

EQUAZIONE DEL CALORE *

A. SOMMARIVA[†]

Conoscenze richieste. Derivate di ordine superiore. Derivate parziali. Formula di Taylor. Equazioni differenziali. Teoremi di Gershgorin. Calcolo matriciale e vettoriale. Matrici definite positive. Conoscenza di Matlab/Octave.

Conoscenze ottenute. Equazione del calore. Metodo delle linee. Discretizzazione con Eulero implicito ed Eulero esplicito. Stabilità dei due metodi.

1. Introduzione. Si consideri l'equazione del calore [1, p.414]

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + G, & 0 < x < 1, t > 0 \\ u(0, t) = d_0(t), u(1, t) = d_1(t), & t \geq 0 \\ u(x, 0) = f(x), & 0 \leq x \leq 1 \end{cases} \quad (1.1)$$

Sia $m > 0$ un numero naturale e si ponga $h_x = 1/m$ ed $x_j = jh_x$ con $j = 0, 1, \dots, m$. Si può mostrare che per $j = 1, 2, \dots, m-1$ e $\xi_j \in (x_{j-1}, x_{j+1})$

$$\frac{\partial^2 u}{\partial x^2}(x_j, t) = \frac{u(x_{j+1}, t) - 2u(x_j, t) + u(x_{j-1}, t))}{h_x^2} - \frac{h_x^2}{12} \frac{\partial^4 u}{\partial x^4}(\xi_j, t) \quad (1.2)$$

e che quindi per $j = 1, \dots, m-1$, da (1.2)

$$\frac{\partial u(x_j, t)}{\partial t} = \frac{u(x_{j+1}, t) - 2u(x_j, t) + u(x_{j-1}, t))}{h_x^2} \quad (1.3)$$

$$+ G(x_j, t) - \frac{h_x^2}{12} \frac{\partial^4 u(\xi_j, t)}{\partial x^4} \quad (1.4)$$

Tralasciando il termine finale e posto $u_j(t) := u(x_j, t)$ otteniamo quindi per $j = 1, \dots, m-1$ il sistema di equazioni differenziali

$$u'_j(t) = \frac{u_{j+1}(t) - 2u_j(t) + u_{j-1}(t))}{h_x^2} + G(x_j, t) \quad (1.5)$$

Risolto (1.5), si avrà una approssimazione della soluzione dell'equazione del calore per $x_j = jh_x$ e $t \geq 0$. Il procedimento appena descritto è noto in letteratura come *metodo delle linee*. Nel risolvere il sistema dobbiamo far attenzione alle condizioni sul bordo

$$u_0(t) = d_0(t), \quad u_m(t) = d_1(t)$$

e ricordare che la condizione iniziale del sistema di equazioni differenziali è

$$u_j(0) = f(x_j), \quad j = 1, \dots, m-1.$$

Il sistema differenziale (1.5) può essere riscritto matricialmente. Posto

$$\mathbf{u}(t) := [u_1(t), \dots, u_{m-1}(t)]^T$$

*Ultima revisione: 26 gennaio 2010

[†]Dipartimento di Matematica Pura ed Applicata, Università degli Studi di Padova, stanza 419, via Trieste 63, 35121 Padova, Italia (alvise@euler.math.unipd.it). Telefono: +39-049-8271350.

