DICACIM programme A.Y. 2019–20
Numerical Methods for Partial Differential Equations

Lab 3
Numerical solution of parabolic problems

Paola Gervasio

DICATAM, Università degli Studi di Brescia (Italy)

uniBS, May, 2020

- Caso 1d: scaricare il file `FEM_1d_heat.zip` dalla pagina moodle del corso,
- Caso 2d (e 3d): PDE toolbox di MATLAB

Problema 1 ($d = 1$)

Siano $\Omega = (0, 1)$ e $T = 1$. Approssimare la soluzione u di

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{\partial^2 u}{\partial x^2} = \sin(x)(cos(t) - sin(t)) & (x, t) \in \Omega \times (0, T) \\ u(0, t) = 0, \;\; u(1, t) = \sin(1)\cos(t) & t \in (0, T) \\ u(x, 0) = \sin(x) & x \in \Omega \end{cases}$$

con FEM-$\mathbb{P}_1$ in spazio e $\theta-$metodo in tempo, con $\theta \in \{0, 0.5, 1\}$. Sapendo che la soluzione esatta è $u(x, t) = \sin(x)\cos(t)$, verificare l'ordine di convergenza del $\theta-$metodo con tre diversi valori di h: $h = 0.008$, $h = 0.004$, $h = 0.002$.

Stima di convergenza del $\theta-$metodo

Nell ipotesi che u_0, f e u siano sufficientemente regolari, si ha:

$$\|u(t^n) - u_h^n\|^2_{L^2(\Omega)} + 2\alpha\Delta t \sum_{k=1}^{n} \|u(t^k) - u_h^k\|^2_{H^1(\Omega)} \leq C(u_0, f, u)(\Delta t^{q(\theta)} + h^{2r})$$

dove

$$q(\theta) = \begin{cases} 2 & \text{se } \theta \neq 1/2 \\ 4 & \text{se } \theta = 1/2, \end{cases}$$

e r è il grado locale FEM.
Quindi, se h è sufficientemente piccolo (tale che $h^{2r} < \Delta t^{q(\theta)}$) si ha

$$\|u(T) - u_h^{N_t}\|_{L^2(\Omega)} \sim \begin{cases} \Delta t & \text{Eulero } (\theta = 0,\ \theta = 1) \\ \Delta t^2 & \text{Crank-Nicolson } (\theta = 1/2) \end{cases}$$

(vedere anche la lezione 7)

Il file FEM_1d_heat.zip contiene il file fem_1d_heat_solver.m.

```
>> help fem_1d_heat_solver
  fem_1d_solver: solve du/dt-mu d^2u/dx^2+sigma u=f
                 in Omega x (t0,T)
  with Dirichlet and/or Neumann boundary conditions
  by either P1-fem or P2-fem on a uniform grid.

  [nodes,un1]=fem_1d_heat_solver(geom,problem_data,...
                      parameters)

  Input: geom: struct with fields:
           geom.xa, geom.xb = end-points of Omega
           geom.t0, geom.tf = end-points of (t0,T)
         problem_data: struct with coefficients and
                       functions data
                  .mu =  mu (positive constant)
...
  Output: nodes = column array with the nodes of the mesh
          un1 = column array of the numerical solution
                at the final time T
```

Soluzione.

Scrivere uno script e:

1. definire i dati
2. richiamare `fem_1d_heat_solver` con:
   - $\theta = 1$ (EI), $h = 0.008$, $\Delta t = 1/20,\ 1/40,\ 1/80,\ 1/160,\ 1/320$
   - $\theta = 1$ (EI), $h = 0.004$, $\Delta t = ...$
   - $\theta = 1$ (EI), $h = 0.002$, $\Delta t = ...$
   - $\theta = 1/2$ (CN), $h = 0.008$, $\Delta t = ...$
   - $\theta = 1/2$ (CN), $h = 0.004$, $\Delta t = ...$
   - $\theta = 1/2$ (CN), $h = 0.002$, $\Delta t = ...$
   - $\theta = 0$ (EE), $h = 0.008$, $\Delta t = 1/320$, si ha $\Delta t < ch^2$? (per garantire la stabilità assoluta?)
3. disegnare la soluzione
4. calcolare gli errori con `fem_1d_errors`.

