Université de Rouen
Master 2, MFA 2010–2011
Outils informatiques pour les mathématiques

Initiation Scilab. Corrigé

1. Résolution d'équations non linéaires

Scilab possède un solveur approché de résolution de systèmes d'équations non linéaires du type $f(x) = 0$ nommé *fsolve* prenant en argument d'entrée un point initial et une fonction (définie séparement dans un script de type fonction ou avec l'instruction *deff*.)

Sur l'exemple de la recherche de la solution de l'équation $\cos(x) = x$, on compare le nombre d'évaluations et le temps de calcul pour obtenir un résultat à la précision 10^{-15}. Le script ci-dessous permet de comparer les deux méthodes (temps de calcul, nombre d'itérations). Les vitesses de convergence des méthodes de dichotomie, du point fixe ainsi que celles de Newton et de la fausse position sont illustrées dans la Figure 1 (voir Script 1 correspondant).

```
format('v',20); //  pour afficher 17 chiffres  apres la virgule
write(%io(2),'comparaison dichotomie/pt fixe pour cos(x)=x');
prec=evstr(x_dialog('precision souhaité:','1E-15'));//(l'utilisateur
// modifie  la precision
deff('y=f(x)','y=cos(x)-x')
deff('y=g(x)','y=cos(x)');
timer();
a=0;b=1;fa=f(a);fb=f(b);n1=int(-log(2*prec)/log(2))+1;// nombre
//d'iterations n1 correspondant a la precision prec
for i=1:n1;
c=(a+b)/2;fc=f(c);
if (fa*fc<0) then
  b=c;fb=fc;
else
  a=c;fa=fc;
end
end
t1=timer();
x=1/2;n2=int(log(prec*2*(1-sin(1)))/log(sin(1)))+1;
for i=1:n2;
x=g(x);
end
t2=timer();
disp('resultat (meth1/meth2):');disp([(a+b)/2,x]);
disp('temps de calcul (meth1/meth2):');disp([t1,t2]);
disp('nombre de boucles (meth1/meth2):');disp([n1,n2]);
deff('y=f(x)','y=4/(1+x^2)');
t1=1/2*(f(0)+f(1));t=log(abs(t1-%pi));
for p=1:10;
n=2^(p-1);s=0;
for j=1:n
 s=s+f((2*j-1)/(2*n));
end
```

```
t1=1/2*(t1+1/n*s);t=[t,log(abs(t1-%pi))];
end
plot2d(0:10,t)
```

2. Résolution d'équations ou de systèmes d'équations différentielles

L'instruction *ode* permet la résolution approchée de tout problème de Cauchy sur un intervalle donné. Le second membre de l'EDO $y' = f(t, y)$ est en particulier rentré comme argument sous la forme d'une fonction ayant toujours deux variables — même quand l'équation est autonome— t (variable réelle) et y (vecteur) :

```
function $[f]=$MonSecondMembre$(t,u)$
Ici, le code donnant les composantes de $f$ en fonction de $t$ et  de  $u$.
endfunction
```

Avant toute étude d'équation différentielle, il est cependant prudent de s'assurer du bon conditionnement du probème de Cauchy d'un point de vue continu ou numérique. Les exemples ci-dessous soulignent l'intérêt d'une telle précaution.

3.

Certaines difficultés peuvent apparaître dans la mise en œuvre des algorithmes de résolution numériques. On rappelle qu'un problème de Cauchy est dit mathématiquement bien posé s'il admet une unique solution et que celle-ci dépend continûment de la donnée initiale.

Exemple de problème mal posé. — On considère le problème :

$$y = 2\sqrt{|y|}, \quad t \in [0, +\infty[; y(0) = 0..$$

Ce problème admet les solutions $y(t) = 0$ et $y(t) = t^2$ et plus généralement

$$\begin{cases} y(t) = 0, & t \in [0, a] \\ y(t) = (t-a)^2, & t \in [a, +\infty[\end{cases}$$

L'utilisation de la méthode d'Euler explicite conduit aux solutions approchées suivantes : i) si $y_0 = 0$ alors $y_{app}(t) = 0$
ii) si $y_0 = \varepsilon$ (donnée initiale perturbée), alors $y_{app}(t) \simeq (t + \sqrt{\varepsilon})^2$ (quand $h \to 0$)
Ici, il n'y a ni unicité, ni continuité de la solution. Ce problème de Cauchy est mathématiquement mal posé et son approximation est inutile.

