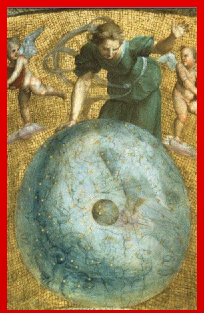


Uranie

Plateforme Incertitude & Optimisation

F. Gaudier, G. Arnaud, J.M. Martinez

fabrice.gaudier@cea.fr
gilles.arnaud@cea.fr
jean-marc.martinez@cea.fr

Atelier ESNT

Orme des merisiers, Saclay

13 Février 2014

Outline

- Introduce the ROOT framework
- The Uranie platform
- Applications :
  - Launcher
  - Optimisation

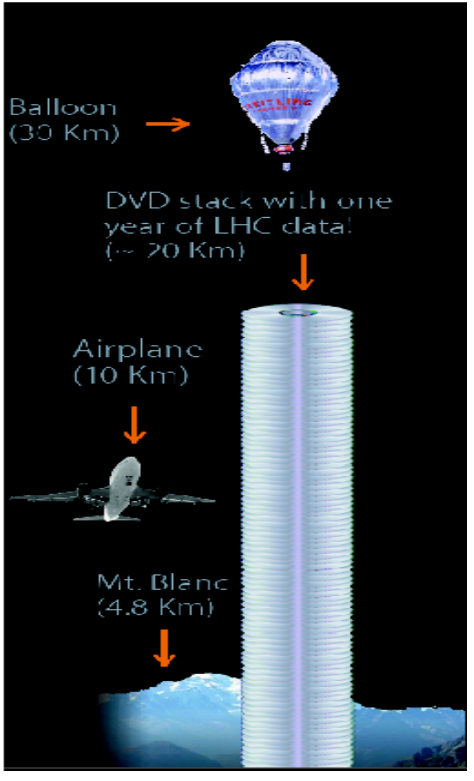
CERN Large Hadron Collider (LHC)

- particle accelerator
- 27 km circumference tunnel in Geneva
- 4 experiments (ATLAS, CMS, ALICE, LHCb)

  Study the structure of matter

  – Search for the Higgs boson

  – Search for new physics
- Data quantity generated : 20 PetaBytes/year
- ROOT is the framework to store, treat and analyze this data

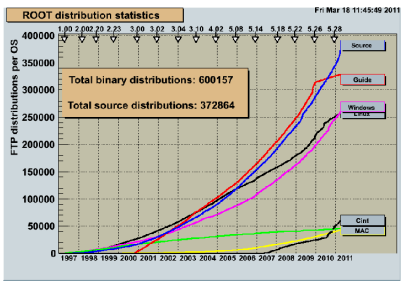


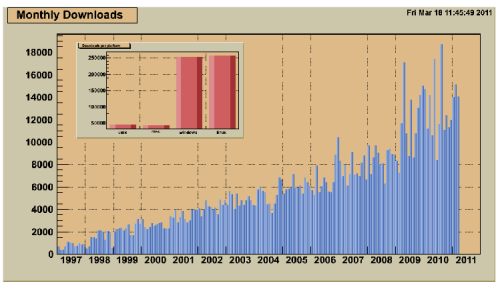
Fons Rademakers / CERN

ROOT (http://root.cern.ch)

ROOT is an object-oriented framework for large scale data analysis and data mining.

- 20 years of development (C++ with 3-4 releases/year)
- multi-plateform (Unix, Windows, Mac OS X)
- Offer :
  - A C++ interpreter, but also python (PyROOT), ruby
  - A hierarchical object-oriented database (machine independent, highly compressed, supporting schema evolution and object versioning)
  - Shared librairies (*automatic loading with "rootmap"*)
  - Advanced statistical analysis tools (subprojects *RooStats*, *RooFit*, *TMVA*)
  - Advanced visualization tools
- **LGPL License**

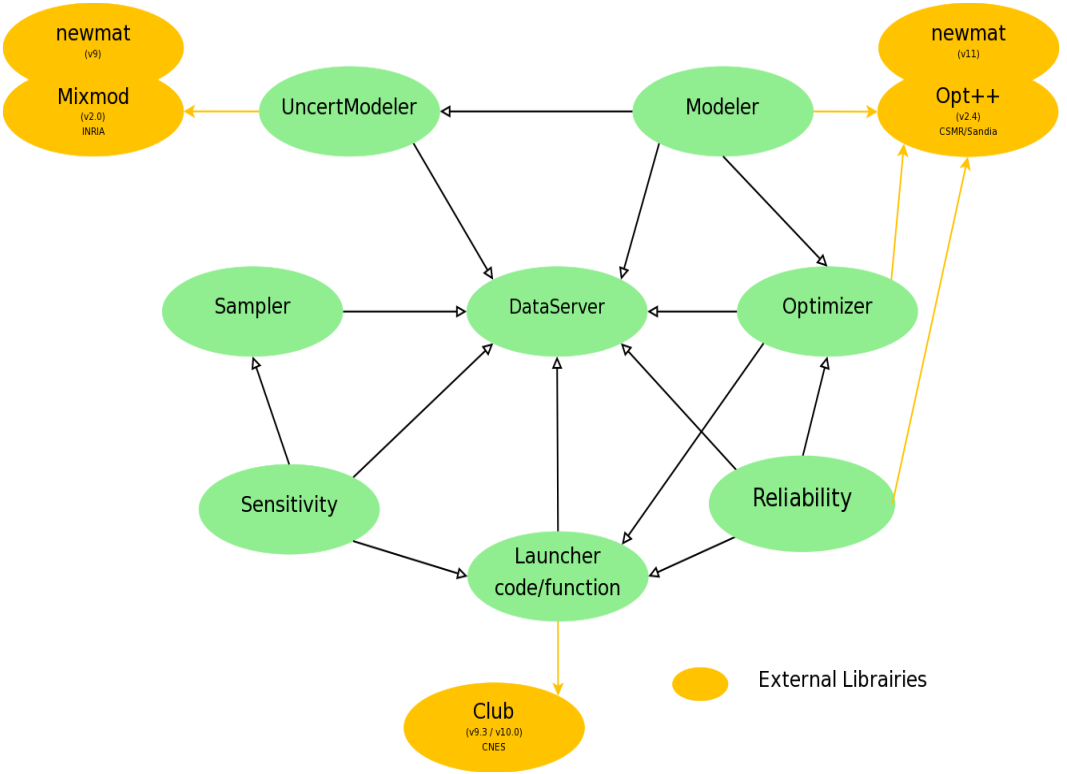


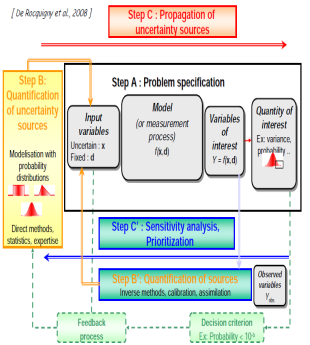


URANIE : CEA/DEN Uncertainty Platform

-  ROOT (CERN),  MIXMOD (Gaussian Mixtures - INRIA),  OPT++ (Optimization - Sandia)
- Data access :
  - Flat file with header ("Salomé Table")
  - TTree (internal ROOT)
  - SQL Data base (MySQL, PostgreSQL, ...)
- Uncertainty/Sensitivity/Optimisation methods in URANIE
  - Design Of Experiments (SRS, LHS, ROA, qMC, MCMC, Copulas)
  - Clustering methods
  - Surrogate models (Polynomial, Artificial Neural Networks, Kriging, GLM)
  - Non Intrusive Spectrale Projection : Generalized Polynomial Chaos
  - Inverse Quantification of Uncertainty (CIRCE)
  - Sensitivity Analysis (Local, Morris, Regressions (*Pearson, Spearmann*), Sobol, FAST & RBD)
  - Optimization, Multi-Criteria (**Vizir** library : Genetic Algorithms)
  - Computing distribution (**HPC** : TGCC, CCRT)

URANIE : Fonctional diagram





URANIE : Batch mode

> root myScript.C

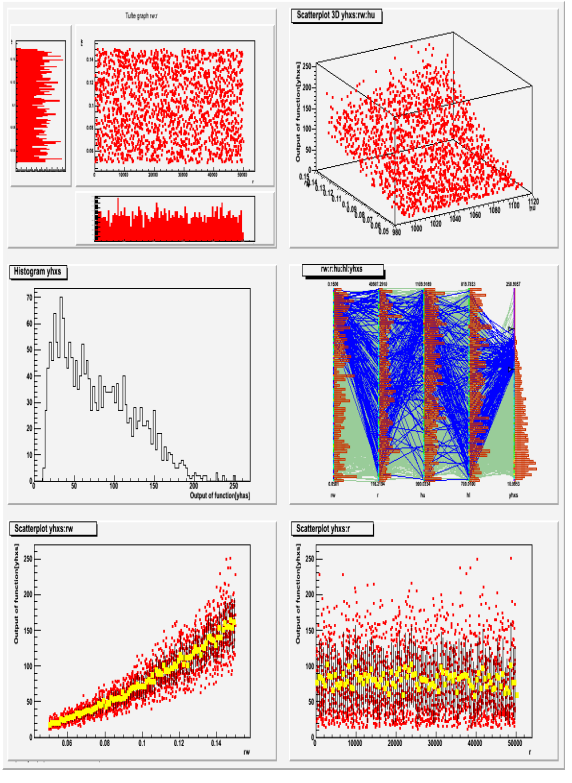
emacs: myScript.C

File Edit View Cmds Tools Options Buffers C++ Help

myScript.C

```
{
  // Create the TDataServer of the study
  TDataServer * tds = new TDataServer("tdsBorehole","Launch the Borehole function");
  // Add the eight attributes of the study with uniform law
  tds->addAttribute( new TUniformDistribution("rw", 0.05, 0.15));
  tds->addAttribute( new TUniformDistribution("r", 100.0, 50000.0));
  tds->addAttribute( new TUniformDistribution("tu", 63070.0, 115600.0));
  tds->addAttribute( new TUniformDistribution("tl", 63.1, 116.0));
  tds->addAttribute( new TUniformDistribution("hu", 990.0, 1110.0));
  tds->addAttribute( new TUniformDistribution("hl", 700.0, 820.0));
  tds->addAttribute( new TUniformDistribution("l", 1120.0, 1680.0));
  tds->addAttribute( new TUniformDistribution("kw", 9855.0, 12045.0));

  // Generate the sampling from the TDataServer (LHS, 1000)
  TSampling *samplhs = new TSampling(tds, "lhs", 1000);
  samplhs->generateSample();
  tds->exportData("waterhole_sampler_lhs.dat");

  // Load the analytical functions
  gROOT->LoadMacro("UserFunctions.C");

  // Evaluate the flowrateModel analytical function
  TLauncherFunction * tlf = new TLauncherFunction(tds, flowrateModel,"","ymod");
  tlf->run();
  tds->exportData("waterholelhs.dat");

  // Clean the TDS
  tds->delAttribute("ymod");
  delete tds->getTuple();

  // Generate the sampling from the TDataServer (SRS, 2000)
  TSampling *sampsrs = new TSampling(tds, "srs", 2000);
  sampsrs->generateSample();

  // Evaluate the HoXuSurrogateModel function
  TLauncherFunction * tlfhxs = new TLauncherFunction(tds, HoXuSurrogateModel,"rw:r:tu:tl:hu:hl:l:kw","yhxs");
  tlfhxs->run();

  // Visualisation
  TCanvas  *Canvas = new TCanvas("c1", "Graph for the Macro launchFunctionSampling",8,118,989,830);
  Canvas->Divide(2,3);
  Canvas->cd(1); tds->drawTufte("rw:r","","same");
  Canvas->cd(2); tds->draw("yhxs:rw:hu");
  Canvas->cd(3); tds->draw("yhxs");
  Canvas->cd(4); tds->draw("rw:r:hu:hl:yhxs","","para");
    TParallelCoord* para = (TParallelCoord*)gPad->GetListOfPrimitives()->FindObject("ParaCoord");
    TParallelCoordVar* axis = (TParallelCoordVar*)para->GetVarList()->FindObject("yhxs");
    axis->AddRange(new TParallelCoordRange(axis,150.0,200.0));
    para->AddSelection("blue");
    para->GetCurrentSelection()->SetLineColor(kBlue);
  Canvas->cd(5); tds->drawProfile("yhxs:rw","","same");
  Canvas->cd(6); tds->drawProfile("yhxs:r","","same");

  Canvas->SaveAs("uranieScriptGraph.png");
}
```

