DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

General overview of the URANIE platform

J-B. Blanchard , F. Gaudier
jean-baptiste.blanchard@cea.fr

Summary

In a nutshell

- ROOT
- URANIE

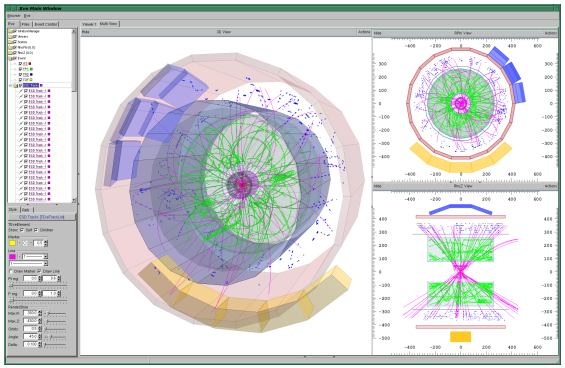
The URANIE purposes

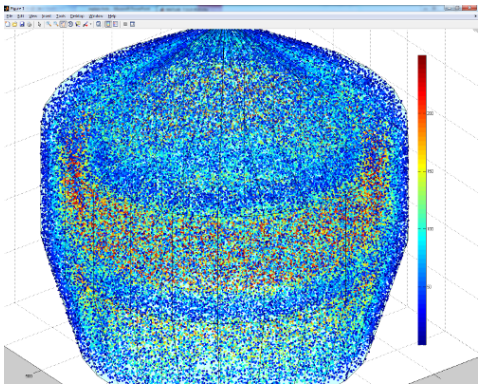
- Gentle reminder on the platform goals
- Internal organisation
- A simple script

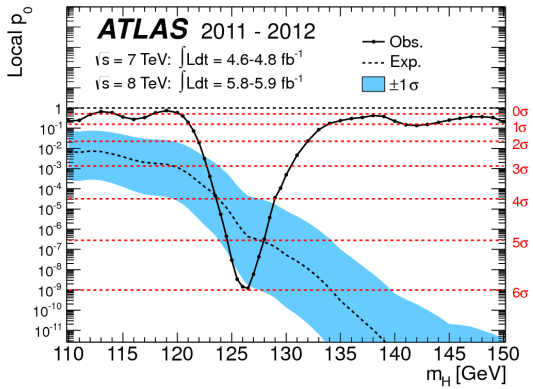
The ROOT platform

Developed at CERN to help analyse the huge amount of data delivered by the successive particle accelerators

- Written in C++ (3/4 releases a year)
- Multi platform (Unix/Windows/Mac OSX)
- Started and maintained over more than 20 years
- It brings:
  - a C++ interpreter, but also Python and Ruby interface
  - a hierarchical object-oriented database (machine independent and highly compressed)
  - advanced visualisation tool (graphics are very important in HEP)
  - statistical analysis tools (*RooStats*, *RooFit*...)
  - and many more (3D object modelling, distributed computing interface...)
- LGPL





