Étude numérique du système de Lotka-Volterra

$$\begin{cases} s' &= \quad a\,s - b\,s\,r \\ r' &= -c\,r + d\,s\,r \end{cases}$$

1 La méthode d'Euler

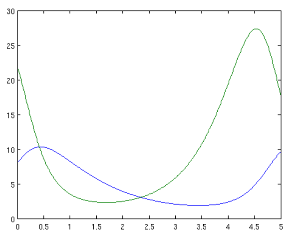
La fonction LVEuler ci-dessous (`LVEuler.m`) permet de calculer les valeurs approchées de la solution du système de Lotka-Volterra valant (s_0, r_0) en $t = 0$, en utilisant la méthode d'Euler avec un pas de $0,01$.

```
% fonction calculant la solution approchee du systeme de Lotka-Volterra
% s' = as-bsr
% r' = -cr +dsr
% valant r0 s0 en 0
% par le methode d'Euler de pas h
% sur un intervalle [0,T]

function M=LVEuler(s0,r0,a,b,c,d,T,h)

% le maillage de l'intervalle de temps
t=0:h:T;
% le nombre de noeuds du maillage.
Z= size(t);
z=Z(2);

% calcul des valeurs approchees par la methode d'Euler
r(1)=r0;
s(1)=s0
for i=2:z
  s(i)=s(i-1)+h*(a*s(i-1)-b*s(i-1)*r(i-1));
  r(i)=r(i-1)+h*(-c*r(i-1)+d*s(i-1)*r(i-1));
end

% On met les resultats dans une unique matrice M qui sortira les valeurs obtenues
M(:,1)=t;
M(:,2)=s;
M(:,3)=r;
```

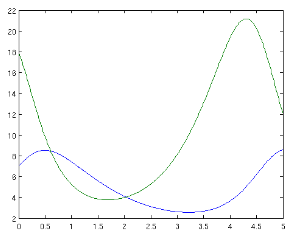
Le script suivant (`LVEscript.m`) trace quelques solutions approchées données par cette méthode pour un pas de $0,01$ puis les courbes paramétrées par ces solutions (*i.e.* les trajectoires).

```
%on nettoie le graphique
clf

%les parametres
a=2;
b=0.4;
c=1;
d=0.1;

%La duree
T=5
% Le pas
h=0.01

% les differentes conditions initiales en t0=0
CI0=[22 , 8 ; 18 , 7 ; 14, 6 ; 10, 5];
Z0=size(CI0);
z0=Z0(1);

%le calcul et le dessin des solutions issues de t0=0 et x0 dans CI0
for i=1:z0
  M=LVEuler(CI0(i,1),CI0(i,2),a,b,c,d,T,h);
  r=M(:,3);
  s=M(:,2);
  t=M(:,1);
  plot(t,r,t,s)
  pause
%on fait une pause entre chaque trace
%appuyer sur une touche pour passer a la boucle suivante
end
% nettoyage
clf

% tracer des courbes parametrees (sardines(t), requins(t))
for i=1:z0
  M=LVEuler(CI0(i,1),CI0(i,2),a,b,c,d,T,h);
  r=M(:,3);
  s=M(:,2);
  plot(s,r)
  hold on
  pause
%on fait une pause entre chaque trace
%appuyer sur une touche pour passer a la boucle suivante
end
```

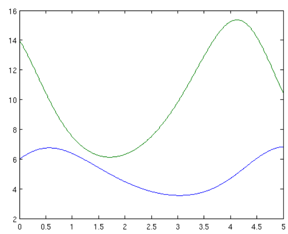
On obtient



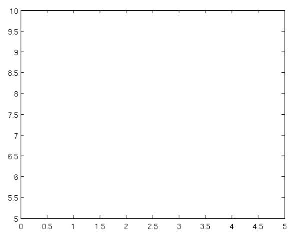
Solution issue de [22,8]



Solution issue de [18,7]

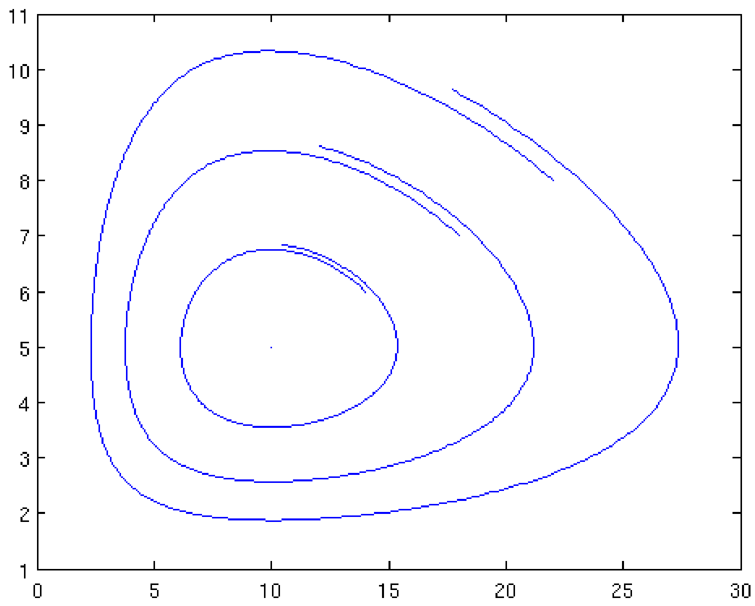


Solution issue de [14-6]



