Space-time Gaussian processes for the approximation of partially converged simulations

Victor Picheny

Space-time Gaussian processes for the approximation of partially converged simulations

V. Picheny[1]

[1] Ecole Centrale de Paris, Grande voie des vignes, Chatenay-Malabry, France

Keywords: Partial convergence, computer experiments, metamodeling

1 Introduction

Using computer experiments and metamodels for facilitating optimization and statistical analysis of engineering systems has become commonplace (Sacks et al. (1989), Jones et al. (1998)). However, despite the continuous growth of computational capabilities, the complexity of the simulators still drastically limit the number of available experiments, which are often insufficient to build accurate metamodels.

An efficient solution to alleviate the computational cost consists of using degraded versions of the expensive simulator to provide faster but less accurate evaluations of the simulator output. Such approximations can be obtained by using coarser mesh (in Finite Element methods), a simpler partial differential equation problem, or geometry simplification for instance. The degraded simulator is often called low-fidelity (LF) model and the expensive version high-fidelity (HF) model. Using metamodels in this context has been addressed by many authors in the litterature. For instance, Alexandrov et al. (2000) and Gano et al. (2006) used metamodels to approximate the difference between LF and HF models. Kennedy and O'Hagan (2000) proposed a so-called auto-regressive model to integrate data with various fidelities. All these approaches assumes (1) that a discrete (small) number of fidelities is available, and (2) that LF responses are smoother than HF responses.

A less explored but promising alternative is to use partially converged simulations as a low-fidelity model, by stopping artificially the convergence of the simulator solver at early stage. Such approach has many advantages, among which the use of a single simulator instead of a different simulator for each fidelity level, and the possibility of having as many levels of accuracy as desired.

Using metamodels with such data is an open and difficult question, that differs from the classical multifidelity framework since unconverged responses are likely to be a lot rougher than converged ones, and the number of fidelity levels can be very large. In Forrester et al. (2006), it is observed that all simulations within the design space tend to converge in unison, so partially converged responses are corrected using a constant shifting value and fused with fully converged responses to build a classical metamodel. Although demonstrated to be quite efficient already, this approach hinders the potential of partial convergence, since it allows the use of only two fidelity levels, and using a constant shift requires simulations to achieve a relatively high level of convergence.

This work addresses the issue of fitting a metamodel to partially converged simulation data, when convergence level potentially varies from one design to another. To do so, we propose to use a Gaussian process model in the joint space of design parameters and computational time. The model is constructed by building a covariance function that reflects accurately the actual structure of the error.

In the next section, we describe a Computational Fluid Dynamics (CFD) simulator optimization problem, which response illustrates the important behaviors of partially converged simulations. Then, we present a Gaussian process model for the joint design-time space, followed by learning issues and solutions specific to this model. Finally, the model is applied to the analysis of the CFD problem.

2 A Motivating example: 2D pipe flow simulation

To motivate our approach and highlight the important points of partial convergence, we consider the optimization problem of an S-shaped pipe, which form is defined parametrically. A two-dimensional CFD model is build using OpenFOAM and its solver simpleFoam (steady-state, incompressible, turbulent flow). A constant flow velocity is imposed at the pipe input, and a null pressure at the

output. The pipe contour is defined with the help of 13 parameters. The objective is to maximize the uniformity of the flow velocity at the end of the S-section. Figure 1 shows the shape parameters and the CFD model for the nominal parameters. The objective function is the velocity standard deviation between P9 and P10, which we will refer to as f_{SD}.

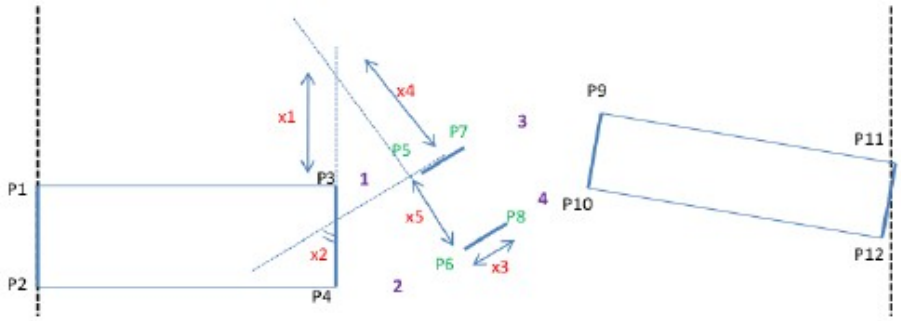

FIGURE 1. Five first shape parameters for the 2D pipe model. The remaining parameters define Bezier curves between the points P3-P5, P4-P6, P7-P9 and P8-P10. All the points are fixed except P5, P6, P7 and P8.