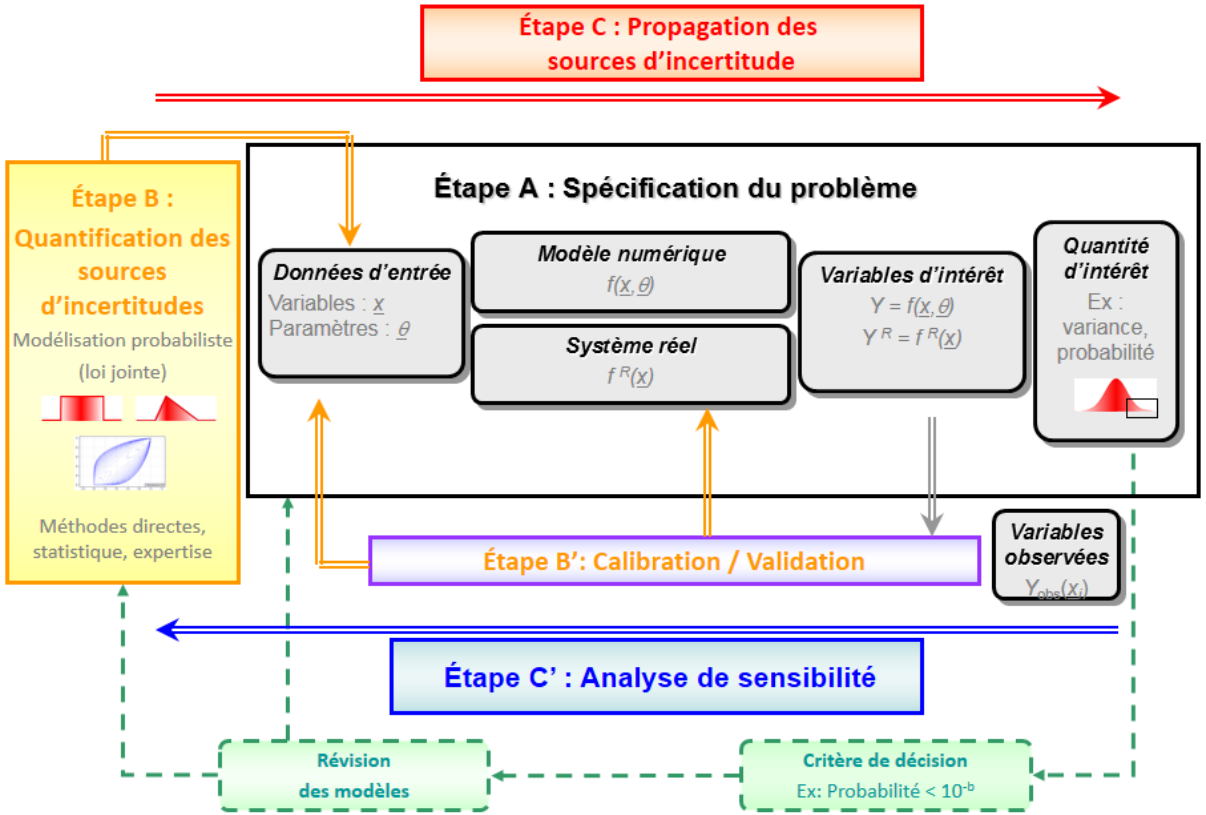


The URANIE platform

Developed at CEA/DEN to help groups handling with sensitivity meta-modelling and optimisation problems.

- Written in C++ (∼2 releases a year), based on ROOT
- Multi platform (developed on Unix and tested on Windows)
- It brings simple data access:
  - Flat ASCII file, XML . . .
  - TTree (internal ROOT format)
  - SQL database access
- Provides advanced visualisation tools (on top of ROOT's one)
- Allows some analysis to be run in parallel through various mechanism
  - simple fork processing
  - using dedicated tool (mpirun)
  - through graphical card (GPU)
- Main purpose is tools for:
  - construction of design-of-experiment
  - uncertainty propagation
  - surrogate models generation
  - sensitivity analysis
  - optimisation problem
  - reliability analysis
- LGPL

Various possible analysis: breakdown into steps



Main steps:

