09/2017

Bref résumé : résolutions d'équations, systèmes d'équations linéaires et équations différentielles sous SCILAB

1. *Définition d'une fonction mathématique sous SCILAB*

Si est f(x_1,x_2,...) est une fonction quelconque d'une ou plusieurs variables x_1,x_2, ..., la fonction f peut se définir de deux manières sous Scilab : en ligne dans le programme principal en utilisant le mot clé « **deff** » dans le programme principal ou « **function** » dans le programme ou un fichier indépendant.

L'écriture générale du code est la suivante :

deff('[s1,s2,...]=mafonction(e1,e2,....)', 'text', [,opt])

ou :

function [s1,s2,...]=mafonction(e1,e2,....)
Instructions
endfunction

[s1,s2,...] sont les arguments de sorties de la fonction et (e1,e2,....) sont les arguments d'entrées de la fonction.
Exemple :
f(x,y)=x+y^2 et calcul du résultat pour x=3 et y=5

```
deff("[somme]=mafonction(x,y)","somme=x+y^2")
somme=mafonction(3,5)
```

Si l'on souhaite calculer aussi la différence, on peut utiliser la même ligne. Dans ce cas, l'argument de sortie est un vecteur de deux éléments : la somme et la différence.

```
deff("[somme, difference]=mafonction(x,y)","somme=x+y^2,difference= x-y^2")
[somme,difference]=mafonction(3,5)
```

On peut écrire aussi f(x,y) en tant que fonction indépendante :

```
function [somme]=mafonction(x, y)
 somme=x+y^2
endfunction
mafonction(3,5)
```

La définition de la fonction peut se faire dans le programme ou dans un fichier indépendant. Supposons qu'un programme contenant seulement la définition de la fonction mafonction a été créé et enregistré sous le nom de fichier *fonction1.sci*. Un programme différent peut faire appel à cette fonction et l'exécuter. Les deux programmes doivent être dans le même répertoire du travail. Autrement, il faut donner le chemin complet du fichier fonction1.sci.

```
exec("fonction1.sci")
mafonction(3,5)
```

Remarques concernant l'utilisation du mot clé *function* :

Que ce soit dans un fichier différent ou dans le même programme, la fonction peut être appelée directement sans arguments de sortie. Si dans l'invite de commande Scilab, on

demande la variable « somme », un message d'erreur indique que « somme » n'est pas définie. En effet, « somme » est dite variable locale et elle est utilisée seulement dans le corps de la fonction. Pour que les variables locales soient accessibles dans le programme principal, il faut les déclarer en tant que variables globales en utilisant le mot clé **global dans le code correspondant à la définition de la fonction et dans le programme principal qui l'appelle**. En conclusion, la communication entre la fonction et le programme principal se fait grâce au mot clé : **global**.

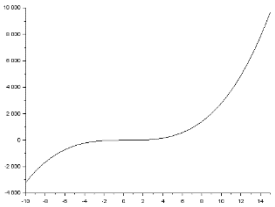
2. *Résolution d'équations sous SCILAB*

La résolution des équations sous Scilab peut se faire avec le mot clé **fsolve**.

```
function y=fct(x)
y=3*x^3-2*x^2-2
endfunction

x=[-10:0.1:15];
plot2d(x,fct(x));

xz=fsolve([-6,6],fct)
```



La résolution est numérique et dépend de l'intervalle d'initialisation de la recherche de la solution. Ici [-6,6]. La solution doit être vérifiée.

2.1 La résolution d'un système d'équations linéaires

Considérons le système d'équations suivantes :

$4*x_1+3*x_2+5*x_3 = 25$
$-x_1 - x_2 +10*x_3 = 30$
$x_1 +10*x_2 - x_3 = 2$

x_1, x_2 et x_3 sont les inconnus du système.

Sous scilab, ce système peut s'écrire sous forme de matrices : B=A*X.

```
A= [4 3 5; -1 -1 10; 1 10 -1]
B=[25; 30; 2]
```

L'exécution de ces deux lignes donne :

```
->A
 A  =
    4.    3.    5.
  - 1.  - 1.    10.
    1.    10.  - 1.

-->B
 B  =
    25.
    30.
    2.
```

Le « ; » indique un changement de ligne lors de l'écriture de la matrice. L'espace entre deux éléments indique que les deux éléments sont sur la même ligne de la matrice ou du tableau.

Quelques opérations sur les matrices (tableaux) :
A(1,1) est le premier élément de la matrice A

A(:,1) est un vecteur colonne contenant toutes les lignes mais de la première colonne seulement.
A(1, :) est un vecteur ligne contenant tous les éléments de la ligne 1.

sum(A(1, :)) : donne la somme des éléments de la première colonne de A (ici =4)
sum(A(:,1)) : donne la somme des éléments de la première ligne de A (ici =12)
On peut de la même manière utiliser les fonctions : min, max, mean, stdev, etc.
mean(A(2, :)) : 8/3

A*B est le produit matriciel de A et B. Il n'est possible que lorsque le nombre de colonne de la matrice de gauche est égal au nombre de ligne de la matrice de droite. A*B est donc possible mais B*A n'est pas autorisé.
Attention de ne pas confondre le produit de deux matrices et la multiplication de deux vecteurs ou deux colonnes élément par élément. Si P et Q sont deux vecteurs ou de lignes : P**.***Q est la multiplication élément par élement de P par Q. L'opérateur utilisé est : **.***
P=[2 3 4 8] ;
Q=[4 4 3 2] ;
P.*Q=[8 12 12 16]

A' est le transposé de A : les lignes deviennent colonnes et les colonnes deviennent les lignes.
transposeA=A'
transposeA=[4. - 1. 1. ; 3. - 1. 10. ; 5. 10. - 1.]
Notez que A*B = B'*A' ;

Pour trouver la solution du système linéaire décrit ci-dessus qu'on note X :
X=AB (division matricielle à gauche : x=A\B est une solution de A*X=B)
Le terme division est un abus de langage. Il s'agit en réalité d'une multiplication de l'inverse de A par B.

