BANNWARTH Nicolas
CLERC Emmanuelle
SERRANO Elvyre
Avril 2006

Projet de calcul scientifique

Groupe 9

Sommaire

Introduction

Le but du projet de calcul scientifique est de résoudre le système d'équations aux dérivées ordinaires suivantes :

$$(S)= \begin{cases} dx/dt = -y -z \\ dy/dt = x + ay \\ dz/dt = b + z(x-c) \end{cases} ;$$

Dans un premier temps, nous avons donc déterminé les postions d'équilibre en fonction des paramètres a, b et c du système.

Puis nous avons résolu numériquement grâce à Matlab les solutions du système avec les valeurs des paramètres fixées : a=0,398, b=2 et c=4.

Puis dans un dernier temps, nous avons étudié la périodicité de la solution en fonction de la variation du paramètre a.

I) Les positions d'équilibre

1. But de la recherche

La recherche des positions d'équilibre est très utile pour la poursuite du sujet. En effet elles sont nécessaires pour déterminer numériquement les solutions du système (S). Elles représentent les conditions initiales du système à 10^{-3} près.

2. Détermination des positions d'équilibre

- Considérons le système différentiel suivant :

$$(S)= \begin{cases} dx/dt = -y -z \\ dy/dt = x + ay \\ dz/dt = b + z(x-c) \end{cases} ;$$

- Etudions les positions d'équilibre en fonction des paramètres a, b et c.

F(t)=(x(t),y(t),z(t)) admet des points fixes ou des positions d'équilibre si et seulement si :

$$(S') = \begin{cases} dx/dt = 0 \\ dy/dt = 0 \\ dz/dt = 0 \end{cases}$$

$$\text{Donc } (S') = \begin{cases} dx/dt = 0 \\ dy/dt = 0 \\ dz/dt = 0 \end{cases} \Leftrightarrow \begin{cases} -y -z = 0 \\ x + ay = 0 \\ b + z(x-c) = 0 \end{cases} \Leftrightarrow \begin{cases} y = -z \\ x = - ay \\ z = -b / (x-c) \end{cases} \Leftrightarrow \begin{cases} y = -z \\ x = - ay \\ z = -b / (x-c) \end{cases}$$

$$(S') \Leftrightarrow \begin{cases} y = -z \\ x = - ay \\ z = b / (ay+c) \end{cases} \Leftrightarrow \begin{cases} y = -z \\ x = - ay \\ b / (ay+c) = -y \end{cases} \Leftrightarrow \boxed{\begin{cases} y = -z \\ x = - ay \\ ay^2 + cy + b = 0 \end{cases}}$$

- Déterminons les valeurs de y :

Pour déterminer les valeurs de y, il faut résoudre l'équation (E) : $ay^2+cy+b=0$. Dans un premier temps, nous calculons le discriminant, puis nous déterminons les solutions réelles de

l'équation. En effet, seules les solutions réelles nous intéressent car nous étudions un système différentiel réel.

(E) $\Leftrightarrow$ $ay^2+cy+b=0$ d'où $\Delta=c^2-4ab$;

Donc nous déterminons les deux solutions suivantes ($\Delta \geq 0$):

$y_1 = (-c +\sqrt{\Delta})/2a = (-c +\sqrt{(c^2-4ab)})/2a$;
$y_2 = (-c -\sqrt{\Delta})/2a = (-c -\sqrt{(c^2-4ab)})/2a$;

- Déterminons les positions d'équilibre en fonctions des valeurs de y et calculons les pour les valeurs des paramètres (a=0,398, b=2 et c=4) :

D'où :

$$\begin{cases} x_1 = (c - \sqrt{(c^2-4ab)}) / 2 \approx 0,2100279332 \\ y_1 = (-c +\sqrt{(c^2-4ab)})/2a \approx -0,5277083748 \\ z_1 = (c -\sqrt{(c^2-4ab)})/2a \approx 0,5277083748 \end{cases}$$

et

$$\begin{cases} x_2 = (c + \sqrt{(c^2-4ab)}) / 2 \approx 3,789972067 \\ y_2 = (-c -\sqrt{(c^2-4ab)})/2a \approx -9,522542881 \\ z_2 = (c +\sqrt{(c^2-4ab)})/2a \approx 9,522542881 \end{cases}$$

II) La résolution du système

1. Utilisation de Runge Kutta d'ordre 4

Le système d'équations aux dérivées ordinaires (S) n'est pas linéaire, nous ne pouvions donc pas le résoudre grâce à la méthode de diagonalisation de matrice que nous avions vu en classe préparatoire. Ne sachant pas résoudre ce système sans passer par une approximation, nous avons donc décidé d'utiliser un schéma numérique.

