Serie 8 - Soluzione
Problemi ai limiti e ai valori iniziali

Esercizio 1.1

1. *Si verifichi che $u_{ex}(x)$ è la soluzione esatta del problema di diffusione con i dati indicati.*

   Iniziamo verificando che $-\mu\, u''_{ex}(x) = f(x)$ per ogni $x \in (0,1)$. Essendo $\mu = 1$, $u'_{ex}(x) = e^{3x}(1 + x - 3x^2) - 3$, otteniamo che $u''_{ex}(x) = -f(x)$ per ogni $x \in (0,1)$. Verifichiamo anche il soddisfacimento delle condizioni al contorno di Dirichlet, ovvero che $u_{ex}(0) = 1$ e $u_{ex}(1) = -2$.

2. *Si ricavino le espressioni generali della matrice A e del vettore $\mathbf{b}$ che compaiono nel sistema lineare condensato $A\,\mathbf{u}_h = \mathbf{b}$ e ottenuto mediante approssimazione con le differenze finite centrate.*

   Abbiamo:

$$A = \frac{\mu}{h^2}\,\text{tridiag}(-1, 2, -1) \qquad \text{e} \qquad \mathbf{b} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix} + \begin{bmatrix} \alpha\,\mu/h^2 \\ 0 \\ \vdots \\ 0 \\ \beta\,\mu/h^2 \end{bmatrix}.$$

3. *Si scriva uno script o funzione Matlab® per risolvere il problema ai limiti ponendo $(N + 1) = 20$. Confrontare in un grafico la soluzione approssimata con quella esatta u_{ex}.*

   Consideriamo i seguenti comandi Matlab® per ottenere il risultato di Figura 1.

```
a = 0;  b = 1;
alpha = 1;  beta = -2;
mu = 1;
f = @( x ) exp( 3 * x ) .* ( - 4 + 3 * x + 9 * x.^2 );
uex = @( x ) exp( 3 * x ) .* ( x - x.^2 ) + 1 - 3 * x;

N = 20 - 1;
h = ( b - a ) / ( N + 1 );
xnodes = linspace( a, b, N + 2 );

A = sparse( 1 : N, 1 : N, 2, N, N ) ...
    + sparse( 2 : N, 1 : N - 1, -1, N, N ) + sparse( 1 : N - 1, 2 : N, -1, N, N );
A = mu / h^2 * A;

bv = ( f( xnodes( 2 : end - 1 ) ) )';
bv( 1 ) = bv( 1 ) + alpha * mu / h^2;
bv( end ) = bv( end ) + beta * mu / h^2;
```

```
uh = A \ bv;

uh = [ alpha; uh; beta ];

xplot = linspace( a, b, 1001 );
figure( 1 );
plot( xplot, uex( xplot ), '-b', xnodes, uh, 'xr', 'LineWidth', 2, 'MarkerSize', 10 );
grid on;        xlabel( 'x' );      ylabel( 'uex, uh ')
legend( 'u_{ex}', 'uh DF centrate', 'location', 'Best' );
```

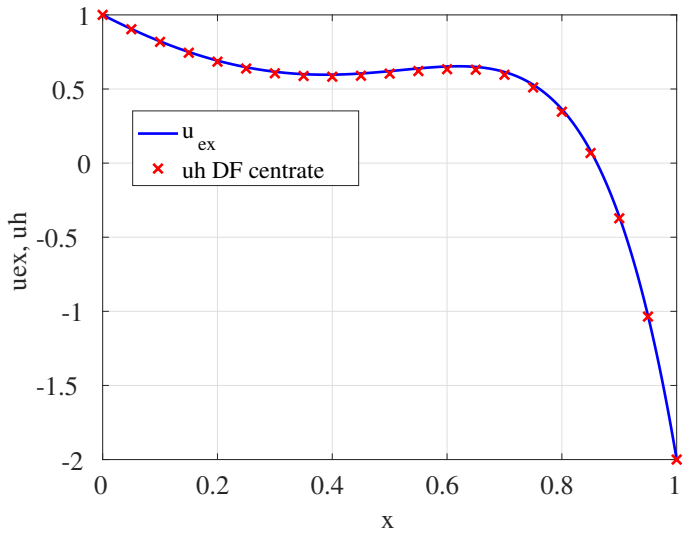

Figura 1: Esercizio 1.1, punto 3. Confronto tra soluzione esatta $u_{ex}(x)$ e soluzione approssimata $\mathbf{u}_h$

4. *Si risolva ora il problema per* $(N+1) = 10, 20, 40, 80, \ldots, 160$. *Si rappresenti graficamente, in funzione di* h, *l'andamento dell'errore* $e_h = \max_{j=0,\ldots,N+1} |u_{ex}(x_j) - u_j|$. *Si confronti il risultato ottenuto con quanto previsto dalla teoria.*

   Consideriamo i seguenti comandi Matlab® per ottenere il grafico dell'andamento dell'errore e_h vs. h in Figura 2.

```
Nv = [ 10 20 40 80 160 ] - 1;
hv = ( b - a ) ./ ( Nv + 1 );
errv = [];
for N = Nv

    xnodes = linspace( a, b, N + 2 );
    h = ( b - a ) / ( N + 1 );

    A = sparse( 1 : N, 1 : N, 2, N, N ) ...
        + sparse( 2 : N, 1 : N - 1, -1, N, N ) + sparse( 1 : N - 1, 2 : N, -1, N, N );
    A = mu / h^2 * A;

    bv = ( f( xnodes( 2 : end - 1 ) ) )';
```

```
    bv( 1 ) = bv( 1 ) + alpha * mu / h^2;
    bv( end ) = bv( end ) + beta * mu / h^2;

    uh = A \ bv;

    uh = [ alpha; uh; beta ];

    errv = [ errv, max( abs( uh - uex( xnodes' ) ) ) ];
end

figure( 2 )
loglog( hv, errv, '-x', hv, hv.^2, '--k', 'LineWidth', 2, 'MarkerSize', 10 );
xlabel( ' h [log]' );   ylabel( 'err [log]')
legend( 'eh', '(h,h^2)', 'Location', 'Best' ); grid on
```

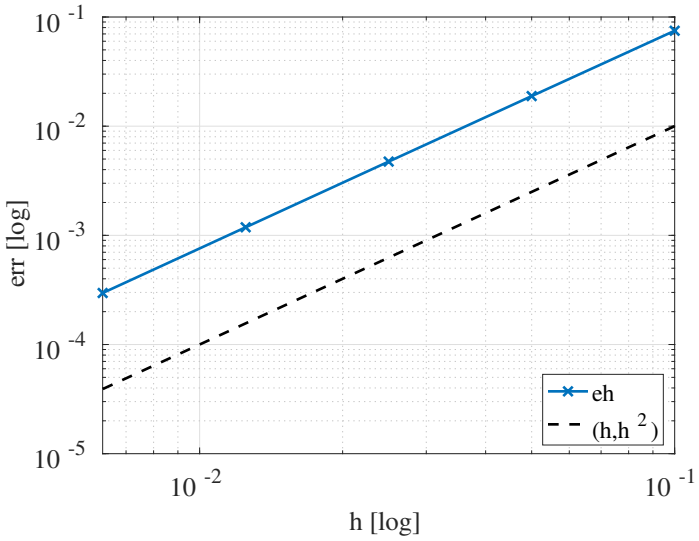

Figura 2: Esercizio 1.1, punto 4. e_h vs. h, scala logaritmica su entrambi gli assi

Osserviamo che l'andamento dell'errore e_h vs. h in scala logaritmica per entrambi gli assi risulta pari a una retta parallela a (h, h^2), ovvero l'errore converge con ordine 2 rispetto ad h. Il risultato è in accordo con la teoria, infatti $e_h \leq C\, h^2$ se $f \in C^2([0,1])$ (ovvero $u \in C^4([0,1])$), come effettivamente avviene in questo caso.