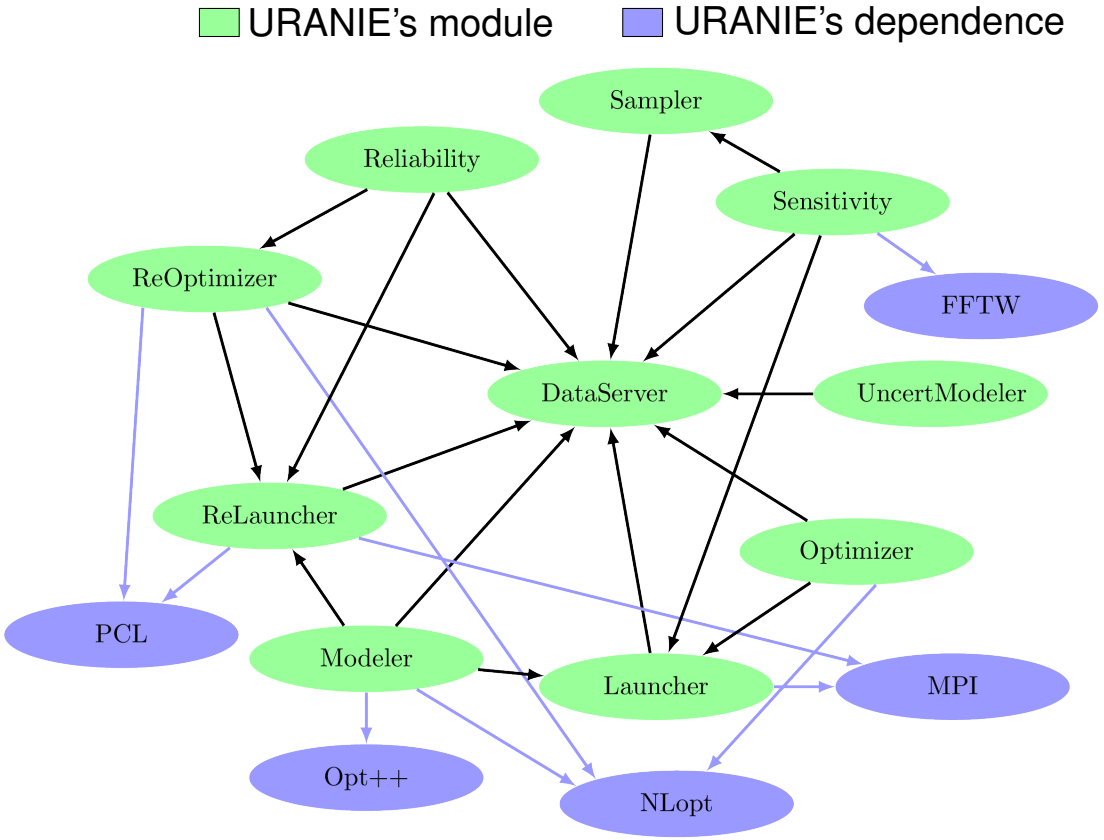
- A: problem definition
  - → Uncertain input variables
  - → Variable/quantity of interest
  - → Model construction
- B: uncertainty sources quantification
  - → Choice of pdfs
  - → Choice of correlations
- C: uncertainty propagation
  - → Evolution of output variability w.r.t input uncertainty
- C': sensitivity analysis
  - → Uncertainty source sorting

These steps are usually model dependent, it might be useful to iterate to help converging to proper conclusions

The URANIE platform

Organised in modules:

- Some are more technical ones:
  - ➙ DataServer: data handling and first statistical treatment
  - ➙ Launcher/ReLauncher: interface to code/functions
- Many are dedicated ones:
  - ➙ Sampler: creation of design-of-experiment
  - ➙ Modeler: surrogate-model generation
  - ➙ Optimizer/Reoptimizer: mono/multi criteria optimisation problems
  - ➙ Sensitivity: input variable sorting w.r.t impact on the output