Nous avons donc dû déterminer quel type de schéma était le plus approprié pour cette résolution. Afin d'avoir un modèle programmable sur Matlab et assez fin, nous avons choisi d'utiliser un schéma de Runge Kutta d'ordre 4. Les schémas d'Euler et de Runge Kutta d'ordre inférieur ne donnent pas une précision satisfaisante.

2. Programmation du système

Afin de réaliser Runge Kutta à l'ordre 4, nous avons commencé par déterminer le schéma à la main pour le système (S). Nous avons donc écrit les équations de k_0, k_1, k_2, k_3 et k_4 en x(t), en y(t) et en z(t).

Puis nous avons programmé le système, afin de calculer de proche en proche les valeurs de x(t), y(t) et z(t). Pour cela nous avons donc du déterminer le pas de Runge Kutta, les conditions initiales et le nombre d'itérations que nous devions réaliser afin d'obtenir une bonne approximation de la solution du système.

- Vérification de Runge Kutta :

Notre programme est directement lié à notre énoncé, nous ne pouvions donc pas tester d'autre fonction. Néanmoins pour vérifier la validité de notre résultat, nous avons appliqué notre problème à une autre programmation de Runge Kutta (cf Annexe 2).

- Détermination des conditions initiales :

Comme nous l'indiquions dans la première partie du rapport, la détermination des conditions d'équilibre était essentielle afin d'initialiser le programme.

Pour lancer le programme de Runge Kutta, il faut utiliser des valeurs initiales pour x(t), y(t) et z(t) qui soit proche de la situation d'équilibre. Nous avons donc choisi après plusieurs essais d'utiliser les valeurs d'équilibre à 10^{-3} près.

Pour la deuxième position d'équilibre, le schéma tend vers l'infini (figure n°0). Nous étudions donc la première position d'équilibre.

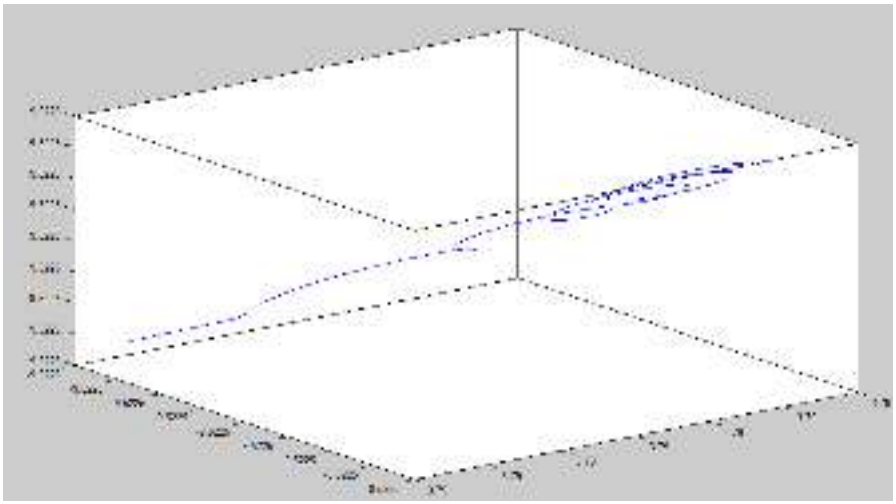

Figure n°0 : Solution avec N=10000 et h=0,01 pour le deuxième point d'équilibre

- Détermination du pas :

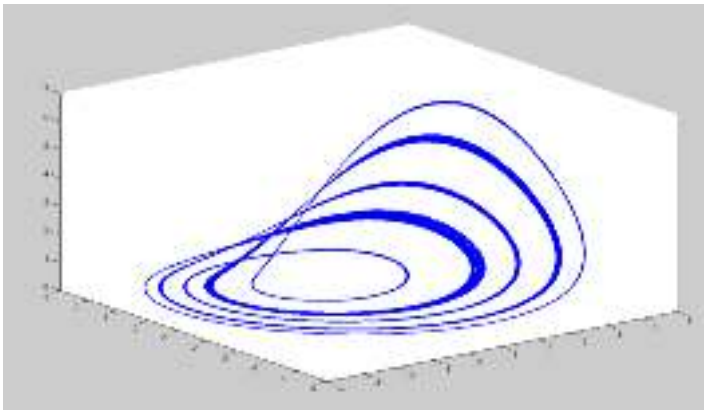

Figure n°1 : Solution avec un pas de 0,1

Avec un pas de 0.1, la solution obtenue est assez peu précise. Il faut donc trouver un compromis entre la précision de la solution et la rapidité d'exécution du programme par la machine.

