Appunti di Tutorato

Metodi Numerici per le Equazioni Differenziali

Michele Ginesi

Attenzione! Queste note non hanno pretesa nè di completezza nè di correttezza. Si tratta infatti solo di una traccia su come risolvere alcuni esercizi del corso *Metodi Numerici per le Equazioni Differenziali* dell'Università degli Studi di Verona. Sono presenti sia mancanze che, probabilmente, errori.

Capitolo 1

Boundary Value Problems

In questa prima parte considereremo i problemi con condizioni al contorno (in inglese, Boudary Values Problem o BVP). Si tratta di equazioni differenziali al secondo ordine in un intervallo (a, b) munite di *condizioni al contorno*, ovvero di valori, determinati dal problema, imposti alla soluzione (o alla sua derivata prima).

1.1 Caso lineare

Inizieremo considerando tre problemi lineari, con la stessa soluzione e stessa equazione differenziale, ma considereremo diversi tipi di condizioni al contorno. Per tutti i tre seguenti problemi, la soluzione è

$$\hat{u}(x) = x^3 - 3x^2 + x + 1,$$

e l'equazione differenziale che considereremo è

$$u + u' + u'' = x^3 + x - 4, \quad x \in (0, 1).$$

D'ora in avanti, denoteremo con $\mathbf{D}_1$ e $\mathbf{D}_2$ le matrici che discretizzano, rispettivamente, la derivata prima e seconda, ovvero

$$\mathbf{D}_1 = \frac{1}{2h} \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ -1 & 0 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 0 & 1 \\ 0 & \cdots & 0 & 0 & -1 & 0 \end{bmatrix},$$

ed

$$\mathbf{D}_2 = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 1 & -2 \end{bmatrix}.$$

Quindi, la discretizzazione del nostro problema differenziale diventa

$$\mathbf{Au} = \mathbf{b}$$

dove

$$\mathbf{A} = \mathbf{I} + \mathbf{D}_1 + \mathbf{D}_2 \quad \text{e} \quad \mathbf{b}_i = \mathbf{x}_i^3 + \mathbf{x}_i - 4$$

per ogni nodo i della discretizzazione del dominio $(0, 1)$. Tuttavia, il problema così scritto non tiene (ancora) conto delle condizioni al contorno.
Quanto fatto fin'ora può essere riassunto nella seguente porzione di codice (in cui il numero di elementi `m` in cui dividiamo l'intervallo è già stato definito)

```
x = linspace(0,1,m);
x = x.';
h = 1/(m-1);
D1 = toeplitz(sparse(1,2,-1/(2*h),1,m),sparse(1,2,1/(2*h),1,m));
D2 = toeplitz(sparse([1,1],[1,2],[-2,1]/(h^2),1,m));
A = D2 + D1 + speye(m);
b = x.^3+x-4;
```

Nel primo caso, considereremo condizioni al bordo di Dirichlet. Il problema diviene quindi

$$\begin{cases} u(x) + u'(x) + u''(x) = x^3 + x - 4 \quad x \in (0, 1) \\ u(0) = 1 \\ u(1) = 0 \end{cases} \tag{1.1}$$

In questo caso l'imposizione delle condizioni al contorno è molto semplice: è sufficiente modificare la prima riga della matrice $\mathbf{A}$ in $[1, 0, \ldots, 0]$ e il primo elemento di $\mathbf{b}$ in $u(0) = 1$; mentre l'ultima riga di $\mathbf{A}$ deve essere modificata in $[0, \ldots, 0, 1]$, e l'ultimo elemento di $\mathbf{b}$ in $u(1) = 0$. Questo procedimento viene eseguito attraverso i comandi

```
A(1,1:2) = [1,0];
A(m,m-1:m) = [0,1];
b(1) = 1;
b(m) = 0;
```

A questo punto è sufficiente risolvere il sistema $\mathbf{Au} = \mathbf{b}$.

Proviamo ora a risolvere un problema con le condizioni di Neumann:

$$\begin{cases} u(x) + u'(x) + u''(x) = x^3 + x - 4 \quad x \in (0,1) \\ u'(0) = 1 \\ u'(1) = -2 \end{cases}$$

In questo caso l'imposizione delle condizioni al bordo è *leggermente* più complicata. Chiamiamo

$$\bar{u} = u'(0) \quad \text{e} \quad \tilde{u} = u'(1).$$

Abbiamo allora le seguenti discretizzazioni della derivata al primo e ultimo nodo (utilizzando due nodi fantasma u_0 ed u_{m+1})

$$u_1 + \frac{u_2 - u_0}{2h} + \frac{u_0 - 2u_1 + u_2}{h^2} = x_1^3 + x_1 - 4$$

$$u_m + \frac{u_{m+1} - u_{m-1}}{2h} + \frac{u_{m-1} - 2u_m + u_{m+1}}{h^2} = x_m^3 + x_m - 4$$

inserendo le seguenti espressioni per i nodi fantasma

$$u_0 = u_2 - 2h\bar{u} \qquad u_{m+1} = u_{m-1} + 2h\tilde{u}$$

otteniamo, come prima e ultima riga del sistema da risolvere

$$u_1 \left(1 - \frac{2}{h^2}\right) + u_2 \frac{2}{h^2} = -5 + \frac{2}{h}$$
$$u_m \left(1 - \frac{2}{h^2}\right) + u_{m-1} \frac{2}{h^2} = \frac{4}{h}$$

A titolo d'esempio, vediamo i passaggi per l'estremo sinistro:

$$\begin{aligned} u_1 + \frac{u_2 - u_0}{2h} + \frac{u_0 - 2u_1 + u_2}{h^2} &= x_1^3 + x_1 - 4 \\ \Leftrightarrow u1 + \frac{u_2 - u_2 + 2h}{2h} + \frac{u_2 - 2h - 2u_1 + u_2}{h^2} &= -4 \\ \Leftrightarrow u_1 + \frac{2h}{2h} + \frac{2u_2 - 2u_1 - 2h}{h^2} &= -4 \\ \Leftrightarrow u_1 \left(1 - \frac{2}{h^2}\right) + u_2 \left(\frac{2}{h^2}\right) &= -5 + \frac{2}{h} \end{aligned}$$

Queste condizioni vengono applicate coi seguenti comandi:

```
A(1,1:2) = [1-2/h^2,2/h^2];
A(m,m-1:m) = [2/h^2,1-2/h^2];
b(1) = -5+2/h;
b(m) = 4/h;
```

Infine, consideriamo anche un problema con condizioni di Robin (all'estremo sinistro del dominio):

$$\begin{cases} u(x) + u'(x) + u''(x) = x^3 + x - 4 \quad x \in (0,1) \\ u'(0) - u(0) = 0 \\ u(1) = 0 \end{cases}$$

Ovviamente la condizione all'estremo destro è uguale al primo esempio (1.1). Vediamo invece come applicare la condizione di Robin ($u_1 = \bar{u}$). La prima riga del sistema è

$$u_1 + \frac{u_2 - u_0}{2h} + \frac{u_0 - 2u_1 + u_2}{h^2} = x_1^3 + x_1 - 4$$

sostituendo

$$u_0 = u_2 - 2h\bar{u} = u_2 - 2hu_1$$

otteniamo

$$u_1 \left(2 - \frac{2}{h} - \frac{2}{h^2}\right) + u_2 \frac{2}{h^2} = -4$$

e dobbiamo quindi applicare le seguenti modifiche

```
A(1,1:2) = [2-2/h-2/h^2,2/h^2];
A(m,m-1:m) = [0,1];
b(1) = -4;
b(m) = 0;
```

In Figura 1.1 si nota l'ordine di convergenza dei tre "diversi" problemi.

