Simulation of nervous centres in closed-loop of perception-action

Ph.D. thesis in Communication Engineering by Francisco Naveros

Doctoral Program in Information and Communication Technologies.

Department of Computer Architecture and Technology

University of Granada

2017

Advised by

Prof. Eduardo Ros

Dr. Niceto R. Luque

Dr. Jesús A. Garrido

Editor: Universidad de Granada. Tesis Doctorales
Autor: Francisco Naveros Arrabal
ISBN: 978-84-9163-262-7
URI: http://hdl.handle.net/10481/47117

Simulation of nervous centres in closed-loop of perception-action

Francisco Naveros Arrabal

Department of Computer Architecture and Technology

University of Granada

A dissertation submitted for the degree of

Philosophiae Doctor (PhD) in Communications Engineering

April 2017

UNIVERSIDAD DE GRANADA

Simulation of nervous centres in closed-loop of perception-action

(Simulación de centros nerviosos en ciclo cerrado de percepción-acción)

Dissertation presented by:

Francisco Naveros Arrabal

To apply for the:

International PhD degree in Communications Engineering

[illegible]

Signed: Francisco Naveros Arrabal

Declaración

El doctorando / *The doctoral candidate* [**Francisco Naveros Arrabal**] y los directores de la tesis / *and the thesis supervisor/s*: [**Eduardo Ros Vidal, Niceto Rafael Luque Sola and Jesús Alberto Garrido Alcázar**]

Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la direccion de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones

/

Guarantee, by signing this doctoral thesis, that the work has been done by the doctoral candidate under the direction of the thesis supervisor/s and, as far as our knowledge reaches, in the performance of the work, the rights of other authors to be cited (when their results or publications have been used) have been respected.

Lugar y fecha / *place and date*:

Granada, a 30 de Marzo de 2017.

Director/es de la Tesis / *Thesis supervisor/s*:

Fdo. Eduardo Ros Vidal

[illegible]

Fdo. Niceto Rafael Luque Sola

[illegible]

Fdo. Jesús Alberto Garrido Alcázar

[illegible]

Doctorando / *Doctoral candidate*:

Fdo. Francisco Naveros Arrabal

[illegible]

Acknowledgements

I would like to show my gratitude to my three thesis supervisors: Prof. Eduardo Ros, Dr. Jesús Garrido and Dr. Niceto Luque. Their respective positive, meticulous and realist points of view have helped me to face the development of this thesis. Eduardo has been always a beacon on the horizon, guiding me through the exciting world of computational neuroscience. Jesús taught me to be meticulous with the work done. Finally, I would specially like to show my gratitude to Niceto. Without his hard work this thesis would not have been possible. We have worked hand in hand developing the cerebellar models and writing the paper presented in this dissertation.

I would also like to show my gratitude to my colleague Richard Carrillo. He has always been willing to help me with anything I have needed. I also want to remember all my colleagues, first in my degree (Trini, Alberto, Soraya, Quique, Antonio, Migue, Juanjo and Javi) and after in my PhD (Fran, Jose Luis, Pablo, Irene, Leo, Karl, Quique, Pablo, etc.).

I would also like to express my gratitude to the group led by Angelo Arleo at the Pierre & Marie Curie University (UPMC) in Paris (France). He gave me the opportunity to work in his lab during six months. This half year gave me the chance to grow professionally (I could implement a vestibule-ocular reflex experiment using his iCub robot) and personally (Paris is an amazing city full of culture and history). I would also like to thanks the Institut des Systèmes Intelligents et de Robotique (ISIR), UPMC and especially Ryan Lober and Jorhabib Eljaik for giving me the opportunity to work with their iCub robot.

Moreover I would like to remember my friends from Alhama de Granada (Carlos, Vicente, Maria, Juanfran, Maricarmen, Jesús, Fernando, etc.). I especially want to thank the support of my best friends Juan Francisco and Silverio with whom I have spent my whole life sharing experiences.

Finally, I wish to thank my family, who has supported me in this period of mi life: my father Francisco and my mother Maria del Pilar, my brothers Adolfo and Emilio, my uncles and cousins, and last but not least, my girlfriend María Trinidad. They never fully understood what I was doing in my PhD, but they always believed in me.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AdEx: adaptive exponential integrate-and-fire
AMPA: α-amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid
API: application programming interface
CF: climbing fibre
CPU: central processing unit
CS: conditioned stimulus
CUDA: compute unified device architecture
DCN: deep cerebellar nuclei
EBCC: eye-blink classical conditioning
EC: electrical coupling
GABA: gamma-aminobutyric acid
GC: granular cell
GoC: Golgi cell
GPU: graphic processing unit
HH: Hodgkin-Huxley
IO: inferior olive
LIF: leaky integrate-and-fire
LTD: long-term depression
LTP: long-term potentiation
NMDA: N-methyl-D-aspartic acid
OpenMP: open multi-processing
PC: Purkinje cell
RAM: random access memory
PF: parallel fibre
RT: real-time
STDP: spike-timing-dependent plasticity
TCP/IP: transmission control protocol / internet protocol
US: unconditioned stimulus
VOR: vestibulo-ocular reflex
YARP: yet another robot platform

Abstract

The human brain, thanks to the evolution, has developed efficient biological structures able to perform a wide range of complex tasks. This is also the case of one of its centres: the cerebellum. This structure plays a fundamental role in different motor control tasks such as coordination of movements or calibration of sensorimotor relationship. In this thesis computational models of the cerebellum have been developed in order to achieve better understanding of the biological mechanisms that confer the cerebellum its motor control and learning capabilities. Simulated and real robots have been used in this work as emulated bodies to control. Thus we aim to validate several hypotheses about the cerebellum operation when performing different motor control tasks such as object manipulation, eye blink classical conditioning (EBCC) or vestibulo-ocular reflex (VOR) experiments.

The scope of this thesis is the development of biologically inspired control systems based in cerebellar models able to perform different motor control tasks using biomorphic robots in real time. This work can be subdivided in three main blocks: (i) development of all the tools needed for this study, (ii) development of a cerebellar model based in data obtained with biological experiments, and (iii) validation of the cerebellar models embedding these ones in control schemes able to perform different motor control tasks with biomorphic robots in real time.

An upgraded version of the EDLUT simulator has been used as the main simulation tool for this study. EDLUT is an efficient spiking neural network simulator developed by our research group at the University of Granada. It was conceived as a small tool capable of efficiently simulate medium-scale spiking neural networks. In this thesis we have systematically improved the efficiency and functionality of EDLUT. We have parallelized its simulation in multicore CPU-GPU co-processing architectures, developed new and efficient event-driven and time-driven simulation methods and implemented new neuron models and learning rules. Additionally, we have included new modules and features in EDLUT related with the robotic control in real time. EDLUT-simulation spiking neural models can now connect with many (simulated or real) robotic devices using TCP/IP connections. Communication interfaces able to translate the cerebellar signals (spikes) in robotic signal (analogical signals) and vice versa have also been implemented within EDLUT. Finally, EDLUT incorporates a real time supervisor able to ensure that a simulation is performed in real time. Thus, EDLUT is now more than a simple spiking neural network simulator. It is a simulation tool able to create biologically inspired control schemes based in spiking neural networks to perform different motor control tasks using biomorphic robots in real time.

Starting from a cerebellar model previously developed by our research group, two new plasticity mechanisms at deep cerebellar nuclei (DCN) level have been proposed and implemented, conferring to the cerebellar model with learning consolidation and gain adaptation capabilities. We have also propose a new neural model for Purkinje cells able to replicate its tri-modal spike modes (tonic, silence and bursting) and a new Inferior Olive (IO) layer interconnected with electrical coupling able to better codify the error signal. Finally, a new synaptic connection from the IO to the DCN cells have been proposed and included. All these new elements, based in theoretical hypotheses and experimental results in the literature, have increased the biological plausibility of our cerebellar model.

Finally we have analysed how each one of the above-mentioned elements affects the behaviour of the whole cerebellar model when performing motor control experiments as a test-bench: a manipulation object task with a robotic arm, an EBCC experiment with a simulated environment or a VOR experiment with a robotic head.

This work has been partially supported by the FPU national grant program, the European projects REALNET (FP7-270434) and HBP (FP7-604102), the Marie Curie grant (658479- Spike Control) and the French government research program "Investissements d'avenir through the Robotex Equipment of Excellence (ANR-10-EQPX-44)". We gratefully acknowledge the support of NVIDIA Corporation with the donation of two Titan GPUs for current EDLUT development.

Resumen

El cerebro humano, gracias a la evolución, ha desarrollado eficientes estructuras biológicas capaces de realizar un amplio rango de tareas complejas. Este es también el caso de uno de sus centros: el cerebelo. Esta estructura juega un papel fundamental en diferentes tareas de control motor tales como la coordinación de movimientos o la calibración de la relación sensoriomotora. En esta tesis hemos desarrollado modelos computacionales del cerebelo para entender mejor los mecanismos biológicos que le confieren al cerebelo sus capacidades de control y aprendizaje motor. En este trabajo se han usado robots simulados y reales como cuerpos emulados que controlar. De esta forma tratamos de validar las hipótesis propuestas sobre la operación del cerebelo cuando este realizar diferentes tareas de control motor tales como manipulación de objetos o recreación de los experimentos de condicionamiento clásico del parpadeo (EBCC) o reflejo vestíbulo-ocular (VOR).

Esta tesis pretende abarcar el desarrollo de sistemas de control biológicamente inspirados basados en modelos cerebelares capaces de realizar diferentes tares de control motor usando robots biomorficos en tiempo real. Este trabajo se puede subdividir en tres bloques principales: (i) desarrollo de todas las herramientas necesarias para este estudio, (ii) desarrollo de un modelo cerebelar basado en datos obtenidos de experimentos biológicos, y (iii) validación de los modelos cerebelares embebiéndolos en esquemas de control capaces de realizar diferentes tareas de control motor usando robots biomorficos en tiempo real.

Una versión actualizada del simulador EDLUT ha sido usada como la principal herramienta de simulación para este estudio. EDLUT es un simulador eficiente de redes neuronales de impulsos desarrollado por nuestro grupo de investigación en la Universidad de Granada. Este fue concebido como una pequeña herramienta capaz de simular de forma eficiente redes neuronales de media escala. En esta tesis hemos mejorado sistemáticamente la eficiencia y funcionalidad de EDLUT. Hemos paralelizado su simulación en arquitecturas de coprocesamiento CPU-GPU multinúcleo, desarrollado nuevos y eficientes métodos de simulación dirigidos por eventos y por tiempo e implementado nuevos modelos de neurona y leyes de aprendizaje. Además hemos incluido en EDLUT nuevos módulos y características relacionadas con el control robótico en tiempo real. Las redes neuronales simuladas en EDLUT pueden ser ahora conectadas con diversos robots (simulados o reales) usando conexiones TCP/IP. También se han implementado en EDLUT interfaces de comunicación capaces de traducir las señales del cerebelo (impulsos) en señales para el robot (señales analógicas) y viceversa. Por último, EDLUT incorpora un supervisor de tiempo real capaz de asegurar que una simulación se realiza en tiempo real. Por lo tanto, EDLUT es ahora más que un simple simulador de redes neuronales de impulsos. EDLUT es una herramienta de simulación capaz de crear sistemas de control biológicamente inspirados basados en redes neuronales de impulsos para realizar diferentes tareas de control motor utilizando robots biomórficos en tiempo real.

Partiendo de un modelo cerebelar previamente desarrollado por nuestro grupo de investigación, dos nuevos mecanismos de plasticidad sináptica a nivel de los núcleos cerebelos profundos (DCN) han sido propuestos e implementados, confiriéndole al modelo nuevas capacidades de consolidación del aprendizaje y adaptación de la ganancia. También hemos

propuesto un nuevo modelo de neurona capaz de reproducir los tres estados de una célula de Purkinje (tónico, silencioso y ráfaga), así como una nueva capa de la Oliva Inferior (IO) con acoplamiento eléctrico entre sus neuronas. Por último, una nueva conexión sináptica desde la IO hasta los DCN ha sido propuesta e incluida en el modelo. Todos estos nuevos elementos, basados en hipótesis teóricas y resultados experimentales en la literatura, han incrementado la plausibilidad biológica de nuestro modelo cerebelar.

Por último hemos analizado cómo cada uno de los elementos anteriormente mencionados afecta al comportamiento del modelo cerebelar cuando este realiza experimentos de control motor como banco de pruebas: una tarea de manipulación de objetos con un brazo robótico, un experimento EBCC con un entorno simulado o un experimento VOR con una cabeza robótica.

Este trabajo ha sido parcialmente apoyado por una beca nacional del programa FPU, los proyector Europeos REALNET (FP7-270434) y HBP (FP7-604102), la beca Marie Curie (658479-Spike Control) y el programa de investigación “Investissements d'avenir through the Robotex Equipment of Excellence (ANR-10-EQPX-44)” del gobierno Francés. Agradecemos también el apoyo de la corporación NVIDIA por la donación de dos GPUs Titan para el desarrollo de EDLUT.

Chapter 1: Introduction

One of the main challenges that human beings face in 21st century is to understand the biological principles of consciousness and mental processes through which we perceive, act, learn and remember (Kandel et al. 2000). The knowledge of these biological principles in nervous systems can bring us several benefits. Firstly, further understanding of the brain helps us to design better and more effective treatments for neurodegenerative diseases such as Huntington, Parkinson and all forms of dementia. Secondly, the brain can be considered as one of the best information processing systems: huge storage and processing capabilities with very low power consumption and reliability against failures (aging). Deeper knowledge on how biological systems process the information will allow the development of new generations of processing architectures able to replicate this astonishing performance.

The cerebellum (Latin for "little brain") is a very important brain region for vertebrates such as humans. It is well known that the cerebellum plays a fundamental role in different motor control features such as coordination and accurate movements (Thach, Goodkin, and Keating 1992), although it seems to be also involved in other cognitive functions such as attention and language, and in regulating fear and pleasure responses (Wolf, Rapoport, and Schweizer 2009). It has been observed that cerebellar damages produce disorders in fine movement, equilibrium, posture, and motor learning (Fine, Ionita, and Lohr 2002).

Understanding the biological mechanisms involved in the cerebellum require its study from many different methodological approaches, been all of them interconnected. This thesis addresses the study of the cerebellum from the point of view of computational neuroscience. It is an interdisciplinary science that links diverse fields such as neuroscience, cognitive science and psychology with electrical engineering, computer science, mathematics, and physics. Due to the fast increase of computational resource availability in the last decades, computational neuroscience has emerged as a powerful tool able to test and validate brain hypotheses proposed by neuroscientists. Computational neuroscientists use mathematical models and computer simulations to study neural systems at different levels. Depending on the level of detail and the size of the structure under simulation (from a molecular level in a single neuron to a large and intricate neural network with distributed plasticity), the computational requirements may drastically differ.

Computational models of various brain regions have been developed and studied for more than thirty years in order to analyse brain function. Computational neuroscience is the natural complement of experimental brain research, since it focuses on specific mechanisms and models which are only partially observable in physiological studies. In particular, the cerebellar control loop (the focus of this thesis) has been extensively modelled since Marr (Marr 1969) and Albus (Albus 1971) proposed elegant explanations on how the cerebellum operates as a forward-controller in mammals.

The simulation of a nervous system model connected to a body (embodiment) can help to better understand how certain capabilities of the nervous system emerge based on cellular characteristics, network topology, or local synaptic adaptation mechanisms. In this thesis we have focussed on understanding the role of the cerebellum in coordinated movements (voluntary or reflex) and object manipulation. For that reason, we need a "body" able to perform these motor tasks when it is controlled by our cerebellar models. We have used both virtual (simulated) and real robots to emulate a body. Nevertheless, several considerations have to be taken into account. While the human body encodes the sensorimotor information using neural population coding (each neuron encoding the signal presents a distribution of responses over some set of inputs), robots use encoders which return the relative position and velocity (analogical signals) of each joint. A translation from analogical domain sensor inputs of robots to spike-based patterns compatible with the spiking neural network is required. Additionally, the output responses of the cerebellum based in spikes must be translated to the analogical domain of the robot actuators. Finally, real robots impose a real-time (RT) constriction. When we carry out a simulation connected to a real robot, the simulation time must evolve at the same speed than the RT (physical time), otherwise the robot cannot be properly operated. The dynamics of the robot (such as momentum of different links) and dynamic environment make this RT operation to be a hard/strict constraint.

In this thesis we have embedded several cerebellar models in different closed-loop schemes able to control the movement of biomorphic robots in RT. This integration aims the following goals: (i) validation of the hypotheses presented in each cerebellar model using different motor control tasks as benchmarks, and (ii) development of innovative control schemes for the next generation of biomorphic robots.

For this study we have developed and upgraded our own spiking neural network simulator: EDLUT (Ros et al. 2006, Garrido et al. 2011, Luque et al. 2014, Naveros et al. 2015, Naveros et al. 2017). This tool is oriented to the efficient simulation of medium-scale (tens of thousands of neurons) neural networks using simplified point neuron models (Leaky Integrate-and-Fire (LIF), Adaptive exponential integrate-and-fire (AdEx), Izhikevich and Hodgkin-Huxley (HH)) and plasticity mechanisms in RT. In our case we have used EDLUT to develop cerebellar models embedded in closed-loop schemes able to control the movement of biomorphic robots in RT. The RT constraint, which demands the execution of the simulated neural network at a determined speed, is fundamental when we operate with real robots in which the time is a physical variable that we cannot control. In this thesis we have drastically improved EDLUT performance using different parallelisation and simulation techniques, thus allowing the simulation of more complex and larger neural networks in RT. Additionally, a specialised closed-loop control scheme together with a RT supervisor has been also developed to deal with the RT constriction.

1.1 The cerebellum

The cerebellum has the appearance of a separate structure attached to the bottom of the brain, tucked underneath the cerebral hemispheres. This structure has been deeply investigated for more than a century since Camillo Golgi and Santiago Ramón y Cajal studied the anatomical organization of the cerebellar cortex (Golgi 1906, Cajal 1894). Although the cerebellum just accounts for approximately 10% of the brain's volume, it contains over 50% of the total number of neurons in the brain (Llinas, Walton, and Lang 2004). It can be subdivided in two major parts: an internal section called deep cerebellar nuclei (DCN) covered by a highly convoluted sheet of tissue called cerebellar cortex. While the DCN represents the output structure of the cerebellum, the cerebellar cortex conveys the main input to the cerebellum and contains most of the neurons in the cerebellum (Purves et al. 2008). Within the cerebellar cortex there are several types of neurons with a highly regular arrangement, being Purkinje cells (PCs) and granule cells (GCs) the most characteristic ones. This complex neural organization gives rise to a massive signal-processing capability (Eccles, Ito, and Szentágothai 1967).

In light of some studies, the central nervous system has been suggested to plan and execute sequentially voluntary movements. Thus, the cerebellum would not initiate voluntary movement, but it would rather contribute to its coordination, precision and accurate timing using a feedback loop for muscle movement (Ito 1970, 2006). In accordance to this hypothesis, the brain might first plan the optimal trajectory in task-space coordinates, translate them into intrinsic-body coordinates, and finally, generate the necessary motor commands (Houk, Buckingham, and Barto 1996, Nakano et al. 1999, Todorov 2004, Hwang and Shadmehr 2005, Izawa et al. 2012, Passot, Luque, and Arleo 2013).

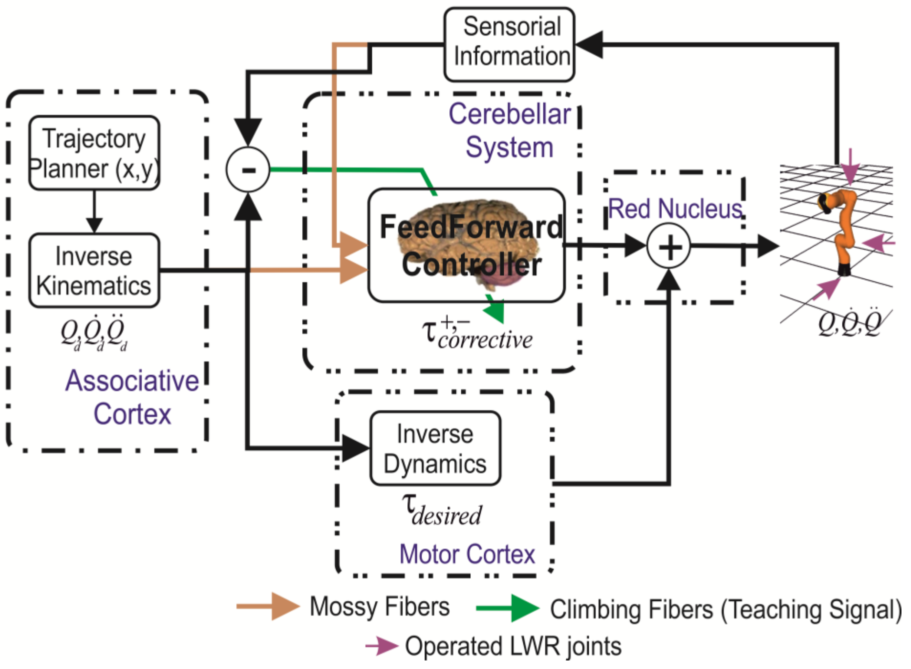


Figure 1 ***Feedforward cerebellar control loop (adapted from (Luque et al. 2016)).*** *The adaptive cerebellar module embedded in a feedforward control loop delivers corrective torque values ($\tau_{corrective}$) to compensate for deviation in the crude inverse dynamics module when manipulating objects of significant weight.*

On the one hand, the cerebrocerebellum–parvocellular red nucleus system is thought to provide a crude internal neural model of the inverse-dynamics of the musculoskeletal system, which is acquired whilst monitoring the desired trajectory (Kawato, Furukawa, and Suzuki 1987). On the other hand, the spinocerebellum–magnocellular red nucleus system is thought to hold an internal neural accurate model of the musculoskeletal body dynamics learnt through sensing voluntary movements (Kawato, Furukawa, and Suzuki 1987). According to the previous theory, the associative cortex would be in charge of providing the desired trajectory in body coordinates and conveying them to the motor cortex. Then it would generate the optimal motor commands to operate our limbs using an inverse dynamic model and would send this ones to the lower motor neurons in the brainstem and spinal cord (Siciliano and Khatib 2016). The cerebellum also receives the desired trajectory together with the sensory information coming from the spinal cord and other parts of the brain (Siegel and Sapru 2006) and generates motor command corrections according to predictable errors occurring when executing a movement using the musculoskeletal body dynamic model. Thus, the crude inverse-dynamic model works together with the dynamic model provided by the cerebellum embedded in a feedforward control loop (Fig. 1).

Learning and adaptation mechanisms play a crucial role in cerebellar motor control. Several theoretical models developed in last decades try to explain how the cerebellum calibrates the relation between sensorial inputs and motor commands using synaptic plasticity mechanisms. Most of these models extend the theories proposed by David Marr (Marr 1969) and James Albus (Albus 1971) based on the observation that each PC receives two dramatically different types of inputs: (i) thousands of weak inputs coming from GC along the parallel fibres (PF) and (ii) a single and extremely strong input coming from an Inferior Olive (IO) cell along a climbing fibre (CF). The strength of these CFs is so intense (due to the numerous synaptic contacts existing between each CF and PC dendrites) that a simple action potential from the IO generates a so-called complex spike in the target PC (a fast initial large-amplitude spike followed by a high-frequency burst and a period of non-activity (Schmolesky et al. 2002)).

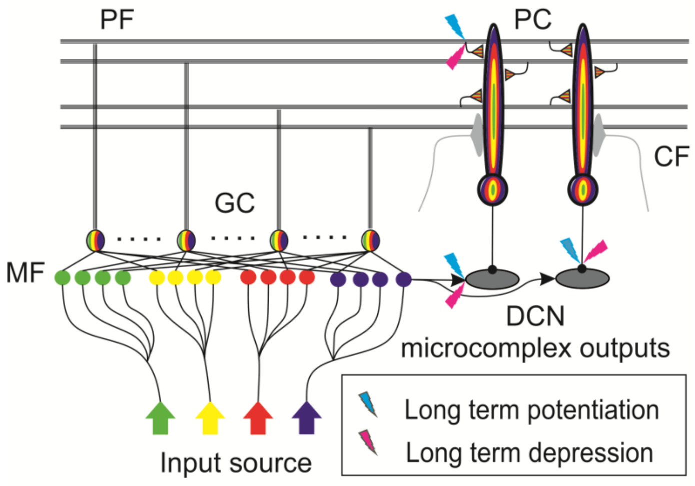


Figure 2 ***Cerebellar architecture (adapted from (Luque et al. 2016)).***
Our cerebellar model is composed of mossy fibres (MF), granule cells (GC), parallel fibres (PF), Purkinje cells (PC), climbing fibres (CF) and deep cerebellar nuclei cells (DCN). Long-term synaptic plasticity for PC and DCN afferents are indicated with two coloured symbols; long-term potentiation (LTP) in blue and long-term depression (LTD) in magenta.

The basic concept of Marr-Albus theory relies on the fact that PFs (i.e. the axons of GCs) propagate sensorimotor information to PCs while CFs (i.e. the axons of IO cells) propagate teaching signals that codify the movement error. These teaching signals induce long-lasting changes in the strength of PFs. Observations of long-term depression (LTD) in PF inputs have provided support for theories of this type, but their validity remains controversial (Purves et al. 2008). In this thesis we have extended this model including two additional plasticity mechanisms at DCN level. These new plasticity mechanisms help to consolidate the learning of

PC level at DCN level and optimise the operational range of DCN cells. Figure 2 represents a scheme of the cerebellar model including the three proposed plasticity mechanisms.

The eye blink classical conditioning (EBCC) (Thompson 1990) and the vestibulo-ocular reflex (VOR) (Leigh and Zee 2015) are broadly assumed as the paradigms that better reveal cerebellar learning characteristics. The EBCC is a relatively simple procedure that consists of pairing an auditory or visual stimulus (the conditioned stimulus (CS)) with an eye blink eliciting unconditioned stimulus (US) (e.g. a mild puff of air to the cornea). By contrast, the VOR is a bit more complicated procedure in which the activation of the vestibular system causes eye movements. This reflex stabilizes images on the retinas during head movement by producing eye movements in the opposite direction to the head movement, thus preserving the image on the centre of the visual field. Both experiments have been recreated using our cerebellar models.

In this thesis we have focused on understanding the mechanisms used by the cerebellum in two main functions related to motor control:

- Coordination of voluntary movements: The realization of most of the movements involves the activation of a number of different muscle groups in a temporally coordinated fashion. One major function of the cerebellum is to coordinate the timing and force of the operation of different muscle groups to produce fluid limb or body movements.
- Motor learning: The cerebellum plays a major role in adapting and fine-tuning motor commands to make accurate movements through a trial-and-error process. The cerebellum must be constantly adjusting the sensorimotor relationships in order to compensate changes in its body (e.g. a tennis player that uses a new racquet with different weight or dimensions) or in the environment (e.g. to play in a grass court or in a hard court).

1.2 Computational neuroscience: simulation tools

In the last years, the spiking neural network simulation tools have experienced a fast evolution from little tailor-made simulators for specific experiments to very complex and general purpose simulators able to perform a wide range of experiments. Currently there exists an extensive number of simulation tools specialised in different kind of simulations. (i) NEURON (Hines and Carnevale 1997) and GENESIS (Bower and Beeman 1998) are the reference when detailed biophysical models of neurons are to be simulated. (ii) NEST (Gewaltig and Diesmann 2007) is the reference when huge neural networks (billions of neurons and trillions of synapses) are to be simulated in supercomputers using simplified point neuron models (where the morphology is not relevant). (iii) Spikey (Pfeil et al. 2012, Schemmel et al. 2010) is a neuromorphic simulator based in analogical circuits (condensers and resistors) over a silicon wafer. It performs simulations between 1000 and 10,000 times faster than RT. This incredible speed-up rate can be used to perform intensive parameter analysis of neural networks in short periods of time. (iv) SpiNNaker (Khan et al. 2008) is a tailor-made massively parallel computer architecture based in ARM processors distributed in a toroidal topology. The whole machine is planned to contain one million ARM processors and will be able to perform simulations with up to one billion simple neurons. A deeper review about this topic can be found in (Brette et al. 2007).

The work developed in this thesis follows the research lines of our group at the University of Granada in the last years. It aims the development of biologically inspired cerebellar models embedded in closed-loop control schemes able to perform different motor control tasks with real robots in RT. Due to the shortage of available tools able to meet the requirements needed for this embodiment experiments, our group decided to develop our own simulation tool: EDLUT.

1.2.1 Simulation strategies: time-driven vs event-driven

Each spiking neural network simulator must perform at least three different tasks: (i) computation of the dynamic evolution of each neuron (normally defined by a set of differential equations that need to be integrated over the time), (ii) generation and propagation of spikes, and (iii) computation of the plasticity mechanisms that modify the synaptic weights. These tasks can be carried out using two different families of simulation methods: time-driven and event-driven simulation methods. The main difference between both methods refers to the way in which they manage the simulation time evolution, especially for the computation of the neural dynamic evolution. A deeper review about this topic can be found in (Brette et al. 2007).

Most of the simulators use time-driven simulation methods based on fixed-step integration since this scheme allows the simulation of the majority of neuron, synapse and learning rule models. These methods divide the simulation time into short time steps of fixed size and evaluate the dynamic evolution of each neuron integrating its differential equation system in each step (Iserles 2009). The main drawback of fixed-step integration methods is the limited efficiency when the neural activity to process is sparse or when the neural complexity is high (in terms of number of differential equations per neuron). In this case it could be recommended to use variable-step integration methods (Iserles 2009). These methods iteratively adapt the simulation step size as a function of the neural dynamic evolution of each neuron (this evolution depends on the neural activity and the model complexity). However, variable-step integration methods are not exempt of drawbacks (deeply discussed in the second journal article included in this thesis (Naveros et al. 2017)). Thus, variable-step integration methods are usually restricted to simulators specialised in the simulation of a few neurons with very complex neural modes (e.g. NEURON or GENESIS).

Alternatively, event-driven simulation methods emerged as a solution to the low efficiency of fixed-step integration methods when neural networks with low activity had to be simulated. There are several regions in the brain characterised by this sparse activity (e.g. the GCs at the cerebellar cortex account for half of the neurons of the whole brain, receive between three and six input synapses with very sparse activity and most of them remain silent and barely generates spikes). The main strength of event-driven simulation methods relies in the fact that the neural dynamic evolution is only computed and updated when a new event modifies the normal evolution of a neuron (i.e. when an input spike is received or an output spike is produced). Several event-driven simulation methods have been developed and incorporated in different spiking neural network simulators (Mattia and Del Giudice 2000, Delorme and Thorpe 2003, Reutimann, Giugliano, and Fusi 2003, Rudolph and Destexhe 2006, Pecevski, Kappel, and Jonke 2014). The main drawback of these methods is the fact that only relatively simple neural models based on equations that can be evaluated at arbitrary times can be implemented (e.g.

Spike-Response Model). This restriction limits the number of neural models that can be simulated using these techniques (limitation that time-driven methods do not suffer).

1.2.2 The evolution of EDLUT simulator

Our research group at the University of Granada has been studying the cerebellum for more than a decade. Inside the cerebellum, the GCs are, by far, the most numerous neurons (accounting for half of the neurons of the whole brain) and present low input connectivity ratios with sparse activity levels. Trying to exploit these features our research group developed the very first EDLUT version following an event-driven simulation scheme based in look-up tables (Ros et al. 2006). This innovative method computes in an initial stage the neural dynamic evolution of whatever kind of neural model with low complexity (up to four or five differential equations) and stores this one in look-up tables that will be posteriorly used during the simulation time. Neural models such as LIF, AdEx, Izhikevich and HH can be pre-computed in look-up tables. For a deeper review about this topic we recommend the reading of Dr. Carrillo's thesis (Carrillo 2009).

However, there exist different types of neurons in the cerebellum with very different connectivity and activity ratios. As we have previously commented, the GCs are perfect candidates to be simulated using event-driven methods due to its sparse activity. By contrast, PCs (large neurons integrating activity from up to 100,000 input synapses) fit better with time-driven simulation methods. Thus, the efficient simulation of the cerebellum requires two different technical approaches. For that reason, Dr. Jesús Garrido (co-supervisor of this thesis) upgraded the EDLUT kernel toward a hybrid event- and time-driven simulation scheme able to use both simulation methods simultaneously (Garrido et al. 2011). Just fixed-step integration methods were implemented. Thus EDLUT could take advantage of the strengths and mitigate the drawbacks of both simulation methods (event-driven methods based in look-up tables for relatively simple neural models with low activity rates (as the GCs) and time-driven methods based in fixed-step integration methods for complex neural models with high activity rates (as the PCs). For a deeper review about this topic we recommend the reading of Dr. Garrido's thesis (Garrido 2012).

1.3 Motivation

Traditionally, the neuroscience has studied biological systems by using *in vitro* and *in vivo* experiments. *In vitro* (Latin for within the glass) refers to the technique of performing a given procedure in a controlled environment outside of a living organism. Thus the experiment can be perfectly controlled although just simple behaviours and characteristics can be studied. By contrast, *in vivo* (Latin for "within the living") refers to experimentation using a whole, living organism as opposed to a partial or dead organism. This technique allows more complex systems and behaviours to be studied, although the control over the experimental conditions and the measures that can be registered are notably more limited. Recently, computational neuroscience with its *in silico* (performed on computer or via computer simulation) experiments has emerged as a third approach to study the brain. In this thesis we have used the results obtained with experimental observation (mainly in rats and mouse) to create a computational model of the cerebellum. This model serves a double function: (i) validate the hypothesis raised by *in vitro* and *in vivo* experiments, and (ii) propose new hypothesis that can be validated by *in vitro* and *in vivo* experiments. Thus, computational neuroscience experiments are a perfect complement for animal experimentation.

Performing embodiment experiments in RT is a complex task that requires a specialised simulation tool. The journal articles published before this thesis and related with the development of EDLUT (Ros et al. 2006, Garrido et al. 2011) show the extraordinary potential of EDLUT to efficiently simulate cerebellar models. However, previous EDLUT versions lacked the efficiency to perform experiments in RT with reasonably large and complex models. A remarkable improvement of EDLUT performance as well as the development of a RT mechanism and a robot interface are then required.

Focusing in the cerebellum, it is well know that this structure plays a fundamental role in the coordination and learning of movements. Neuroscientists perform behavioural experiments such as EBCC, VOR or manipulation tasks to study the cerebellum, observing how it controls the body. We replicate this methodology by studying our cerebellar models embedded in closed-loop schemes able to control the movement of simulated or real robots in RT. Thus we can better appreciate the influence of cerebellar models in the robot behaviour. This configuration allows the exploration of different features of the cerebellar model (neuron models, interconnectivity topology, plasticity mechanisms, etc.) and how they affect the robot behaviour. Additionally, robot behaviours can be compared with the ones observed by the neuroscientists using animals (or humans), thus validating or refuting the theories studied.

From the point of view of robotics systems, the cerebellum is able to coordinate the movement of hundreds of muscles with astonishing reliability and low power consumption. Additionally, it is able to learn new tasks by repeating the movements and observing the consequences. By mimicking these features in robotic controllers can deeply improve the functionality and autonomy of ground-breaking biomorphic robots.

Finally, from the point of view of information processing, the human brain represents a very powerful and energy efficient processing system robust against failures (aging). Understanding how the biological systems manage to process information from hundreds of sources will allow the development of new generations of processing architectures with never-seen-before capabilities.

1.4 Objectives

This thesis aims to create cerebellar models that explain the biological mechanisms allowing the cerebellum to perform different motor control tasks. In order to validate these models, they have been embedded in different closed-loop control schemes and have been exposed to demanding motor control tasks using simulated or real robots in RT. The secondary aim of this work is the implementation of novel biologically inspired control schemes able to control the new generation of biomorphic robots. Finally, the third aim this work addresses is the development of the simulation tools that enable this study, exploring new and efficient simulation methods, control schemes, communication interfaces and RT supervisors.

In order to achieve these objectives, this thesis addresses the following separate goals:

- Development of the simulation tools required to create spiking-neural-network-based control systems for real robots. This goal can be subdivided in:
  - ✓ Development of an ultra-fast spiking network simulator (EDLUT) for medium-scale networks. Implementation of different parallelization techniques in Central Processing Units (CPUs) and Graphics Processing Units (GPUs) to increase its computational performance.
  - ✓ Implementation of efficient event-driven and time-driven simulation methods for neural models such as LIF, AdEx, Izhikevich and HH.
  - ✓ Integration of the neural network simulator within a robotic environment (able to manage simulated and real robots) to facilitate the experimentation using biological control schemes.
  - ✓ Enhancing the simulation software with the capability of ensuring RT simulations with robots.
- Implementation of a cerebellar model using the tools previously developed. This goal can be subdivided in:
  - ✓ Study of how the different elements of the cerebellar model (neurons and synapses) contribute to its motor control skills. It requires the implementation and evaluation of complex neuron and synapsis models in EDLUT, with especial attention to the nucleo-olivary system (Purkinje, DCN and IO cells).
  - ✓ Study of different plasticity mechanisms at Purkinje and DCN level that allow the cerebellar models to acquire new skills.
- Validation of the proposed cerebellar models by means of its inclusion in closed-loop control schemes performing different motor control tasks with simulated or real robots in RT.

1.5 Our contribution

With the RT restriction in mind this thesis has deeply improved the efficiency and functionality of EDLUT. Regarding the efficiency, our main contributions are covered in the first (Naveros et al. 2015) and second (Naveros et al. 2017) journal articles included in this thesis, but also in the additional results pending for publication included in the fourth chapter of this thesis. Nowadays, most CPUs include several physical or virtual (hyper threading) cores. For that reason we have parallelised EDLUT kernel (event-driven and time-driven simulation methods) in CPU by using Open Multi-Processing (OpenMP). This is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C++. Additionally, a few years ago NVIDIA developed Compute Unified Device Architecture (CUDA), a parallel computing platform and API model that allows software developers to use CUDA-enabled GPUs for general purpose processing programs exploiting data parallelism. Each CUDA-enabled GPU has hundreds or even thousands of CUDA cores able to execute in parallel the same code over multiple data. The computation of the neural dynamic evolution in time-driven models using fixed-step integration methods is a task that perfectly copes with the parallel architecture of CUDA. For that reason we have upgraded EDLUT toward a CPU-GPU co-processing platform in which the GPU (using CUDA) computes time-driven neural models while the CPU (using OpenMP) calculates time-driven and event-driven neural models. The spike generation and propagation, as well as the plasticity mechanism are processed by the CPU in parallel using OpenMP. All these new features are covered in (Naveros et al. 2015).

We have also developed new and efficient event-driven and time-driven simulation methods to make them capable to deal with complex neural models. For the event-driven methods we have restructured the look-up tables and developed an efficient activity propagation mechanism. For time-driven methods in CPU and GPU we have developed a new bi-fixed-step integration method that takes advantage of the strength and mitigates the weaknesses of fixed-step and variable-step integration methods. This upgrade is covered in (Naveros et al. 2017).

Finally, in the additional results section of this thesis we have exposed how different developments related with the computation of neurons, synapses and plasticity mechanisms have been included in EDLUT.

Regarding the functionality of EDLUT, our main contributions are covered in the remaining journal articles included in this thesis (Luque et al. 2014, Luque et al. 2016) and the additional results section. New neural models and learning rules required for our cerebellar models have been implemented in EDLUT. We have also equipped EDLUT with an integrated robotic software framework. This framework allows the implementation of different closed-loop control schemes in which our cerebellar models are embedded. EDLUT includes now: (i) an interface that translates analogical signals generated by the robots into spikes that the cerebellar model understands and vice versa, (ii) a communication interface using TCP/IP connections between the control scheme and the robot interface (that can be executed in the same or different computers), and (iii) a RT supervisor that ensures the RT constraint. Additionally, EDLUT kernel has been modified to allow the simulation of neural elements (spikes generation and propagation, plasticity mechanism, neural dynamic evolution, etc.) to be temporally disabled in order to comply with the RT constraints. This feature is controlled by the RT supervisor, which may speed-up or speed-down the simulation.

In addition to this, we have made extensive usage of these implemented tools to study the cerebellum. We have studied how the different neurons, synapses and plasticity mechanisms that have been experimentally reported in the cerebellum contribute to its motor control skills. More concretely we have focused our study in the closed-loop containing the IO, Purkinje and DCN cells (closing the loop from DCN to IO through the body). Although the IO cells are not considered as part of the cerebellum, it is believed that their axons (CFs) carry a teaching signal encoding the error to the cerebellum and influence the synaptic weight modification of PFs (Ito and Kano 1982, Ito 2001). We have studied the relationship between these three neurons in several demanding motor control and motor learning tasks with robots: (i) manipulation of heavy/light objects using a simulated robotic arm (Luque et al. 2014, Luque et al. 2016), (ii) recreation of an EBCC experiment using a simulated environment (Antonietti et al. 2016), and (iii) recreation of a VOR experiments using a simulated and real iCub robot (additional results section).

Regarding the plasticity mechanisms, three spike-timing dependent plasticity (STDP) mechanisms implementing LTP and LTD at three different groups of synapses (PFs to PCs, MFs to DCN and PCs to DCN) are evaluated in this thesis (Luque et al. 2016). We have seen how in a first learning stage the CF activity at PC level influences the synaptic weight adaptation of their afferent PFs (axons coming from GCs) to minimise the error in the motor tasks. This error signal is computed in the IO cells and convey to the PCs through the CFs. In a second learning stage the PC activity at DCN level influences the synaptic weight adaptation of their afferent MFs. Thus, the learning is consolidated by transferring part of the information stored in the PFs/PCs synapses to the MFs/DCN afferent synapses. Additionally, these mechanisms act as a gain adaptation mechanism able to optimize the operational range of DCN cells.

Finally, we have proposed in the last stage of this thesis (additional results section) a new cerebellar model with three main contributions: (i) a detailed neural model for PCs able to replicate its tri-modal spike modes, namely tonic, silence and bursting (Forrest 2008), (ii) an IO layer interconnected with electrical coupling able to better codify the error signal, and (iii) a new synaptic connection from IO to DCN cells. We have validated this new cerebellar model using a VOR experiment with a simulated and real iCub robot in RT. This is the first experiment in which we have used a real robot (instead of simulated ones as previously). We have also developed in this work an asynchronous inner/outer closed-loop control scheme to facilitate the control of real robots. This control scheme takes advantage of the difference between the biological propagation delay of spikes through nerves and the artificial propagation delay of spikes through artificial systems to relax the RT constraint. This control scheme has been explained in more detail in the second chapter of this thesis.

1.6 Project framework

The work described in this document has been developed in the framework of two European projects: “Realistic Real-time Networks: computation dynamics in the cerebellum” (REALNET (FP7-270434)) and “Human Brain Project” (HBP (FP7-604102)). The last part of this thesis (additional results section) has been developed in collaboration with the research group lead by Angelo Arleo at the Pierre and Marie Curie University (UPMC), Paris.

The REALNET project was a continuation of the European project SENSOPAC (IST-028056). REALNET (funded under the 7th EU Framework Information and Communication Technologies work program) started in February, 2011 and finished in February, 2014. The main goal of this project was to elaborate realistic spiking networks and use them, together with experimental recording of network activity, to investigate the theoretical basis of central network computation. A cerebellar circuit was used as a use case for this study. Based on experimental data, this project developed the very first realistic real-time model of the cerebellum and connected it to robotic systems to evaluate circuit functioning under closed-loop conditions. Within this project our research group at the University of Granada was focused in the development of EDLUT simulator to cope with realistic biological structures in RT increasing the performance of the simulations.

More recently our research group is involved in the HBP project. This is a European Commission Future and Emerging Technologies Flagship. It was launched in October 2013, and is scheduled to run for ten years. The HBP aims to put in place a cutting-edge, ICT-based scientific Research Infrastructure for brain research, cognitive neuroscience and brain-inspired computing. The HBP is divided in 12 subprojects. Our research group at the University of Granada is currently integrated in the subproject 10: Neurorobotics Platform. This one offers scientists and technology developers a software and hardware infrastructure that allows pre-validated brain models to be connected to detailed simulations of robot bodies and environments. Inside this subproject, the University of Granada is integrated in the Work Package 10.1 Closed-Loop Experiments (Data-Driven Brain Models), developing the subtask 10.1.4 Cerebellar motor control. Here we have followed with the development of our cerebellar model, validating this one in closed-loop experiments.

Finally, the doctoral candidate has made a six-month stay in the Angelo Arleo's group at the UPMC, partly funded by a mobility FPU grant of the Spanish Education Ministry. During this period we have developed and validated a more realistic cerebellar model embedded in a closed-loop control scheme able to recreate a VOR experiment using a simulated and real iCub robot in RT.

1.7 Chapter organization

This document has been organized in chapters for the sake of readability and organization. Each chapter can be described as follows:

- Chapter 1 is a brief introduction of computational neuroscience and cerebellar models applied to close-loop experiments using biomorphic robots. This chapter also describes the motivation of this thesis.
- Chapter 2 contextualizes the work presented in thesis regarding to previous works.
- Chapter 3 enumerates the main contributions of this thesis and future work.
- Chapter 4 is a compendium of the journal articles that support this thesis, including a brief description about the journals and the quality indexes of the publications. It also includes additional results pending for publication.
- Chapter 5 is the chapter 1 in Spanish.
- Chapter 6 is the chapter 3 in Spanish.
- Annex includes the four journal articles that compose this thesis.

Chapter 2: Thesis contextualization

The work developed in this thesis follows the research lines that our group at the University of Granada has developed in the last years. Our group aims the development of biologically inspired cerebellar models able to perform different motor control tasks with real robots in RT. All the journal articles that compose this thesis are oriented to this goal.

In the previous chapter we have put in context EDLUT. We have exposed the different simulation strategies that can be used (event-driven and time-driven), as well as a list of the most important simulation tools. We have also exposed the evolution of EDLUT, passing from an event-driven simulation scheme to a hybrid event- and time-driven simulation scheme. Finally, we have summarised all the new contributions to EDLUT compiled in the journal articles and additional results section included in this thesis.

In this chapter we put in context the cerebellar models and control schemes developed in this thesis, linking them with previous versions developed by our research group. Thus, we hope it will be easier for the reader to understand the cerebellar models presented in this thesis.

2.1 Previous cerebellar models

The cerebellar models and control schemes presented in this thesis are evolutions of the ones proposed by Dr. Niceto Luque (co-supervisor of this thesis) in his own thesis. A brief description of the work done by Dr. Luque in his thesis (Luque 2014) can help to better understand the work done in this one.

The main aim of Dr. Luque's thesis lied in the study and implementation of functional cerebellar models embedded in different controls schemes to perform motor control task with simulated robotics arms. This work was structured in four journal articles as follows.

The first article (Luque, Garrido, Carrillo, Coenen, et al. 2011) proposed a simplified model of a cerebellum formed by four different kinds of neurons (MFs, PCs, DCN and CFs) and one learning mechanism (a STDP rule in PFs driven by CFs). This model did not include GCs. It connected directly the MFs with the PCs, acting this synapses as the PFs. The main contribution of this article was the development of a STDP rule able to calibrate the sensorimotor relationship and compensate the sensory propagation delay in a spiking cerebellar model. This plasticity was implemented using LTP and LTD mechanisms. The LTP mechanism produced a fixed synaptic efficacy increase in the PFs that propagated spikes to a PC. The LTD mechanism produced a synaptic efficacy decrease in the PFs that propagated spike to a PC before the arrival to the same cell of a spike coming from a CF. This mechanism was able to compensate the propagation delay using a mathematical function with a peak in the propagation delay.

This cerebellar model was embedded in a control loop. For the experimental work they used a simulated version of a biomorphic robotic arm with low power actuators that tried to emulate a real human arm. They studied how the cerebellar model, thanks to the STDP mechanism, was able to build up corrective models to compensate deviation in the target trajectory of the robotic arm when the dynamics of the robot was altered due to manipulation of heavy objects (whose mass significantly affected the basic model dynamics).

The second article (Luque, Garrido, et al. 2011a) proposed a more realistic cerebellar model including a GC layer. As in the previous article, the cerebellar model was embedded in a control loop to manage the movement of a simulated robotic arm when it manipulated different objects that could modify its basic model dynamics. In this work they explored different configurations for the sensor representation supply to the cerebellum through the MFs (explicit, implicit and the combination of both ones), and how these one could be efficiently used for a corrective model abstraction corresponding to different manipulated objects.

The third article (Luque, Garrido, Carrillo, Tolu, et al. 2011) studied how the previous cerebellar model performing the same manipulation task could be embedded in diverse control loops: forward, recurrent, and a combination of both architectures. These ones were able to infer corrective models which compensated deviation in the robot trajectory when the dynamics and kinematics of the controlled robotic arm were altered due to the manipulation of different object. They also studied how the noise (related to the inherent noise of the muscle spindle signal) introduced to the cerebellum through the MFs could affect to the model. The main goal of this article was to make a comparative evaluation of these control architectures which showed how forward and recurrent architectures complement each other in the framework of manipulation task and how robustly they behaved in the presence of noise.

The last article (Luque et al. 2012) was a study of how the sensory-motor information in a common robot scenario could be encoded in an optimal representation that transfer from the robotic analogical domain to the biological digital (spikes) domain.

2.2 Cerebellar models embedded in closed-loop control scheme for robot control in real time

The human brain, thanks to the evolution during millions of years, has developed one of the best control systems known. During the last years it has emerged a new tendency that tries to emulate these biological control systems in order to design robotic controllers (Arbib, Metta, and van der Smagt 2008). Since it is known that the cerebellum is involved in control and learning of smooth coordinated movements, an accurate understanding of the cerebellar operation should help to improve biologically inspired control systems for biomorphic robots. As we have previously commented the best way to understand the biological mechanisms that confer the cerebellum its performance is studying it as part of a more complex system. For this reason we have embedded our cerebellar models in different control schemes able to perform motor control tasks using biomorphic robots (embodiment). Thus we can explore the effect that experimentally-supported characteristics have in the operation of biomorphic robots.

When the associative cortex generates the desired trajectory in body coordinates (Fig. 1), it is sent to the motor cortex and the cerebellum (reaching the PCs through the PFs, Fig. 2). Similarly, the cerebellum also receives sensory information coming from the spinal cord and other brain regions. The motor cortex and the cerebellum generate then the motor commands that are transported to the muscles. This propagation through the nerves is at a limited speed, producing a certain delay in the effective motor response (Flash and Hogan 1985). Once the muscles receive the motor commands, the contraction/relaxation is produced, generating in this way the movement.

Additionally, the sensorial system sends information to the brain regarding the consequence of the movements (e.g. the effective positions/velocities of the body along the movement). However, these signals also arrive to the brain with some propagation delay. The sensorial information is processed in several brain regions (including the cerebellum and the Inferior Olivary nucleus). It is thought that the IO cells compare the desired and real trajectories and generate teaching signals encoding the error of the movement (Fig. 1). The teaching signals arrive to the PCs in the cerebellum through the CFs (Fig. 2). Nevertheless, the CF activity conveying the teaching signal is delayed, due to consecutive brain-to-motor and sensorial-to-brain delays, with respect to the PF activity (representing desired trajectory information). Depending on the distance between the brain and the muscles, the total propagation delay (the sum of both propagation delays) can take values between 100 and 150 ms (Flash and Hogan 1985). It is believed that the cerebellum is able to compensate this propagation delay using a learning mechanism at PC level that correlates CF and PF activities. When a spike arrives to a PC through a CF at time T, it modifies the synaptic weights of all those PFs that propagated spikes to that PC using a convolution kernel with a peak in time T minus the total propagation delay (Luque, Garrido, Carrillo, Coenen, et al. 2011).

This propagation delay is a biological constraint due to the limited propagation speed of nerves. By contrast, the propagation speed of artificial systems is usually much higher and its magnitude depends on the system topology (where the control system and the robot are physically placed) and the technology used to communicate the system modules. In a normal configuration in which the control scheme and the robot interface are executed in two different computer in the same room connected through an Ethernet connection, the artificial propagation delay can be a few milliseconds.

In a closed-loop scheme able to control the movement of a robot, the control scheme must periodically send motor commands to the robot and the robot sends back information to the control system. A reasonable period for this communication is 2ms. This bidirectional communication can be implemented using two different communication techniques: synchronous and asynchronous communications.

In the synchronous approach, both elements (cerebellar model and robot interface) deploy the simulation at the same speed (both ones use the same “simulation time”). Thus, if one element blocks the time evolution (e.g. the cerebellar model has to process an activity peak), the other element will have to pause its operation (e.g. the robot does not receive motor commands). This is not a big deal if we work with simulated robots in which the time is another variable that can be conveniently set. However, this is a very important issue when we work with real robots. In this case the time is a physical variable that evolves at a constant speed (RT). We cannot allow the cerebellar model to block the simulation. We need a RT supervisor that ensures that each simulation time period of 2ms is performed in 2ms of RT (speeding-up the simulation by momentary disabling some neural elements such as spikes or plasticity mechanisms or speeding-down the simulation by halting it). This first option has been implemented in the third journal article included in this thesis (Luque et al. 2014).

By contrast, the asynchronous approach can take advantage of the difference between the biological and the artificial propagation delays. Since biologically inspired control schemes based on cerebellar models are able to deal with this long delay (between 50 and 75ms in each direction), we can use this feature in our own benefit by uncoupling the simulation of the control scheme (cerebellar model) and the robot interface for a period of time equivalent to the biological propagation delay (between 50 and 75ms) minus the artificial propagation delay (usually a few milliseconds). For instance, when we are controlling the movement of a robotic arm we may set the communication period to 2ms and the biological propagation delay to 50ms (100ms in total). If the artificial propagation delay (the time that our artificial system spends in connect our cerebellar model with the robot) is 10ms, there is a difference of 40ms between the biological and artificial propagation delays. By using buffers to accumulate the messages sent in both directions, we can uncouple the simulation time of both elements until 40ms (the simulation time of the cerebellar model must be always greater or equal than the simulation time of the robot, but the difference between both ones must be lower than 40ms). This asynchronous approach is harder to build than the synchronous one, but is the best option when we have to deal with real robots. In this case, the simulation time of the robot interface must evolve at the same speed that the physical time (RT). However, the simulation time of the cerebellar model has a margin of up to 40ms with respect to the RT. When the cerebellar activity is low, the cerebellar model can advance the simulation time up to 40ms with respect to the RT. When the cerebellar activity is high, the cerebellar model can use these 40ms to process all the activity without blocking the robot operation and without disabling any neural element (just are disabled when the simulation time is close to the RT boundary). This additional time reduces the RT exigencies with respect to the synchronous approach and allows easier implementation of the RT supervisor. This second choice has been implemented in the additional results section of this thesis.

2.3 New cerebellar models

The cerebellar models presented in this thesis have been developed in close collaboration with Dr. Luque (expert in this field). In the fourth article of this thesis (Luque et al. 2016) we have proposed two new plasticity mechanisms related with the DCN cells. Thus our cerebellar model implements plasticity mechanism in three different groups of synapses (PFs to PCs, MFs to DCN and PCs to DCN). The two new plasticity mechanisms have a dual functionality. The first one is to act as a gain adaptation mechanism able to optimize the working range of DCN cells. The second one allows the consolidation of the synaptic memories that are formed at PFs to PCs by transferring this synaptic topology to the MFs to DCN cells. This new cerebellar model has been tested over two different closed-loop schemes able to control the movement of a simulated robotic arm: (i) a feedback control loop to deliver corrective actions to compensate the existing difference between a controlled variable (real trajectory) and a demanding reference variable (desired trajectory), and (ii) a feedforward control loop to deliver corrective torque values to compensate for deviations in the crude inverse dynamic module when the robotic arm manipulates an object of significant weight along a predefined trajectory.

Additionally, the cerebellar model proposed in the previous article has been also used to recreate an EBCC experiment (Antonietti et al. 2016) at the Polytechnic of Milan. We have collaborated with this University in the framework of the European project REALNET. We have supplied the cerebellar model running in EDLUT that they have used to recreate the EBCC experiment.

Finally, the EBCC (already simulated with our cerebellar model) together with the VOR are broadly assumed as the paradigms that better reveal cerebellar learning. For that reason we have implemented in the last stage of this thesis a VOR experiment with a real robot (iCub) in RT. This work is included in the additional result section of this thesis. This is the first work in which we have been able to carry out an experiment using a real robot in RT. Taking the step from simulated robots to real robots has been an arduous task. We have implemented an asynchronous inner/outer closed-loop control scheme able to take advantage of the difference between the biological and artificial propagation delays in order to manage the movement of the iCub robot in RT. We have also implemented a RT supervisor similar to the one proposed in (Luque et al. 2014) able to ensure the RT restriction (in this case, since the simulation is asynchronous, the RT supervisor has to manage the time evolution of the cerebellar model and robot interface). Regarding the cerebellar model, we have included three new biological features in the new version proposed for this experiment: (i) a more complex neural model for the PCs able to recreate its tri-modal spike modes namely tonic, silence and bursting, (ii) a proper IO layer with electrical coupling between its neurons, and (iii) a new synapse from IO to DCN cells. All this new features have been tested in the VOR experiment.

Chapter 3: Conclusions and future work

This chapter shows a summary of the main contributions presented in this thesis, the conclusion that can be extracted and a proposal for future work.

3.1 Revisiting the thesis objectives

This thesis aims three complementary objectives:

- The development of the simulation tools that enable this study, exploring new and efficient simulation methods, control schemes, communication interfaces and RT supervisors. This has been fulfilled through the development of the EDLUT platform integrating different features that allow very efficient computation. A complete study of its performance capability and how it can be optimised has been carried out.
- Creating cerebellar models that explain the biological mechanisms allowing the cerebellum and some related centres to perform different motor control tasks. This objective is fulfilled by the development of cerebellar models which integrate new features such as distributed plasticity mechanisms (PFs to PCs driven by CFs, MFs to DCN cells driven by PCs and PCs to DCN cells), complex cell models (such as the tri-modal PC model), an IO layer able to better codify de error signal using electrical coupling and a specific synapse from IO to DCN layers.
- Implementation of novel biologically inspired control schemes able to control the new generation of biomorphic robots. The cerebellar models have been embedded in different closed-loop control schemes and have been exposed to demanding motor control tasks using simulated or real robots in RT. This has been done within biologically inspired control schemes, thus fulfilling this third objective.

3.2 Main contributions

- An ultra-fast spiking neural network simulator (EDLUT) based on hybrid event- and time-driven simulations has been developed and upgraded. A wide range of simulation techniques have been applied to this simulator with the idea of increasing its computational performance. Thus we have made possible the goal of performing motor control tasks with real robot in RT using biologically inspired cerebellar models.
  - ✓ The EDLUT kernel, able to perform hybrid event- and time-driven simulations, has been parallelized in multicore CPUs using OpenMP.
  - ✓ The time-driven simulation methods have been also parallelized in CPU-GPU co-processing platforms using CUDA. The GPU just computes the neural dynamic evolution while the spike generation and propagation is executed in the CPU.
  - ✓ New event-driven and time-driven simulation methods specialised in the simulation of relatively complex neural models (AdEx and HH) have been developed. A new mechanism able to re-organise the look-up tables together with a new protocol to generate and process synchronous activity have been developed for the event-driven methods. For the time-driven methods in CPU and GPU, a new bi-fixed-step integration method has been developed. This integration method is a hybrid between

  fixed-step and variable-step integration methods able to take advantage of the strengths and mitigate the weaknesses of both ones.
  - ✓ Several upgrading in the computation of the neural dynamic evolution, generation and propagation of spikes and updating of synaptic weight using different plasticity mechanism have been developed.
  - ✓ EDLUT kernel has been modified in order to allow the momentary disabling of neural elements (spikes generation and propagation, plasticity mechanism, neural dynamic evolution, etc.) to comply with the RT constraint imposed by real robots. This feature is controlled by a RT supervisor, which can speed-up or speed-down the simulation.
- Additional features beyond the simulation of spiking neural networks have been integrated in EDLUT.
  - ✓ It allows now the implementation of different closed-loop control schemes in which our cerebellar models are embedded using synchronous or asynchronous simulations.
  - ✓ A robot plant simulator able to simulate a wide range of robots.
  - ✓ A communication interface able to convert the analogical signals of robots to spikes that neural networks can process and vice versa.
  - ✓ A communication interface able to connect the control scheme with the robot interface of whatever simulated or real robot using TCP/IP connections.
- An incremental version of a biologically plausible cerebellar model has been developed.
  - ✓ Three different plasticity mechanisms have been proposed in our cerebellar model (PFs to PCs driven by CFs, MFs to DCN cells driven by PCs and PCs to DCN cells). These plasticity mechanisms generate three main contributions to the cerebellar model behaviour.
    - ▪ Calibrate in a first stage the sensorimotor relationship using the first plasticity mechanism at PC level.
    - ▪ Consolidate in a second stage the synaptic memory that is formed in the first plasticity mechanism at PC level by replicating this synaptic memory distribution in the second plasticity mechanism at DCN level.
    - ▪ The third plasticity mechanism acts as a gain adaptation mechanism able to optimize the working range of DCN cells.
  - ✓ A new complex neural model able to replicate the tri-modal spike modes of PCs has been included in the cerebellar model.
  - ✓ A proper IO layer implementing electrical coupling between its neurons has been included in the cerebellar model. This electrical coupling helps to better codifying the error signal.
  - ✓ A new synapse from IO to DCN cell has been evaluated.
- The cerebellar models proposed have been validated in different motor control and motor learning tasks.
  - ✓ Manipulation tasks of heavy/light objects that impact the basic model dynamics of a simulated robotic arm in RT.
  - ✓ Recreation of an EBCC experiment using a simulated environment.
  - ✓ Recreation of a VOR experiment using a simulated and a real iCub robot in RT.

3.3 Conclusions

In this thesis we have tried to bring some light to the motor control theory related to the cerebellum, proposing and studying cerebellar models based on hypothesis and results extracted from the biology. These cerebellar models have been evaluated in different motor control tasks using simulated and real robots (embodiment) in RT. Additionally, in this thesis we have developed and integrated in a single tool (EDLUT) all the elements required for these embodiment experiments.

When the simulation of spiking neural networks capable of interacting with simulated or real robots in RT is required, an efficient spiking neural network simulator with RT simulation capabilities is mandatory. With this idea in mind we have developed EDLUT, an open source spiking neural network simulator with a hybrid event- and time-driven simulation scheme parallelized in multicore CPU-GPU co-processing platforms and a RT supervisor. This hybrid scheme offers several simulation techniques that can be used conjointly to speed-up the simulation of different neural layers within the same neural networks (e.g. event-driven methods for the GC layer and time-driven methods for PC layer). Additionally, this simulator integrates a robotic software framework able to implement different closed-loop control schemes to manage simulated or real robots in RT.

Regarding the cerebellar models proposed in this thesis we have shown how a biologically inspired cerebellar model can be used to perform different motor controls tasks such as manipulation of heavy/light objects, EBCC and VOR experiments. Several new features have been proposed and studied in our cerebellar model: (i) Two additional plasticity mechanisms at DCN level able to consolidate the learning at Purkinje level and adjust the output range of DCN cells, (ii) a complex neural model for PCs able to recreate its tri-modal spike modes, (iii) a new IO layer with electrical coupling able to better codify the error, and (iv) a new synapses from IO to DCN cells. Nevertheless, we are yet quite far of fully understanding all the features of a cerebellum, being able to recreate a perfect cerebellar model. For that reason we need to follow with this study in the future.

3.4 Future work

The first step after this thesis will be the publication of all the results that are yet pending for publication. These ones will be divided in two separated (although related) articles. The first one will address the cerebellar model from the point of view of neuroscience. We will present how all the elements included in our cerebellar model are biologically plausible and how they contribute to the cerebellar model. The second article will be much more technical, focusing in the technical issues that we have had to deal with in order to recreate a VOR experiment using a simulated and real iCub robot in RT.

After that we will try to follow exploring the capabilities of our cerebellar model. Recent *in vivo* and *in vitro* studies (Bidoret et al. 2009, Bouvier et al. 2016) have observed how the plasticity mechanism of PFs to PCs is driven by the activity of CFs. These studies propose that the LTD and LTP mechanisms of PFs require a determined activity to modify the synaptic weight. This activity must be in bursts of at least 2 spikes for LTD or 5 spikes for LTP in order to induce a synaptic weight modification. Additionally, the spikes inside these bursts must have a high frequency (200Hz). For shorter bursts or with lower frequencies no synaptic weight modifications are observed. Additionally, the authors of these studies have proposed a mathematical model able to recreate this behaviour. In future versions of our cerebellar model we will replace our plasticity mechanism at PF level with this new version, evaluating how it can modify the dynamics of the model with respect to our previous version. Additionally, since this new learning rule requires the GC layer to generate bursts at high frequency, we will need to re-design a new GC layer able to meet these new requirements. With this goal in mind we will collaborate with Dr. Jesús Garrido (co-supervisor of this thesis and expert in the GC layer). He has extensively studied the MF, GC and Golgi cell (GoC) layers, evaluating different distributed plasticity mechanisms (Garrido et al. 2016, Mapelli et al. 2015, Luque, Garrido, et al. 2011a, D'Angelo et al. 2009, D’Angelo et al. 2016). We will integrate his MF, GC and GoC layers with our PC, DCN and IO layers, thus building our most complex and biologically plausible cerebellar model. Finally we will include in our cerebellar model two new types of interneurons: basket and stellate cells. These cells receive excitatory synapses from the PFs and inhibit the PCs. The working hypothesis is that these layers help to better differentiate the PF patters at PC level and optimise the working range of PCs.

Regarding the development of our simulator EDLUT, we have drastically improved in this thesis the computational performance of this tool, making it capable of simulate more complex cerebellar models in RT. In the next steps after this thesis we are planning to follow with the development of our cerebellar model, increasing its neural and mathematical complexity, but at the same time we also want to perform RT experiments with real robots. As we have seen in this thesis, the combination of both options at the same time will require to go on with the development of EDLUT, exploring new ways to increase its computational performance. In this sense we will explore the parallelization of EDLUT in clusters, but taking always into account the RT constraint that real robots impose.

We will also explore how to optimise the new plasticity mechanism at PF level. This new plasticity mechanism uses two state variables and one differential equation for each PF. The state variables experiment a direct increment each time a spike arrives thought the PF or CF respectively and an exponential decrement over the time. In our last cerebellar model we have implemented 400,000 PFs. Thus we will have to evaluate 800,000 exponential decrement functions and 400,000 differential equations, generating a huge computational workload. We will implement and compare three different configurations: fixed-step and variable-step integration methods in CPU and fixed-step integration methods in CPU-GPU co-processing platforms (the direct increment of the state variables will be computed in CPU, while the exponential decrement and the differential equation integration will be computed in GPU).

Finally, as we have previously expressed, we will further evaluate the new features of our cerebellar model when it is involved in different motor control task using simulated or real robots in RT.

Chapter 4: Results

This chapter summarises the results obtained during this thesis. The chapter is divided in two main sections; published results and pending-for-publication results. The first section enumerates and ranks the journal articles obtained whereas the second section presents a more in depth explanation of those results that are to be published.

4.1 Published journal articles included in this thesis

1. **A Spiking Neural Simulator Integrating Event-Driven and Time-Driven Computation Schemes Using Parallel CPU-GPU Co-Processing: A Case Study.**

   ***Naveros, F.**, Luque, N. R., Garrido, J. A., Carrillo, R. R., Anguita, M., & Ros, E. (2015). A spiking neural simulator integrating event-driven and time-driven computation schemes using parallel CPU-GPU co-processing: a case study. IEEE transactions on neural networks and learning systems, 26(7), 1567-1574.*

   Status: Published in 2015

   Impact Factor (JCR 2015): 4.854

   Category:

   - Computer Science, Artificial Intelligence. Ranking 7/130, Quartile in Category: **Q1.**
   - Computer Science, Hardware & Architecture. Ranking 1/51, Quartile in Category: **Q1.**
   - Computer Science, Theory & Methods. Ranking 3/105, Quartile in Category: **Q1**.
   - Engineering, Electrical and Electronic. Ranking 10/257, Quartile in Category: **Q1**.

2. **Event- and Time-Driven Techniques Using Parallel CPU-GPU Co- Processing for Spiking Neural Networks.**

   ***Naveros, F.**, Garrido, J. A., Carrillo, R. R., Ros, E., & Luque, N. R. (2017). Event-and Time-Driven Techniques Using Parallel CPU-GPU Co-processing for Spiking Neural Networks. Frontiers in Neuroinformatics, 11.*

   Status: Accepted in 2017

   Impact Factor (JCR 2015): 3,047

   Category:

   - Mathematical & Computational Biology. Ranking 6/56, Quartile in Category: **Q1.**
   - Neurosciences. Ranking 107/256, Quartile in Category: Q2.

3. **Integrated neural and robotic simulations. Simulation of cerebellar neurobiological substrate for an object-oriented dynamic model abstraction process.**

   *Luque, N. R., Carrillo, R. R., **Naveros, F.**, Garrido, J. A., & Sáez-Lara, M. J. (2014). Integrated neural and robotic simulations. Simulation of cerebellar neurobiological substrate for an* *object-oriented dynamic model abstraction process. Robotics and Autonomous Systems, 62(12), 1702-1716.*

Status: Published in 2014

Impact Factor (JCR 2014): 2,326

Subject Category:

- Automation & Control Systems. Ranking 31/52, Quartile in Category: Q3
- Computer Science, Artificial Intelligence. Ranking 68/123, Quartile in Category: Q3
- Robotics. Ranking 11/23, Quartile in Category: Q2

4. **Distributed Cerebellar Motor learning: A Spike-Timing-Dependent Plasticity Model.**

*Luque, N. R., Garrido, J. A., **Naveros, F.**, Carrillo, R. R., D'Angelo, E., & Ros, E. (2016). Distributed cerebellar motor learning: a spike-timing-dependent plasticity model. Frontiers in computational neuroscience, 10.*

Status: Published in 2016

Impact Factor (JCR 2015): 2,635

Subject Category:

- Mathematical & Computational Biology. Ranking 7/56, Quartile in Category: **Q1**.
- Neurosciences. Ranking 133/256, Quartile in Category: Q3.

4.1.1 Other published journal articles in collaboration with EU partners

5. **Spiking Neural Network With Distributed Plasticity Reproduces Cerebellar Learning in Eye Blink Conditioning Paradigms.**

*Antonietti, A., Casellato, C., Garrido, J. A., Luque, N. R., **Naveros, F.**, Ros, E., D'Angelo, E., Pedrocchi, A. (2016). Spiking neural network with distributed plasticity reproduces cerebellar learning in eye blink conditioning paradigms. IEEE Transactions on Biomedical Engineering, 63(1), 210-219.*

Status: Published in 2016

Impact Factor (JCR 2015): 2,468

Subject Category:

- Engineering, Biomedical. Ranking 22/76, Quartile in Category: Q2.

4.1.2 International peer-reviewed proceedings

1. **CPU-GPU hybrid platform for efficient spiking neural-network simulation.**

***Naveros, F.**, Luque, N. R., Garrido, J. A., Carrillo, R. R., & Ros, E. (2013). CPU-GPU hybrid platform for efficient spiking neural-network simulation. BMC Neuroscience, 14(1), P328. This is a contribution to the CNS 2013, Paris.*

4.2 Results pending for publication

As the reader may envisage, the work here presented is the result of a gradual process whose zenith is yet to come. The ultimate goal of the thesis here presented is to integrate the cerebellar circuit modelling within RT “behavioural and cognitive tasks”. This is a compromise approach based on the assumption that most cerebellar functions involve the cooperative computation of several neural subcircuits.

Many computational models about the cerebro-cerebellar loop have been proposed since (Marr 1969, Albus 1971), thus providing elegant explanations about the core of the forward controller operation that cerebro-cerebellar loop seems to carry out. Nevertheless, these computational theories tend to focus in one part of the cerebellar circuitry and then extrapolating the obtained conclusions to the completely cerebro-cerebellar system. It is also true that functional features are not either suddenly going to emerge from modelling all the cerebellar parts together since small deviations in many of the estimated network parameters (Sporns 2006) can cause large deviations in resultant global behaviour. But, it is also true that simulating nervous systems "connected" to a body (agent or robot with sensors and actuators) can be of great interest for studying how certain capabilities of the nervous system (e.g. the role of the cerebellum in coordinated movements and object manipulation) are based on cellular characteristics, nervous system topology or local synaptic adaptation mechanisms. This last work presents an **integrative approach** that tries to build the bridge between task specific experimentation and Systems Neuroscience models.

This integrative approach allows studying the role of certain nervous systems under what it is called "behavioural/cognitive tasks". This is closely related to the concept of "embodied cognition" (Pfeifer and Bongard 2007), in which the main aim of the CNS is to solve and facilitate the body interaction with the environment. Therefore, it is crucial to study nervous system models within the framework of its interaction with a body (sensors and actuators) and environment.

To that aim, the ingredients for our embodied cognition approach are:

1. The vestibulo-ocular reflex (VOR) experiment as our behavioural/cognitive task. The VOR is a reflexive eye movement largely used for testing cerebellar malfunctions.
2. The cerebellar model as the neural structure responsible for facilitating the body interaction.
3. The humanoid iCub robot as the front end human-like body.

In subsequence sections, we explain how each ingredient has been implemented, how they operate and how they have been put together to work as a whole (embodied system). Here we have included a brief summary of the work done to avoid any copyright infringements with the eventual publications.

4.2.1 What is the vestibulo-ocular reflex (VOR)?

The VOR is a reflexive eye movement that stabilises the images on the retina during head rotations by contralateral eye movements that maintain the image in the centre of the visual field (Fig. 3A). The VOR is effective up to head direction speeds of 50 deg/sec, and its latency period is extremely short (~10 ms). The VOR depends on the vestibular system, which detects head rotation. Therefore, VOR does not depend on vision and it works both in light and darkness conditions. The vestibular system can detect both rotational and translational head movements through the stimulation of semicircular canals and otolithic organs respectively (Rabbitt, Damiano, and Grant 2004). Thus, both angular and linear movements elicit the VOR: rotational (r-VOR) and translational (t-VOR). Head oscillations can involve a combination of translational and rotational movements, which results in combined r- and t-VOR.

