Projet Bis

Cottereau Laure et Dalissier Eric

1 Le modèle de croissance logistique

1.1 Question 1

Nous avons l'équation différentielle proposée par Verhulst en 1936 :

$$N'(t) = rN(t)(1 - \frac{N(t)}{K}) \tag{1}$$

Donc on a : $N'(t) = rN(t) - r\frac{N^2(t)}{K}$
Une équation de Bernoulli est du type :
$y' = p(t)y + q(t)y^\alpha$
L'équation (1) est bien une équation de Bernoulli avec $p(t) = r$, $q(t) = -\frac{r}{K}$ et $\alpha = 2$
On pose :
$z(t) = N(t)^{1-\alpha} = \frac{1}{N(t)}$
L'équation (1) devient alors :

$$-\frac{z'(t)}{z^2(t)} = r\frac{1}{z(t)}(1 - \frac{1}{Kz(t)})$$

On a alors l'équation différentielle suivante, que l'on résoudra de manière standard :

$$-z'(t) = rz(t) - \frac{r}{K} \tag{2}$$

Résolution de l'équation homogène :

$$\frac{z'(t)}{z(t)} = -r$$

$ln|z(t)| = -rt + C1$ avec $C1 \in \mathbb{R}$

$z(t) = e^{-rt}e^{C1}$

Solution particulière :

Pour calculer la solution particulière, nous allons utliser la méthode de variation de la constante. Quand on pose $C(t)$=e^{C_1},cela nous donne :

$z'(t) = -re^{-rt}C(t) + C'(t)e^{-rt}$

On remplace dans l'équation (2)

$-re^{-rt}C(t) + C'(t)e^{-rt} = -re^{-rt}C(t) + \frac{r}{K}$

$\Rightarrow C'(t)e^{-rt} = \frac{r}{K}$

D'où $C(t) = \frac{1}{K}e^{rt}$

La solution de notre équation est donc :

$$z(t) = e^{-rt}C + \frac{1}{K}e^{-rt}e^{rt} \tag{3}$$

Si maintenant on remplace $z(t) = \frac{1}{N(t)}$, on a après simplification :

$N(t) = \frac{K}{KCe^{-rt} + 1}$

Maintenant, calculons C grâce à la condition initiale $N(0) = N_0$:

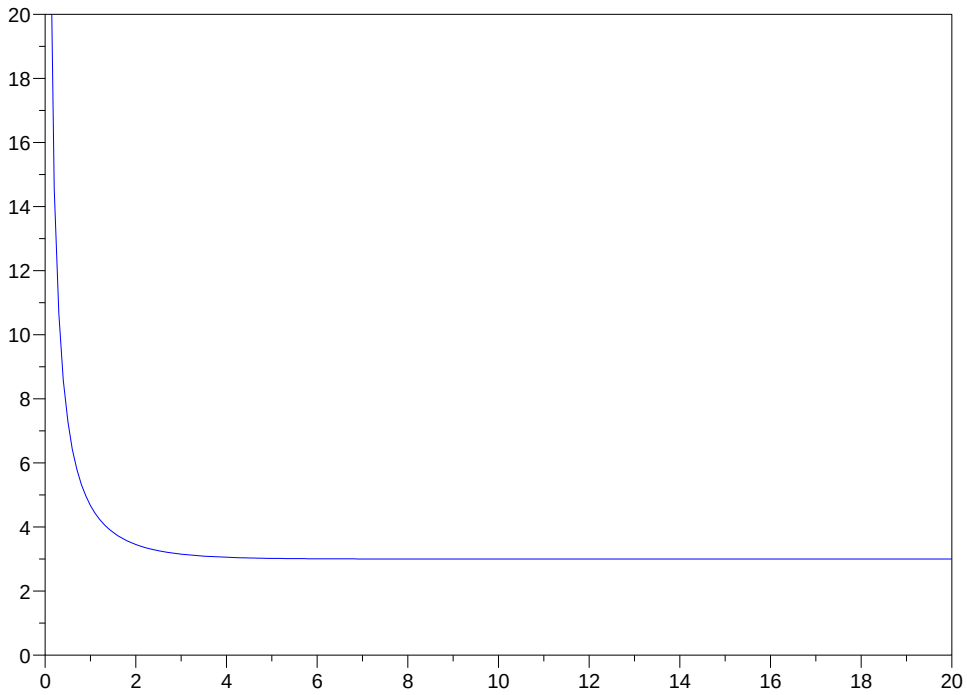
D'où $C = \frac{K - N_0}{N_0 K}$

Finalement, on obtient la solution de l'équation (1) qui est :

$$N(t) = \frac{K}{\frac{K-N_0}{N_0}e^{-rt} + 1} \tag{4}$$

Quand on fait tendre t vers $+\infty$, on trouve :