Raw-----XEmacs: myScript.C //:24 (C++ Font Abbrev)----L18--C2--All----------



URANIE : XML User Interface

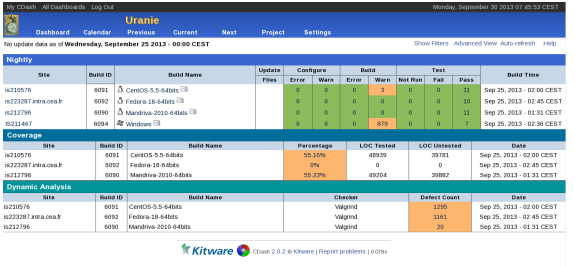
```
1<?xml version="1.0" encoding="iso-8859-1"?>
2<!DOCTYPE Problem SYSTEM "/uranie.dtd">
3<Problem>
4  <Header name="boreholeXML" title="Launch the Borehole function in XML" debug="0">
5    <Application name="uranie" version="1.0"/>
6  </Header>
7  <DataDictionary>
8    <DataField name="rw" law="uniform" min="0.05" max="0.15"/>
9    <DataField name="r"  law="uniform" min="100.0" max="50000.0"/>
10   <DataField name="tu" law="uniform" min="63070.0" max="115600.0"/>
11   <DataField name="tl" law="uniform" min="63.1" max="116.0"/>
12   <DataField name="hu" law="uniform" min="990.0" max="1110.0"/>
13   <DataField name="hl" law="uniform" min="700.0" max="820.0"/>
14   <DataField name="l"  law="uniform" min="1120.0" max="1680.0"/>
15   <DataField name="kw" law="uniform" min="9855.0" max="12045.0"/>
16 </DataDictionary>
17 <Sampler method="LHS" N="1000" export="waterhole_sampler_lhs.dat"/>
18 <Launcher macro="UserFunctions.C" function="flowrateModel" output="ymod" export="waterholelhs.dat"/>
19 <Sampler method="SRS" N="2000"/>
20 <Launcher function="HoXuSurrogateModel" input="rw:r:tu:tl:hu:hl:l:kw" output="yhxs"/>
21</Problem>
```

```
void evaluateXMLFile (TString xmlFile = "uranieproblem.xml")
{
TXMLProblem * xmlProblem = new TXMLProblem(xmlFile);
xmlProblem->submit();
}
```

The URANIE project : v3.4 - 2013/12

- **110 000** lines & **234** classes
- Version of ROOT :
  v5.34.13 (2013 Nov.)
  v5.32 (2011 Dec.)
- Compilation with **cmake**
  (Linux-Makefiles/Windows-Visual Project)
- Unitary tests with **CppUnit**
- Coverage with **gcov** in **CDash**
- Memory check with **valgrind**
- **Exceptions** (Warning, Error, Deprecated)
- Open source since 2013/05

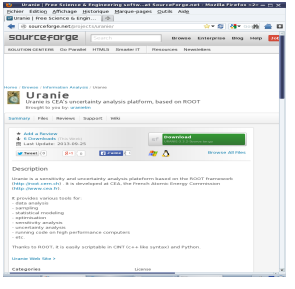
http://sourceforge.net/projects/uranie



CDash reports

```
--- <WARNING> URANIE::MessageLogger : ==================== WARNING ======================
--- <WARNING> URANIE::MessageLogger : Deppreciated constructor since v0.3 to v0.5
--- <WARNING> URANIE::MessageLogger : Using the same constructor with a TDataServer object
--- <WARNING> URANIE::MessageLogger : ================= End Of Warning ==================
```

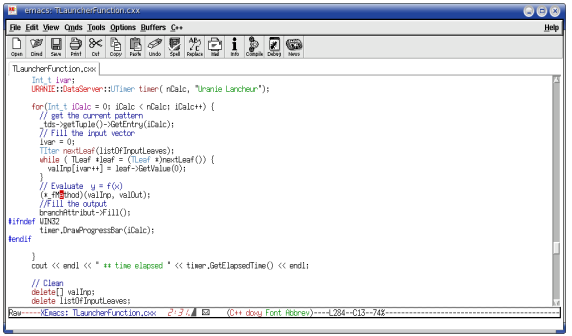
Deprecated message

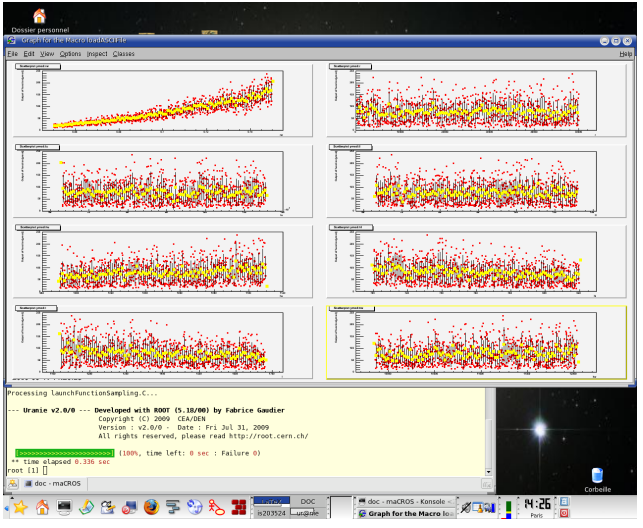


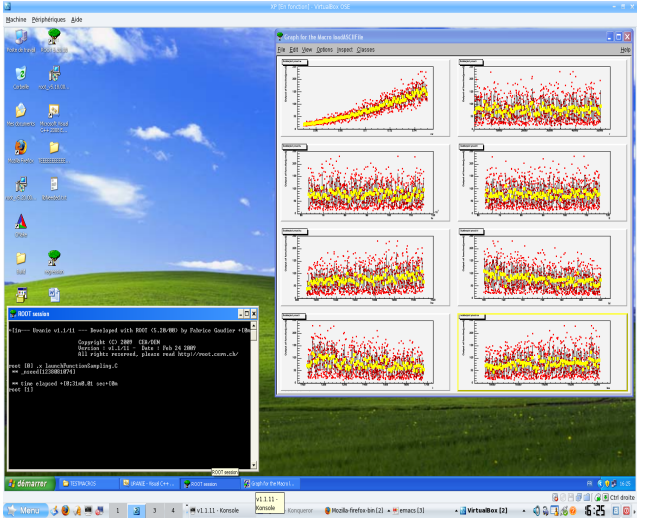
Uranie SourceForge site

The URANIE project : Windows

- Configure with **cmake**
- Compilation with **Visual C++** (**free edition**)
- **with <u>very few</u>** `"#ifdef WIN32"`
- Same Macro on Linux and Windows





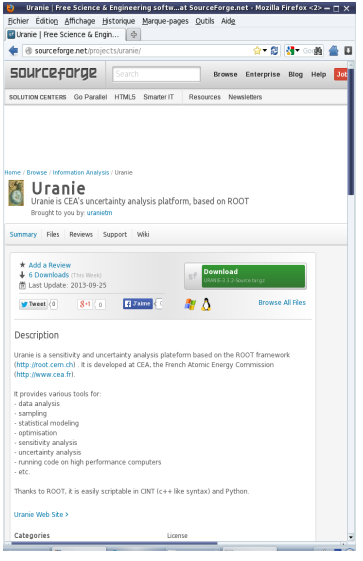


Projects using URANIE

- LEONAR tool for severe accidents in french nuclear reactor (**CEA-EDF Septen**)
- PSI-Matador Methodology : Dosimetry computation in french nuclear reactor (**CEA-EDF**)
- SIVIT project : (**CEA DTCD**)
  - ◇ GUI Salomé/Uranie on Windows
- EHPOC project : Meteor/Pleiades codes (**CEA DEC**)
- Multi-criteria optimization (**CEA CESTA/CELIA**)
- Sensitivity Analysis for ThermalHydraulic codes Flica/Cathare (**Areva TA**)
- ALLIANCES platform (**CEA/ANDRA/EDF**)
  - ◇ is to provide a working environment for the simulation and analysis of phenomena to be taken into account for waste storage and disposal studies
- European project **NURESIM/NURISP**
  - ◇ The European Platform for NUclear REactor SIMulations, NURESIM, is a Common European Standard Software Platform for modeling, recording, and recovering computer data for nuclear reactors simulations

The URANIE project : Training

- Uranie Training sessions
  - 2013/10/14-15 : **"Optimisation user"** session - SAC Areva-TA (2)
  - 2013/09/16-18 : **"user"** session - CAD CEA/DEN (12)
  - 2013/04/2-4 : **"user"** session - SAC NURESAFE (12)
  - 2012/10/14-17 : **"user"** session - SAC (12)
  - 2012/04/11-13 : **"user"** session - CAD CEA/DEN-DSM (12)
  - 2011/10/17-19 : **"user"** session - SAC CEA/DEN (8)
  - 2011/04/4-6 : **"user"** session - NURISP (5)
  - 2011/01/19-21 : **"user"** session - SAC CEA/DEN (12)
  - ...
  - 2008-2009 : Areva-TA ($\sim$ 10)
  - 2006-2008 : CEA/DEN ($\sim$ 50 persons)
- ROOT Training sessions at INSTN (12 students, 3/4 days)
  - February 2011 : **DEN-SAC**
  - February 2010 : **DEN-SAC**
- Developpment is piloted by user request
- bugtracker
- Distribution : Open source since 2013/05

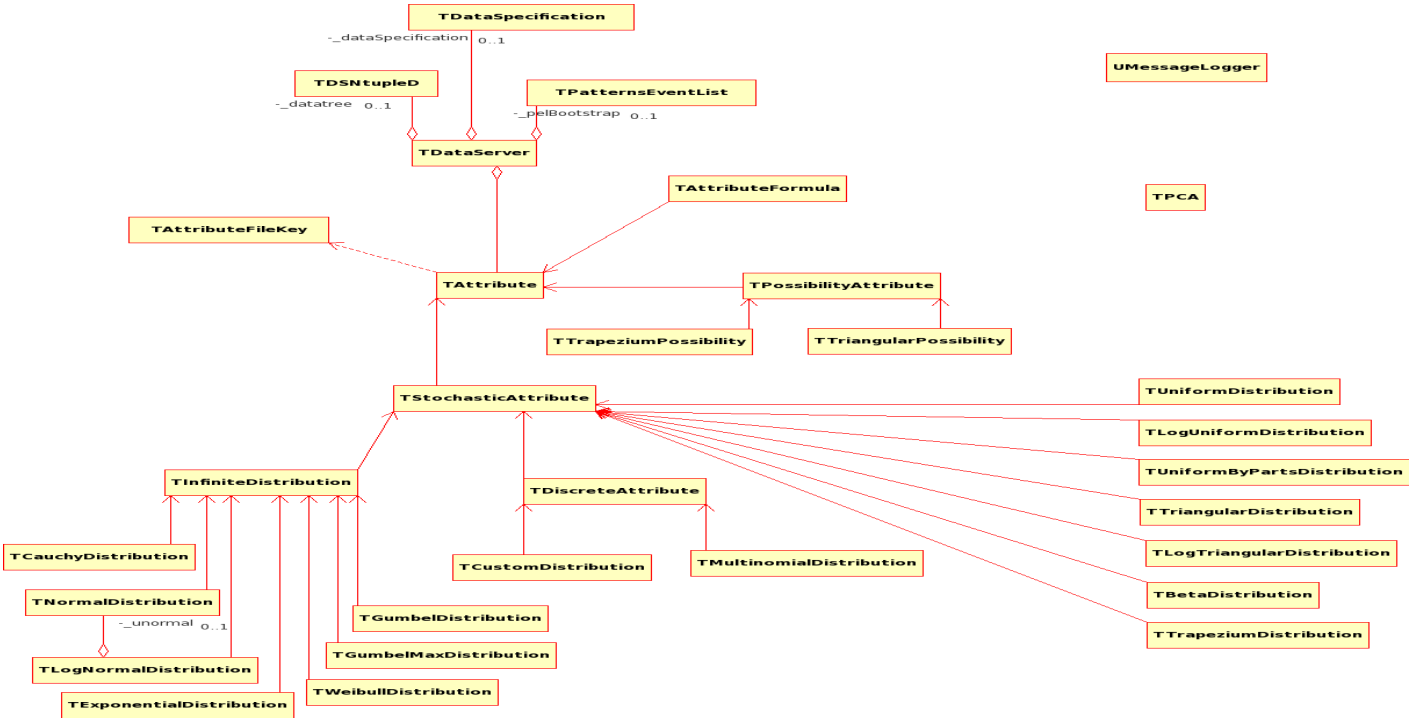


http://sourceforge.net/projects/uranie

"DataServer" library

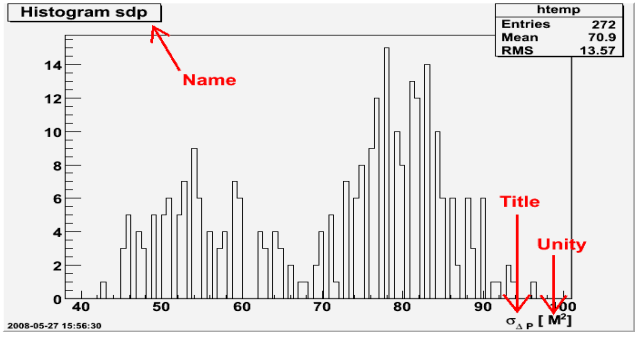
Management of the attributes (∼ variables)
- create/transform attributes
- Load data from external files/formats (ASCII,TTREE,SQL)