OpenFOAM allows us to monitor the velocity field for each solver step, so we can measure the convergence directly on the objective function. First, we generate 20 designs using Latin hypercube sampling (LHS), and for each solver step, we compute f_{SD}. Figure 2 shows the evolution of the 50 designs for all time steps. Although converging to different values, all the convergence curves have very similar shapes (which goes in the sense of convergence in unison, as described in Forrester et al. (2006)).

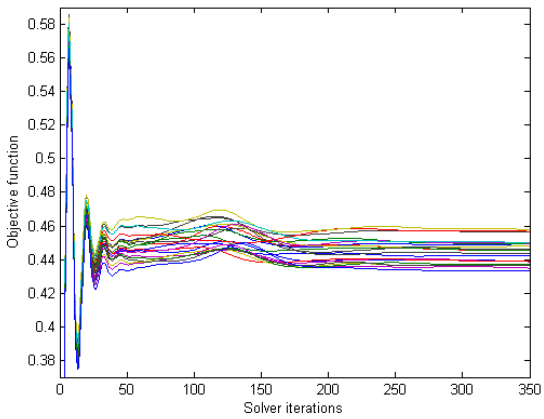

FIGURE 2. Response convergence for 20 designs.

Now, in order to represent the data in the joint design-time space, we fix all the parameters to their nominal value but x_2 (which is the most sensitive parameter), and 100 designs are generated for x_2 values uniformly distributed between its bounds. For all designs, 500 solver iterations are used for convergence. Figure 3 shows three designs and their converged velocity fields, for minimum (left), mean (center) and maximum (right) values of x_2.
The objective function f_{SD} and the convergence error are then shown in the x_2 and time t plan (Figure 4). The convergence error is here taken as the current objective function value minus the value at step 500.
First, we can observe a strong regularity of the response in both x_2 and t directions, which means that two close designs with the same number of convergence steps will have similar responses. Obviously, when t increases, the error decreases and tends towards zero, so the response becomes invariant with respect to t. One can also observe that the error fluctuations have a higher frequency for small t than for high t. These are the three key characteristics that we want to be included in our model, as we describe in the next section.

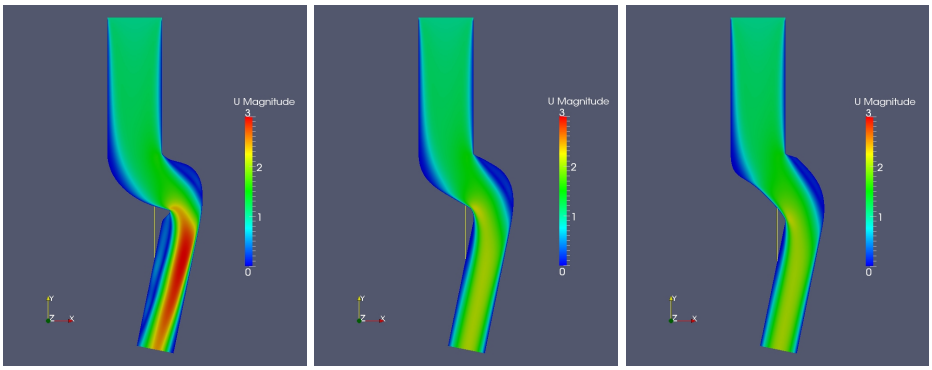

FIGURE 3. Three designs and velocity fields for x_2 taking its minimum (left), mean (center) and maximum values (right).

