INVERSE PROBLEM OF PREDICTING STOCHASTIC FATIGUE DAMAGE AND RELIABILITY IN COMPOSITES MATERIALS

by

JUAN CHIACHIO RUANO

A thesis submitted to the Department of Structural Mechanics and Hydraulic Engineering,

in partial fulfillment of the requirements for the degree of

DIPLOMA DE ESTUDIOS AVANZADOS

Supervisor: Dr. Guillermo Rus Carlborg

Department of Structural Mechanics and Hydraulic Engineering

University of Granada, Campus de Fuentenueva,

18071 Granada, Spain

June 2011

ABSTRACT

INVERSE PROBLEM OF PREDICTING STOCHASTIC FATIGUE DAMAGE AND RELIABILITY IN COMPOSITES MATERIALS

The prediction of the fatigue behavior of composites materials is an unsolved problem with important economical and safety implications. The majority of the fatigue models existing in the literature work under restricted experimental conditions and hence they are difficult to extend. Additionally a vast number of them are of the deterministic type, thus they can not account the inherent variability of the fatigue process. In this work, a stochastic phenomenological evolutive damage model is presented as an extension of the classic model of Bogdanoff and Kozin, based on Markov chains. New model parameterizations are proposed and the Inverse Problem for parameter identification is solved from stochastic damage data by means of a genetic algorithm. The parameter identification is done by accounting all the statistical information contained within the data, defining a new residual based on statistical distance. Additionally, a new residual based on the concept of cumulative entropy has been defined, which considers the information gained when predictions approach data. Finally the statistical prediction of the complete damage process is introduced into the reliability formulation, leading to a coherent prediction of the long term reliability.

RESUMEN

PROBLEMA INVERSO DE PREDICIÓN DE DAÑO ESTOCÁSTICO POR FATIGA Y FIABILIDAD EN MATERIALES COMPUESTOS

La predicción del comportamiento a fatiga de los materiales compuestos es un problema abierto con importantes implicaciones económicas y de seguridad. La mayoría de los modelos de fatiga existentes en la literatura funcionan bajo determinadas condiciones experimentales por lo que son difícilmente extensibles. Adicionalmente, una buena parte de estos modelos son de tipo determinista, por lo que no pueden tener en cuenta la variabilidad inherente al proceso de fatiga. En este trabajo se plantea un modelo estocástico fenomenológico de evolución de daño, como extensión del modelo estocástico clásico de Bogdanoff y Kozin, basado en cadenas de Markov. Se han propuesto diferentes parametrizaciones del modelo y se ha resuelto el Problema Inverso de identificación de parámetros a partir de datos estocásticos mediante algoritmos genéticos. La identificación de parámetros se ha realizado teniendo en cuenta toda la información estadística contenida en los datos, mediante la definición original de un residual basado en distancia estadística. Adicionalmente, se ha planteado un residual basado en el concepto de entropía acumulada, que tiene en cuenta el contenido de información ganado a medida que las predicciones se aproximan a los datos. Finalmente la predicción estadística del daño es introducida en el criterio de fallo del material compuesto, dando lugar a una predicción coherente de la fiabilidad a largo plazo.

ACKNOWLEDGMENTS

I would like to thank the responsible for the direction of my research, Dr. Guillermo Rus Carlborg of the Department of Structural Mechanics. He brought me back to University to work in the exciting area of composites materials. His philosophical thinking has had a great influence on my work throughout this time. I couldn't forget to my friends and colleges of the Non Destructive Evaluation Laboratory. There have been a lot of enjoyable moments in the collaborations and I have learned much from them, especially in the areas outside my research topic. Also I would like to thank my colleges of the Department of Structural Mechanics for their friendly help and advise.

And finally, I need to express my sincere gratitude to my family. I'm in debt with them for the comprehension I receive in my Phd work.

This work has been supported by the Ministry of Education of Spain through FPU grant no. P2009-4641.

AUTORIZACIÓN

D. .....

Profesor del departamento de

.....

de la Universidad de Granada, como director del Proyecto Fin de Máster de D.

.....

Informa:

que el presente trabajo, titulado:

.....

Ha sido realizado y redactado por el mencionado alumno bajo nuetra dirección, y con esta fecha autorizo a su presentación.

Granada, a ..... de ..... de 20.....

Fdo. .....

Los abajo firmantes autorizan a que la presente copia de Proyecto Fin de Máster se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a ..... de ..... de 20.....

(Firmas y números de DNI / NIE del alumno y de los tutores)

Informe de valoración del proyecto

El tribunal constituido para la evaluación del Proyecto Fin de Máster titulado:

.....

Realizado por el alumno:

.....

Y dirigido por el tutor:

.....

Ha resuelto la calificación de:

- ☐ SOBRESALIENTE (9-10 puntos)
- ☐ NOTABLE (7-8.9 puntos)
- ☐ APROBADO (5-6.9 puntos)
- ☐ SUSPENSO

Con la nota[1]: ☐ puntos.

El Presidente: .....

El Secretario: .....

El Vocal: .....

Granada, a ..... de ..... de 20.....

[1]Solamente con un decimal.

Contents

Resumen **iii**

Abstract **1**

1 Introduction **1**
1.1 Motivation and Objectives 1
1.2 Thesis Organization 3

2 Inverse Stochastic Modeling for Fatigue Damage in Composites **4**
2.1 Introduction 5
2.2 Methodology 8
2.2.1 Cumulative damage model using Markov chains . . . 8
2.2.2 Forward problem 10
2.2.3 Inverse problem 12
2.2.4 Model selection by Cross Validation 15
2.3 Numerical results 16
2.3.1 Experimental data 16
2.3.2 GA convergence 19
2.3.3 Inverse problem solution 21
2.3.4 Cross Validation 32
2.4 Discussion 32

3 Reliability in Composites under Damage Conditions **35**
3.1 Introduction 35
3.2 Reliability Formulation 37
3.2.1 Limit State Function 38
3.2.2 Monte Carlo method 39
3.3 Reliability under damage conditions 40
3.4 Numerical example 43
3.5 Conclusions 48

A Monotone piecewise cubic interpolation **49**

B IP-MARKOV algorithm **51**

List of Figures **85**

List of Tables **88**

Bibliography **89**

Chapter 1

Introduction

1.1 Motivation and Objectives

Composite materials are used extensively in the construction of high performance structures such as aeronautical, marine, mechanical and civil structures, which often require high reliability standards. These structures are typically subjected to dynamic loads and hence they are susceptible to long term fatigue failures. The perception of the phenomenon of fatigue has been usually associated with the behavior of homogeneous, isotropic, metallic materials and hence there has been a strong tendency to treat fatigue in composites as though they were metals. It has typically led to oversized designs and more occasionally to some catastrophic failures [1].

Unlike metals, composites are inhomogeneous and anisotropic materials. They accumulate damage in a general damage area rather than a localized area, and failure does not always occur by the propagation of a single macroscopic crack. In addition damage starts early, after only a few or a few hundred loading cycles, and a sharp initial reduction of the stiffness is usually observed. This early damage process is caused by micro-scale damage mechanisms, including fibre breakage, matrix cracking, debonding and delamination. These mechanism can occur sometimes independently

and others interactively, and their prevalences may be strongly affected by both materials variables and testing conditions [2, 3].

The efficient and reliable use of the composite materials in any application will require to account for this damage accumulation process under service conditions such as fatigue loadings. But due to the quite complex nature of the fatigue phenomenon [3–5], a reliable study of the fatigue response should take into account the inherent randomness of the process. There exists physical uncertainty that comes from the material random properties, the random spatial distribution of defects, and imperfections within the material structure. Also, there can be loading uncertainty generated by the randomness of the applied mechanical loads and the environmental conditions. Several studies have been reported in the literature focusing on the uncertainty of composites fatigue phenomena [6–9].

It follows that, in addition to continue understanding the physical mechanisms by which fatigue damage occurs in composites, new phenomenological procedures are needed to predict this cumulative process in a statistical framework, and therefore the reliability and life of the material. In this context, the Inverse Problem together with Genetic Algorithms [10] are shown to be effective to train stochastic damage models from nondestructive damage data. These models are able to make statistical predictions of damage at any time and therefore allow to make predictions of the long term reliability, by inserting them into a suitable failure criteria. This procedure has not been reported before and hence it is the main contribution of this research.

1.2 Thesis Organization

This thesis is organized as follows: The present chapter deals with the motivation and organization of the research work presented herein. Chapter 2 is dedicated to the inverse stochastic problem of predicting damage in composites and it is presented in the format of a scientist paper prepared to be submitted to *Journal of Composite Science and Technology*. Chapter 3 treats the assessment of long term reliability of composites materials under stochastic damage conditions, which is also prepared to be submitted to *Journal of Composites Structures*. Finally, this document is closed with two appendices: Appendix A develops the formulation of the Monotonic Cubic Hermite Interpolation while Appendix B is dedicated to the **IP-MARKOV** algorithm, developed *ad-hoc* for this research.

Chapter 2

Inverse Stochastic Modeling for Fatigue Damage in Composites

Fatigue in composite materials is a complex-multiscale damage cumulative process, detectable from the beginning of the lifecycle [2, 3]. Numerous fatigue models have been proposed in the literature, but they are difficult to extend outside laboratory conditions. Due to the material heterogeneity and random spatial distribution of initial defects, composites show a significant scatter in their fatigue responses and hence deterministic models fit to reality is only relative. A stochastic phenomenological approach that considers damage evolution by means of parametrized Markov chains has been presented in this work. Three new model parameterizations are proposed and compared. An inverse framework has been proposed to find the optimal model parameters that minimize the residual mismatch between model predictions and experimental data by means of a genetic algorithm. Two novel definitions of the residual mismatch based on (1) the statistical distance and (2) cumulative entropy have been proposed. Finally models and residuals are ranked and compared by the cross-validation method based on their predictability. This methodology has been validated against experimental fatigue damage data taken from literature.

2.1 Introduction

Fatigue modeling is an unsolved problem of the composites science and technology. Several scale-level damage mechanisms such as matrix microcracking, fiber breakage, fiber-matrix debonding or delamination take place early during the fatigue process within diffuse areas of the laminate [2, 3]. Their predominance and extension are subjected to uncertainty due mainly to the random distribution of initial defects and heterogeneity [6, 11]. This inherent uncertainty is not considered by most fatigue models, which leads to oversized and uneconomical designs. This research proposes a novel methodology to account for the uncertainty in fatigue damage modeling in a coherent statistical sense.

Numerous fatigue models have been proposed since the boost in the development of the composites technology in the early seventies. Influenced by the metal fatigue experience (S-N curves), *life models* were the first to be accepted and used. Because fatigue failure in composites occur very differently than in metals, some researchers turned their attention to the real damage mechanisms occurring along the fatigue process. *Phenomenological models* were then proposed based on the progression of one or more variables related to any measurable manifestation of damage, such us stiffness and strength reduction, delamination size or matrix crack density. Good examples of both classes of models can be found in [12].

The vast majority of the existing fatigue models are deterministic approaches, and hence they are not able to account for the inherent variability of the process. In many other cases fatigue models are not practical for engineering purposes. Probabilistic phenomenological damage approaches are found to be more suitable for composites materials [32] , but unfortunatelly the extension of such methods is not as mature as deterministic models.

Bogdanoff and Kozin [7] were the first to introduce a stochastic phe-

nomenological model to simulate the lifetime for processes of fatigue, wear and crack propagation of engineering materials. A Markov chain model was proposed to take into account the variability of the process and a *time transformation-condensation* method was developed to take into account the nonstationarity. Later, Rowatt and Spanos [13] extended this model to composite materials. A time transformation-condensation method was also applied in their work to predict the fatigue lifetime from compliance stochastic data. Ganessan [14] discussed the limitations on the validity of the Weibull model for fatigue damage and proposed the use of Markov chains as a suitable approach to model the compliance evolution of composites laminates. Recently, Wei and Johnson [11] have proposed a Markov chain model to predict stochastic *S-N* curves from fatigue damage data. They also provide a good review of stochastic cumulative damage models.

In all of these models, the key task is the inference of the probability transition matrix (PTM) of the process, which summarizes the probability transitions between damage or lifetime states, as a specified function of some unknown parameters. Additionally, the majority of the existing Markov fatigue models do not account explicitly for damage evolution, only for lifetime evolution in which explicit formulas for model parameters are available [7]. Unfortunately fatigue damage modeling in composite materials often require more complicated parameterizations and there no exists explicit formulas to infer them. Consequently statistical superior methods like maximum likelihood estimation [15] or numerical search strategies, as those proposed herein, are required.

Three new Markov damage models are proposed, one stationary (model A) based on a bilinear variation of the stationary PTM elements, and two non stationary models (model B and C) based on a novel *unitary time transformation* concept. Piecewise monotonic cubic splines [16, 17] are originally

used to parameterize the time scale within the non stationary models. An inverse procedure is proposed to find the optimal model parameters which minimize a cost functional that summarize the mismatch between experimental and model predicted measurements. All the process is conducted by a genetic algorithm (GA) to avoid local minima. The main contribution at this point is the formulation of two novel definitions of residuals based upon (1) the statistical distance and (2) the cumulative entropy [18] between model predicted and empirical distribution functions. This model strategy allows for a robust description of the stochastic process since an unlimited number of samples of the experimental process (coupons or specimens) can be taken into account.

Since several models and residuals are proposed a comparative criterion for model-residual selection is required. The cost functional is the first available criterion to compare between models, but it is restricted to the same residual type. In addition, it is a measure of the fitting accuracy for a given set of data, but it does not provide information about the model ability to fit new data. Hence the *prediction error* calculated by the Cross Validation method [19–21] is proposed to be used as absolute criterion to model-residual rank selection and also as a measure of the predictability of a model. This task is particularly relevant given that large samples of fatigue data are usually not available, hence complicated models trained with a reduced experimental set can overfit data and hence reduce the model prediction ability.

All proposed models have been able to simulate the complete experimental stochastic process with a reasonable good fit. However the predictability of models has been rather influenced by the different parameterizations. Additionally, the GA-driven inverse procedure has revealed efficiency in terms of computational cost, as the convergence has taken less than one minute

per model, with a personal computer. As a preliminary conclusion it has been found that models with a tendency to overfit data are able to improve their predictability when trained with the entropic residual.

2.2 Methodology

2.2.1 Cumulative damage model using Markov chains

The evolution of fatigue damage as a function of time is proposed to be modeled by Markov chains, under the main hypothesis established by the *Markov property*, which states that the *future* of the process depends only on its *present* state, which is independent of the *past*. This phenomenological stochastic approach is based on the theory of Markov chains [22] and assumes the following underlying assumptions: [7, 13]

1. Damage is a nondecreasing random variable and it passes through an integer and finite number of states, $j = 1, 2, ..., s$, until the "absorbing" state s is reached.

2. Time is discretized within integer units of *duty cycles* $n = 0, 1, ..., N$. One duty cycle (DC) is a suitably defined period of load cycles in which damage is randomly accumulated.

3. Damage can only increase from a state to the next within a DC.

It follows from the previous remarks that the proposed model is a finite-state (1), discrete-time embedded (2) Markov process in which the damage accumulation mechanism is of the unit-jump type (3). At each integer time n, there is an integer-valued random variable (rv) D_n called the *damage state* at time n and the damage process is family of rv's $\{D_n; n \geqslant 0\}$. This integer-time process can also be viewed as a continuous process $\{D(t); t \geqslant 0\}$

by taking $D(t) = D_n$ for $t \in [n, n+1)$, but changes only occur at integer times, which is usually coincident with fatigue cycles.

Let then the rv D_n represents the damage state at time or duty cycle n. Thus the probability of D_n to be in state j at time n is denoted by

$$\mathrm{P}\left[D_n = j\right] = p_n(j) \tag{2.1}$$

The probability mass function of the rv D_n at time n is given by the $(1 \times s)$ vector

$$\mathbf{p_n} = \{p_n(1), p_n(2), ..., p_n(s)\} \tag{2.2}$$

where

$$\sum_{j=1}^{s} p_n(j) = 1 \tag{2.3}$$

Let now define the probability of damage being in the state k at time $n+1$ given that the process has passed through the states $\{1, 2, ..., j\}$ at the discrete times $\{0, 1, ..., n\}$, as

$$p_{1k} = \mathrm{P}\left[D_{n+1} = k | D_n = j, D_{n-1} = j-1, ..., D_0 = 1\right] \tag{2.4}$$

By the Markov property the future behavior of the process is independent of its past states, since the present state is the only influencer, so that (2.4) can be simplified as

$$p_{1k} = p_{jk} = \mathrm{P}\left[D_{n+1} = k | D_n = j\right] \tag{2.5}$$

Moreover, by assumption (d) damage may increase from a given state j to the one just above $j+1$ within a DC , or in other case, it may remain in the same state j.

Hence all possible transitions within the DC n can be summarized in a $s \times s$ sized double-diagonal Probability Transition Matrix (PTM), as

$$P_n = \begin{pmatrix} p_{11}^{(n)} & p_{12}^{(n)} & & & \\ & p_{22}^{(n)} & p_{23}^{(n)} & & \\ & & \ddots & \ddots & \\ & & & p_{s-1,s-1}^{(n)} & p_{s-1,s}^{(n)} \\ & & & & 1 \end{pmatrix} \tag{2.6}$$

Additionally the PTM satisfies the Chapman-Kolmogorov identity [22], so:

$$\sum_{k=1}^{s} p_{jk}^{(n)} = 1; \quad j = 1, ..., s-1 \tag{2.7}$$

and hence

$$p_{jk}^{(n)} = 1 - p_{jj}^{(n)} > 0 \tag{2.8}$$

From the Markov chains theory [22], the probability distribution of the rv D_N (2.2) is completely determined by the probability mass function of the initial damage, $\mathbf{p}_0$, and the probability transition matrices, P_n , where $n = 0, 1, ..., N$, as

$$\mathbf{p}_N = \mathbf{p}_0 \prod_{n=0}^{N} P_n \tag{2.9}$$

Equation (2.9) provides the fundamental probabilistic information of the stochastic damage model and it is central within the methodology proposed.

2.2.2 Forward problem

The number of independent variables needed to define the Markov model described above are $N \times (s-1)$. The process is supposed to start at the no-damage state, thus $\mathbf{p}_0 = \{1, 0, ..., 0\}$. An unusual stochastic process of 5 states and 20 discrete times would have 80 variables to infer, hence a description of the PTMs as functions of some unknown parameters is mandatory. A two parameter model, the size s of the PTM and the ratio $r_j^{(n)} = p_{jj}^{(n)}/p_{jk}^{(n)}$ can be used as the simplest parameterization assuming a stationary ($r_j^{(n)} = r_j$) and state-independent process ($r_j = r$) [7]. However fatigue in composite materials is often a non-stationary and state-dependent damage process and then require more elaborated parameterizations.

