Toolbox Nisp - Scilab
Revue OPUS

M. Baudin

INRIA

31/03/2009

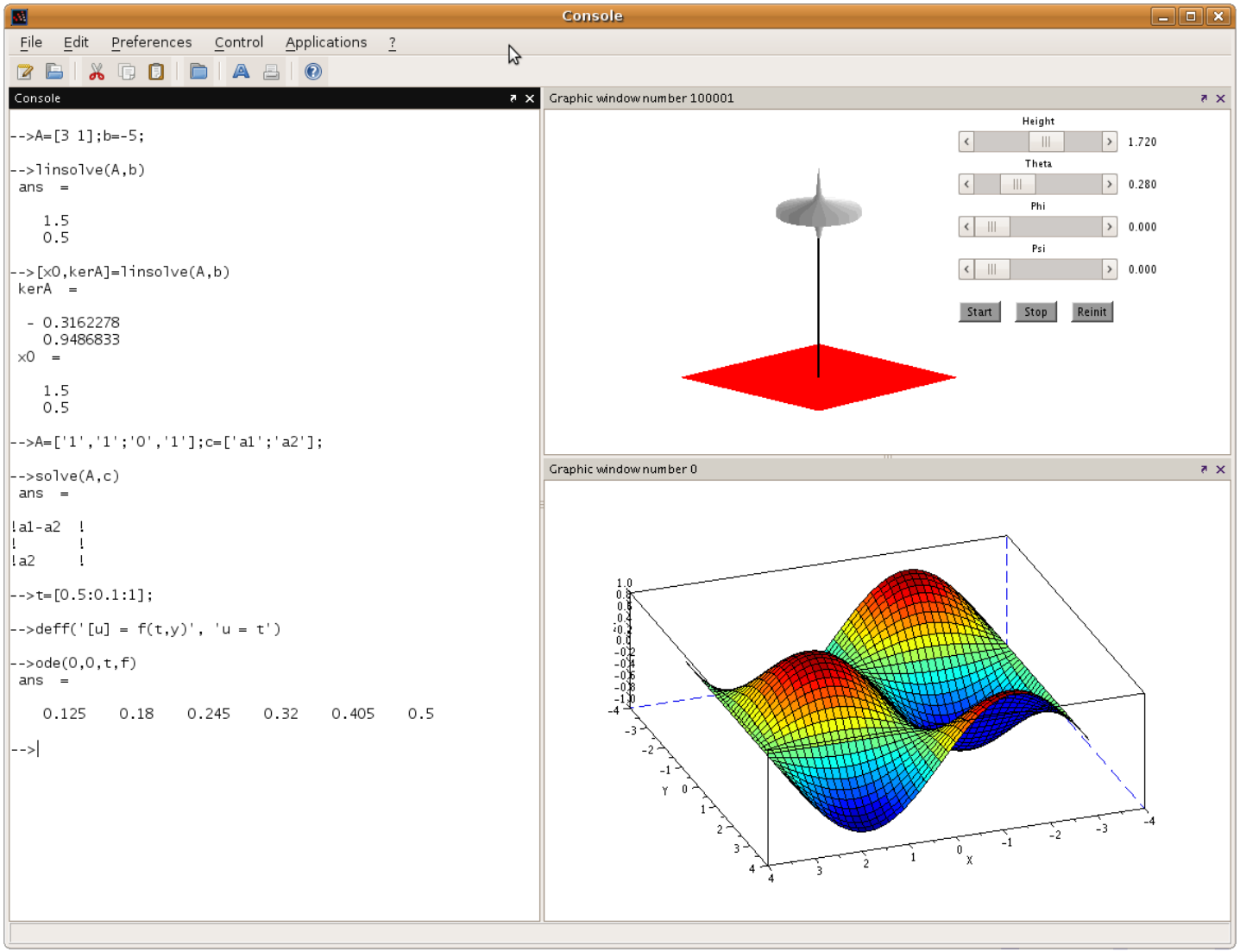
Outline

Scilab System

- Développé par Fondation Digitéo
- Consortium Scilab
- Licence CeCILL (Open Source)
- Windows (9X/2000/XP/Vista), GNU/Linux
- Manuels Utilisateurs Anglais, Français en ligne
- Extensible : Toolbox
- Fonctionnalités de calcul scientifique :
  - Algèbre linéaire
  - Polynômes
  - Interpolation
  - ODE, DEA
  - Scicos: éditeur bloc-diagramme
  - Optimisation, Traitement du Signal, Statistiques, etc...

