Uranie Platform v1.1

35th ESReDA Seminar

F. Gaudier

CEA/DEN/DANS/DM2S/SFME/LGLS

Marseille, November, 19 2008

CERN Large Hadron Collider (LHC)
ROOT (http://root.cern.ch)

CERN Large Hadron Collider (LHC)

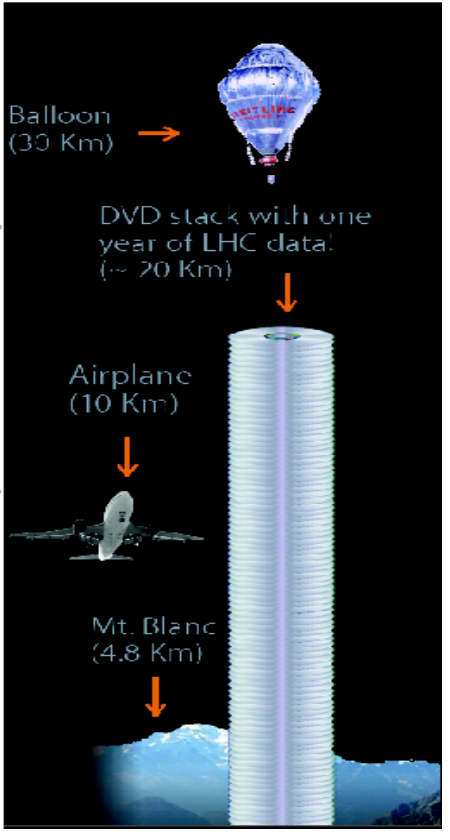
- particle accelerator
- 27 km circumference tunnel in Genève
- 4 experiments (ATLAS, CMS, ALICE, LHCb)

Study the matter structure

— Research the Higgs boson

— Research new physic

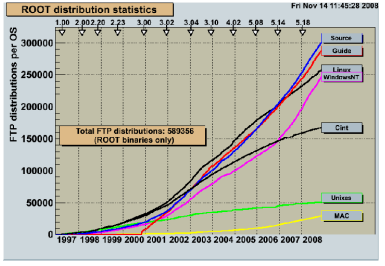
- Data quantity generated : 20 PetaBytes/year
- ROOT is the framework to store, treat and analyze this data

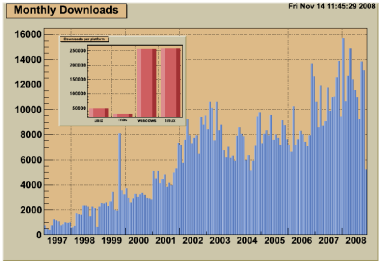


ROOT (http://root.cern.ch)

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

- 10 years of development (C++ with 3-4 releases/year)
- multi-plateform (Unix, Windows, Mac OS X)
- Offer :
  - A C++ interpreter
  - 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
  - Advanced visualization tools
- **License LGPL**