Three alternative models are proposed and compared with the simplest model (*sr* model): A five parameter state-dependent stationary model (model A), a six parameter state-independent non-stationary model (model B) and a four parameter state-independent non-stationary model (model

C). The first model assumes a monotonic bilinear variation of q_j while the PTM matrix remains invariant for the entire process. In models B and C the nonstationarity is accounted by a transformation of the unitary time scale x, to the transformed scale y, by means of a parameterized monotonic cubic spline [1] $y : y\left(x; \alpha_1, \beta_1, \alpha_2, \beta_2\right)$ allowing the probabilities of transition between states p and q remain invariants during the process. Mathematically:

Model A: $\boldsymbol{\theta_A} = \{s, q_1, q_{s-1}, \alpha, \beta\}$

$$\mathbf{p}_n = \mathbf{p}_0 \begin{pmatrix} p_1 & q_1 & & & \\ & p_2 & q_2 & & \\ & & \ddots & \ddots & \\ & & & p_{s-1} & q_{s-1} \\ & & & & 1 \end{pmatrix}^n \tag{2.10a}$$

$$q_j = q_1 + (q_{s-1} - q_1) \cdot \phi(\xi; \alpha, \beta) \tag{2.10b}$$

$$\phi(\xi; \alpha, \beta) = \begin{cases} \frac{\beta}{\alpha}\xi & if\ \xi < \alpha \\ \frac{\beta}{\alpha}(\xi - \alpha) + \beta & if\ \xi \geqslant \alpha \end{cases} \tag{2.10c}$$

$$\xi = \frac{j-1}{s-1}, \quad j = 1, ..., s \tag{2.10d}$$

$$p_j = 1 - q_j \tag{2.10e}$$

[1] See Appendix A

Model B: $\boldsymbol{\theta_B} = \{s, p, \alpha_1, \beta_1, \alpha_2, \beta_2\}$

$$\mathbf{p}_n = \mathbf{p}_0 \begin{pmatrix} p & q & & & \\ & p & q & & \\ & & \ddots & \ddots & \\ & & & p & q \\ & & & & 1 \end{pmatrix}^{m(n)} \tag{2.11a}$$

$$m(n) = n \cdot y\left(x; \alpha_1, \beta_1, \alpha_2, \beta_2\right) \tag{2.11b}$$

$$x, y \in [0, 1] \tag{2.11c}$$

$$\alpha_1 < \alpha_2 \in [0, 1] \tag{2.11d}$$

$$\beta_1 < \beta_2 \in [0, 1] \tag{2.11e}$$

$$q = 1 - p \tag{2.11f}$$

Model C: $\boldsymbol{\theta_C} = \{s, p, \alpha_1, \beta_1\}$

$$\mathbf{p}_n = \mathbf{p}_0 \begin{pmatrix} p & q & & & \\ & p & q & & \\ & & \ddots & \ddots & \\ & & & p & q \\ & & & & 1 \end{pmatrix}^{m(n)} \tag{2.12a}$$

$$m(n) = n \cdot y\left(x; \alpha_1, \beta_1\right) \tag{2.12b}$$

$$x, y \in [0, 1] \tag{2.12c}$$

$$\alpha_1, \beta_1 \in [0, 1] \tag{2.12d}$$

$$q = 1 - p \tag{2.12e}$$

2.2.3 Inverse problem

The estimation of model parameters by the Inverse Problem (IP) can be stated as the minimization problem of the discrepancy between model predicted and experimental measurements. The approach used herein is to

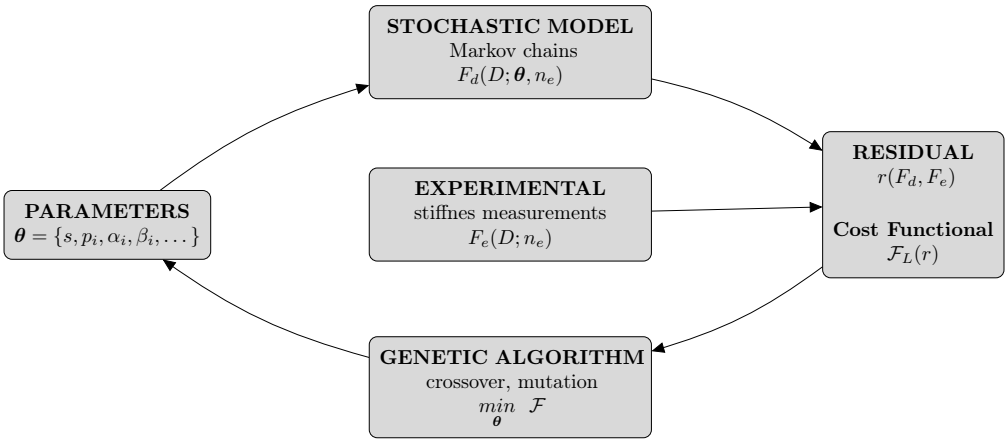

Figure 2.1: Inverse procedure

use a Genetic Algorithm (GA)[10] to iteratively search the set of model parameters $\boldsymbol{\theta}$ that minimizes a *cost functional* that quantify the model-data mismatch. Other search algorithms such gradient-based or simulated annealing can be used for the same aim but GA is preferred by its efficiency exploring the whole model space avoiding local minima.

Let $F_e(D;n_e)$ be the empirical cumulative distribution function of damage D at time n_e and $F_d(D;\boldsymbol{\theta},n_e)$ the CDF of damage D at time n_e predicted by a model parameterized by $\boldsymbol{\theta}$. A population $\Psi_g=\{\boldsymbol{\theta}^{(1)};\cdots;\boldsymbol{\theta}^{(h)}\}$ of h possible solutions or chromosomes is randomly generated. Each chromosome $\boldsymbol{\theta}^{(i)}$ is introduced as a input within the forward problem (eqs 2.10, 2.11) and the cost functional integrates the discrepancy r between $F_e(D;\mathbf{t}_e)$ and $F_d(D;\boldsymbol{\theta}^{(i)},\mathbf{t}_e)$ along the empirical times $\mathbf{t}_e=\{0,\cdots,n_e,\cdots,N_e\}$. Genetic operators such as crossover and mutation are iteratively applied to obtain new populations until the maximum number of generations is reached.

Three different expressions for the evaluation of the discrepancy are proposed based on well-established *statistical distance* concepts. The first of them uses the integral of the squared difference between F_e and F_d as a ℓ_2-norm type distance [23, 24]:

$$r(\boldsymbol{\theta},n_e)=\int_0^1\left[F_e(D;n_e)-F_d(D;\boldsymbol{\theta},n_e)\right]^2\,\mathrm{d}D \tag{2.13}$$

The second type proposed is a ℓ_1 variant of the former definition (2.13) and it is defined as[25]:

$$r(\boldsymbol{\theta}, n_e) = \int_0^1 |F_e(D; n_e) - F_d(D; \boldsymbol{\theta}, n_e)| \, \mathrm{d}D \tag{2.14}$$

Finally an alternative definition of residual based on the concept of cumulative entropy ($\mathcal{E}_c$) [18, 26] is proposed. From this concept, a modified version of the Jensen-Shannon divergence is adopted as residual. This residual can be interpreted as a measure of the information gained when F_d closes to F_e. It is defined as:

$$r(\boldsymbol{\theta}, n_e) = \mathcal{E}_c\Big(\frac{1}{2}F_d + \frac{1}{2}F_e\Big) - \frac{1}{2}\Big(\mathcal{E}_c(F_d) + \mathcal{E}_c(F_e)\Big) \tag{2.15}$$

where

$$\mathcal{E}_c = -\int_0^1 F(D) log F(D) \mathrm{d}D \tag{2.16}$$

The discrepancy between $F_e(D; \mathbf{t}_e)$ and $F_d(D; \boldsymbol{\theta}, \mathbf{t}_e)$ for all $n_e \in \mathbf{t}_e$ is stored within a *residual vector* $\mathbf{r}$, defined for each candidate $\boldsymbol{\theta}$ as:

$$\mathbf{r}(\boldsymbol{\theta}) = \{r(\boldsymbol{\theta}, 1), \cdots, r(\boldsymbol{\theta}, N_e)\} \tag{2.17}$$

Since two residual vectors cannot be compared directly, a scalar number is derived by means a cost functional $\mathcal{F}$ defined as the ℓ_2 norm of the residue vector (**??**):

$$\mathcal{F}(\boldsymbol{\theta}) = \|\mathbf{r}(\boldsymbol{\theta}, n_e)\|_2 = \sqrt{\sum_{n_e=0}^{N_e} r(\boldsymbol{\theta}, n_e)^2} \tag{2.18}$$

To improve the identifiability and the convergence speed of the GA, an alternative definition of the cost functional has been adopted [27]:

$$\mathcal{F}_L = \log(\mathcal{F} + \epsilon) \tag{2.19}$$

where ϵ is a small non-dimensional value (here adopted $\epsilon = 10^{-20}$) that ensures the existence of $\mathcal{F}_L$ when $\mathcal{F}$ tends to zero.

2.2.4 Model selection by Cross Validation

Cross Validation (CV) is a standard heuristic for finding the right model architecture among a heterogeneous class of models based on a comparative of their prediction error (P_E), i.e. the expected loss of the estimated model evaluated on future observations [19, 28]. In the application of CV, some samples are left out for validation (validation set), while other samples are used for calibration (calibration set). If only one sample is left out for validation, the method is known as leave-one-out cross validation (LOO-CV). This last method has been proven to be asymptotically inconsistent, in the sense that the PE estimation does not converge to the true PE as the data set approaches to infinity , so it will not be used here [29]. This deficiency of LOO-CV is overcome by using leave-multiple-out cross-validation, or simply called cross-validation, which provides a nearly unbiased estimate of the PE.

The available data set $\mathcal{D} = \{D_1, \ldots, D_{N_e}\}$ is randomly split into K disjunt subsets $\mathcal{D}_1, \ldots, \mathcal{D}_K$ of approximately equal size. Each subset $\mathcal{D}_i$ contains a collection of v random variables $\mathcal{D}_i = \{D_{i1}, \ldots, D_{iv}\}$ where $v = {}^{N_e}/_{K}$, each one with mean and standard deviation $(\mu_{D_{ij}}, \sigma_{D_{ij}})$

For each $i \in \{1, \cdots, K\}$ the model candidate $\mathcal{M}$ is fitted on $\mathcal{D} - \mathcal{D}_i$ and evaluated on $\mathcal{D}_i$ as:

$$P_{Ei} = \frac{1}{v} \sum_{j=1}^{v} (\mu_{D_{ij}} - \hat{\mu}_{D_{ij}})^2 + (\sigma_{D_{ij}} - \hat{\sigma}_{D_{ij}})^2 \tag{2.20}$$

The prediction error calculated as (2.20) is averaged over the K folds,

hence:

$$P_E^{(n)} = \frac{1}{K} \sum_{i=1}^{K} P_{Ei} \tag{2.21}$$

As the CV estimate of (P_E) is a random number which depends on a random division of the data set, the method is repeated N times using different splits into folds in order to obtain a Monte Carlo estimation of the random variable P_E: $\{P_E^{(1)}, \cdots, P_E^{(n)}, \cdots, P_E^{(N)}\}$.

2.3 Numerical results

2.3.1 Experimental data

In this section, the modeling procedure described above is illustrated. Stochastic damage data for sixteen quasi-isotropic open-hole S2-glass laminates have been taken from the work of Wei et al. [11]. Details regarding the manufacture of samples, experimental set-up, measurements, etc were reported in this work and hence, they are not repeated here. In essence, each specimen is subjected to a constant amplitude $T - T$ fatigue loading $(R = 0.1, f = 5\,Hz, \sigma_{max} = 0.5\sigma_u)$ and twenty five measurements of longitudinal stiffness are registered as fatigue response within a not-regularly spaced time interval.

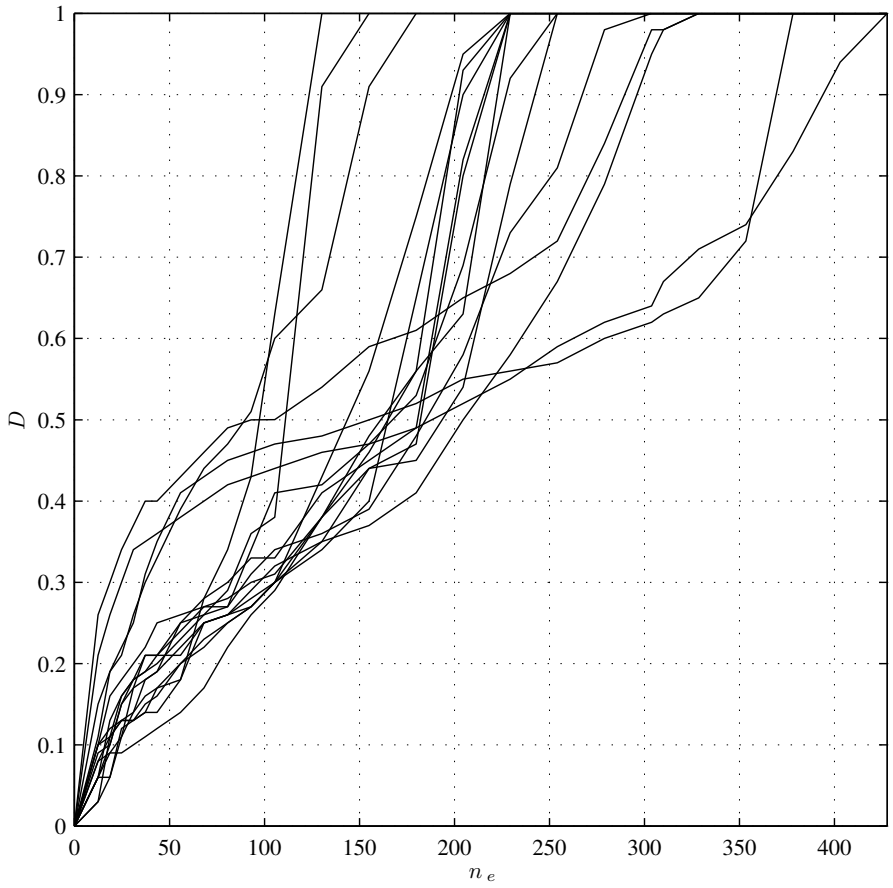

Figure 2.2: Experimental samples of damage as a stiffness reduction over time. The scattering increase with time

The *absorbing* state is reached ($D_{n_e} = 1$) when the stiffness decreases up to 60% of E_0, as reported in [11]. Hence damage at sample time n_e is indirectly measured from the stiffness data E_{n_e} as:

$$D_{n_e} = \begin{cases} \dfrac{(E_0 - E_{n_e})}{0.4\, E_0} & \text{if} \quad E_{n_e} \geqslant 0.6\, E_0 \\ 1 & \text{if} \quad E_{n_e} < 0.6\, E_0 \end{cases} \tag{2.22}$$

where E_0 is the initial stiffness for which $D_{n_e} = 0$. Damage data calculated as (2.22) are plotted here as sample realizations in Figure 2.2.

Empirical cumulative distribution functions of damage are calculated at each n_e from damage data of the 16 specimens as:

$$F_e(D; n_e) = \frac{1}{16} \sum_{i=1}^{16} \mathbf{1}_{[0,D)}(D_{n_e}^{(i)}) \tag{2.23}$$

The selection of the proper value of DC is carried out by means a parametric study concerning the IP accuracy ($\mathcal{F}_L$) as a function of DC duration in fatigue cycles. The inverse algorithm is run ten times for each DC value and the IP error is calculated by averaging the cost functional. The process is repeated for each model and each residual type and the results are presented in Figure 2.3. Lower values of DC lead to good fitting accuracies but at a higher computational expense, and viceversa. Thus, as a compromise solution, one duty cycle is taken to be 500 load cycles for this study, hence

$$n_e = \frac{t_e}{500} \tag{2.24}$$

where t_e is the number of fatigue cycles.

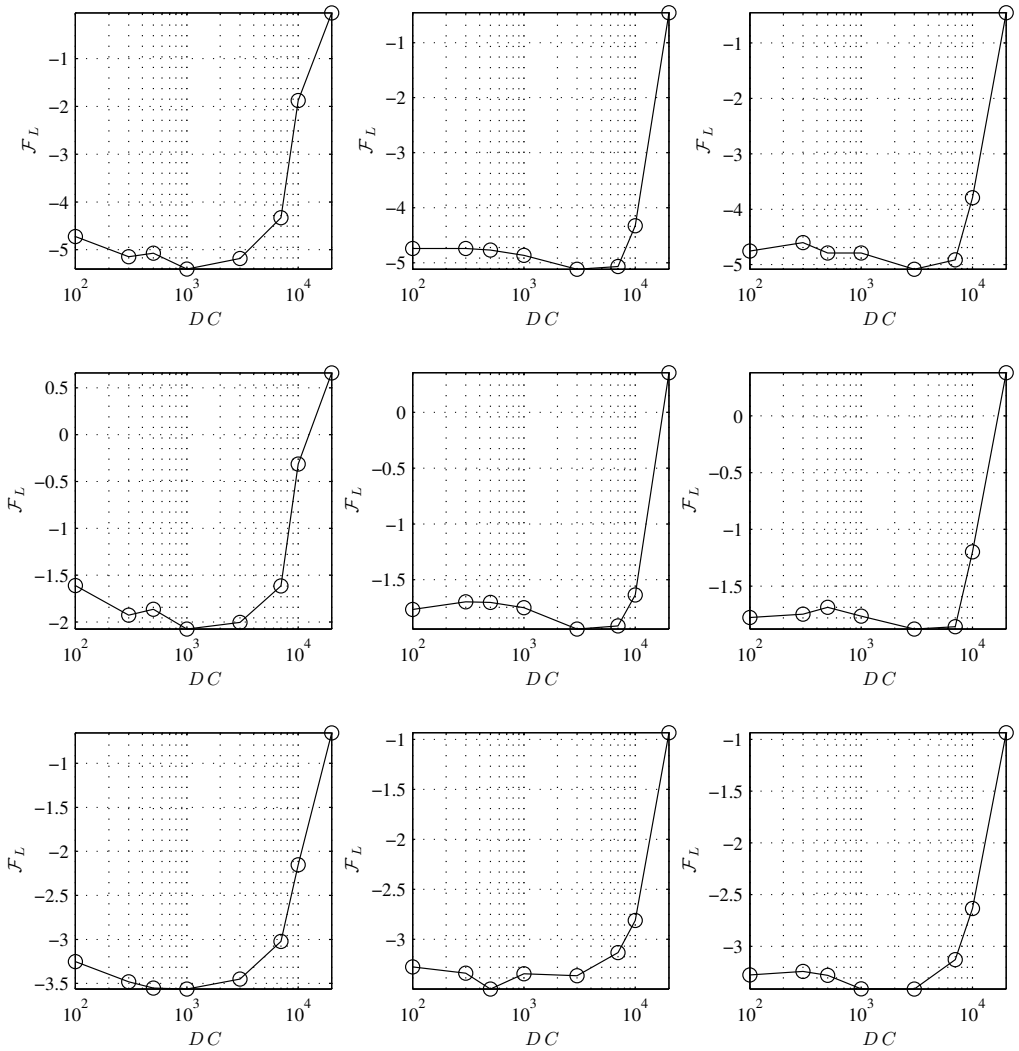

Figure 2.3: Influence of the DC election over the cost functional. By columns from left to right: Model A, model B, model C, respectively. By rows from top to bottom: Residual 1, residual 2, residual 3, respectively. Clearly there exists an upper accuracy limit for DC.

2.3.2 GA convergence

A high number of generations together with large populations can provide excellent convergence results for the GA but it is at the expense of a high computational cost. In this section the search algorithm is studied establishing a compromise between the IP accuracy and the computational cost. In Figure 2.4 the cost functional ($\mathcal{F}_L$) is represented for different values of population size and generations, for each model-residual election.

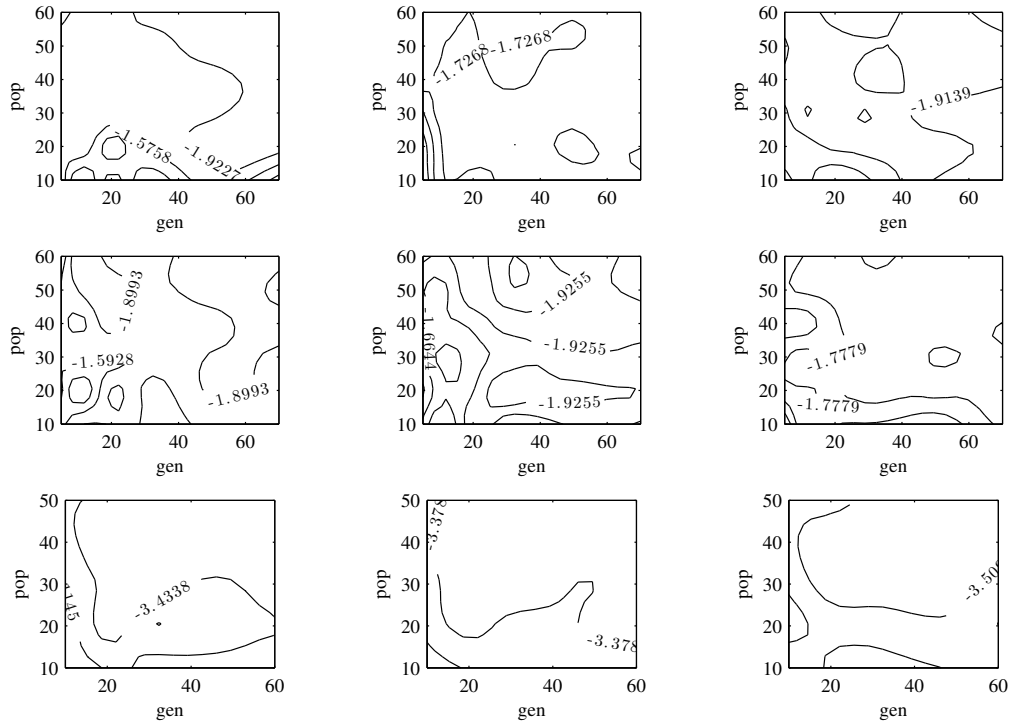

Figure 2.4: GA convergence. By columns from left to right: Model A, model B, model C, respectively. By rows from top to bottom: Residual 1, residual 2, residual 3, respectively. Note that models trained with entropic residual provides smoother GA convergences.