Potrebbero capitare casi in cui la linearità del problema sia "meno intuitiva". Ad esempio nel seguente problema (che ha come soluzione esatta $u(x) = e^{x^2}$)

$$\begin{cases} \frac{u''(x)}{4x^2+2} - u(x) = 0 \quad x \in (0,1) \\ u(0) = 1 \\ u(1) = e \end{cases}$$

il "coefficiente" di u'' non è una costante ma una certa funzione di x. Il problema è comunque lineare in u, che è la nostra incognita! In questo caso la versione discretizzata del problema in riga i è

$$\frac{1}{4x_i^2 + 2} \left(\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}\right) - u_i = 0$$

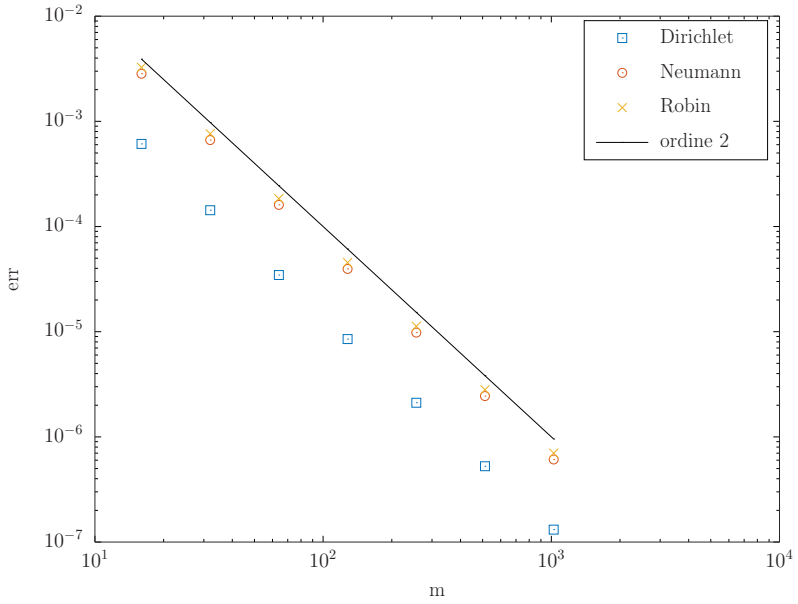


Figura 1.1: Risultati ottenuti nell'approssimazione del problema differenziale $u(x) + u'(x) + u''(x) = x^3 + x - 4$ con tre diverse condizioni al contorno. Si può notare come in tutti i casi abbiamo convergenza quadratica.

quindi possiamo scrivere il problema in forma lineare come

$$\left[\begin{array}{c|ccc|c} 1 & 0 & \cdots & 0 & 0 \\ \hline 0 & & & & 0 \\ \vdots & \frac{1}{h^2}\frac{1}{4x_i^2+2} & \frac{-2}{h^2}\frac{1}{4x_i^2+2} & \frac{1}{h^2}\frac{1}{4x_i^2+2} & \vdots \\ 0 & & & & 0 \\ \hline 0 & 0 & \cdots & 0 & 1 \end{array}\right] - \left[\begin{array}{c|ccc|c} 0 & \cdots & \cdots & \cdots & 0 \\ \hline \vdots & & & & \vdots \\ \vdots & & \tilde{I} & & \vdots \\ \vdots & & & & \vdots \\ \hline 0 & \cdots & \cdots & \cdots & 0 \end{array}\right] = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \mathrm{e} \end{bmatrix}$$

dove $\tilde{I}$ è la matrice identità di misura $(m-2) \times (m-2)$. La risoluzione del problema si implementa come segue:

```
x = linspace(0,1,m);
x = x';
xx = 1./(4*x.^2+2);
A = spdiags([[xx(2:m-1);0;0],[0;-2*xx(2:m-1);0],[0;0;xx(2:m-1)]],...
 [-1,0,1],m,m)/(h^2);
A = A - speye(m);
A(1,1) = 1;
A(m,m) = 1;
```

```
b = zeros(m,1);
b(1) = 1;
b(m) = exp(1);
u = A\b;
```

1.2 Caso non lineare

Vedremo adesso qualche esempio nel caso non lineare, e vedremo anche come verificare il corretto ordine di convergenza pur non conoscendo la soluzione analitica.
Iniziamo studiando il seguente problema:

$$\begin{cases} u''(x) + \frac{1}{8}u(x)u'(x) = 4 + \frac{1}{4}x^3 \quad x \in (1,3) \\ u(1) = 17 \\ u(3) = \frac{43}{3} \end{cases} \tag{1.2}$$

che ha come soluzione esatta

$$\hat{u}(x) = x^2 + \frac{16}{x}.$$

Vogliamo implementare un metodo di Newton: la funzione che vogliamo azzerare è

$$F(u) = \begin{bmatrix} u_1 \\ \hline \vdots \\ \frac{u_{i+1}-2u_i+u_{i-1}}{h^2} + \frac{1}{8}\left(\frac{u_{i+1}-u_{i-1}}{2h}\right)u_i \\ \vdots \\ \hline u_m \end{bmatrix} - \begin{bmatrix} 17 \\ \hline \vdots \\ 4 + \frac{1}{4}x_i^3 \\ \vdots \\ \hline \frac{43}{3} \end{bmatrix} \tag{1.3}$$

e può essere implementata come segue

```
b = 4+x.^3/4;
b(1) = 17;
b(m) = 43/3;
F = @(u) [u(1);(D2*u+1/8*((D1*u).*u))(2:m-1);u(m)] - b;
```

Se denotiamo con $\tilde{\mathbf{D}}_2$ la matrice $\mathbf{D}_2$ privata di prima e ultima riga, otteniamo che possiamo scrivere la matrice Jacobiana come

$$J(u) = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \hline & & \tilde{\mathbf{D}}_2 & & \\ \hline 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} + \frac{1}{16h}\begin{bmatrix} 0 & \cdots & & \cdots & 0 \\ \hline & -u_i & u_{i+1}-u_{i-1} & u_i & \\ \hline 0 & \cdots & & \cdots & 0 \end{bmatrix}$$

che viene implementata come segue:

```
D2(1,1:2) = [1,0];
D2(m,m-1:m) = [0,1];
Jfun = @(u) D2 + spdiags([[-u(2:m-1);0;0],[0;u(3:m)-u(1:m-2);0],...
  [0;0;u(2:m-1)]],[-1,0,1],m,m)/(16*h);
```

A questo punto prendiamo come guess iniziale una linea dal 17 a $\frac{43}{3}$ ed applichiamo il metodo di Newton esatto

```
tol = 1e-12;
maxit = 50;
it = 0;
u = linspace(17,43/3,m);
u = u';
res = -Jfun(u)\F(u);
while ((norm(res,inf)>tol) && (it<maxit))
  u = u+res;
  res = -Jfun(u)\F(u);
  it = it+1;
end
```

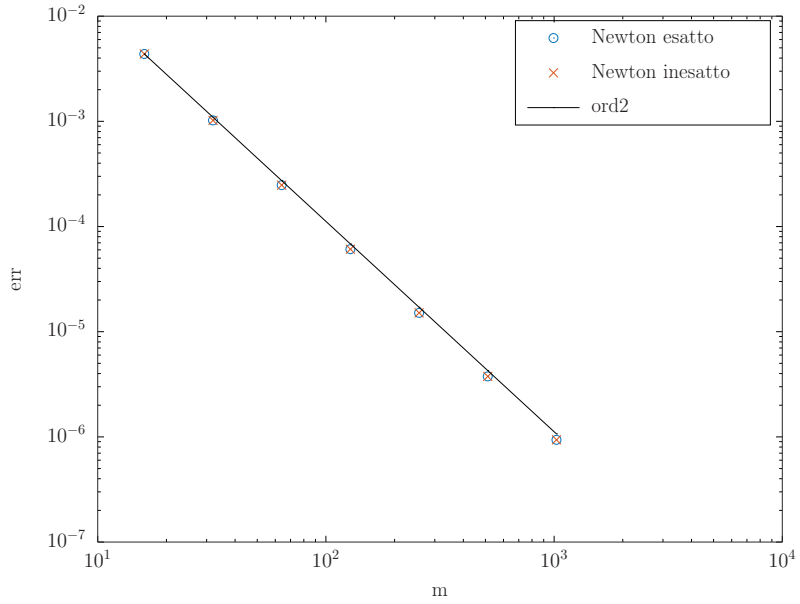
ed anche il metodo di Newton inesatto

```
it = 0;
u = linspace(17,43/3,m);
u = u';
JF = Jfun(u);
res = -JF\F(u);
while ((norm(res,inf)>tol) && (it<maxit))
  u = u+res;
  res = -JF\F(u);
  it = it+1;
end
```

I due metodi danno la stessa convergenza della soluzione (come si vede in Figura 1.2), l'unica differenza sta nel fatto che il metodo esatto usa 4 iterazioni per convergere, mentre quello inesatto ne usa 13.