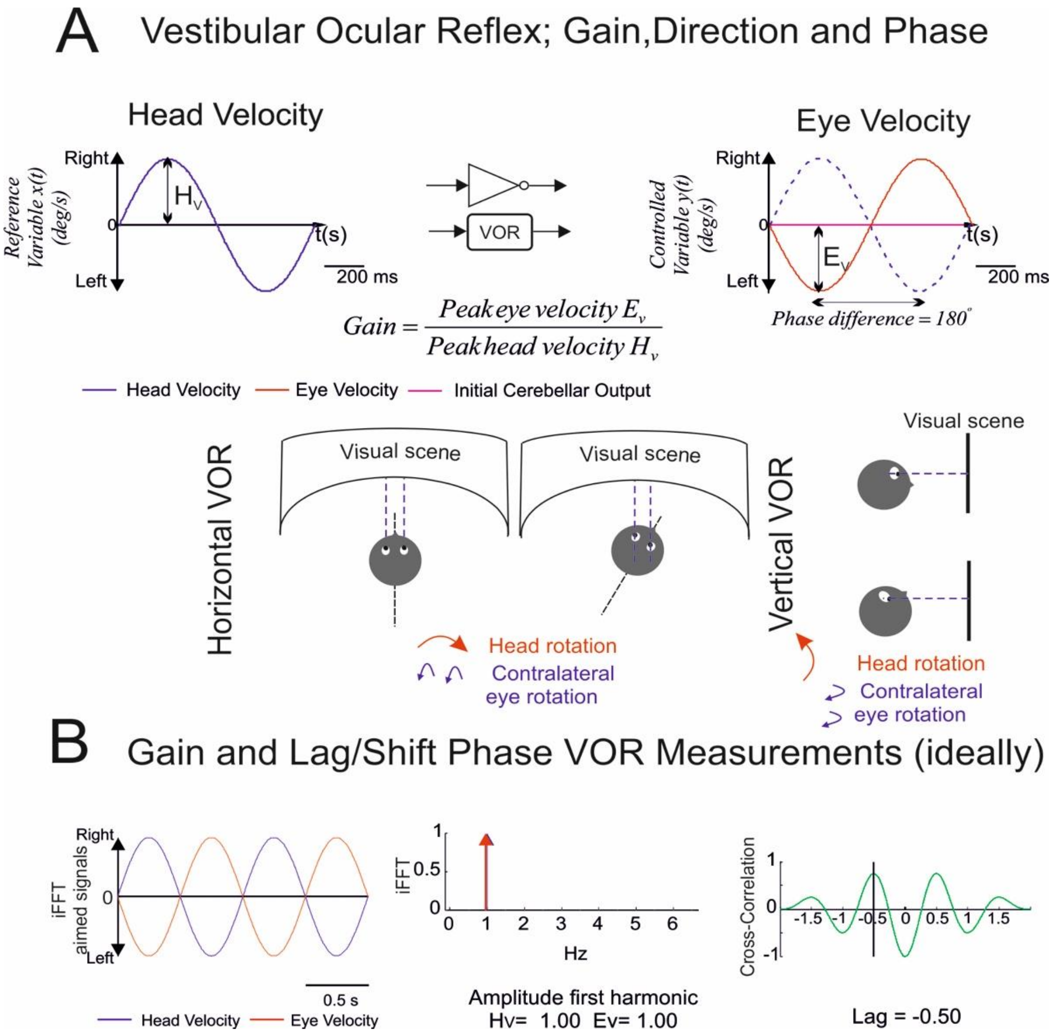


Figure 3 ***Vestibular Ocular Reflex (VOR).***

(A) *VOR stabilises the images on the retina during head rotations by contralateral eye movements that maintain the image still on the fovea.* ***(B)*** *In VOR, head movements are compared against induced eye movements using the VOR-gain and the VOR-phase. The VOR-gain is calculated by using a Fourier analysis. VOR-gain is the ratio of the first harmonic amplitudes obtained from the input signal and the actual cerebellar output signal. The lag of the VOR-phase is calculated by using cross-correlation between the reference variable (input) and the actual cerebellar output.*

In head oscillation tests, head and eye movements are usually compared thanks to a sine-wave stimulus used as reference variable (head velocity) and controlled variable (eye velocity). The temporal difference between these two curves, called VOR phase shift, is traditionally given in degrees whilst the ratio of amplitude of eye rotation and head rotation, called VOR gain, is non-dimensional (Fig. 3B). For natural head rotation frequencies (0.5-5.0 Hz), the VOR gain is close to 1.0 whilst phase shift is close to 180 degrees. That is, equally-sized head and eye movements are in counter-phase as they are synchronously occurring in opposite directions (Leigh and Zee 2015).

VOR's nature is purely feed-forward since it induces prompt compensatory eye movements as consequence of head movements. VOR is mediated by a control system in which adaptation is directly driven by sensorimotor errors: the cerebellum (Figure 4). The existing mismatch between head movements (signalled by the vestibular organ) and the incoming information to the cerebellum about eye movements represents sensory errors, which are called retinal slips. The feed-forward adaptive control mediated by the cerebellum aims at minimising these retinal slips.

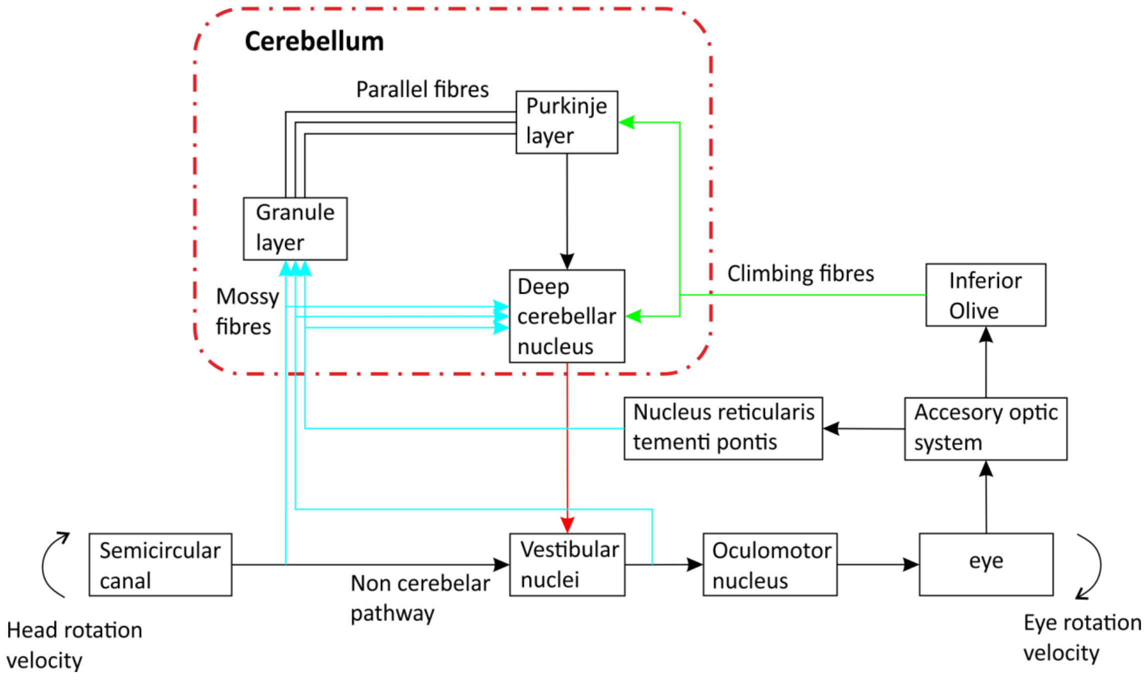


Figure 4 ***Vestibular and cerebellar scheme.***
Connections from the vestibular organ via the vestibular nucleus to the oculomotor nucleus forming the three-neuron reflex arc and connections to the cerebellum.

The VOR, together with EBCC, are broadly assumed as the paradigm that better reveals cerebellar learning.

4.2.2 Cerebellar spiking neural network model

The cerebellar model is a complete upgrading of (Luque et al. 2016). It is modelled as a forward controller capable of compensating head movements by producing contralateral eye movements. The connectivity and the topology of the simulated cerebellar network involve five neural sub-populations (Eccles, Ito, and Szentágothai 1967, Ito 1984, Voogd and Glickstein 1998, Medina and Mauk 1999, 2000). The population of MFs conveys the sensory signals from the vestibular organ and the eyes, providing the input to the cerebellar network. MFs project excitatory afferents onto GC and DCN. PCs integrate the input from the PFs (i.e. the axons of GCs) as well as the error signal (difference between the head and eye movements) from the CFs (i.e. the axons of IO cells). Finally, the DCN cells generate the output activity of the cerebellum. DCNs close the cerebellar loop with the excitatory synapses coming from the MFs and the IO together with the inhibitory synapses coming from the PCs.

Mossy fibres (MFs). 100 MFs have been modelled as LIF neurons. These fibres generate excitatory synapses that connect with all DCN cells. MF activity is generated following a sinusoidal shape (1Hz with a step size of 0.002 ms) to encode head movements (Lisberger and Fuchs 1978, Arenz et al. 2008, Clopath et al. 2014) consistently with the functional principles of VOR in cerebellar-control (Clopath et al. 2014). The overall MF activity consists of non-overlapping activations of equally sized neural subpopulations that allow a constant firing rate.

Granular cells (GCs). The granular layer includes 2000 GCs implementing a state generator (Yamazaki and Tanaka 2005, 2007, 2009, Honda et al. 2011). This means that the granular layer generates a sequence of active neuron populations without recurrence in the presence of a constant MF input (Fujita 1982). The granular layer generates non-overlapped spatiotemporal patterns that are repeatedly activated in the same sequence during each one-second learning trial. The passage of time is represented by 500 different states that consist of four activated non-overlapped GCs per time-step (2ms).

Purkinje cells (PCs). PCs have been modelled using a detailed neural model consisting of a single compartment with five ionic currents (two groups of 100 cells for agonist/antagonist muscles). This new model is able to replicate the complex spikes observed in PCs. THE previous PC model (Luque et al. 2016) did not exhibit the tri-modal spike modes namely tonic, silence and bursting (Forrest 2008)).

Inferior olive cells (IOs) and climbing fibres (CFs). 200 IO cells have been modelled as LIF neurons (two groups of 100 cells for agonist/antagonist muscles). Each CF, from an IO cell, contacts with one PC and one DCN. IO cells are interconnected via gap-junctions whose electrical coupling activates ensembles of IO cells. Gap-junctions are electrically coupled following preferred directions. There is not always a direct correlation between vicinity and the direction of the electrical coupling transmission within the inferior olive network (Devor and Yarom 2002). The nearest IO cells are not always the cells that are coupled in the first place. We have tried to mimic this “preferred directions” by using a certain IO network topology. Each group of 100 cells is divided in four 5x5 squares. The preferred paths are disposed radially from one corner of each 5x5 square to the other three corners. Finally, the external input activity of IO cells has been generated with a probabilistic Poisson process. Given the normalised error signal $\varepsilon(t)$ and a random number $\eta(t)$ between 0 and 1, an IO cell receives an input spike if $\varepsilon(t)>\eta(t)$ (Boucheny et al. 2005, Luque, Garrido, et al. 2011b). These input stimuli together with the electrical coupling between IO cells generate the CF activity. Thus, a single CF spike encodes well-timed information regarding the instantaneous error. Furthermore, the probabilistic spike sampling of the error ensures a proper representation of the whole error region over trials, while maintaining the CF activity below 10 Hz per fibre (similar to electrophysiological data (Kuroda et al. 2001)). The evolution of the error can be sampled accurately even at such a low frequency (Carrillo et al. 2008, Luque, Garrido, et al. 2011b).

Deep Cerebellar Nuclei (DCN) cells. 200 DCN cells have been modelled as LIF neurons (two groups of 100 cells for agonist/antagonist muscles). Each DCN cell receives an inhibitory afferent from a PC and an excitatory afferent from the IO cell that is also contacting that PC. DCN cells also receive excitatory projections from all MFs (which determine the baseline DCN activity). Thus, the subcircuit IO–PC–DCN is organised in a single microcomplex. DCN spike trains are translated into analogue output signals through averaging the spiking output given by each DCN subpopulation (one subpopulation for each agonist/antagonist group of muscles) (Eqs. 1 and 2):

$$DCN_i(t) = \int_{t}^{t+T_{step}} \delta_{DCN_{spike}}(t) \cdot dt \tag{1}$$

$$DCN_{output}(t) = \sum_{i=1}^{N=100} DCN_i(t) \tag{2}$$

We have summarised in Table 1 the neural topology used by our cerebellar model.

Table 1. ***Neural network topology***

Neurons		*Synapses*			
Pre-synaptic cells (number)	*Post-synaptic cells (number)*	*Number*	*Type*	*Initial weight*	*Weight range*
2000 GC	*200 PC*	*400000*	*AMPA*	*4*	*[0, 10]*
200 IO	*200 PC*	*200*	*AMPA*	*40*	*-*
100 MF	*200 DCN*	*20000*	*AMPA*	*0*	*[0, 1]*
200 PC	*200 DCN*	*200*	*GABA*	*1.5*	*-*
200 IO	*200 DCN*	*200*	*NMDA*	*7*	*-*
IO to IO		*320*	*EC*	*5*	*-*

4.2.3 Neural models

DCN cell model

DCN cells are modelled as LIF neurons with excitatory (AMPA and NMDA) and inhibitory (GABA) chemical synapses (Eqs. 3 to 9).

$$C_m \frac{dV}{dt} = I_{internal} + I_{external} \tag{3}$$

$$I_{internal} = -g_L \cdot (V + E_L) \tag{4}$$

$$I_{external} = -(g_{AMPA}(t) + g_{NMDA}(t) \cdot g_{NMDA\_INF}) \cdot (V - E_{AMPA}) - g_{GABA}(t) \cdot (V - E_{GABA}) \tag{5}$$

$$g_{AMPA}(t) = g_{AMPA}(t_0) \cdot e^{\frac{(t-t_0)}{\tau_{AMPA}}} \tag{6}$$

$$g_{NMDA}(t) = g_{NMDA}(t_0) \cdot e^{\frac{(t-t_0)}{\tau_{NMDA}}} \tag{7}$$

$$g_{GABA}(t) = g_{GABA}(t_0) \cdot e^{\frac{(t-t_0)}{\tau_{GABA}}} \tag{8}$$

$$g_{NMDA\_INF} = \frac{1}{1 + e^{-62 \cdot V} \cdot \frac{1.2}{3.57}} \tag{9}$$

Where C_m denotes de membrane capacitance, V the membrane potential, $I_{internal}$ the internal currents and $I_{external}$ the external currents. E_L is the resting potential and g_L the conductance responsible for the passive decay term toward the resting potential. Conductaces g_{AMPA}, g_{NMDA} and g_{GABA} integrate all the contributions received by each receptor type (AMPA, NMDA, GABA) through individual synapses. These conductances are defined as decaying exponential functions (Gerstner and Kistler 2002, Ros et al. 2006).

IO cell model

IO cells are modelled as LIF neurons with excitatory (AMPA) and inhibitory (GABA) chemical synapses and electrical coupling (EC) synapses (Eqs. 10 to 15). From a functional point of view, the interneuron communication through electrical synapses differs significantly from the communication through chemical synapses. The main difference lies in the communication speed. Chemical synapses hold a significant synaptic delay. There is an elapsed time since the action potential reaches the presynaptic connection and the released neurotransmitter interacts with the receptor producing the response in the postsynaptic cell (a few milliseconds). By contrast, the electrical synapse delay is negligible. This high speed in intercellular communication allows the simultaneous functional coupling (synchronisation) of neural networks interconnected by electrical synapses.

The electrical-synapse intercellular channels allow the bidirectional flow of ions in almost any situation. Most of the intercellular channels conforming the electrical synapses are voltage dependent, that is, their conductance varies according to the difference of potential on both sides of the membranes conforming the electrical bond. In some specialised gap junctions, this “sensitivity” to the voltage difference allows the conduction of the depolarising currents in just one direction (rectifying electrical synapses). IO cells exhibit this peculiar electrophysiological property. They are coupled by electrotonic gap junctions located on the dendrites (Llinas, Baker, and Sotelo 1974, Sotelo, Llinas, and Baker 1974).

$$C_m \frac{dV}{dt} = I_{internal} + I_{external} \tag{10}$$

$$I_{internal} = -g_L \cdot (V + E_L) \tag{11}$$

$$I_{external} = -g_{AMPA}(t) \cdot (V - E_{AMPA}) - g_{GABA}(t) \cdot (V - E_{GABA}) - I_{EC} \tag{12}$$

$$g_{AMPA}(t) = g_{AMPA}(t_0) \cdot e^{\frac{(t-t_0)}{\tau_{AMPA}}} \tag{13}$$

$$g_{GABA}(t) = g_{GABA}(t_0) \cdot e^{\frac{(t-t_0)}{\tau_{GABA}}} \tag{14}$$

$$I_{EC} = \sum_{i=N}^{N} w_i \cdot (V - V_i) \cdot \left(0.6 \cdot e^{-\frac{(V-V_i)^2}{50^2}} + 0.4\right) \tag{15}$$

Where C_m denotes de membrane capacitance, V the membrane potential, $I_{internal}$ the internal currents and $I_{external}$ the external currents. E_L is the resting potential and g_L the conductance responsible for the passive decay term toward the resting potential. Conductaces g_{AMPA} and g_{GABA} integrate all the contributions received by each chemical receptor type (AMPA, GABA) through individual synapses. These conductances are defined as decaying exponential functions (Gerstner and Kistler 2002, Ros et al. 2006). Finally, I_{EC} represent the total current injected through the EC synapses (Schweighofer, Doya, and Kawato 1999), where w_i denotes the synaptic weight of the synapses between the neuron i and the target neuron, V the membrane potential of the target neuron and V_i the membrane potential of neuron i. N is the total number of input synapses received by the target neuron. Finally, for a correct operation of the electrical synapses, this model needs to emulate the depolarization and hyperpolarization phases of an action potential. We have incorporated a simple threshold process that enables the generation of a triangular voltage function each time the LIF neuron fires (Bezzi et al. 2004).

Purkinje cell model

Purkinje cells present three spiking modes: tonic, bursting and silence (Forrest 2008). PF excitatory inputs drive PCs into its tonic/silence modes whereas CF excitatory inputs drive PCs into its bursting/silence modes. Simple spikes of PCs are elicited typically at high frequencies (Thach 1967, Raman and Bean 1999) and complex spikes at low frequencies. Complex spikes consist of a fast initial large-amplitude spike followed by a high-frequency burst (Schmolesky et al. 2002). This burst is made of several slower spikelets of smaller amplitude separated from one another by 2 - 3ms (Schmolesky et al. 2002, Najafi and Medina 2013, Eccles, Ito, and Szentágothai 1967). After each burst, a spike pause prevents PCs from resuming either their tonic or bursting firing for a period that depends on the length of the complex spike (Mathy et al. 2009).

The detailed PC model is based on (Miyasho et al. 2001, Middleton et al. 2008) and consists of a single compartment with five ionic currents and two excitatory (AMPA) and inhibitory (GABA) chemical synapses (Eqs. 16 to 22).

$$C_m \frac{dV}{dt} = I_{internal} + \frac{I_{external}}{Membrane\ Area} \tag{16}$$

$$\begin{aligned} I_{internal} = & -g_k \cdot n^4 \cdot (V+95) - g_{Na} \cdot m_0[V]^3 \cdot h \cdot (V-50) - g_{Ca} \cdot c^2 \cdot (V-125) - g_L(V+70) \\ & - g_M \cdot M \cdot (V+95) \end{aligned} \tag{17}$$

$$I_{external} = -g_{AMPA}(t) \cdot (V - E_{AMPA}) - g_{GABA}(t) \cdot (V - E_{GABA}) \tag{18}$$

$$g_{AMPA}(t) = g_{AMPA}(t_0) \cdot e^{\frac{(t-t_{0)}}{\tau_{AMPA}}} \tag{19}$$

$$g_{GABA}(t) = g_{GABA}(t_0) \cdot e^{\frac{(t-t_{0)}}{\tau_{GABA}}} \tag{20}$$

Where V denotes the membrane potential, $I_{internal}$ the internal currents and $I_{external}$ the external currents. The membrane capacitance (C_m) and the membrane area are defined in Table 2. Conductaces g_{AMPA} and g_{GABA} integrate all the contributions received by each chemical receptor type (AMPA, GABA) through individual synapses. These conductances are defined as decaying exponential functions (Gerstner and Kistler 2002, Ros et al. 2006). Finally, g_K is a delayed rectifier potassium current, g_{Na} a transient inactivating sodium current, g_{Ca} a high-threshold non-inactivating calcium current, g_L a leak current, and g_M a muscarinic receptor suppressed potassium current (see Table 3).

Table 2. ***Geometrical parameters.***

Geometrical parameters	
Cylinder length of the soma	*15µm*
Radius of the soma	*8 µm*
Membrane Capacitance (C_m)	*1 µF/cm^2*
Axial resistivity	*100 Ω/cm (axon) 250 Ω/cm (dendrites)*
Number of segments	*1*

Table 3. ***Ionic conductance densities.***

Conductance type	*Soma (mho/cm^2)*
gK delayed rectifier potassium current	*0.01*
gNa transient inactivation sodium current	*0.125*
gCa high threshold	*0.001*
gM muscarinic receptor	*0.75*
gL leak current (anomalous rectifier)	*0.02*

The dynamics evolution of each gating variable (n, h, c, and M) can be computed with the following differential equation:

$$\dot{x} = \frac{x_0[V] - x}{\tau_x[V]} \tag{21}$$

where x indicates the variables n, h, c, and M. The implemented equilibrium function is determined by the term $x_0[V]$ and time constant $\tau_x[V]$ (Table 4).

Table 4. ***Ionic conductance kinetic parameters.***

Conductance type	*Steady–state Activation/Inactivation*	*Time constant (ms)*
g_K delayed rectifier potassium current	$x_0[V] = \frac{1}{1 + e^{\frac{-V-29.5}{10}}}$	$\tau_x[V] = 0.25 + 4.35 \cdot e^{\frac{-\lvert V+10 \rvert}{10}}$
g_{Na} transient inactivating sodium current	$x_0[V] = \frac{1}{1 + e^{\frac{V-59.4}{10.7}}}$	$\tau_x[V] = 0.15 + \frac{1.15}{1 + e^{\frac{V+33.5}{15}}}$
$\boldsymbol{m_0[V]}$	$m_0[V] = \frac{1}{1 + e^{\frac{-V-48}{10}}} \cdot m$	
	Forward Rate Function (α)	***Backward Rate Function (β)***
g_{Ca} high threshold	$\alpha = \frac{1.6}{1 + e^{-0.0072(V-5)}}$	$\beta = \frac{0.02 \cdot (V + 8.9)}{e^{\frac{V+8.9}{5}}}$
g_M muscarinic receptor suppressed potassium current	$\alpha = \frac{0.3}{1 + e^{\frac{-V-2}{5}}}$	$\beta = 0.001 \cdot e^{\frac{-V-70}{18}}$
	Steady–state Activation/Inactivation	***Time constant(ms)***
	$x_0[V] = \frac{\alpha}{\alpha + \beta}$	$\tau_x[V] = \frac{1}{\alpha + \beta}$

The sodium activation variable has been replaced and approximated by its equilibrium function $m_0[V]$. The M-current presents a temporal evolution significantly slower than the rest of variables. Each internal spike in the neuron generates a fast increase of M-current that takes several milliseconds to return to its stable state. A high M-current prevents the PC from entering in its tonic mode (when the neuron generates spikes due to the PFs activity). Complex spike cause an M-current rapid increase that depends, in turn, on the size of the spikelet within the burst. PC tonic mode resumes when M-current decreases.

First, we have validated the detailed PC model in the NEURON simulator. Subsequently, we have reduced the PC model for fast spiking neural network simulation. In the reduced PC model, I_K and I_{Na} currents are implemented through a simple threshold process that triggers the generation of a triangular voltage function each time the neuron fires (Bezzi et al. 2004). This triangular voltage depolarisation drives the state of ion channels similarly to the original voltage depolarisation during the spike generation. The final internal current is:

$$I_{internal} = -g_{Ca} \cdot c^2 \cdot (V - 125) - g_L(V + 70) - g_M \cdot M \cdot (V + 95) \tag{22}$$

4.2.4 Synaptic plasticity

The overall input-output function of the cerebellar network model has been made adaptive through STDP mechanism at different sites. These STDP mechanisms balance long-term potentiation (LTP) and long-term depression (LTD) (see (Luque et al. 2016) for an in-depth review of the implemented synaptic mechanisms).

PFs–PCs synaptic plasticity: The LTD/LTP balance at PFs–PCs synapses is based on the following rules (Eqs. 23 and 24):

$$LTD\ \Delta w_{PF_j-PC_i}(t) = \int_{-\infty}^{IO_{spike}} k\left(\frac{t - t_{IO_{spike}}}{\tau_{LTD}}\right) \cdot \delta_{GC_{spike}}(t) \cdot dt \qquad if\ PF_j\ is\ active\ at\ t \tag{23}$$

$$LTP\ \Delta w_{PF_j-PC_i}(t) = \propto \qquad always \tag{24}$$

where *$\Delta W_{PFj-PCi}(t)$* denotes the weight change between the *j*th PF and the target *i*th PC; τ_{LTD} is the time constant that compensates the sensorimotor delay; δ_{GC} is the delta Dirac function corresponding to an afferent spike from a PF; and the kernel function *k(x)* is defined as:

$$k(x) = e^{-x} \cdot \sin(x)^{20} \tag{25}$$

The effect on the presynaptic spikes arriving through PFs is maximal over the 100 ms time window before CF spike arrival, thus accounting for the sensorimotor pathway delay (Kawato and Gomi 1992, Luque, Garrido, et al. 2011a, Luque, Garrido, et al. 2011b, Luque, Garrido, Carrillo, Tolu, et al. 2011). Note that the kernel *k(x)* allows the computation to be run on an event-driven simulation scheme as EDLUT (Ros et al. 2006, Luque, Garrido, et al. 2011a, Luque, Garrido, et al. 2011b, Luque, Garrido, Carrillo, Tolu, et al. 2011), which avoids integrating the whole kernel upon each new spike arrival. Finally, as shown in Eq. 24, the amount of LTP at PFs is fixed, with an increase in synaptic efficacy equal to α each time a spike arrives through a PF to the targeted PC.

MFs–DCN synaptic plasticity: The LTD/LTP dynamics at MFs – DCN synapses is based on the following rules (Eqs. 26 and 27):

$$LTD\ \Delta w_{MF_j-DCN_i}(t) = \int_{-\infty}^{\infty} k\left(\frac{t - t_{PC_{spike}}}{\sigma_{MF-DCN}}\right) \cdot \delta_{MF_{spike}}(t) \cdot dt \qquad if\ MF_j\ is\ active\ at\ t \tag{26}$$

$$LTP\ \Delta w_{MF_j-DCN_i}(t) = \propto \qquad always \tag{27}$$

with *$\Delta W_{MFj-DCNi(t)}$* denoting the weight change between the *j*th MF and the target *i*th DCN. $\sigma_{MF\text{-}DCN}$ standing for the temporal width of the kernel; δ_{MF} representing the delta Dirac function that defines a MF spike; and the integrative kernel function *k(x)* defined as:

$$k(x) = e^{-|x|} \cdot \cos(x)^2 \tag{28}$$

Note that there is no needs to compensate the sensorimotor pathway delay at this site because it is already done in the previous learning rule (τ_{LTD} in Eq. 23).

The STDP rule defined by Eq. 26 produces a synaptic efficacy decrease (LTD) when a spike from the PC reaches the targeted DCN neuron. The amount of synaptic decrement (LTD) depends on the activity arrived through the MFs. This activity is convolved with the integrative kernel defined in Eq. (28). This LTD mechanism considers those MF spikes that arrive after/before the PC spike arrival within the time window defined by the kernel. The amount of LTP at MF - DCN synapses is fixed, with an increase in synaptic efficacy equal to α each time a spike arrives through a MF to the targeted DCN.

4.2.5 VOR plant model

The VOR plant model implemented represents the VOR as a continuous-time mathematical model with two poles, whose parameters are adjusted by means of recursive methods to fit experimental and clinical observations (Skavenski and Robinson 1973, Robinson 1981, Gordon, Furman, and Kamen 1989):

$$e(kT), E(s)\text{: } eye\ motion\ (output) \tag{29}$$

$$h(kT), H(s)\text{: } head\ motion\ (input) \tag{30}$$

$$VOR(s) = \frac{E(s)}{H(s)} = \frac{K \cdot T_{c1} \cdot s}{(T_{c1} \cdot s + 1) \cdot (T_{c2} \cdot s + 1)} \cdot e^{-s\tau_{delay}} \tag{31}$$

There are four parameters in the model: *$Q=[K, T_{C1}, T_{C2}, \tau_{delay}]$*. The delay parameter τ_{delay} captures the fact that there exists some delay in communicating the signals from the inner ear to the brain and eyes. This delay is consequence of the time needed for chemical neurotransmitters to traverse the synaptic clefts between nerve cells. Based on the number of synapses involved in the VOR, this delay is expected to be around 5 ms (Skavenski and Robinson 1973, Robinson 1981). The gain parameter K models the fact that the eyes do not perfectly cope the movement of the head. This parameter is assumed to be between 0.6 and 1 (Skavenski and Robinson 1973, Robinson 1981). The T_{c1} parameter represents the dynamics associated with the semicircular canals as well as some additional neural processing. The canals are high-pass filters, because after a subject has been put into rotational motion, the neural active membranes in the canals slowly relax back to resting position, so the canals stop sensing motion. Based on mechanical characteristics of the canals, combined with additional neural processing which prolongs this time constant to improve the accuracy of the VOR, the T_{c1} parameter is estimated to be around 15 seconds, and assumed to be between 10 and 30 seconds (Skavenski and Robinson 1973, Robinson 1981). Finally, the T_{c2} parameter captures the dynamics of the oculomotor plant, i.e. the eye and the muscles and tissues attached to it. For the T_{c2} parameter, we will assume that it is between 0.005 and 0.05 seconds.

To find the temporal response for the VOR transfer function, we need to calculate the inverse Laplace transform. The inverse Laplace transform outcome consists of an equivalent differential equation system defined in the same time domain than the spiking cerebellar network (note that the delay is modelled and inserted within the control loop).

$$VOR(s) = \frac{E(s)}{H(s)} = \frac{K \cdot T_{c1} \cdot s}{(T_{c1} \cdot s + 1) \cdot (T_{c2} \cdot s + 1)} = \frac{\frac{K \cdot T_{c1}}{T_{c1} \cdot T_{c2}} \cdot s}{s^2 + s \cdot \frac{(T_{c1} \cdot T_{c2})}{T_{c1} \cdot T_{c2}} + \frac{1}{T_{c1} \cdot T_{c2}}} \cdot \frac{Z(s)}{Z(s)} = \frac{(b_1 \cdot s + b_0) \cdot Z(s)}{(s^2 + a_1 \cdot s + a_0) \cdot Z(s)} \tag{32}$$

Where:

$$a_0 = \frac{1}{T_{c1} \cdot T_{c2}}; \; a_1 = \frac{(T_{c1} \cdot T_{c2})}{T_{c1} \cdot T_{c2}}; \; b_0 = 0; \; b_1 = \frac{K \cdot T_{c1}}{T_{c1} \cdot T_{c2}} \tag{33}$$

Thus we obtain:

$$E(s) = (b_1 \cdot s + b_0) \cdot Z(s) = e(t) = b_1 \cdot \frac{dz}{dt} + b_0 \cdot z(t) \tag{34}$$

$$H(s) = (s^2 + a_1 \cdot s + a_0) \cdot Z(s) = h(t) = \frac{d^2z}{dt} + a_1 \cdot \frac{dz}{dt} + a_0 \cdot z(t) \tag{35}$$

Where the state variables are:

$$y = e(t); \; x_1 = z(t); \; x_2 = \dot{x}_1 = \frac{dx_1}{dt} = \frac{dz}{dt} \tag{36}$$

Substituting:

$$\frac{d^2z}{dt} = -a_1 \cdot \frac{dz}{dt} - a_0 \cdot z(t) + h(t); \; \dot{x}_2 = -a_1 \cdot x_2 - a_0 \cdot x_1 + h(t) \tag{37}$$

It is a differential equation system given by:

$$\dot{x}_1 = x_2 \tag{38}$$

$$\dot{x}_2 = -a_1 \cdot x_2 - a_0 \cdot x_1 + h(t) \tag{39}$$

$$y = b_0 \cdot x_1 + b_1 \cdot x_2 \tag{40}$$

And the canonical form of the linear difference equation system is:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ h(t) \end{bmatrix} \tag{41}$$

$$y = \begin{bmatrix} b_0 & b_1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{42}$$

4.2.6 ICub robot

The iCub robot, used as front-end body, consists of 53 motors able to operate head, arms, hands, waist and legs. The ICub robot can see, hear and move. It can also sense it body position (proprioception) and sense its movement (using accelerometers and gyroscopes) (Tsagarakis et al. 2007). The VOR protocol only requires the head and eye movement. The neck consists of a 3-d.o.f. serial chain of rotations, with the 3 d.o.f. placed in a configuration that best represents human movements. The eye mechanism has also a total of 3 d.o.f. Both eyes can pan (independently) and tilt (simultaneously). The pan movement is driven by a belt system, with the iCub motor behind the eye ball. The eyes (common) tilt movement is actuated by a belt system placed in the middle of the two eyes. Each belt subsystem has a tension adjustment mechanism. The calculation of the actuators characteristics is based on the desired specifications and the moment of inertia, as well as the various components, weight, given by CAD software. For driving this mechanism, DC micro motors (Faulhaber) equipped with planetary gearheads (Gysin) and optical encoders have been used. Each joint uses an overload clutch system that increases the robustness of the mechanism, by absorbing (by sliding) different kinds of impacts and efforts during its interaction with the external world (Beira et al. 2006).

For the robot interface we have used YARP (Yet Another Robot Platform) (Metta, Fitzpatrick, and Natale 2006). Using this tool we can send motor command and read robot sensor information. Finally, we have used the virtual version of iCub as a tool for testing our cerebellar algorithms to quickly check for any major problems prior to use the physical robot.

4.2.7 Control loop

Structural embodiment requires the robot agent, the cerebellar neural network and the environment to be coupled in a way that they can produce effects on one another. The information (i.e. sensory or motor information) must flow in either direction between the environment and the robot agent and its neural network. The control loop allows structural embodiment, thus facilitating the communication flow amongst the body, its neural structure and the environment.

A control scheme and a robot interface (using YARP) able to manage the interaction between neural network simulation and the robot agent sensors and actuators are mandatory. Both elements are independent and interconnected via TCP/IP connections (they can be run in the same or in different computers). The inner/outer control loop architecture allows us to uncouple the robot internal clock from the cerebellar neural internal clock. The control loop establishes a common time framework through with both elements are able to maintain a dialogue (2ms). The cerebellar neural simulation (inner loop) operates in simulation time whereas the robot sensorimotor system (outer loop) operates in RT. The control loop makes compatible these two time domains. This conciliation is vital for several reasons (Ott et al. 2006).

- The outer loop and robot interface program operate in RT (Fig. 5). Conversely, the inner loop may operate faster or slower than RT (simulation time) depending on the cerebellar neural network features and simulation conditions (Fig. 5). Thus, these two control loops operate asynchronously. Particularly, our robotic sensorimotor delay is approximately 15 ms whereas the biological sensorimotor delay has been fixed to 50 ms. The inner loop uses the temporal difference between these two delays to allow neural dynamic pre-processing. The computed neural outcome is made available to the outer loop prior to RT without neural information losses. The control-loop global structure stores synchronous message between loops in buffers. The RT supervisor manages the inner/outer control loop updates and accesses to these buffers to ensure the overall exact time evolution and synchronisation.

- The cerebellar output commands computed within the inner loop (non-linear terms) are meant to achieve high neural dynamic precision at high computation speed with independency of what the outer loop needs. The structure of the inner loop control remains fixed; what control designers may modify more freely to customise the control system architecture is mainly in the outer loop. Thus, the outer loop can be totally modified without restrictions to achieve several other goals without the need to modify the dedicated inner loop control. For instance, additional compensation terms may be included in the outer loop to enhance robustness to parametric uncertainty, unknown dynamics, external disturbances that may occur when controlling a real servomotor, etc. What we propose with this control architecture is a way to isolate the VOR control from the regulation in velocity and position of the motors that command the robot eye and head movements.

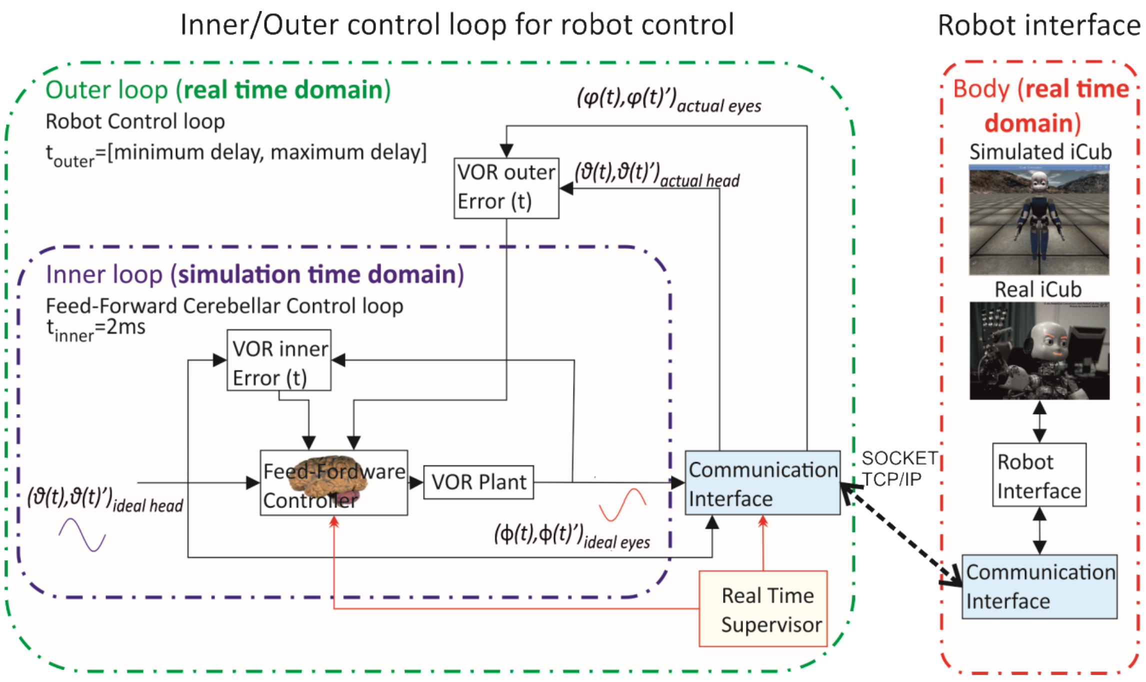


Figure 5 ***Cerebellar inner/outer VOR robot control loop.***

The inner loop corresponds to the forward architecture that supplies the cerebellar corrections in VOR protocols (horizontal and vertical VOR). Reference signals are always velocities in angular coordinates. The cerebellar controller operates two pairs of muscles in agonist/antagonist configuration to cope the eye velocity movements (controlled variable) with the head velocity (reference variable in counter-phase). Assuming an ideal robot where neither motor dynamics nor environmental interaction is involved, inner loop operation may operate the robot in open loop. The outer control loop corresponds to the feedback architecture that supplies the position/velocity corrections to the motors. Since this control loop architecture allows us to uncouple the inner from the outer loop, the use of two different time domains (simulate and real) is possible.

Drawing an analogy between the inner/outer control loop (Fig. 5) and the presented composed control architecture, the inner loop corresponds to the forward architecture that supplies the cerebellar corrections and the outer control loop corresponds to the feedback architecture that supplies the position/velocity corrections to the motors.

4.2.8 RT supervisor and EDLUT upgrading towards RT

A RT supervisor, implemented within EDLUT, operates as a watchdog. In our experiment, RT constitutes a boundary that the neural dynamic computation must not surpass. RT supervisor compares the simulation time of the inner loop with RT. The RT supervisor is constantly calculating a variable named "restriction level". This variable can take five different values. Each different value triggers different countermeasures affecting the neural dynamics computation within the inner loop (Table 5). The higher the value of the variable, the more drastic the countermeasures upon the RT pursuit.

Table 5. ***Real time restriction levels.***

Restriction level	*Contingency tasks*
-1	*The simulation time is way ahead RT. The neural dynamic computation time has to be halted for a short period of time.*
0	*Standard simulation. Neural dynamic computation is ahead RT. No countermeasures are needed.*
1	*The neural dynamic computation time is close to RT. learning rules are disengaged to speed-up the neural dynamic computation.*
2	*The neural dynamic computation time is close to RT even with learning rules already disabled. Spikes propagation and neuron model updates are then disengaged too to further speed-up the neural dynamic computation.*
3	*The neural dynamic computation time is close to RT even with learning rules, spikes propagation and neuron model updates disengaged. All the non-vital neural dynamic computation is disengaged (i.e. internal spike generation, periodic weight saving operation, etc.).*

The experimental VOR process normally maintains its restriction level values between -1 and 0. Only when a large computational load is to be processed (i.e. an occasional large neural activity workload), restriction level values may take values from 1 to 3. EDLUT disengages some neural dynamic computation elements thus causing slightly modifications over the final cerebellar outcome. A restriction level value between 1 and 3 constantly means that EDLUT does not meet the neural dynamic computation requirements. The outcome, therefore, would drastically differ from what it was expected.

Six computational threads have been used for this experiment; 2 for the inner/outer loop, 2 for both communication interfaces, 1 for the RT supervisor and 1 for the robot interface using YARP (see Fig. 5).

Upgrading neural model. The neural dynamic integration.

Purkinje, DCN and IO cells are not worth to be implemented as time-driven neural models in GPU due to their low numbers. On the other hand, the high neural activity to be processed discourages the implementation of event-driven neural models. Time-driven neural models are, therefore, used in CPU.

The neural dynamic integration generates a high computational workload in time-driven neural model. To diminish the overall computational workload, we have pre-computed in look-up tables some costly mathematical functions:

a) The exponential functions used to compute the evolution of the conductances related with the chemical and electrical synapses (Eqs. 6, 7, 8, 13, 14, 15, 19 and 20).
b) G_{NMDA_INF} in DCN model (Eq. 9).
c) $x_0[V]$ and $\tau_x[V]$ functions in PC model (Table 3).

Upgrading the generation and propagation of spikes

The generation of internal spikes and propagated spikes events, their insertion and extraction from the event queues and their final processing constitute a time-consuming computational load in EDLUT (Ros et al. 2006). Two new complex events able to unify in a single event several internal spikes or propagated spike events helped to minimise the timing in generating and propagating spikes.

Upgrading plasticity mechanisms

The plasticity mechanisms also constitute a time-consuming computational load (approximately the half of the simulation time). EDLUT incorporates two specific improvements that reduce the impact of the computation of the plasticity mechanisms over the simulation time.

- ***Pre-computation of kernel functions in look-up tables:*** The LTD kernels (eqs. 25 and 28) allow accumulative computation in an event-driven simulation scheme (Luque et al. 2016). These kernels are now pre-computed in look-up tables. The accumulation of the kernel effect of each spike is now stored in a buffer instead of using state variables. This buffer accumulates the arrival time of the last spikes through each PF and MF (delta Dirac function corresponding to an afferent spike in eqs. 23 and 26). Then, the LTD functions (eqs. 23 and 26) use the precomputed kernels and the time of the last spikes to compute the synaptic weight modification.
- ***Minimization of cache failures when reading synapses in RAM:*** EDLUT allocates the neural network (neurons and synapses) in RAM memory. Correlative neurons and synapses defined in the network file are loaded in correlative RAM memory positions (initial order in Fig. 6). This correlative order in RAM does not ensure the minimisation of cache failures when the synapses are to be read and/or modified during the neural network simulation. However, this correlative order can be reorganised for that purpose.

  There are mainly two ways to reorganise the synapses: following an OUTPUT or INPUT configuration (output and input order in Fig. 6):

  1. OUTPUT configuration minimises the number of cache failures when the spikes are to be propagated and EDLUT needs access to the output synapses of each neuron.
  2. INPUT configuration minimises the number of cache failures when a neuron generates a post-synaptic spike in traditional STDP rules or receives a teaching signal in driven STDP rules and EDLUT needs to modify the synaptic weights of all the “input” synapses to that neuron with STDP rules.

  OUTPUT and INPUT configurations are, in principle, mutually exclusive. Nevertheless, we have designed a mechanism that minimises the number of “INPUT” cache failures when the synapses are reordered following an “OUTPUT” configuration. This is a two-step mechanism; firstly, EDLUT computes the synaptic weight modification for all the input synapses, but it stores these values in an auxiliary array linked to the target neuron instead of directly modify the synaptic weights. So far, EDLUT does not read the input synapses and does not cause cache failures. Secondly, before spike propagation through the affected synapses by STDP learning, EDLUT reads the auxiliary array at the target neurons and modifies the synaptic weights. EDLUT, therefore, only reads "OUTPUT" synapses.

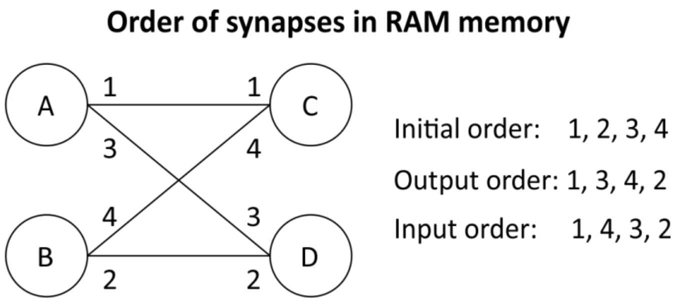


Figure 6 ***Synaptic order in RAM memory.***
This figure represents three different ways to order the synapses in RAM memory.

4.2.9 Results

The first step in the roadmap to embed the cerebellar network on a real body has been to use the simulated iCub robot as the front-end. A Rotational VOR experimentation in the horizontal plane with the iCub robot has been used as test bed. The cerebellum learns to compensate the eye movements (cerebellar adaptation) to stabilise the images captured by the cameras in the eyes while iCub robot described a rotational head movement.

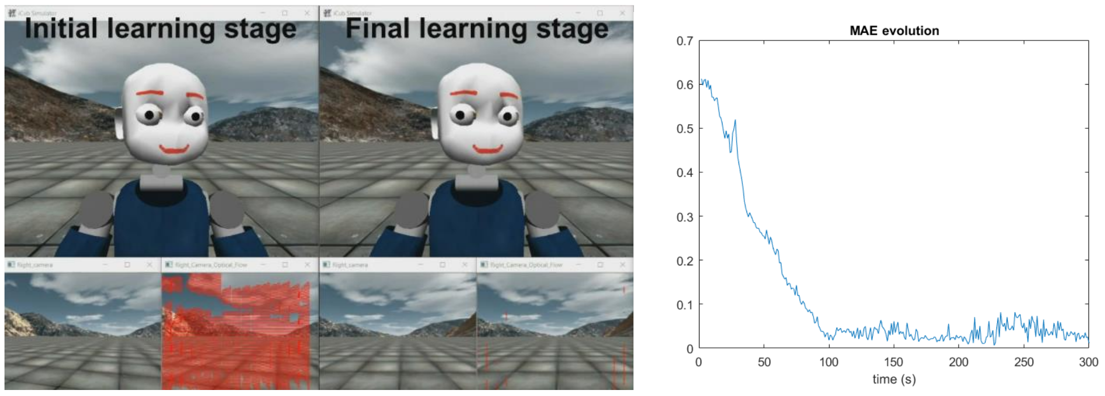


Figure 7 ***The iCub simulator performing the VOR test.***
(Left) *Image showing the VOR performance before (initial learning stage) and after (final learning stage) learning. Down in the figure, for each learning state there are two windows: the visual scene given by the ocular cameras (left) and its optical flow (right), which gives a better description of the accuracy of the reflex itself. The optical flow represents the pattern of apparent motion of objects, surfaces, and edges in the visual scene given by the ocular cameras. This is caused by the relative motion between an observer (a camera) and the scene. For a perfect VOR, the optical flow should be zero. The gradient of a pixel movement is given by the length of coloured arrows. The greater the length of the arrows are, the greater the optical flow and the greater the movement.* ***(Right)*** *Mean Absolute Error (MAE) curve obtained during VOR adaptation. VOR is learnt in about 100 seconds with an overall error about 10% of the initial value.*

As a metric to measure how well the cerebellum is recreating the VOR, we have used two different options. The first one computes the optical flow of the visual scene captured by the cameras in the eyes. For a perfect VOR, the optical flow must be zero. The second one computes the mean absolute error (MAE) over the sum of the head and eyes trajectories for each one-second trial. For a perfect VOR, the amplitude of both signals must be the same with opposed signs, being zero the MAE. As indicated in figure 7, the VOR is learnt roughly in about 100 seconds with high precision (an overall error about 10% of the initial value).

Once all the system has been tested with the simulated iCub, we move on to the real robot. The main difference between the simulated and the real iCub is the artificial propagation delay (considerably higher in the real robot but lower than the biological sensorimotor delay). The results obtained with the real robot (Fig. 8) are consistent with the results obtained with the iCub simulator (Fig. 7).

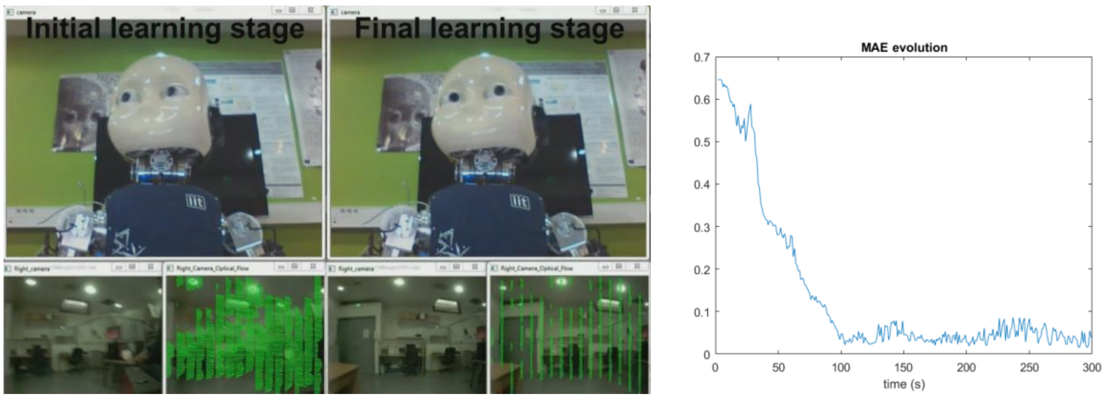


Figure 8 ***The iCub robot performing the VOR test.***
(Left) *Image showing the VOR performance before (initial learning stage) and after (final learning stage) learning. Down in the figure, for each learning state there are two windows: the visual scene given by the ocular cameras (left) and its optical flow (right), which gives a better description of the accuracy of the reflex itself. The optical flow represents the pattern of apparent motion of objects, surfaces, and edges in the visual scene given by the ocular cameras. This is caused by the relative motion between an observer (a camera) and the scene. For a perfect VOR, the optical flow should be zero. The gradient of a pixel movement is given by the length of coloured arrows. The greater the length of the arrows are, the greater the optical flow and the greater the movement.* ***(Right)*** *Mean Absolute Error (MAE) curve obtained during VOR adaptation. VOR is learnt in about 100 seconds with an overall error about 10% of the initial value.*

4.2.10 Results into perspective

A brief summary reporting robotics VOR experiments may provide a measure of significance and impact of our preliminary results. Two main families can be found combining cerebellar controllers and robotics solving the embodied cognition approach: machine learning and cerebellar-based families.

The machine learning family solves the embodied cognition approach without devoting attention to the biological restrictions imposed by the neural structures within the nervous system; the end (the performance) justifies the means. The algorithms mediating the cerebellar role operation are claimed to be inspired in the cerebellar architecture, functionality or both. However, the parallelisms to be drawn between the cerebellar operation/architecture and the algorithms proposed are generally constrained to a general overview of the cerebellar adaptive mechanisms. The solutions provided are usually purely speculative and difficult to refute/validate from a cellular/neural network point of view. Some examples are here summarised:

- A learning system derived from the biologically inspired principle of feedback-error learning (FEL) (Kawato 1990) combined with non-parametric statistical learning networks is presented in (Shibata and Schaal 2001). FEL approximately maps the sensory error into motor error. The motor error, subsequently, is used to train a neural network through supervised learning. While the biology used the cerebellum for this neural network implementing supervised learning, the authors of this work propose a recursive least squares algorithm (RLS) based on a Newton-like method. RLS presents a very fast convergence and robustness without the need for costly parameter adjustments. This system is able to acquire a high performance visual stabilization reflex in a humanoid robot but the biological plausibility is lacking.
- Marr-Albus theory commonly assumes the teaching signal (from CFs) as the motor error. This assumption demands complex neural structures that are able to estimate non-observable motor errors from their observables sensory consequences. A recurrent control architecture with a controller that decorrelates the sensory error from the motor error is presented in (Porrill, Dean, and Stone 2004). This model considers the cerebellum like a bank of adaptive linear filters supervised by the CF activity (Fujita 1982). This model was used to simulate a VOR.
- The locally weight projection regression (LWPR) (Vijayakumar and Schaal 2000) is a nonlinear function approximator that operates in high dimensional spaces. This algorithm is able to cope with redundant dimensions and irrelevant inputs. A cerebellar model in which the granular and molecular layers (including also the interneurons (Wulff et al. 2009)) have been modelled using this LWPR algorithm is presented in (Tolu et al. 2012, Tolu et al. 2013). The input to the PCs is the output of the LWPR algorithm. This cerebellar model have been used to create a gaze stabilization system in (Vannucci et al. 2016). This system integrates eye stabilization (opto-kinetic reflex (OKR) and vestibule-ocular reflex (VOR)) together with head stabilization (vestibulocollic reflex (VCR)). They have tested this system using a simulated iCub robot.

On the contrary, cerebellar-based families solve the embodied cognition approach by taking the biological restriction imposed by the cerebellar neural structures as granted. The cerebellar algorithm performance is a consequence, not the main cause. The cerebellar algorithms are biologically constrained and they share a family resemblance with the cerebellar anatomy. The solution provided, although speculative, give us a closer and clearer view of the cerebellar computation primaries. The main aim here is to drive basic cerebellar research by proposing working cerebellar hypotheses that can be either refuted or validated from a cellular or neural network point of view. This family can be subdivided in two main categories: analogue cerebellar models and spiking cerebellar models.

- *Analogue cerebellar models.* They usually present higher abstraction levels than spiking models. For that reason, they are usually easier to implement and more computationally efficient, but less biological plausible. An analogue cerebellar model in which the role of distributed plasticity over PC and DCN layer is presented in (Garrido Alcazar et al. 2013). This cerebellar model have been used in (Casellato et al. 2015) to recreate an EBCC and VOR experiment using 2DOF robotic arm in RT (1st DOF emulating the neck, 2nd DOF emulating the eyes). In this case the RT requirements are easy to cope with due to the simplicity and efficiency of the analogue cerebellar model.

- *Spiking cerebellar models.* They are more akin to biology. They try to mimic the cerebellar neural communication by using spikes. These spikes are propagated within cerebellar sub-circuits that attempt to mimic the cerebellar architecture. Interestingly, the emerging behaviour from the dialogue between the neural code and the different cerebellar sub-circuits is intended to cope with the behaviours observed in biology. Conciliating realistic spiking cerebellar models with behavioural outcomes (i.e. VOR) remains an open issue. There exists computational models that partly address this problem (i.e. modelling and interconnecting certain sub-circuits (Solinas, Nieus, and D'Angelo 2010) or certain spiking features (Latorre et al. 2013)). ***Nevertheless, reconstructing the path from cellular to behaviour level remains elusive. Our spiking cerebellar model proposes an attempt to construct this path. The model itself is a biological and computational upgrading of (Luque et al. 2016) (i.e. complex spikes in PCs, electrical coupling at IO, etc.). To the best of our knowledge, this is one amongst the first initiatives that are able to combine this level of neural detail with several neural adaptive mechanisms working all together to operate a humanoid in a VOR experiment in RTs.***

Capítulo 5: Introducción

Uno de los retos principales a los que la humanidad se enfrenta en el siglo XXI es entender los principios biológicos de la consciencia y los procesos mentales a través de los que percibimos, actuamos, aprendemos y recordamos (Kandel et al. 2000). El conocimiento de estos principios en el sistema nervioso puede reportarnos numerosos beneficios. En primer lugar, un mayor entendimiento del cerebro nos ayudará a diseñar tratamientos mejores y más eficientes contra enfermedades neurodegenerativas tales como Huntington, Parkinson y todo tipo de demencias. En segundo lugar, el cerebro puede ser considerado como uno de los mejores sistemas de procesamiento de información: una enorme capacidad de almacenamiento y procesamiento de información junto con bajos niveles de consumo energético y fiabilidad contra fallos por envejecimiento. Un mayor conocimiento de cómo los sistemas biológicos procesan la información nos permitirá desarrollar nuevas generaciones de arquitecturas de procesamiento capaces de replicar estas extraordinarias capacidades.

El cerebelo ("pequeño cerebelo" del Latín) es una región del cerebro muy importante para los vertebrados tales como los humanos. Se sabe que el cerebelo juega un papel fundamental en distintas características del control moto tales como la coordinación y precisión de movimientos (Thach, Goodkin, and Keating 1992), aunque parece que también está envuelto en otras funciones cognitivas como puedan ser la atención y el lenguaje o en la regulación de las respuestas frente al miedo y el placer (Wolf, Rapoport, and Schweizer 2009). Se ha observado que daños en el cerebelo producen desordenes en la precisión de movimientos, equilibrio, postura y aprendizaje motor (Fine, Ionita, and Lohr 2002).

Entender los mecanismos biológicos que le confieren al cerebelo sus capacidades requiere de su estudio usando diferentes metodologías, estando todas ellas interconectadas. Esta tesis aborda el estudio del cerebelo desde el punto de vista de la neurociencia computacional. Esta es una ciencia interdisciplinar que uno diversos campos de estudio tales como la neurociencia, ciencia cognitiva y psicología con la ingeniería electrónica, ciencias de la computación, matemáticas y física. Debido al rápido incremento de recursos computacionales disponibles en las últimas décadas, la neurociencia computacional ha emergido como una poderosa herramienta capaz de testear y validad las hipótesis propuestas por los neurocientíficos sobre el cerebro. Los neurocientíficos computacionales usan modelos matemáticos y simulaciones por ordenador para estudiar el sistema nervioso a distintos niveles. Dependiendo del nivel de detalle y el tamaño de la estructura simulada (desde niveles moleculares en una sola neurona hasta grandes e intrincadas redes neuronales con plasticidad distribuida), los requerimientos computacionales pueden diferir drásticamente.

Modelos computacionales de varias regiones del cerebro han sido desarrollados y estudiados por más de treinta años con el objetivo de analizar la función cerebral. La ciencia computacional es un complemento natural a la investigación experimental en el cerebro, ya que se centra en mecanismos específicos y modelos que únicamente pueden ser parcialmente observados en estudios fisiológicos. En concreto, el lazo de control cerebelar (el foco de esta tesis) ha sido extensamente modelado desde que Marr (Marr 1969) y Albus (Albus 1971) propusieron una explicación elegante de como el cerebelo opera como un controlador anticipativo en los mamíferos.

La simulación de un modelo de sistema nervioso conectado a un cuerpo puede ayudar a entender mejor como ciertas capacidades del sistema nervioso emergen en base a distintas características celulares, topologías de red o mecanismos de adaptación sináptica a nivel local. En esta tesis nos hemos centrado en entender el papel que desempeña el cerebelo en la coordinación de movimientos (voluntarios o reflejos) y en la manipulación de objetos. Por esta razón necesitamos un "cuerpo" capaz de realizar estas tareas motoras cuando es controlado por nuestros modelos de cerebelo. En esta tesis hemos usado tanto robots virtuales (simulados) como reales para emular este cuerpo. Sin embargo, varias consideraciones han de tenerse en cuenta. Mientras que el cuerpo humano codifica la información sensoriomotora usando codificación en población neuronal (cada neurona que codifica la señal presenta una distribución de respuesta sobre el conjunto de entradas), los robots usan codificadores que devuelven la posición y velocidad relativas (señales analógicas) de cada articulación. Una traducción o conversión desde el dominio analógico de los sensores del robot hasta un patrón basado en impulsos compatible con las redes neuronales de impulsos es requerido. Además, la respuesta de salida del cerebelo basada en impulsos debe ser también convertida al dominio analógico empleado por los actuadores (motores) del robot. Finalmente, los robots reales imponen una condición o restricción de tiempo real (RT). Cuando llevamos a cabo una simulación conectada a un robot real, el tiempo de simulación debe evolucionar a la misma velocidad que el RT (tiempo físico), de lo contrario el robot no puede ser correctamente controlado. La dinámica del robot (cantidad de movimiento de los distintos elementos) y la dinámica del ambiente hacen de esta operación en RT una ardua restricción.

En esta tesis hemos embebido varios modelos cerebelares en diferentes esquemas de lazo cerrado capaces de controlar el movimiento de robot biomórficos en RT. Esta integración persigue los siguientes objetivos: (i) validación de las hipótesis presentadas en cada uno de los modelos cerebelares usando diferentes tareas de control motor como estándar de comparación, y (ii) desarrollo de innovadores esquemas de control para la próxima generación de robot biomórficos.

Para este estudio hemos desarrollado y actualizado nuestro propio simulador de redes neuronales de impulsos: EDLUT (Ros et al. 2006, Garrido et al. 2011, Luque et al. 2014, Naveros et al. 2015, Naveros et al. 2017). Esta herramienta está orientada a la simulación eficiente de redes neuronales de media escala (decenas de miles de neuronas) usando modelos puntuales simplificados de neuronas (Leaky Integrate-and-Fire (LIF), Adaptive exponential integrate-and-fire (AdEx), Izhikevich and Hodgkin-Huxley (HH)) y mecanismos de plasticidad sináptica en RT. En nuestro caso hemos usado EDLUT para crear modelos cerebelares embebidos en esquemas de lazo cerrado capaces de controlar el movimiento de robots biomórficos en RT. La limitación o restricción de RT, la cual demanda ejecutar la simulación de la red neuronal a una determinada velocidad, es fundamental cuando trabajamos con robots reales en los cuales el tiempo es una variable física que no podemos controlar. En esta tesis hemos mejorado drásticamente el rendimiento de EDLUT usando diferentes técnicas de paralelización y simulación, permitiendo así la simulación de redes neuronales más grandes y complejas en RT.

También se han desarrollado un esquema de control en lazo cerrado y un supervisor de RT para tratar con la constricción de RT.

5.1 El cerebelo

El cerebelo tiene la apariencia de una estructura separada situada al fondo del cerebro, debajo de los hemisferios cerebrales. Esta estructura ha sido ampliamente estudiada durante más de un siglos desde que Camilo Golgi y Santiago Ramón y Cajal estudiaron la organización anatómica del córtex cerebelar (Golgi 1906, Cajal 1894). Aunque el cerebelo solo cuenta con aproximadamente el 10% del volumen del cerebro, este contiene aproximadamente el 50% del número total de neuronas del mismo (Llinas, Walton, and Lang 2004). El cerebelo se subdivide en dos partes principales: una sección interna formada por los núcleos cerebelosos profundos (DCN) cubierta por una capa de tejido altamente plegada llamada córtex cerebeloso. Mientras que los DCN representan la estructura de salida del cerebelo, el córtex cerebeloso representa la entrada principal al cerebelo, conteniendo la mayoría de las neuronas del mismo (Purves et al. 2008). Dentro del córtex cerebeloso hay varios tipos de neuronas con una distribución altamente regular, siendo las células de Purkinje (PCs) y las células granulares (GCs) las más características. Esta compleja organización neural da lugar a una masiva capacidad de procesamiento de señales (Eccles, Ito, and Szentágothai 1967).

A la luz de algunos estudios, se ha sugerido que el sistema nervioso central planifica y ejecuta secuencialmente los movimientos voluntarios. De esta forma el cerebelo no iniciaría los movimientos voluntarios, aunque sí que contribuiría a su coordinación y precisión usando bucles de retroalimentación para el movimiento muscular (Ito 1970, 2006). De acuerdo con esta hipótesis, el cerebro podría en primer lugar planear una trayectoria óptima en la espacio de coordenadas físico (donde se realiza la tarea), traducirla al espacio de coordenadas intrínseco del cuerpo, y finalmente generar los comandos motores necesarios para llevar a cabo dicha tarea (Houk, Buckingham, and Barto 1996, Nakano et al. 1999, Todorov 2004, Hwang and Shadmehr 2005, Izawa et al. 2012, Passot, Luque, and Arleo 2013).

Por un lado se cree que el sistema formado por el núcleo rojo parvocelular-neocerebelo proporciona un modelo neural crudo interno de la dinámica inversa del sistema musculo-esqueletal, el cual es adquirido mientras se monitorean las trayectorias deseadas (Kawato, Furukawa, and Suzuki 1987). Por otro lado se cree que el sistema formado por el núcleo rojo magnocelular-espinocerebelo proporciona un preciso modelo neural interno de la dinámica del sistema musculo-esqueletal, el cual es adquirido mientras se monitorean los movimiento voluntarios (Kawato, Furukawa, and Suzuki 1987). De acuerdo a esta teoría, el córtex asociativo estaría al cargo de proporcionar la trayectoria deseada en el espacio de coordenadas intrínseco del cuerpo y propagarlo hasta el córtex motor. Este generaría entonces los comandos motores óptimos para operar nuestros miembros usando el modelo dinámico inverso, enviándolos a las neuronas motoras inferiores en el tronco encefálico y la médula espinal (Siciliano and Khatib 2016). El cerebelo también recibiría las trayectorias deseadas junto con la información sensorial preveniente de la médula espinal y otras partes del cerebro (Siegel and Sapru 2006). Usando esta información y el modelo dinámico musculo-esqueletal, el cerebelo sería capaz de generar correcciones en los comandos motores de acuerdo a errores predecibles que se producen cuando se realiza un movimiento. De esta forma, el modelo crudo dinámico inverso trabaja conjuntamente con el modelo dinámico provisto por el cerebelo embebido en lazo de control anticipativo (Fig. 9).

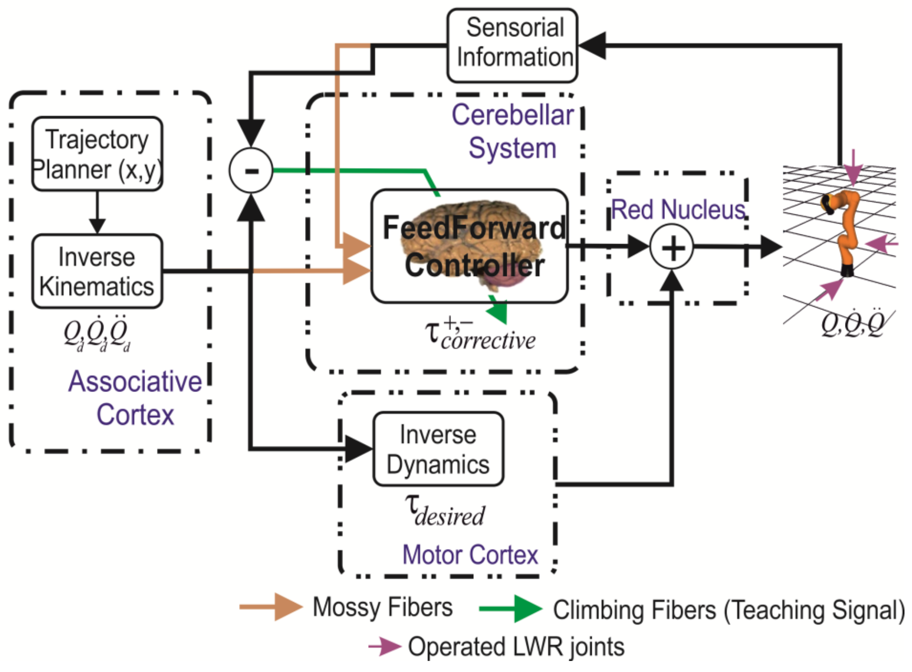


Figure 9 ***Ciclo de control cerebelar anticipativo (adaptado de (Luque et al. 2016)).*** *El módulo cerebelar adaptativo embebido en un ciclo de control anticipativo genera valores correctivos de torque ($\tau_{corrective}$) para compensar la desviación producida por el modelo dinámico crudo inverso cuando se manipulan objetos con un peso significativo.*

Los mecanismos de aprendizaje y adaptación del cerebelo juegan un papel crucial en el control motor. Varios modelos teóricos desarrollados en las últimas décadas tratan de explicar cómo el cerebelo es capaz de calibrar la relación entre las entradas sensoriales y los comandos motores usando mecanismos de plasticidad sináptica. La mayoría de estos modelos son extensiones de las teorías propuestas por David Marr (Marr 1969) y James Albus (Albus 1971) basadas en la observación que cada PC recibe dos tipos de sinapsis de entrada drásticamente distintas: (i) miles de entradas débiles provenientes de las GC a través de las fibras paralelas (PF) y (ii) una única y extremadamente fuerte entrada proveniente de una célula de la Oliva Inferior (IO) a través de una fibra trepadora (CF). La fuerza de estas CF es tan intensa (debido a los numerosos contactos sinápticos existentes entre cada CF y el árbol dendrítico de cada PC) que un solo potencial de acción de una célula de la IO genera un impulso complejo en la PC de destino (un impulso inicial de gran amplitud seguido de una ráfaga de alta frecuencia y un periodo de no actividad (Schmolesky et al. 2002)).

El concepto básico de la teoría Marr-Albus recae en el hecho de que las PFs (i.e. los axones de las GC) propagan la información sensoriomotora a las PCs mientras que las CFs (i.e. los axones de las células de la IO) propagan señales de aprendizaje que codifican el error de movimiento. Estas señales de aprendizaje inducen cambios de larga duración en la fuerza de las PFs. Observaciones de depresiones a largo plazo (LTD) in las PFs de entrada dan soporte a teorías de este tipo, pero su validez sigue siendo controvertida (Purves et al. 2008). En esta tesis hemos extendido este modelo incluyendo dos nuevos mecanismos de plasticidad a nivel de los DCN. Estos nuevos mecanismos de plasticidad ayudan a consolidar el aprendizaje en las PCs a nivel de los DCN y optimiza el rango de operación de los DCN. La figura 10 representa un esquema del modelo de cerebelo incluyendo los tres mecanismos de plasticidad propuestos.

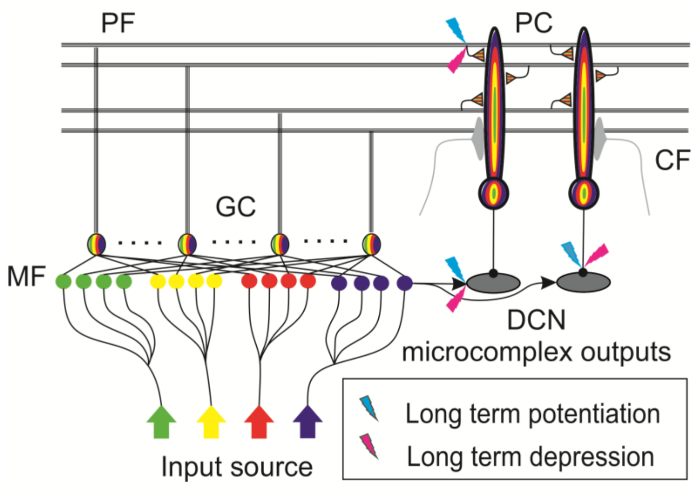


Figure 10 ***Arquitectura cerebelar (adaptado de (Luque et al. 2016)).***
Nuestro modelo cerebelar se compone de fibras musgosas (FM), células granulares (GC), fibras paralelas (PF), células de Purkinje (PC), fibras trepadoras (CF) y células de los núcleos cerebelosos profundos (DCN). La plasticidad sináptica a largo plazo para las aferentes de las PCs y los DCN se indican con dos símbolos coloreados; potenciación a largo plazo (LTP) en azul y depresión a largo plazo (LTD) en magenta.

El condicionamiento clásico del parpadeo (EBCC) (Thompson 1990) y el reflejo vestíbulo-ocular (VOR) (Leigh and Zee 2015) son ampliamente reconocidos como los paradigmas que mejor revelan las características del aprendizaje cerebelar. El EBCC es un procedimiento relativamente sencillo que consiste en emparejar un estímulo auditorio o visual (estímulo condicionado (CE)) con un estímulo incondicionado (UE) (e.g. un leve soplo de aire en la córnea) que provoca un parpadeo en el ojo. Por el contrario, el VOR es un procedimiento un poco más complicado en el cual la activación del sistema vestibular causa movimientos en el ojo. Este reflejo estabiliza las imágenes en la retina durante el movimiento de la cabeza produciendo el movimiento de los ojos en la dirección opuesta al movimiento de la cabeza, preservando así la imagen en el centro del campo visual. Ambos experimentos han sido recreados usando nuestros modelos cerebelares.

En esta tesis nos hemos centrado en entender los mecanismos usados por el cerebelo en dos funciones principales relacionadas con el control motor:

- Coordinación de movimientos voluntarios: La realización de la mayoría de los movimientos requieren de la activación de diferentes grupos musculares coordinados en el tiempo. Una de las funciones principales del cerebelo es coordinar el tiempo y la fuerza que deben aplicar cada grupo muscular para producir movimientos fluidos del cuerpo.
- Aprendizaje motor: El cerebelo juega un papel fundamental en la adaptación y ajuste fino de los comandos motores para realizar movimientos precisos a través del proceso de ensayo y error. El cerebelo debe estar constantemente ajustando la relación sensoriomotora para compensar cambios en el cuerpo (e.g. un jugador de tenis que usa una nueva raqueta con un peso o dimensiones diferentes) o en el entorno (e.g. jugar en una pista de césped o en una pista rápida).