From Figure 2.4 the parameters for the GA search are selected.

Parameter	Model A	Model B	Model *C*
Population size	50	60	50
No. of generations	60	60	50
Prob. of crossover	0.80	0.80	0.80
Prob. of mutation	0.10	0.10	0.10
Prob. of selection	0.70	0.70	0.70

Table 2.1: Parameter setup for GA

The algorithm is stopped when the total number of generation reaches the values shown in Table 2.1 or when the convergence fall to the tolerance limit, fixed at 10^{-30}.

2.3.3 Inverse problem solution

A set of optimal model parameters have been found for each model (Table 2.2). CDFs of damage predicted by models have been compared with those experimentally determined by Equation (2.23). Additionally, model predicted and experimental determined mean and coefficient of variation of damage are respectively compared and plotted in Figure 2.14.

Parameter	Residual 1	Residual 2	Residual 3
MODEL A			
s	24	24	26
q_1	0.249181	0.280566	0.250915
q_{s-1}	0.078543	0.078058	0.079019
α	0.122154	0.120968	0.180314
β	0.999000	0.999000	0.998997
MODEL B			
s	25	25	27
p	0.880683	0.881128	0.903399
α_1	0.087957	0.088012	0.157744
β_1	0.076331	0.076446	0.097282
α_2	0.226531	0.226555	0.281078
β_2	0.357724	0.357813	0.357695
MODEL C			
s	28	28	28
p	0.916719	0.914797	0.916739
α_1	0.505151	0.626855	0.540549
β_1	0.407527	0.540407	0.437133
MODEL sr			
s	34	34	34
r	6.9719	7.0781	7.5046

Table 2.2: Inverse Problem solution.

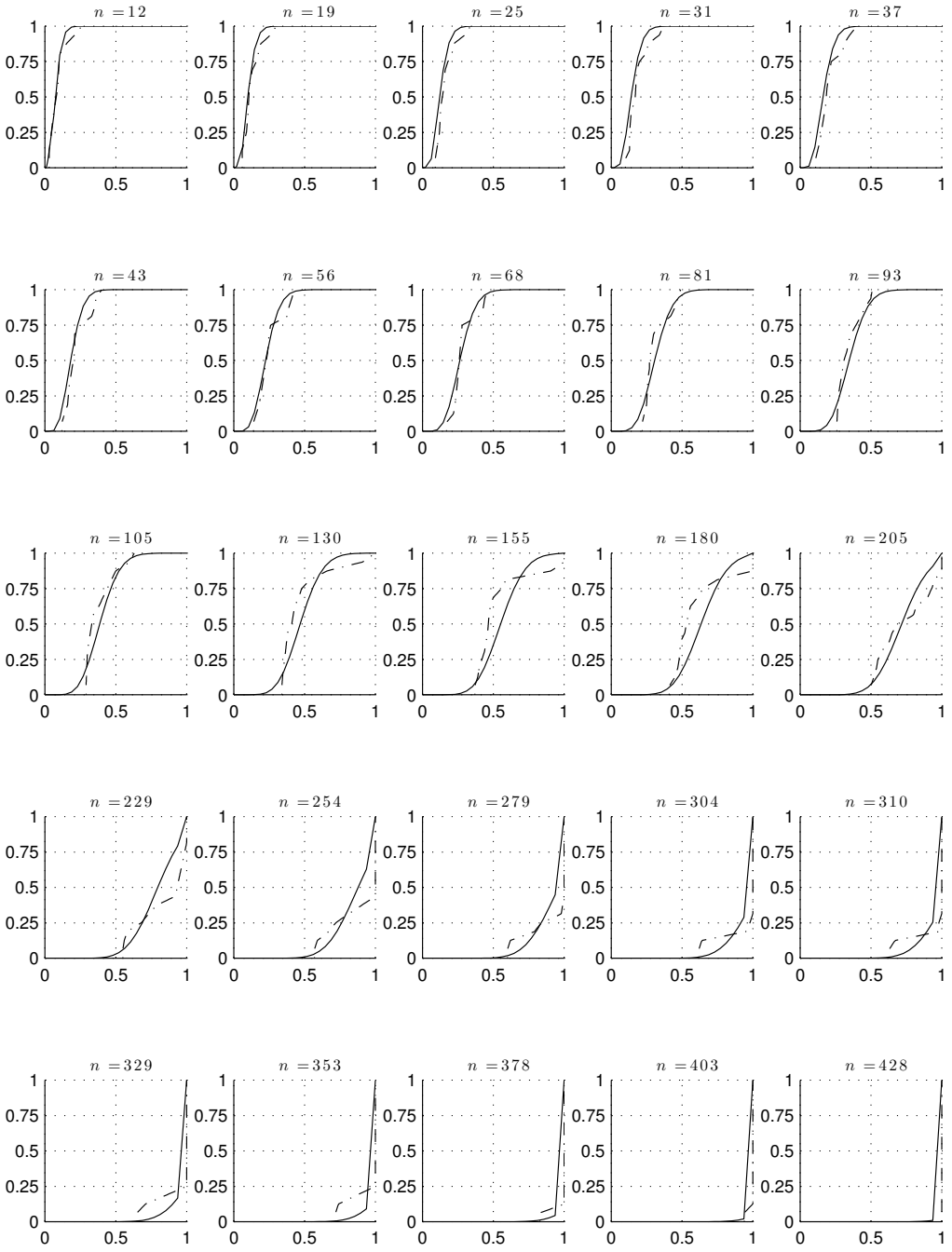

Figure 2.5: Model prediction of the complete stochastic process. Model A trained with residual 1.

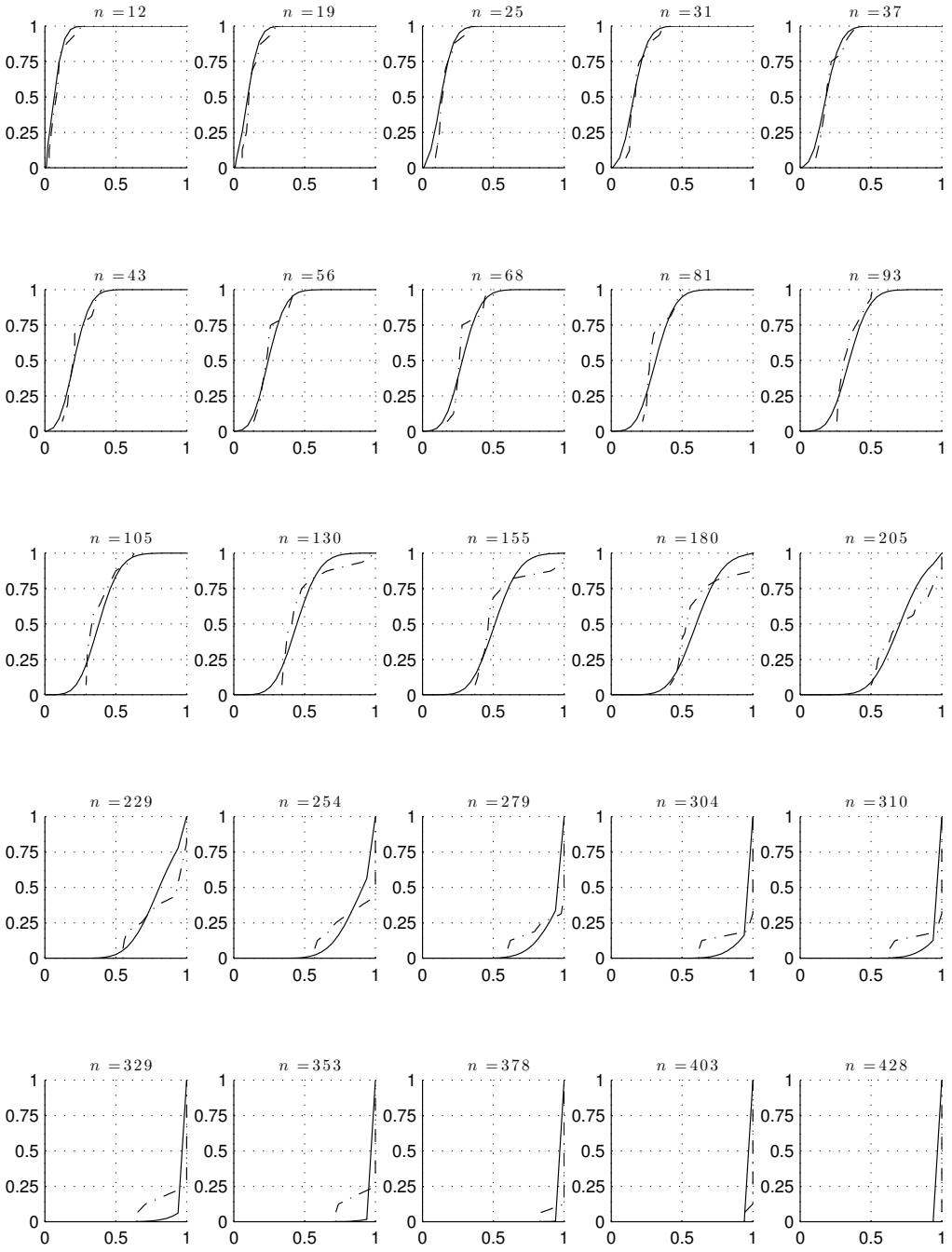

Figure 2.6: Model prediction of the complete stochastic process. Model B trained with residual 1.

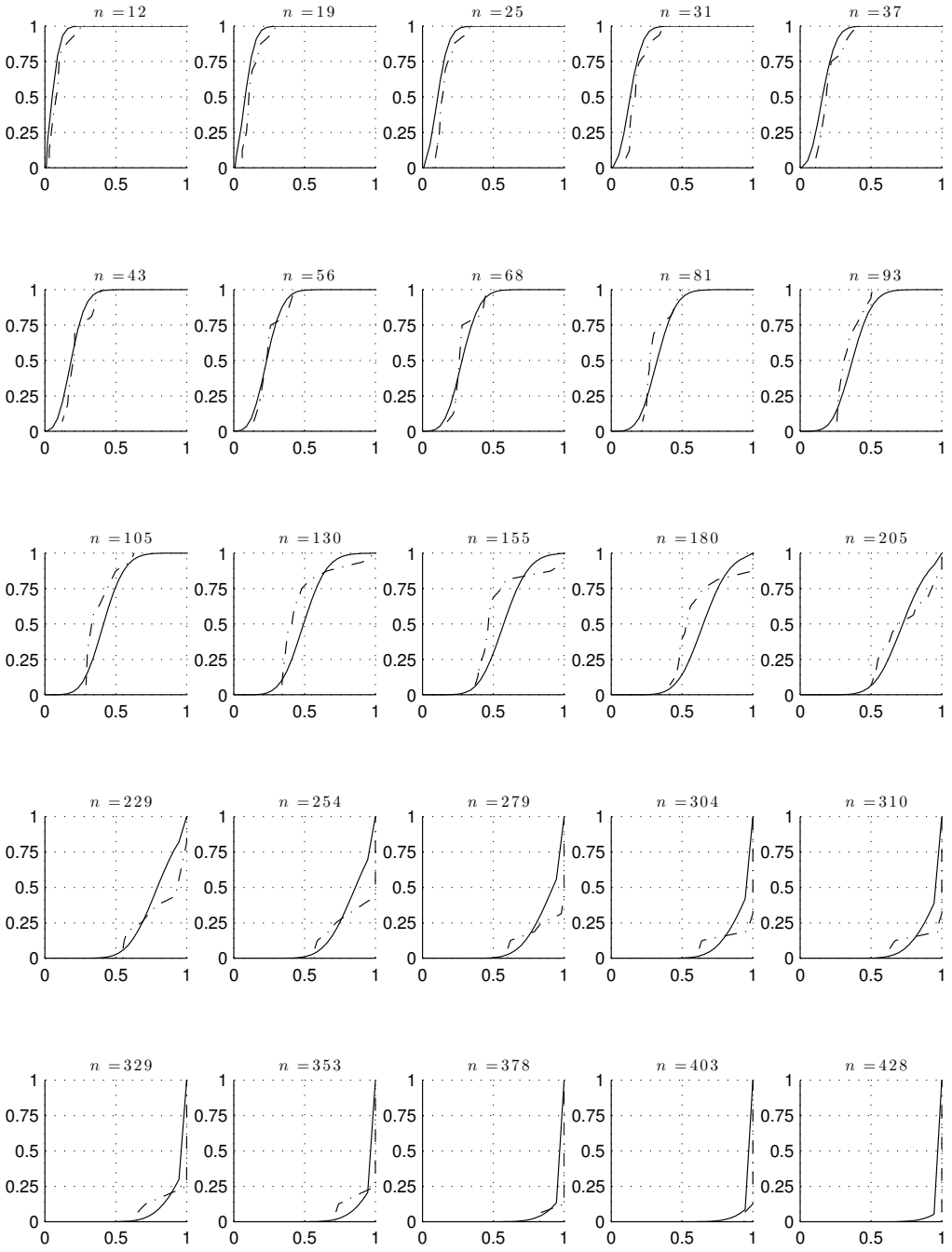

Figure 2.7: Model prediction of the complete stochastic process. Model C trained with residual 1.

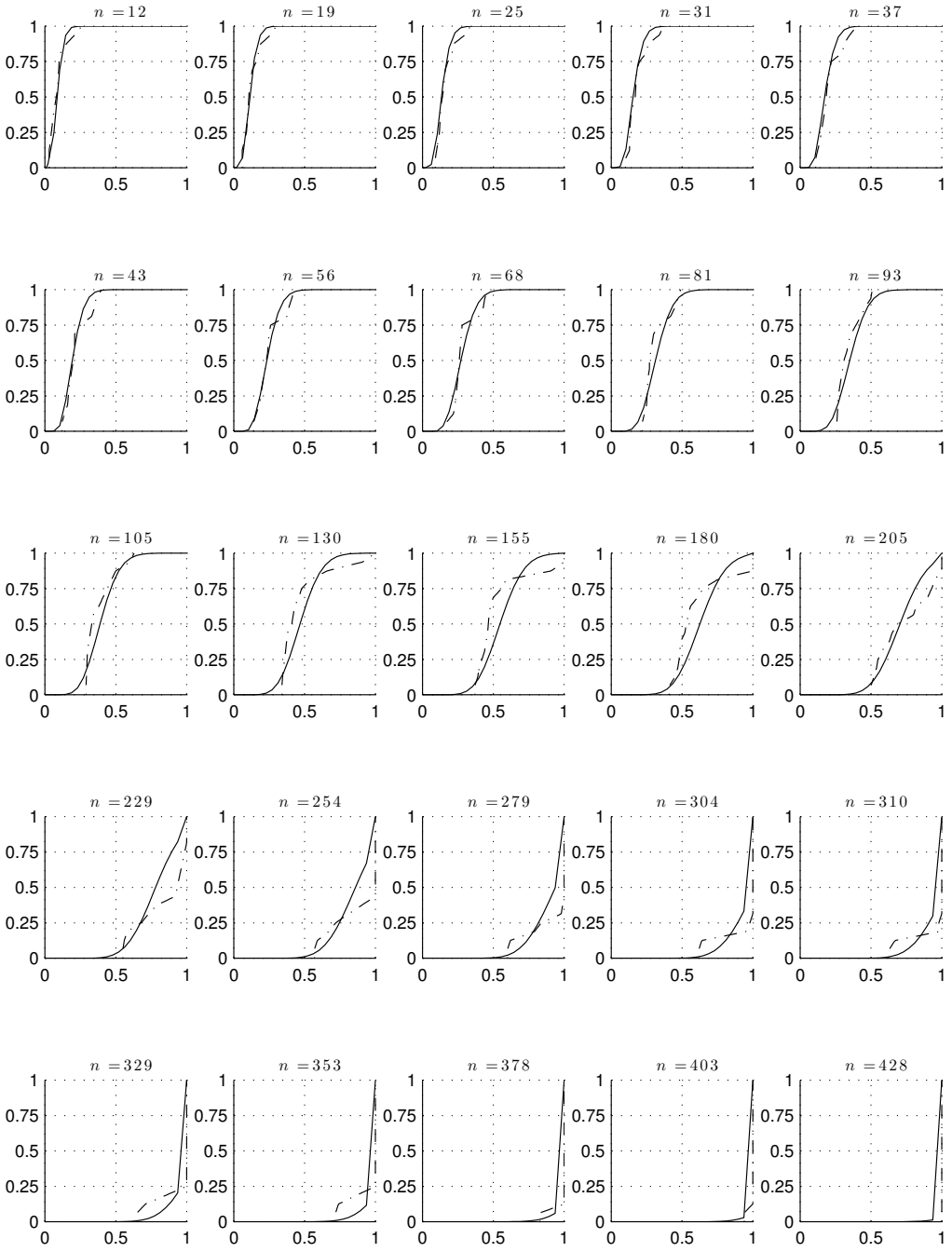

Figure 2.8: Model prediction of the complete stochastic process. Model A trained with residual 2.

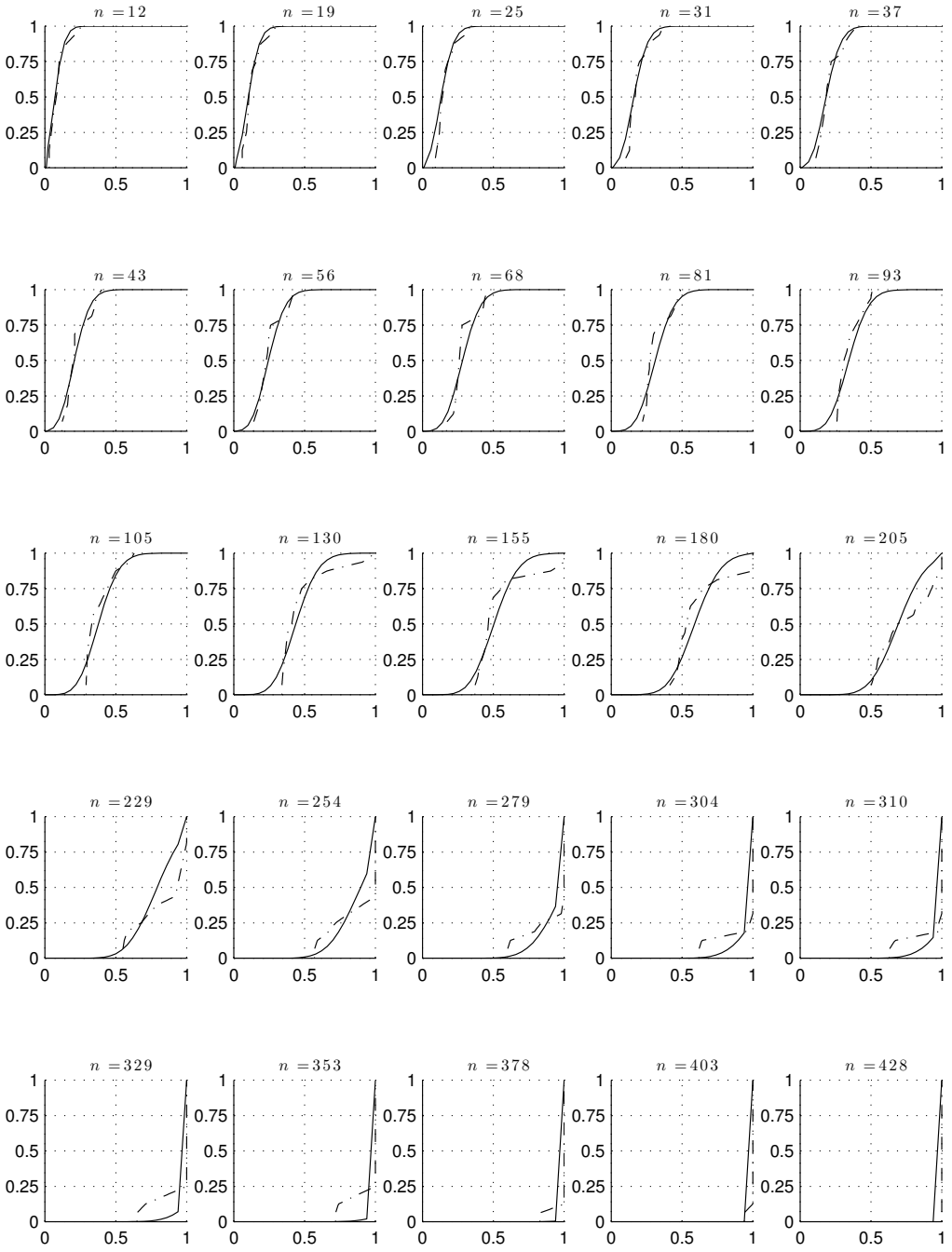

Figure 2.9: Model prediction of the complete stochastic process. Model B trained with residual 2.