$\lim_{t \to +\infty} N(t) = K$



$K = 3$ et $r = 1$

1.2 Question 2

Voir l'annexe. On remarque que les solutions approchés à l'aide de la méthode d'Euler et de la méthode de Runge-Kutta à d'ordre 3 pour les paramètres $r = 1$ et $K = 3$, approche la solution exacte.
Pour évaluer l'erreur nous avons programmé les fonctions "*compare*1" pour Euler et "*compare*2" pour Runge-Kutta (annexe). On remarque que la méthode de Runge-Kutta a une erreur plus faible que la méthode d'Euler.
Plus le pas de temps est petit, plus l'erreur est petite.
Pour mesurer la vitesse de convergence, on évalue l'erreur. Pour un même pas de temps, la méthode de Runge-Kutta converge plus vite que celle d'Euler.

2 Le modèle du ver du bourgeon de l'épinette

On part de l'équation proposé par Ludwig-Jones et Holling en 1978. La population du ver suit cette équation :

$$\frac{dN(t)}{dt} = rN(t)(1 - \frac{N(t)}{K}) - \frac{BN^2}{A^2 + N^2} \tag{5}$$

2.1 Question 1

Pour justifier le passage de l'équation 5 à l'équation que nous utiliserons par la suite, nous allons faire des changements de variables. Les variables seront r, K, A, B qui sont des constantes, et u une fonction de τ.
Tout d'abord le changement $uA = N$ d'où $Adu = dN$. Cela implique :

$$A\frac{du}{dt} = rAu(1 - \frac{Au}{K}) - \frac{A^2Bu^2}{A^2 + A^2u^2}$$

Faisons maintenant le changement de variable $\tau = \frac{B}{A}t$, et donc $d\tau = \frac{B}{A}dt$, on trouve :

$$B\frac{du}{d\tau} = rAu(1 - \frac{Au}{K}) - \frac{A^2Bu^2}{A^2(1 + u^2)}$$

Après avoir divisé par B et simplifié le dernier terme de l'équation par A^2, on fait le changement $\rho = \frac{A}{B}r$:

$$\frac{du}{d\tau} = \rho u(1 - \frac{Au}{K}) - \frac{u^2}{(1 + u^2)}$$

Il ne nous reste plus maintenant qu'à poser $q = \frac{K}{A}$

$$\frac{du}{d\tau} = \rho u(1 - \frac{u}{q}) - \frac{u^2}{(1 + u^2)} \tag{6}$$

Afin de rechercher les solutions constantes de (6), on est amené à résoudre l'équation non linéaire suivante :