Graphs and treatments specific to uncertainties



"DataServer" library - URANIE ASCII file format

```
#NAME: geyser
#TITLE: geyser data
#DATE: Mon Mar 12 23:41:09 2007
#COLUMN_NAMES: x1| sdp
#COLUMN_TITLES: x_1| #sigma_{#Delta P}
#COLUMN_UNITS: Sec| M^{2}
------ empty line ------
3.600 79.000
1.800 54.000
...
```

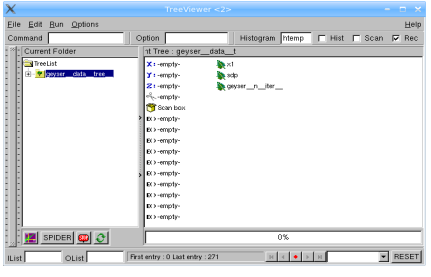


tds->draw("sdp");

Only the "#COLUMN_NAMES:" line is mandatory

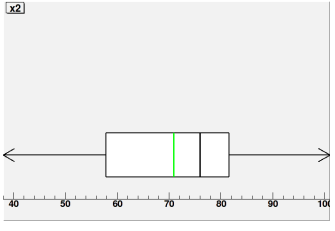
WARNING : the empty line between the header and the matrix data

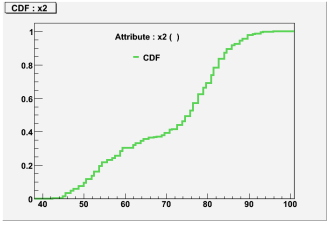
```
TDataServer *tds = new TDataServer();
tds->fileDataRead("geyser.dat");
tds->draw("sdp");
tds->startViewer();
```

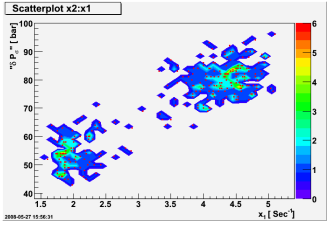


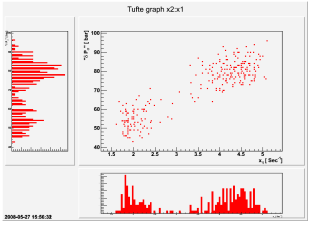
"DataServer" library - statistical graphs

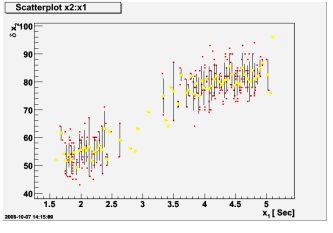
```
tds->drawBoxPlot("x2");
tds->drawCDF("x2","x1<3.0");
tds->drawScatterplot("x2:x1");
tds->drawTufte("x2:x1");
tds->drawProfile("x2:x1","","same");
tds->drawPairs();
```

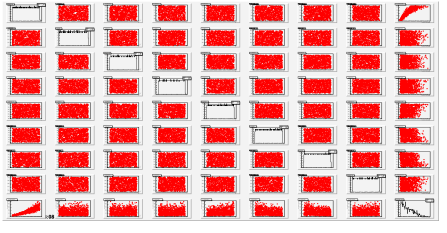








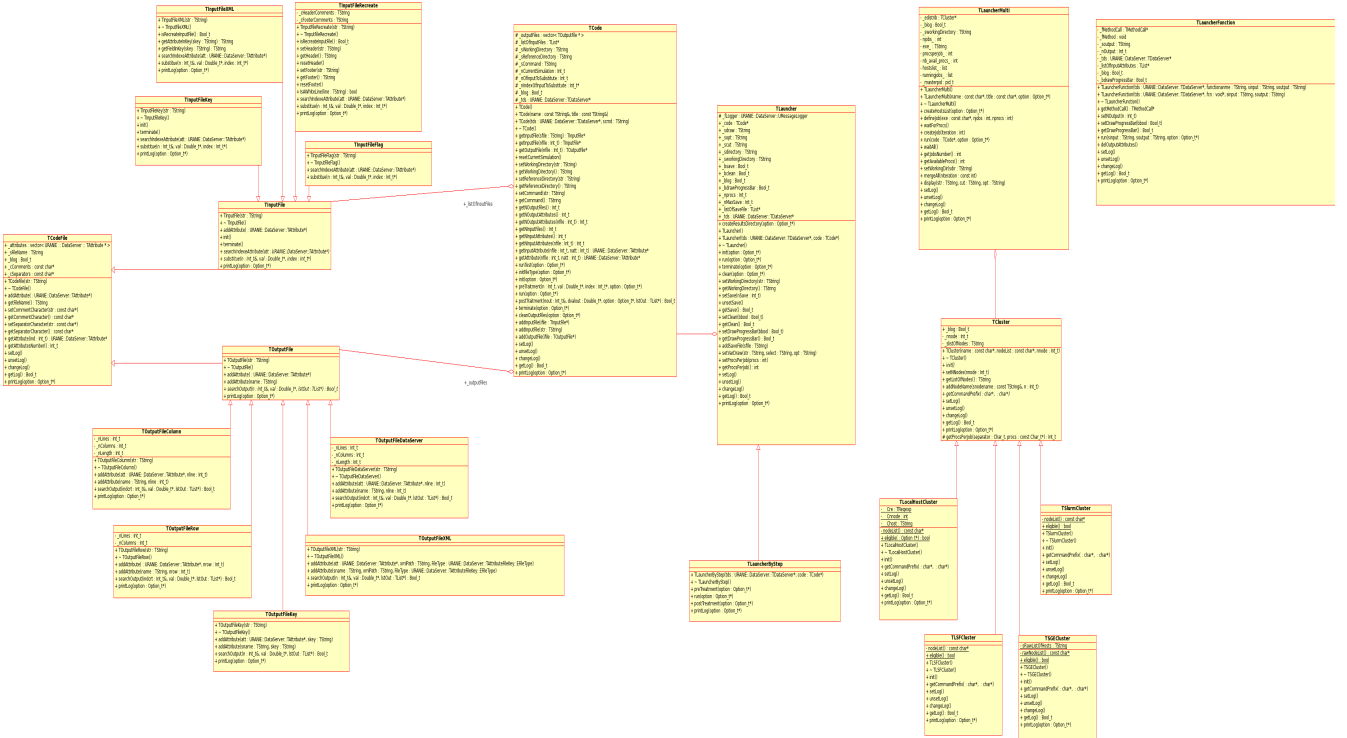




"Launcher" library

Distribute the model evaluations (sequential, cluster) for:

- Analytical function
- External code



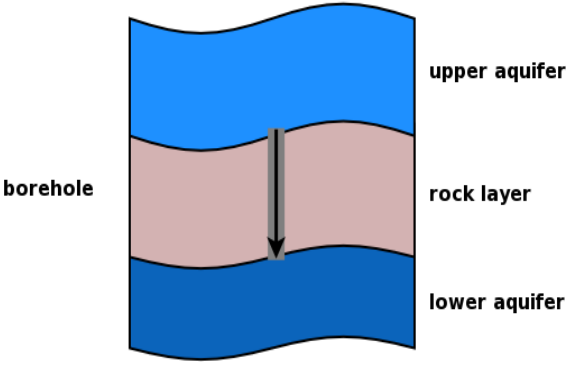
Analytical Function

void myFunction (double *x, double *y)

Analytical function of R^8 in R :

(Bernoulli's law and flow is steady-state laminar and isothermal)

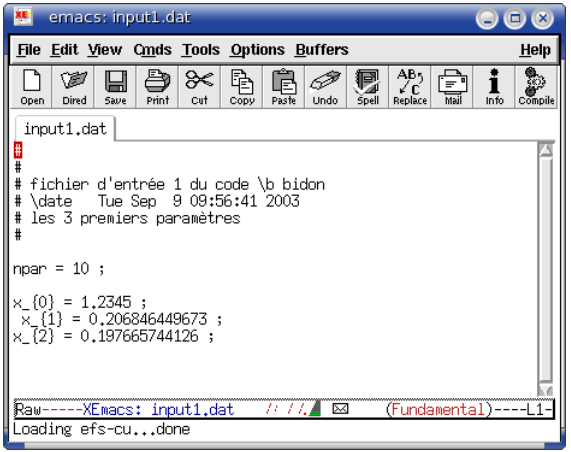
$$y = \frac{2\pi T_u \left(H_u - H_l\right)}{\ln(\frac{r}{r_\omega}) \left[1 + \frac{2LT_u}{\ln(\frac{r}{r_\omega}) r_\omega^2 K_\omega} + \frac{T_u}{T_l}\right]}$$



Fang, p 35

```
void flowrateModel(double *x, double *y)
{
  Double_t drw = x[0], dr  = x[1];
  Double_t dtu = x[2], dtl = x[3];
  Double_t dhu = x[4], dhl = x[5];
  Double_t dl  = x[6], dkw = x[7];

  Double_t dnum = 2.0 * TMath::Pi() * dtu * ( dhu -dhl);
  Double_t dlnronrw = TMath::Log( dr / drw);
  Double_t dden = dlnronrw * ( 1.0 +  ( 2.0 * dl * dtu ) / ( dlnronrw * drw * drw * dkw) + dtu / dtl );

  y[0] = dnum / dden;
}
```

External code : Input Files with " Key - Value"

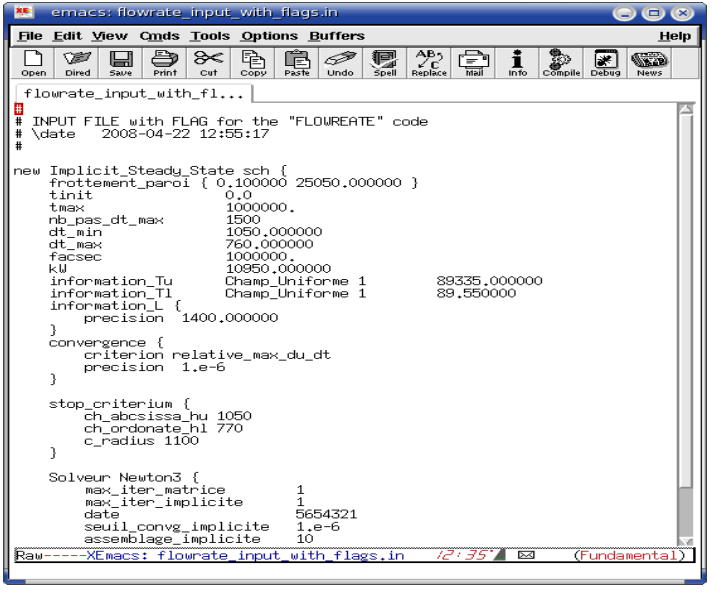



```
TAttribute *x1 = new TAttribute("x_{1}", 0.20, 0.40);
x1->setFileKey("input1.dat");

TAttribute *x2 = new TAttribute("x2", 0.15, 0.25);
x2->setFileKey("input1.dat", "x_{2}");
```

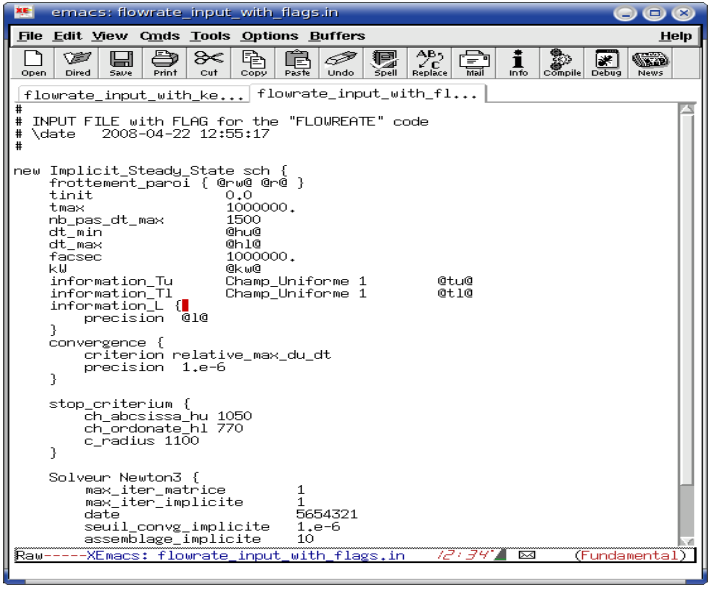
Hypothesis : unicity of the key

```
void TAttribute::setFileKey(
    TString sfile,
    TString skey="",
    TString sformatToSubstitute="%e",
    TAttributeFileKey::EFileType FileType=TAttributeFileKey::kKey);
```

