Comment calcule-t-on numériquement la solution d'une équation ?

Petits pièges de la simulation numérique (I)

Texte de E. Grenier.

Prenons une équation qui décrit un phénomène physique, chimique ou biologique. Deux questions se posent au mathématicien :

- existe-t-il toujours une solution ? est-elle unique ? quelles sont ses propriétés ? (est-elle positive ? où sont ses zéros ? est-elle croissante ?).
- comment peut-on calculer effectivement cette solution ? peut-on le faire rapidement ?

La première question est théorique et abstraite, et nul doute que ce soit une question de "mathématiques". La seconde ressemblerait plus à une question d'informatique, ou à une question d'ingénieur. Qu'en est-il ? Suffit-il de programmer l'équation sur sa calculatrice pour avoir la solution ? Est-ce évident ? direct ? ou faut-il tout d'abord réflechir ? L'accélération du calcul n'est il qu'une question de Megahertz ? de rapidité du processeur de calcul ?

Si il suffit de rentrer une équation dans une calculette ou un superordinateur, la seconde question n'est pas une question de mathématiques, et il n'y a rien à faire.

Si par contre il faut un travail préliminaire abstrait pour trouver un schéma efficace, du coup la théorie des algorithmes numériques redevient un sujet à part entière des mathématiques. L'informatique (ou plus exactement la programmation) se charge de la mise en place effective (mise en place sur des gros ordinateurs, parallélisation,...), et l'ingénierie fournit les cas concrets à calculer.

Le but de cette note est de tenter de répondre à ces questions sur un cas simple ....

1 Transport ...

Commençons par une équation aux dérivées partielles très simple. Soit $u(t,x)$ une fonction de deux variables réelles t et x. Une équation aux dérivées partielles (edp en abrégé) est une équation reliant des dérivées partielles de u. Une des équations les plus simples est

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0. \tag{1}$$

Elle relie la dérivée partielle par rapport au temps avec la dérivée partielle par rapport à x.

La solution de (1) est particulièrement simple. Toute fonction de la forme

$$u(t,x) = \phi(x-t) \tag{2}$$

où ϕ est une fonction réelle dérivable arbitraire est solution !. Par exemple si on connaît u à $t=0$, c'est-à-dire si on connaˆ$u(0,x)$ pour tout $x \in R$, alors on connaˆ$u(t,x)$ pour tout $t \in R$ et pour tout $x \in R$ simplement par

$$u(t,x) = u(0,x-t). \tag{3}$$

Comment peut-on calculer la solution de (1) si on se donne u à $t=0$. Elémentaire allez vous dire ! Il suffit d'utiliser (3) !.

Oui, mais imaginons qu'on ne connaisse pas (3) ? Que fait-on ? Peut-on calculer la solution numériquement ?

Question sans intérêt allez vous dire, puisqu'on connaˆparfaitement la solution ...

Pas tout à fait, car bien sûr les équations que l'on cherche effectivement à résoudre sont plus compliquées, et pour elles on ne connaˆpas de solution explicite. L'équation présentée ici est simplement un modèle simplifié pour présenter les idées, sans rentrer dans des difficultés techniques. Une équation effectivement utilisée est par exemple l'équation de Hopf

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} = 0. \tag{4}$$

Là, il existe une formule "semi-explicite", c'est-à-dire il existe effectivement une formule, mais elle est à peu près inexploitable tant elle est compliquée.

Un système d'équations plus réaliste est un système issu de la dynamique des gaz (fluides compressibles barotropes)

$$\frac{\partial \rho}{\partial x} + \frac{\partial(\rho u)}{\partial x} = 0, \tag{5}$$

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial p(\rho)}{\partial x} = 0, \tag{6}$$

où p est une fonction dérivable de R vers R.

Là plus de solution explicite du tout ... et le système est physiquement intéressant (des variantes sont effectivement utilisées dans des codes industriels). Il semble clair qu'avant de résoudre (5,6) il vaut mieux commencer par résoudre (1) ! C'est ce que nous allons faire maintenant.