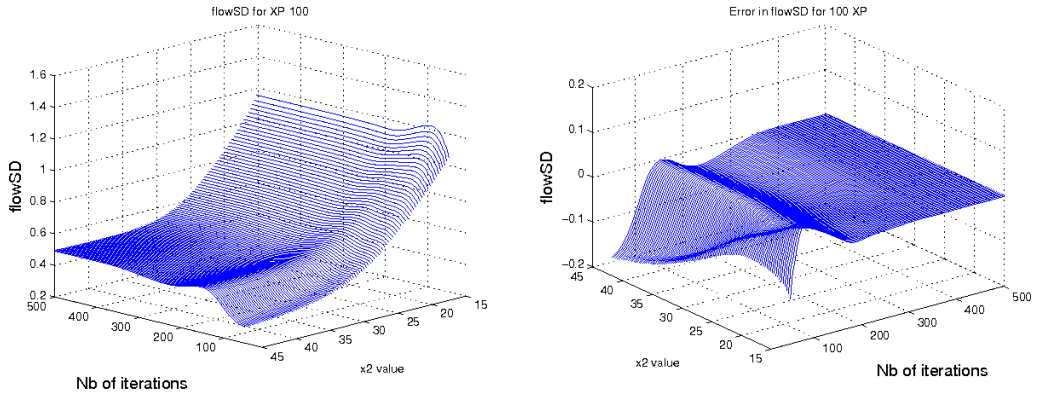

FIGURE 4. Evolution of objective function (left) and objective function error (right) as a function of x_2 and t. Time axis direction is inversed in the left figure to increase readability

3 A Gaussian process surrogate for partially converged simulations

Let us first introduce some notation. We denote by y the response of a numerical simulator or function that is to be studied: $y : x \in D \subset \mathbb{R}^d \longrightarrow y(x) \in \mathbb{R}$. In the framework of computer experiments, Gaussian process is a popular metamodel in particular because of its flexibility and rich statistical interpretation. Assuming a covariance structure for the data, a metamodel is built by conditioning a gaussian process on n observations $\mathbf{Y}_n$ evaluated at a set of input parameters $\mathbf{X}_n \in D^n$ called the design of experiments. The choice of the covariance is particularly crucial, and is often based on user's previous knowledge on the response behavior. In the framework of kriging (Matheron (1969)), stationary covariances are used.
When partial convergence is considered, an observation y_i is defined by both input parameters $x \in D$ and computational time $t \in R^{+*}$ (typically proportional to the number of solver iterations). In order to predict such types of responses, Gaussian processes are particularly adapted since they allow the definition of models that can inherit the structure of the function to approximate.
Indeed, we consider that the observed function is a realization of a random process $Y(x,t)$, which is the sum of a process F independent of t, and a process G that depends on both x and t:

$$Y(x,t) = F(x) + G(x,t) \tag{1}$$

F is the response given by the simulator with complete convergence, and then can be modeled with the usual assumptions: stationarity, ergodicity, etc. (as for a kriging model in a classical framework). G is the error term due to partial convergence, and has a more complex structure. In the x space, it can be observed that two runs with close sets of input parameters converge in a similar fashion, hence their partially converged response are correlated. In the t direction, except for the first few iterations that often show large oscillations, the convergence is smooth so the responses evaluated at successive time steps are also correlated. Moreover, G is naturally instationary in t, since the convergence error tends to zero when the computational time increases. More generally, it can be assumed that the error variance decreases monotonically with computational time.

Hence, we propose a covariance function for G of the following form (with the notation $u = (x, t)$):

$$k_G(u, u') = \sigma^2(t, t') r_x(x, x') r_t(t, t') \tag{2}$$

where r_x and r_t are correlation functions, and $\sigma^2(t, t')$ is the process variance. In the following, we make appropriate choices so k_G is still a kernel. For details on this topic, see Rasmussen and Williams (2006) (chapter 4) for instance.
The correlation r_x can be taken as stationary, i.e. $r_x(x, x') = r_x(|x - x'|)$, for instance, the matern 3/2 function:

$$r_x(|x - x'|) = \left(1 + \sqrt{3}\frac{|x - x'|}{\theta}\right) \exp\left(-\frac{|x - x'|}{\theta}\right) \tag{3}$$