External code : Input Files with "flag"



Original file



User Flag file

```
attrw->setFileFlag("myfile.in", "@tu@");
attrw->setFileKey("myfile.in", "@tu@", "%f",TAttributeFileKey::kFlag);
```

Hypothesis : unicity of the key not required but intervention of the user

```
[file ../../flowrate_input_with_xml.xml does not exist]
...
```

```
attrw->setFileXMLAttribute("input.xml", "wall_friction/@rw");
atthu->setFileXMLField("input.xml", "parameter[tonode='mesh and toport='dt_hu']/value/double");
```

Hypothesis : unicity of the key not required and no intervention of the user

- **TAttributeFileKey::kNewRow** 4^{th} argument in the **setFileKey** method

```
2.481733e+02 6.112975e-03 1.055352e-06 2.635758e-03 2.217372e+02 1.888999e+00 ...
```

- **TAttributeFileKey::kNewColumn** 4^{th} argument in the **setFileKey** method

```
2.481733e+02
6.112975e-03
...
```

- **TAttributeFileKey::kNewKey** 4^{th} argument in the **setFileKey** method

```
t = 2.481733e+02 ;
kl = 6.112975e-03 ;
kc = 1.055352e-06 ;
...
```

Hypothesis : The input files does not exist

- **TOutputFileRow** class

```
0.20E+05 0.5579978E-25 0.2789989E-25
0.30E+05 0.5121863E-20 0.2560931E-20
0.40E+05 0.8212720E-17 0.4106360E-17
0.50E+05 0.1432418E-14 0.7162090E-15
...
```

- **TOutputFileColumn** class

```
0.20E+05 0.30E+05 0.40E+05 0.50E+05 ....
0.5579978E-25 0.5121863E-20 0.8212720E-17 0.1432418E-14 ...
0.2789989E-25 0.2560931E-20 0.4106360E-17 0.7162090E-15 ...
...
```

- **TOutputFileKey** class

```
yhat = 3.591931e+01;
d = 2.415401e+03;
...
```

- **TOutputFileDataServer** class

```
#COLUMN_NAMES: yhat | d

3.591931e+01 2.415401e+03
...
```

Distribution

- PC multicores

```
launcher->run("localhost=5");
```

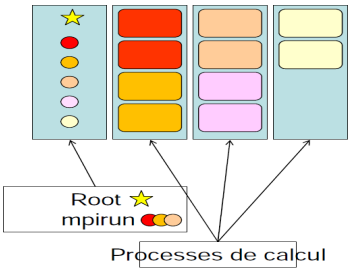
- Cluster (SLURM, LSF, SGE)

```
#BSUB -n 10
#BSUB -J FlowrateSampling
#BSUB -o FlowrateSampling.out
#BSUB -e FlowrateSampling.err
source /home/cont002/uranie/uranie-titane.cshrc
rm -f FlowrateSampling.out FlowrateSampling.err
root -l -q lanceurFLOWRATE_SAMPLING.C
```

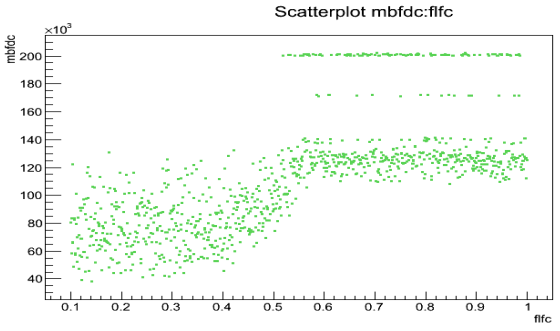
> bsub < BsubFile

Utilisation d'Uranie au CCRT

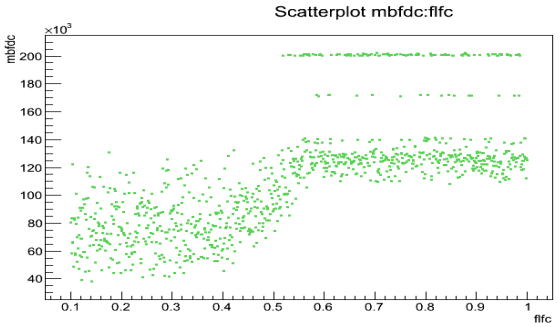
- Le mécanisme de lancement des calculs dans URANIE est transparent pour l'utilisateur : le script URANIE est le même que l'on soit sur un poste de travail ou sur le CCRT;
- La séquence est la suivante :
  - Le plan d'expérience est généré (en fonction de la méthode et des incertitudes sur les paramètres d'entrée)
  - URANIE analyse la machine locale à l'aide de variables d'environnement et déduit le nombre de processeurs disponibles
  - Un pool de processeurs est ensuite géré pour répartir les calculs au fur et à mesure sur les processeurs disponibles
- Travail réalisé par les équipes du Support Applicatif CCRT;
- Difficulté liée au fait qu'il est impossible de lancer *mpirun* depuis *mpirun*;
- Méthode choisie :
  - Le noeud maître gère la distribution des calculs au fur et à mesure;
  - Lorsqu'un groupe de processeurs est disponible, le process maître est forké et lance un *mpirun*;
  - La fin de l'exécution du cas est détectée en analysant l'état du process fils.

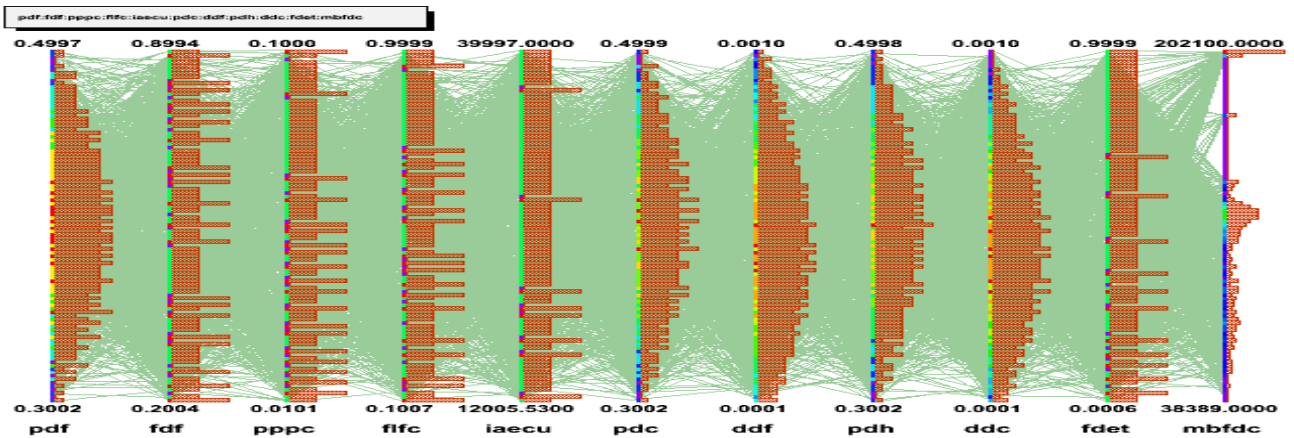


Uranie Application : $R^{32} \rightarrow R^{23}$ (1/3)



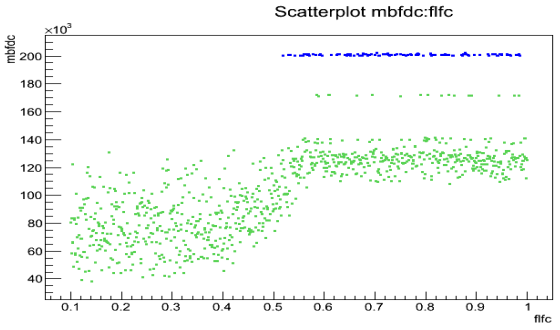
Uranie Application : $R^{32} \rightarrow R^{23}$ (2/3)

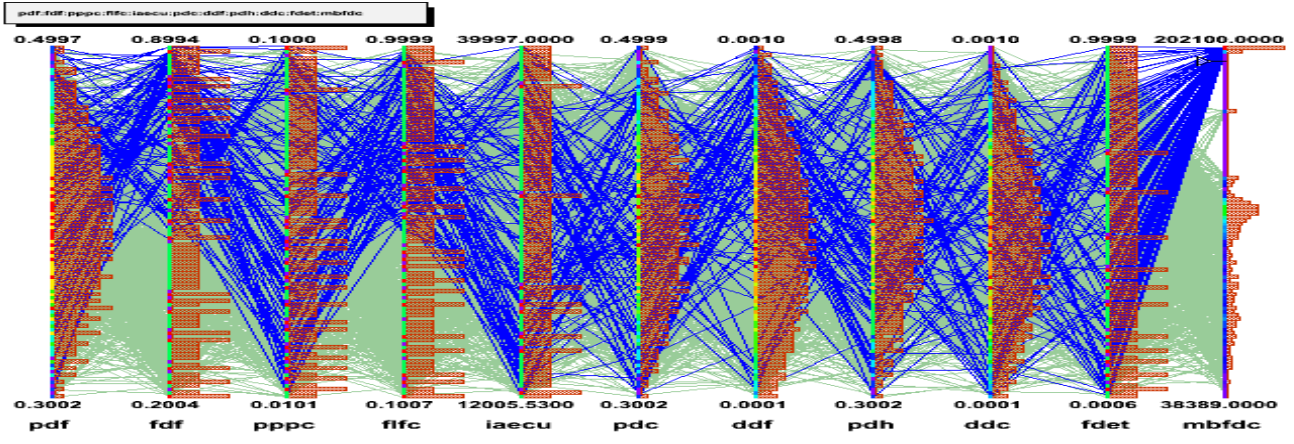




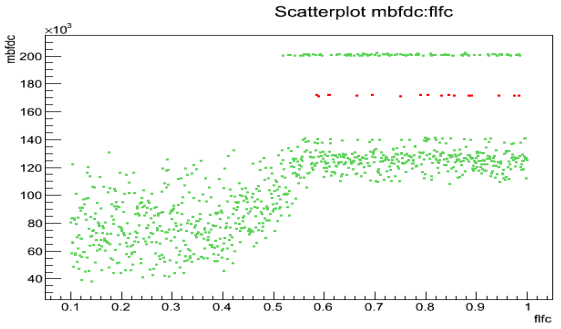
tds->Draw("pdf:fdf:pppc:fflfc:iaecu:pdc:ddf:pdh:ddc:fdet:mbfdc","","para");

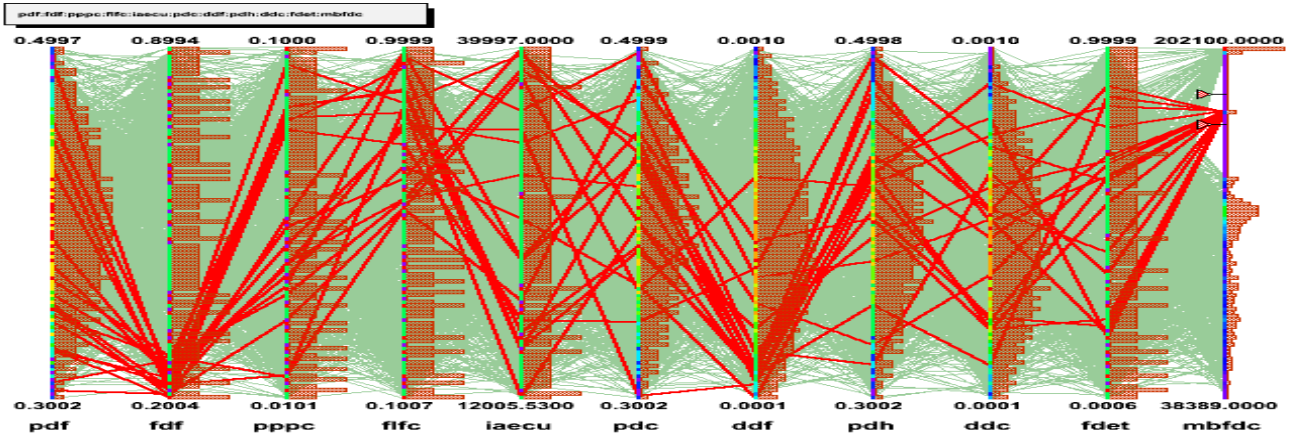
Uranie Application : $R^{32} \rightarrow R^{23}$ (2/3)