$$ru(1 - \frac{u}{q}) - \frac{u^2}{1 + u^2} = 0 \tag{7}$$

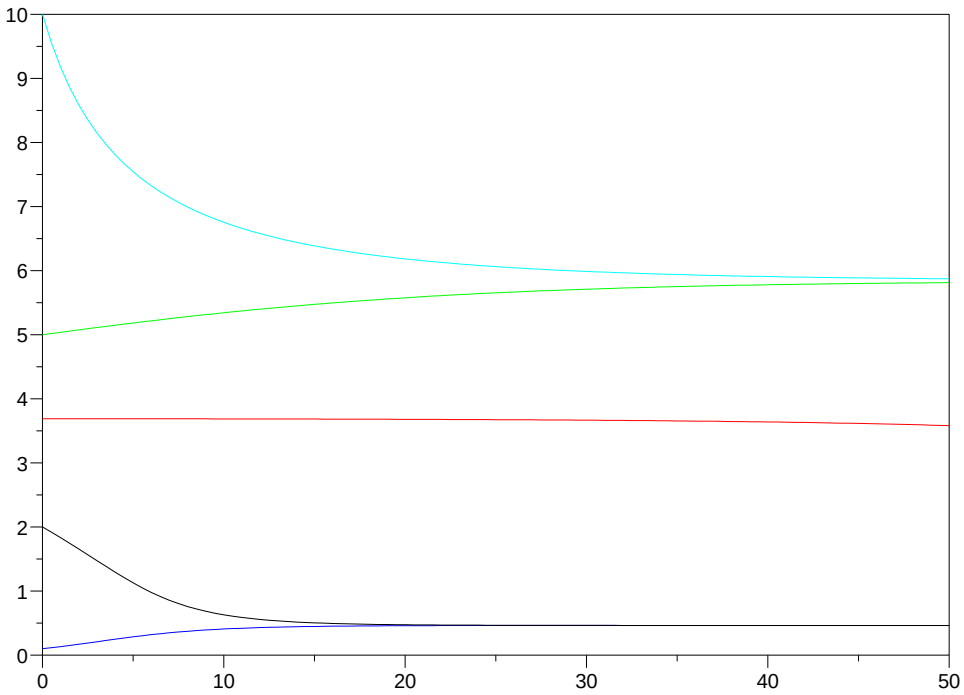
2.2 Question 2

On a utilisé la fonction *ode* de Scilab afin de représenter de manière approchée la solution de l'équation (6) pour les paramètres $r = 0.4$ et $q = 10$ et les valeurs initiales respectives $u_0 = \{0,1; 2; 5; 10\}$. On remarque que $u(t)$ ne converge pas vers les mêmes valeurs quand on change les conditions initiales :

- pour $u_0 = 0.1$ et $u_0 = 2$, $u(t)$ converge vers 0.4633717
- pour $u_0 = 5$ et $u_0 = 10$, $u(t)$ converge vers 5.8739895

En prenant de nouvelles valeurs de u_0, on remarque :

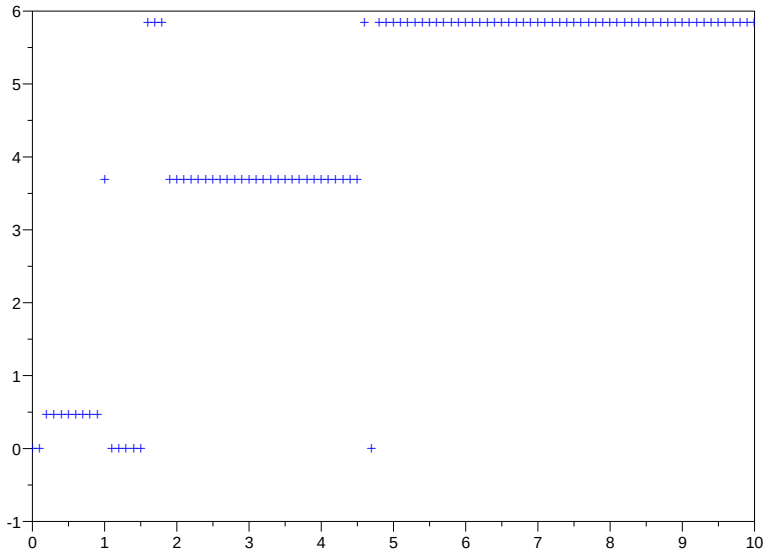
- $u_O \in]0; 3, 69]$, on a la convergence vers 0.4633717
- $u_O \approx 3, 6932251$, on a un point de transition
- $u_O \in [3, 7; +\infty[$, on a la convergence vers 5.8739895



2.3 Question 3

Nous avons reprogrammé la méthode de $Newton$ pour notre problème (voir annexe). En faisant varier u_0, on remarque que le point fixe change. Nous avons donc créé la fonction $trac$ (voir annexe) qui appliqé à la fonction $Newton$, nous permet de voir les différents points fixes en fonction de u_0. Nous obtenons le tableau et le graphique suivant :

u_0	$Points fixes$
0	0
$0,1$	$-1,43110^8$
0.2	0.4633704
0.9	0.4633704
1.0	3.6932251
1.1	$-1.042E-10$
1.5	$-1.149E-09$
1.6	5.8434045
1.8	5.8434045
1.9	3.6932251
4.5	3.6932251
4.6	5.8434045
4.7	$-6.137E-10$
4.8	5.8434045
10	5.8434045



Nous avons choisi de ne pas noter les valeurs à des intervalles dans lesquels le point fixe

était inchangé.