Vediamo ora come implementare il metodo di Newton nel (leggermente) più complicato caso delle condizioni di Neumann. Si consideri il seguente problema:

$$\begin{cases} u(x)u'(x) + \frac{u''(x)}{2} = 0 \quad x \in (1,2) \\ u(1) = 1 \\ u'(2) = -\frac{1}{4} \end{cases}$$



Iniziamo vedendo come implementare le condizioni di Neumann: il problema discretizzato si scrive, in ultima riga,

$$u_m \left(\frac{u_{m+1} - u_{m-1}}{2h} \right) + \frac{1}{2} \left(\frac{u_{m+1} - 2u_m + u_{m-1}}{h^2} \right) = 0 \tag{1.4}$$

approssimando u_{m+1} con

$$u_{m+1} = u_{m-1} + 2h\, u'(2) = u_{m-1} + 2h \frac{-1}{4} = u_{m-1} - \frac{1}{2}h$$

otteniamo che (1.4) si riscrive come

$$u_m \left(-\frac{1}{4} - \frac{1}{h^2} \right) + u_{m-1} \left(\frac{1}{h^2} \right) - \frac{1}{4h} = 0.$$

Abbiamo quindi che la nostra funzione da azzerare è

$$F(u) = \begin{bmatrix} u_1 \\ \hline \vdots \\ u_i \left(\frac{u_{i+1} - u_{i-1}}{2h} \right) + 2 \left(\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} \right) \\ \vdots \\ \hline u_m \left(-\frac{1}{4} - \frac{1}{h^2} \right) + u_{m-1} \left(\frac{1}{h^2} \right) \end{bmatrix} - \begin{bmatrix} 1 \\ \hline \vdots \\ 0 \\ \vdots \\ \hline \frac{1}{4h} \end{bmatrix}$$

che ha, come Jacobiano

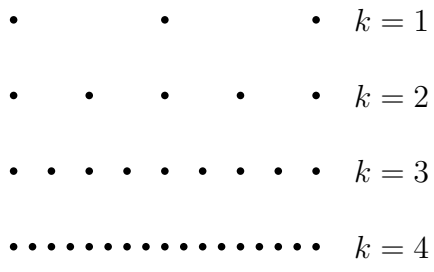
$$J(u) = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ \hline & -u_i & u_{i+1} - u_{i-1} & u_i & \\ \hline 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \frac{1}{2h} + \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \hline & & \tilde{\mathbf{D}}_2 & & \\ \hline 0 & \cdots & 0 & \frac{1}{h^2} & -\frac{1}{4} - \frac{1}{h^2} \end{bmatrix}$$

La funzione e lo Jacobiano si implementano come segue:

```
F = @(u) [u(1);(u.*(D1*u)+0.5*D2*u)(2:m-1);u(m)*(-1/4-1/h^2) + ...
  u(m-1)/h^2] - b;
D2(1,1:2) = [1,0]*2;
D2(m,m-1:m) = [1/h^2,-1/4-1/h^2]*2;
Jfun = @(u) 0.5*D2 + spdiags([[-u(2:m-1);0;0],[0;u(3:m)-u(1:m-2);0],...
  [0;0;u (2:m-1)]],[-1,0,1],m,m)/(2*h);
```

A questo punto è sufficiente implementare il metodo di Newton.

Nel caso in cui non sia possibile avere la soluzione esatta, il miglior modo per verificare l'ordine di convergenza è stimare prima una soluzione su una griglia molto fine, ed utilizzarla come soluzione esatta. Per fare ciò, bisogna stare attenti a scegliere il numero giusto di intervalli in cui dividere l'intervallo: la soluzione migliore (o, perlomeno, quella che trovo più comoda) è usare $2^k + 1$ nodi, con $k \in \mathbb{N}$. In questo modo le varie "soluzioni" si interpolano nei punti giusti, come si nota nel seguente schema:



Quindi se la soluzione di riferimento U viene stimata con M nodi, e quella approssimata u con m, abbiamo che, per interpolare correttamente, stimeremo l'errore come segue

```
err = norm(u-U(1:(M-1)/(m-1):M),inf);
```

A titolo d'esempio, consideriamo il problema

$$\begin{cases} u''(x) + \sin(u'(x)) + u(x) = x \quad x \in (0,1) \\ u(0) = 0 \\ u(1) = 1 \end{cases}$$

Questo problema si implementa con la funzione

$$F(u) = \begin{bmatrix} u_1 \\ \hline \vdots \\ \frac{u_{i+1}-2u_i+u_{i-1}}{h^2} + \sin\left(\frac{u_{i+1}-u_{i-1}}{2h}\right) + u_i - x_i \\ \vdots \\ \hline u_m - 1 \end{bmatrix}$$

che ha come Jacobiano

$$J(u) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ \hline & & \tilde{D2} & & \\ \hline 0 & 0 & \cdots & 0 & 0 \end{bmatrix} + I + \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ \hline & -\cos\left(\frac{u_{i+1}-u_{i-1}}{2h}\right) \quad 0 \quad \cos\left(\frac{u_{i+1}-u_{i-1}}{2h}\right) & & \\ \hline 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \frac{1}{2h}$$

implementate come segue

```
b = linspace(0,1,m);
b = b';
b(1) = 0;
b(m) = 1;
F = @(u) [u(1);(D2*u+sin(D1*u)+u)(2:m-1);u(m)] - b;
D2(1,1:2) = [0,0];
D2(m,m-1:m) = [0,0];
Jfun = @(u) D2 + speye(m) + spdiags([[-cos((u(3:m)-u(1:m-2))/(2*h));0;0],...
  [0;0;cos((u(3:m)-u(1:m-2))/(2*h))]],[-1,1],m,m)/(2*h);
```

1.3 Nodi non equispaziati

Se denotiamo con h_i

$$h_i = x_{i+1} - x_i$$

abbiamo che la riga i di $D1$ ha, in colonna $i-1$ ed $i+1$ rispettivamente

$$-\frac{1}{h_{i-1}+h_i}, \quad \frac{1}{h_{i-1}+h_i},$$

mentre la riga i di $D2$, in colonna $i-1, i, i+1$ avrà rispettivamente

$$\frac{2}{h_{i-1}(h_{i-1}h_i)}, \quad \frac{-2}{h_{i-1}h_i}, \quad \frac{2}{h_{i-1}(h_{i-1}h_i)}.$$

Capitolo 2

Ordinary Differential Equations

Ricordiamo lo "schema base" per descrivere una ODE: si tratta di un problema della forma

$$\begin{cases} y^{(d)} = f(t, y, y', y'', \ldots, y^{d-1}) \\ y(t_0) = y_0 \\ y'(t_0) = y_1 \\ y''(t_0) = y_2 \\ \vdots \\ y^{(d-1)}(t_0) = y_{d-1} \end{cases}$$

La strategia che useremo per l'implementazione è scrivere

$$\mathbf{y}' = f(\mathbf{y})$$

dove

$$\mathbf{y} = \begin{bmatrix} y \\ y' \\ \vdots \\ y^{(d-2)} \\ y^{(d-1)} \\ t \end{bmatrix} \qquad f = \begin{bmatrix} y' \\ y'' \\ \vdots \\ y^{(d-1)} \\ f(t, y, y', \ldots, y^{(d-1)}) \\ 1 \end{bmatrix}$$

Nel caso di un problema omogeneo ($f(t, y, y', \ldots, y^{(d)}) = f(y, y', \ldots, y^{(d)})$) è sufficiente rimuovere l'ultima riga sia da y che da f.