2 Un premier essai ....

Bien sûr un ordinateur ne peut contenir qu'un nombre fini de données. On ne pourra donc pas calculer $u(t,x)$ pour tous les t et tous les x mais seulement pour un nombre fini d'entre eux. Ce n'est pas grave, les autres t et x peuvent être recalculés par des techniques dites d'interpolation.

Pour se ramener à un nombre fini de x, on "discrétise" x, c'est-à-dire on divise un intervalle de calcul $0 \leq x \leq A$ en petites "boîtes" de longueur $h > 0$. Supposons que l'on crée N boîtes. On a

$$h = \frac{A}{N}.$$

Les boîtes B_j sont définies par

$$B_j = [jh, (j+1)h[.$$

Plus N est grand, plus les boîtes sont petites et plus le calcul sera précis ... mais plus il sera long.

De même façon on discrétise le temps. Soit k le pas de temps. On coupe le temps en "tranches" de longueur k. Soit

$$t_n = kn.$$

Une façon simple de calculer u est d'approximer u par une constante, notée u_i^n sur chaque "maille":

$$u(t,x) \sim u_i^n \qquad \text{pour} \qquad x \in B_j, \qquad t_n \leq t < t_{n+1}.$$

Il reste à transcrire l'équation (1) en des relations entre les u_i^n, c'est-à-dire à exprimer $\partial u/\partial t$ et $\partial u/\partial x$ en fonction des u_i^n.

Le plus simple est de remplacer la dérivée par rapport au temps par une dérivée discrète

$$\frac{\partial u}{\partial t} \sim \frac{u_i^{n+1} - u_i^n}{k}$$

et de façon similaire

$$\frac{\partial u}{\partial x} \sim \frac{u_{i+1}^n - u_{i-1}^n}{2h}.$$

A noter qu'on peut aussi approcher la dérivée temporelle par $(u_i^{n+1} - u_i^{n-1})/2h$, mais cela ne change rien au résultat.

L'équation (1) devient alors

$$\frac{u_i^{n+1} - u_i^n}{k} + \frac{u_{i+1}^n - u_{i-1}^n}{2h} = 0. \tag{7}$$

Tout ceci est bien naturel allez vous dire !

Si on se donne u_i^0 alors (1) permet de calculer de proche en proche tous les u_i^n... sauf pour $i = 0$ ou $i = A/h$. Pour $i = 0$ en effet, (7) fait intervenir u_{-1}^n qui n'est pas définit !

Passons sur ce détail. Une des façons de faire est de considérer une solution périodique en espace, de période A. Cela revient à dire que $u_{-1}^n = u_{A/h-1}^n$ et $u_{A/h+1}^n = u_2^n$. Une autre méthode est d'imposer des "conditions aux limites" en $x = 0$ et $x = A$. C'est un peu plus délicat et ne sera pas discuté ici.

A noter que (7) est particulièrement facile à programmer. Nous donnerons ici les programmes en langage Matlab, toutefois il est facile de les transposer en n'importe quel autre langage de programmation. Ce genre de schéma numérique peut se programmer sur n'importe quelle petite calculatrice !

Passons aux résultats !

Prenons le cas périodique, de période 1, $h = 0.01$, $k = 0.005$, et essayons de résoudre l'équation (1) avec donnée initiale