2.4 Question 4

La méthode de *Newton* nous permet de calculer les points fixes de l'équation (7), elle nous donne donc les solutions constantes vérifiant l'équation (6), c'est à dire les points d'équilibres. Sur le graphique, nous pouvons voir qu'il y a 4 points d'équilibres :
$U = \{\ 0\ ;\ 0.4633704\ ;\ 3.6932251\ ;\ 5.8434045\ \}$

Etudions leurs stabilités :

- 0 est un point instable,car les valeurs proches de 0 ne tendent pas vers 0. Exemple : pour $U = 0.2$, on a le point fixe qui est 0.4633704
- 0.4633704 est un point stable. En effet, les valeurs des points fixes pour $u_0 = [0.2, 0.9]$ sont égales à 0.4633704
- 3.6932251 est un point stable. En effet, les valeurs des points fixes pour $u_0 = [1.9, 4.5]$ sont égales à 3.6932251
- 5.8434045 est un point stable. En effet, les valeurs des points fixes pour $u_0 = [4.8; +\infty[$ sont égales à 5.8434045

2.5 Question 5

Pour représenter un régionnement de l'espace des paramètres (q, r) en fonction du nombre de points d'équilibre de l'équation (6), nous allons résoudre l'équation (7).
On a :

$$ru(1 - \frac{u}{q}) - \frac{u^2}{1 + u^2} = 0$$

$$ru(1 - \frac{u}{q})(1 + u^2) - u^2 = 0$$

On développe : $-\frac{r}{q}u^4 + ru^3 - (1 + \frac{r}{q})u^2 + ru = 0$

On met u en facteur, on a ainsi le 1^{er} point d'équilibre $u = 0$. Par la suite ce point d'équilibre ne sera pas pris en compte dans le nombre de points d'équilibre.

On obtient : $-\frac{r}{q}u^3 + ru^2 - (1 + \frac{r}{q})u + r = 0$

On multiplie par q et -1, on divise par r :

$$u^3 - qu^2 + (1 + \frac{q}{r})u - q = 0$$

Pour savoir comment varie la fonction, nous allons étudier la dérivée.
En dérivant, on obtient :

$$3u^2 - 2qu + 1 + \frac{q}{r} = 0$$

On calcule le discriminant :

$$\Delta = 4[q^2 - 3(1 + \frac{q}{r})]$$

Quand Δ est positif, on a 3 points d'équilibre(car les solutions sont réelles).
Quand Δ est négatif ou vaut 0, on a qu 1 seul point d'équilibre. Car dans ce cas, la fonction est monotone.
Nous avons fait un programme qui nous rend en fonction de (q,r) le nombre de points d'équilibre de l'équation (7).(Voir annexe)
Le dernier programme permet de voir un régionnement de l'espace des paramètres (q, r) en fonction du nombre de points d'équilibre de l'équation (6). (voir Annexe)
Nous pouvons remarquer que quand r et q sont grands, l'équation (7) a 3 points d'équilibre.

Annexes

Partie 1

Voici les programmes utilisés :

Euler :

```
function u = eulerexplicite(f,u0,t,n)
lt=length(t)
u(1)=u0
  for i=2 :lt
  t1=t(i)
  u0=u(i-1)'
  t0=t(i-1)
  h=(t1-t0)/n
    for j=1 :n
    u0=u0+h*f(t0,u0)
    t0=t0+h
    end
  u(i)=u0'
  end
endfunction
```

Runge-Kutta d'ordre 3 :

```
function v = rungekutta(f,v0,t,n)
lt=length(t)
v(1)=v0
  for i=2 :lt
  t1=t(i)
  v0=v(i-1)'
  t0=t(i-1)
  h=(t1-t0)/n
    for j=1 :n
    v1=v0
    v2=v0+h/2*f(t0+0.5*h,v1)
    v3=v0+h/2*f(t0+0.5*h,v2)
    v4=v0+h*f(t0+h,v3)
    v0=v0+h/6*(f(t0,v1)+2*[f(t0+0.5*h,v2)+f(t0+0.5*h,v3)]+f(t0+h,v4))
    t0=t0+h
    end
  v(i)=v0'
  end
endfunction
```

Compare1(permet d'avoir l'erreur entre la fonction et la méthode d'Euler)