Scilab Logo

Some Graphics



Nisp : Non Intrusive Spectral Projection

- Scientifique
  - Polynômes de chaos
  - Analyse d'incertitude et de sensibilité
- Logiciel
  - C++ (3 classes, 30 fichiers)
  - Linux
  - Utilise Numerical Recipes

Toolbox Nisp/Scilab dans OPUS

1. Mettre à disposition une libraire de calcul dans un projet open source
   - Portage Windows
   - Remplacer composants propriétaires (Numerical Recipes) par des composants libres
   - Assouplir la librairie (ex : exit, cout)
2. Mettre à disposition un outil d'enseignement
   - Interface Scilab

Nisp dans Scilab - Objectif

```
1  // Un petit exemple, adapté du document de spécification
2  // OPUS - Chaos Polynomial
3  function y = Exemple (x)
4  y(1) = x(1) * x(2);
5  endfunction
6  // Représentation des paramètres incertains
7  vu1 = randomvariable_new("Normale",[1.0  0.5]);
8  value1 = randomvariable_getvalue(vu1);
9  vu2 = randomvariable_new("Uniforme",[1.0  2.5]);
10 value2 = randomvariable_getvalue(vu2);
11 // Création d'une collection de variables aléatoires
12 gu = setrandomvariable_new();
13 gu = setrandomvariable_add(gu,vu1);
14 gu = setrandomvariable_add(gu,vu2);
15 etc...
```

Nisp dans Scilab - Réalisation (1/2)

- Portage Windows
- Interface de la classe RandomVariable

Nisp dans Scilab - Réalisation (2/2)

```
1  // Try randvar_getlog
2  vu1 = randvar_new("Normale",1.0,0.5);
3  randvar_getlog(vu1);
4  randvar_destroy(vu1);
5
6  // Try randvar_getvalue
7  vu1 = randvar_new("Normale",1.0,0.5);
8  nbshots = 1000;
9  values = zeros(nbshots);
10 for i=1:nbshots
11   values(i) = randvar_getvalue(vu1);
12 end
13 computed = mean (values);
14 assert_close ( computed , 1.0 , 1.e-1 );
15 computed = st_deviation (values);
16 assert_close ( computed , 0.5 , 1.e-1 );
17 randvar_destroy(vu1);
```

Perspectives

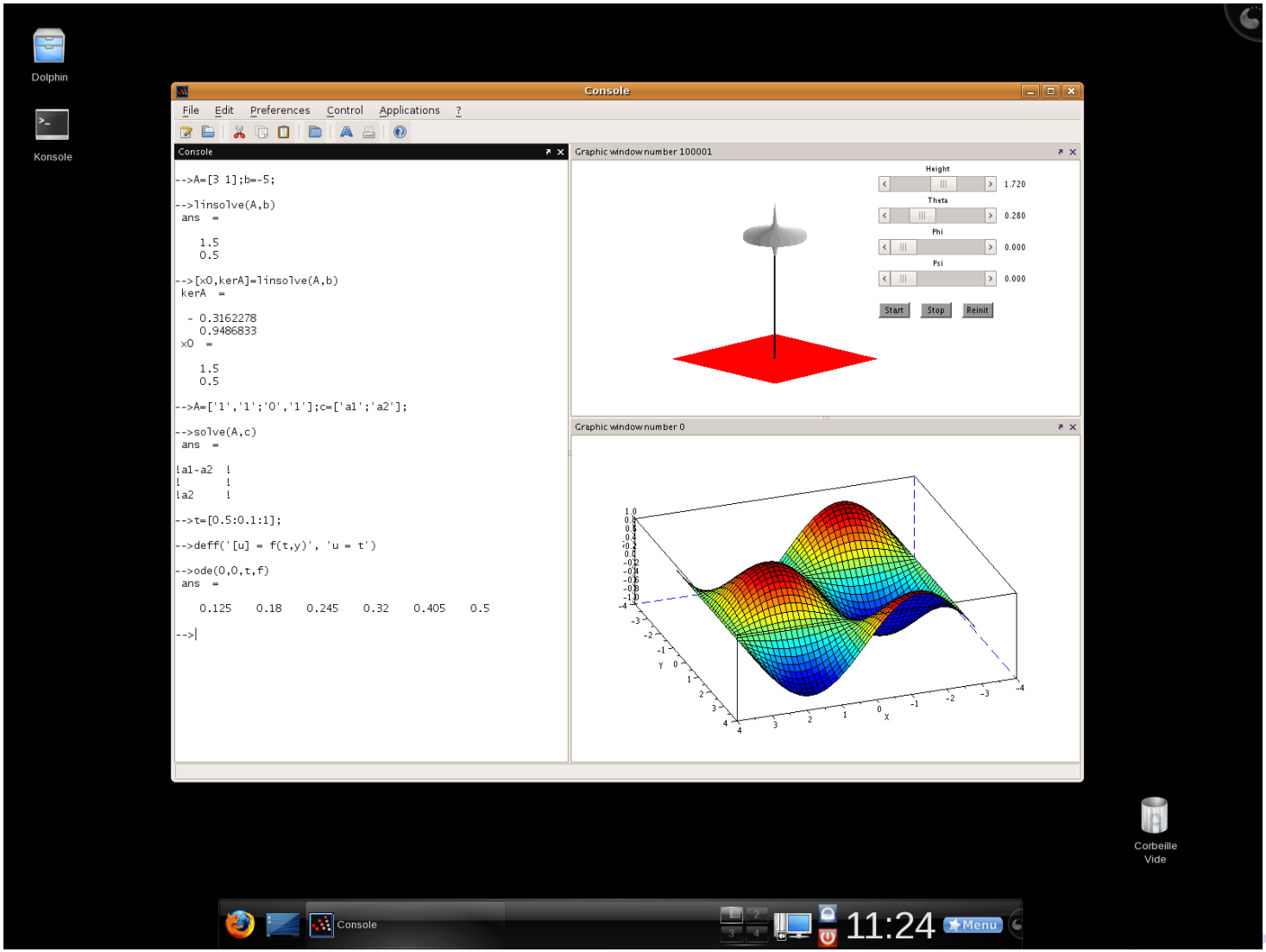
- Décision du projet OPUS
- Modifier la librairie NISP (licences, exit, cout, etc...)
- Interfacer les classes SetRandomVariable, PolynomialChaos
- Tests, Documentation, Mise en ligne