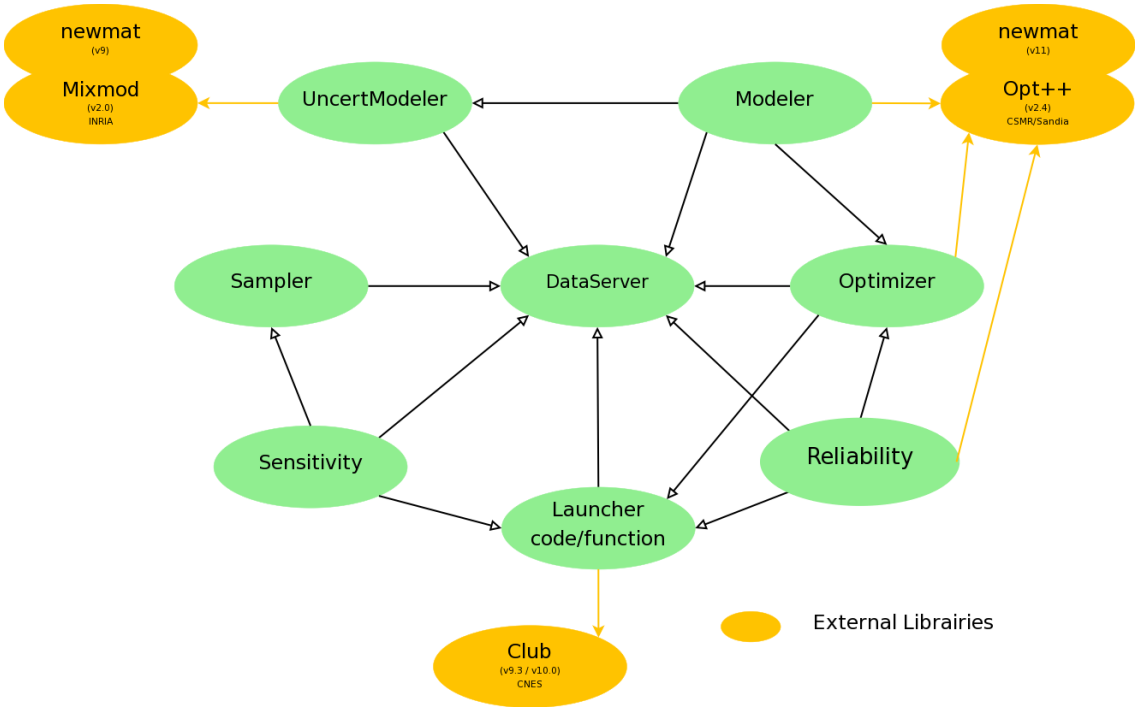
URANIE v1.0

URANIE : CEA/DEN Uncertainty Platform
URANIE : Fonctional diagram
URANIE : Graphical User Interface
URANIE - XML User Interface
URANIE : Batch mode
Projects using URANIE
"DataServer" module - URANIE ASCII file format

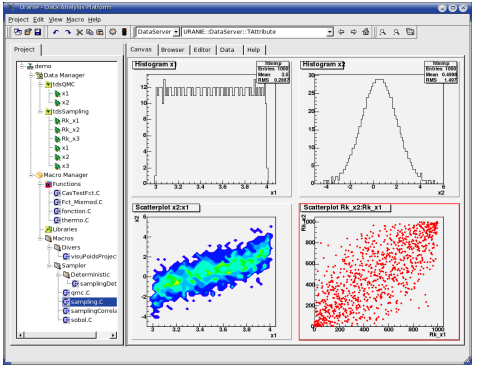
URANIE : CEA/DEN Uncertainty Platform

- ROOT (CERN), MIXMOD (Gaussian Mixtures - INRIA), OPT++ (Optimization - Sandia), CLUB (Computation - CNES),
- QT (Graphical User Interface)
- Data access :
  - Flat file with header ("Salomé Table")
  - TTree (internal ROOT)
  - SQL Data base (MySQL, PostgreSQL, ...)
- Uncertainty/Sensitivity methods in URANIE
  - Design Of Experiments (SRS, LHS, ROA, qMC, MCMC, Copulas)
  - Surrogate models (Polynomial , Artificial Neural Networks, Splines)
  - Non Intrusive Spectrale Projection : Generalized Polynomial Chaos
  - Sensitivity analysis (Pearson, Spearmann, Sobol, Fast, Morris)
  - Optimization, Multi-Criteria (Genetic Algorithms)
  - Computing distribution