$$u(0, x) = \sin(2\pi x).$$

La solution est

$$u(t, x) = \sin\Big(2\pi(x - t)\Big).$$

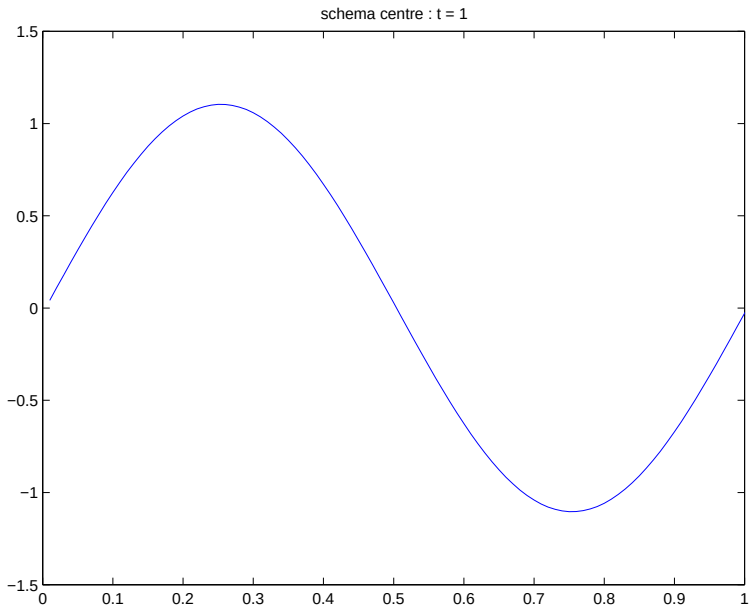

Figure 1: Schéma centré, $t = 1$

Initialisons le schéma par

$$u_i^0 = \sin(2\pi i h).$$

Et traçons u_i^n pour divers n.

Bon, pour n petit, ça marche ....

mais ne crions pas trop tôt victoire ... rapidement la solution se met à vibrer .... de plus en plus ..... et explose complètement !

Bon, c'est l'échec complet !

Recommençons donc en diminuant h et k.... c'est plus long à calculer, mais cela semble mieux .... enfin ... pas tant que cela !

Faisons un petit bilan, le schéma (7), pourtant le plus naturel et le plus simple ne marche pas du tout ! Impossible de calculer la solution de cette équation, pourtant la plus simple des équations aux dérivées partielles.

