… *solutions for robust engineering*

Bayesian updating for degradation prediction
20 Juin 2014

Rodrigue DECATOIRE

Phimeca Engineering

Outline

- Introduction
- Bayesian updating with Open TURNS
  - Methodology
  - Markov Chain Monte Carlo
- Illustration
- Pros and Cons

Introduction

Civil engineering background

- A lot of structures and infrastructures in reinforced concrete built in the post-war years ;
- Victims of important pathologies, such as carbonation (penetration of carbon dioxide) ;
- Costly maintenance and repair actions.

Objectives

- Identify parameters of degradation models based on the results of an inspection
- Updating those parameters at the next inspection in order to improve the predictions made with the models
- In order to reduce the operating costs of the concerned structures and infrastructures

PHIMECA

Bayesian updating with Open TURNS

Methodology

- Let discrepancy between the measures of the model outputs and the model predictions be modelled by a centered gaussian distribution

$$\mathbf{E} = \{e(z_p, t_q), p = (1, \cdots, P); (q = 1 \cdots, Q)\}$$

- We can explain the measures as a combination of the model predictions and of the model/measurement errors:

$$y(z_p, t_q) = M(\mathbf{X}, t) + e(z_p, t_q)$$

- The inputs vector $\boldsymbol{X}$ is composed of:
  - The vector $\boldsymbol{X^{nobs}}$ of the inputs which are not measured
  - The vector $\boldsymbol{X^{obs}}$ of the inputs which are measured, with their own measurement error modelled by a centered normal distribution such as:

$$\boldsymbol{X^{obs}} \sim \{N(x^{obs}(z_p, t_q), \sigma^{obs}), p = (1, \cdots, P); (q = 1 \cdots, Q)\}$$

- Under those assumptions, the likelihood of the observations writes:

$$L\left(\sigma_e, Y^{obs}\right) = \prod_{p=1}^{P} \prod_{q=1}^{Q} \varphi\left(\frac{M\left[x^{obs}(z_p, t_q), x^{nobs}, t_q\right] - y(z_p, t_q)}{\sigma_e}\right)$$

- Finally, the posterior distribution for the inputs which are not observed follows:

$$\boxed{f_{X^{nobs},\sigma}(x, \sigma) = \frac{1}{c} p_{X^{nobs}}(x^{nobs}) p_{\sigma}(\sigma) L\left(\sigma_e, Y^{obs}\right)}$$

Bayesian updating with Open TURNS

Markov Chain Monte Carlo (MCMC)

- In practice, the computation of the constant c can be cumbersome ;
- With the MCMC algorithm, sample of the posterior distribution are computed without computing this constant.

Initial point $\mathbf{X}_0^{\mathbf{nobs}}$ $\boldsymbol{\sigma}_{e0}$ → Generation of N values until convergence is reached → MCMC: Truncation of the first B samples (Burning period) → Use of the samples to evaluate the quantity of interest

Metropolis-Hastings

1. $x = x_{t-1}$
   candidate $\tilde{x}$ generated by an instrumental distribution $q(\tilde{x}|x_t)$
2. Evaluation of $\alpha = min\left(1, \dfrac{f_{\mathbf{X}^{nobs},\sigma}(\tilde{x},\sigma)q(x_t|\tilde{x})}{f_{\mathbf{X}^{nobs},\sigma}(\tilde{x},\sigma)q(\tilde{x}|x_t)}\right)$
3. $x_t = \tilde{x}$ with a probability of acceptance α and
   $x_t = x_{t-1}$ a probability of rejection $1-\alpha$

Illustration

A simple degradation model

- Carbonation depth

$$\begin{cases} d(t) & = & A(t-t_0)^n \quad \text{for } t > t0 \ ; \\ d(t) & = & 0 \quad \text{otherwise.} \end{cases}$$

- Variables

$$\begin{cases} A & \sim & U(1\times 10^{-9}, 1\times 10^{-3}) \\ n & \sim & U(0.5, 1) \\ \sigma_e & \sim & U(1\times 10^{-12}, 1) \\ t_0 & = & 1 \text{ year} \end{cases}$$

- Observations : 150 observations available at 4 different dates, simulated by a finite element model
- Bayesian updating with
  - Open TURNS
  - PyMC (Python Monte Carlo) – module dedicated to bayesian analysis

Markov Chain

- 500,000 simulations, (50,000 for the Burn In period, thinning is equal to 5)