OPUS project

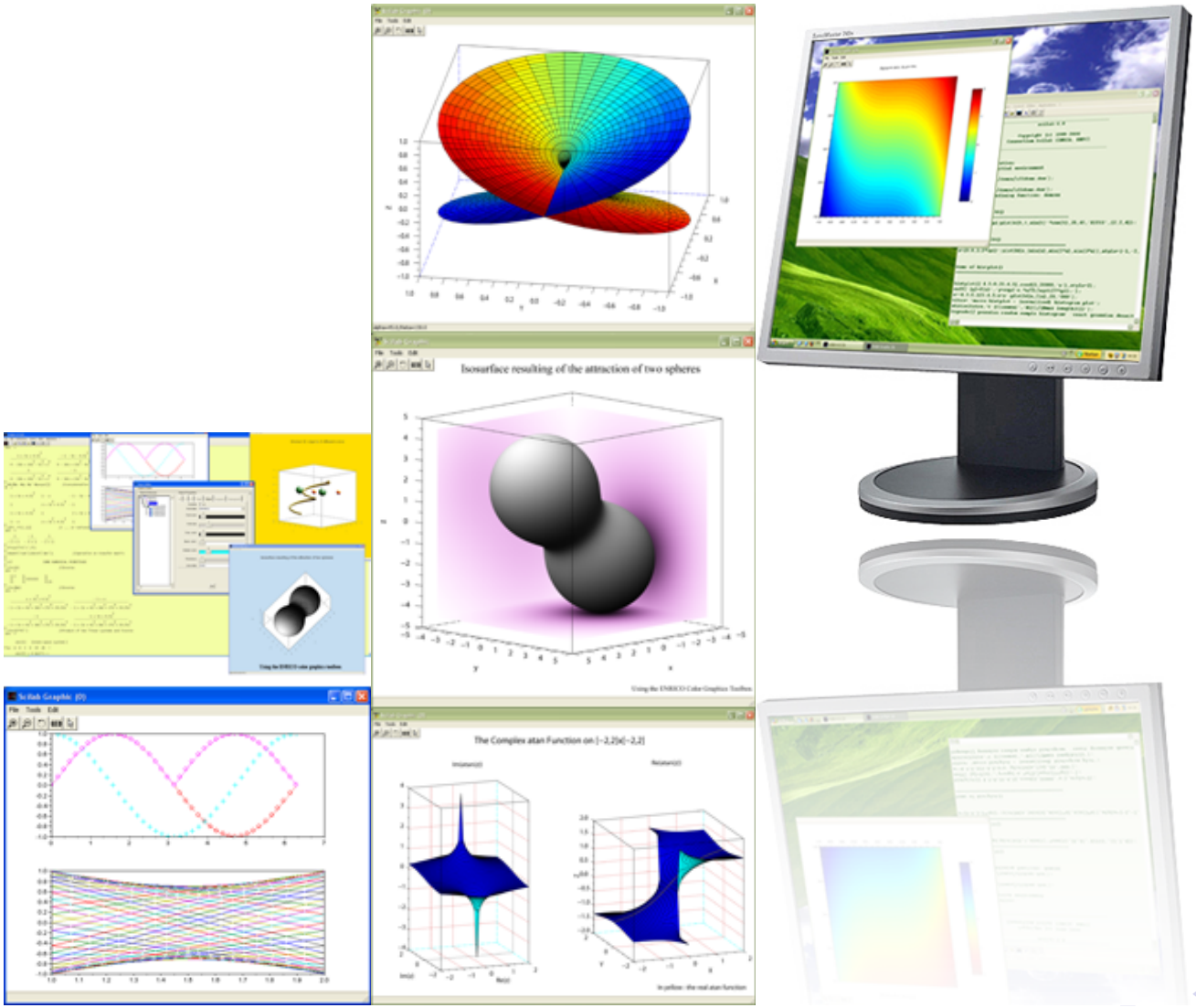
Scilab Features

- 2-D and 3-D graphics, animation
- Linear algebra, sparse matrices
- Polynomials and rational functions
- Interpolation, approximation
- Simulation: ODE solver and DAE solver
- Scicos: a hybrid dynamic systems modeler and simulator
- Classic and robust control, LMI optimization
- Differentiable and non-differentiable optimization
- Signal processing
- Statistics
- Interface with Fortran, Tcl/Tk, C, C++, Java, LabVIEW

Some Graphics



Some Graphics



Nisp dans Scilab

- Portage Windows standard C90
  - gcc sous Linux : standard C99
  - Microsoft Visual Studio 2008 : standard C90
  - création des projets Visual Studio 2008
  - création des dll, avec export/import explicite
  - remplacement de erfc par calerf
  - remplacement de srandom par srand
  - suppression de l'initialisation par PID
  - utilisation de la macro _USE_MATH_DEFINES pour définir la constante PI
- Licence
  - utilisation de Numerical Recipes
- Flexibilité de la librairie
  - instructions exit
  - affichages dans le terminal

Nisp dans Scilab - Objectif (1/5)

```
1  // Un petit exemple, adapté du document de spécification
2  // OPUS - Chaos Polynomial
3  function y = Exemple (x)
4  y(1) = x(1) * x(2);
5  endfunction
6  // Représentation des paramètres incertains
7  vu1 = randomvariable_new("Normale",[1.0  0.5]);
8  value1 = randomvariable_getvalue(vu1);
9  vu2 = randomvariable_new("Uniforme",[1.0  2.5]);
10 value2 = randomvariable_getvalue(vu2);
11 // Création d'une collection de variables aléatoires
12 gu = setrandomvariable_new();
13 gu = setrandomvariable_add(gu,vu1);
14 gu = setrandomvariable_add(gu,vu2);
```

Nisp dans Scilab - Objectif (2/5)

```
1  // Représentation des variables stochastiques
