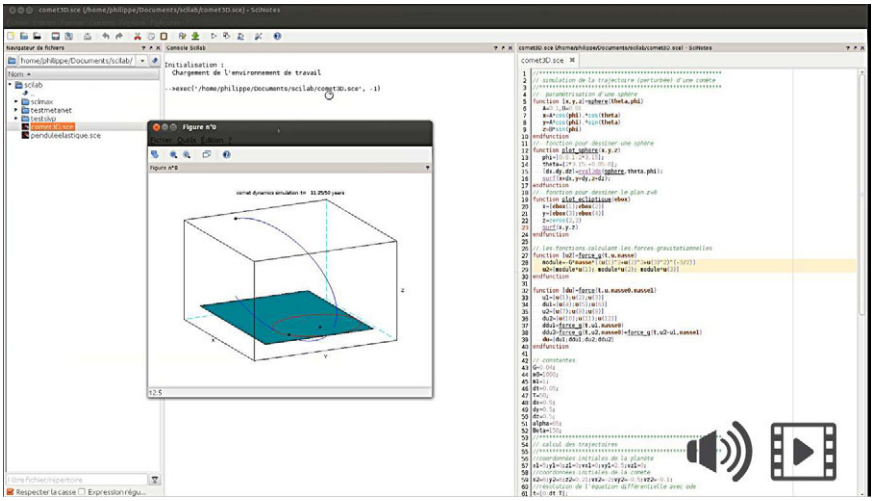
1

Aperçu de Scilab

Scilab est un logiciel de calcul numérique libre et multiplate-forme destiné aussi bien au théoricien qu'à l'ingénieur. Il dispose d'un langage de programmation de haut niveau, bien adapté au formalisme mathématique, et de modules complémentaires dédiés à des applications scientifiques telles que le traitement du signal et des images, l'analyse statistique, les simulations de mécanique, de dynamique des fluides, l'optimisation numérique, la modélisation et la simulation de systèmes dynamiques hybrides, etc. Pour vous donner une petite idée de ses capacités, prenons l'exemple d'une comète en orbite autour du Soleil, dont la trajectoire est perturbée par la présence d'une planète évoluant sur une orbite stable autour du Soleil. Malgré la complexité du problème, vous pouvez facilement, avec Scilab, simuler un tel système et obtenir une animation en trois dimensions du mouvement des différents corps, comme sur la vidéo ci-dessous.

Figure 1.1 : *Simulation d'une orbite cométaire (vidéo)*



Cette animation est obtenue à l'aide d'un programme écrit en langage Scilab et présenté dans Exemple 1.1 suivant.