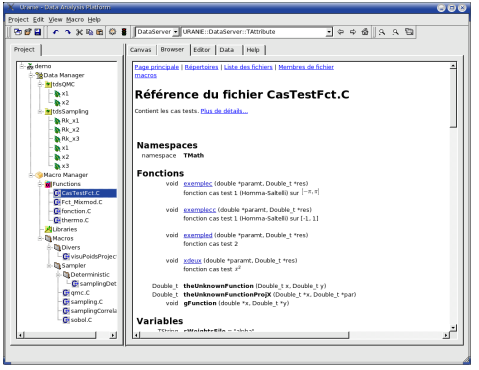
URANIE : Fonctional diagram



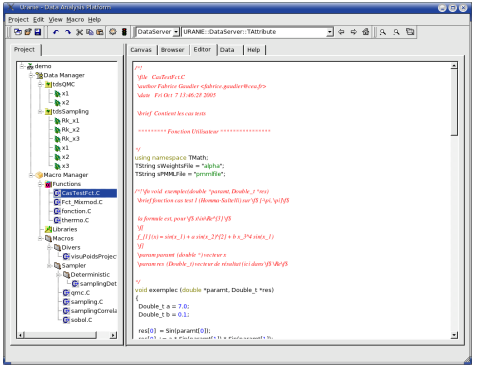
URANIE : Graphical User Interface



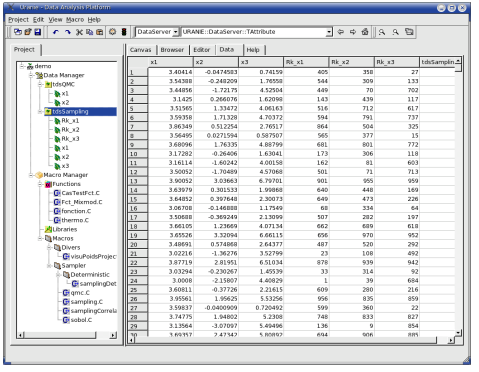
Visualization



User Help



Editor



Spreadsheet

ROOT

URANIE . . .

benchmark . .

Sensitivity . . .

URANIE - XML User Interface

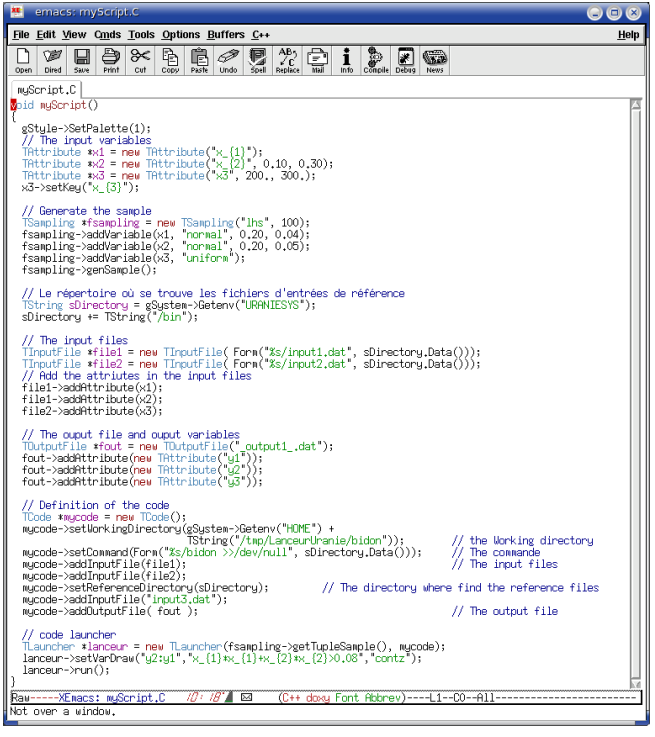
XML file (problem_uranie.xml)

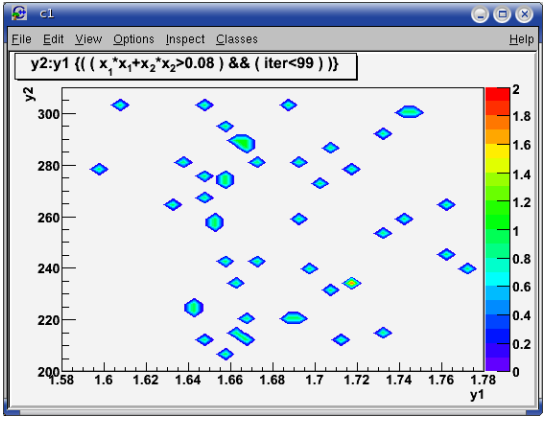
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Problem SYSTEM "/home/uranie/tools/share/uranie/uranie.dtd" >
<Problem>
    <Header name="Etude" title="projet GENTR">
        <Application name="uranie" version="0.4"/>
    </Header>
    <DataDictionary>
        <DataField name="x1" law="uniform" min="0.5" max="1.5"/>
        <DataField name="x2" law="normal" mean="2.5" std="0.25"/>
    </DataDictionary>
    <Sampler method="SRS" N="1500" export="data/sampler_SRS_1500.dat"/>
    <Sampler method="LHS" N="1000" export="data/sampler_LHS_1000.dat"/>
</Problem>
```

