Analyse numérique des équations différentielles

Grégory Vial

10 mars 2003

Le but de ces quelques pages est de montrer comment programmer facilement et efficacement les schémas classiques de résolution numérique des équations différentielles à l'aide de matlab. On met en évidence leurs ordres de convergence respectifs sur un exemple simple. Cependant, un ordre de consistance élevé n'est pas toujours la garantie d'une bonne approximation, notamment dans le cas de problèmes raides ou de systèmes hamiltonniens.

Le problème de Cauchy

On s'intéresse à l'équation différentielle avec condition initiale suivante :

$$(1) \qquad \left\{ \begin{array}{rcl} y'(t) & = & f(t,y(t)), \quad 0 \leq t \leq T, \\ y(0) & = & y_0, \end{array} \right.$$

où $f : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ est une fonction régulière et $y_0 \in \mathbb{R}^d$.

Programmation des méthodes classiques

Pour l'approximation numérique, on note h le pas de la subdivision uniforme $(t_n)_{n=0}^N$ de l'intervalle $[0,T]$: $t_n = nh$ et y_n est une approximation de $y(t_n)$ pour $0 \leq n \leq N$.

On a choisi de présenter ici les cinq méthodes ci-dessous.

Euler explicite
$$y_{n+1} = y_n + hf(t_n, y_n)$$

Heun
$$y_{n+1} = y_n + \frac{h}{2}\Big[f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n))\Big]$$

Runge-Kutta (RK4)
$$\left| \begin{array}{rcl} k_1^n & = & f(t_n, y_n) \\ k_2^n & = & f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1^n) \\ k_3^n & = & f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2^n) \\ k_4^n & = & f(t_{n+1}, y_n + hk_3^n) \\ y_{n+1} & = & y_n + \frac{h}{6}\Big(k_1^n + 2k_2^n + 2k_3^n + k_4^n\Big) \end{array} \right.$$

Euler implicite
$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

Cranck-Nicolson
$$\left| \begin{array}{rcl} y_{n+\frac{1}{2}} & = & y_n + \frac{h}{2} f\left(t_n + \frac{h}{2}, y_{n+\frac{1}{2}}\right) \\ y_{n+1} & = & y_n + hf\left(t_n + \frac{h}{2}, y_{n+\frac{1}{2}}\right) \end{array} \right.$$

Les fonctions matlab suivantes définissent ces méthodes :

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Méthode d'Euler explicite
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=eulE(y0,h,T);
y=[y0];yy=y0;
N=floor(T/h);t=0;
for i=1:N,
    yy=yy+h*F(t,yy);
    y=[y,yy];t=t+h;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Méthode de Heun
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=heun(y0,h,T);
y=[y0];yy=y0;
N=floor(T/h);t=0;
for i=1:N,
    yy=yy+h/2*(F(t,yy)
      +F(t+h,yy+h*F(t,yy)));;
    y=[y,yy];t=t+h;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Méthode RK4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=RK4(y0,h,T);
y=[y0];yy=y0;
N=floor(T/h);t=0;
for i=1:N,
    k1=f(t,yy);
k2=f(t+h/2,yy+h/2*k1);
k3=f(t+h/2,yy+h/2*k2);
k4=f(t+h,yy+h*k3);
    yy=yy+h/6*(k1+2*k2+2*k3+k4);
    y=[y,yy];t=t+h;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Méthode d'Euler implicite
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=eulI(y0,h,T);
y=[y0];yy=y0;
N=floor(T/h);t=0;
for i=1:N,
    yy=newton(t,h,yy);
    y=[y,yy];t=t+h;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Méthode de Cranck-Nicolson
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=CN(y0,h,T);
y=[y0];yy=y0;
N=floor(T/h);t=0;
for i=1:N,
    y2=newton(t+h/2,h/2,yy);
    yy=y2+h/2*F(t+h/2,y2);
    y=[y,yy];t=t+h;
end;
```

Les trois premières méthodes sont explicites : la détermination de y_{n+1} à partir de y_n est un simple calcul. Les deux dernières, en revanche, sont implicites : il est nécessaire de résoudre une équation à chaque itération. Si f est non-linéaire, cette équation est non-linéaire ; c'est pourquoi on a programmé une méthode de Newton.