5. *Come varia il numero di condizionamento $K_2(A)$ per $(N+1) = 10, 20, 40, 80, \ldots, 160$, ovvero per $h \to 0$? Con quale metodo numerico è computazionalmente conveniente risolvere il sistema lineare condensato? Posto $(N+1) = 160$, come si confronta l'errore computazionale associato alla soluzione del sistema lineare tramite metodo diretto con l'errore di troncamento e_h associato allo schema alle differenze finite?*

   Sappiamo che $K_2(A) = \dfrac{C}{h^2}$, come possiamo verificare con i seguenti comandi Matlab® e da Figura 3.

```
K2v = [];
```

```
for N = Nv
    h = ( b - a ) / ( N + 1 );

    A = sparse( 1 : N, 1 : N, 2, N, N ) ...
        + sparse( 2 : N, 1 : N - 1, -1, N, N ) + sparse( 1 : N - 1, 2 : N, -1, N, N );
    A = mu / h^2 * A;
    K2v = [ K2v, condest( A ) ];
end
figure( 3 )
loglog( hv, K2v, '-x', hv, hv.^(-2), '--k', 'LineWidth', 2, 'MarkerSize', 10 );
xlabel( ' h [log]' );  ylabel( 'K2(A) [log]')
legend( 'K2(A)', '(h,1/h^2)', 'Location', 'Best' );  grid on
```

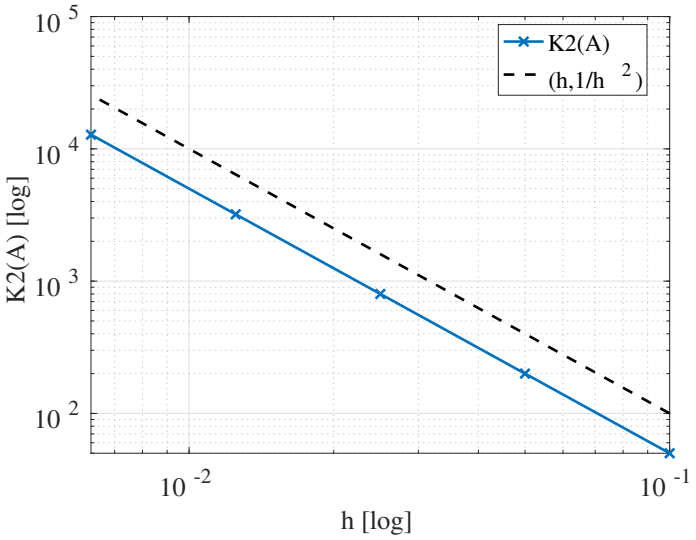

Figura 3: Esercizio 1.1, punto 5. $K_2(A)$ vs. h e (h, h^{-2}), scala logaritmica su entrambi gli assi

Dato che la matrice A è tridiagonale, il metodo computazionalmente più vantaggioso per risolvere il sistema lineare condensato è l'algoritmo di Thomas, che impiega $8N - 7$ operazioni.

Verifichiamo che l'errore computazionale stimato associato alla soluzione del sistema lineare tramite metodo diretto è inferiore all'errore di troncamento e_h associato allo schema numerico delle differenze finite per questo valore di h. Ricordiamo che l'errore computazionale può essere stimato tramite la seguente stima a posteriori $\|\widehat{\mathbf{u}}_h - \mathbf{u}_h\| \leq K_2(A) \dfrac{\|\mathbf{b} - A\mathbf{u}_h\|}{\|\mathbf{b}\|} \|\mathbf{u}_h\|$. Usiamo i seguenti comandi Matlab® :

```
bv = ( f( xnodes( 2 : end - 1 ) ) )';
bv( 1 ) = bv( 1 ) + alpha * mu / h^2;
bv( end ) = bv( end ) + beta * mu / h^2;
uh = A \ bv;
eh = max( abs( uh - uex( xnodes( 2 : end - 1 )' ) ) )
eh =
```

```
   2.9640e-04
err_sist_lin_stimato = condest( A ) * norm( bv - A * uh ) / norm( bv ) * norm( uh )
err_sist_lin_stimato =
   5.6609e-11
```

Abbiamo dunque verificato che per questo valore di h, l'errore computazionale è trascurabile rispetto all'errore e_h.

Esercizio 1.2

1. *Si verifichi che $u_{ex}(x)$ è la soluzione esatta del problema di diffusione–reazione con i dati indicati.*

   Iniziamo verificando che $-\mu\, u''_{ex}(x) + \sigma(x)\, u_{ex}(x) = f(x)$ per ogni $x \in (0,1)$. Essendo $\mu = 1$, $\sigma(x) = 1 + \sin(2\pi x)$, $u'_{ex}(x) = 3e^{3x}$, otteniamo che $u''_{ex}(x) - \sigma(x)\, u_{ex}(x) = -f(x)$ per ogni $x \in (0,1)$. Verifichiamo anche il soddisfacimento delle condizioni al contorno di Dirichlet, ovvero che $u_{ex}(0) = 1$ e $u_{ex}(1) = e^3$.

2. *Si ricavino le espressioni generali della matrice A e del vettore $\mathbf{b}$ che compaiono nel sistema lineare condensato e ottenuto mediante approssimazione con le differenze finite centrate.*

   Rispetto all'Esercizio 1.1, abbiamo:

   $$A = \frac{\mu}{h^2}\,\mathrm{tridiag}(-1, 2, -1) + \mathrm{diag}\left((\sigma(x_1), \ldots, \sigma(x_N))^T \right).$$

3. *Si scriva uno script o funzione Matlab® per risolvere il problema ai limiti ponendo $(N+1) = 20$. Confrontare in un grafico la soluzione approssimata con quella esatta u_{ex}.*

   Consideriamo i seguenti comandi Matlab® per ottenere il risultato di Figura 4.

```
a = 0;      b = 1;
alpha = 1;      beta = exp( 3 );
mu = 1;
sigma = @( x ) 1 + sin( 2 * pi * x );
f = @( x ) exp( 3 * x ) .* ( sin( 2 * pi * x ) - 8 );
uex = @( x ) exp( 3 * x );

N = 20 - 1;
h = ( b - a ) / ( N + 1 );
xnodes = linspace( a, b, N + 2 );

A = sparse( 1 : N, 1 : N, 2, N, N ) ...
    + sparse( 2 : N, 1 : N - 1, -1, N, N ) + sparse( 1 : N - 1, 2 : N, -1, N, N );
A = mu / h^2 * A;
A = A + sparse( diag( sigma( xnodes( 2 : end - 1 ) ) ) );

bv = ( f( xnodes( 2 : end - 1 ) ) )';
bv( 1 ) = bv( 1 ) + alpha * mu / h^2;
```

```
bv( end ) = bv( end ) + beta * mu / h^2;

uh = A \ bv;

uh = [ alpha; uh; beta ];

xplot = linspace( a, b, 1001 );
figure( 1 );
plot( xplot, uex( xplot ), '-b', xnodes, uh, 'xr', 'LineWidth', 2, 'MarkerSize', 10 );
grid on;  xlabel( 'x' );  ylabel( 'uex, uh ')
legend( 'u_{ex}', 'uh DF centrate', 'location', 'Best' );
```

