Résolution Numérique de l'Équation de la chaleur

Nous avons déjà fait une introduction aux méthodes numériques pour résoudre l'équation de la chaleur dans le cas des systèmes minces:

$$\rho c_p V \frac{\partial T(t)}{\partial t} = -hS(T(t) - T_{air}), \quad T(0) = T_0.$$

où $T(t)$ était la température supposée uniforme dans le corps considéré. On va ici calculer un cas où la température varie dans le corps. On va avoir la température fonction de l'espace et du temps: $T(x,t)$.

Pour résoudre, on avait écrit la dérivée sous forme discrète de manière à calculer la nouvelle température au temps $t + \Delta t$ notée T^{n+1} en fonction de la température T^n au temps précédent t :

$$\frac{T^{n+1} - T^n}{\Delta t} = -\frac{hS}{\rho c_p V}(T^n - T_{air}),$$

pas de temps après pas de temps en partant de $T^0 = T_0$ on peut calculer la température à tous les temps. Nous allons voir dans ce chapitre comment discrétiser les dérivées en x.

1 Le problème

1.1 Échelon de température

Nous examinons un cas simple: la lamelle 1D soumise à un choc thermique (sans source de chaleur volumique). Nous rappelons les équations à résoudre:

• équation de la chaleur dans un milieu immobile isotrope homogène, avec des coefficients thermodynamiques constants, l'équation est écrite sans dimensions:

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2}$$

conditions aux limites en $x = -1$ et $x = 1$ on se donne $T(\pm 1, t) = 0$ et au temps $t = 0$, on se donne $T(x, 0) = 1$ température pariétale imposée.

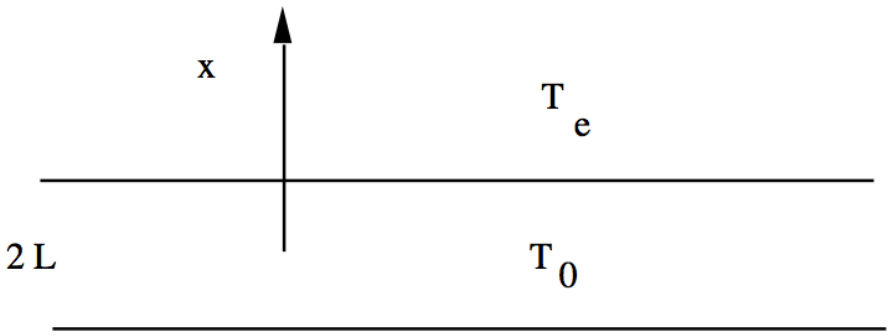

Figure 1: une lamelle infinie de température uniforme au temps initial.

1.2 Problème discret

On se place aux points x_i appelés points du maillage.

$$x_i = -1 + 2 * i/n$$

l'indice i varie de 0 à n (par exemple 100, soit 101 points et 100 segments). On appelle $\Delta x = 2/n$ le pas de discrétisation spatiale. On cherche la valeur $T(x_i, t_k)$ la température en ces points aux temps t_k discrets eux aussi. On a pour le temps:

$$t_{k+1} = t_k + \Delta t$$

le pas de temps de discrétisation est Δt.

Dans chaque tranche i, la variation d'énergie par rapport au temps $\partial T/\partial t$ est par définition des accroissements

$$\frac{\partial T(x,t)}{\partial t} \sim \frac{T(x_i, t+\Delta t) - T(x_i, t)}{\Delta t}$$

la densité de flux venant de $i-1$ vers i est de même:

$$-\frac{\partial T(x_i,t)}{\partial x} \sim -\frac{T(x_i,t) - T(x_{i-1})}{\Delta x}$$

la densité de flux allant de i vers $i+1$ est de même:

$$-\frac{\partial T(x_{i+1},t)}{\partial x} \sim -\frac{T(x_{i+1},t) - T(x_i)}{\Delta x}$$

la dérivée de la densité de flux est donc bien:

$$\Big(-\frac{\partial T(x_{i+1},t)}{\partial x}) - (-\frac{\partial T(x_i,t)}{\partial x})\Big)/\Delta x$$

ce qui donne la forme approchée discrétisée de la dérivée seconde de la température

$$\frac{T(x_{i+1},t)-2T(x_i,t)+T(x_{i-1},t)}{\Delta x}$$

On en déduit donc la forme discrète de l'équation de la chaleur,

$$\frac{T(x_i,t+\Delta t)}{\Delta t}=\frac{T(x_{i+1},t)-2T(x_i,t)+T(x_{i-1},t)}{\Delta x}$$

ou, puisque l'on veut faire une itération en temps et trouver la nouvelle distribution de chaleur au temps $t+\Delta t$ connaissant la température au temps t aux points x_i.