```
uranie -s problem_uranie.xml
```

URANIE : Batch mode

> root myScript.C





ROOT

URANIE . . .

benchmark . .

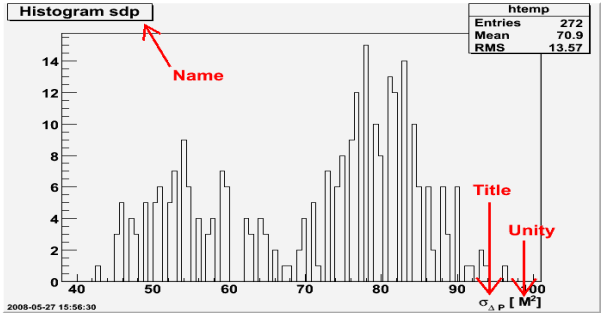
Sensitivity . . .

Projects using URANIE

- LEONAR tool for severe accidents in french nuclear reactor (**CEA-EDF**)
- PSI-Matador Methodology : Dosimetry computation in french nuclear reactor (**CEA-EDF**)
- EHPOC project : Meteor code (**CEA**)
- Sensitivity Analysis for Cathare code (**CEA/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.

”DataServer” module - 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
...
```

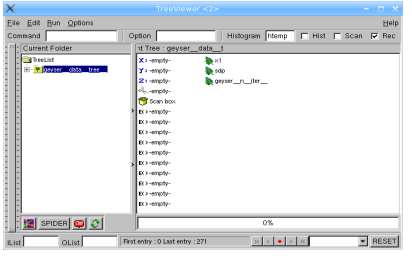


tds->draw(”sdp”);

Only the `"#COLUMN_NAMES:"` line is obligatory

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

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



benchmark "FlowRate"

benchmark "FlowRate"

Analytical function of R^8 in R defined by :

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

with

1. $r_\omega \sim \mathcal{U}[0.05, 0.15]$: radius of borehole (m)
2. $r \sim \mathcal{U}[100, 50\ 000]$: radius of influence (m)
3. $T_u \sim \mathcal{U}[63\ 070, 115\ 600]$: Transmissivity of upper aquifer (m^2/year)
4. $T_l \sim \mathcal{U}[63.1, 116]$: Transmissivity of lower aquifer (m^2/year)
5. $H_u \sim \mathcal{U}[990, 1\ 110]$: Potentiometric head of upper aquifer (m)
6. $H_l \sim \mathcal{U}[700, 820]$: Potentiometric head of lower aquifer (m)
7. $L \sim \mathcal{U}[1\ 120, 1\ 680]$: length of borehole (m)
8. $K_\omega \sim \mathcal{U}[9\ 855, 12\ 045]$: hydraulic conductivity of borehole (m)

Fang, p 35

benchmark "FlowRate" : Analytical Function

Prototype : **void myFunction (Double_t *inp, Double_t *out)**

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

```
#include "TMath.h"

void WaterBoreHole(double *x, Double_t *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;
}
```

benchmark "FlowRate" : Code

Usage: flowreate [-d] [-k] [[-f] file] [[-k] file] [-h|-?]