```
function x=newton(t,h,yn)
x=yn+h*F(t,yn);
err=1;eps=1e-9;
while err>eps,
  b=-Phi(t,x,h,yn);
  A=DPhi(t,x,h);
  y=A\b;
  x=x+y;
  err=norm(y);
end;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Fonctions annexes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=Phi(t,x,h,yn);
y=x-h*F(t,x)-yn;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y=DPhi(t,x,h)
y=eye(length(x))-h*DF(t,x);
```

Précision des méthodes : ordre de convergence

Pour chacune des cinq méthodes présentées au paragraphe précédent, une étude de l'erreur de consistance permet de montrer l'estimation suivante (on suppose $y \in \mathscr{C}^{p+1}$).

$$\max_{0\leq n\leq N} |y_n - y(t_n)| \leq Ch^p \sup_{0\leq t\leq T} |y^{p+1}(t)|.$$

Méthode	p
Euler explicite	1
Heun	2
RK4	4
Euler implicite	1
Cranck-Nicolson	2

TAB. 1 – Ordres de convergence des méthodes.

Le nombre p est appelé *ordre de convergence* ; le tableau ci-dessus précise sa valeur pour les méthodes décrites plus haut.

Pour plus de détails pour l'analyse des schémas, on pourra se reporter à [1].

Afin de mettre en évidence la différence de précision entre les différentes méthodes, on les a appliquées sur l'exemple suivant :

$$y' = -y \quad \text{avec} \quad y(0) = 2.$$

Le script suivant permet de comparer les vitesses de convergence de la quantité

$$\max_{0 \leq n \leq N} |y_n - y(t_n)|$$

en fonction de h.

```
function y=DF(t,x);
y=-1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=F(t,x);
y=-x;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T=5;
y0=2;
nn=10:10:100;
err=zeros(5,length(nn));
k=0;
for n=nn
  k=k+1;
  h=1/n;
  t=0:h:T;
  yex=y0*exp(-t);
  err(1,k)=norm(eulE(y0,h,T)-yex
                          ,'inf');
  err(2,k)=norm(heun(y0,h,T)-yex
                          ,'inf');
  err(3,k)=norm(RK4(y0,h,T)-yex
                          ,'inf');
  err(4,k)=norm(eulI(y0,h,T)-yex
                          ,'inf');
  err(5,k)=norm(CN(y0,h,T)-yex
                          ,'inf');
end

