Projet P3 : Analyse Numérique
Méthode d'Adams-Bashforth-2

Conrad Hillairet et Alexandre Vieira

2 août 2013

Table des matières

Introduction

Les équations différentielles sont utiles dans de nombreux domaines scientifiques. On les utilise bien souvent en physique, en chimie ou même dans la finance.
Bien qu'on sache comment résoudre des équations différentielles, toutes ne sont pas forcément résolvable avec les connaissances actuelles. Pour cela, on cherche à avoir une solution approchée de ces équations via des méthodes itératives. Il en existe plusieurs sortes :

— Les méthodes à un pas, où chaque itération ne dépend que de la précédente. On peut citer comme exemple la méthode d'Euler
— Les méthodes à pas liés, dont les suites récurrentes sont au moins d'ordre deux.

La méthode d'Adams-Bashforth-2, développée dans ce rapport, est une méthode à pas liés. Elle est définie ainsi :

$$\begin{cases} y_{t+1} &= y_t + \frac{h}{2}(3f_t - f_{t-1}) \\ y(x_0) &= y_0 \end{cases}$$

où :

$$f_t = f(x_t, y_t)$$

$$y_t = y(x_t)$$

Ce rapport se partagera ainsi :
— Nous commencerons par présenter la méthode d'Euler et la méthode d'Adams-Bashforth-2, ainsi que différents théorèmes
— Nous présenterons ensuite différentes implémentations en C
— Nous testerons enfin nos codes sur des exemples que nous commenterons.

1 Présentation des méthodes d'Euler et d'Adams-Bashforth-2

1.1 Méthode d'Euler

Cette méthode est utilisée pour initialiser la méthode d'Adams. Elle a en effet besoin de deux points à chaque itération, or, le problème de Cauchy n'en donne qu'un seul.

1.1.1 Définition de la méthode

Pour définir la méthode d'Euler, repartons du système différentiel, qu'on définit sur $[x_t, x_{t+1}]$, $t \in \mathbb{N}$, intervalle de longueur h :

$$\begin{cases} dy = f(x,y)dx \\ y(x_t) = y_t \end{cases}$$

Ce système équivaut à :

$$\int_{y_t}^{y_{t+1}} dy = \int_{x_t}^{x_{t+1}} f(x,y)dx$$

On approche la deuxième intégrale par la méthode des rectangles à gauche :

$$\int_{x_t}^{x_{t+1}} f(x,y)dx \approx \underbrace{(x_{t+1} - x_t)}_{=h} f(x_t, y_t)$$

Ainsi :

$$y_{t+1} \approx y_t + hf(x_t, y_t)$$

1.1.2 Stabilité et convergence

— Si f est lipschitzienne par rapport à y, la méthode est stable.
— La méthode est convergente.

1.2 Méthode d'Adams-Bashforth-2

1.2.1 Définition de la méthode

L'idée de cette méthode est simple : on interpole à l'ordre 1 la fonction $f(x,y)$ entre x_{t-1} et x_t. Ainsi :

$$\begin{aligned} f(x,y) &\approx f(x_t, y_t) + \frac{f(x_{t-1}, y_{t-1}) - f(x_t, y_t)}{x_{t-1} - x_t}(x - x_t) \\ &\approx f_t + \frac{f_{t-1} - f_t}{x_{t-1} - x_t}(x - x_t) \end{aligned}$$

On pose $L_2(f,x) = f_t + \frac{f_{t-1} - f_t}{x_{t-1} - x_t}(x - x_t)$.

On repart à présent du système différentiel de départ, qu'on définit entre x_t et x_{t+1}. On considère qu'entre chaque x_i et x_{i+1}, on a un pas constant h :

$$\begin{cases} \frac{dy}{dx} &= f(x,y) \\ y(x_t) &= y_t \end{cases}$$

Ce système équivaut à :

$$y_{t+1} = y_t + \int_{x_t}^{x_{t+1}} f(x,y)dx$$

On utilise ici l'approximation de f. On calcule pour cela $L_2(f, x_t)$ et $L_2(f, x_{t+1})$.

$$\begin{aligned} L_2(f, x_t) &= f_t \\ L_2(f, x_{t+1}) &= f_t + \underbrace{\frac{f_{t-1} - f_t}{x_{t-1} - x_t}}_{=-h} \underbrace{(x_{t+1} - x_t)}_{=h} \\ &= 2f_t - f_{t-1} \end{aligned}$$

D'où :

$$\begin{aligned} \int_{x_t}^{x_{t+1}} f(x, y)dx &\approx \int_{x_t}^{x_{t+1}} L_2(f, x)dx \\ &\approx \frac{h}{2}(L_2(f, x_{t+1}) + L_2(f, x_t)) \\ &\approx \frac{h}{2}(3f_t - f_{t-1}) \end{aligned}$$

D'où la méthode d'Adams-Bashforth-2 :

$$y_{t+1} = y_t + \frac{h}{2}(3f_t - f_{t-1})$$

1.2.2 Convergence de la méthode

Démontrons qu'il existe η tel que :

$$\varepsilon(t, h) = \frac{5}{12}h^3 y^{(3)}(\eta)$$

Où $\varepsilon(t, h)$ est l'erreur sur la méthode.
Considérons d'abord l'approximation de la solution au point t :

$$\begin{aligned} y(t) &\approx y(t-h) + \frac{h}{2}\left[3f(t-h, y(t-h)) - f(t-2h, y(t-2h))\right] \\ &\approx y(t-h) + \frac{h}{2}[3y'(t-h) - y'(t-2h)] \end{aligned}$$

On utilise à présent la formule de Taylor :

$$\begin{aligned} y(t-h) &= y(t) - hy'(t) + \frac{h^2}{2}y''(t) - \frac{h^3}{6}y^{(3)}(\eta) \\ y'(t-h) &= y'(t) - hy''(t) + \frac{h^2}{2}y^{(3)}(\eta) \\ y'(t-2h) &= y'(t) - 2hy''(t) + 2h^2 y^{(3)}(\eta) \end{aligned}$$

On a donc :

$$\begin{aligned} \varepsilon(t, h) &= y(t) - y(t-h) - \frac{h}{2}[3y'(t-h) - y'(t-2h)] \\ &= hy'(t) - \frac{h^2}{2}y''(t) + \frac{h^3}{6}y^{(3)}(\eta) - \frac{3h}{2}y'(t) + \frac{3h^2}{2}y''(t) - \frac{3h^3}{4}y^{(3)}(t) + \frac{h}{2}y'(t) - h^2 y''(t) + h^3 y^{(3)}(\eta) \\ &= hy'(t)\left(1 - \frac{3}{2} + \frac{1}{2}\right) + h^2 y''(t)\left(-\frac{1}{2} + \frac{3}{2} - 1\right) + h^3 y^{(3)}(\eta)\left(\frac{1}{6} - \frac{3}{4} + 1\right) \\ &= \frac{5}{12}h^3 y^{(3)}(\eta) \end{aligned}$$

Donc la méthode converge.