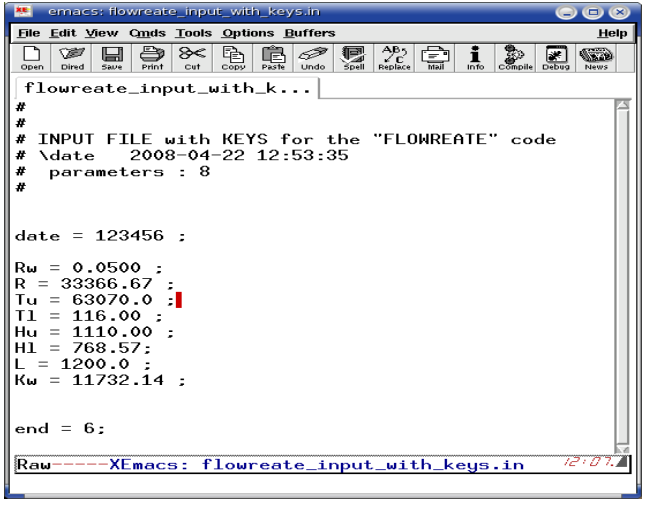
-d: debug mode

-v: verbose mode

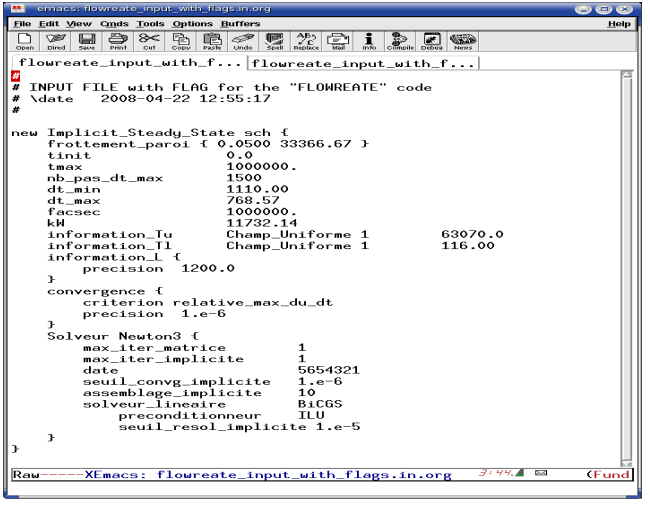
-f: input file with flag [flowreate_input_with_flag.in]

-k: input file with key [flowreate_input_with_keys.in]

-h,-?: this help message



with Keys



with Flags

Polynomial regression from Ho and Xu(2000)

Uniform Design from Ho and Xu(2000) : 32 points

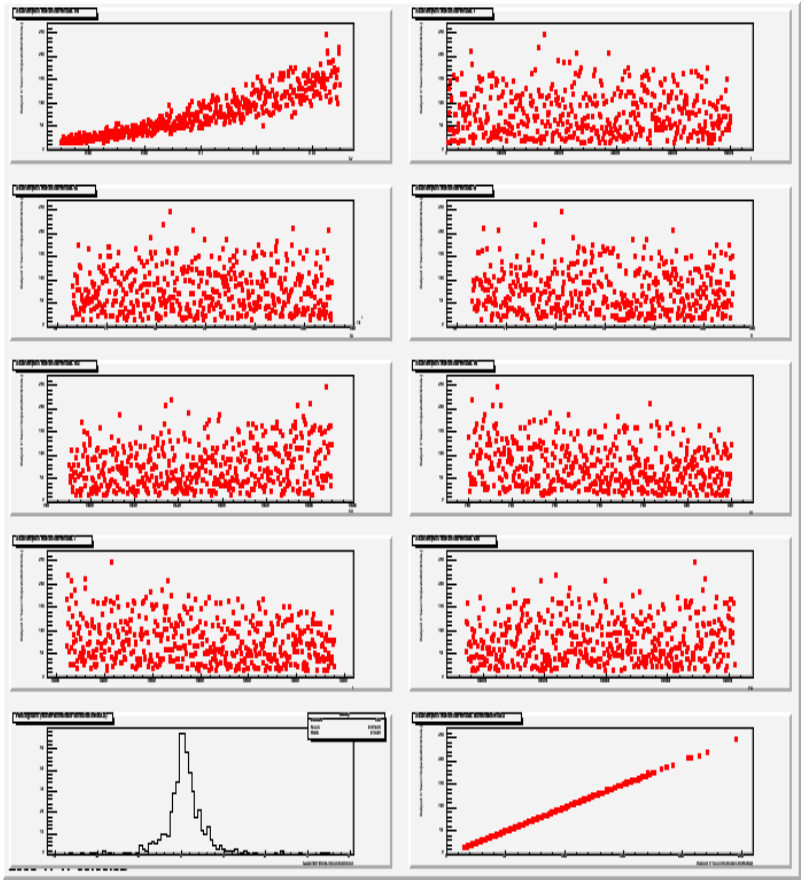
$$\begin{aligned}\hat{log}(y) &= 4.156 + 1.9903(\log r_\omega + 2.3544) - 0.0007292(L - 1400) - .003554(H_l - 760) \\ &+ .0035068(H_u - 1050) + .000090868(K_\omega - 10950) - .00087286(\log(r) - 8.8914) \\ &+ .000015325(H_u - 1050)(H_l - 760) + .00000026487(L - 1400)^2 \\ &- .0000071759(H_l - 760)^2 - .0000068021(H_u - 1050)^2\end{aligned}$$

```
void WaterBoreHoleHoXu(double *x, Double_t *y)
{
  Double_t drw = x[0], dr  = x[1];
  Double_t dtu = x[2], dtl = x[3];
  Double_t dhu = x[4] - 1050.0, dhl = x[5] -   760.0;
  Double_t dl  = x[6] - 1400.0, dkw = x[7] - 10950.0;

  Double_t yHoXu = 4.1560 + 1.9903 * (TMath::Log(drw) + 2.3544);
  yHoXu -= .0007292 * dl;
  yHoXu -= .003554 * dhl;
  yHoXu += .0035068 * dhu;
  yHoXu += .000090868 * dkw;
  yHoXu += .000015325 * dhu * dhl;
  yHoXu += .00000026487 * dl * dl;
  yHoXu -= .0000071759 * dhl * dhl;
  yHoXu -= .0000068021 * dhu * dhu;
  yHoXu -= .00087286 * (TMath::Log(dr) - 8.8914);

  y[0] = TMath::Exp(yHoXu);

}
```