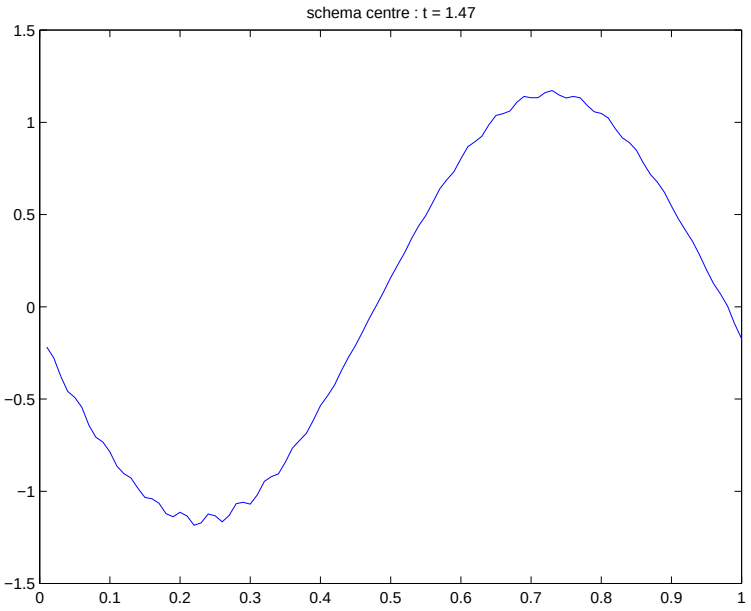

Figure 2: Schéma centré, $t = 1.47$

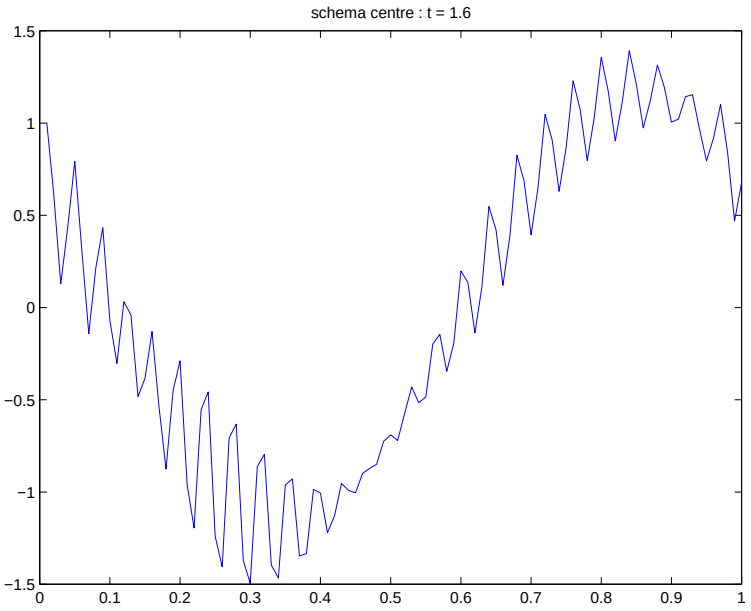

Figure 3: Schéma centré, $t = 1.6$

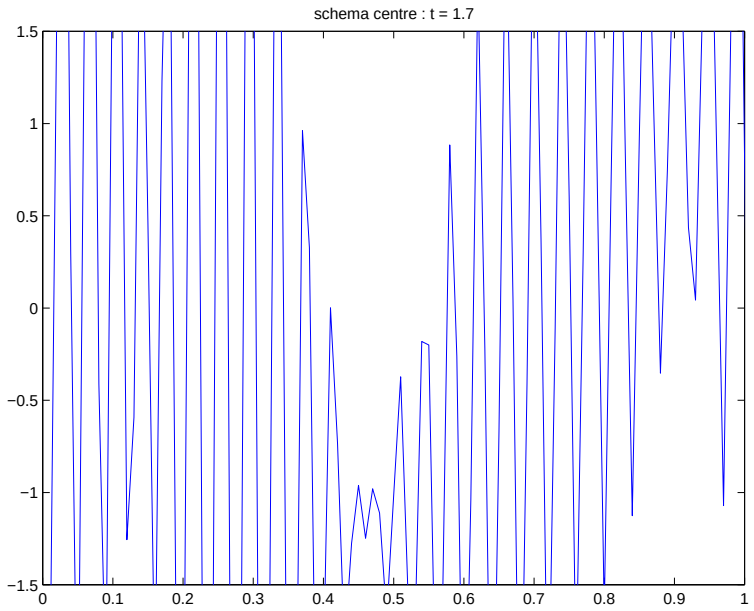

Figure 4: Schéma centré, $t = 1.7$

3 Essayons autre chose

Ah, dirons les esprits chagrins, la formule n'est pas tout à fait symétrique en t et en x. Pourquoi pas essayer

$$\frac{u_i^{n+1} - u_i^n}{k} + \frac{u_{i+1}^n - u_i^n}{h} = 0. \tag{8}$$

Essayons !

Bon ce n'est pas mieux ...

Essayons autre chose

$$\frac{u_i^{n+1} - u_i^n}{k} + \frac{u_i^n - u_{i-1}^n}{h} = 0. \tag{9}$$

Ah, ça n'oscille plus ! et c'est beaucoup plus stable .... ça marche !