5.2 Neurociencia computacional: herramientas de simulación

En los últimos años, las herramientas de simulación de redes neuronales de impulsos han experimentado una rápida evolución desde pequeños simuladores hechos a medida para experimentos específicos hasta complejos simuladores de propósito general capaces de llevar a cabo un amplio rango de experimentos. Actualmente existe un extenso número de herramientas de simulación especializadas en distintos ámbitos de simulación. (i) NEURON (Hines and Carnevale 1997) y GENESIS (Bower and Beeman 1998) son la referencia cuando se quiere simular modelos biofísicos detallados de neuronas. (ii) NEST (Gewaltig and Diesmann 2007) es la referencia cuando se quiere simular redes neuronales inmensas (miles de millones de neuronas y billones de sinapsis) en superordenadores usando modelos puntuales simplificados de neurona (donde la morfología no es relevante). (iii) Spikey (Pfeil et al. 2012, Schemmel et al. 2010) es un simulador neuromórfico basado en circuitos analógicos (condensadores y resistencias) sobre una oblea de silicio. Este realiza simulaciones entre 1000 y 10,000 veces más rápido que RT. Esta increíble tasa de aceleración se puede usar para llevar acabo análisis intensivos de parámetros en redes neuronales en cortos periodos de tiempo. (iv) SpiNNaker (Khan et al. 2008) usa una arquitectura de cómputo masivamente paralela hecha a medida basada en procesadores ARM distribuidos en una topología toroidal. Se planea que la máquina al completo contenga un millón de procesadores ARM y sea capaz de llevar a cabo simulaciones con hasta mil millones de neuronas simples. Para una revisión más profunda sobre este tema (Brette et al. 2007).

El trabajo desarrollado en esta tesis sigue las líneas de investigación que nuestro grupo en la Universidad de Granada ha desarrollado durante los últimos años. Su objetivo principal es desarrollar modelos cerebelares biológicamente inspirados embebidos en esquemas de control en lazo cerrado capaces de llevar a cabo diferentes tareas de control motor con robots reales en RT. Debido a la escasez de herramientas capaces de cumplir con los requisitos necesarios para estos experimentos con robots, nuestro grupo decidió desarrollar nuestra propia herramienta de simulación: EDLUT.

5.2.1 Estrategias de simulación: dirigidas por tiempo vs dirigidas por eventos

Cada simulador de redes neuronales de impulsos debe realizar al menos tres tareas diferentes: (i) computar la evolución dinámica de cada neurona (normalmente definida por un conjunto de ecuaciones diferenciales que deben ser integradas en el tiempo), (ii) generación y propagación de los impulsos, y (iii) computación de los mecanismos de plasticidad que modifican los pesos sinápticos. Estas tareas se pueden llevar a cabo usando dos familias distintas de métodos de simulación: métodos de simulación dirigidos por tiempo o dirigidos por eventos. La principal diferencia entre ambos métodos recae en la forma en que cada uno maneja la evolución del tiempo, especialmente para computar la evolución de la dinámica neuronal. Para una revisión más profunda sobre este tema (Brette et al. 2007).

La mayoría de los simuladores usan métodos de simulación dirigidos por tiempo basados en integradores de paso fijo, dado que estos esquemas permiten la simulación de la mayoría de los modelos de neurona, sinapsis y ley de aprendizaje. Estos métodos dividen el tiempo de simulación en pequeños pasos de tiempo de tamaño fijo y evalúan la evolución dinámica de cada neurona integrando sus ecuaciones diferenciales en cada paso (Iserles 2009). El principal inconveniente de estos métodos de integración de paso fijo es su limitada eficiencia cuando la actividad neuronal a procesar es dispersa o cuando la complejidad neuronal es elevada (en términos de número de ecuaciones diferenciales por neurona). En este caso podría ser recomendable usar métodos de integración de paso variable (Iserles 2009). Estos métodos adaptan de forma iterativa el tamaño del paso de integración en función de la evolución dinámica de cada neurona (esta evolución depende a su vez de la actividad neuronal y la complejidad del modelo). Sin embargo, los métodos de integración de paso variable no están exentos de inconvenientes (discutidos todos ellos en el segundo artículo incluido en esta tesis (Naveros et al. 2017)). De esta forma, los métodos de integración de paso variables están normalmente restringidos a simuladores especializados en la simulación de unas pocas neuronas con modelos muy complejos (e.g. NEURON o GENESIS).

En respuesta a este problema surgieron los métodos de simulación dirigidos por eventos como posible solución a la baja eficiencia de los métodos de integración de paso fijo cuando se simulaban redes neuronales con baja actividad. Hay varias regiones del cerebro que se caracterizan por poseer una actividad dispersa (e.g. las GCs en el córtex cerebelar representan la mitad de la neuronas de todo el cerebro, reciben entre tres y seis sinapsis de entrada con una actividad muy dispersa y la mayoría de ellas permanecen en silencio y apenas generan impulsos). La principal fortaleza de estos métodos de simulación dirigidos por eventos recae en el hecho de que la evolución dinámica neuronal solo se computa y actualiza cuando un nuevo evento modifica la evolución normal de una neurona (i.e. cuando un impulso de entrada es recibido o un impulso de salida es generado). Varios métodos de simulación dirigidos por eventos han sido desarrollados e incorporados en diferentes simuladores de redes neuronales (Mattia and Del Giudice 2000, Delorme and Thorpe 2003, Reutimann, Giugliano, and Fusi 2003, Rudolph and Destexhe 2006, Pecevski, Kappel, and Jonke 2014). El principal inconveniente de todos estos métodos es el hecho de que únicamente modelos neuronales relativamente sencillos basados en ecuaciones que pueden ser evaluadas en tiempos arbitrarios pueden ser implementados (e.g. Spike-Response Model). Esta restricción limita el número de modelos de neurona que pueden ser simulados usando estas técnicas (limitación que los métodos dirigidos por tiempo no sufren).

5.2.2 La evolución del simulador EDLUT

Nuestro grupo de investigación en la Universidad de Granada ha estado estudiando el cerebelo durante más de una década. Dentro del cerebelo, las GCs son, de lejos, las neuronas más numerosas (representando la mitad de las neuronas de todo el cerebro). Estas neuronas presentan ratios de conectividad de entrada bajos junto con niveles de actividad dispersos. Tratando de explotar esta característica, nuestro grupo de investigación desarrolló la primera versión de EDLUT usando un esquema de simulación dirigido por eventos basados en tablas de consulta (Ros et al. 2006). Este innovador método computa en una fase inicial la evolución dinámica neuronal de cualquier modelo de neurona con una baja complejidad (hasta cuatro o cinco ecuaciones diferenciales) y la almacena en tablas de consulta que posteriormente usará durante el tiempo de simulación. Modelos neuronales tales como LIF, AdEx, Izhikevich y HH pueden ser precomputados en tablas de consulta. Para una revisión más profunda de este tema recomendamos la lectura de la tesis del Dr. Carrillo (Carrillo 2009).

Sin embargo, existen diferentes tipos de neuronas en el cerebelo con ratios de conectividad y actividad muy diferentes. Tal y como hemos comentado anteriormente, las GCs son el perfecto candidato para una simulación dirigida por eventos debido a su actividad dispersa. Por el contrario, las PCs (grandes neuronas integrando la actividad de hasta 100,000 sinapsis de entrada) encajan mejor con los métodos de simulación dirigidos por tiempo. De esta forma, la simulación eficiente del cerebelo requiere de dos técnicas de simulación diferentes. Por este motivo, el Dr. Jesús Garrido (co-director de esta tesis) actualizó el núcleo de EDLUT hacia un esquema de simulación hibrido dirigido por eventos y por tiempo capaz de usar ambos métodos conjuntamente en la misma red (Garrido et al. 2011). En este caso solo se implementaron métodos de integración de paso fijo. Así EDLUT podía aprovechar los puntos fuertes y mitigar las debilidades de ambos métodos de simulación (métodos dirigidos por eventos basados en tablas de consulta para modelos de neurona relativamente sencillos con ratios de actividad bajos (como las GCs) y métodos de simulación dirigidos por tiempo basados en métodos de integración de paso fijo para modelos de neurona más complejos con tasas de actividad más elevadas (como las PCs). Para una revisión más profunda de este tema recomendamos la lectura de la tesis del Dr. Garrido (Garrido 2012).

5.3 Motivación

Tradicionalmente, la neurociencia ha estudiado los sistemas biológicos usando experimentos *in vitro* e *in vivo*. El término *in vitro* (“entre vidrio” del Latín) se refiere a la técnica que lleva a cabo un procedimiento en un entorno controlado fuera de un organismo vivo. De esta forma el experimento puede ser perfectamente controlado, aunque únicamente se pueden estudiar comportamientos y características relativamente sencillos. Por el contrario, el término *in vivo* (“dentro de lo vivo” del Latín) se refiere a los experimentos usando organismo vivos completos (en oposición a organismo muertos o parte de ellos). Esta técnica permite el estudio de sistemas y comportamientos mucho más complejos, aunque el control sobre las condiciones del experimento y las mediciones que se pueden realizar están notablemente más limitadas. Recientemente, la neurociencia computacional con sus experimentos *in silico* (hechos por computadora o vía simulación computacional) ha emergido como una tercera opción para el estudio del cerebro. En esta tesis hemos usado los resultados obtenidos mediante la observación experimental (principalmente en ratas y ratones) para crear un modelo computacional del cerebelo. Este modelo presenta una doble funcionalidad: (i) validad las hipótesis derivadas de los experimentos *in vivo* e *in vitro*, y (ii) proponer nuevas hipótesis que pueden ser validadas por los experimentos *in vivo* e *in vitro*. De esta forma podemos decir que los experimentos de la neurociencia computacional son el perfecto complemento a los experimentos con animales.

Realizar esto experimentos con un cuerpo artificial en RT es una tarea compleja que requiere de herramientas de simulación especializadas. Los artículos de revista publicados previamente a esta tesis relacionados con el desarrollo de EDLUT (Ros et al. 2006, Garrido et al. 2011) muestran el extraordinario potencial de EDLUT para la simulación eficiente de modelos cerebelares. Sin embargo, las versiones previas de EDLUT a esta tesis carecían de la eficiencia necesaria para realizar experimentos en RT con modelos razonablemente grandes y complejos. De esta forma era necesaria una mejora considerable del rendimiento de EDLUT, el desarrollo de mecanismos de RT y una interfaz para los robots.

Centrándonos en el cerebelo, es bien sabido que esta estructura juega un papel fundamental en la coordinación y aprendizaje de movimientos. Los neurocientíficos realizan experimentos conductuales tales como EBCC, VOR o manipulación de objetos para estudiar el cerebelo, observando como este es capaz de controlar el cuerpo. Nosotros replicamos esta metodología estudiando nuestros modelos cerebelares embebidos en esquemas de lazo cerrado capaces de controlar el movimiento de robots simulados o reales en RT. De esta forma podemos apreciar mejor la influencia que ejercen nuestros modelos cerebelares en el comportamiento del robot. Esta configuración permite explorar diferentes características del modelo (modelos de neuronas, topología de interconexionado, mecanismos de plasticidad sináptica, etc.) y como afectan al comportamiento del robot. Además, estos comportamientos del robot pueden ser comparados con los observados por los neurocientíficos en animales (o humanos), validando o refutando de esta forma las teorías estudiadas.

Desde el punto de vista de los sistemas robóticos, el cerebelo es capaz de coordinar el movimientos de cientos de músculos con una increíble precisión y bajo consumo energético. También es capaz de aprender nuevas tareas o habilidades repitiendo el mismo movimiento y observando las consecuencias. Replicar estas características en los controladores robóticos puede mejorar profundamente la funcionalidad y autonomía de las nuevas generaciones de robots biomórficos.

Por último, desde el punto de vista del procesamiento de información, el cerebro humano representa un sistema de procesamiento realmente potente, energéticamente eficiente y robusto ante fallos por envejecimiento. Entender como los sistemas biológicos son capaces de procesar la información de cientos de fuentes nos ayudará a desarrollar nuevas generaciones de arquitecturas de procesamiento con capacidades nunca antes vistas.

5.4 Objetivos

Esta tesis pretende crear modelos cerebelares que expliquen los mecanismos biológicos que le confieren al cerebelo la capacidad de realizar diferentes tareas de control motor. Con el objetivo de validar los modelos, estos han sido embebidos en diferentes esquemas de control de lazo cerrado y expuestos a exigentes tareas de control motor usando robots simulados o reales en RT. Otro objetivo de esta tesis es implementar nuevos esquemas de control biológicamente inspirados para controlar las nuevas generaciones de robots biomórficos. El último objetivo de esta tesis es desarrollar las herramientas de simulación que permitan llevar a cabo los dos objetivos anteriormente propuestos, explorando nuevos y eficientes métodos de simulación, esquemas de control, interfaces de comunicación y supervisores de RT.

Para conseguir estos objetivos, esta tesis aborda los siguientes puntos:

- Desarrollo de las herramientas de simulación requeridas para crear un sistema de control para robots reales basado en redes neuronales de impulsos. Este objetivo se puede subdividir en:
  - ✓ Desarrollo de un simulador de redes neuronales (EDLUT) ultra rápido para redes de media escala. Implementar diferentes técnicas de paralelización en CPUs y GPUs para incrementar su rendimiento computacional.
  - ✓ Implementar nuevos métodos de simulación más eficiente dirigidos por eventos y por tiempo orientados a modelos de neurona tales como LIF, AdEx, Izhikevich y HH.

    - ✓ Integración de un entorno robótico (capaz de controlar robots simulados y reales) en el simulador de redes neuronales para facilitar la experimentación usando esquemas de control bioinspirados.
    - ✓ Mejorar el simulador añadiéndolo la capacidad de asegurar simulaciones en RT con robots.
- Implementación de un modelo cerebelar usando las herramientas anteriormente desarrolladas. Este objetivo se puede subdividir en:
    - ✓ Estudiar como los diferentes elementos del modelo cerebelar (neuronas y sinapsis) contribuyen a sus habilidades motoras. Esto requiere la implementación y evaluación de complejos modelos de neurona y sinapsis en EDLUT, tomando especial atención a las PCs, DCN y IO.
    - ✓ Estudiar diferentes mecanismos de plasticidad sináptica a nivel de las PCs y DCN que le permiten al modelo cerebelar adquirir nuevas habilidades.
- Validación de los modelos cerebelares propuestos mediante su inclusión en esquemas de control de lazo cerrado para realizar diferentes tareas de control motor con robots simulados o reales en RT.

5.5 Nuestra contribución

Siempre con la restricción de RT en mente, esta tesis a mejorar profundamente la eficiencia y funcionalidad de EDLUT. Con respecto a la eficiente, nuestras contribuciones principales se recogen en los dos primeros artículos incluidos en esta tesis (Naveros et al. 2015, Naveros et al. 2017), pero también en los resultados adicionales pendientes de publicación incluidos en el cuarto capítulo de esta tesis. Hoy en día, la mayoría de las CPUs incluyen varios núcleos físicos o virtuales. Por esta razón hemos paralelizado el núcleo de EDLUT (métodos dirigidos por eventos y por tiempo) en CPU usando OpenMP. Esta es una API que soporta la programación multiplataforma de programas de multiprocesamiento usando memoria compartida en C++. Además, hace unos cuando años NVIDIA desarrollo CUDA, una nueva arquitectura de procesamiento en GPUs y una API. CUDA permite desarrollar código que se puede ejecutar en las GPUs con arquitectura CUDA para realizar tareas de propósito general explotando el paralelismo a nivel de dato. El cómputo de la evolución dinámica neuronal en los métodos de simulación dirigidos por tiempo que usan integración de paso fijo es una tarea que cumple perfectamente con los requisitos de la arquitectura paralela de CUDA. Por esta razón hemos actualizado EDLUT hacia una plataforma de co-procesamiento CPU-GPU en la que la GPU (usando CUDA) computa modelos de neurona dirigidos por tiempo mientras que la CPU (usando OpenMP) computa modelos de neurona dirigidos por eventos y por tiempo. Tanto la generación y propagación de impulsos como los mecanismos de plasticidad se procesan paralelamente en la CPU usando OpenMP. Todas estas nuevas características se cubren en (Naveros et al. 2015).

También hemos desarrollados nuevos y eficientes métodos de simulación dirigidos por eventos y por tiempo capaces de tratar con modelos de neurona más complejos. Para los métodos dirigidos por eventos hemos reestructurado las tablas de consulta y desarrollado un nuevo mecanismo para generar y propagar de forma mucho más eficiente la actividad síncrona. Para los métodos dirigidos por tiempo de la CPU y la GPU hemos desarrollado los métodos de integración de doble paso fijo. Estos son capaces de aprovechar los puntos fuertes y mitigar las debilidades de los métodos de integración de paso fijo y paso variable. Todas estas nuevas características se cubren en (Naveros et al. 2017).

Por último, en la sección de resultados adicionales de esta tesis hemos expuesto como diferentes desarrollos relacionados con la computación de neuronas, sinapsis y mecanismos de plasticidad se han incluido en EDLUT.

En lo referente a la funcionalidad de EDLUT, nuestras principales contribuciones están cubiertas en el resto de artículos que componen esta tesis (Luque et al. 2014, Luque et al. 2016) y en la sección de resultados adicionales. Hemos implementados en EDLUT nuevos modelos de neurona y leyes de aprendizaje necesarios para nuestros modelos cerebelares. También hemos equipado EDLUT con un entorno de trabajo para robots. Este entorno permite implementar diferentes esquemas de control en los que poder embeber nuestros modelos cerebelares. EDLUT incluye ahora: (i) una interfaz que transforma las señales analógicas generadas por los robots en impulsos que las redes neuronales son capaces de entender y viceversa, (ii) una interfaz de comunicación TCP/IP capaz de conectar los esquema de control con la interfaz del robot (que pueden ser ejecutados en el mismo o en diferentes computadores), y (iii) un supervisor de RT que asegura el cumplimiento de la restricción de RT. Además, el núcleo de EDLUT ha sido modificado para permitir que la simulación de algunos elementos neuronales (generación y propagación de impulsos, mecanismo de plasticidad, evolución de la dinámica neuronal, etc.) puedan ser temporalmente deshabilitados con el objetivo de cumplir la restricción de RT. Esta nueva característica es controlada por el supervisor de RT, el cual puede acelerar o decelerar la simulación.

Además de esto, en esta tesis hemos hecho un extenso uso de todas estas herramientas para estudiar el cerebelo. Hemos estudiado como diferentes neuronas, sinapsis y mecanismos de plasticidad que han sido reportados experimentalmente en el cerebelo contribuyen a sus habilidades de control motor. Más concretamente hemos centrado nuestro estudio en el lazo cerrado formado por la IO, PCs y DCN (cerrando el lazo desde los DCN hasta la IO a través del cuerpo). Aunque la IO no se considera parte del cerebelo, se cree que sus axones (CFs) transportan hasta el cerebelo una señal de aprendizaje que codifica el error e influye en la modificación del peso sináptico de las PFs (Ito and Kano 1982, Ito 2001). Hemos estudiado la relación existente entre estos tres grupos de neuronas en varias tareas de control y aprendizaje motor con robots: (i) manipulación de objetos pesados/ligeros usando un brazo robótico simulado (Luque et al. 2014, Luque et al. 2016), (ii) recreación de un experimento EBCC usando un entorno simulado (Antonietti et al. 2016), y (iii) recreación de un experimento VOR usando un robot iCub simulado y también real (sección de resultados adicionales).

Con respecto a los mecanismo de plasticidad sináptica, tres mecanismo de plasticidad dependiente del tiempo del impulso (STDP) implementando LTP y LTD en tres grupos de sinapsis (PFs a PCs, MFs a DCN y PCs a DCN) son evaluados en esta tesis (Luque et al. 2016). Hemos visto como en una primera fase del aprendizaje la actividad de las CFs a nivel de las PCs influye en la adaptación del peso sináptico de sus PFs aferentes (axones provenientes de las GCs) con el objetivo de minimizar el error en la tarea motora. Esta señal de error se computa en las células de la IO y se transmiten a las PCs a través de las CFs. En una segunda fase del aprendizaje, la actividad de las PCs que llega a los DCN influyen en la adaptación del peso sináptico de sus MFs aferentes. De esta forma el aprendizaje es consolidado mediante la transferencia de parte de la información almacenada en las sinapsis PFs/PCs hacia las sinapsis MFs/DCN. Además, estos mecanismos actúan como un mecanismo de adaptación de ganancia capaz de optimizar el rango operacional de las células de los DCN.

Por último, en la última fase de esta tesis hemos propuesto un nuevo modelo cerebelar con tres contribuciones principales (sección de resultados adicionales): (i) un modelo neuronal detallado de las PCs capaz de replicar sus tres estados de actividad llamados tónico, silencioso y ráfaga (Forrest 2008), (ii) una nueva capa IO interconectada mediante acoplamiento eléctrico capaz de codificar la señal de error de forma más eficiente, y (iii) una nueva conexión sináptica de la IO hasta los DCN. Hemos valida este nuevo modelo cerebelar usando un experimento VOR con un robot iCub simulado y real en RT. Este es el primer experimento en el que hemos usado un robot real (en vez de un robot simulado como en los anteriores trabajos). Para este trabajo hemos desarrollado un esquema de control de lazo cerrado de entrada/salida asíncrono para facilitar el control del robot real. Este esquema de control aprovecha la diferencia de tiempo entre el tiempo de propagación biológico de los impulsos a través de los nervios y el tiempo de propagación artificial de los impulsos a través de los sistemas artificiales para relajar la constricción de RT. Este esquema de control ha sido detallado en mayor profundidad en el segundo capítulo de esta tesis.

5.6 Marco del proyecto

El trabajo descrito en este documento ha sido desarrollado en el marco de dos proyectos Europeos: “Realistic Real-time Networks: computation dynamics in the cerebellum” (REALNET (FP7-270434)) y “Human Brain Project” (HBP (FP7-604102)). La última parte de esta tesis (sección de resultados adicionales) ha sido desarrollada en colaboración con el grupo de investigación liderado por Angelo Arleo en la Universidad Pierre y Marie Curie (UPMC), París.

El proyecto REALNET era una continuación del proyecto Europeo SENSOPAC (IST-028056). REALNET (fundado bajo el 7º programa marco de la UE de tecnologías de la información y la comunicación) empezó en febrero de 2011 y terminó en febrero de 2014. El objetivo principal de este proyecto era elaborar una red neuronal realista y usarla, junto con registros experimentales de la actividad de redes biológicas, para investigar las bases teóricas de cómputo del sistema nervioso central. Se usó un circuito cerebelar como caso de estudio. Basado en datos experimentales, este proyecto desarrolló el primer modelo realista en tiempo real de un cerebelo y lo conectó a un sistema robótico para evaluar la funcionalidad del circuito bajo condiciones de lazo cerrado. Dentro de este proyecto, nuestro grupo de investigación perteneciente a la Universidad de Granada estaba centrado en el desarrollo del simulador EDLUT para hacerlo capaz de simular estas estructuras biológicas realistas en tiempo real incrementando el rendimiento de las simulaciones.

Más recientemente nuestro grupo está envuelto en el proyecto HBP. Este es un proyecto insignia de la Comisión Europeo para las tecnologías emergentes del futuro. Fue lanzado en octubre de 2013, y está programado que dure diez años. El HBP tiene como objetivo poner en marcha una infraestructura científica de vanguardia para investigar el cerebro, la neurociencia cognitiva y la computación inspirada en el cerebro. El HBP se divide en 12 subproyectos. Nuestro grupo de investigación perteneciente a la Universidad de Granada se encuentra actualmente integrado en el subproyecto 10: Plataforma Neurorobótica. Esta plataforma ofrece a los científicos y desarrolladores tecnológicos una infraestructura software y hardware que permite conectar modelos de cerebelo prevalidados a cuerpos robóticos y entornos detalladamente simulados. Dentro de este subproyecto, la Universidad de Granada se integra en el paquete de trabajo 10.1 Experimentos en lazo cerrado (modelos de cerebro dirigidos por datos), desarrollando la subtarea 10.1.4 control motor cerebelar. Aquí hemos continuado con el desarrollo de nuestro modelo cerebelar, validándolo en experimentos de lazo cerrado.

Finalmente, el doctorando ha hecho una estancia de seis meses en el grupo de investigación liderado por Angelo Arleo en la UPMC, financiado parcialmente por una beca de movilidad asociada a las FPUs del Ministerio de Educación Español. Durante este periodo hemos desarrollado y validado un modelo cerebelar más realista embebido en un esquema de control de lazo cerrado capaz de recrear un experimento VOR usando un robot simulado y real en RT.

5.7 Organización de los capítulos

Este documento ha sido organizado en capítulos por razones de legibilidad y organización. Cada capítulo contiene:

- El capítulo 1 es una breve introducción a la neurociencia computacional y los modelos cerebelares aplicados a experimentos de lazo cerrado usando robots biomorficos. Este capítulo también describe la motivación de esta tesis.
- El capítulo 2 contextualiza el trabajo presentado en esta tesis con respecto a trabajos previos.
- El capítulo 3 enumera las contribuciones principales de esta tesis y el trabajo futuro.
- El capítulo 4 es un compendio de los artículos de revista que dan soporte a esta tesis, incluyendo una breve descripción sobre las revistas y los índices de calidad de las publicaciones. Este capítulo también incluye una sección de resultado adicionales pendientes de publicar.
- El capítulo 5 es el capítulo 1 escrito en español.
- El capítulo 6 es el capítulo 3 escrito en español.
- El anexo incluye los cuatro artículos de revista que componen esta tesis.

Capítulo 6: Conclusiones y Trabajo Futuro

Este capítulo muestra un resumen de las principales contribuciones presentadas en esta tesis, las conclusiones que pueden ser extraídas y una propuesta para trabajo futuro.

6.1 Revisando los objetivos de la tesis

Esta tesis persigues tres objetivos complementarios:

- Desarrollar las herramientas de simulación que habilitan este estudio, explorando nuevos y eficientes métodos de simulación, esquemas de control, interfaces de comunicación y supervisores de RT. Esto se ha logrado a través del desarrollo de la plataforma EDLUT, integrando diferentes técnicas que permiten un cómputo muy eficiente. Se ha realizado un estudio completo de su capacidad de rendimiento y como optimizarlo.
- Crear modelos cerebelares que expliquen los mecanismos biológicos que permiten que el cerebelo y algunos centros relacionados realicen diferentes tareas de control motor. Este objetivo se cumple mediante el desarrollo de modelos cerebelares que integran nuevas características biológicas tales como mecanismos de plasticidad sináptica distribuidos en distintas capas (de PFs a PCs dirigido por las CFs, de MFs a los DCN dirigido por las PCs y de las PCs a los DCN), complejos modelos neuronales (modelo tri-modal de las PCs), una capa de la IO capaz de codificar mejor la señal de error usando acoplamiento eléctrico y una nueva sinapsis de las capas IO a DCN.
- Implementación de novedosos esquemas de control biológicamente inspirados capaces de controlar las nuevas generaciones de robots biomórficos. Los modelos cerebelares han sido embebidos en diferentes esquemas de control en lazo cerrado y expuestos a exigentes tareas de control motor utilizando robots simulados o reales en RT. Esto se ha hecho usando esquemas de control biológicamente plausibles, cumpliendo así con este tercer objetivo.

6.2 Contribuciones principales

- Hemos desarrollado y actualizado un eficiente simulador de redes neuronales de impulsos (EDLUT) basado en simulaciones hibridas dirigidas por eventos y por tiempo. Se le han aplicado un amplio rango de técnicas de simulación con la idea de incrementar su rendimiento computacional. Así hemos hecho posible el objetivo de realizar tareas de control motor con robots reales en RT utilizando modelos cerebelares biológicamente inspirados.
  - ✓ El núcleo de EDLUT, capaz de realizar simulaciones hibridas dirigidas por eventos y por tiempo, ha si paralelizado en CPUs multinúcleo usando OpenMP.
  - ✓ Los métodos de simulación dirigidos por tiempo han sido también paralelizados en plataformas de coprocesamiento CPU-GPU usando CUDA. La GPU únicamente computa la evolución de la dinámica neuronal mientras que la generación y propagación de impulsos se siguen procesando en la CPU.
  - ✓ Se han desarrollado nuevos métodos de simulación dirigidos por eventos y por tiempo especializados en la simulación de modelos de neurona relativamente complejos (AdEx y HH). Para los métodos dirigidos por eventos se ha desarrollado un nuevo mecanismo

    capaz de reordenar las tablas de consulta junto con un nuevo protocolo para generar y procesar actividad síncrona de forma mucho más eficiente. Para los métodos dirigidos por tiempo de la CPU y la GPU se ha desarrollado un nuevo método de integración de doble paso fijo. Este método es un hibrido entre los métodos de paso fijo y paso variable capaz de explotar las fortalezas y mitigar las debilidades de ambos al mismo tiempo.
    - ✓ Se han desarrollado varias mejoras en lo referente a la computación de la evolución de la dinámica neuronal, la generación y propagación de impulsos y la actualización de los pesos sinápticos usando mecanismo de plasticidad.
    - ✓ El núcleo de EDLUT ha sido modificado con el objetivo de permitir la deshabilitación momentánea de elementos neurales (generación y propagación de impulsos, mecanismos de plasticidad, evolución de la dinámica neuronal, etc.) para cumplir con las restricciones de RT impuestas por los robots reales. Esta característica la controla automáticamente un supervisor de RT, el cual puede acelerar o decelerar la simulación.
- Se han integrado en EDLUT nuevas características adicionales más allá de la simulación de redes neuronales de impulsos tales como:
    - ✓ Implementar diferentes esquemas de control en lazo cerrado en los que poder embeber nuestros modelos cerebelares usando simulaciones síncronas o asíncronas.
    - ✓ Un simulador capaz de simular la planta de un amplio rango de robots.
    - ✓ Una interfaz de comunicación capaz de convertir las señales analógicas de los robots en impulsos que las redes neuronales puedan entender y viceversa.
    - ✓ Una interfaz de comunicación capaz de conectar el esquema de control con la interfaz de cualquier robot simulado o real usando conexiones TCP/IP.
- También se ha desarrollado una versión incremental de un modelo cerebelar biológicamente plausible.
    - ✓ Se han propuesto tres mecanismos de plasticidad diferentes en nuestro modelo de cerebelo (de PFs a PCs dirigido por las CFs, de MFs a los DCN dirigido por las PCs y de las PCs a los DCN). Estos mecanismos generan tres contribuciones principales al comportamiento del modelo cerebelar.
        - ▪ Calibrar en una primera fase la relación sensoriomotora usando el primer mecanismo de plasticidad a nivel de las PCs.
        - ▪ Consolidar en una segunda fase la memoria sináptica formada con el primer mecanismo de plasticidad a nivel de las PCs mediante la replicación de esta distribución de memoria sináptica en el segundo mecanismo de plasticidad a nivel de los DCN.
        - ▪ El tercer mecanismo de plasticidad actúa como un mecanismo de adaptación de ganancia que optimiza el rango de trabajos de los DCN.
    - ✓ Se ha incluido en el modelo cerebelar un nuevo modelo de neurona más complejo capaz de replicar los tres estados de actividad de las PCs.
    - ✓ También se ha incluido una nueva capa IO que implementa acoplamiento eléctrico entre sus neuronas. Este acoplamiento ayuda a codificar mejor la señal de error.
    - ✓ Por último se ha evaluado una nueva sinapsis desde la IO a los DCN.
- Los modelos cerebelares propuestos han sido validados en diferentes tareas de control y aprendizaje motor.

- ✓ Manipulación de objetos pesados/ligeros que modifican el modelo básico dinámico de un brazo robótico simulado en RT.
- ✓ Recreación de un experimento EBCC usando un entorno simulado.
- ✓ Recreación de un experimento VOR usando un robot iCub simulado y real en RT.

6.3 Conclusiones

En esa tesis hemos tratado de arrojar alguna luz sobre la teoría de control motor relacionada con el cerebelo, proponiendo y estudiando modelos cerebelares basados en hipótesis y resultados extraídos de la biología. Estos modelos han sido evaluados en diferentes tares de control motor usando robots simulados y reales en RT. Además, en esta tesis hemos desarrollado e integrado en una sola herramienta (EDLUT) todos los elementos requeridos por este tipo de experimentos.

Cuando se requiere la simulación de redes neuronales de impulsos capaces de interactuar con robots simulados o reales en RT, es obligatorio utilizar un simulador que asegure esta condición de RT. Con esta idea en mente hemos desarrollado EDLUT, un simulador de código abierto para redes neuronales de impulsos con un esquema de simulación híbrido dirigido por eventos y por tiempo. Dicho simulador ha sido paralelizado en plataformas de coprocesamiento CPU-GPU multinúcleo habilitando un supervisor de RT. Este esquema hibrido nos brinda múltiples técnicas de simulación que pueden ser usadas conjuntamente para acelerar la simulación de diferentes capas neuronales dentro de la misma red (e.g. métodos dirigidos por eventos para las GCs y dirigidos por tiempo para las PCs). Además, este simulador integra el software necesario para implementar esquemas de control en lazo cerrado capaces de controlar robots simulados o reales en RT.

En cuanto a los modelos cerebelares propuestos en esta tesis, hemos demostrado cómo un modelo biológicamente inspirado puede ser usado para diversas tares de control motor tales como manipulación de objetos o los experimentos EBCC y VOR. Así mismo se han propuesto y estudiado varías características nuevas sobre nuestro modelo de cerebelo: (i) Dos nuevos mecanismo de plasticidad sináptica a nivel de los DCN capaces de consolidar el aprendizaje a nivel de las PCs y ajustar el rango dinámico de salida de los DCN, (ii) un nuevo modelo de neurona más complejo capaz de recrear los tres estados de actividad de las PCs, (iii) una nueva capa IO capaz de codificar mejor la señal de error utilizando acoplamiento eléctrico entre sus neuronas, y (iv) una nueva sinapsis desde la IO a los DCN. Sin embargo, todavía estamos bastante lejos de entender todas las características de cerebelo, siendo capaces de recrear un modelo cerebelar perfecto. Por este mismo motivo debemos seguir investigando esta área en el futuro.

6.4 Trabajo futuro

El primer paso que deberemos dar una vez finalizada esta tesis será proceder a la publicación de los resultados pendientes de publicación del cuarto capítulo. Estos resultados se dividirán en dos artículos separados (aunque ambos relacionados). El primero abordará el modelo de cerebelo desde el punto de vista de la neurociencia. Presentaremos como todos los elementos incluidos en nuestro modelo son biológicamente plausibles, a la vez que demostraremos como contribuyen al comportamiento de nuestro modelo cerebelar. El segundo artículo será mucho más técnico, centrándonos en los problemas técnicos que hemos tenido que abordar para poder recrear un experimento VOR usando un robot iCub simulado y real en RT.

Después de esto trataremos de seguir explorando las capacidades de nuestro modelo cerebelar. Recientes estudios *in vivo* e *in vitro* (Bidoret et al. 2009, Bouvier et al. 2016) han observado como los mecanismo de plasticidad de las PFs a las PCs están controlados por la actividad de las CFs. Estos estudios proponen que los mecanismo LTD y LTP de las PFs requieren una determinada actividad para modificar el peso sináptico. Esta actividad debe darse en ráfagas de 2 impulsos para LTD o 5 impulsos para LTP y producir así una modificación del peso sináptico. Además, los impulsos dentro de estás ráfagas deben poseer una alta frecuencia (200Hz). Para ráfagas más cortas o con una menor frecuencia no se observaron modificaciones del peso sináptico. Así mismo, los autores de estos estudios han propuesto un modelo matemático capaz de recrear este comportamiento. En futuras versiones de nuestro modelo cerebelar sustituiremos nuestra ley de aprendizaje en las PFs por esta nueva versión, evaluando como este cambio puede modificar la dinámica del modelo respecto con la versión anterior. De hecho, dado que esta nueva ley de aprendizaje requiere que las GCs generen ráfagas de actividad a una determinada frecuencia, necesitaremos también rediseñar la capa de las GCs para que cumplan estos requisitos. Con este objetivo en mente colaboraremos con el Dr. Jesús Garrido (codirector de esta tesis y experto en la capa de las GCs). Él ha estudiado extensamente las capas formadas por las MFs, GCs y células Golgi (GoCs), evaluando diferentes mecanismos de plasticidad distribuida (Garrido et al. 2016, Mapelli et al. 2015, Luque, Garrido, et al. 2011a, D'Angelo et al. 2009, D’Angelo et al. 2016). Trataremos de integrar sus capas MFs, GCs y GoCs con nuestras capas IO, PCs y DCN, construyendo de esta forma nuestro modelo cerebelar biológicamente plausible más complejo hasta la fecha. Por último incluiremos en nuestro modelo dos nuevos tipos de interneuronas: basket y stellate. Estas células reciben sinapsis excitatorias de las PFs e inhiben a las PCs. La hipótesis propuesta es que estas capas intermedias ayudan a las PCs a diferenciar mejor los patrones en las PFs, al tiempo que optimizan el rango de trabajo de las PCs.

Con respecto al desarrollo de nuestro simulador EDLUT, a lo largo de esta tesis hemos mejorado drásticamente el rendimiento computacional de esta herramienta, haciéndola capaz de simular modelos cerebelares más complejos en RT. Tras esta tesis planeamos seguir desarrollando nuestro modelo cerebelar, incrementando su complejidad neuronal y matemática, pero al mismo tiempo deseamos poder seguir realizando experimentos con robots reales en RT. Como hemos visto en esta tesis, la combinación de ambas opciones al mismo tiempo requerirá seguir con el desarrollo de EDLUT, explorando nuevas formas de incrementar su rendimiento computacional. En este sentido exploraremos la paralelización de EDLUT en clusters, pero teniendo siempre en cuenta la restricción de RT que imponen los robots reales.

También exploraremos como optimizar el nuevo mecanismo de plasticidad en las PFs. Este nuevo mecanismo utiliza dos variables de estado y una ecuación diferencial para cada PF. Las variables de estado experimentan un incremento directo cada vez que un impulso llega a través de una PF o CF respectivamente y un decremento exponencial con el tiempo. En nuestro último modelo cerebelar hemos implementado 400,000 PFs. De esta forma deberemos evaluar 800,000 funciones de decremento exponencial y 400,000 ecuaciones diferenciales, generando una enorme carga computacional. Implementaremos y compararemos tres configuraciones distintas: métodos de integración de paso fijo y de paso variable en CPU y de paso fijo en una plataforma de coprocesamiento CPU-GPU (el incremento directo de las variables de estado se realizará en la CPU, mientras que los decrementos exponenciales así como la integración de la ecuación diferencial se realizará en la GPU).

Por último, como hemos expresado anteriormente, evaluaremos las nuevas características de nuestro modelo cerebelar cuando este esté envuelto en diferentes tares de control motor usando robots simulados o reales en RT.

BIBLIOGRAPHY

Albus, James S. 1971. "A theory of cerebellar function." *Mathematical Biosciences* no. 10 (1–2):25-61. doi: http://dx.doi.org/10.1016/0025-5564(71)90051-4.

Antonietti, Alberto, Claudia Casellato, Jesús A Garrido, Niceto R Luque, Francisco Naveros, Eduardo Ros, Egidio D'Angelo, and Alessandra Pedrocchi. 2016. "Spiking neural network with distributed plasticity reproduces cerebellar learning in eye blink conditioning paradigms." *IEEE Transactions on Biomedical Engineering* no. 63 (1):210-219.

Arbib, Michael A, Giorgio Metta, and Patrick van der Smagt. 2008. "Neurorobotics: from vision to action." In *Springer Handbook of Robotics*, 1453-1480. Springer.

Arenz, Alexander, R Angus Silver, Andreas T Schaefer, and Troy W Margrie. 2008. "The contribution of single synapses to sensory representation in vivo." *Science* no. 321 (5891):977-980.

Beira, Ricardo, Manuel Lopes, Miguel Praça, José Santos-Victor, Alexandre Bernardino, Giorgio Metta, Francesco Becchi, and Roque Saltarén. 2006. Design of the robot-cub (icub) head. Paper read at Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.

Bezzi, Michele, Thierry Nieus, Olivier J-M Coenen, and Egidio D'Angelo. 2004. "An integrate-and-fire model of a cerebellar granule cell." *Neurocomputing* no. 58:593-598.

Bidoret, Céline, Annick Ayon, Boris Barbour, and Mariano Casado. 2009. "Presynaptic NR2A-containing NMDA receptors implement a high-pass filter synaptic plasticity rule." *Proceedings of the National Academy of Sciences* no. 106 (33):14126-14131.

Boucheny, Christian, Richard Carrillo, Eduardo Ros, and J-MD Coenen Olivier. 2005. Real-time spiking neural network: An adaptive cerebellar model. Paper read at International Work-Conference on Artificial Neural Networks.

Bouvier, Guy, David Higgins, Maria Spolidoro, Damien Carrel, Benjamin Mathieu, Clément Léna, Stéphane Dieudonné, Boris Barbour, Nicolas Brunel, and Mariano Casado. 2016. "Burst-Dependent Bidirectional Plasticity in the Cerebellum Is Driven by Presynaptic NMDA Receptors." *Cell Reports* no. 15 (1):104-116. doi: 10.1016/j.celrep.2016.03.004.

Bower, JM, and D Beeman. 1998. "The book of Genesis: exploring realistic neural models with the GEneral NEural SImulation System. 1998." *Telos, Springer, New York*.

Brette, R., M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, M. Zirpe, T. Natschlager, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. El Boustani, and A. Destexhe. 2007. "Simulation of networks of spiking neurons: A review of tools and strategies." *Journal of Computational Neuroscience* no. 23 (3):349-398. doi: 10.1007/s10827-007-0038-6.

Cajal, Santiago Ramon Y. 1894. "The Croonian Lecture: La fine structure des centres nerveux." *Proceedings of the Royal Society of London* no. 55 (331-335):444-468.

Carrillo, Richard. 2009. *Simulación eficiente de estructuras neuronales basadas en el sistema nervioso*, Department of Computer Architecture and Technology, University of Granada.

Carrillo, Richard R, Eduardo Ros, Christian Boucheny, and J-MD Coenen Olivier. 2008. "A real-time spiking cerebellum model for learning robot control." *Biosystems* no. 94 (1):18-27.

Casellato, Claudia, Alberto Antonietti, Jesus A. Garrido, Giancarlo Ferrigno, Egidio D'Angelo, and Alessandra Pedrocchi. 2015. "Distributed cerebellar plasticity implements generalized multiple-scale memory components in real-robot sensorimotor tasks." *Frontiers in Computational Neuroscience* no. 9 (24). doi: 10.3389/fncom.2015.00024.

Clopath, Claudia, Aleksandra Badura, Chris I De Zeeuw, and Nicolas Brunel. 2014. "A cerebellar learning model of vestibulo-ocular reflex adaptation in wild-type and mutant mice." *Journal of Neuroscience* no. 34 (21):7203-7215.

D'Angelo, Egidio, SKE Koekkoek, Paola Lombardo, S Solinas, E Ros, J Garrido, Martijn Schonewille, and CI De Zeeuw. 2009. "Timing in the cerebellum: oscillations and resonance in the granular layer." *Neuroscience* no. 162 (3):805-815.

D'Angelo, Egidio, Alberto Antonietti, Stefano Casali, Claudia Casellato, Jesus A Garrido, Niceto Rafael Luque, Lisa Mapelli, Stefano Masoli, Alessandra Pedrocchi, and Francesca Prestori. 2016. "Modeling the Cerebellar Microcircuit: New Strategies for a Long-Standing Issue." *Frontiers in Cellular Neuroscience* no. 10.

Delorme, Arnaud, and Simon J Thorpe. 2003. "SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons." *Network: Computation in Neural Systems* no. 14 (4):613-627.

Devor, Anna, and Yosef Yarom. 2002. "Generation and propagation of subthreshold waves in a network of inferior olivary neurons." *Journal of neurophysiology* no. 87 (6):3059-3069.

Eccles, JC, M Ito, and J Szentágothai. 1967. The cerebellum as a neuronal machine. Springer, New York.

Fine, Edward J, Catalina C Ionita, and Linda Lohr. 2002. The history of the development of the cerebellar examination. Paper read at Seminars in neurology.

Flash, Tamar, and Neville Hogan. 1985. "The coordination of arm movements: an experimentally confirmed mathematical model." *Journal of neuroscience* no. 5 (7):1688-1703.

Forrest, Michael. 2008. *Biophysics of Purkinje computation*, Department of Computer Science, University of Warwick.

Fujita, M. 1982. "Adaptive filter model of the cerebellum." *Biological cybernetics* no. 45 (3):195-206.

Garrido Alcazar, Jesus A, Niceto Rafael Luque, Egidio D‘Angelo, and Eduardo Ros. 2013. "Distributed cerebellar plasticity implements adaptable gain control in a manipulation task: a closed-loop robotic simulation." *Frontiers in neural circuits* no. 7:159.

Garrido, Jesus A, Richard R Carrillo, Niceto R Luque, and Eduardo Ros. 2011. Event and time driven hybrid simulation of spiking neural networks. Paper read at International Work-Conference on Artificial Neural Networks.

Garrido, Jesús A, Niceto R Luque, Silvia Tolu, and Egidio D’Angelo. 2016. "Oscillation-driven spike-timing dependent plasticity allows multiple overlapping pattern recognition in inhibitory interneuron networks." *International journal of neural systems* no. 26 (05):1650020.

Garrido, Jesús A. 2012. *Simulation of biological neuronal structures. Design and functional study of the cerebellum*, Department of Computer Architecture and Technology, University of Granada.

Gerstner, Wulfram, and Werner M Kistler. 2002. *Spiking neuron models: Single neurons, populations, plasticity*: Cambridge university press.

Gewaltig, Marc-Oliver, and Markus Diesmann. 2007. "NEST (neural simulation tool)." *Scholarpedia* no. 2 (4):1430.

Golgi, Camillo. 1906. "The neuron doctrine: theory and facts." *Nobel lecture* no. 1921:190-217.

Gordon, JL, JMR Furman, and EW Kamen. 1989. System identification of the vestibulo-ocular reflex: application of the recursive least-squares algorithm. Paper read at Bioengineering Conference, 1989., Proceedings of the 1989 Fifteenth Annual Northeast.

Hines, M. L., and N. T. Carnevale. 1997. "The NEURON Simulation Environment." *Neural Computation* no. 9 (6):1179-1209. doi: 10.1162/neco.1997.9.6.1179.

Honda, Takeru, Tadashi Yamazaki, Shigeru Tanaka, Soichi Nagao, and Tetsuro Nishino. 2011. "Stimulus-dependent state transition between synchronized oscillation and randomly repetitive burst in a model cerebellar granular layer." *PLoS Comput Biol* no. 7 (7):e1002087.

Houk, James C, Jay T Buckingham, and Andrew G Barto. 1996. "Models of the cerebellum and motor learning." *Behavioral and brain sciences* no. 19 (3):368-383.

Hwang, Eun Jung, and Reza Shadmehr. 2005. "Internal models of limb dynamics and the encoding of limb state." *Journal of neural engineering* no. 2 (3):S266.

Iserles, Arieh. 2009. *A first course in the numerical analysis of differential equations*: Cambridge university press.

Ito, Masao. 1970. "Neurophysiological aspects of the cerebellar motor control system." *International journal of neurology* no. 7 (2):162-176.

Ito, Masao. 1984. *The cerebellum and neural control*: Raven Pr.

Ito, Masao. 2001. "Cerebellar long-term depression: characterization, signal transduction, and functional roles." *Physiological reviews* no. 81 (3):1143-1195.

Ito, Masao. 2006. "Cerebellar circuitry as a neuronal machine." *Progress in neurobiology* no. 78 (3):272-303.

Ito, Masao, and Masanobu Kano. 1982. "Long-lasting depression of parallel fiber-Purkinje cell transmission induced by conjunctive stimulation of parallel fibers and climbing fibers in the cerebellar cortex." *Neuroscience letters* no. 33 (3):253-258.

Izawa, Jun, Sarah E Pekny, Mollie K Marko, Courtney C Haswell, Reza Shadmehr, and Stewart H Mostofsky. 2012. "Motor Learning Relies on Integrated Sensory Inputs in ADHD, but Over-Selectively on Proprioception in Autism Spectrum Conditions." *Autism Research* no. 5 (2):124-136.

Kandel, Eric R, James H Schwartz, Thomas M Jessell, Steven A Siegelbaum, and A James Hudspeth. 2000. *Principles of neural science*. Vol. 4: McGraw-hill New York.

Kawato, Mitsuo. 1990. "Feedback-error-learning neural network for supervised motor learning." *Advanced neural computers* no. 6 (3):365-372.

Kawato, Mitsuo, Kazunori Furukawa, and R Suzuki. 1987. "A hierarchical neural-network model for control and learning of voluntary movement." *Biological cybernetics* no. 57 (3):169-185.

Kawato, Mitsuo, and Hiroaki Gomi. 1992. "A computational model of four regions of the cerebellum based on feedback-error learning." *Biological cybernetics* no. 68 (2):95-103.

Khan, Muhammad Mukaram, David R Lester, Luis A Plana, A Rast, Xin Jin, Eustace Painkras, and Stephen B Furber. 2008. SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor. Paper read at Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on.

Kuroda, Shinya, Kenji Yamamoto, Hiroyuki Miyamoto, Kenji Doya, and Mitsuo Kawato. 2001. "Statistical characteristics of climbing fiber spikes necessary for efficient cerebellar learning." *Biological cybernetics* no. 84 (3):183-192.

Latorre, Roberto, Carlos Aguirre, Mikhail I Rabinovich, and Pablo Varona. 2013. "Transient dynamics and rhythm coordination of inferior olive spatio-temporal patterns."

Leigh, R John, and David S Zee. 2015. *The neurology of eye movements*. Vol. 90: Oxford University Press, USA.

Lisberger, SG, and AF Fuchs. 1978. "Role of primate flocculus during rapid behavioral modification of vestibuloocular reflex. II. Mossy fiber firing patterns during horizontal head rotation and eye movement." *Journal of Neurophysiology* no. 41 (3):764-777.

Luque, Niceto R. 2014. *Bio-inspired robotic control schemes using biologically plausible neural structures*, Department of Computer Architecture and Technology, University of Granada.

Luque, Niceto R, Richard R Carrillo, Francisco Naveros, Jesús A Garrido, and MJ Sáez-Lara. 2014. "Integrated neural and robotic simulations. Simulation of cerebellar neurobiological substrate for an object-oriented dynamic model abstraction process." *Robotics and Autonomous Systems* no. 62 (12):1702-1716.

Luque, Niceto R, Jesus A Garrido, Richard R Carrillo, J-MD Coenen Olivier, and Eduardo Ros. 2011a. "Cerebellar input configuration toward object model abstraction in manipulation tasks." *IEEE Transactions on Neural Networks* no. 22 (8):1321-1328.

Luque, Niceto R, Jesús A Garrido, Francisco Naveros, Richard R Carrillo, Egidio D'Angelo, and Eduardo Ros. 2016. "Distributed cerebellar motor learning: a spike-timing-dependent plasticity model." *Frontiers in computational neuroscience* no. 10.

Luque, Niceto R, Jesus A Garrido, Jarno Ralli, Juanlu J Laredo, and Eduardo Ros. 2012. "From sensors to spikes: Evolving receptive fields to enhance sensorimotor information in a robot-arm." *International journal of neural systems* no. 22 (04):1250013.

Luque, Niceto R, Jesús Alberto Garrido, Richard R Carrillo, Silvia Tolu, and Eduardo Ros. 2011. "Adaptive cerebellar spiking model embedded in the control loop: context switching and robustness against noise." *International Journal of Neural Systems* no. 21 (05):385-401.

Luque, Niceto Rafael, Jesús Alberto Garrido, Richard Rafael Carrillo, J-M D Coenen Olivier, and Eduardo Ros. 2011b. "Cerebellarlike corrective model inference engine for manipulation tasks." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* no. 41 (5):1299-1312.

Luque, NR, JA Garrido, RR Carrillo, OJMD Coenen, and E Ros. 2011. "Cerebellar-like corrective-model abstraction engine for robot movement control." *IEEE Transaction on system, man, and cybernetics-Part B* no. 41 (5).

Llinas, R, R Baker, and C Sotelo. 1974. "Electrotonic coupling between neurons in cat inferior olive." *Journal of Neurophysiology* no. 37 (3):560-571.

Llinas, RR, Kerry D Walton, and EJ Lang. 2004. "Ch. 7 Cerebellum." *Shepherd GM. The Synaptic Organization of the Brain*.

Mapelli, Lisa, Martina Pagani, Jesus A Garrido, and Egidio D’Angelo. 2015. "Integrated plasticity at inhibitory and excitatory synapses in the cerebellar circuit."

Marr, David. 1969. "A theory of cerebellar cortex." *The Journal of Physiology* no. 202 (2):437-470.1.

Mathy, Alexandre, Sara SN Ho, Jenny T Davie, Ian C Duguid, Beverley A Clark, and Michael Häusser. 2009. "Encoding of oscillations by axonal bursts in inferior olive neurons." *Neuron* no. 62 (3):388-399.

Mattia, Maurizio, and Paolo Del Giudice. 2000. "Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses." *Neural Computation* no. 12 (10):2305-2329.

Medina, Javier F, and Michael D Mauk. 1999. "Simulations of cerebellar motor learning: computational analysis of plasticity at the mossy fiber to deep nucleus synapse." *Journal of Neuroscience* no. 19 (16):7140-7151.

Medina, Javier F, and Michael D Mauk. 2000. "Computer simulation of cerebellar information processing." *nature neuroscience* no. 3:1205-1211.

Metta, Giorgio, Paul Fitzpatrick, and Lorenzo Natale. 2006. "Yarp: Yet another robot platform." *International Journal of Advanced Robotic Systems* no. 3 (1):8.

Middleton, Steven J, Claudia Racca, Mark O Cunningham, Roger D Traub, Hannah Monyer, Thomas Knöpfel, Ian S Schofield, Alistair Jenkins, and Miles A Whittington. 2008. "High-frequency network oscillations in cerebellar cortex." *Neuron* no. 58 (5):763-774.

Miyasho, Tsugumichi, Hiroshi Takagi, Hideo Suzuki, Shigeo Watanabe, Masashi Inoue, Yoshihisa Kudo, and Hiroyoshi Miyakawa. 2001. "Low-threshold potassium channels and a low-threshold calcium channel regulate Ca 2+ spike firing in the dendrites of cerebellar Purkinje neurons: a modeling study." *Brain research* no. 891 (1):106-115.

Najafi, Farzaneh, and Javier F Medina. 2013. "Beyond "all-or-nothing" climbing fibers: graded representation of teaching signals in Purkinje cells." *Frontiers in neural circuits* no. 7:115.

Nakano, Eri, Hiroshi Imamizu, Rieko Osu, Yoji Uno, Hiroaki Gomi, Toshinori Yoshioka, and Mitsuo Kawato. 1999. "Quantitative examinations of internal representations for arm trajectory planning: minimum commanded torque change model." *Journal of Neurophysiology* no. 81 (5):2140-2155.

Naveros, Francisco, Jesus A Garrido, Richard R Carrillo, Eduardo Ros, and Niceto R Luque. 2017. "Event- and Time-Driven Techniques Using Parallel CPU-GPU Co-Processing for Spiking Neural Networks." *Frontiers in Neuroinformatics* no. 11:7.

Naveros, Francisco, Niceto R Luque, Jesús A Garrido, Richard R Carrillo, Mancia Anguita, and Eduardo Ros. 2015. "A spiking neural simulator integrating event-driven and time-driven computation schemes using parallel CPU-GPU co-processing: a case study." *IEEE transactions on neural networks and learning systems* no. 26 (7):1567-1574.

Ott, Ch, Oliver Eiberger, Werner Friedl, B Bauml, Ulrich Hillenbrand, Ch Borst, Alin Albu-Schaffer, Bernhard Brunner, H Hirschmuller, and S Kielhofer. 2006. A humanoid two-arm system for dexterous manipulation. Paper read at Humanoid Robots, 2006 6th IEEE-RAS International Conference on.

Passot, Jean-Baptiste, Niceto Rafael Luque, and Angelo Arleo. 2013. "Coupling internal cerebellar models enhances online adaptation and supports offline consolidation in sensorimotor tasks." *Frontiers in computational neuroscience* no. 7:95.

Pecevski, Dejan, David Kappel, and Zeno Jonke. 2014. "NEVESIM: event-driven neural simulation framework with a Python interface." *Frontiers in neuroinformatics* no. 8:70.

Pfeifer, R, and J Bongard. 2007. How the Body shapes the Way we think: How the body shapes the way we think: A new view of intelligence. MIT Press, Cambridge, MA.

Pfeil, Thomas, Andreas Grübl, Sebastian Jeltsch, Eric Müller, Paul Müller, Mihai A Petrovici, Michael Schmuker, Daniel Brüderle, Johannes Schemmel, and Karlheinz Meier. 2012. "Six networks on a universal neuromorphic computing substrate." *arXiv preprint arXiv:1210.7083*.

Porrill, John, Paul Dean, and James V Stone. 2004. "Recurrent cerebellar architecture solves the motor-error problem." *Proceedings of the Royal Society of London-B* no. 271 (1541):789-796.

Purves, Dale, DAVID Fitzpatrick, WC Hall, GJ Augustine, and AS Lamantia. 2008. Neuroscience . Sunderland, Mass: Sinauer. ISBN 0-87893-697-1.

Rabbitt, Richard D, Edward R Damiano, and J Wallace Grant. 2004. "Biomechanics of the semicircular canals and otolith organs." In *The vestibular system*, 153-201. Springer.

Raman, Indira M, and Bruce P Bean. 1999. "Ionic currents underlying spontaneous action potentials in isolated cerebellar Purkinje neurons." *Journal of Neuroscience* no. 19 (5):1663-1674.

Reutimann, Jan, Michele Giugliano, and Stefano Fusi. 2003. "Event-driven simulation of spiking neurons with stochastic dynamics." *Neural Computation* no. 15 (4):811-830.

Robinson, DA. 1981. "The use of control systems analysis in the neurophysiology of eye movements." *Annual review of neuroscience* no. 4 (1):463-503.

Ros, Eduardo, Richard Carrillo, Eva M Ortigosa, Boris Barbour, and Rodrigo Agís. 2006. "Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics." *Neural computation* no. 18 (12):2959-2993.

Rudolph, Michelle, and Alain Destexhe. 2006. "Analytical integrate-and-fire neuron models with conductance-based dynamics for event-driven simulation strategies." *Neural computation* no. 18 (9):2146-2210.

Schemmel, Johannes, Daniel Briiderle, Andreas Griibl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. 2010. A wafer-scale neuromorphic hardware system for large-scale neural modeling. Paper read at Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on.

Schmolesky, Matthew T, John T Weber, Chris I Zeeuw, and Christian Hansel. 2002. "The making of a complex spike: ionic composition and plasticity." *Annals of the New York Academy of Sciences* no. 978 (1):359-390.

Schweighofer, Nicolas, Kenji Doya, and Mitsuo Kawato. 1999. "Electrophysiological properties of inferior olive neurons: a compartmental model." *Journal of neurophysiology* no. 82 (2):804-817.

Shibata, Tomohiro, and Stefan Schaal. 2001. "Biomimetic gaze stabilization based on feedback-error-learning with nonparametric regression networks." *Neural Networks* no. 14 (2):201-216.

Siciliano, Bruno, and Oussama Khatib. 2016. *Springer handbook of robotics*: Springer.

Siegel, Allan, and Hreday N Sapru. 2006. *Essential neuroscience*: Lippincott Williams & Wilkins.

Skavenski, ALEXANDER A, and DAVID A Robinson. 1973. "Role of abducens neurons in vestibuloocular reflex." *Journal of Neurophysiology* no. 36 (4):724-738.

Solinas, Sergio, Thierry Nieus, and Egidio D'Angelo. 2010. "A realistic large-scale model of the cerebellum granular layer predicts circuit spatio-temporal filtering properties." *Frontiers in cellular neuroscience* no. 4 (12).

Sotelo, C, R Llinas, and R Baker. 1974. "Structural study of inferior olivary nucleus of the cat: morphological correlates of electrotonic coupling." *Journal of Neurophysiology* no. 37 (3):541-559.

Sporns, Olaf. 2006. "Small-world connectivity, motif composition, and complexity of fractal neuronal connections." *Biosystems* no. 85 (1):55-64.

Thach, W Thomas, HP Goodkin, and JG Keating. 1992. "The cerebellum and the adaptive coordination of movement." *Annual review of neuroscience* no. 15 (1):403-442.

Thach, WT. 1967. "Somatosensory receptive fields of single units in cat cerebellar cortex." *Journal of neurophysiology* no. 30 (4):675-696.

Thompson, Richard F. 1990. "Neural mechanisms of classical conditioning in mammals." *Philosophical Transactions of the Royal Society of London B: Biological Sciences* no. 329 (1253):161-170.

Todorov, Emanuel. 2004. "Optimality principles in sensorimotor control." *Nature neuroscience* no. 7 (9):907-915.

Tolu, Silvia, Mauricio Vanegas, Jesus A Garrido, Niceto R Luque, and Eduardo Ros. 2013. "Adaptive and predictive control of a simulated robot arm." *International journal of neural systems* no. 23 (03):1350010.

Tolu, Silvia, Mauricio Vanegas, Niceto R Luque, Jesús A Garrido, and Eduardo Ros. 2012. "Bio-inspired adaptive feedback error learning architecture for motor control." *Biological cybernetics*:1-16.

Tsagarakis, Nikolaos G, Giorgio Metta, Giulio Sandini, David Vernon, Ricardo Beira, Francesco Becchi, Ludovic Righetti, Jose Santos-Victor, Auke Jan Ijspeert, and Maria Chiara Carrozza. 2007. "iCub: the design and realization of an open humanoid platform for cognitive and neuroscience research." *Advanced Robotics* no. 21 (10):1151-1175.

Vannucci, Lorenzo, Silvia Tolu, Egidio Falotico, Paolo Dario, Henrik Hautop Lund, and Cecilia Laschi. 2016. Adaptive gaze stabilization through cerebellar internal models in a humanoid robot. Paper read at Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on.

Vijayakumar, Sethu, and Stefan Schaal. 2000. Locally weighted projection regression: An O (n) algorithm for incremental real time learning in high dimensional space. Paper read at Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000).

Voogd, Jan, and Mitchell Glickstein. 1998. "The anatomy of the cerebellum." *Trends in cognitive sciences* no. 2 (9):307-313.

Wolf, Uri, Mark J Rapoport, and Tom A Schweizer. 2009. "Evaluating the affective component of the cerebellar cognitive affective syndrome." *The Journal of neuropsychiatry and clinical neurosciences* no. 21 (3):245-253.

Wulff, Peer, Martijn Schonewille, Massimiliano Renzi, Laura Viltono, Marco Sassoè-Pognetto, Aleksandra Badura, Zhenyu Gao, Freek E Hoebeek, Stijn Van Dorp, and William Wisden. 2009. "Synaptic inhibition of Purkinje cells mediates consolidation of vestibulo-cerebellar motor learning." *Nature neuroscience* no. 12 (8):1042-1049.

Yamazaki, Tadashi, and Shigeru Tanaka. 2005. "Neural modeling of an internal clock." *Neural computation* no. 17 (5):1032-1058.

Yamazaki, Tadashi, and Shigeru Tanaka. 2007. "The cerebellum as a liquid state machine." *Neural Networks* no. 20 (3):290-297.

Yamazaki, Tadashi, and Shigeru Tanaka. 2009. "Computational models of timing mechanisms in the cerebellar granular layer." *The cerebellum* no. 8 (4):423-432.

Annex: Journal articles included in this thesis

In this annex we have included the four journal articles that compose this thesis.

A Spiking Neural Simulator Integrating Event-Driven and Time-Driven Computation Schemes Using Parallel CPU-GPU Co-Processing: A Case Study

F. Naveros, N. R. Luque, J. A. Garrido, R. R. Carrillo, M. Anguita, E. Ros

Abstract — **Time-driven simulation methods in traditional CPU architectures perform well and precisely when simulating small-scale spiking neural networks. Nevertheless, they still have drawbacks when simulating large-scale systems. Conversely, event-driven simulation methods in CPUs and time-driven simulation methods in graphic processing units (GPUs) can outperform CPU time-driven methods under certain conditions. With this performance improvement in mind, we have developed an event-and-time-driven spiking neural network simulator suitable for a hybrid CPU-GPU platform. Our neural simulator is able to efficiently simulate bio-inspired spiking neural networks consisting of different neural models which can be distributed heterogeneously in both small layers and large layers or subsystems. For the sake of efficiency, the low-activity parts of the neural network can be simulated in CPU by using event-driven methods whilst the high-activity subsystems can be simulated in either CPU (a few neurons) or GPU (thousands or millions of neurons) by using time-driven methods. In this work, we have undertaken a comparative study of these different simulation methods. For benchmarking the different simulation methods and platforms, we have used a cerebellar-inspired neural-network model consisting of a very dense granular layer and a Purkinje layer with a smaller number of cells (according to biological ratios). Thus, this cerebellar-like network includes a dense diverging neural layer (increasing the dimensionality of its internal representation and sparse coding) and a converging neural layer (integration) as many other biologically inspired and also artificial neural networks.**

Index Terms — **co-processing CPU-GPU, EDLUT, event-driven execution, real time, simulation, spiking neural network, time-driven execution.**

Manuscript received November 28, 2013. This work has been supported by the EU grant REALNET (FP7-270434). We gratefully acknowledge the support of NVIDIA Corporation with the donation of two Titan GPUs for current EDLUT development.

F. Naveros is a PhD student; N. R Luque and R. R. Carrillo are post-doc researchers; Mancia Anguita is an associate professor and Eduardo Ros is a full professor (U. of Granada), Periodista Daniel Saucedo s/n, Granada (Spain). (E-mail: fnaveros@ugr.es, nluque@ugr.es, rcarrillo@ugr.es, manguita@ugr.es, eros@ugr.es).

J. A. Garrido is a post-doc researcher at the Neurophysiology Unit, Dept. of Brain and Behavioural Sciences, University of Pavia, Pavia, Italy and Consorzio Interuniversitario per le Scienze Fisiche della Materia (CNISM), Pavia, Italy (E-mail: jesus.garrido@unipv.it)

I. INTRODUCTION

ONE of the main challenges addressed by neuroscientists in the twenty-first century consists of understanding the biological principles of consciousness and mental processes through which we perceive, act, learn and remember [1]. But understanding the computing principles involved requires simulations at different levels of detail and at different scales. Within the context of neurorobotics, the possibility of simulating biologically plausible neural networks connected to an active agent such as a robot [2] [3] [4] allows us to study behavioral features using body-brain closed-loop experiments. Nevertheless, this imposes strict constraints on computation time (ideally, real-time simulations for perception-action set ups with real robots).

Studying brain computational principles is the main goal of computational neuroscience, where widely used neural simulators such as GENESIS [5], NEURON [6], Brian [7], and NEST [8] play a fundamental role. These neural simulators usually calculate neuronal dynamics using time-driven simulation methods. These methods divide the total simulation time into small time steps and update each neural-state variable in each time step by means of numerical analysis methods [9]. This iterative updating process represents a heavy computational load, which depends linearly on the number of neurons and neural variables. This computational load may also depend on the number of synapses if we use synapse models with intrinsic dynamics that need to be computed continuously. For synapses where the parameters can be calculated within the event-driven scheme, the computational load would only be slightly affected by the spike propagation and learning rule application processes. Hence, this total computational load makes large-scale neural network simulations in real-time almost intractable if directly iterative numerical methods are used.

Alternatively, different approaches and techniques have been developed over the last decades; event-driven neural network simulations [10] [11] [12], neural network simulations in high-performance platforms such as field programmable gate-array circuits (FPGAs) [13] [14], very large-scale integration (VLSI) circuits [15], and graphic processor units (GPUs) [16] [17] [18] [19] [20], and finally, distributed neural network simulations in clusters [21] represent different approaches for large scale simulation.

As a starting point we chose the open source simulator EDLUT (event-driven neural simulator based on look-up tables). The very first EDLUT version incorporated only an event-driven simulation scheme into its design [10]. A further step in the technological development of this simulator made it capable of performing hybrid event-and-time-driven simulations [22]. Finally, to take full advantage of new multi-core CPU architectures, time-driven methods have been parallelized by using OpenMP.

This article describes how EDLUT has been further developed so as to perform event-and-time-driven simulations in a hybrid CPU-GPU platform. This hybrid platform allows computing large-scale simulations efficiently on a single computer. Whilst event-driven simulation methods based on look-up tables are always running on CPU, iterative time-driven simulation methods can be run either on CPU (single core and multi-core CPU architectures) or GPU, depending on the number of neurons to be computed (a layer of a few neurons can be more efficiently computed on CPU architecture, whilst GPU architectures are better suited for larger scale neural systems). This paper also presents a comparative study of all these simulation methods operating in our hybrid CPU-GPU simulator for event-and-time-driven simulations.

II. SIMULATION METHODS IN A HYBRID CPU-GPU ARCHITECTURE

According to Brette [23], an event-and-time-driven simulator, such as EDLUT, could be divided into three processing sections: a) neuronal dynamics (using time-driven and/or event-driven methods), b) spike propagation, and c) event queue management. Thus the computation time can be divided in $t_{\_neu\_dynamics}$, $t_{\_spike\_prop}$ and $t_{\_queue\_manag}$, respectively. In fact, EDLUT uses several techniques to improve the neuronal dynamic computation, thus making this simulator perform better when the neural integration phase (a) prevails over others (b and c).

Parallelizing time-driven methods in both CPU and GPU, would improve the performance of the simulator whilst maintaining its flexibility and accuracy [19]. This parallelization procedure enables the use of more complex time-driven neuron models and shorter integration time steps, thus obtaining more realistic and accurate results. Furthermore, event-driven methods (based on look-up tables) are best suited for simulating simple neuron models with high levels of performance and precision.

Within the field of computational neuroscience, GPUs have also proven useful in speeding up the computation of time-driven simulation methods in some neural networks thanks to their particular parallel architecture. These GPUs depend on CPU hosts for their operations. When the CPU only initializes the simulation and the GPU computes the whole simulation, some constraints and simplifications have to be taken into account in order to parallelize the entire simulation in the GPU properly, as in NeMo [24] and GeNN [25], (i.e. adopting deterministic delay propagation, time-driven simulation, fixed step integration methods, avoiding event-driven simulation, etc.). Alternatively, when the GPU operates conjointly with the CPU (CPU-GPU platform) such constraints are no longer required since any parallelizable task can be independently performed on GPU, whilst sequential ones can be performed in CPU as in Brian [7] and our simulator. This means that whilst EDLUT is running, the GPU updates the neural-state variables in time-driven methods and the CPU generates, propagates and receives spikes, processes learning rules, and updates the neural state variables in both time-driven and event-driven methods.

To better understand how this hybrid CPU-GPU platform operates, we need to bear in mind GPU pros and cons as well as the implications of establishing a CPU-GPU communication process. The synchronization and communication bridge between the CPU and GPU represents the main bottleneck of this platform. The dialogue between these two architectures starts when the CPU commands the GPU to update its time-driven neurons. Before starting to work, the GPU consumes time (this is called *synchronization time*). Furthermore, there is another overhead time which occurs when the CPU updates the GPU with the newest variable values obtained (this overhead time is called *transfer time*). The number of neurons in the GPU must be high enough to compensate both overhead times and then to achieve good speed-up rates (compared to a CPU-only processing engine). Therefore it is clear that when using a hybrid CPU-GPU platform it becomes crucial to minimize both *synchronization* and *transfer times*. The GPU has to implement fixed step integration methods capable of updating all GPU neurons simultaneously, although the CPU can implement both fixed and variable step integration methods.

For the comparative study, we have chosen the leaky integrate-and-fire neural model (LIF) because it requires only a few state variables to be implemented. A low number of state variables helps to maintain the synchronization and information transference between CPU and GPU per integration step time to a minimum and also facilitates the model implementation onto look-up tables for event-driven methods.

A. Leaky integrate-and-fire model (LIF)

The leaky integrate-and-fire model [26] was implemented using an event-driven method in CPU and a time-driven method in both CPU and GPU. Event-driven methods make use of characterization tables stored in a binary file, which are pre-calculated in a previous off-line stage [10]. They contain the particular look-up tables that characterize the dynamics of each cell and allow updating neural state variables discontinuously at any simulation time. Time-driven methods make use of a text file containing all the particular parameters which are required to configure each cell type (i.e. granule cell, Purkinje cell, etc.).

The neural state is characterized by the membrane potential (V_{m-c}), which is expressed by (1):

$$C_m \frac{dV_{m-c}}{dt} = g_{AMPA}(t)(E_{AMPA} - V_{m-c}) + g_{GABA}(t)(E_{GABA} - V_{m-c}) + G_{rest}(E_{rest} - V_{m-c}) \tag{1}$$

where C_m denotes the membrane capacitance, E_{AMPA} and E_{GABA} represent the reversal potential of each synaptic conductance, and E_{rest} is the resting potential (with G_{rest} as the conductance responsible for the passive decay term towards the resting potential). The conductances g_{AMPA} and g_{GABA} integrate all the contributions received through individual synapses and are defined as decaying exponential functions. The parameters of the neural model and a more detailed description can be found in [26].

As shown in (1), the state of a neuron can be defined by using just three state variables:

- V_{m-c} represents the membrane potential. When this variable reaches a specific threshold the neuron generates an output spike.
- g_{AMPA} and g_{GABA} represent excitatory and inhibitory conductances that modify the membrane potential. These conductances decrease exponentially in each integration step and increase proportionally to the synaptic weight of their connections when an input spike arrives.

To solve the LIF neuron model differential equation, a 4th - order Runge-Kutta method was implemented. This differential equation is processed off-line in event-driven methods [10] (to build up the neural characterization look-up-tables) and on-line in time-driven methods [22]. In event-driven methods, the size of the look-up table is critical, affecting the accuracy of the simulation. The execution time is also affected because the number of cache failures is higher for large tables, but this impact is small. In time-driven methods, the key factor of the simulation is the integration step size, which drastically affects the accuracy and the execution time.

Although in this study we only use this neuron model, our neural simulator also includes other neuron models, such as a Leaky Integrate and Fire neuron model with four synaptic conductance types (available for all the techniques described in this work), a Spike Response Model (for event-driven and time-driven in CPU), a Hodgkin-Huxley neuron model (so far only for event-driven in CPU), etc.

B. Implementation

As indicated in Section II.A the neural-state of the LIF neuron model can be defined by using just three state variables; its membrane potential and its excitatory and inhibitory conductances. In the time-driven scheme, a vector in the CPU or GPU global memory stores the three neural-state variables of each neuron belonging to the same neural model (granule cell, Purkinje cell, etc.).