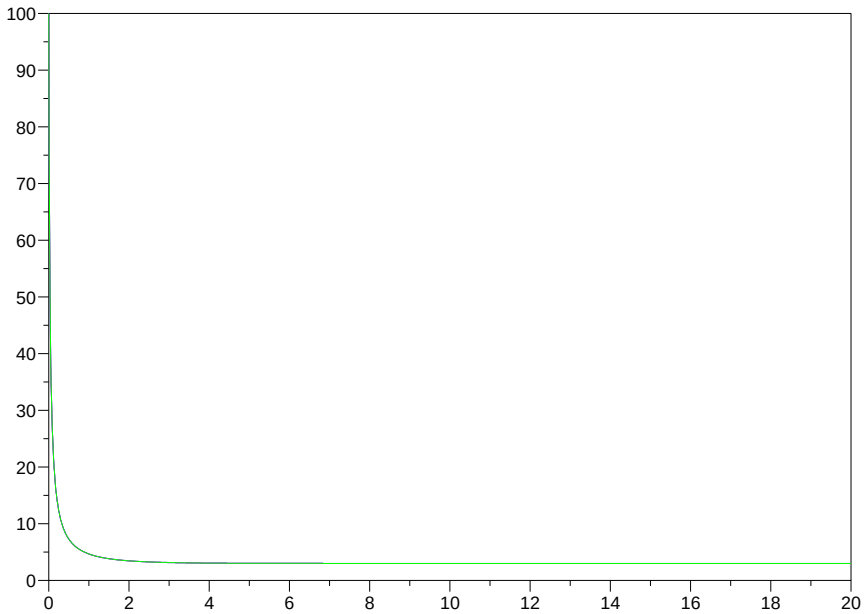
```
function [x]=compare1(t,k,m)
lt=length(t)
  for i=1 :lt
  k1(i)=k(t(i))
  end
x=k1-eulerexplicite(m,N0,t,n)
endfunction
```

Compare2(permet d'avoir l'erreur entre la fonction et la méthode de Runge-Kutta)

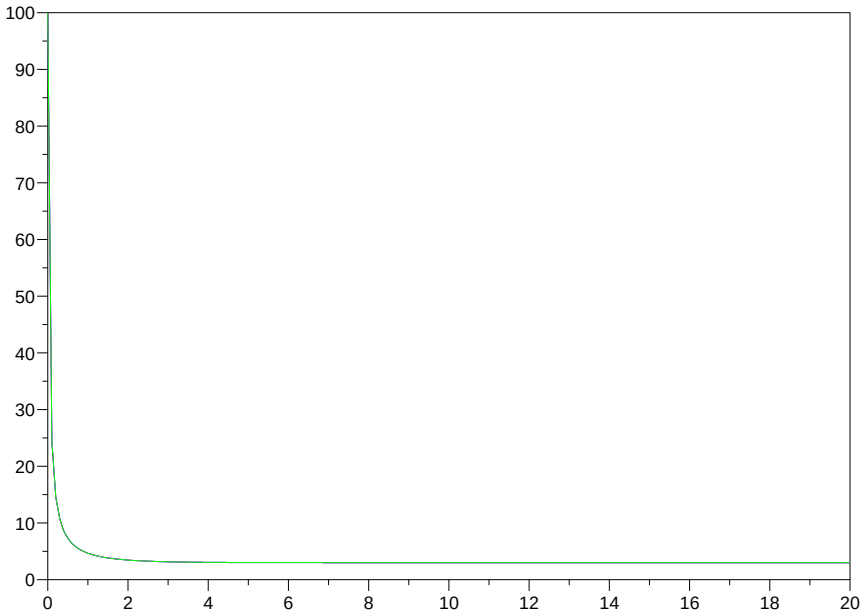
```
function [x]=compare2(t,k,m)
lt=length(t)
  for i=1 :lt
  k1(i)=k(t(i))
  end
x=k1-rungekutta(m,N0,t,100)
endfunction
```

Voici les graphiques représentant en fonction du temps avec différents pas, les solutions de l'équation 1 par la méthode d'Euler (en rose) et de Runge-Kutta (en vert), ainsi que les solutions exactes (en bleu) :