The next following slides will discuss the content of the main dedicated modules

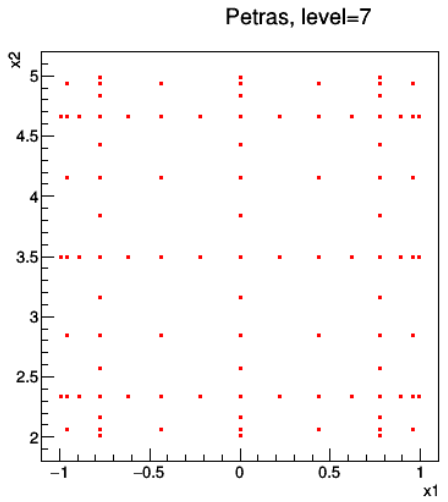
A glimpse at the main modules

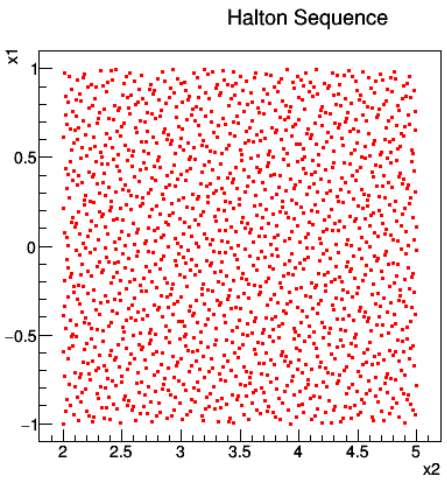
The sampler module

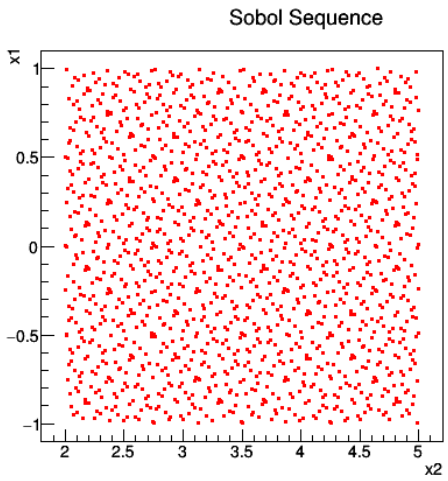
Used to generate the design-of-experiments, basis of many analysis.
Some methods can deal with correlation as well.

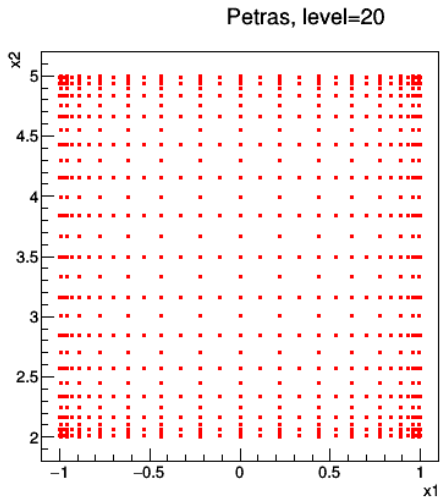
Two main categories

- Stochastic designs:
  - Simple Random Sampling (SRS)
  - Latin Hypercube Sampling (LHS)
  - One-At-a-Time Sampling (OAT)
  - Archimedian copulas
  - Random fields...
- Deterministic designs:
  - Regular quasi Monte-Carlo: Halton/Sobol sequence
  - Sparce grid sampling: Petras
  - Space filling design







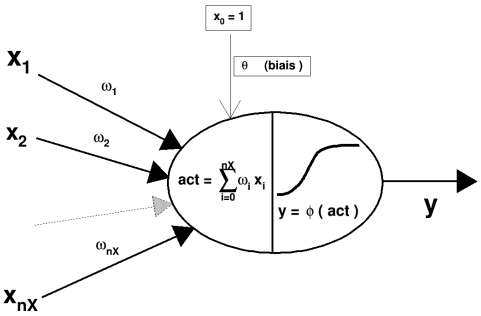


The modeler module

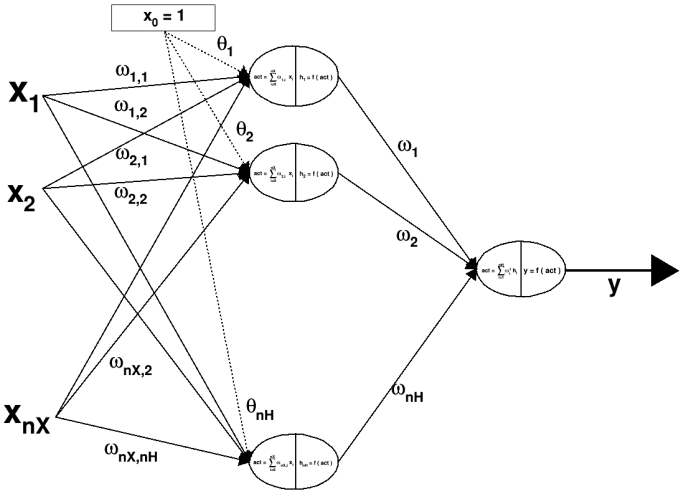
Create a surrogate-model: a numerical model reproducing the behaviour of provided data