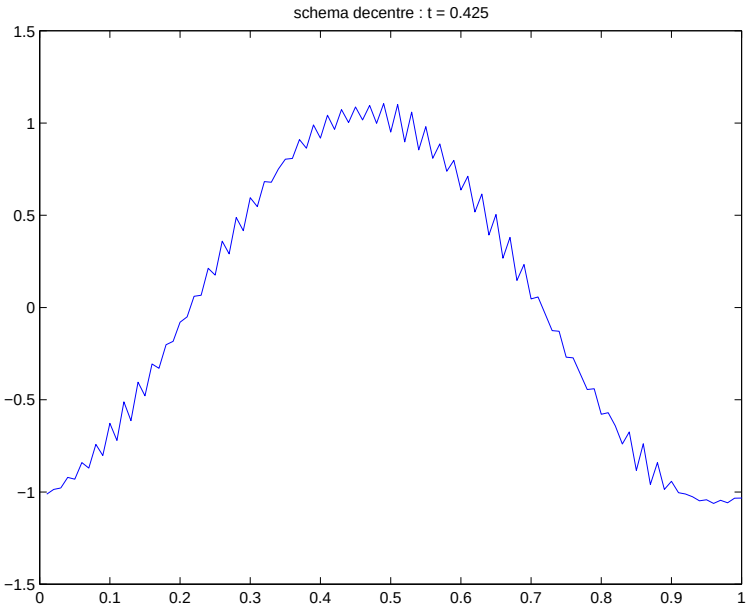

Figure 5: Schéma décentré (8), $t = 0.425$

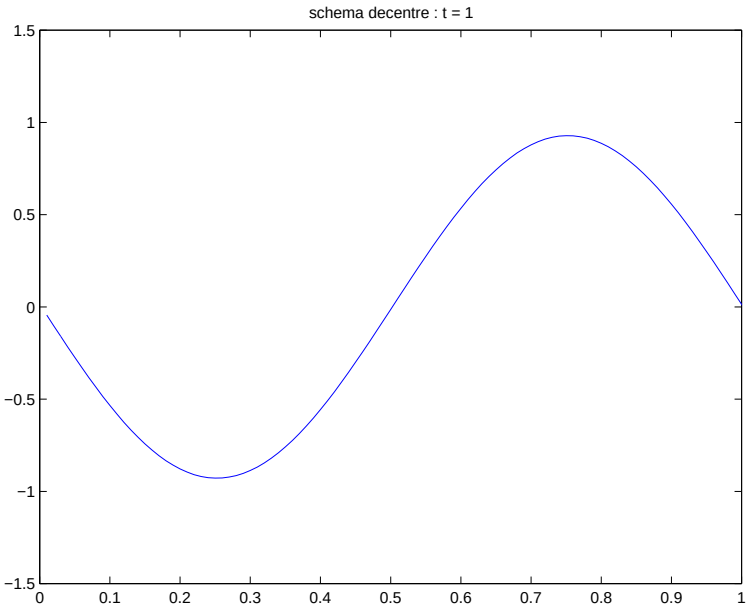

Figure 6: Schéma décentré (9), $t = 1$, $h = 0.01$, $k = 0.005$

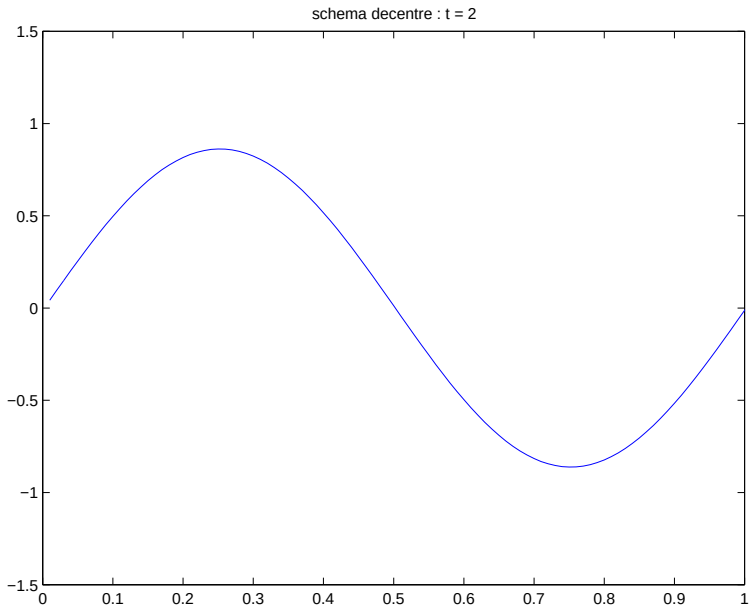

Figure 7: Schéma décentré (9), $t = 2$, $h = 0.01$, $k = 0.005$

Voilà donc que (9) marche, alors que ce schéma est on ne peut plus voisin de (8) qui lui ne marche pas du tout ! Il devient clair qu'une programmation "en aveugle" sans réflechir n'a pas beaucoup de chance de marcher ....