Utiliser la méthode d'Euler explicite à pas constant pour résoudre les EDO suivantes :

$$y' = f(t, y(t)) = 3y - 1, y(0) = 1/3 \quad \text{sur} \quad [0, 30] \quad \text{partant de} \quad y_0 = 0.33333$$

Ce problème de Cauchy est bien posé car f est Lipschitzienne par rapport à y. Par ailleurs, ce problème est numériquement mal posé puisque, comme le montrent les calculs ci-dessous, la continuité de la solution par rapport à la donnée initile n'est pas "suffisamment bonne" ou encore que la constante de Lipschitz n'est pas petite relativement à la précision des calculs. Le prob—ème suivant montre que même un prob—ème numériquement bien posé peut soulever des difficultés inattendues :

$$y' = -150y + 30, y(0) = 1/5 \quad \text{sur} \quad [0, 1].$$

Pour cette donnée initiale, la solution est $y(t) = 1/5$. Si l'on considère $\tilde{y}(0) = 1/5 + \varepsilon$ comme donnée initiale perturbée, alors la solution est $\tilde{y}(t) = 1/5 + \varepsilon e^{-150t}$. On montre que la méthode d'Euler explicite avec un pas $h = 1/50$ approche, l'extrémité de l'intervalle, $y(50)$ par $\tilde{y}_{50} = 1/5 + 2^{50}\varepsilon \simeq 1/5 + 10^{15}\varepsilon$! (voir Figure 3) Bien que le problème soit bien posé, il est nécessaire de prendre un pas de discrétisation assez petit, et faire des calculs assez couteûx. Ce problème est dit mal conditionné ou raide. Ce type de problèmes sont résolus avec Scilab avec l'instruction ode en précisant le type *stiff*[1] en premier argument (voir l'aide en ligne).

[1] raide

a) Écrire une fonction scilab donnant le second membre de l'équation différentielle d'ordre 2, dite de Van Der Pol :

$$y'' = 0.4(1 - y^2)y' - y.$$

b) tiliser la commande *ode* pour résoudre l'équation sur $[0, 2]$ en partant de données sur l'inconnue y et sa dérivée y' à l'instant $t = 0$. On reformule cette équation d'ordre 2 comme un système de deux équations différentielles du premier ordre en posant $u_1 = y(t)$ et $u_2 = y'(t)$. On obtient :

$$\frac{d}{dt}\begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} u_2(t) \\ 0.4(1 - u_1^2(t))u_2(t) - u_1(t) \end{bmatrix}.$$

```
function[f]=VanDerPol(t,u)
f(1)=u(2)
 f(2)=0.4*(1-u(1)^2)*u(2)-u(1)
endfunction
```

Ensuite, un appel à *ode* pour résoudre l'équation de t_0 à T, en partant de u_0 (vecteur colonne), et en voulant récupérer la solution aux instants $t(1) = t_0$, $t(2)$,..., $t(m) = T$, se fait comme suit :

```
t= linspace (t0,T,m)
 [u]=ode(u0,t0,t, VanDerPol)
```

Résolution de l'equation de Van Der Pol (voir Figure 4)

```
m=500; T=30;
t= linspace(0,T,m) //les instants pour lesquels on récupère la solution
u0=[-2.5;2.5]  //donnee initiale
[u]=ode(u0,0,t,VanDerPol);
plot2d(u(1,:)',u(2,:)',2)//voir l'aide en ligne pour le graphisme
```

4. Espace de phase

L'espace de phase est un système de coordonnées à deux dimensions qui sont l'altitude y et la vitesse y'. Dans le cas où l'espace des phases est un plan, on peut obtenir une idée de la dynamique en dessinant simplement le champ de vecteurs dans un rectangle $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$ avec l'instruction graphique *fchamp* dont la syntaxe est :

```
fchamp(MonsecondMembre,t,x,y)
```

où MonsecondMembre est le nom de la fonction Scilab du second membre de l'EDO, T est l'instant pour lequel on veut dessiner le champ, et x, y sont des vecteurs lignes à nx et ny composantes, donnant les points de la grille sur lequels seront dessinés les flèches représentant le champ de vecteurs

a) Le champ de vecteurs issu de l'équation de Van Der Pol (Figure 5)

```
 n=30;
delta=5;
x=linspace(-delta,delta,n);
xbasc()
fchamp(VanDerPol,0,x,x)
```

b) L'instruction *portrait* (reprenant uniquement la fonction précédente MonSecondMembre en argument) permet pour sa part de tracer le portrait de phase d'un système dynamique plan.