N=[];
for i=1:5
  N=[N;nn];
  pente=polyfit(log(nn)
                ,log(err(i,:)),1);
  pente(1)
end
plot(log(N'),log(err'))
grid on
legend('Euler explicite','Heun','RK4'
,'Euler implicite','Cranck Nicolson')
     xlabel('Entier n tel que h=1/n')
ylabel('log|erreur|')
title('Erreurs en coordonnées
                  logarithmiques')
```

La figure 1 représente les courbes obtenues pour les cinq schémas avec $T = 5$ et $h = 0.1$, ainsi que la solution exacte. Afin d'observer les ordres de convergence des méthodes, on a tracé – en coordonnées logarithmiques – la norme de $y_h - y$ en fonction de h (y_h désigne la solution approchée pour la valeur h du pas).

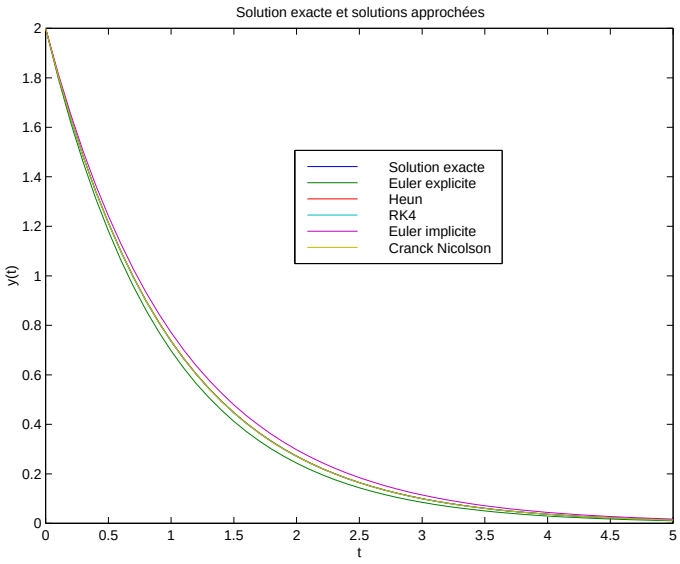

FIG. 1 – Approximation de la solution de $y' = -y$.

Le graphe de la figure 2 permet d'estimer les ordres de convergence, qui coïncident avec les ordres théoriques donnés dans le tableau 1. Précisément la commande `polyfit`, qui effectue une régression linéaire, donne les ordres suivants :

Ordre théorique	1	2	4	1	2
Ordre estimé	1.015	2.027	4.030	0.985	2.000

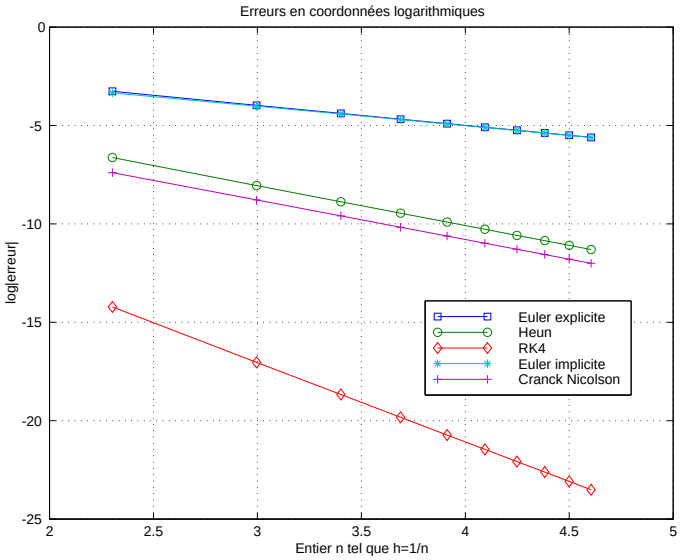

FIG. 2 – Ordres de convergence des différentes méthodes.

Nécessité de méthodes implicites : problèmes raides

Le paragraphe précédent a montré que les méthodes utilisées n'étaient plus ou moins précises. Cependant, il ne faut pas croire que la méthode RK4 est toujours la plus performante des cinq méthodes décrites.

Considérons en effet le problème de Cauchy suivant :

$$y' = -\lambda y + 1 + \lambda t \quad \text{avec} \quad y(0) = 2, \tag{2}$$

dont la solution exacte est donnée par

$$y(t) = t + 2e^{-\lambda t}. \tag{3}$$

Le tableau ci-dessous consigne les erreurs entre la solution exacte et la solution approchée pour $\lambda = 500$ et $h = 0.01$.

Méthode	$\max_{0\le n\le N} \lvert y_n - y(t_n)\rvert$	
pas	$h = 0.01$	$h = 0.001$
Euler explicite	2.1990e+12	0.2358
Heun	7.7519e+18	0.0455
RK4	1.0983e+23	5.8281e-04
Euler implicite	0.3115	0.1526
Cranck-Nicolson	0.8653	0.0159

TAB. 2 – Performance des méthodes pour le système (2) avec $\lambda = 500$ et $h = 0.01$ et $h = 0.001$.

On observe ici que seules les méthodes implicites fournissent une approximation raisonnable de la solution pour le pas $h = 0.01$. L'exemple (2) est extrait de [2] ; on peut étudier le comportement des méthodes d'Euler sur cette équation. Notons (y_n^{e}) et (y_n^{i}) les suites respectivement construites par les méthodes d'Euler explicite et implicite. Par définition,