benchmark "FlowRate" : Macro Sampling

```
{
  TDataServer *tds = new TDataServer("samplllhs", "Sampling LHS");
  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));

  TSampling *sampling = new TSampling(tds, "lhs", 500);
  sampling->generateSample();

  TLauncherFunction * tlf = new TLauncherFunction(tds, WaterBoreHole,"","waterborehole");
  tlf->run();

  TLauncherFunction * tlf = new TLauncherFunction(tds, WaterBoreHoleHoXu,"","boreholeHoXu");
  tlf->run();

  TCanvas *c = new TCanvas();
  c->Clear();  c->Divide(2,5);
  c->cd(1); tds->draw("waterborehole:rw");
  c->cd(2); tds->draw("waterborehole:r");
  c->cd(3); tds->draw("waterborehole:tu");
  c->cd(4); tds->draw("waterborehole:tl");
  c->cd(5); tds->draw("waterborehole:hu");
  c->cd(6); tds->draw("waterborehole:hl");
  c->cd(7); tds->draw("waterborehole:l");
  c->cd(8); tds->draw("waterborehole:kw");
  c->cd(9); tds->draw("waterborehole-boreholeHoXu");
  c->cd(10); tds->draw("waterborehole:boreholeHoXu");
  c->SaveAs("../fiGURE/flowrateborholeSampling.png");
}
```

benchmark "FlowRate" : Macro Sampling



benchmark "FlowRate" : XML Interface

```
1<?xml version="1.0" encoding="iso-8859-1"?>
2<?xml-stylesheet type="text/xsl" href="/home/uranie/tools/share/uranie/uranie.xslt"?>
3<!DOCTYPE Problem SYSTEM "/home/uranie/tools/share/uranie/uranie.dtd">
4<Problem>
5  <Header name="boreholeXML" title="Launch the Borehole function in XML" debug="0">
6    <Application name="uranie" version="1.0"/>
7  </Header>
8  <DataDictionary>
9    <DataField name="rw" law="uniform" min="0.05" max="0.15"/>
10   <DataField name="r"  law="uniform" min="100.0" max="50000.0"/>
11   <DataField name="tu" law="uniform" min="63070.0" max="115600.0"/>
12   <DataField name="tl" law="uniform" min="63.1" max="116.0"/>
13   <DataField name="hu" law="uniform" min="990.0" max="1110.0"/>
14   <DataField name="hl" law="uniform" min="700.0" max="820.0"/>
15   <DataField name="l"  law="uniform" min="1120.0" max="1680.0"/>
16   <DataField name="kw" law="uniform" min="9855.0" max="12045.0"/>
17 </DataDictionary>
18 <Sampler method="LHS" N="100" export="waterhole_sampler_lhs.dat"/>
19 <Launcher macro="FunctionsToCompile.C" function="WaterBoreHole" output="wh" export="waterholelhs.dat"/>

20 <Sampler method="SRS" N="2000" export="waterhole_sampler_srs.dat"/>
21 <Launcher function="WaterBoreHoleHoXu" input="r:rw:tu:tl:hu:hl:l:kw" export="waterholeholusrs.dat"/>

22</Problem>
```

