INTEGRACIÓ NUMÈRICA D'EDP
Grau de Matemàtiques UAB, 2016–2017

Pràctica en C : Diferències finites per equacions d'evolució

1 Plantejament

L'objectiu d'aquesta pràctica és resoldre un problema d'evolució mitjançant diferències finites. El problema triat és un problema mixte per l'equació de la calor sobre una regió rectangular. Concretament,

$$\begin{array}{rcl} u_t - \Delta u &=& f \quad \text{a} \quad [0,T]\times\Omega, \\ u &=& g \quad \text{a} \quad [0,T]\times\partial\Omega, \\ u &=& h \quad \text{a} \quad \{0\}\times\Omega. \end{array} \tag{1}$$

on $\Omega = [0, L_x] \times [0, L_y]$.

Discretitzem $[0,T]$ amb pas $\delta_t = T/n_t$, $[0, L_x]$ amb pas $\delta_x = L_x/n_x$ i $[0, L_y]$ amb pas $\delta_y = L_y/n_y$. Per a $k = 0 \div n_t$, $i = 0 \div n_x$, $j = 0 \div n_y$, denotem per $U_{k,i,j}$ les aproximacions que obtindrem de $u(k\delta_t, i\delta_x, j\delta_y)$. Imposarem les condicions de frontera de (1) prenent

$$\begin{array}{lll} U_{0,i,j} = h(i\delta_x, j\delta_y), & i = 0 \div n_x,\ j = 0 \div n_y, & \\ U_{k,i,0} = g(k\delta_t, i\delta_x, 0), \quad U_{k,i,n_y} = g(k\delta_t, i\delta_x, L_y), & k = 1 \div n_t,\ i = 0 \div n_x, & \\ U_{k,0,j} = g(k\delta_t, 0, j\delta_y), \quad U_{k,n_x,j} = g(k\delta_t, L_x, j\delta_y), & k = 1 \div n_t,\ j = 1 \div n_y - 1. & \end{array} \tag{2}$$

En prendre diferències finites endavant de primer ordre en el temps i centrades de segon ordre en l'espai a l'equació de la calor a (1), obtenim

$$\begin{array}{rcl} U_{k+1,i,j} &=& (1 - 2\mu_x - 2\mu_y)U_{k,i,j} \\ && +\mu_x(U_{k,i+1,j} + U_{k,i-1,j}) + \mu_y(U_{k,i,j+1} + U_{k,i,j-1}) \\ && \delta_t f_{k,i,j}, \end{array} \tag{3}$$

on $\mu_x = \delta_t/\delta_x^2$, $\mu_y = \delta_t/\delta_y^2$, $U_{k,i,j} \approx u(k\delta_t, i\delta_x, j\delta_y)$ i $f_{k,i,j} = f(k\delta_t, i\delta_x, j\delta_y)$. L'equació anterior és un *esquema explícit en diferències finites* perquè permet obtenir recurrenment les aproximacions $\{U_{k,i,j}\}_{k=0\div\delta_t, i=0\div\delta_x, j=0\div\delta_y}$ de la següent forma:

Algorisme 1.1

Omplir $\{U_{0,i,j}\}_{i=0\div n_x, j=1\div n_y-1}$ *a partir de (2)*
$\forall k = 0 \div n_t - 1$
  Omplir $\{U_{k+1,i,0}, U_{k+1,i,n_y}, U_{k+1,0,j}, U_{k+1,n_x,j}\}_{i=0\div n_x, j=1\div n_y-1}$ *a partir de (2)*
  $\forall i = 1 \div n_x - 1$
    $\forall j = 1 \div n_y - 1$
      Omplir $U_{k+1,i,j}$ *a partir de (3).*

Aquest mètode de resolució numèrica del problema (1) és atractiu per la seva simplicitat, però presenta dos inconvenients:

- Per a assegurar convergència de l'aproximació obtinguda cap a la solució exacta, és a dir, perquè

$$U_{k,i,j} - u(k\delta_t, i\delta_x, j\delta_y) \xrightarrow{\delta_t,\delta_x,\delta_y\to 0} 0,$$

cal imposar la condició

$$1 - 2\mu_x - 2\mu_y \geq 0. \tag{4}$$

- Suposant que se satisfà la condició (4), l'aproximació obtinguda és quadràtica en l'espai però lineal en el temps. Concretament,

$$|U_{k,i,j} - u(k\delta_t, i\delta_x, j\delta_y)| = O(\delta_t + \delta_x^2 + \delta_y^2).$$

Això ens obliga a prendre passos δ_t molt petits respecte dels passos δ_x, δ_y per a tenir bones aproximacions.