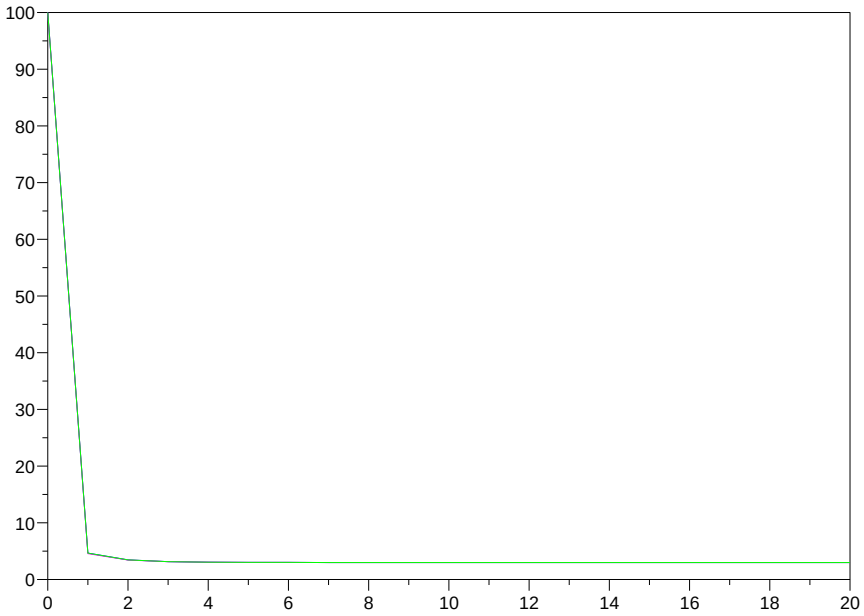
– t=[0 ;20] avec un pas de 0.01



– t=[0 ;20] avec un pas de 0.1

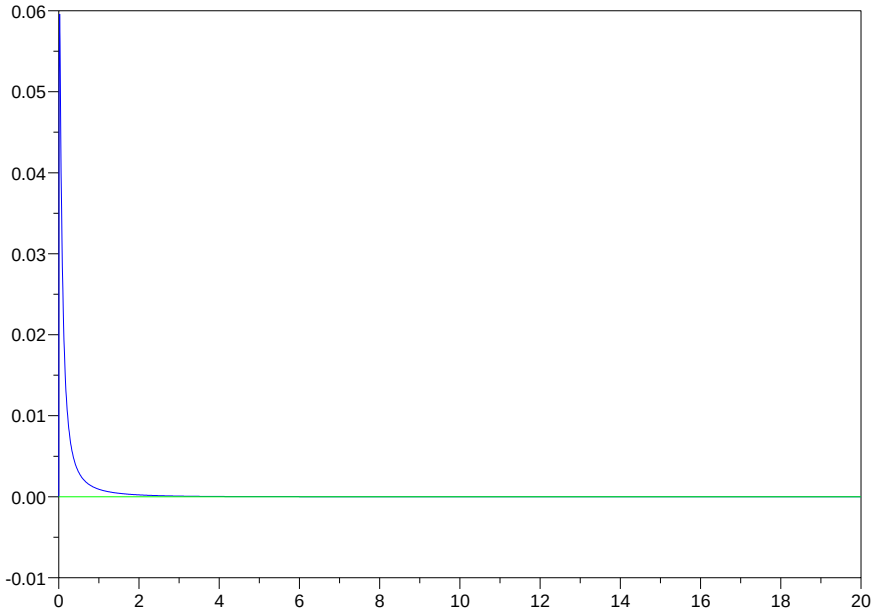


– t=[0 ;20] avec un pas de 1

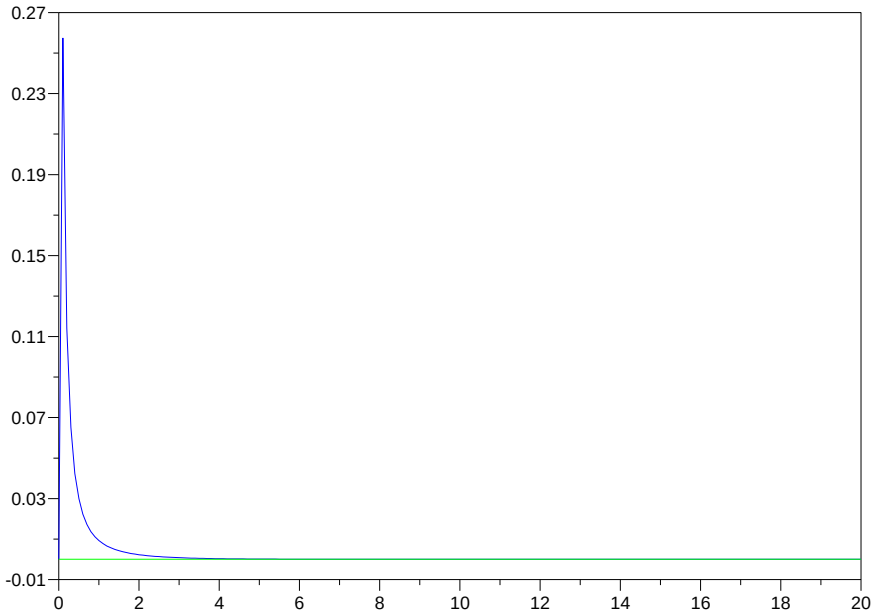


Voici les graphiques des erreurs, en bleu l'erreur de la méthode d'Euler et en vert celle de Runge-Kutta :