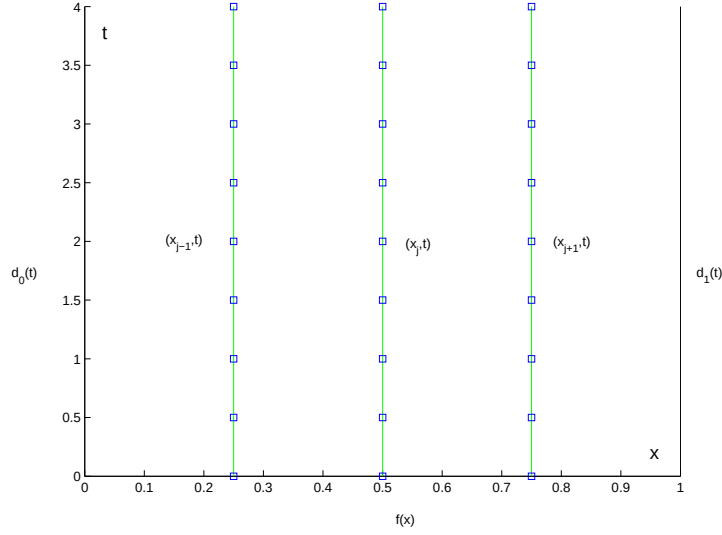


FIGURA 1.1. La discretizzazione del dominio.

$$\mathbf{u}_0(t) := [f(x_1), \dots, f(x_{m-1})]^T$$

$$\mathbf{g}(t) := \left[\frac{1}{h_x^2} d_0(t), \dots, \frac{1}{h_x^2} d_1(t) \right]^T + [G(x_1, t), 0, \dots, 0, G(x_{m-1}, t)]^T$$

$$\Lambda = \frac{1}{h_x^2} \begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix} \quad (1.6)$$

otteniamo che (1.5) è equivalente al sistema di equazioni differenziali (lineari)

$$\mathbf{u}'(t) = \Lambda \mathbf{u}(t) + \mathbf{g}(t), \quad \mathbf{u}(0) = \mathbf{u}_0(t) \quad (1.7)$$

Tra i metodi più comuni nel risolvere il problema differenziale (di Cauchy)

$$\begin{cases} \mathbf{u}'(t) = F(t, \mathbf{u}(t)) \\ \mathbf{u}(0) = \mathbf{u}_0 \end{cases} \quad (1.8)$$

citiamo il metodo di Eulero esplicito

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + hF(t_n, \mathbf{u}_n) \\ \mathbf{u}_0 \text{ assegnato} \end{cases} \quad (1.9)$$

e quello di Eulero implicito

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + hF(t_{n+1}, \mathbf{u}_{n+1}) \\ \mathbf{u}_0 \text{ assegnato} \end{cases} \quad (1.10)$$

Nel nostro caso da (1.7)

$$F(t, \mathbf{v}(t)) := \Lambda \mathbf{v}(t) + \mathbf{g}(t)$$

e quindi il metodo di Eulero esplicito genera la successione

$$\begin{cases} \mathbf{v}_{n+1} = \mathbf{v}_n + h_t(\Lambda \mathbf{v}_n + \mathbf{g}(t_n)) \\ \mathbf{v}_0 \text{ assegnato} \end{cases} \quad (1.11)$$

mentre Eulero implicito determina

$$\begin{cases} \mathbf{v}_{n+1} = \mathbf{v}_n + h_t(\Lambda \mathbf{v}_{n+1} + \mathbf{g}(t_{n+1})) \\ \mathbf{v}_0 \text{ assegnato} \end{cases} \quad (1.12)$$

o equivalentemente

$$\begin{cases} (I - h_t \Lambda) \mathbf{v}_{n+1} = \mathbf{v}_n + h_t \mathbf{g}(t_{n+1}) \\ \mathbf{v}_0 \text{ assegnato} \end{cases} \quad (1.13)$$

Osserviamo che a differenza del metodo esplicito, in (1.12) ad ogni iterazione si richiede la soluzione di un'equazione (che nel nostro caso è lineare). Usando i primi due teoremi di Gerschgorin, si può mostrare che la matrice $(I - h_t \Lambda)$ è definita positiva (e quindi non singolare).

A partire da Eulero esplicito ed Eulero implicito si definiscono i cosiddetti θ -metodi in cui

$$\begin{cases} \mathbf{v}_{n+1} = (1 - \theta)(\mathbf{v}_n + h_t(\Lambda \mathbf{v}_n + \mathbf{g}(t_n))) \\ \quad + \theta(\mathbf{v}_n + h_t(\Lambda \mathbf{v}_{n+1} + \mathbf{g}(t_{n+1}))) \\ \mathbf{v}_0 \text{ assegnato} \end{cases}$$

Si noti che per $\theta = 0$ si ottiene il metodo di Eulero esplicito mentre per $\theta = 1$ si ottiene il metodo di Eulero implicito.