1.2.3 Stabilité de la méthode

Si f est lipschitzienne, alors la méthode converge.
Définissons pour cela deux méthodes :

$$\left\{ \begin{array}{rcc} y_{j+1} & = & y_j + \frac{h}{2}[3f(x_j, y_j) - f(x_{j-1}, y_{j-1})] \\ y(x_0) & = & y_0 \end{array} \right.$$

$$\left\{ \begin{array}{rcc} z_{j+1} & = & z_j + \frac{h}{2}[3f(x_j, z_j) - f(x_{j-1}, z_{j-1})] + \varepsilon_j \\ z(x_0) & = & z_0 \end{array} \right.$$

$$\begin{array}{rcl} |y_{j+1} - z_{j+1}| & \leq & |y_j - z_j| + \dfrac{h}{2}[3|f(x_j, y_j) - f(x_j, z_j)| + |f(x_{j-1}, y_{j-1}) - f(x_{j-1}, z_{j-1})|] + |\varepsilon_j| \\ & \leq & |y_j - z_j| + \dfrac{b-a}{2}[3M|y_j - z_j| + M|y_{j-1} - z_{j-1}|] + |\varepsilon_j| \text{ car } f \text{ lipschitzienne} \\ & \leq & C(|y_j - z_j| + |y_{j-1} - z_{j-1}| + |\varepsilon_j|) \text{ avec } C = \dfrac{3}{2}(b-a)M + 1 > 0 \\ & \leq & C((1+C)|y_{j-1} - z_{j-1}| + |y_{j-2} - z_{j-2}| + |\varepsilon_j| + |\varepsilon_{j-1}|) \\ & \leq & C'(|y_{j-1} - z_{j-1}| + |y_{j-2} - z_{j-2}| + |\varepsilon_j| + |\varepsilon_{j-1}|) \\ & \leq & \vdots \\ & \leq & C''\left(|y_1 - z_1| + |y_0 - z_0| + \displaystyle\sum_{i=1}^{j} |\varepsilon_i|\right) \end{array}$$

Or, on a $|y_1 - z - 1| \leq C(|y_0 - z_0| + |\varepsilon_0|)$ car la méthode est stable. Ainsi :

$$|y_{j+1} - z_{j+1}| \leq C^{(3)}\left(|y_0 - z_0| + \sum_{i=0}^{j} |\varepsilon_i|\right)$$

D'où :

$$\max_{0 \leq j \leq n} |y_j - z_j| \leq C^{(3)}\left(|y_0 - z_0| + \sum_{i=0}^{n-1} |\varepsilon_i|\right)$$

On peut également utiliser le théorème suivant :
Une méthode est stable si les racines du polynôme $\rho(\lambda) = 0$ sont à l'intérieur du cercle unité ou elles sont simples si elles sont sur le cercle unité du plan complexe.
Le polynôme $\rho(\lambda)$ est défini par :

$$\rho(\lambda) = \sum_{i=1}^{k} \alpha_i \lambda^i$$

où les α_i sont les coefficients devant les y_{n-i}

Le polynôme est dans notre cas :

$$\lambda^2 - \lambda = \lambda(\lambda - 1)$$

Les racines sont donc 0 et 1, qui sont simples. La méthode est donc bien stable.

1.2.4 Consistance de la méthode

On utilise pour cela le théorème suivant : Si on vérifie les conditions suivantes :

$$\sum_{i=1}^{k} \alpha_i = 0 \text{ et } \sum_{i=1}^{k} i\alpha_i + \beta_i = 0$$

où α_i est définie comme précédemment et les β_i sont les coefficients devant les β_i, alors la méthode est consistante.

Dans notre exemple, on a la méthode :

$$y_{t+1} = y_t + \frac{h}{2}(3f_t - f_{t-1})$$

D'où :

$$\alpha_1 = 1,\ \alpha_2 = -1$$

et :

$$\beta_1 = \frac{3}{2},\ \beta_2 = -\frac{1}{2}$$

On a donc : $\sum \alpha_i = 0$ et :

$$\sum i\alpha_i + \beta_i = 1 - 2 + \frac{3}{2} - \frac{1}{2} = 0$$

La méthode est donc consistante.
Comme on l'a retrouvé précédemment, la méthode est bien convergente, car elle est consistante et stable.

2 Présentation du code C

Tout ce projet a été écrit en C. Nous présentons ici les différents codes écrits pour implémenter les différentes méthodes.
Nous avons également écrit un script bash ainsi qu'un script gnuplot pour automatiser le lancement des différents calculs. Ils seront présenter en Annexe.

2.1 Code pour la méthode d'Euler

```
void euler(int n,double h,double *V0,double *V1,double *F)
{
        int i;

        for(i=0;i<n;i++)
                V1[i]=V0[i]+h*F[i];
}
```

2.2 Code pour la méthode d'Adams-Bashforth

```
void adamsBashforth(int n,double h,double *V1,double *V,double *F0,double *F1)
{
        int i;

        for(i=0;i<n;i++)
                V[i]=V1[i]+h/2.*(3*F1[i]-F0[i]);
}
```

2.3 main

```
void main()
{
        int n;
        double h;
        double *V0;
        double *V1;
        double *V;
        double *F0;
        double *F1;
        double xmax;
        int nmax;
        int i,j;

        //Recuperation des informations generales sur la methode
        scanf("%d %lf %lf",&n,&xmax,&h);

        //Creation des tableaux dynamiques
        V0=(double *) malloc(n*sizeof(double));
        V1=(double *) malloc(n*sizeof(double));
        V=(double *) malloc(n*sizeof(double));
        F0=(double *) malloc(n*sizeof(double));
        F1=(double *) malloc(n*sizeof(double));

        //Lecture et ecriture des conditions initiales
        for(i=0;i<n;i++)
        {
```

```
                scanf("%lf",&V0[i]);
                printf("%lf ",V0[i]);
        }
        printf("\n");

        //Calcul du nombre d'iterations
        nmax=trunc((xmax-V0[0])/h);

        //Initialisation avec la methode d'Euler explicite
        f(V0,F0);
        euler(n,h,V0,V1,F0);
        for(i=0;i<n;i++)
                printf("%lf ",V1[i]);
        printf("\n");

        //Methode d'Adams-Bashforth
        f(V1,F1);
        for(i=2;i<(nmax+1);i++)
        {
                adamsBashforth(n,h,V1,V,F0,F1);
                for(j=0;j<n;j++)
                {
                        printf("%lf ",V[j]);
                        V0[j]=V1[j];
                        V1[j]=V[j];
                        f(V0,F0);
                        f(V1,F1);
                }
                printf("\n");
        }
}
```