$$T(x_i,t+\Delta t)=T(x_i,t)+\Delta t\frac{T(x_{i+1},t)-2T(x_i,t)+T(x_{i-1},t)}{\Delta x}.$$

1.3 Mise en oeuvre

A l'aide d'un éditeur de texte, on tape les lignes suivantes de l'entête C:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
```

puis, on programme tel quel, dans un premier temps, les paramètres sont codés en "dur". La dimension des tableaux est fixée à 100, on définit le tableau des températures `T` et des nouvelles températures `Tn`. On définit un fichier de sortie.

```
int main ( ) {
  int Nmax=100;
  int i,it;
  double x[Nmax+1];          // dimension des tableaux
  double T[Nmax+1];
  double Tn[Nmax+1];
  double L=1;
  double dx=2*L/Nmax,dt=0.00001,t=0; // parametres
  FILE *f;
```

On dit bonjour, et on initialise les points du maillage: les x_i qui sont les valeurs du tableau `x[i]`.

```
printf(" Resolution de l'equation de la Chaleur MECAVENIR fev 08\n");

 for (i=0; i<=Nmax; i++)
{  /*RAZ */
 x[i] = -L + i*2*L/Nmax;
 T[i]=1;
 Tn[i]=1;
}
```

on peut maintenant avancer en temps avec la boucle sur puis, à chaque temps, résoudre en espace suivant la formule démontrée. On veille à bien mettre les conditions aux limites à 0.

```
// Calcul, avancee en temps
for(it=0;it<100000;it++)
{  t=t+dt;  // le temps augmente
  T[0]=0;  // on force les conditions aux limites
  T[Nmax]=0;
  Tn[0]=0;
  Tn[Nmax]=0;

for (i=1; i<Nmax; i++)
{ Tn[i]=T[i]+(dt/dx/dx)*(T[i+1]-2*T[i]+T[i-1]);} // equation chaleur dicretisee
```

On "swape" les tableaux, `T` devient `Tn`

```
for (i=0; i<=Nmax; i++)
    {T[i]=Tn[i];}    //echange le nouveau devient l'ancien
```

On affiche le temps, on sauve dans un fichier (syntaxe spéciale `fprintf`, et le `FILE *f` au début et attention au `%lf` du format "long float") et on attend que l'on fasse retour chariot `getchar()` pour passer au temps suivant. On ferme les dernières accolades.

```
   printf("t=%lf\n",t);
  // sauvegarde dans un fichier pour tracer avec gnuplot
  f= fopen("resultat.dat","w");
  for (i=0; i<=Nmax;i++)
  { fprintf(f,"%lf %lf \n",x[i], T[i]); }
  fclose(f);