Solution issue de [10-5]

Puis le portrait de phase approché



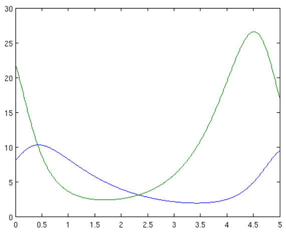
2 La Méthode de Runge-Kutta

La fonction LVRK ci-dessous (`LVRK.m`) permet de calculer les valeurs approchées de la solution du système de Lotka-Volterra valant (s_0, r_0) en $t = 0$, en utilisant la méthode de Runge-Kutta avec un pas de h.

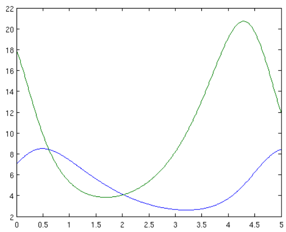
```
% fonction calculant la solution approchee du systeme de Lotka-Volterra
% s' = as-bsr
% r' = -cr +dsr
% par le methode de RungeKutta de pas h
% valant r0 s0 en 0
% sur un intervalle [0,T]

function M=LVRK(s0,r0,a,b,c,d,T,h)

%le maillage de l'intervalle de temps
t=0:h:T;
%le nombre de noeuds du maillage.
Z= size(t);
z=Z(2);

% calcul des valeurs approchees par runge kutta
r(1)=r0;
s(1)=s0
for i=2:z
  k11=a*s(i-1)-b*s(i-1)*r(i-1);
  k12=-c*r(i-1)+d*s(i-1)*r(i-1);

  k21=a*(s(i-1)+h*k11/2)-b*(s(i-1)+h*k11/2)*(r(i-1)+h*k12/2);
  k22=-c*(r(i-1)+h*k12/2)+d*(s(i-1)+h*k11/2)*(r(i-1)+h*k12/2);

  k31=a*(s(i-1)+h*k21/2)-b*(s(i-1)+h*k21/2)*(r(i-1)+h*k22/2);
  k32=-c*(r(i-1)+h*k22/2)+d*(s(i-1)+h*k21/2)*(r(i-1)+h*k22/2);
```

```
  k41=a*(s(i-1)+h*k31)-b*(s(i-1)+h*k31)*(r(i-1)+h*k32);
  k42=-c*(r(i-1)+h*k32)+d*(s(i-1)+h*k31)*(r(i-1)+h*k32);

  s(i)=s(i-1)+h*(k11+2*k21+2*k31+k41)/6;
  r(i)=r(i-1)+h*(k12+2*k22+2*k32+k42)/6;
end

%On met les resultats dans une unique matrice M qui sortira les valeurs obtenues
M(:,1)=t;
M(:,2)=s;
M(:,3)=r;
```

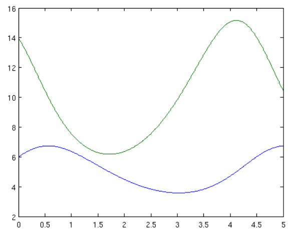
En reprenant le script précédent en faisant appel à `LVRK.m` à la place de `LVEuler.m`, on obtient :



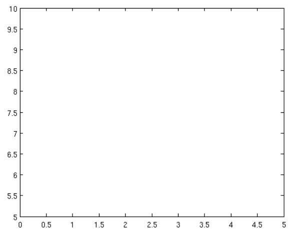
Solution issue de [22,8]



Solution issue de [18,7]