3 Présentation des différents exemples développés

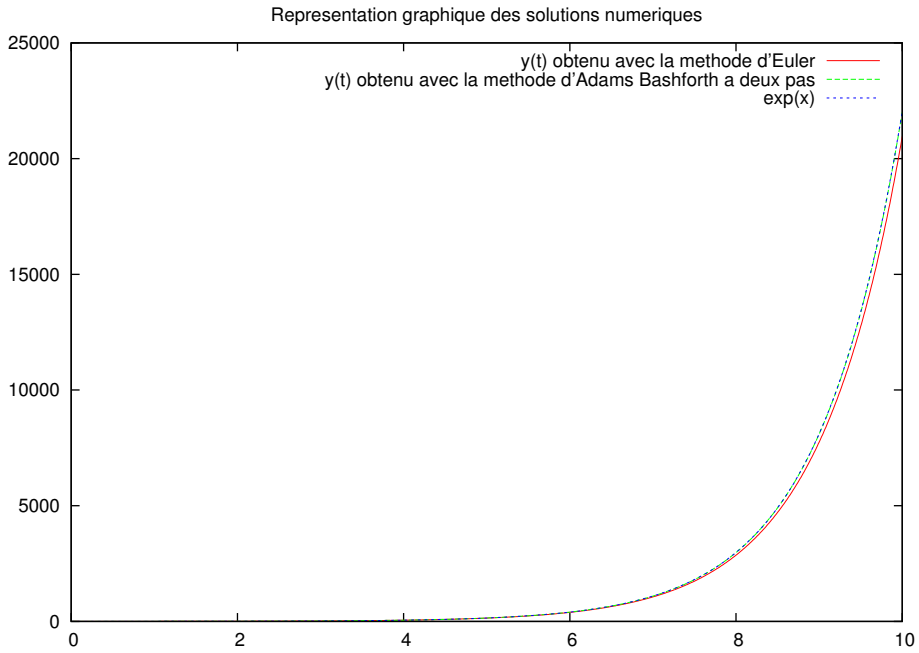
3.1 Exemple introductif

On commence par un exemple extrêmement simple :

$$\dot{y} = y$$

$$y(0) = 1$$

La solution est ici évidente : $y(x) = e^x$.
Cet exemple nous sert surtout à comparer la méthode d'Adams-Bashforth-2 avec celle d'Euler, et de voir la précision qu'on a pour un pas donné. Dans cet exemple, nous avons pris un pas de 0.01.

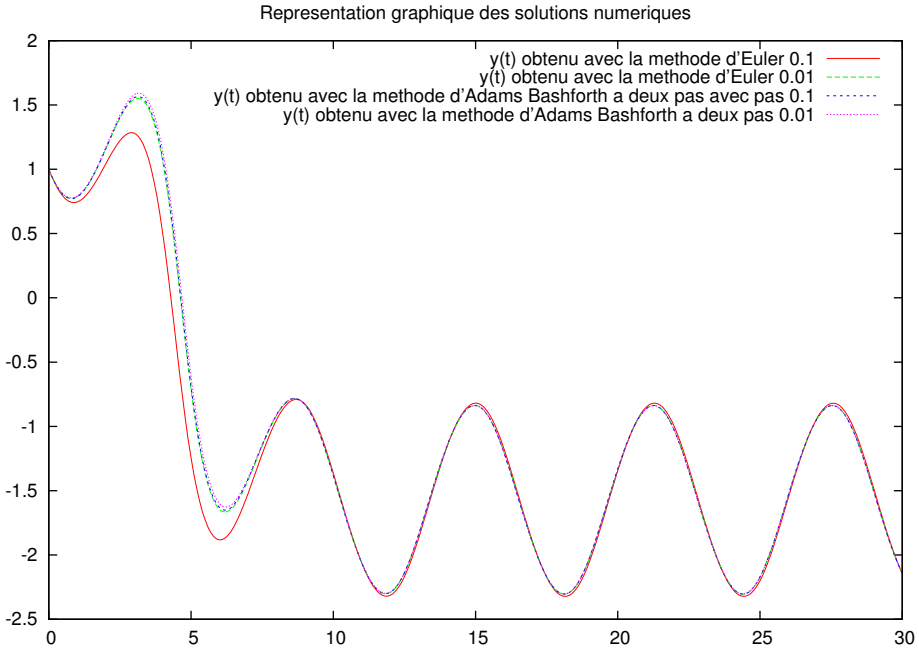


On voit très bien que pour un même pas, la méthode d'Adams-Bashforth-2 est déjà plus précise que celle d'Euler. La courbe obtenue par cette méthode s'approche énormément de la solution exacte, alors qu'on voit déjà que la méthode d'Euler s'en éloigne dans l'intervalle étudié.
Voyons maintenant dans d'autres exemples comment fonctionne cette méthode.

3.2 Premier exemple

Dans cet exemple, l'équation était la suivante :

$$y' = \sin(x) - \cos(y), y(0) = 1$$



On remarque tout d'abord que :

— On a un certain écart entre un pas de 0.1 et un pas de 0.01 pour la méthode d'Adams-Bashforth-2. Néanmoins, les solutions numériques obetnues semblent converger vers une même représentation graphique.

— Cet exemple illustre aussi bien le fait que la méthode d'Adams-Bashforth soit d'ordre 2 et la méthode d'Euler soit d'ordre 1. En effet, on observe un écart non négligeable entre la méthode d'Euler et de la méthode d'Adams-Bashforth pour un pas de 0.1. Notons toutefois que cet écart diminue avec le pas.

3.3 Deuxième exemple

L'équation était :

$$xdx - ydy = yx^2dy - xy^2dx$$
$$y(1) = 1$$

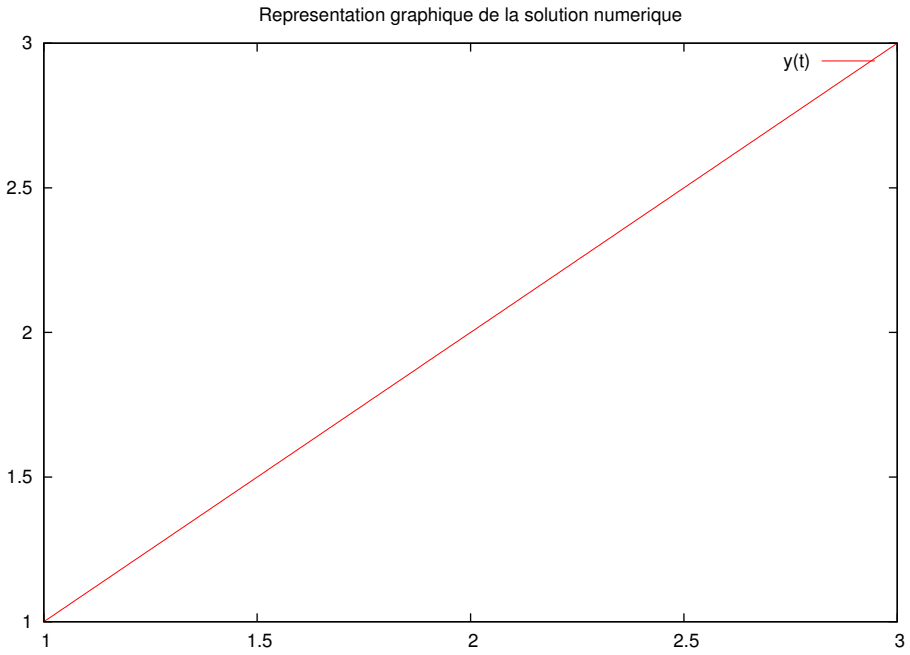
On transforme l'équation pour avoir :

$$\frac{dy}{dx} = \frac{x(1+y^2)}{y(1+x^2)}$$

Essayons de résoudre l'équation :

$$\begin{aligned} \frac{y}{1+y^2}dy &= \frac{x}{1+x^2}dx \\ \frac{1}{2}\ \ln(1+y^2) &= \frac{1}{2}\ln(1+x^2) + k \\ y^2 &= C(1+x^2) - 1 \end{aligned}$$

Or, on veut $y(1) = 1$, d'où $1 = 2C - 1 \Rightarrow C = 1$
On a donc $y^2 = x^2$



— La courbe ressemble beaucoup à ce qu'on doit obtenir (une droite affine). La convergence semble donc assurée.

