Toolbox Nisp - Scilab
Revue OPUS

M. Baudin

INRIA

31/03/2009

Outline

1. Scilab
   - Overview
   - Features
   - Logo
   - Graphics
2. Nisp dans Scilab
   - Activités
   - Script (Objectif)
   - Script (Réalisation)
   - Perspectives

Scilab System

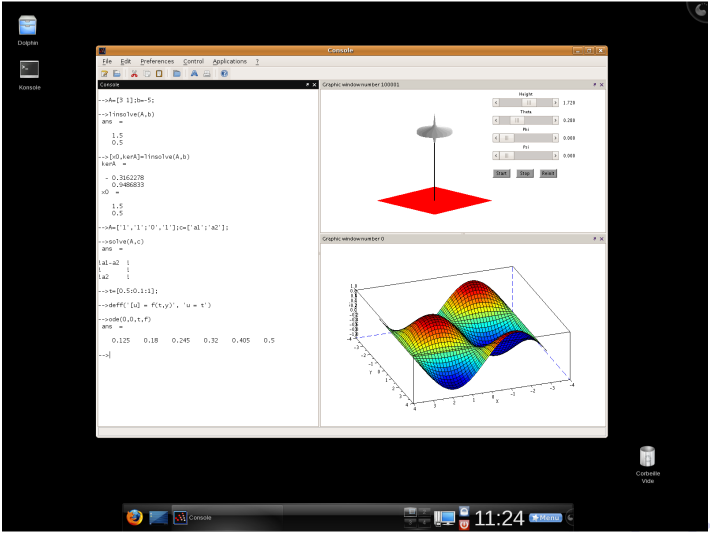
- CeCILL license (free software)
- An open system : new user-defined data types (and operations, with overloading).
- Contributions can be downloaded from Scilab Web site.
- Built on external libraries
- Windows (9X/2000/XP/Vista), GNU/Linux (most Unix systems)
- Source code
- On-line help
- English (French) user manuals
- binary versions available