A l'aide de l'instruction *ode*, résoudre sur $[0, 25]$ le système d'équations dit de Lokta-Volterra intervenant en dynamique des populations ($x(t)$ désigne la densité de prédateurs, $y(t)$ celle de proies) :

$$\begin{cases} x'(t) &= xy/2 - x/2 \\ y'(t) &= -xy + y \end{cases}$$

avec $x(0) = y(0) = 2$.

```
//Résolution du système (Figure 7):
function[f]=LoktaVoltera(t,u)
f(1)=1./2*u(1)*u(2)-u(1)./2
f(2)=-u(1)*u(2)+u(2)
endfunction
m=150;
u0=[2;2];
t=linspace(0,25,m);
[u]=ode(u0,0,t,LoktaVoltera);
subplot(1,3,1)
n=100;
deltax=5;
deltay=5;
x=linspace(-deltax,deltax,n);
y=linspace(-deltay,deltay,n);
fchamp(LoktaVoltera,0,x,y,strf="000")
xselect()
plot2d(u(1,:)',u(2,:)',5,"121")
xtitle('Trajectoire des phases:interaction des populations', 'y','x')
subplot(1,3,2)
plot2d(t,u(1,:)',4)
xtitle('Evolution des predateurs', 't','x')
subplot(1,3,3)
plot2d(t,u(2,:)',2)
xtitle('Evolution des proies','t','y')
```

La commande *portrait* utilise comme arguments d'entrée MonSecondMembre du système différentiel ainsi que quelques options graphiques (voir l'aide en ligne). L'instruction portrait(LoktaVoltera) renvoie des fenètres de dialogues permettant de tarcer le champ de vecteurs, et de rentrer une nouvelle donnée initiale à l'aide de la sourie. Une trajectoire dans la plan de phase, partant de cette donnée initiale, est alors dessinée. On peut renouveler l'opération plusieures fois.

Le script ci-dessous permet d'avoir plusieures trajectoires dans le plan de phase correspondant à des données initiales rentrées avec l'instruction *xclick* de scilab (voir Figure 9).

```
n=30;
 deltax=6;
 deltay=4;
 x=linspace(0,deltax,n);
 y=linspace(0,deltay,n);
xbasc()
fchamp(LoktaVoltera,0,x,y,strf="041")
xselect()
xtitle(' Champs de vecteurs et quleques trajectoires   du systeme de Lokta Voltera')
m=500; T=30;
t=linspace(0,T,m) les instants pour lesquels on recupere la solution
couleurs=[21 2  3 4  5  6 19 28 32 9 13 22 18 21 12 30 27]
num=-1;
while  %t
 [c_i,c_x,c_y]=xclick();
if  c_i==0  then
plot2d(c_x,c_y,style=-9,strf="000")   //un petit o pour marquer la CI
u0=[c_x;c_y]  // donnee initiale
[u]=ode(u0,0,t,LoktaVoltera);
num=modulo(num+1,length(couleurs));
plot2d(u(1,:)',u(2,:)',style=couleurs(num+1),strf="000")
elseif  c_i==2 then
break
end
```

```
end
```

Remarques. — 1) On observe le caractère périodique des solutions (Figure 7). Ces solutions ont la même période puisque le cycle dans la plan de phase est fermé. Nous constatons aussi que si le nombre de proies augmente, alors le nombre des prédateurs va augmenter quelques temps plus tard, et de même si la population des proies diminue (Figure 8).
2) Les deux populations sont décalées d'un quart de cycle ; la population dévorée commence donc à décliner lorsque la population des prédateurs entreprend sa phase de croissance rapide.
3) De la Figure 9, on constate que le point $(1, 1)$ est un point d'équilibre puisque $f(1, 1) = (0, 0)$. Ce point n'est, par ailleurs, pas attracteur. On note aussi que ce point réalise la moyenne de toutes les courbes. En effet, en observant que