– t=[0 ;20] avec un pas de 0.01



– t=[0 ;20] avec un pas de 0.1



Pour tracer, ces différentes courbes nous avons défini les fonctions de la manière suivante :

- la fonction (4) :
  function y=h(t)
  y=K./((((K-N0)./N0).*exp(-r.*t))+1)
  endfunction

- la fonction (1) :
  function y=f(t,d)
  y=r.*d.*(1-(d./K))
  endfunction

- pour tracer les 3 fonctions sur le même graphique (ici n=100) :
  plot2d(t,h(t),style=2)
  plot2d(t,eulerexplicite(f,N0,t,n),style=6)
  plot2d(t,rungekutta(f,N0,t,100),style=3)

- pour tracer les graphiques de comparaison :
  plot2d(t,compare(t,h,f),style=2)
  plot2d(t,compare2(t,h,f),style=3)

Partie 2

Voici les programmes utilisés :

Newton :

```
function [x,iter]=Newton(x0,f,jacf,eps,itermax)
x=x0
fx=f(x)
iter=[0 :0.5 :10] ;
c=[0 :0.5 :10] ;
g=jacf(x)
  while norm(fx)>eps & iter<itermax
  Dx=fx/g
  x=x-Dx
  fx=f(x)
  iter=iter+1
  g=jacf(x)
  end
if norm(fx)>eps & iter>=itermax then
printf(”pas de convergence”)
end
endfunction
```

trac :

```
function u=trac(f,t)
n=length(t)
  for i=1 :n
  x=t(i)
  u(i)=f(x)
  end
endfunction
```

nbreptfixe :

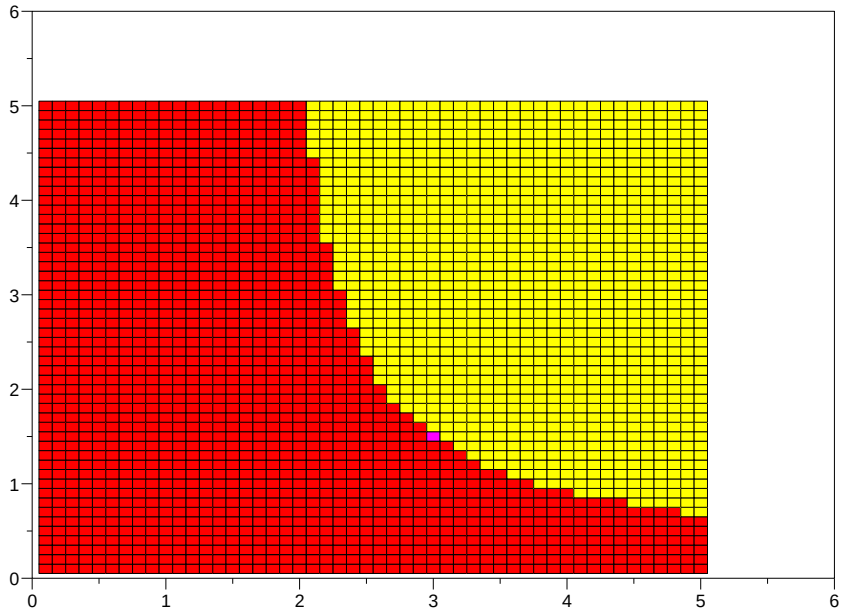
```
function e=nbrptfixe(q,r)
m=4∗(qˆ 2-(3∗(1+(q/r))))
if m>0 then
e=3 //il y a 3 points fixes
  elseif m==0 then
  e=2 //il y a 1 point fixe
    else m<0
    e=1 //il y a 1 point fixe
end
endfunction
```

dessin :