La solution de :

$4*x_1+3*x_2+5*x_3 = 25$
$-x_1 - x_2 +10*x_3 = 30$
$x_1 +10*x_2 - x_3 = 2$
X = [x1;x2;x3] ;

X=A\B
X=inv(A)*B
->X= [1.968599 ; 0.3260870 ; 3.2294686]

Pour plus de détails sur les matrices, référez-vous à l'exercice (p.32 du TD).

2.2 La résolution d'un système d'équations différentielles de premier ordre

Une équation différentielle est une équation entre une fonction et sa ou ses dérivées. La fonction constitue l'inconnue. Les inconnues ne sont donc pas des scalaires mais des fonctions.
La fonction suivante :
dP/dt = r*P est une équation différentielle de premier ordre (dérivée première) où la fonction est P = dP/rdt.
Toutes les équations différentielles ne possèdent pas une solution analytique comme dans le cas ci-dessus où P = P0 * exp(rt) (si t0 = 0 et P au temps 0 = P0). Dans ce cas, il faut faire appel à des méthodes de résolutions numériques. La solution rendue par ses méthodes n'est pas une fonction analytique mais un tableau de valeurs qui donne la valeur prise par la fonction Y à chaque pas de temps.

La méthode la plus simple est la **méthode d'Euler** qui s'appuie sur la méthode itérative suivante :
dP/dt = r*P ou dP = r*P*dt ; c'est de la forme dy=f(x,y) dx
L'approximation d'Euler donne :
y(x+dx)-y(x)=f(x,y) dx = y(x+dx)=y(x)+f(x,y) dx.
Le calcul se fait d'une manière itérative :
y(n+1)=y(n)+f(x(n),y(n))*dx.

Dans l'exemple ci-dessus f(x ,y)=r*P. On suppose que lorsque t=0, P(0)=100. On prend dt=0.1 et r=1.1
t(0)=0, P(0)=100 ;
t(1)=t(0)+dt=0.1, P(1)=P(0)+f(t(0),P(0))*dt = 100+ 1.1*P(0)*0.1=111
t(2)=t(1)+dt=0.1+0.1 ; P(2)=P(1)+f(t(1),P(1))*dt = 111+1.1*P(1)*0.1=123.21
ect.

Sous Scilab :

```
r=1.1
Pinit=100
dt=0.1
P(1)=Pinit
k=1
temps(1)=0
for i=0:dt:5
   k=k+1
   P(k)=P(k-1)+r*P(k-1)*dt
   temps(k)=temps(k-1)+dt
end
plot2d(temps,P)
```

Une autre façon de faire est d'utiliser la commande « **ode** » de Scilab. Son application nécessite l'écriture de la fonction. ODE pour ordinary differential equations trouve une solution numérique de la même manière que la méthode d'Euler mais elle est plus précise. Pour utiliser la méthode de Runge-Kutta d'ordre 4 plus précise que la méthode d'Euler, ode peut être appelée avec l'option **rk**.

```
function ydot=fct(t, y)
   global r
    ydot=r*y
endfunction
global r
dt=0.1
r=1.1
t=0:dt:5
result=ode("rk",100,0,t,fct)
plot2d(t,result)
```

-ode(100,0,t,fct) correspond à l'appel suivant : ode(y0,t0, t vecteur temps de simulation, fonction).

Pour un système d'équations différentielles de premier ordre :
Relations proies-prédateurs :

L'évolution des populations de proies X et de prédateurs Y est définie comme suit :

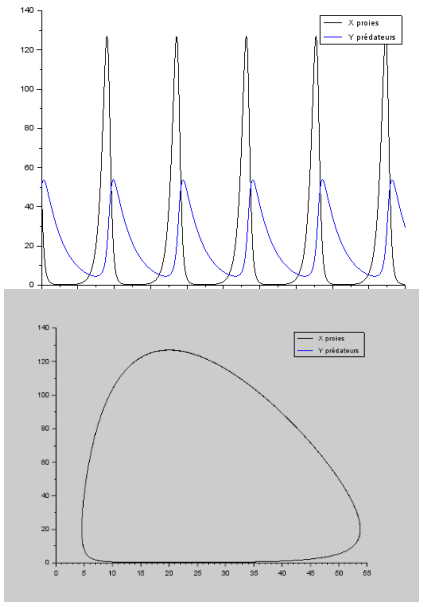
$$\frac{dX}{dt} = r \times X - g \times X \times Y$$

r est le taux d'accroissement de la population de proies et g le taux d'efficacité de la prédation. Cette relation suppose que les ressources sont non limitantes et que la seule limitation de l'accroissement de la population est provoquée par les prélèvements par les prédateurs.

$$\frac{dY}{dt} = e \times g \times X \times Y - m \times Y$$

```
sol=[];
function [ydot]=f(t, y)
global r g e m
proies=r*y(1)-g*y(1)*y(2);
pred=e*g*y(1)*y(2)-m*y(2);
ydot=[proies;pred];
endfunction
global r g e m
r=1; g=0.05; e=0.2; m=0.2;
t=0:0.1:100;
y0=[50;50]
sol=ode("rk",y0,0,t,f);

effectif=sol';
//permet de transformer le tableau sol en un tableau de deux
colonnes
proies=effectif(:,1);
predateurs=effectif(:,2);
xset("window",1)
//permet d'ouvrir une fenêtre 1 pour la figure suivante
figure(1)
plot2d(t,[proies predateurs])
figure(2)
plot2d(proies,predateurs)
```



.///