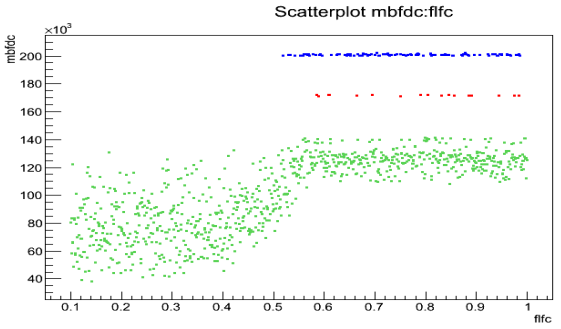


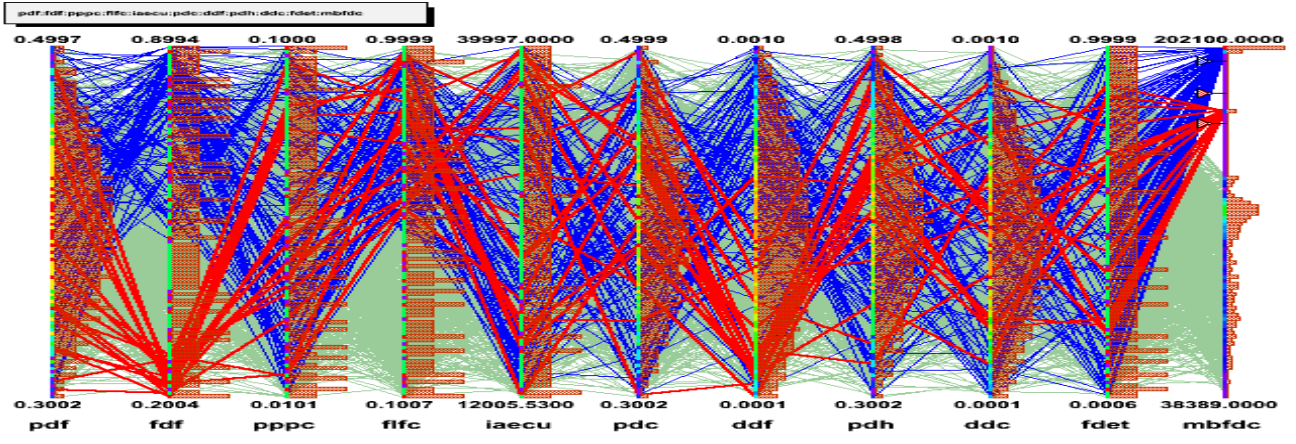
Uranie Application : $R^{32} \rightarrow R^{23}$ (2/3)



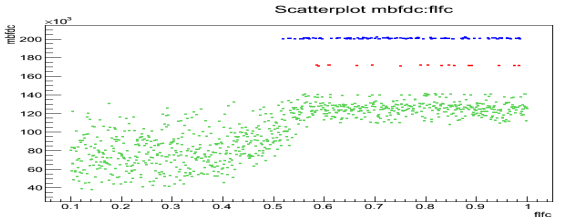


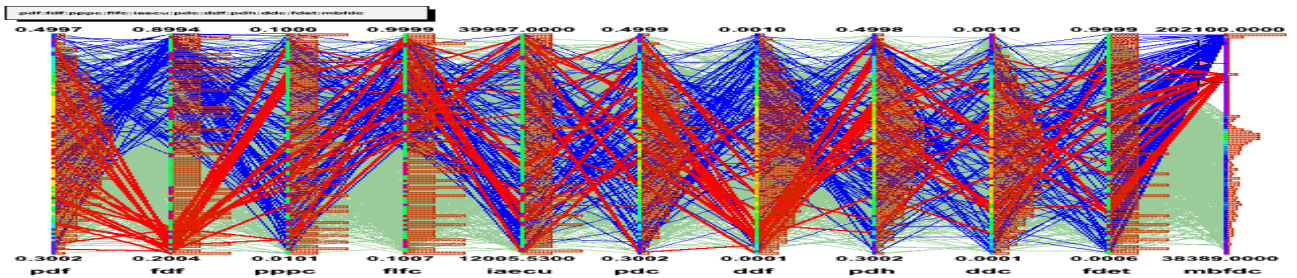
Uranie Application : $R^{32} \rightarrow R^{23}$ (2/3)

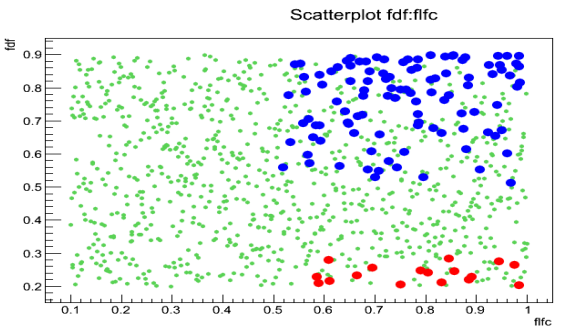




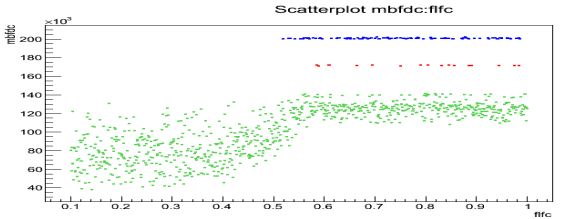
Uranie Application : $R^{32} \rightarrow R^{23}$ (3/3)

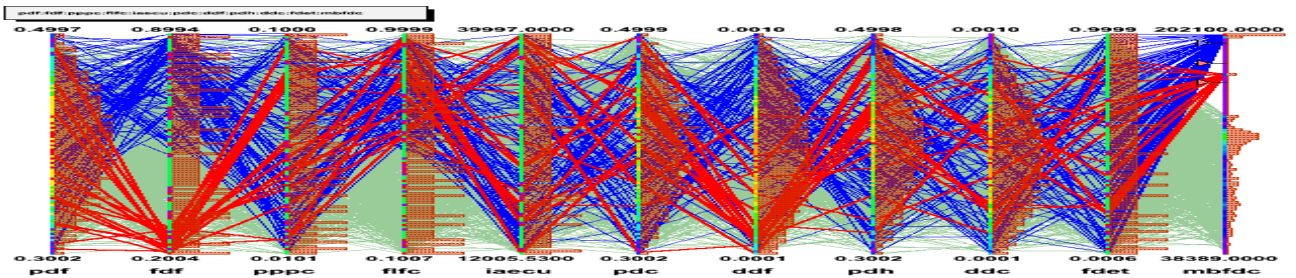


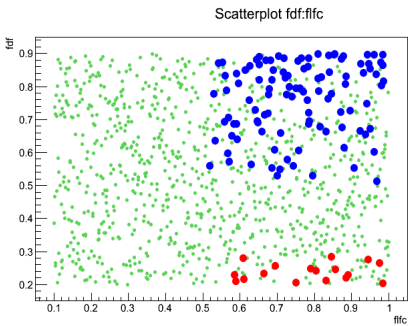


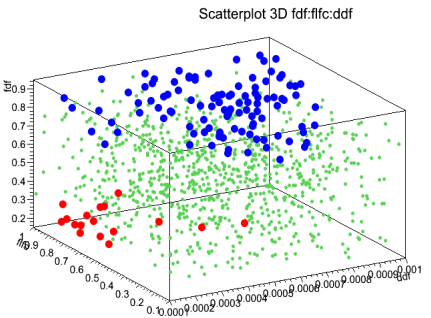


Uranie Application : $R^{32} \rightarrow R^{23}$ (3/3)









Uranie Application with ANDRA : GdR MoMaS

- Context

  Make Sensitivity Analysis with Surrogate Model : Artificial Neurals Networks ("ANN")

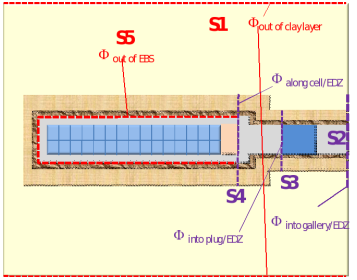
- CPU time single calculation : 1 < t < 5 hours (Cluster)

- Original DataSet

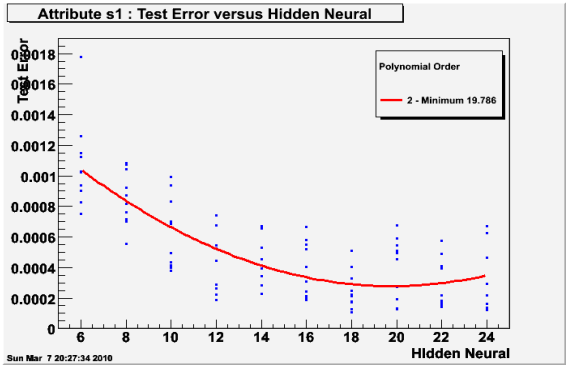
  - nX = 17 input variables
  - nY = 5 output variables
  - Build the 5 ANN : nS = 1500 patterns

- Neural networks

  - different architectures (MLP)
  - Input with a logarithm PDF $x := \log(x)$
  - Output $y := \frac{1}{1+log(y)}$
  - cross validation
  - Validate the 5 ANN on another dataset with nS = 1000 patterns



Guillaume Pépin (ANDRA)



Computation Sensitivity Indexes (SI)

First Order : $$S_i = \frac{Var[\, IE[\, Y \mid x_i \,]\,]}{Var[Y]}$$

1. **Regression Analysis** $$\hat{y} = b_0 + \sum b_i \, x_i$$

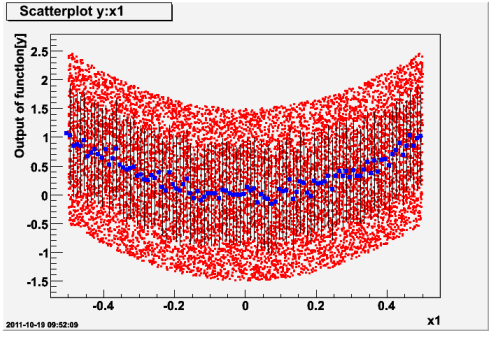
   - on the Values : "SRC" ("Standardised Regression Coefficient")
   - on the Ranks : "SRRC" ("Standardised Rank Regression Coefficient")

   The regression is valid when $R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$ close to 1.0 (≥ 0.7)

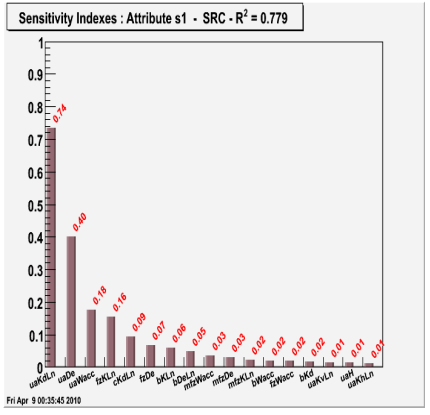
2. **Brute-Force Method**

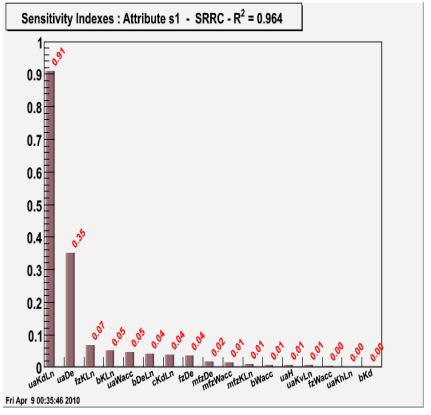
   Cost : nX * nCV * nPts

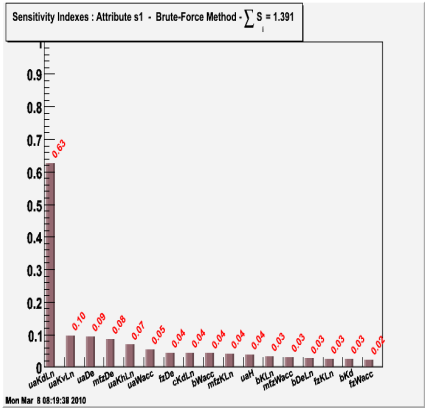
   - nX = number of inputs parameters
   - nCV = n. of conditionnal values
   - nPts = n. of pts over cond. values