Illustration

Proposal distributions

- Gaussian distributions with standard deviation

$$\begin{cases} \sigma_A^{proposal^{(i)}} & = & 2.5 * \sigma_A * c_A^{(i)} \\ \sigma_{t_0}^{proposal^{(i)}} & = & 2.5 * \sigma_{t_0} * c_{t_0}^{(i)} \\ \sigma_n^{proposal^{(i)}} & = & 2.5 * \sigma_n * c_n^{(i)} \\ \sigma_{\sigma_e}^{proposal^{(i)}} & = & 1 \times 10^{-3} * \sigma_{\sigma_e} * c_{\sigma_e}^{(i)} \end{cases}$$

- At the (i)-th tuning step of the distributions

Tuning of the distributions

$$\begin{cases} c_\bullet^{(0)} & = & 1 & \\ c_\bullet^{(i)} & = & c_\bullet^{(i-1)} & \text{if } 0.15 < a_{r_\bullet}^{(i)} < 0.2 \\ c_\bullet^{(i)} & = & 0.1 * c_\bullet^{(i-1)} & \text{if } a_{r_\bullet}^{(i)} < 0.15 \\ c_\bullet^{(i)} & = & 10 * c_\bullet^{(i-1)} & \text{if } a_{r_\bullet}^{(i)} > 0.2 \end{cases}$$

- With $a_{r_\bullet}^{(i)}$ the acceptance rate of the $\bullet$ random variable
- The proposal distributions are tuned every 10,000 simulations