2.1 Eulero esplicito

Il primo metodo che vedremo è il cosiddetto *Eulero esplicito*: denotiamo con k il passo tempolare, allora il metodo si riassume nella seguente formula

$$\mathbf{y}_{n+1} = \mathbf{y}_n + kf(\mathbf{y}_n)$$

dove $\mathbf{y}_{n+1}$rappresenta il passo che dobbiamo ancora calcolare, $\mathbf{y}_n$ l'ultimo passo calcolato.
Come primo esempio vediamo un problema omogeneo del primo ordine:

$$\begin{cases} y' = 5y - 3 \\ y(2) = 1 \end{cases} \tag{2.1}$$

che ha come soluzione

$$y(t) = \frac{2}{5}e^{5(t-2)} + \frac{3}{5}$$

In questo caso l'implementazione è molto semplice e viene implementata con i seguenti comandi

```
t = linspace(2,3,m);
k = 1/(m-1);
y = zeros(1,m);
y(1) = 1;
for n = 1:m-1
  y(n+1) = y(n) + k*(5*y(n)-3);
end
```

Vediamo un problema lineare di ordine più grande:

$$\begin{cases} y''' = -3y'' - 3y' - y \\ y(0) = 7 \\ y'(0) = -7 \\ y''(0) = 11 \end{cases} \tag{2.2}$$

Il problema in forma matriciale può essere scritto nel seguente modo:

$$\begin{bmatrix} y \\ y' \\ y'' \end{bmatrix}' = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -3 & -3 \end{bmatrix} \begin{bmatrix} y \\ y' \\ y'' \end{bmatrix}$$

Quindi ad ogni iterazione calcoleremo $\mathbf{y}_{n+1}$ come segue:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + k \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -3 & -3 \end{bmatrix} \mathbf{y}_n$$

L'implementazione è la seguente:

```
k = 1/(m-1);
y(:,1) = [7;-7;11];
A = k*[0,1,0;0,0,1;-1,-3,-3];
for n=1:m-1
  y(:,n+1) = y(:,n) + A*y(:,n);
end
```

Passiamo ad un problema del primo ordine non omogeneo:

$$\begin{cases} y' = -3y + \sin(2t) \\ y(\pi) = 0 \end{cases} \tag{2.3}$$

che ha come soluzione esatta

$$y(t) = \frac{3}{13}\sin(2t) - \frac{2}{13}\cos(2t) + \frac{2}{13}e^{3\pi - 3t}$$

In questo caso abbiamo la seguente versione discretizzata del problema:

$$\begin{bmatrix} y \\ t \end{bmatrix}' = \begin{bmatrix} -3y + \sin(2t) \\ 1 \end{bmatrix}$$

con dato iniziale

$$\begin{bmatrix} y_0 \\ t_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \pi \end{bmatrix}$$

E si implementa quindi come segue:

```
t = linspace(pi,2*pi,m);
k = pi/(m-1);
f = @(y) [-3*y(1)+sin(2*y(2));1];
y = zeros(2,m);
y(:,1) = [0;pi];
for n = 1:m-1
  y(:,n+1) = y(:,n) + k*feval(f,(y(:,n)));
end
```

Consideriamo infine un problema del terzo ordine non omogeneo:

$$\begin{cases} y''' = -y'' - 4y' - 2y + 2\cos^2(t) \\ y(0) = 0 \\ y'(0) = 0 \\ y''(0) = 2 \end{cases} \tag{2.4}$$

che ha come soluzione esatta

$$y = \sin^2(t)$$

Quindi il nostro problema in forma discretizzata diventa:

$$\begin{bmatrix} y \\ y' \\ y'' \\ t \end{bmatrix}' = \begin{bmatrix} y' \\ y'' \\ -y'' - 4y' - 2y + 2\cos^2(t) \\ 1 \end{bmatrix}$$

con dato iniziale

$$\begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}$$

La soluzione si implementa come segue:

```
t = linspace(0,1,m);
k = 1/(m-1);
f = @(y) [y(2);y(3);-y(3)-4*y(2)-2*y(1)+2*cos(y(4))^2;1];
y = zeros(4,m);
y(:,1) = [0;0;2;0];
for n = 1:m-1
  y(:,n+1) = y(:,n) + k*feval(f,(y(:,n)));
end
```

I risultati di questa implementazione sono mostrati in Figura 2.1

2.2 Metodo dei trapezi

Denotiamo con $\mathbf{y}_n$ l'ultima approssimazione già calcolata e $\mathbf{y}_{n+1}$ la soluzione (da approssimare) al passo successivo. Allora il metodo dei trapezi si riassume nella formula

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{k}{2}\big(f(\mathbf{y}_n, t_n) + f(\mathbf{y}_{n+1}, t_{n+1})\big)$$

Il metodo si traduce nel risolvere

$$F(\mathbf{y}_n, \mathbf{y}_{n+1}) \doteq \mathbf{y}_{n+1} - \frac{k}{2} f(\mathbf{y}_{n+1}, t_{n+1}) - \mathbf{y}_n - \frac{k}{2} f(\mathbf{y}_n, t_n) = 0$$

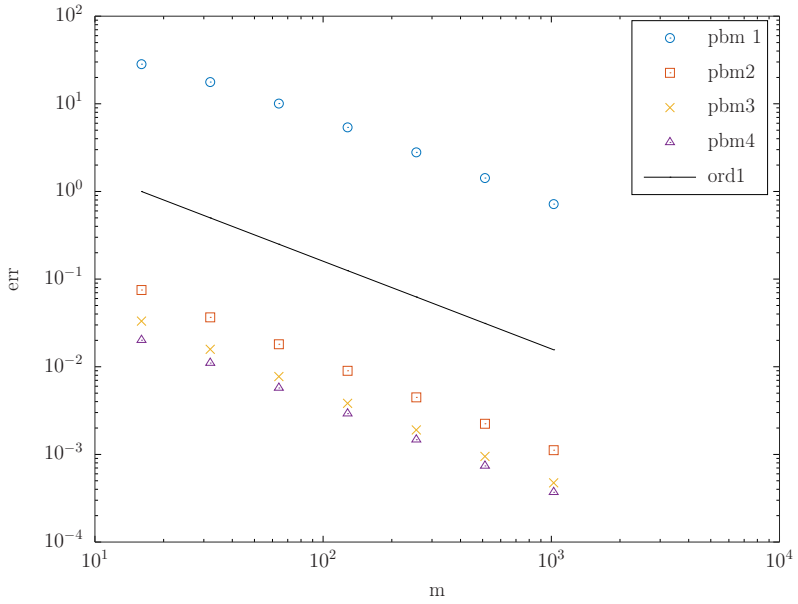


Figura 2.1: Risultati nell'approssimazione dei problemi (2.1),(2.2), (2.3), e (2.4) mediante il metodo Eulero esplicito. Si noti la convergenza di ordine 1.