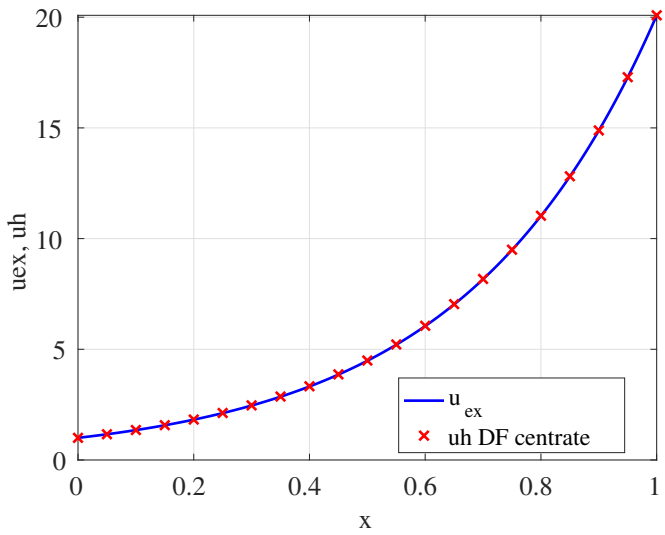

Figura 4: Esercizio 1.2, punto 3. Confronto tra soluzione esatta $u_{ex}(x)$ e soluzione approssimata $\mathbf{u}_h$

4. *Si risolva ora il problema per* $h = 0.04, 0.02, 0.01, \ldots, 0.0025$. *Si rappresenti graficamente, in funzione di* h, *l'andamento dell'errore* e_h. *Si confronti il risultato ottenuto con quanto previsto dalla teoria.*

   Consideriamo i seguenti comandi Matlab® per ottenere il grafico dell'anadamento dell'errore e_h vs. h in Figura 5.

```
hv = [ 0.04 0.02 0.01 0.005 0.0025 ];
Nv = round( ( b - a ) ./ hv ) - 1;
errv = [];
for N = Nv
    xnodes = linspace( a, b, N + 2 );
    h = ( b - a ) / ( N + 1 );

    A = sparse( 1 : N, 1 : N, 2, N, N ) ...
        + sparse( 2 : N, 1 : N - 1, -1, N, N ) + sparse( 1 : N - 1, 2 : N, -1, N, N );
    A = mu / h^2 * A;
    A = A + sparse( diag( sigma( xnodes( 2 : end - 1 ) ) ) );
```

```
    bv = ( f( xnodes( 2 : end - 1 ) ) )';
    bv( 1 ) = bv( 1 ) + alpha * mu / h^2;
    bv( end ) = bv( end ) + beta * mu / h^2;

    uh = A \ bv;

    uh = [ alpha; uh; beta ];

    errv = [ errv, max( abs( uh - uex( xnodes' ) ) ) ];
end

figure( 2 )
loglog( hv, errv, '-x', hv, hv.^2, '--k', 'LineWidth', 2, 'MarkerSize', 10 );
xlabel( ' h [log]' );  ylabel( 'err [log]')
legend( 'eh', '(h,h^2)', 'Location', 'Best' );  grid on
```

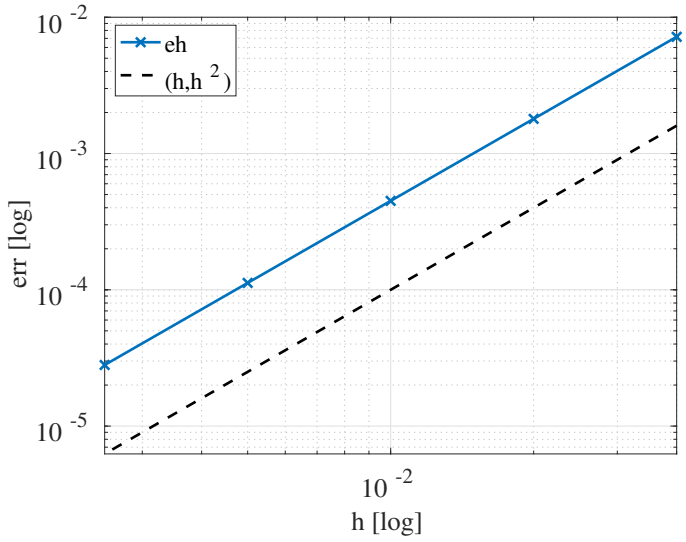

Figura 5: Esercizio 1.2, punto 4. e_h vs. h, scala logaritmica su entrambi gli assi

L'andamento dell'errore e_h vs. h in scala logaritmica per entrambi gli assi risulta pari a una retta parallela a (h, h^2), ovvero l'errore converge con ordine 2 rispetto ad h. Il risultato è in accordo con la teoria, infatti $e_h \leq C\,h^2$ se $f \in C^2([0,1])$ (ovvero $u \in C^4([0,1])$), come effettivamente avviene in questo caso.

Esercizio 1.3

Si pongano $a = 0$, $b = 1$, $\alpha = \beta = 0$ *e* $\mu = 1$.

1. *Posto* $f(x) = -\sin\left(\frac{\pi}{2}x\right)$, *si approssimi il problema tramite lo schema alle differenze finite centrate con un passo* $h = 0.05$ *e si rappresenti la soluzione approssimata. Inoltre, si determini opportunamente lo sforzo approssimato* $\sigma(x)$ *e si determinino i valori delle reazioni vincolari* q_a *e* q_b.

   Procediamo in maniera del tutto analoga all'Esercizio 1.1. Osserviamo però che è necessario approssimare il valore di $\sigma(x) = \mu\, u'(x)$. Usiamo le differenze finite centrate per i nodi interni, ovvero:

$$\sigma_j \approx \mu \frac{u_{j+1} - u_{j-1}}{2h} \qquad \text{per } j = 1, \ldots, N;$$

   mentre rispettivamente le differenze finite in avanti e all'indietro per σ_0 e σ_{N+1}, ovvero:

$$\sigma_0 \approx \mu \frac{u_1 - u_0}{h},$$

$$\sigma_{N+1} \approx \mu \frac{u_{N+1} - u_N}{h}.$$

   Sappiamo che questa approssimazione avrà un ordine di accuratezza solo pari ad 1, mentre le differenze finite centrate di ordine 2.