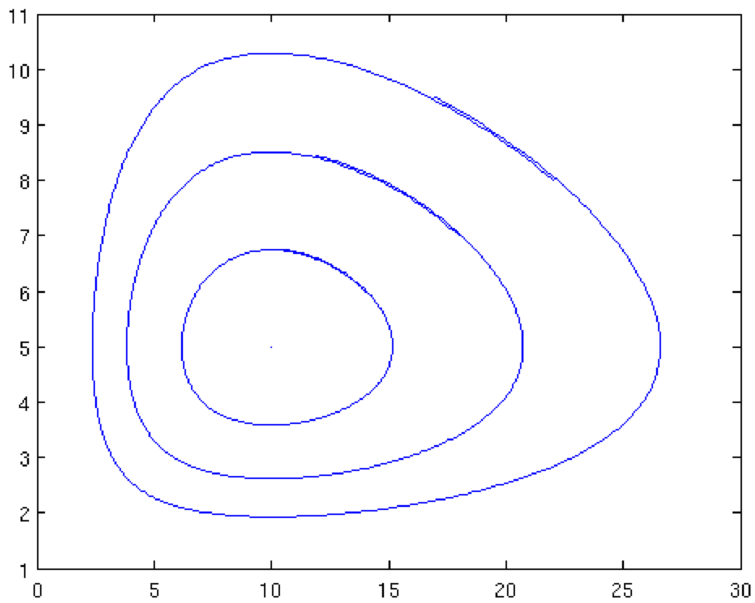


Solution issue de [14-6]



Solution issue de [10-5]

Puis le portrait de phase approché



3 Solution exacte

On trouve facilement une intégrale première du système sous la forme

$$L(r,s) = \alpha r + \beta s + \gamma \ln r + \delta \ln s.$$

En demandant que $L(s(t), r(t))$ soit constante lorsque r et s sont une solution du système de Lotka-Volterra, on trouve

$$\alpha = b, \beta = d, \gamma = -a \text{ et } \delta = -c.$$

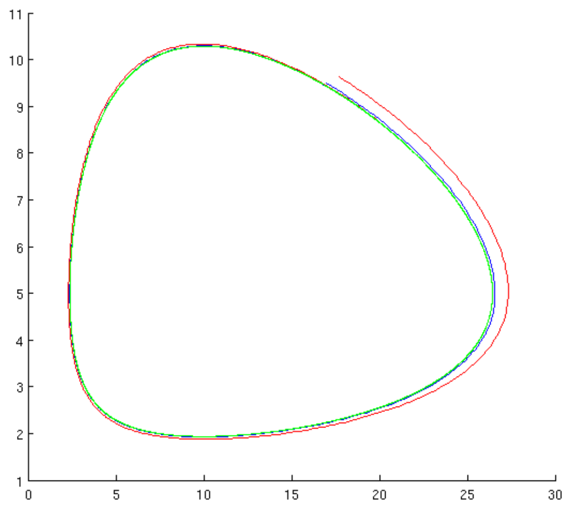
Pour chaque solution passant pas (s_0, r_0) on a donc $L(s(t), r(t)) = L(s_0, r_0)$. En particulier la solution issue de $(22, 8)$ est la courbe d'équation

$$br + ds - a \ln r - c \ln s = b8 + d22 - a \ln 8 - c \ln 22.$$

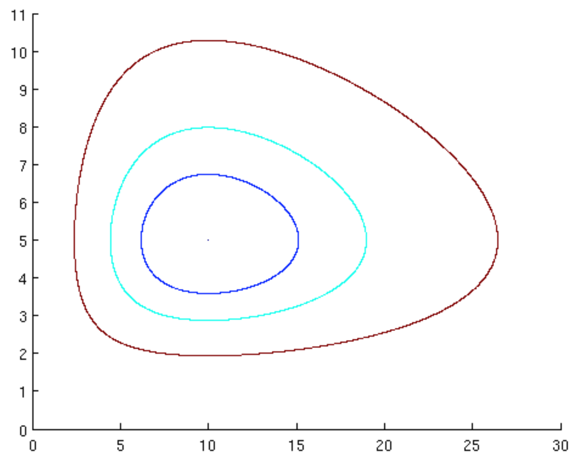
La fonction `compar.m` ci-dessous (écrite par Rémy Ramadour) trace les 2 solutions numériques obtenues par les méthode d'Euler et de Runge-Kutta ainsi que la solution analytique issues de la condition initiale (s_0, r_0).

```
TP03 de C02 de Rémy Ramadour -- Automne 2007

function compar(s0,r0)

% compare les valeurs exactes avec les deux approximations :  Euler et Runge-Kutta

clf
h = 0.01 ;
a = 2 ;
b = 0.4 ;
c = 1 ;
d = 0.1 ;
T=5;
[X,Y] = meshgrid(0.1:h:10, 0.1:h:20) ;
Z = b*Y - a*log(Y) + d*X - c*log(X) ;
v = 0.4*r0 - 2*log(r0) + 0.1*s0 - log(s0) ;

A = LVEuler(s0,r0,a,b,c,d,T,h) ;
B = LVRK(s0,r0,a,b,c,d,T,h) ;

hold on

plot(A(:,2),A(:,3),'r');
pause
plot(B(:,2),B(:,3), 'y') ;
pause
[C,h] = contour(X,Y,Z,v,'g') ;
```

Ce qui donne



solutions calculées par `compar.m`



Les quatres solutions "exactes" tracées avec `contour`