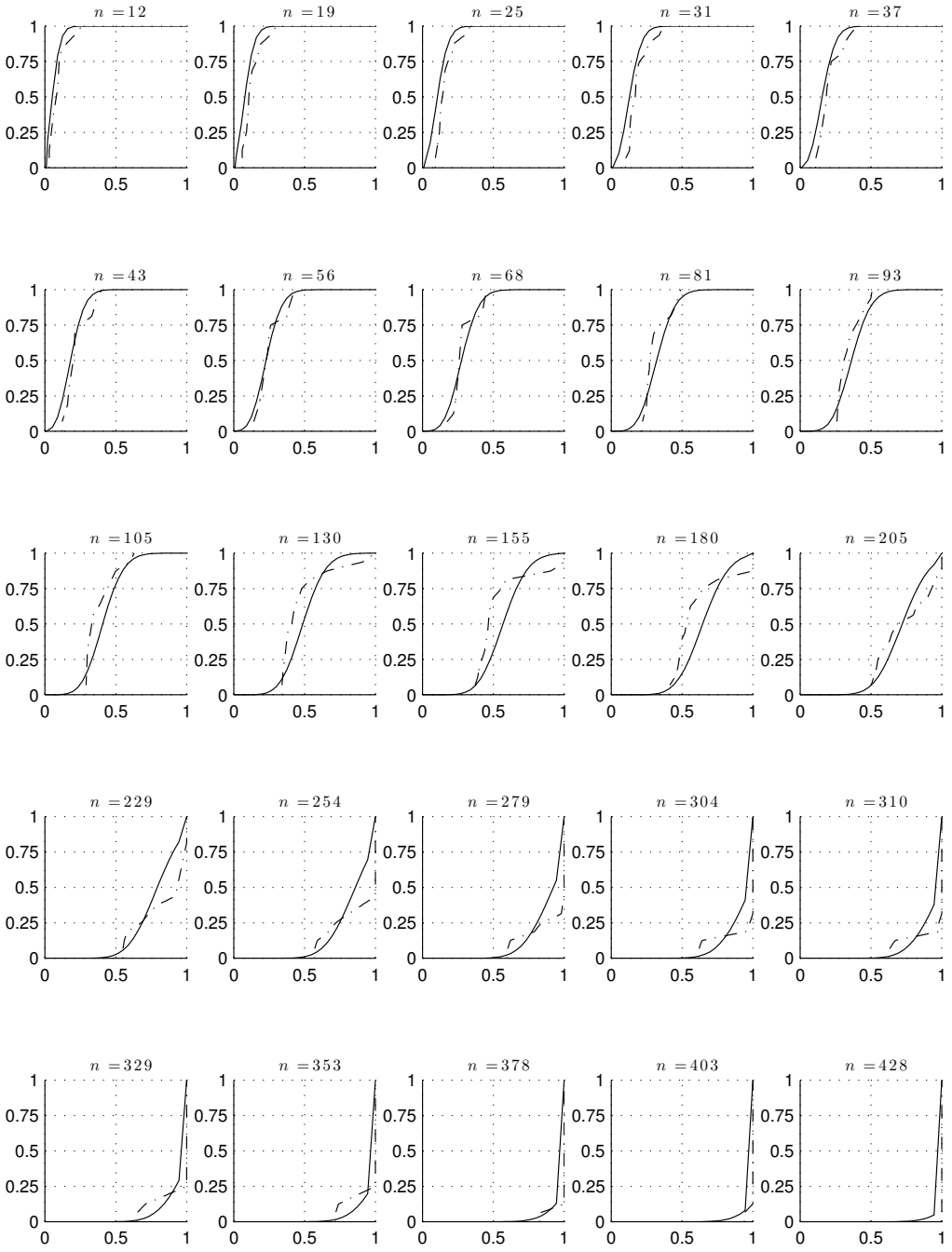

Figure 2.10: Model prediction of the complete stochastic process. Model C trained with residual 2.

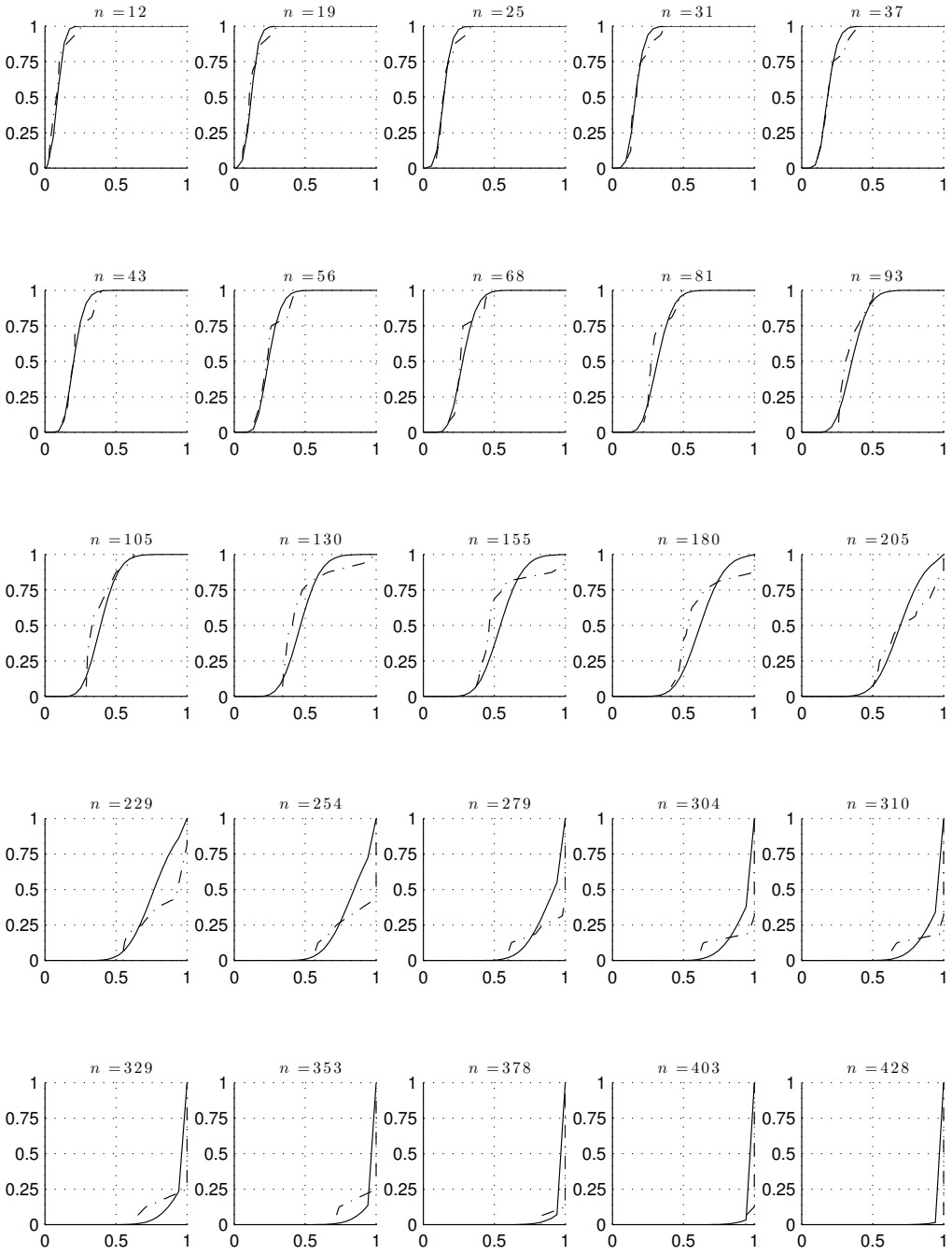

Figure 2.11: Model prediction of the complete stochastic process. Model A trained with residual 3.

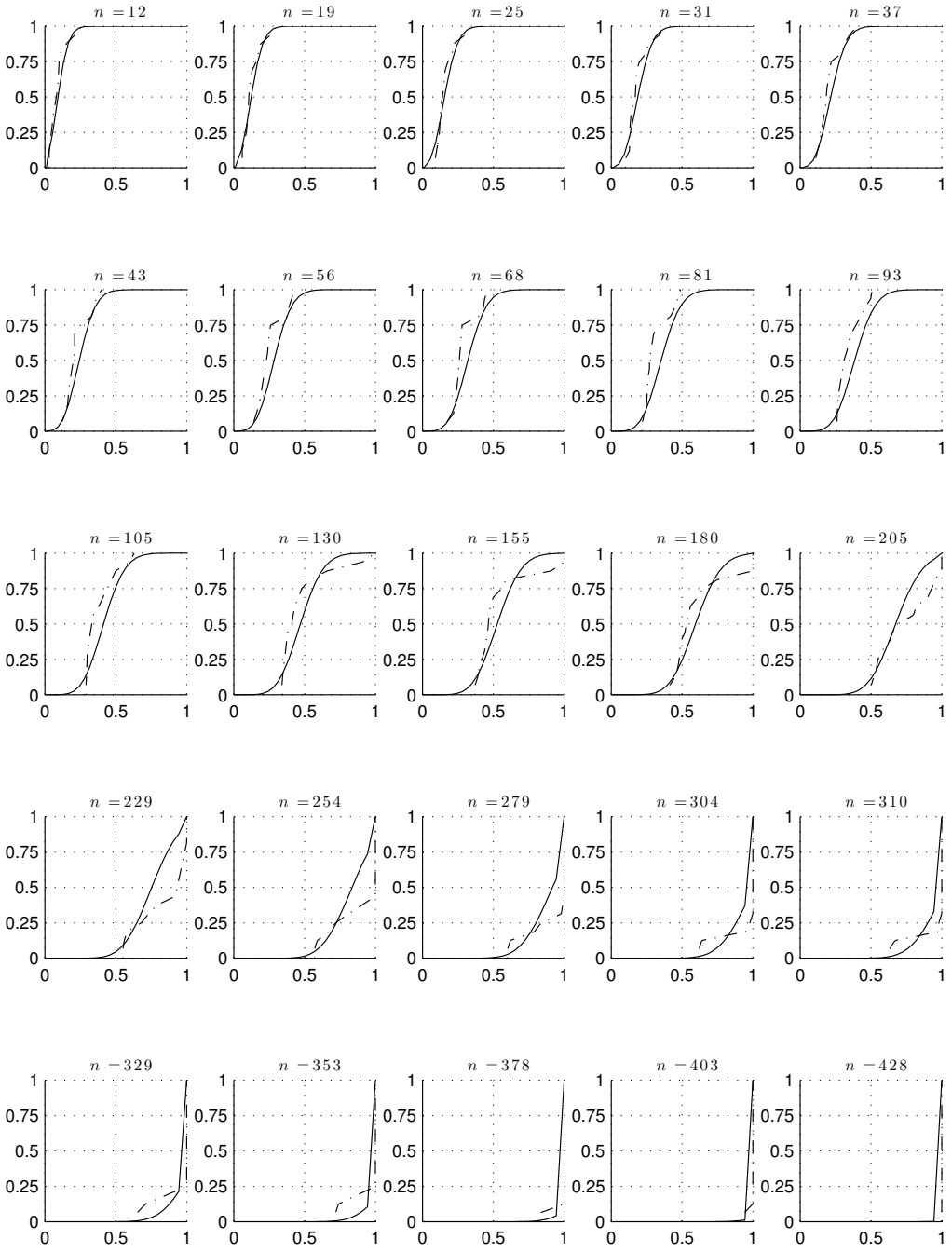

Figure 2.12: Model prediction of the complete stochastic process. Model B trained with residual 3.

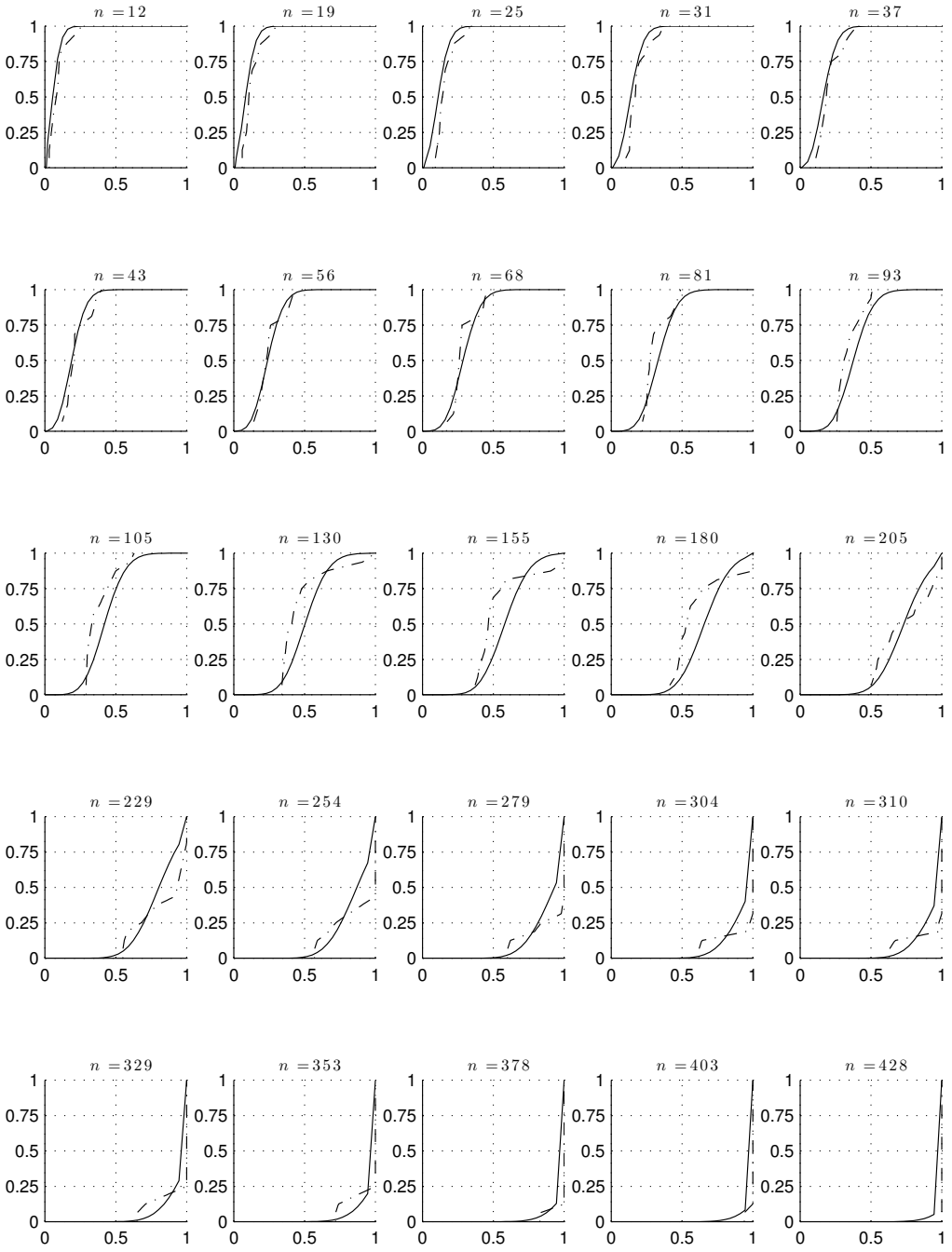

Figure 2.13: Model prediction of the complete stochastic process. Model C trained with residual 3.

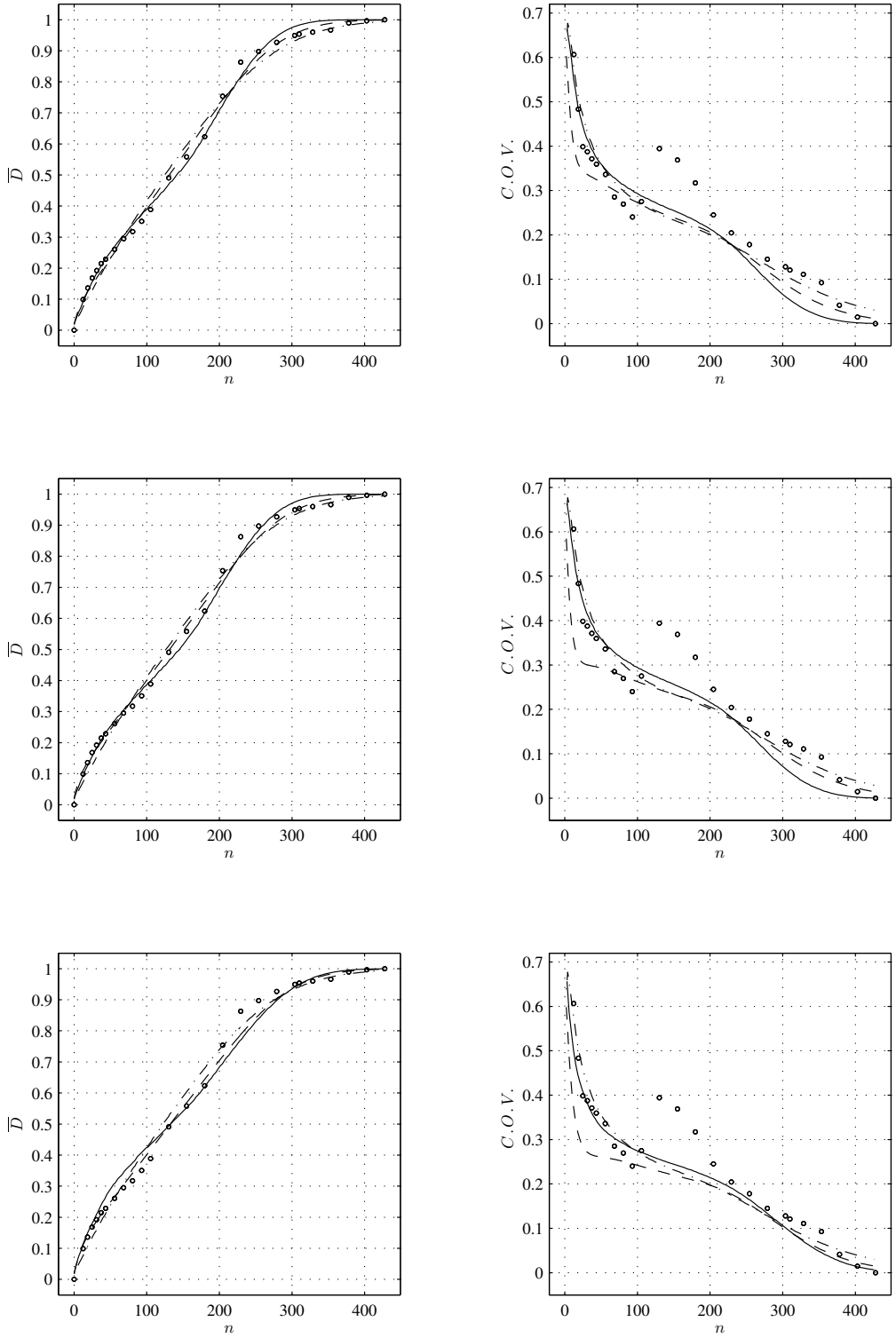

Figure 2.14: Moments predicted at times not covered by data. Dashed, solid and dot-dashed line: Model A, model B and model C, respectively. Dots: Experimental data. Rows from up to bottom: Residual 1, residual 2, residual 3, respectively.