   Consideriamo i seguenti comandi Matlab® per ottenere le reazioni vincolari e il risultato di Figura 6.

```
a = 0;  b = 1;
alpha = 0;  beta = 0;
mu = 1;
f = @( x ) - sin( pi / 2 * x );

h = 0.05;
N = round( ( b - a ) / h ) - 1;
xnodes = linspace( a, b, N + 2 );

A = sparse( 1 : N, 1 : N, 2, N, N ) ...
    + sparse( 2 : N, 1 : N - 1, -1, N, N ) + sparse( 1 : N - 1, 2 : N, -1, N, N );
A = mu / h^2 * A;

bv = ( f( xnodes( 2 : end - 1 ) ) )';
bv( 1 ) = bv( 1 ) + alpha * mu / h^2;
bv( end ) = bv( end ) + beta * mu / h^2;

uh = A \ bv;

uh = [ alpha; uh; beta ];

sigmah = [ ( uh( 2 ) - uh( 1 ) ) / h;
           ( uh( 3 : end ) - uh( 1 : end - 2 ) ) / 2 / h;
           ( uh( end ) - uh( end - 1 ) ) / h ];

xplot = linspace( a, b, 1001 );
```

```
figure( 1 );
subplot( 3, 1, 1 );
plot( xplot, f( xplot), '-b', 'LineWidth', 2, 'MarkerSize', 10 );
grid on; xlabel( 'x' ); ylabel( 'f(x) ')

subplot( 3, 1, 2 );
plot( xnodes, uh, 'xr', 'LineWidth', 2, 'MarkerSize', 10 );
grid on; xlabel( 'x' ); ylabel( 'uh ')

subplot( 3, 1, 3 );
plot( xnodes, sigmah, 'sk', 'LineWidth', 2, 'MarkerSize', 10 );
grid on; xlabel( 'x' ); ylabel( 'sigmah ')

qa = - sigmah( 1 )
qa =
    0.2308
qb = sigmah( end )
qb =
    0.3805
```

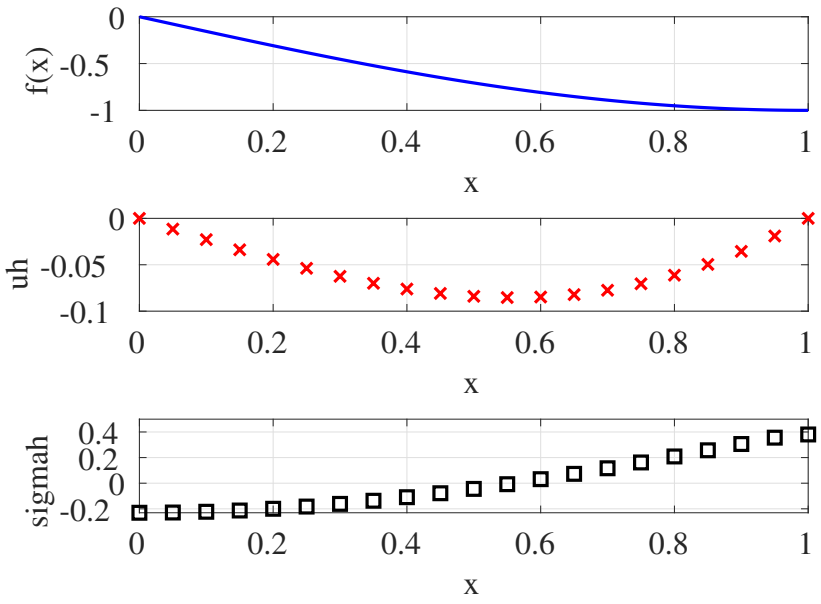

Figura 6: Esercizio 1.3, punto 3. $f(x)$, $\mathbf{u}_h$ e σ_h

2. *Si ripeta il punto 1) con* $f(x) = -(2x-1)\, H\left(x - \frac{1}{2}\right)$, *dove* $H(x)$ *è la funzione scalino (funzione Heaviside) tale che* $H(x) = \begin{cases} 0 & se\ x < 0 \\ 1 & se\ x \geq 0 \end{cases}$.

   Consideriamo i comandi precedenti ma con la seguente modifica.

```
f = @( x ) - ( 2 * x - 1 ) .* ( x >= 1/2 )
```

   Otteniamo il risultato di Figura 7 e le seguenti reazioni vincolari:

```
qa =
    0.0412
qb =
    0.1838
```

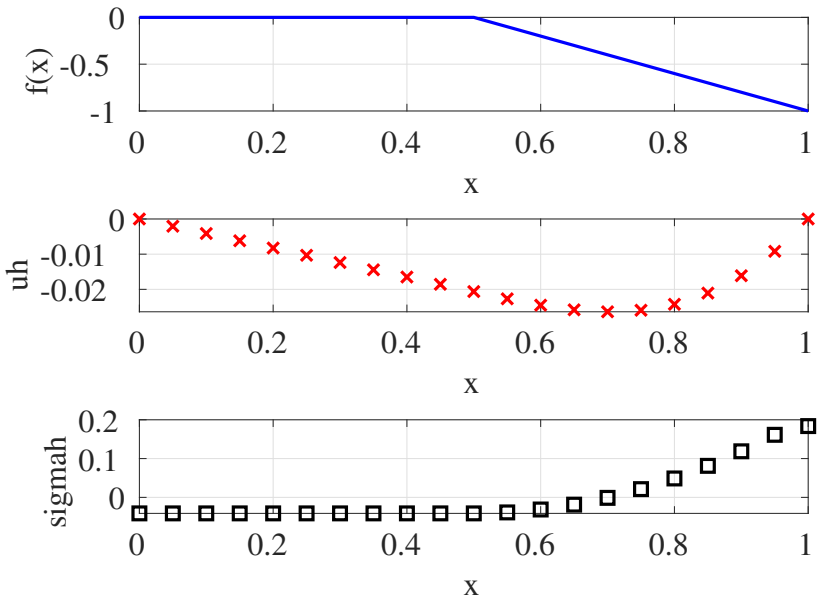

Figura 7: Esercizio 1.3, punto 2. $f(x)$, $\mathbf{u}_h$ e σ_h

Esercizio 2.1

1. *Si verifichi che* $u_{ex}(x)$ *è la soluzione esatta del problema di diffusione–trasporto con i dati indicati e per ogni* μ, $\eta \in \mathbb{R}$.

   Iniziamo verificando che $-\mu\, u''_{ex}(x) + \eta\, u'_{ex}(x) = f(x)$ per ogni $x \in (0,1)$. Essendo $u_{ex}(x) = \dfrac{e^{\eta/\mu\, x} - 1}{e^{\eta/\mu} - 1}$, otteniamo che $-\mu\, u''_{ex}(x) + \eta\, u'_{ex}(x) = 0$ per ogni $x \in (0,1)$ e per ogni μ ed η. Verifichiamo anche il soddisfacimento delle condizioni al contorno di Dirichlet, ovvero che $u_{ex}(0) = 1$ e $u_{ex}(1) = 1$.