When using CPU methods, these neural-state variables are stored in the CPU global memory. Therefore, with each spike arrival, the CPU updates its corresponding conductance. When using GPU methods, however, these neural-state variables are stored in the GPU global memory and cannot be directly modified by the CPU. In this case, an auxiliary incremental conductance vector with two values per neuron is created in the CPU global memory. The CPU stores the input spike effect for both conductances in this vector (excitatory and inhibitory conductance) at each integration step. When the GPU neurons are updated the auxiliary vector is transferred from the CPU to the GPU and is added to the conductance values stored in the GPU global memory. Thus, only one single transfer from the CPU to the GPU takes place for each integration step.

Moreover, after the GPU neurons are updated, the GPU has to indicate to the CPU which neurons generate an output spike. To do this a Boolean vector, in which each position represents a neuron, is used. The GPU assigns the value ‘True’ to the corresponding Boolean vector position when its related neuron has to generate an output spike (its membrane potential reaches a specific threshold) or otherwise assigns the value ‘False’. When the neural updating has finished, the Boolean vector is transferred from the GPU to the CPU. Once again, just a single transfer from the GPU to the CPU takes place for each integration step. Finally, the CPU generates the output spikes in the corresponding neurons, inserting them into the event queue [10] [22], which stores all relevant events throughout the simulation process. It is worth mentioning that this CPU-GPU memory management and also the CPU-GPU communication procedure can be used with more complex neuron models (i.e. Izhikevich or Hodgkin-Huxley neuron models).

Once the structure of the neural-state variables, their distribution within the memory hierarchies in a hybrid architecture, and the communication protocol between the CPU and GPU have been stated, it is time to define the five steps required to update the neural-state variables of each group of time-driven neurons in GPU:

1. Transferring the auxiliary incremental conductance vector from the CPU to the GPU.
2. Updating the neural-state variables with the new incoming conductance values. This updating is computed according to the neural model associated to

each neuron group. If a particular neuron has to generate an output spike, the GPU assigns the value 'True' in its corresponding Boolean vector position or 'False' otherwise.

3. Transferring the Boolean vector from the GPU to the CPU.

4. Resetting the auxiliary incremental conductance vector.

5. Generating the output spikes in the CPU by using the Boolean vector (these generated output spikes are stored in the event queue).

Nevertheless, using the mapped-memory version (see appendix A), the GPU can perform the first and third steps automatically since it is capable of reading and writing CPU memory (the so-called auxiliary incremental conductance vector and Boolean vector).

All these processes can be seen in the flow diagram of the new simulator (Fig. 1).

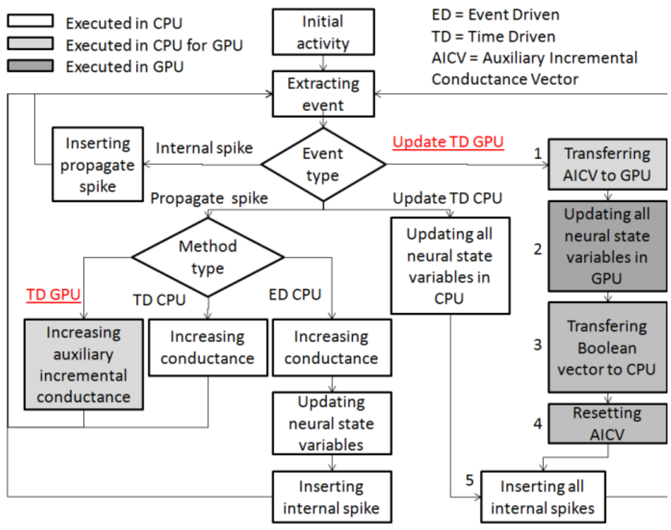


Fig. 1. EDLUT simulator flow diagram. This figure shows the original flow diagram (white blocks) for event-driven and time-driven methods in CPU and the added blocks for the time-driven method in the GPU. These new blocks are executed in the CPU (light-gray blocks) or GPU (dark-gray blocks). The light-gray blocks manage the input information towards the GPU and the output information from it. The dark-gray block updates the neural-state variables in the GPU. An internal spike event [10] generates a spike inside a neuron. A propagate spike event [10] delivers all output spikes after an internal spike event.

The hardware used for running these simulations consists of an ASUS P8Z68-VPRO motherboard, an Intel core i7 2600K 2nd-generation 3.4 GHz CPU (four real cores, eight virtual cores), 32 GB RAM memory DDR3 1333 MHz, and finally, an NVIDIA GTX 470 GPU with 1280 MB RAM memory GDDR5. This GPU model supports mapped memory (see appendix A). All the CPU-GPU simulations use this technique.

C.Neural network topology

To measure the performance of the three simulation methods in our hybrid CPU-GPU implementation we ran different biologically plausible neural network simulations. A neural network representing an abstraction of the granular and Purkinje layers of the cerebellum [27] was built as shown in Fig. 2. The cerebellum was structured into micro-zones [28], each of which included ten thousand neurons distributed in three different layers, as described below:

- *Mossy fiber layer:* an input layer consisting of 800 neurons that receive and convey the input network activity to the second layer. This layer is implemented by means of an event-driven method since it is not necessary to update its neural-state variables.
- *Granular layer:* consisting of 9,120 neurons that mimic cerebellar granule cells. These neurons have four, randomly chosen, input connections from the first layer corresponding to mossy fiber/granular layer connectivity in biological systems [29] [30]. These input connections have delays of 1ms. This layer is the most extensive and time-consuming section of the network. We implement this layer using a time-driven method, both in CPU and GPU, and an event-driven method in CPU to compare the performance achieved with each computing scheme.
- *Purkinje cell layer:* an output layer consisting of 80 output neurons that mimic cerebellar Purkinje cells. The connections between the granular and Purkinje layers are configured in such a way that each Purkinje neuron has an 80% probability of being contacted by all the granular neurons of its own micro-zone and a 20% probability of being contacted by all the granular neurons in its two adjacent micro-zones [31]. These input connections have delays of 3ms. The number of input synapses (on average) to these neurons is very high (roughly 10 940 synapses per neuron), making an event-driven model somewhat unsuitable for this layer [22]. In addition, the quantity of neurons is not high enough to be run efficiently in a GPU; consequently, in the comparative study in following sections, this layer is implemented in a CPU using a time-driven method.

All the connections are excitatory synapses, there being about 91.2 synapses per neuron in the network on average. This particular network does not use any learning rules, although the simulator can implement different STDP learning rules.

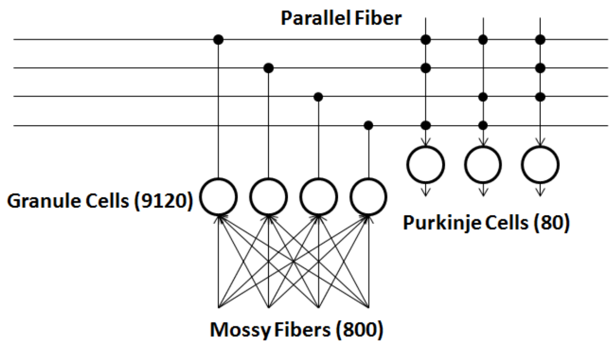


Fig. 2. A neural network representing an abstraction of the granular and Purkinje layers of the cerebellum. The parallel fibers are the axons of the granular cells that eventually contact different Purkinje cells.

This modular structure, organized in micro-zones in the cerebellum, can be seen in other areas of the brain (for instance cortical columns in the cortex [1]). These local neural structures have very intensive local connectivity but sparser connectivity with other micro-areas (such as neighborhood micro-zones or cortical columns in the cortex).

The input activity supplied to each micro-zone has been taken from a real simulation of a complete cerebellar model in a manipulation task using six micro-zones. This activity derives from the sensorimotor (proprioceptor) activity obtained from a 3-joint robot arm executing a figure-of-eight-like trajectory in one second [4].

The network performance was measured under three different input activity levels which generate 3, 10, and 20 Hz average firing rate over all the neural network activity respectively during 10 seconds' simulation time. These three input activity levels were obtained from the original input spike pattern. These new inputs correspond to 3, 6, and 9 executions of the figure-of-eight-like trajectory during 10 seconds' simulation time.

For further details the reader may check and download the EDLUT source code from the project's official website [32]. The configuration files, neuron models and input activity files used in this comparative study can be provided upon request.

III. Results

This simulator incorporates different simulation techniques which can work conjointly. As indicated in the previous section, we are simulating a modular three-layer cerebellar spiking neural network where the input mossy fiber layer is always simulated by means of an event-driven method; the granular layer can be simulated by means of an event-driven method in CPU or a time-driven method in both CPU or GPU, and finally the Purkinje cell layer is always computed using a time-driven method in CPU. The characteristics of each simulation technique at granular layer level are evaluated in two different experiments:

1) Different integration step sizes to be used in time-driven methods and different look-up table sizes to be used in event-driven methods running in the already established three-layer cerebellar network particularized to a pre-fixed neural network size.

2) Two fixed integration step sizes and two fixed look-up table sizes to be used in the established three-layer cerebellar network scaled to different neural network sizes. These look-up tables are constructed for the event-driven simulation scheme, in order to obtain a similar accuracy to the time-driven simulation scheme with the two fixed integration steps.

First experimental set up: We have evaluated the accuracy and performance of the simulation techniques (time-driven methods in CPU and GPU, and event-driven methods in CPU) when adopting these different approaches for the granular layer simulation. The fixed neural network consists of a micro-zone containing 10 thousand neurons and 620 thousand synapses. This neural network operates under three defined averaged firing rate neural network activities (3, 10, and 20 Hz).

In order to compare the neural dynamics computation time ($t_{\_neu\_dynamics}$) against the spike propagation and queue management time ($t_{\_spike\_prop}$ and $t_{\_queue\_manag}$), a profiling of the time-driven CPU technique has been performed. The Y axis of Fig.3 shows the execution time ratio between the total simulation time ($t_{\_neu\_dynamics}$ + $t_{\_spike\_prop}$ + $t_{\_queue\_manag}$) and the time of the spike propagation and queue management ($t_{\_spike\_prop}$ + $t_{\_queue\_manag}$). Fig.3 shows the ideal speed-up rate that could be achieved if the neural dynamics computation time ($t_{\_neu\_dynamics}$) were reduced to zero when the time-driven CPU technique is used (see Amdahl's law [33]).

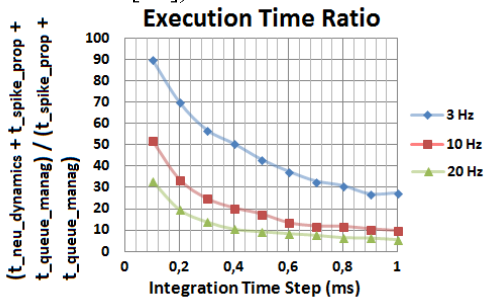


Fig. 3. Ideal speed-up rate that could be achieved by the time-driven CPU technique. We show how the speed-up rate would behave if the neural dynamics computation time ($t_{\_neu\_dynamics}$) were reduced towards zero and only the spike propagation and the queue management had to be processed ($t_{\_spike\_prop}$ and $t_{\_queue\_manag}$). Simulations run using different integration-step with a neural network consisting of one micro-zone of 10 thousand neurons and 620 thousand synapses. Three averaged firing rate neural network activities are used (3, 10, and 20 Hz respectively).

Once the neural dynamics computation time impact has been pondered for the time-driven CPU technique, the aim of this experiment lies in measuring the impact of using different integration-steps and look-up table sizes in GPU and CPU respectively. We use the Van Rossum distance [34] with respect to a 1 µs-integration-step-time-driven simulation in order to compare the accuracy of the results. We also compared the speed-up rate obtained with the time-driven method in GPU and the event-driven method in CPU compared to the stand alone time-driven method in CPU.

As shown in Fig. 4.a., whilst decreasing the integration step size, both time-driven CPU and GPU simulation techniques achieve smaller Van Rossum distances, which means a higher accuracy (the smaller the Van Rossum distance values, the higher accuracy obtained and vice-versa). The event-driven CPU simulation technique at the second layer level (granular layer) (Fig. 4.c.), demands an increase of the look-up table size (see Table I) from 2.2 MB to 1750 MB to remain fairly comparable in terms of accuracy with respect to 1ms and 0.1ms time-driven simulations. We should bear in mind that the maximum accuracy that can be reached by the event-driven CPU simulations is limited by the amount of memory available in the CPU for storing look-up tables.

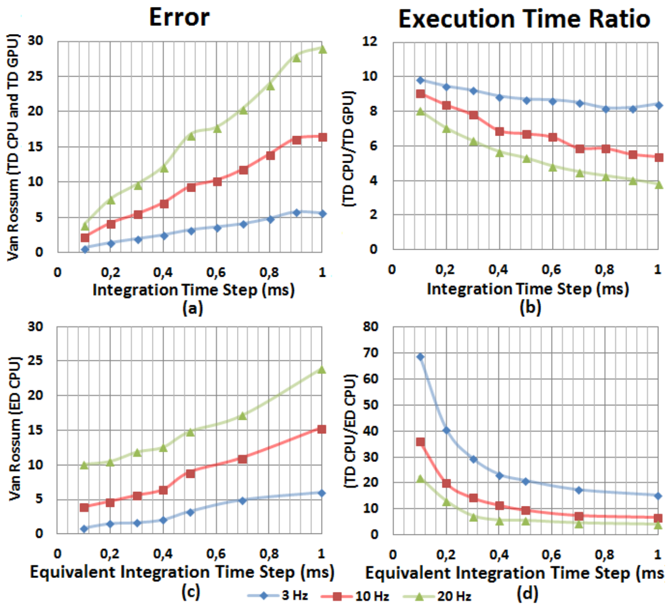


Fig. 4. Simulations with different integration-step and look-up table sizes (equivalent in accuracy as indicated in Table I) in a neural network consisting of one micro-zone of 10 thousand neurons and 620 thousand synapses divided into three layers: the first one is event-driven, the third one is time-driven in CPU and the second one is computed using the three different simulation techniques: TD CPU→ time-driven in CPU; TD GPU→ time-driven in GPU; ED CPU→ event-driven in CPU; and three averaged firing rate neural network activities (3, 10, and 20 Hz respectively). **(a)** Van Rossum distance for both TD CPU and TD GPU with respect to a 1 µs-integration-step-time-driven simulation with tau Van Rossum value=2ms [34]. **(b)** Speed-up rate, which is given as TD CPU/TD GPU execution time ratio. **(c)** Van Rossum distance for ED CPU (for look-up table sizes between 2.2 MB and 1750 MB, see Table I). **(d)** Speed-up rate, which is given as TD CPU/ED CPU execution time ratio.

TABLE I
RELATIONSHIP BETWEEN THE INTEGRATION STEP SIZE OF FIG.4.A AND FIG.4.B AND THE LOOK-UP TABLE SIZE OF FIG.4.C AND FIG.4.D

Integration time step (ms)	Look-Up table Size (MB)
1	2.2
0.7	3.1
0.5	4.2
0.4	52.2
0.3	245
0.2	960
0.1	1750

Furthermore, a progressive reduction of the integration step size leads the time-driven GPU technique to achieve higher speed-up rates (Fig. 4.b.) whilst maintaining the degree of accuracy that the time-driven CPU technique obtains. The event-driven CPU technique is even able to reach higher speed-up rates (Fig. 4.d.) with similar levels of accuracy than time-driven CPU technique. When comparing both plots (Fig. 4.b.and Fig. 4.d.) with Fig. 3, it is clear that the event-driven technique performs better for the proposed small neural network. The event-driven technique reaches speed-up rates quite close to the ideal one whilst the GPU parallel hardware (due to the size of this small neural network) is not capable of operating at its maximum performance.

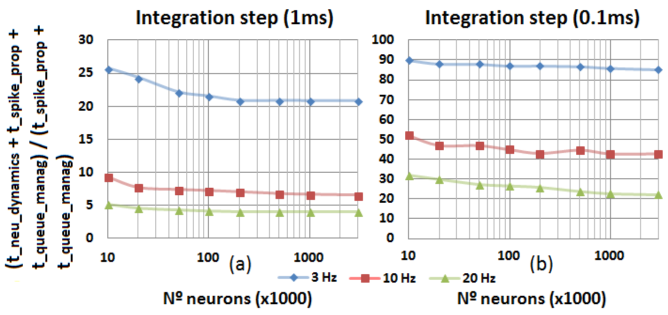


Fig. 5. Ideal speed-up rate that could be achieved in the time-driven CPU technique. We show how the speed-up rate would behave if the neural dynamics computation time ($t_{\_neu\_dynamics}$) were reduced towards zero and only the spike propagation and the queue management had to be processed ($t_{\_spike\_prop}$ and $t_{\_queue\_manag}$). Two fixed integration steps (1ms and 0.1ms) with three averaged firing rate neural network activities (3, 10, and 20 Hz) are used.

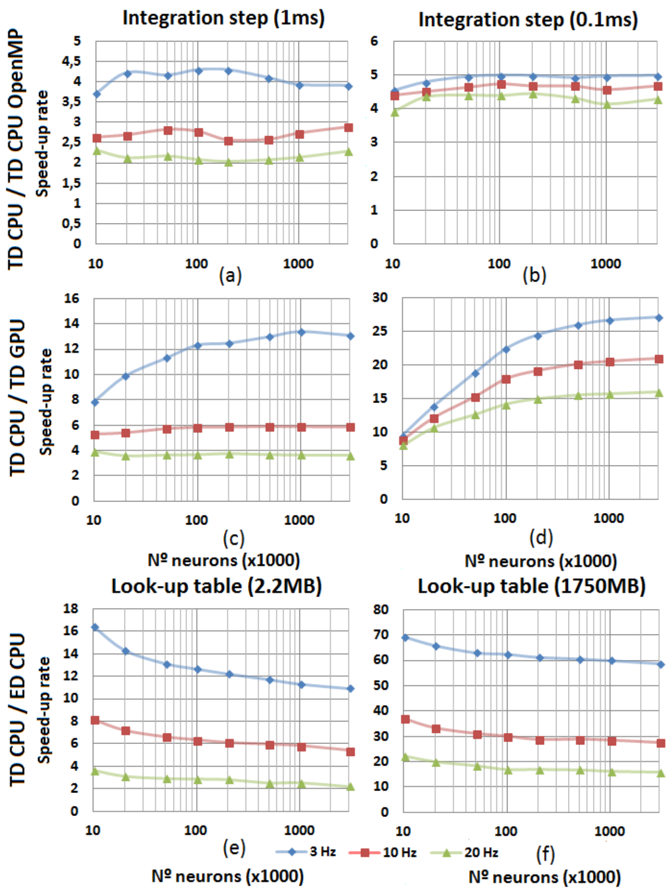


Fig. 6. Speed-up rates achieved by TD CPU OpenMP (time-driven in multi-core CPU with 8 cores) **(a, b)**, TD GPU (time-driven in GPU) **(c, d)** and ED CPU (event-driven in CPU) **(e, f)** techniques with respect to TD CPU (time-driven in CPU) according to the number of neurons. Two fixed integration steps (1ms and 0.1ms) and two look-up tables (2,2 MB and 1750 MB) to achieve similar levels of accuracy with three averaged firing rate neural network activities (3, 10, and 20 Hz) are used.

Second experimental set up (scalability): We have evaluated how our different simulation techniques evolve in terms of performance when the number of micro-zones increases. The techniques to be used at second layer (granular layer) are again time-driven methods in GPU and CPU (adding the possibility of using OpenMP to take full advantage of the multi-core architecture that the CPU presents), and event-driven methods in CPU. To that purpose we have implemented a network with an incremental number of micro-zones from 1 to 300 micro-zones (i.e. from 10 thousand neurons and 620 thousand synapses to 3 million neurons and 274 million synapses).

First of all, the time-driven CPU technique demands a profiling (as already made for the first experiment in Fig.3) to compare and take into account the neural dynamics computation time against the spike propagation and queue management computation time. Fig. 5 shows the ideal speed-up rate that could be achieved if the neural dynamics computation time were reduced to zero when the time-driven CPU technique is used (see Amdahl's law [33]).

As shown in the first four plots of Figure 6, the shorter the integration step or the larger the neural network in time-driven techniques, the higher the speed-up rate of the parallel implementations. This speed-up rate reaches a maximum when those tasks to be parallelized in either CPU (6.a and 6.b) or GPU (6.c and 6.d) are large enough to take full advantage of the hardware. Conversely, event-driven techniques (Figs. 6.e and 6.f) operate quite close to the optimum achievable performance (Figs. 5.a. and 5.b.). The inverse relationship between the neural network size and the speed-up rate in these figures is caused by the event queue management [10] [22].

IV. Discussion

As stated in the abstract, time-driven simulation techniques computed in CPU architectures achieve high performance at high precision when simulating small-scale spiking neural networks. Nevertheless, thanks to brand new features of the current PC architecture, our neural simulator is able to outperform this standalone CPU traditional approach using:

- Multi-core CPU and GPU parallel architectures for time-driven simulations: the gradual increase of processing units in both architectures motivates the parallelization of processes for high performance computing.
- Memory used as a computing resource in event-driven simulations based on look-up tables: the huge amount of available memory in current PCs (up to 32 GB on our hardware) makes this approach a powerful tool when simulating spiking neural networks [10] using event-driven schemes.

The description of how these different simulation methods (event-driven and time-driven) and simulation platforms (multi-core CPU and GPU) are made naturally compatible constitutes the main contribution of this article (Fig.1). This hybrid simulator allows us to compute non-homogeneous simulations in the part of the neural network which is computed using event-driven techniques (usually neural network layers with either low or sparse activity, independently of their size) whilst other parts of this neural network (the ones with higher activity or more complex neural models) can adopt time-driven simulation techniques (in CPU for small layers and using GPU for large layers). This means that the best simulation technique can be selected, taking into account the most suitable part of the neural system, aimed to obtain the best possible performance.

Furthermore, the implementation of all these simulation techniques on the same platform allows us to characterize the performance evolution with each approach when different test-bed neural network sizes are simulated. For the evaluation study, we have used a cerebellar-like spiking neural network as a basis [27] which is a good example, as are many others in the brain, where very large layers (such as the granular layer) with sparse activity are interconnected to other smaller layers (such as the Purkinje layer) with a higher average activity.

Regarding the simulation accuracy, time-driven methods, as is well known, only require the decrease of the integration step size to improve accuracy. Under these circumstances, when a high accuracy is demanded, the GPU simulation techniques reach their maximum performance compared to stand-alone CPU simulation techniques. On the other hand, event-driven methods based on look-up tables have to increase the size of their tables in order to achieve similar accuracy levels to time-driven methods (with shorter integration time steps). The final size of the look-up table depends on the number of neural state variables needed to define the neural model behavior and the required accuracy. Hence, event-driven methods perform optimally for simple neural models such as the one used in our simulations (LIF), with a medium level of accuracy. On the other hand, time-driven methods, especially when they run in GPU, are better suited for more complex neuron models.

With regards to the performance when scaling up the neural network, we have run simulations from thousands of neurons and several hundreds of thousands of synapses to millions of neurons and several hundreds of millions of synapses, under different conditions (integration step sizes, look-up table sizes, and averaged firing rate activities). This allows the evaluation of the scalability of our different simulation techniques and the performance of the different processing platforms:

- *Time-driven CPU with 8 cores and GPU vs. time-driven mono-core CPU*: the larger the neural network or the smaller the averaged firing rate activity or the smaller the integration step are, the higher the speed-up rate (from x2 to x5 using OpenMP parallelism in CPU and from x4 to x27 using GPU).
- *Event-driven CPU vs. time-driven CPU*: the higher accuracy or the smaller averaged firing rate activity,

the higher the speed-up rate. Moreover, the smaller the neural network the easier the event queue management and the higher the speed-up rate (from x2 to x70) when using the event-driven technique.

All this provides a clear picture of the relevance of adopting the right simulation technique and computing resources based on the neural network features. As we have shown, today's technology offers the possibility of simulating millions of neurons on a conventional PC (tens of thousands of neurons can be computed in real time). This is of crucial importance for embedded systems, in which the simulation needs to interact with a robot (body or sensors/actuators). For instance, using a 1ms integration step (equivalent to a 2,2MB look-up table in terms of accuracy when using an event-driven scheme) and a 10 Hz averaged firing rate neural network activity, our traditional time-driven CPU technique can simulate in real-time a neural network consisting of 10 thousand neurons and 624 thousand synapses. Our parallelized time-driven OpenMP CPU technique can simulate up to 20 thousand neurons and 1.53 million synapses in real-time. Finally, our time-driven GPU technique and event-driven CPU technique are able to simulate up to 50 thousand neurons and 4.13 million synapses in real-time. The numbers of neurons and synapses in these examples are related with the neural structure described in Section II.

The performance of the event-driven scheme does not depend directly on the size of the network but rather on the number of events to be processed. The maximum neural network size that EDLUT can simulate is constrained by the available RAM memory in the whole system (taking into account both CPU and GPU RAM memory). Using the cerebellar scheme set out above, we are able to evaluate up to 3 million neurons and 274 million synapses with high average firing rate neural network activity (20 Hz) using 32 GB CPU RAM and 1.28 GB GPU RAM. This simulation on a conventional single computer takes only 987.44 seconds to compute 10 seconds' of simulation time with our time-driven GPU technique and 1623.96 seconds with our event-driven CPU technique.

V. CONCLUSIONS

In order to develop a powerful simulator capable of dealing with neural networks of very different properties (e.g. neural networks ranging from small to very large, from low and sparse activity to very high activity and able to accommodate from simple to complex neural models), it is mandatory to integrate different simulation techniques (with their own pros and cons) on the same simulator.

Throughout this work, we have described a simulator which integrates different simulation methods (event-driven and time-driven schemes) with different simulation techniques (time-driven methods in CPU and GPU, and event-driven methods in CPU) that make use of different processing platforms (single-core and multi-core CPU as well as GPU) in the same simulation. We have studied in detail the pros and cons of each option. To the best of our knowledge this is the first simulator that includes all these different simulation alternatives, thus allowing a simulation performance characterization study such as the one presented in this paper.

Finally, the next step in our large-scale neural network simulator development will be to implement our simulator EDLUT in CPU/GPU clusters, thus improving the spike propagation and queue management time. At the same time, we intend to work on implementing more complex and realistic neural models (i.e. Izhikevich and Hodgkin-Huxley models), taking advantage of the large computing power of the GPU (these neuron models are defined by more than one differential equation, however their state variables can still be kept within GPU memory thus transferring just their conductances from CPU to GPU).

APPENDIX A

EDLUT has been re-programmed to make it compatible with CUDA for NVIDIA GPUs.

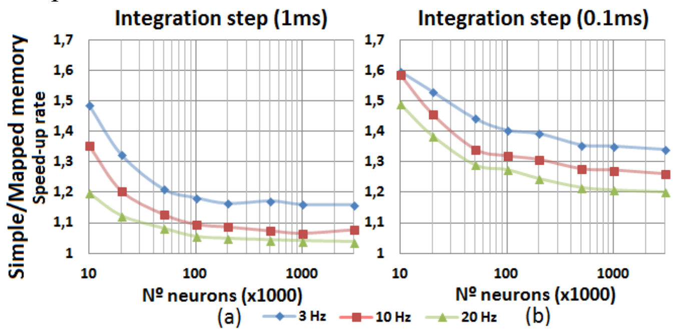


Fig. 7. Speed-up rate achieved by the mapped memory version of TD GPU (time-driven in GPU) compared to the simple version of TD GPU according to the number of neurons. Two fixed integration steps (1ms and 0.1ms) **(a, b)** are used.

The GPU memory should be able to store the neural-state variables of millions or even tens of millions of neurons. The GPU global memory is used to store these variables. To reduce the time consuming impact of the CPU-GPU communication our simulation scheme is able to adjust its operation depending on certain GPU features:

- Legacy GPUs: the CPU writes/reads the GPU global memory. This option is available in all GPU models.

- Non-legacy GPUs: some advanced GPU models incorporate the so-called mapped memory technique. This feature allows the GPU to directly access pre-allocated sections of CPU memory. The simulator is able to check automatically for the presence of this GPU feature and adopts the best possible option.

Fig. 7 shows the differences between the two GPU implementations. The mapped memory version performs faster than the version without mapped memory, especially when the neural network size and/or the integration step size are small.

REFERENCES

[1] E. Kandel, J. Schwartz and T. Jessell, Principles of neural science, 4th ed., Elseiver, 2000.

[2] R. R. Carrillo, E. Ros, C. Boucheny and O. Coenen, "A real-time spiking cerebellum model for learning robot control," *Biosystems,* vol. 94, no. 1-2, pp. 18-27, 2008.

[3] N. Luque, J. Garrido, R. Carrillo, O. Coenen and E. Ros, "Cerebellar Input Configuration Toward Object Model Abstraction in Manipulation Tasks," *IEEE T. Neural Networ.,* vol. 22, no. 8, pp. 1321-1328, 2011.

[4] N. Luque, J. Garrido, R. Carrillo, O. Coenen and E. Ros, "Cerebellarlike Corrective Model Inference Engine for Manipulation Tasks," *IEEE T. Syst. Man Cyb.,* vol. 41, no. 5, pp. 1299-1312, 2011.

[5] J. Bower and D. Beeman, The Book of GENESIS: Exploring Realistic Neural Models with the GEneral Simulation System, 2nd ed., Heidelberg: Springer, 1998.

[6] M. Hines and N. Carnevale, "The NEURON simulation environment," *Neural Comput.,* vol. 9, no. 6, pp. 1179-1209, 1997.

[7] D. Goodman and R. Brette, "The brian simulator," *Frontiers in neuroscience,* vol. 3, no. 2, pp. 192-7, 2009.

[8] M.-O. Gewaltig and M. Diesmann, "NEST (NEural Simulation Tool)," *Scholarpedia,* vol. 2, no. 4, p. 1430, 2007.

[9] R. O'Reilly and Y. Munakata, "Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain," *MIT Press,* 2000.

[10] E. Ros, R. Carrillo, E. Ortigosa, B. Barbour and R. Agís, "Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics," *Neural Comput.,* vol. 18, no. 12, pp. 2959-2993, 2006.

[11] M. Rudolph-Lilith, M. Dubois and A. Destexhe, "Analytical Integrate-and-Fire Neuron Models with Conductance-Based Dynamics and Realistic Postsynaptic Potential Time Course for Event-Driven Simulation Strategies," *Neural Comput.,* vol. 24, no. 6, pp. 1426-1461, 2012.

[12] A. Delorme and S. Thorpe, "SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons," *Network-Comp. Neural,* vol. 14, no. 4, pp. 613-627, 2003.

[13] E. Ros, E. Ortigosa, R. Agís, R. Carrillo and M. Arnold, "Real-Time Computing Platform for Spiking Neurons (RT-Spike)," *IEEE T. Neural Networ.,* vol. 17, no. 4, pp. 1050-1063, 2006.

[14] M. Pearson, A. Pipe, B. Mitchinson, K. Gurney, C. Melhuish, I. Gilhespy and M. Nibouche, "Implementing Spiking Neural Network for Real-Time Signal-Processing and Control Applications: A Model-Validated FPGA Approach," *IEEE T. Neural Networ.,* vol. 18, no. 5, pp. 1472-1487, 2007.

[15] H. Chen, S. Saïghi, L. Buhry and S. Renaud, "Real-Time Simulation of Biologically Realistic Stochastic Neuron in VLSI," *IEEE T. Neural Networ.,* vol. 21, no. 9, pp. 1511-1517, 2010.

[16] A. K. Fidjeland and M. P. Shanahan, "Accelerated Simulation of Spiking Neural Networks Using GPUs," in *IJCNN*, Barcelona, Spain, 2010.

[17] J. Nageswaran, N. Dutt, J. Krichmar, A. Nicolau and A. Veidenbaum, "Efficient simulation of large-scale Spiking Neural Networks using CUDA graphics processors," in *IJCNN*, Atlanta, 2009.

[18] A. Ahmadi and H. Soleimani, "A GPU base simulation of multilayer spiking neural networks," in *19th ICEE*, Amirkabir, 2011.

[19] R. Brette and D. Goodman, "Simulation spiking neural network on GPU," *Network,* vol. 23, no. 4, pp. 167-182, 2012.

[20] M. Richert, J. M. Nageswaran, N. Dutti and J. L. Krichmar, "An efficient simulation environment for modeling large-scale cortical processing," *Frontiers in neuroinformatics,* vol. 5, p. 19, 2011.

[21] C. Chen and T. Taha, "Spiking Neural Networks on High Performance Computer Clusters," in *Conference on Optics and Photonics for Information Processing V,* San Diego, 2011.

[22] J. Garrido, R. Carrillo, N. Luque and E. Ros, "Event and Time Driven Hybrid Simulation of Spiking Neural Networks," *Advances in Computational Intelligence,* pp. 554-561, 2011.

[23] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. J. Harris, M. Zirpe, T. Natschlaeger, D. Pecevske, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. El Boustani and A. Destexhe, "Simulation of networks of spiking neurons: A review of tools and strategies," *J. Comput. Neurosci.,* vol. 23, no. 3, pp. 349-398, 2007.

[24] A. K. Fidjeland, E. B. Roesch, M. P. Shanahan and W. Luk, "NeMo: A Platform for Neural Modelling of Spiking Neurons Using GPUs.," in *ASAP 2009. 20th IEEE International Conference on*, Boston, 2009.

[25] T. Nowotny, "GeNN.," [Online]. Available: http://sourceforge.net/projects/genn/.

[26] W. Gerstner and W. Kistler, "Spiking Neuron Models," *Cambridge University Press,* 2002.

[27] J. Voogd and M. Glickstein, "The anatomy of the cerebellum," *Trends Neurosci.,* vol. 21, no. 9, pp. 370-375, 1998.

[28] O. Oscarsson, "Spatial distribution of climbing and mossy fibre inputs into the cerebellar cortex," in *In Afferent and Intrinsic Organization of Laminated Structures in the Brain*, Berlin, Springer-Verlag, 1976, pp. 34-42.

[29] J. Eccles, M. Ito and J. Szentágothai, The Cerebellum as a Neuronal Machine, Springer-Verlag, 1967.

[30] P. Chadderton, T. Margrie and M. Häusser, "Integration of quanta in cerebellar granule cells during sensory processing," *Nature,* vol. 428, no. 6985, pp. 856-860, 2004.

[31] S. Solinas, T. Nieus and E. D'Angelo, "A realistic large-scale model of the cerebellum granular layer predicts circuit spatio-temporal filtering properties," *Frontiers in Cellular Neuroscience,* vol. 4, no. 12, 2010.

[32] "EDLUT official web site," [Online]. Available: http://edlut.googlecode.com.

[33] G. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," in *AFIPS Conference Proceedings*, 1967.

[34] C. Houghton and T. Kreyz, "On the efficient calculation of Van Rossum distances," *Network-Comp. Neural,* vol. 23, no. 1-2, pp. 48-58, 2012.

METHODS
published: 07 February 2017
doi: 10.3389/fninf.2017.00007

Event- and Time-Driven Techniques Using Parallel CPU-GPU Co-processing for Spiking Neural Networks

Francisco Naveros[1], *Jesus A. Garrido*[1], *Richard R. Carrillo*[1], *Eduardo Ros*[1]* and *Niceto R. Luque*[2, 3]*

[1] Department of Computer Architecture and Technology, Research Centre for Information and Communication Technologies, University of Granada, Granada, Spain, [2] Vision Institute, Aging in Vision and Action Lab, Paris, France, [3] CNRS, INSERM, Pierre and Marie Curie University, Paris, France

Modeling and simulating the neural structures which make up our central neural system is instrumental for deciphering the computational neural cues beneath. Higher levels of biological plausibility usually impose higher levels of complexity in mathematical modeling, from neural to behavioral levels. This paper focuses on overcoming the simulation problems (accuracy and performance) derived from using higher levels of mathematical complexity at a neural level. This study proposes different techniques for simulating neural models that hold incremental levels of mathematical complexity: leaky integrate-and-fire (LIF), adaptive exponential integrate-and-fire (AdEx), and Hodgkin-Huxley (HH) neural models (ranged from low to high neural complexity). The studied techniques are classified into two main families depending on how the neural-model dynamic evaluation is computed: the event-driven or the time-driven families. Whilst event-driven techniques pre-compile and store the neural dynamics within look-up tables, time-driven techniques compute the neural dynamics iteratively during the simulation time. We propose two modifications for the event-driven family: a look-up table recombination to better cope with the incremental neural complexity together with a better handling of the synchronous input activity. Regarding the time-driven family, we propose a modification in computing the neural dynamics: the bi-fixed-step integration method. This method automatically adjusts the simulation step size to better cope with the stiffness of the neural model dynamics running in CPU platforms. One version of this method is also implemented for hybrid CPU-GPU platforms. Finally, we analyze how the performance and accuracy of these modifications evolve with increasing levels of neural complexity. We also demonstrate how the proposed modifications which constitute the main contribution of this study systematically outperform the traditional event- and time-driven techniques under increasing levels of neural complexity.

Keywords: event- and time-driven techniques, CPU, GPU, look-up table, spiking neural models, bi-fixed-step integration methods

OPEN ACCESS

Edited by:
Pedro Antonio Valdes-Sosa, Joint China Cuba Lab for Frontiers Reaearch in Translational Neurotechnology, Cuba

Reviewed by:
Hans Ekkehard Plesser, Norwegian University of Life Sciences, Norway
Jayram Moorkanikara Nageswaran, Brain Corporation, USA

****Correspondence:***
Eduardo Ros
eros@ugr.es
Niceto R. Luque
niceto.luque@inserm.fr

Received: *09 August 2016*
Accepted: *18 January 2017*
Published: *07 February 2017*

Citation:
Naveros F, Garrido JA, Carrillo RR, Ros E and Luque NR (2017) Event- and Time-Driven Techniques Using Parallel CPU-GPU Co-processing for Spiking Neural Networks. Front. Neuroinform. 11:7. doi: 10.3389/fninf.2017.00007

INTRODUCTION

Artificial neural networks (NNs) have been studied since the early 1940's (Mcculloch and Pitts, 1943). These NNs were born as mathematically tractable algorithms that attempted to abstract the learning mechanisms underlying our brain. The natural evolution of these NNs has lately resulted in diverse paradigms including Spiking Neural Networks (SNNs) (Ghosh-Dastidar and Adeli, 2009). These SNNs render a higher biological plausibility by bringing the concept of spike-timing into play. The idea behind the spike-timing concept is based on equipping the neural units (neurons) with the capability to emit spikes when their membrane potentials reach a specific dynamic range (firing regime). Leaky integrate-and-fire (LIF) models, for instance, emit spikes when their membrane potentials reach a specific firing threshold. When a spike is fired, it travels from the source neuron to the target neurons. The spike arrivals to the target neurons may increase or decrease their corresponding membrane potentials depending on their synaptic types and synaptic weights. The spike timing, that is, when a spike is either produced or received, constitutes the foundation for processing the neural information in SNNs and is fundamental to understand brain processing based on spike-timing codification.

Spiking Neural Networks (SNNs) will be considered as highly parallelizable algorithms in which each neural-processing unit (neuron) sends and receives data (spikes) from other neurons. These SNNs are mainly defined by three key factors:

(a) The neural model that defines each neural-processing unit (neurons).
(b) The neural network topology, that is, how the neural-processing units (neurons) are interconnected.
(c) The learning mechanisms that drive adaptation within the SNN at both neural and network level.

The parallelizable algorithmic nature of SNNs makes them perfect candidates for being implemented within a wide variety of specific hardware platforms, such as field programmable gate-array circuits (FPGAs) (Ros et al., 2006b; Agis et al., 2007), very large-scale integration circuits (VLSI) (Pelayo et al., 1997; Schemmel et al., 2010) or specific purpose clusters, such as SpiNNaker (Furber et al., 2013) which are better suited for parallel processing. However, the wide-spread availability of general-purpose computers has drifted the SNN algorithmic development effort toward using hardware architectures better suited for sequential processing (Neumann, 1958). These general-purpose hardware architectures designed for sequential processing (also for parallel processing in the case of GPUs) do require tailor-made (customized) solutions that allow highly parallelizable SNN algorithms to run efficiently.

Two main groups of techniques are traditionally used for simulating the neural units (neurons) of SNNs within general-purpose computers: event-driven and time-driven techniques (Brette et al., 2007). Whilst the first technique only computes the neural dynamics of a neuron when it is affected by a spiking-event (generation and propagation of neural activity), the second one iteratively updates the neural dynamics of all neurons in each simulation step time. Both groups have pros and cons (Brette et al., 2007) and the best choice depends on the SNN inner features. In this study, we have focused our efforts on developing tailor-made event-driven and time-driven solutions to overcome the architectural and processing computational problems derived from using a general-purpose computer for simulating SNNs. We have studied how the mathematical complexity of several neural models may affect the simulation accuracy and computational performance when different simulation techniques are used over a standard SNN configuration.

METHODS

In this section we further explain the mechanisms that allow us to study the relationship amongst the neural dynamic complexity, simulation accuracy, and computational performance in SNNs. The benchmark analysis of well-established neural models helps to better understand this relationship. Three well-known neural models are chosen, based on their mathematical complexity and biological plausibility (see Appendix A for further details):

(a) The leaky integrate-and-fire (LIF) (Gerstner and Kistler, 2002) model. It is composed of one differential equation and two exponential decay functions for both excitatory and inhibitory conductances. It is extremely efficient in computational terms; however it cannot account for a wide range of biological properties.
(b) The adaptive exponential integrate-and-fire (AdEx) (Brette and Gerstner, 2005) model. It is composed of two differential equations and two exponential decay functions for both excitatory and inhibitory conductances. This model is only slightly more complex than the LIF from a computational point of view; however it can be considered more biologically plausible since it is able to reproduce a wide range of firing regimes (bursting, short-term adaptation, etc.).
(c) The Hodgkin-Huxley (HH) (Hodgkin and Huxley, 1952) model. It is composed of four differential equations and two exponential decay functions for both excitatory and inhibitory conductances. Its neural dynamics requires more computational resources; however, its differential equations closely match the neural processes that govern the spike generation. This biophysical model reproduces rather realistic physiological properties (considering ion channel activation and deactivation features).

To run the benchmark analysis, we use the spiking neural network simulator EDLUT (Ros et al., 2006a) as the working framework. EDLUT is an efficient open source simulator mainly oriented to real time simulations that allows the processing of specific parts of the neural network using different neural dynamic evaluation techniques. To adopt an extensively used benchmark methodology, we follow the recommendations given by Brette et al. (2007) to evaluate the performance of the neural network simulation techniques (different neural network sizes, connectivity ratios, firing rates, and neural models). By means of this benchmark, we specifically evaluate how the mathematical model complexity of neurons affects the computational performance and simulation accuracy when different simulation techniques are used. The synthetic nature of the benchmark here proposed is based on previous benchmark studies (Brette et al., 2007). This benchmark emulates those neural networks which are composed of neurons with a medium to low number of input synapses (up to 1,280 input synapses per neuron). The simulation performance results may change significantly when simulating biologically realistic experiments (e.g., cortical networks) which require a much larger number of incoming synapses (up to 10,000 synapses per neuron) and firing rates (van Albada et al., 2015).

The source code is available for readers at URL: www.ugr.es/~nluque/restringido/Event_and_Time_Driven_Techniques.rar (user: REVIEWER, password: REVIEWER). All the additional material needed for the benchmark analyses (neural network, synaptic weight, input activity and neuron model description files, as well as the scripts to compile the look-up tables) are also located at the same URL.

Neural Dynamic Evaluation Techniques-Why and for What?

EDLUT implements two different neural dynamic evaluation techniques: (a) event-driven neural techniques based on pre-computed look-up tables for CPU platforms (Ros et al., 2006a), and (b) time-driven neural techniques for both CPU (Garrido et al., 2011) and GPU (Naveros et al., 2015) platforms. EDLUT allows the combination of both neural techniques on the same simulation.

Event-driven techniques are better suited for neural network layers with low and sparse activity whose network units (neurons) present low mathematical complexity. Pre-compiling and allocating the dynamic evolution of a neural model within look-up tables allows the updating of its neural state discontinuously, i.e., at different time intervals. Thus, the neural-state update process during simulation becomes very efficient, requiring only a few accesses to these look-up tables. This technique can be applied to a wide variety of neurons of diverse mathematical complexity. The bottleneck of this computational scheme lies in two factors: (a) the dimensionality of the look-up tables, and (b) the number of look-up table readouts (the number of input and output spikes that are to be processed). The higher the neural mathematical complexity is, the higher the number of state variables, and therefore, the higher the dimensionality and the number of look-up tables needed. Higher numbers of look-up tables involve time-consuming data queries. Larger look-up tables also involve slower look-up table readouts. The number of readouts, in turn, depends on the number of events to be processed (input propagated spikes and output internal spikes; Ros et al., 2006a).

On the other hand, the time-driven neural technique outperforms the event-driven neural technique for neural network layers that present high interconnectivity ratios, high neural activities and high levels of neural mathematical complexity. This technique takes full advantage of parallel computing resources at CPU and GPU platforms. CPU time-driven techniques perform better for small and medium-size groups of neurons with a low-medium mathematical complexity (from one neuron to several thousands of neurons, depending on the mathematical complexity), whereas GPU time-driven techniques perform better for large-size groups of neurons with high mathematical complexity (from thousands to millions of neurons; Naveros et al., 2015).

When the neural network layers present high heterogeneity, both simulation techniques should be used concurrently. One example of a heterogeneous neural network can be found in the cerebellum, where the large granular layer with low and sparse activity (Luque et al., 2011a; Garrido et al., 2013b, 2016) is combined with other smaller layers dealing with higher activity rates (i.e., large-convergence neurons, such as Purkinje cells Luque et al., 2016).

Event-Driven Neuron Models

The implementation of event-driven neuron models has previously been stated in Mattia and Del Giudice (2000), Delorme and Thorpe (2003), Reutimann et al. (2003), Rudolph and Destexhe (2006), and Pecevski et al. (2014). Particularly, the neural simulator EDLUT implements an event-driven neural technique based on look-up tables. See Ros et al. (2006a) for a comprehensive description on EDLUT event-driven simulation techniques. Compared to previous studies (Naveros et al., 2015), in this paper we propose two independent contributions over EDLUT event-driven simulation techniques. The first contribution increases the performance by compacting the look-up table structure and improving the look-up table indexing. The second one increases the performance by improving the processing of synchronous activity. With the integration of these two new simulation techniques with the original one we can simulate each neuron model using four different configurations for event-driven neuron models: direct (the original one), combined, synchronous and combined synchronous event-driven neuron models. Below, we summarize the properties of these two new simulation techniques.

Combined Look-Up Tables for Complex Neuron Models

EDLUT pre-compiles the solution of each neural model equation into look-up tables (a look-up table per state variable). EDLUT inherently requires up to two additional state variables. The first additional state variable stores the timing of a predicted spike-firing event, whereas the second state variable stores its ending (in some cases both variables remain equal). Each look-up table dimension is indexed by a neural state variable. EDLUT neural simulation uses look-up table data queries to update the neural state variables.

The higher the mathematical complexity the neural model is, the more state variables that are needed, and the more look-up tables that are then required. Concomitantly, the dimensionality of the look-up tables also increases with the number of coupled state variables. The dimensionality and number of look-up tables are, therefore, imposed by the neural model complexity. The only way to control the look-up table size is by adjusting the look-up table granularity (the number of coordinates per dimension). Obviously, the degree of granularity has a direct impact on the accuracy and performance of the neural simulation.

Computing data queries using large look-up tables constitutes the most time consuming operation of all the neural dynamic evaluations. Therefore, reducing the number of look-up table readouts needed per neural model would reduce the neural dynamic evaluation time, thus increasing the overall simulation performance. Aiming to reduce the number of look-up tables, we have created a new event-driven method that recombines the look-up tables that share index values. Thus, we are able to reduce the number of look-up tables and make them more compact than the original ones (considering all of them as a whole). See **Figure 1**. The combined look-up tables are described as follows (see Appendix A for further details about neural model descriptions):

Leaky Integrated-and-Fire Model (LIF)

- One look-up table with four dimensions storing the forecast values of the membrane potential: $V = f(\Delta t, g_{AMPA}, g_{GABA}, V)$. The neural state variables associated to each dimension are the elapsed time since the last update (Δt), the previous excitatory (g_{AMPA}) and inhibitory (g_{GABA}) conductances and the previous membrane potential (V).

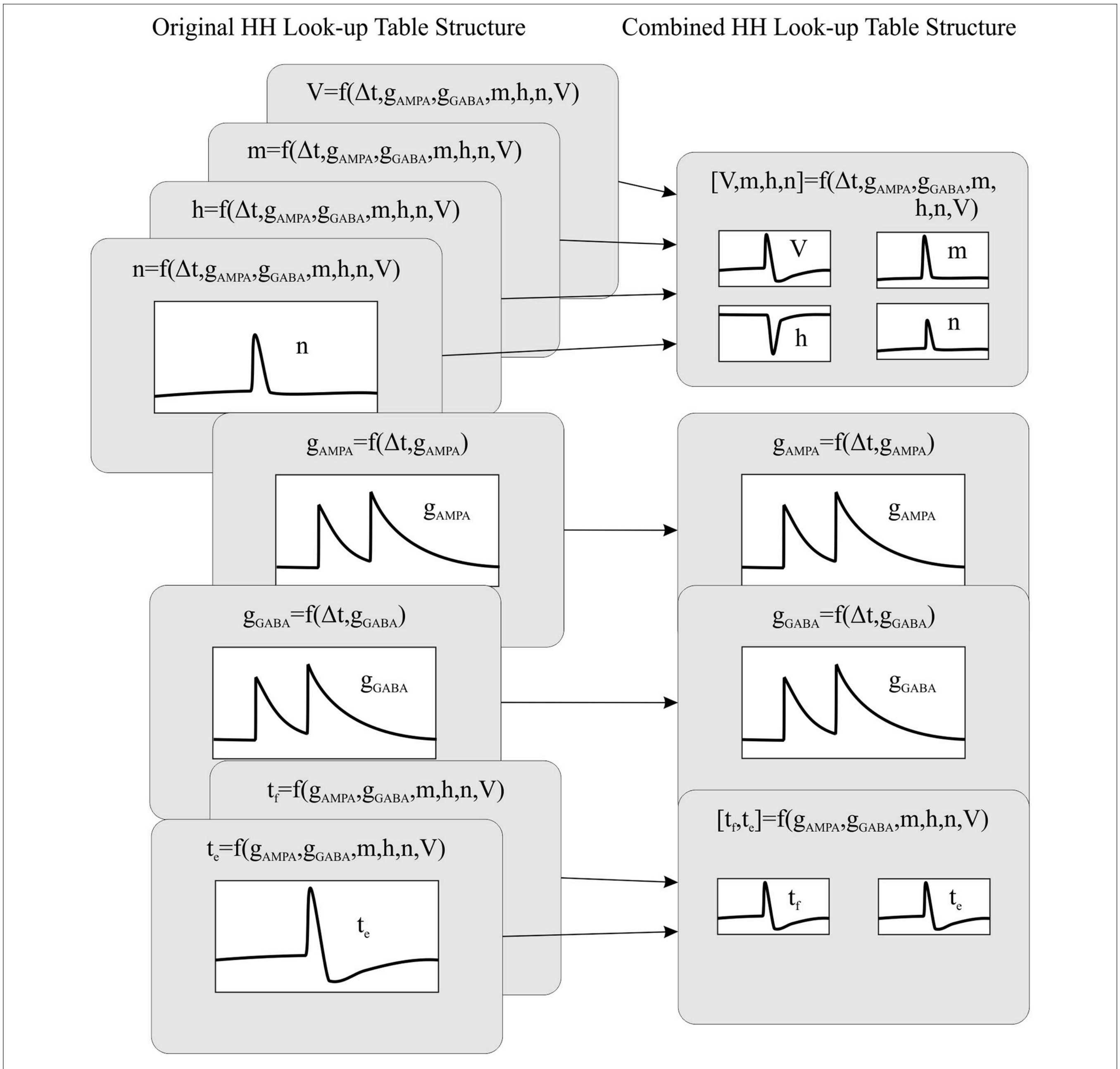

Original HH Look-up Table Structure
Combined HH Look-up Table Structure
V=f(Δt,g_AMPA,g_GABA,m,h,n,V)
m=f(Δt,g_AMPA,g_GABA,m,h,n,V)
h=f(Δt,g_AMPA,g_GABA,m,h,n,V)
n=f(Δt,g_AMPA,g_GABA,m,h,n,V)
n
[V,m,h,n]=f(Δt,g_AMPA,g_GABA,m,h,n,V)
V
m
h
n
g_AMPA=f(Δt,g_AMPA)
g_AMPA
g_GABA=f(Δt,g_GABA)
g_GABA
t_f=f(g_AMPA,g_GABA,m,h,n,V)
t_e=f(g_AMPA,g_GABA,m,h,n,V)
t_e
[t_f,t_e]=f(g_AMPA,g_GABA,m,h,n,V)
t_f
t_e

FIGURE 1 | The recombination mechanism of look-up tables for a HH model. The left side of the panel shows the original look-up table structure (eight tables) whilst the right side of the panel shows the recombined look-up table structure (four tables).

- Two look-up tables of two dimensions storing the forecast values of the excitatory and inhibitory conductances: $g_{AMPA} = f(\Delta t, g_{AMPA})$ and $g_{GABA} = f(\Delta t, g_{GABA})$. The neural state variables associated to each dimension are the elapsed time since the last update (Δt) and the previous excitatory (g_{AMPA}) or inhibitory (g_{GABA}) conductances.
- One look-up table of three dimensions storing the forecast values about the time of the next firing event and the ending of the refractory period $(t_f, t_e) = f(g_{AMPA}, g_{GABA}, V)$. The neural state variables associated to each dimension are the current excitatory (g_{AMPA}) and inhibitory (g_{GABA}) conductances and the current membrane potential (V). Although the LIF model presents a constant refractory period (and could be easily implemented ad-hoc), we use the look-up table t_e, which stores the evolution of the refractory period, to maintain the same event-driven simulation structure for all the neural models (LIF, AdEx and HH).

Adaptive Exponential Integrate-and-Fire Model (AdEx)

- One look-up table of five dimensions storing the forecast values of the membrane potential and membrane adaptation variable: $[V, w] = f(\Delta t, g_{AMPA}, g_{GABA}, w, V)$. The neural state variables associated to each dimension are the elapsed time since the last update (Δt), the previous excitatory (g_{AMPA}) and inhibitory (g_{GABA}) conductances, the previous membrane adaptation variable (w) and the previous membrane potential (V).
- Two look-up tables of two dimensions storing the forecast values of the excitatory and inhibitory conductances: $g_{AMPA} = f(\Delta t, g_{AMPA})$ and $g_{GABA} = f(\Delta t, g_{GABA})$. The neural state variables associated to each dimension are the elapsed time since the last update (Δt) and the previous excitatory (g_{AMPA}) or inhibitory (g_{GABA}) conductances.
- One table of four dimensions for storing the forecast values about the time of the next firing event: $t_f = f(g_{AMPA}, g_{GABA}, w, V)$. The neural state variables associated to each dimension are the current excitatory (g_{AMPA}) and inhibitory (g_{GABA}) conductances, the current membrane adaptation variable (w) and the current membrane potential (V). Just one additional table is needed since this model does not include a refractory period.

Hodgkin-Huxley Model (HH)

- One look-up table of seven dimensions storing the forecast values of the membrane potential and the three ionic current activation variables: $[V, m, h, n] = f(\Delta t, g_{AMPA}, g_{GABA}, m, h, n, V)$. The neural state variables associated to each dimension are the elapsed time since the last update (Δt), the previous excitatory (g_{AMPA}) and inhibitory (g_{GABA}) conductances, the previous ionic current activation values *(m, h, and n)* and finally the previous membrane potential (V).
- Two look-up tables of two dimensions storing the forecast values of the excitatory and inhibitory conductances: $g_{AMPA} = f(\Delta t, g_{\mathrm{AMPA}})$ and $g_{\mathrm{GABA}} = f(\Delta t, g_{\mathrm{GABA}})$. The neural state variables associated to each dimension are the elapsed time since the last update (Δt) and the previous excitatory (g_{AMPA}) or inhibitory (g_{GABA}) conductances.
- One look-up table of six dimensions storing the forecast values about the time of the next firing event and the start of the hyperpolarization phase: $[t_f, t_e] = f(g_{AMPA}, g_{GABA}, m, h, n, V)$. The neural state variables associated to each dimension are the current excitatory (g_{AMPA}) and inhibitory (g_{GABA}) conductances, the current ionic current activation values (*m*, *h*, and *n*) and finally the current membrane potential (V). The look-up table t_e prevents EDLUT from duplicating internal spikes during the HH depolarization phase.

The combination of look-up tables minimizes the overall number of look-up tables for complex neuron models, since this combination allows one look-up table to store several state variables. This also means that a single look-up table readout can now update several state variables at a time. Thus, we are able to increase the computational performance of complex neuron models without modifying their accuracy.

Synchronous Event-Driven Neuron Models

Each time that an event-driven neuron is affected by an event, (input propagated spikes or output internal spikes) its neural state ought to be updated. After this update, EDLUT must predict whether the new neural state will make the neuron emit a spike in subsequent time steps (Ros et al., 2006a). EDLUT implements a two-stage mechanism able to handle the generation and propagation of the spikes. When EDLUT predicts a spike firing at any time, an internal-spike event is then created and inserted in the event queue. If another event modifies the spike-firing prediction, the internal-spike event is discarded; otherwise, the spike is eventually generated in the neural soma. A propagated-spike event is then generated and inserted in the event queue. This propagated-spike event is responsible for delivering the generated spike through all the neural output synapses. It holds a time stamp equivalent to the timing of its corresponding internal-spike event plus the propagation delay. When a neuron possesses several synapses with different propagation delays, the propagated-spike mechanism generates sequential propagated-spike events depending on the propagation delay values. The synaptic propagation delays are always fixed at multiples of 0.1 ms. If not, EDLUT rounds the delay within the network file to the nearest multiple of 0.1 ms.

To sum up, EDLUT triggers a three-step process in each neuron that receives a spike through a propagated-spike event (the most common event):

(a) When a spike arrives to an event-driven neuron, its neural state variables are updated.
(b) The spike increments the neural conductance associated with the synapse that propagates the spike.
(c) A prediction about the generation of a spike is made. If this prediction is positive, an internal spike event is inserted in the event queue.

This three-step process presents performance losses when the neural input activity is synchronous. When n spikes reach a neuron at the same time, the first n-1 predictions (and its correspondent internal spikes) would be discarded and only the *n*th taken into account. Knowing beforehand the number of synchronous spike arrivals per time step and per neuron would allow us to compute only one prediction per neuron, the *n*th prediction.

We have developed a new synchronous event-driven method able to efficiently compute synchronous neural input activity and able to generate synchronous output activity. When a group of synchronous spikes arrives to a neuron (being simulated within a synchronous event-driven method) the neural state variables are updated conjointly and a single internal spike prediction is done. This process is done in three steps:

(a) When the first spike of a group of synchronous spikes arrives to a synchronous event-driven neuron, its neural state variables are updated. Thus, these neural state variables will be already updated for the rest of the synchronous spikes.
(b) Each synchronous spike increments the neural conductance associated with the synapse that propagates the spike.
(c) Once EDLUT verifies that all the synchronous spikes have been processed thanks to an additional event, only one prediction about the generation of an output spike is made. If this prediction is positive, just one internal spike event is inserted in the event queue.

Thus, we only make one neural state update and one activity prediction for each group of synchronous spikes. Obviously, the additional event that helps to verify the processing of all the synchronous input spikes may cause a performance loss if the neural input activity is asynchronous (incoming activity is not grouped into tight time slots).

This new synchronous event-driven technique can also synchronize the neural spike propagation, thus allowing the efficient interconnection amongst synchronous event-driven neurons. This technique uses a parameter named synchronization period (t_{sync}) that is defined in the description file of each event-driven neural model. The synchronization period value is fixed and equal or greater than zero. Each internal-spike event can be processed at any time step; however, its corresponding propagated-spike events are generated as if the internal-spike were processed at multiples of t_{sync}. When t_{sync} is zero, the output activity is asynchronous and the neural network units (neurons) behave as direct event-driven neuron models. If t_{sync} is greater than zero, the output activity is then synchronous and the neural network units (neurons) can be interconnected to other synchronous event-driven models, thus increasing the overall performance but at the expense of accuracy. These synchronous models efficiently compute input activity coming from either time-driven or synchronous event-driven neuron models. Conversely, when the input activity comes from other types of event-driven neuron models the computational performance drops.

These kinds of synchronous neural layers can typically be found in neural networks that process sensory information, such as the olfactory (Schoppa, 2006) (30–70 Hz), auditory (Doesburg et al., 2012) (30–50 Hz), or visual (Eckhorn et al., 1990) (35–80 Hz) systems.

Time-Driven Neuron Models

EDLUT implements time-driven neuron models for both CPU (Garrido et al., 2011) and GPU (Naveros et al., 2015) platforms. These models are defined by a set of differential and non-differential equations that have to be computed during the simulation time. These equations must be solved using differential equation solvers given within a certain integration method. There are mainly two families of integration methods regarding their integration step size: a) fixed-step integration methods, and b) variable-step integration methods (Iserles, 2009).

Fixed-Step Integration Methods

Fixed-step integration methods are suited for parallelization in both CPU and GPU platforms (Naveros et al., 2015) since these methods favor synchronicity during the integration process. A single integration event manages the integration process of a large number of neurons (just one event for each integration step must be generated, inserted in the event queue, extracted from the event queue, processed and deleted). Thus, the computation overhead caused by non-directly related tasks to the integration processes remains low. However, the maximum fixed-step size that can be used is constrained by the stiffness of the differential equations that define each neural model. This constraint makes fixed-step integration methods to not be well suited for solving complex neural models whose differential equations are rather stiff.

Variable-Step Integration Methods

Variable-step integration methods iteratively adapt their integration step size to the neural dynamics. They are iteratively maintaining a balance between the simulation step size and the accuracy as the integration process deploys. This adaptation mechanism makes variable-step integration methods better suited for solving complex neural models whose differential equations are rather stiff. However, this flexibility comes at a cost:

- Their parallelization in CPU platforms is arduous and almost intractable in GPU platforms.
- The computation overhead caused by the estimation of the integration step size can be high.
- The integration process of each neuron has to be managed individually (one event per neuron). For each integration step an event must be generated, stored in the event queue, extracted from the event queue, processed and deleted. This overhead is determinant when the number of neurons is relatively high (thousands of neurons) and the activity is also high.
- The performance of these methods is heavily related with the neural activity. A high network activity increases the firing ratios of the neural network units (neurons). In this case, the solutions for the neural differential equations are mostly located around the neural firing working points. To maintain accuracy around a firing working point, the integration step size needs to be reduced. The computation time per neural unit is then increased.

The computational overhead caused by all these drawbacks makes these methods unadvisable for efficient simulations when the number of neural units (neurons) is relatively high. For this reason, we do not implement variable-step integration methods in EDLUT.

A New Integration Method; The Bi-Fixed-Step Integration Method

This new integration method tries to take advantage of the strengths and mitigate the weaknesses of fixed-step and variable-step integration methods. This integration method uses two fixed-steps for each neuron: a global fixed-step size (T_g) and local fixed-step size (T_l). T_g is multiple of T_l ($M_{gl} = T_g/T_l$). T_g synchronizes the integration processes of all network units (neurons) that are defined by the same neural model. This allows us to manage the integration process of a group of similar neurons using just a single integration event, as the fixed-step integration methods do. On the other hand, T_l can scale down the integration step size of one neuron when needed. When T_l is used instead of T_g, the integration method performs M_{gl} consecutive integrations within T_g. This allows us to adapt the integration step size to the dynamic evolution of each neuron, as the variable-step integration methods do. **Figure 2** shows how the implementation of this integration method within CPU and GPU platforms differs in order to cope with their different hardware properties.

When EDLUT runs a simulation, the generated "events" are sorted depending on their time stamps in an event queue. When a new event is processed, its corresponding time stamp establishes the new "simulation time." A new bi-fixed integration event produces a simulation time updating which is multiple of the global time step. The spike generation process cannot be triggered at any local time step but at the global step time, otherwise the generated spikes would carry incoherent time stamp values (lower than the actual simulation time). Therefore, the spikes to be generated are detected at local time steps but only generated at global time steps.

Bi-fixed-step integration method for differential equation solvers in CPU

Two additional elements per neuron model are defined: a hysteresis cycle given by two membrane potential thresholds (V_s as the upper bound and V_e as the lower bound) and a period of time T_p for the hyperpolarization phase in neural models, such as the HH. These parameters drive the switching of the integration step size from global to local and vice versa.

This method starts the integration process of a certain group of neurons using the global integration step size T_g for each neuron. The membrane potential of each neuron is then compared with the threshold value V_s after each global integration step. When the resulting membrane potential is $>V_s$, the integration result is discarded and the integration step size is scaled down to the local step size T_l just for that neuron. The integration process is reinitialized using the new local step size. This local step size is maintained until the membrane potential decays to V_e and a certain period of time T_p has passed. Once this double condition is filled, the integration step size is re-scaled up to the global step size T_g (see **Figure 2A** for a complete workflow diagram). **Figure 3** shows an example of this adaptation mechanism over the LIF, AdEx, and HH models.

The state variables, in most neuron models, usually present a slow evolution during simulation time (very low velocity gradient). It is at the spike phase when these state variables evolve faster (very high velocity gradient). By using a V_s threshold lower than the actual "functional" spike threshold we are able to predict a spike phase beforehand. The bi-fixed-step integration method uses this prediction to reduce the integration step size before an eventual spike phase. T_g extra period sets the hyperpolarization time after the spike generation. During T_g, the state variables present very high velocity gradients and, therefore, reduced integration step sizes are to be maintained (e.g., T_g can be used to properly integrate the hyperpolarization phase of HH models after the depolarization phase).

This method is easily parallelizable in CPU and can be managed with just one integration event, as in fixed-step integration methods. Additionally, it can outperform the simulation of complex neuron model with stiff equations thanks to its adaptation mechanism, as in variable-step integration methods.

Bi-fixed-step integration method for differential equation solvers in GPU

GPU hardware requires all the simulated neurons of a neural model to run exactly the same code at the same time. Additionally, these neurons must also access the RAM memory following a concurrent scheme. As reported in Naveros et al. (2015), the synchronization period and transference of data between CPU and GPU processors account for most of the performance losses in a hybrid CPU-GPU neural simulation. To minimize these losses, CPU and GPU processors are synchronized at each T_g global integration step time. Then the GPU integrates all its neurons using the local integration step T_l. After the integration process, the GPU reports to the CPU which neurons fired a spike (see **Figure 2B** for a complete workflow diagram).

This method is easily parallelizable in GPU and can be managed with just one integration event, as in fixed-step integration methods. Additionally, a short local step (T_l) can accurately compute the simulation of complex neuron model with stiff equations whereas a large global step (T_g) reduces the number of synchronizations and data transferences between CPU and GPU processors and increases the overall performance. This bi-fixed-step integration method is suited to comply with hybrid CPU-GPU platforms since it maximizes the GPU workload and minimizes the communication between both processors.

To sum up, EDLUT can now operate with time-driven neuron models that can use different fixed-step or bi-fixed-step integration methods for both the CPU and GPU platforms. The following differential equation solvers have been implemented using fixed-step integration methods: Euler, 2nd and 4th order Runge-Kutta, and 1st to 6th order Backward Differentiation Formula (BDF). The following differential equation solvers have been implemented using bi-fixed-step integration methods:

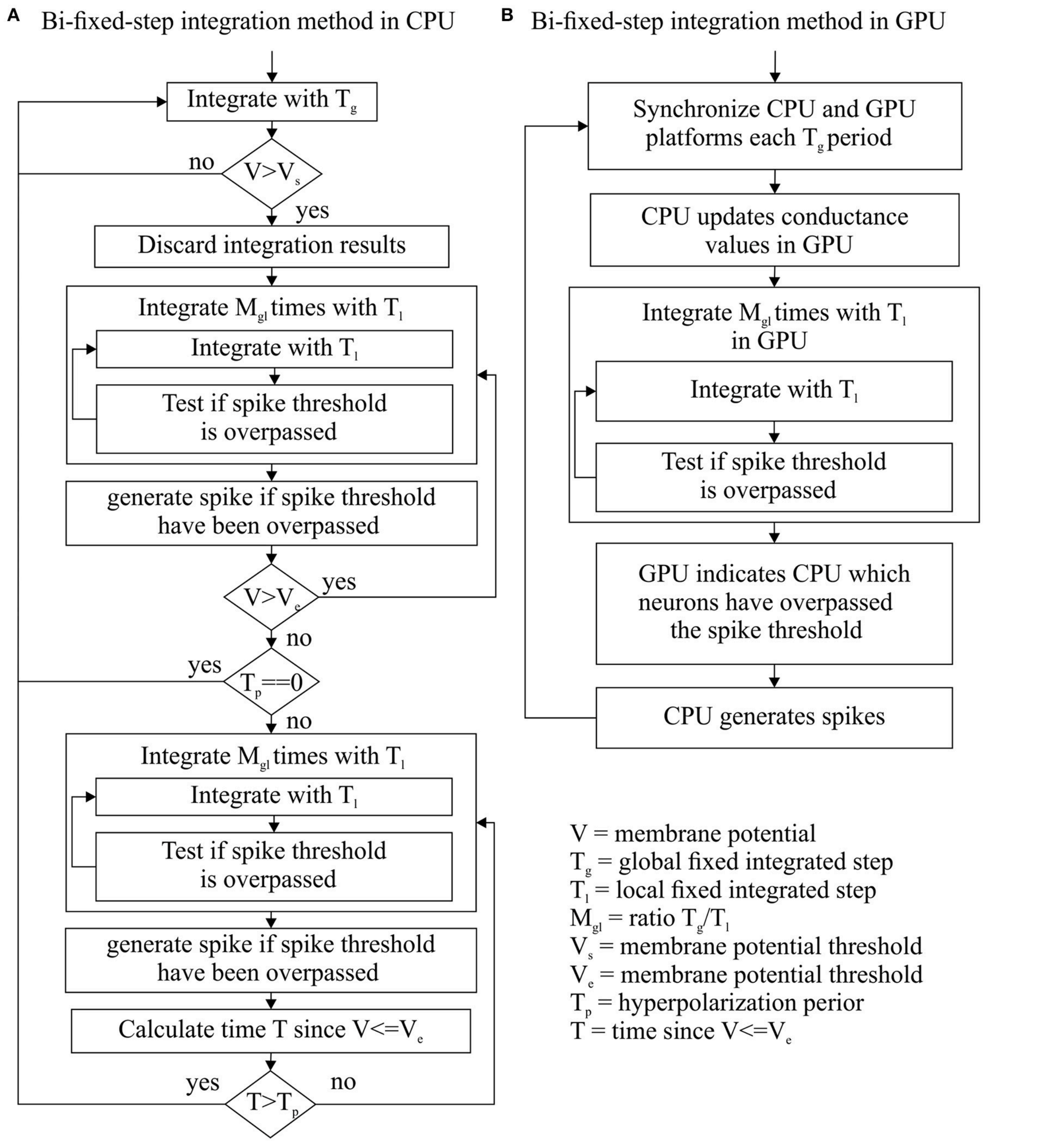

A Bi-fixed-step integration method in CPU
Integrate with T_g
no
$V>V_s$
yes
Discard integration results
Integrate M_{gl} times with T_l
Integrate with T_l
Test if spike threshold is overpassed
generate spike if spike threshold have been overpassed
$V>V_e$
yes
no
yes
$T_p==0$
no
Integrate M_{gl} times with T_l
Integrate with T_l
Test if spike threshold is overpassed
generate spike if spike threshold have been overpassed
Calculate time T since $V<=V_e$
yes
$T>T_p$
no
B Bi-fixed-step integration method in GPU
Synchronize CPU and GPU platforms each T_g period
CPU updates conductance values in GPU
Integrate M_{gl} times with T_l in GPU
Integrate with T_l
Test if spike threshold is overpassed
GPU indicates CPU which neurons have overpassed the spike threshold
CPU generates spikes
V = membrane potential
T_g = global fixed integrated step
T_l = local fixed integrated step
M_{gl} = ratio T_g/T_l
V_s = membrane potential threshold
V_e = membrane potential threshold
T_p = hyperpolarization perior
T = time since $V<=V_e$

FIGURE 2 | Bi-fixed-step integration method flow diagram for CPU (A) and GPU **(B)** platforms. Since T_l is divisor of T_g, these integration methods can integrate a period of time T_g making M_{gl} consecutive integrations with a step-sizes of T_l.

Euler, 2nd and 4th order Runge-Kutta, and 2nd order Backward Differentiation Formula (BDF). This last differential equation solver implements a fixed-leading coefficient technique (Skeel, 1986) to handle the variation of the integration step size.

In this paper, we have only evaluated the simulation accuracy and computational performance of 4th order Runge-Kutta solvers in both fixed-step and bi-fixed-step integration methods in both CPU and GPU platforms. **Table 1** shows the integration parameters used by each neuron model for each integration method.