2.3.4 Cross Validation

The whole data set reported in [11] and used here consists on a collection of 25 random variables (rv), one for each time in which damage is measured $\{D_1, \ldots, D_{25}\}$. These rv are randomly divided into ten folds ($K = 10$) and the model candidate is trained and evaluated 10 times, one for each division, following the methodology above. Given that an identical integer number of rv occupying each fold is not possible, 5 folds are occupied by 2 rv while the other 5 are occupied by 3 rv.

Residual	Model A		Model B		Model C		Model *sr*	
	μ	σ	μ	σ	μ	σ	μ	σ
1	**0.0263**	**0.0008**	0.0499	0.0069	**0.0298**	**0.0020**	**0.0310**	**0.0007**
2	0.0265	0.0006	0.0530	0.0050	0.0322	0.0049	0.0315	0.0008
3	0.0280	0.0014	**0.0364**	**0.0029**	0.0346	0.0026	0.0317	0.0008

Table 2.3: Monte Carlo estimation of mean and variance Prediction Error

The prediction error, calculated following Equation 2.20, is averaged over the 10 divisions and the whole process is repeated 25 times to obtain $N = 25$ samples of the prediction error. The PE mean and standard deviation is calculated for all models and all residuals, and are summarized in Table 2.3.

2.4 Discussion

The three models proposed are capable to accurately simulate the temporal evolution of CDF of damage with a reduced set of parameters (Figures 2.5 to 2.13). The mean and coefficient of variation of damage are also closely predicted at times not covered by data. In a principle, the residual election seems not to have a decisive importance in the fitting accuracy of models, however Figure 2.14 reveals that model B trained with residual 3 fits worse than it does with residual 1 and 2. Additionally, if one look at Table 2.3,

the prediction error of model B is drastically decreased when trained with residual 3, what means that this loss of fitting accuracy is the expense of a gain in the predictability of the model.

Regarding to the IP solution of model parameters, models trained with residuals 1 and 2 provide almost identical solutions, varying moderately from the solution using residual 3, as shown in Table 2.2. Moreover, it is noted that the number of damage states increases for the "weakest"models C and *br*. This is due to the fact that a higher number of states increases the model fitting accuracy for a fixed value of DC [30], which show evidence that the number of damage states has to be introduced within the problem, as an optimization variable.

The selection of the suitable value of DC in units of fatigue cycles plays an important role for the fitting accuracy of models. If DC increases the computational cost decreases but the accuracy limit can decreases. On the contrary, too small values of DC lead to an increase of computation time but also to numerical imprecisions caused by raise large matrices with near zero entries to large exponents. The sensitivity analysis presented in Figure 2.3 reflects both effects. Nonstationary models allows for higher values of DC without loss of accuracy and they seem to be more immunes to numerical imprecisions for low DC, hence they seem to be less sensitive than stationary model against the choose of DC.

Relating to the predictive capacity of models evaluated by the prediction error estimated by the (Monte Carlo) Cross-Validation method, Table 2.3 model A is the best predictor for the given set of data while model B does worst. Model B trained with residuals 1 and 2 exhibit a clear tendency to overfit data B, however it disappears when it is trained with residual 3. At the moment, there is not enough information to generalize this observation, so we prefer only to account for it.

As a discussable limitation associated with the proposed methodology is that it is "data-drive" in nature. Fortunately, the structure of this methodology minimizes the amount of data needed for the model construction and is more immune to noise data by incorporating the idea of a residual based on statistical distance between CDF, avoiding to infer model parameters from moments of data. This fact and its inherent simplicity greatly increases the applicability of the method to "real life" situations.

Chapter 3

Reliability in Composites under Damage Conditions

A statistically consistent method to asses the long term fatigue reliability in the framework of a macro-scale cumulative damage process is proposed. The stochastic damage model discussed in Chapter 2 is originally incorporated into the reliability problem. It allows to account for the real "path" of successive damage states inferred from stochastic data to predict the "path" of the long term failure probability. This methodology is validated against experimental data taken from the literature. A modified quadratic Tsai-Wu failure criteria is adopted. Finally the reliability problem has been resolved by the Monte Carlo method together with the Bootstrap technique.

3.1 Introduction

The gradual deterioration of the composite material under fatigue loadings induces changes in both strength and stiffness and hence leads to a continuous redistribution of stresses within the damage areas [31]. The reliability assessment depends upon stresses and strengths, which are stochastic processes under fatigue conditions. Hence the variation of the reliability along

the fatigue process should be predicted by establishing consistent relationships between a stochastic damage model and a failure criterion, in the framework of the continuum damage mechanics [50]. This methodology allows to estimate the long term fatigue reliability accounting for the real "path" of successive damage states by a stochastic damage model inferred from data.

In the reliability literature, only few works have considered the damage as a variable inserted into the composite failure function to derive reliability. Kam [32] considered a limit state function from a damage model based on a linear relation of time to failure. Others authors considered damage as a deterministic non linearity into the composite failure function. In the work of Richard [33], damage was studied as an elasto-viscoplastic model to derive relations between stress and strains. Carbillet et al. [34] derived an extension of this work for strongly non linear behavior caused by damage. As a drawback, all of this approaches are based on assumptions over cumulative damage modeling. Finally, Van Paepegem and Degrieck [31, 35] proposed a coupled formulation of reliability with damage by means of the concept of the effective stress from the continuum damage mechanics. This approach is follow herein.

In this paper an inverse problem is applied to infer the fatigue damage process, modeled as parameterized Markov chains. Three different parameterizations for the fatigue damage model are proposed. To obtain the change of probabilistic failure, model predicted probability distribution functions of damage, are considered inside a failure criterion to account the reserve of failure due to the stochastic damage accumulation. Through this, the path of successive damage states is not only considered [31, 35] but also the full statistical information of damage through time from data. Failure probability is calculated by Monte Carlo method, [36] which is a numerical

method based on computational simulations widely used in composites reliability as reference or exact method [37, 38]. The boostrap method is used to overcome the statistical uncertainty from the sampling method.

As a result of this work, distributions of failure probability over lifetime are obtained and compared with those obtained directly from empirical data. In order to compare the efficiency of the stochastic damage model to derive the long term failure probability, the model is compared with benchmark data coming from from probability density functions of damage identified by the test of Kolmogorov-Smirnov.

3.2 Reliability Formulation

The essence of the reliability problem is the probability integral:

$$P_f = \int_{\mathbf{X}|g(\mathbf{X})\leq 0} f_X(\mathbf{X})\mathrm{d}(\mathbf{X}) \tag{3.1}$$

where $f_X(\mathbf{X})$ is the probability density function of the vector of random variables $\mathbf{X}$ that represent uncertain quantities that influences the state of the structure. $g(\mathbf{X}) \leq 0$ denotes a subset of the outcome space where failure occurs.

For mathematical analysis, it is necessary to describe the failure domain $g(\mathbf{X}) \leq 0$ in an analytical form, which is widely named as limit state function (LSF). The next section 3.2.1 is dedicated to expose the LSF of Tsai and Wu [39], widely used for failure analysis and reliability in composites. A Monte Carlo method to solve numerically the integral (3.1) will be exposed in the section 3.2.2.

Both cited topics about Equation 3.1, together with the discussion about what to consider as random variables, take almost all the literature discussion of composite reliability.

3.2.1 Limit State Function

There are several failure criteria for unidirectional composite laminates such as maximum stress, maximum strain, Tsai-Hill, Hoff-man, Tsai-Wu, etc [40–43]. Under such variability of failure criteria, in certain research works of reliability for composites materials [37, 44–47], several possibles LSF are probed and compared to experimental or reference reliability data when available. However, the Tsai-Wu [39] quadratic criteria is widely used in reliability by its physical plausibility and its mature knowledge achieved from several decades. Hence, without lack of generality, this criterion is used herein.

The Tsai-Wu failure criterion is used to determine the failure of orthotropic materials and takes into account the interactions between different stress and strength components. It is formulated as:

$$F_x\sigma_x + F_y\sigma_y + F_{xx}\sigma_x^2 + F_{yy}\sigma_y^2 + F_{ss}\sigma_{xy}^2 + 2F_{xy}\sigma_x\sigma_y = 1 \tag{3.2}$$

where

$$F_x = \frac{1}{R_x} - \frac{1}{R'_x} \tag{3.3a}$$

$$F_y = \frac{1}{R_y} - \frac{1}{R'_y} \tag{3.3b}$$

$$F_{xx} = \frac{1}{R_x R'_x} \tag{3.3c}$$

$$F_{yy} = \frac{1}{R_y R'_y} \tag{3.3d}$$

$$F_{ss} = \frac{1}{R_s^2} \tag{3.3e}$$

$$F_{xy} = -0.5\sqrt{F_{xx}F_{yy}} \tag{3.3f}$$

The subscripts x and y indicate longitudinal and transversal orientation respectively, while s means shear. R_x is the ultimate longitudinal tensile

strength, R'_x is the longitudinal ultimate compressive strength, R_y is the ultimate transverse tensile strength, R'_y is the ultimate transverse compressive strength and R_s is the in-plane shear strength.

A mathematical expression for unidirectional composite failure may be written as follows:

$$g(\mathbf{X}) = g(x_1, x_2, \ldots, x_n) \leqslant 0 \tag{3.4}$$

where $g(\mathbf{X})$ represents the safety margin and $\mathbf{X}$ is the n-dimensional vector of random variables $\mathbf{X} = \{\sigma_x, \sigma_y, \sigma_{xy}, R_x, R'_x, R_y, R'_y, R_s\}$. Substituting equation (3.2) into (3.4), the limit state function $g(\mathbf{X})$ at the critical point in the composite material, becomes:

$$g(\mathbf{X}) = 1 - (F_x\sigma_x + F_y\sigma_y + F_{xx}\sigma_x^2 + F_{yy}\sigma_y^2 + F_{ss}\sigma_{xy}^2 + 2F_{xy}\sigma_x\sigma_y) \tag{3.5}$$

3.2.2 Monte Carlo method

Given the random set $\mathbf{X}$ of random variables each one characterized by its marginal density function $f_{x_i}(x_i)$, the failure probability defined in Equation (3.1) can be written as:

$$\begin{aligned} P_f = \int_{\mathbf{X}|g(\mathbf{X})\leq 0} f_X(\mathbf{X})\mathrm{d}(\mathbf{X}) = \\ \int_{\mathbf{X}} I\left[g(\mathbf{X})\right] f_X(\mathbf{X})\mathrm{d}(\mathbf{X}) \end{aligned} \tag{3.6}$$

where $f_X(\mathbf{X})$ is the joint probability distribution function for the random variables, and $I\left[g(\mathbf{X})\right]$ is an indicative function defined by:

$$I[g(\mathbf{X})] = \begin{cases} 1 & \text{if } g(\mathbf{X}) \leq 0 \\ 0 & \text{if } g(\mathbf{X}) > 0 \end{cases} \tag{3.7}$$

Unfortunately, the definition of random variables for stresses and strengths, and the Tsai-Wu criterion lead to a very complex expression to compute the probability of failure analytically. An effective way to compute this probability of failure is by a Monte Carlo simulation.

The principle of the Monte Carlo method is to sample each uncertain parameter x_i by independent samples according to its density function $f_{x_i}(x_i)$. In each iteration, a value is generated for each design variable which is then tested in the failure criterion $g(\mathbf{X})$. The failure probability will then be the number of failure simulation respect to the total number of simulations.

Given that Equation (3.6) represents the expected value of the indicative function (3.7), then an estimate of the failure probability can be written as:

$$P_f \cong \frac{1}{n_s} \sum_{j=1}^{n_s} I[g(\mathbf{X}_j)] \tag{3.8}$$

where n_s is the number of simulations, $\mathbf{X}_j$ the vector of random variables of the j$^{\text{th}}$ sample and $\sum_{j=1}^{n_s} I[g(\mathbf{X}_j)]$ represents the sum of the number of simulation in the failure domain (n_f). Equation (3.8) may also be written as:

$$P_f = \frac{n_f}{n_s} \tag{3.9}$$

In MCM a high computational cost is expected for small failure probabilities, given that the total number of required simulations increases drastically as is evidenced in Equation (3.15). Hence, attention has been focused on the develop of more efficient simulation methods, among them the most popular one the *importance sampling method* [48]. In this paper, this drawback has been solved alternatively by a vectorized computation [49].

3.3 Reliability under damage conditions

The random accumulation of fatigue damage over time leads to a redistribution of stresses and also to a strength decrease, which affects the failure function $g(\mathbf{X})$.

To use this information in a reliability model, the damage evolution must be accounted into the failure function. To this end, a recent coupled

approach of residual stiffness and strength to simulate the progressive failure by a modified Tsai-Wu (or other) failure criterion, has been adopted [31].

This approach is based on the concept of the effective stress, $\tilde{\sigma}$ [50], as the stress calculated over the effective area of the damaged cross-section A, that resists the force F:

$$\tilde{\sigma} = \frac{F}{A(1-D)} = \frac{\sigma}{1-D} \tag{3.10}$$

The stress and strain are related by the commonly used equation in continuum damage mechanics of Lemaitre and Chaboche [51], Krajcinovic [52]:

$$\epsilon = \frac{\tilde{\sigma}}{E_0} = \frac{\sigma}{E_0(1-D)} \tag{3.11}$$

where ϵ is the nominal strain, E_0 is the undamaged Young's modulus and D is a macroscopic measure for the fatigue damage, defined as $D = 1 - {}^{E}/_{E_0}$ with E the actual-residual stiffness. Then, when $E = 0 \Rightarrow D = 1$.

In this paper, a generalization of the damage variable is adopted to consider failure not only when stiffness equals zero but also when it reaches a target stiffness loss value, as follows:

$$D = \frac{E_0 - E}{(1-\xi)E_0} \tag{3.12}$$

with ξ the target percentage loss of stiffness.

Following this approach, a modified Tsai-Wu failure criterion can be achieve by considering the effective stress into the quadratic failure function. So, the limit state function for reliability evaluation in the uniaxial case, results as follows:

$$g(D) = 1 - \left(\frac{\sigma}{1 - \underbrace{D}_{\text{rv}}}\right)^2 \left(\frac{1}{R_x R'_x}\right) + \left(\frac{\sigma}{1 - \underbrace{D}_{\text{rv}}}\right) \left(\frac{1}{R_x} - \frac{1}{R'_x}\right) \tag{3.13}$$

with R_x and R'_x as indicated previously in Equations (3.3).

The only random variable considered in this framework is the macroscopic damage D, as the factor that induces stochastic changes in both stress and strengths values. Hence, a stochastic model for the evolution of D over time together with the adoption of an appropriate failure criterion $g(D)$ are needed to formulate mathematically the probability integral for the failure probability evaluation, as:

$$P_f = \int_D I\left[g(D)\right] f_D(D)\mathrm{d}D = \int_{D/g(D)\leq 0} f_D(D)\mathrm{d}D \tag{3.14}$$

where$f_D(D)$ is the probability density function derived from the stochastic Markov model developed in Chapter 2 (Equation 2.9).

By the Monte Carlo method, the solution of Equation (3.14) can be obtained as:

$$P_f \cong \frac{1}{n_s}\sum_{j=1}^{n_s} I[g(D_j)] = \frac{n_f}{n_s} \tag{3.15}$$

where n_s is the number of simulations, D_j is the random damage value of the j^{th} sample and $\sum_{j=1}^{n_s} I[g(D_j)]$ represents the sum of the number of simulation in the failure domain (n_f).

Due to the stochastic information proceeding from Equation (2.9) are of the non-parametric type, a population of samples $\mathfrak{D} \subseteq D$ must be derived from the model predicted density functions of damage by the Rejection Method, Metropolis Hasting, Gibss or others [53]. In this paper, the Rejection Method with a sample size of 5000 has been used.

The statistical uncertainty associate to sampling D by rejection, derives an error of evaluation for the failure probability once this sample has been utilized as simulation in MCM. For conferring confidence, the calculus was performed using the Bootstrap cross-validation technique [54], which are Monte Carlo simulations that treat the original sample D as the pseudo-population or as an estimate of the population by sampling B times with

replacement over $\mathfrak{D}$ obtaining the *bootstrap replicates* $\mathfrak{D}^b$, as shown in Equation (3.16)

$$\hat{P}_f^{*b} = P_f(\mathfrak{D}^b) = \frac{1}{n_s}\sum_{j=1}^{n_s} I[g(\mathfrak{D}_j^b)]; \qquad b = 1 \cdots B \tag{3.16}$$

In this work, $B = 100$ bootstraps were needed to controlled the bias in the failure probability.

3.4 Numerical example

The proposed framework is illustrated in an example considering the previously mentioned stochastic damage data from the work of Wei et al. [11]. Details regarding the experimental set-up, measurements, etc were reported in this work and hence, they are not repeated here. Each specimen is subjected to a constant amplitude $T - T$ fatigue loading ($R = 0.1, f = 5\,Hz, \sigma_{max} = 0.5\sigma_u$) and twenty five measurements of longitudinal stiffness are registered as fatigue response within a not-regularly spaced time interval. A graphical representation of the damage samples coming from this data set was reported in Figure 2.2, Chapter 2

Equation (3.16) is applied to obtain an estimation of the failure probability $\hat{P}_{f_{t_i}}^{*b}$ from empirical damage states D_{n_e}. The same procedure is reproduced with the model predicted probability functions of damage at times not covered by data. The three Markov model parameterizations proposed in Chapter 2 are introduced within Equation (3.16) here. Additionally, each calculation is repeated to take into account the three definitions of residual (Equations 2.13 to 2.15) proposed in Chapter 2, with which damage models have been trained.

In order to compare the efficiency of the stochastic damage model proposed herein and to derive a benchmark for the failure probability evolution, the method is also repeated with new probability density functions of dam-

age identified by the test of Kolmogorov-Smirnov with a confident level of 95%. In this last case, it was not necessary to use the bootstrap technique, given that a parametric definition of the distribution of damage is available, as it is provided by the test. Finally, in those calculations employing the bootstrap technique, the maximum likelihood value of each $\hat{P}^{*b}_{f_{t_i}}$ estimated, is selected as the most representative value of failure probability at each time.

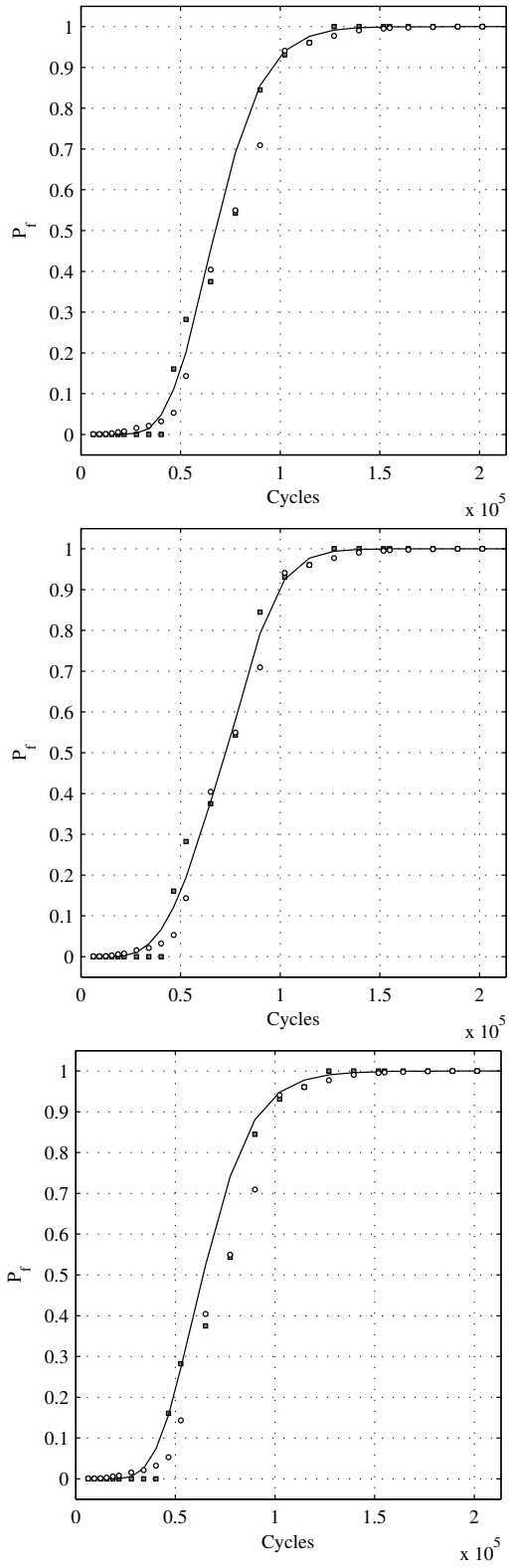

Figure 3.1: Failure probability predicted by models trained with residual 1. From top to bottom: Model A, model B and model C, respectively. Solid line: Model predicted. Square marks: Predicted from empirical damage. Circle mark: Predicted by K-S test.

