Processus gaussiens pour l'utilisation massive et automatique de simulateurs numériques

François Bachoc, Karim Ammar, Jean-Marc Martinez

University Paul Sabatier, Toulouse
CEA Saclay

Journées MAS 2016

Outline

- This talk is a case study of the metamodeling of a nuclear simulation code
  - large number of simulation results
  - unexpected numerical issues
- Objectives
  - provide a concrete example of the metamodeling process
  - see if there is a need for new theory and methodology
- The paper :

F. Bachoc, K. Ammar and J.M. Martinez "Improvement of code behaviour in a design of experiments by metamodeling", *Nuclear Science and Engineering (2016) 183(3) 387-406*.

Fuel pin simulation

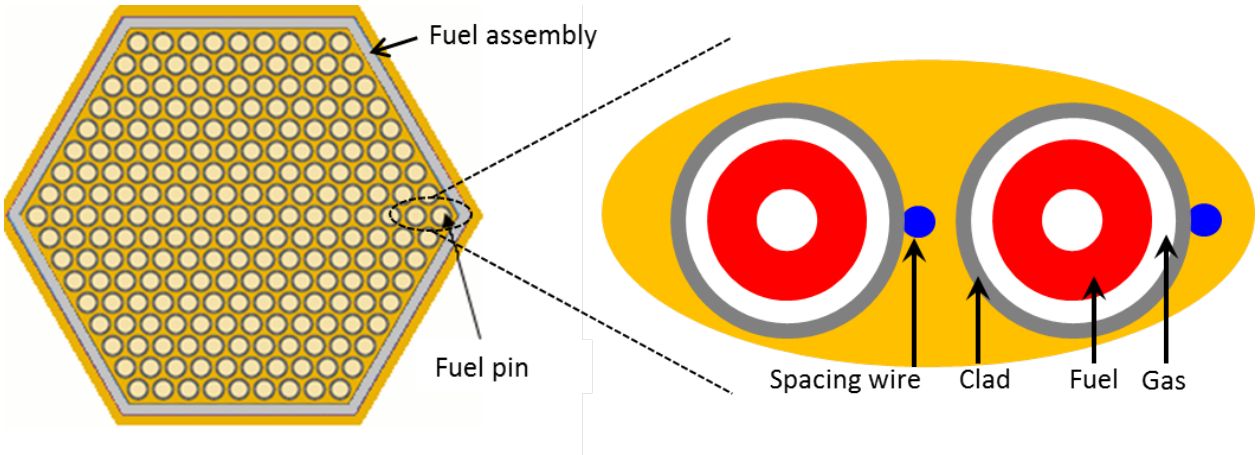


FIGURE: A schematic representation of a fuel pin and a fuel assembly in nuclear fast-neutron reactors.

Simulation process

- The nuclear radiation is an input parameter (may come from an other type of simulation)
- The power and heat maps are output parameters

Aspects of interest

- Efficiency (produced-energy / fuel ratio)
- Security (is the fuel too hot ? Is the support altered ?)

The Germinal parametric study

We represent the input and output maps by a finite number of parameters :