  getchar();
```

```
 }
return 1;
}
```

Le programme est prêt, on ouvre deux fenêtres de terminal, dans une il suffit de compiler: `gcc chaleur.c -o chaleur`, cela crée l'exécutable. On le lance. Chaque fois que l'on appuie sur enter, le temps augmente. Dans le deuxième terminal que l'on a lancé, on lance `gnuplot`. On peut tracer au fur et à mesure l'évolution de la température.

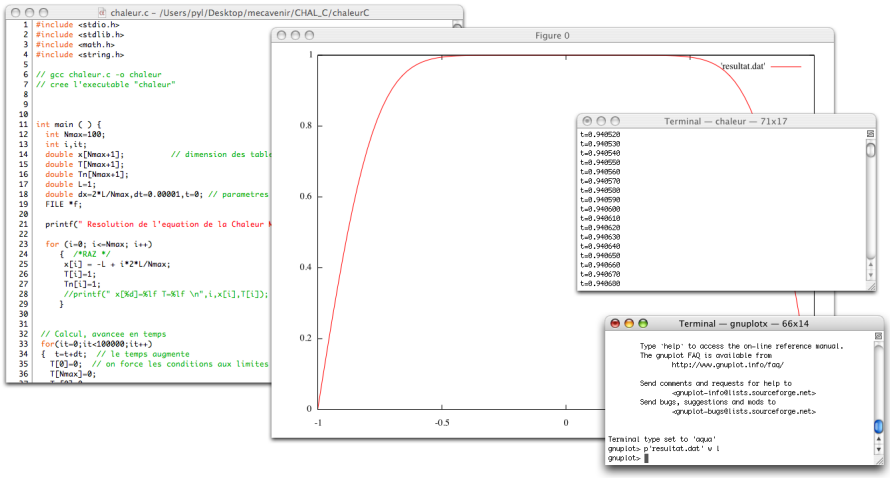

Figure 2: Les différents terminaux.

1.4 Stabilité

On remarquera que ce programme est assez lent, on peut faire des sorties uniquement tous les 100 pas de temps.

On pourrait penser augmenter le pas de temps pour aller plus vite. Mais, si on augmente trop le pas de temps, le calcul explose. Vérifier que $\Delta t = \Delta x^2/2$ est la limite supérieure à ne pas franchir. Au delà, le programme devient instable, il explose.

Le schéma que l'on a écrit est explicite, pour pouvoir résoudre avec un pas de temps plus grand, il faudrait faire un schéma dit "implicite". Cela complique un peu la résolution.

2 Résolution numérique sur le Web

On peut résoudre en direct par un calcul en différences finies l'équation de la chaleur. Il s'agit d'un programme plus complet que celui présenté ici en C, il est en Java. On change à la volée les conditions aux limites, le lien est
`http://www.lmm.jussieu.fr/ ~ lagree/SOURCES/Appliquette-JavaChal/guiChalomega_v_implicite/index.html`

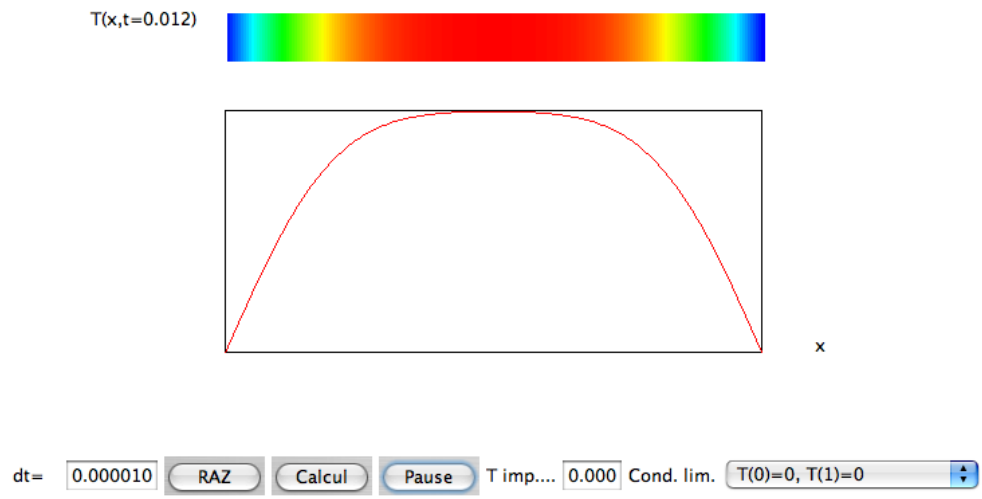

Figure 3: Résolution en direct par différences finies de l'équation de la chaleur.

3 Bibliographie

I. Danaila , F. Hecht , O. Pironneau (2003)
"Simulation numérique en C++ Cours et exercices corrigés", Dunod .
W. H. Press, S. A. Teukolsky, W.T. Vetterling, B. P. Flannery ()
"Numerical Recipes in C/C++: The Art of Scientific Computing "
D. Euvrard (1994)
"Résolution numérique des équations aux dérivées partielles de la physique, de la mécanique et des sciences de l'ingénieur : Différences finies, éléments finis, problèmes en domaines non bornés" Masson

Consulter aussi `http://www.lmm.jussieu.fr/~lagree/COURS/MECAVENIR` le cours complet de thermique de P.-Y. Lagrée.

Annexe 1

On remet le programme en entier

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

// gcc chaleur.c -o chaleur
// cree l'executable "chaleur"



int main ( ) {
  int Nmax=100;
  int i,it;
  double x[Nmax+1];          // dimension des tableaux
  double T[Nmax+1];
  double Tn[Nmax+1];
  double L=1;
  double dx=2*L/Nmax,dt=0.00001,t=0; // parametres
  FILE *f;

  printf(" Resolution de l'equation de la Chaleur MECAVENIR fev 08\n");

  for (i=0; i<=Nmax; i++)
 {  /*RAZ */
  x[i] = -L + i*2*L/Nmax;
  T[i]=1;
  Tn[i]=1;
  //printf(" x[%d]=%lf T=%lf \n",i,x[i],T[i]);
 }


 // Calcul, avancee en temps
 for(it=0;it<100000;it++)
 {  t=t+dt;  // le temps augmente
   T[0]=0;  // on force les conditions aux limites
```

```
  T[Nmax]=0;
  Tn[0]=0;
  Tn[Nmax]=0;

 for (i=1; i<Nmax; i++)
 { Tn[i]=T[i]+(dt/dx/dx)*(T[i+1]-2*T[i]+T[i-1]);} // equation chaleur dicretisee

 for (i=0; i<=Nmax; i++)
     {T[i]=Tn[i];}    //echange le nouveau devient l'ancien

      printf("t=%lf\n",t);

  // sauvegarde dans un fichier pour tracer avec gnuplot
  f= fopen("resultat.dat","w");
  for (i=0; i<=Nmax;i++)
  { fprintf(f,"%lf %lf \n",x[i], T[i]); }
  fclose(f);

  getchar();

 }

return 1;
}
```