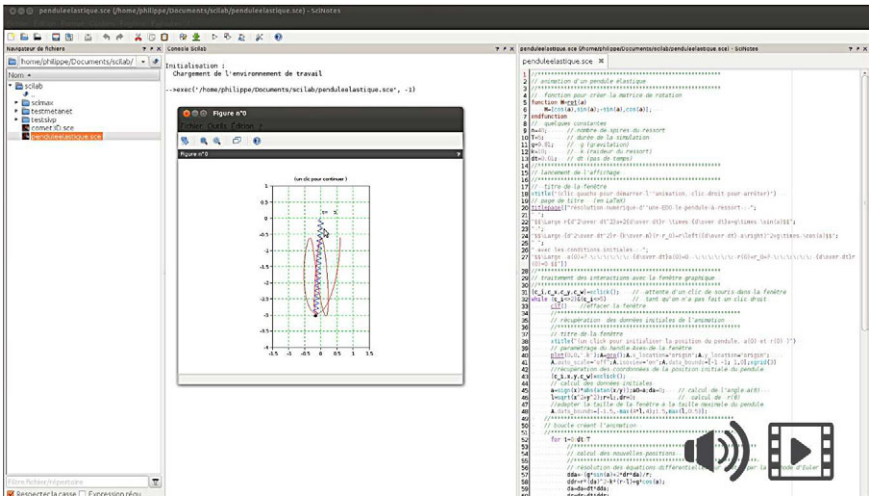
Exemple 1.1 : *Script permettant de visualiser la trajectoire d'une comète autour du Soleil*

```
//*******************************************************
// simulation de la trajectoire (perturbée) d'une comète
//*******************************************************
//  paramétrisation d'une sphère
function [x,y,z]=sphere(theta,phi)
   A=0.1,B=0.01
   x=A*cos(phi).*cos(theta)
   y=A*cos(phi).*sin(theta)
   z=B*sin(phi)
endfunction
//  fonction pour dessiner une sphère
function plot_sphere(x,y,z)
   phi=[0:0.1:2*3.15];
   theta=[2*3.15:-0.05:0];
   [dx,dy,dz]=eval3dp(sphere,theta,phi);
   surf(x+dx,y+dy,z+dz);
endfunction
//  fonction pour dessiner le plan z=0
function plot_ecliptique(ebox)
   x=[ebox(1);ebox(2)]
   y=[ebox(3);ebox(4)]
   z=zeros(2,2)
   surf(x,y,z)
endfunction

// les fonctions calculant les forces gravitationnelles
function [u2]=force_g(t,u,masse)
   module=-G*masse*((u(1)^2+u(2)^2+u(3)^2)^(-3/2))
   u2=[module*u(1); module*u(2); module*u(3)]
endfunction

function [du]=force(t,u,masse0,masse1)
   u1=[u(1);u(2);u(3)]
   du1=[u(4);u(5);u(6)]
   u2=[u(7);u(8);u(9)]
   du2=[u(10);u(11);u(12)]
   ddu1=force_g(t,u1,masse0)
   ddu2=force_g(t,u2,masse0)+force_g(t,u2-u1,masse1)
   du=[du1;ddu1;du2;ddu2]
endfunction

// constantes
G=0.04;
m0=1000;
m1=1;
dt=0.05;
T=50;
dx=0.5;
dy=0.5;
dz=0.5;
alpha=65;
Beta=150;
```

```
//*********************************************************
// calcul des trajectoires
//*********************************************************
//coordonnées initiales de la planète
x1=5;y1=0;z1=0;vx1=0;vy1=2.5;vz1=0;
//coordonnées initiales de la comète
x2=6;y2=6;z2=0.21;vx2=-2;vy2=-0.5;vz2=-0.1;
//résolution de l'équation différentielle avec ode
t=[0:dt:T];
u0=[x1; y1; z1; vx1; vy1; vz1; x2; y2; z2; vx2; vy2; vz2];
u=ode(u0,0,t,list(force,m0,m1));
//  récupération des résultats
X=[u(1,:)',u(7,:)'];
Y=[u(2,:)',u(8,:)'];
Z=[u(3,:)',u(9,:)'];
//*********************************************************
// lancement de la fenêtre graphique
//*********************************************************
ebox=[min(X),max(X),min(Y),max(Y),min(Z),max(Z)];
N=length(t);                        // nombre d'étapes
drawlater()
plot_ecliptique(ebox) // tracé du plan de l'écliptique
plot_sphere(0,0,0)                  // soleil
plot_sphere(X(1,1),Y(1,1),Z(1,1))  // planète
plot_sphere(X(1,2),Y(1,2),Z(1,2))  // comète
A=gca();
A.axes_visible=["off" "off" "off"];
A.rotation_angles=[alpha Beta];
A.data_bounds=ebox;
drawnow()
//*********************************************************
// boucle principale créant l'animation graphique
//*********************************************************
for k=1:5:N
   Beta=Beta+k/300;                     //  angle de vue
   realtimeinit(0.05)                  // unité de temps
   drawlater()           // ouverture du buffer graphique
   clf()                 // on efface le buffer graphique
   plot_ecliptique(ebox) // tracé du plan de l'écliptique
   param3d1(X(1:k,:),Y(1:k,:),...        // affichage des
   list(Z(1:k,:),[5,2]))                 // trajectoires
   plot_sphere(0,0,0)                       // le soleil
   plot_sphere(X(k,1),Y(k,1),Z(k,1))       // la planète
   plot_sphere(X(k,2),Y(k,2),Z(k,2))        // la comète
   title('comet dynamics simulation : t='+msprintf(...
   '%2.2f',t(k))+'/'+string(T)+' years')      // le titre
   xinfo(string(t(k)))              // affichage du temps
   A=gca();  // redimensionnement de la fenêtre graphique
   A.axes_visible=["off" "off" "off"];
   A.rotation_angles=[alpha Beta];  // rotation pt de vue
   A.data_bounds=ebox;
   drawnow()             // affichage du buffer graphique
   realtime(k)// pause pour ajuster le rythme d'affichage
end
```