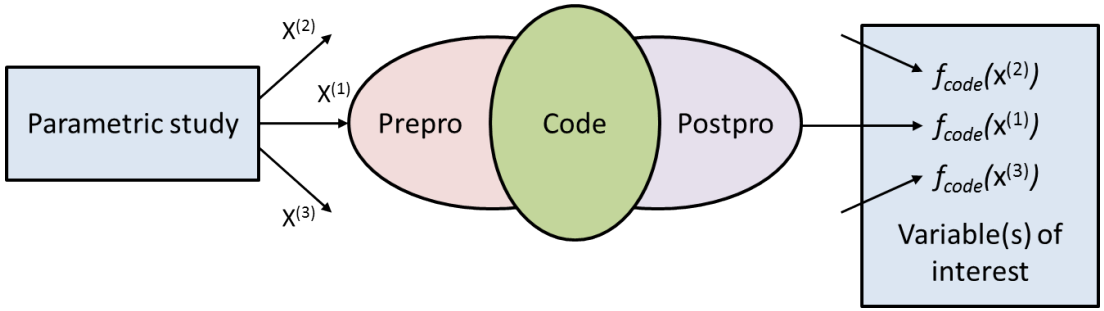


FIGURE: Illustration of the code manager for Germinal

- 11 input parameters in $[0,1]^{11}$ (normalized)
  - x_1 : cycle length
  - $x_2, \ldots, x_7$: nature of the fuel pin (material and geometry)
  - x_8, x_9, x_{10} : input power map (would come from a neutronic simulation in case of code coupling)
  - x_{11} : volume of expansion for fission gas.
- 1 output parameter in $\mathbb{R}$ (non normalized)
  - $f_{code}(\boldsymbol{x}) = f_{code}(x_1, \ldots, x_{11})$: fusion margin
  - 'fusion margin' = 'fuel melting temperature' $-$ 'maximal temperature along simulation'
  - $f_{code}(\boldsymbol{x}) < 0 \Rightarrow$ security issue

Set of simulation results

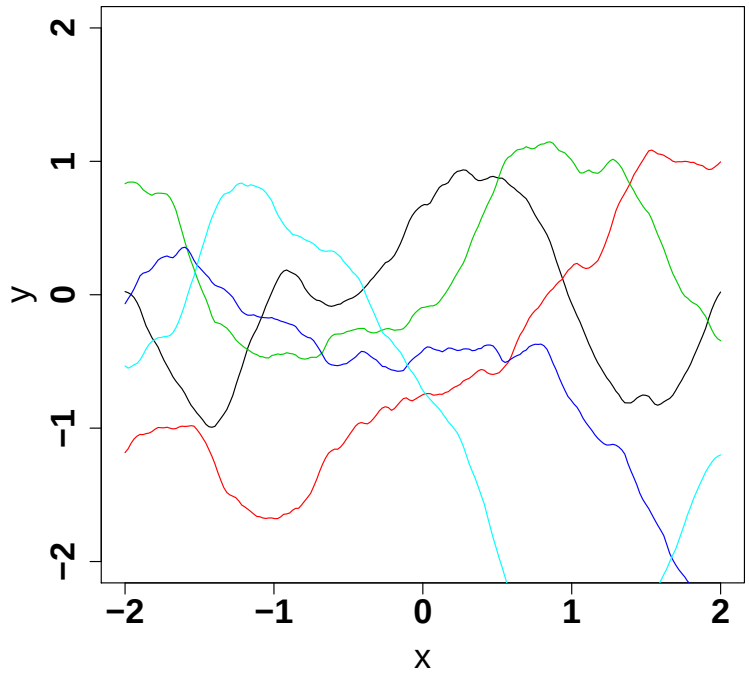
- One Germinal simulation takes around 1 minute
- $n = 3807$ simulation results obtained as follows
  - $LHS-$ maximin (space filling) set of points in $[0, 1]^{11}$
  - We remove the input parameters which are non-viable (e.g. infeasible geometry) (no need for Germinal simulation for this)
- Our set of simulation results is written $y_1 = f_{code}(\boldsymbol{x}^{(1)}), ..., y_n = f_{code}(\boldsymbol{x}^{(n)})$

Metamodeling = construction of an approximation $\hat{f}_{code} : [0, 1]^{11} \to \mathbb{R}$ of f_{code} so that

- $\hat{f}_{code}$ is only based on $\boldsymbol{x}^{(1)}, y_1, ..., \boldsymbol{x}^{(n)}, y_n$ (black box)
- $\hat{f}_{code}$ is much quicker to evaluate than f_{code}

Gaussian process model (Kriging)

Assume that f_{code} is a realization of a random function Y



- Enables to work with $\mathcal{L}\left[Y | Y(\boldsymbol{x}^{(1)}) = f_{code}(\boldsymbol{x}^{(1)}), ..., Y(\boldsymbol{x}^{(n)}) = f_{code}(\boldsymbol{x}^{(n)})\right]$