The correlation r_t has to account for the increasing smoothness of the error (high oscillations for the first steps, then smooth convergence). To do so, we propose to define for the correlation a range depending on time, for instance with the matern 3/2 function:

$$r_t(t, t') = \left(1 + \sqrt{3}\frac{|t - t'|}{\theta(t, t')}\right) \exp\left(-\frac{|t - t'|}{\theta(t, t')}\right) \tag{4}$$

with: $\theta(t, t') = \theta_0 + \frac{\Delta_\theta}{2}(t + t')$, $\theta_0, \Delta_\theta \in R^+$.
The process variance $\sigma^2(t, t')$ must decreases towards zero to ensure that the error is null when $t \to +\infty$. Here we choose a decreasing exponential form for the variance:

$$\sigma^2(t, t') = \sigma_G^2 \exp(-\alpha \frac{t + t'}{2}) \tag{5}$$

Finally, the kernel of the process Y is the sum of the kernels of F and G (Rasmussen and Williams (2006), p.95):

$$k_Y(u, u') = k_F(x, x') + k_G(u, u') \tag{6}$$

Using this kernel, we are able to perform simulation, conditional simulation, hence learning with gaussian processes.
Let $\mathbf{Y}_n = [y_1, \ldots, y_n]^T$ be a set of observations, $\mathbf{X}$ the matrix of design parameters, $\mathbf{T}$ the vector of times and $\mathbf{U} = [\mathbf{X}, \mathbf{T}]$ the experimental matrix. In the fashion of Simple Kriging, the mean and variance of Y at $u^* = (x^*, t^*)$ conditional on the observations $\mathbf{Y}_n$ are given by (Matheron (1969)):

$$m(u^*) = \mathbf{k}_Y(u^*)^T \mathbf{K}_Y \mathbf{Y}_n \tag{7}$$

$$s^2(u^*) = k_Y(u^*, u^*) - \mathbf{k}_Y(u^*)^T \mathbf{K}_Y \mathbf{k}_Y(u^*) \tag{8}$$

with: $\mathbf{K}_{Yi,j} = k_Y(u_i, u_j)$ and $\mathbf{k}_Y = [k_Y(u^*, u_1) \ldots k_Y(u^*, u_n)]$.
The functions $m(.)$ and $s^2(.)$ define the gaussian process model, which provide a prediction mean and variance for any given design with convergence level. As for the simple kriging model, m is equal to the observations and s is equal to zero at the points of the DOE.
In most applications, the value of interest is the actual response, i.e. the asymptotic value for $t = +\infty$. From equation 6, the covariance $k_Y(u, u^*)$ is defined for $u^* = (x^*, +\infty)$ is simply equal to $k_F(x, x^*)$. Then, we can define an asymptotic prediction independent of t, equal to:

$$m_\infty(x^*) = \mathbf{k}_F(x^*)^T \mathbf{K}_Y \mathbf{Y}_n \tag{9}$$

$$s^2_\infty(x^*) = \sigma_F^2 - \mathbf{k}_F(x^*)^T \mathbf{K}_Y \mathbf{k}_F(x^*) \tag{10}$$

One can notice that these equations take the form the equations of a Simple Kriging with correlated residuals.

4 Learning issues and solutions

In simple kriging, the covariance parameters are most of the time learned using an optimization process, for instance by maximizing the likelihood of the observations, or by minimizing the cross-validation error. This step is particularly critical for the accuracy of the kriging model, and is known to be difficult, in particular when the number of observations is small and the number of parametes large.
Our GP model requires the knowledge of the parameters of the covariance function of k_Y. Assuming anisotropy in the x space and matern 3/2 shape for all covariances, we have:

- for the stationary covariance k_F: $d+1$ parameters, σ_F^2, $\theta_F^1, \ldots, \theta_F^d$,
- for the stationary correlation r_x: d parameters, $\theta_G^1, \ldots, \theta_G^d$,
- for the correlation r_t: two parameters, θ_0 and Δ_θ,
- for the process variance σ^2: two parameters, σ_G^2 and α.