Poursuivons un peu. Essayons d'accélerer un peu le calcul. Pour aller plus vite, facile ... il suffit d'augmenter k. Regardons les résultats pour divers k. Soit $h = 0.01$. Regardons $k = 0.005$, $k = 0.1$. Le premier marche, pas le second ... quelle est la limite ? En fait pour avoir la stabilité pour toute donnée initiale il faut $k < h$.

Bien, passons à l'équation voisine

$$\frac{\partial u}{\partial t} - \frac{\partial u}{\partial x} = 0.$$

Maintenant ... (9) ne marche pas, mais (8) marche !

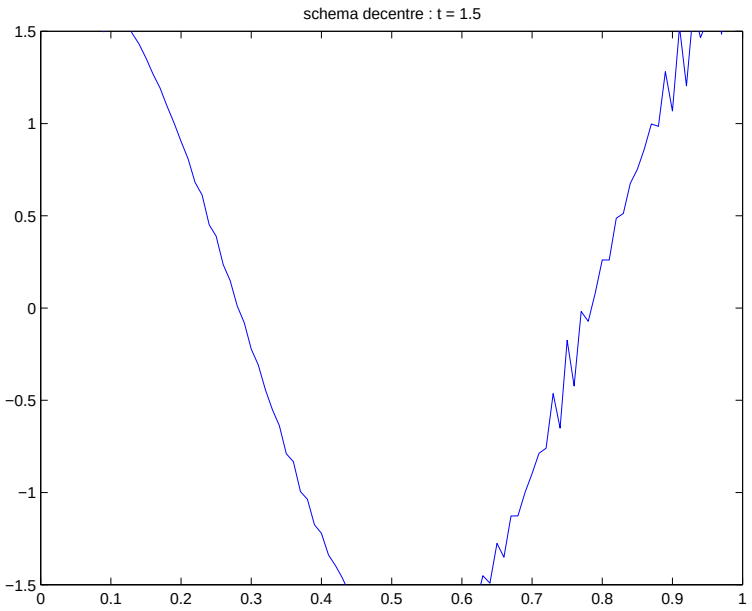

Figure 8: Schéma décentré, $t = 1.5$, $h = 0.01$, $k = 0.1$

4 Bilan

On vient de voir sur un exemple simple que le schéma numérique le plus simple que l'on puisse imaginer ne marche pas ... le schéma (9) marche bien, mais le trouver est plutôt une question de chance si on n'a pas de théorie ! Dans le cas de systèmes plus complexes on sent bien qu'un discrétisation "au hasard" ne va pas marcher ...

Comment analyser ces schémas ?

Réecrivons (7)

$$u_i^{n+1} = u_i^n + \frac{k}{2h} u_{i-1}^n - \frac{k}{2h} u_{i+1}^n. \tag{10}$$

Essayons de majorer $|u_i^{n+1}|$. Soit

$$\alpha_n = \sup |u_i^n|.$$

On a

$$|u_i^{n+1}| \leq |u_i^n| + \frac{k}{2h} |u_{i-1}^n| + \frac{k}{2h} |u_{i+1}^n|$$

soit

$$\alpha_{n+1} \leq \Big(1 + \frac{k}{h}\Big)\alpha_n.$$

On n'a donc pas mieux que

$$\alpha_n \leq \Big(1 + \frac{k}{h}\Big)^n \alpha_0.$$

Pour $k = h$ cela donne

$$\alpha_n \leq 2^n \alpha_0$$

ce qui n'est pas glorieux ... au temps $t = 1$ cela donne

$$\alpha_{1/k} = 2^{1/k}\alpha_0$$

qui est très grand quand k est petit... En fait on n'a pas mieux ... et c'est l'originie des instabilités !