Après avoir effectué des tests de rapidité et de qualité de la solution trouvée, nous avons décidé que un pas de 0,01 était un bon compromis. Nous avons donc travaillé durant tout le projet avec ce pas.

- Détermination du nombre d'itérations :

Pour un nombre d'itérations trop petit la solution n'est pas complète. Avec le pas choisi, le nombre d'itération minimum est voisin de 50 000.

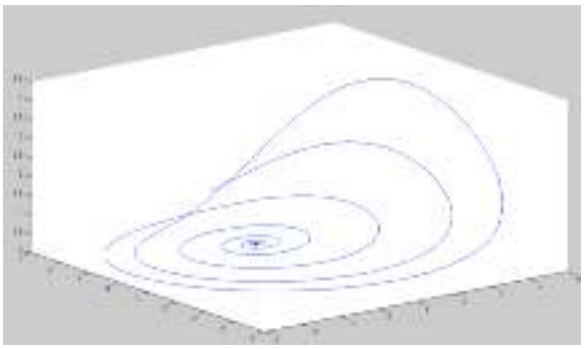

Figure n°2 : Solution avec 10000 itérations

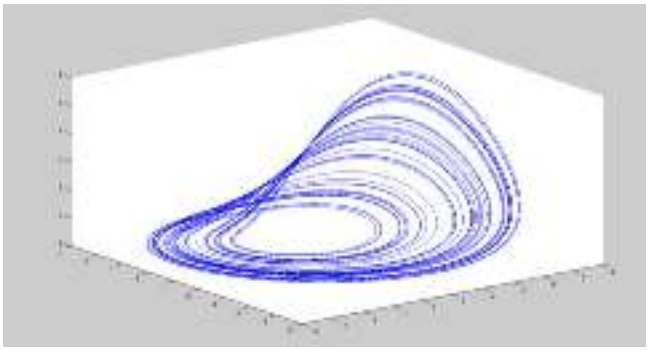

Figure n°3 : Solution avec 50000 itérations

3. Résultats

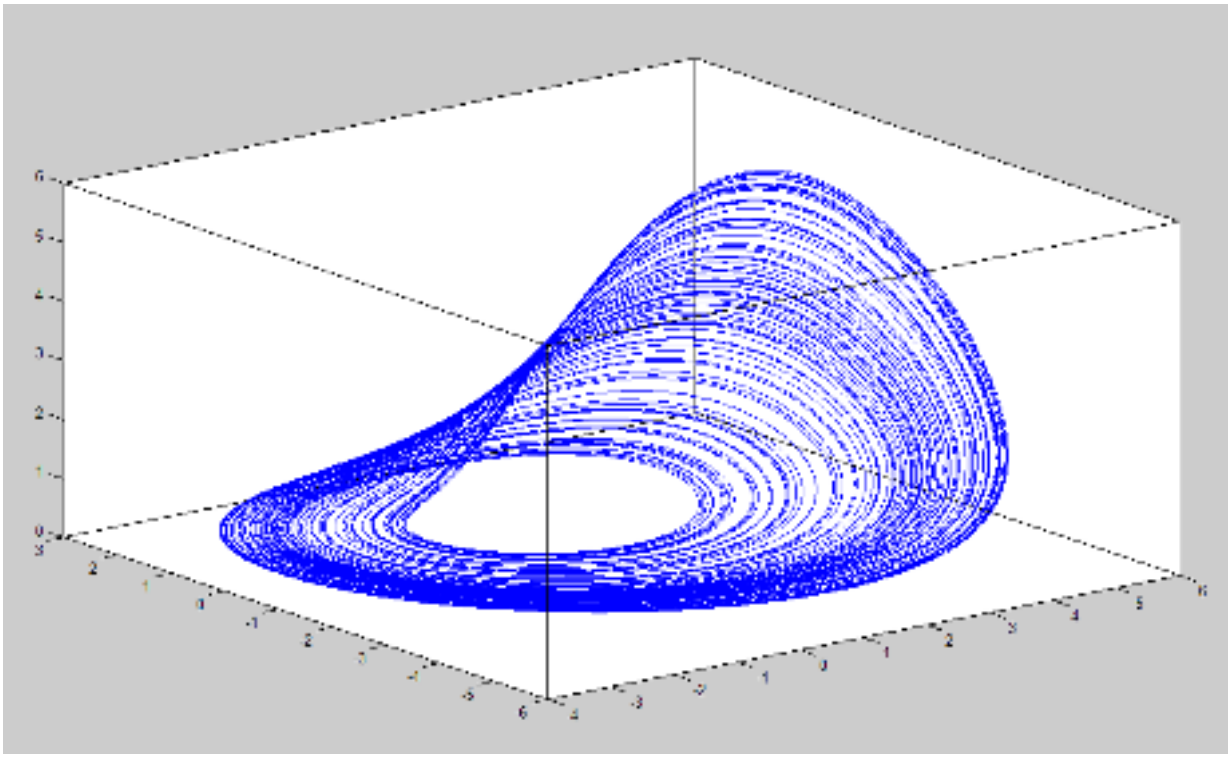

Figure n°4 : Solution du système d'équation aux dérivées ordinaires (S)

Cette figure représente la solution du système d'équation aux dérivées ordinaires (S) avec le paramétrage suivant :

- Conditions initiales : $\begin{cases} x_0 = 0{,}2110279332 \\ y_0 = -0{,}5287083748 \\ z_0 = 0{,}5287083748 \end{cases}$

- Pas : h = 0,01
- Nombre d'itérations : N = 50 000

III) Recherche de la périodicité de la solution

1. Méthode de recherche

Le but de la troisième question du sujet est de trouver la périodicité de la solution obtenue en fonction du paramètre a. Dans le paragraphe précédent, la solution a été trouvée avec le paramètre a égal à 0,398.

Nous avons donc fait varier ce paramètre, dans le programme Matlab, afin d'observer les modifications de la solution. Nous avons procédé de la manière suivante :

- Nous avons choisi la valeurs du paramètre a.
- Nous avons fait tourner le programme Matlab au minimum deux fois car à la fin de chaque passage le programme garde en mémoire la dernière position sur la courbe et continue à tourner à partir de celle-ci.
- Enfin nous avons compté le nombre de boucle obtenue sur le graphique.
- Puis nous avons recommencé en diminuant la valeur de a de 10^{-3} en 10^{-3}.

Afin d'affiner le résultat obtenu, nous avons travaillé avec des valeurs de a de plus en plus petites.

Cette méthode nous a donc permis d'obtenir la périodicité de la solution du système d'équations aux dérivées ordinaires.

2. Résultats

En appliquant la méthode ci-dessus nous avons obtenu les résultats suivants :

a début de période	0,35000000	0,37400000	0,38200000	0,38425000	0,38515000	0,38515835	0,38515950
a fin de période	0,37300000	0,38175000	0,38400000	0,38500000	0,38515825	0,38515900	0,38515956
Longueur de l'intervalle	0,02300000	0,00775000	0,00200000	0,00075000	0,00000825	0,00000065	0,00000006
Nombre de courbes	1	2	4	8	16	32	64