2. *Si ricavino le espressioni generali della matrice A e del vettore $\mathbf{b}$ che compaiono nel sistema lineare condensato $A\,\mathbf{u}_h = \mathbf{b}$ e ottenuto mediante approssimazione con le differenze finite centrate.*

   Abbiamo:

$$A = \frac{\mu}{h^2}\,\text{tridiag}(-1, 2, -1) + \frac{\eta}{2h}\,\text{tridiag}(-1, 0, 1)$$

   e

$$\mathbf{b} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix} + \begin{bmatrix} \alpha\,\left(\mu/h^2 + \eta/(2h)\right) \\ 0 \\ \vdots \\ 0 \\ \beta\,\left(\mu/h^2 - \eta/(2h)\right) \end{bmatrix}.$$

3. *Si scriva uno script o funzione Matlab® per risolvere il problema ai limiti ponendo $\mu = 1$, $\eta = 1$, $(N + 1) = 20$. Confrontare in un grafico la soluzione approssimata con quella esatta u_{ex}.*

   Consideriamo i seguenti comandi Matlab® per ottenere il risultato di Figura 8.

```
a = 0;  b = 1;
alpha = 0;  beta = 1;
mu = 1; eta = 1;
f = @( x ) 0 * x;
uex = @( x ) ( exp( eta / mu * x ) - 1 ) / ( exp( eta / mu ) - 1 );

N = 20 - 1;
h = ( b - a ) / ( N + 1 );
xnodes = linspace( a, b, N + 2 );

A = sparse( 1 : N, 1 : N, 2, N, N ) ...
+ sparse( 2 : N, 1 : N - 1, -1, N, N ) ...
+ sparse( 1 : N - 1, 2 : N, -1, N, N );
A = mu / h^2 * A;
A = A + eta / ( 2 * h ) * ( sparse( 2 : N, 1 : N - 1, -1, N, N ) ...
+ sparse( 1 : N - 1, 2 : N, 1, N, N ) );

bv = ( f( xnodes( 2 : end - 1 ) ) )';
bv( 1 ) = bv( 1 ) + alpha * ( mu / h^2 + eta / ( 2 * h ) );
bv( end ) = bv( end ) + beta * ( mu / h^2 - eta / ( 2 * h ) );

uh = A \ bv;

uh = [ alpha; uh; beta ];

xplot = linspace( a, b, 1001 );
figure( 1 );
plot( xplot, uex( xplot ), '-b', xnodes, uh, ':xr'  );
grid on; xlabel( 'x' ); ylabel( 'uex, uh ')
legend( 'u_{ex}', 'uh DF centrate', 'location', 'Best' );
```

4. *Posto ora $\mu = 10^{-2}$, si ripeta il punto 3). Si commenti il risultato ottenuto. Come è possibile garantire l'assenza di oscillazioni numeriche con lo schema alle differenze finite centrate agendo sul valore di h?*

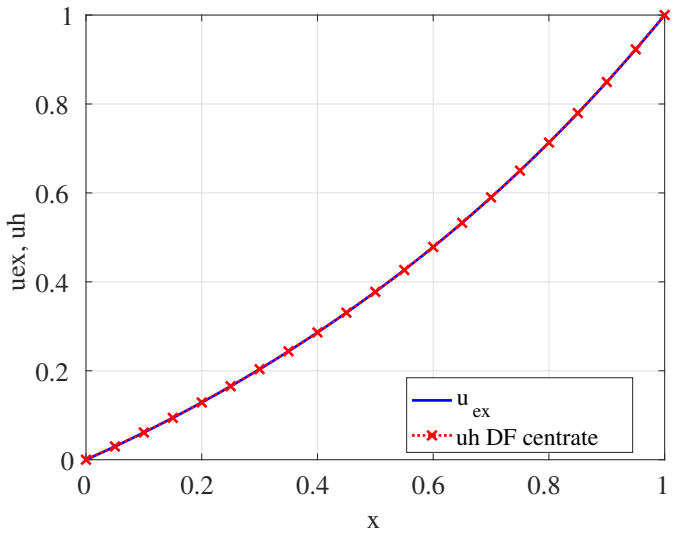

Figura 8: Esercizio 2.1, punto 3. Confronto tra soluzione esatta $u_{ex}(x)$ e soluzione approssimata $\mathbf{u}_h$ per $\mu = 1$, $\eta = 1$ e $h = 1/20$

Ripetiamo i comandi Matlab® del punto 3) per ottenere il risultato di Figura 9. In questo caso la soluzione approssimata è affetta da oscillazioni numeriche; infatti il numero di Péclet locale $\mathbb{P}e_h = \dfrac{|\eta|h}{2\mu} = 2.5$ è maggiore di 1. Per evitare l'incorrere di tale fenomeno numerico è possibile scegliere h tale che $h < h_{max} = \dfrac{2\mu}{|\eta|} = 0.02 = \dfrac{1}{50}$, ovvero tale che $\mathbb{P}e_h < 1$. Si può verificare ripetendo i comandi precedenti con $h < h_{max}$.

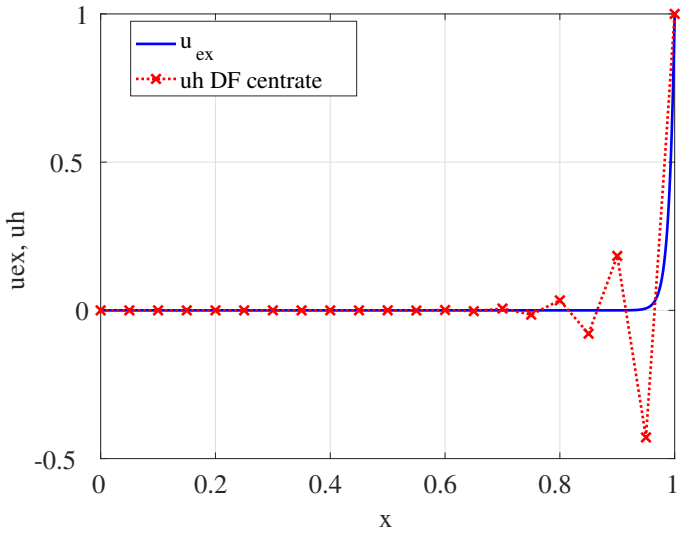

Figura 9: Esercizio 2.1, punto 4. Confronto tra soluzione esatta $u_{ex}(x)$ e soluzione approssimata $\mathbf{u}_h$ per $\mu = 10^{-2}$, $\eta = 1$ e $h = 1/20$

5. *Si risolva ora il problema con i dati di cui al punto 4) per* $(N+1) = 30, 60, 120, 240, 480$. *Si rappresenti graficamente, in funzione di* h, *l'andamento dell'errore* $e_h = \max_{j=0,\ldots,N+1} |u_{ex}(x_j) - u_j|$. *Si confronti il risultato ottenuto con quanto previsto dalla teoria.*

   Consideriamo i seguenti comandi Matlab® per ottenere il grafico dell'andamento dell'errore e_h vs. h in Figura 10.

```
Nv = [ 30 60 120 240 480 ] - 1;
hv = ( b - a ) ./ ( Nv + 1 );
errv = [];
for N = Nv
xnodes = linspace( a, b, N + 2 );
h = ( b - a ) / ( N + 1 );
A = sparse( 1 : N, 1 : N, 2, N, N ) ...
+ sparse( 2 : N, 1 : N - 1, -1, N, N ) ...
+ sparse( 1 : N - 1, 2 : N, -1, N, N );
A = mu / h^2 * A;
A = A + eta / ( 2 * h ) * ( sparse( 2 : N, 1 : N - 1, - 1, N, N ) ...
+ sparse( 1 : N - 1, 2 : N, 1, N, N ) );
bv = ( f( xnodes( 2 : end - 1 ) ) )';
bv( 1 ) = bv( 1 ) + alpha * ( mu / h^2 + eta / ( 2 * h ) );
bv( end ) = bv( end ) + beta * ( mu / h^2 - eta / ( 2 * h ) );

uh = A \ bv;
uh = [ alpha; uh; beta ];
errv = [ errv, max( abs( uh - uex( xnodes' ) ) ) ];
end
loglog( hv, errv, '-x', hv, hv, '-.k', hv, hv.^2, '--k' );
xlabel( ' h [log]' );   ylabel( 'err [log]');
legend( 'eh', '(h,h)', '(h,h^2)', 'Location', 'Best' ); grid on
```

   Osserviamo che l'andamento dell'errore e_h vs. h in scala logaritmica per entrambi gli assi risulta pari a una retta parallela a (h, h^2) per h "sufficientemente" piccolo, ovvero l'errore converge con ordine 2 rispetto ad h. Il risultato è in accordo con la teoria, infatti $e_h \leq C\, h^2$ se $f \in C^2([0,1])$ (ovvero $u \in C^4([0,1])$), come effettivamente avviene in questo caso.