Annexe 2

On remet le programme en entier en mettant la lecture d'un fichier de données, et l'interfaçage avec gnuplot et des pointeurs autorisant un dimensionnement dynamique des tableaux.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
// PYL 02/08
// gcc chaleur2.c -o chaleur2
// cree l'executable "chaleur2"
//#define GNUPLOT_PATH "/usr/local/bin/gnuplot"
#define GNUPLOT_PATH "gnuplot"


int main ( ) {
  int Nmax=100;
  int i,it=0;
  double *x,*T,*Tn;   //pointeurs
  double L=1,tmax=1;
  double dx=2*L/Nmax,dt=0.00001,t=0; // parametres
   FILE *gp;         // pour gnuplot
   FILE *f;


  printf(" Resolution de l'equation de la Chaleur MECAVENIR fev 08\n");
 f =fopen("D.IN","r");
 fscanf(f,"dt=%lf              \n",&dt);
 fscanf(f,"tmax=%lf              \n",&tmax);
 fscanf(f,"Nmax=%d               \n",&Nmax);
 fclose(f);

   printf("dt=%lf tmax=%lf Nmax=%d \n",dt,tmax,Nmax);
   printf("\n");
   //getchar();
   dx=2*L/Nmax;
```

```
    x =  (double*)calloc(Nmax+1,sizeof(double));
    T =  (double*)calloc(Nmax+1,sizeof(double));
    Tn=  (double*)calloc(Nmax+1,sizeof(double));
      // dimension dynamique des tableaux

  for (i=0; i<=Nmax; i++)
 {  /*RAZ */
  x[i] = -L + i*dx;
  T[i]=1;
  Tn[i]=1;
 }


// ouverture d'un tuyau pour gnuplot!!!
 gp = popen(GNUPLOT_PATH, "w");
 if(gp == NULL){
 fprintf(stderr, "Aye pas de %s.", GNUPLOT_PATH);
     exit(EXIT_FAILURE);
       }else{
 fprintf(gp, "set xlabel \"x\";set ylabel \"T(x,t)\";set  title \"chaleur PYL\" \n");


 // Calcul, avancee en temps
while(t<tmax)
 {  it++;
  t=t+dt;  // le temps augmente
   T[0]=0;  // on force les conditions aux limites
   T[Nmax]=0;
   Tn[0]=0;
   Tn[Nmax]=0;

 for (i=1; i<Nmax; i++)
 { Tn[i]=T[i]+(dt/dx/dx)*(T[i+1]-2*T[i]+T[i-1]);} // equation chaleur dicretisee

 for (i=0; i<=Nmax; i++)
     {T[i]=Tn[i];}    //echange le nouveau devient l'ancien


   // sauvegarde dans un fichier pour tracer avec gnuplot
   int dtt= .01/dt;
```

```
   if(it%dtt==0){
     printf("t=%lf \r",t);
   f= fopen("resultat.dat","w");
   for (i=0; i<=Nmax;i++)
   { fprintf(f,"%lf %lf \n",x[i], T[i]); }
   fclose(f);

   fprintf(gp,"p [][0:1]'resultat.dat'  u 1:2 not w l  \n");
   fflush(gp);}

 }

return 1;
}
}
```

dans le fichier `D.IN`, il y a

```
dt=0.000001
tmax=5
nx=520
```