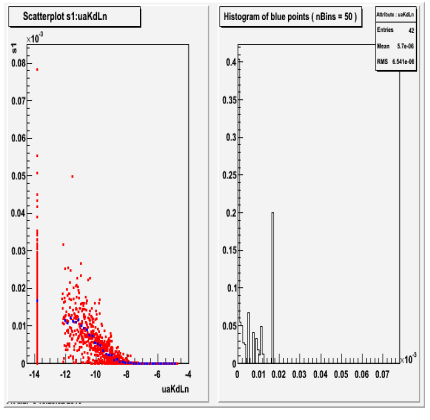


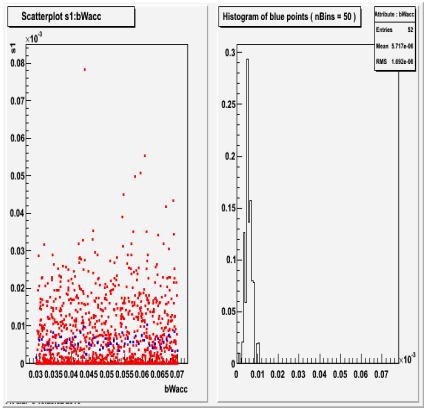
Original Database (nS = 1500) & "s1"

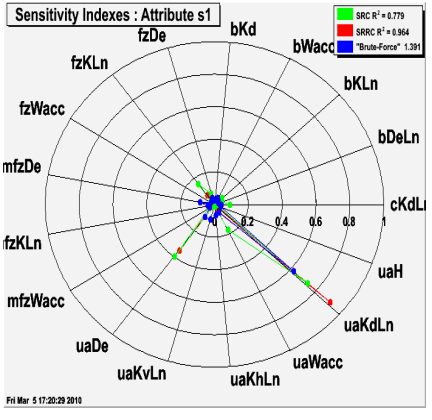




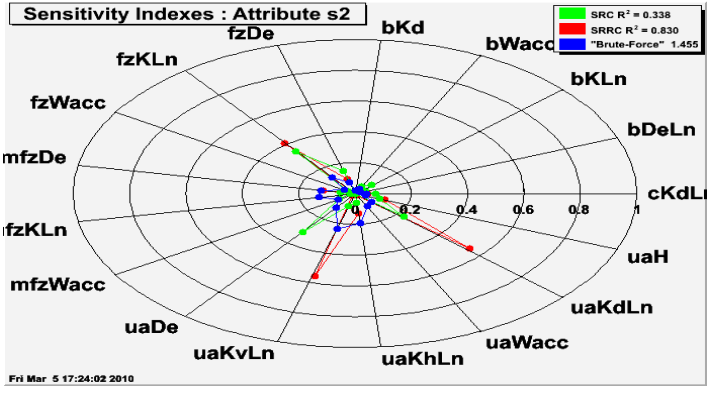


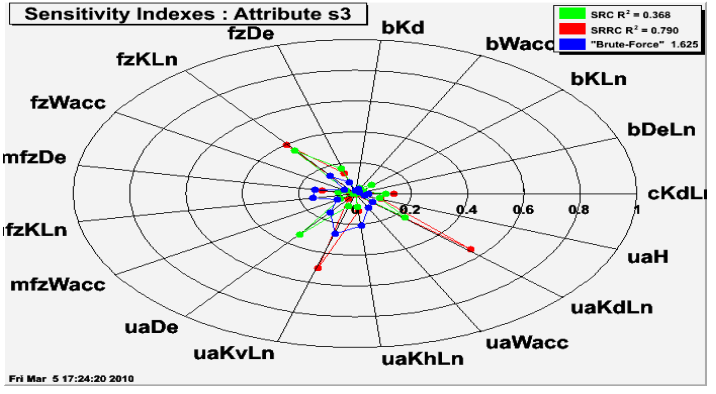


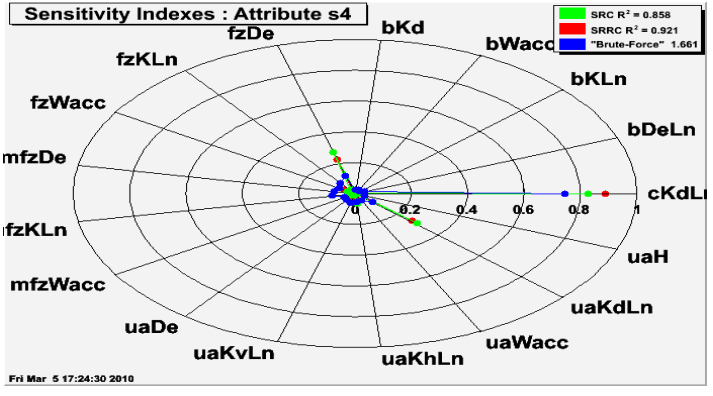


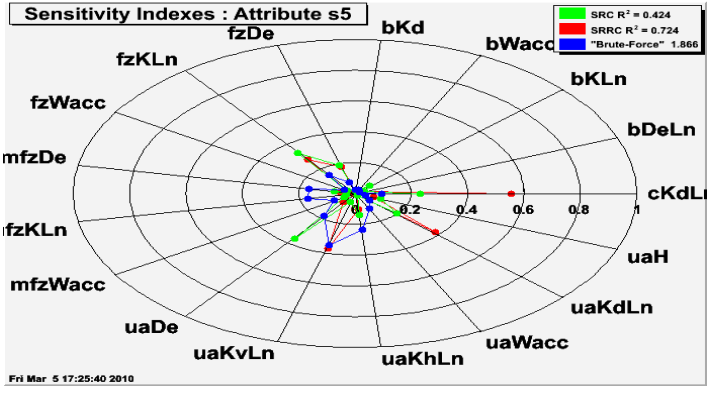


Original Database (nS = 1500) & other outputs







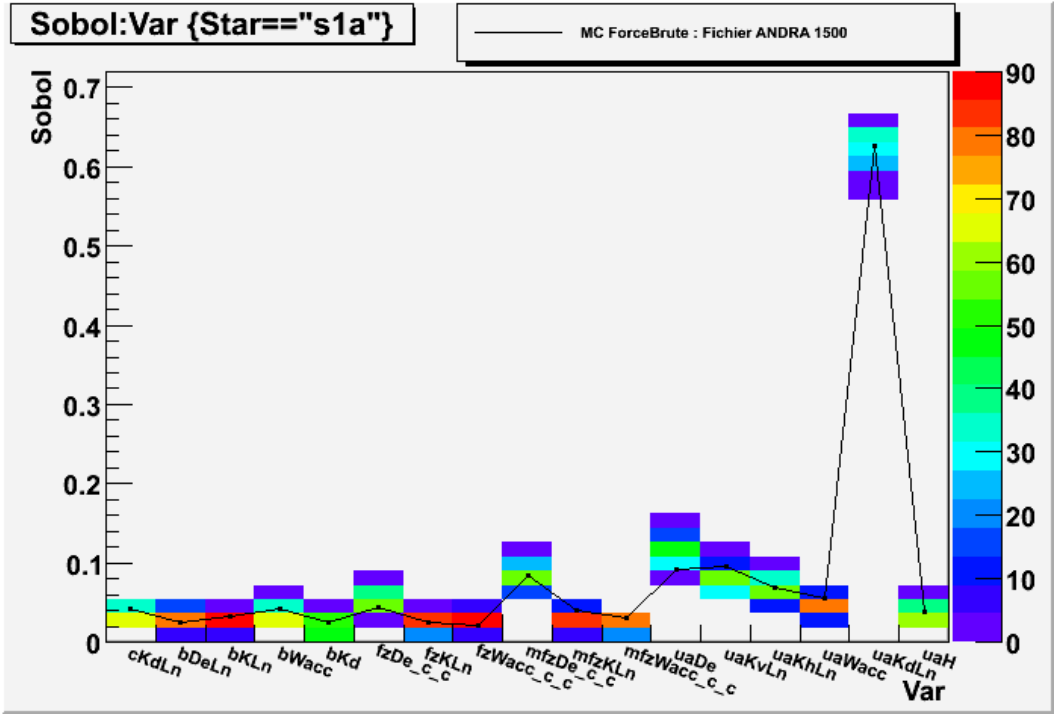


Estimation of SI with ANN and nS = 1500

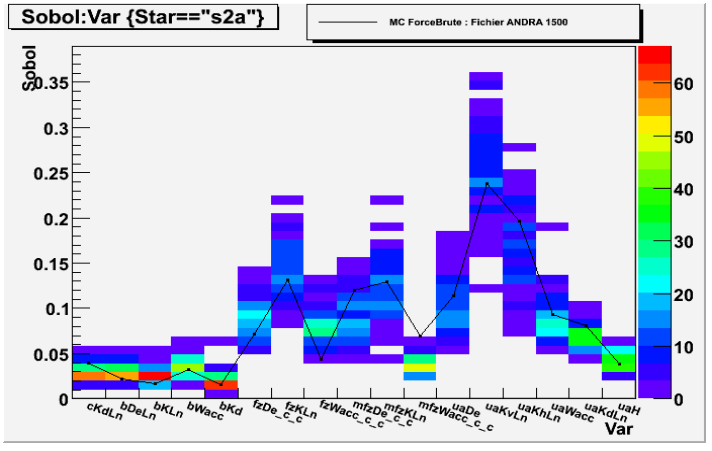
Design nL = 100 databases with size nS = 1500 points

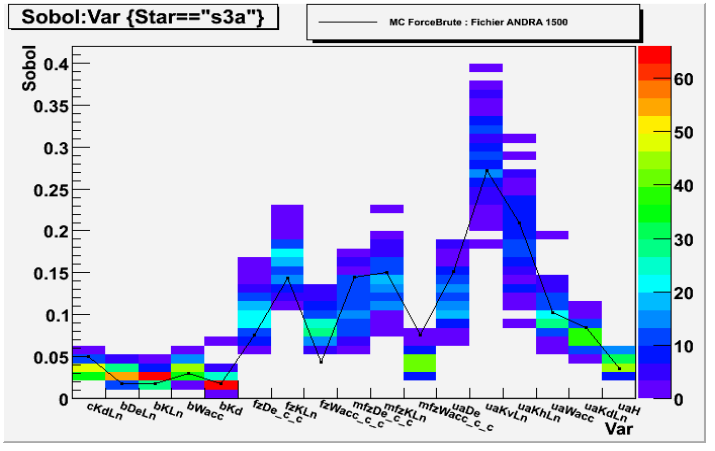
Simulate these databases using ANN surrogate models

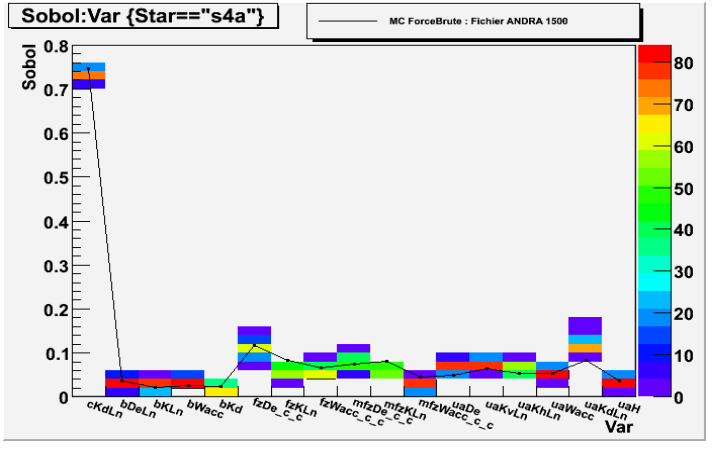
Compute the first order SI by the Brute-Force method