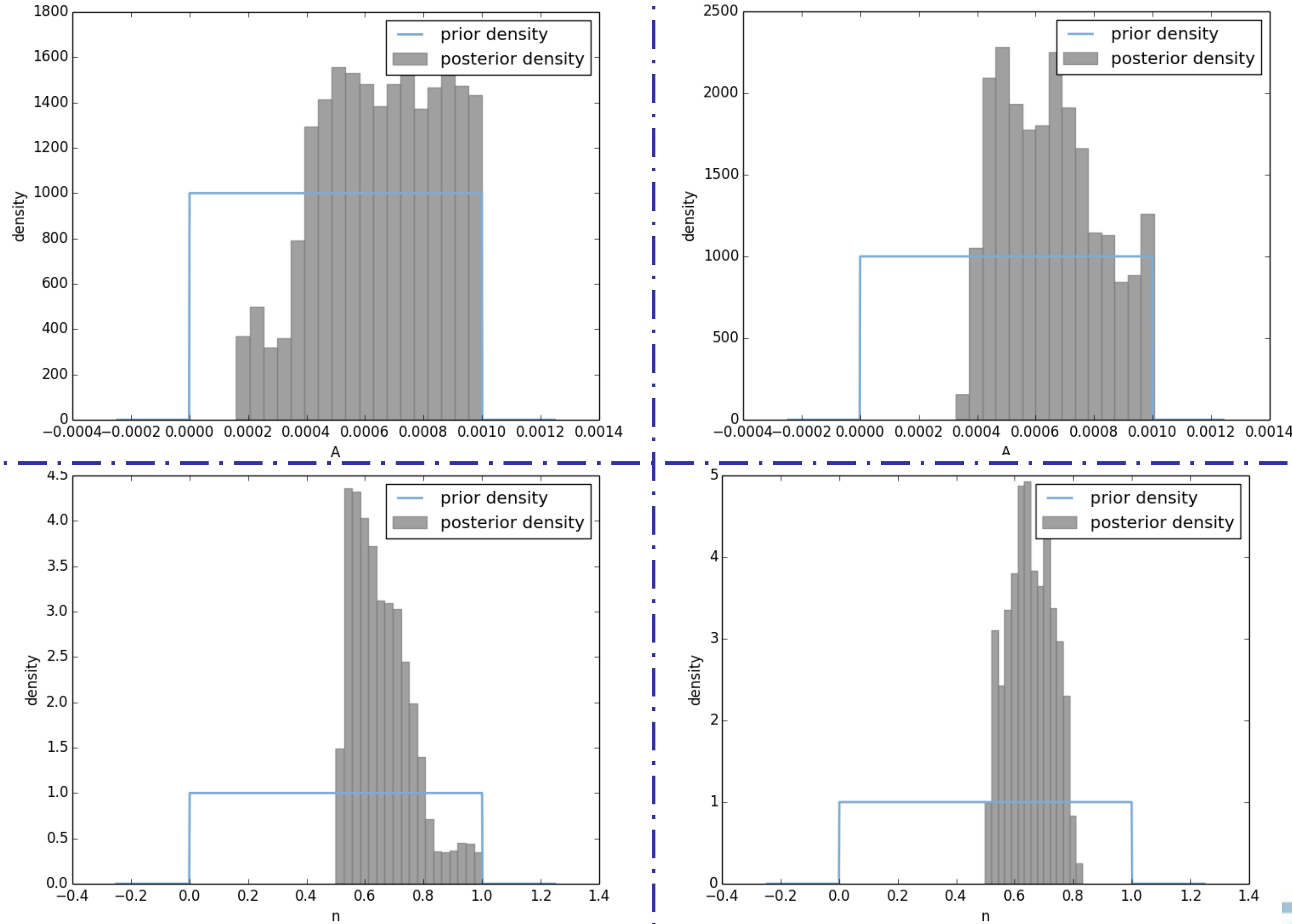
Illustration

Results

Pymc | Open TURNS

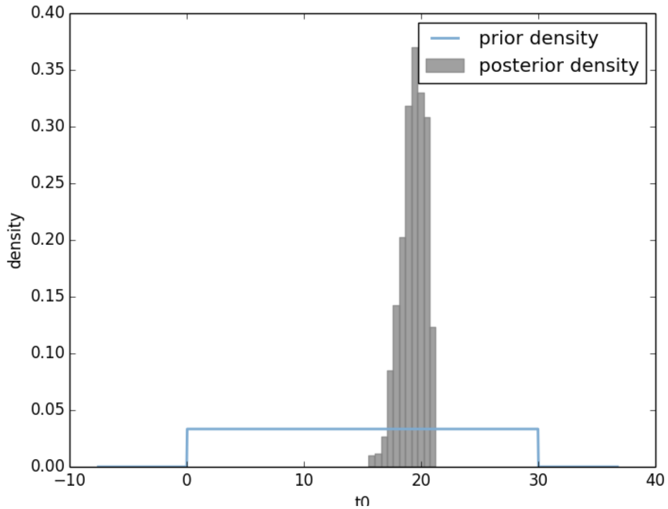
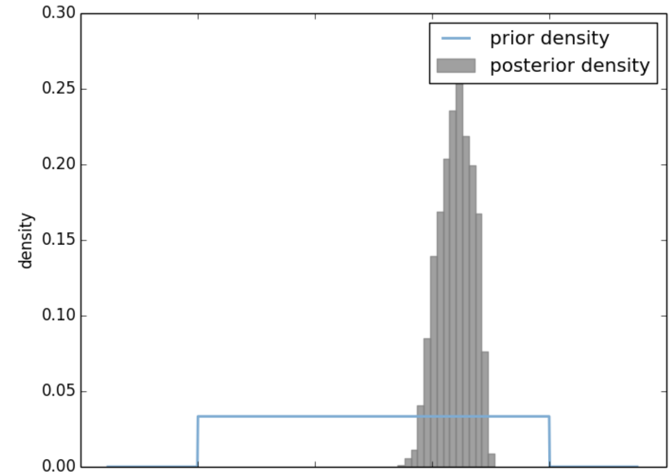
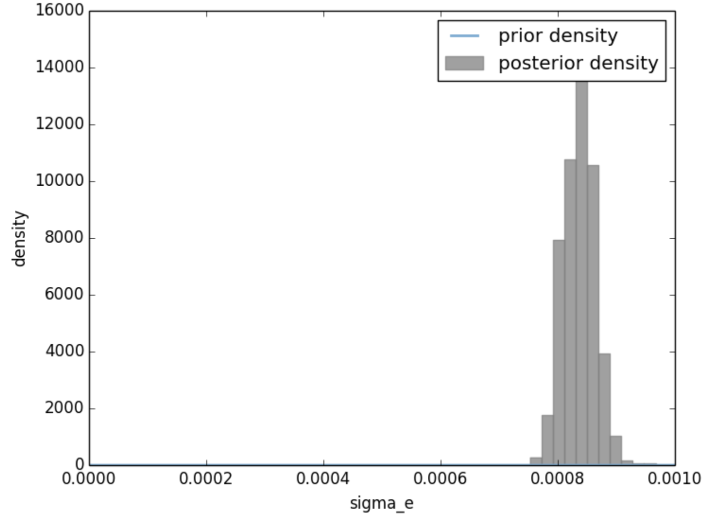
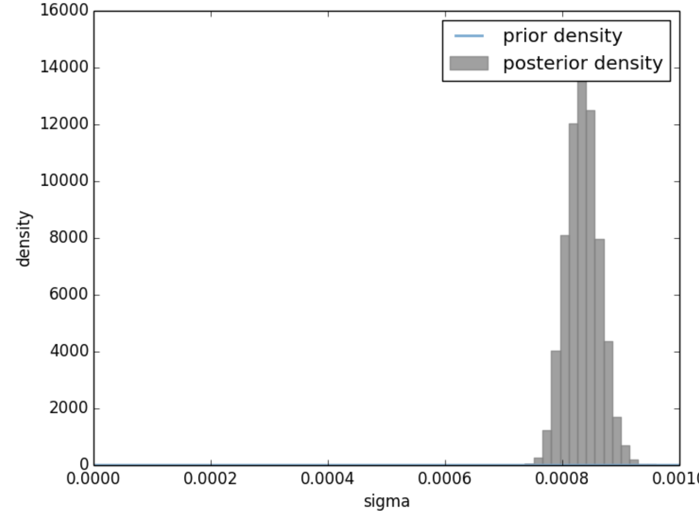
A