2. Un esperimento numerico. In questa sezione studiamo numericamente l'equazione del calore [1, p.414], [4, p.735]

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + G, \quad 0 < x < 1, t > 0 \\ u(0, t) = d_0(t), u(1, t) = d_1(t), t \geq 0 \\ u(x, 0) = f(x), 0 \leq x \leq 1 \end{cases} \quad (2.1)$$

per

$$\begin{cases} G(x, t) = (-0.1 + \pi^2)(\exp(-0.1 \cdot t) \sin(\pi x)) \\ d_0(t) = 0, d_1(t) = 0 \\ f(x) = \sin(\pi x) \end{cases} \quad (2.2)$$

avente quale soluzione

$$u(x, t) = \exp(-0.1 \cdot t) \sin(\pi x).$$

Con la funzione $\mathbf{g}$ costruiamo il vettore $\mathbf{g}(t)$

```
function gt=g(t,x,delta,d0,d1,G)
```

```
% CALCOLA
```

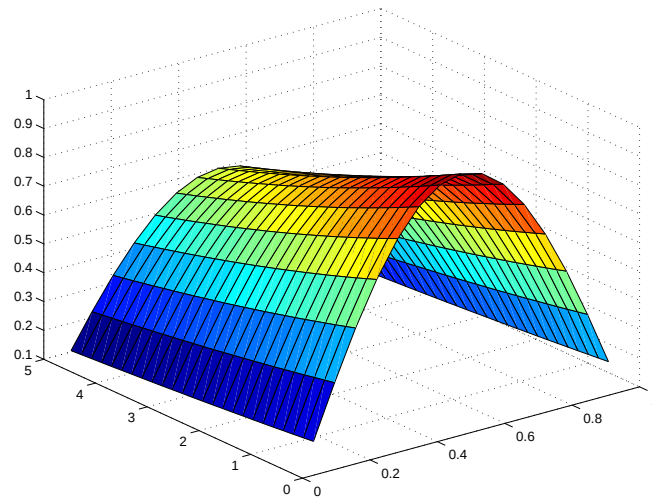


FIGURA 2.1. Grafico della soluzione dell'equazione del calore (2.1).

```
% g(t)=(1/hx^2)*(d0(t),0,...,0,d1(t))+[G(x1,t),...,G(x_(m-1),t)].
```

```
gt=feval(G,x,t);
```

```
gt(1)=gt(1)+(1/delta^2)*feval(d0,t);
```

```
gt(length(gt))=gt(length(gt))+(1/delta^2)*feval(d1,t);
```