6. *Si ricavino le espressioni generali della matrice* A *e del vettore* $\mathbf{b}$ *che compaiono nel sistema lineare condensato considerando l'uso della tecnica* upwind. *Per il caso* $\eta > 0$, *ciò corrisponde ad approssimare* $u'(x_j)$ *tramite le differenze finite all'indietro, ovvero:*

$$u'(x_j) \simeq \frac{u(x_j) - u(x_{j-1})}{h} \qquad \textit{per } j = 1, 2, \ldots, N.$$

   Se invece $\eta < 0$, *ciò corrisponde ad approssimare* $u'(x_j)$ *tramite le differenze finite in avanti, ovvero:*

$$u'(x_j) \simeq \frac{u(x_{j+1}) - u(x_j)}{h} \qquad \textit{per } j = 1, 2, \ldots, N.$$

   Si mostri che tale tecnica coincide con l'uso delle differenze finite centrate con viscosità artificiale $\mu_h = \mu\,(1 + \mathbb{P}e_h)$, *dove* $\mathbb{P}e_h = \dfrac{|\eta|\, h}{2\mu}$ *è il numero di Péclet locale.*

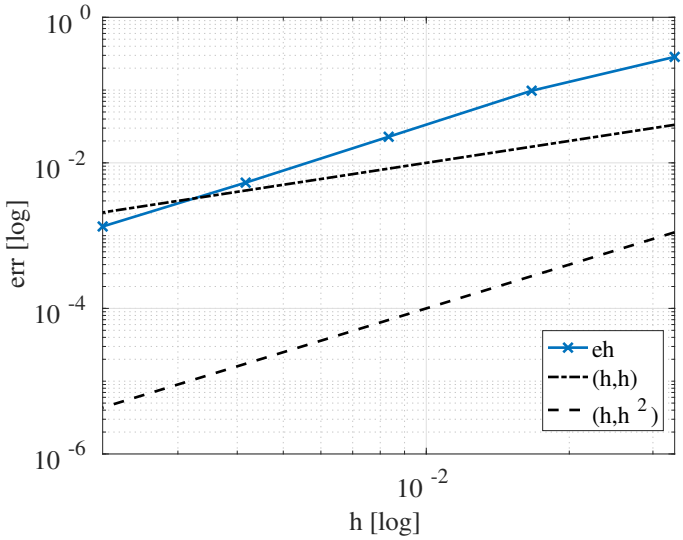

Figura 10: Esercizio 2.1, punto 5. e_h vs. h, scala logaritmica su entrambi gli assi; schema alle differenze finite centrate

Posto $\mu_h = \mu\,(1 + \mathbb{P}e_h) = \mu\left(1 + \dfrac{|\eta|\,h}{2\,\mu}\right)$, abbiamo:

$$A = \frac{\mu_h}{h^2}\,\text{tridiag}(-1, 2, -1) + \frac{\eta}{2h}\,\text{tridiag}(-1, 0, 1)$$

e

$$\mathbf{b} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix} + \begin{bmatrix} \alpha\,\left(\mu_h/h^2 + \eta/(2h)\right) \\ 0 \\ \vdots \\ 0 \\ \beta\,\left(\mu_h/h^2 - \eta/(2h)\right) \end{bmatrix}.$$

7. *Si ripetano i punti 4) e 5) tramite l'uso della tecnica di* upwind.

   Tramite i seguenti comandi Matlab® otteniamo il risultato di Figura 11 che, come vediamo, è esente da oscillazioni numeriche.

   Con i seguenti comandi Matlab® otteniamo il grafico di Figura 12.

```
Nv = [ 30 60 120 240 480 ] - 1;
hv = ( b - a ) ./ ( Nv + 1 );
errv = [];
for N = Nv
xnodes = linspace( a, b, N + 2 );
h = ( b - a ) / ( N + 1 );

Peh = abs( eta ) * h / ( 2 * mu );
muh = mu * ( 1 + Peh );
```

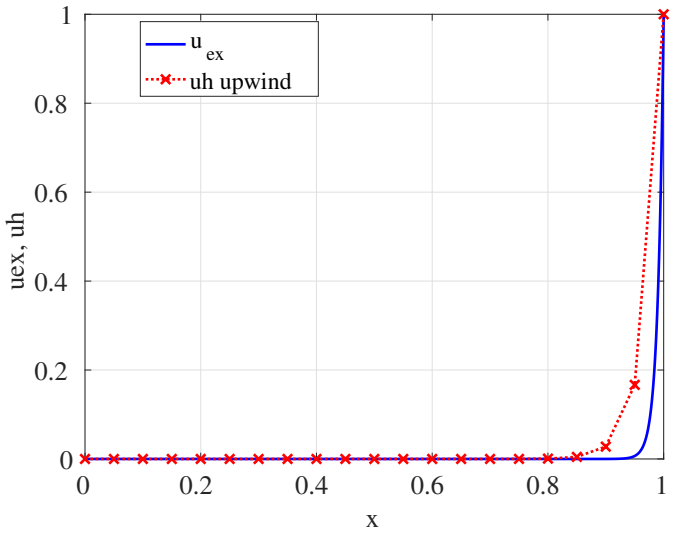

Figura 11: Esercizio 2.1, punto 7. Confronto tra soluzione esatta $u_{ex}(x)$ e soluzione approssimata $\mathbf{u}_h$ per $\mu = 10^{-2}$, $\eta = 1$ e $h = 1/20$; schema upwind.

```
A = sparse( 1 : N, 1 : N, 2, N, N ) ...
+ sparse( 2 : N, 1 : N - 1, -1, N, N ) + sparse( 1 : N - 1, 2 : N, -1, N, N );
A = muh / h^2 * A;
A = A + eta / ( 2 * h ) * ( sparse( 2 : N, 1 : N - 1, - 1, N, N ) ...
+ sparse( 1 : N - 1, 2 : N, 1, N, N ) );

bv = ( f( xnodes( 2 : end - 1 ) ) )';
bv( 1 ) = bv( 1 ) + alpha * ( muh / h^2 + eta / ( 2 * h ) );
bv( end ) = bv( end ) + beta * ( muh / h^2 - eta / ( 2 * h ) );

uh = A \ bv;

uh = [ alpha; uh; beta ];

errv = [ errv, max( abs( uh - uex( xnodes' ) ) ) ];
end

figure( 4 )
loglog( hv, errv, '-x', hv, hv, '-.k', hv, hv.^2, '--k', ...
          'LineWidth', 2, 'MarkerSize', 10 );
xlabel( ' h [log]' );
ylabel( 'err [log]')
legend( 'eh', '(h,h)', '(h,h^2)', 'Location', 'Best' );
grid on
```

L'errore e_h vs. h in scala logaritmica per entrambi gli assi risulta pari a una retta parallela a (h, h) per h "sufficientemente" piccolo, ovvero l'errore converge con ordine 1 rispetto ad h. Il ridotto ordine di convergenza, rispetto allo schema alle differenze finite centrate, è dovuto all'uso della tecnica *upwind* che prevede l'approssimazione della derivata prima di u tramite differenze finite all'indietro o in avanti (a seconda del segno di η); tali schemi sono accurati di ordine 1. Abbiamo dunque in questo caso $e_h^{up} \leq C\, h$.