Test-Bed Experiments

We have adapted the benchmark proposed by Brette et al. (2007) as our initial neural network setup for our experiments. The

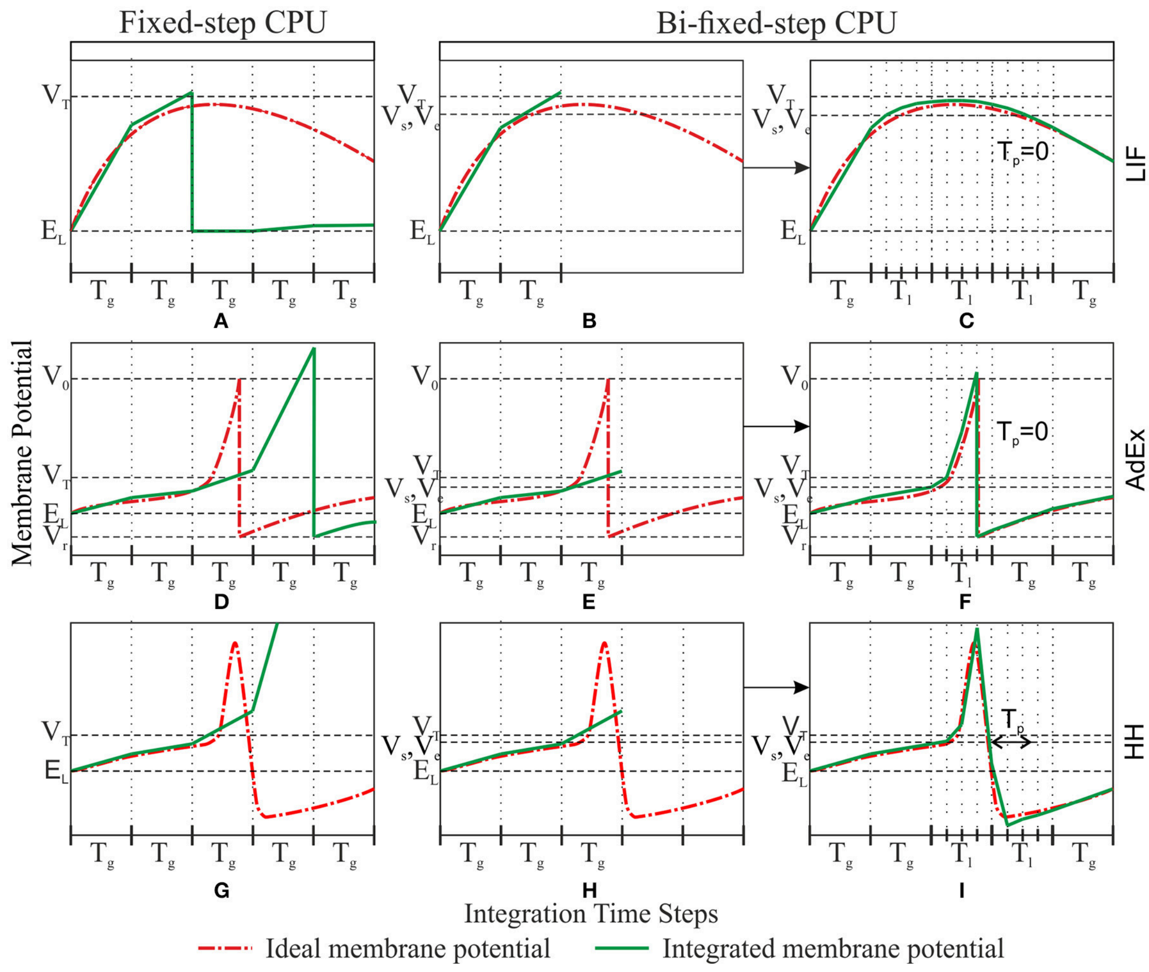


FIGURE 3 | Comparison between fixed-step and bi-fixed-step integration methods in CPU for LIF, AdEx, and HH models. Each row shows the ideal membrane potential and the integrated membrane potential for a LIF, AdEx, and HH model, respectively. The left-hand column **(A,D,** and **G)** shows the fixed-step integration results. The central and right-hand columns show the bi-fixed-step integration results. The central column **(B,E,** and **H)** shows the moment when the membrane potential overpasses the threshold V_s. The last integrated result is then discarded and the integration step size is scaled down to T_l. The right-hand column **(C,F** and **I)** shows the moment when the membrane potential underpasses the threshold V_e, a time $> T_p$ has elapsed and the integration step size is then scaled up to T_g.

initial setup consists of 5000 neurons divided into two layers. The first layer (input layer) holds 1000 excitatory neurons and it just conveys the input activity to the second layer. The second layer consists of 4000 neurons where 80% are excitatory neurons and the remaining 20% are inhibitory neurons. The neurons at this 2nd layer are modeled as LIF, AdEx or HH models.

Each second-layer neuron is the target of 10 randomly chosen neurons from the first layer. Each second-layer neuron is also the target of eighty randomly chosen neurons from the same layer, following a recurrent topology. All these synapses include a 0.1 ms propagation delay. This neural network topology is summarized in **Table 2**.

The input activity supplied to each input neuron is randomly generated using a Poisson process with exponential inter-spike-interval distribution and mean frequency of 5 Hz. This input activity generates a mean firing rate activity of 10 Hz in the second layer.

We measured the simulation accuracy and computational performance of the aforementioned integration methods over three different neural models (LIF, AdEx, and HH). These neural models are simulated using two different dynamic evaluation techniques: event-driven and time-driven techniques.

Within event-driven dynamic evaluation techniques, four different event-driven integration methods are applied:

(a) Direct event-driven integration methods.
(b) Combined event-driven integration methods.
(c) Synchronous event-driven integration methods.
(d) Combined synchronous event-driven integration methods.

Within time-driven dynamic evaluation techniques, two different integration methods implementing a 4th order Runge-Kutta differential equation solver are applied in both CPU and GPU platforms:

TABLE 1 | Summary of parameters for event-driven and time-driven simulation techniques for LIF, AdEx, and HH models.

	LIF	AdEx	HH
Synchronization period (ms)	1.0	1.0	1.0
Fixed-step size (ms)	0.5	0.5	1.0/15.0
Global fixed-step size (ms)	1.0	1.0	1.0
Local fixed-step size (ms)	0.25	0.25	1.0/15.0
Threshold V_S (mV)	−53.0	−50.0	−57.0
Threshold V_e (mV)	−53.0	−50.0	−57.0
Period T_p (ms)	0.0	0.0	1.0

(a) Fixed-step integration methods.
(b) Bi-fixed-step integration methods.

To properly compare all these integration methods, we studied the simulation accuracy that each can offer. We used the van Rossum distance (Van Rossum, 2001) with a tau of 1 ms (maximum size of integration step periods for event-driven methods and synchronization periods for even-driven methods) as a metric of accuracy. We use this metric to compare a reference activity file and a tested activity file (both files containing the spike time stamps associated to all the spikes emitted by the neural network units). The reference activity files are obtained using a time-driven simulation technique running in CPU with a fixed-step integration method using a 4th order Runge-Kutta solver and a fixed 1 μs integration step size for each neural model (LIF, AdEx, and HH). The tested activity files are obtained using the mentioned integration methods of the two different dynamic evaluation techniques (event-driven and time-driven) for each neural model (LIF, AdEx, and HH).

Additionally, we wanted to study the computational performance of each integration method. We measured the execution time spent by each integration technique over a set of four different experiments that simulate 1 s of neural activity. These four experiments characterize the computational performance of the mentioned integration methods.

The hardware running these Benchmark analyses consists of an ASUS Z87 DELUXE mother board, an Intel Core i7-4,770k processor, 32 GB of DDR3 1,333 MHz RAM memory, and a NVIDIA GeForce GTX TITAN graphic processor unit with 6144 MB RAM memory GDDR5 and 2,688 CUDA cores. The compilers used are those that are integrated in visual studio 2008 together with CUDA 6.0.

All the experiments are CPU parallelized by using two OpenMP threads as described in a previous paper (Naveros et al., 2015). These threads parallelize the spike generation and propagation for the event-driven and time-driven models. The neural dynamic computation of the event-driven and time-driven models in CPU is also parallelized by using the two OpenMP threads. The neural dynamic computation of the time-driven models in GPU is parallelized by using all the GPU cores.

Simulation Parameter Analyses

We quantified the effects of using different simulation techniques on the simulation accuracy and the computational performance. In particular, we measured the impact of scaling the look-up table size and the synchronization time-period for event-driven techniques. For time-driven techniques, we measured the impact of scaling the integration time-step sizes.

For these analyses, our initial neural network setup is modified as defined in **Table 3**. A third neural layer replicating the second layer properties is inserted and the recurrent topology is modified. The 2nd and the 3rd layer are now interconnected by those synapses from the 2nd layer that were previously modeling the recurrent topology of our initial setup. This initial recurrent topology rapidly propagated and increased small errors through the recurrent synapses. Under these circumstances, accuracy cannot be properly measured. Adding this 3rd layer allowed us to circumvent this problem and better evaluate the accuracy degradation in a well-defined experiment.

We stimulated this new setup with five different input patterns generated using a Poisson process with exponential inter-spike-interval distribution and mean frequency of 5 Hz. We measured the simulation accuracy of the 3rd neural layer by applying the van Rossum distances as previously explained. Thus, we were able to evaluate the effect of the synchronization period over accuracy when several layers of synchronous event-driven models are interconnected. Similarly, we also evaluated the effect of the integration step size over accuracy when several layers of time-driven models are interconnected. The computational performance is given by the mean execution time that the new setup spends in computing 1 s of simulation when it is stimulated with the five different input activity patterns.

Scalability Analyses

We quantified the effects of scaling up the number of neurons within our initial setup over the computational performance. In particular, we measured the impact of scaling up the second layer size for the event-driven and time-driven techniques proposed.

For these analyses, our initial neural network setup is modified as defined in **Table 4**. Nine different variations over our initial setup are simulated. The 2nd layer is geometrically scaled up from 1,000 to 256,000 by a common ratio and scale factor of $r = 2$ and $a = 1{,}000$, respectively (number of neurons $= a{\cdot}r^n$, where $n \epsilon$ [0, 8]).

Input Activity Analyses

We quantified the effects of scaling up the input activity levels over the computational performance. In particular, we measured the impact of scaling up our neural network mean-firing rate through different input activity levels for the event-driven and time-driven techniques proposed.

For these analyses, our initial neural network setup is modified as defined in **Table 5**. Fifteen different input activity levels scaled up from 2 to 16 Hz stimulate our neural network. These input activity levels generate mean firing rates in the second neural layer of between 2 and 40 Hz.

Connectivity Analyses

We quantified the effects of scaling up the number of synapses over the computational performance. In particular, we measured

TABLE 2 | Summary of cells and synapses implemented.

Initial Network Configuration	Pre-synaptic cell (number)	Post-synaptic cell (number)	Number of synapses	Number of excitatory synapses	Number of inhibitory synapses
	Input layer (1,000) input neurons	2nd layer (4,000) LIF, AdEx, or HH neurons	40,000	40,000 (7 nS)	
	2nd layer (4,000) LIF, AdEx, or HH neurons	2nd layer (4,000) LIF, AdEx, or HH neurons	320,000	256,000 (0.5 nS)	64,000 (2.5 nS)

Initial values providing the reference framework for comparisons in subsequent experiments.

TABLE 3 | Summary of cells and synapses implemented for parameter analysis experiment (accuracy and performance).

Network Configuration	Pre-synaptic cell (number)	Post-synaptic cell (number)	Number of synapses	Number of excitatory synapses	Number of inhibitory synapses
	Input layer (1,000) input neurons	2nd layer (4,000) LIF, AdEx, or HH neurons	40,000	40,000 (7 nS)	
	Input layer (1,000) input neurons	3rd layer (4,000) LIF, AdEx, or HH neurons	40,000	40,000 (7 nS)	
	2nd layer (4,000) LIF, AdEx or HH neurons	3rd layer (4,000) LIF, AdEx, or HH neurons	320,000	256,000 (0.5 nS)	64,000 (2.5 nS)

TABLE 4 | Summary of cells and synapses implemented for neural network scalability experiment.

Network Configuration	Pre-synaptic cell (number)	Post-synaptic cell (number)	Number of synapses	Number of excitatory synapses	Number of inhibitory synapses
	Input layer (1,000) input neurons	2nd layer (from 1,000 to 256,000) LIF, AdEx, or HH neurons	From 10,000 to 256,0000	From 10,000 to 2,560,000 (7 nS)	
	2nd layer (from 1,000 to 256,000) LIF, AdEx, or HH neurons	2nd layer (from 1,000 to 256,000) LIF, AdEx, or HH neurons	From 80,000 to 20,480,000	From 64,000 to 16,384,000 (0.5 nS)	From 16,000 to 4,096,000 (2.5 nS)

TABLE 5 | Summary of cells and synapses implemented for neural network input activity scaling (input average firing rate) experiment.

Network Configuration	Pre-synaptic cell (number)	Post-synaptic cell (number)	Number of synapses	Number of excitatory synapses	Number of inhibitory synapses
	Input layer (1,000) input neurons	2nd layer (16,000) LIF, AdEx, or HH neurons	160,000	160,000 (7 nS)	
	2nd layer (16,000) LIF, AdEx, or HH neurons	2nd layer (16,000) LIF, AdEx, or HH neurons	1,280,000	1,024,000 (0.5 nS)	256,000 (2.5 nS)

TABLE 6 | Summary of cells and synapses implemented for neural network interconnection scalability experiment.

Network Configuration	Pre-synaptic cell (number)	Post-synaptic cell (number)	Number of synapses	Number of excitatory synapses	Number of inhibitory synapses
	Input layer (1000) input neurons	2nd layer (16,000) LIF, AdEx, or HH neurons	160,000	160,000 (7 nS)	
	2nd layer (16,000) LIF, AdEx, or HH neurons	2nd layer (16,000) LIF, AdEx, or HH neurons	From 160,000 to 20,480,000	From 128,000 to 16,384,000 (from 0.5 to 0.03125 nS)	From 32,000 to 4,096,000 (from 2.5 to 0.15625 nS)

the impact of increasing the number of recurrent synapses for the event-driven and time-driven techniques proposed.

For these analyses, our initial neural network setup is modified as defined in **Table 6**. The number of recurrent synapses are geometrically scaled up from 10 to 1,280 by a common ratio and scale factor of $r = 2$ and $a = 10$, respectively (number of recurrent synapses $= a{\cdot}r^n$, where $n \in [0, 7]$). Maintaining the mean firing rate in the second neural layer at approximately

10 Hz requires the recurrent synaptic weights to be adapted depending on the number of recurrent synapses. The initial neural network maintains 80 synapses as in the previous cases (the weights of excitatory and inhibitory synapses are set to 0.5 and 2.5 nS, respectively). Neural networks with a lower number of synapses [10, 40] require the synaptic weights to remain the same. Neural networks with larger number of synapses [160, 1280] require the synaptic weights to be divided by the common ratio $r = 2$ in each iteration (the weights of excitatory and inhibitory synapses are ranged from 0.25 to 0.03125 and from 1.25 to 0.15625, respectively).

RESULTS

This section shows the results obtained by the four test-bed experiments described in the methods section. Each experiment evaluated eight different neural dynamic simulation techniques over LIF, AdEx, and HH neuron models in terms of accuracy and/or performance (see Methods). Thus, we evaluated how the proposed simulation techniques perform with neural models of different mathematical complexity.

Simulation Parameter Results: The Look-Up Table, Synchronization Period and Integration Step Size Implications

In this experiment, we computed the neural network defined in **Table 3** using five different input spike patterns with a Poisson process and mean firing rate of 5 Hz. We measured the simulation accuracy over the third neural layer and the computational performance over the whole simulation time when the look-up table size, the synchronization period, or the integration step size are scaled up.

Look-Up Table Size Implications

As previously stated, the look-up table size directly affects the neural model simulation accuracy and computational performance for event-driven simulation techniques. This is of special importance for neural models with high mathematical complexity. The number of state variables in a model determines the number of look-up tables and their dimensions. Since the number of state variables is given by the neural model (LIF, AdEx, or HH), the granularity level of each look-up table dimension is the only independent parameter that can be freely selected to adjust the look-up table size. The more complex the model is, the lower granularity levels that are required to keep the look-up table size manageable (HH granularity level <AdEx granularity level <LIF granularity level). Consequently, the higher the complexity of the neural models is, the lower accuracy that is obtained when the total look-up table sizes are fixed.

Here, we have evaluated four pre-compiled look-up tables with different levels of granularity for each neural model. In subsequent experiments, the event-driven models will use the largest look-up tables to keep the highest possible level of accuracy.

Figure 4 shows the simulation accuracy and computational performance of direct and combined event-driven integration methods with respect to the four sets of look-up tables with different levels of granularity for each of our three neural models (LIF, AdEx, and HH). As shown, the larger the look-up table size is, the higher the accuracy (smaller van Rossum distances with respect to the reference simulation pattern; **Figure 4A**) but at the cost of a worse performance (higher execution times; **Figure 4B**). The simulation accuracy for both integration methods (direct and combined) remains the same since the look-up table recombination of the second integration method does not affect the neural dynamics. The simulation accuracy for one

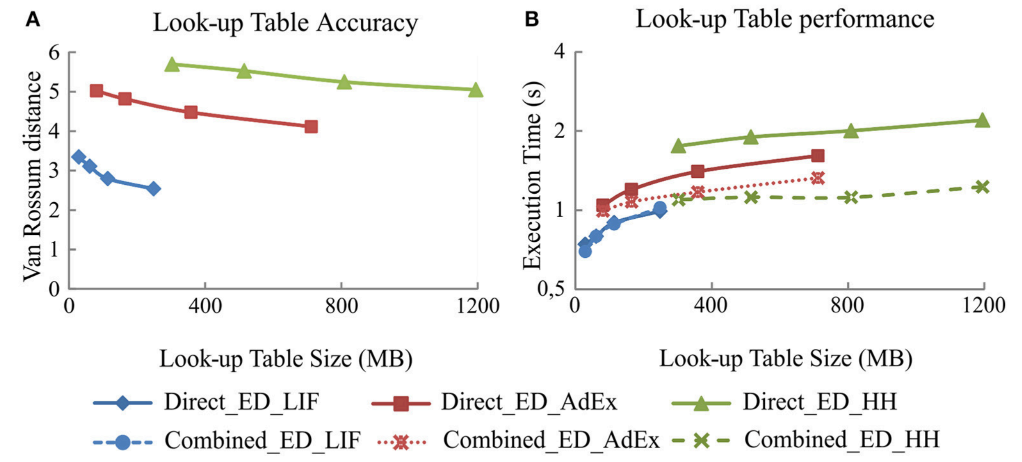


FIGURE 4 | Simulation accuracy and computational performance for direct and combined event-driven integration methods. (A) Mean simulation accuracy obtained with direct and combined event-driven integration methods depending on the look-up table sizes for five different input spike patterns. **(B)** Computation time spent by direct and combined event-driven integration methods in running 1 s of simulation over five different input spike patterns (averaged). Four different look-up table sizes for each neural model are used. The neural network defined in **Table 3** is simulated using both integration methods over LIF, AdEx, and HH models. The standard deviation of simulation accuracy and computational performance obtained is negligible; we only represent the mean values. Both integration methods present identical accuracy results.

of these integration methods is, therefore, representative for both in the plots of **Figure 4**.

Figure 4B also compares the computational performance of direct and combined event-driven integration methods. The more mathematically complex the neural model is, the more look-up tables can be combined and the higher the computational performance results of combined event-driven integration methods with respect to the direct ones.

Synchronization Period Size Implications

The synchronization period of synchronous and combined synchronous event-driven integration methods affects the neural model simulation accuracy and the computational performance. As in the previous case, the simulation accuracy of both integration methods remains the same since the look-up table recombination does not affect the neural dynamics. The simulation accuracy for one of these integration methods is, therefore, representative for both in plots of **Figure 5**.

Both integration methods minimize the number of spike predictions when the input activity is synchronous (see methods). Adjusting the step size of the synchronization period in the second neural layer allows us to control the synchronicity of the input activity driven toward the third neural layer (see **Table 3**).

Figure 5 shows to what extent the synchronization period affects the simulation accuracy (**Figures 5A,C**, and **E**) and the computational performance (**Figures 5B,D**, and **F**) for each of our three neural models (LIF, AdEx, and HH), respectively. The larger the synchronization period is, the higher the computational performance (shorter execution times). Regarding simulation accuracy, event-driven methods are comparable in accuracy to time-driven methods for LIF and AdEx models. Conversely, event-driven methods present larger accuracy errors for the HH model due to RAM capacity limitations (huge look-up tables would be required to achieve similar accuracy rates).

Integration Step Size Implications

Likewise, simulation accuracy and computational performance in time driven simulation techniques using fixed-step and bi-fixed-step integration methods are tightly related to the integration step sizes. The simulation accuracy for both methods in CPU and GPU platforms is almost identical. For the sake of readability, we only show the accuracy results of CPU methods in **Figure 5**. **Figure 5** shows to what extent the decrease of the integration step size affects the simulation accuracy and the computational performance. The smaller the integration step sizes are, the more accurate the results that are obtained (**Figures 5A,C**, and **E**) but at the cost of slower simulations (**Figures 5B,D**, and **F**). When comparing the computational performance of fixed-step and bi-fixed-step integration methods in both platforms, CPU and GPU, it is demonstrated that the more complex the neural model is, the better performance results that are obtained by the bi-fixed-step methods with respect to the fixed-step ones.

Scalability Results: Implications When Increasing the Number of Network Units

This section studies the computational performance for the event-driven and time-driven simulation techniques when the neural network size is scaled up. We have measured the computational performance of our two different dynamic evaluation techniques when they are applied to different neural network sizes (**Table 4**) under equal input activity patterns (a set of random input patterns with 5 Hz mean frequency).

Figure 6 shows in the column on the left (**Figures 6A,D**, and **G**) the computational performance of our four event-driven integration methods (direct, combined, synchronous, and combined synchronous event-driven integration methods) for LIF, AdEx and HH neural models, respectively. The central column (**Figures 6B,E**, and **H**) shows the computational performance of our four time-driven simulation methods (fixed-step and bi-fixed-step integration methods in both CPU and GPU platforms) for the same three neural models. The column on the right (**Figures 6C,F,** and **I**) shows the speed-up achieved by the combined synchronous event-driven methods, the fixed-step and bi-fixed-step integration methods in GPU with respect to the direct event-driven methods, the fixed-step and bi-fixed-step integration methods in CPU for the same three neural models.

The six CPU methods (direct, combined, synchronous, and combined synchronous event-driven integration methods as well as fixed-step and bi-fixed-step integration methods) present a linear behavior. The computation time linearly increases with the number of neurons. Similarly, the two GPU integration methods (fixed-step and bi-fixed-step integration methods) perform linearly with the number of neurons (the computation time increases with the number of neurons). However, when the number of neurons to be simulated is under a certain boundary, the time spent in the synchronization and transference of data between CPU and GPU processors dominates over the neural computation time. In this case, the speed-up of GPU methods with respect to the CPU ones decreases, as shown in **Figure 6**, right column.

Input Activity Results: Implication When Increasing the Mean Firing Activity

This section studies the computational performance of the event-driven and time-driven simulation techniques as the mean firing activity of the neural network increases. The neural network described in **Table 5** has been simulated using fifteen different input activity patterns whose mean firing rate frequency ranges from 2 to 16 Hz.

Figure 7 shows in the column on the left (**Figures 7A,C**, and **E**) the computational performance of our four event-driven integration methods (direct, combined, synchronous, and combined synchronous event-driven integration methods) for LIF, AdEx, and HH neural models, respectively. The column on the right (**Figures 7B, D**, and **F**) shows the computational performance of the four time-driven simulation methods (fixed-step and bi-fixed-step integration methods in both CPU and GPU platforms) for the LIF, AdEx, and HH neural models, respectively.

Figure 7 clearly shows how event-driven schemes are sensitive to the level of input activity, whilst the impact of the input activity on time-driven integration methods is marginal. When comparing amongst event-driven integration methods, the results clearly show the improvements of the combined and

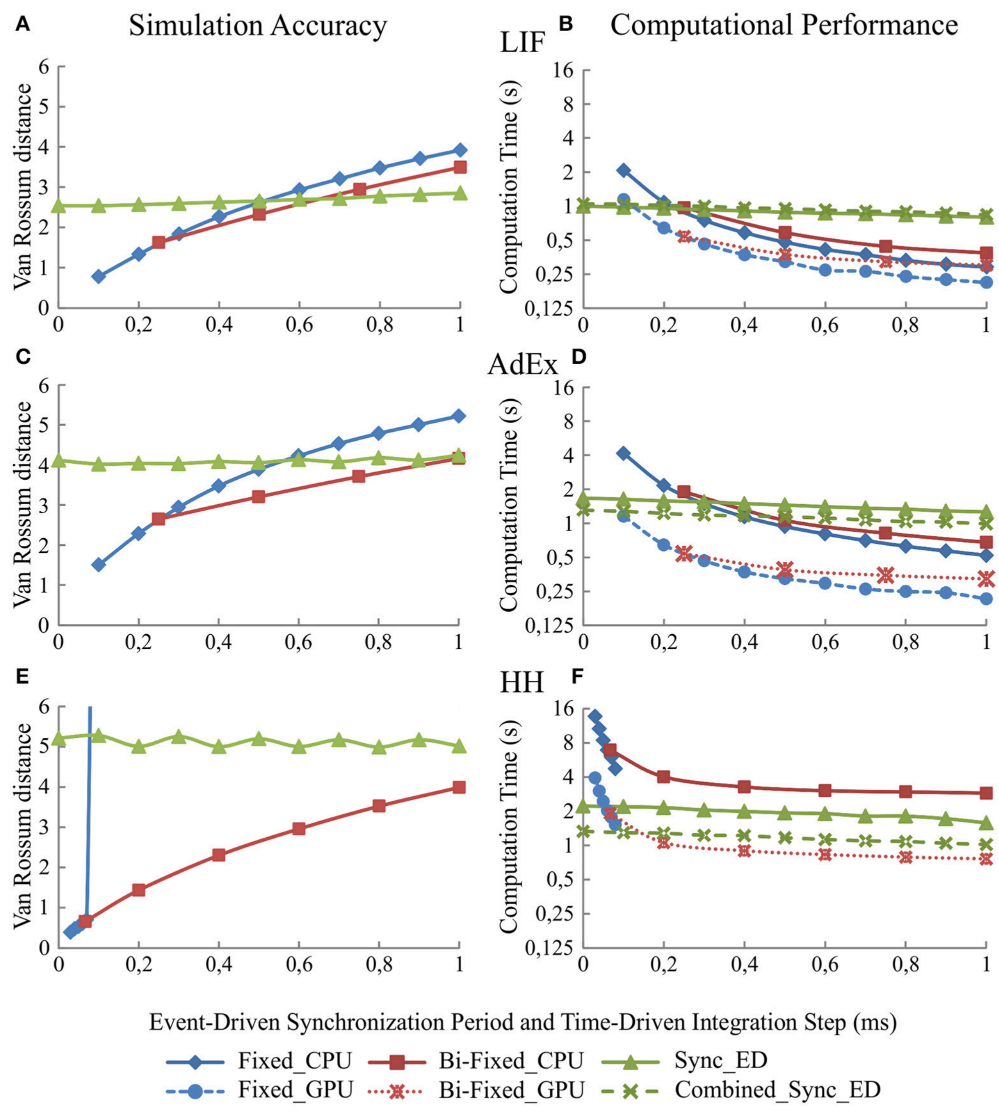


FIGURE 5 | Simulation accuracy and computational performance of synchronous and combined synchronous event-driven integration methods depending on the synchronization periods. Simulation accuracy and computational performance of fixed-step and bi-fixed-step integration methods in CPU and GPU platforms depending on the integration step size. One-second simulation of the neural networks defined in **Table 3** is shown. Five different input spike patterns of 5 Hz that generate mean firing rate activities of 10 Hz in the third neural layer are used. The left-hand column **(A,C,** and **E)** of the panel shows the simulation accuracy and the right-hand column **(B,D,** and **F)** the computational performance obtained by the synchronous and combined synchronous event-driven integration methods depending on the synchronization period for LIF, AdEx, and HH models, respectively (the synchronization period is plotted over x-axis). Both columns also show the simulation accuracy and computational performance of fixed-step and bi-fixed-step integration methods in CPU and GPU platforms depending on the integration steps (the global integration step size of fixed-step and bi-fixed-step integration methods are plotted over x-axis. The local step sizes for bi-fixed-step integration methods are 0.25 ms for LIF and AdEx models and 1/15 ms for HH model). The standard deviation of the simulation accuracy and the computational performance obtained is negligible; we only represent the mean values. Synchronous and combined synchronous event-driven integration methods present identical accuracy results. CPU and GPU time-driven integration methods present almost identical accuracy results. The stiffness of HH model constrains the maximum step size that fixed-step integration methods can use. Beyond this step size, the differential equations cannot properly be integrated for this model.

synchronous integration methods. The slope of the result series (i.e., the impact of the average activity on the simulation performance) decreases when these integration methods are adopted in the simulation scheme. When comparing amongst time-driven methods, the improvements of using bi-fixed-step methods (leading to 2-fold performance levels compared to fixed-step methods) and GPU as co-processing engine (leading to 5-fold performance levels compared to CPU approaches) are also clear in the obtained results. Amongst the four time-driven integration methods proposed, the bi-fixed-step method in CPU is the most severely affected by the increasing of the mean firing activity within the neural network. The overhead time spent in deploying the adaptation mechanism (see methods) makes these integration methods unadvisable for scenarios where

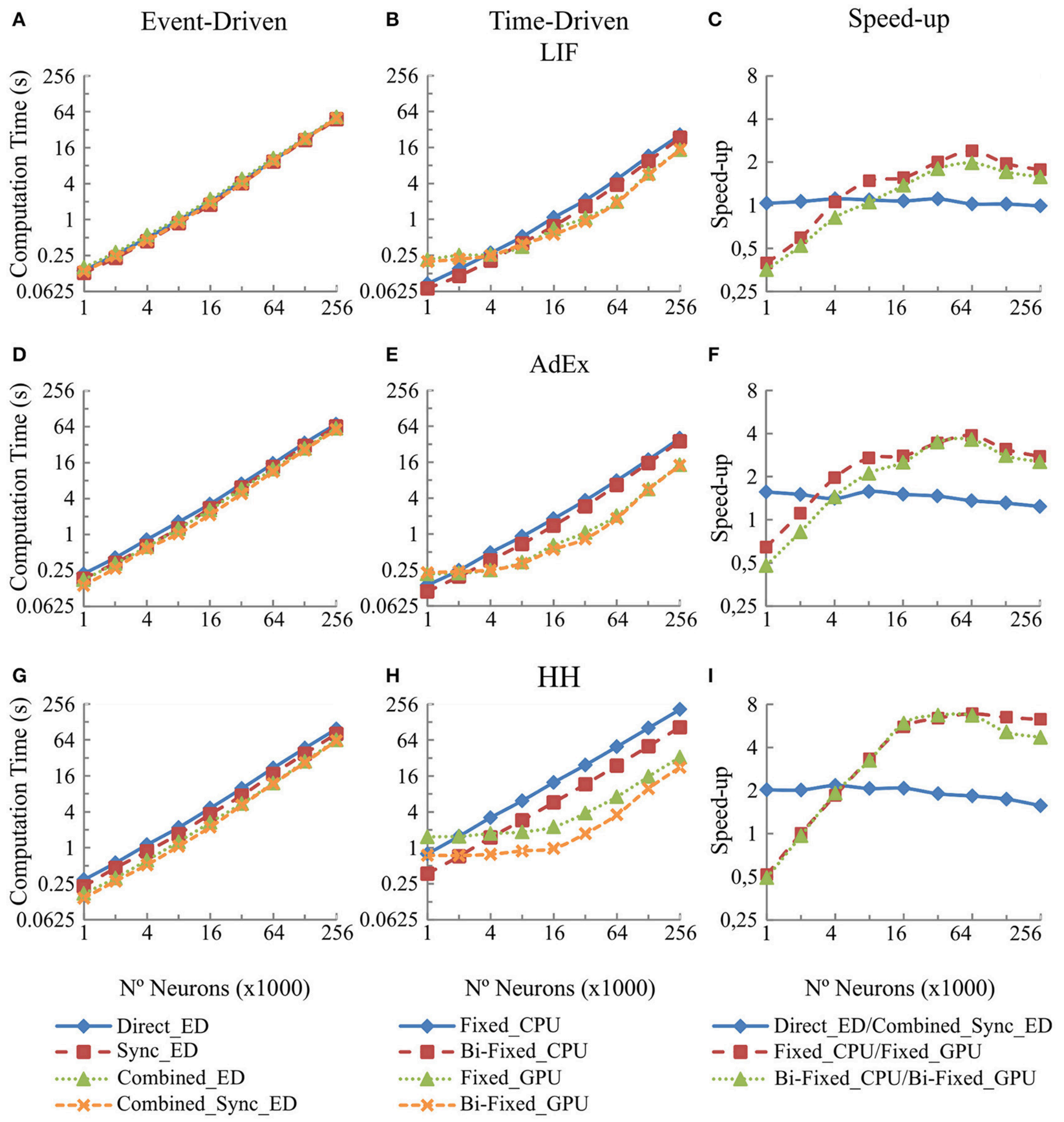


FIGURE 6 | Computational performance obtained from the eight integration methods proposed for LIF, AdEx and HH neural models depending on the number of neurons within the second neural layer. One-second simulation of the neural networks defined in **Table 4** is shown. A mean input activity of 5 Hz that generates a mean firing rate activity of 10 Hz in the second neural layer is used. The left-hand column **(A,D,** and **G)** of the panel shows the computational performance of the four event-driven integration methods (direct, combined, synchronous, and combined synchronous event-driven methods) for LIF, AdEx, and HH neural models, respectively. The central column **(B,E,** and **H)** of the panel shows the computational performance of the four time-driven integration methods (fixed-step and bi-fixed-step integration methods in both CPU and GPU platforms) for the same three neural models. The right-hand column **(C,F,** and **I)** of the panel shows the speed-up achieved by the combined synchronous event-driven methods, the fixed-step and bi-fixed-step integration methods in GPU respect to the direct event-driven methods, the fixed-step and bi-fixed-step integration methods in CPU for the same three neural models.

the neural network presents very high levels of constant firing activity.

Connectivity Results: Implications When Increasing the Number of Synapses in the Recurrent Topology

This section studies the computational performance for the event-driven and time-driven simulation techniques as the number of synapses in the recurrent topology of our neural network increases. The neural network described in **Table 6** has been simulated using a random input activity with a mean firing rate of 5 Hz.

Figure 8 shows in the column on the left (**Figures 8A,C,** and **E**) the computational performance of our four event-driven integration methods (direct, combined, synchronous, and combined synchronous event-driven integration methods) for LIF, AdEx, and HH neural models, respectively. The column on

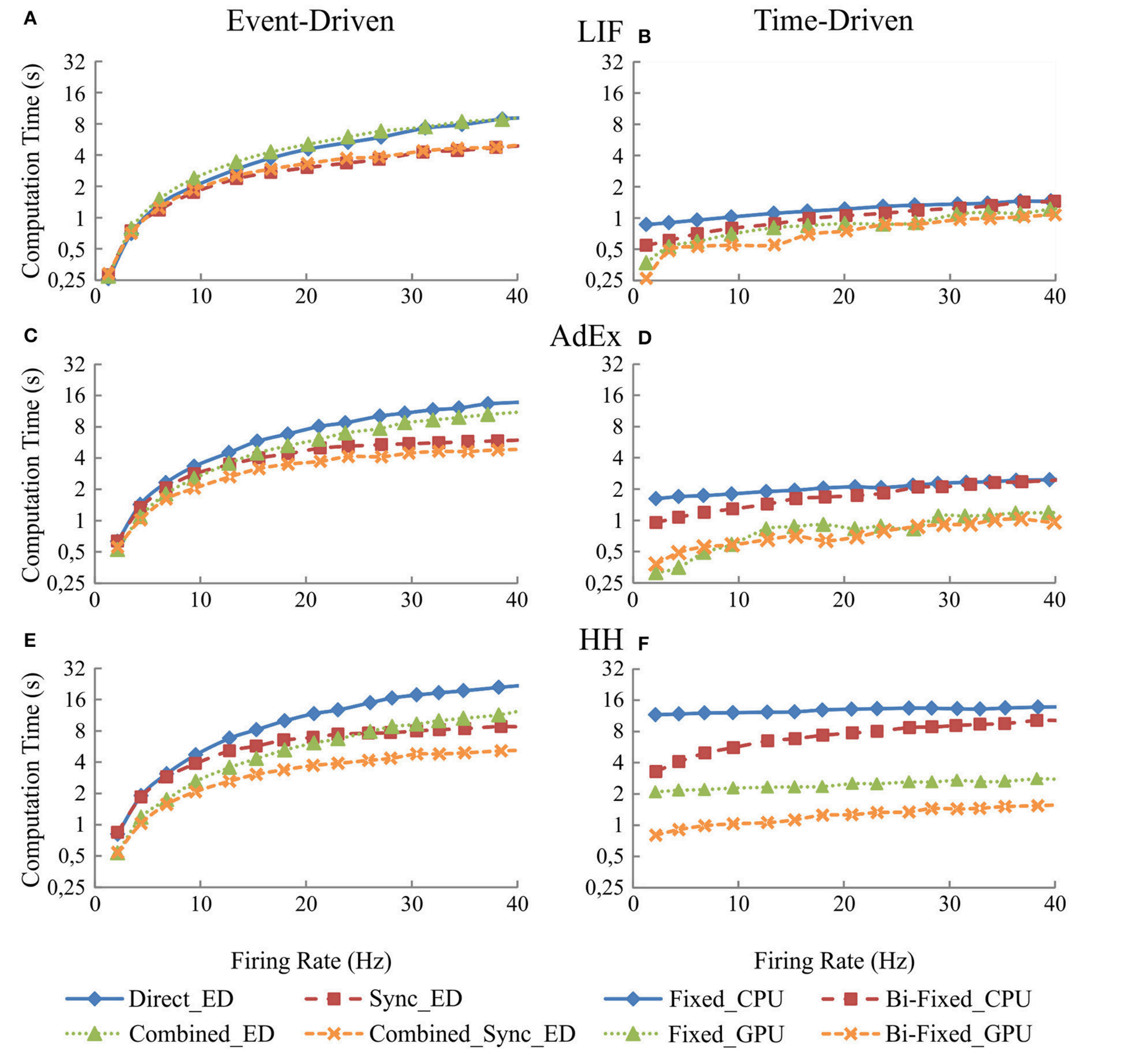


FIGURE 7 | Computational performance obtained from the eight simulation methods proposed for LIF, AdEx, and HH neural models depending on the mean firing rate activity in the second neural layer. One-second simulation of the neural networks defined in **Table 5** is shown. The neural input activity ranges from 1 to 10 Hz on average. The mean neural activity obtained at the second layer is plotted over x-axis. The left-hand column **(A,C,** and **E)** of the panel shows the computational performance of the four event-driven integration methods (direct, synchronous, combined, and combined synchronous event-driven integration methods) for LIF, AdEx, and HH neural models, respectively. The right-hand column **(B,D,** and **F)** of the panel shows the computational performance of the four time-driven integration methods (fixed-step and bi-fixed-step integration methods in both CPU and GPU platforms) for the same three neural models.

the right (**B, D**, and **F**) shows the computational performance of our four time-driven integration methods (fixed-step and bi-fixed-step integration methods in both CPU and GPU platforms) for the LIF, AdEx, and HH neural models, respectively. The firing rate activity remains quite stable (between 8 and 12 Hz), although the number of propagated spikes increases due to the higher number of synapses. The computation time (the measured variable) depends on the computational workload. This workload, in turn, depends on the number of internal spikes and recurrent synapses (number of propagated spikes = number of internal spikes · number of recurrent synapses). The number of propagated spikes is plotted in x-axis instead of the number of recurrent synapses to better compare the computation time of all the simulation methods under equivalent neural activity conditions. Each mark in **Figure 8** corresponds to a number of recurrent synapses (10, 20, 40, 80, 160, 320, 640, and 1280) since this is the parameter that can be directly set in the network definition and thus in the simulation experiment.

The simulation performance in event-driven integration methods significantly decreases as the number of propagated

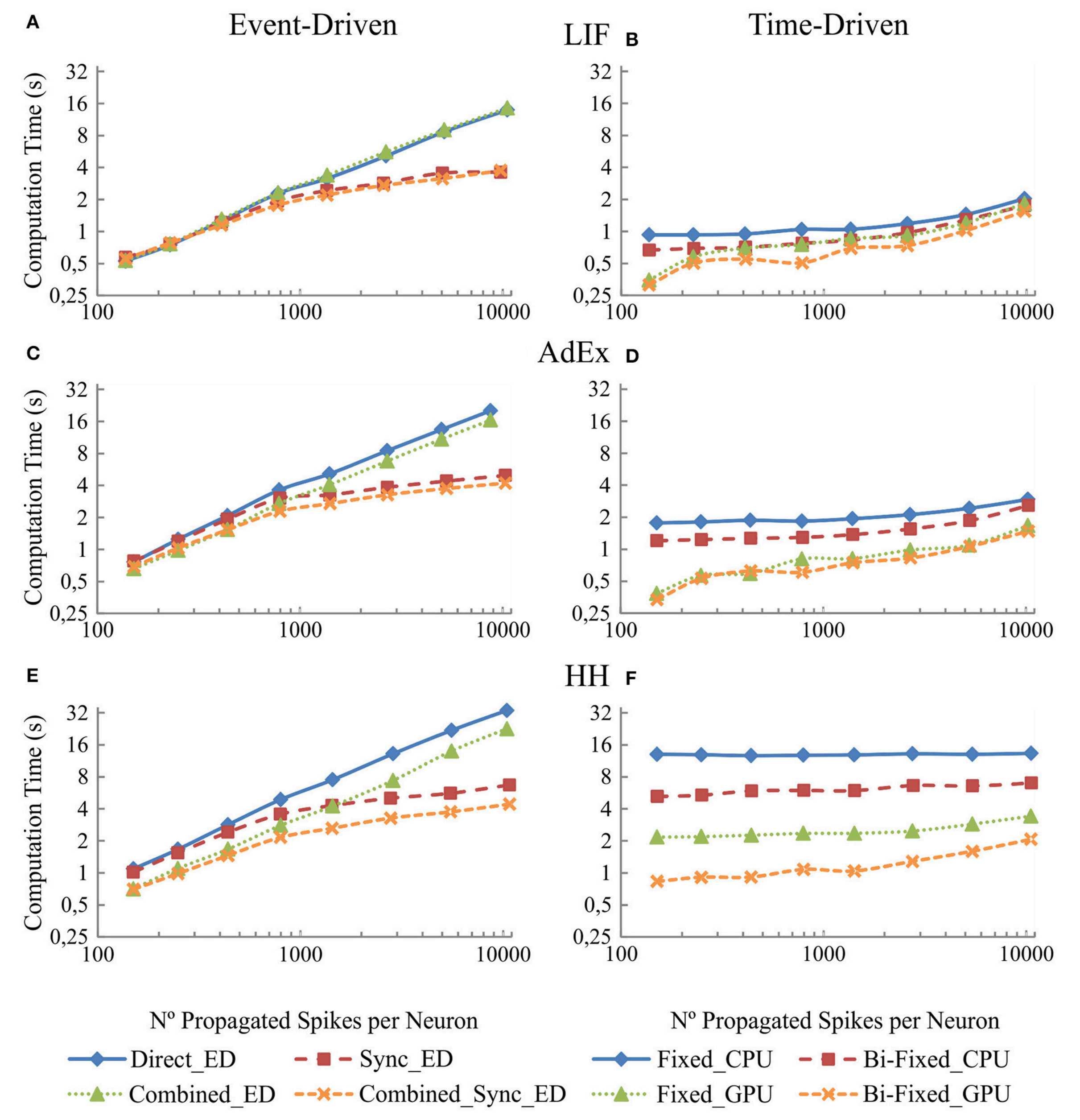


FIGURE 8 | Computational performance of the eight simulation methods proposed for LIF, AdEx, and HH neural models depending on the number of propagated spikes that are determined by the number recurrent synapses. One-second simulation of the neural networks defined in **Table 6** is shown. The number of recurrent synapses is geometrically scaled up (10, 20, 40, 80, 160, 320, 640, and 1,280). A mean input activity of 5 Hz is used. This input activity generates a mean firing rate activity of between 8 and 12 Hz within the second neural layer. The number of propagated spikes increments proportionally with the number of synapses (number of propagated spikes = number of internal spikes · number of recurrent synapses). The mean number of propagated spikes that arrives to the second layer is plotted over x-axis. The left-hand column **(A,C,** and **E)** of the panel shows the computational performance of the four event-driven integration methods (direct, synchronous, combined, and combined synchronous event-driven integration methods) for LIF, AdEx, and HH neural models, respectively. The right-hand column **(B,D,** and **F)** shows the computational performance of the four time-driven integration methods (fixed-step and bi-fixed-step integration methods in both CPU and GPU platforms) for the same three neural models.

spikes increases. Nevertheless, we can see a significant improvement when synchronous event-driven integration methods are used since they are optimized for computing higher levels of synchronous activity. Conversely, the simulation performance in time-driven integration methods suffers little direct impact as the number of propagated spikes increases. The results show the improvement achieved with the bi-fixed-step integration methods either with or without GPU co-processing.

When comparing amongst event- and time-driven methods, GPU time-driven methods have the best-in-class performance (see **Figures 7**, **8**). Incremental levels of input activity cause an incremental number of propagated spikes thus favoring GPU time-driven methods. On the other hand, tend-to-zero input activity levels favor event-driven methods. The event-driven performance obtained under these low input activities is usually equal to or better than GPU time-driven performance.

DISCUSSION

Throughout this paper, different neural dynamic evaluation techniques are developed. Within the event-driven methods: the combined integration methods based on the combination of look-up tables and the synchronous integration methods based on the optimization of processing synchronous activities. These two integration method are clear improvements with respect to previously described event-driven neural dynamic evaluation techniques (Ros et al., 2006a). As far as the time-driven methods are concerned, the bi-fixed-step integration methods and the CPU-GPU co-processing significantly increase the performance of time-driven neural dynamic evaluation techniques.

The quality level of each proposed integration method is given in terms of neural accuracy and computational performance when simulating three neural models of incremental mathematical complexity (LIF, AdEx, and HH). These neural models are set up (**Table 7**) for reproducing similar activity patterns. All the simulation methods shall provide similar accuracy results to make them comparable. Fixed-step and bi-fixed-step time-driven integration methods for LIF and AdEx models are set up (**Table 1**) for obtaining similar accuracy results than event-driven methods (**Figure 5**). LIF and AdEx models are compiled in look-up tables of 249 and 712 MB, respectively (**Figure 4**).

The higher complexity of the HH model imposes a large storage memory capacity. An event-driven HH model with comparable accuracy levels to bi-fixed-step time-driven HH model would require up to 14 GB of storage memory capacity (estimation extrapolated from **Figure 4A**). In this benchmark, the HH model has been compiled in look-up tables of 1195 MB that obtain larger accuracy errors results than the equivalent time-driven methods.

TABLE 7 | Summary of parameters for LIF, AdEx and HH neural models.

LIF		AdEx		HH	
C	0.19e–9 F	C	110 pF	C	120 pF
E_L	−0.065 V	E_L	−65 mV	E_L	−65 mV
g_L	10e–9 S	g_L	10 nS	g_L	10 nS
V_T	−0.050 V	V_T	−50 mV	V_T	−52 mV
T_{ref}	0.0025 s	Δ_T	2 mV	g_{Na}	20 nS
E_{AMPA}	0.0 V	τ_w	50 ms	E_{Na}	50 mV
E_{GABA}	−0.080 V	A	1 nS	g_{Kd}	6000 nS
τ_{AMPA}	0.005 s	B	9 pA	E_K	−90 mV
τ_{GABA}	0.010 s	V_r	−80 mV	E_{AMPA}	0.0 mV
		E_{AMPA}	0.0 mV	E_{GABA}	−80 mV
		E_{GABA}	−80 mV	τ_{AMPA}	5 ms
		τ_{AMPA}	5 ms	τ_{GABA}	10 ms
		τ_{GABA}	10 ms		

Event-Driven Main Functional Aspects

The main functional aspects in relation to the event-driven integration methods can be summarized as follows:

- The number of state variables defining a neural model represents, broadly speaking, the complexity of a neural model. When this number increases linearly, the memory requirements to allocate the pre-compiled look-up tables of the event-driven neural models increases geometrically. Thus, reducing the level of granularity of each dimension is the only way to reduce the total look-up table size, but this reduction directly affects the simulation accuracy (as shown in **Figure 4A**). The more complex the neural models are or the smaller the look-up table sizes are, the higher van Rossum distance values (less accuracy) that are obtained. Boundaries in accuracy and memory capacity constrain the maximum neural complexity that these event-driven techniques can handle.
- The recombination of look-up tables improves the computational performance, maintaining the simulation accuracy. Actually, the combined event-driven integration methods slightly increase the computation time when the neural model complexity increases because the neural state update process of several variables using combined look-up tables is slightly more complex than the update of just one variable. Larger look-up table sizes cause higher rates of cache failures and, therefore, losses in computational performance (see **Figure 4**). This means that the computational performance is more impacted by the total look-up table size than by the mathematical complexity (the number of state variables) of the neural model, although both the mathematical complexity and the look-up table size are related.
- The computation mechanism used by synchronous methods to deal with synchronous activity significantly improves the computational performance. When a synchronous event-driven neuron receives input synapses coming from other synchronous event-driven neurons or time-driven neurons, the computational performance enhancement depends on either the synchronization period or the integration step size of the previous layers. The larger the synchronization period or the integration step size of the previous layers are, the more synchronous the activity that arrives to the synchronous model and the higher performance levels with respect to the direct non-synchronized integration methods (see **Figures 5–8**). Regarding the simulation accuracy, the look-up tables are precompiled maintaining a certain degree of precision. A lager synchronization period only generates a negligible error in the spike generation time which, in turn, causes small oscillations in the van Rossum distance measurements (**Figure 5**).
- Both neural dynamic evaluation techniques (the combination of look-up tables and synchronization of activity) are simultaneously applied by the combined synchronous event-driven method. This simulation technique outperforms the

rest of event-driven techniques. This is more significant when the mathematical complexity of the neural models increases (see **Figures 5–8**).

- The main factor that finally constrains the computational performance of all these event-driven methods is the number of events that need to be processed. These events are mainly internal and propagated spikes (Ros et al., 2006a) that linearly increase with the neural activity. Time-driven integration methods are preferred rather than event-driven integration methods for those neural networks with high levels of neural activity (see **Figures 7**, **8**). Conversely, there are particular cases in which the event-driven integration methods can be the best option. There are, actually, biologically realistic SNNs in which parts of their inner layers present a very low and sparse neural activity, such as the granular cells in the cerebellum (D'Angelo et al., 2016) or the mushroom bodies within the olfactory system in Drosophila (Serrano et al., 2013). The importance of these particular networks cannot be overlooked (i.e., just the granular cerebellar layer accounts for half of the neurons of the whole brain, its neurons receive between three and six input synapses with a low and very sparse activity, with most of them remaining silent and barely generating spikes). In these cases, event-driven integration methods perform better than time-driven integration methods.

Time-Driven Main Functional Aspects

The main functional aspects in relation to the time-driven integration methods can be summarized as follows:

- Hybrid CPU-GPU integration methods perform better than CPU methods. This is specifically relevant when the mathematical complexity of the neural models increases. GPU hardware architecture performs better computing parallel tasks than CPU architecture. The computation of the neural dynamics is a pure parallelizable task and consequently, GPU-friendly. In a hybrid CPU-GPU platform, the GPU only processes the neural dynamics, whilst the spike generation and propagation are processed in the CPU. When the mathematical complexity of the neural models increases, the workload assigned to the GPU increases, whilst the workload of the CPU remains equal. For this reason, CPU-GPU neural models perform better than purely CPU neural models, especially when the mathematical complexity of the neural models increases. This increase in performance is shown in **Figures 5–8**.
- Bi-fixed-step integration methods outperform fixed-step integration methods for both CPU and GPU platforms when the mathematical complexity of the neural model increases (see **Figures 5–8**). Complex neural models usually demand small integration step sizes to better cope with the stiffness of their neural model equations during the spike shape generation. **Figures 5E,F** show how the maximum step size on a fixed-step integration method is constrained due to the differential equation stiffness (HH model). The adaptation mechanism used by the CPU bi-fixed-step integration methods improves the simulation performance by enlarging the simulation step size during those neural dynamic intervals out of the spike phase.
- The adaptation mechanism of the integration step size for GPU bi-fixed-step integration methods increases performance thanks to the minimization of the time spent in the synchronization and transfer of data between the CPU and GPU processors.
- Whilst CPU integration methods are better suited for small-medium groups of neurons (from one neuron to several thousands of neurons, depending on the mathematical complexity), the GPU integration methods are better suited for larger numbers of neurons (from thousands to millions of neurons). The computation time invested in the synchronization period and data transferences between CPU and GPU platforms dominates over the computation time invested in solving the neural dynamics when the number of neurons within the network is small (see **Figure 6**). In this case, the computational performance of the GPU integration methods reaches a plateau.
- The adaptation mechanism that the bi-fixed-step integration method uses in CPU may decrease the computational performance when the mean firing rate over the neural network is quite high. When the neural activity increases, the ratio of use between the local and global step also increases. The computational workload for the neural dynamic increases and the performance drops (see how the computation time increases in **Figure 7**).

EDLUT Hybrid Architecture into Perspective

EDLUT is a simulator mainly oriented to efficiently simulate medium-scale neural networks (tens of thousands of neurons) pursuing real time simulations. EDLUT uses point neural models, such as LIF, AdEx or HH. EDLUT information transmission relies on spike timing rather than on the particular spike shape. What matters is when the spike is emitted rather than how the spike is generated. Neurons are just means to an end needed toward understanding the behavior of the neural network behind. The neural communication mechanisms are deployed at network level at very high simulation speeds on a single multicore computer, thus facilitating real time embodiment experiments (Carrillo et al., 2008; Luque et al., 2011a,b, 2014a,b, 2016; Garrido et al., 2013a; Casellato et al., 2014; Antonietti et al., 2016). In these neurorobotic experimental set-ups the neural network and the body are coupled as a single entity.

Conversely, NEURON (Hines and Carnevale, 1997) is mainly designed for the simulation of very complex and detailed neural models. What matters here is how the spike was generated rather than when it was emitted. Understanding neurons themselves is the goal. To be as biologically plausible as possible, NEURON is conceived to deal with high levels of mathematical complexity that usually require time-driven simulation methods (either fixed- or variable-step integration methods). The computational cost here highly depends on the mathematical complexity which makes the simulation of hundreds or tens of hundreds neurons conforming a network almost computationally intractable. Using

NEURON for the benchmark analysis proposed here would be out of context.

NEURON, lately, seems to be increasing its field of application toward medium- large-scale neural networks (see Lytton et al., 2016) that are comprised of highly simplified neural models (i.e., Izhikevich or the four dimensional HH models). Note that the time-driven simulation techniques here proposed may have a direct impact on NEURON if this tendency is finally consolidated.

In contrast, BRIAN (Goodman and Romain, 2009) and NEST (Gewaltig and Diesmann, 2007) are simulators often considered to be playing in the same league as EDLUT. As is the case with EDLUT, Brian claims to be mainly oriented to efficiently simulate medium-scale neural networks (tens of thousands of neurons) while NEST is designed for very large-scale neural networks (up to 1.86 billion neurons connected by 11.1 trillion synapses on the Japanese K supercomputer; Kunkel et al., 2014). These simulators mainly implement point neuron models, although some models with few compartments can be simulated. Similarly, they consider neurons to be just means to an end. They use neurons to understand the behavior of the neural network behind. Both are natively implementing time-driven simulation methods in CPU and particularly BRIAN also implements a hybrid CPU-GPU co-processing scheme for time-driven models. Having said that, the conclusions and approaches proposed in the paper regarding time-driven methods would have a direct impact on Brian and a substantial impact on NEST since CPU-GPU co-processing is still missing. The other fundamental pillar of the methodology proposed here, the event-driven scheme, is not included in BRIAN but it does exist in NEST. Whilst the event-driven EDLUT framework (originally an event-driven scheme) was adapted to also perform time-driven neural simulations (Garrido et al., 2011), the time-driven NEST framework (originally a time-driven scheme) was adapted to also perform event-driven neural simulations (Morrison et al., 2007; Hanuschkin et al., 2010). Thus, both simulators can perform combined event- and time-driven simulations. In fact, NEST proposes an event-driven method that presents similarities to our synchronous event-driven method. Both event-driven methods minimize the number of spike predictions by processing all the synchronous input spikes conjointly and thus make only one prediction.

CONCLUSIONS

The way forward in computational neuroscience lies in the simulation of biologically plausible computational models of different nervous centers (cerebellum, inferior olive, cuneate nucleus, etc.) to better understand how the information is processed within these nervous centers. Computational neuroscience allows the study of these nervous center models without experimental restrictions using neural models that have been developed and validated according to experimental cellular data.

These nervous center models can be simulated in different conditions and circumstances to give a consistent idea about how they may operate. In many cases, these models are becoming a fundamental tool in the neuroscience hypothesis-experimentation cycle. The computational models allow researchers to test their hypotheses in simulation. This fact leads to making a better hypothesis and better experiments designed with a greater probability of success.

The road to model and simulate nervous centers has been progressively paved with increasing levels of mathematical complexity to include more and more biological features. However, this mathematical complexity comes at a computational cost (i.e., neural accuracy and computational performance). In this paper, we have proposed several new neural dynamic evaluation techniques to cope with the incremental mathematical complexity of well-known neural models (LIF, AdEx, and HH):

(a) The combined synchronous event-driven integration method combines the look-up tables to minimize the number of look-up table data queries needed to update the neural state variables during the simulation process. Additionally, this method also minimizes the look-up table data queries, making just one prediction about the emission of an output spike for each group of synchronous input spikes that arrive to each neuron.
(b) The bi-fixed-step integration method (optimized also in GPU) in which the neural dynamic equations that define the complex neural models (as HH) are accurately solved by switching between two time steps of different lengths during the simulation process.

All these integration methods, with their own pros and cons, are meant to be used concurrently to increase the computational performance when simulating heterogeneous SNNs (such as those previously studied in Naveros et al., 2015). These heterogeneous SNNs consist of several layers with different neural properties, thus trying to mimic the neural heterogeneity found in different brain regions, such as the cerebellum (D'Angelo et al., 2016; Luque et al., 2016) or the cuneate nucleus (Bologna et al., 2011). The simulation platform used in this study integrates all these neural dynamic evaluation techniques in such a way that parts of the neural network (with low and sparse activity) can be simulated efficiently with event-driven methods (which have been optimized to more efficiently deal with relatively-complex neural models and synchronous activity) and parts of the neural networks (with higher activity in terms of number of spikes) can be simulated with time-driven methods (which have been optimized with bi-fixed-step integration methods and the capability of using highly parallel hardware, such as GPU engines). See Appendix B for a simulation accuracy study of neural networks with combined event- and time-driven methods.

Choosing the most appropriate method or combination of methods for each neural center model to be simulated is a trade-off amongst three elements:

1. The neural network architecture (number of neurons, neural model complexity, number of input and output synapses, mean firing rates, etc.).

2. The hardware restrictions (number of CPU and GPU cores, RAM size).
3. The simulation requirements and target (minimizing the execution time, maximizing accuracy, etc.).

Finally, this study has been done using neural networks with medium-low connectivity ratios (from 10 to 1280 input synapses per neuron) oriented to fast simulations. However, the simulation performance results may change significantly when simulating neural networks with larger connectivity ratios (for example 10,000 input synapses per neuron). In this case the spike propagation task is usually more time consuming than the neural dynamics update task for time-driven methods. Nevertheless, as can be seen in **Figure 8**, our synchronous event-driven method improves its performance in relation to the direct event-driven method when the number of synapses increases.

AUTHOR CONTRIBUTIONS

All authors listed have made substantial, direct and intellectual contribution to the work, and approved it for publication. FN, RC, JG, and NL conceived and designed the experiments. FN performed the experiments. FN, RC, NL, JG, and ER analyzed the data. FN, NL, and RC contributed reagents/materials/analysis tools. FN, NL, and ER wrote the paper.

ACKNOWLEDGMENTS

This study was supported by the European Union NR (658479-Spike Control), the Spanish National Grant NEUROPACT (TIN2013-47069-P) and by the Spanish National Grant PhD scholarship (AP2012-0906). We gratefully acknowledge the support of NVIDIA Corporation with the donation of two Titan GPUs for current EDLUT development.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://journal.frontiersin.org/article/10.3389/fninf.2017.00007/full#supplementary-material

REFERENCES

Agis, R., Ros, E., Diaz, J., Carrillo, R., and Ortigosa, E. M. (2007). Hardware event-driven simulation engine for spiking neural networks. *Int. J. Electron.* 94, 469–480. doi: 10.1080/00207210701308625

Antonietti, A., Casellato, C., Garrido, J. A., Luque, N. R., Naveros, F., Ros, E., et al. (2016). Spiking neural network with distributed plasticity reproduces cerebellar learning in eye blink conditioning paradigms. *IEEE Trans. Biomed. Eng.* 63, 210–219. doi: 10.1109/TBME.2015.2485301

Bologna, L. L., Pinoteau, J., Brasselet, R., Maggiali, M., and Arleo, A. (2011). Encoding/decoding of first and second order tactile afferents in a neurorobotic application. *J. Physiol. Paris* 105, 25–35. doi: 10.1016/j.jphysparis.2011.08.002

Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* 94, 3637–3642. doi: 10.1152/jn.00686.2005

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., et al. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *J. Comput. Neurosci.* 23, 349–398. doi: 10.1007/s10827-007-0038-6

Carrillo, R. R., Ros, E., Boucheny, C., and Coenen, O. J. (2008). A real-time spiking cerebellum model for learning robot control. *BioSystems* 94, 18–27. doi: 10.1016/j.biosystems.2008.05.008

Casellato, C., Antonietti, A., Garrido, J. A., Carrillo, R. R., Luque, N. R., Ros, E., et al. (2014). Adaptive robotic control driven by a versatile spiking cerebellar network. *PLoS ONE* 9:e112265. doi: 10.1371/journal.pone.0112265

D'Angelo, E., Antonietti, A., Casali, S., Casellato, C., Garrido, J. A., Luque, N. R., et al. (2016). Modelling the cerebellar microcircuit: new strategies for a long-standing issue. *Front. Cell. Neurosci.* 10:176. doi: 10.3389/fncel.2016.00176

Delorme, A., and Thorpe, S. J. (2003). Spikenet: an event-driven simulation package for modelling large networks of spiking neurons. *Network* 14, 613–627. doi: 10.1088/0954-898X_14_4_301

Doesburg, S. M., Green, J. J., McDonald, J. J., and Ward, L. M. (2012). Theta modulation of inter-regional gamma synchronization during auditory attention control. *Brain Res.* 1431, 77–85. doi: 10.1016/j.brainres.2011.11.005

Eckhorn, R., Reitboeck, H. J., Arndt, M., and Dicke, P. (1990). Feature linking via synchronization among distributed assemblies: simulations of results from cat visual cortex. *Neural Comput.* 2, 293–307. doi: 10.1162/neco.1990.2.3.293

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., et al. (2013). Overview of the spinnaker system architecture. *IEEE Trans. Comp.* 62, 2454–2467. doi: 10.1109/TC.2012.142

Garrido, J. A., Carrillo, R. R., Luque, N. R., and Ros, E. (2011). Event and Time driven hybrid simulation of spiking neural networks. *Adv. Comput. Intel.* 6691, 554–561 doi: 10.1007/978-3-642-21501-8_69

Garrido, J. A., Luque, N. R., D'Angelo, E., and Ros, E. (2013a). Distributed cerebellar plasticity implements adaptable gain control in a manipulation task: a closed-loop robotic simulation. *Front. Neural Circuits* 7:159. doi: 10.3389/fncir.2013.00159

Garrido, J. A., Luque, N. R., Tolu, S., and D'Angelo, E. (2016). Oscillation-driven spike-timing dependent plasticity allows multiple overlapping pattern recognition in inhibitory interneuron networks. *Int. J. Neural Syst.* 26:1650020. doi: 10.1142/S0129065716500209

Garrido, J. A., Ros, E., and D'Angelo, E. (2013b). Spike timing regulation on the millisecond scale by distributed synaptic plasticity at the cerebellum input stage: a simulation study. *Front. Comput. Neurosci.* 7:64. doi: 10.3389/fncom.2013.00064

Gerstner, W., and Kistler, W. (2002). *Spiking Neuron Models*. Cambridge: Cambridge University Press.

Gewaltig, M.-O., and Diesmann, M. (2007). NEST (neural simulation tool). *Scholarpedia* 2:1430. doi: 10.4249/scholarpedia.1430

Ghosh-Dastidar, S., and Adeli, H. (2009). Spiking neural networks. *Int. J. Neural Syst.* 19, 295–308. doi: 10.1142/S0129065709002002

Goodman, D. F. M., and Romain, B. (2009). The brian simulator. *Front. Neurosci.* 3, 192–197. doi: 10.3389/neuro.01.026.2009

Hanuschkin, A., Kunkel, S., Helias, M., Morrison, A., and Diesmann, M. (2010). A general and efficient method for incorporating precise spike times in globally time-driven simulations. *Front. Neuroinform.* 4:113. doi: 10.3389/fninf.2010.00113

Hines, M. L., and Carnevale, N. T. (1997). The NEURON Simulation Environment. *Neural Comput.* 9, 1179–1209. doi: 10.1162/neco.1997.9.6.1179

Hodgkin, A. L., and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiology* 117, 500–544. doi: 10.1113/jphysiol.1952.sp004764

Iserles, A. (2009). *A First Course in the Numerical Analysis of Differential Equations*. Cambridge: Cambridge University Press.

Kunkel, S., Schmidt, M., Eppler, J. M., Plesser, H. E., Masumoto, G., Igarashi, J., et al. (2014). Spiking network simulation code for petascale computers. *Front. Neuroinform.* 8:78. doi: 10.3389/fninf.2014.00078

Luque, N. R., Garrido, J. A., Carrillo, R. R., Coenen, O. J., and Ros, E. (2011a). Cerebellar input configuration toward object model abstraction in manipulation tasks. *IEEE Trans. Neural Netw.* 22, 1321–1328. doi: 10.1109/TNN.2011.2156809

Luque, N. R., Garrido, J. A., Carrillo, R. R., Coenen, O. J., and Ros, E. (2011b). Cerebellarlike corrective model inference engine for manipulation tasks. *IEEE Trans. Syst. Man, and Cybern. B Cybern.* 41, 1299–1312. doi: 10.1109/TSMCB.2011.2138693

Luque, N. R., Carrillo, R. R., Francisco, N., Jesús, A. G., and Sáez-Larac, M. J. (2014a). Integrated neural and robotic simulations. Simulation of cerebellar neurobiological substrate for an object-oriented dynamic model abstraction process. *Rob. Auton. Syst.* 62, 1702–1716. doi: 10.1016/j.robot.2014.08.002

Luque, N. R., Garrido, J. A., Carrillo, R. R., D'Angelo, E., and Ros, E. (2014b). Fast convergence of learning requires plasticity between inferior olive and deep cerebellar nuclei in a manipulation task: a closed-loop robotic simulation. *Front. Comput. Neurosci.* 8:97. doi: 10.3389/fncom.2014.00097

Luque, N. R., Garrido, J. A., Naveros, F., Carrillo, R. R., D'Angelo, E., and Ros, E. (2016). Distributed cerebellar motor learning; a spike-timing-dependent plasticity model. *Front. Comput. Neurosci.* 10:17. doi: 10.3389/fncom.2016.00017

Lytton, W. W., Seidenstein, A. H., Dura-Bernal, S, McDougal, R. A., Schürmann, F., and Hines, M. L. (2016). Simulation neurotechnologies for advancing brain research: parallelizing large networks in NEURON. *Neural Computat.* 28, 2063–2090. doi: 10.1162/NECO_a_00876

Mattia, M., and Del Giudice, P. (2000). Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Comput.* 12, 2305–2329. doi: 10.1162/089976600300014953

Mcculloch, W. S., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biol.* 5, 115–133. doi: 10.1007/bf02478259

Morrison, A., Straube, S., Plesser, H. E., and Diesmann, M. (2007). Exact subthreshold integration with continuous spike times in discrete-time neural network simulations. *Neural Comput.* 19, 47–79. doi: 10.1162/neco.2007.19.1.47

Naveros, F., Luque, N. R., Garrido, J. A., Carrillo, R. R., Anguita, M., and Ros, E. (2015). A spiking neural simulator integrating event-driven and time-driven computation schemes using parallel CPU-GPU co-processing: a case study. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 1567–1574. doi: 10.1109/TNNLS.2014.2345844

Neumann, J. (1958). *The Computer and the Brain*. New Haven, CT: Yale University Press.

Pecevski, D., Kappel, D., and Jonke, Z. (2014). NEVESIM: event-driven neural simulation framework with a Python interface. *Front. Neuroinform.* 8:70. doi: 10.3389/fninf.2014.00070

Pelayo, F. J., Ros, E., Arreguit, X., and Prieto, A. (1997). VLSI implementation of a neural model using spikes. *Analog Integr. Circuits Signal Process.* 13, 111–121. doi: 10.1023/A:1008240229616

Reutimann, J., Giugliano, M., and Fusi, S. (2003). Event-driven simulation of spiking neurons with stochastic dynamics. *Neural Comput.* 15, 811–830. doi: 10.1162/08997660360581912

Ros, E., Carrillo, R., Ortigosa, E. M., Barbour, B., and Agís, R. (2006a). Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics. *Neural Comput.* 18, 2959–2993. doi: 10.1162/neco.2006.18.12.2959

Ros, E., Ortigosa, E. M., Agis, R., Carrillo, R., and Arnold, M. (2006b). Real-time computing platform for spiking neurons (RT-Spike). *IEEE Trans. Neural Netw.* 17, 1050–1063. doi: 10.1109/TNN.2006.875980

Rudolph, M., and Destexhe, A. (2006). Analytical integrate-and-fire neuron models with conductance-based dynamics for event-driven simulation strategies. *Neural Comput.* 18, 2146–2210. doi: 10.1162/neco.2006.18.9.2146

Schemmel, J., Bruderle, D., Grubl, A., Hock, M., Meier, K., and Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. *IEEE Int. Symp. Circuits Syst.* 1947–1950. doi: 10.1109/iscas.2010.5536970

Schoppa, N. E. (2006). Synchronization of olfactory bulb mitral cells by precisely timed inhibitory inputs. *Neuron* 49, 271–283. doi: 10.1016/j.neuron.2005.11.038

Serrano, E., Nowotny, T., Levi, R., Smith, B. H., and Huerta, R. (2013). Gain control network conditions in early sensory coding. *PLoS Comput. Biol.* 9:e1003133. doi: 10.1371/journal.pcbi.1003133

Skeel, R. D. (1986). Construction of variable-stepsize multistep formulas. *Math. Comput.* 47, 503–510. doi: 10.1090/S0025-5718-1986-0856699-X

van Albada, S. J., Helias, M., and Diesmann, M. (2015). Scalability of asynchronous networks is limited by one-to-one mapping between effective connectivity and correlations. *PLoS Comput. Biol.* 11:e1004490. doi: 10.1371/journal.pcbi.1004490

Van Rossum, M. C. W. (2001). A novel spike distance. *Neural Comput.* 13, 751–763. doi: 10.1162/089976601300014321

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

APPENDIX A

LIF, AdEx and HH models are here described in full detail. They have been configured in such a way that their I-F curves are similar and generate similar activity levels at the neural system. Thus, we can properly compare their results over our four tests.

Leaky integrated-and-fire model (LIF)

The neural state is defined by the membrane potential (*V*), which is expressed by (Eq.1):

$$C\frac{dV}{dt} = -g_L(V - E_L) + I \tag{1}$$

where *C* denotes the membrane capacitance, g_L the conductance responsible for the passive decay term towards the resting potential, E_L the resting potential and *I* the synaptic interaction (defined in section d). LIF model (Gerstner and Kistler 2002) has a firing threshold V_T. Once *V* reaches this threshold, an output spikes is generated and the membrane potential is reset to E_L. The membrane potential remains being E_L for the whole refractory period T_{ref} (all these parameters are included in Table 7).

Adaptive exponential integrated-and-fire model (AdEx)

The neural state is defined by the membrane potential (*V*) and the adaptation variable (*w*), which are expressed by (Eq. 2):

$$\begin{aligned} C\frac{dV}{dt} &= -g_L(V - E_L) + g_L \Delta_T e^{(\frac{V - V_T}{\Delta_T})} + I - w \\ \tau_w \frac{dw}{dt} &= a(V - E_L) - w \end{aligned} \tag{2}$$

where *C* denotes the membrane capacitance, g_L the conductance responsible for the passive decay term towards the resting potential, E_L the resting potential and Δ_T the threshold slope factor. V_T is the effective threshold potential, *I* the synaptic interaction (defined in section d), τ_w the adaptation time constant and *a* the sub threshold adaptation conductance. AdEx model (Brette and Gerstner 2005) has a reset threshold at zero mV. When *V* exceeds this threshold, an output spikes is generated and the state variables are set to $V=V_r$ and $w=w+b$. V_r denotes the reset potential and *b* the spike triggered adaptation mechanism (all this parameters are included in Table 7).

Hodgkin-Huxley model (HH)

The neural state is defined by the membrane potential (*V*) and three dimensionless gating variables (*m*), (*h*) and (*n*) that take values between 0 and 1. These variables are associated with the sodium channel activation, the sodium channel inactivation and the potassium channel activation respectively (Eq. 3):

$$\begin{aligned} C\frac{dV}{dt} &= -g_L(V - E_L) - \bar{g}_{Na} m^3 h (V - E_{Na}) - \bar{g}_K n^4 (V - E_K) + I \\ \frac{dm}{dt} &= \alpha_m(V)(1 - m) - \beta_m(V) m \\ \frac{dh}{dt} &= \alpha_h(V)(1 - h) - \beta_h(V) h \\ \frac{dn}{dt} &= \alpha_n(V)(1 - n) - \beta_n(V) n \end{aligned} \tag{3}$$

where *C* denotes the membrane capacitance, g_L the conductance responsible for the passive decay term towards the resting potential and E_L the resting potential. $\bar{g}_{NA}$ and $\bar{g}_K$ are the maximum value of the sodium and potassium conductances and *I* the synaptic interaction (defined in section d) (all these parameters are included in Table 7). HH model equations (Hodgkin and Huxley 1952) do not inherently implement a firing threshold. Conversely, EDLUT needs an eventual threshold for the membrane potential as to predict a spike generation. Setting a functional firing threshold (-30 mV) allows us to simulate HH models using EDLUT computational kernel. α_i and β_i are rate constants for the i^th ion channel which is only voltage-dependent (Eq. 4). α_i and β_i are pre-compiled and stored in a look-up table for time-driven neural models running in CPUs. Pre-compiling α_i and β_i parameters in look-up tables is not convenient for GPU neural models since the RAM memory access is neither coalescent nor constant for all neurons at the same time.

$$\begin{aligned} \alpha_m &= 0{,}32\frac{13 - V + V_T}{e^{\frac{(13 - V - V_T)}{4}} - 1}; & \beta_m &= 0{,}28\frac{V - V_T - 40}{e^{\frac{(V - V_T - 40)}{5}} - 1} \\ \alpha_h &= 0{,}128\ e^{\frac{(17 - V + V_T)}{18}}; & \beta_h &= \frac{4}{1 + e^{\frac{(40 - V + V_T)}{5}}} \\ \alpha_n &= 0{,}032\frac{15 - V + V_T}{e^{\frac{15 - V + V_T}{5}} - 1}; & \beta_n &= 0{,}5e^{\frac{10 - V + V_T}{40}} \end{aligned} \tag{4}$$

where V_T adjusts the threshold potential.