Learning these $2d+5$ parameters in a single optimization loop seems unrealistic here, since the objective function is likely to be highly multimodal, and ensuring a good exploration may be too expensive computationally.
Besides, with partial convergence, the design of experiments takes a particular form, which can be used to simplify the learning process. Indeed, when an observation is made at x with time t, the response can be calculated without any computational effort for all the time steps smaller than t. In other words, one has access to the response convergence for the design x from zero to t: $\{y(x,0), y(x,1), \ldots, y(x,t)\}$. In the following, we refer to a series of data for the same x and increasing t as *response* (or *error*) *trajectory*.
Then, we propose to decompose the kernel parameters learning into successive steps: first, we learn the process variance parameters, then the r_t parameters, and finally all the other parameters.

4.1 Learning the process variance

The process variance function accounts for the convergence speed of the simulator (the variance of the error due to partial convergence). This speed might differ from one design to another, especially if the design space is large, but it is reasonable to consider speed as uniform, and then learn it from a small number of simulations.
We assume here that the user has performed a small number K of fully converged simulations, well spread in the design space. The error trajectories can be known exactly by substracting the converged responses to the partially converged response trajectories. We have then realizations of the process G for K designs and N times: $g(x_1,t_1), \ldots, g(x_1,t_N), \ldots, g(x_K,t_1), \ldots, g(x_K,t_N)$.
Since they are yet unknown, we assume that the correlations in x and t are both null for the process G. Following that assumption, we have:

$$G(x_i,t_j) \sim N\left(0, \sigma^2 \exp(-\alpha t_j)\right) \quad \text{independently} \tag{11}$$

$$\overline{G}(x_i,t_j) = \frac{G(x_i,t_j)}{\sqrt{\sigma^2 \exp(-\alpha t_j)}} \sim N(0,1) \quad \text{independently} \tag{12}$$

Then, the parameters σ^2 and α are chosen to ensure that $\overline{G}(x_i,t_j)$ follows a standard normal distribution. One can remark that for a given α, there exists a unique σ^2 that ensures that the sample $\{\overline{G}(x_i,t_j)\}_{i,j}$ has a variance equal to one:

$$\hat{\sigma}^2 = \frac{1}{K \times N} \sum_{i,j} \overline{G}(x_i,t_j)^2 - \left(\frac{1}{K \times N} \sum_{i,j} \overline{G}(x_i,t_j) \right)^2 \tag{13}$$

Then, α can be found by performing a (mono-dimensional) optimization, for instance using the Kolmogorov-Smirnov distance between empirical standard normal distributions.

4.2 Range parameters for the time correlation

Once the process variance parameters are known, the time correlation parameters can be assessed using the same data, assuming that the correlation in x is null. The r_t parameters account for the regularity in the t direction, which depends on the simulator solver and can be considered as uniform for the design space, so a small number of trajectories are sufficient to learn the parameters.
Several possibilities exist at this point, but all involve an optimization loop. For instance, the parameters can be found by minimizing leave-one-out (or leave-k-out) cross-validation errors. Here we choose to use the likelihood information. It is well-known that maximizing the likelihood of a gaussian vector realization $\mathbf{Y}_n$ is equivalent to minimizing the following quantity:

$$l = \log\det \mathbf{K}_G + \mathbf{Y}_n^T \mathbf{K}_G^{-1} \mathbf{Y}_n \tag{14}$$

Hence, the time correlation parameters are chosen by solving the optimization problem:

$$[\theta_0^*, \Delta_\theta^*] = \arg\min l(\theta_0, \Delta_\theta) \tag{15}$$

Such kind of optimization is not easy to solve, since it is not convex in general, and may require the use of a global optimizer. Here, the possibility of data vizualization (trajectory by trajectory) might be useful to set realistic bounds for the two variables and help the optimization process.

4.3 Other parameters

Finally, the other parameters for the covariance of F (σ_F^2, $\theta_F^1, \ldots, \theta_F^d$) and the correlation r_x of G ($\theta_G^1, \ldots, \theta_G^d$) have to be learned together, using log-likelihood minimization for instance:

$$\left[\sigma_F^2, \theta_F^1, \ldots, \theta_F^d, \theta_G^1, \ldots, \theta_G^d\right] = \arg\min \left(\log\det \mathbf{K}_Y + \mathbf{Y}_n^T \mathbf{K}_Y^{-1} \mathbf{Y}_n\right) \tag{16}$$

If the amount of fully converged simulations is large, it might be preferable to learn the covariance parameters of F first based on the converged response only, and then the parameters of G based on all the data.