A-t-on mieux pour (9) ? Le schéma (9) se réecrit

$$u_i^{n+1} = \Big(1 - \frac{k}{h}\Big)u_i^n + \frac{k}{h}u_{i-1}^n. \tag{11}$$

On a alors

$$|u_i^{n+1}| \leq \Big(1 - \frac{k}{h}\Big)|u_i^n| + \frac{k}{h}|u_{i-1}^n|$$

ce qui donne

$$\alpha_{n+1} \leq \alpha_n$$

si $1 - k/h > 0$, c'est-à-dire si $k < h$ ... tiens, revoilà la condition de stabilité ! La relation $\alpha_{n+1} < \alpha_n$ est garante de stabilité, puisqu'on est assuré que le maximum de $|u_i^n|$ décroit.

Et le schéma (8) ? Il donne

$$u_i^{n+1} = \Big(1 - \frac{k}{h}\Big)u_i^n + \frac{k}{h}u_{i-1}^n. \tag{12}$$

On a alors

$$|u_i^{n+1}| \leq \Big(1 + \frac{k}{h}\Big)|u_i^n| + \frac{k}{h}|u_{i-1}^n|$$

ce qui donne ...

$$\alpha_{n+1} \leq \Big(1 + 2\frac{k}{h}\Big)\alpha_n$$

et on est reparti dans le décor, comme pour (7) !

Bien sûr pour justifier que les solutions de (9) convergent vers la solution de l'edp quand les paramètres de discrétisation h et k tendent vers 0 (avec $k < h$), il faut encore travailler .... La preuve fait appel à des estimations *a priori* et à des arguments de compacité entre espaces fonctionnels.

5 Les programmes

Les programmes sont écrits en langage "matlab" (langage très pratique pour les calculs numériques mais hélas payant !, voir http://www.mathworks.com).

5.1 Schéma centré (7)

```
%

% Schema centre

%



% Parametre:



h = 0.01;

k = 0.005;



N = 1/h;



% Initialisation
```

```
U='';

U(2:N+1) = sin(2*pi*h*(2:N+1));

U(1) = U(N+1);

U(N+2) = U(2);


% Boucle


for t=0:k:10,

   U(2:N+1) = U(2:N+1) + (k/(2*h))*U(1:N) - (k/(2*h))*U(3:N+2);

   U(1) = U(N+1);

   U(N+2) = U(2);

   t

   plot([1:N+2]*h  + i*U);

   pause


end
```

5.2 Schéma décentré (8)

```
%

% Schema centre

%


% Parametre:


h = 0.01;

k = 0.005;


N = 1/h;


% Initialisation


U='';

U(2:N+1) = sin(2*pi*h*(2:N+1));

U(1) = U(N+1);

U(N+2) = U(2);
```

```
% Boucle


for t=0:k:10,
   U(2:N+1) = U(2:N+1)*( 1+ (k/(2*h))) - (k/(2*h))*U(3:N+2);
   U(1) = U(N+1);
   U(N+2) = U(2);
   t
   plot([1:N+2]*h  + i*U);
   pause


end
```

5.3 Autre schéma décentré (9)

```
%
% Schema centre
%


% Parametre:
```

```
h = 0.01;

k = 0.005;


N = 1/h;


% Initialisation


U='';

U(2:N+1) = sin(2*pi*h*(2:N+1));

U(1) = U(N+1);

U(N+2) = U(2);


% Boucle


for t=0:k:10,

   U(2:N+1) = U(2:N+1)*( 1 - (k/(2*h))) + (k/(2*h))*U(1:N);

   U(1) = U(N+1);

   U(N+2) = U(2);

   t
```

```
   plot([1:N+2]*h  + i*U);

   pause


end
```