$$\ln(x)' = y/2 - 1 \quad \text{et} \quad \ln(y)' = -x + 1$$

et en intégrant sur une période, on vérifie que

$$\frac{1}{T}\int_0^T y(t)\,dt = \frac{1}{T}\int_0^T x(t)\,dt = 1$$

5. Test d'ajustement χ^2

Soit $x^m = (x_1, \ldots, x_m)$ un échantillon. Et soit l'hypothèse $\mathcal{H}$: "les variables aléatoires sous-jacentes $(X_1, \ldots, X_m)$ suivent la loi $\mathcal{L}$". Une question est de savoir si cette hypothèse est réaliste. Sur cet échantillon, on peut calculer les statistiques élémentaires (moyenne et écart type empirique) et si celles-ci semble raisonnablement proches de l'espérance et de l'écart type de $\mathcal{L}$, on peut alors mettre en œuvre un test statistique. Le test χ^2 s'applique sur une loi discrète prenant un nombre fini de valeurs. Par exemple supposons que la loi de $\mathcal{L}$ est donnée par $(v_i, p_i),\ 1 \leq i \leq n\}$. Le test consiste à calculer la quantité :

$$y = \sum_{i=1}^{n} \frac{(o_i - mp_i)^2}{mp_i}$$

où o_i est le nombre de résultats x_i égaux à v_i et à comparer la valeur obtenue y à un seuil y_{seuil}, le test étant alors positif si $y \leq y_{seuil}$.
Exemple. — On dispose d'un dé numéroté de 1 à 6 et on veut savoir s'il est parfaitement équilibré, c'est-à-dire si la probabilité d'obtenir chaque numéro est effectivement $1/6$: ici X' la variable aléatoire correspondant au dé possédé (on ne connait pas les probabilités théoriques d'apparition de chaque numéro) et X est la variable définie par $P(X = i) = 1/6$ pour $i = 1, \ldots, 6$. On cherche à savoir si X' a pour loi celle de la variable aléatoire X.

6.

On a effectué 200 fois l'expérience suivante avec le même dé : on le jette autant de fois qu'il le faut jusqu'à obtenir un 1 (mais on arrête lorsque le 1 n'est pas sorti au bout de 10 lancés). On a obtenu les résultats suivants : par exemple il y a 36 expériences où le 1 est sortie lors

nombre de jets	1	2	3	4	5	6	7	8	9	10	≥ 11
nombre d'expériences	36	25	26	27	12	12	8	7	8	9	30

TAB. 1 – Résultats

du premier lancé, 25 où le 1 est sorti au deuxième lancé, etc...
Effecttuer un test χ^2 pour essayer de répondre à la question.
Souvent, une simulation consiste en premier lieu, à obtenir un vecteur :

$$x^m = (x_1, \ldots, x_m)$$

dont les composantes sont considérées comme les réalisations de variables aléatoires indépendantes et de même loi X_1, $X_2, \ldots, X_m$ (on notera X une variable aléatoire indépendante suivant la même loi.) A partir de l'échantillon x^m on cherche à approcher

les caractéristiques de la loi sous-jacente comme l'espérance, l'écart-type, la fonction de répartition (ou la densité), ou si on emet une hypothèse sur la loi en question, à déterminer ses paramètres, ou encore si les paramètres sont connus, à vérifier via un ou des tests statistiques que notre échatillon est (avec un risque !) bien issu de variable aléatoire qui suivent effectivement la loi en question, etc...

Intervalle de confiance. — Une fois l'espérance empirique obtenue à partir de notre échantillon, on aimerait connaitre un intervalle, dit de confiance, I_c centré en $\bar{x}^m$, la moyenne, (obtenue par la fonction scilab mean (xm)) pour affirmer que :

$$E[X] \in I_c \quad \text{avec une probabilité} \quad 1-\alpha$$

où $\alpha = 0.05$ ou 0.01 (intervalles de confiance à respectivement 95% et 99%).