Figure n°5 : Résultats des périodes

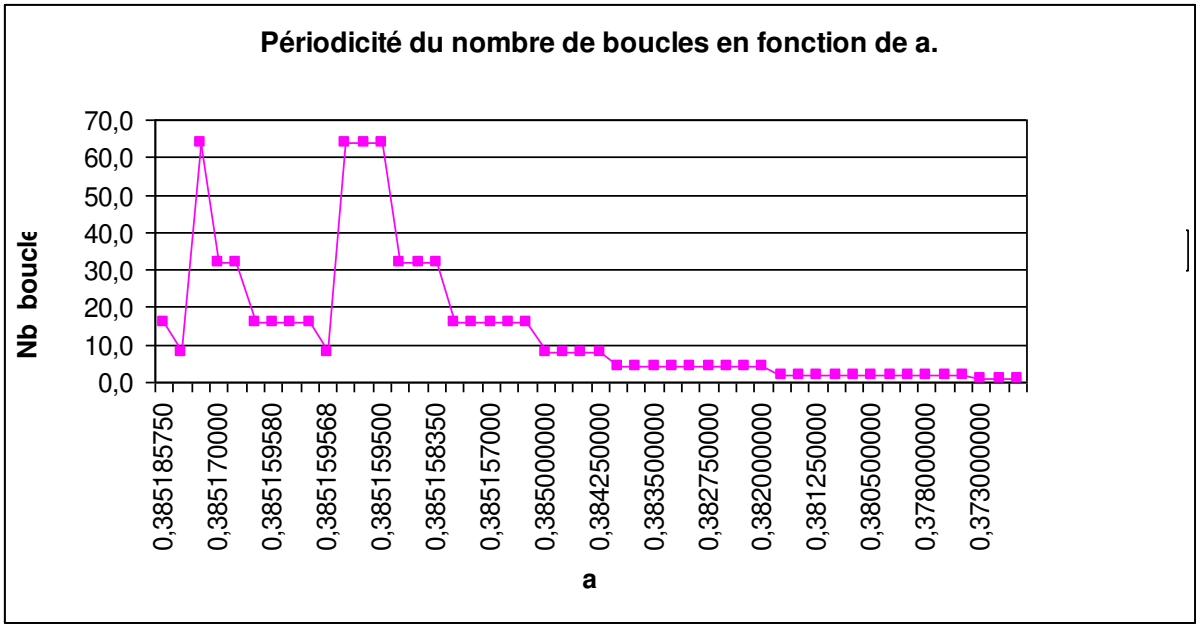

Figure n°6 : Résultat graphique de la périodicité (échelle non linéaire)

Voici quelques exemples des courbes obtenues :