Covariance function parametrization

- We consider that the Gaussian process is centered, $\forall \boldsymbol{x}, \mathbb{E}(Y(\boldsymbol{x})) = 0$
- The Gaussian process is hence characterized by its covariance function $C(\boldsymbol{x}^{(a)}, \boldsymbol{x}^{(b)}) = \mathrm{Cov}(Y(\boldsymbol{x}^{(a)}), Y(\boldsymbol{x}^{(b)}))$

We assume

$$Y(\boldsymbol{x}) = Y_c(\boldsymbol{x}) + Y_d(\boldsymbol{x})$$

- Y_c : continuous
- Y_d : discontinuous $\longrightarrow$ numerical instabilities

Parametrization :

$$C(\boldsymbol{x}^{(a)}, \boldsymbol{x}^{(b)}) \text{ of the form } \sigma^2 \bar{C}_\ell(\boldsymbol{x}^{(a)}, \boldsymbol{x}^{(b)}) + \delta^2 \mathbf{1}\{\boldsymbol{x}^{(a)} = \boldsymbol{x}^{(b)}\}$$

- σ^2, ℓ variance and correlation lengths
- δ^2 : nugget variance
- Estimates $\hat{\sigma}^2$, $\hat{\ell}$ and $\hat{\delta}$ obtained by Maximum Likelihood (modified because of large n, see the paper)

Prediction

Gaussian process Y observed at $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}$ and predicted at $\boldsymbol{x}$
Let

- $\boldsymbol{y} = (f_{code}(\boldsymbol{x}^{(1)}), ..., f_{code}(\boldsymbol{x}^{(n)}))'$
- $\boldsymbol{R} = \{\hat{\sigma}^2 \bar{C}_{\hat{\ell}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) + \hat{\delta}^2 \mathbf{1}\{\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(j)}\}\}_{1 \leq i,j \leq n}$
- $\boldsymbol{r}(\boldsymbol{x}) = \{\hat{\sigma}^2 \bar{C}_{\hat{\ell}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + \hat{\delta}^2 \mathbf{1}\{\boldsymbol{x}^{(i)} = \boldsymbol{x}\}\}_{1 \leq i \leq n}$

Then, conditionally to $\boldsymbol{y}$, $Y(\boldsymbol{x})$ follows a Gaussian distribution with mean

$$\hat{f}_{code}(\boldsymbol{x}) = \boldsymbol{r}(\boldsymbol{x})^t \mathbf{R}^{-1} \boldsymbol{y}$$

and variance

$$\hat{\sigma}^2_{code}(\boldsymbol{x}) = \hat{\sigma}^2 - \boldsymbol{r}(\boldsymbol{x})^t \mathbf{R}^{-1} \boldsymbol{r}(\boldsymbol{x}) + \hat{\delta}^2$$

- $\hat{f}_{code}(\boldsymbol{x})$: metamodel function
- $\hat{\sigma}^2_{code}(\boldsymbol{x})$: error indicator
- $\hat{\sigma}^2_{code}(\boldsymbol{x}) \geq \hat{\delta}^2$: importance of the nugget variance

Other metamodels and computation time

Two other metamodels

- Kernel methods
  - similar formulas as Gaussian processes but different interpretation
  - no error indicator
- neural networks
  - no support points (construction of an explicit non-linear function)
  - no error indicator

Computation time for prediction at q new points :

- construction phase cost (1 time) + evaluation phase cost (proportional to q)

Costs for Kriging :

- construction phase : few hours
- evaluation phase : 0.004 seconds per input $\boldsymbol{x}$

Costs for neural networks :

- construction phase : few hours
- evaluation phase : 0.00015 seconds per input $\boldsymbol{x}$

Costs for kernel methods :

- no construction phase
- evaluation phase : 0.004 seconds per input $\boldsymbol{x}$

Prediction criteria

Test base $(\boldsymbol{x}_t^{(1)}, f_{code}(\boldsymbol{x}_t^{(1)})), ..., (\boldsymbol{x}_t^{(n_t)}, f_{code}(\boldsymbol{x}_t^{(n_t)}))$, with $n_t = 1613$

- Root Mean Square Error (RMSE) (should be minimal)

$$RMSE^2 = \frac{1}{n_t} \sum_{i=1}^{n_t} \left(\hat{f}_{code}(\boldsymbol{x}_t^{(i)}) - f_{code}(\boldsymbol{x}_t^{(i)}) \right)^2$$