Cet exemple donne un petit aperçu des possibilités de Scilab aussi bien en matière graphique qu'au niveau du calcul numérique. Avec un peu de maîtrise, vous pourrez rapidement aller plus loin et créer des programmes permettant à un utilisateur d'interagir avec l'interface graphique de Scilab pour générer ses propres animations, telle que celle présentée sur la vidéo ci-dessous.

Figure 1.2 : *Simulation d'un pendule accroché à un ressort (vidéo)*



Dans cet exemple, nous simulons la trajectoire d'un pendule accroché à un ressort (de raideur k) que l'on laisse osciller librement après l'avoir lâché sans élan depuis une position donnée. L'évolution du pendule se modélise par un système d'équations différentielles faisant intervenir deux variables :

$$\begin{cases} r \times \frac{\mathrm{d}^2 a}{\mathrm{d}t^2} + 2\frac{\mathrm{d}r}{\mathrm{d}t} \times \frac{\mathrm{d}a}{\mathrm{d}t} = g \times \sin(a) \\ \frac{\mathrm{d}^2 r}{\mathrm{d}t^2} - \frac{k}{m}(r - r_0) = r \times \left(\frac{\mathrm{d}a}{\mathrm{d}t}\right)^2 + g \times \cos(a) \end{cases}$$

avec :

- a l'angle du pendule par rapport à la verticale ;
- r la longueur du ressort constituant le pendule ;

et les conditions initiales : $a(0) = a_0, \quad \frac{\mathrm{d}a(0)}{\mathrm{d}t} = 0, \quad r(0) = r_0, \quad \frac{\mathrm{d}r(0)}{\mathrm{d}t} = 0.$

Pour étudier un tel système, très sensible aux conditions initiales, il est important de pouvoir modifier facilement la position de départ du pendule et observer l'influence de chaque paramètre sur le mouvement. Ici Scilab permet de créer une interface graphique qui simplifie la saisie des paramètres et de se concentrer sur l'analyse des résultats. Avec le script de l'Exemple 1.2, vous pouvez modifier la position initiale du pendule d'un simple clic de souris dans la fenêtre graphique puis étudier son mouvement.