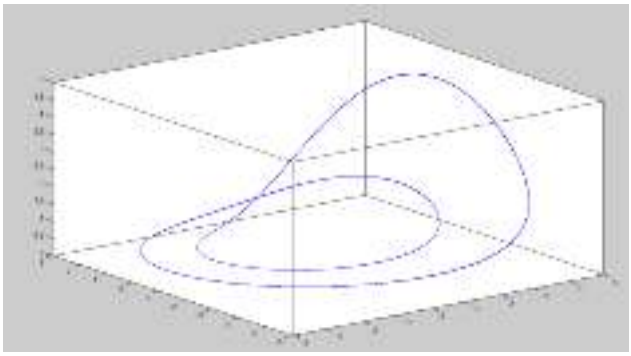

Figure n°7 : Solution avec a = 0,371

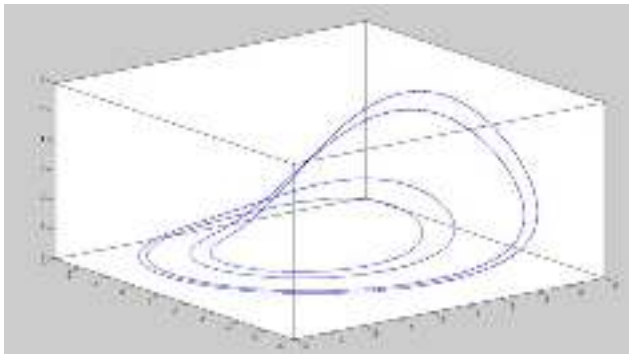

Figure n°8 : Solution avec a = 0,38

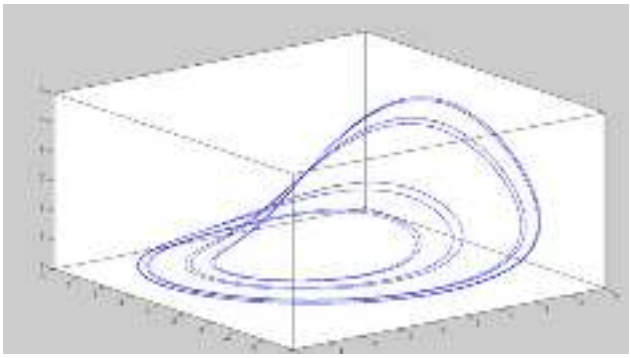

Figure n°9 : Solution avec a = 0,38375

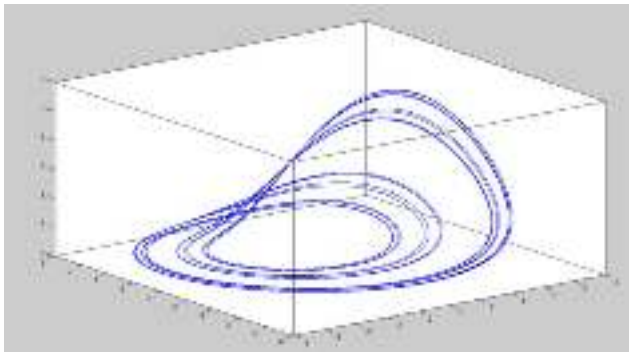

Figure n°10 : Solution avec a = 0,385

Conclusion

Grâce à un schéma numérique, nous avons obtenu les solutions d'un système d'équations aux dérivées ordinaires non linéaire.

La précision de Matlab, ne donnant que huit décimales, nous n'avons pu pousser plus loin la recherche de la périodicité de la solution en fonction de a.

Annexe 1 : Programme Matlab

```
function runge2(x,y,z,a)
hold off;

%Initialisation des constantes
N=50000;
k=1;
 b=2; c=4;
h=0.01; % pas


X=zeros(1,N);  %Vecteurs qui stockent x(t),y(t) et z(t)
Y=zeros(1,N);
Z=zeros(1,N);

X(1)=x;
Y(1)=y;
Z(1)=z;

for k=1:N

   x(k)=-Y(k)-Z(k);
   y(k)=X(k)+a*Y(k);
   z(k)=b+Z(k)*(X(k)-c);
   %(Z(k)*x(k)+z(k)*X(k))=Z(k)*(-Y(k)-Z(k))+X(k)*(b+Z(k)*(X(k)-c))


   k1(1)=x(k);
   k2(1)=x(k)+h/2*-(y(k)-z(k));
   k3(1)=x(k)+h/2*(-y(k)-z(k)+h/2*(-x(k)-a*y(k)+c*(z(k)-(Z(k)*x(k)+z(k)*X(k)))));
   k4(1)=x(k)+h*(-y(k)-z(k)+h/2*(-
(x(k)+a*y(k)+(Z(k)*x(k)+z(k)*X(k)))+c*z(k))+h/2*(y(k)+z(k)-
a*(x(k)+a*y(k))+c*((Z(k)*x(k)+z(k)*X(k))-c*z(k)-(-Z(k)*y(k)-z(k)*Y(k)-
2*Z(k)*z(k)+b*x(k)+z(k)*(X(k))^2+2*X(k)*x(k)*Z(k)-c*(Z(k)*x(k)+z(k)*X(k))))));


   k1(2)=y(k);
   k2(2)=y(k)+h/2*(x(k)+a*y(k));