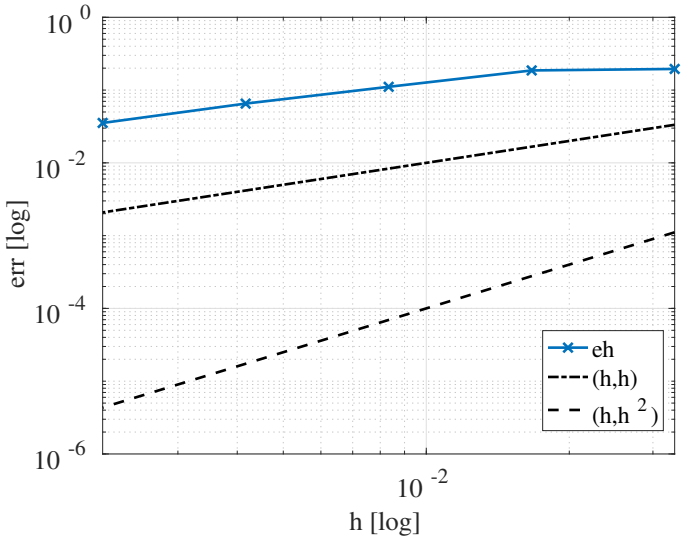

Figura 12: Esercizio 2.1, punto 7. e_h vs. h, scala logaritmica su entrambi gli assi; schema upwind

In alternativa, è possibile usare lo schema alle differenze finite centrate con viscosità artificiale di Scharfetter–Gummel per garantire l'assenza di oscillazioni numeriche e ordine di convergenza 2.

Esercizio 3.1

1. *Si verifichi che $u_{ex}(x)$ è la soluzione esatta del problema con i dati indicati.*

   Verifichiamo che $f(x) = -\mu\, u''_{ex}(x) + \eta\, u'_{ex}(x) + \sigma(x)\, u_{ex}(x)$ per ogni $x \in (0,1)$. Inoltre, verifichiamo che $u_{ex}(0) = \alpha = 1$ e $\mu\, u'_{ex}(b) = \gamma = -e^3$.

2. *Si ricavino le espressioni generali della matrice A e del vettore $\mathbf{b}$ che compaiono nel sistema lineare condensato $A\,\mathbf{u}_h = \mathbf{b}$ ottenuto mediante approssimazione con le differenze finite centrate e lo schema delle differenze finite all'indietro per l'approssimazione di $u'(x_{N+1})$.*

   Posto:

   $$A_{NN} = \frac{\mu}{h^2}\,\text{tridiag}(-1,2,-1) + \frac{\eta}{2h}\,\text{tridiag}(-1,0,1) + \text{diag}\left((\sigma(x_1),\ldots,\sigma(x_N))^T\right) \in \mathbb{R}^{N\times N},$$

   abbiamo:

   $$A = \left[\begin{array}{ccc|c} & & & 0 \\ & A_{NN} & & \vdots \\ & & & 0 \\ & & & \left(-\mu/h^2 + \eta/(2h)\right) \\ \hline 0 & \cdots & 0 \quad -\mu/h & \mu/h \end{array}\right] \in \mathbb{R}^{(N+1)\times(N+1)}.$$

Ponendo

$$\mathbf{b}_N = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix} + \begin{bmatrix} \alpha \left(\mu/h^2 + \eta/(2h)\right) \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^N,$$

otteniamo:

$$\mathbf{b} = \left[\begin{array}{c} \mathbf{b}_N \\ \hline \gamma \end{array}\right] \in \mathbb{R}^{(N+1)}.$$

3. *Si scriva uno script o funzione Matlab® per risolvere il problema ai limiti di cui al punto 2) con* $(N+1) = 100$. *Confrontare in un grafico la soluzione approssimata con quella esatta* u_{ex}.

   Consideriamo i seguenti comandi Matlab® per ottenere il grafico di Figura 13.

```
a = 0;      b = 1;
alpha = 1;      gamma = - exp( 3 );
mu = 1;     eta = 0;        sigma = @( x ) 0 * x;
f = @( x ) exp( 3 * x ) .* ( - 4 + 3 * x + 9 * x.^2 );
uex = @( x ) exp( 3 * x ) .* ( x - x.^2 ) + 1;

N = 100 - 1;
h = ( b - a ) / ( N + 1 );
xnodes = linspace( a, b, N + 2 );

A = sparse( 1 : N, 1 : N, 2, N + 1, N + 1 ) ...
+ sparse( 2 : N, 1 : N - 1, -1, N + 1, N + 1 ) ...
+ sparse( 1 : N, 2 : N + 1, -1, N + 1, N + 1 );
A = mu / h^2 * A;
A = A + eta / ( 2 * h ) * ( sparse( 2 : N, 1 : N - 1, -1, N + 1, N + 1 ) ...
+ sparse( 1 : N, 2 : N + 1, 1, N + 1, N + 1 ) );
A = A + sparse( 1 : N, 1 : N, sigma( xnodes( 2 : end - 1 ) ), N + 1, N + 1 );
A( end, N : N + 1 ) = mu / h * [ -1 1 ];

bv = ( f( xnodes( 2 : end ) ) )';
bv( 1 ) = bv( 1 ) + alpha * ( mu / h^2 + eta / ( 2 * h ) );
bv( end ) = gamma;

uh = A \ bv;
uh = [ alpha; uh ];

xplot = linspace( a, b, 1001 );
plot( xplot, uex( xplot ), '-b', xnodes, uh, ':xr' );
grid on; xlabel( 'x' ); ylabel( 'uex, uh ')
legend( 'u_{ex}', 'uh DF centrate & indietro', 'location', 'Best' );
```

4. *Si risolva ora il problema con lo schema di cui al punto 2) per* $(N+1) = 50, 100, 200, 400, 800$. *Si rappresenti graficamente, in funzione di* h, *l'andamento dell'errore* e_h. *Si confronti il risultato ottenuto con quanto previsto dalla teoria.*

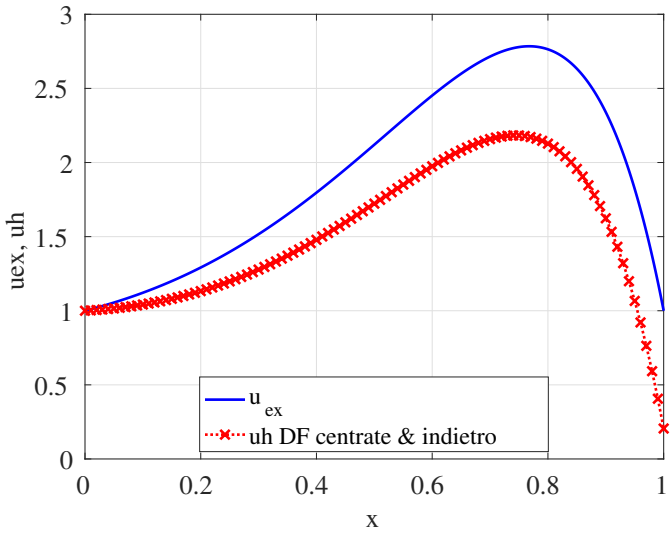

Figura 13: Esercizio 3.1, punto 3. Confronto tra soluzione esatta $u_{ex}(x)$ e soluzione approssimata $\mathbf{u}_h$; schema alle differenze finite centrate con differenze finite all'indietro per $u'(x_{N+1})$