$$\begin{aligned} y_{n+1}^{\mathrm{e}} &= y_n^{\mathrm{e}} + h(\lambda(t_n - y_n^{\mathrm{e}}) + 1) \; ; \\ y_{n+1}^{\mathrm{i}} &= (1 + \lambda h)^{-1} \left(y_n^{\mathrm{i}} + \lambda h t_{n+1} + h\right). \end{aligned}$$

Comme λ est grand, la solution (3) est rapidement proche de $t \mapsto t$; on cherche à savoir si cette propriété est conservée par les schémas. Il est facile de voir que, pour h fixé, la quantité y_{n+1}^{i} est proche de t_{n+1} pour les grandes valeurs de λ. En revanche, si on suppose que y_n^{e} est proche de t_n, alors y_n^{e} n'est proche de t_{n+1} que si $h\lambda \ll 1$. Cette condition impose un choix du pas h très petit si l'on veut approcher correctement la solution de (2), une telle équation est appelée *raide*. Il est difficile de donner une définition de la *raideur* ; on retiendra que la solution d'un problème raide met en jeu des phénomènes : l'un est lent et régulier – terme t dans (3) – le second est rapide et a un effet à temps court – terme $2e^{-\lambda t}$.

Les problèmes raides interviennent souvent dans les applications où différentes échelles coexistent (réactions chimiques avec des vitesses caractéristiques très différentes, voir [4] par exemple). Les méthodes explicites doivent être utilisées avec un pas de temps très petit pour être efficaces (ce que montrent les résultats du tableau 2 pour $h = 0.001$), c'est pourquoi on leur préfère les méthodes implicites pour la résolution numérique des problèmes raides.

Méthodes symplectiques : systèmes hamiltonniens

On a vu dans le paragraphe précédent qu'une méthode implicite d'ordre 1 pouvait être plus performante qu'une méthode explicite d'ordre 4 dans le cas d'un problème raide. Intéressons-nous maintenant au problème linéaire suivant :

$$(4) \qquad \left\{ \begin{array}{rcl} x' & = & -y \\ y' & = & x \end{array} \right. \quad \text{avec} \quad x(0) = 1 \quad \text{et} \quad y(0) = 0.$$

La solution est donnée par $[x(t) = \cos t, y(t) = \sin t]$ et la trajectoire dans la plan de phase est le cercle unité. La quantité

$$H(t) = x^2(t) + y^2(t)$$

est constante ; on la nomme *hamiltonnien* du système (4).

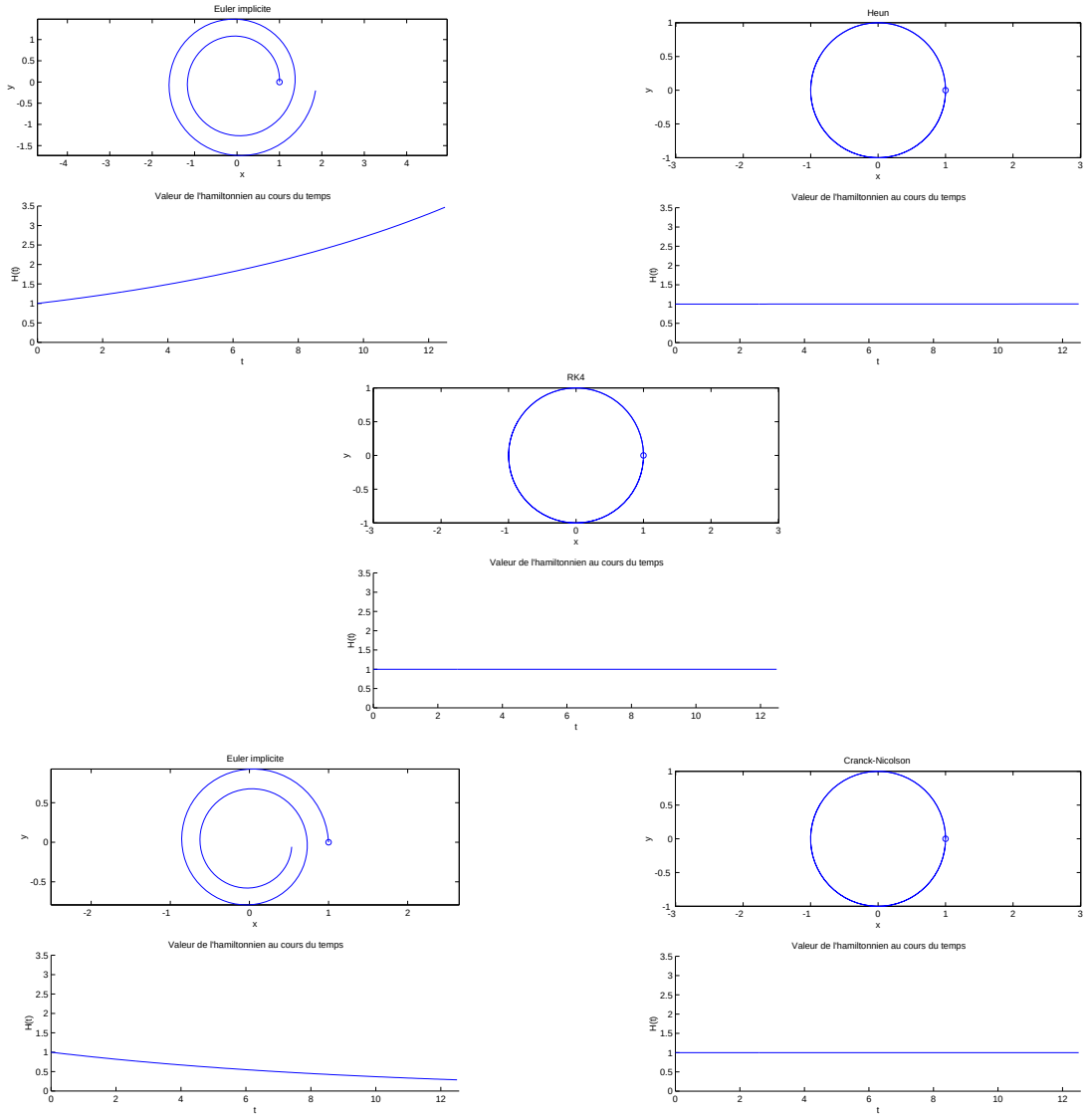

FIG. 3 – Trajectoire et hamiltonnien pour les cinq méthodes.