— Avec des pas différents (0.1, 0.01 et 0.0001), la courbe était exactement la même.

3.4 Troisième exemple

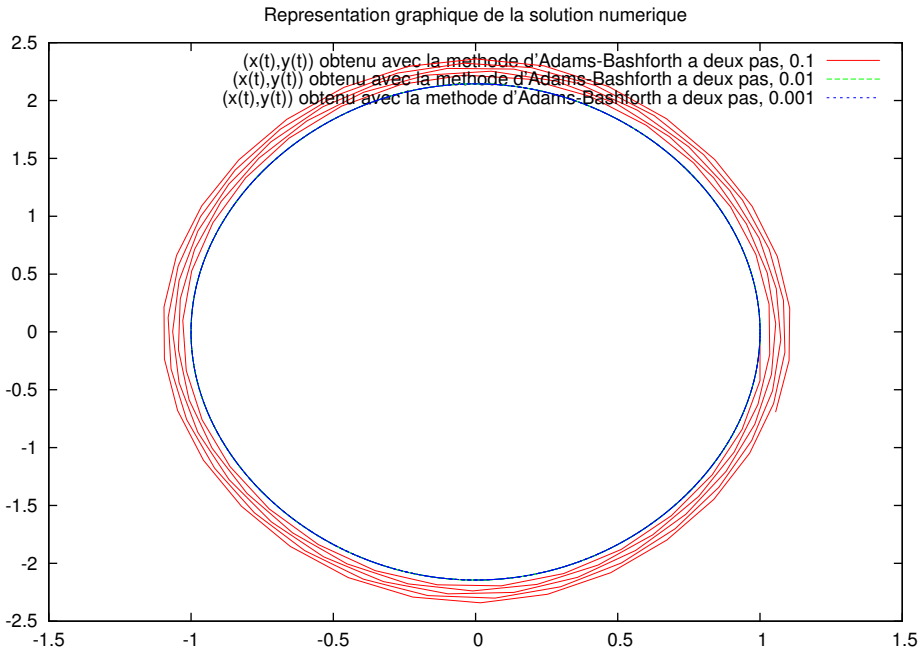
Nous avions un système d'équation de la forme :

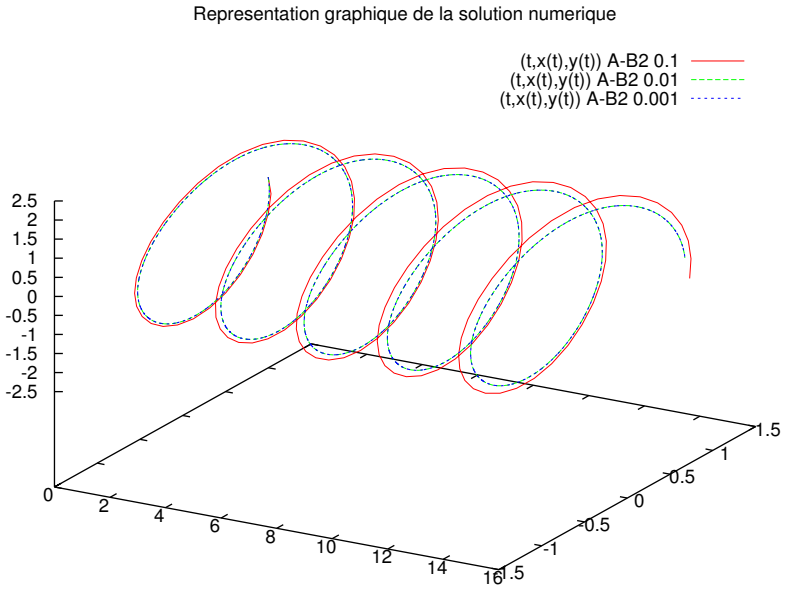
$$\begin{cases} y_1' &= y_2 \\ y_2' &= -5\sin(y_1) \\ y_1(0) &= \theta_0 \\ y_2(0) &= 0 \end{cases}$$

On a pris pour cela deux valeurs de θ_0 : 0.5 et 1.
Nous avons décidé sur cet exemple de comparer les résultats obtenus par la méthode d'Adams-Bashforth et par la méthode d'Euler.

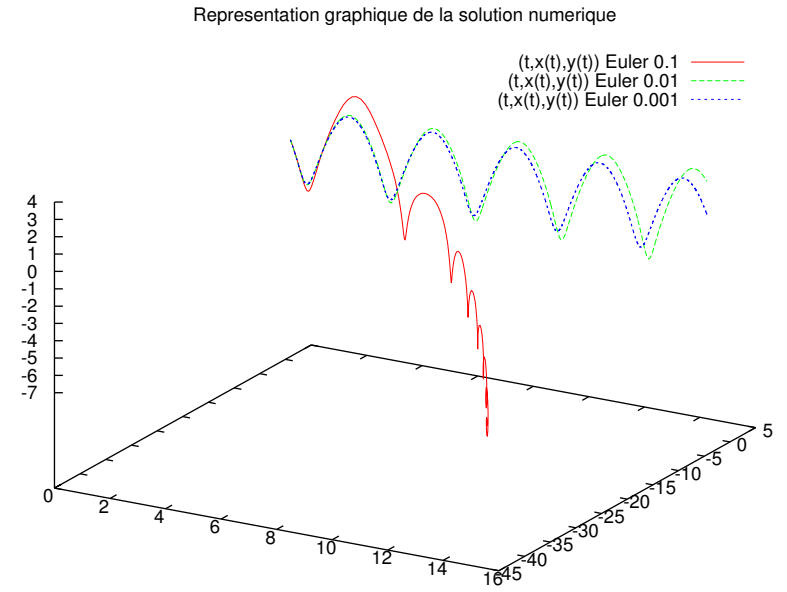
3.4.1 Résultats avec $\theta_0 = 1$

En commençant avec la méthode d'Adams-Bashforth-2, on voit que la métode semble converger, en réduisant le pas, vers une certaine solution. Avec un pas de 0.1, on semble quand même s'éloigner assez rapidement de la solution :