- Efficiency criterion (Q^2) (should be maximal)

$$Q^2 = 1 - \frac{RMSE^2}{sd_{code}^2}$$

  - sd_{code} empirical standard deviation of the test base

- Empirical quantile q_γ for $\gamma \in (0, 1)$: empirical quantile γ of the set of errors $\left| \hat{f}_{code}(\boldsymbol{x}_t^{(i)}) - f_{code}(\boldsymbol{x}_t^{(i)}) \right|$, for $i = 1, ..., n_t$

Estimation of the prediction criteria

- Estimation of RMSE

$$\widehat{RMSE}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{f}_{code}(\boldsymbol{x}^{(i)}) - f_{code}(\boldsymbol{x}^{(i)})\right)^2$$

  - For Gaussian processes and kernel methods, $\tilde{f}_{code}(\boldsymbol{x}^{(i)})$ is obtained by virtual leave-one-out
  - For neural networks $\tilde{f}_{code}(\boldsymbol{x}^{(i)}) = \hat{f}_{code}(\boldsymbol{x}^{(i)})$

- Estimation of Q^2

$$\widehat{Q}^2 = 1 - \frac{\widehat{RMSE}^2}{\widehat{sd}^2_{code}},$$

where $\widehat{sd}_{code}$ is the standard deviation of the output on the learning base.

Prediction results

	$\widehat{RMSE}$	RMSE	$\widehat{Q}^2$	Q^2	$q_{0.9}$	$q_{0.95}$
Neural network	34.5°	38.5°	0.990	0.987	61.6°	76.7°
Kriging	35.6°	36.1°	0.989	0.989	57.4°	72.7°
Kernel methods	44.3°	44.5°	0.983	0.983	68.5°	88.8°

TABLE: Prediction results for the fusion margin output of the Germinal code. The standard deviation of the output on the test base is 342°

- Accuracy ranking : Kriging > neural networks > kernel methods
- For Kriging and kernel methods, $\widehat{RMSE}$ and $\widehat{Q}^2$ are accurate
- For neural networks, $\widehat{RMSE}$ and $\widehat{Q}^2$ are slightly over-optimistic
- $\hat{\delta} = 28.5°$: an important part of the prediction error comes from the numerical instabilities