Exemple 1.2 : *Script permettant de modifier la position initiale du pendule*

```
//*********************************************************************
// animation d'un pendule élastique
//*********************************************************************
//  fonction pour créer la matrice de rotation
function M=rot(a)
    M=[cos(a),sin(a);-sin(a),cos(a)];
endfunction
//  quelques constantes
n=40;       // nombre de spires du ressort
T=5;        // durée de la simulation
g=9.81;     //  g (gravitation)
k=10;       //  k (raideur du ressort)
dt=0.01;    // dt (pas de temps)
//*********************************************************************
// lancement de l'affichage
//*********************************************************************
//  titre de la fenêtre
xtitle("(clic gauche pour démarrer l''animation, clic droit pour
 arrêter)")
// page de titre (en LaTeX)
titlepage(["résolution numérique d''une EDO le pendule à ressort : ";"
 "; "$$\Large r{d^2\over dt^2}a+2{d\over dt}r \times {d\over dt}a=g
\times \sin(a)$$";
" "; "$$\Large {d^2\over dt^2}r-{k\over m}(r-r_0)=r\left({d\over dt}
 a\right)^2+g\times \cos(a)$$";" "; " avec les conditions initiales :
 "; "$$\Large  a(0)=? \;\;\;\;\;\;\; {d\over dt}a(0)=0  \;\;\;\;\;\;\;
 r(0)=r_0=? \;\;\;\;\;\;\; {d\over dt}r(0)=0 $$"])
//*********************************************************************
// traitement des interactions avec la fenêtre graphique
//*********************************************************************
// attente d'un clic de souris dans la fenêtre
[c_i,c_x,c_y,c_w]=xclick();
while (c_i<>2)&(c_i<>5)   // tant qu'on n'a pas fait un clic droit
      clf()    //effacer la fenêtre
      //*************************************************************
      // récupération des données initiales de l'animation
      //*************************************************************
      // titre de la fenêtre
      xtitle("(un click pour initialiser la position du pendule, a(0) et
 r(0) )")
      // paramétrage du handle Axes de la fenêtre
      plot(0,0,'.k');A=gca();A.x_location="origin";
      A.y_location="origin";A.auto_scale="off";A.isoview="on";
```

```
A.data_bounds=[-1 -1; 1,0];xgrid(3)
// récupération des coordonnées de la position initiale
// du pendule :
[c_i,x,y,c_w]=xclick();
// calcul des données initiales :
a=sign(x)*abs(atan(x/y));a0=a;da=0;  // calcul de l'angle a(0)
l=sqrt(x^2+y^2);r=l;,dr=0;           //  calcul de  r(0)
// adapter la taille de la fenêtre à la taille maximale
// du pendule :
A.data_bounds=[-1.5,-max(4*l,4);1.5,max(l,0.5)];
//*********************************************************
// boucle créant l'animation
//*********************************************************
for t=0:dt:T
     //*****************************************************
     // calcul des nouvelles positions
     //*****************************************************
     // résolution des équations différentielles sur a et r
     // par la méthode d'Euler
     dda=-(g*sin(a)+2*dr*da)/r;
     ddr=r*(da)^2-k*(r-l)+g*cos(a);
     da=da+dt*dda;
     dr=dr+dt*ddr;
     a=a+dt*da;
     r=r+dt*dr;
     // calcul de la ligne  traçant le ressort
     ressortr=linspace(0,r,n)';        // étirement du ressort
     // coordonnées transversales à l'axe du ressort -> /\/\/\
     ressorta=[0;(-1).^[0:n-3]';0]*(l/10);
     //rotation de l'image du ressort selon l'angle a
     x=[x;r*sin(a)];
     y=[y;-r*cos(a)];
     M=-rot(-a);
     N=[ressortr,ressorta]*M;
     ressorty=N(:,1);ressortx=N(:,2);
     //*****************************************************
     // affichage du pendule
     //*****************************************************
     drawlater()  // écriture dans le buffer graphique
     clf()        // effacer la fenêtre
     plot(ressortx,ressorty) //affichage du ressort du pendule
     xstring(0,0.1,["t=" string(t)]) // temps écoulé
     // la boule du prendule :
     xfarc(r*sin(a)-0.05,-r*cos(a)+0.05,0.1,0.1,0,360*64)
     // redimensionnement de la fenêtre graphique
     A=gca();A.data_bounds=[-1.5,-max(4*l,4);1.5,max(l,0.5)];
     A.auto_scale="off";A.isoview="on";
     A.axes_visible=["off" "off" "off"];
     drawnow()            // afficher le buffer graphique
     realtime(t);           // delai d'affichage
end
//*********************************************************
// choix d'une nouvelle animation ou d'une sortie du script
//*********************************************************
xtitle("(un clic pour continuer )")    // titre de la fenêtre
plot(x,y,'-r')                       // affichage trajectoire
```

```
      A=gca();A.isoview="on";xgrid(3); //afficher une grille (verte)
      // attente d'un clic de souris dans la fenêtre graphique :
      [c_i,x,y,c_w]=xclick();
      clf();                          // choix d'une nouvelle action
      xtitle("(clic gauche pour démarrer l''animation, clic droit pour
 arrêter)")
      plot(0,0,'.k');A=gca();A.x_location="origin";
      A.y_location="origin";
      // attente d'un clic de souris dans la fenêtre :
      [c_i,x,y,c_w]=xclick();
end
```

Ces deux exemples font appel à des notions de base d'utilisation de Scilab, que nous allons vous présenter dans la suite de ce livre. À la fin de votre lecture, vous serez à même de les réaliser par vous-même, comme nous vous le montrerons au dernier chapitre Deux études de cas : le pendule et l'orbite cométaire.

Attention › *Pour une bonne exécution des scripts ci-dessus, utilisez Scilab 5.4.1 ou +.*