Si tratta di un metodo implicito, ma questo non significa che sia sempre necessario utilizzare il metodo di Newton! Ad esempio nel caso del problema (2.1)

$$f(y) = 5y - 3$$

quindi

$$F(\mathbf{y}_{n+1}) = \mathbf{y}_{n+1} - \frac{k}{2}(5\mathbf{y}_{n+1} - 3 + 5\mathbf{y}_n - 3) - \mathbf{y}_n$$

poiché conosciamo il valore $\mathbf{y}_n$, abbiamo che la soluzione al passo successivo è

$$\mathbf{y}_{n+1} = \frac{\mathbf{y}_n\left(1 + \frac{5}{2}k\right) - 3k}{1 - \frac{5}{2}k}$$

e l'implementazione si scrive come segue:

```
for n = 1:m-1
  y(n+1) = (y(n)*(1+5*k/2)-3*k)/(1-5*k/2);
end
```

Vediamo ora il metodo dei trapezi applicato al problema (2.2). Iniziamo chiamando A la matrice

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -3 & -3 \end{bmatrix}$$

Il metodo dei trapezi ci chede quindi di risolvere, ad ogni passo,

$$\mathbf{y}_{n+1} - \frac{k}{2} A\mathbf{y}_{n+1} - \frac{k}{2} A\mathbf{y}_n - \mathbf{y}_n = 0$$

ovvero

$$\left(I - \frac{k}{2}A\right)\mathbf{y}_{n+1} = \left(I + \frac{k}{2}A\right)\mathbf{y}_n$$

dove $\mathbf{y}_{n+1}$ è l'incognita, mentre $\mathbf{y}_n$ è noto. Il metodo si implementa come segue

```
y(:,1) = [7;-7;11];
A = k/2*[0,1,0;0,0,1;-1,-3,-3];
for n=1:m-1
  y(:,n+1) = (eye(3)-A)\((eye(3)+A)*y(:,n));
end
```

Consideriamo ora il problema (2.3). La funzione da azzerare in questo caso è

$$F\left(\begin{bmatrix} y_{n+1} \\ t_{n+1} \end{bmatrix}\right) = \begin{bmatrix} y_{n+1} \\ t_{n+1} \end{bmatrix} - \frac{k}{2}\left(\begin{bmatrix} -3y_{n+1} + \sin(2t_{n+1}) \\ 1 \end{bmatrix} + \begin{bmatrix} -3y_n + \sin(2t_n) \\ 1 \end{bmatrix}\right) - \begin{bmatrix} y_n \\ t_n \end{bmatrix}$$

Potrebbe sembrare necessario l'utilizzo del metodo di Newton, ma così non è poiché l'unica parte non lineare è una funzione di t, di cui conosciamo il valore ad ogni step. La funzione da azzerare è quindi

$$F(y_{n+1}) = y_{n+1} - \frac{k}{2}\big(-3y_{n+1} + \sin(2t_{n+1}) - 3y_n + \sin(2t_n)\big) - y_n$$

che ha valore zero per

$$y_{n+1} = -\frac{\frac{k}{2}(-\sin(2t_{n+1}) + 3y_n - \sin(2t_n))}{1 + \frac{3k}{2}} - y_n$$

e si implementa come segue:

```
t = linspace(pi,2*pi,m);
k = pi/(m-1);
y = zeros(1,m);
y(1) = 0;
for n = 1:m-1
  y(n+1) = -(k/2*(-sin(2*t(n+1))+3*y(:,n)-sin(2*t(n))) - y(:,n))/(1+3*k/2);
end
```

Vediamo come risolvere numericamente il problema (2.4). Dobbiamo trovare lo zero della seguente funzione

$$F\left(y_{n+1}\right) = \begin{bmatrix} \\ \mathbf{y}_{n+1} \\ \\ \end{bmatrix} - \frac{k}{2}\left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -4 & -1 \end{bmatrix}\begin{bmatrix} \\ \mathbf{y}_{n+1} \\ \\ \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 2\cos^2(t_{n+1}) \end{bmatrix}\right) + \dots$$
$$- \begin{bmatrix} \\ \mathbf{y}_{n} \\ \\ \end{bmatrix} - \frac{k}{2}\left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -4 & -1 \end{bmatrix}\begin{bmatrix} \\ \mathbf{y}_{n} \\ \\ \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 2\cos^2(t_{n}) \end{bmatrix}\right)$$

Se chiamiamo A la matrice

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -4 & -1 \end{bmatrix}$$

il problema si traduce nel risolvere il seguente sistema

$$-\left(I - \frac{k}{2}A\right)\begin{bmatrix} \\ \mathbf{y}_{n+1} \\ \\ \end{bmatrix} = -k\begin{bmatrix} 0 \\ 0 \\ \cos^2(t_N) \end{bmatrix} - \begin{bmatrix} \\ \mathbf{y}_{n} \\ \\ \end{bmatrix} - k\begin{bmatrix} 0 \\ 0 \\ \cos^2(t_O) \end{bmatrix} - \frac{k}{2}A\begin{bmatrix} \\ \mathbf{y}_{n} \\ \\ \end{bmatrix}$$

E si implementa come segue:

```
t = linspace(0,1,m);
k = 1/(m-1);
y = zeros(3,m);
y(:,1) = [0;0;2];
A = [0,1,0;0,0,1;-2,-4,-1];
for n = 1:m-1
  b = -k*[0;0;cos(t(n+1))^2] - y(:,n)-k*[0;0;cos(t(n))^2] - k/2*A*y(:,n);
  B = -(eye(3)-k/2*A);
  y(:,n+1) = B\b;
end
```

I risultati di queste approssimazioni sono riportate in Figura 2.2

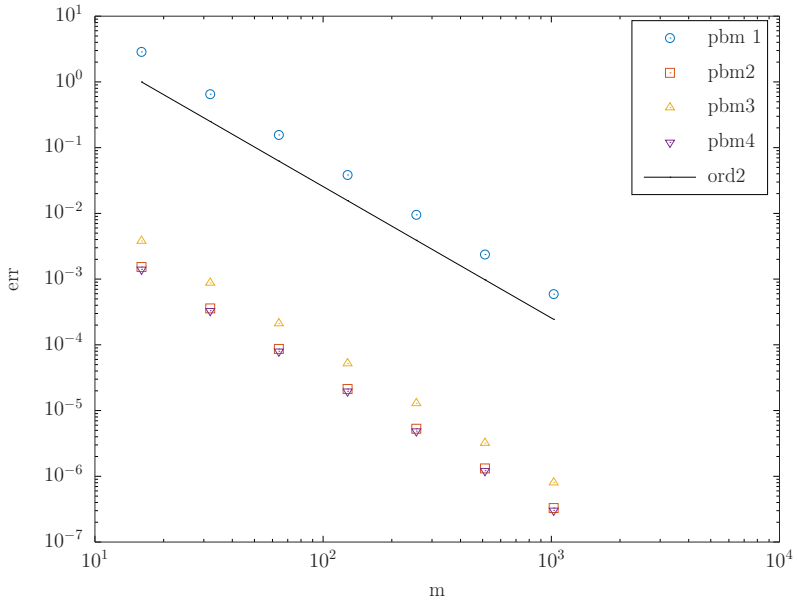


Figura 2.2: Risultati nell'approssimazione dei problemi (2.1),(2.2), (2.3), e (2.4) mediante il metodo dei trapezi. Si noti la convergenza di ordine 2.