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

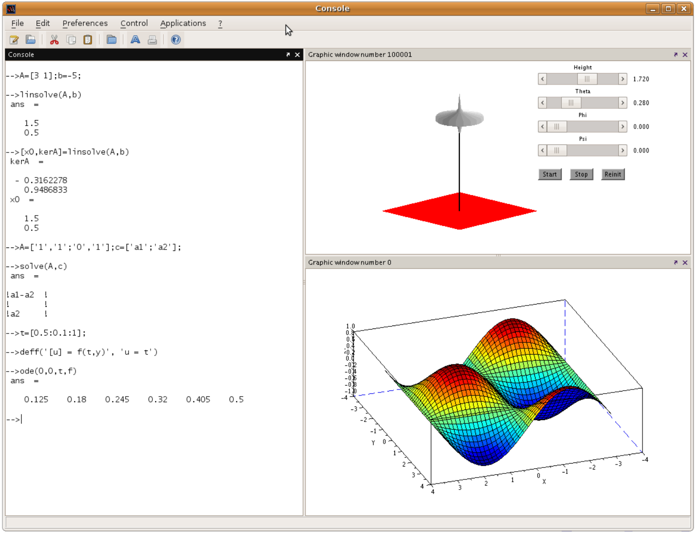
Scilab Logo

Some Graphics



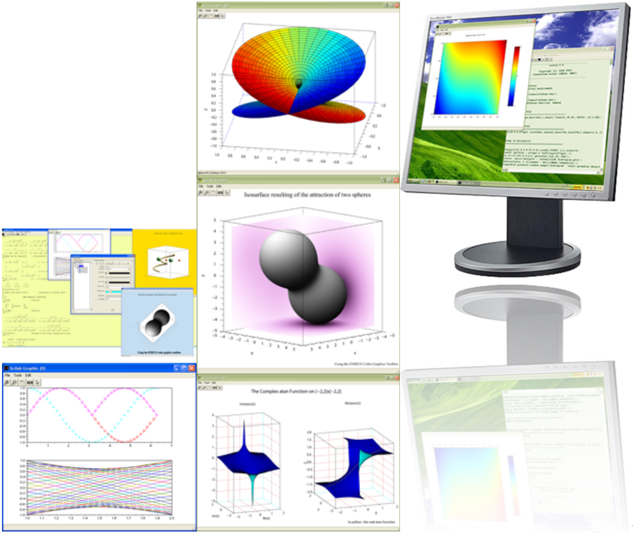
```
-->A=[3 1];b=-5;

-->linsolve(A,b)
 ans  =

    1.5
    0.5

-->[x0,kerA]=linsolve(A,b)
 kerA  =

  - 0.3162278
    0.9486833
 x0  =

    1.5
    0.5

-->A=['1','1';'0','1'];c=['a1';'a2'];

-->solve(A,c)
 ans  =

!a1-a2  !
!       !
!a2     !

-->t=[0.5:0.1:1];

-->deff('[u] = f(t,y)', 'u = t')

-->ode(0,0,t,f)
 ans  =

    0.125    0.18    0.245    0.32    0.405    0.5

-->
```

Some Graphics



```
-->A=[3 1];b=-5;

-->linsolve(A,b)
 ans  =

    1.5
    0.5

-->[x0,kerA]=linsolve(A,b)
 kerA  =

  - 0.3162278
    0.9486833
 x0  =

    1.5
    0.5

-->A=['1','1';'0','1'];c=['a1';'a2'];

-->solve(A,c)
 ans  =

!a1-a2  !
!       !
!a2     !

-->t=[0.5:0.1:1];

-->deff('[u] = f(t,y)', 'u = t')

-->ode(0,0,t,f)
 ans  =

    0.125    0.18    0.245    0.32    0.405    0.5

-->
```

Some Graphics


Isosurface resulting of the attraction of two spheres
Using the ENRICO Color Graphics Toolbox
Scilab Graphic (0)
File Tools Edit
The Complex atan Function on [-2,2]x[-2,2]

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)

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

Nisp dans Scilab - Objectif (2/5)

```
// Représentation des variables stochastiques
vx1 = randomvariable_new("Normale");
vx1 = randomvariable_new("Uniforme");
gx = setrandomvariable_new();
gx = setrandomvariable_add(gx,vx1);
gx = setrandomvariable_add(gx,vx2);
// Création du Polynôme de chaos
pc = polynomialchaos_new(gx);
```

Nisp dans Scilab - Objectif (3/5)

```
// Plan d'expériences
degre = 2;
gx = setrandomvariable_buildsample(gx,"Quadrature",degre);
gu = setrandomvariable_buildsample(gu,gx);
np = setrandomvariable_size(gu);
pc = polynomialchaos_setsizetarget(pc,np);
nx = polynomialchaos_getdimensioninput(pc);
ny = polynomialchaos_getdimensionoutput(pc);
indata = zeros(nx);
outdata = zeros(ny);
for k=1:np
  indata = setrandomvariable_getsample(gu,k);
  outdata = Exemple1(indata);
  pc = polynomialchaos_settarget(pc,k,outdata);
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)

```
// Tracé des données
inputx1 = zeros(np);
inputx2 = zeros(np);
for k=1:np
  indata = setrandomvariable_getsample(gu,k);
  inputx1(k) = indata(1);
  inputx2(k) = indata(2);
end
plot(inputx1,inputx2);
histplot(20,inputx1);
histplot(20,inputx2);
```

Nisp dans Scilab - Réalisation (1/2)

- Interface de la classe RandomVariable
- Windows, Linux

Nisp dans Scilab - Réalisation (2/2)

```
// Try randvar_getlog
vu1 = randvar_new("Normale",1.0,0.5);
randvar_getlog(vu1);
randvar_destroy(vu1);

// Try randvar_getvalue
vu1 = randvar_new("Normale",1.0,0.5);
nbshots = 1000;
values = zeros(nbshots);
for i=1:nbshots
  values(i) = randvar_getvalue(vu1);
end
computed = mean (values);
assert_close ( computed , 1.0 , 1.e-1 );
computed = st_deviation (values);
assert_close ( computed , 0.5 , 1.e-1 );
randvar_destroy(vu1);
```

Perspectives

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

OPUS project