Synapse properties

All these models hold the same synaptic interaction mechanism based on excitatory and inhibitory conductances. Equation 5 defines the input current injected to each neuron and the exponential decay functions for both conductances:

$$\begin{aligned} I &= -g_{AMPA}(t)(V - E_{AMPA}) - g_{GABA}(t)(V - E_{GABA}) \\ g_{AMPA}(t_1) &= g_{AMPA}(t_0) e^{\frac{t_0 - t_1}{\tau_{AMPA}}} \\ g_{GABA}(t_1) &= g_{GABA}(t_0) e^{\frac{t_0 - t_1}{\tau_{GABA}}} \end{aligned} \tag{5}$$

where *V* represents the membrane potential and E_{AMPA} and E_{GABA} the reversal potential of the excitatory and inhibitory synaptic conductances. g_{AMPA} and g_{GABA} stands for the excitatory and inhibitory conductances that integrate all the contributions received through individual synapses. Both conductances are defined as exponential decay function where τ_{AMPA} and τ_{GABA} represent the exponential decay time constants, t_0 the last ending time and t_1 the current time (all these parameters are included in Table VII).

Whilst the LIF model includes the refractory period, the HH model includes the depolarization and hyperpolarization periods. The maximum operating frequencies for these models are constrained by these time periods. Conversely, the maximum operating frequency for an AdEx model depends on several parameters. Setting the AdEx excitatory conductance to a maximum of 30 nS, limits its maximum operating frequency without modifying the AdEx internal dynamics.

APPENDIX B

One arising question derived from the simulation methods here proposed is the possibility of combining event-driven and time-driven integration methods in the same simulation. The EDLUT simulator allows us to not only evaluate the accuracy of these simulation methods individually but also the accuracy obtained when they are combined. This appendix compares the simulation accuracy results obtained when combining event- and time-driven methods with the results obtained when event-driven and time-driven methods are used separately.

To perform this comparison, a “control neural network” was first defined to be used as an initial reference. This control neural network was the original benchmark configuration (see section 2.4.1 Simulation parameter analysis) in which the second and third neural layers were computed by either event-driven methods only or time-driven methods only. We compared the “control benchmark configuration” with two “combined” neural configurations for the three neural models described (LIF, AdEx and HH):

- A first configuration in which the second neural layer was computed by event-driven methods and the third neural layer was computed by time-driven methods.
- A second configuration in which the second neural layer was computed by time-driven methods and the third neural layer was computed by event-driven methods.

For the sake of simplicity, only the direct method was taken into account amongst the event-driven integration methods. The combined and synchronous event-driven methods presented quite similar behaviors in terms of accuracy. Each neural model used two different look-up table sizes taken from Figure 4: LIF: 29 MB worst and 249 MB best case scenario, AdEx: 81 MB worst and 712 MB best case scenario and HH: 303 MB worst and 1195 MB best case scenario.

Similarly, only the fixed-step and bi-fixed-step integration methods in CPU were taken into account amongst time-driven integration methods. The GPU time-driven methods presented quite similar behaviors in terms of accuracy. Each neural model was simulated using different integration step sizes [0 - 1 ms]. The local step size for bi-fixed-step integration methods was 0.25 ms for both LIF and AdEx models, and 1/15 ms for HH model.

The uncombined simulation methods define a region (Fig. 1, left column) delimited by the worst and the best accuracy results obtained when simulating our “control neural network”. As expected, the results obtained for the combined simulation methods are mostly included in this region. The combined simulations present comparable accuracy results with respect to the uncombined event-driven or time-driven simulation methods. As shown in Figure 1, when the integration step size tends to zero, the accuracy obtained with time-driven methods improves, whilst event-driven methods provide worse accuracy. Combined methods obtain accuracy results right in between these two extremes; not as good as time-driven methods but not as bad as event-driven methods. Likewise, when the simulation step size becomes larger (1ms), the event-driven method accuracy remains better than the time-driven method accuracy. The combined methods, again, obtain accuracy results approximately in between these two methods.

Figure 1 also reveals a fact that ought to be taken into consideration when combining event- and time-driven methods. The sequence in combining event- and time-driven methods for simulating consecutive neural layers matters. The commutative property cannot be applied. The accuracy obtained with the event- and time-driven network configuration and the time- and event-driven network configuration differs. The event- and time-driven combination always presents higher van Rossum distances (worse accuracy) than the time- and event-driven combination. This is caused by two main factors:

- Worse accuracy is usually obtained by event-driven methods compared to time-driven methods in equal conditions. A second neural layer using event-driven methods delivers larger accuracy errors into the third neural layer. These accuracy errors are then propagated and increased in the third neural layer.

- Event-driven methods, in the second neural layer, generate spikes asynchronously, that is, with a high time resolution. However, time-driven methods, in the third neural layer, process all the input spikes synchronously at each global integration step time. That is, all the inputs spikes are computed in time slots of multiple of the global integration step. The time resolution obtained by the second neural layer is diminished by the third neural layer.

The benchmark used here, due to its particular features (i.e. connectivity ratios, firing activity, etc.), does not offer the best deal for event- and time-driven combined methods in terms of performance. On the contrary, as demonstrated in (Naveros et al. 2015), some cerebellar-based benchmarks are able to use event- and time-driven combined methods to improve computational performance without compromising accuracy. The optimal approach is to adopt an event-driven scheme for sparse activity neurons and a time-driven scheme for intensive activity neurons. As argued throughout this paper, choosing time-driven only, even-driven only or event- and time-driven combined methods is always a tradeoff between accuracy, computational performance and the peculiarities of the neural network to be computed.

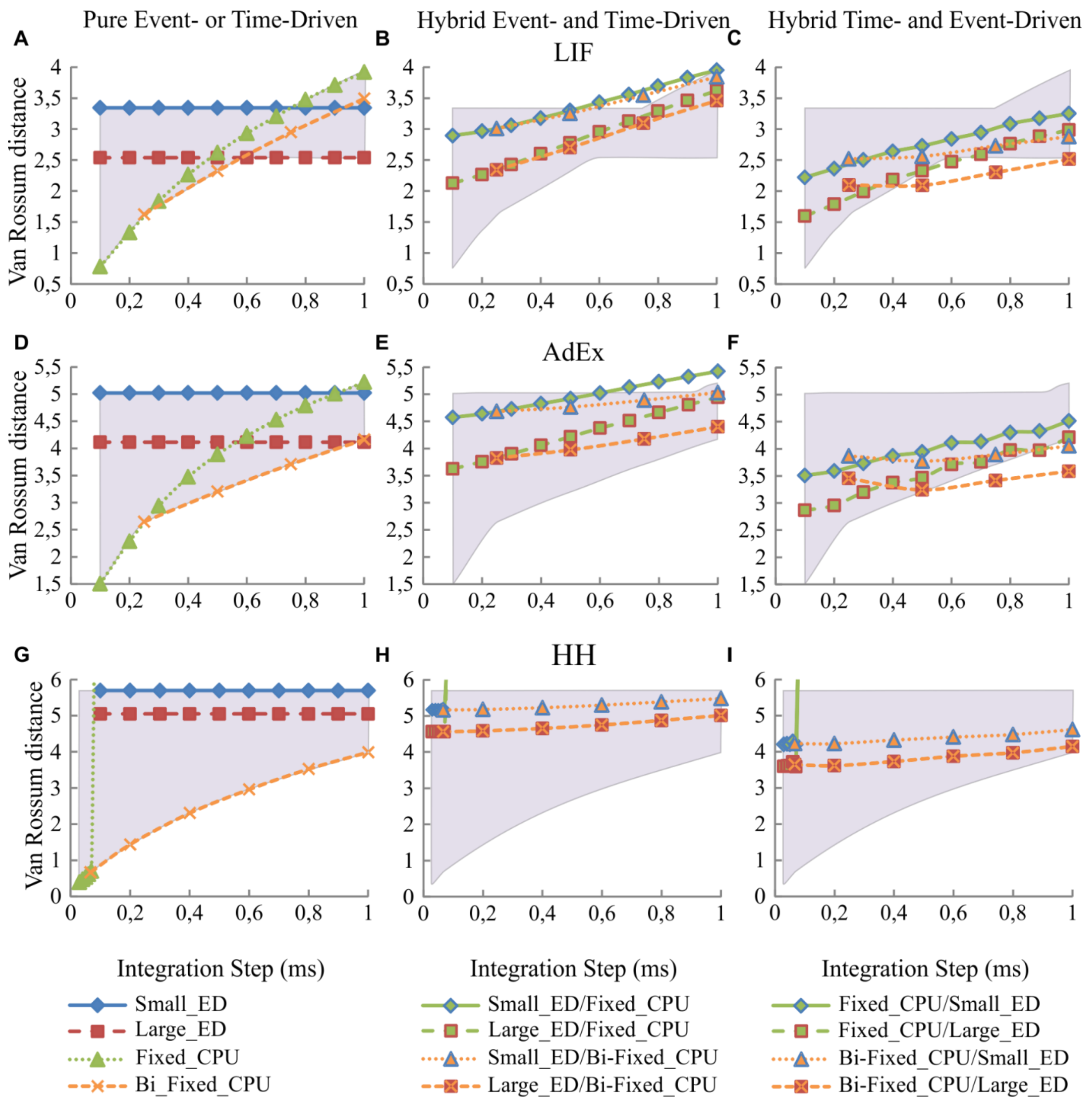


FIGURE 1 | Simulation accuracy of uncombined event- and time-driven methods vs combined event- and time-driven methods for LIF, AdEx and HH neural models. A 1-s simulation of the neural networks defined in **Table 3** is shown. Five different input spike patterns of 5 Hz that generate mean firing rate activities of 10 Hz in the third neural layer are used. The left-hand column (**A,D,** and **G**) of the panel shows the simulation accuracy when either event-driven methods only or time-driven methods only are used in the second and third neural layer. The central column (**B,E,** and **H**) shows the accuracy results obtained when combining event-driven methods for the second neural layer and time-driven methods for the third neural layer. The right-hand column (**C,F,** and **I**) shows the accuracy results obtained when combining time-driven methods for the second neural layer and event-driven methods for the third neural layer. Worst and best accuracy results obtained by the uncombined simulation methods (left column) define the boundaries of an accuracy area (grey shaded area). This area is compared with the accuracy results obtained with the combined simulation methods (central and right columns). The accuracy results obtained by these combined methods are expected to define a boundary area mostly included within the area defined by the uncombined simulation methods. The direct event-driven methods use two look-up table sizes per each neural model (LIF: 29 and 249 MB; AdEx: 81 and 712 MB; HH: 303 and 1,195 MB). The fixed and bi-fixed-step integration methods in CPU evaluate several integration step sizes for each neural model. The global integration step size for fixed and bi-fixed-step integration methods is plotted over x-axis. The standard deviation of the simulation accuracy obtained is negligible; we only represent the mean values.

Integrated neural and robotic simulations. Simulation of cerebellar neurobiological substrate for an object-oriented dynamic model abstraction process

Niceto R. Luque[ǂ1], Richard R. Carrillo[ǂ1], Francisco Naveros[1], Jesús .A. Garrido[2] and M.J. Sáez-Lara[3]

[1] *Department of Computer Architecture and Technology, University of Granada, Periodista D. Saucedo Aranda s/n, E-18071 Granada, Spain*
[2]*Brain Connectivity Center, IRCCS Istituto Neurologico Nazionale C. Mondino, Via Mondino 2, Pavia, I-27100, Italy*
[1] *Department of Biochemistry and Molecular Biology, University of Granada, CIBM, PTS, s/n, E-18071, Granada, Spain*

ǂ these authors contributed equally to this work.

Highlights: ►We integrated EDLUT neural simulator within a simulated robotic environment. ►As an embodiment example, we implemented a cerebelar-like structure controlling a simulated arm. ►The neural robotic simulator combines signals in analog/spike domains. ►Neural simulator, interface, and robotic platform operate conjointly in real time.

Keywords: neurobotics, cerebellum, spiking neural network, close-loop simulation, embodied neuroscience.

Acknowledgments: This work was supported by the EU grants REALNET FP7-ICT270434 (in which the cerebellar simulations were developed) and HBP FP7 Flagship Project 604102 (in which interface libraries are being developed). And by the national grant ARC-VISION (TEC2010-15396).

List of Abbreviations: CNS Central Nervous System, PF parallel fiber, MF mossy fiber, CF climbing fiber, GC granule cell, GoC Golgi cell, PC Purkinje cell, DCN deep cerebellar nuclei, IO inferior olive, UBC the unipolar brush cell, MLI molecular layer interneuron, MAE mean average error.

Abstract

Experimental studies of the Central Nervous System (CNS) at multiple organization levels aim at understanding how information is represented and processed by the brain's neurobiological substrate. The information processed within different neural subsystems is *neurocomputed* using distributed and dynamic patterns of neural activity. These emerging patterns can be hardly understood by merely taking into account individual cell activities. Studying how these patterns are elicited in the CNS under specific behavioral tasks has become a groundbreaking research topic in system neuroscience. This methodology of synthetic behavioral experimentation is also motivated by the concept of embodied neuroscience, according to which the primary goal of the CNS is to solve/facilitate the body–environment interaction.

With the aim to bridge the gap between system neuroscience and biological control, this paper presents how the CNS neural structures can be connected/integrated within a body agent; in particular, an efficient neural simulator based on EDLUT has been integrated within a simulated robotic environment to facilitate the implementation of object manipulating closed loop experiments (action–perception loop). This kind of experiment allows the study of the neural abstraction process of dynamic models that occurs within our neural structures when manipulating objects.

The neural simulator, communication interfaces, and a robot platform have been efficiently integrated enabling real time simulations. The cerebellum is thought to play a crucial role in human-body interaction with a primary function related to motor control which makes it the perfect candidate to start building an embodied nervous system as illustrated in the simulations performed in this work.

1. Introduction

Computational models of various brain regions have been developed and studied for more than thirty years in order to analyze central brain functions. Computational neuroscience (CN) is the natural complement of experimental brain research, since it focuses on specific mechanisms and models which are only partially observed in anatomical or physiological studies. In particular, the cerebro-cerebellar loop has been extensively modeled since Marr and Albus [1,2], providing elegant explanations on how the forward controller operation of the cerebro-cerebellar loop seems to work. Nevertheless, these computational theories tend to focus on one part of the cerebellar circuitry and then, to extrapolate the obtained conclusions to the whole cerebro-cerebellar system. Simulating nervous systems "connected" to a body (agent or robot with sensors and actuators) is of interest for studying how certain capabilities of the nervous system (e.g. the role of the cerebellum in coordinated movements and object manipulation) are based on cellular characteristics, nervous system topology, or local synaptic adaptation mechanisms. This represents an integrative approach which aims to build the bridge between task specific experimentation (equivalent to "awake animal testing") and system neuroscience models.

This integrative approach allows us to study the role of certain nervous systems within "behavioral tasks" [3]. For this purpose, it is crucial to study nervous system models within the framework of their interaction with a body (sensors and actuators) and the environment.

This paper describes an integrated approach to the cerebellar circuit modeling within real time "behavioral tasks". The paper describes briefly: (a) a cerebellar model based on point neurons capable of being simulated in real-time. The model maintains biological interconnectivity ratios in functional medium-scale networks (rather than an ad-hoc neural network particularly designed for a specific behavioral task) that are embedded in biologically plausible control loops. (b) Testing the role of plasticity at parallel fibers-Purkinje cells. (c) Embedding the neural system model into a cerebro-cerebellar control loop connected to a Light Weight Robot (LWR) performing repetitive fast manipulations along benchmark trajectories. In order to address these three aims, we have integrated a neural simulator based on EDLUT [4] with a simulated robotic environment to facilitate the implementation of object-manipulating closed-loop experiments (action–perception closed loops).

These experiments allow us to study the neural abstraction process of dynamic models (of objects being manipulated) that occurs within our neural structures in fast manipulation tasks [5–7]. The neural simulator, the communication interface, and the simulated robotic platform have been developed and integrated taking into account computational efficiency as a major requirement in order to enable real time simulations. This platform allows us to study different neural representation and processing schemes in a specific task within a brain–body interaction framework.

1.1. Functional cerebellar models; a brief overview

Among Embodied System Neuroscience models, the well-organized structure of the cerebellum has received special attention from researchers belonging to very different fields. On one hand, neurophysiologists have studied and proposed detailed models and descriptions according to experimentally recorded cells and synaptic properties. However, it is not yet clear how specific properties of these current detailed models facilitate specific tasks at a behavioral level. On the other hand, engineers have proposed artificial approaches (only related with biology at a very high level) for biologically relevant tasks such as accurate and coordinated movements. Based on these opposed approaches, several cerebellar modeling frameworks have been proposed:

In state-generator models, the granule cell layer presents on/off type "granule" entities that provide a sparse coding of the state space (Marr–Albus Model [1,2], CMAC [8–10] model, or Yamazaki and Tanaka model [11–14]). These models succeed in explaining some traditional cerebellum-involving tasks such as eyelid conditioning [15] or motor control tasks [6,7,16]. In functional models, only the functional abstraction of specific cerebellar operations is considered: MPFIM model [17], Adaptive Filter model [18–22], APG model [23], or LWPR model [24,25]. Although in some cases, these models are also used to explain how the cerebellum works, these can be seen as problem solving approaches (that use internal structures not constrained to biologically plausible features). These functional models are also used to study the potential role of the cerebellum in tasks such as eyelid conditioning, the vestibule ocular reflex (VOR), or movement correction [24,25]. Finally, cellular level models capture the biophysical features of the cerebellar neuronal topology and processing, and can be evaluated in the framework of neurophysiological experiments. These models aim to be as biologically plausible as possible. But due to their inherent complexity, their application in the context of large-scale cerebellar modeling and computation remains limited. The very first approximations in this field were developed based on the simplified models of Schweighofer–Arbib [26,27].

1.2. How to embody the cerebellar circuitry

The cerebellar network has been at the core of neurocomputational theories since the 1960s, when Eccles proposed the Beam Theory [28] and Marr and Albus, the Motor Learning Theory [1,2]. Later on, Ito developed the forward controller theory [4,29–32]. Since then, the view has been crystallized on two main concepts that can be synthesized as follows; the way the cerebellum operates is by decorrelating the inputs in the granular layer and detecting known patterns in Purkinje cells. Pattern recognition is regulated by memory storage at the parallel-fiber-Purkinje-cell synapse. When unfamiliar patterns are detected repeatedly, the Purkinje cells change their firing rate and regulate activity in the deep cerebellar nuclei (DCN), thereby emitting the corrective terms used for highly accurate motions (skillful control performance).

Despite its attractiveness and simplicity, this theory only partially accounts for the capabilities of the cerebellum. Furthermore, recent experimental data indicate that the cerebellar system is much more complex than initially stated. Just to make a very short survey, the mechanisms of the granular layer go far beyond simple decorrelation [33], long-term synaptic plasticity does not occur only at the parallel fibers (PF) [33–35], the inferior olive (IO) operates as a complex timing system and not simply to drive Purkinje cell plasticity [36], the Purkinje cells and the DCN cells have operative states that go far beyond the concept of firing rate regulation [37]. The core idea is that our knowledge on the functioning of neuronal networks of the cerebellum is still rather vague, and that we have to develop new computational tools to investigate cerebellar network dynamics beyond the current existing paradigms.

The available neurophysiological data (which is essential for understanding the functional organization of the cerebellum and related structures) has to be analyzed to investigate the particular processing capabilities of each neuron and of its internal dynamics. Emphasis must be put on proving how the network processing capabilities are supported by the low-level characteristics of each neuron type. Many of the specific cerebellar neural types have already been implemented in Python–NEURON–EDLUT software simulators [38,39] and there are even specific repositories gathering different kinds of models [40,41].

1.3. Modeling the cerebellar circuits

When modeling the cerebellar circuit with a bottom-up approach, the cerebellar network needs to be modeled aiming at the construction and generation of a complete cerebellar functional network, tested in realistic functional conditions and endowed with plasticity rules. This process demands the comprehension of the interplay that occurs between the Granular-and-molecular-layer subcircuit and the PC–DCN–IO subcircuit.

Whilst the granular layer and molecular layer neurons can be largely reconstructed starting from precise existing models, the DCN-and-IO subcircuits are not modeled in detail. Therefore, the PC–DCN–IO circuit requires basic modeling to achieve functional properties. An initial model of the DCN can be constructed based on [42,43]. As a starting point, the IO can be modeled at a functional level, i.e. as a module translating “error related signals” into activity that modulates learning at the PF–PC synapses. Also at this stage, although different plasticity sites have been reported [34,35], most cerebellar functional models are based solely on the PF–PC adaptation mechanism modulated by the IO activity (which delivers the teaching signals).

Once all the subcircuits and long-term synaptic plasticity are implemented and tested separately, the functional operation of a complete circuit can be tested. The first step lies on developing an appropriate connectivity between the modular subcircuits. The connection map between the IO and PCs via climbing fibers, the convergence of PCs to DCN neurons [44] and the mossy fiber (MF) projections to the DCN [45], and the granular layer have been extensively described in the literature and should be reconstructed respecting the known convergence/divergence ratios.

2. Material and methods

2.1. *The real-time neural simulator: EDLUT*

Common event-driven simulators [46,47] use simple neural models whose dynamics are described by equations which can be discontinuously evaluated at arbitrary times (e.g. current based integrate-and-fire models). But even when using simple neural models, the firing-time prediction which is necessary for an event-driven simulation may be complex [48,49].

An EDLUT (Event-Driven neural simulator based on LookUp Tables) was implemented [4] to simulate neural models whose internal dynamics is defined by a set of differential equations (for instance, the Hodgkin and Huxley model [50]) adopting an event-driven simulation scheme. This software is an open source project [51] for efficient simulation of biological neural networks. It is of particular interest in the field of neurobotics and embedded neural computing in which real-time processing is required, for example, for experiments which include perception–action loops.

EDLUT uses an intensive preliminary simulation stage in which a neural model is characterized, i.e. massive simulations of a single cell are done with different initial conditions. At this stage, samples of the neural variables at different times are stored in lookup tables. This preliminary stage can be seen as a cell model compilation stage. These tables are calculated using time-consuming numerical analysis (e.g. Runge-Kutta method). However, once they are generated, the network simulation can be run efficiently through the event-driven method, just by accessing tables when the neural state must be updated or predicted.

EDLUT uses lookup tables which store all the possible values (with certain precision) of the neural-model state variables [52] in addition to the future states (firing times) [53]. Therefore, a whole neural model is encoded in each set of model-characterization tables. In this way, the simulator takes advantage of the increasing memory resources available to perform efficient simulations with very limited computation requirements. The event-driven simulation scheme based on lookup tables uses memory access intensively, instead of CPU computation power for the neural variable updates.

The initial EDLUT processing scheme allowed fast simulation of complex neural models. Nonetheless, this scheme is constrained by the number of state variables of a neural model because this determines the number of dimensions of the required lookup tables. But in later versions [54], the EDLUT was upgraded to provide a hybrid time-and-event driven simulation method. This hybrid scheme allows the concurrent simulation of some neuronal models using the event-driven method (the models which can be translated into lookup tables) and other models using the time-driven method in the same network.

2.2. *The cerebellar model*

We have used leaky integrate-and-fire neural models (LIF) [50] whose synapses are modeled as conductances. The general model has been then adapted for different neural types. The LIF neural state is characterized by the membrane potential (V_{m-c}) expressed by Eq. (1):

$$\begin{aligned} C_m \frac{dV_{m-c}}{dt} &= g_{AMPA}(t)(E_{AMPA} - V_{m-c}) + \\ & g_{GABA}(t)(E_{GABA} - V_{m-c}) + G_{rest}(E_{rest} - V_{m-c}) \end{aligned} \tag{1}$$

where Cm stands for the membrane capacitance, E_{AMPA} and E_{GABA} denote the reversal potentials of the synaptic conductances, and finally, E_{rest} represents the resting potential (being G_{rest} the conductance responsible for the passive decay term towards the resting potential). The g_{AMPA} and g_{GABA} conductances integrate all the contributions received through individual synapses and are defined as decaying exponential functions. The parameters of the neural model [5–7] and a more detailed description can be found in [5–7,51].

Therefore, the state of a neuron is defined with just three variables:

V_{m-c} represents the membrane potential. When this variable reaches a specific threshold, the neuron generates an output spike.

g_{AMPA} and g_{GABA} represent excitatory and inhibitory conductances respectively that affect the membrane potential. These conductances decrease exponentially in each integration step and increase proportionally to the synaptic weight of their connections when an input spike arrives.

To solve the LIF neuron model differential equation, the EDLUT simulator incorporates different integrative methods. This differential equation is processed off-line using a short integration step to achieve good accuracy (it does not directly affect the computation time during system neural simulations, since the neural model is computed and stored in lookup tables in a preliminary neural characterization stage).

All the different characterized neural types have been interconnected following a cerebellar topology structured into micro-zones distributed in different layers, as described below (Fig. 1):

Mossy fibers (MFs) (248). These mossy fibers drive the contextual information and sensory joint information (related with the manipulated object and desired/actual positions and velocities). The mossy fiber model is based on leaky integrate-and-fire neuron dynamics whose input current is provided by a set of overlapping receptive fields covering the joint value space of the input signals (see Fig. 2).

Granular layer (GCs) (1500). This layer behaves as an abstraction of a simplified cerebellar granular layer. The information provided by the mossy fibers is translated into a sparse representation. Each granular cell (GC) receives four excitatory input connections; three connections randomly chosen from joint-related mossy fibers and the other one, from a context-related mossy fiber [7].

Parallel fibers (PFs) (1500). They represent the output axons of the granular layer. The manipulated object model abstraction is stored in learned weights at the PF–PC connections.

Climbing fibers (CFs) (48). The climbing fibers are the axons of the Inferior Olive cells. This layer consists of 6 groups of 8 climbing fibers each. The IO output (encoding a teaching signal related to the error) is translated into spikes using leaky integrate-and-fire neuron dynamics whose input current is in this case proportional to

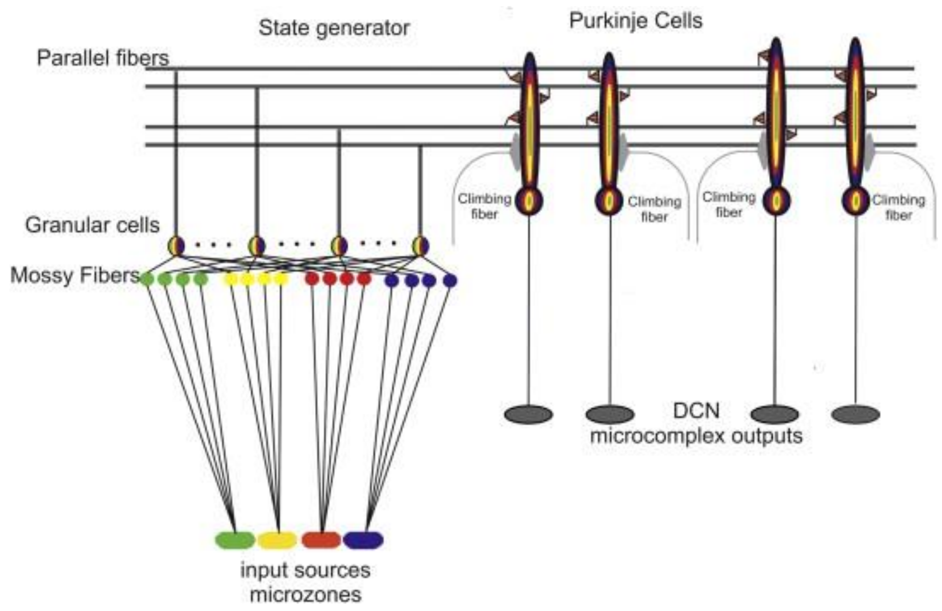


Fig. 1. Cerebellar architecture. Color representation indicates signals from different sources such as different cuneate receptive fields or proprioceptors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

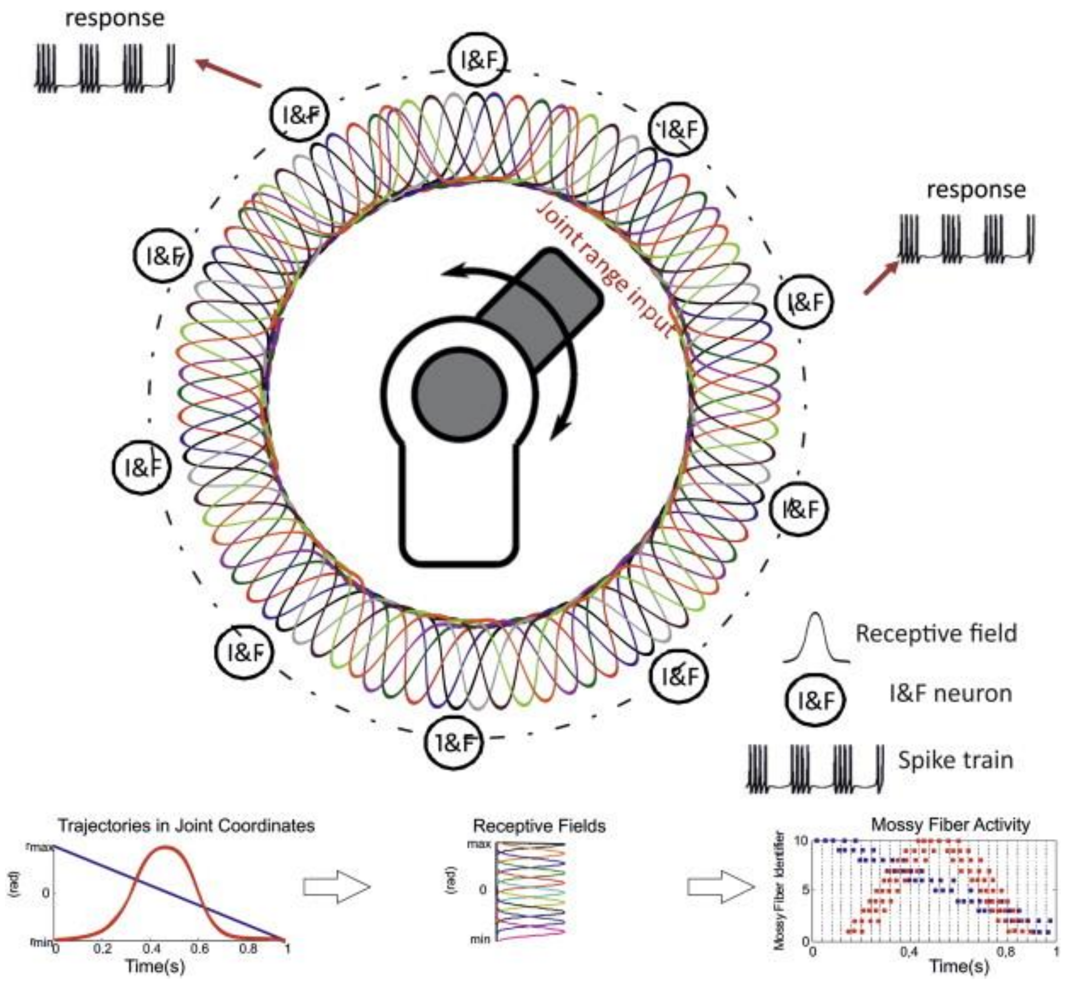


Fig. 2. Population coding of input (proprioceptors) signals. The joint angle position (input signal) provided by a joint encoder which covers the joint range is translated into a population coding whilst a certain trajectory is followed using a set of tuning receptive fields (Gaussian like curves) which represent the current injected into spiking neurons by different sensory receptors (proprioceptors). Each proprioceptor's value (output signal) is integrated using an integrate-and-fire neuron model and determines the activity response of an input neuron (as is the case of Mossy Fiber neurons belonging to the cerebellar circuitry). Lower plots illustrate how two trajectories (encoder angle varying in time) defined in a single joint produce spiking patterns when the contributions to the integrate neurons are integrated through the sensory receptive fields.

the error signal. The CFs drive the IO outputs to the Purkinje cells for supervised learning at PF–PC connections. More details on this learning rule can be found in [6].

Purkinje cells (PC) (48). These cells are divided into 6 groups of 8 cells. Each GC is connected to 80% of the PCs which are also receiving their corresponding teaching signals from the CFs.

Deep cerebellar nuclei cells (DCN) (24). The cerebellar output is generated using 6 groups of these cells (2 groups per joint) whose activity is capable of providing corrective torques for a specified cerebellar input. Corrective torque values per joint are encoded by a couple of these groups, one group compensating positive errors (agonist) whilst the other one is dedicated to compensate negative errors (antagonist). Each DCN neuron group receives excitation from every MF cell and inhibition from the two corresponding PCs. The sub-circuit PC–DCN–IO then is organized into six microzones; three of them generating joint positive corrections (one per joint) and the other three, generating joint negative corrections (one per joint). Mind that as it will be explained below, we use three joints in our robot experiments.

2.3. *The cerebellar model*

Neural population coding is traditionally used for sensorimotor representation. Each neuron belonging to a certain system presents a distribution of responses over some set of inputs. Hence, the response of many system neurons over a set of certain inputs represents the system state [55,56]. In a reaching movement, the arm direction is encoded by means of neurons whose input current changes with the cosine of the difference between the stimulus angle and the preferred direction of the cell [57] (Cosine tuning). Each cell has a preferred direction and receives input current depending on how a movement is aligned to its preferred feature. However, a simple reaching movement involves extracting spatial information including visual acquisition of the target, coordination of multi-modal proprioceptive signals, and a proper motor command generation to drive a proper motor response towards the target [58].Common reaching movements towards a target that we have already seen involve an internal representation of the target and limb positions, and also a coordinate transformation between different internal reference frames. A spiking population coding is used as internal representation and can be adapted as indicated below to be embedded into a control loop.

The integration of computational models with neurophysiological observations in order to understand the main problems in motor control requires not only the cerebellum functionality to be considered but also its biological architecture (cell-network topology). This requires the development of two "translation processes" in order to interact with a robot agent: (1) Translation from analog domain sensor inputs to spike based patterns compatible with a spiking cerebellar network. (2) Translation from spike domain cerebellar outputs to analog domain actuator commands to be delivered to the robot agent.

2.3.1. From sensors to spikes

When a target reaching movement is executed, different body parts, such as muscles, tendons, or joints are articulated depending on their body location [59] along the followed trajectory. Sensory proprioceptors are activated according to the movement; thus, a time-varying set of stimuli is produced, and its corresponding neural population varying activity is generated. In contrast, in a robot scenario, the only available proprioception sensory information is supplied by an encoder output per link. Hence, a translation from the joint position/velocity measures to a time-varying set of stimuli is required. At this point, finding out an optimal biologically plausible encoding scheme that allows "biological decoders" (as the ones we assume at the granular and molecular layers of the cerebellum) to take advantage of the representation is a non-trivial issue. It is assumed that the firing rate of an individual sensory receptor follows a neural response which is characterized by Eq. (2) (also equivalent to a cosine tuning curve, that is, the firing rate of the neurons varies with the angle between the preferred direction of the sensory receptors and the sensed position) [60]. Therefore, a reaching movement execution will be represented with a sparse population of active cells which are varying with time. This coding mechanism leads to a representation of the current sensorial state during the trajectory execution in an unambiguous way.

The output of each receptor is given by Eq. (2);

$$I_{Ni}(t) = r_0 + r_{max} \sum_n e^{-\left(\theta-\theta_{pref}-2\pi n\right)^2/2\sigma^2} \qquad (2)$$

Where $[r_0, r_{max}]$ is the joint range in radians, θ is the actual position, θ_{pref} is the preferred direction of the receptor, σ is the amplitude of the receptive field associated to the receptor, and finally, $2\pi n$ is a subtractive term used to refer the actual position to the first-360-degrees (the maximal range of any revolute joint is 360°).

Receptors are distributed along the range of each joint, being their receptive fields overlapped (as peripheral nerve receptive fields are). Each value of a proprioceptor output signal is integrated using an integrate-and-fire neuron model whose dynamics is defined in Eq. (3) (see illustration in Fig. 2). In the case of an arm system, this determines the output activity that drives the Cuneate Nucleus (CN) activity emulating the way the Mossy Fiber activity from cells in the CN handles information from forelimb muscle spindles [61].

$$\tau_{mi}\frac{dv_i}{dt} = -v_i(t) + R_i I_{Ni} \quad (3)$$

Related to the leakage integrate and fire cell dynamics, τ_{mi} is the resting time constant, v_i the membrane potential, I_{Ni} the input current, and R_i is related to the resting conductance of the membrane.

2.3.2. *From spikes to actuators, decoding the cerebellar output*

Spiking modeled neurons elicit pulsed signals usually named action potentials or spikes. It is believed that the shape of these spikes only carries minimal information whilst the core of the information is carried by the spike time arrival [62,63]. The action potential waveforms (voltage curve profile) elicited by those neurons is usually translated into a set of binary symbols (0 or 1) representing an instant in which an action potential occurs (1) or does not (0). The generated binary waveform conforms a spike train and the obtained pattern of spikes belonging to a certain time-frame generates the spike binary code; the columns corresponding to the array of spikes are also named *neural activation patterns*. It is then clear that, somehow, the translation of these neural activation patterns into meaningful analog output signals has to be implemented for interfacing actual robot actuators with analog signals.

Assuming that the goal is to decode rather than to analyze the behavior of biological neurons, it seems reasonable to use a mathematical approach such as linear filtering, particularly, a Finite Impulse Response filter (FIR), to accomplish this task [64].

Defining the spike train as $x(t) = \sum_{j=t}^{N} \delta(t - t_j)$ where t_j stands for the set of firing times of the corresponding neuron and being the FIR response defined as $h(t)$ then the stimulus can be written as follows.

$$stimulus(t) = (h * t)(t) = \sum_{j=t}^{N} h(t - t_j) \quad (4)$$

As noticed from Eq. (4), converting spike trains into analog signals is a quite straightforward implementation. Nevertheless, despite the widespread use of FIR filters for such purpose, an undesired delay is introduced in the generated analog signal. This delay is strongly related to the number of filter coefficients as well as to the shape of the filter kernel. To mitigate this effect and to make the conversion more efficient, an exponentially-decaying kernel can be implemented, as seen in Eq. (5). Thus, at each time step, the output signal value only depends on its previous value and on the input spikes in the same time step. Therefore, this filter can be implemented by recursively updating the last value of the output signal. Actually, the choice of such exponential kernel is double folded. The kernel is able to mitigate the delay problem and bears a strong resemblance to postsynaptic currents [62,63], thus facilitating a possible biological interpretation.

$$Kernel = h(t) = e^{-\frac{M}{\tau}} \quad (5)$$

Where M is the number of filter taps (one integration step size) and τ is the decay factor.

2.3.3. *Equivalency with an integrative neuron*

Integrative neurons are capable of both analyzing and interpreting sensory input just taking into account their actual state, the incoming information, and their previous states as well. Once the computation of those three elements is done, the resulting information can be transmitted to motor neurons or other integrative neurons. Assuming a leaky integrate-and-fire model for the integrative neuron, the model looks like Eq. (3). This model forces the input current to exceed a threshold $I_{th} = V_{th}/R_i$ for the cell i to fire; otherwise, it will simply leak out any charge in the membrane potential. The firing frequency is thus defined in Eq. (6):

$$f(I) = \left\{\begin{matrix} 0, & I_{Ni} \le I_{th} \\ \left[t_{ref} - \tau_{mi} log\left(1 - \frac{V_{th}}{I_{Ni}R_i}\right)\right]^{-1}, & I_{Ni} > I_{th} \end{matrix}\right\} \quad (6)$$

Where t_{ref} is a refractory period and τ_{mi} is the resting time constant. Solving the differential equation (3), the membrane potential is expressed as follows:

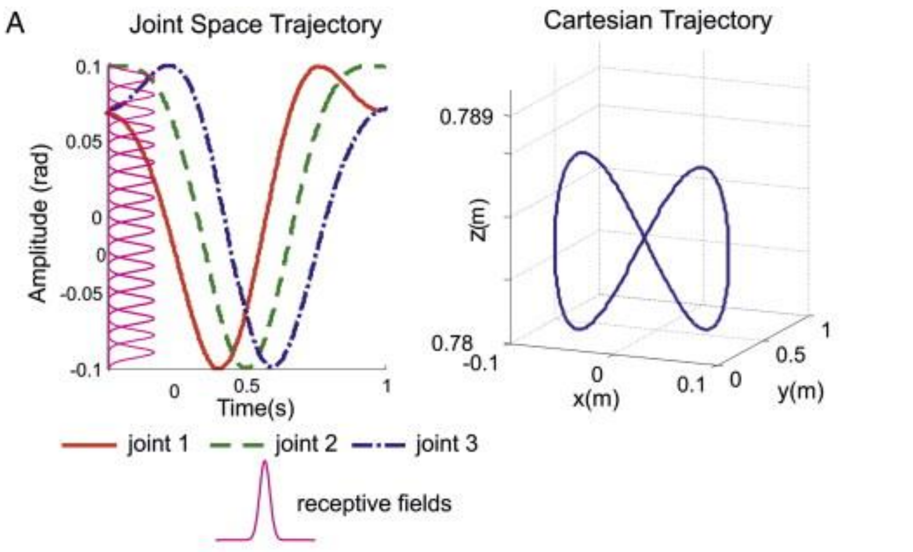
$$v_i(t) = R_i I_{Ni}(t) + \frac{(V_{Rest} - R_i I_{Ni}(t))}{e^{(t/\tau_{mi})}} \quad (7)$$

The functionality of the selected FIR described in Eqs. (4) and (5) can be read in terms of a biological interpretation just by making an analogy between the proposed exponential-decaying kernel and the behavior of an integrative neuron whose dynamics is defined using Eq. (7) (stimulus(t) ≈ V_j(t)). The resulting shapes of both sides of this analogy hold a remarkable resemblance due to the exponential-decaying kernel that governs both the neural dynamics and the FIR kernel. An engineering strategy usually adopts the FIR based approach, because it allows us to easily adapt the output values to the control signal which is demanded for accurate control. In such a way, the effect of each spike elicited by any cerebellar nuclei cell (output cerebellar cells) can be easily pondered thanks to the FIR filter, thus facilitating the correlation between the cerebellar output spikes and their corresponding corrective output signals. It is clear then that this conversion can be also processed by using any Integrate and Fire-like neuron; however, doing so, the influence of each spike on the output does not always remain clear.

2.4. Cerebellar control loop; a plausible implementation

It is widely assumed that the cerebellum, acting as a control module, is embedded in a feedforward control loop [65–67]. A feedforward control system is able to evaluate both the incoming sensory information from the environment and the information provided by the system itself (proprioception) before the motor control action is sent to the body. This means that the controller manages the sensory information to deliver the best motor commands to accomplish the desired movement. At that point, we must bear in mind that once a pure feedforward system sends the corresponding control actions, it is not possible to modify them.

On one hand, a feedforward control system is able to deliver the precise set of motor commands for the body-plant and to make corrections during the movement without continuously checking the motor control output [26,27]. Conversely, the feedforward controller requires a previous trial-and-error learning process in order to later recognize (in a recall stage) all the possible sensorial states that may be reached. In a real manipulation task, the environmental conditions are constantly changing and the feedforward controller must continuously tune its motor commands to cope with these changeable environmental conditions [68]. According to this scheme, the cerebellum operates as a feedforward controller for the motor commands which are originated in the motor cortex (Fig. 3). The brain is able to plan and learn the optimal trajectory of a movement in intrinsic coordinates [23,68–71]. This operation consists of three main tasks: the desired trajectory computation in external coordinates, the task-space translation into body coordinates, and the motor command generation [72]. In order to deal with the aforementioned changeable environmental conditions, the system needs to incorporate a Feedback-Error Learning (FEL) scheme [73] by means of the cerebellum operating in conjunction with a crude inverse dynamic model of the arm-plant [74]. It has been proposed that the association cortex provides the motor cortex with the desired trajectory in body coordinates. In the motor cortex, the motor command is calculated by using an inverse dynamic arm model (for a review, see [75]). *The spinocerebellum–magnocellular red nucleus system* provides an accurate model of musculoskeletal dynamics, which is learned with practice by sensing the motor command consequences in terms of executed movements (proprioception). The *cerebrocerebellum-parvocellular red nucleus system*, which projects back to the motor cortex, provides a crude inverse-dynamic model of the musculoskeletal system, which is acquired whilst monitoring the desired trajectory [73]. The crude inverse-dynamic model works together with the dynamic model provided by the cerebellum embedded in a feedforward control loop thus updating motor commands according to predictable errors occurring when executing a movement. It learns and stores models of the skeleto-muscular system providing the precise timing control of agonist–antagonist muscle pair groups in addition to the needed force and stiffness control [76]. Obviously, the muscle flexion–contraction precise timing and the needed force in a manipulation task depend on the weight to be



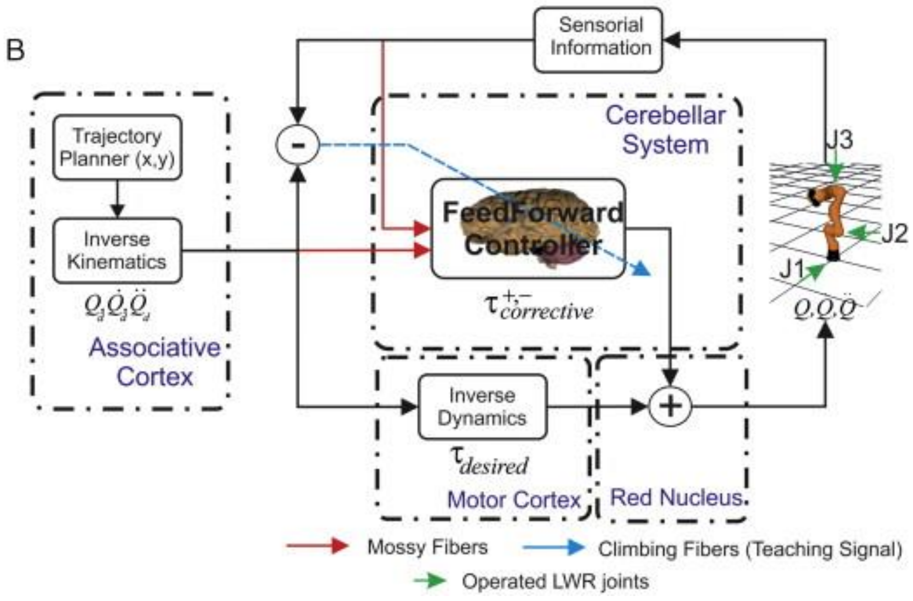


Fig. 3. (A) Benchmark trajectory to be performed consisting of sinusoidal components. The trajectory is shown in both joint coordinate and Cartesian coordinates (eight-like trajectory). The receptive fields are distributed covering the whole range determined by the joint coordinates. (B) Implemented cerebellar control loop. The cerebellum infers a corrective model that produces effective corrective commands in order to compensate the existing mismatch between the crude inverse dynamic robot model and the actual base dynamic plant model. The desired arm states are generated according to the Cartesian trajectory to be followed (positions (Qd), velocities ($\dot{Q}$d) and accelerations ($\ddot{Q}$d)) by the trajectory generator (a crude inverse kinematicmodel representing the output of the associative cortex and othermotor areas). These desired armstates in joint coordinates are used at each time step to compute desired torque commands (crude inverse dynamic robot model). They are also used as input to the cerebellum which produces the predictive corrective commands (τcorrective) which are added to these crude torque commands (τdesired). The final total torque addition is supplied to the robot plant. The difference between the actual robot trajectory and the desired one is used to calculate the climbing fiber activity which is supplied to the cerebellum as a teaching input signal (for adapting PF–PC synaptic weights).

handled (more concretely, on the dynamic model of the object under manipulation), the cerebellum being crucial for delivering this proper timing, force, and coordination; these appropriate corrective terms are learned through a trial-and-error process [68].

2.5. *Simulated robot integration: robot and training trajectory*

Behavioral experiments with an embodied cerebellar system require the integration of a real or simulated robot in the control loop. The simulated robot is intended to follow a specific trajectory whilst the cerebellar model learns to provide corrective torques for the robot actuators. The robot-control experiment results are intended to assess the effects on performance caused by concrete neural properties, cerebellar subcircuits, or adaptive mechanisms (synaptic plasticity). This robot-control experimentation demands human-like robots whose intrinsic dynamics is somewhat similar to their biological counterparts. This requirement motivates the use of lightweight robots (LWR) such as the Kuka lightweight robot developed by DLR [77,78].

As mentioned above, the main role of the cerebellum seems to be related to human motor control, especially in those tasks where timing and force are critical. Therefore, those manipulation tasks able to modify the dynamics of the arm-plant whilst performing certain movements would constitute the paradigm to follow. These LWR robots are capable of being dynamically modified when manipulating different payload contexts under certain kind of movements. This motivates the definition of a benchmark trajectory capable of revealing the dynamic properties of a LWR. According to the proposals in [76,79], fast movements in a smooth pursuit task consisting of vertical and horizontal sinusoidal components are good candidates in order to reveal the robot dynamics. Examples of different benchmark trajectories can be checked in [74,76,80]. Considerations related to the communication interface delay and the friction force of the robot joints need to be taken into account (see Appendix).

2.6. The integrated neurobotics simulation platform

These techniques are now included into an integrated software platform able to combine realistic robotic experiments (running in real time) with cerebellar like modules that work as corrective engines. This platform aims to facilitate the study of how the adaptive neural information coding mechanisms underlying the ability of humans to interact with their environment is handled by means of an effective adaptation at the cerebellum. The simulator of the robotic LWR arm, the control loop, and the cerebellar module were implemented in C/C++ following previous developments [5–7,24,25]. The software platform source code has been made available at: https://code.google.com/p/edlut/source/browse/branches/EDLUT_with_Robot.

The core of the neural simulator was implemented taking EDLUT [4] source code as the basis. EDLUT was then provided with an interface library as well as with a robot library able to dynamically define and model different lightweight robot configurations. In this work, we use a rough approximation of a Kuka LWR [77].

2.7. A practical running example

The aim of this working example is to show how a cerebellar model based on [7] within a “perception–action” closed-loop [5–7,74] is used in order to control a simulated LWR [77] arm by means of the developed software platform. Vertical and horizontal sinusoidal composed trajectory-following tasks [5–7,74] will be run in order to reveal the robot dynamics (see Fig. 3) with different payloads to be manipulated. The input pathways to the artificial cerebellum will be MFs and CFs. The cerebellar output is translated into torque commands for each joint through conversion modules [5–7], following the approach described in the previous section.

3. Results

Using this cerebellar architecture, a 1000 trial execution (each trial takes one second) following the principles already presented in [7] has been performed, obtaining the raster plot shown in Fig. 4. This figure represents a snapshot of one trial execution representing a cerebellar simulation of one second eight-like trajectory operating 3 revolute joints (joint 1, joint 2, and joint 3 indicated in Fig. 3) of a LWR defined in [5–7] when manipulating a 10 kg payload. This snapshot corresponds to two particular moments during the learning process; the initial learning stage (left column plots) and the final learning stage (right column plots). Mind that, as can be seen in Fig. 4, at the initial learning stage (0–1 s period), no cerebellar action has been learned yet (Fig. 4(E)), whilst at the final learning stage

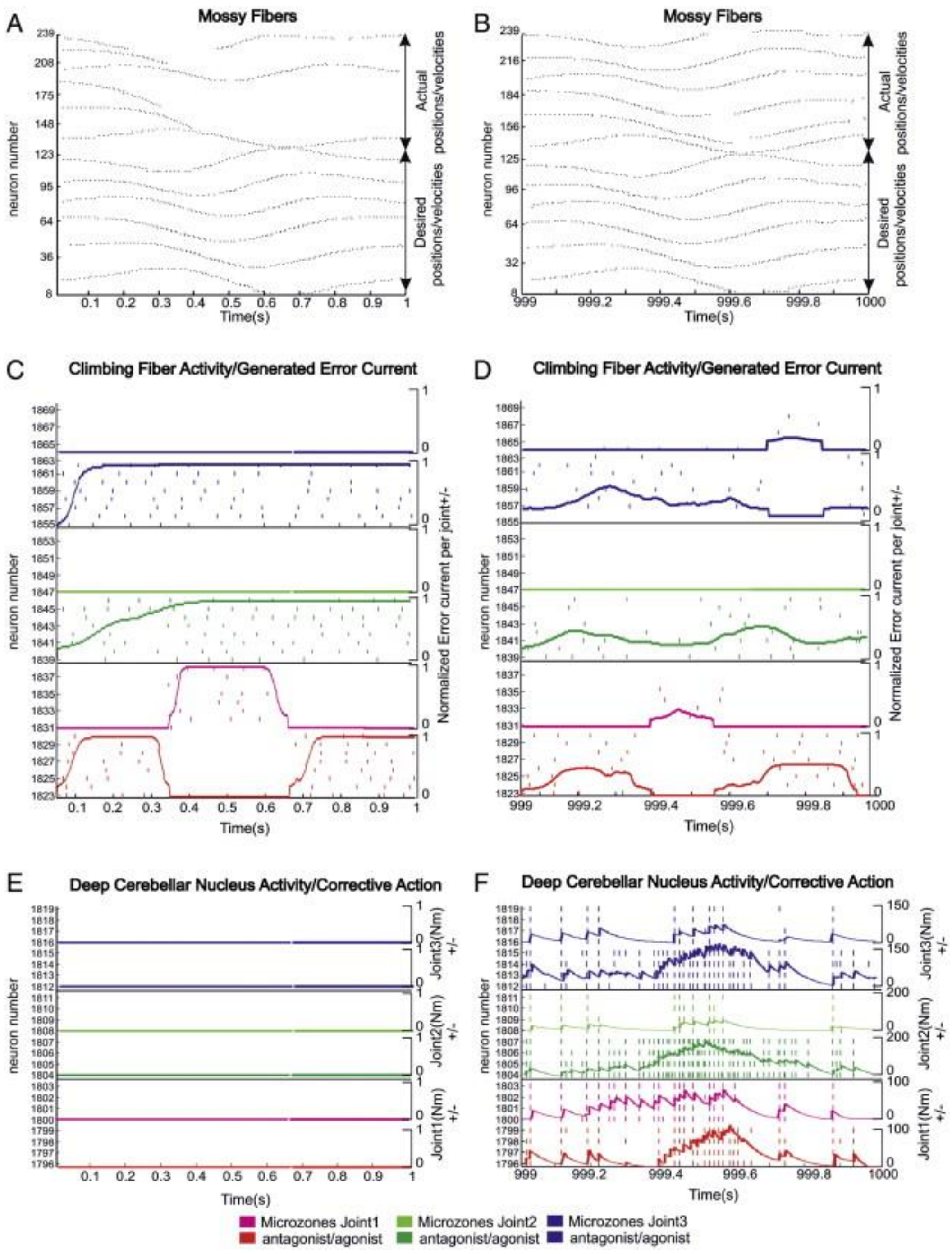


Fig. 4. Cerebellar activity monitoring one second simulation snapshot at the beginning of the learning process (left plots) and at the end of the learning process (right plots). Left Y axes are used for the neuron number in the network. The bottom legend indicates how these neurons are related to different joints and agonist or antagonist micro-complexes by using different colors. Plots C–F include two overlapped representations, the spike patterns related to the left Y axis and a continuous line referred to the right Y axis at each plot. (A) (B) Translation of the desired/actual joint positions/velocities into mossy fiber activity at the beginning of the learning process (A) and at the final learning stage (B). (C) (D) Evolution of the climbing fiber activity during the learning process and its corresponding error current proportional to the actual position and velocity error. (C) High error current translated into spikes at the initial learning stage. (D) Lower error current translated into spikes at the end of the learning process. (E) (F) Cerebellar output during the learning process and the corresponding generated analog corrective action. (E) Cerebellar output at the beginning of the learning process. No spikes are elicited at the DCNs, the corrective actions are zero. (F) Cerebellar output at the end of the learning process. The spike output activity is translated into corrective actions for each robot joint. Each couple of micro-complexes is related to a certain robot joint (agonist and antagonist terms). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

(999–1000 s period), the learning process is well settled down (Fig. 4(F)) and corrective terms are delivered through DCNs.

As we see, all plots represent activity along time using dots (in plots A and B, each dot represents a spike) or short vertical markers when the number of neurons being monitored is lower (plots C, D, E, and F). Fig. 4(A) represents a raster plot of the input activity that is reaching the cerebellar architecture through mossy fibers at the initial learning stage. As explained before, mossy fibers are able to elicit a set of spike trains related to the desired and actual positions and velocities (according to the scheme illustrated in Fig. 2) presented by the robot arm along the eight-like trajectory movement. Each joint position and velocity is translated into spikes by using three groups (one for position and another one for velocity for each joint) of 20 mossy fibers. Each of these groups is activated by its corresponding set of receptive fields (Fig. 2) that are covering the operative range of the input variable. At this initial learning stage, the actual trajectory is far from the desired one, thus position/velocity values only activate part of the population of mossy fibers (compared to the activation of the mossy fibers encoding the desired trajectory). However, as Fig. 4(B) shows, at the final learning stage, both actual position/velocity values can properly cover the operative range of the input variables. It can be seen that at this final learning stage, the activation of the mossy fibers related to the desired trajectory is similar to the activation profile of the mossy fiber group related to the actual trajectory (encoding actual position and velocity along the movement execution).

The activity of mossy fibers reaches the granular cell layer. The granular layer operates adopting the model functionality described in [13–15] by Yamazaki and Tanaka, that is, it behaves as a state generator. A state generator machine is capable of representing each time step (in our simulations, this is 0.002 s) as an unambiguous time stamp (with a unique spike pattern representation), thus facilitating the learning process (see [7]).

As indicated in the description of the cerebellar architecture, the Purkinje cell activity is divided into 6 well-defined sets of spike trains representing the generated spiking activity related to the output agonist/antagonist joint micro complexes for each robot joint (joint 1, joint 2, and joint 3). Each pair of these 6 well-defined sets is related to each agonist/antagonist corrective action for the three joints. As aforementioned, the inferior olive activity (spiking patterns Fig. 4(C), (D)) is in charge of encoding the error signal (Fig. 4(C), (D) colored lines) that has to be compensated by the cerebellar corrective terms; here, we can see that there are also 6 well-defined areas related to micro-complexes encoding the positive/negative corrective actions for the three robot joints. Fig. 4(C), (D) illustrates how the inferior olive spike distribution during the trajectory execution remains proportional to the received error signals which in turn, are related to the actual position/velocity errors. In these figures, it is shown how just a positive corrective action is demanded in joints 2 and 3 whilst both positive and negative actions are demanded in joint 1 along the whole eight-like trajectory execution. The error directionality (either positive or negative error) is also illustrated in Fig. 4(C) and (D). Obviously, at the initial learning stage, the amplitude of the encoded error signal to be translated into spikes is high as well as the number of spikes elicited by the inferior olive since the learning process has barely started (Fig. 4(C)). On the contrary, once the learning process is well settled down, the expected amplitude of the encoded error signal to be translated into spikes and the numbers of elicited spikes by the inferior olive decreased significantly at this final learning stage (Fig. 4(D)). The Inferior Olive cell activity is constrained between 1 and 10 Hz, according to neurophysiological data [81].

Finally, DCN generated output activity is plotted in Fig. 4(E) and (F). At the beginning of the learning stage, a negligible cerebellar output is provided (Fig. 4(E)) whilst at the final learning stage (Fig. 4(F)), an appropriate cerebellar output corrective action is generated. Error corrections are accomplished by changes in the activity of PCs that, in turn, affect the activity of the DCN, which eventually is translated into analog torque correction signals (also plotted in Fig. 4(E) and (F), with continuous lines) following principles

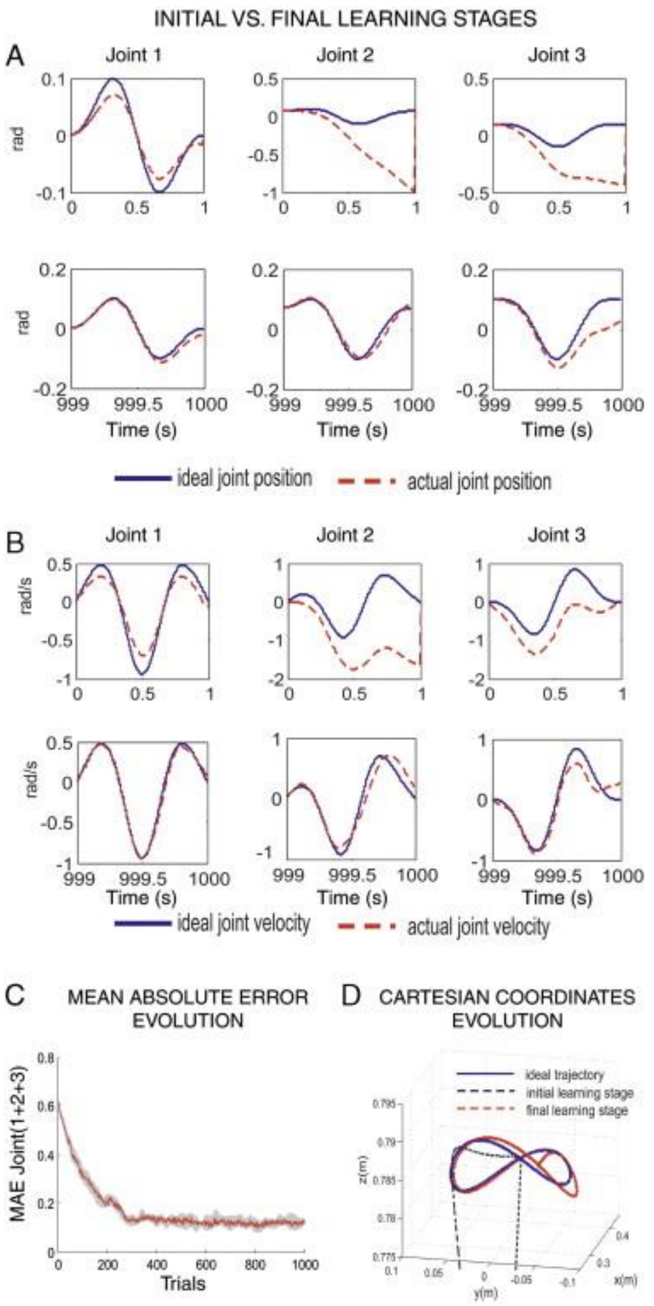


Fig. 5. Robotic performance (system behavior) in a manipulation task. The manipulation of a 10 kg payload whilst executing an eight-like trajectory reveals the inner robot dynamics. The benchmark trajectory execution takes one second in each trial. (A) Snapshot of the execution of the eight-like trajectory in joint coordinates (position) belonging to the initial learning stage (top plots) and the final learning stage (bottom plots). (B) Snapshot of the execution of the eight-like velocity trajectory in joint coordinates (velocity) belonging to the initial learning stage (top plots) and the final learning stage (bottom plots). (C) Averaged Mean Absolute Error (during each trial) obtained along the learning process computing the addition of the individual MAEs corresponding to each robot joint. Four different simulations with different initial random values at PF–PC synaptic weights have been used. The shadowed area is defined between the maximum and minimum values among the four simulations in each trial. The red curve is the average of the four simulations. (D) Cartesian coordinate evolution during the learning process. At the initial learning stage, the LWR is not capable of properly handling the attached payload. At the final learning stage, the cerebellum is able to provide the appropriate corrective torque values achieving almost the aimed target trajectory. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

already presented in the previous section. Each group of 4 DCN cells encodes the positive or negative corrective term which is eventually translated into a joint corrective torque. The higher/lower the activity at each micro complex is, the higher/lower its corresponding corrective torque is. In fact, the final activity at the DCN (which represents the actual corrective terms being produced) is the result of the subtraction of the PC activity (since its connection to DCN is inhibitory) which is specific and learned (through supervised learning at the granular cell-Purkinje cell synapses) from a general (nonspecific) activity term (from mossy fibers) which is approximately constant. Mind that, although the corrections of the DCN after learning may seem very irregular with high frequency terms (continuous lines Fig. 4(E) and (F)), the actual contributions are smoothed out by the motor system (in this case, the actual motor gears).

3.1. *Robotic input/output*

As briefly described in this work, cerebellar neural models are a current open issue whose operating basis is not yet well determined due to their working complexity principles. New tools for massive simulations (with multiple parameters) and state monitoring capabilities are necessary to identify how certain neural/subcircuit/neural layer features are related to the cerebellar functionality. Therefore, relating the cerebellar operation with the system in which the cerebellum is embodied seems to be the natural step forward. The presented integrated software platform is able to establish this interconnection between these two elements as shown in Figs. 4 and 5. Monitored snapshots of the whole cerebellar activity are related with their corresponding robot performance curve (system behavior). These snapshots facilitate the interpretation of the results giving a better insight about what is going on during embodied experimentation (behavioral experiments as the manipulation task illustrated in the previous section). Fig. 5 is an illustrative example of the sort of performance curves that can be obtained by using the presented software platform. Here, the robot arm is manipulating a 10 kg payload whilst executing a one second eight-like trajectory able to reveal the inner robot dynamics. Fig. 5(A) and (B), represent a snapshot of this one second eight-like trajectory execution in joint coordinates belonging to the initial learning stage (first row plots) and the final learning stage (position and velocity) (second row plots). The target trajectory at each joint is plotted in blue (continuous line) whilst the actual trajectory at each joint is plotted in red (dashed line). The error directionality (position and velocity error) is shown in these plots (either positive or negative error). As mentioned before, during the manipulation of objects with a significant weight, the arm–object platform dynamics differ from the original arm dynamics. This translates into a continuous negative error at the 2nd and 3rd joints which activates just one of the two inferior olive micro complexes (related to each joint) during the simulation. Additionally, Fig. 5(C) shows the Mean Absolute Error (MAE) obtained along the learning process. Finally, plot 5(D) represents just an example of how the obtained Cartesian coordinates of the tip of the robot arm evolve during the learning process. As shown, at the initial learning stage, the LWR is not capable of properly handling the attached payload; there is no acquired cerebellar corrective model for the 10 kg payload. Therefore, no corrective torque values are supplied yet. At the final learning stage, the cerebellum is able to provide the appropriate corrective torque terms achieving almost the aimed target trajectory.

Fig. 6 shows the same kind of experimentation conducted in Fig. 5 but extrapolated to different masses so as to reveal the capabilities and features that the learning at PF–PC synapses endows. Fig. 6(A) and (B), represent the MAE evolution whilst the robot arm is manipulating different payloads (10, 6 and 2.5 kg respectively) whether independently or consecutively. In Fig. 6(A) the learning process is reset, which means that all the synaptic weights at PF–PC are randomly chosen at the end of the learning of each payload whilst in Fig. 6(B) the learning process is not reset at the end of each payload learning. As can be seen, the learning is not destructive; the incoming learning process takes advantage of the previous learning process as indicated by the lower initial starting MAE error after switching between contexts. Fig. 6(C) points out the normalized performance that each of the aforementioned experiments achieves. Fig. 6(D) demonstrates how the learning process is compatible with incremental learning. Here, the payloads are switched every 50 trials (between 10 kg/6 kg in the left plot and 6 kg/2.5 kg in the right plot) thus showing how the learning process can simultaneously abstract two different payloads (two different dynamic models) that are only marginally interfering with each other.

3.2. Real time simulation

The computation load when simulating spiking neurons is high and needs to be done efficiently for controlling robots in real time. When any event-driven simulator is confronted with a massive amount of data to be processed online, this approach suffers due to the discontinuous flow of data to be computed. In fact, the learning process must be done online, in real time, as the robot is moving. A mechanism to ensure real time when processing all the neural activity involved during the simulation process has been implemented. During a neural simulation, all neural updates have to be processed in chronological order. However, during the neural simulation, future events may appear (i.e. events that occurred due to delayed spike firings or neural connections presenting delays). To manage this situation, a heap data structure able to efficiently insert and extract ordered events is required. Controlling the CPU time consumption of each time step allows real-time simulation. Although the calculation of the dynamics and kinematics of the robot (for instance, using a Newton Euler algorithm [74]) involves a constant number of operations at each time step, the neural simulation computational cost depends on the neural activity.

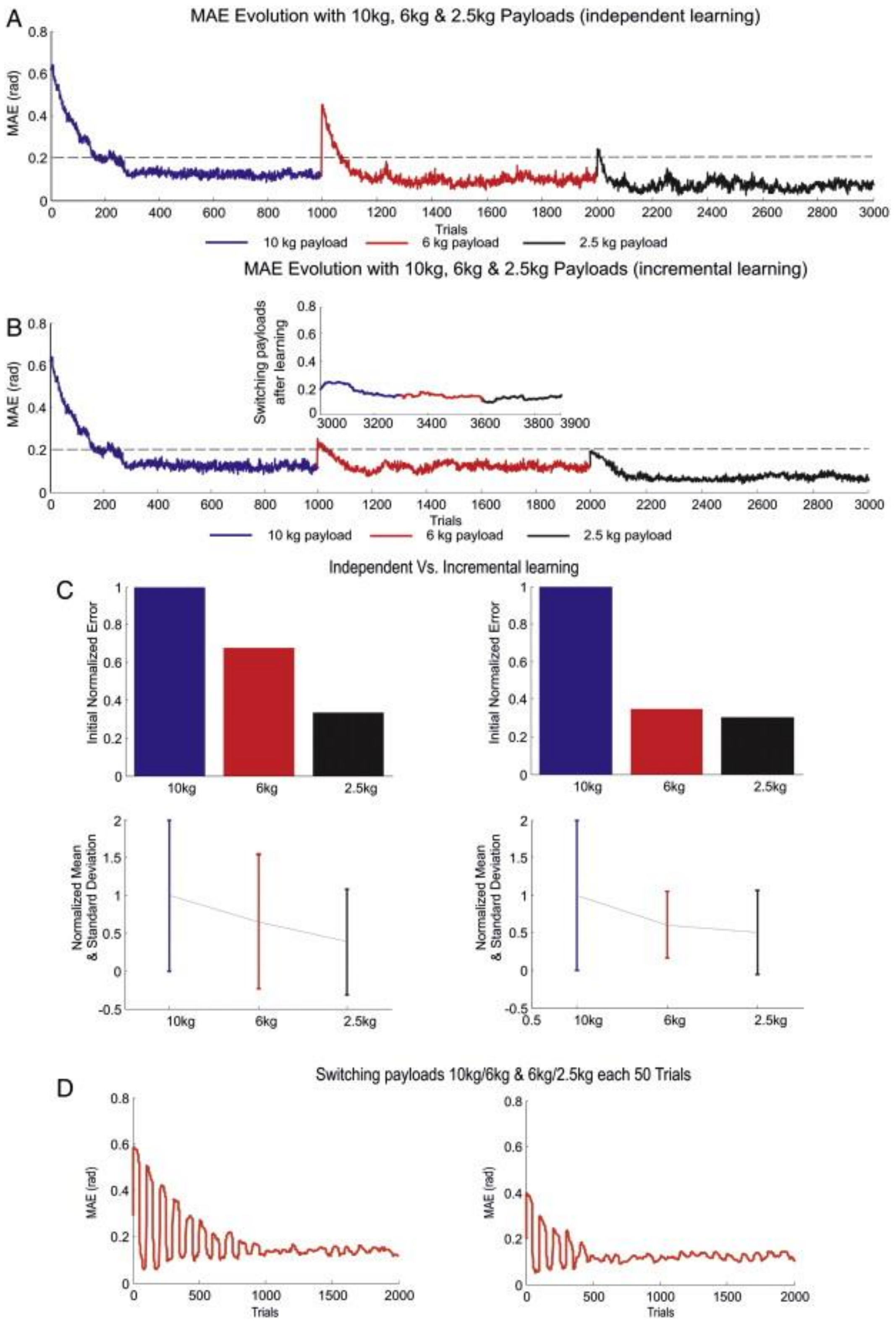


Fig. 6. Independent Learning vs. Incremental Learning. (A) Manipulation of 10, 6, and 2.5 kg independently. The learning process is reset (synaptic weights at PF–PC are randomly chosen at the end of the learning process of each payload). (B) Manipulation of 10, 6, and 2.5 kg consecutively. The learning process is not reset at the end of the each payload learning. The learning is not destructive; the incoming learning process takes advantage of the previous learning process as indicated by the lower initial starting MAE error after switching between contexts (objects under manipulation). The zoom in the graph shows how the system behaves when these new objects are presented again (300 iterations each). This demonstrates that the learning is done with only low interference between the object model dynamics being learned (abstracted). (C) Normalized initial error values (obtained in the ten-first trial errors per payload, 10 kg initial error has

been taken as the worse possible scenario) obtained at the beginning of the learning process with independent learning (left plots) and consecutive or incremental learning (right plots). The normalized average and standard deviation of MAE values (of the last 100 trials of each learning process) with independent learning (left plot) and consecutive or incremental learning (right plot) are also shown. In any case, incremental learning outperforms independent learning. (D) Incremental learning. Switching payloads every 50 trials (between 10 kg/6 kg in the left plot and 6 kg/2.5 kg in the right plot). It is shown how the learning can simultaneously abstract two different payloads (two different dynamic models) only marginally interfering with each other.

We have implemented a watchdog timer supervising each simulation time step. When the simulation process is consuming more time than a certain predefined constraint percentage of the total robot communication step time, the simulator skips noncritical event processing, thus keeping the simulation running in time (see Fig. 7). In our example, the total computation time has to remain below 2 ms, since the communication between the neural simulator and the robot platform is sliced in 2 ms intervals. As shown in Fig. 7(A), the computation time of each simulation slice (of 2 ms) consumes less than 2 ms. The "computation time" includes the cerebellar simulation time, the robotic simulation time, and the communication time between them. At each simulation step, the cerebellum updates and computes its internal neural states thus eliciting a set of generated spikes. There exists a close relationship between the number of generated spikes and the consumed computational time (Fig. 7(A) and (B)). In the end, a trade-off decision has to be taken. A watchdog ensures that the boundary will not be surpassed.