mentre con `cnheat` risolviamo l'equazione del calore via un θ -metodo (1.14),

```
function [V_hist,x_mid,t_mid,hx,ht]=cnheat(theta,tfin,m,f,d0,d1,G,time_step_factor)
```

```
% INPUT.
```

```
% theta : PARAMETRO DEL THETA METODO (TRA 0 ED 1, POSSIBILMENTE 0,1 INCLUSI).
```

```
% tfin : Istante FINALE.
```

```
% m : DETERMINA IL PASSO SPAZIALE CON PASSO  $h_x=1/m$ .
```

```
% f :  $u(x,0)=f(x)$ 
```

```
% d0, d1 :  $u(0,t)=d0(t)$ ,  $u(1,t)=d1(t)$ .
```

```
% G :  $(D_t)u=(D^2_x)u+G$ .
```

```
% time_step_factor: IL PASSO TEMPORALE E'  $time\_step\_factor*(h_x^2)/2$ .
```

```
%
```

```
% OUTPUT.
```

```
% V_hist : SOLUZIONE Istante PER Istante. RIGHE INIZIALI -> BASSO t.
```

```
% x_mid : PUNTI INTERNI NELLA VARIABILE x.
```

```
% t_mid : PUNTI INTERNI NELLA VARIABILE t.
```

```
% hx : PASSO SPAZIALE.
```

```
% ht : PASSO TEMPORALE.
```

```
hx=1/m; % DELTA=hx
```

```
matorder=m-1;
```

```
ht=time_step_factor*(hx^2)/2; % STEP TEMPORALE.
```

```

x=(0:hx:1)';
x_mid=x(2:length(x)-1,:);           % PUNTI INTERNI.

u0=feval(f,x_mid);                   % VETTORE SOLUZIONE AL TEMPO "0".

t=(0:ht:tfin)';                      % TEMPI DA ANALIZZARE.
t_mid=t(2:length(t),1);              % TEMPI t > 0.

% COSTRUZIONE MATRICE LAMBDA.
submat=zeros(1,matorder); eye(matorder-1) (zeros(1,matorder-1))';
supmat=submat';
lambda_matrix=(1/hx^2)*(diag(-2*ones(m-1,1))+submat+supmat);

if theta < 1
    gt_prev=g(0,x_mid,hx,d0,d1,G);
end

V_old=u0;
V_hist=[V_old'];
err_hist=[];

for index=2:length(t)

    % STUDIANDO u'(t_curr)=LAMBDA*u(t_curr)+g(t_curr),
    % DOVE g(t)=(1/hx^2)*(d0(t),0,...,0,d1(t))+[G(x1,t),...,G(x_(m-1),t)].
    t_curr=t(index);
    gt_curr=g(t_curr,x_mid,hx,d0,d1,G);
    switch theta
    case 1
        A=eye(size(lambda_matrix))-ht*lambda_matrix;
        b=V_old+ht*gt_curr;
        V_new=A\b;
    case 0
        V_new=V_old+ht*(lambda_matrix*V_old+gt_prev);
        gt_prev=gt_curr;
    otherwise
        A=eye(size(lambda_matrix))-(ht*theta)*lambda_matrix;
        b=V_old+(ht*(1-theta))*lambda_matrix*V_old+...
            (ht*(1-theta))*gt_prev+(ht*theta)*gt_curr;
        V_new=A\b;
        gt_prev=gt_curr;
    end
    V_hist=[V_hist; V_new'];
    V_old=V_new;
end

```

Le funzioni precedenti vengono utilizzate da `demoheatcn` per risolvere l'equazione del calore definita da (2.1)-(2.2).

1. Il parametro `m` determina il passo spaziale $h_x = 1/m$;
2. se `theta=0` allora si utilizza il metodo di Eulero esplicito, mentre se `theta=1` Eulero implicito;
3. la variabile `tfin` determina l'istante finale;
4. il parametro `timestepfactor` determina il passo temporale a partire da quello

spaziale; se $h_{tmax} = h_x^2/2$ per $h_x = 1/m$ il passo usato da Eulero esplicito è

`timestepfactor*htmax`

5. in seguito la demo valuta uno dei metodi per $m = 2^k$ con $k = 2, 3, 4$, calcolando le $\text{ratio } e_{2h_x}/e_{h_x}$ dove si è posto

$$e_{h_x} = \max_i |v_i(t_{\text{fin}}) - u_{x_i, t_{\text{fin}}}|$$

in cui $v_i(t) := v^{(h_x)}(x_i, t_{\text{fin}})$ è la soluzione ottenuta dal metodo scegliendo il parametro temporale uguale a $h_x = 1/m$;

6. il parametro `mvect` all'interno dello switch iniziale, determina gli m da analizzare.