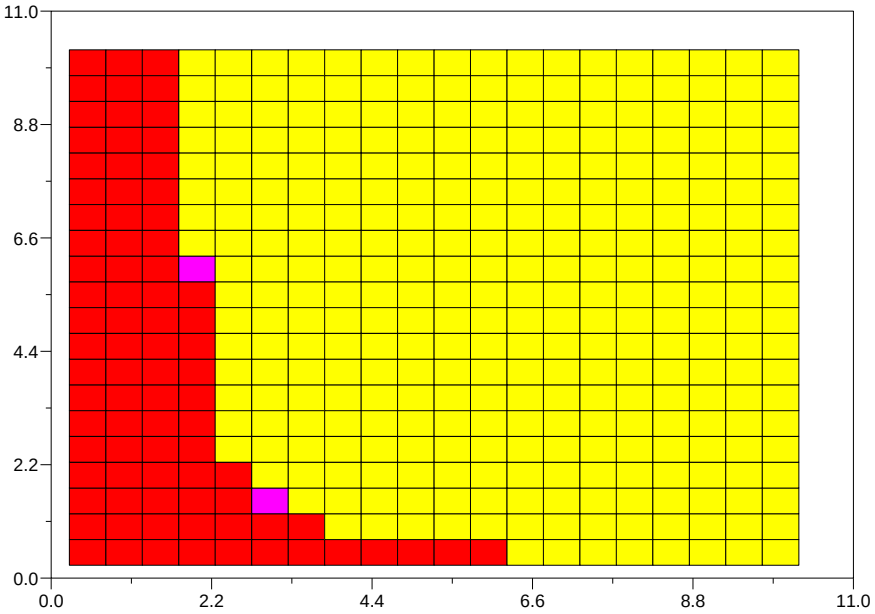
```
function dessin(q,r)
linter1=(q(length(q))-q(1))/(length(q)-1)
linter2=(r(length(r))-r(1))/(length(r)-1)
lr=length(r)
lq=length(q)
for j=2 :lr
for i=2 :lq
solint=nbrptfixe(q(i),r(j))
if solint==1 then
xset("color",5)
xfrect(q(i)-(linter1 /2),r(j)+(linter2 /2),linter1,linter2)
//pour avoir le quadrillage
xset("color",1)
xrect(q(i)-(linter1 /2),r(j)+(linter2 /2),linter1,linter2)
  elseif solint==2 then
  xset("color",6)
  xfrect(q(i)-(linter1 /2),r(j)+(linter2 /2),linter1,linter2)
  //pour avoir le quadrillage
  xset("color",1)
  xrect(q(i)-(linter1 /2),r(j)+(linter2 /2),linter1,linter2)
    else solint=3
    xset("color",7)
    xfrect(q(i)-(linter1 /2),r(j)+(linter2 /2),linter1,linter2)
    //pour avoir le quadrillage
    xset("color",1)
    xrect(q(i)-(linter1 /2),r(j)+(linter2 /2),linter1,linter2)
end
end
end
//pour avoir les axes
xset("color",1)
plot2d(0,0,rect=[0,0,q(lq)+1,r(lr)+1])
endfunction
//jaune=3 solutions
//mauve=1 solution
//rouge=1 solution
```

Voici les représentations de *dessin* (r en fonction de q), en jaune l'équation a 3 solutions et $\Delta > 0$, en mauve $\Delta = 0$ et il y a 1 solution, et enfin en rouge $\Delta < 0$ et il n'y a qu'une solution :

– $(q, r) \in [0; 5]^2$ avec un pas de 0.1



– $(q, r) \in [0; 10]^2$ avec un pas de 0.5



Pour tracer, ces différentes courbes nous avons défini les fonctions de la manière suivante :

- la fonction (6) :
  function y=l(t,u)
  y=r.∗u.∗(1-(u./q))-((u.^ 2)./(1+(u.^ 2)))
  endfunction

- la fonction (7), mais en fonction d un seul paramètre :
  function y=b(u)
  y=r∗u∗(1-(u/q))-((u^ 2)/(1+(u^ 2)))
  endfunction

- la dérivée de la fonction (7) :
  function g=jacb(u)
  g=r-((2∗r∗u)/q)-((2∗u)/(1+u^ 2)^ 2)
  endfunction

- pour tracer la fonction *ode* de h :
  plot2d(t,ode(N1,0,t,l),style=2)

- pour tracer le graphique de la question 3 :
  deff ('y=sol(x)','y=Newton(x,b,jacb,1e-10,500)')
  plot2d(a,trac(sol,a),style=4)

- pour tracer le graphique de *dessin* :
  q=[0 :0.5 :10] ;
  r=[0 :0.5 :10] ;
  dessin(q,r)