2.2.1 Un problema non lineare

Vediamo ora come applicare il metodo di Eulero ed il metodo dei trapezi a un problema non lineare. Si consideri il seguente problema

$$\begin{cases} y''' = 96\frac{y}{y'} \\ y(1) = 1 \\ y'(1) = 4 \\ y''(1) = 12 \end{cases} \tag{2.5}$$

L'implementazione del metodo di Eulero è molto semplice:

```
y_E = zeros(3,m);
y_E(:,1) = [1;4;12];
f = @(y) [y(2);y(3);96*y(1)/y(2)];
for n = 1:m-1
  y_E(:,n+1) = y_E(:,n) + k*f(y_E(:,n));
end
```

Il problema si traduce, per il metodo dei trapezi,nell'azzerare

$$F\left(\begin{bmatrix}\\ \mathbf{y}_{n+1}\\ \\ \end{bmatrix}\right) = \begin{bmatrix}\\ \mathbf{y}_{n+1}\\ \\ \end{bmatrix} - \frac{k}{2}\begin{bmatrix} \mathbf{y}_{n+1}^{(2)} \\ \mathbf{y}_{n+1}^{(3)} \\ 96\frac{\mathbf{y}_{n+1}^{(1)}}{\mathbf{y}_{n+1}^{(2)}} \end{bmatrix} - \begin{bmatrix}\\ \mathbf{y}_{n}\\ \\ \end{bmatrix} - \frac{k}{2}\begin{bmatrix} \mathbf{y}_{n}^{(2)} \\ \mathbf{y}_{n}^{(3)} \\ 96\frac{\mathbf{y}_{n}^{(1)}}{\mathbf{y}_{n}^{(2)}} \end{bmatrix}$$

In questo caso si rende necessario il metodo di Newton. Calcoliamo quindi lo Jacobiano

$$J = I - \frac{k}{2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 96\frac{1}{\mathbf{y}_{n+1}^{(2)}} & -96\frac{\mathbf{y}_{n+1}^{(1)}}{\left(\mathbf{y}_{n+1}^{(2)}\right)^2} & 0 \end{bmatrix}$$

Il metodo si implementa come segue:

```
F = @(yo,yn) yn - k/2*f(yn) - yo - k/2*f(yo);
JF = @(yn) eye(3) -k/2*[0,1,0;0,0,1;96/yn(2),-96*yn(1)/(yn(2)^2),0];
y_T = zeros(3,m);
y_T(:,1) = [1;4;12];
maxit = 30;
tol = 1e-10;
for n = 1:m-1
  y_T(:,n+1) = y_T(:,n);
  res = -JF(y_T(:,n+1))\F(y_T(:,n),y_T(:,n+1));
  it = 0;
  while it<maxit && norm(res)>tol
    y_T(:,n+1) = y_T(:,n+1) + res;
    res = -JF(y_T(:,n+1))\F(y_T(:,n),y_T(:,n+1));
    it = it+1;
  end
end
```

In Figura 2.3 si possono osservere i risultati ottenuti con questi metodi

2.3 Una generalizzazione: il θ-metodo

Il θ-metodo si riassume nella seguente formula

$$\mathbf{y}_{n+1} = \mathbf{y}_n + k(1-\theta)f(\mathbf{y}_n, t_n) + k\theta f(\mathbf{y}_{n+1}, t_{n+1}) \quad \theta \in [0,1]$$

o, equivalentemente, nell'azzerare la funzione

$$F(\mathbf{y}_{n+1}) = \mathbf{y}_{n+1} - \mathbf{y}_n - k(1-\theta)f(\mathbf{y}_n, t_n) - k\theta f(\mathbf{y}_{n+1}, t_{n+1})$$

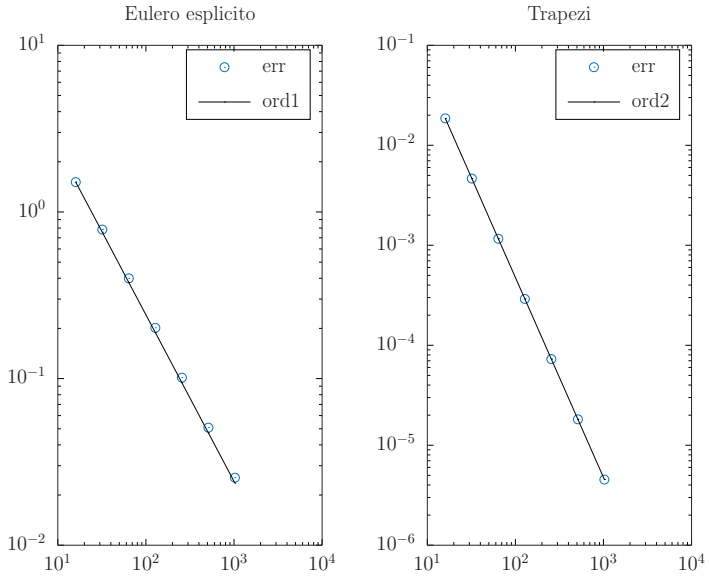


Figura 2.3: Risultati nell'approssimazione del problema (2.5) mediante il metodo di Eulero ed il metodo dei trapezi.

Prendiamo come esempio il problema (2.4). La formula del θ-metodo diventa

$$y_{n+1} = \mathbf{y}_n + k(1-\theta)(A\mathbf{y}_n + T_n) + k\theta(A\mathbf{y}_{n+1} + T_{n+1})$$

con

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -4 & -1 \end{bmatrix} \qquad T_n = \begin{bmatrix} 0 \\ 0 \\ \cos^2(t_n) \end{bmatrix}$$

quindi

$$(I - k\theta A)\mathbf{y}_{n+1} + (I + k(1-\theta)A)\mathbf{y}_n + k\theta T_{n+1} + (1-k\theta)T_n$$

Si noti che per $\theta = 0$ abbiamo il metodo di Eulero, per $\theta = 0.5$ abbiamo il metodo dei trapezi, mentre per $\theta = 1$ abbiamo il metodo di Eulero implicito. Il metodo viene implementato come segue

```
for n = 1:m-1
  b = k*theta*[2*cos(t(n+1))^2;0;0] + (1-theta)*k*[2*cos(t(n))^2;0;0] +...
    (eye(3)+k*(1-theta)*A)*y(:,n);
  B = eye(3)-k*theta*A;
  y(:,n+1) = B\b;
end
```

In Figura 2.4 si vede l'ordine di convergenza per diversi valori di θ.

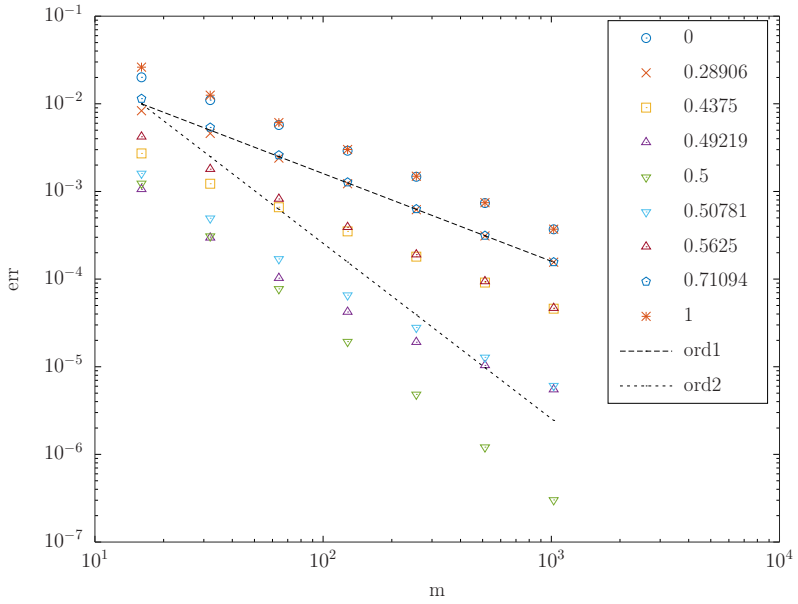


Figura 2.4: Risultati nell'approssimazione del problema (2.4) mediante il θ-metodo: in legenda vengono riportati i vari valori di θ. Si noti che sono tutti di ordine 1, tranne che per il valore 0.5 (metodo dei trapezi).