Si on utilise, dans formule donnant y, les variables aléatoires $X_1, \ldots, X_m$ au lieu de l'échantillon $x_1, \ldots, x_m$ on définit alors une variable aléatoire Y qui suit approximativement (pour m suffisamment grand) la loi du χ^2 à $n-1$ degré de liberté. La valeur seuil est alors obtenue par scilab :

```
y_seuil=cdfchi(''X'',n-1,1-alpha,alpha)//
```

Si l'on note J la variable aléatoire correspondant au résultat de l'expérience alors J suit la loi géométrique $G(p)$ avec $p = 1/6$ si le dé est non truqué. Avec les données de l'expérience, on assemble dans une même classe tous les résultats correspondants à un nombre de lancés supérieur strictement à 10. Ce qui nous donne les probabilités :

$$P(J=1) = p,\ P(J=1) = p,\ P(J=2) = qp,\ P(J=3) = q^2p, \ldots,\ P(J=10) = q^9p,$$

et $P(J > 11) = q^{10}$ où $q = 1-p$. Du tableau, donné plus haut, nous avons les occurences o_i, on écrit alors les script suivant pour calculer y :

```
occ=[36 ; 25 ; 26; 27; 12; 12; 8; 7; 8; 9; 30];// les oi
p=1/6; q=5/6;
pr=[p*q.^(0:9),q^10];// les probabilités  pi
y=sum( (occ-m*pr).^2./(m*pr));
y_seuil=cdfchi(''X'',10,0.95,0.05)
```

Ce dé semble correct car y est plus petit que y_{seuil}.

1 Scripts

1.1 Script 1

```
//Resolution de cos(x)=x  par differentes methodes
deff('y=f(x)','y=cos(x)-x');
deff('y=fprime(x)','y=-sin(x)-1');
xbasc()
xset('window',0);
x=0:0.01:3;
subplot(2,3,1)// pour diviser la fenetre graphique
plot2d(x,f(x),2,axesflag=5);
xtitle('La courbe f(x)=cos(x)-x')
xgrid(6)//La grille permet de localiser la solution avec la commande
//Zoom du menu graphique
// methode de dichotomie (bissection)
a=0;b=3;c=(a+b)/2;nb=0;
vb=[];// Initialisation  de vb au vecteur vide [], puis
//en concatenant dans une boucle chaque nouvel  element avec
//des affectation de type vb=[vb,x0]
while (f(c)~=0)&(nb < 50)
  c=(a+b)/2;
```

```
  if (f(c)*f(a)<0) then
    b=c;
  else
    a=c;
  end
  nb=nb+1;
  vb=[vb;c];
end
subplot(2,3,2)
plot2d((0:1:nb-2)',vb(1:(nb-1)),7)
xtitle('La Dichotomie','n','cn')
// methode de  la corde  (fausse position)
a=0;b=3;nfp=0;
y1=f(a);y2=f(b);c=b-y2*(b-a)/(y2-y1);
vfp=[];
while (f(c)~=0)&(y2~=y1)&(nfp < 100)
  c=b-y2*(b-a)/(y2-y1);
  if (f(c)*f(a)<0) then
    b=c;y2=f(c);
  else
    a=c;y1=f(c);
  end
  nfp=nfp+1;
  vfp=[vfp;c];
end
subplot(2,3,3)
plot2d((0:nfp-2)',vfp(1:(nfp-1)),3);
xtitle('La fausse position','n', 'cn')
// methode de Newton
x0=1;vn=[];nn=1;
while (f(x0)~=0)&(fprime(x0)~=0)&(nn<30)
  x0=x0-f(x0)/fprime(x0);
  nn=nn+1;
  vn=[vn;x0];
end
subplot(2,3,4)
plot2d((0:nn-2)',err(1:(nn-1)),5);
xtitle('Newton','n', 'xn')
//
// methode de point fixe
//
deff('y=g(x)','y=cos(x)')
x0=1;vpf=[];npf=1;
while (g(x0)~=x0)&(npf<40)
  x0=g(x0);
  npf=npf+1;
  vpf=[vpf;x0];
end
subplot(2,3,5)
plot2d((0:npf-2)',vpf(1:(npf-1)),9);
xtitle('Point fixe','n','xn')
```