Una versione di `demoheatcn` è

```
%-----
% QUESTO CODICE SEGUE [ATKINSON, AN INTRODUCTION TO NUMERICAL ANALYSIS, p.414].
% NECESSITA DELLA FUNCTION: "g".
%-----

demoexample=2; % SETTA EQUAZIONI E TEMPI DIVERSI. VALORI: 1,2.

theta=1;      % [theta=0]   EULERO ESPLICITO.
              % [theta=0.5] CRANK-NICOLSON.
              % [theta=1]   EULERO IMPLICITO.

mvect=[4 8 16 32 64]; % VETTORE DI "m".
time_step_factor_vect=ones(size(mvect));

switch demoexample
case 1

    tfin=10;      % TEMPO FINALE.

    G=inline('(-0.1+pi^2)*(exp((-0.1)*t).*sin(pi*x))','x','t');
    d0=inline('zeros(size(t))','t');
    d1=inline('exp((-0.1)*t).*sin(pi)','t');
    f=inline('sin(pi*x)','x');

    solution=inline('exp((-0.1)*t).*sin(pi*x)','x','t');

case 2

    tfin=0.1;      % TEMPO FINALE.

    G=inline('zeros(size(x))','x','t');
    d0=inline('zeros(size(t))','t');
    d1=inline('zeros(size(t))','t');
    f=inline('sin(pi*x)','x');

    solution=inline('exp((-pi^2)*t).*sin(pi*x)','x','t');

end
```

```

err_hist_prev_m=[];

fprintf('\n \t [THETA]: %3.3f [TFIN]: %3.3f',theta,tfin);

for mindex=1:length(mvect)

    m=mvect(mindex); % SELEZIONE m TRA UNA LISTA.
    time_step_factor=time_step_factor_vect(mindex); % TIME FACTOR.
    err_hist=[];

    [V_hist,x_mid,t_mid,hx,ht]=cnheat(theta,tfin,m,f,d0,d1,G,time_step_factor);

    [X, Y]=meshgrid(x_mid,t_mid); % VALUTAZIONE ERRORI.
    U=feval(solution,X,Y);      % SOL. ESATTA NELLA GRIGLIA.

    % ERRORE COMPIUTO PER "m" FISSATO.
    err=norm( U(size(U,1),:)-V_hist(size(V_hist,1),:), inf);
    err_hist=[err_hist; err];

    fprintf('\n \t [m]: %3.0f [ERROR]: %2.2e [hx]: %2.2e [ht]: %2.2e', m, err,hx,ht);

    if length(err_hist_prev_m) > 0
        fprintf(' [RATIO]: %2.2f', err_hist_prev_m(size(err_hist_prev_m,1))/err );
    end

    err_hist_prev_m=err_hist;

end

```

2.1. Eulero esplicito. Per motivi di stabilità tipici di Eulero esplicito [1, p. 416], il passo temporale h_t deve essere inferiore o uguale a $h_x^2/2$. Vediamo su vari esempi cosa succede numericamente.

Dopo aver settato in `demoheat` il parametro `theta=0` scegliamo per esempio `timestepfactor=1.5`. Quindi dalla shell di Matlab/Octave digitiamo quanto segue

```

>> demoheatcn

[THETA]: 0.000 [TFIN]: 5.000 [TIME STEP FACTOR]: 1.50e+000

[m]:   4 [ERROR]: 1.40e+004 [hx]: 2.50e-001 [ht]: 4.69e-002
[m]:   8 [ERROR]: 7.32e+100 [hx]: 1.25e-001 [ht]: 1.17e-002 [RATIO]: 0.00
[m]:  16 [ERROR]: NaN [hx]: 6.25e-002 [ht]: 2.93e-003 [RATIO]: NaN

>>

```

Evidentemente bisogna scegliere un parametro `timestepfactor` più piccolo. Proviamo ad esempio `timestepfactor=1.1`.

```

>> demoheatcn

[THETA]: 0.000 [TFIN]: 5.000 [TIME STEP FACTOR]: 1.10e+000

```

```
[m]: 4 [ERROR]: 3.25e-002 [hx]: 2.50e-001 [ht]: 3.44e-002
[m]: 8 [ERROR]: 2.89e+011 [hx]: 1.25e-001 [ht]: 8.59e-003 [RATIO]: 0.00
[m]: 16 [ERROR]: 3.77e+149 [hx]: 6.25e-002 [ht]: 2.15e-003 [RATIO]: 0.00
```

>>

Il metodo non fornisce evidentemente risultati apprezzabili. Scegliamo ora `timestepfactor=1.0`: il metodo di Eulero esplicito finalmente converge.