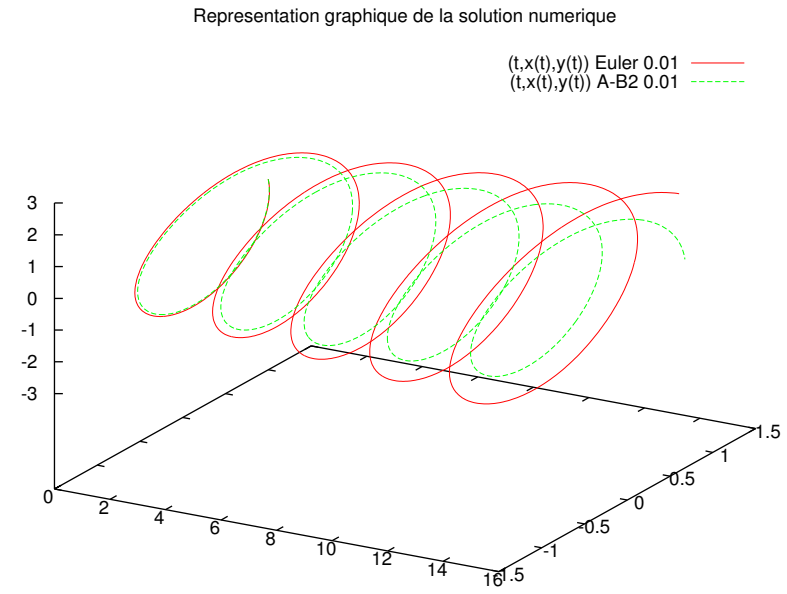


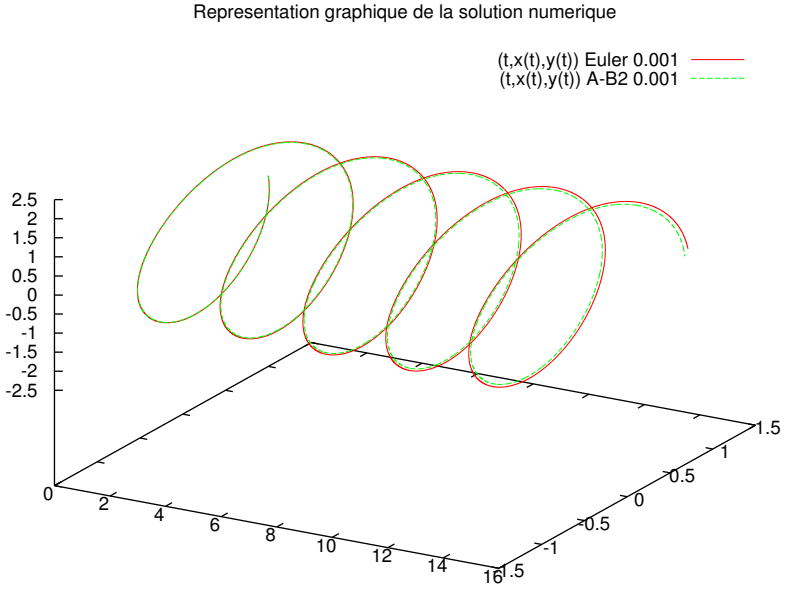


En comparaison, la méthode d'Euler semble bien moins puissante : elle ne donne pas une bonne solution approchée avec un pas de 0.1 :



Si on compare à présent directement la méthode d'Euler et la méthode d'Adams-Bashforth-2, la méthode d'Adams-Bashforth semble bien plus sûre que la méthode d'Euler, comme on peut le voir sur ces graphiques :

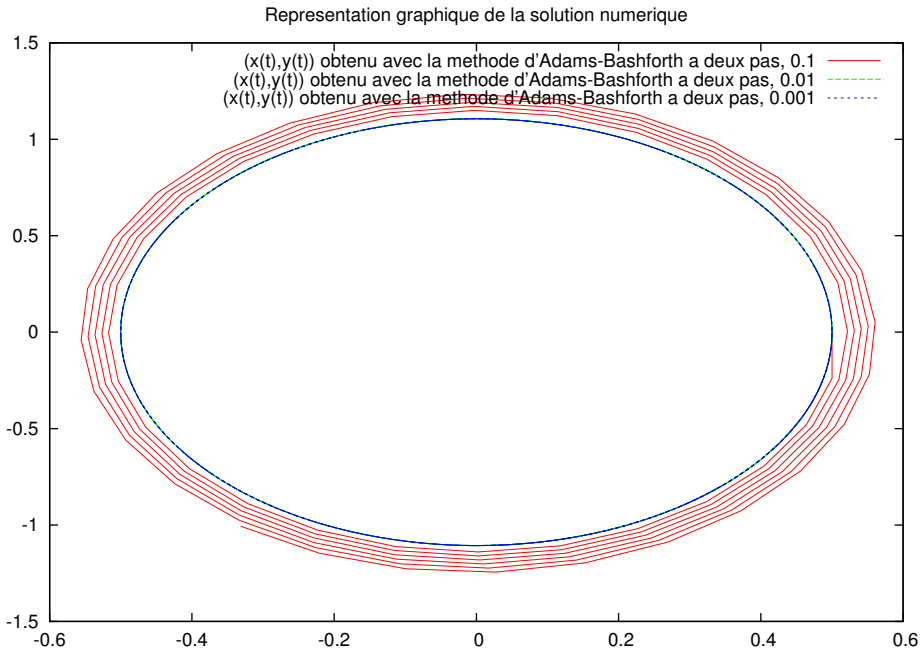


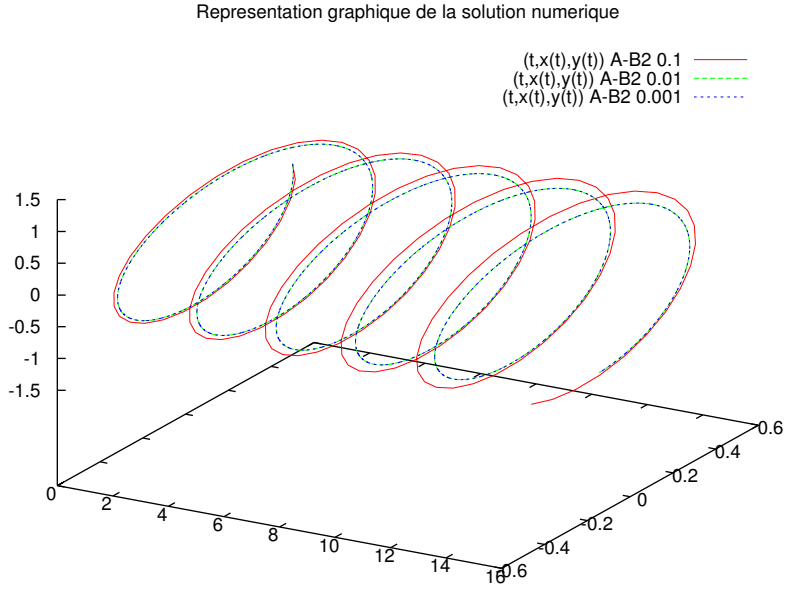


3.4.2 Résultats avec $\theta_0 = 0.5$

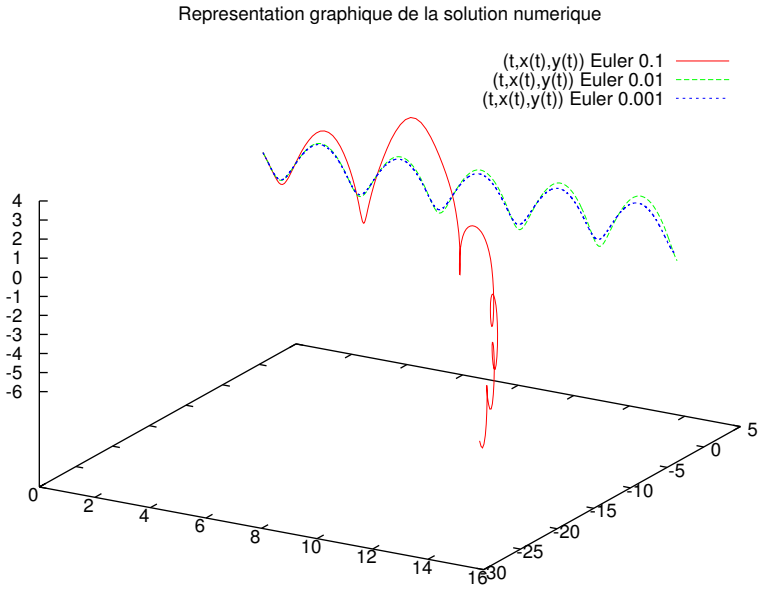
Les résultats ne changent pas vraiment ici :