On observe sur la figure 3 les solutions obtenues avec les différentes méthodes pour $h = 0.1$ et $T = 4\pi$. On a représenté la trajectoire dans le plan de phase ainsi que l'évolution de l'hamiltonnien $H(t)$ au cours du temps. Les méthodes d'Euler (explicite et implicite) ne conservent pas l'hamiltonnien, alors que les trois autres méthodes le gardent presque constant. Précisément, on peut vérifier que

$$\begin{array}{lrcl}
\text{Euler explicite} & x_{n+1}^2 + y_{n+1}^2 & = & (1+h^2)(x_n^2 + y_n^2)\ ; \\
\text{Heun} & x_{n+1}^2 + y_{n+1} & = & \left(1 + \frac{h^4}{4}\right)(x_n^2 + y_n^2)\ ; \\
\text{RK4} & x_{n+1}^2 + y_{n+1} & = & \left(1 - \frac{h^6}{72} + \frac{h^8}{576}\right)(x_n^2 + y_n^2)\ ; \\
\text{Euler implicite} & x_{n+1}^2 + y_{n+1}^2 & = & \left(1+h^2\right)^{-1}(x_n^2 + y_n^2)\ ; \\
\text{Cranck-Nicolson} & x_{n+1}^2 + y_{n+1}^2 & = & x_n^2 + y_n^2\ ;
\end{array}$$

ce qui explique les comportements obtenus sur la figure 3. Remarquons que seule la méthode de Cranck-Nicolson conserve exactement l'hamiltonnien : elle est dite *symplectique*; les méthodes de Heun et RK4 sont dites *asymptotiquement symplectiques*.

Pour plus de détails sur les méthodes symplectiques, voir [3]. On retiendra que dans le cas de systèmes du type (4), où une quantité est conservée au cours du temps, il est crucial que le schéma utilisé conserve – au moins approximativement – cette quantité.

On trouve un exemple typique d'application conduisant à un système hamiltonnien dans le modèle proie-prédateur de Lotka-Volterra. Les méthodes numériques ont des comportement voisins dans le cas linéaire étudié ici (il est à noter que l'hamiltonnien n'est plus constant, même pour la méthode de Cranck-Nicolson, pour laquelle il oscille autour de sa valeur théorique).

Références

[1] M. Crouzeix, A. L. Mignot. *Analyse numérique des équations différentielles*. Collection mathématiques appliquées pour la maîtrise. Masson, Paris 1984.

[2] K. Dekker, J.-G. Verwer. *Stability of Runge-Kutta methods for stiff nonlinear differential equations*. CWI Monographs. North-Holland, Amsterdam 1984.

[3] E. Hairer, S. P. Nørsett, G. Wanner. *Solving ordinary differential equations. I.* Springer-Verlag, Berlin, second edition 1993. Nonstiff problems.

[4] E. Hairer, G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition 1996. Stiff and differential-algebraic problems.