>> `demoheatcn`

```
[THETA]: 0.000 [TFIN]: 5.000 [TIME STEP FACTOR]: 1.00e+000
```

```
[m]: 4 [ERROR]: 3.25e-002 [hx]: 2.50e-001 [ht]: 3.13e-002
[m]: 8 [ERROR]: 7.93e-003 [hx]: 1.25e-001 [ht]: 7.81e-003 [RATIO]: 4.10
[m]: 16 [ERROR]: 1.97e-003 [hx]: 6.25e-002 [ht]: 1.95e-003 [RATIO]: 4.02
```

>>

In definitiva affinché il metodo di Eulero esplicito converga, il passo temporale dev'essere scelto dell'ordine di $h_x^2/2$, che in molti casi risulta essere troppo piccolo e rende il metodo non competitivo dal punto di vista computazionale.

A tal proposito, supponiamo di dover risolvere

$$\begin{cases} \frac{\partial u}{\partial t} = a \frac{\partial^2 u}{\partial x^2} + G(x, t), & 0 < x < 1, t > 0 \\ u(0, t) = d_0(t), u(1, t) = d_1(t), & t \geq 0 \\ u(x, 0) = f(x), & 0 \leq x \leq 1 \end{cases} \quad (2.3)$$

In effetti si può dimostrare che se h_x è lo step spaziale e h_t lo step temporale allora l'errore non si amplifica nel tempo e il metodo è stabile se

$$\gamma = a \frac{h_t}{h_x^2} \leq \frac{1}{2}$$

Si può provare che tale condizione è necessaria e sufficiente affinché il metodo sia stabile e che se tanto la soluzione quanto d_0 , d_1 , g ed f sono sufficientemente regolari allora con ovvia notazione

$$\max_{i,k} |u(x_i, t_k) - u_{i,k}| = O(h_t + h_x^2).$$

Per $a = 1$ la sopracitata condizione diventa

$$\frac{h_t}{h_x^2} \leq \frac{1}{2}$$

cioè

$$h_t \leq \frac{h_x^2}{2},$$

chiarendo perchè per `timestepfactor=1` il metodo di Eulero esplicito converga.

2.2. Eulero implicito. Vediamo in questa sezione il comportamento di Eulero implicito. Osserviamo che a differenza di Eulero esplicito richiede la soluzione di sistemi lineari tridiagonali, ma ciò non è un problema dal punto di vista computazionale (il costo è di $5m$ per ogni t_i).

Proviamo il comportamento per `timestepfactor=1.5`, dopo aver posto `theta=1`. Il metodo di Eulero implicito, a differenza di Eulero esplicito converge. Infatti


```
>> demoheatcn
```

```
[THETA]: 1.000 [TFIN]: 5.000 [TIME STEP FACTOR]: 1.50e+000
```

```
[m]: 4 [ERROR]: 3.26e-002 [hx]: 2.50e-001 [ht]: 4.69e-002  
[m]: 8 [ERROR]: 7.95e-003 [hx]: 1.25e-001 [ht]: 1.17e-002 [RATIO]: 4.11  
[m]: 16 [ERROR]: 1.97e-003 [hx]: 6.25e-002 [ht]: 2.93e-003 [RATIO]: 4.03
```

```
>>
```