n



Illustration

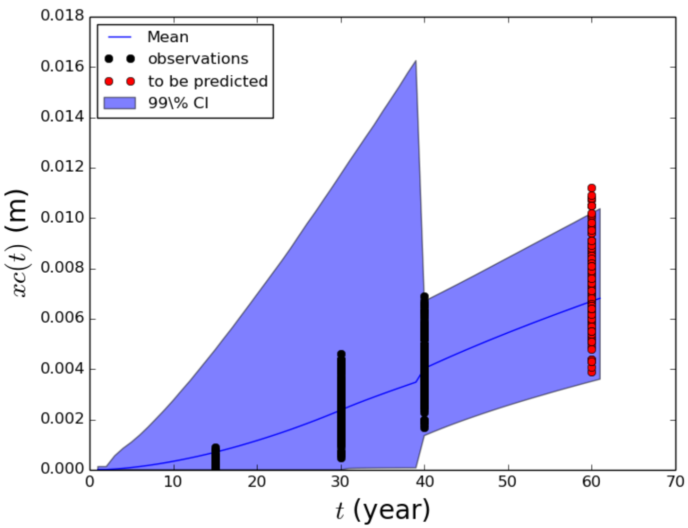
Results

	Pymc	Open TURNS
t_0		
σ_e		

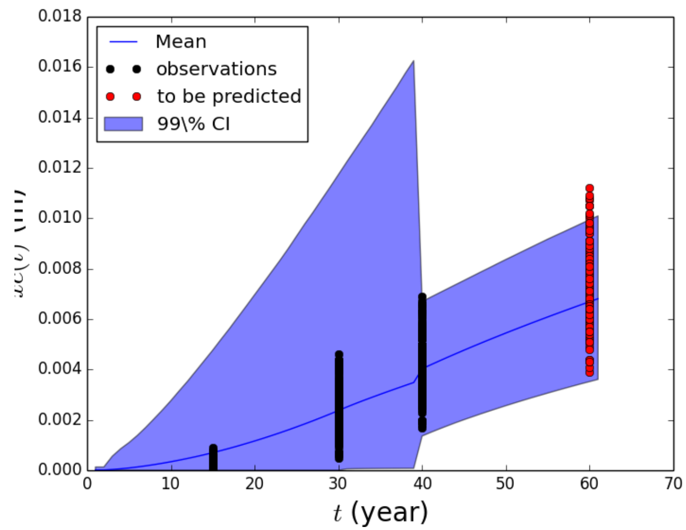
Illustration

Results

Pymc



Open TURNS



- The prediction quality is increased
- Despite the difference in the posterior, the updated predictions are equivalent

Pros and Cons

Behind the scene…

- Open TURNS beneficiates from its large application field, this is appreciable
- The possibility to tune the proposal density differently following range of the acceptance rate (as PyMC does it) is missing
- As well as the capacity to stop and resume the simulations
- However the choice of the proposal density appears to be much easier (only two distributions are available for that in PyMC)
- Yet, the computational time of PyMC is faster in this case(4h30 minutes against 5h20 for Open TURNS)
  - Coming from the use of multiple OpenTURNS Python functions ?