Problema 2 ($d = 1$)

Consideriamo una barra omogenea di alluminio lunga tre metri e con sezione uniforme. Siamo interessati a simulare l'evoluzione della temperatura nella barra partendo da una opportuna condizione iniziale e qualora la sorgente sia nulla, risolvendo l'equazione del calore

$$\rho C_p \frac{\partial u}{\partial t} - \nabla \cdot (k \nabla u) = f \quad in\ \Omega \times (t_0, T).$$

Se noi imponiamo condizioni adiabatiche sulla superficie laterale della barra (ovvero condizioni di Neumann omogenee) e condizioni di Dirichlet alle estremità della barra, la temperatura dipende solo dalla coordinata assiale (denotata con x). Quindi il problema può essere modellato dall'equazione del calore monodimensionale, completata con la condizione iniziale al tempo $t = t_0$ e da condizioni al bordo di Dirichlet agli estremi del dominio computazionale ridotto $\Omega = (0, L)$, con $L = 3$m. L'alluminio puro ha conducibilità termica pari a $k = 237$ W/(m K), una densità $\rho = 2700$kg/m^3 e una capacità di calore specifico $c = 897$ J/(kg K). Infine consideriamo la condizione iniziale $T(x, 0) = 500$ K se $x \in (1, 2)$, $T(x, 0) = 250$ K altrimenti (si noti che tale funzione è discontinua), e condizioni al bordo di Dirichlet $T(0, t) = T(3, t) = 250$ K.

Oscillazioni di CN in corrispondenza della discontinuità del dato iniziale.

$d \geq 2$ PDE Toolbox di MATLAB

1. creare l'oggetto,
2. definire la **geometria** (Ω),
3. definire i dati: coefficienti, termine sorgente, condizione iniziale (NEW),
4. definire le **condizioni al bordo**,
5. costruire la **mesh** ($\mathcal{T}_h$),
6. **risolvere** il sistema di equazioni (calcolare u_h^n),
7. rappresentare **graficamente** la soluzione (NEW).

Definire i dati del problema

L'equazione differenziale di riferimento è:

$$\begin{cases} \rho C_p \frac{\partial u}{\partial t} - \nabla \cdot (k \nabla u) = f & \text{in } \Omega \times (t_0, T) \\ u = g_D & \text{on } \partial\Omega_D \times (t_0, T) \\ \mu \mathbf{n} \cdot \nabla u = g_N & \text{on } \partial\Omega_N \times (t_0, T) \\ \mu \mathbf{n} \cdot \nabla u + \alpha u = \alpha u_{ext} & \text{on } \partial\Omega_R \times (t_0, T) \\ u = u^0 & \text{in } \Omega \times \{t_0\} \end{cases}$$

g_D = temp. esterna
g_N = flusso di calore
scambio termico convettivo

dove i dati sono

- ρ = densità di massa del materiale,
- C_p = calore specifico,
- k = conducibilità termica,
- f = sorgente di calore (interna),
- α = coefficiente di scambio termico convettivo,
- u_{ext} = temperatura esterna.

Discretizzazione in tempo

MATLAB utilizza `ode15s.m`, uno schema di discretizzazione in tempo a passo adattivo. (Calcolo Scientifico, Cap. 8)
Il passo adattivo viene scelto a partire da due parametri:

- `AbsoluteTolerance` (default $= 10^{-6}$): è una soglia sotto la quale il valore di una singola componente della soluzione non è più considerata importante. Questa proprietà determina l'accuratezza della soluzione quando questa si avvicina a zero.
- `RelativeTolerance` (default $= 10^{-3}$): è una misura dell'errore relativo su ogni componente della soluzione numerica. Esso controlla il numero di cifre corrette per ogni componente della soluzione, eccetto per le componenti più piccole di `AbsoluteTolerance` . `RelativeTolerance=1e-3` corrisponde a circa 0.1% di accuratezza sulla soluzione esatta.