1.2 Script 2

```
 function [f]=SecondMembre(t,y)
f=3*y-1;
```

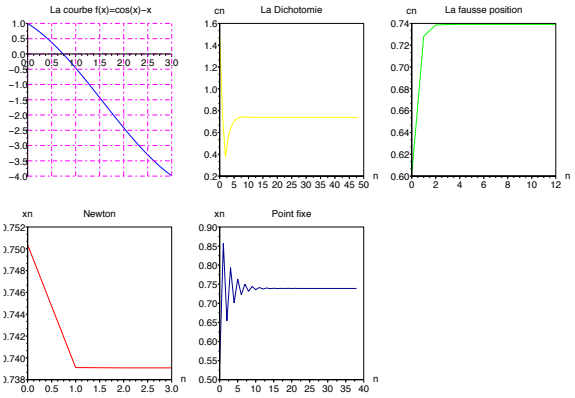

FIG. 1 – Convergences de différentes méthodes

```
endfunction
function [g]=SecondMembre2(t,y)
g=-150*y+30;
endfunction
function[z]=Euler(SecondMembre,a,b,n)
h=(b-a)./n;
t=0:(b-a);
x=evstr(x_dialog(['La donnee initiale';'Entrer une nouvelle donnee'],'1./3'));
v=[];
for i=1:n
x=x+h*SecondMembre2(t,x)
v=[v,x];
z=v;
end;
xset('window',1)
//subplot(2,2,1)
plot2d((0:n-2)',z((1:n-1)),5)
xtitle('Approximation par Euler explicite')
endfunction
```