2.4 Metodi Multistep

Si dicono metodi multistep quei metodi che utilizzano, per approssimare la soluzione a un certo punto t_n, non solo la soluzione al tempo t_{n-1}, ma anche le soluzioni di tempi precedenti.

2.4.1 Adams-Bashforth

Il metodo si riassume nella formula

$$y_{n+s} = y_{n+s-1} + k \sum_{j=0}^{s-1} b_j \, f(t_{n+j}, y_{n+j})$$

dove

$$b_j = \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^{s-1} \frac{s-1-i+r}{j-1} \mathrm{d}r$$

Vediamo alcuni valori

s	b_0	b_1	b_2	b_3	b_4
2	-1/2	3/2			
3	5/12	-4/3	23/12		
4	-3/8	37/24	-59/24	55/24	
5	251/720	-637/360	109/30	-1387/360	1901/720

Vediamo ad esempio come risolvere il problema (2.4) con $s = 3$:

```
b0 = 5/12;
b1 = -4/3;
b2 = 23/12;
T = linspace(0,1,m);
k = 1/(m-1);
count++;
Y = NaN(3,m);
Y(:,1) = [0,0,2];
Y(:,2) = (eye(3) - k/2*A)\((k/2*A+eye(3))*Y(:,1) + ...
  k*[0;0;cos(T(1))^2 + cos(T(2))^2]);
Y(:,3) = Y(:,2) + k *(-0.5*f(T(1),Y(:,1)) + 1.5*f(T(2),Y(:,2)));
for n = 3:m-1
Y(:,n+1) = Y(:,n) + k*(b0*f(T(n-2),Y(:,n-2)) + ...
  b1*f(T(n-1),Y(:,n-1)) + b2*f(T(n),Y(:,n)));
end
```

2.4.2 Adams-Multon

Si può generalizzare il metodo Adams-Bashfort rendendolo implicito:

$$y_{n+s} = y_{n+s-1} + k \sum_{j=0}^{s} b_j f(t_{n+j}, y_{n+j})$$

dove l'ordine massimo ($\mathcal{O}(k^{s+1})$) è raggiunto dal metodo Adams-Multon, con

s	b_0	b_1	b_2	b_3	b_4
0	1				
1	1/2	1/2			
2	-1/12	2/3	5/12		
3	1/24	-5/24	19/24	3/8	
4	-19/720	106/720	-264/720	646/720	251/720

Consideriamo ora il problema (2.5). Si tratta di risolvere

$$\mathbf{y}_{n+2} - \mathbf{y}_{n+1} - k(b_0 f(\mathbf{y}_n) + b_1 f(\mathbf{y}_{n+1}) + b_2 f(\mathbf{y}_{n+2})) = 0$$

dove

$$f(\mathbf{y}) = f\left(\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \end{bmatrix}\right) = \begin{bmatrix} y^{(2)} \\ y^{(3)} \\ 96\frac{y^{(1)}}{y^{(2)}} \end{bmatrix}$$

È chiaramente un problema non lineare nell'incognita $\mathbf{y}_{n+1}$, e dovremmo quindi usare il metodo di Newton: l'implementazione si svolge come segue (dove, per calcolare il secondo passo[1] usiamo il metodo dei trapezi)

```
% Trapezi per il primo passo
Ftrap = @(yn,yo) yn - 0.5*k*f(yn) - yo - 0.5*k*f(yo);
Jtrap = @(yn) eye(3) - 0.5*k*j(yn);
% Forma generale del metodo multistep
b0 = -1/12;
b1 = 2/3;
b2 = 5/12;
FF =  @(yn2,yn1,yn) yn2-(yn1+k*(b0*f(yn)+b1*f(yn1)+b2*f(yn2)));
JJ = @(yn2) eye(3) -k*b2*j(yn2);
% %
Y = NaN(3,m);
Y(:,1) = [1;4;12];
yn = ones(3,1);
yo = Y(:,1);
res = -Jtrap(yn)\Ftrap(yn,yo);
it = 0;
while it<maxit && norm(res)>tol
yn = yn + res;
res = -Jtrap(yn)\Ftrap(yn,yo);
it++;
endwhile
yn = yn + res;
Y(:,2) = yn;
% Inizio Multon
for n = 1:m-2
yn = Y(:,n);
yn1 = Y(:,n+1);
yn2 = Y(:,n+1); %initial guess
res = -JJ(yn2)\FF(yn2,yn1,yn);
it = 0;
  while it<maxit && norm(res)>tol
```

[1] il primo è, chiaramente, il dato iniziale

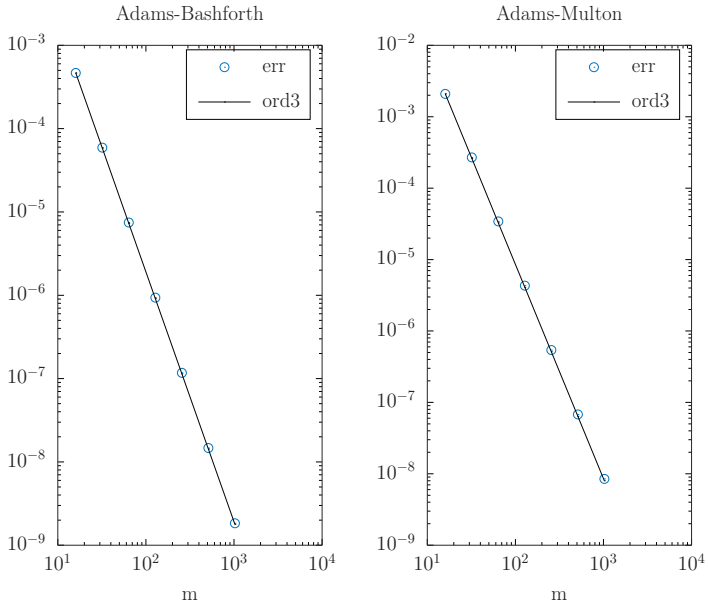


Figura 2.5: Convergenza per i metodi multistep di Adams-Bashforth e Adams-Multon.

```
    yn2 = yn2 + res;
    res = -JJ(yn2)\FF(yn2,yn1,yn);
    it++;
  endwhile
Y(:,n+2) = yn2;
endfor
```

In Figura (2.5) si vede il corretto ordine di convergenza

2.5 Metodi Runge-Kutta

Nella forma più generale, un metodo RK può essere scritto come

$$\mathbf{y}_{n+1} = \mathbf{y}_n + kF(t_n, \mathbf{y}_n, h; f), \quad n \geq 0 \tag{2.6a}$$

dove F è la funzione incremento

$$F(t_n, \mathbf{y}_n, h; f) = \sum_{i=1}^{s} b_i K_i \tag{2.6b}$$

dove i coefficienti K_i sono dati da

$$K_i = f\left(t_n + c_i h, \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij} K_j\right), \tag{2.6c}$$

in cui s è il numero di stadi del metodo e i coefficienti a_{ij}, b_i ed c_i vengono raccolti nella *matrice di Butcher*

$$\begin{array}{c|cccc} c_1 & a_{1\,1} & a_{1\,2} & \cdots & a_{1\,s} \\ c_2 & a_{2\,1} & a_{2\,2} & \cdots & a_{2\,s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s\,1} & a_{s\,2} & \cdots & a_{s\,s} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \quad \text{o} \quad \begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$$

Equivalenetemente alla formulazione (2.6), si può scrivere un metodo Runge-Kutta utilizzando la seguente formulazione:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + k\sum_{j=1}^{s} b_j f(t_n + kc_j, \xi_j),$$

con

$$\xi_i = \mathbf{y}_n + k\sum_{j=1}^{s} a_{ij}\mathbf{f}\left(t_n + kc_j, \xi_j\right).$$

È tuttavia preferibile utilizzare la formulazione (2.6) in quanto permette di utilizzare il metodo evitando di calcolare la funzione f negli stessi punti.