Plot of the prediction errors

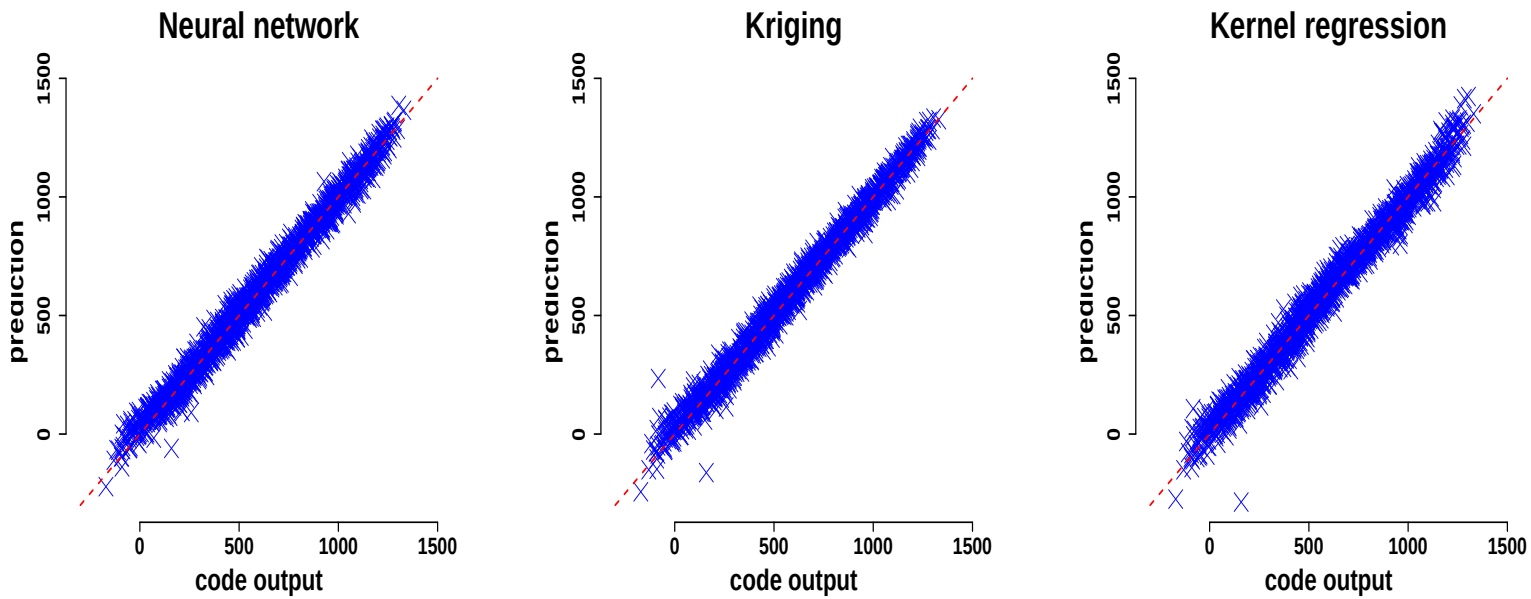


FIGURE: Plot of the metamodel predictions in the test base (y-axis), as a function of the Germinal output values (x-axis), for the neural networks (left), Kriging (middle) and kernel methods (right). The dashed lines is defined by $y = x$.

- There are some very large errors $\Longrightarrow$ outlier computations (possibly because of numerical issues)

Visualization

We have encountered two types of code manager issues :

- outliers (large prediction errors)
- code instabilities (large nugget effect)

We decided to visualize them in $1d$ with points on a line segment of $\mathbb{R}^{11}$:

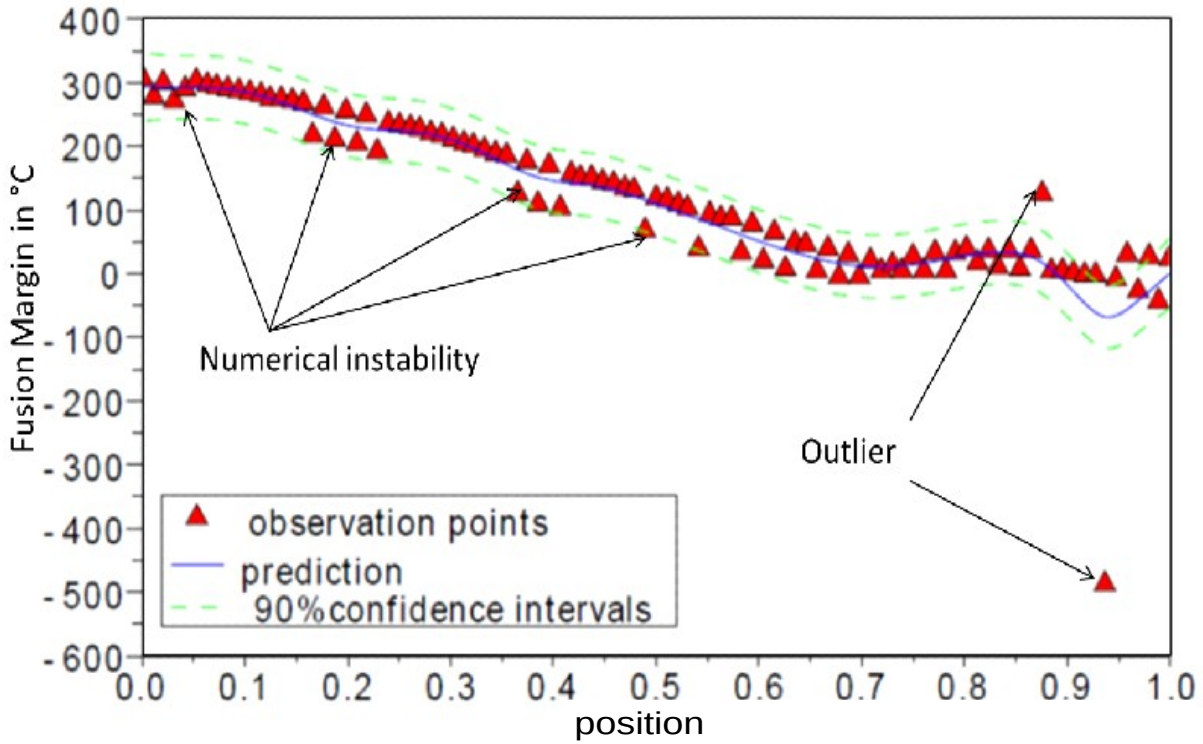


FIGURE: One-dimensional representation of the Germinal code function with 97 points on a line segment. We also plot the Gaussian process prediction with those 97 support points (the nugget estimate seems appropriate)

Improvement of the code manager

- The outlier computations are well flagged by the three metamodels. We can investigate a few outlier computations in details and improve the code manager
- We can investigate successive non-consistent computations from the $1d$ plot and find and solve the following issue :

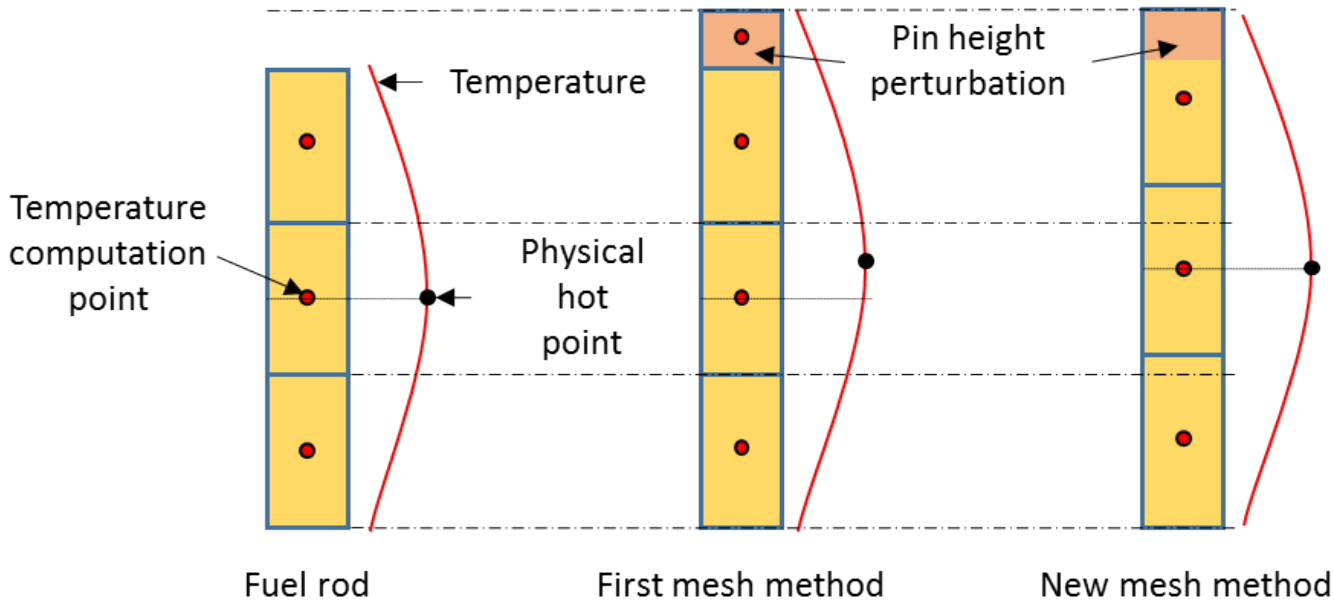


FIGURE: Simplified illustration of the cause of the numerical instabilities. The issue is the automatic mesh generation. We update the code preprocessor to solve it