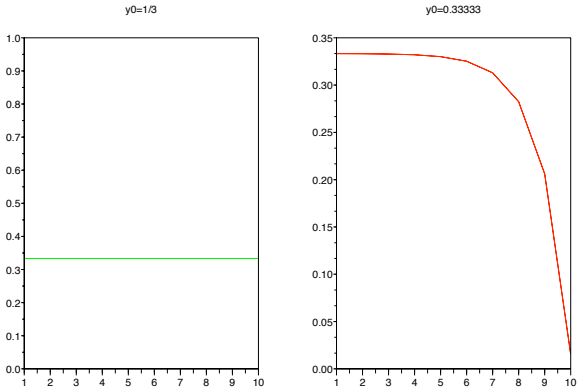

FIG. 2 – Problème numériquement mal posé

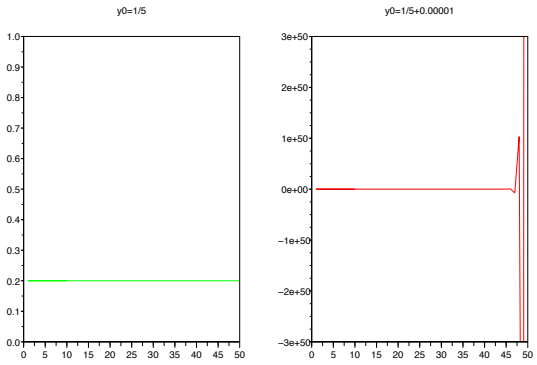

FIG. 3 – Problème raide ou mal conditionné

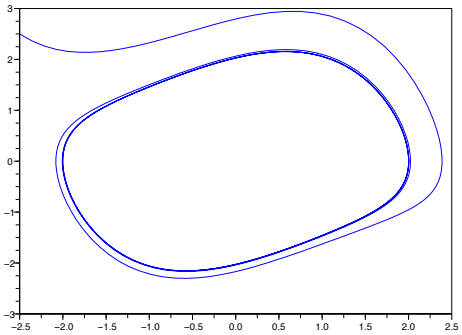

FIG. 4 – La courbe solution de Van Der Pol

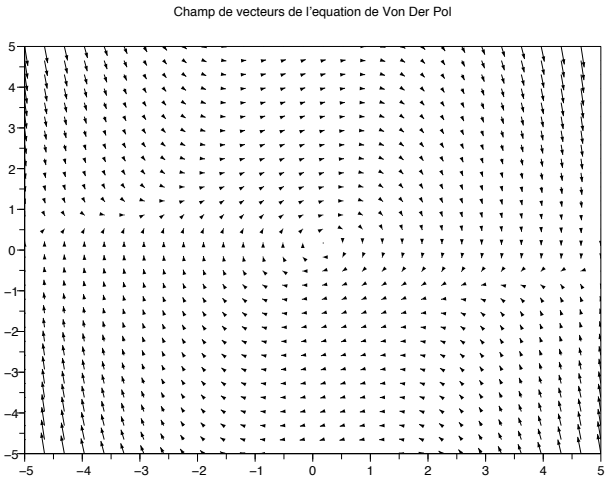

FIG. 5 – Champ de vecteurs de l'équation de Van Der Pol

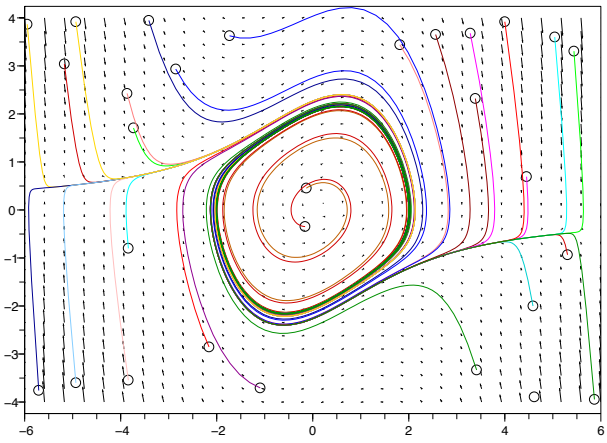

FIG. 6 – Quelques trajectoires dans le plan de phase pour l'équation de Van Der Pol

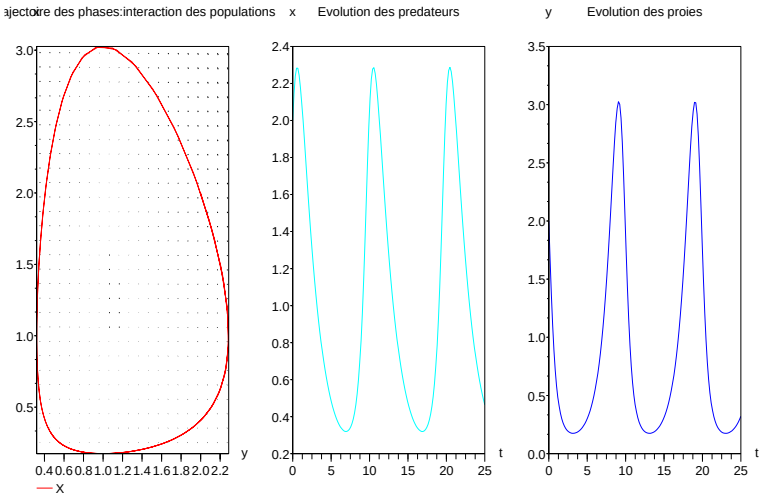

FIG. 7 – Périodicité et interaction des populations du système de Lokta Voltera

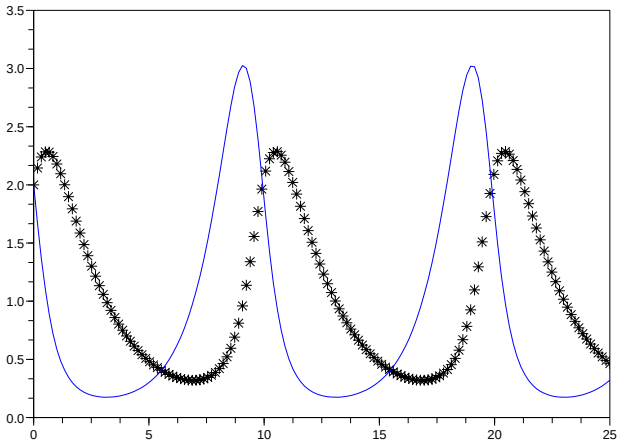

FIG. 8 – Comparaison des deux popoulations

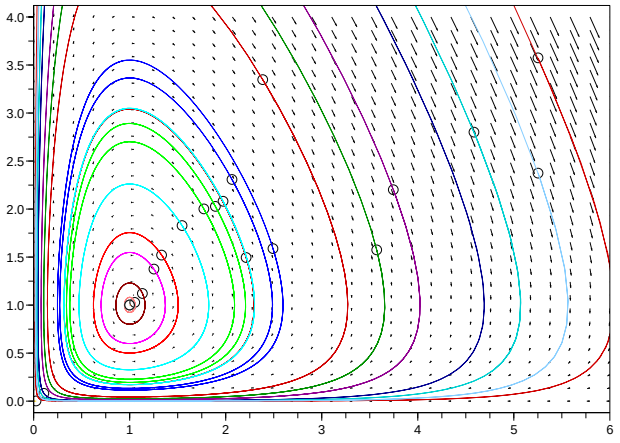

FIG. 9 – Quelques trajectoires dans le plan de phase pour le système de Lokta Voltera