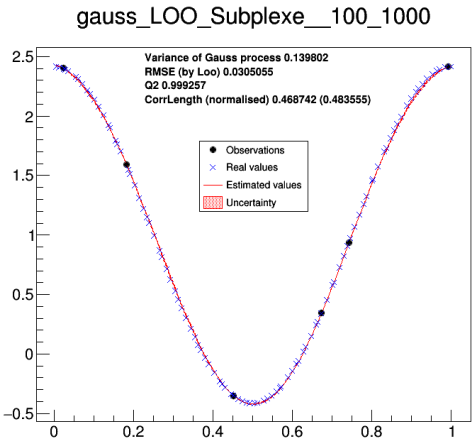
Several possible models to be chosen:

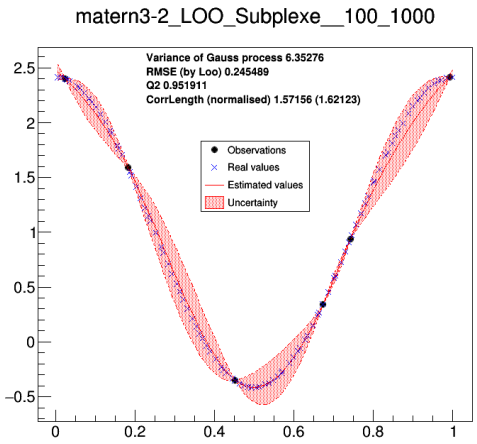
- Polynomial regressions
- Generalised linear models
- Artificial Neural Networks (ANN/MLP)
- Chaos Polynomial + ANISP
- Kriging



➡ Models can be exported in different format (C++, fortran, PMML) in order to be re-used later on.







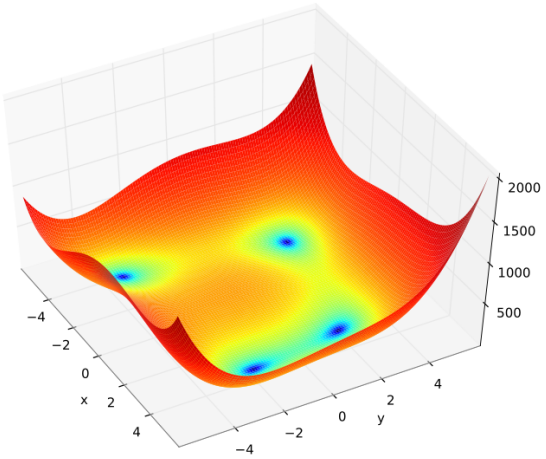
The optimizer module

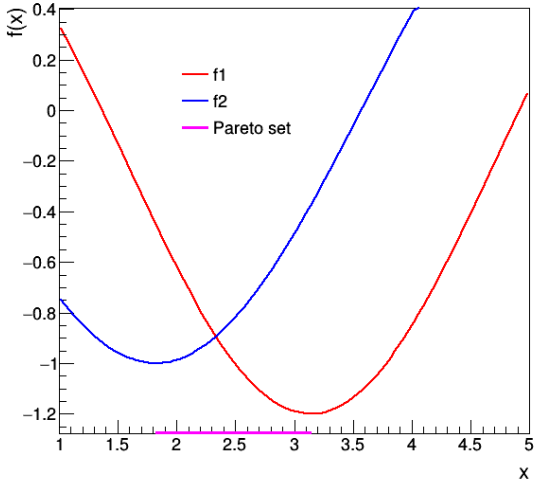
Dealing with optimisation problem usually means:

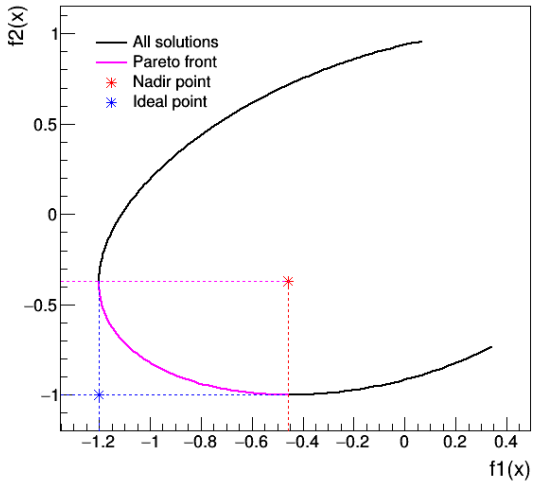
- single objective (SO) or multi objectives (MO) to be minimised
- parameters that have an impact on objective
- possible constraint on these parameters

Many possible implementation for this, based on:

- Minuit: ROOT's SO optimisation library without constraint
- Opt++: SO optimisation library with/without constraint
- NLopt: SO optimisation library with/without constraint
- Vizir: CEA's MO optimisation library with/without constraint, based on stochastic algorithms (*e.g.* genetic algorithms)





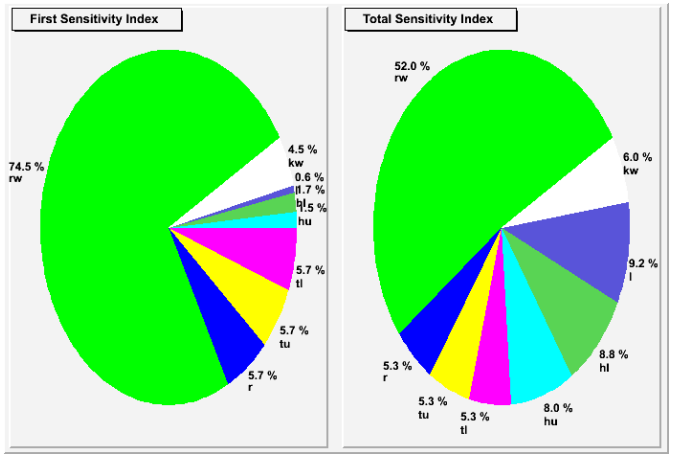


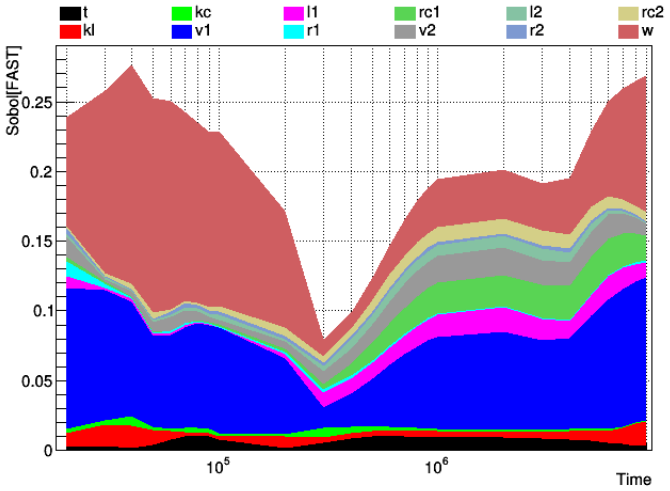
The sensitivity module

Tools to evaluate the sensitivity of the outputs of a code/function to its inputs.

Several kind of methods available:

- Local: finite differences $(\frac{\delta Y_i}{\delta X_j}(x_0))$
- Regression:
  - Pearson (values)
  - Spearman (ranks)
- Screening: OAT, Morris...
- Sobol indexes:
  - FAST (Fourier Amplitude Sensitivity Test)
  - RBD (Random Balance Design)
  - Sobol/Saltelli Methods