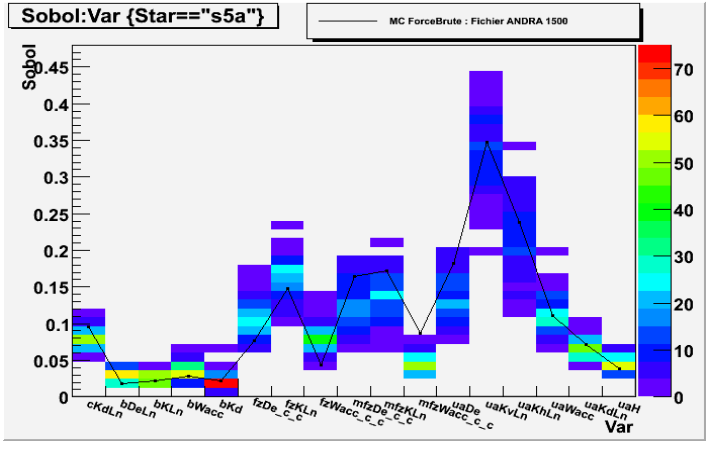


Estimation of SI with ANN and nS = 1500

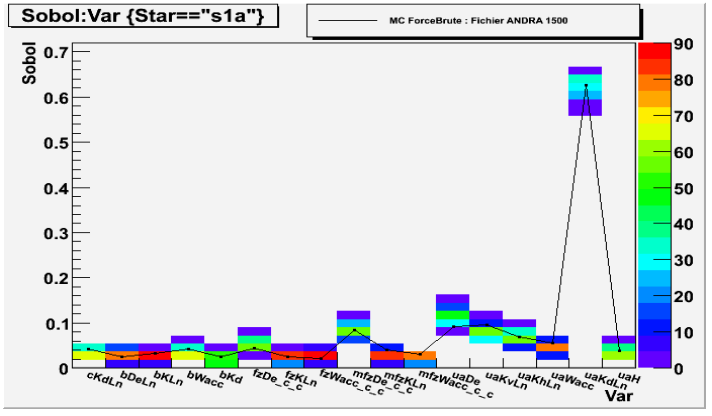


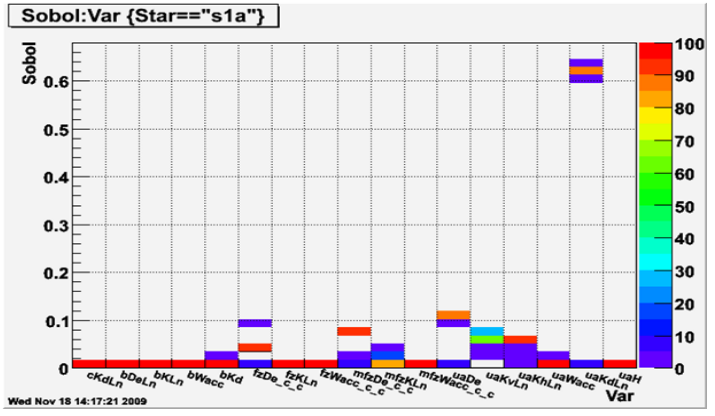


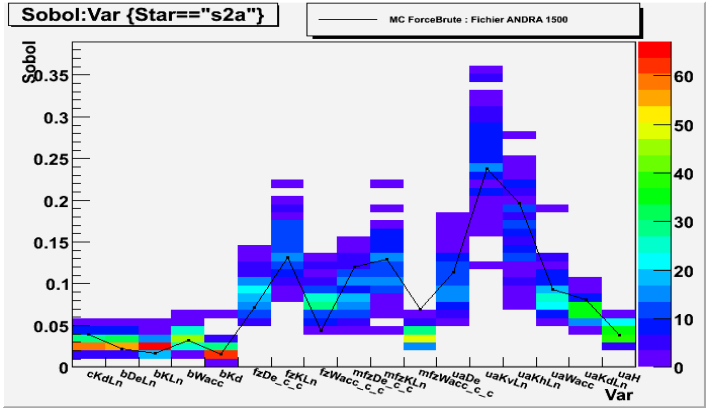


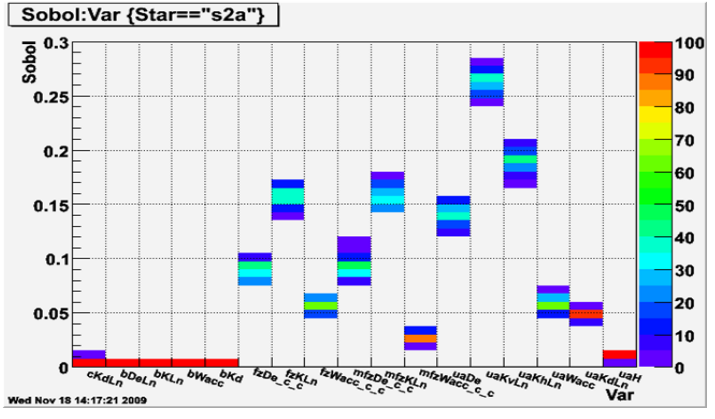


Same with nS = 20000 (« s1 » and « s2 »)





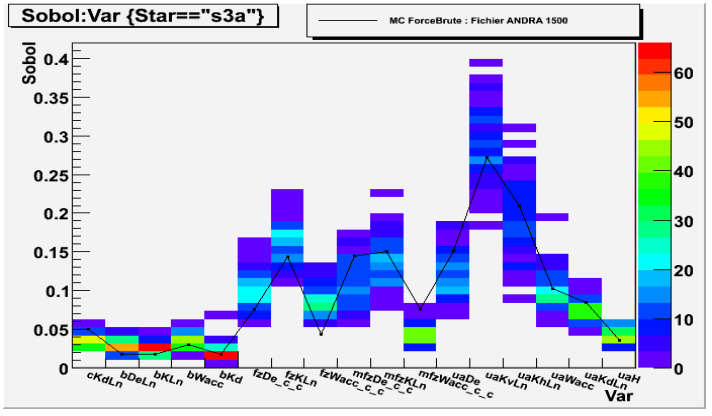


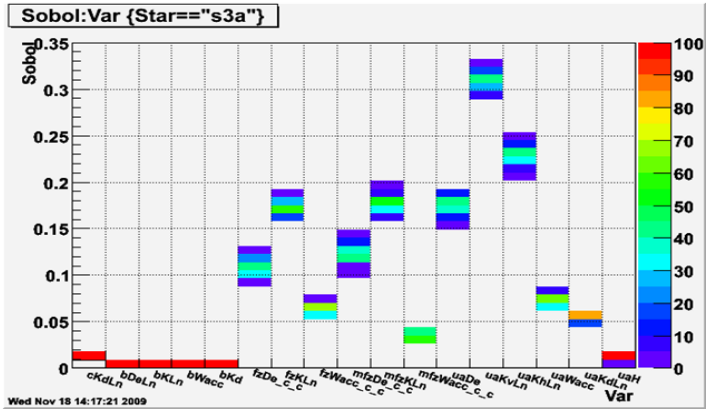


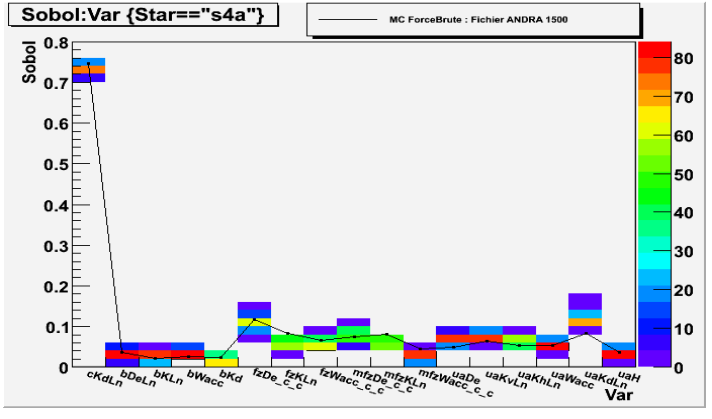
1 500 points

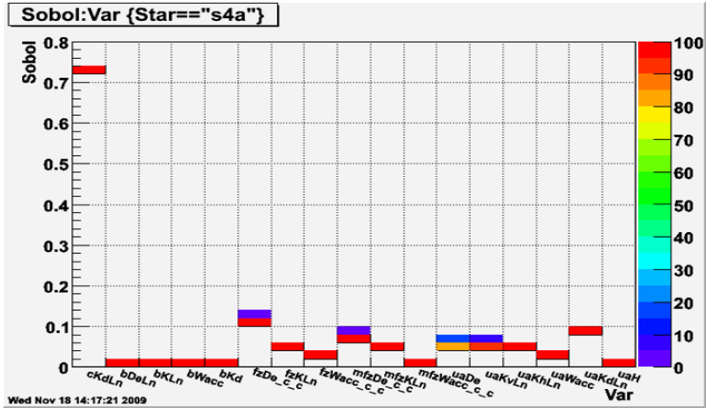
20 000 points

Same with nS = 20000 (« s3 » and « s4 »)







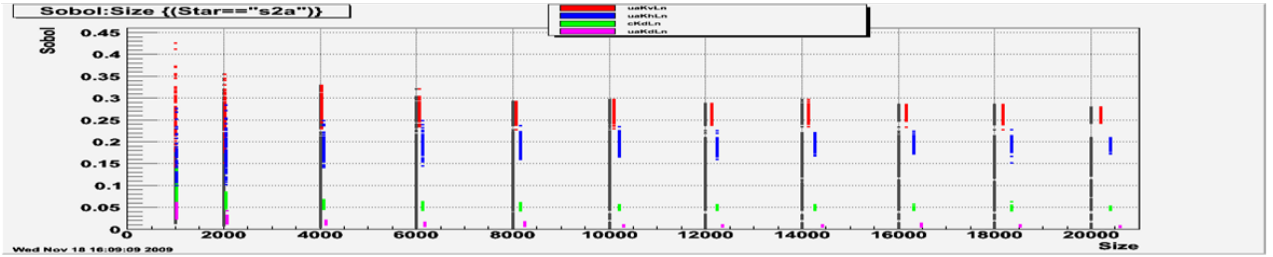


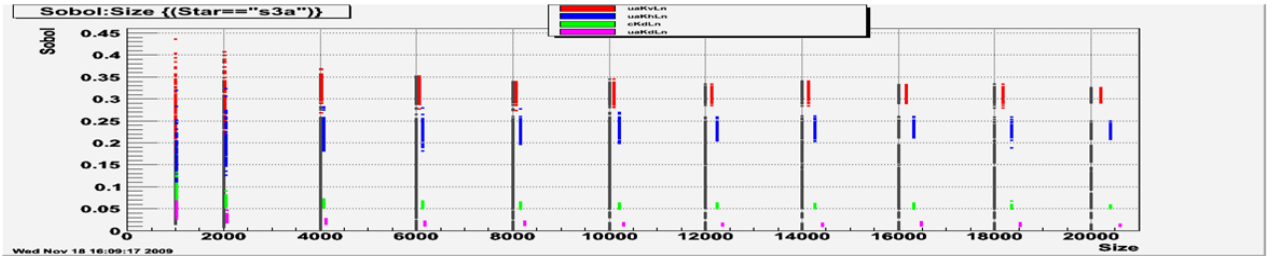
1 500 points

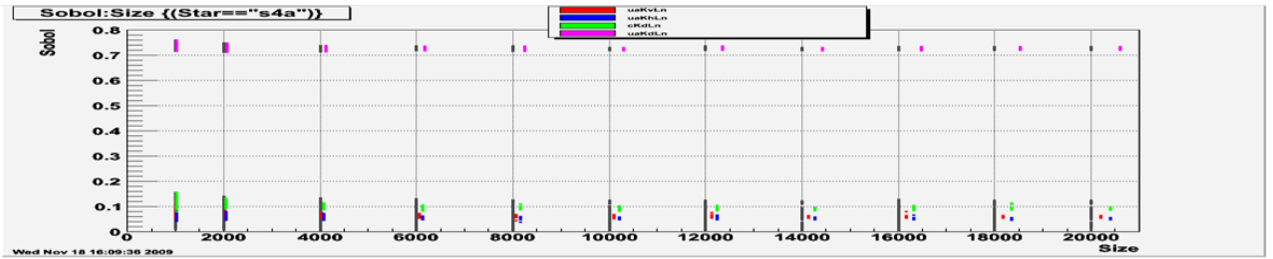
20 000 points

Uranie Application : All simulations (1/2)

Same loop (nL = 100 DoE) but for nS=2000 to 20000 by step 2000 :