5 Application to the 2D pipe flow example

In this section, we illustrate the learning steps of the previous section applied to the data on the CFD example.

5.1 Learning time parameters

We assume that three fully converged runs (randomly chosen on the initial DOE) are available for learning the time parameters. The corresponding data (1500 error values) is represented in Figure 5.

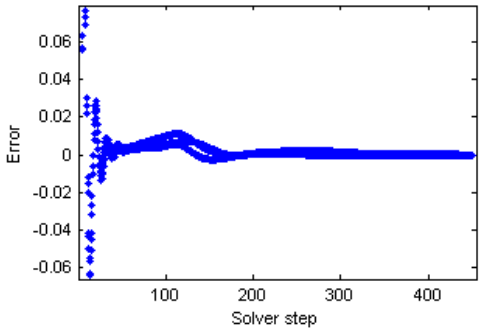

FIGURE 5. Available data for learning time parameters.

The first step is to assess the values of σ^2 and α. Figure 6 shows the graph of the Kolmogorov-Smirnov (KS) distance as a function of α (left), and the empirical distribution of the sample normalized with the best α (right). We can see that the KS distance has a marked minimum for $\alpha = 0.0128$. For this value, we have $\sigma^2 = 5.4 \times 10^{-4}$. The empirical and standard normal distributions differ substantially; this can be imputed to the strong dependency on time, which has been neglected here.
σ^2 and α provide an estimate of the error variance for any t. Figure 7 draws the square error in logarithmic scale as well as the obtained variance model. We see that the variance trend matches quite well the data.
Then, the parameters θ_0 and Δ_θ are learned by maximum likelihood. The optimal parameters are $\theta_0 = 5$ and $\Delta_\theta = 0.03$ (so the range is equal to 20 when $t = 500$). To validate visually our model, we represent the actual error trajectory of a new design (randomly chosen), and the associated Gaussian process model based on 20 observations of this trajectory, uniformly chosen between $t = 0$ and $t = 500$. The trajectory and GP model (mean and 95% confidence interval) are shown in Figure 8. Note that such kind of data is not realistic, since in a real case the response would be known for all the intermediate time steps, but the shape of the GP mean and confidence interval

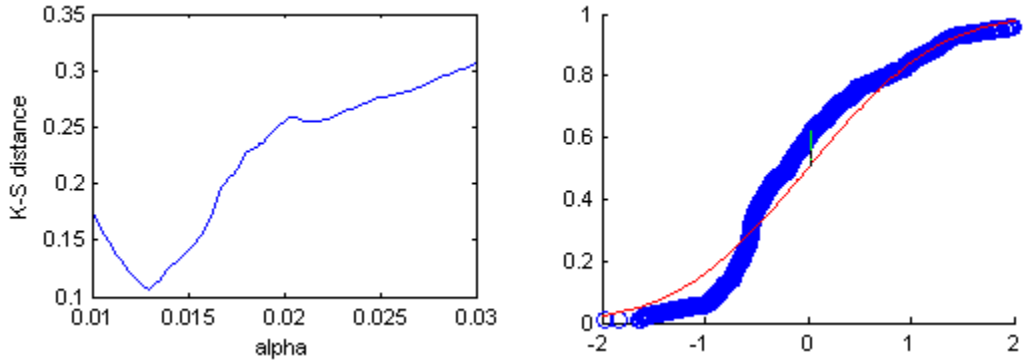

FIGURE 6. Left: evolution of the Kolmogorov-Smirnov distance with α (left). Right: Empirical distribution of the sample normalized with the best α and CDF of the standard normal distribution.