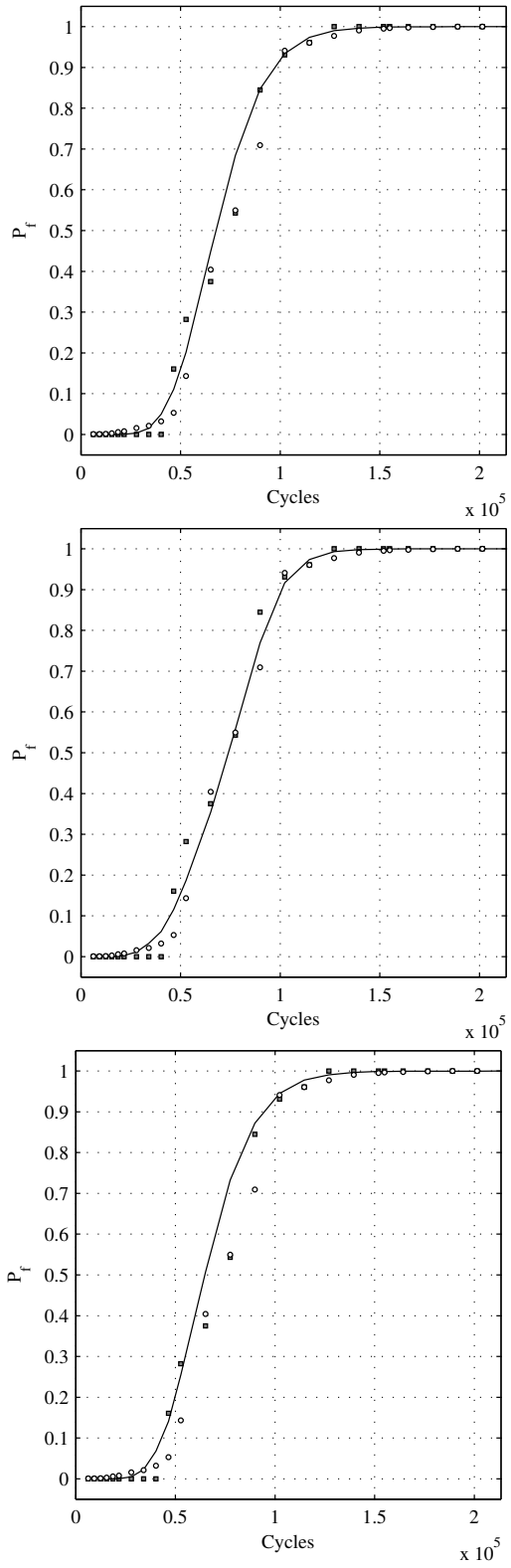

Figure 3.2: Failure probability predicted by models trained with residual 2. From top to bottom: Model A, model B and model C, respectively. Solid line: Model predicted. Square marks: Predicted from empirical damage. Circle mark: Predicted by K-S test.

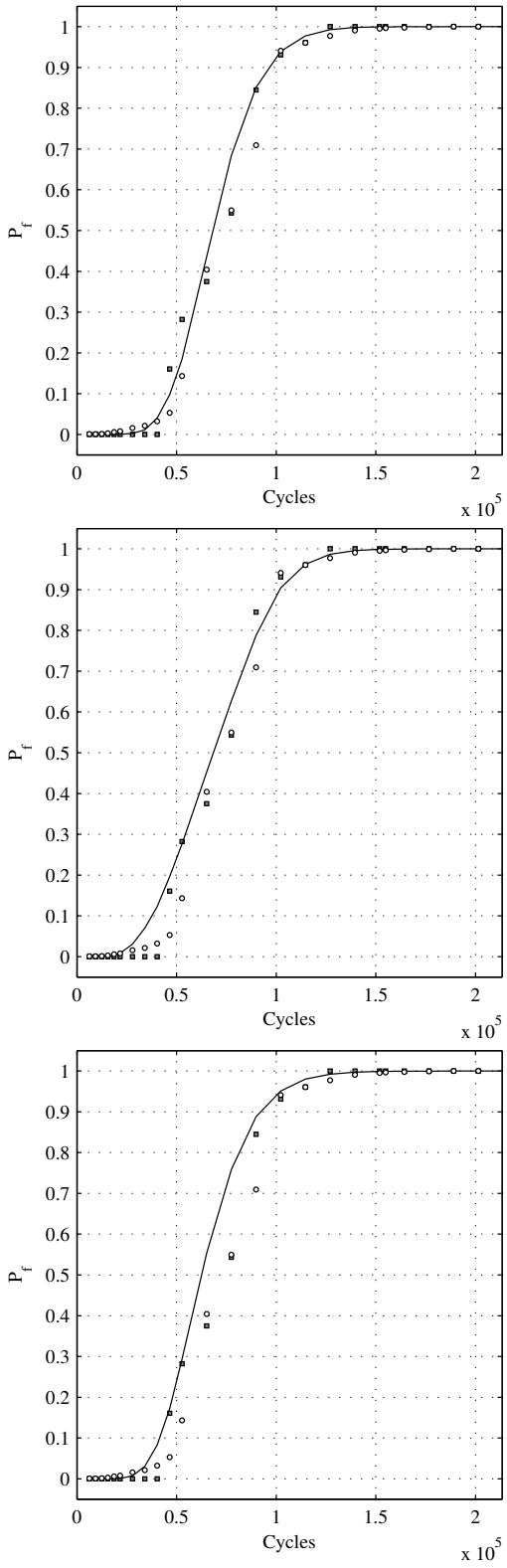

Figure 3.3: Failure probability predicted by models trained with residual 3. From top to bottom: Model A, model B and model C, respectively. Solid line: Model predicted. Square marks: Predicted from empirical damage. Circle mark: Predicted by K-S test.

3.5 Conclusions

The results show general good agreement between model and experimental predicted failure probability. However it is appreciable the use of different parameterizations for the Markov damage models in the accuracy of the failure probability prediction. The non stationary damage model B, which showed the best fitting accuracies in Chapter 2, fits also better the failure probability, as expected. This damage model also showed a considerable tendency to overfit damage data, which went down when it was trained with the entropic residual. Thus, it is also reasonable to expect it to predict worse new experimental data, coming for example from a model updating scheme.

Regarding the residual election it seems not to have a decisive importance in the fitting accuracy of the failure probability, providing almost the same results. This can be attributed to the fact that the inherent error of the sampling method covers the differences between using one or other residual for the same model architecture.

Finally, it is also important to observe that the proposed framework is general in nature and it is extensible to a broader class of materials, given their failure criteria and a stochastic macroscopic damage model. In composite materials, other failure criteria different than Tsai-Wu can be used and different material variables, such us compliance, matrix cracking density or delamination area can be established as a suitable measure of macroscopic damage.

Appendix A

Monotone piecewise cubic interpolation

Let the mesh $\{\alpha_i\}_i^n$ be a partition of the unitary space $\mathcal{X} \in [0,1]$ with $\alpha_1 < \alpha_2 \cdots < \alpha_n$, and let $\{\beta i\}$ be the corresponding data points in the transformed unitary space $\mathcal{Y} \in [0,1]$ such that $\beta_i = \beta_i(\alpha_i)$. The mesh spacing is $\Delta\alpha_{i+1} = \alpha_{i+1} - \alpha_i$ and the slope between two consecutive data points is $S_{i+1} = \frac{\Delta\beta_{i+1}}{\Delta\alpha_{i+1}}$. The cubic Hermite interpolant is then defined as

$$y(x) = c_1 + (x - \alpha_i)c_2 + (x - \alpha_i)^2 c_3 + (x - \alpha_i)^3 c_4 \tag{A.1}$$

where

$$c_1 = \beta_i \tag{A.2a}$$

$$c_2 = \dot{\beta}_i \tag{A.2b}$$

$$c_3 = \frac{3S_{i+1} - \dot{\beta_{i+1}} - 2\dot{\beta}_i}{\Delta\alpha_{i+1}} \tag{A.2c}$$

$$c_4 = -\frac{2S_{i+1} - \dot{\beta_{i+1}} - \dot{\beta}_i}{\Delta\alpha_{i+1}^2} \tag{A.2d}$$

The global monotonicity of the interpolant function (A.1) depends on how $\{\dot{\beta}_i\}$ are calculated. There exits several methods in the literature to

approximate $\dot{\beta}_i$ preserving the piecewise continuity and monotonicity [16], among then the Fristch-Butland method [17] is adopted herein for its superiority and efficiency:

$$\dot{\beta}_i = \frac{3S^i_{min}S^i_{max}}{S^i_{max} + 2S^i_{min}} \tag{A.3}$$

where

$$S^i_{min} = min(S_{i-1}, S_{i+1}) \tag{A.4a}$$

$$S^i_{min} = max(S_{i-1}, S_{i+1}) \tag{A.4b}$$

Note that this method can not ensure the continuity of the second order derivative of the interpolant, which is acceptable for our purpose.

Appendix B

IP-MARKOV algorithm

This appendix provides a summary of the algorithm **IP-MARKOV** developed for the research work presented herein. The code consists of a collection of Matlab® files developed *ad hoc* in conjunction with other Matlab® functions, among them the GA function holds a central importance. The main structure of the algorithm is represented in Figure B.1 and a description of the main part of the code is provided below.

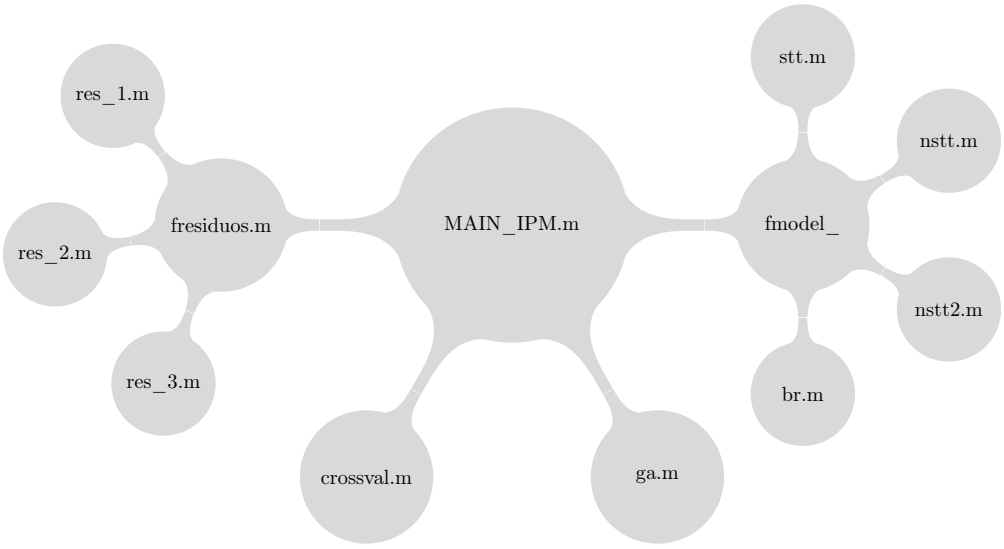

Figure B.1: IP-MARKOV algorithm scheme

```
1
2  %% MAIN_IPM.m
3
4  format compact;
5  clear all;
6  close all;
7  load newdata;  %load experimental data (damage)
8  load datatime; %load experimental data (time)
9  norm_compl=newdata';
10 datatime;
11
12 global PMFe
13 global T
14 global D_e
15 global mdl
16 global R
17
18 %**********************input parameters**********************
19 DC=500; %number of cycles in a DC
20 nx=2^7; %number of experimental points
21 tol=15; % percentual range (100*1/tol) tolerance of data
22 mdl='nstt'; %Model type
23 R=7; %Residual type
24 %************************************************************
25
26 abs_st=1; %absorbing state
27 norm_compl=absrvnt(norm_compl,abs_st);
28 norm_compl=treatdata(norm_compl);
29 dutytime=datatime/DC;
30 T_e=dutytime;
31 T=T_e;
32 [D_e,PMFe]=non_smoothing(norm_compl,T,nx);
33 D_e=treatdata(D_e);
34 PMFe=treatdata(PMFe);
```

```
35 [PMFe]=adjs_zero(PMFe);
36 mu_samples=mean(norm_compl,1);
37 desv_samples=sqrt(var(norm_compl,1,1));
38 median_samples=median(norm_compl,1);
39
40 %% GA model search
41
42 switch mdl
43
44     case 'stt'
45         InitialPopulation_Data=[24 0.267735429366 0.077890063227...
46         0.127544763315 0.998997224269];
47         OPTIONS = gaoptimset('PopulationSize',50,'Generations',65,...
48         'StallTimeLimit',Inf,'PlotFcns',{@gaplotbestf,...
49          @gaplotdistance},'TolFun',1e-300,...
50          'MutationFcn',@mutationadaptfeasible,...
51          'InitialPopulation' ,InitialPopulation_Data);
52
53         nval=5;
54         lb=[20,0.001,0.001,0.001,0.001];
55         ub=[50,0.999,0.999,0.999,0.999];
56         [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],[],[],[],...
57         lb,ub,[],OPTIONS);
58
59         save(['rslts/mat/RESULTS',num2str(R),'_DC',num2str(DC),'_',mdl])
60         M(1)=round(M(1));
61         fid = fopen(['rslts/RSLTS_R',num2str(R),'_DC',num2str(DC),'_',...
62         mdl,'.txt'], 'wt');
63
64     case 'nstt'
65         InitialPopulation_Data=[25 0.880693943710 0.087919369077...
66         0.076286122394 0.226390924072 0.357695025208];
67         OPTIONS = gaoptimset('PopulationSize',60,'Generations',50,...
68         'StallTimeLimit',...
```

```
69          Inf,'PlotFcns',{@gaplotbestf, @gaplotdistance},...
70          'TolFun',1e-300,'InitialPopulation' ,InitialPopulation_Data);
71
72          nval=6;
73          lb=[20,0.001,0.001,0.001,0.001,0.001];
74          ub=[60,0.999,0.999,0.999,0.999,0.999];
75          [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],[],[],[],...
76          lb,ub,[],OPTIONS);
77
78          save(['rslts/mat/RESULTS',num2str(R),'_DC',num2str(DC),'_',mdl])
79          M(1)=round(M(1));
80          fid = fopen(['rslts/RSLTS_R',num2str(R),'_DC',num2str(DC),...
81          '_',mdl,'.txt'], 'wt');
82
83      case 'nstt2'
84          InitialPopulation_Data=[28.3980974441175,0.916719210331823,...
85              0.505150560019371,0.407526844531409];
86          OPTIONS = gaoptimset('PopulationSize',50,'Generations',60,...
87          'StallTimeLimit',Inf,'PlotFcns',{@gaplotbestf,...
88           @gaplotdistance},'TolFun',1e-300,...
89           'InitialPopulation' ,InitialPopulation_Data);
90
91          nval=4;
92          lb=[20,0.001,0.001,0.001];
93          ub=[50,0.999,0.999,0.999];
94          [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],[],[],[],...
95          lb,ub,[],OPTIONS);
96
97          save(['rslts/mat/RESULTS',num2str(R),'_DC',num2str(DC),'_',mdl])
98          M(1)=round(M(1));
99          fid = fopen(['rslts/RSLTS_R',num2str(R),'_DC',num2str(DC),'_',...
100         mdl,'.txt'], 'wt');
101
102     case 'br'
```

```
103         InitialPopulation_Data=[30 8.015276954562];
104         OPTIONS = gaoptimset('PopulationSize',30,'Generations',40,...
105             'StallTimeLimit',Inf,'PlotFcns',{@gaplotbestf,...
106             @gaplotdistance},'TolFun',1e-300,'MutationFcn',...
107             @mutationadaptfeasible,'InitialPopulation',...
108             InitialPopulation_Data);
109
110         nval=2;
111         lb=[20,1];
112         ub=[50,15];
113         [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],[],[],[],...
114         lb,ub,[],OPTIONS);
115
116         save(['rslts/mat/RESULTS',num2str(R),'_DC',num2str(DC),'_',mdl])
117         M(1)=round(M(1));
118         fid = fopen(['rslts/RSLTS_R',num2str(R),'_DC',num2str(DC),...
119         '_',mdl,'.txt'], 'wt');
120 end
```

```
1  %fresiduos.m
2
3  %*****************************************************************
4  function objetive=fresiduos(V)
5  global PMFe
6  global T
7  global D_e
8  global R
9  global mdl
10
11 switch mdl
12
13     case 'stt'
14
```

```
15          b=round(V(1));
16          q1=abs(V(2));
17          qb_1=abs(V(3));
18          alfa1=abs(V(4));
19          beta1=abs(V(5));
20          [PMFd,D_d]=fmodel_stt(b,q1,qb_1,alfa1,beta1,T);
21
22      case 'nstt'
23
24          b=round(V(1));
25          p=abs(V(2));
26          alfa1=abs(V(3));
27          beta1=abs(V(4));
28          alfa2=abs(V(5));
29          beta2=abs(V(6));
30          [PMFd,D_d]=fmodel_nstt(b,p,alfa1,beta1,alfa2,beta2,T);
31
32      case 'nstt2'
33
34          b=round(V(1));
35          p=abs(V(2));
36          alfa1=abs(V(3));
37          beta1=abs(V(4));
38          [PMFd,D_d]=fmodel_nstt2(b,p,alfa1,beta1,T);
39
40          case 'br'
41
42          b=round(V(1));
43          r=abs(V(2));
44          [PMFd,D_d]=fmodel_br(b,r,T);
45
46  end
47
48  if numel(PMFd)==0
```

```
49      disp('JCHR_Warning: PMFd matrix is empty')
50      objetive=1000;
51  else
52      fctr=1;
53      obj_vector=[];
54      for i=2:size(PMFd,1)
55          obj_vector=[obj_vector;eval(['residual_R',num2str(R),...
56          '(fctr*D_e(i,:),fctr*D_d,PMFe(i,:),PMFd(i,:))'])];
57      end
58      if numel(obj_vector)==0;
59          disp('JCHR_Warning: Objetive vector is empty')
60          objetive=1000;
61      else
62          objetive=log(sqrt(sum(obj_vector.^2))+1e-20);
63      end
64  end
65  %*****************************************************************
```

```
1   %fmodel_stt.m
2
3   %*****************************************************************
4
5   function [PMFd,D_d,mu_d,desv_d]=fmodel_stt(b,q1,qb_1,alfa1,beta1,T)
6
7   b=round(b);
8   xd=[0 alfa1 1];
9   yd=[0 beta1 1];
10  q=q1+(qb_1-q1)*pchip(xd,yd,linspace(0,1,b-1));
11  p=1-q;
12  p=[p 1];
13  P1=diag(p);
14
15  %Initial probability distribution of damage
```

```
p0=zeros(1,b);
p0(1,1)=1;
for i=1:(b-1)
    P1(i,i+1)=q(1,i);
end
D_d=cat(2,0.01,1/b*((1:(b-1))-0.5),1);
PMFd=[];
mu_d=[];
desv_d=[];
for i=1:numel(T)
    PTM=binprod(P1,T(i));
    pt=p0*PTM;
    pt=cat(2,0,pt);
    med=sum(D_d.*pt);
    stdev=sqrt(sum(((D_d-med).^2).*pt));
    mu_d=[mu_d,med];
    desv_d=[desv_d,stdev];
    CDF_D=pt(1);
    for n=2:numel(pt)
        CDF_D=[CDF_D,CDF_D(n-1)+pt(n)];
    end
    PMFd=[PMFd;CDF_D];

end

%*****************************************************************
```

```
%fmodel_nstt.m

%*****************************************************************
function [PMFd,D_d,mu_d,desv_d]=fmodel_nstt(b,p,alfa1,beta1,alfa2,beta2,T)

b=round(b);
```

```
alfa2m=alfa1+alfa2*(1-alfa1);
beta2m=beta1+beta2*(1-beta1);
q=1-p;
p0=zeros(1,b);
p0(1,1)=1;
for j=1:(b-1)
    P1(j,j)=p;
    P1(j,j+1)=q;
end

P1(b,b)=1;
xx=0:0.001:1;
x=[0 alfa1 alfa2m 1];
y=[0 beta1 beta2m 1];
yy=pchip(x,y,xx);
PTMy=eye(size(P1));
PMFd=[];
X=max(T);
t_0=0;
x_time=t_0+xx*(X-t_0);
y_time=t_0+yy*(X-t_0);
real_rt=interp1(y_time,x_time,T);
D_d=cat(2,0.01,1/b*((1:(b-1))-0.5),1);
mu_d=[];
desv_d=[];
for i=1:numel(real_rt)
    if i==1
        n=floor(real_rt(i))-0;
    elseif real_rt(i-1)==0;
        n=floor(real_rt(i))-ceil(real_rt(i-1));
    else
        n=floor(real_rt(i))-ceil(real_rt(i-1))+1;
    end