Eyes-on: a simple example

emacs@jbpc

File Edit Options Buffers Tools C++ Help

Save Undo

```
void OneKrigingOnly()
{
  /*Reading a database (y vs x1) from a simple function.
  Informations are stored in the TDataServer object*/
  TDataServer *tdsObs = new TDataServer("tdsObs","observations");
  tdsObs->fileDataRead("utf-1D-train.dat");

  /*Defining a kriging model with the training database
   by defining a certain number of options */
  TGPBuilder *gpb = new TGPBuilder(tdsObs,"x1","y","maternII");
  //Find the best possible parameters by optimisation
  gpb->findOptimalParameters("ML", 20, "BFGS", 100);
  //Build the best obtained kriging model.
  TKriging *kg = gpb->buildGP();

  /*Reading now a test basis (constructed with the same dummy function)
  This is mostly for x-check and illustration purposes*/
  TDataServer *tdsEstim = new TDataServer("tdstest","base de test");
  tdsEstim->fileDataRead("utf-1D-test.dat");

  //Applying the kriging on the test basis => launching the model on every points
  TLauncher2 *lkrig = new TLauncher2(tdsEstim, kg, "x1", "yEstim:vEstim");
  lkrig->solverLoop();

  //Plotting the results
  gROOT->LoadMacro("PlottingKriging.C");
  PlottingKriging(tdsObs, tdsEstim);

}
```

-:--- OneKrigingOnly.C All L1 (C++/l Abbrev)

Loading cc-langs...done

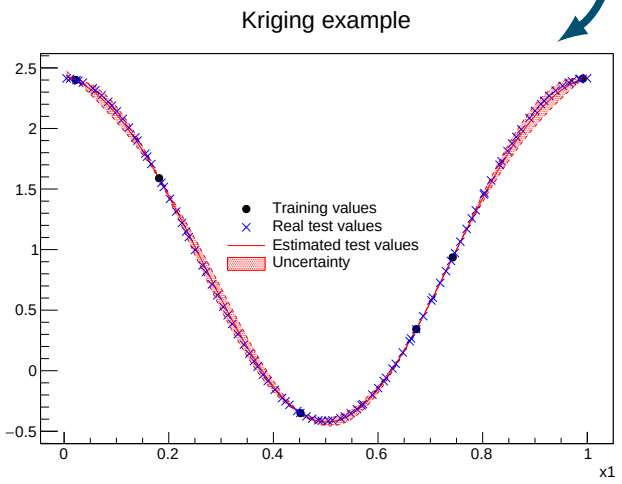
called by

```
(mar.10mai 0:37)-(blanchard@jbpc:...ipts/modeler)-(1|%) root OneKrigingOnly.C
root [0]
Processing /home/blanchard/.root_logon...
Processing OneKrigingOnly.C...

--- Uranie v3.7/0 --- Developed with ROOT (5.34/23) by Fabrice Gaudier
                      Copyright (C) 2013  CEA/DEN
                      Version : v3.7/0 -  Date : Thu Jul 23, 2015
                      All rights reserved, please read http://root.cern.ch/

TGPBuilder::findOptimalParameters: starting screening procedure (20 evaluations)
TGPBuilder::findOptimalParameters: starting optimisation procedure (BFGS algorithm
!,,,,;,,,,!,,,,
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2]
```

gives



The best is yet to come

ROOT

- Reference guide: https://root.cern.ch/guides/reference-guide
- User guide and manual: https://root.cern.ch/root-user-guides-and-manuals
- How-to: https://root.cern.ch/howtos

URANIE as up to now

- HTML documentation and examples embedded with the sources
- Doxygen documentation available also if compiled by the user

URANIE in few months

- New forge will be created
  - → Make available the latest sources
  - → Create an How-to with up-to-date examples
  - → Re-organise the full documentation, making it available in both web/pdf format.
- New release will be done with large modifications
  - → Going to next ROOT release
  - → Giving more data handling flexibility (more complex objects available)

You're more than welcome to give it a try !

Commissariat à l'énergie atomique et aux énergies alternatives
Centre de Saclay | 91191 Gif-sur-Yvette Cedex
T. +33 (0)1 69 08 73 20 | F. +33 (0)1 69 08 68 86

DEN/DANS
DM2S
STMF

Etablissement public à caractère industriel et commercial | R.C.S Paris B 775 685 019