This illustrative simulation is composed by 1871 neurons and 69 603 synapses. We have used simple point neurons (parameterized according to different cerebellar neuron types) with three state variables (membrane potential and the excitatory and inhibitory conductances). Thus, 5613 state variables need to be continuously updated. During one second of simulation, the network produces 9890 spikes and 69 603 synaptic weight modifications (through spike time dependent plasticity at the parallel fiber to Purkinje cell synapses). All this needs to be computed within the real-time constraint. The simulation was run on a CPU consisting of a Pentium i7 3770k 3.4 GHz processor with 8 GB RAM all mounted on an ASUS P8Zseries motherboard.

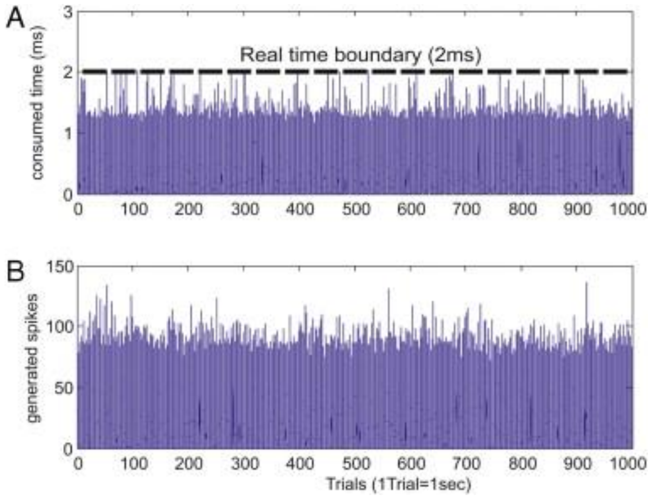


Fig. 7. Real time monitoring. The total computation time has to remain below 2 ms, because the communication between the neural simulator and the robot platform (real or simulated) is sliced in intervals of 2 ms.

4. Discussion

Along this paper, we have outlined how the EDLUT neural simulator has been equipped with an integrated robotic software framework. The dialog between these two elements, the EDLUT and the robotic software, is mediated by an efficient bidirectional interface (analog signals to spike patterns and vice versa) able to process sensory data from the robot agent and generate the appropriate robot motor commands. As a running embodied nervous system example, we have implemented and described a cerebellar architecture within a robotic control closed-loop where the robot features allow the exploitation of the cerebellar potential in a manipulating control task. This manipulation task aims to follow a specific desired trajectory consisting of sinusoidal components with the robotic arm manipulating a punctual mass. This punctual mass (representing the object under manipulation) affects the global dynamic model of the arm + object plant. The cerebellar system aims to provide corrective torque terms to compensate the existing mismatch between the arm dynamic model and the one of the arm + object under manipulation. These corrective torque terms are refined as the cerebellum acquires the dynamic model of the object under manipulation. This can be considered an abstraction process based on just the synaptic plasticity mechanism between the parallel fibers and the Purkinje cells.

The interest of this integrated neurobotics software platform can be outlined in two main points: for accelerating the development of biologically plausible control architectures cooperating with robot agents and for studying how certain capabilities of the cerebellum in coordinated motion and object manipulation are based on cellular characteristics, nervous system topology, or local synaptic adaptation mechanisms. In fact, a rich dynamical environment (i.e. highly reconfigurable robot model dynamics and reconfigurable cerebellar control loops) is a powerful tool to explore neurophysiological hypotheses from a functional point of view. All this also needs to be complemented with an appropriate monitoring and evaluation methodology. Here, it has been addressed not only just the way in which the neural activity can be plotted and interpreted by considering the micro-complex biologically plausible cerebellar organization, but also the neural activity contributions to agonist and antagonist motor system outputs thanks to the continuous monitoring of the target and actual joint trajectories.

Furthermore, the performance obtained is also remarkable. Although a simulation achieving real-time could be considered to be irrelevant, it is a critical non-trivial issue in embodied system neuroscience. When doing experiments with a real neuro-operated body, real-time operation becomes a major requirement. We have shown how this integrated software framework fulfils real-time requirement enabling a future real-robot cerebellar spiking control. In fact, the software framework integrating the neural simulator, the robotic simulator and all the communication and monitoring components has been developed with demanding real-time constraints.

5. Conclusions

In this paper, we show how a cerebellar structure integrated in the control loop as an adaptive feedforward model can learn to abstract model dynamics of objects being manipulated. We use an integrated simulation platform consisting of a real-time spiking neural simulator (EDLUT) and a simulated robot (LWR). This platform allows us to monitor the cell activity at different layers in terms of spike patterns as well as the contribution that they produce in terms of actual corrective torques within the control loop before learning the object model, and also eventually in the corrected trajectory (closer to the goal trajectory) after the learning process converges. The possibility of monitoring each cell activity allows us to interpret how the whole network works, receiving distributed spike patterns from the mossy fibers, producing sparse coding at the granular layer and adapting the weights between the granular layer and the Purkinje cells through supervised learning driven by the inferior olive activity (which is related to the actual error at each instant of the trajectory execution). The cerebellum integrated in the control loop with the presented configuration (actual and desired positions/velocities reaching the cerebellum through mossy fibers), performs the model abstraction process, as a function approximation problem (with the object under manipulation on-the-loop).

In the final experiments done (Fig. 6(A), (B) and (C)), we demonstrate that the presented architecture can learn dynamic models incrementally (with low interference with each other). In fact, learning a new model takes advantage of previous learned weights (related to previous objects under manipulation) but without destroying these previous models (Fig. 6(D)).

Acknowledgements

This work was supported by the EU grants REALNET FP7-ICT270434 (where the cerebellar simulations were developed) and HBP FP7 Flagship Project 604102 (monitoring tools are being developed), and by the national grants ARC-VISION (TEC2010-15396) and PYR (2014-16).

Appendix A. considerations related to benchmark trajectory accuracy; communication interface delay

When a real robot is connected to the controller (cerebellar base controller), its communication interface introduces a delay each time that the joint positions are obtained and the joint motor torques are set. This delay limits the frequency in which the controller can interact with the robot. Thus, the robot communication interface determines the minimum control cycle time. The robot trajectory accuracy decreases as the control cycle time increases, since, for example, the robot motor torque set points remain constant during each cycle. Therefore, the suitability of a concrete communication interface (bus) depends on the trajectory accuracy decline which is acceptable. It is of importance then to take this limitation into account when developing realistic real-time software towards embodied system neuroscience. Spiking cerebellar updating usually demands simulation step times in the millisecond scale (1–2 ms) [5–7] making this bus delay consideration an important factor to be considered when designing cerebellar control stages.

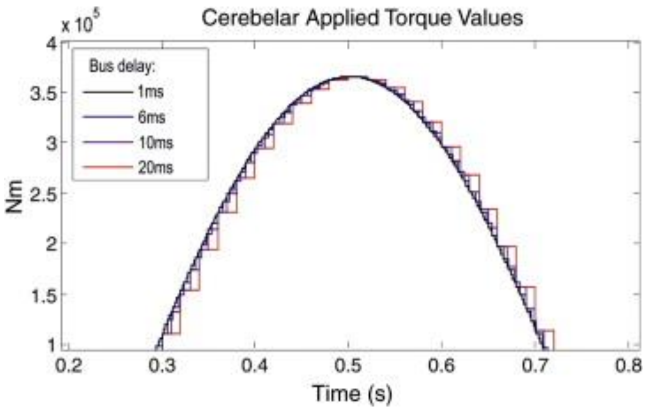


Fig. A.1. Possible consequences of the interface delay: snapshot of the cerebellar torque supplied to a LWR robot [77] (after being kept constant for several milliseconds as indicated in different traces).

Just as an example, Fig. A.1 illustrates the inaccuracy introduced by different bus transmission delays for different conducted experiments using a simulated lightweight robot [77] and an eight-shaped test trajectory. In order to simulate the effect of a communication bus, the torque generated by the controller is repeatedly kept constant for a period (control cycle time). When the robot input torque is increasing, the bus delay produces an average torque below the desired one (with negligible bus delay). The opposite occurs when the input torque is decreasing. Therefore, the joint angle error caused by the transmission time is related to the desired angle value and velocity during the trajectory execution.

Appendix B. Considerations related to the friction force of the robot joints

There are several forces that affect the expected robot dynamic model. When these forces are not properly taken into account, an open-loop controller for an ideal robot may fail to produce accurate movements. The most relevant perturbing forces that can be easily found in simple robotic arms can be summarized as follows:

Force exerted by the wires attached to the robot motors (for supping current and measuring angle encoder inputs/outputs): These forces remain relatively low. They can pull or push the robot's joints when the arm is in certain positions, facilitating or hindering the movement in certain directions. Since these forces are usually very low, it can be assumed that they will be compensated thanks to the adaptability of the cerebellar controller.

Inner dry friction forces of the robot joints: The two regimes of dry friction are static friction (the joint remains static) and kinetic friction (between moving surfaces of the joint). Sometimes the static friction of some robots is very significant. This friction force can be also compensated by the cerebellar controller. Nevertheless, when the magnitude of this force is comparable or higher than the rest of the force that the cerebellar controller must exert (to compensate for other deviations from the ideal dynamic model), the precision of the adaptive cerebellar module to compensate these other deviations is low. This occurs because if the cerebellar output force range increases, the resolution of its output per force unit decreases. This output range increase is equivalent to multiplying the output by a factor; therefore, the inaccuracy of this output would also be multiplied.

Accurately compensating the effect of the friction forces can sometimes become considerably complex, depending on the used compensation technique (this force is not the same in all the possible joint angles); in fact, the friction term proves to be crucial when controlling light-weight robot arms with high-ratio gear boxes because there are no standard methodologies/techniques to control these robots without massive modeling [76]. However a complex technique to fully compensate this force is not needed since the cerebellar module can conveniently compensate it (when it is relatively low). Thus, in this case, the goal of the compensation technique should not be to fully compensate for these perturbations, but to keep them in a range domain where the cerebellar module can learn to accurately correct the movement deviations.

References

[1] J.S. Albus, A theory of cerebellar function, Math. Biosci. 10 (1971) 25–61.

[2] D. Marr, A theory of cerebellar cortex, J. Physiol. 202 (1969) 437–470.

[3] S.F. Giszter, K.A. Moxon, I.A. Rybak, J.K. Chapin, Neurobiological and neurorobotic approaches to control architectures for a humanoid motor system, Robot. Auton. Syst. 37 (2001) 219–235.

[4] E. Ros, R. Carrillo, E.M. Ortigosa, B. Barbour, R. Agís, Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics, Neural Comput. 18 (2006) 2959–2993.

[5] N.R. Luque, J.A. Garrido, R.R. Carrillo, S. Tolu, E. Ros, Adaptive cerebellar spiking model embedded in the control loop: context switching and robustness against noise, Int. J. Neural Syst. 21 (2011) 385–401.

[6] N.R. Luque, J.A. Garrido, R.R. Carrillo, O.J.M.D. Coenen, E. Ros, Cerebellarlike corrective model inference engine for manipulation tasks, IEEE Trans. Syst. Man Cybern. B 41 (2011) 1299–1312.

[7] N.R. Luque, J.A. Garrido, R.R. Carrillo, O.J.M.D. Coenen, E. Ros, Cerebellar input configuration toward object model abstraction in manipulation tasks, IEEE Trans. Neural Netw. 22 (2011) 1321–1328.

[8] J.S. Albus, Data storage in the cerebellar model articulation controller (CMAC), Trans. ASME, J. Dyn. Syst. Meas. Control 3 (1975) 228–233.

[9] C. Sabourin, O. Bruneau, Robustness of the dynamic walk of a biped robot subjected to disturbing external forces by using CMAC neural networks, Robot. Auton. Syst. 51 (2005) 81–99.

[10] C.K. Tham, Reinforcement learning of multiple tasks using a hierarchical CMAC architecture, Robot. Auton. Syst. 15 (1995) 247–274.

[11] T. Yamazaki, S. Nagao, A computational mechanism for unified gain and timing control in the cerebellum, PLoS One 3 (2012) e33319.

[12] T. Yamazaki, S. Tanaka, Neural modeling of an internal clock, Neural Comput. 17 (2005) 1032–1058.

[13] T. Yamazaki, S. Tanaka, The cerebellum as a liquid state machine, Neural Netw. 20 (2007) 290–297.

[14] T. Yamazaki, S. Tanaka, Computational models of timing mechanisms in the cerebellar granular layer, Cerebellum 8 (2009) 423–432.

[15] T. Yamazaki, S. Tanaka, A spiking network model for passage-of-time representation in the cerebellum, Eur. J. Neurosci. 26 (2007) 2279–2292.

[16] P. Manoonpong, T. Geng, T. Kulvicius, B. Porr, F. Wörgötter, Adaptive, fast walking in a biped robot under neuronal control and learning, PLoS Comput. Biol. 3 (2007) e134.

[17] W. Wolpert, M. Kawato, Multiple paired forward and inverse models for motor control, Neural Netw. 11 (1998) 1317–1329.

[18] P. Dean, J. Porril, Adaptive filter models of the cerebellum. Computational analysis, Cerebellum 7 (2008) 567–571.

[19] P. Dean, J. Porrill, The cerebellum as an adaptive filter: a general model? Funct. Neurol. 25 (2010) 173–180.

[20] P. Dean, J. Porrill, C.F. Ekerot, H. Jörntel, The cerebellar microcircuit as an adaptive filter: experimental and computational evidence, Nat. Rev. Neurosci. 11 (2010) 30–34.

[21] M. Fujita, Adaptive filter model of the cerebellum, Biol. Cybernet. 45 (1982) 195–206.

[22] J. Porrill, P. Dean, Cerebellar motor learning: when is cortical plasticity not enough? PLoS Comput. Biol. 3 (2007) e197.

[23] J.C. Houk, J.T. Buckingham, A.G. Barto, Models of cerebellum and motor learning, Behav. Brain Sci. 19 (1996) 369–383.

[24] S. Tolu, M. Vanegas, N.R. Luque, J.A. Garrido, E. Ros, Bio-inspired adaptive feedback error learning architecture for motor control, Biol. Cybernet. 106 (2012) 507–522.

[25] S. Tolu, M. Vanegas, J.A. Garrido, N.R. Luque, E. Ros, Adaptive and predictive control of a simulated robot arm, Int. J. Neural Syst. 23 (2013).

[26] N. Schweighofer, J. Spoelstra, M.A. Arbib, M. Kawato, Role of the cerebellum in reaching movements in human. II. A neural model of the intermediate cerebellum, Eur. J. Neurosci. 10 (1998) 95–105.

[27] N. Schweighofer, M.A. Arbib, M. Kawato, Role of the cerebellum in reaching movements in human. I. Distributed Inverse dynamics control, Eur. J. Neurosci. 10 (1998) 86–94.

[28] J.C. Eccles, Circuits in the cerebellar control of movement, Proc. Natl. Acad. Sci. 58 (1967) 336–343.

[29] M. Ito, Adaptive modification of the vestibulo-ocular reflex in rabbits affected by visual inputs and its possible neuronal mechanisms, in: R. Granit, O. Pompeiano (Eds.), Progress in Brain Research, Elsevier, 1979, pp. 757–761.

[30] M. Ito, The Cerebellum and Neural Control, Raven Press, New York, 1984.

[31] M. Ito, Synaptic plasticity in the cerebellar cortex and its role in motor learning, Can. J. Neurol. Sci.. Le J. Can. Sci. Neurol. 20 (Suppl 3) (1993) S70–S74.

[32] M. Ito, Control of mental activities by internal models in the cerebellum, Nat. Rev. Neurosci. 9 (2008) 304–313.

[33] E. D'Angelo, C.I. De Zeeuw, Timing and plasticity in the cerebellum: focus on the granular layer, Trends Neurosci. 32 (2009) 10.

[34] Z. Gao, B.J. vanBeugen, C.I. De Zeeuw, Distributed synergistic plasticity and cerebellar learning, Nat. Rev. Neurosci. 13 (2012) 1–17.

[35] C. Hansel, D.J. Linden, E. D'Angelo, Beyond parallel fiber LTD: the diversity of synaptic and non-synaptic plasticity in the cerebellum, Nat. Neurosci. 4 (2001) 467–475.

[36] G.A. Jacobson, D. Rokni, Y. Yarom, A model of the olivo-cerebellar system as a temporal pattern generator, Trends Neurosci. 31 (2008) 617–625.

[37] G.A. Jacobson, I. Lev, Y. Yarom, D. Cohen, Invariant phase structure of olivocerebellar oscillations and its putative role in temporal pattern generation, Proc. Natl. Acad. Sci. 106 (2009) 3579–3584.

[38] R.R. Carrillo, E. Ros, S. Tolu, T. Nieus, E. D'Angelo, Event-driven simulation of cerebellar granule cells, Biosystems 94 (2008) 10–17.

[39] S. Solinas, T. Nieus, E. D'Angelo, A realistic large-scale model of the cerebellum granular layer predicts circuit spatio temporal filtering properties, Front. Cell. Neurosci. 4 (2010).

[40] P. Gleeson, V. Steuber, R.A. Silver, S. Crook, NeuroML, in: Computational Systems Neurobiology, Springer, 2012, pp. 489–517.

[41] M.L. Hines, T. Morse, M. Migliore, N.T. Carnevale, G.M. Shepherd, ModelDB: a database to support computational neuroscience, J. Comput. Neurosci. 17 (2004) 7–11.

[42] N.C. Rowland, D. Jaeger, Coding of tactile response properties in the rat deep cerebellar nuclei, J. Neurophysiol. 94 (2005) 1236–1251.

[43] N.C. Rowland, D. Jaeger, Responses to tactile stimulation in deep cerebellar nucleus neurons result from recurrent activation in multiple pathways, J. Neurophysiol. 99 (2008) 704–717.

[44] T.M. Teune, J. van der Burg, C.I. de Zeeuw, J. Voogd, T.J. Ruigrok, Single Purkinje cell can innervate multiple classes of projection neurons in the cerebellar nuclei of the rat: a light microscopic and ultrastructural triple-tracer study in the rat, J. Comp. Neurol. 392 (1998) 164–178.

[45] W. Zhang, W.D. Linden, Long-term depression at the mossy fiber-deep cerebellar nucleus synapse, J. Neurosci. 26 (2006) 6935–6944.

[46] A. Delorme, J. Gautrais, R. van Rullen, S. Thorpe, SpikeNET: a simulator for modeling large networks of integrate and fire neurons, Neurocomputing 26 (1999) 989–996.

[47] L. Watts, Event-driven simulation of networks of spiking neurons, Adv. Neural Inf. Process. Syst. (1994) 927–934.

[48] M. D'Haene, B. Schrauwen, J. Van Campenhout, D. Stroobandt, Accelerating event-driven simulation of spiking neurons with multiple synaptic time constants, Neural Comput. 21 (2009) 1068–1099.

[49] T. Makino, A discrete-event neural network simulator for general neuron models, Neural Comput. Appl. 11 (2003) 210–223.

[50] W. Gerstner, W.M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.

[51] J.A. Garrido, in: Edlut official website, 2012.

[52] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J.M. Bower, M. Diesmann, A. Morrison, P.H. Goodman, Simulation of networks of spiking neurons: a review of tools and strategies, J. Comput. Neurosci. 23 (2007) 349–398.

[53] J. Reutimann, M. Giugliano, S. Fusi, Event-driven simulation of spiking neurons with stochastic dynamics, Neural Comput. 15 (2003) 811–830.

[54] J.A. Garrido, R.R. Carrillo, N.R. Luque, E. Ros, Event and time driven hybrid simulation of

spiking neural networks, in: Advances in Computational Intelligence, Springer, 2011, pp. 554–561.

[55] A. Pouget, P. Dayan, R. Zemel, Information processing with population codes, Nat. Rev. Neurosci. 1 (2000) 125–132.

[56] S. Wu, S. Amari, H. Nakahara, Population coding and decoding in a neural field: a computational study, Neural Comput. 14 (2002) 999–1026.

[57] T. Flash, T.J. Sejnowski, Computational approaches to motor control, Curr. Opin. Neurobiol. 11 (2001) 655–662.

[58] B. Amirikian, A.P. Georgopoulos, Modular organization of directionally tuned cells in the motor cortex: is there a short-range order? Proc. Natl. Acad. Sci. 100 (2003) 12474–12479.

[59] K.R. Boff, J.E. Lincoln, Engineering Data Compendium. Human Perception and Performance. Vol. 3, Harry G Armstrong. Aerospace Medical Research Lab Wright-Patterson Afb Oh, 1988.

[60] N.V. Swindale, Orientation tuning curves: empirical description and estimation of parameters, Biol. Cybernet. 78 (1998) 45–56.

[61] N.R. Luque, J.A. Garrido, J. Ralli, J.J. Laredo, E. Ros, From sensors to spikes: evolving receptive fields to enhance sensorimotor information in a robot-arm, Int. J. Neural Syst. 22 (2012) 1250013.

[62] J.D. Victor, Spike train metrics, Curr. Opin. Neurobiol. 15 (2005) 585–592.

[63] M.C. van Rossum, A novel spike distance, Neural Comput. 13 (2001) 751–763.

[64] B. Schrauwen, J. Van Campenhout, BSA, a fast and accurate spike train encoding scheme, in: Proceedings of the International Joint Conference on Neural Networks, 2003, IEEE, 2003, pp. 2825–2830.

[65] G.C. Goodwin, Adaptive Prediction and Control, Prentice Hall, NJ, 1984.

[66] R.C. Miall, D.J. Weir, D.M. Wolpert, J.F. Stein, Is the cerebellum a Smith predictor? J. Mot. Behav. 25 (1993) 203–216.

[67] D.M. Wolpert, R.C. Miall, Forward models for physiological motor control, Neural Netw. 9 (1996) 1265–1279.

[68] A.J. Bastian, Learning to predict the future: the cerebellum adapts feedforward movement control, Curr. Opin. Neurobiol. 16 (2006) 645–649.

[69] E.J. Hwang, R. Shadmehr, Internal models of limb dynamic and the encoding of limb state, J. Neural Eng. 2 (2005) 266–278.

[70] E. Nakano, H. Imamizu, R. Osu, Y. Uno, H. Gomi, T. Yoshioka, M. Kawato, Quantitative examinations of internal representations for arm trajectory planning. Minimum commanded torque change model, J. Neurophysiol. 81 (1999) 2140–2155.

[71] E. Todorov, Optimality principles in sensorimotor control (review), Nat. Neurosci. 7 (2004) 907–915.

[72] E.R. Kandel, J.H. Schwartz, T.M. Jessell, Principles of Neural Science, McGraw-Hill Professional Publishing, New York, 2000.

[73] M. Kawato, K. Furukawa, R. Suzuki, A hierarchical neural-network model for control and learning of voluntary movement, Biol. Cybernet. 57 (1987) 169–185.

[74] J.A. Garrido, N.R. Luque, E. D'Angelo, E. Ros, Distributed cerebellar plasticity implements adaptable gain control in a manipulation task: a closed-loop robotic simulation, Front. Neural Circuits 7 (2013).

[75] B. Siciliano, O. Khatib, Springer Handbook of Robotics, Springer, 2008.

[76] P. van der Smagt, Benchmarking cerebellar control, Robot. Auton. Syst. 32 (2000) 237–251.

[77] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, G. Hirzinger, The DLR lightweight robot: design and control concepts for robots in human environments, Int. J. Ind. Robot 34 (2007) 376–385.

[78] G. Hirzinger, J. Butterfab, M. Fischer, M. Grebenstein, M. Hähnle, H. Liu, N. Shäfer, I. Sporer, A mechatronics approach to the design of light-weight arms and multifingered hands, in: ICRA, 2000, pp. 46–54.

[79] R.E. Kettner, S. Mahamud, H. Leung, N. Sittko, J.C. Houk, B.W. Peterson, A.G. Barto, Prediction of complex two-dimensional trajectories by a cerebellar model of smooth pursuit eye movement, J. Neurophysiol. 77 (1997) 2115–2130.

[80] H. Hoffmann, G. Petkos, S. Bitzer, S. Vijayakumar, Sensor-assisted adaptive motor control under continuously varying context, 2007.

[81] N. Schweighofer, K. Doya, M. Kawato, Electrophysiological properties of inferior olive neurons: a compartmental model, J. Neurophysiol. 82 (1999) 804–817.

Niceto Rafael Luque received his M.S. and Ph.D. in Computer science and Networks from the University of Granada (Spain) in 2007 and 2013 respectively. He also received a B.S in Electronics Engineering and an M.S. in Automatics and Industrial Electronics from the University of Córdoba (Spain) in 2003 and 2006, respectively. He is currently participating as a postdoctoral researcher in an EU project related to adaptive learning mechanisms and bio-inspired control (REALNET). His main research interests include biologically processing control schemes, light weight robot, and spiking neural networks.

Richard Rafael Carrillo received a Ph.D. degree in Computer and Electronics Engineering from the University of Cagliari (Italy) in 2008 and in Computer Science from the University of Granada (Spain) in 2009. He is currently participating as a postdoctoral researcher in an EU project related to the implementation of realistic neural networks. He is interested in the efficient simulation of spiking neural networks and modeling of neurons and functional nervous circuits.

Francisco Naveros received an M.S. in Telecom Engineering and an M.S. in Computer science and Networks from the University of Granada (Spain) in 2011 and 2012 respectively. He is currently a Ph.D. student at the University of Granada. His main research interest includes high performance simulation of spiking neural networks using GPUs.

Jesús Alberto Garrido received his M.S. degree in Computer Science and his Ph.D. degree in Computer Architectures and Networks from the University of Granada (Spain) in 2007 and 2011 respectively. He is currently working as a postdoctoral researcher at the Department of Brain and Behavioral Sciences at the University of Pavia (Italy). He is participating in an EU project related to adaptive learning mechanisms and bio-inspired control (REALNET). His main research interests include biologically processing control schemes, learning models, and spiking neurons.

María José Sáez-Lara received the Ph.D. degree from the University of Granada (Spain) in 2003, where she is currently an Associative Professor at the Biochemistry and Molecular Biology Department. Her research interests include Computational Biology, Molecular Biology processes and Genetics.

frontiers in Computational Neuroscience

ORIGINAL RESEARCH
published: 02 March 2016
doi: 10.3389/fncom.2016.00017

Distributed Cerebellar Motor Learning: A Spike-Timing-Dependent Plasticity Model

Niceto R. Luque[1*†], *Jesús A. Garrido*[1†], *Francisco Naveros*[1], *Richard R. Carrillo*[1], *Egidio D'Angelo*[2,3] *and Eduardo Ros*[1]

[1] Department of Computer Architecture and Technology, Research Centre for Information and Communications Technologies of the University of Granada (CITIC-UGR), Granada, Spain, [2] Brain Connectivity Center, Istituto di Ricovero e Cura a Carattere Scientifico, Istituto Neurologico Nazionale Casimiro Mondino, Pavia, Italy, [3] Department of Brain and Behavioural Sciences, University of Pavia, Pavia, Italy

Deep cerebellar nuclei neurons receive both inhibitory (GABAergic) synaptic currents from Purkinje cells (within the cerebellar cortex) and excitatory (glutamatergic) synaptic currents from mossy fibers. Those two deep cerebellar nucleus inputs are thought to be also adaptive, embedding interesting properties in the framework of accurate movements. We show that distributed spike-timing-dependent plasticity mechanisms (STDP) located at different cerebellar sites (parallel fibers to Purkinje cells, mossy fibers to deep cerebellar nucleus cells, and Purkinje cells to deep cerebellar nucleus cells) in close-loop simulations provide an explanation for the complex learning properties of the cerebellum in motor learning. Concretely, we propose a new mechanistic cerebellar spiking model. In this new model, deep cerebellar nuclei embed a dual functionality: deep cerebellar nuclei acting as a gain adaptation mechanism and as a facilitator for the slow memory consolidation at mossy fibers to deep cerebellar nucleus synapses. Equipping the cerebellum with excitatory (e-STDP) and inhibitory (i-STDP) mechanisms at deep cerebellar nuclei afferents allows the accommodation of synaptic memories that were formed at parallel fibers to Purkinje cells synapses and then transferred to mossy fibers to deep cerebellar nucleus synapses. These adaptive mechanisms also contribute to modulate the deep-cerebellar-nucleus-output firing rate (output gain modulation toward optimizing its working range).

Keywords: cerebellar nuclei, spike-timing-dependent plasticity, motor learning consolidation, cerebellar modeling, cerebellar motor control

OPEN ACCESS

Edited by:
Stefano Fusi,
Columbia University, USA

Reviewed by:
Thomas Wennekers,
University of Plymouth, UK
Herve Rouault,
Janelia Research Campus, USA

****Correspondence:***
Niceto R. Luque
nluque@ugr.es

[†] These authors have contributed equally to this work.

Received: *02 June 2015*
Accepted: *15 February 2016*
Published: *02 March 2016*

Citation:
Luque NR, Garrido JA, Naveros F, Carrillo RR, D'Angelo E and Ros E (2016) Distributed Cerebellar Motor Learning: A Spike-Timing-Dependent Plasticity Model. Front. Comput. Neurosci. 10:17. doi: 10.3389/fncom.2016.00017

INTRODUCTION

Since Marr (1969) and Albus (1971), the cerebellar loop has been extensively modeled providing smart explanations on how the forward-controller operations in biological systems seem to work. The classic long-term synaptic plasticity between parallel fibers (PF) and Purkinje cells (PC) [driven by the inferior olive (IO) action] stands at the core of those processes related to sensorimotor adaptation and motor control. However, this adaptation mechanism can be enhanced with

Abbreviations: PF, parallel fiber; MF, mossy fiber; CF, climbing fiber; GC, granule cell; GoC, Golgi cell; PC, Purkinje cell; DCN, deep cerebellar nuclei; IO, inferior olive; MLI, molecular layer interneuron; MAE, mean average error; EBCC, eye blink classical conditioning; VOR, vestibulo-ocular reflex.

complementary plasticity sites at the cerebellar circuit. Particularly, in this work we explore how STDP at Deep Cerebellar Nuclei efficiently complements the classical PF–PC long-term plasticity as an efficient adaptive gain term and memory consolidation resource.

Plasticity in Deep Cerebellar Nuclei

It is worth revisiting the original theories based on the structural analysis of cerebellar connectivity (Eccles, 1967; Eccles et al., 1967; Marr, 1969; Albus, 1971; Fujita, 1982). In those theories, the cerebellum was proposed to act as a timing and learning machine. The granular layer was hypothesized to recode the input spatiotemporal activity into sparse somatosensory activity. Then, only the relevant patterns were learnt and stored at PF–PC synapses under the supervised control of the teaching signal supplied by climbing fibers (CF). In light of different electrophysiological findings, it has been suggested that the CFs convey sensory feedback from comparing proprioceptive and predicted signals. CFs could indeed provide quantitative error estimation (Bazzigaluppi et al., 2012; De Gruijl et al., 2012) that, in turn, would be able to improve motor performance through specifically depressing the PF (PF–LTD) synapses that are more correlated to motor errors.

Although since the early 70s, plasticity in the cerebellar cortex was widely accepted and established, demonstrations of synaptic plasticity in cerebellar learning at cerebellar nucleus cells were studied significantly later. It was at the end of the 1990s when the analysis of the circuit-cerebellar basis for learning eye-movement yielded insight into a plausible two-state learning mechanism (Shadmehr and Brashers-Krug, 1997; Shadmehr and Holcomb, 1997). That is, whilst a fast learning process occurs in the cerebellar cortex (granular and molecular layer, involving PF–PC plasticity), a slow consolidation process occurs in deeper structures (possibly, at the deep cerebellar nuclei, DCN; Shadmehr and Brashers-Krug, 1997; Shadmehr and Holcomb, 1997; Medina and Mauk, 2000; Ohyama et al., 2006).

The main idea behind this speculative scheme lies on assuming that PF and PC outcomes are mediated by upstream-processing-nervous centers and, in turn, PC outcome shapes the output of its corresponding DCN-target neurons (Miles and Lisberger, 1981; Zhang and Linden, 2006; Zheng and Raman, 2010). This two-state learning mechanism was motivated by the fact that DCN neurons are innervated by excitatory synapses from mossy fibers (MFs) as well as by inhibitory synapses from PCs. The interplay between these excitatory and inhibitory connections has not been well-established yet. However, evidence of synaptic-plasticity traces at MFs (Racine et al., 1986; Medina and Mauk, 1999; Ohyama et al., 2006; Pugh and Raman, 2006; Zhang and Linden, 2006; Yang and Lisberger, 2014) and at PC synapses (Morishita and Sastry, 1996; Aizenman et al., 1998; Ouardouz and Sastry, 2000; Masuda and Amari, 2008) in the cerebellar nuclei and their vestibular nucleus (VN) counterparts has recently been encountered. This motivates the development of an adequate mechanistic model toward better understanding the potential of the DCN plasticity role.

Deep nucleus plasticity is assumed to be supervised and, according to different hypotheses, it is thought to be responsible for storing granular layer patterns that are correlated with the teaching signal generated by PCs (Hansel et al., 2001; Boyden et al., 2004; Gao et al., 2012). This plasticity comprises several mechanisms generating LTP and LTD at MF–DCN (Bagnall and du Lac, 2006; Pugh and Raman, 2006) and PC–DCN synapses (Morishita and Sastry, 1996; Aizenman et al., 1998; Ouardouz and Sastry, 2000). MF–DCN and PF–DCN plasticity are indeed thought to be important in controlling cerebellar learning in the context of the eye-blink classic conditioning (EBCC; Medina and Mauk, 1999, 2000). The equivalent forms of plasticity in the VN are also important in controlling cerebellar learning in the vestibulo-ocular reflex (VOR; Masuda and Amari, 2008).

Recent works based on a simplistic cerebellar model have proposed that the MF–DCN and PC–DCN synaptic plasticity mechanisms are an adaptive cerebellar-gain control (Garrido et al., 2013a; Luque et al., 2014b). Nevertheless, those works were focused just on the functional role of these DCN learning rules, without answering the question of how these learning rules may take place as STDP mechanisms. Two main issues were addressed within these computational approaches:

- Firstly, the proposed adaptive gain controller (Garrido et al., 2013a; Luque et al., 2014b) at the cerebellum was equipped with suitable learning and memory mechanisms whose nature is still under debate (Carey, 2011; Yang and Lisberger, 2014).
- Secondly, the gain-control system involving the cerebellum was capable of optimizing its performance within wider operative ranges; concretely, keeping PF–PC adaptation mechanisms within their optimal working range.

Conversely, these approaches still lack two key features that are addressed in the present work in a more realistic and biologically plausible scenario:

i. Whilst MF–DCN and PC–DCN plasticity played a key role in generating the gain controller, the way through which the slow learning consolidation process occurred was still missing. The level of detail of those previous computational approaches prevented this feature from being properly addressed.
ii. It was not clear how to implement the analog conceptual model of these previous approaches into a spiking-based model compatible with spiking signal processing and then endowed with long-term spike-timing-dependent plasticity mechanisms.

Now, in this work, we have studied the impact of distributed cerebellar spike-time synaptic plasticity on both gain adaptation and learning consolidation when performing a manipulating task. To that purpose, we have used a cerebellar spiking-based model embedded in closed loops. The working hypothesis assumes that there exist three learning sites; one located in the cerebellar cortex (PF–PC) and the other two located at the DCN innervations (MF–DCN and PC–DCN), all including LTP and LTD (**Figure 1A**). We found that our simulations captured the adaptive features proposed in the analog models regarding self-adaptive-gain control recalibration over a broad dynamic range involving manipulation of a heavy mass. Furthermore, we confirmed how MF–DCN innervations broadly stored what was already learnt at PF–PC. PC–DCN was also revealed as a

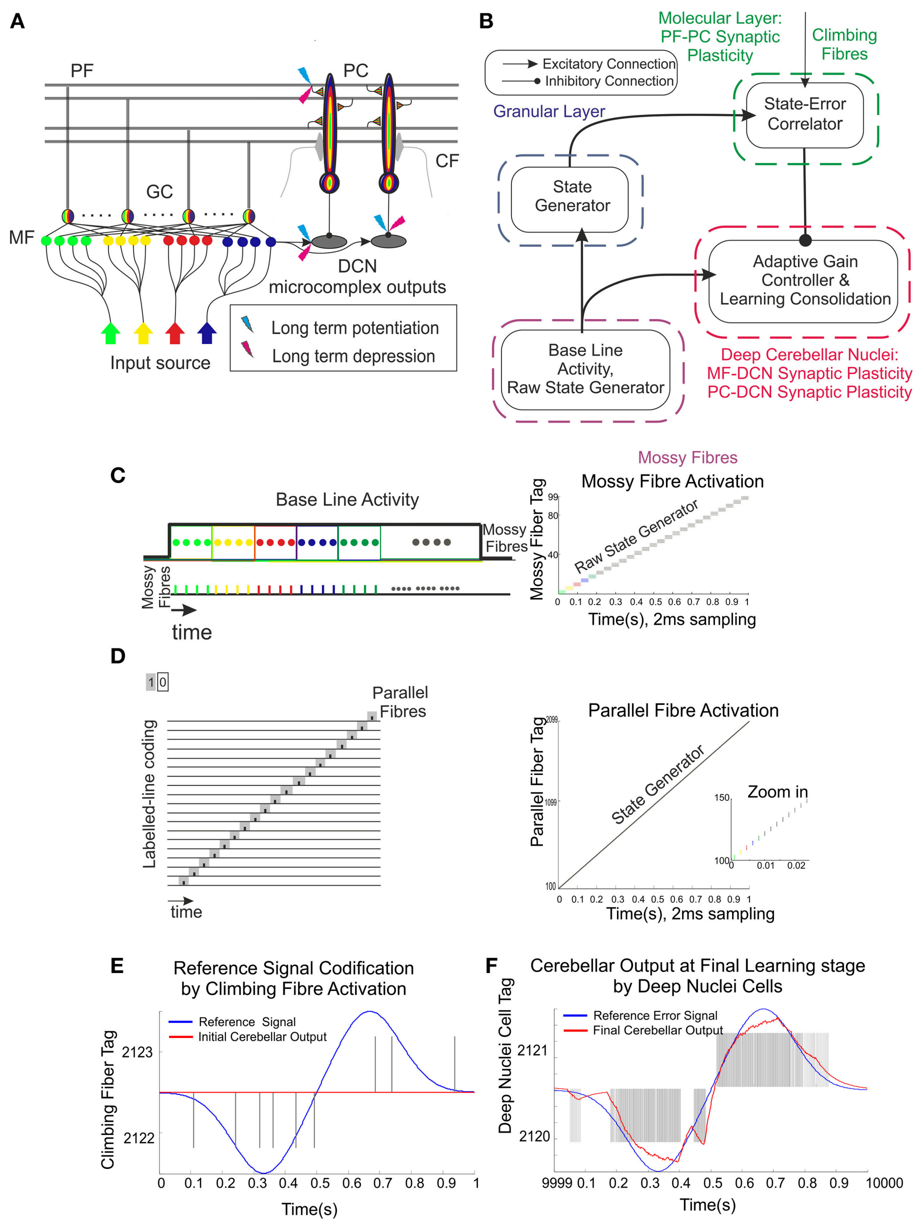


FIGURE 1 | Schematic representation of the main cerebellar layers, cerebellar cells and connections, as well as plasticity sites considered. Working hypothesis of cerebellar learning in a manipulation task. **(A)** Cerebellar architecture. Colored representation indicates signals from different sources such as different cuneate receptive fields or proprioceptors. Pathways involved in long-term synaptic plasticity for DCN and PC afferents are indicated with two colored symbols; long-term potentiation in blue and long-term depression in magenta. PF, parallel fiber; MF, mossy fiber; CF, climbing fiber; GC, granule cell; PC, Purkinje cell; DCN, deep cerebellar nuclei. **(B)** Conceptual cerebellar block-diagram. Each cerebellar layer is put in relation to its functionality according to the cerebellar model hypothesis being adopted. MF input layer conveys sequences of spikes acting as time-evolving states (raw state generator) which present a constant firing rate, thus supplying the excitatory activity required by the DCN to start operating. The cerebellar granular layer operates as a state generator that is reinitialized with the onset of a new trial. The PC function acts as a state-error correlator; each state is correlated with the error signal that reaches the PCs through the CFs and represents the difference between the controlled variable (actual cerebellar output value) and the reference variable (set point). By repeating pairings of PF states and CF error signals, trial after trial, an association between these two sets is formed thanks to the PF–PC long-term plasticity action driven by the activity at CFs (supervised learning). A learnt corrective action is therefore deployed to anticipate the incoming error. This association implies either a reduction or increase of PC firing at different step times. Finally, the temporally correlated signals from PCs are inverted (due to the inhibitory nature of the PC–DCN connection) and conveyed to the DCN which, in turn, receives inputs coming from MF afferents (excitatory). The DCN operates like an adder/subtractor able to adaptively modulate the output DCN gain which enables learning consolidation (adapted from Garrido et al., 2013a). **(C)** During each manipulation trial, the onset of the movement makes MFs convey sequences of spikes that present a constant firing rate and time-evolving states simultaneously. This MF constant firing-rate initialization, in turn, allows PFs to start generating a non-recurrent sequence of firing states (Yamazaki and Tanaka, 2007b, 2009). To that aim, groups of non-overlapped MFs are correlatively activated during the simulation. Each colored MF group represents a certain state able to determine univocally a certain time-period within the simulation **(D)** The figure presents the GC coding

(Continued)

FIGURE 1 | Continued
strategy: in our model, the states correspond directly to non-overlapped GCs activated at each time-step simulation. Each PF group represents a certain state able to determine univocally a certain simulation step-time within the simulation **(E)** Each CF carries the teaching spikes. CF cell response follows a probabilistic Poisson process. A single spike reports time-information regarding the instantaneous error and the probabilistic spike sampling of the error ensures that the whole error region is accurately represented over trials **(F)** The generated DCN spike train is translated into meaningful analog output signals by using a Finite Impulse Response filter (FIR).

fundamental plasticity site in charge of adapting the DCN-output firing rate.

MATERIALS AND METHODS

Within this section, the working principles of the proposed mechanistic spiking cerebellar model are described. Furthermore, the major functional hypotheses related to the granular layer, PC layer, and DCN are linked to the cerebellar underlying structure (**Figure 1B**). The section is also divided into two main blocks. The first block describes the cerebellar topology to be used and the implemented spike-timing-dependent plasticity mechanisms. The second block consists of two case studies: Case study A uses a simplified cerebellar control loop seeking to reveal the functional interplay amongst distributed plasticity cerebellar sites, whereas Case study B uses a cerebellar control loop designed to operate a simulated robotic arm (able to manipulate heavy masses) that can exploit the potential of using distributed cerebellar plasticity in quantitative and qualitative evaluation experiments.

Cerebellar Computational Model Considerations

A cerebellar spiking model was implemented using the EDLUT simulator (http://edlut.googlecode.com; Ros et al., 2006; Naveros et al., 2015). This model intended to capture the essence of the main properties of synaptic cerebellar topology and its neuronal elements. This work aimed to investigate the synaptic-weight plasticity at multiple connections. The simulations were done using leaky integrate-and-fire (LIF) neural models whereas synapses were simplified using conductance based exponential models (Gerstner and Kistler, 2002). The work was focused on the IO–PC–DCN subcircuit, thus the granular layer was also simplified. That is, the granular layer was implemented as a state generator following the liquid-state-machine principles (Yamazaki and Tanaka, 2007a, 2009; see the Cerebellar-Network Organization Section). All the implemented code is at the disposal of the reader at http://www.ugr.es/∼nluque/restringido/CODE.rar (user: REVIEWER, password REVIEWER).

Cerebellar Network Organization

The connectivity and topology of the cerebellar network sought to abstract the general cerebellar principles taking inspiration from Eccles et al. (1967), Ito (1984), Voogd and Glickstein (1998) and Medina and Mauk (1999, 2000). Our cerebellar model consisted of four main layers (**Figure 1A**) connected as indicated in **Table 1**:

- ***Mossy fibers (MFs):*** (100) MFs were modeled as leaky I&F neurons. According to existing models of eyelid-conditioning cerebellar control (Medina and Mauk, 1999; Yamazaki and Tanaka, 2007b, 2009), MFs are hypothesized to convey sequences of spikes which present a constant firing rate during the conditioned-stimulus-presentation phase. In our simplified model, MFs were correlatively activated in non-overlapped and equally-sized neural clusters ensuring a constant firing rate during the execution of each learning trial whilst they remained silent when the learning trial came to its end. The learning trial start was defined by the onset of MF activity thus forcing the granular layer to generate its state sequence, and supplying the base-line excitatory activity that DCN needed to start operating (**Figure 1C**).
- ***Granular cells (GCs):*** (2000) similarly to other models (Yamazaki and Tanaka, 2005, 2007a, 2009; Honda et al., 2011), the granular layer was implemented as a state generator, that is, the granular layer generated a sequence of active neuron populations without recurrence. The sequential activation of these neuron populations was able to represent the passage of time. When the learning process began, the granular layer produced non-overlapped time patterns that were repeatedly activated in the same sequence during each learning trial (1 s; **Figures 1C,D**). Having 1 s learning process in a 2 ms time-step simulation demanded 500 different states, which involved four non-overlapped GCs activated per time-step simulation. PF–PC synaptic conductances were set to an initial value (5 nS) at the beginning of the simulation, and were modified by the STDP mechanism during the training process. Note that the whole model aims to adopt cell realistic ratios, although the actual number of simulated neurons is much smaller than a full size rat model. A reduced version of the cerebellum (2000 GCs) where each PC just received activity from 2000 PFs was modeled. Since in a full model of the cerebellum, each PC should receive activity from about 150,000 PFs (Brunel et al., 2004), PF–PC weight values were scaled to obtain a similar relative PC excitation.
- ***Purkinje Cells (PCs):*** We have defined two case studies: (20) Purkinje cells in case study A, (60) Purkinje cells in case study B.

  Case study A; the cerebellar circuit was modeled as a closed loop able to supply a corrective signal to counterbalance the existing difference between a controlled variable (actual cerebellar output value) and a demanding reference variable (set point). This was equivalent to a cerebellar model compensating the error that one degree-of-freedom (DoF) manipulator could undergo (see Control Loop Section and **Figure 3A**). Within this loop, 20 PCs inhibited two DCNs that, in turn, counterbalanced the error curve. CFs (2) were

TABLE 1 | Summary of cerebellar cells and synapses implemented in Case A and Case B simulations.

Case A			Case B		
Presynaptic cell (number)	**Postsynaptic cell**	**Number of synapses**	**Presynaptic cell (number)**	**Postsynaptic cell**	**Number of synapses**
Mossy Fibers(100)	Granular Cells	8000	Mossy Fibers(100)	Granular Cells	8000
	Deep Cerebellar Nuclei	200		Deep Cerebellar Nuclei	600
Climbing Fibers(2)	Purkinje Cells	20	Climbing Fibers(6)	Purkinje Cells	60
Granular Cells(1000)	Purkinje Cells	40,000	Granular Cells(1000)	Purkinje Cells	120,000
Purkinje cell(20)	Deep Cerebellar Nuclei	20	Purkinje cell(60)	Deep Cerebellar Nuclei	60
Deep Cerebellar Nuclei(2)	–	–	Deep Cerebellar Nuclei(6)	–	–

assumed to transmit the difference between the set point curve and the actual one. CFs closed the loop providing input to the PCs. This layer was divided into two groups of 10 Purkinje cells each all receiving activity through each granular layer cell. One group was in charge of correcting the negative errors (providing activity toward enhancing output cerebellar corrective activity) and the other group was in charge of corrective positive errors. This set up mimics the existing interplay between agonist and antagonist muscles at biological systems. Each 10 PC group was innervated by its corresponding CF that, in turn, was also in charge of carrying the teaching signal corresponding to the negative or the positive part of the error being estimated. Every subgroup of PCs finally inhibited a cell of the DCN that, again, counterbalanced the negative or the positive part of the error curve.

Case study B; the cerebellar circuit was modeled within a closed loop designed to operate a simulated robotic arm of 3 DoFs (see Control Loop Section and **Figure 3B**). In this set up, we scaled up the cerebellar model. Within this loop, 60 PCs inhibited six DCN that, in turn, counterbalanced the error undergone by the simulated robotic arm. CFs (6) were assumed to transmit the difference between the simulated robotic arm desired-trajectory curves and the actual ones. CFs closed the loop providing teaching input to the PCs. The PC layer was divided into three groups of 20 PC cells each that were in charge of correcting their corresponding simulated robotic arm DoF. Each group was also subdivided into two groups of 10 Purkinje cells and innervated by each granular layer cell. Each subgroup of the PCs was aimed to provide the positive or negative necessary corrections. Each PC subgroup was innervated by its corresponding CF which, in turn, carried the teaching signal corresponding to either the negative or the positive part of the actual error at each DoF. Every group of PCs finally inhibited a cell of the DCN that, again, counterbalanced the negative or the positive part of the actual error.

- ***Climbing fibers (CFs):*** (2) Climbing fibers in case study A. (6) Climbing fibers in case study B. Each CF carried the teaching spikes (obtained from error signals) from the IO to a PC subgroup. CF cell response followed a probabilistic Poisson process. Given the normalized error signal $\varepsilon(t)$ and a random number $\eta(t)$ between 0 and 1, the cell fired a spike if $\varepsilon(t) > \eta(t)$; otherwise, it remained silent (Boucheny et al., 2005; Luque et al., 2011a). In this way, a single spike reported accurately timed information regarding the instantaneous error; furthermore, the probabilistic spike sampling of the error ensured that the whole error region was accurately represented over trials with a constrained CF activity below 10 spikes per second, per fiber. Hence, the error evolution is accurately sampled even at a low frequency (Carrillo et al., 2008; Luque et al., 2011a). This firing behavior is similar to the ones obtained in physiological recordings (Kuroda et al., 2001; **Figure 1E**).
- ***Deep Cerebellar Nuclei (DCN):*** (2) Deep Cerebellar Nucleus cells in case study A, (6) Deep Cerebellar Nucleus cells in case study B. The generated DCN spike train is translated into meaningful analog output signals by using a Finite Impulse Response filter (FIR). We adopted this mathematical approach (Schrauwen and van Campenhout, 2003) because we assumed, at this stage, that the goal is to decode rather than to analyze the behavior of biological neurons.

Defining the spike train as $x(t) = \sum_{j=t}^{N} \delta(t - t_j)$, where t_j stands for the set of firing times of the corresponding neuron, N is the number of events in the spike train, and being the FIR response defined as *h(t),* then the stimulus can be written as follows (Equation 1):

$$stimulus\,(t) = \left(h * x\right)(t) = \sum_{j=t}^{N} h\left(t - t_j\right) \quad j = 1 \; to \; N \tag{1}$$

Despite the widespread use of FIR filters for such purpose, an undesired delay is introduced in the generated analog signal. This delay is strongly related to the number of filter coefficients and to the shape of the filter kernel. In order to mitigate this effect and to make the conversion more efficient, an exponentially-decaying kernel is implemented Equation (2). At each time step, the output signal value only depends on its previous value and on the input spikes in the same time step and, therefore, this filter is implemented by recursively updating the last value of the output signal. Actually, the choice of such exponential kernel is double folded. The kernel is able to mitigate the delay problem and bears a strong resemblance to postsynaptic currents (van Rossum, 2001; Victor, 2005), thus facilitating a biological interpretation. Furthermore, as demonstrated in Luque et al. (2014a), this FIR filter is equivalent to an integrative neuron (**Figure 1F**).

$$Kernel = e^{-\frac{M}{\tau}},\ \text{where}\ M = 1 \tag{2}$$

where M is the number of filter taps (one-tap per integration step 0.002 s) and τ is the decaying factor.

In case study A, the cerebellum output was generated by a single group of these DCN cells; one of the cells handled positive error corrections whereas the other one handled negative error corrections. Each DCN neuron received excitation from every MF and inhibition from its corresponding 10 PC group. In this way, the sub-circuit PC–DCN–IO was organized in a single microzone.

In case study B, the cerebellum output was generated by three groups of these cells. The cerebellar corrective output (torque) for each DoF was encoded by a group of these cells (two subgroups per DoF) whose activity provided corrective actions to the specified robot-arm commands. Each neuron group in the DCN received excitation from every MF and inhibition from its corresponding PC group. In this way, the sub-circuit PC–DCN–IO was organized in three microzones.

In both cases, DCN synaptic conductances were set to initial values of 0 nS at the beginning of the simulation, and were modified by the STDP mechanisms during the training process.

Synaptic Plasticity

The impact of distributed cerebellar synaptic plasticity on gain adaptation and learning consolidation using close-loop experiments has been explored. It has been assumed that there are at least three learning sites, one in the cerebellar cortex (PF–PC) and two at the DCN (MF–DCN and PC–DCN), all of them generating LTP and/or LTD. Unlike the previous analog cerebellar model (Garrido et al., 2013a; Luque et al., 2014b), where each cerebellar layer was implemented as a set of parameter values corresponding to the firing rate of the neural population, the spiking model presented here preserves the timing information of the elicited spikes at each cerebellar layer and the adaptation mechanisms are based on Spike Time Dependent Synaptic Plasticity (STDP). Now we summarize these multiple forms of synaptic plasticity.

PF–PC Synaptic Plasticity

This is, by far, the most widely investigated cerebellar plasticity mechanism as evidenced by the vast number of studies supporting the existence of multiple forms of LTD (Ito and Kano, 1982; Boyden et al., 2004; Coesmans et al., 2004) and LTP (Hansel et al., 2001; Boyden et al., 2004; Coesmans et al., 2004) plasticity mechanisms. Two important features were considered when implementing this synaptic plasticity mechanism:

i. The synaptic efficacy change for each PF connection had to be driven by pre-synaptic activity (spike-timing-dependent plasticity) and had to be instantaneous.
ii. Since the sensorimotor pathway delay is roughly ∼100 ms, the learning mechanism had to learn to provide *corrective predictions* to compensate this inner sensorimotor delay (**Figure 2A**).

To this aim, this plasticity mechanism was implemented including LTD and LTP as follows (Luque et al., 2011a):

- LTD produced a synaptic efficacy decrease when a spike from the IO reached the target PC through the CF. The amount of the weight decrement depended on the previous activity arrived through the PF. This previous activity was convolved with an integrative kernel as defined by Equation (3).

$$k(x) = e^{-x} \cdot \sin(x)^{20} \tag{3}$$

  where x is used as intermediate variable to get a compacted definition of the kernel, x is then substituted in Equation (4) by the independent variable t.

  This mainly took into account those PF spikes which arrived 100 ms before the CF spike arrival. This correction was facilitated by a time-logged "eligibility trace," which evaluated the past activity of the afferent PF (Sutton and Barto, 1981; Barto et al., 1983; Kettner et al., 1997; Boucheny et al., 2005). This trace aimed to calculate the correspondence in time between spikes from the IO (error-related activity) and the previous activity of the PF that was temporally correlated to this error signal. The eligibility trace idea stemmed from experimental evidence showing that a spike in the climbing fiber afferent to a Purkinje cell was more likely to depress a PF–PC synapse if the corresponding PF had been firing between 50 and 150 ms before the IO spike (through CF) arrived at the PC (Kettner et al., 1997; Boucheny et al., 2005; Ros et al., 2006).
- LTP produced a fixed increase in synaptic efficacy each time a spike arrived through a PF to the corresponding targeted PC as defined by Equation (4). This mechanism allowed us to capture how the LTD process could be inverted when the PF stimulation was followed by spikes from the IO or by a strong depression of the Purkinje cell membrane potential (according to neurophysiologists studies; Lev-Ram et al., 2003).

The chosen mathematical-model kernel allowed accumulative computation in an event-driven simulation scheme as adopted by the EDLUT simulator (Ros et al., 2006; Luque et al., 2011a,b). This avoids the necessity of integrating the whole correlation kernel upon each new arrival of a spike. This correlation kernel, despite being computationally efficient, suffered from a second marginal peak whose impact could be considered to be negligible (<5% of the main peak height). This is indicated in the following Equation (4).

$$\begin{aligned} LTD.\triangle W_{PF_j-PC_i}(t) &= \int_{-\infty}^{IO_{spike}} k\left(\frac{t - t_{IO_{spike}}}{\tau_{LTD}}\right) \cdot \delta_{GC_{spike}}(t) \cdot dt \\ &\qquad \text{if } PF_j \text{ is active at } t \\ LTP.\triangle W_{PF_j-PC_i}(t) &= \alpha\ \text{Const. otherwise} \end{aligned} \tag{4}$$

where $\Delta W_{PFj-PCi}(t)$ represents the weight change between the j^{th} PF and the target i^{th} PC. τ_{LTD} stands for the time constant that compensates the sensorimotor delay and δ_{GC} stands for the delta Dirac function defining a GC spike. For an in-depth review of the inner features of this kind of kernel (see Ros et al., 2006; Luque et al., 2011a).

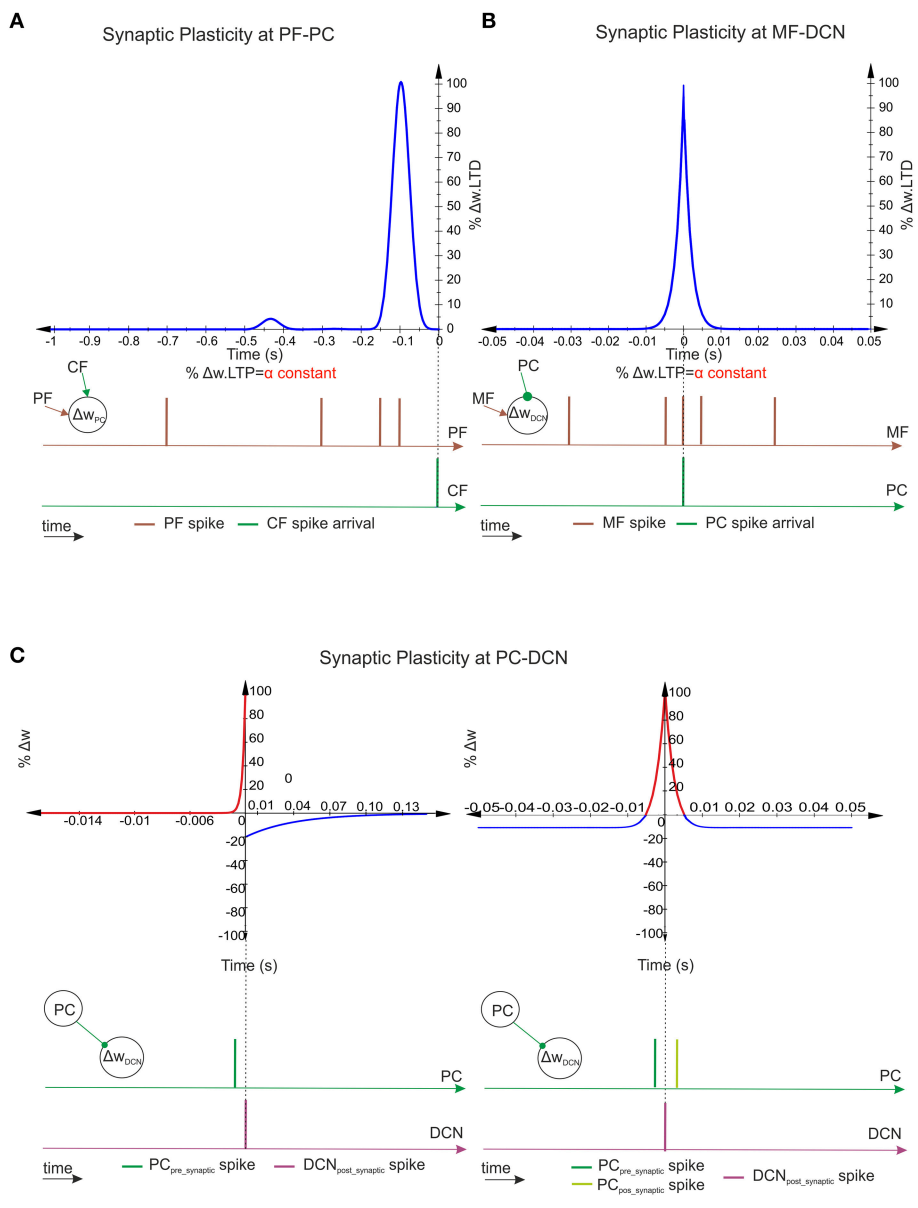


FIGURE 2 | Spike-timing–dependent learning rules at PF–PC, MF–DCN, and PC–DCN synapses. (A) Representation of PF–PC LTD correlation kernel. A synaptic efficacy decrease occurs at PF–PC innervations when a spike from the IO reaches a target PC through a CF. The weight decrement percentage depends on the previous activity arrived through the corresponding PF (100 ms before the CF spike arrival) in order to compensate the sensorimotor pathway. The PF–PC LTP—synaptic-efficacy increase is considered to remain constant. **(B)** Representation of MF–DCN LTD correlation kernel. A synaptic efficacy decrease occurs at MF–DCN innervations when a spike from the PC reaches a target DCN. Near-coincident pre- and post-synaptic MF–DCN spikes which arrive close to PC–DCN spike arrival cause a depression at MF excitatory synapses. MF–DCN LTP-synaptic-efficacy increase is also considered to remain constant. **(C)** Representation of two PC–DCN alternative correlation kernels. Classical inhibitory STDP modifies the synapse efficacy at PC–DCN innervations depending on DCN activity. Near-coincident pre-synaptic PC–DCN spikes before post-synaptic DCN-action potentials cause long-term potentiation action whereas PC–DCN spike arrivals after post-synaptic DCN-action potentials cause long-term depression action. The second inhibitory-STDP kernel potentiates the synapse efficacy at PC–DCN innervations after a DCN-action potential each time a near-coincident pre- and postsynaptic PC-spike arrives whereas every presynaptic PC spike leads to synaptic depression.

MF–DCN Synaptic Plasticity

MF–DCN synaptic plasticity has been reported to depend on the intensity of the DCN cell excitation (Racine et al., 1986; Medina and Mauk, 1999; Bastian, 2006; Pugh and Raman, 2006; Zhang and Linden, 2006; **Figure 2B**). It has been implemented by means of a mathematical kernel defined by Equation (5):

$$k(x) = e^{-|x \cdot \beta|} \cdot \cos(x)^2 \tag{5}$$

where x is used as intermediate variable to get a compacted definition of the kernel, x is then substituted in Equation (6) by the independent variable t. β is a constant factor used for mitigating the impact of the second marginal peak that this kernel suffers.

$$LTD.\triangle W_{MF_j-DCN_i}(t) = \int_{-\infty}^{+\infty} k\left(\frac{t - t_{PC_{spike}}}{\sigma_{MF-DCN}}\right) \cdot \delta_{MF_{spike}}(t) \cdot dt \text{ if } PC_j \text{ is active at } t$$
$$LTP.\triangle W_{MF_j-DCN_i}(t) = \alpha \text{ Const. otherwise} \quad (6)$$

where $\Delta W_{MFj-DCNi(t)}$ denotes the weight change between the j^{th} MF and the target i^{th} DCN, σ_{MF-DCN} stands for the window-time width of the kernel, and δ_{MF} stands for the delta Dirac function that defines a MF spike. As evidenced, there is no need to compensate the sensorimotor pathway delay at this plastic site since it is already compensated by the PF–PC kernel. LTD and LTP actions are then characterized as follows:

- LTD produced a synaptic efficacy decrease when a spike from the PC reached a targeted DCN. The amount of the weight decrement depended on the activity arrived through the MFs. This activity was convolved with the integrative kernel defined in Equation (5). This mainly considered those MF spikes that arrived after/before the PC–DCN spike arrival within the window-time-width defined by the kernel.
- LTP produced a fixed increase in synaptic efficacy each time a spike arrived through an MF to the corresponding targeted DCN as defined by Equation (6). This mechanism allowed the compensation of the LTD if necessary and prevents any weight saturation as proven in Luque et al. (2011a).

Despite the fact that this MF–DCN synaptic-plasticity mechanism looks very much like the mathematical expression given by PF–PC synaptic plasticity, it presents two significant differences:

i. The first one lies on the reduced capability of MFs, compared to PFs, to generate sequences of non-recurrent states. As aforementioned, the MF–DCN activity compared to the analog approaches described at Garrido et al. (2013a) and Luque et al. (2014b) is now capable of codifying the passage of time. It does so by using groups of active mossy neurons that are sequentially activated. However, it uses a significantly lower number of consecutive non-recurrent time stamps than the 500 able to be generated by the granular layer (Yamazaki and Tanaka, 2007b, 2009; Yamazaki and Nagao, 2012).

   In Garrido et al. (2013a) and Luque et al. (2014b), the MF–DCN connection was implemented as a state generator able of generating only one state; the amplitude at this state was equivalent to the base current able to excite DCN cells. Plasticity at this site was capable of varying the amount of injected current that operated DCN by modifying the amount of excitation that DCN received at this connection (gain controller). However, a single-state generator was not able to generate the mentioned 500 time stamps (PF–PC). Nevertheless, having a state generator made out of clusters of non-overlapped neurons at MF–DCN allows us to roughly store or "translate" the timing sequence that is generated by a state generator holding 500 states. Given the fact that the cerebellar networks holds 2000 GCs, the simulation step-size is 2 ms, and the trajectory time is 1 s; 500 different states are, therefore, generated by groups of four non-overlapped neurons at PF–PC level. The 100 MFs have been clustered in groups of four non-overlapped neurons obtaining 25 states at MF–DCN level to roughly store the PF–PC synaptic weight distribution facilitated by those 500 different states.

ii. The second main difference concerns the connection driving LTD and LTP. Whilst the PF–PC plasticity was driven by the CF activity, the MF–DCN plasticity was driven by the PC activity. This mechanism optimized the activity range in the whole inhibitory pathway comprising MF–PF–PC–DCN connections: high PC activity caused MF–DCN LTD, whilst low PC activity caused MF–DCN LTP. This mechanism implemented an effective cerebellar gain controller able to adapt its output activity to minimize the amount of inhibition generated in the MF–PF–PC–DCN inhibitory loop.

PC–DCN Synaptic Plasticity

PC–DCN synaptic plasticity was reported to depend on the intensity of DCN and PC cells (Morishita and Sastry, 1996; Aizenman et al., 1998; Ouardouz and Sastry, 2000; Masuda and Amari, 2008). Moreover, plasticity at inhibitory synapses was revealed as a fundamental homeostatic mechanism in balancing the excitatory and inhibitory cell inputs (Medina and Mauk, 1999; Kleberg et al., 2014) at DCNs capable of conforming synaptic memories related to activity patterns (Vogels et al., 2011). Taking inspiration from (Medina and Mauk, 1999) and recent studies (Vogels et al., 2011; Kleberg et al., 2014), the synaptic plasticity mechanism was implemented following two possible valid kernels (**Figure 2C**):

i. A classical inhibitory-STDP learning rule (iSTDP; Equation 7)

$$LTP.\triangle W_{PC_j-DCN_i}(t) = e^{-\left(\frac{t_{DCN_{post}} - t_{DCN_{pre}}}{\tau_1}\right)} \quad if\ t_{DCN_{post}} > t_{DCN_{pre}}$$
$$LTD.\triangle W_{PC_j-DCN_i}(t) = e^{-\left(\frac{t_{DCN_{pre}} - t_{DCN_{post}}}{\tau_2}\right)} \quad if\ t_{DCN_{pre}} > t_{DCN_{post}} \quad (7)$$

   where $\Delta W_{PCj-DCNi(t)}$ is the weight change between the j^{th} PC and the target i^{th} DCN. τ_1 stands for the time constant for the LTP expression and τ_2 stands for the time constant for the LTD expression.

ii. An inhibitory-STDP learning rule based on near-coincident pre and postsynaptic spikes able to potentiate inhibitory synapses, whereas every presynaptic spike causes synaptic depression (Equation 10).

$$\triangle W_{PC_j-DCN_i}(t) = \int_{-\infty}^{+\infty} \left(LTP_{max} \cdot e^{-\left|\frac{t_{DCN_{post}} - t_{DCN_{pre}}}{\sigma_{PC-DCN}}\right|} \cdot \cos\left(\frac{t_{DCN_{post}} - t_{DCN_{pre}}}{\sigma_{PC-DCN}}\right)^2 - LTD_{max}\right) \cdot dt \quad (8)$$

where $\Delta W_{PCj-DCNi(t)}$ is the weight change between the j^{th} PC and the target i^{th} DCN, σ_{PC-DCN} stands for window-time width of the kernel, and LTP$_{max}$/LTP$_{max}$ stand for the maximum weight depression or potentiation change per simulation step.

These two plausible kernels mainly consider those spikes received by DCN through PC innervation within the window-time-width defined per each kernel.

Cerebellar Control Loop

Case study A (Figure 3A)

The adopted control loop for the cerebellar architecture of Case-study-A was based on the traditional forward cerebellar control architecture. Within this architecture, the cerebellum attempted to minimize the existing difference between the controlled variable (actual cerebellar output value) and the reference variable (set point) via manipulation of the controlled variable. The reference variable (Equation 9) was a 1-s curve (2 ms time-step simulation) made out of Gaussian functions that was repeatedly iteratively presented to the cerebellar model.

$$\text{reference variable}\,(t) = e^{-\frac{\left(t-\frac{T}{4}\right)^2}{\sigma_{ref}^2}} - e^{-\frac{\left(t-\frac{3\cdot T}{4}\right)^2}{\sigma_{ref}^2}} \qquad (9)$$

where σ_{ref} stands for the Gaussian standard deviation and T stands for the time period.

This curve (Equation 9) changed its direction and module from the minimum possible value (normalized) to its maximum possible value twice per period. The cerebellar output action demanded a fine balance between the negative/positive output micro-complex actions to match the reference variable. It is worth mentioning that the IO frequency ranged between 1 and 10 Hz. Thus, according to the network already presented, each IO codified whether the error was positive or negative during 0.5 s (depending on the activated CF). Hence, no more than five spikes per IO and period (1 s) were obtained in the worst possible scenario. These directional and module changes combined with the IO biological low rate sampling constraint made the cerebellum operate at the limits of its learning performance.

Case study B (Figure 3B)

The adopted control loop was based on the traditional feed-forward architecture along with a crude inverse dynamic model of the simulated robotic arm. An inverse kinematic module translated the desired trajectory into arm-joint coordinates and fed an inverse dynamic module based on a recursive Newton-Euler algorithm. This algorithm generated crude step-by-step motor commands (torques) corresponding to the desired trajectory.

In light of some studies, the central nervous system has been suggested to plan and execute sequentially voluntary movements. In accordance to this hypothesis, the brain might first plan the optimal trajectory in task-space coordinates, translate them into intrinsic-body coordinates, and finally, generate the necessary motor commands (Houk et al., 1996; Nakano et al., 1999;

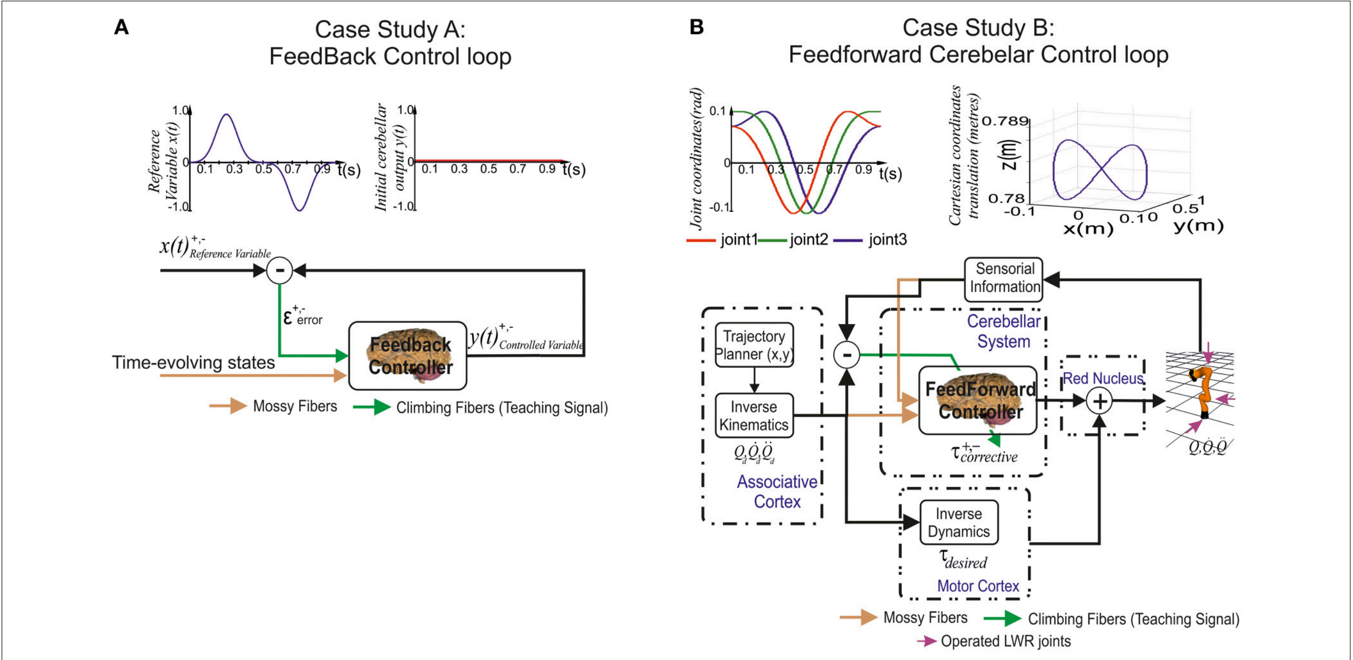


FIGURE 3 | Case study cerebellar control loops. (A) Case study A, the adaptive cerebellar module embedded in a control loop delivers corrective actions to compensate the existing difference between a controlled variable [actual cerebellar output value $y(t)$] and a demanding reference variable [set point $x(t)$]. **(B)** Case study B**,** the adaptive cerebellar module embedded in a feed forward control loop delivers corrective torque values ($\tau_{\text{corrective}}$) to compensate for deviations in the crude inverse dynamic module when manipulating an object of significant weight along an eight-like trajectory. In this feed-forward control loop, the cerebellum receives a teaching error-dependent signal and the desired arm state (Q_d, $\dot{Q}_d$, $\ddot{Q}_d$) so as to produce the adaptive corrective actions.