```

```
    if n<0;

        disp('JCHR_error: fmodel, line 63')
        break
    else
        if n==0;
            Qy=eye(size(P1));
        elseif n==1
            Qy=P1;
        else
            Qy=binprod(P1,n);
        end
        PTMy=PTMy*Qy;
        pt=p0*PTMy;
        pt=cat(2,0,pt);
        med=sum(D_d.*pt);
        stdev=sqrt(sum(((D_d-med).^2).*pt));
        mu_d=[mu_d,med];
        desv_d=[desv_d,stdev];
        CDF_D=pt(1);
        for n=2:numel(pt)
            CDF_D=[CDF_D,CDF_D(n-1)+pt(n)];
        end
        PMFd=[PMFd;CDF_D];
    end
end

%*****************************************************************
```

```
%fmodel_nstt2.m

%*****************************************************************
function [PMFd,D_d,mu_d,desv_d]=fmodel_nstt2(b,p,alfa1,beta1,T)
```

```
5
6  b=round(b);
7  q=1-p;
8  p0=zeros(1,b);
9  p0(1,1)=1;
10 for j=1:(b-1)
11     P1(j,j)=p;
12     P1(j,j+1)=q;
13 end
14
15 P1(b,b)=1;
16 xx=0:0.001:1;
17 x=[0 alfa1 1];
18 y=[0 beta1 1];
19 yy=pchip(x,y,xx);
20 PTMy=eye(size(P1));
21 PMFd=[];
22 X=max(T);
23 t_0=0;
24 x_time=t_0+xx*(X-t_0);
25 y_time=t_0+yy*(X-t_0);
26 real_rt=interp1(y_time,x_time,T);
27 D_d=cat(2,0.01,1/b*((1:(b-1))-0.5),1);
28 mu_d=[];
29 desv_d=[];
30 for i=1:numel(real_rt)
31     if i==1
32         n=floor(real_rt(i))-0;
33     elseif real_rt(i-1)==0;
34         n=floor(real_rt(i))-ceil(real_rt(i-1));
35     else
36         n=floor(real_rt(i))-ceil(real_rt(i-1))+1;
37     end
38
```

```
    if n<0;

        disp('JCHR_error: fmodel, line 63')
        break
    else
        if n==0;
            Qy=eye(size(P1));
        elseif n==1
            Qy=P1;
        else
            Qy=binprod(P1,n);
        end
        PTMy=PTMy*Qy;
        pt=p0*PTMy;
        pt=cat(2,0,pt);
        med=sum(D_d.*pt);
        stdev=sqrt(sum(((D_d-med).^2).*pt));
        mu_d=[mu_d,med];
        desv_d=[desv_d,stdev];
        CDF_D=pt(1);
        for n=2:numel(pt)
            CDF_D=[CDF_D,CDF_D(n-1)+pt(n)];
        end
        PMFd=[PMFd;CDF_D];
    end
end

%*****************************************************************



%CROSSVAL_5.m
%*****************************************************************
function [CV_L2]=CROSSVAL_5(r,m,nsamples,nfolds)

```

```
load newdata;
load datatime;
norm_compl=newdata';
datatime;
global PMFe
global T
global D_e
global R
global mdl
nzro_compl=[];
for i=2:size(norm_compl,2)
    nzro_compl=cat(2,nzro_compl,norm_compl(:,i));
end
abs_st=1;
nzro_compl=absrvnt(nzro_compl,abs_st);
nzro_compl=treatdata(nzro_compl);
norm_compl=nzro_compl;
DC=500;
nx=2^7;
tol=15;
mdl=m;
R=r;
N=nsamples;
nf=nfolds;
dutytime=datatime/DC;
Tmax=floor(max(dutytime));
T_e=dutytime;
T=T_e(2:end);

%% EVALUATION SETS
pkv=1:N;
cnt=0;
set1=[];set2=[];set3=[];set4=[];set5=[];

```

```
for i=1:nf
    r1=ceil(rand*(N-cnt)); set1=[set1,pkv(r1)];...
    ordset1=ordvector(set1);
    pkv=takeIND(pkv,r1);cnt=cnt+1;
    r2=ceil(rand*(N-cnt)); set2=[set2,pkv(r2)];...
    ordset2=ordvector(set2);
    pkv=takeIND(pkv,r2);cnt=cnt+1;
    r3=ceil(rand*(N-cnt)); set3=[set3,pkv(r3)];...
    ordset3=ordvector(set3);
    pkv=takeIND(pkv,r3); cnt=cnt+1;
    r4=ceil(rand*(N-cnt)); set4=[set4,pkv(r4)];...
    ordset4=ordvector(set4);
    pkv=takeIND(pkv,r4); cnt=cnt+1;
    r5=ceil(rand*(N-cnt)); set5=[set5,pkv(r5)];...
    ordset5=ordvector(set5);
    pkv=takeIND(pkv,r5); cnt=cnt+1;
end

evtst1=[]; Tev1=[];
for j=1:numel(ordset1)
    evtst1=cat(2,evtst1,nzro_compl(:,ordset1(j)));
    Tev1=cat(1,Tev1,T(ordset1(j)));
end
evtst2=[]; Tev2=[];
for j=1:numel(ordset2)
    evtst2=cat(2,evtst2,nzro_compl(:,ordset2(j)));
    Tev2=cat(1,Tev2,T(ordset2(j)));
end
evtst3=[]; Tev3=[];
for j=1:numel(ordset3)
    evtst3=cat(2,evtst3,nzro_compl(:,ordset3(j)));
    Tev3=cat(1,Tev3,T(ordset3(j)));
end
evtst4=[]; Tev4=[];
```

```
for j=1:numel(ordset4)
    evtst4=cat(2,evtst4,nzro_compl(:,ordset4(j)));
    Tev4=cat(1,Tev4,T(ordset4(j)));
end
evtst5=[]; Tev5=[];
for j=1:numel(ordset5)
    evtst5=cat(2,evtst5,nzro_compl(:,ordset5(j)));
    Tev5=cat(1,Tev5,T(ordset5(j)));
end
v1=(1:N);
for i=1:numel(ordset1)
    n=ordset1(i); v1=takeVAL(v1,n);
end

trst1=[]; T1=[];
for i=1:numel(v1)
    trst1=cat(2,trst1,nzro_compl(:,v1(i)));
    T1=cat(1,T1,T(v1(i)));
end
v2=(1:N);
for i=1:numel(ordset2)
    n=ordset2(i); v2=takeVAL(v2,n);
end
trst2=[]; T2=[];
for i=1:numel(v1)
    trst2=cat(2,trst2,nzro_compl(:,v2(i)));
    T2=cat(1,T2,T(v2(i)));
end
v3=(1:N);
for i=1:numel(ordset3)

    n=ordset3(i); v3=takeVAL(v3,n);
end
trst3=[]; T3=[];
```

```
for i=1:numel(v1)
    trst3=cat(2,trst3,nzro_compl(:,v3(i)));
    T3=cat(1,T3,T(v3(i)));
end
v4=(1:N);
for i=1:numel(ordset4)

    n=ordset4(i); v4=takeVAL(v4,n);
end
trst4=[]; T4=[];
for i=1:numel(v4)
    trst4=cat(2,trst4,nzro_compl(:,v4(i)));
    T4=cat(1,T4,T(v4(i)));
end
v5=(1:N);
for i=1:numel(ordset5)

    n=ordset5(i); v5=takeVAL(v5,n);
end
trst5=[]; T5=[];
for i=1:numel(v5)
    trst5=cat(2,trst5,nzro_compl(:,v5(i)));
    T5=cat(1,T5,T(v5(i)));
end
%% K-FOLD CROSS VALIDATION
M_mtrx=[];
FVAL_mtrx=[];
for i=1:nf
    train_fold=eval(['trst',num2str(i)]);
    Ttr=eval(['T',num2str(i)]);
    T=Ttr;
    [D_e,PMFe]=non_smoothing(train_fold,Ttr,nx);
    D_e=treatdata(D_e);
    PMFe=treatdata(PMFe);
```

```
141     [PMFe]=adjs_zero(PMFe);
142
143     switch mdl
144
145         case 'stt'
146
147             InitialPopulation_Data=[25 0.307872879831 0.082244313778,...
148             0.102992817449 0.998999986567];
149             OPTIONS = gaoptimset('PopulationSize',30,'Generations',60,...
150                 'StallTimeLimit',Inf,'PlotFcns',...
151                 {@gaplotbestf, @gaplotdistance},'TolFun',1e-300,...
152                 'MutationFcn',@mutationadaptfeasible,...
153                 'InitialPopulation',InitialPopulation_Data);
154
155             nval=5;
156             lb=[20,0.001,0.001,0.001,0.001];
157             ub=[40,0.999,0.999,0.999,0.999];
158
159         case 'nstt'
160
161         InitialPopulation_Data=[25 0.880327732772 0.087943270540,...
162         0.076286122394 0.226661946392 0.357993584874];
163         OPTIONS = gaoptimset('PopulationSize',40,'Generations',...
164         60,'StallTimeLimit',Inf,'PlotFcns',{@gaplotbestf,...
165         @gaplotdistance},'TolFun',1e-300,...
166         'InitialPopulation' ,InitialPopulation_Data);
167
168             nval=6;
169             lb=[20,0.001,0.001,0.001,0.001,0.001];
170             ub=[50,0.999,0.999,0.999,0.999,0.999];
171
172          case 'nstt2'
173
174
```

```
175         InitialPopulation_Data=[21 0.932380641799 0.956837454518,...
176         0.944203728701];
177         OPTIONS = gaoptimset('PopulationSize',40,'Generations',40,...
178         'StallTimeLimit',Inf,'PlotFcns',{@gaplotbestf,...
179         @gaplotdistance},'TolFun',1e-300,'InitialPopulation',...
180         InitialPopulation_Data);
181
182         nval=4;
183         lb=[20,0.001,0.001,0.001];
184         ub=[40,0.999,0.999,0.999];
185
186
187             case 'br'
188
189         InitialPopulation_Data=[23 10.735400407087];
190         OPTIONS = gaoptimset('PopulationSize',20,'Generations',30,...
191         'StallTimeLimit',Inf,'PlotFcns',{@gaplotbestf,...
192          @gaplotdistance},'TolFun',1e-300,'MutationFcn',...
193          @mutationadaptfeasible,...
194         'InitialPopulation' ,InitialPopulation_Data);
195
196         nval=2;
197         lb=[20,1];
198         ub=[40,20];
199
200     end
201
202     [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],[],[],[],lb,ub,[],OPTIONS);
203     M(1)=round(M(1));
204     M_mtrx=[M_mtrx;M];
205     FVAL_mtrx=[FVAL_mtrx,FVAL];
206
207     hold off
208 end
```

```
209         save (['rslts/mat/CROSSVAL/RESULTS',num2str(R),'_DC',...
210         num2str(DC),'_',mdl])
211
212 %% EVALUATING MODEL
213
214 L2_norm=[];
215
216 for i=1:nf
217     eval_fold=eval(['evtst',num2str(i)]);
218     Tev=eval(['Tev',num2str(i)]);
219     [D_ev,PMFev]=non_smoothing(eval_fold,Tev,nx);
220     D_ev=treatdata(D_ev);
221     PMFev=treatdata(PMFev);
222     [PMFev]=adjs_zero(PMFev);
223
224     switch mdl
225         case 'stt'
226
227             mu_samples=mean(eval_fold,1);
228             desv_samples=sqrt(var(eval_fold,1,1));
229             b=M_mtrx(i,1);
230             q1=M_mtrx(i,2);
231             qb_1=M_mtrx(i,3);
232             alfa1=M_mtrx(i,4);
233             beta1=M_mtrx(i,5);
234             [PMFdev,D_dev,mu_dev,desv_dev]=fmodel_stt(b,q1,qb_1,...
235             alfa1,beta1,Tev);
236
237         case 'nstt'
238
239             mu_samples=mean(eval_fold,1);
240             desv_samples=sqrt(var(eval_fold,1,1));
241             b=M_mtrx(i,1);
242             p=M_mtrx(i,2);
```

```
243             alfa1=M_mtrx(i,3);
244             beta1=M_mtrx(i,4);
245             alfa2=M_mtrx(i,5);
246             beta2=M_mtrx(i,6);
247             [PMFdev,D_dev,mu_dev,desv_dev]=fmodel_nstt(b,p,alfa1,...
248             beta1,alfa2,beta2,Tev);
249
250         case 'nstt2'
251
252             mu_samples=mean(eval_fold,1);
253             desv_samples=sqrt(var(eval_fold,1,1));
254             b=M_mtrx(i,1);
255             p=M_mtrx(i,2);
256             alfa1=M_mtrx(i,3);
257             beta1=M_mtrx(i,4);
258             [PMFdev,D_dev,mu_dev,desv_dev]=fmodel_nstt2(b,p,alfa1,...
259             beta1,Tev);
260
261          case 'br'
262
263             mu_samples=mean(eval_fold,1);
264             desv_samples=sqrt(var(eval_fold,1,1));
265             b=M_mtrx(i,1);
266             r=M_mtrx(i,2);
267             [PMFdev,D_dev,mu_dev,desv_dev]=fmodel_br(b,r,Tev);
268
269     end
270     np=numel(Tev);
271     L2_norm=[L2_norm;sqrt(sum((mu_samples-mu_dev).^2)+...
272     sum((desv_samples-desv_dev).^2))/np];
273
274     Rs=[];
275     for j=1:numel(Tev)
276         Rs=[Rs;eval(['residual_R',...
```

```
277         num2str(R),'(D_ev(j,:),D_dev,PMFev(j,:),PMFdev(j,:))'])];
278     end
279     R_norm=[R_norm;sum(Rs)/np];
280 end
281 CV_L2=mean(L2_norm);
282 %*****************************************************************
```

```
1  %********************AUXILIAR FUNCTIONS************************
2
3  function [PTM]=binprod(P,X)
4  %(computes the exact multiplication of sparse matrices
5  %raised to large exponents)
6
7  if X<0;disp('JCHR_Warning: negative time in binprod!!.
8  Converted to positive');
9  end
10
11 X=abs(round(X));
12
13 if X==1
14     PTM=P;
15 elseif X==0
16     PTM=eye(size(P));
17 else
18
19 PTM=eye(size(P)); %initialize
20 ex=0; %initialize
21
22     while X-ex>1
23
24         n=floor(log2(X-ex));
25         M=P;
26         for i=1:n
```

```
27              M=M*M;
28          end
29          PTM=PTM*M;
30          ex=ex+2^(n); %exponent of PTM in each iteration
31
32          if ex == X;
33              break
34              PTM;
35          elseif X-ex==1
36              PTM=PTM*P;
37              break
38          end
39
40      end
41  end
```

```
1   function [D_e,PMFe]=non_smoothing(norm_compl,T,nx)
2   %(computes the raw empirical CDF of damage)
3
4   D_e=[];
5   PMFe=[];
6   D_ac=[];
7
8   for n=1:numel(T) %using not-measured data is not allowed
9
10          D_ac=[D_ac,norm_compl(:,n)];
11
12      % empirical cdf of damage for t=ti
13
14      [stairs_ecdf,Dmg] = ecdf(D_ac(:,n));
15
16          for j=1:numel(Dmg)-1
17          if Dmg(j)==Dmg(j+1) || Dmg(j+1)≤Dmg(j)
```

```
            Dmg(j+1)=Dmg(j)+1e-30;
        end
        end

    PMFe=[PMFe;linspace(0,1,nx)];

    D_e=[D_e;interp1(stairs_ecdf,Dmg,linspace(0,1,nx))];

end
```

```
function [PMFe]=adjs_zero(PMFe)
%(converts in absolute zeros the near-zero values...
%of the first column of the experimental matrix PMFe,...
%avoiding computational errors)

for i=1:size(PMFe,1)
    if PMFe(i,1)>0 && PMFe(i,1)<1
        PMFe(i,1)=0;
    end
end
```

```

function M=absrvnt(M,abs_st)
%(modify data to account for the absorbing state, D=1)

for i=1:size(M,1)
    for j=1:size(M,2)
        if M(i,j)>abs_st
            M(i,j)=abs_st;
        end
    end
end
```

```
% DC-FCOST SENSITIVITY ANALYSIS
%*************************************************************
format compact;
clear all;
load newdata;
load datatime;
norm_compl=newdata';
datatime;

global PMFe
global T
global D_e
global mdl
global R

nzro_compl=[]; %time zero is avoided (trivial case)
for i=2:size(norm_compl,2)
nzro_compl=cat(2,nzro_compl,norm_compl(:,i));
end
abs_st=1; %absorbing state
nzro_compl=absrvnt(nzro_compl,abs_st);

nzro_compl=treatdata(nzro_compl);
norm_compl=nzro_compl;

res=[6,5,7];
model={'stt','nstt','nstt2'};
FCOSTCOMP={};
RESCOMP={};

for k=1:numel(res)
    for l=1:numel(model)

%input parameters
```

```
35
36 DC=[100,300,500,1000,3000,7000,10000,20000];
37 nx=2^7;
38 tol=15;
39
40 fcostmatrix=[];
41 results=[];
42
43 for w=1:numel(DC)
44
45    results=[];
46    Nrep=5;
47     for x=1:Nrep
48
49 dutytime=datatime/DC(w); %display time data
50 Tmax=floor(max(dutytime));
51 T_e=dutytime; %experimental time
52
53 T=T_e(2:end); %vector of time where model has to be evaluated.
54 %[D_e,PMFe,bndwth,dens]=smoothing(norm_compl,T,nx,tol);
55 [D_e,PMFe]=non_smoothing(norm_compl,T,nx);
56
57 D_e=treatdata(D_e);
58 PMFe=treatdata(PMFe);
59 [PMFe]=adjs_zero(PMFe);
60
61 mu_samples=mean(norm_compl,1);
62 desv_samples=sqrt(var(norm_compl,1,1));
63 median_samples=median(norm_compl,1);
64
65  %GA model search
66
67                 switch mdl
68
```

```
69                      case 'stt'
70
71                          OPTIONS = gaoptimset('PopulationSize',...
72                          60,'Generations',70,...
73                         'StallTimeLimit', Inf,'TolFun',...
74                          1e-30,'MutationFcn',@mutationadaptfeasible,...
75                          'PlotFcns',{@gaplotbestf});
76
77
78                          nval=5;
79                          lb=[20,0.001,0.001,0.001,0.001];
80                          ub=[40,0.999,0.999,0.999,0.999];
81
82                          [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],...
83                          [],[],[],lb,ub,[],OPTIONS);
84                          M(1)=round(M(1));
85                          results=[results;M(1),FVAL];
86
87
88                      case 'nstt'
89
90
91                          OPTIONS = gaoptimset('PopulationSize',...
92                          60,'Generations',55,...
93                         'StallTimeLimit', Inf,'TolFun',...
94                          1e-30,'MutationFcn',@mutationadaptfeasible,...
95                          'PlotFcns',{@gaplotbestf});
96
97                          nval=6;
98                          lb=[20,0.001,0.001,0.001,0.001,0.001];
99                          ub=[40,0.999,0.999,0.999,0.999,0.999];
100
101                         [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],...
102                         [],[],[],lb,ub,[],OPTIONS);
```

```
103                         M(1)=round(M(1));
104                         results=[results;M(1),FVAL];
105
106                     case 'nstt2'
107
108
109                         OPTIONS = gaoptimset('PopulationSize',...
110                         50,'Generations',50,...
111                        'StallTimeLimit', Inf,'TolFun',...
112                         1e-30,'MutationFcn',@mutationadaptfeasible,...
113                         'PlotFcns',{@gaplotbestf});
114
115                         nval=4;
116                         lb=[20,0.001,0.001,0.001];
117                         ub=[40,0.999,0.999,0.999];
118
119                         [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],...
120                         [],[],[],lb,ub,[],OPTIONS);
121                         M(1)=round(M(1));
122                         results=[results;M(1),FVAL];
123
124
125                     case 'br'
126
127                         OPTIONS = gaoptimset('PopulationSize',...
128                         30,'Generations',30,...
129                        'StallTimeLimit', Inf,'TolFun',...
130                         1e-30,'MutationFcn',@mutationadaptfeasible,...
131                         'PlotFcns',{@gaplotbestf});
132
133                         nval=2;
134                         lb=[20,1];
135                         ub=[40,15];
136
```

```
137                         [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,[],...
138                         [],[],[],lb,ub,[],OPTIONS);
139                         M(1)=round(M(1));
140                         results=[results;M(1),FVAL];
141
142                 end
143     end
144                 fcostmatrix=[fcostmatrix;DC(w),...
145                 ceil(mean(results(:,1))),...
146                 mean(results(:,2))];
147
148 end
149         FCOSTCOMP{k,l}=fcostmatrix
150         %RESCOMP{k,l}=results
151
152     end
153 end
154
155   %% PLOTING THE DC-FCOST CURVES
156
157         fig2=figure;
158         set(fig2,'PaperUnits','centimeters',...
159         'PaperPosition',[.5 .5     18  18]);
160    cont=1;
161   for i=1:size(FCOSTCOMP,1) %residuals
162       for j=1:size(FCOSTCOMP,2) %models
163          matrix=FCOSTCOMP{i,j};
164         subplot(size(FCOSTCOMP,1),...
165         size(FCOSTCOMP,2),cont)
166         semilogx((matrix(:,1)),matrix(:,3),'-o','color',[0 0 0])
167
168         axis square
169         %axis([0 max(T) 0 1.03*max(desv_d./mu_d)])
170         xlabel('$DC$','Interpreter','latex','FontSize',8)
```

```
171         ylabel('$\mathcal{F}_{L}$','Interpreter',...
172             'latex','FontSize',8)
173         hold on
174            Ax1=gca;
175            set(Ax1,'Xlim',[(min(matrix(:,1))),(max(matrix(:,1)))],...
176           'Ylim',[min(matrix(:,3)),max(matrix(:,3)) ],...
177           'YGrid','on','XGrid','on','FontName','Times New Roman',...
178           'FontSize',8)
179         cont=cont+1;
180
181       end
182   end
183
184   print('-depsc',['eps\DC_comparation.eps']);
```

```
1
2  % GA CONVERGENCE ANALYSIS
3  %*********************************************************
4
5  format compact;
6  clear all;
7  load newdata; norm_compl=newdata';
8  datatime;
9
10 global PMFe
11 global T
12 global D_e
13 global mdl
14 global R
15
16 %input parameters
17 DC=500; %number of cycles in a DC (≤500)
18 nx=2^7; %number of experimental points
```

```
19 tol=15; % percentual range (100*1/tol) tolerance of data
20
21 nzro_compl=[]; %time zero is avoided (trivial case)
22 for i=2:size(norm_compl,2)
23     nzro_compl=cat(2,nzro_compl,norm_compl(:,i));
24 end
25
26 abs_st=1; %absorbing state
27 nzro_compl=absrvnt(nzro_compl,abs_st);
28
29 nzro_compl=treatdata(nzro_compl);
30 norm_compl=nzro_compl;
31
32 dutytime=datatime/DC; %display time data
33 Tmax=floor(max(dutytime));
34 T_e=dutytime; %experimental time
35
36 T=T_e(2:end); %vector of time where model has to be evaluated.
37 %[D_e,PMFe,bndwth,dens]=smoothing(norm_compl,T,nx,tol);
38 [D_e,PMFe]=non_smoothing(norm_compl,T,nx);
39
40 D_e=treatdata(D_e);
41 PMFe=treatdata(PMFe);
42 [PMFe]=adjs_zero(PMFe);
43
44 mu_samples=mean(norm_compl,1);
45 desv_samples=sqrt(var(norm_compl,1,1));
46 median_samples=median(norm_compl,1);
47
48 % GA model search
49
50 res=[6,5,7];
51 model={'stt';'nstt';'nstt2'};
52 FCOSTCOMP={};
```

```
nrep=1;
for k=1:numel(res)
    for l=1:numel(model)

        pop=[10,20,30,50,50];
        gen=[10,20,30,50,60];

        fcostmatrix=[];
        for i=1:numel(gen)
            for j=1:numel(pop)

                fvalrepmat=[];
                for b=1:nrep

                switch mdl

                    case 'stt'

                        OPTIONS = gaoptimset('PopulationSize',...
                        pop(j),'Generations',gen(i),...
                       'StallTimeLimit', Inf,'TolFun',...
                        1e-30,'MutationFcn',@mutationadaptfeasible,...
                        'PlotFcns',{@gaplotbestf});

                        nval=5;
                        lb=[20,0.001,0.001,0.001,0.001];
                        ub=[40,0.999,0.999,0.999,0.999];

                        [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,...
                        [],[],[],[],lb,ub,[],OPTIONS);


                    case 'nstt'