   k3(2)=y(k)+h/2*(x(k)+a*y(k)+h/2*(-y(k)-z(k)+a*(x(k)+y(k))));
   k4(2)=y(k)+h*(x(k)+a*y(k)+h/2*(-y(k)-z(k)+a*(x(k)+y(k))+h/2*(-x(k)-a*y(k)-
(Z(k)*x(k)+z(k)*X(k))+c*z(k)+a*(-y(k)-z(k)+a*(x(k)+a*y(k))))));

   k1(3)=z(k);
   k2(3)=z(k)+h/2*((Z(k)*x(k)+z(k)*X(k))-c*z(k));
   k3(3)=z(k)+h/2*((Z(k)*x(k)+z(k)*X(k))-c*z(k)+h/2*(-
(Z(k)*y(k)+Y(k)*z(k)+2*Z(k)*z(k))+b*x(k)+2*Z(k)*x(k)*X(k)+z(k)*(X(k))^2-
c*(Z(k)*x(k)+X(k)*z(k))-c*(Z(k)*x(k)+z(k)*X(k)-c*z(k))));
   k4(3)=z(k)+h*((Z(k)*x(k)+z(k)*X(k))-c*z(k)+h/2*(-(Y(k)*z(k)+Z(k)*y(k))-2*Z(k)*z(k)-
c*(Z(k)*x(k)+z(k)*X(k))+c^2*z(k)+h/2*(-((Z(k)*x(k)+z(k)*X(k)))-
```

```
a*(Y(k)*z(k)+Z(k)*y(k))+b*y(k)+x(k)*Y(k)*Z(k)+X(k)*Y(k)*z(k)+X(k)*y(k)*Z(k)-
c*(X(k)*y(k)+Y(k)*x(k))+2*b*z(k)+2*((Z(k))^2*x(k)+2*Z(k)*z(k)*X(k))-
2*c*(2*Z(k)*z(k))-b*y(k)-b*z(k)-2*(x(k)*Z(k)*Y(k)+X(k)*Y(k)*z(k)+X(k)*y(k)*Z(k))-
2*((Z(k))^2*x(k)+2*Z(k)*z(k)*X(k))+b*2*X(k)*x(k)+3*Z(k)*x(k)*(X(k))^2+(X(k))^3*z(k)-
c*(2*Z(k)*X(k)*x(k)+(X(k))^2*z(k))+c*(Z(k)*y(k)+Y(k)*z(k)+2*Z(k)*z(k)-b*x(k)-
z(k)*(X(k))^2-2*Z(k)*X(k)*x(k)+(c-1)*((Z(k)*x(k)+z(k)*X(k)))+c*z(k)))));

   X(k+1)=X(k)+h/6*(k1(1)+2*k2(1)+2*k3(1)+k4(1));
   Y(k+1)=Y(k)+h/6*(k1(2)+2*k2(2)+2*k3(2)+k4(2));
   Z(k+1)=Z(k)+h/6*(k1(3)+2*k2(3)+2*k3(3)+k4(3));


end

% fenetre graphique : trace la surface

plot3(X,Y,Z);
hold on;
box on;
d=input('taper une touche','s');

clear figure(1)
X=X(N);
Y=Y(N);
Z=Z(N);
runge2(X,Y,Z,a)

end
```