Sensitivity Analysis

Sensitivity Indexes from Regression (SRC)
Morris Method (27 evaluations)
Sobol Indexes from "Saltelli" Method
Sobol Indexes from "Fast" Method
Sobol Indexes from Polynomial Chaos

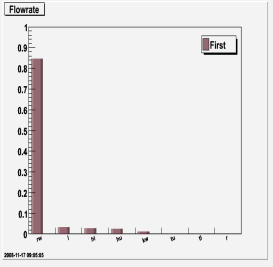
Sensitivity Indexes from Regression (SRC)

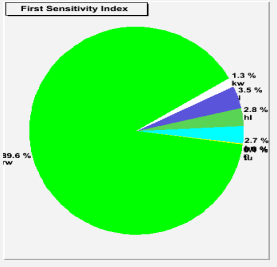
TRegression * treg = new TRegression(tds, "rw:r:tu:tl:hu:hl:l:kw","waterborehole", "SRC");

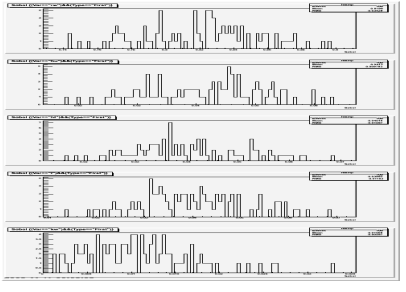
treg->computeIndexes();

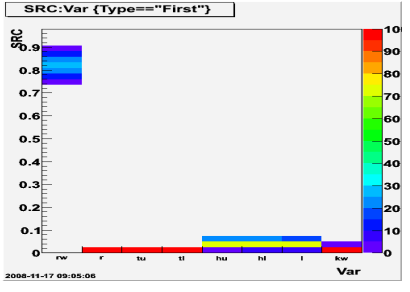
treg->drawIndexes("Flowrate", "", "nonewcanv,hist,first");

treg->drawIndexes("Flowrate", "", "nonewcanv,pie,first");









ROOT

URANIE . . .

benchmark . .

Sensitivity . . .

Morris Method (27 evaluations)

TMorris * tmor = new TMorris(tds, WaterBoreHole, 3, 10)

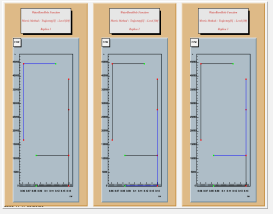
tmor->computeIndexes();

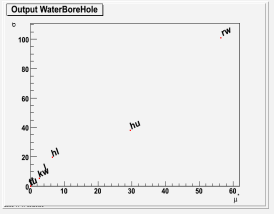
tmor->drawSample("rw:r", 1);

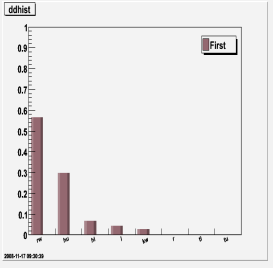
tmor->drawIndexes("mustar");

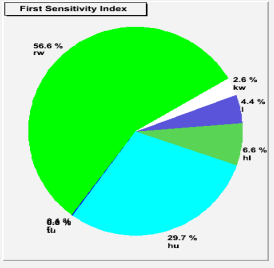
tmor->drawIndexes("Flowrate", "", "nonewcanv,hist,first");

tmor->drawIndexes("Flowrate", "", "nonewcanv,pie,first");