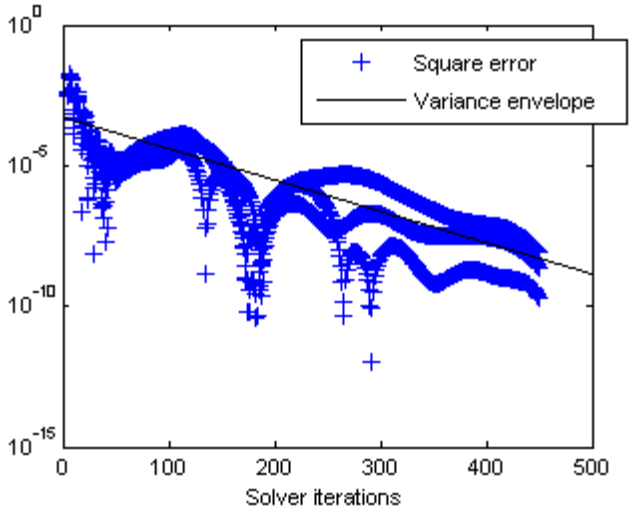

FIGURE 7. Square error and variance model.

reflects the accuracy of the model. Here, the regularity of the model mean is similar to the one of the actual process, except for the very first time steps, where it shows very high variability. The confidence intervals are also quite realistic, and account for the fact that the process becomes flatter for large t.

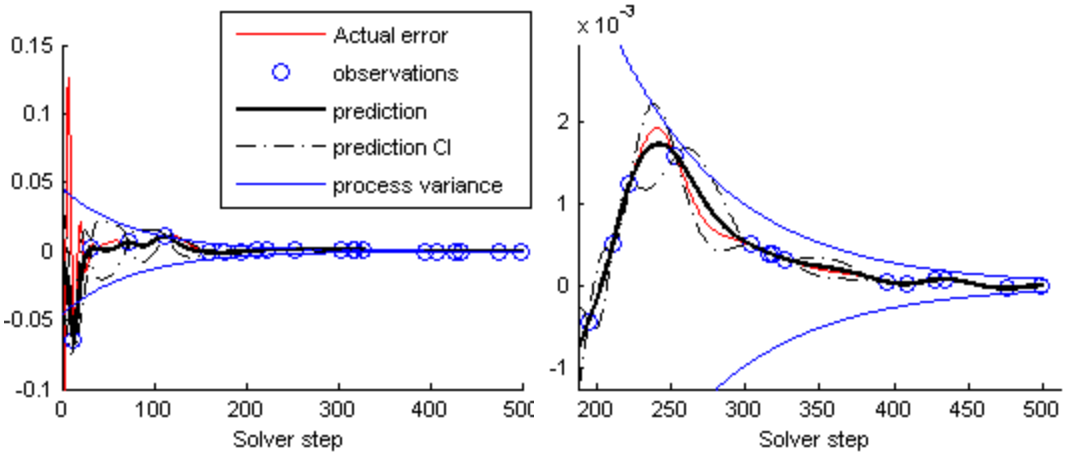

FIGURE 8. Example of error trajectory approximation using a GP model (left: complete trajectory, right: detail).

5.2 Space-time approximation

Finally, we present the results in the joint design-time space, where x_2 is the only varying design parameter, as described in section 2. For this very simplified setup, we do not consider the problem of learning the covariance parameters in the x direction, so σ_F^2, θ_F and θ_G are assumed to be known.
The DOE here consists of five partially converged simulations, with different convergence times; for each design, the response is sampled every five time steps (to limit the total number of observations and reduce the covariance matrix size). The Gaussian process model, DOE, and asymptotic responses are represented in Figure 9.