Se $a_{ij} = 0$ per $j \geq 0$, per ogni $i = 1, 2, \ldots, s$, i coefficienti K_i possono essere calcolati esplicitamente e parleremo allora di *metodo Runge Kutta esplicito*. In caso contrario, il metodo RK è detto *implicito*, in questo caso, per trovare i coefficienti K_i si dovrebbe risolvere un sistema non lineare e sarebbe oneroso; quindi considereremo solo un tipo di RK impliciti, i metdodi *semi-impliciti*, in cui $a_{ij} = 0$ se $j > i$ e quindi ogni K_i è dato dalla soluzione dell'equazione non lineare

$$K_i = f\left(t_n + c_i h, \mathbf{y}_n + h a_{ii} K_i + h\sum_{j=1}^{i-1} a_{ij} K_j\right)$$

Supporremo sempre le condizioni

$$c_i = \sum_{j=1}^{s} a_{ij} \qquad \sum_{i=1}^{s} b_i = 1.$$

Nel caso di un metodo semi-implicito, l'unica differenza sta nel fatto che per approssimare `K(i)` bisognerà usare, forse ma *non necessariamente*, il metodo di Newton.

Consideriamo il tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{2.7}$$

per un metodo esplicito ed il tableau

$$\begin{array}{c|cc} \frac{3+\sqrt{3}}{6} & \frac{3+\sqrt{3}}{6} & \\ \frac{3-\sqrt{3}}{6} & -\frac{\sqrt{3}}{3} & \frac{3+\sqrt{3}}{6} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{2.8}$$

per un metodo semiimplicito. Vediamo come utilizzare il metodo dato dal tableau (2.7) per il problema (2.4). Per prima cosa definiamo i dati

```
A = sparse([2,3,4],[1,2,3],[0.5,0.5,1],4,4);
b = [1,2,2,1]/6;
c = [0,1,1,2]/2;
y0 = [0;0;2];
AA = [0,1,0;0,0,1;-2,-4,-1];
f = @(t,y) AA*y + [0;0;2*cos(t)^2];
```

con questi, risolviamo il problema cose segue

```
for n = 1:m-1
  K = zeros(3,4);
  K(:,1) = feval(f,T(n),Y(:,n));
  K(:,2) = feval(f,T(n) + h*c(2),Y(:,n) + h*A(2,1)*K(:,1));
  K(:,3) = feval(f,T(n) + h*c(3),Y(:,n) + ...
    h*(A(3,1)*K(:,1)+A(3,2)*K(:,2)));
  K(:,4) = feval(f,T(n) + h*c(4),Y(:,n) + ...
    h*(A(4,1)*K(:,1)+A(4,2)*K(:,2)+A(4,3)*K(:,3)));
  Y(:,n+1) = Y(:,n) + h*K*b';
endfor
```

Nel caso invece volessimo usare il tableau (2.8), il calcolo dei coefficienti K_i si implementa come segue

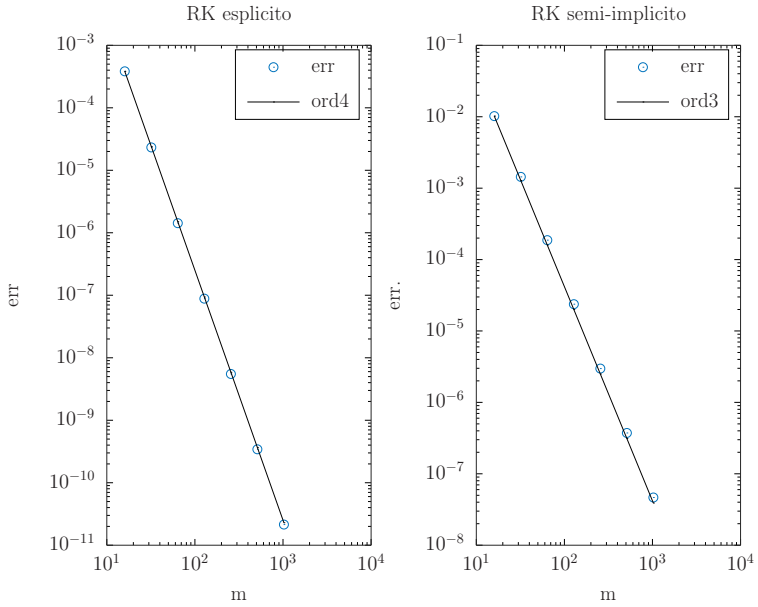


Figura 2.6: Convergenza per il metodo di tableau (2.7) e (2.8) per la risoluzione del problema (2.4)

```
K = zeros(3,2);
K(:,1) = (eye(3)-h*A(1,1)*AA)\(AA*Y(:,n)+[0;0;2*cos(T(n)+h*c(1))^2]);
K(:,2) = (eye(3)-h*A(2,2)*AA)\(AA*(Y(:,n)+h*A(2,1)*K(:,1))+...
  [0;0;2*cos(T(n)+c(2)*h)^2]);
```

Si osservi che, nonostante il metodo sia (semi)implicito, non è stato necessario utilizzare il metodo di Newton.
In Figura 2.6 viene mostrato il corretto ordine di convergenza di questi due metodi.

2.6 Integratori esponenziali

Consideriamo il sistema

$$\begin{cases} \mathbf{y}(t) = A\mathbf{y}(t) + \mathbf{b}(t, \mathbf{y}(t)), & t > 0 \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}$$

il metodo Eulero esponenziale è

$$\begin{aligned} \mathbf{y}_{n+1} &= \exp(kA)\mathbf{y}_n + k\varphi_1(kA)\mathbf{b}(t_n, \mathbf{y}_n) \\ &= \mathbf{y}_n + k\varphi_1(kA)(A\mathbf{y}_n + \mathbf{b}(t_n, y_n) \end{aligned}$$

Consideriamo il problema lineare non autonomo

$$\begin{cases} \mathbf{y}(t) = A\mathbf{y}(t) + \mathbf{b}(t), \quad t > 0 \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}$$

il metodo esponenziale-punto medio è

$$\begin{aligned} \mathbf{y}_{n+1} &= \exp(kA)\mathbf{y}_n + k\varphi_1(kA)\mathbf{b}(t_n + k/2) \\ &= \mathbf{y}_n + k\varphi_1(kA)(A\mathbf{y}_n + \mathbf{b}(t_n + k/2)) \end{aligned}$$

Per un problema differenziale in forma autonoma

$$\begin{cases} \mathbf{y}(t) = f(\mathbf{y}(t)), \quad t > 0 \\ \mathbf{y}(t_0) = \mathbf{y}_0 \end{cases}$$

il metodo di Eulero-Rosenbrock esponenziale è

$$\begin{aligned} \mathbf{y}_{n+1} &= \exp(kJ_n)\mathbf{y}_n + k\varphi_1(kJ_n)\mathbf{b}_n(\mathbf{y}_n) \\ &= \mathbf{y}_n + k\varphi_1(kJ_n)\mathbf{f}(\mathbf{y}_n)) \end{aligned}$$

dove

$$\mathbf{b}_n(\mathbf{y}(t)) = \mathbf{f}(\mathbf{y}(t)) - J_n\mathbf{y}(t).$$

Si noti che dato un problema in forma non autonoma

$$\mathbf{y}' = f(\mathbf{y}) + g(t)$$

si può passare ad una forma autonoma ponendo

$$\begin{bmatrix} \tilde{\mathbf{y}} \end{bmatrix} \doteq \left[\begin{array}{c} \mathbf{y} \\ \hline t \end{array}\right]$$