Annexe 2 : Programme Matlab de vérification

```
%h est le pas de temps, T la durée de l'expérience, x0, y0 et z0 les conditions initiales

function [y]=run(h,T,x0,y0,z0)
%Permet de résoudre la réaction BZ: h est le pas de temps, T la durée de
%l'expérience, x0, y0 et z0 les conditions initiales, j représente le pas
%d'incrémentation lors de la représentation graphique
tic

global j

format long;

%On définit les conditions initiales
X0=[x0;y0;z0];
t0=0;
t = t0;
X = X0;
X_old = X0; X_new = [];

disp('Bienvenue sur le programme de résolution d''équations différentielles');
disp(' ');

%On introduit les modes de fonctionnement
disp('Vous avez le choix entre plusieurs méthodes de résolution:');
disp('Tapez 1 pour la méthode de Runge-Kutta d''ordre 4');
disp('Tapez 2 pour la méthode de Runge-Kutta d''ordre 4 (avec contrôle du pas)');
disp(' ');

%On met en place la variable choix_valide pour éviter tout bug
choix_valide=0;
while choix_valide~=1
   choix=input('Quel mode voulez-vous choisir?');
   if choix==1
      choix_valide=1;
      %Solution par la méthode de Runge-Kutta ordre 4
      for tt = t0:h:(t0+T-h)
         k1 = Elvyre(tt, X_old);
         k2 = Elvyre(tt, X_old+(h/2)*k1);
         k3 = Elvyre(tt, X_old+(h/2)*k2);
         k4 = Elvyre(tt, X_old+h*k3);
         X_new = X_old + h/6*(k1+2*k2+2*k3+k4);
         t = [t , tt];
         X= [X, X_new];
         X_old = X_new;
      end;
```

```
    %On affiche le graphe
    plot3(X(1,:),X(2,:),X(3,:));
    box on;
    toc
    break
  elseif choix==2
    choix_valide=1;
    %Solution par la méthode de Runge-Kutta ordre 4 (avec contrôle de
    %pas)
    alpha=input('Tolérance alpha:');
    alphaprime=input('Tolérance alphaprime: (d''habitude alphaprime est alpha/2^(p+1) ');
    i=1;
    j=0;
    nb_iterations=input('Nombre d''itérations ?');
    while t(i)<=T;
      d=alpha*h;
      while d>= alpha*h
        %Calcul avec le pas h
        k1 = h*Elvyre(t, X_old);
        k2 = h*Elvyre(t, X_old+k1/2);
        k3 = h*Elvyre(t, X_old+k2/2);
        k4 = h*Elvyre(t, X_old+k3);
        X_chapeau = X_old + (1/6)*(k1+2*k2+2*k3+k4);

        %Calcul avec le pas h/2, 1ère itération
        l1 = h/2*Elvyre(t, X_old);
        l2 = h/2*Elvyre(t, X_old+l1/2);
        l3 = h/2*Elvyre(t, X_old+l2/2);
        l4 = h/2*Elvyre(t, X_old+l3);
        X_auxiliaire1 = X_old+1/6*(l1+2*l2+2*l3+l4);

        %Calcul avec le pas h/2, 2ème itération
        m1 = h/2*Elvyre(t, X_auxiliaire1);
        m2 = h/2*Elvyre(t, X_auxiliaire1+m1/2);
        m3 = h/2*Elvyre(t, X_auxiliaire1+m2/2);
        m4 = h/2*Elvyre(t, X_auxiliaire1+m1);
        X_auxiliaire2 = X_auxiliaire1+1/6*(m1+2*m2+2*m3+m4);

        d=abs(X_chapeau-X_auxiliaire2)/31;
        h=h/2;
      end
      X_new=X_chapeau;
      X= [X, X_new];
      X_old = X_new;

      if d < (alphaprime*h)
        h=2*h;
      end;

      t(i+1)=t(i)+h;
```

```
        i=i+1;

        affichage_pas3(nb_iterations,t,i,h,X);
      end

      %On affiche le graphe
      subplot(311);
      plot(t(1,:),X(1,:),'r','Linewidth', 1.5);
      xlabel('Temps','Fontname', 'Arial Narrow', 'FontSize', 12);
      ylabel('Concentration en x','Fontname', 'Arial Narrow', 'FontSize', 12);
      title('Résolution par la méthode de Runge-Kutta d''ordre 4 (avec contrôle du pas');
      subplot(312);
      plot(t(1,:),X(2,:),'b','Linewidth', 1.5);
      xlabel('Temps','Fontname', 'Arial Narrow', 'FontSize', 12);
      ylabel('Concentration en y','Fontname', 'Arial Narrow', 'FontSize', 12);
      subplot(313);
      plot(t(1,:),X(3,:),'g','Linewidth', 1.5);
      xlabel('Temps','Fontname', 'Arial Narrow', 'FontSize', 12);
      ylabel('Concentration en z','Fontname', 'Arial Narrow', 'FontSize', 12);
      box on;
      toc
      break
   end
   disp('Vous vous etes trompés. Veuillez taper un nombre entre 1 et 2');
   disp(' ');
end
```