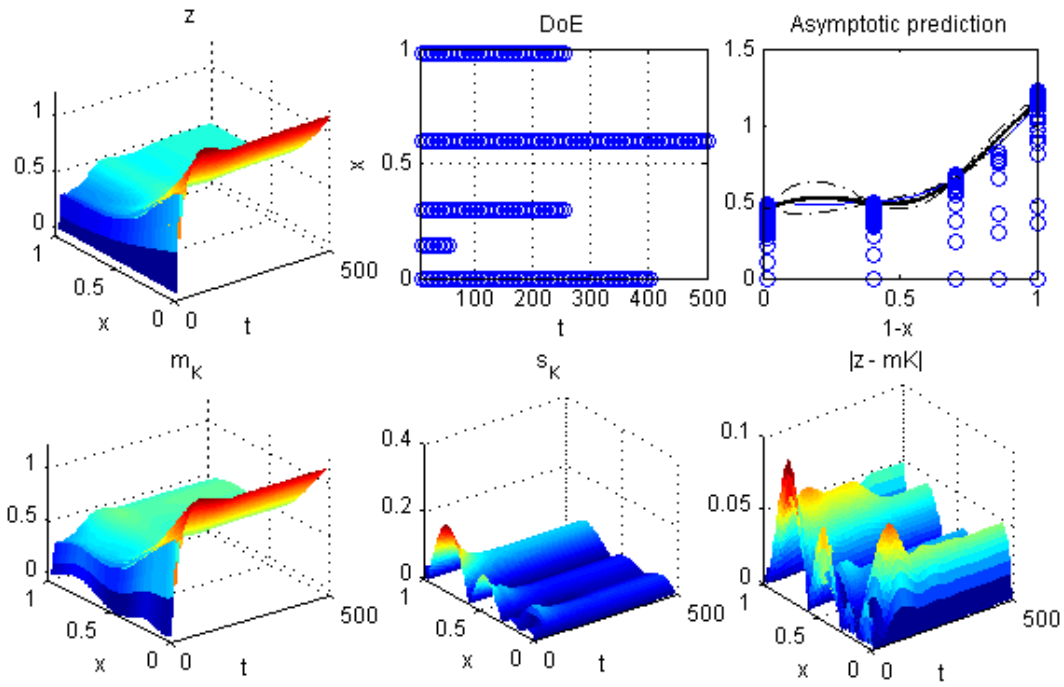

FIGURE 9. Prediction of the output of the CFD simulator based on five observations with different convergence levels. Upper graphs: exact response, design of experiments, and (right): asymptotic prediction (black), exact response (red) and observations (blue). Lower graphs: mean and standard deviation of the gaussian process model; difference between the model mean and actual response.

Due to the high amount of data regarding the problem complexity, the model mean, as well as the asymptotic prediction, are very accurate. The asymptotic prediction is somehow similar to a kriging with nugget effect, but take into account the negative bias of the observations for small t. The shape of the prediction variance s^2 reflects the actual process structure, with higher values for small t. For large t, the variance due to partial convergence tends to zero but s^2 remains high to account for the uncertainty in the asymptotic process.

6 Conclusion

In this work, we have proposed a way to approximate data from partially converged simulations, using Gaussian processes in the joint design-time space. The main idea was to build a covariance kernel that reflects accurately the actual structure of the response considered: the observed response is modeled as the sum of a stationary process depending on design parameters only and an error process which variance decreases towards zero when time tends to infinity. In addition, we proposed a complete procedure for the learning of the model parameters, by decomposing in into a series of simple optimization problems. Finally, we have applied our model to a real simulator, and showed that it approximates accurately partially converged data in small dimension.
Future research may include the application of this model to higher dimension problems, and optimization under partial convergence using EGO-like strategies. Another important issue is the selection of useful data, since the presented model tends to have large covariance matrices, which can be computationally costly.

Bibliography

Alexandrov, N., Lewis, R., Gumbert, C., Green, L., Newman, P., 2000. Optimization with variable-fidelity models applied to wing design. AIAA paper 841 (2000), 254.

Forrester, A., Bressloff, N., Keane, A., 2006. Optimization using surrogate models and partially converged computational fluid dynamics simulations. Proceedings of the Royal Society A 462 (2071), 2177.

Gano, S., Renaud, J., Martin, J., Simpson, T., 2006. Update strategies for kriging models used in variable fidelity optimization. Structural and Multidisciplinary Optimization 32 (4), 287–298.

Jones, D., Schonlau, M., Welch, W., 1998. Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13 (4), 455–492.

Kennedy, M., O'Hagan, A., 2000. Predicting the output from a complex computer code when fast approximations are available. Biometrika 87 (1), 1.

Matheron, G., 1969. Le krigeage universel. Cahiers du centre de morphologie mathématique 1.

Rasmussen, C., Williams, C., 2006. Gaussian processes for machine learning. Springer.

Sacks, J., Welch, W., Mitchell, T., Wynn, H., 1989. Design and analysis of computer experiments. Statistical science, 409–423.