Per curiosità proviamo per `timestepfactor=10`, quindi con un passo temporale h_t relativamente grande, ottenendo

```
>> demoheatcn
```

```
[THETA]: 1.000 [TFIN]: 5.000 [TIME STEP FACTOR]: 1.00e+001
```

```
[m]: 4 [ERROR]: 3.26e-002 [hx]: 2.50e-001 [ht]: 3.13e-001  
[m]: 8 [ERROR]: 7.96e-003 [hx]: 1.25e-001 [ht]: 7.81e-002 [RATIO]: 4.10  
[m]: 16 [ERROR]: 1.98e-003 [hx]: 6.25e-002 [ht]: 1.95e-002 [RATIO]: 4.02
```

```
>>
```

Si può mostrare che in effetti, il metodo è A-stabile, e quindi non richiede alcun vincolo sullo step temporale. Ciò significa che per ogni valore di h_x e h_t la propagazione dell'errore avanzando nel tempo è *sotto controllo* o come si dice il metodo è *incondizionatamente stabile* [2, p.171]. Inoltre se tanto la soluzione quanto d_0 , d_1 , g ed f sono sufficientemente regolari allora con ovvia notazione

$$\max_{i,k} |u(x_i, t_k) - u_{i,k}| = O(h_t + h_x^2).$$

3. Esercizi.

1. Rifare i test precedenti per $\theta = 0.5$ (metodo di Crank-Nicolson).
2. Rifare i test precedenti per il parametro `demoexample=2` che corrisponde alla risoluzione del problema

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + G, & 0 < x < 1, t > 0 \\ u(0, t) = d_0(t), u(1, t) = d_1(t), & t \geq 0 \\ u(x, 0) = f(x), & 0 \leq x \leq 1 \end{cases} \quad (3.1)$$

per

$$\begin{cases} G(x, t) = 0 \\ d_0(t) = 0, d_1(t) = 0 \\ f(x) = \sin(\pi x) \end{cases} \quad (3.2)$$

avente quale soluzione

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

4. Una stima dell'errore. Per quanto concerne la stima dell'errore, si può dimostrare [1, p. 415] che se d_0 , d_1 , G ed f sono sufficientemente regolari allora

$$\max_{j \in 0, \dots, m, t \in [0, T]} |u(x_j, t) - v(x_j, t)| \leq C_T h_x^2$$

dove C_T è una costante indipendente da h_x , $u_j = u(x_j, \cdot)$ è la soluzione esatta dell'equazione del calore mentre v_j è la soluzione approssimata, ottenute dalla discretizzazione numerica (1.5). Nei metodi precedenti in effetti si è visto che la ratio è uguale a 4, segno che la stima è effettivamente realizzata. Infatti se dimezziamo h l'errore diventa 4 volte inferiore. Ciò è conseguenza del fatto che

$$e_{2h_x} \approx C_T (2h_x)^2$$

$$e_{h_x} \approx C_T (h_x)^2$$

implica

$$\frac{e_{2h_x}}{e_{h_x}} \approx \frac{C_T (2h_x)^2}{C_T h_x^2} = 4.$$

Notiamo che si parla della soluzione esatta di (1.5) e non di quella offerta dal metodo numerico. Abbiamo visto infatti che per cattive scelte dello step temporale, il metodo di Eulero esplicito offre risultati non sensati.

RIFERIMENTI BIBLIOGRAFICI

- [1] K. Atkinson, *Introduction to Numerical Analysis*, Wiley, 1989.
- [2] K. Atkinson and W. Han, *Theoretical Numerical Analysis*, Springer Verlag, 2001.
- [3] D. Bini, M. Capovani e O. Menchi, *Metodi numerici per l'algebra lineare*, Zanichelli, 1988.
- [4] V. Comincioli, *Analisi Numerica, metodi modelli applicazioni*, Mc Graw-Hill, 1990.
- [5] S.D. Conte e C. de Boor, *Elementary Numerical Analysis, 3rd Edition*, Mc Graw-Hill, 1980.
- [6] The MathWorks Inc., *Numerical Computing with Matlab*, <http://www.mathworks.com/moler>.
- [7] A. Quarteroni e F. Saleri, *Introduzione al calcolo scientifico*, Springer Verlag, 2006.
- [8] A. Suli e D. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003.