fonctions annexes:

- function Y = Elvyre(t,X)

```
% déclaration des variables globales définies à l'extérieur de la fonction
%global epsilon1,epsilon2,q;

% Vérification des dimensions 3 x 1
[m,n] = size(X);
 if ((m~=3) | (n~=1))
   error('Vous vous êtes plantés dans les conditions initiales')
 end

Y1 = -X(2)-X(3);
Y2 = X(1)+0.398*X(2);
Y3 = 2+X(3)*(X(1)-4);

% Définition du vecteur colonne Y
Y = [ Y1; ...
     Y2; ...
     Y3
```

];

- function affichage_pas(nb_iterations,t,i,h,X)

```
%Permet d'afficher le résultat au bout d'un certain nombre d'itérations

global j

if j==nb_iterations
   j=0;
   t(i)
   h

   figure
   %On affiche le graphe
   subplot(311);
   plot(t(1,:),X(1,:),'r','Linewidth', 1.5);
   xlabel('Temps','Fontname', 'Arial Narrow', 'FontSize', 12);
   ylabel('Concentration en x','Fontname', 'Arial Narrow', 'FontSize', 12);
   title('Résolution par la méthode de Runge-Kutta d''ordre 4 (avec contrôle du pas)');
   subplot(312);
   plot(t(1,:),X(2,:),'b','Linewidth', 1.5);
   xlabel('Temps','Fontname', 'Arial Narrow', 'FontSize', 12);
   ylabel('Concentration en y','Fontname', 'Arial Narrow', 'FontSize', 12);
   subplot(313);
   plot(t(1,:),X(3,:),'g','Linewidth', 1.5);
   xlabel('Temps','Fontname', 'Arial Narrow', 'FontSize', 12);
   ylabel('Concentration en z','Fontname', 'Arial Narrow', 'FontSize', 12);
   box on;

   input('');
end
j=j+1;
```