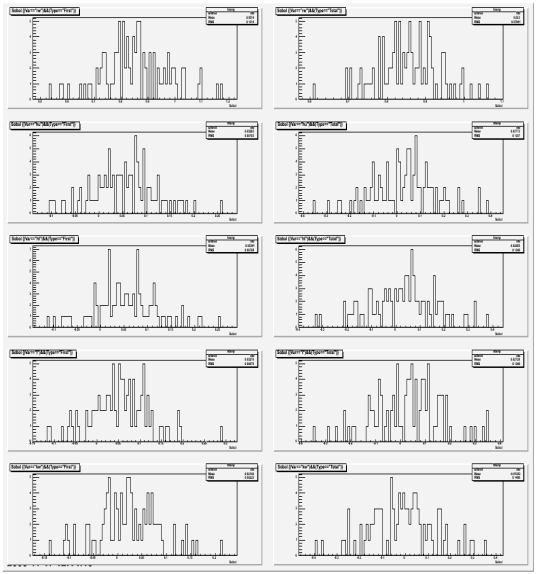


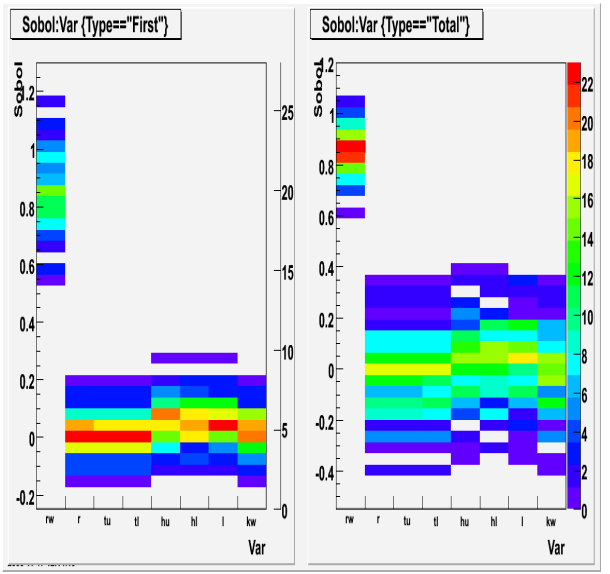


Sobol Indexes from "Saltelli" Method

```
TSobol * tsob = new TSobol(tds, WaterBoreHole, 40000);
tsob->computeIndexes();
```

Monte-Carlo : 100 simulations with 40 000 evaluations each one:





ROOT

URANIE . . .

benchmark . .

Sensitivity . . .

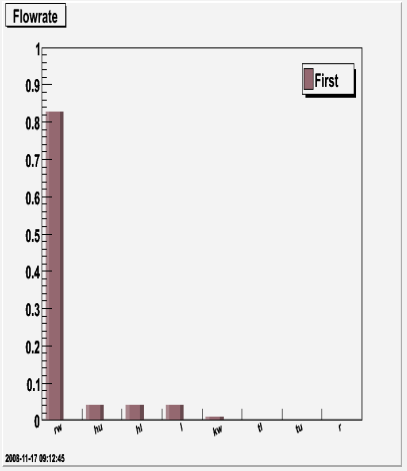
Sobol Indexes from "Fast" Method

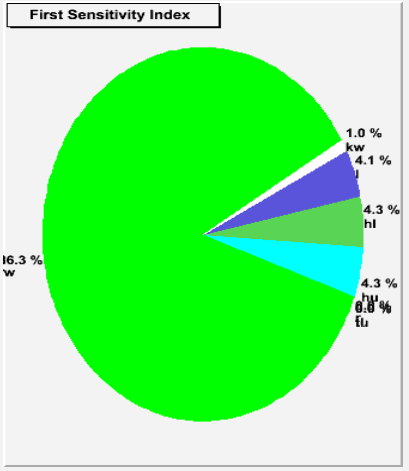
TFast * tfast = new TFast(tds, WaterBoreHole, 500);

tfast->computeIndexes();

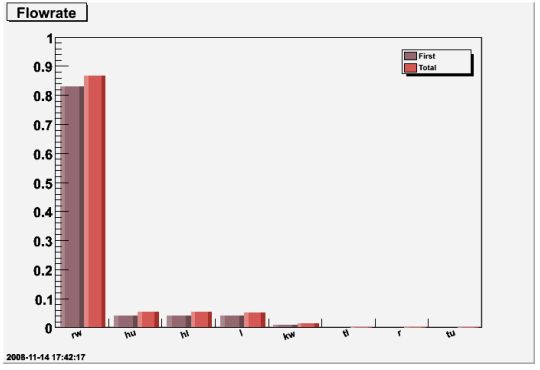
tfast->drawIndexes("Flowrate", "", "nonewcanv,hist,first");

tfast->drawIndexes("Flowrate", "", "nonewcanv,pie,first");





Sobol Indexes from Polynomial Chaos



Variables	S_i	ΣS_i
R_ω	0.828934	0.828934
H_l	0.041385	0.870319
H_u	0.041385	0.911705
l	0.039342	0.951047
(R_ω, H_l)	0.011827	0.962875
(R_ω, H_u)	0.011827	0.974703
(R_ω, l)	0.011124	0.985828