— La convergence est toujours la même avec la méthode d'Adams-Bashforth-2 :

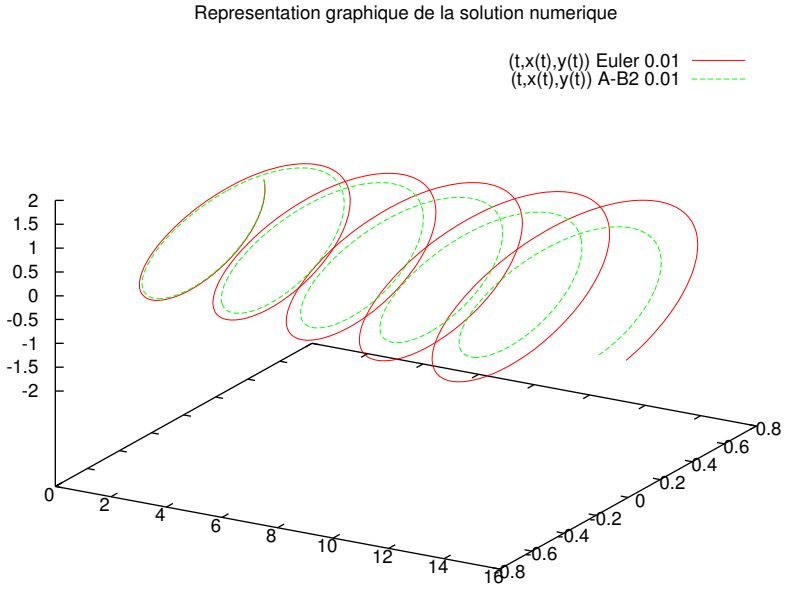


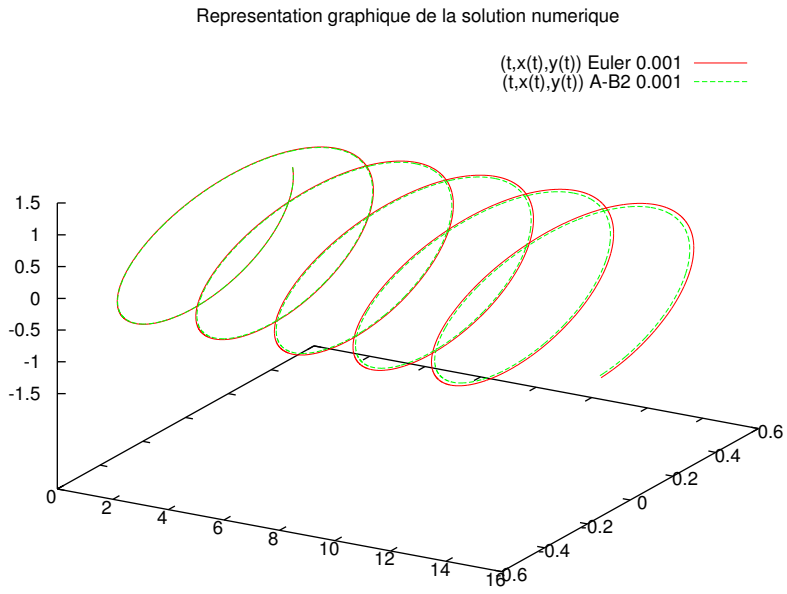


— La méthode d'Euler semble toujours moins puissante et n'est toujours pas digne de confiance avec un pas de 0.1 :



— Si on compare les deux méthodes, la méthode d'Adams est toujours plus sûre que la méthode d'Euler :





3.5 Quatrième exemple

Nous avions un système d'équation de la forme :

$$\begin{cases} y_1' &= -2y_1 - 998y_2 \\ y_2' &= -1000y_2 \\ y_1(0) &= 2 \\ y_2(0) &= 1 \end{cases}$$

On peut facilement calculer la solution de ce système. Directement, on voit que :

$$y_2(t) = e^{-1000t}$$

Il nous reste donc à résoudre :

$$y_1'(t) = -2y_1(t) - 998e^{-1000t}$$

Solution de l'équation homogène :

$$y_{1_H}(t) = Ce^{-2t}$$

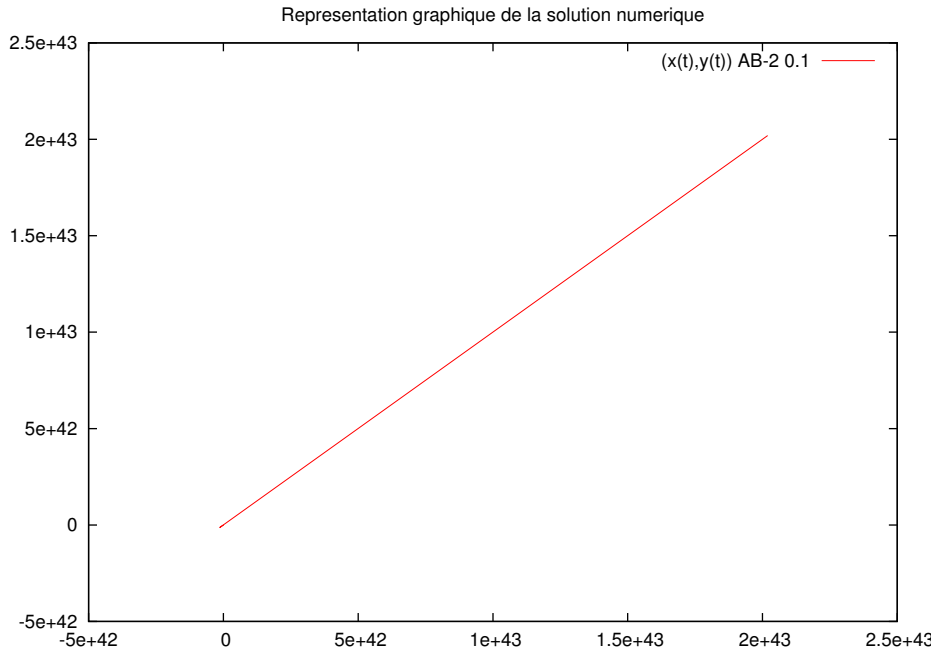
Solution particulière à l'équation :