2  vx1 = randomvariable_new("Normale");
3  vx1 = randomvariable_new("Uniforme");
4  gx = setrandomvariable_new();
5  gx = setrandomvariable_add(gx,vx1);
6  gx = setrandomvariable_add(gx,vx2);
7  // Création du Polynôme de chaos
8  pc = polynomialchaos_new(gx);
```

Nisp dans Scilab - Objectif (3/5)

```
1  // Plan d'expériences
2  degre = 2;
3  gx = setrandomvariable_buildsample(gx,"Quadrature",degre);
4  gu = setrandomvariable_buildsample(gu,gx);
5  np = setrandomvariable_size(gu);
6  pc = polynomialchaos_setsizetarget(pc,np);
7  nx = polynomialchaos_getdimensioninput(pc);
8  ny = polynomialchaos_getdimensionoutput(pc);
9  indata = zeros(nx);
10 outdata = zeros(ny);
11 for k=1:np
12   indata = setrandomvariable_getsample(gu,k);
13   outdata = Exemple1(indata);
14   pc = polynomialchaos_settarget(pc,k,outdata);
15 end
```

Nisp dans Scilab - Objectif (4/5)

```
// Analyse des coefficients
pc = polynomialchaos_setdegre(pc,degre);
pc = polynomialchaos_compute(pc,gx,"Integration");
// Edition de l'analyse de sensibilité
average = polynomialchaos_getmean(pc);
var = polynomialchaos_getvariance(pc);
mprintf("Mean     = %e\n",average);
mprintf("Variance     = %e\n",var);
mprintf("Indice de sensibilité du 1er ordre\n");
isx1 = polynomialchaos_getindicefirstorder(pc,1);
isx2 = polynomialchaos_getindicefirstorder(pc,2);
mprintf("Variable x1 = %e\n",isx1);
mprintf("Variable x2 = %e\n",isx2);
```

Nisp dans Scilab - Objectif (5/5)

```
1  // Tracé des données
2  inputx1 = zeros(np);
3  inputx2 = zeros(np);
4  for k=1:np
5    indata = setrandomvariable_getsample(gu,k);
6    inputx1(k) = indata(1);
7    inputx2(k) = indata(2);
8  end
9  plot(inputx1,inputx2);
10 histplot(20,inputx1);
11 histplot(20,inputx2);
```