After the code manager improvement

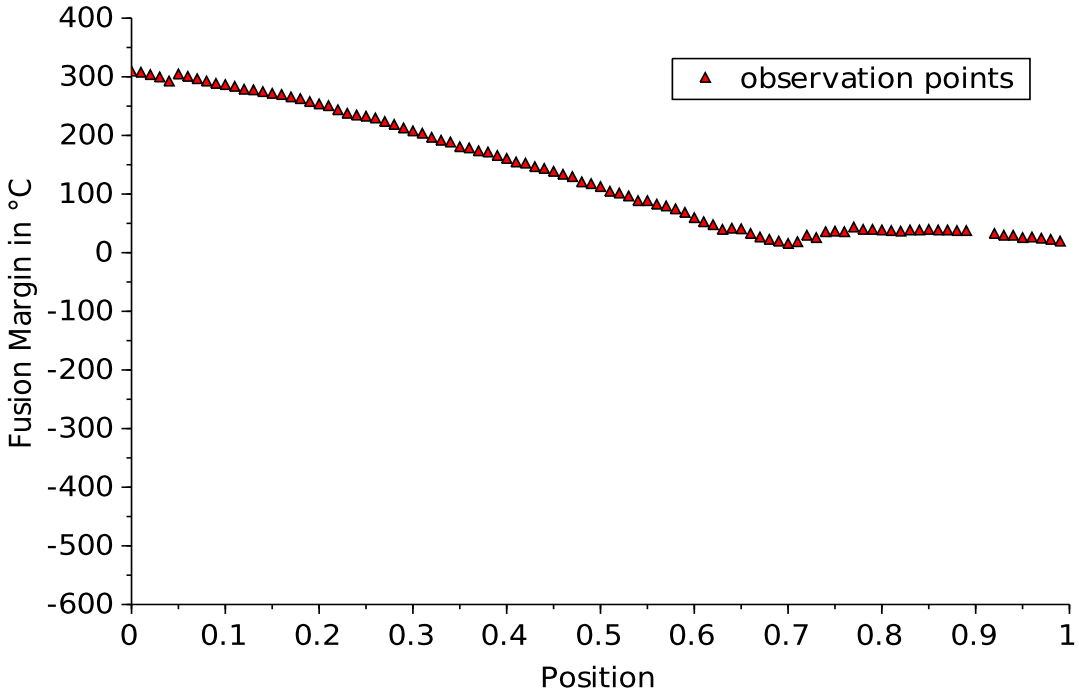


FIGURE: New computations, for the same input points on a line, after update of the code preprocessor

- the outliers are now automatically flagged by the code postprocessor
- the numerical instabilities are removed

New prediction results

	$\widehat{RMSE}$	RMSE	$\hat{Q}^2$	Q^2	$q_{0.9}$	$q_{0.95}$
Neural network	27.5°	31.3°	0.993	0.991	48.7°	63.4°
Kriging	27.2°	27.6°	0.993	0.993	43.2°	54.0°
Kernel methods	38.3°	38.5°	0.986	0.986	60.8°	75.3°

TABLE: Prediction results for the fusion margin output of the updated Germinal code. The standard deviation of the output on the test base is 326.2°

- The predictions are more accurate after the code manager update
- $\hat{\delta} = 19.8°$: smaller than before the code manager update (28.5°)
- The relative differences between the metamodels are larger $\Longrightarrow$ can be explained by the smaller nugget variance
- It is possible that other types of numerical instabilities are present

Conclusion

Recap :

- Comparison of metamodels :
  - Kriging most accurate and provides most uncertainty quantification tools
  - neural networks provide fastest prediction
  - kernel methods do not need a construction phase
- Illustrations of numerical issues in simulations
  - Some simulations can fail completely $\Longrightarrow$ well detected by all metamodels, then manual investigation
  - Groups of simulations can be non-consistent (numerical instabilities) $\Longrightarrow$ so far detected with nugget variance and $1d$ plot
- Addressing these numerical issues improves the accuracy of the metamodels

Qestions :

- Robust Gaussian processes to outlier computations
- Detection procedures for pairs of computations with numerical instabilities
- Asymptotic guarantees for nugget variance estimation, even when the continuous process is misspecified

For more details :

F. Bachoc, K. Ammar and J.M. Martinez "Improvement of code behaviour in a design of experiments by metamodeling", *Nuclear Science and Engineering (2016) 183(3) 387-406*.

Thank you for your attention !