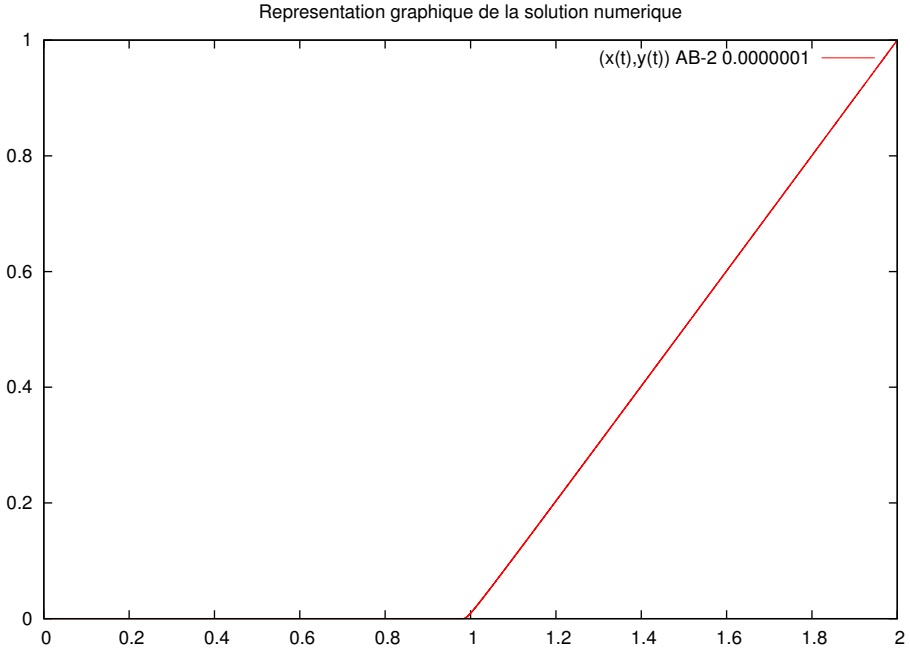
$$y_{1_P}(t) = e^{-1000t}$$

D'où :

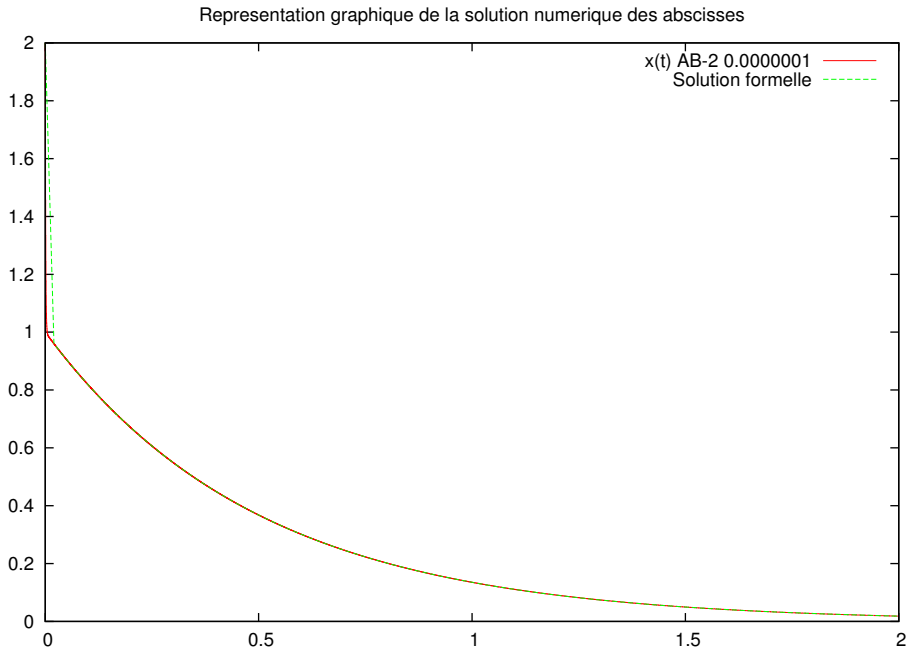
$$y_1(t) = e^{-2t} + e^{-1000t} = e^{-2t} + y_2(t)$$

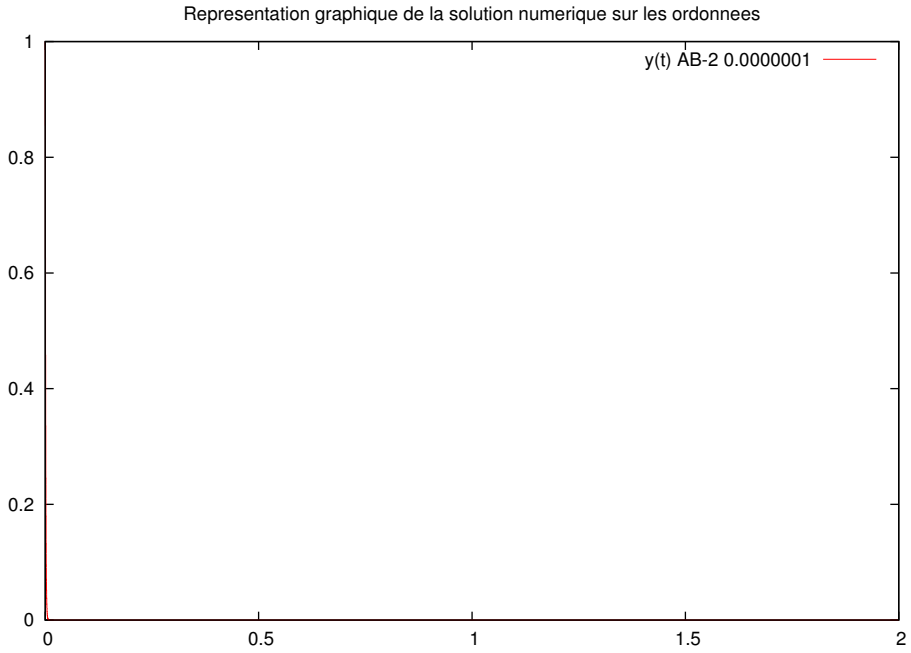
Lors de nos tests, on a directement remarqué un problème dû à la stabilité de la méthode. En effet, avec un pas trop grand, nos solutions explosaient. Nous avons dû prendre un pas de l'ordre du dix millième pour avoir une solution correcte. Voici deux tracés des solutions obtenues : le premier avec un pas de 0.1, le deuxième avec un pas de 0.0000001 :





— La méthode ne paraît donc pas trop adaptée pour cette équation.
— Quand on obtient des points qui semblent corrects, la courbe n'est pas très exploitable. Cela est dû au fait que les abscisses diminuent beaucoup plus lentement que les ordonnées (vu la solution de l'équation), comme on peut le voir sur les deux graphiques suivants :