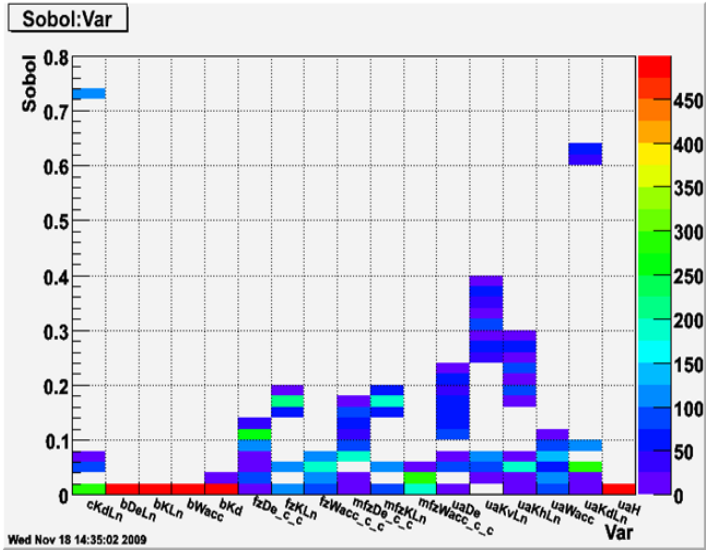




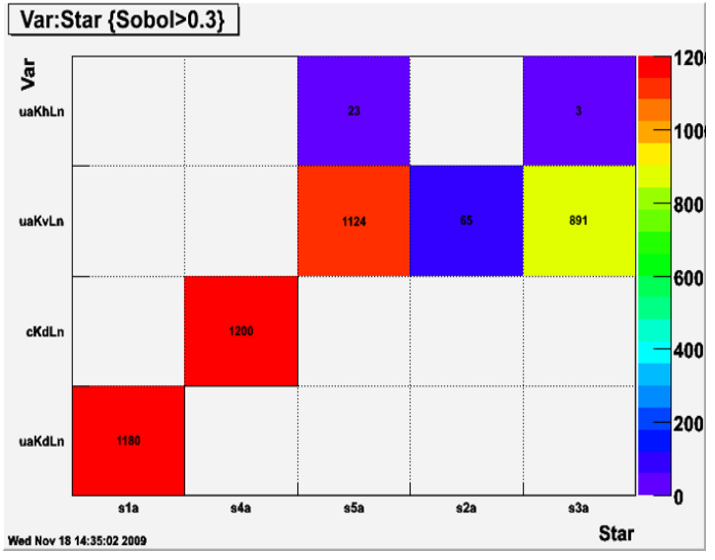


Uranie Application : All simulations (2/2)

1. 5 inputs parameters are not influence (first order) on all the 5 outputs
2. Only 4 inputs parameters are sensitives (for all 5 outputs)



ttre->Draw("Sobol:Var","","BOXCOL2Z")



ttre->Draw("Var:Star","Sobol>0.3","BOXCOL2Ztext")

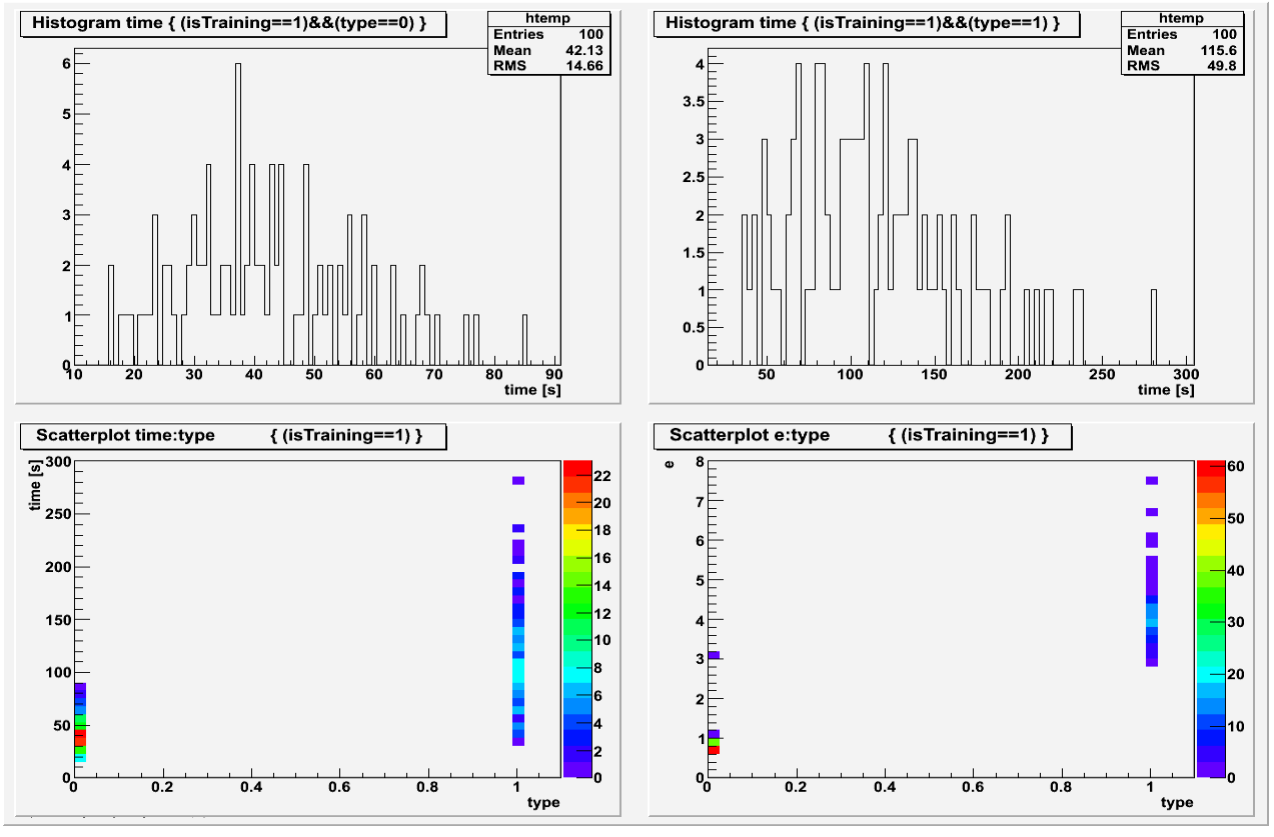
GPU - Description of the test

- Output : **st4** with $nH = 16$ neurons in the hidden layer ($\mathbb{R}^{17} \rightarrow \mathbb{R}^{16} \rightarrow \mathbb{R}$)
- Learning :
  - 1 database with 1 400 patterns
  - 100 learning
- Running :
  - 6 databases with 1 500, 5 000, 10 000, 20 000 and 50 000 patterns
  - 10 running for each databases
- criteria for comparison : time (t) and error (e)

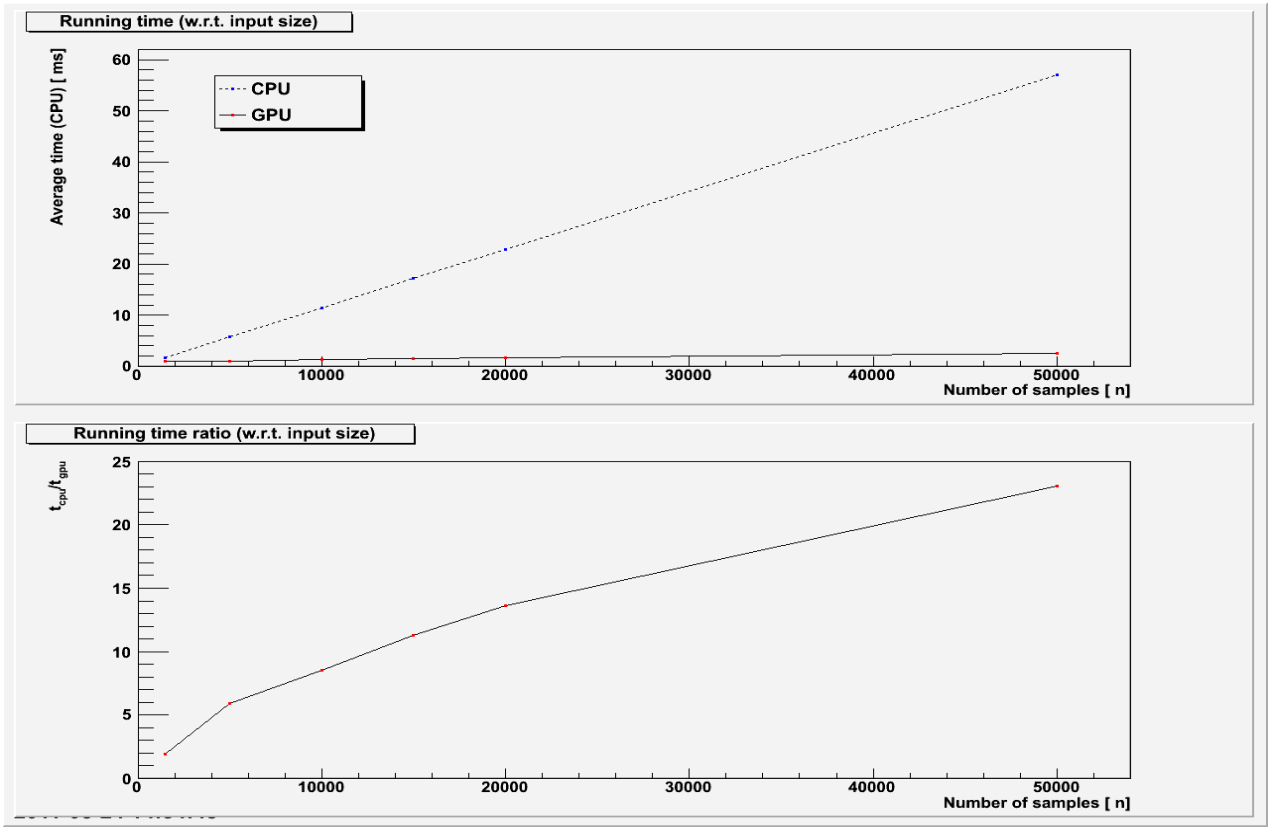
1. CPU : 2 quadri cores Xeon 5500 (Boost.uBLAS)
2. GPU : 8 Fermi Tesla C2050 (CUDA/cuBLAS)

GPU - Learning Performance

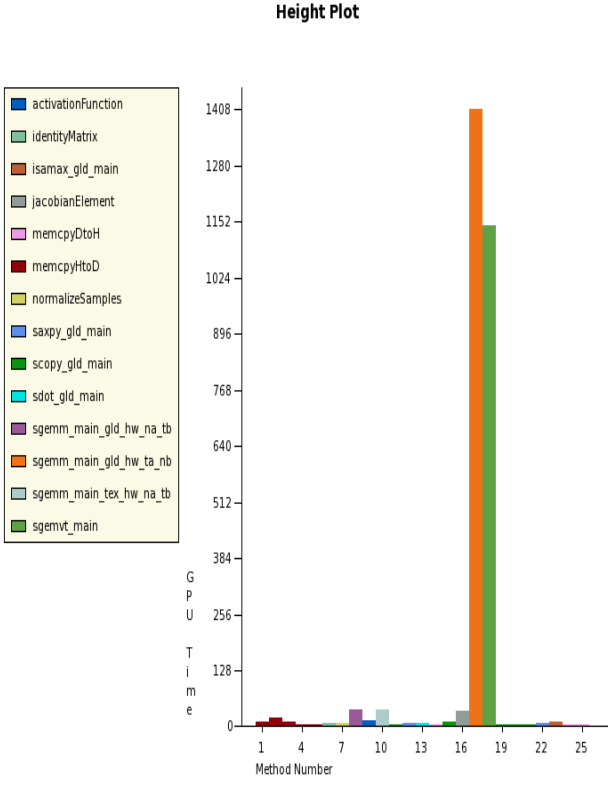
Mean : GPU 42 s versus CPU 115 s

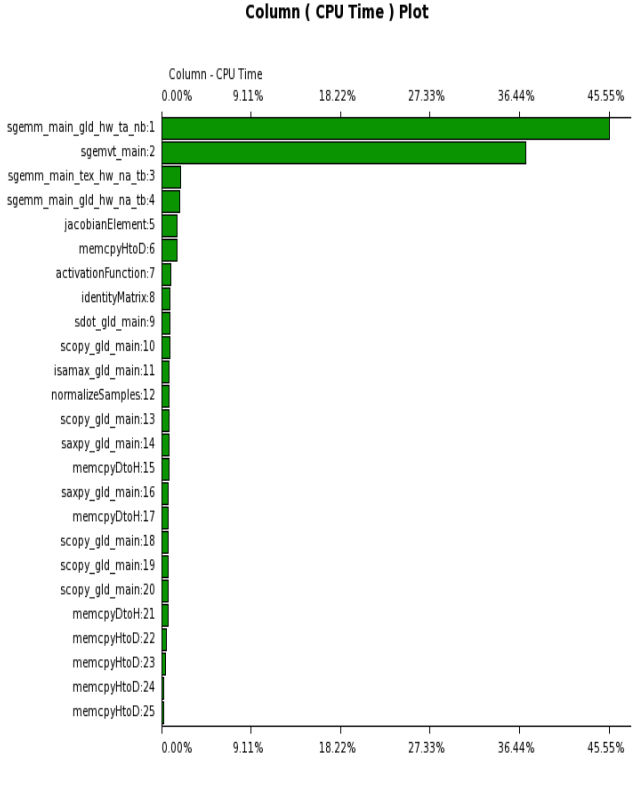


GPU - Evaluation Performance



GPU - Profiling





GPU - Conclusions

1. GPU is $2 \rightarrow 23$ faster than CPU for evaluation ANN
2. GPU is only 3 faster than CPU for learning ANN for small database
   same speed-up for large database ?
3. main time consumer : matrix computing