Per cambiare questi parametri (prima di risolvere):

```
model.SolverOptions.RelativeTolerance = 1.0000e-02;
model.SolverOptions.AbsoluteTolerance = 1.0000e-04;
```

Problema 3

Il dominio $\Omega = R_1 \setminus (R_2 \cup R_3)$, con $R_1 = (0,1) \times (0,2)$, $R_2 = (0.3, 0.4) \times (0.3, 0.9)$ e $R_3 = (0.5, 0.6) \times (1.1., 1.7)$, rappresenta la sezione verticale di un blocco di materiale con due cavità.
La densità di massa del materiale è pari a 7.2 kg/m^3, il calore specifico è pari a $C_p = 500$ J/(kg K), mentre la conducibilità termica è $k = 50$ W/(m K). La parete sinistra del blocco è scaldata a temperatura costante $T = 373$K, la parete destra è a contatto con l'aria a temperatura di $T_{ext} = 293$K (il coefficiente di scambio termico convettivo è $\alpha = 25$W/(m^2 K)), mentre attraverso le altre pareti non avviene scambio di calore.
Simulare l'evoluzione della temperatura all'interno della regione Ω nell'intervallo temporale $(0, 20)$ s, supponendo che la temperatura iniziale sia pari a $T_0 = 273.15$K.

Geometria

```
model=createpde; % oggetto PDEModel
R1=[3,4,0,1,1,0,0,0,2,2]';
R2=[3,4,0.3,0.4,0.4,0.3,0.3,0.3,0.9,0.9]';
R3=[3,4,0.5,0.6,0.6,0.5,1.1,1.1,1.7,1.7]';
gd=[R1,R2,R3];
ns=(char('R1','R2','R3'))';
sf='R1-R2-R3';
g=decsg(gd,sf,ns);
% importiamo la geometria nel modello
geometryFromEdges(model,g)
% per visualizzare la geometria
figure(1); clf
pdegplot(model,'EdgeLabels','on')
```

Coefficienti di PDE model

Ricordiamo che la forma generale dell'equazione è:

$$m\frac{\partial^2 u}{\partial t^2} + d\frac{\partial u}{\partial t} - \nabla \cdot (c\nabla u) + au = f$$

e che le condizioni al bordo sono

$$\begin{aligned} &\textit{Dirichlet}: && hu = r && \text{on } \partial\Omega_D \\ &\textit{Neumann}: && \mathbf{n} \cdot (c\nabla u) + qu = g && \text{on } \partial\Omega_N \end{aligned}$$

```
% coefficienti
rho=7.2; % kg/m^3
Cp=500; % J/(kg K)
k=50; % W/(m K)
specifyCoefficients(model,'m',0,'d',rho*Cp,'c',k,...
                    'a',0,'f',0);
% condizioni al bordo
applyBoundaryCondition(model,'dirichlet',...
                'Edge',7,'u',373.15); % sinistra
applyBoundaryCondition(model,'neumann',...
                'Edge',1,'q',25,'g',293); % destra
% definisco la condizione iniziale
setInitialConditions(model,273.15);
```

Mesh e risoluzione

```
% genero la mesh
generateMesh(model,'Hmax',0.05,...
                'GeometricOrder','quadratic');
% istanti in cui si vuole rappresentare la soluzione
tlist=0:1:20;  % secondi
%  per risolvere
results=solvepde(model,tlist);
```

Plot della soluzione

```
% calcolo il gradiente
[gradx,grady]=evaluateGradient(results);
% limiti per la barra dei colori
% (altrimenti ad ogni istante cambiano)
clim=[min(min(results.NodalSolution)),...
      max(max(results.NodalSolution))];

figure(2); clf
nt=size(tlist,2); % numero di istanti
for n=1:nt
% etraggo la soluzione numerica all'istante n
   u=results.NodalSolution(:,n);
   pl=pdeplot(model,'XYData',u,'Contour','on',...
     'FlowData',[-gradx(:,n),-grady(:,n)]);
   set(gca,'CLim',clim)
   title(['t=',num2str(tlist(n))]);
   pause(0.5)
end
```