El problema de la no convergència es pot solucionar discretitzant l'equació de la calor en el temps mitjançant una diferència finita endarrera de primer ordre, en comptes d'endavant, de manera que obtindríem un *esquema implícit*. Existeix una variant d'aquesta estratègia, coneguda com a *mètode de Crank–Nicolson*, que consisteix a promitjar les discretitzacions espacials en els instants de temps $k\delta_t$ i $(k+1)\delta_t$. Concretament,

$$\begin{aligned}
&\frac{U_{k+1,i,j} - U_{k,i,j}}{\delta_t} \\
&-\frac{1}{2}\Big(\frac{U_{k+1,i+1,j} - 2U_{k+1,i,j} + U_{k+1,i-1,j}}{\delta_x^2} + \frac{U_{k,i+1,j} - 2U_{k,i,j} + U_{k,i-1,j}}{\delta_x^2}\Big) \\
&-\frac{1}{2}\Big(\frac{U_{k+1,i,j+1} - 2U_{k+1,i,j} + U_{k+1,i,j-1}}{\delta_y^2} + \frac{U_{k,i,j+1} - 2U_{k,i,j} + U_{k,i,j-1}}{\delta_y^2}\Big) \\
&\qquad = \frac{1}{2}(f_{k+1,i,j} + f_{k,i,j}).
\end{aligned} \tag{5}$$

Si ara volguéssim fer l'anàl·leg de l'algorisme 1.1, suposant que som al pas k i coneixem $\{U_{k,i,j}\}_{i=0\div n_x, j=0\div n_y}$, l'equació (5) no ens permet trobar $U_{k+1,i,j}$ a partir dels valors del pas k, donat que apareixen simultàniament els valors $\{U_{k,i,j}, U_{k,i\pm1,j}, U_{k,i,j\pm1}\}$ i $\{U_{k+1,i,j}, U_{k+1,i\pm1,j}, U_{k+1,i,j\pm1}\}$. Variant $j = 1 \div n_y - 1, i = 1 \div n_x - 1$, l'equació (5) és un sistema lineal en $\{U_{k+1,i,j}\}_{j=1\div n_y-1, i=1\div n_x-1}$.

Suposem l'ordenació d'incògnites

$$\begin{array}{lllll}
U_{k+1,1,1} & , \; U_{k+1,2,1} & , \; \cdots & , \; U_{k+1,n_x-1,1}, \\
U_{k+1,1,2} & , \; U_{k+1,2,2} & , \; \cdots & , \; U_{k+1,n_x-1,2}, \\
 & & \cdots & , \\
U_{k+1,1,n_y-1} & , \; U_{k+1,2,n_y-1} & , \; \cdots & , \; U_{k+1,n_x-1,n_y-1}.
\end{array}$$

Aleshores, amb les mateixes notacions de (3), el mètode de sobrerelaxació amb paràmetre ω per a aquest sistema es pot descriure com segueix: prenem $\{U^{(0)}_{k+1,i,j}\}_{i,j}$

aproximació inicial de la solució de (5), i, per a $m \geq 0$, calculem $\{U^{(m+1)}_{k+1,i,j}\}_{i,j}$ a partir de $\{U^{(m)}_{k+1,i,j}\}_{i,j}$ mitjançant

$$
\begin{array}{l}
\forall j = 1, \ldots, n_y - 1 \\
\quad \forall i = 1, \ldots, n_x - 1 \\
\quad U^{(m+1)}_{k+1,i,j} = (1-\omega)U^{(m)}_{k+1,i,j} + \dfrac{\omega}{1+\mu_x+\mu_y}\Big[(1-\mu_x-\mu_y)U_{k,i,j} \\
\qquad\qquad + \dfrac{\mu_x}{2}\Big(U^{(m)}_{k+1,i+1,j} + U^{(m+1)}_{k+1,i-1,j} + U_{k,i+1,j} + U_{k,i-1,j}\Big) \\
\qquad\qquad + \dfrac{\mu_y}{2}\Big(U^{(m)}_{k+1,i,j+1} + U^{(m+1)}_{k+1,i,j-1} + U_{k,i,j+1} + U_{k,i,j-1}\Big) \\
\qquad\qquad + \dfrac{\delta_t}{2}(f_{k+1,i,j} + f_{k,i,j})\Big].
\end{array}
\tag{6}
$$

Com a aproximació inicial $\{U^{(0)}_{k+1,i,j}\}_{i,j}$, podem prendre els valors $\{U_{k,i,j}\}_{i,j}$, determinats al pas de temps anterior.