```

```
87                          OPTIONS = gaoptimset('PopulationSize',...
88                          pop(j),'Generations',gen(i),...
89                         'StallTimeLimit', Inf,'TolFun',...
90                          1e-30,'MutationFcn',@mutationadaptfeasible,...
91                          'PlotFcns',{@gaplotbestf});
92
93                          nval=6;
94                          lb=[20,0.001,0.001,0.001,0.001,0.001];
95                          ub=[50,0.999,0.999,0.999,0.999,0.999];
96
97                          [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,...
98                          [],[],[],[],lb,ub,[],OPTIONS);
99
100                     case 'nstt2'
101
102                         OPTIONS = gaoptimset('PopulationSize',...
103                         pop(j),'Generations',gen(i),...
104                        'StallTimeLimit', Inf,'TolFun',...
105                         1e-30,'MutationFcn',@mutationadaptfeasible,...
106                         'PlotFcns',{@gaplotbestf});
107
108                         nval=4;
109                         lb=[20,0.001,0.001,0.001];
110                         ub=[40,0.999,0.999,0.999];
111
112                         [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,...
113                         [],[],[],[],lb,ub,[],OPTIONS);
114
115                     case 'br'
116
117                         OPTIONS = gaoptimset('PopulationSize',...
118                         pop(j),'Generations',gen(i),...
119                        'StallTimeLimit', Inf,'TolFun',...
120                         1e-30,'MutationFcn',@mutationadaptfeasible,...
```

```
121                         'PlotFcns',{@gaplotbestf});
122
123                         nval=2;
124                         lb=[15,1];
125                         ub=[30,15];
126
127                         [M,FVAL,EXITFLAG]=ga(@fresiduos,nval,...
128                         [],[],[],[],lb,ub,[],OPTIONS);
129
130                 end
131
132                 fvalrepmat=[fvalrepmat;FVAL]
133
134                 end
135                 FVAL=mean(fvalrepmat);
136                 fcostmatrix=[fcostmatrix;gen(i),pop(j),FVAL]
137
138             end
139
140         end
141
142      FCOSTCOMP71{k,l}=fcostmatrix;
143     end
144
145 end
146
147  %Plot contour plot
148
149        load FCOSTCOMP
150          matrix=FCOSTCOMP{1,2};
151
152          fig2=figure;
153          set(fig2,'PaperUnits','inches','PaperPosition',[.8 1 5 5]);
154
```

```
155         xlin=linspace(gen(1),gen(end),100);
156         ylin=linspace(pop(1),pop(end),100);
157         [X,Y]=meshgrid(xlin,ylin);
158         Z=griddata(matrix(:,1),matrix(:,2),matrix(:,3),X,Y,'v4');
159
160         [C,h]=contour(X,Y,Z,10);axis tight; colormap ([.5 .5 .5])
161         text_handle=clabel(C,h,'FontSize',8,'Interpreter','latex');
162         Ax3=gca;
163         set(Ax3,'Xlim',[gen(1),gen(end)],'Ylim',[pop(1),...
164             pop(end)], 'YGrid','off','XGrid','off',...
165             'FontName','latex')
166          xlabel('Generations')
167          ylabel('Population')
168          save GACOST FCOSTCOMP
169          print('-depsc',['eps\GA_comparation.eps']);
```

List of Figures

2.1 Inverse procedure 13

2.2 Experimental samples of damage as a stiffness reduction over time. The scattering increase with time 17

2.3 Influence of the DC election over the cost functional. By columns from left to right: Model A, model B, model C, respectively. By rows from top to bottom: Residual 1, residual 2, residual 3, respectively. Clearly there exists an upper accuracy limit for DC. 19

2.4 GA convergence. By columns from left to right: Model A, model B, model C, respectively. By rows from top to bottom: Residual 1, residual 2, residual 3, respectively. Note that models trained with entropic residual provides smoother GA convergences. 20

2.5 Model prediction of the complete stochastic process. Model A trained with residual 1. 22

2.6 Model prediction of the complete stochastic process. Model B trained with residual 1. 23

2.7 Model prediction of the complete stochastic process. Model C trained with residual 1. 24

2.8 Model prediction of the complete stochastic process. Model A trained with residual 2. 25

2.9 Model prediction of the complete stochastic process. Model B trained with residual 2. 26

2.10 Model prediction of the complete stochastic process. Model C trained with residual 2. 27

2.11 Model prediction of the complete stochastic process. Model A trained with residual 3. 28

2.12 Model prediction of the complete stochastic process. Model B trained with residual 3. 29

2.13 Model prediction of the complete stochastic process. Model C trained with residual 3. 30

2.14 Moments predicted at times not covered by data. Dashed, solid and dot-dashed line: Model A, model B and model C, respectively. Dots: Experimental data. Rows from up to bottom: Residual 1, residual 2, residual 3, respectively. . . . 31

3.1 Failure probability predicted by models trained with residual 1. From top to bottom: Model A, model B and model C, respectively. Solid line: Model predicted. Square marks: Predicted from empirical damage. Circle mark: Predicted by K-S test. 45

3.2 Failure probability predicted by models trained with residual 2. From top to bottom: Model A, model B and model C, respectively. Solid line: Model predicted. Square marks: Predicted from empirical damage. Circle mark: Predicted by K-S test. 46

3.3 Failure probability predicted by models trained with residual 3. From top to bottom: Model A, model B and model C, respectively. Solid line: Model predicted. Square marks: Predicted from empirical damage. Circle mark: Predicted by K-S test. 47

B.1 IP-MARKOV algorithm scheme 51

List of Tables

2.1 Parameter setup for GA 20
2.2 Inverse Problem solution. 21
2.3 Monte Carlo estimation of mean and variance Prediction Error 32

Bibliography

[1] B. Harris, Fatigue in composites, CRC Press, 2003.

[2] R. Talreja, A continuum mechanics characterization of damage in composite materials, Proc. R. Soc. London 399 (1985) 195–216.

[3] K. Reifsnider, Z. Gao, A micromechanics model for composites under fatigue loading, International Journal of Fatigue 13 (2) (1991) 149–156, ISSN 01421123.

[4] H. Hahn, R. Kim, Fatigue behavior of composite laminate, Journal of Composite Materials 10 (2) (1976) 156.

[5] R. Huston, Fatigue life prediction in composites, International Journal of Pressure Vessels and Piping 59 (1-3) (1994) 131–140, ISSN 03080161.

[6] V. Bolotin, Stochastic models of cumulative damage in composite materials, Engineering Fracture Mechanics 8 (1) (1976) 103–113, ISSN 00137944.

[7] J. Bogdanoff, F. Kozin, Probabilistic models of comulative damage, John Wiley & Sons, ISBN 0-471-88180-5, 1985.

[8] K. Sobczyk, Stochastic models for fatigue damage of materials, Advances in applied probability 19 (3) (1987) 652–673.

[9] K. Sobczyk, B. Spencer, Random fatigue: from data to theory, Academic Pr, 1992.

[10] D. Goldberg, Genetic algorithms in search, optimization, and machine learning, Addison-wesley, ISBN 0201157675, 1989.

[11] B.-S. Wei, S. Johnson, R. Haj-Ali, A stochastic fatigue damage method for composite materials based on Markov chains and infrared thermography, International Journal of Fatigue 32 (2) (2010) 350–360, ISSN 01421123.

[12] J. Degrieck, W. Van Paepegem, Fatigue damage modeling of fibre-reinforced composite materials: Review, Applied Mechanics Reviews 54 (4) (2001) 279, ISSN 00036900.

[13] J. Rowatt, P. Spanos, Markov chain models for life prediction of composite laminates, Structural Safety 20 (1998) 117–135, ISSN 01674730.

[14] R. Ganesan, A data-driven stochastic approach to model and analyze test data on fatigue response, Computers & Structures 76 (4) (2000) 517–531, ISSN 00457949.

[15] P. Billingsley, Statistical Methods in Markov Chains, The Annals of Mathematical Statistics 32 (1) (1961) 12–40.

[16] J. Hyman, Accurate monotonicity preserving cubic interpolation, SIAM journal of scientific computing 4 (4) (1983) 645–654.

[17] R. Carlson, F. Fritsch, Monotone piecewise cubic interpolation, SIAM journal on numerical analysis 22 (2) (1985) 386–400.

[18] A. Di Crescenzo, M. Longobardi, On cumulative entropies, Journal of Statistical Planning and Inference 139 (12) (2009) 4072–4087, ISSN 0378-3758.

[19] P. Zhang, Model selection via multifold cross validation, The Annals of Statistics 21 (1) (1993) 299–313.

[20] B. Efron, G. Gong, A leisurely look at the bootstrap, the jackknife, and cross-validation, The American Statistician 37 (1) (1983) 36–48.

[21] B. Efron, R. Tibshirani, Improvements on cross-validation: The .632 + bootstrap method, Journal of the American Statistical Association 92 (438) (1997) 548–560.

[22] R. Gallager, Discrete stochastic processes, Springer, ISBN 0792395832, 1996.

[23] H. Cramer, On the composition of elementary errors, Skandinavisk Aktuarietidskrift 11 (13-74) (1928) 141–180.

[24] R. Von Mises, Wahrscheinlichkeitsrechnung und ihre Anwendung in der Statistik und theoretischen Physik, Bull. Amer. Math. Soc. 38, 169-170. 2 (9904).

[25] F. Schmid, M. Trede, An L1-variant of the Cramer-von Mises test, Statistics Probability Letters 26 (1) (1996) 91 – 96, ISSN 0167-7152.

[26] J. Navarro, M. Asadi, Some new results on the cumulative residual entropy, Journal of Statistical Planning and Inference 140 (1) (2010) 310–322, ISSN 0378-3758.

[27] R. Gallego, G. Rus, Identification of cracks and cavities using the topological sensitivity boundary integral equation, Computational Mechanics 33 (2) (2004) 154–163, ISSN 0178-7675.

[28] S. Borra, A. Di Ciaccio, Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods, Computational Statistics & Data Analysis 54 (12) (2010) 2976–2989.

[29] R. Kohavi, A study of cross-validation and bootstrap for accuracy es-

timation and model selection, in: International joint Conference on artificial intelligence, vol. 14, 1137–1145, 1995.

[30] Y. Pappas, P. Spanos, V. Kostopoulos, Markov chains for damage accumulation of organic and ceramic matrix composites, Journal of Engineering Mechanics 127 (9) (2001) 915–926.

[31] W. Van Paepegem, J. Degrieck, A new coupled approach of residual stiffness and strength for fatigue of fibre-reinforced composites, International Journal of Fatigue 24 (7) (2002) 747–762.

[32] T. Kam, Fatigue reliability analysis of composite laminates under spectrum stress, International Journal of Solids and Structures 34 (12) (1997) 1441–1461, ISSN 00207683.

[33] F. Richard, The safety-factor calibration of laminates for long-term applications: behavior model and reliability method, Composites Science and Technology 61 (14) (2001) 2087–2094, ISSN 02663538.

[34] S. Carbillet, F. Richard, L. Boubakar, Reliability indicator for layered composites with strongly non-linear behavior, Composites Science and Technology 69 (1) (2009) 81–87.

[35] W. Van Paepegem, J. Degrieck, Calculation of damage-dependent directional failure indices from the Tsai-Wu static failure criterion, Composites science and technology 63 (2) (2003) 305–310.

[36] N. Metropolis, S. Ulam, The Monte Carlo method, Journal of the American Statistical Association 44 (247) (1949) 335–341.

[37] R. Wetherhold, A. Ucci, Probability methods for the fracture of composite materials, Composite Structures 28 (1) (1994) 113–119.

[38] M. Di Sciuva, D. Lomario, A comparison between Monte Carlo and FORMs in calculating the reliability of a composite structure, Composite Structures 59 (1) (2003) 155–162.

[39] S. Tsai, E. Wu, A general theory of strength for anisotropic materials, Journal of composite materials 5 (1) (1971) 58.

[40] S. P.D., A. Kaddour, M. Hinton, Recommendations for Designers and Researchers Resulting from the World-Wide Failure Exercise, Composites Science and Technology 64 (3-4) (2004) 589–604.

[41] M. Hinton, A. Kaddour, S. P.D., A Further Assessment of the Predictive Capabilities of Current Failure Theories for Composites Laminates: Comparison with Experimental Evidence, Composites Science and Technology 64 (3-4) (2004) 549–588.

[42] A. Kaddour, M. Hinton, S. P.D., A Comparison of the Predictive Capabilities of Current Failure Theories for Composite Laminates, Composites Science and Technology 64 (3-4) (2004) 449–476.

[43] A. Orifici, I. Herszberg, R. Thomson, Composite Structures 86 (1-3) (2008) 194–210.

[44] C. Boyer, A. Béakou, M. Lemaire, Design of a composite structure to achieve a specified reliability level, Reliability Engineering & System Safety 56 (3) (1997) 273–283, ISSN 0951-8320.

[45] T. Kam, E. Chang, Reliability formulation for composite laminates subjected to first-ply failure, Composite Structures 38 (1-4) (1997) 447–452.

[46] T. Kam, S. Lin, K. Hsiao, Reliability analysis of nonlinear laminated composite plate structures, Composite Structures 25 (1-4) (1993) 503–510, ISSN 0263-8223.

[47] A. Onkar, C. Upadhyay, D. Yadav, Probabilistic failure of laminated composite plates using the stochastic finite element method, Composite Structures 77 (1) (2007) 79–91.

[48] A. Harbitz, An efficient sampling method for probability of failure calculation, Structural safety 3 (2) (1986) 109–115.

[49] A. Der Kiureghian, T. Haukaas, K. Fujimura, Structural reliability software at the University of California, Berkeley, Structural safety 28 (1-2) (2006) 44–67.

[50] L. Kachanov, Introduction to continuum damage mechanics, vol. 10, Springer, 1986.

[51] J. Lemaitre, J. L. Chaboche, Phenomenological approach of damage rupture, J. Mech Appl 2 (3) (1978) 317–365.

[52] D. Krajcinovic, Continuum Damage Mechanics, Applied Mechanic Reviews 37 (1984) 1–6.

[53] S. Chib, E. Greenberg, Understanding the metropolis-hastings algorithm, The American Statistician 49 (4) (1995) 327–335.

[54] N. Carrere, Y. Rollet, V. Retel, L. Boubakar, J. Maire, Composites structural modeling with uncertain data, Composites Science and Technology 69 (1) (2009) 60–66.