Todorov, 2004; Hwang and Shadmehr, 2005; Izawa et al., 2012; Passot et al., 2013). According to these studies, the association cortex would be in charge of providing the desired trajectory in body coordinates and conveying them to the motor cortex which, in turn, would generate the optimal motor commands to operate our limbs. On the one hand, the spinocerebellum–magnocellular red nucleus system is thought to hold an internal neural accurate model of the musculoskeletal body dynamics learnt through sensing voluntary movements (Kawato et al., 1987). On the other hand, the cerebrocerebellum–parvocellular red nucleus system is thought to provide a crude internal neural model of the inverse-dynamics of the musculoskeletal system (Kawato et al., 1987). The crude inverse-dynamic model shall work conjointly with the dynamic model (given by the spinocerebellum–magnocellular red nucleus system) in order to get the ongoing motor commands updated to match a possible predictable error when executing a movement.

Together with the feed forward control loop, a simulated-light-weight robot (LWR) arm was integrated. The simulated-robot-plant physical characteristics can be dynamically modified to manipulate different payloads (punctual masses). This LWR (Hirzinger et al., 2000; Albu-Schäffer et al., 2007) model is a 7-DOF arm robot consisting of revolute joints where only the first (labeled as Q1), second (Q2), and fifth joint (Q3) were operated in our experiments while maintaining the others fixed (rigid).

Similarly to Case study A, the main aim when selecting a benchmark trajectory was to challenge the cerebellar learning limits. Case study B needed to reveal the dynamic properties of a simulated-robot-plant. Choosing fast movements in a smooth pursuit task consisting of vertical and horizontal sinusoidal components (Kettner et al., 1997; van Der Smagt, 2000; 1 s for the whole target trajectory) allowed us to study how inertial components (when manipulating objects) were inferred by the cerebellar architecture (Luque et al., 2014b). The selected target trajectory described an "8-shape" defined by Equation (12) in joint coordinates.

$$\mathbf{Q}_n(t) = A_n \cdot \sin\left(\left(-4 \cdot \pi \cdot t^3 + 6 \cdot \pi \cdot t^2\right) + C_n\right) \\ \text{where } n = \left\{1, \cdots, \text{ number of links}\right\} \qquad (10)$$

where A_n and $C_n = n{\cdot}\pi/4$ represent the amplitude and phase of each robot joint. The followed trajectory is based on cubic spline technique so as to provide continuity and a zero initial velocity per link, which fulfills the implementation requirements of a physical robot controller. This trajectory is easy to perform despite the non-linearity in the robot joint angles, since joint velocities and accelerations are constricted to small bounds depending on the amplitude and phase. To finally quantify and evaluate the movement performance in terms of accuracy, the average of the Mean Absolute Error (MAE) per robot joint was calculated. The estimation of this measurement was monitored in each trial, thus allowing the quantification of the global-movement accuracy evolution during the learning process.

RESULTS

We tested the hypothesis of cerebellar gain-controller operation assuming that the MF–DCN synaptic weights were capable of obtaining the maximum corrective cerebellar values whilst the difference between the maximum and minimum corrective cerebellar values were supplied by PC–DCN synaptic weights. We also tested the learning consolidation hypothesis by endowing these two connectivity sites with plasticity, thereby generating an internal adaptive gain controller fully compatible with the two-state learning mechanism proposed by Shadmehr and Brashers-Krug (1997), Shadmehr and Holcomb (1997), Medina and Mauk (2000), and Ohyama et al. (2006). Whilst case study A, due to its inherent simplicity, helped to demonstrate and validate our premises, case study B helped to extrapolate our premises to a more demanding scenario where the cerebellar model delivered to a simulated-robotic arm the corrective actions needed to compensate for dynamic deviations produced when manipulating heavy point masses. Furthermore, case B also helped to evaluate how the distributed learning scheme was scalable in terms of joints.

Illustrative movies of learning simulations for case study A and case study B during the manipulation of a 6-kg load are available in the Supplemental Material.

MF–DCN STDP Allows Learning Consolidation

In order to determine the impact of MF–DCN STDP in learning consolidation, in case-study-A, the cerebellar network was equipped with plasticity at PF–PC and MF–DCN synapses. Our first simulation was carried out to demonstrate not just, how MF–DCN could implement a gain-controller, but also how the PF–PC learning was transferred into MF–DCN synapses.

Within the feed-forward control loop, the cerebellum in case study A, attempted to minimize the existing difference between the controlled variable (current cerebellar output) and the reference variable (following the 1-s curve made out of Gaussian functions; **Figure 4C**). The reference variable was iteratively presented over 2500 iterations. PF–PC synaptic conductances were set to an initial value of 5 nS, MF–DCN initial conditions started from zero, and PC–DCN synaptic weights were fixed with pre-calculated values that ensured a proper inhibitory PC–DCN action. In order to better discern the synaptic weight distribution shape that was transferred from PF–PC synapses into MF–DCN, the initial synaptic weights at those synaptic sites were set to equal values. This set-up configuration facilitated the perception at a glance of a continuous surface representing PF–PC synaptic distribution copying the reference variable. We also made simulations with random initialization of synaptic weights leading us to similar results but in these simulations, it was difficult to obtain a visual verification of the learning consolidation process (see Supplementary Material).

As evidenced, the reference variable changed its direction and module from the minimum possible value (normalized) to its maximum possible value twice each period, which required a fine balance between the cerebellar micro-complex negative/positive output (**Figure 4C**). Despite this demanding

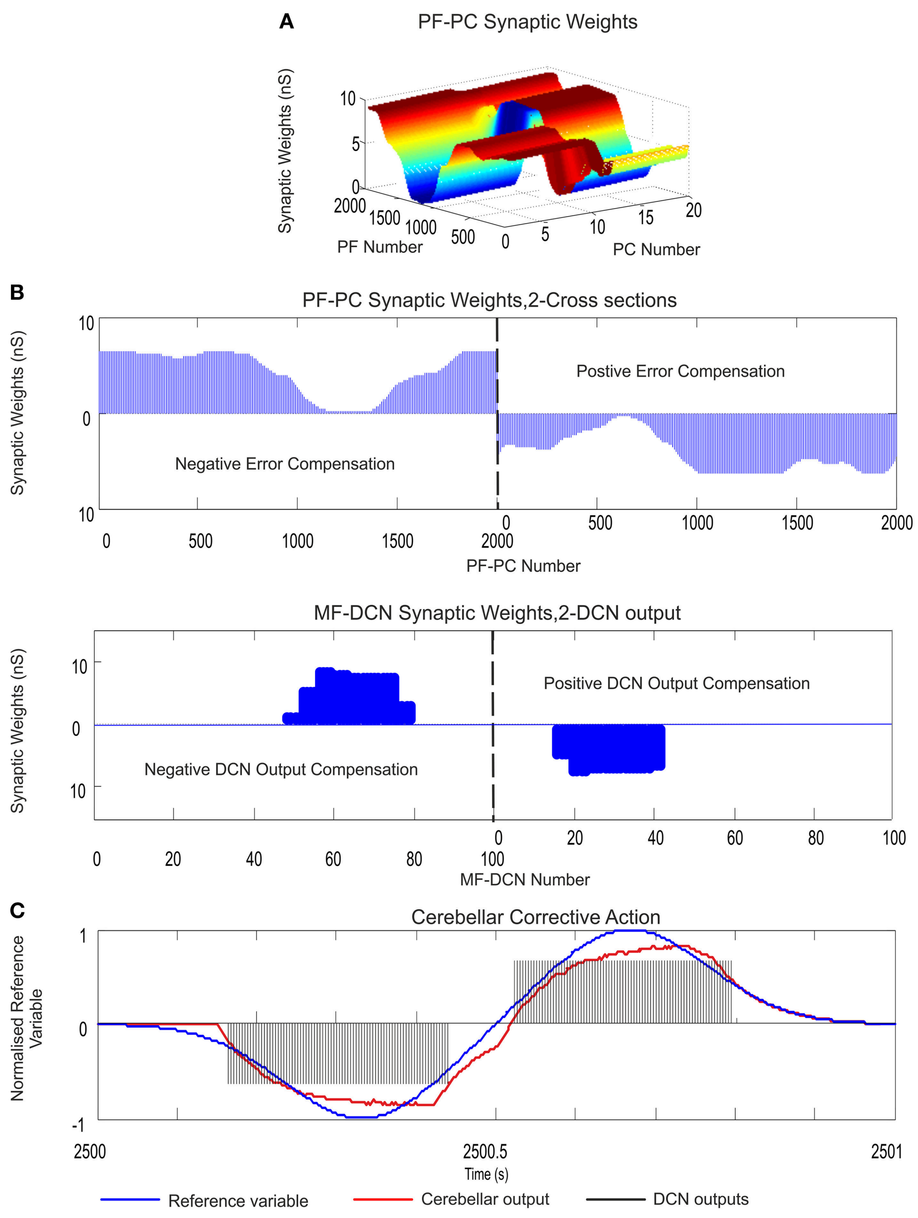


FIGURE 4 | PF–PC and MF–DCN learning rule interplay: learning consolidation. Case study A. PF–PC and MF–DCN synaptic weight distribution at the end of the learning process. The cerebellum tries to counterbalance the existing mismatch between the actual cerebellar output and the reference variable; a 1-s curve made out of Gaussian functions which is iteratively presented to the cerebellum over 2500 iterations. The exponential weight distribution at PF–PC shows that the corrective action is properly stored at these afferents. Learning rule at PC–DCN connections is deactivated. Synaptic weights are fixed with pre-calculated values ensuring a proper inhibitory PC–DCN action. **(A)** Two micro-complexes that are conformed by 10 PCs each and innervated, in turn, by 2000 PFs. These micro-complexes are responsible for the correct balance between the negative and the positive cerebellar correction. One micro-complex is in charge of delivering the positive corrective action whilst the other one delivers the negative corrective action. Each of the two output DCN cells is, in turn, innervated by one of the two micro-complexes. **(B)** The Gaussian-like weight distribution at PF–PC is transferred in counter phase to MF–DCN synapses. The reduced MF number of non-recurrent states enables learning consolidation; however, the obtained synaptic weight-distribution shape adopts a discretized version of the PF–PC weight-distribution shape. **(C)** The injected error is properly counterbalanced thanks to the action of these two learning laws. DCN output activity (spikes in black) is transformed into its proportional analog value (in red) and, later on, subtracted from the reference variable (in blue).

scenario, after the synaptic weight adaptation process at PF–PC connections (**Figures 4A,B**), the Gaussian shape that the reference variable presented, was copied and stored at PF–PC synaptic weights, thus constituting the first learning stage needed to deliver the cerebellar corrective action. Then, PF–PC learning triggered MF–DCN learning process which was able to lead MF–DCN synaptic weights to their local maximum values allowing plasticity to store temporally correlated information (the weight distribution at PF–PC was inversely copied at MF–DCN; **Figure 4D**).

The constrained capability of MFs, compared to PFs, when generating sequences of non-recurrent states was immediately reflected in the MF–DCN synaptic weight shape (**Figure 4B**). Although the reduced MF number of non-recurrent states enabled learning consolidation, the obtained synaptic weight-distribution shape, through the adaptation process at this site, was forced to adopt an inverse discretized version of the PF–PC weight-distribution shape.

MF–DCN and PC–DCN STDP Interplay toward Adaptive Gain Controller

In order to evaluate the existing interplay amongst different forms of plasticity at PF–PC, MF–DCN, and PC–DCN synapses, respectively, case-study-A cerebellar network was sequentially added with the aforementioned adaptive mechanisms (Equations 3, 5, and 7). In previous works, we demonstrated that plasticity at PF–PC synapses could not account for preventing PC activity saturation *per se* (Garrido et al., 2013a; Luque et al., 2014b). To circumvent this limitation, MF–DCN and PC–DCN plasticity mechanisms were implemented, thus allowing PC activity to keep on operating within its optimal working range. Nevertheless, how such analog plasticity mechanisms would be re-designed and counterbalanced to take into account the spiking cerebellar nature remained an open issue.

This has motivated the work presented here. Case-study-A cerebellar network attempted to minimize the existing difference between the controlled variable and the reference variable (1 s duration) over 5000 iterations (**Figure 5C**). PF–PC synaptic conductances were set to an initial value of 5 nS, MF–DCN initial conditions started from zero (**Figure 5D**), and PC–DCN synaptic weights were set to either zero initial values, random values, or higher values than needed (**Figures 5E,F**). As expected, the STDP learning rule located at this site was able to self-regulate PC-DCN synaptic weights in order to adequate the optimal working range demanded by both DCN and PC (**Figures 5E,F**).

Whilst the consolidation process was settling down (what was learnt at PF–PC synapses (**Figure 5A**) was transferred in counter phase to MF–DCN synapses; **Figure 5B**), it was possible to verify the double-learning time-scale behavior already indicated in recent behavioral and computational studies (Shadmehr and Brashers-Krug, 1997; Shadmehr and Holcomb, 1997; Medina and Mauk, 2000; Ohyama et al., 2006; Garrido et al., 2013a; Luque et al., 2014b; **Movies S1**, **S2** in Supplementary Material). MF–DCN and PC–DCN averaged synaptic weights (averaged gains) stabilized slower than those at PF–PC synapses, since learning at MF–DCN and PC–DCN synapses depended on the PC activity. As shown in **Movies S1**, **S2**, there was a fast learning process, in which temporal information was inferred and stored at PF–PC synapses. Meanwhile, there also was a slow learning process, in which the adaptation of cerebellar excitatory and inhibitory gain values in the DCN took place. This second slow learning process could be, in turn, split into two components related to the MF–DCN and PF–PC connections with time-constants of 750–2500 trials and 2500–5000 trials, respectively. **Figures 5D,E**.

iSTDP Shape Impacts on PC–DCN Synapses

Within the case-study-A cerebellar configuration, PC–DCN iSTDP remains as the only inhibitory pathway to the cerebellar nuclei, and therefore, the only mechanism capable of reducing the cerebellar output and preventing MF–DCN from saturation. iSTDP is known to act as a fundamental mechanism in both; balancing the excitatory and inhibitory DCN inputs (Medina and Mauk, 1999; Kleberg et al., 2014), and conforming synaptic memories related to activity patterns (Vogels et al., 2011). Nevertheless, the shape held by the iSTDP at PC–DCN synapses is not yet well-known (the exact adaptation mechanism remains an open issue).

In order to identify the influence that the iSTDP shape may exert on the cerebellar output, two biologically plausible learning kernels were tested. The first one was implemented following the traditional STDP Hebbian kernel shape (Equation 7) whereas the second one was implemented following Medina and Mauk approach (Medina and Mauk, 1999), also adopted in recent studies (Vogels et al., 2011; Kleberg et al., 2014; Equation 10). The first step that needed to be proven was the robustness of the shape of these two kernels. Based on Vogels et al. (2011), these two kernels suited well our experimental test-bench, since both fulfilled two main conditions: the postsynaptic activity potentiated the activated inhibitory synapses together with the fact that in absence of postsynaptic firing, the inhibitory synapses decayed.

Case-study-A was used for a comparative study of both approaches. Again, the cerebellar network attempted to minimize the existing difference between the controlled variable and the reference variable (1-s duration) over 10,000 iterations. PF–PC synaptic conductances were set to an initial value of 5 nS, MF–DCN initial conditions started from zero (**Figure 7A**), and PC–DCN synaptic weights were set to a zero initial value as well (**Figure 7B**). As expected, according to the aforementioned premises, both kernels showed a similar ability to correlate (more concretely, to reverse-correlate) the activity arriving from PCs with DCN output activity (**Figure 6B**). Both kernels did indeed obtain a similar behavior in terms of maximal reverse-correlation values and speed of convergence (**Figure 6A**).

Nevertheless, the second kernel exhibited a better performance in terms of stability and overall gain value (**Figure 7C**) but at the cost of a lower convergence speed (**Figure 7E**). Due to the initial conditions for DCN innervations were set initially to zero, a post-synaptic spike scenario dominated during the learning process, thus making the Hebbian approach faster than the symmetric kernel in terms of convergence speed (see **Figure 2C**). STDP Hebbian kernel shape has been traditionally used for spatiotemporal detection and learning of hidden spike patterns from a neural activity background by correlating post-synaptic and pre-synaptic activity (Masquelier et al., 2009). However, an inhibitory-STDP learning kernel based on near-coincident pre and post-synaptic spike seemed to be more useful for balancing the DCN excitation and inhibition inputs (**Figures 7B,C**) and for selectively propagating the correlated spiking activity from PC to DCN (**Figure 7D**).

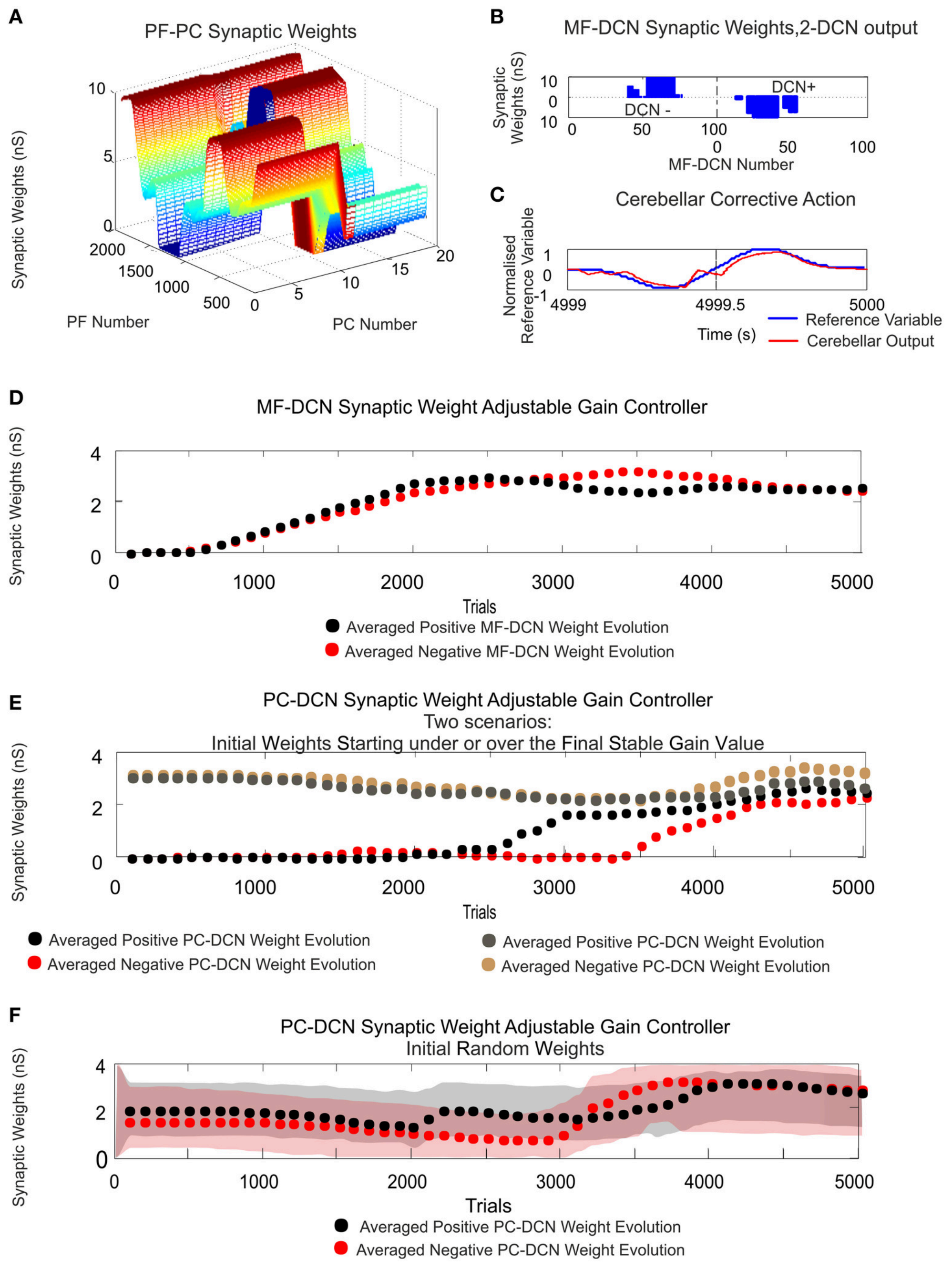


FIGURE 5 | MF–DCN and PC–DCN STDP learning rules working conjointly as an adjustable gain controller. Case study A. **(A)** PF–PC synaptic weight distribution at the end of the learning process. The cerebellum counterbalances the existing difference between the actual cerebellar output and the reference curve which is iteratively presented to the cerebellum over 5000 iterations. The Gaussian-like weight distribution at PF–PC synapses shows that the corrective action is properly stored at these afferents. **(B)** Two micro-complexes that are conformed by 10 PCs each and innervated, in turn, by 2000 PFs are responsible for managing the trade-off between the negative and the positive cerebellar corrective action. The Gaussian-like weight distribution at PFs is inversely transferred at MF–DCN synapses. **(C)** The reference curve acting as an error is counterbalanced. DCN output activity is transformed into its proportional analog value (in red) and, later on, subtracted from the reference variable (in blue). **(D,E)** The initial conditions established for synaptic weights at MF–DCN start from zero whilst PC–DCN innervations start from either a higher or lower value than needed. MF–DCN and PC–DCN averaged synaptic weights (averaged gains) get stabilized more slowly than those at PF–PC synapses, since learning at MF–DCN and PC–DCN synapses depended on the PC activity. MF–DCN and PC–DCN averaged synaptic weights (averaged gains) are modified when PF–PC weights tend to be saturated. This learning process at MF–DCN and PC–DCN connection can be split into two components with time-constants of 750–2500 trials and 2500–5000 trials, respectively. **(F)** The initial conditions established for synaptic weights at MF–DCN start from zero whilst PC–DCN innervations start from random values. Red and gray shaded areas delimit the synaptic weight space in which the synaptic PC–DCN synaptic values evolve during the learning process for each micro-complex. Dotted lines indicate the averaged PC–DCN synaptic weight value obtained per micro-complex. The learning rule at PC–DCN self-regulates the synaptic weights obtaining the optimal firing rate demanded by both DCN and PC.

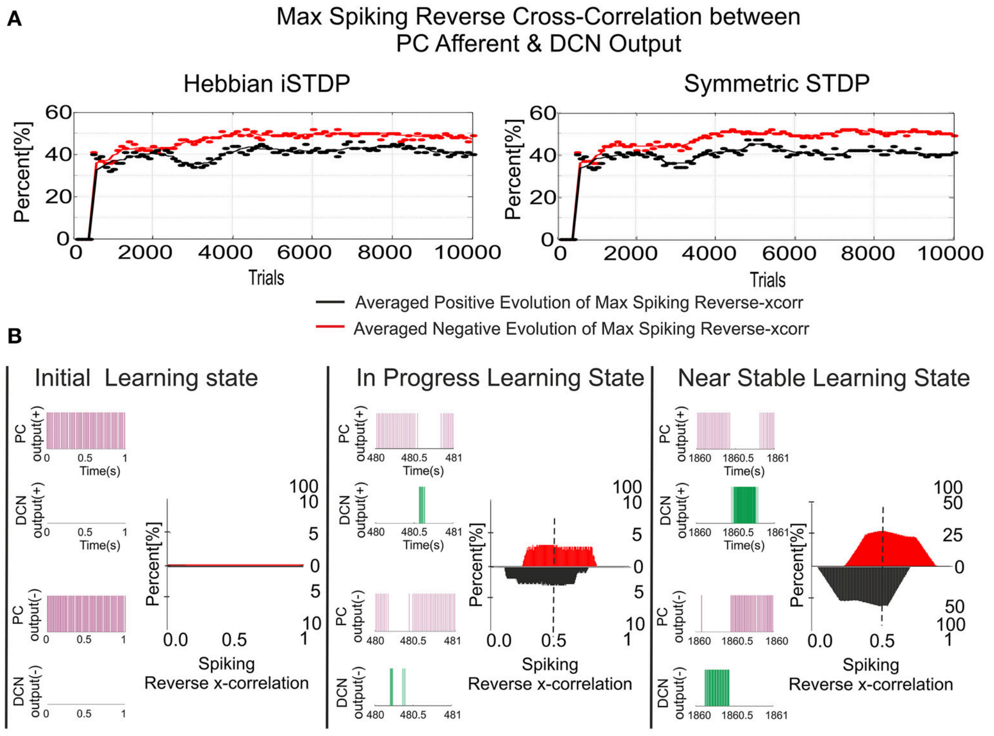


FIGURE 6 | Inhibitory plasticity at PC–DCN synapses with two possible STDP kernels, Classic i-STDP Hebbian rule, or Symmetric i-STDP. Case study A. **(A)** Reverse cross-correlation evolution at PC–DCN synapses. Since the postsynaptic activity potentiates the activated inhibitory synapses and the inhibitory synapses decay in the absence of postsynaptic activity in both kernels, they obtain a similar performance in terms of maximal reverse correlation values and convergence speed (Vogels et al., 2011). **(B)** Distribution of the reverse cross-correlation between spike trains from PC output and DCN output. For the sake of simplicity, only the Classic i-STDP Hebbian rule is shown. Three consecutive snapshots of the DCN input/output activity are shown, where the way in which a PC is able to modulate the DCN activity can be seen.

Testing Distributed Cerebellar Plasticity in a Robotic Manipulation Task

In order to quantify and extrapolate the aforementioned STDP distributed plasticity features, case-study B cerebellar network was faced with a more demanding scenario. Our last simulation was intended to show how the self-regulation of MF–DCN and PC–DCN synapses by means of STDP learning rules is able to deliver to a simulated-robotic arm the corrective actions needed to compensate for dynamic deviations produced when manipulating heavy point masses (6 kg).

Within the feed forward control architecture presented by case-study-B, the cerebellum attempted to minimize the existing difference between the controlled variable (current cerebellar output) and the reference variable (1 s eight-like trajectory to be followed by the robotic manipulator) during a manipulation task repeated 10,000 trials (**Figure 8**). PF–PC synaptic conductances were set to an initial value of 5 nS, MF–DCN initial conditions started from zero (**Figure 8D**), and PC–DCN synaptic weights were set to a zero initial value as well (**Figure 8E**). After DCN synaptic weight adaptation (**Figures 8D,E**), the cerebellum was able to deliver proper corrective torques reducing the error of the robot-arm movement (**Figures 8F,G**). Once the synaptic weights were stabilized, both PC and DCN neurons exploited their dynamic gain adaptation range (**Figures 8D,E**) allowing the cerebellum to operate near its optimal performance.

The cerebellum exhibited its ability to act as both an adaptive gain-controller (**Figures 8D,E**) and a distributed-learning storage architecture (what was learned at PF–PC synapses (**Figure 8A**) was then transferred in counter phase to MF–DCN synapses; **Figures 8B,C**). However, the difference between controlled and reference variable was not directly related because the cerebellar corrective action was delivered in torque commands (**Figure 3B**) and the proprioception state estimations were acquired in joint-angle coordinates.

It should be clarified that the proposed STDP mechanisms, and therefore their involvements, are not restricted to any specific test-bed framework, and could be extrapolated to other common but simpler test-bed frameworks such as EBCC and VOR.

DISCUSSION

This work presents a mechanistic spiking cerebellar model endowed with several STDP learning rules located at different synaptic sites. They are tested embedded in close-loop simulations. These close-loop simulations challenge the cerebellum with two tasks with different degrees of complexity. However, the main observation regarding the

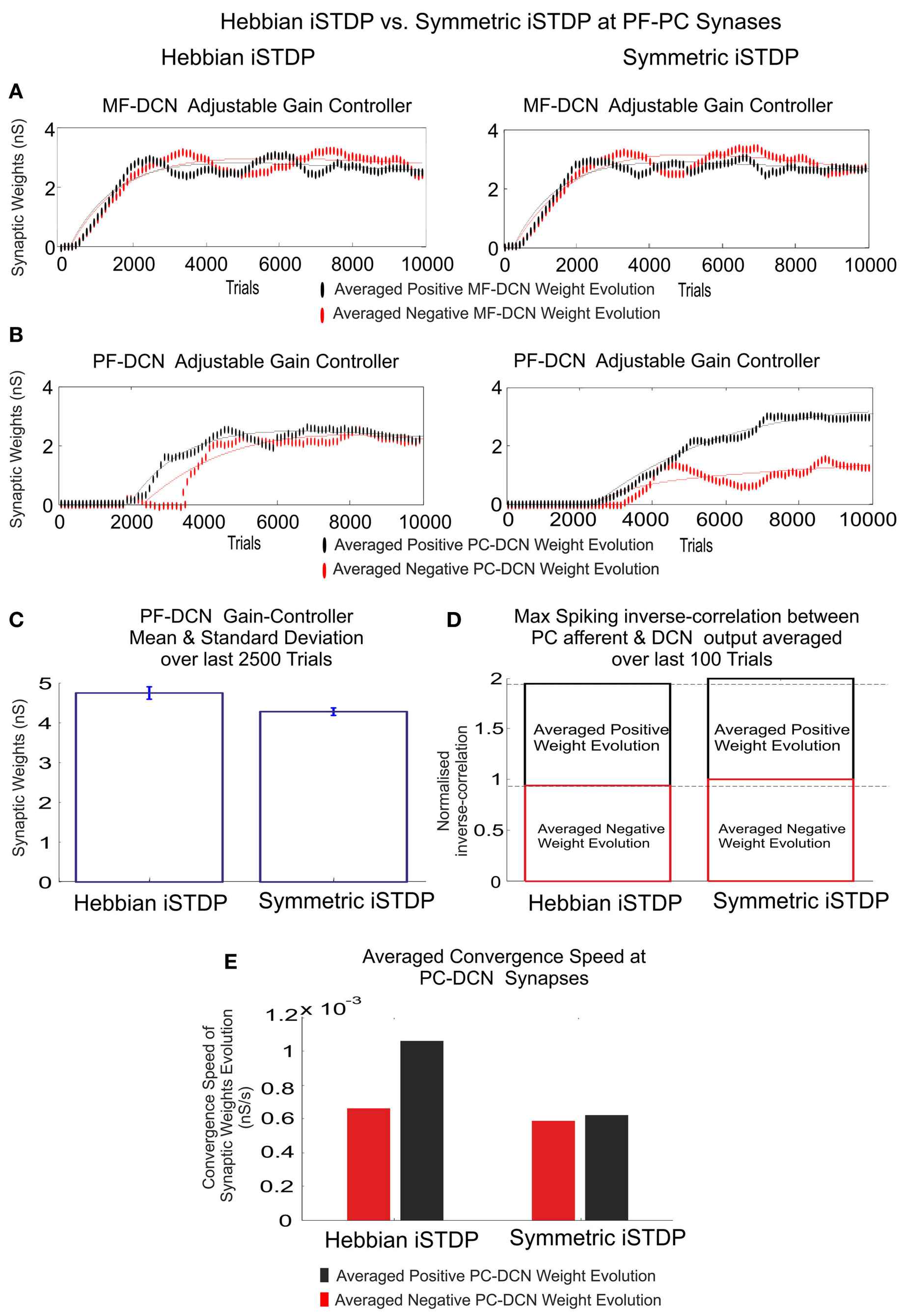


FIGURE 7 | Inhibitory plasticity at PC–DCN synapses. Kernel shape impact. Classic i-STDP Hebbian rule vs. Symmetric i-STDP. Case study A. **(A,B)** Whilst MF–DCN averaged synaptic weights (averaged gains) exhibit a similar behavior in both configurations (Classic i-STDP Hebbian and Symmetric i-STDP), PC–DCN averaged synaptic weights (averaged gains) differ. **(C)** The second kernel presents a better performance in terms of gain stability and global gain (a lower global gain value obtains the same correction action). **(D)** The symmetric i-STDP kernel achieves a better balance for the DCN excitation and inhibition inputs. Symmetric i-STDP better propagates selectively the correlated spiking activity from PC to DCN. Symmetric i-STDP always leads to higher maximal-spiking reverse-correlation values between PC afferent and DCN output for the two DCN cells. **(E)** Averaged Convergence Speed for Classic i-STDP Hebbian kernel is higher than Symmetric i-STDP. Hebbian kernel converges faster at the cost of a lower gain stability and global gain.

learning mechanisms at DCN synapses remains valid in all of them:

i. Plasticity at DCN synapses is double-folded:

- It is able to operate as a gain adaptation mechanism allowing the PFs to prevent saturation, thus making the learning mechanisms between PFs and PCs more accurate

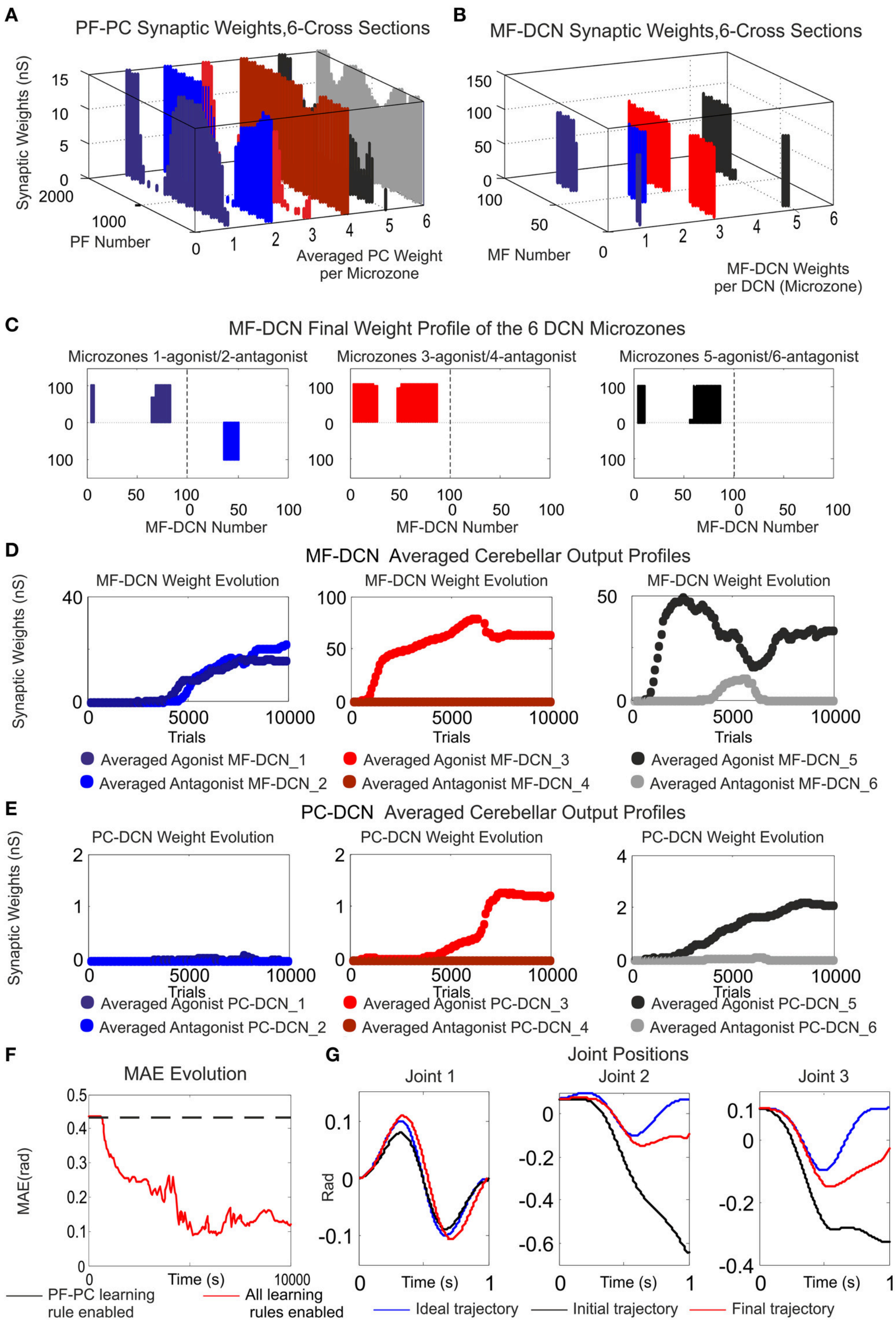


FIGURE 8 | Functionality of PF–PC, MF–DCN, and PC–DCN learning rules working conjointly in a manipulating robotic task. Case B. **(A)** Cross-sections of the synaptic weight distribution surface at PF–PC connections at the end of the learning process. There are two cross-sections per articulated robotic joint; purple and blue cross-sections correspond to the first joint, red and orange to the second joint, and black and gray to the third one. The weight distribution at PF–PC shows that the corrective action is properly stored at these afferents. Each of the six micro-complexes is formed by 10 PCs and innervated, in turn; by 2000 PFs. **(B,C)** Each pair of micro-complexes is in charge for the correct balance between the negative and the positive cerebellar correction per each operated robot joint. One micro-complex of each pair delivers the positive corrective action per joint whilst the other one delivers the negative corrective action per joint. The weight distribution at PF–PC is transferred in counter phase to MF–DCN. Despite the reduced MF number of non-recurrent states, MF–DCN synapses are able to consolidate the learning. The obtained synaptic weight-distribution shape adopts a discretized version of the PF–PC weight-distribution shape. Since the error to be corrected at the

(Continued)

FIGURE 8 | Continued
second and third joint is unidirectional (always negative), only the antagonist correction action is delivered. **(D,E)** MF–DCN and PC–DCN averaged synaptic weights (averaged gains) are stabilized more slowly than those at PF–PC synapses. MF–DCN and PC–DCN synapses depended on PC activity and are modified when some PF–PC weights tend to be saturated. Since PCs corresponding to the first joint are operating in their proper range, there is almost no need for gain regulation (not the case of consolidation) **(F)** Medium Average Error (MAE) evolution. The error curve to be corrected is the difference between controlled (robot actual position and velocities) and reference variables (desired position and velocities **Figure 3B**). Controlled and reference variables are not directly related (or in the same; representation space) to cerebellar corrective actions since corrective actions are delivered in torque values whilst controlled and reference variables are taken in joint values. Despite this demanding scenario, the cerebellum is able to supply corrective torque values which decrease the error up to 77, 8% in a 6 kg manipulation task (almost max robot load) when all the learning rules are active. **(G)** Robotic joint angle corrections obtained at the end of the learning process.

(keeping their plasticity capability within their working range).

- DCN has also proven to be fundamental for the slow memory consolidation process. A plausible two-state learning mechanism (Shadmehr and Brashers-Krug, 1997; Shadmehr and Holcomb, 1997) based on STDP has been shown. According to several evidences (Shadmehr and Brashers-Krug, 1997; Shadmehr and Holcomb, 1997; Medina and Mauk, 2000; Ohyama et al., 2006), the cerebellar cortex seems to undergo a fast learning process at initial learning stages while the consolidation process seems to occur in deeper structures (more likely at DCN innervations).

ii Inhibitory-STDP learning kernel based on near-coincident pre and post-synaptic spike has proven to be rather efficient for balancing the DCN excitation and inhibition inputs and for selectively propagating the correlated spiking activity from PCs to DCN. Nevertheless, it has been shown that the shape of the learning kernel at this site (as concluded also in Vogels et al., 2011) remains valid upon two related conditions:

- Postsynaptic activity shall potentiate those activated inhibitory synapses.
- In absence of postsynaptic firing, the inhibitory synapses shall decay.

Biological Realism and Model Limitations

Some simplifications and assumptions have been made to generate a mathematically tractable cerebellar model that is biologically realistic as well. The limitations imposed were profusely discussed in Garrido et al. (2013a) and Luque et al. (2014b); however, in light of new spiking features held by our approach, those limitations are here revisited:

(a) The main assumption at granular layer level is its functionality as a state generator. The state generator model is grounded in neurophysiologic observations of granule cell connectivity. Granule cells are comprised in a recurrent inhibitory network with Golgi cells, thus pointing to the fact that the input layer of the cerebellum may act as a recurrent circuit. The state-generator model has revealed that modeled granule cells present a randomly repetitive behavior in active/inactive state transitions (Yamazaki and Tanaka, 2009). Furthermore, this model has also shown that the sparse population of active cells changes with the passage of the time (POT) and no recurrence of active cell populations is exhibited. Consequently, a specific time interval can be univocally represented by means of a sequence of active cells belonging to a certain population. In other words, the state-generator model is able to represent the POT by means of a sparse-population coding scheme, thus allowing the cerebellum to operate like a liquid state machine (LSM; Maass et al., 2002; Yamazaki and Tanaka, 2007a) or an Echo state network (Jaeger, 2007). The cerebellar granule cell layer can be seen as an LSM; each LSM neuron receives time varying inputs from external sources (as the cerebellum receives varying sensorimotor inputs through mossy fibers) and from other neurons as well (this role is played in the cerebellum by different interneurons such as Golgi cells, Lugaro cells, unipolar brush cells, etc.). These LSM neurons are randomly connected to each other (as Granule cells are interconnected via Golgi cells in a recurrent loop). This structural analogy leads us to think that the recurrent nature of both neural networks, cerebellar granule layer and LSM, may operate in a similar manner. That is, the time varying inputs are turned into spatio-temporal patterns of neural activations; the granular layer acts as the reservoir of interacting spiking neurons within a recurrent topology, whilst Purkinje cells act as readout neurons. The strength of the cerebellum acting like an LSM lies in the possibility of obtaining whichever needed mathematical operation so as to perform a certain task such as eyelid conditioning or motor control tasks.

Since the exact function of the granular layer is not fully resolved, an assessment of its involvements remains to be established besides a biologically precise representation of plasticity mechanisms underneath (i.e., Solinas et al., 2010) that could substantially modify the core conclusion of this model.

(b) MF input layer was assumed to maintain not only a constant firing rate, but also time-evolving states simultaneously (25 different states with four non-overlapped MFs activated per state). Making use of time-evolving states at MF layer level has, within this article, proven to be vital for the learning consolidation process. Despite this, it was assumed that the granular layer circuit was also capable of generating time-evolving states even in the presence of a constant MF input thanks to its inner dynamics (Fujita, 1982; Yamazaki and Tanaka, 2007a). DCN activity has, indeed, been traditionally related with both the excitatory-activity integration coming from MFs and the inhibitory-activity integration from PCs. The number of MFs and CFs in comparison to granule cells (GCs) is very low. Thus, these fibers (MFs and CFs) are very limited for generating a sparse representation of different cerebellar states. In fact, even though MFs in our model were able to generate 25 different states, their role could be understood more as a baseline global activity or bias term provider per generated state rather than a proper state generator that is more the role of the granular-cell-layer. This fact pointed out that the reported synaptic plasticity at MF–DCN synapses (Racine et al., 1986; Medina and Mauk, 1999; Pugh and Raman, 2006; Zhang and Linden, 2006) could induce the adjustment of gain control through plasticity at DCN synapses whilst the learning consolidation was roughly preserved at these MF–DCN synapses.

(c) Cerebellar feedback is needed to minimize the existing difference between the controlled variable (actual cerebellar output value) and the reference variable (set point) via manipulation of the controlled variable. We assumed the teaching signal to come only through the CFs; however, there is no general agreement regarding neither the type of information conveyed by CFs nor their potential role (Ito, 2013; Luque et al., 2014b). Furthermore, there exist evidences pointing to the fact that cerebellar feedback is bounced back toward the motor cortex (Kawato et al., 1987; Siciliano and Khatib, 2008) together with the teaching signal, which is also received and correlated at a granular layer level (Krichmar et al., 1997; Kistler and Leo van Hemmen, 1999; Anastasio, 2001; Rothganger and Anastasio, 2009). Incorporating these elements is thought to further enhance the level of flexibility and accuracy in motor control and learning.

(d) We have included within the model what is, to our knowledge, the most complex set of STDP plasticity mechanisms interacting with each other within the cerebellar network. Nevertheless, there are multiple sub-forms of plasticity which are still missing such as plasticity at MF–GC, GO–GC, MF–GO connections, etc., as well as PC and GC intrinsic excitability (Hansel et al., 2001; Gao et al., 2012; Garrido et al., 2013b).

(e) The theoretical network here presented is rather oversimplified compared to the real cerebellar network. The physiological implications may have been overlooked but must not be ignored. As an example, the role of the inhibitory PC collaterals, the complex structure of the PC dendritic tree, the operation of DCN cells with their characteristic postsynaptic rebounds, or the theta oscillations and resonance in the granular layer, amongst many other physiological evidences, shall need to be fully addressed. Nevertheless, the way in which all these physiology implications interact, how they reciprocally improve their operations, and how they are understandable in the framework of a complex cerebellar operation remains a future challenge.

(f) MF–DCN and PC–DCN STDP plasticity mechanisms were implemented according to some principles suggested by Medina and Mauk (2000), Masuda and Amari (2008), and Vogels et al. (2011), where DCN played the role of a further cerebellar learning vessel besides PF–PC synapses. However, the underlying mechanism that the cerebellar nuclei may experience in cerebellar learning has only been suggested at experimental single-cell level and supported by behavioral observations (EBCC and VOR). MF–DCN and PC–DCN STDP plasticity mechanisms therefore still have to be specifically demonstrated and characterized.

CONCLUSION

Our results propose an explanation for the existing interplay between the excitatory and inhibitory synapses at DCN afferents by means of STDP mechanisms. This balance allows the PC outcome to shape the output of its corresponding DCN-target neuron which may effectively implement a cerebellar gain control fully compatible with the two-state learning mechanism suggested by Shadmehr and Brashers-Krug (1997), Shadmehr and Holcomb (1997), and Shadmehr and Mussa-Ivaldi (2012). Moreover, those STDP assemblies at MF–DCN and PC–DCN synapses have proven to be effective to explain how long-term memories can be transferred and stored from PF–PC to MF–DCN synapses. In fact, the experimentation revealed how MF–DCN synapses could effectively copy a discretized version of the PF–PC weight distribution shape in counter-phase. This learning consolidation process operated much as was demonstrated in Vogels et al. (2011); that is, PC, MF, or DCN cells do not compete with each other, exhibiting a winner-take-all behavior. On the contrary, the cerebellar PC–DCN, MF–DCN innervations stay inactive until PC activity starts modulating MF–DCN connections (thus favoring excitation), whilst DCN activity is able to self-modulate PC–DCN innervations (thus favoring inhibition). STDP learning rule at inhibitory synapses facilitates a self-organized balance of excitation and inhibition at DCN innervations.

Our results also suggest that the understanding of STDP mechanisms in motor learning requires not only studying their molecular basis. Rather, they show that this understanding must be accompanied by parallel insights regarding how the interactions amongst these plasticity mechanisms and the different cerebellar sub-circuitries allow distributed learning and neural homeostatic balance.

AUTHOR CONTRIBUTIONS

All authors listed, have made substantial, direct and intellectual contribution to the work, and approved it for publication. NL and JG conceived, designed the experiments. NL performed the experiments. NL and ER analyzed the data. NL, FN, and RC contributed reagents/materials/analysis tools. NL, ER, and ED wrote the paper.

ACKNOWLEDGMENTS

This work was supported by the European Union NR (658479 – SpikeControl) and by the Spanish National Grant ER (TIN2013-47069-P partially funded by FEDER). ED was supported by grants of the European Union (CEREBNET FP7-ITN238686, REALNET FP7-ICT270434, Human Brain Project HBP-604102) and the Italian Ministry of Health (RF-2009-1475845).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://journal.frontiersin.org/article/10.3389/fncom.2016.00017

Movie S1 | Learning simulation. Synaptic weight evolution during Case A study is plotted. Simulations were run using plasticity mechanisms at PF–PC, MF-DCN, and PC-DCN along 6000 trials. PF–PC synaptic conductances were set to an initial value of 5 nS; MF-DCN and PC-DCN initial conditions started from zero. Only one every 100 trials is shown. (Top left) 3D view of the synaptic weight distribution at PF–PC synapses. (Top right) Sagittal axis of the synaptic weight distribution at MF-DCN synapses. (Second right) The cerebellum counterbalances the existing difference between the actual cerebellar output (in red) and the reference curve (in blue) which is iteratively presented to the cerebellum over 5000 iterations. (First and second plot of the bottom row). Evolution of the averaged gain at MF-DCN and PC-DCN synaptic weights at first and second micro-complexes which supply agonist (red line) or antagonist (black line) cerebellar corrective actions.

Movie S2 | Learning simulation. Synaptic weight evolution during Case A study is plotted. Simulations were run using plasticity mechanisms at PF–PC, MF-DCN, and PC-DCN along 10,000 trials. PF–PC synaptic conductances were set to an initial value of 5 nS, MF-DCN initial conditions started from zero whilst PC-DCN initial conditions were set to a higher value than demanded. Only one every 100 trials is shown. (Top left) 3D view of the synaptic weight distribution at PF–PC synapses. (Top right) Sagittal axis of the synaptic weight distribution at MF-DCN synapses. (Second right) The cerebellum counterbalances the existing difference between the actual cerebellar output (in red) and the reference curve (in blue) which is iteratively presented to the cerebellum over 10,000 iterations. (First and second plot of the bottom row). Evolution of the averaged gain at MF-DCN and PC-DCN synaptic weights at the first and second micro-complexes which supply an agonist (red line) or antagonist (black line) cerebellar corrective action.

Movie S3 | Learning simulation. Synaptic weight evolution during Case B study is plotted. Simulations were run using plasticity mechanisms at PF–PC, MF-DCN, and PC-DCN along 10,000 trials. PF–PC synaptic conductances were set to an initial value of 5 nS, MF-DCN and PC-DCN initial conditions started from zero. Only one every 100 trials is shown. (Top left) Sagittal axis 3D view of the synaptic weight distribution at PF–PC synapses. (Top right) Sagittal axis 3D view of the synaptic weight distribution at MF-DCN synapses. (Second row plots) Evolution of the averaged gains at MF-DCN synaptic weights from the first to sixth micro-complex. Each micro-complex supplies an agonist (red line) or antagonist (black line) cerebellar corrective action in each robot joint. The error curve to be corrected is the difference between controlled (robot actual position and velocities) and reference variables (desired position and velocities). (Third row plots) Evolution of the averaged gain at PC-DCN synaptic weights from the first to sixth micro-complexes. Each micro-complex supplies an agonist (red line) or antagonist (black line) cerebellar corrective action at its corresponding robot joint.

Figure S1 | (A) PF–PC synaptic weight distribution at the beginning of the learning process at 1, 15, and 25 s (CASE A). The exponential weight distribution at PF–PC shows that the corrective action is properly stored at these afferents. Synaptic weights at PF–PC synapses are randomly initialized unlike in previous experimentations, where these weights were set to equal values in order to better perceive at a glance the shape of the synaptic weight distribution at this site. **(B)** MF-DCN averaged gains for 2, 3, 6, 8, and 10 kg, respectively, when the learning process has settled down (CASE B). MF-DCN synapses depended on PC activity and are modified when some PF–PC weights tend to be saturated. The heavier the payload to be manipulated by the lightweight robot, the more cerebellar gain is demanded for counterbalancing the dynamic existing mismatch between the crude inverse controller and the robot plant. Since the error is unidirectional in joints 2 and 3, the gain is unidirectional as well. In joint 1, the error to be compensated is bidirectional, and therefore, the gain has to be bidirectional.

REFERENCES

Aizenman, C., Manis, P., and Linden, D. (1998). Polarity of long-term synaptic gain change is related to postsynaptic spike *Neuron* 21, 827–835.

Albu-Schäffer, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. (2007). The DLR lightweight robot: design and control concepts for robots in human environments. *Int. J. Ind. Robot* 34, 376–385. doi: 10.1108/01439910710774386

Albus, J. S. (1971). A theory of cerebellar function. *Math Biosci.* 10, 25–61. doi: 10.1016/0025-5564(71)90051-4

Anastasio, T. J. (2001). Input minimization: a model of cerebellar learning without climbing fiber error signals. *Neuroreport* 12, 3825. doi: 10.1097/00001756-200112040-00045

Bagnall, M. W., and du Lac, S. (2006). A new locus for synaptic plasticity in cerebellar circuits. *Neuron* 51, 5–7. doi: 10.1016/j.neuron.2006.06.014

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* 13, 934–846.

Bastian, A. J. (2006). Learning to predict the future: the cerebellum adapts feedforward movement control. *Curr. Opin. Neurobiol.* 16, 645–649. doi: 10.1016/j.conb.2006.08.016

Bazzigaluppi, P., De Grujil, J. R., van der Giessen, R. S., Khosrovani, S., De Zeeuw, C. I., and De Jeu, M. T. G. (2012). Olivary subthreshold oscillations and burst activity revisited. *Front. Neural Circuits* 6:91. doi: 10.3389/fncir.2012.00091

Boucheny, C., Carrillo, R. R., Ros, E., and Coenen, O. J.-M. D. (2005). Real-time spiking neural network: an adaptive cerebellar model. *LNCS* 3512, 136–144. doi: 10.1007/11494669_18

Boyden, E. S., Katoh, A., and Raymond, J. L. (2004). Cerebellum-dependent learning: the role of multiple plasticity mechanisms *Annu. Rev. Neurosci.* 27, 581–609. doi: 10.1146/annurev.neuro.27.070203.144238

Brunel, N., Hakim, V., Isope, P., Nadal, J. P., and Barbour, B. (2004). Optimal information storage and the distribution of synaptic weights: perceptron versus Purkinje cell. *Neuron* 43, 745–757. doi: 10.1016/j.neuron.2004.08.023

Carey, M. R. (2011). Synaptic mechanisms of sensorimotor learning in the cerebellum. *Curr. Opin. Neurobiol.* 21, 609–615. doi: 10.1016/j.conb.2011.06.011

Carrillo, R. R., Ros, E., Boucheny, C., and Coenen, O.-J. M.-D. (2008). A real time spiking cerebellum model for learning robto control. *Biosystems* 94, 18–27. doi: 10.1016/j.biosystems.2008.05.008

Coesmans, M., Weber, J., De Zeeuw, C., and Hansel, C. (2004). Bidirectional parallel plasticity in the cerebellum under climbing. *Neuron* 44, 691–700. doi: 10.1016/j.neuron.2004.10.031

D'Angelo, E., De Filippi, G., Rossi, P., and Taglietti, V. (1998). Ionic mechanism of electroresponsiveness in cerebellar granule cells implicates the action of a persistent sodium current. *J. Neurophysiol.* 80, 493–503.

D'Angelo, E., Nieus, T., Maffei, A., Armano, S., and Rossi, P. (2001). Theta-frequency bursting and resonance in cerebellar granule cells: experimental evidence and modeling of a slow K+-dependent mechanism. *J. Neurosci.* 21, 759–770.

D'Angelo, E., Rossi, P., and Taglietti, V. (1993). Different proportions of N-Methyl-D-Aspartate and Non-N-Methyl-D-Aspartate receptor currents at the mossy fiber granule cell synapse of developing rat *cerebellum*. 53, 121–130.

De Gruijl, J. R., P, B., and De Jeu, M. T. G., De Zeeuw, C. I. (2012). Climbing fiber burst size and olivary sub-threshold oscillations in a network setting. *PLoS Comput. Biol.* 8:e1002814. doi: 10.1371/journal.pcbi.1002814

Eccles, J. C. (1967). Circuits in the cerebellar control of movement. *Proc. Natl. Acad. Sci. U.S.A.* 58, 336–343. doi: 10.1073/pnas.58.1.336

Eccles, J. C., Ito, M., and Szentágothai, J. (1967). *The Cerebellum as a Neuronal Machine.* New York, NY: Springer-Verlag.

Fujita, M. (1982). Adaptive filter model of the cerebellum. *Biol. Cybern.* 45, 195–206. doi: 10.1007/BF00336192

Gao, Z., Vanbeugen, B. J., and De Zeeuw, C. I. (2012). Distributed synergistic plasticity and cerebellar learning. *Nat. Rev. Neurosci.* 13, 1–17. doi: 10.1038/nrn3391

Garrido, J. A., Luque, N. R., D'Angelo, E., and Ros, E. (2013a). Distributed cerebellar plasticity implements adaptable gain control in a manipulation task: a closed-loop robotic simulation. *Front. Neural Circuits* 7:159. doi: 10.3389/fncir.2013.00159

Garrido, J. A., Ros, E., and D'Angelo, E. (2013b). Spike timing regulation on the millisecond scale by distributed synaptic plasticity at the cerebellum input stage: a simulation study. *Front. Comput. Neurosci.* 7:64. doi: 10.3389/fncom.2013.00064

Gerstner, W., and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, UK: Cambridge University Press.

Hansel, C., Linden, D. J., and D'Angelo, E. (2001). Beyond parallel fiber LTD: the diversity of synaptic and non-synaptic plasticity in the cerebellum. *Nat. Neurosci.* 4, 467–475. doi: 10.1038/87419

Hirzinger, G., Butterfab, J., Fischer, M., Grebenstein, M., Hähnle, M., Liu, H., et al. (2000). "A mechatronics approach to the design of light-weight arms and multifingered hands," in *ICRA* (San Francisco, CA), 46–54.

Honda, T., Yamazaki, T., Tanaka, S., Nagao, S., and Nishino, T. (2011). Stimulus-dependent state transition between synchronized oscillation and randomly repetitive burst in a model cerebellar granular layer. *PLoS Comput. Biol.* 7:e1002087. doi: 10.1371/journal.pcbi.1002087

Houk, J. C., Buckingham, J. T., and Barto, A. G. (1996). Models of cerebellum and motor learning. *Behav. Brain Sci.* 19, 369–383. doi: 10.1017/s0140525x00081474

Hwang, E. J., and Shadmehr, R. (2005). Internal Models of limb dynamic and the encoding of limb state. *J. Neural Eng.* 2, 266–278. doi: 10.1088/1741-2560/2/3/S09

Ito, M. (1984). *The Cerebellum and Neural Control*. New York, NY: Raven Press.

Ito, M. (2013). Error detection and representation in the olivo-cerebellar system. *Front. Neural Circuits* 7:1. doi: 10.3389/fncir.2013.00001

Ito, M., and Kano, M. (1982). Long-lasting depression of parallel fiber-Purkinje cell transmission induced by conjunctive stimulation of parallel fibers and climbing fibers in the cerebellar cortex. *Neurosci. Lett.* 33, 253–258. doi: 10.1016/0304-3940(82)90380-9

Izawa, J., Pekny, S. E., Marko, M. K., Haswell, C. C., Shadmehr, R., and Mostofsky, S. H. (2012). Motor learning relies on integrated sensory inputs in ADHD, but over-selectively on proprioception in autism spectrum conditions. *Autism Res.* 5, 124–136. doi: 10.1002/aur.1222

Jaeger, H. (2007). Echo state network. *Scholarpedia* 2:2330. doi: 10.4249/scholarpedia.2330

Kawato, M., Furukawa, K., and Suzuki, R. (1987). A hierarchical neural-network model for control and learning of voluntary movement. *Biol. Cybern.* 57, 169–185. doi: 10.1007/BF00364149

Kettner, R. E., Mahamud, S., Leung, H., Sittko, N., Houk, J. C., Peterson, B. W., et al. (1997). Prediction of complex two-dimensional trajectories by a cerebellar model of smooth pursuit eye movement. *J. Neurophysiol.* 77, 2115–2130.

Kistler, W. M., and Leo van Hemmen, J. (1999). Delayed reverberation through time windows as a key to cerebellar function. *Biol. Cybern.* 81, 373–380. doi: 10.1007/s004220050569

Kleberg, F. I., Fukai, T., and Gilson, M. (2014). Excitatory and inhibitory STDP jointly tune feedforward neural circuits to selectively propagate correlated spiking activity. *Front. Comput. Neurosci.* 8:53. doi: 10.3389/fncom.2014.00053

Krichmar, J., Ascoli, G., Hunter, L., and Olds, J. (1997). "A model of cerebellar saccadic motor learning using qualitative reasoning," in *Biological and Artificial Computation: From Neuroscience to Technology* (Lanzarote: Springer), 133–145. doi: 10.1007/bfb0032471

Kuroda, S., Yammoto, K., Miyamoto, H., Doya, K., and Kawato, M. (2001). Statistical characteristics of climbing fiber spikies necessary for efficient cerebellar learning. *Biol. Cybern.* 84, 183–192. doi: 10.1007/s004220000206

Lev-Ram, V., Mehta, S. B., Kleinfeld, D., and Tsien, R. Y. (2003). Reversing cerebellar long term depression. *Proc. Natl. Acad. Sci. U.S.A.* 100, 15989–15993. doi: 10.1073/pnas.2636935100

Luque, N. R., Carrillo, R. R., Naveros, F., Garrido, J. A., and Sáez-Lara, M. J. (2014a). Integrated neural and robotic simulations. Simulation of cerebellar neurobiological substrate for an object-oriented dynamic model abstraction process. *Rob. Auton. Syst.* 62, 1702–1716. doi: 10.1016/j.robot.2014.08.002

Luque, N. R., Garrido, J. A., Carrillo, R. R., Coenen, O. J. and Ros, E. (2011a). Cerebellarlike corrective model inference engine for manipulation tasks. *IEEE Trans. Syst. Man Cybern. B Cybern.* 41, 1299–1312. doi: 10.1109/TSMCB.2011.2138693

Luque, N. R., Garrido, J. A., Carrillo, R. R., D'Angelo, E., and Ros, E. (2014b). Fast convergence of learning requires plasticity between inferior olive and deep cerebellar nuclei in a manipulation task: a closed-loop robotic simulation. *Front. Comput. Neurosci.* 8:97. doi: 10.3389/fncom.2014.00097

Luque, N. R., Garrido, J. A., Carrillo, R. R., Tolu, S., and Ros, E. (2011b). Adaptive cerebellar spiking model embedded in the control loop: context switching and robustness against noise. *Int. J. Neural Syst.* 21, 385–401. doi: 10.1142/S0129065711002900

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations *Neural Comput.* 14, 2531–2560. doi: 10.1162/089976602760407955

Marr, D. (1969). A theory of cerebellar cortex. *J. Physiol.* 202, 437–470. doi: 10.1113/jphysiol.1969.sp008820

Masquelier, T., Guyonneau, R., and Thorpe, S. J. (2009). Competitive STDP-based spike pattern learning. *Neural Comput.* 21, 1259–1276. doi: 10.1162/neco.2008.06-08-804

Masuda, N., and Amari, S. (2008). A computational study of synaptic mechanisms of partial memory transfer in cerebellar vestibulo-ocular-reflex learning. *J. Comput. Neurosci.* 24, 137–156. doi: 10.1007/s10827-007-0045-7

Medina, J. F., and Mauk, M. D. (2000). Computer simulation of cerebellar information processing. *Nat. Neurosci.* 3(Suppl), 1205–1211. doi: 10.1038/81486

Medina, J., and Mauk, M. (1999). Simulations of cerebellar motor learning: computational analysis of plasticity at the mossy fiber synapse. *J. Neurosci.* 19, 7140–7151.

Miles, F., and Lisberger, S. (1981). Plasticity in the vestibulo-ocular reflex: a new hypothesis. *Annu. Rev. Neurosci.* 4, 273–299. doi: 10.1146/annurev.ne.04.030181.001421

Morishita, W., and Sastry, B. (1996). Postsynaptic mechanisms underlying long-term depression of gabaergic transmission in neurons of the deep cerebellar nuclei. *J. Neurophysiol.* 76, 59–68.

Nakano, E., Imamizu, H., Osu, R., Uno, Y., Gomi, H., Yoshioka, T., et al. (1999). Quantitative examinations of internal representations for arm trajectory planning. minimum commanded torque change model. *J. Neurophysiol.* 81, 2140–2155.

Naveros, F., Luque, N. R., Garrido, J. A., Carrillo, R. R., Anguita, M., and Ros, E. (2015). A spiking neural simulator integrating event-driven and time-driven computation schemes using parallel CPU-GPU co-processing: a case study. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 1567–1574. doi: 10.1109/TNNLS.2014.2345844

Nieus, T., Sola, E., Mapelli, J., Saftenku, E., and Rossi, P. (2006). LTP regulates burst initiation and frequency at mossy fiber-granule cell synapses of rat cerebellum: Experimental observations and theoretical predictions. *J. Neurophysiol.* 95, 686–699. doi: 10.1152/jn.00696.2005

Nusser, Z., Cullcandy, S., and Farrant, M. (1997). Differences in synaptic GABA(A) receptor number underlie variation in GABA mini amplitude. *Neuron* 19, 697–709. doi: 10.1016/S0896-6273(00)80382-7

Ohyama, T., Nores, W. L., Medina, J. F., Riusech, F. A., and Mauk, M. D. (2006). Learning-induced plasticity in deep cerebellar nucleus. *J. Neurosci.* 26, 12656–12663. doi: 10.1523/JNEUROSCI.4023-06.2006

Ouardouz, M., and Sastry, B. (2000). Mechanisms underlying ltp of inhibitory synaptic transmission in the deep cerebellar nuclei. *J. Neurophysiol.* 84, 1414–1421.

Passot, J.-B., Luque, N. R., and Arleo, A. (2013). Coupling internal cerebellar models enhances online adaptation and supports offline consolidation in sensorimotor tasks. *Front. Comput. Neurosci.* 7:95. doi: 10.3389/fncom.2013.00095

Pugh, J., and Raman, I. (2006). Potentiation of mossy NMDA receptor activation followed by postinhibitory rebound current. *Neuron* 51, 113–123. doi: 10.1016/j.neuron.2006.05.021

Racine, R., Wilson, D., Gingell, R., and Sunderland, D. (1986). Long-term potentiation in the interpositus and vestibular nuclei in the rat. *Exp. Brain Res.* 63, 158–162. doi: 10.1007/bf00235658

Ros, E., Carrillo, R. R., Ortigosa, E. M., Barbour, B., and Agís, R. (2006). Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics. *Neural Comput.* 18, 2959–2993. doi: 10.1162/neco.2006.18.12.2959

Rossi, D. J., and Hamann, M. (1998). Spillover-mediated transmission at inhibitory synapses promoted by high affinity alpha(6) subunit GABA(A) receptors and glomerular geometry. *Neuron* 20, 783–795. doi: 10.1016/S0896-6273(00)81016-8

Rothganger, F. H., and Anastasio, T. J. (2009). Using input minimization to train a cerebellar model to simulate regulation of smooth pursuit. *Biol. Cybern.* 101, 339–359. doi: 10.1007/s00422-009-0340-7

Schrauwen, B., and van Campenhout, J. (2003). "BSA, a fast and accurate spike train encoding scheme," in *Proceedings of the International Joint Conference on Neural Networks, IEEE* (Portland, OR), 2825–2830.

Shadmehr, R., and Brashers-Krug, T. (1997). Functional stages in the formation of human long-term motor memory. *J. Neurosci.* 17, 409–419.

Shadmehr, R., and Holcomb, H. H. (1997). Neural correlates of motor memory consolidation. *Science* 277, 821–825. doi: 10.1126/science.277.5327.821

Shadmehr, R., and Mussa-Ivaldi, S. (2012). *Biological Learning and Control: How the Brain Builds Representations, Predicts Events, and Makes Decisions*. Cambridge, MA: MIT Press.

Siciliano, B., and Khatib, O. (2008). *Springer Handbook of Robotics*. Berlin; Heidelberg; Würzburg: Springer-Verlag.

Silver, R. A., Colquhoun, D., Cullcandy, S. G., and Edmonds, B. (1996). Deactivation and desensitization of non-NMDA receptors in patches and the time course of EPSCs in rat cerebellar granule cells. *J. Physiol.* 493, 167–173. doi: 10.1113/jphysiol.1996.sp021372

Solinas, S., Nieus, T., and D'Angelo, E. (2010). A realistic large-scale model of the cerebellum granular layer predicts circuit spatio-temporal filtering properties. *Front. Cell. Neurosci.* 4:12. doi: 10.3389/fncel.2010.00012

Sutton, R. S., and Barto, A. G. (1981). Toward a modern teory of adaptive networks: expectation and prediction. *Phychol. Rev.* 88, 135–171. doi: 10.1037/0033-295X.88.2.135

Tia, S., Wang, J. F., Kotchabhakdi, N., and Vicini, S. (1996). Developmental changes of inhibitory synaptic currents in cerebellar granule neurons: role of GABA(A) receptor alpha 6 subunit. *J. Neurosci.* 16, 3630–3640.

Todorov, E. (2004). Optimality principles in sensorimotor control (review). *Nat. Neurosci.* 7, 907–915. doi: 10.1038/nn1309

van Der Smagt, P. (2000). Benchmarking cerebellar control. *Robot. Auton. Syst.* 32, 237–251. doi: 10.1016/S0921-8890(00)00090-7

van Rossum, M. C. (2001). A novel spike distance. *Neural Comput.* 13, 751–763. doi: 10.1162/089976601300014321

Victor, J. D. (2005). Spike train metrics. *Curr. Opin. Neurobiol.* 15, 585–592. doi: 10.1016/j.conb.2005.08.002

Vogels, T., Sprekeler, H., Zenke, F., Clopath, C., and Gerstner, W. (2011). Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science* 334, 1569–1573. doi: 10.1126/science.1211095

Voogd, J., and Glickstein, M. (1998). The anatomy of the cerebellum. *Trends Neurosci.* 21, 370–375. doi: 10.1016/S0166-2236(98)01318-6

Yamazaki, T., and Nagao, S. (2012). A computational mechanism for unified gain and timing control in the cerebellum. *PLoS ONE* 7:e33319. doi: 10.1371/journal.pone.0033319

Yamazaki, T., and Tanaka, S. (2005). Neural modeling of an internal clock. *Neural Comput.* 17, 1032–1058. doi: 10.1162/0899766053491850

Yamazaki, T., and Tanaka, S. (2007a). The cerebellum as a liquid state machine. *Neural Netw.* 20, 290–297. doi: 10.1016/j.neunet.2007.04.004

Yamazaki, T., and Tanaka, S. (2007b). A spiking network model for passage-of-time representation in the cerebellum. *Eur. J. Neurosci.* 26, 2279–2292. doi: 10.1111/j.1460-9568.2007.05837.x

Yamazaki, T., and Tanaka, S. (2009). Computational models of timing mechanisms in the cerebellar granular layer. *Cerebellum* 8, 423–432. doi: 10.1007/s12311-009-0115-7

Yang, Y., and Lisberger, S. G. (2014). Role of plasticity at different sites across the time course of cerebellar motor learning. *J. Neurosci.* 34, 7077–7090. doi: 10.1523/JNEUROSCI.0017-14.2014

Zhang, W., and Linden, W. D. (2006). Long-term depression at the mossy fiber - deep cerebellar nucleus synapse. *J. Neurosci.* 26, 6935–6944. doi: 10.1523/JNEUROSCI.0784-06.2006

Zheng, N., and Raman, I. (2010). Synaptic inhibition, excitation, and plasticity in neurons of the cerebellar nuclei. *Cerebellum* 9, 56–66. doi: 10.1007/s12311-009-0140-6

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

APPENDIX: NEURAL AND SYNAPSE MODELS

Neuron models were implemented using slightly modified versions of the LIF model (Gerstner and Kistler, 2002). In the LIF model, the neural state is characterized by the membrane potential (V_{m-c}) defined by the differential equation (Equation A1). This equation includes the effect of chemical synapses [α-amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid (AMPA), gamma-aminobutyric acid (GABA) receptors] and the resting conductance (G_{rest}),

$$C_m \cdot \frac{dV_{m-c}}{dt} = g_{AMPA}(t) \cdot (E_{AMPA} - V_{m-c}) + \\ + g_{GABA}(t) \cdot (E_{GABA} - V_{m-c}) + G_{rest} \cdot (E_{rest} - V_{m-c}) \quad \text{(A1)}$$

where C_{m} denotes the membrane capacitance, E_{AMPA} and E_{GABA} stand for the reversal potential of each synaptic conductance, and E_{rest} represents the resting potential (with G_{rest} being the conductance responsible for the passive decay term toward the resting potential). Conductances g_{AMPA} and g_{GABA} integrate all the contributions received by each receptor type (AMPA and GABA) through individual synapses and are defined as decaying exponential functions which provide reasonable accuracy at a low computational cost (Gerstner and Kistler, 2002; Ros et al., 2006; Equation A2).

$$g_{AMPA}(t) = \begin{cases} 0, & t \le t_0 \\ g_{AMPA}(t_0) \cdot e^{-(t-t_0)/\tau_{AMPA}}, & t > t_0 \end{cases}$$
$$g_{GABA}(t) = \begin{cases} 0, & t \le t_0 \\ g_{GABA}(t_0) \cdot e^{-(t-t_0)/\tau_{GABA}}, & t > t_0 \end{cases} \quad \text{(A2)}$$

where t represents the simulation time whilst t_0 denotes the arrival instant of an input spike. g_{AMPA} stands for the AMPA receptor which provides excitation, and g_{GABA} stands for the GABA receptor-mediated conductance, which provides inhibition. Finally, τ_{AMPA} and τ_{GABA} are the decaying time constants of each receptor type. The parameters defining each cell type and synaptic receptor that have been chosen to model granule cell, Purkinje cell, and deep nucleus dynamics are included in **Table A1**.

TABLE A1 | Parameters of the cell types.

Parameter	Granule cell	Purkinje cell	DCN cell
Refractory period (ms)	1	2	1
Membrane capacitance (pF)	2	400	2
*Total excitatory peak conductance	1 nS·100	1.3 nS·175,000·10%*	1 nS·7
Total inhibitory peak conductance	1 nS·200	3 nS·150	30 nS·1
Threshold (mV)	−40	−52	−40
Resting potential (mV)	−70	−70	−70
Resting conductance (nS)	0.2	16	0.2
Resting time constant (τ_{rest}; ms)	10	25	10
Excitatory-synapse time constant (τ_{AMPA}; ms)	0.5	0.5	0.5
Inhibitory-synapse time constant (τ_{GABA}; ms)	10	1.6	10

Parameters obtained from the following papers:
Granule cell (GC; Silver et al., 1996; Tia et al., 1996; Nusser et al., 1997; D'Angelo et al., 1998; Rossi and Hamann, 1998) and Purkinje cell (PC; D'Angelo et al., 1993, 1998, 2001; Nieus et al., 2006). DCN data were extracted from unpublished material from Prof. D'Angelo's lab.
**Where 10% means the ratio of active connections PF–PC (out of the total 175,000 PFs).*