Amb tot això, podem descriure algorísmicament *l'esquema implícit de Crank-Nicolson per a l'equació de la calor* com segueix

Algorisme 1.2

Omplir $\{U_{0,i,j}\}_{i=0\div n_x, j=0\div n_y}$ *a partir de (2)*
$\forall k = 0 \div n_t - 1$
  Omplir $\{U_{k+1,i,0}, U_{k+1,i,n_y}, U_{k+1,0,j}, U_{k+1,n_x,j}\}_{i=0\div n_x, j=1\div n_y-1}$ *a partir de (2)*
  Obtenir $\{U_{k+1,i,j}\}_{i=1\div n_x-1, j=1\div n_y-1}$ *mitjançant la iteració (6)*

Amb aquest procediment, a més d'evitar la necessitat d'imposar la condició (4) per a tenir convergència, obtenim també que l'error és quadràtic tant en el pas temporal com en els passos espacials. Concretament,

$$|u(k\delta_t, i\delta_x, j\delta_y) - U_{k,i,j}| = O(\delta_t^2 + \delta_x^2 + \delta_y^2),$$

de manera que podem prendre passos en el temps del mateix ordre que en l'espai.

2 Guió

1. Implementeu l'esquema explícit en diferències finites (3).

2. Per tal de verificar la correcció de la implementació anterior, resoleu el problema (1) en un cas en què sabeu que heu de trobar la solució exacta. Per exemple, trieu com a u un polinomi de grau 1 en t i grau ≤ 3 en x i y, llavors trobeu f, g i h tals que u sigui solució de (1). Apliqueu l'esquema en diferències finites explícit que heu implementat al punt anterior i comproveu que obteniu la solució exacta.

3. Suposem que $\Omega = [0,1] \times [0,1]$ representa una placa d'acer molt prima, sota la qual posem una font de calor que genera $F(\tau, x, y)$ cal $/(\mathrm{s} \cdot \mathrm{cm}^3)$ a l'instant τ (en s) i al punt (x, y) (en cm), on F està definida per

$$F(\tau, x, y) = \begin{cases} 100 & \text{si } \|(x,y) - (0.5, 0.5)\|_2 < 0.2, \\ 0 & \text{altrament,} \end{cases}$$

Suposem també que, per $\tau = 0$, la placa es troba a 0 °C, i per tot temps τ la vora de la placa es manté també a 0 °C.

L'evolució de la temperatura sobre el domini Ω des de l'instant $\tau = 0$ fins a l'instant $\tau = S$ (en segons) ve donada per la solució $v(\tau, x, y)$ del problema mixte

$$\begin{array}{rcll} c\rho v_\tau - \kappa \Delta v &=& F & \text{a } [0,S] \times \Omega, \\ v &=& G & \text{a } [0,S] \times \partial\Omega, \\ v &=& H & \text{a } \{0\} \times \Omega, \end{array} \tag{7}$$

on c és la calor específica del material, ρ és la seva densitat i κ la seva conductivitat tèrmica, suposant que totes tres són constants. Per a la placa d'acer, els valors són

$$\kappa = 0.13 \, \frac{\text{cal}}{\text{s} \cdot \text{cm} \cdot {}^\circ\text{C}}, \qquad c = 0.11 \, \frac{\text{cal}}{\text{g} \cdot {}^\circ\text{C}}, \qquad \rho = 7.8 \, \frac{\text{g}}{\text{cm}^3}.$$

(a) Comproveu que el canvi de temps donat per

$$u(t, x, y) = v(\tau, x, y), \quad \text{per } t = \frac{\kappa}{c\rho}\tau,$$

transforma el problema (7) en el problema (1), per a

$$f(t, x, y) = \frac{1}{\kappa} F(\frac{c\rho}{\kappa} t, x, y), \quad g(t, x, y) = G(\frac{c\rho}{\kappa} t, x, y), \quad h(x, y) = H(x, y).$$

(b) Comproveu que, si preneu passos $\delta_t, \delta_x, \delta_y$ que no satisfan la condició (4) i apliqueu l'algorisme 1.1, obteniu solucions físicament sense sentit.