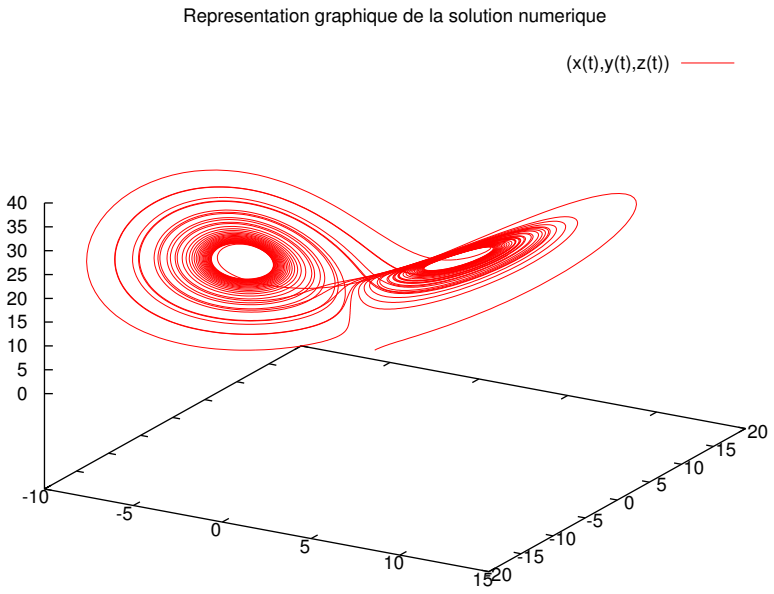


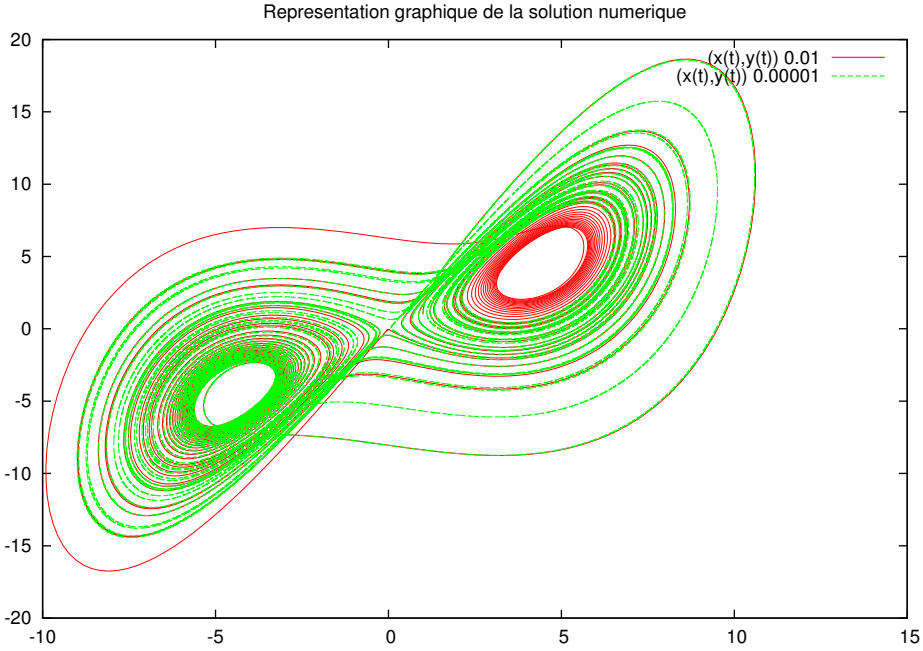


3.6 Cinquième exemple : l'attracteur de Lorenz

Nous avions un système d'équation de la forme :

$$\begin{cases} x' &= -3(x-y) \\ y' &= -xz + 21x - y \\ z' &= xy - z \end{cases}$$



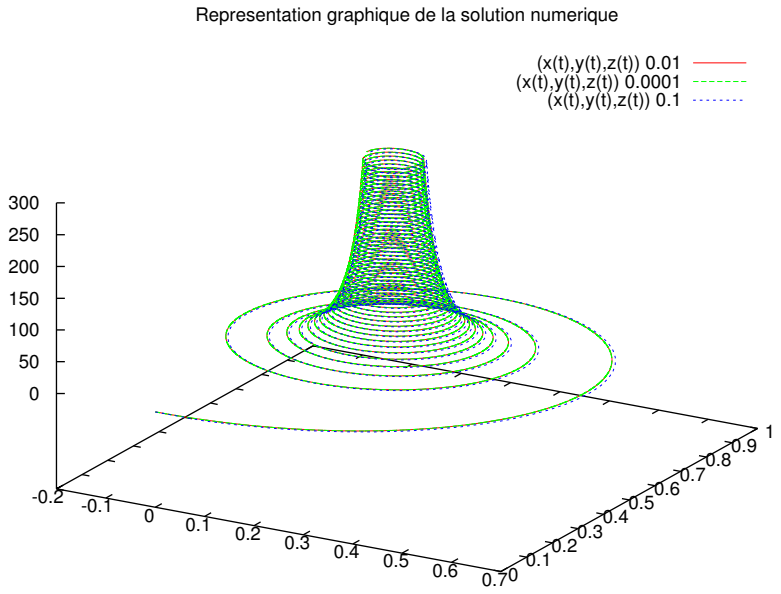


— On retrouve bien l'attracteur de Lorenz.
— Un premier intérêt est de voir que la méthode d'Adams-Bashforth-2 marche encore très bien pour des systèmes non linéaires.
— Le fait qu'on change le pas ne fait pas passer la courbe par les mêmes régions. Le côté chaotique du système explique ces variations. On peut donc soupçonner qu'il existe des méthodes plus adaptées pour les systèmes chaotiques.

3.7 Sixième exemple

Nous avions un système d'équation de la forme :

$$\begin{cases} x' &= \frac{1}{\sqrt{z}+1}\cos(z) \\ y' &= \frac{1}{\sqrt{z}+1}\sin(z) \\ z' &= 1 \end{cases}$$



— L'exemple n'a pas beaucoup d'autre intérêt que de montrer une jolie courbe !
— On voit qu'ici, le pas n'influance pas grandement la précision sur la courbe. L'équation est en même temps relativement simple : les dérivées de x et de y ne dépendent que de z à chaque fois, dont sa dérivée vaut 1.

Conclusion

Ce projet nous aura permis d'implémenter une méthode explicite plus précise que la méthode d'Euler qui est d'ordre 1. On remarque cependant qu'il reste des cas critique (comme l'exemple quatre) ou que pour certains systèmes chaotiques comme l'attracteur de Lorenz, d'autres méthodes peuvent être plus adaptées.
Les résultats obtenus restent tout de même très satisfaisants et auront fini de nous convaincre de l'utilité de cette méthode du fait de son implémentation relativement simple.

A Annexes

A.1 Codes supplémentaires

A.1.1 Script bash

```
        #!/bin/bash


echo "Choisir un exemple."
select choix in `ls | grep "exemple*"`
do
        break
done
cd ${choix}

echo " "
ls|grep "description*" > bidule
dsc=`cat bidule`
rm bidule

dsc=${dsc##*n}
if [ "$dsc" = "1D" ]
then
        cat description1D
elif [ "$dsc" = "2D" ]
then
        cat description2D
else
        cat description3D
fi

echo " "
echo "Que souhaitez-vous faire ?"
echo "1) Modifier le fichier de donnees"
echo "2) Modifier le code source"
echo "3) Lancer le programme"
echo "Autre : quitter le script"
read choix

if [ $choix -eq 1 ]
then
        vim dAdams
        echo " "
elif [ $choix -eq 2 ]
then
        vim adamsBashforth2.c
        echo " "
elif [ $choix -eq 3 ]
then
        gcc adamsBashforth2.c -o adamsBashforth2 -lm
        ./adamsBashforth2 < dAdams > res

        if [ "$dsc" = "1D" ]
        then
```

```
                gnuplot ../1D.g
        elif [ "$dsc" = "2D" ]
        then
                gnuplot ../2D.g
        else
                gnuplot ../3D.g
        fi
        evince image.eps
fi
```

A.1.2 Script gnuplot

1D.g

```
set terminal postscript eps enhanced color
set output "image.eps"
set title "Representation graphique de la solution numerique"
plot "res" u 1:2 w l title "y(t)"
```

2D.g

```
set terminal postscript eps enhanced color
set output "image.eps"
set title "Representation graphique de la solution numerique"
plot "res" u 2:3 w l title "(x(t),y(t))"
```

3D.g

```
set terminal postscript eps enhanced color
set output "image.eps"
set title "Representation graphique de la solution numerique"
splot "res" u 2:3:4 w l title "(x(t),y(t),z(t))"
```