(c) Suposem que volem obtenir una precissió de l'ordre de 10^{-4}. Quina dificultat us suposa emprar l'algorisme 1.1 si voleu arribar fins a $\tau = 1$?

4. Implementeu ara el mètode de Crank-Nicolson (5) mitjançant la iteració SOR (6) (useu $\omega = 1.7$). Com abans, verifiqueu la correcció d'aquesta implementació resolent (1) triant com a u el polinomi del punt 2. Noteu que en aquest cas no obteniu la solució exacta. Quin és l'error que detecteu?

5. Resoleu mitjançant l'esquema implícit el problema del punt 3. Podeu ara arribar "còmodament" fins a temps $\tau = 1$? Feu una animació del procés de difusió de calor i incloeu a la memòria alguns fotogrames (vegeu la figura 1).

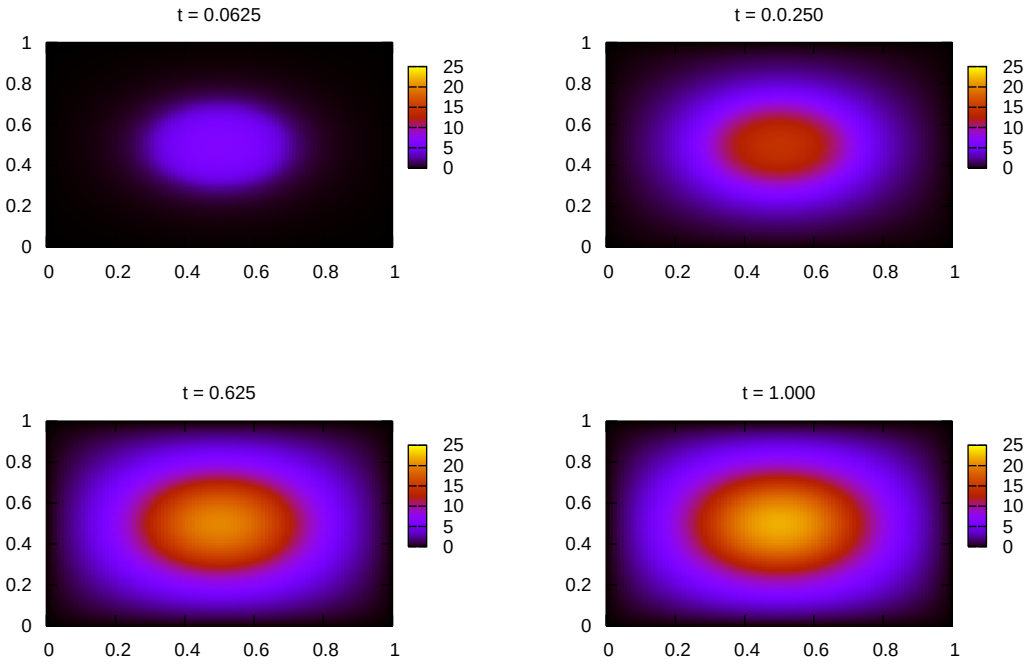

Figura 1: Fotogrames corresponents al punt 5 del guió.

3 Estructura del codi

El codi s'ha de dividir en diversos fitxers:

- Un fitxer `grRDF.h`, llistat completament a continuació.
- Un fitxer `grRDF.c`, amb el codi de les funcions prototipades a `grRDF.h`.
- Tants fitxers `.c` com programes principals necessiteu per als diversos punts del guió. Per exemple, per al punt 2, podeu fer un fitxer anomenat `polexpl.c`, amb un programa principal que resolgui el problema 1 amb l'algorisme 1.1 prenent com a u un polinomi. Més endavant teniu un exemple d'esquelet d'aquest fitxer. Per generar l'executable corresponent, haureu de fer

  ```
  gcc -opolexpl -g -Wall grRDF.c polexpl.c -lm
  ```

  Opcionalment, podeu escriure un `Makefile` amb una entrada per cadascun dels executables que volgueu generar.

```
************************ grRDF.h ************************
/*!
 * \brief Estructura graella d'evoluci\'o.
 *
 * Aquesta estructura est\`a preparada per a contenir una
 * llesca de temps de la graella corresponent a la resoluci\'o
 * per difer\`encies finites d'una EDP d'evoluci\'o sobre un
 * rectangle.
 */
typedef struct {
/*!\brief Pas en x (i.e. \f$\delta_x\f$). */
   double dx;
/*!\brief Pas en y (i.e. \f$\delta_y\f$). */
   double dy;
/*!\brief "Pendent num\`eric" en x (i.e. \f$\mu_x\f$). */
   double mux;
/*!\brief "Pendent num\`eric" en y (i.e. \f$\mu_y\f$). */
   double muy;
/*!\brief Nombre de nodes en x menys 1 (i.e. \f$n_x\f$). */
   int nx;
/*!\brief Nombre de nodes en y menys 1 (i.e. \f$n_y\f$). */
   int ny;
/*!\brief Instant de temps al qual correspon el contingut de
 * la graella. */
   double t;
/*!\brief Pas en temps (i.e. \f$\delta_t\f$) */
   double dt;
/*!\brief Graella.
 *
 * El valor de la funci\'o al punt de coordenades
 * <tt>x=i*dx</tt> i <tt>y=j*dy</tt> \'es <tt>u[j*(nx+1)+i]</tt>
 * */
   double *u;
} grRDF;

/*!
 * \brief Inicialitza un objecte grRDF.
 *
 * Assigna par\`ametres, allocata mem\`oria i posa temps a zero.
 */
void grRDF_init (grRDF *gr, double dx, double dy, double dt,
      int nx, int ny, double (*h)(double x, double y));

/*!
```

```
 * \brief Per a l'equaci\'o de la calor sobre un rectangle, fa
 * un pas en temps del m\'etode expl\'{\i}cit (3) (o sigui, un pas
 * en k de l'algorisme 1.1)
 */
void grRDF_pasCalExpl (grRDF *gr,
      double (*f)(double t, double x, double y),
      double (*g)(double t, double x, double y));

/*!
 * \brief Escriu un objecte grRDF, en aquest format:
 * \verbatim
 * # t <valor de t>
 * <x> <y> <u(x,y)>
 * ...
 * \endverbatim
 * Recorre la graella per files, insertant un salt de l\'{\i}nia
 * a cada canvi de fila, i dos salts de l\'{\i}nia un cop
 * completada la graella
 */
void grRDF_escriure (grRDF *gr, FILE *fp);

/*!\brief Allibera la mem\'oria d'un objecte \c grRDF. */
void grRDF_allib (grRDF *gr);

/*!
 * \brief Per a l'equaci\'o de la calor sobre un rectangle, fa
 * un pas en temps del m\'etode de Crank-Nicolson (5) (o
 * sigui, un pas en k de l'algorisme 1.2)
 */
int grRDF_pasCalCN (grRDF *gr, double w, double tol, int maxit,
      double (*f)(double t, double x, double y),
      double (*g)(double t, double x, double y));

************************ grRDF.c ************************
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/* ... Codi de les funcions de grRDF.h ... */
```

```
************************ polexpl.c ************************
/*!
 * \brief Funcio f del problema mixt per l'equaci\'o de la
 * calor, imposant que la soluci\'o sigui un polinomi.
 */
double f_pol (double t, double x, double y) {
/* ... */
}

/*!
 * \brief Funcio g del problema mixt per l'equaci\'o de la
 * calor, imposant que la soluci\'o sigui un polinomi.
 */
double g_pol (double t, double x, double y) {
/* ... */
}

/*!
 * \brief Funcio h del problema mixt per l'equaci\'o de la
 * calor, imposant que la soluci\'o sigui un polinomi.
 */
double h_pol (double x, double y) {
/* ... */
}

/*!\brief Implementa l'algorisme 1.1 per al punt 2 del gui\'o. */
int main (int argc, char *argv[]) {
   /* ... declaracions diverses ... */
/* Objecte graella */
   grRDF gr;
   /* ... llegir par\`ametres ... */
/* Inicialitzo graella (par\`ametres i llesca t=0 */
   grRDF_init(&gr,dx,dy,dt,nx,ny,h_pol);
/* Escric graella inicial */
   grRDF_escriure(&gr,stdout);
/*
 * Bucle en temps:
 * A cada pas crida grRDF_pasCalExpl(&gr,f_pol,g_pol)
 * i escriu graella mitjan\c{c}ant crida a grRDF_escriure(&gr,stdout)
 */
   /* ... el bucle en temps ... */
/* Tot fet! */
   return 0;
}
```