Consideriamo i seguenti comandi Matlab® :

```
Nv = [ 50 100 200 400 800 ] - 1;
hv = ( b - a ) ./ ( Nv + 1 );
errv = [];
for N = Nv
xnodes = linspace( a, b, N + 2 );
h = ( b - a ) / ( N + 1 );
A = sparse( 1 : N, 1 : N, 2, N + 1, N + 1 ) ...
+ sparse( 2 : N, 1 : N - 1, -1, N + 1, N + 1 ) ...
+ sparse( 1 : N, 2 : N + 1, -1, N + 1, N + 1 );
A = mu / h^2 * A;
A = A + eta / ( 2 * h ) * ( sparse( 2 : N, 1 : N - 1, -1, N + 1, N + 1 ) ...
+ sparse( 1 : N, 2 : N + 1, 1, N + 1, N + 1 ) );
A = A + sparse( 1 : N, 1 : N, sigma( xnodes( 2 : end - 1 ) ), N + 1, N + 1 );
A( end, N : N + 1 ) = mu / h * [ -1 1 ];
bv = ( f( xnodes( 2 : end ) ) )';
bv( 1 ) = bv( 1 ) + alpha * ( mu / h^2 + eta / ( 2 * h ) );
bv( end ) = gamma;
uh = A \ bv;
uh = [ alpha; uh ];
errv = [ errv, max( abs( uh - uex( xnodes' ) ) ) ];
end
loglog( hv, errv, '-x', hv, hv, '-.k', hv, hv.^2, '--k' );
xlabel( ' h [log]' );   ylabel( 'err [log]')
legend( 'eh', '(h,h)', '(h,h^2)', 'Location', 'Best' );
grid on
```

Dato che l'errore e_h vs. h descrive in questo grafico in scala logaritmica su entrambi gli assi una retta parallela a (h, h), deduciamo che il metodo converge con ordine 1 per questo problema con condizioni al contorno miste. Infatti, abbiamo approssimato

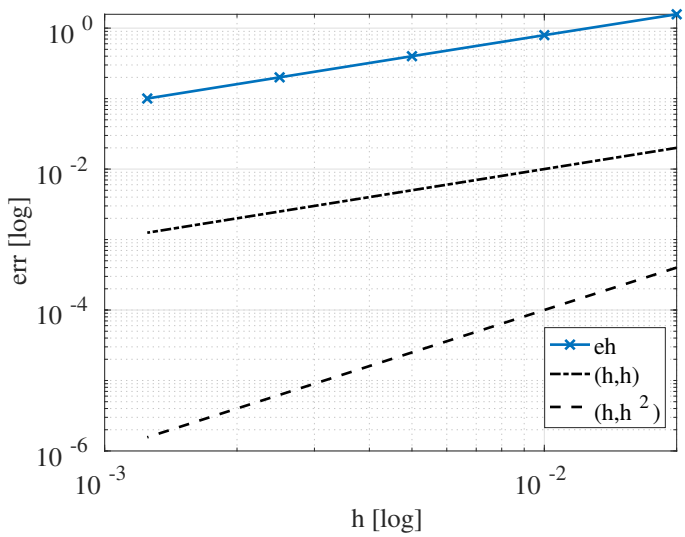

Figura 14: Esercizio 3.1, punto 4. e_h vs. h, scala logaritmica su entrambi gli assi; schema alle differenze finite centrate con differenze finite all'indietro per $u'(x_{N+1})$

$u'(x_{N+1})$ con uno schema alle differenze finite all'indietro, uno schema appunto accurato di ordine 1. Ciò comporta una riduzione di accuratezza complessiva per lo schema alle differenze finite (anche usando le differenze finite centrate per tutti i nodi interni).

5. *Si ripetano i punti 2), 3) e 4) usando ora lo schema accurato di ordine* 2 *per l'approssimazione di* $u'(x_{N+1})$.

   Rispetto al punto 2), abbiamo:

$$A = \left[\begin{array}{cccccc} & & A_{NN} & & & \begin{matrix}0\\ \vdots \\ 0 \\ \left(-\mu/h^2 + \eta/(2h)\right)\end{matrix} \\ \hline 0 & \cdots & 0 & \mu/(2h) & -4\mu/(2h) & 3\mu/(2h) \end{array}\right] \in \mathbb{R}^{(N+1)\times(N+1)}.$$

   Inoltre, consideriamo comandi Matlab® del tutto simili al punto 3)

```
...
A( end, N - 1 : N + 1 ) = mu / ( 2 * h ) * [ 1 -4 3 ];
...
```

   per ottenere il risultato di Figura 15.

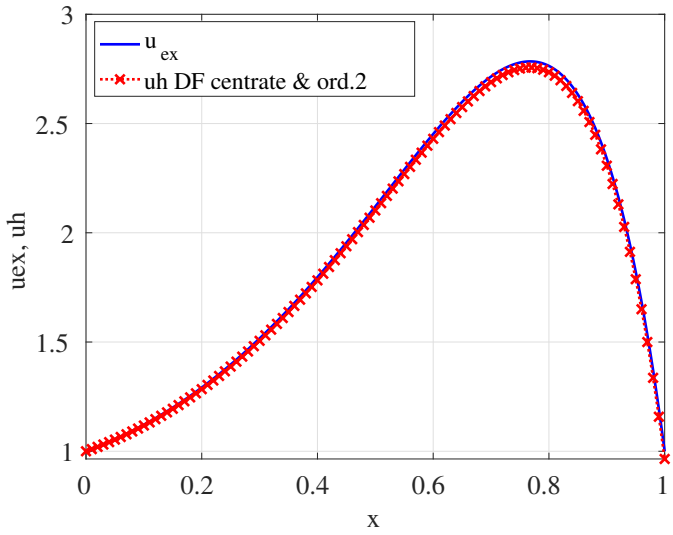

Figura 15: Esercizio 3.1, punto 5. Confronto tra soluzione esatta $u_{ex}(x)$ e soluzione approssimata $\mathbf{u}_h$; schema alle differenze finite centrate con differenze finite all'indietro per $u'(x_{N+1})$ con ordine di accuratezza 2

Analogamente, procedendo come al punto 4), otteniamo il grafico degli errori e_h vs. h. In questo caso, notiamo che l'errore converge con ordine 2 rispetto ad h, infatti la retta in questo grafico in scala logaritmica su entrambi gli assi è parallela a quella di (h, h^2). Il risultato era atteso dato che abbiamo usato uno schema di ordine 2 per approssimare anche $u'(x_{N+1})$.

Esercizio 3.2

Si veda il file `es3.m` allegato.

Esercizio 4.1

1. *Si mostri che* $u_{ex}(x, y) = \sin(\pi x)\,\sin(2\pi y)$ *è la soluzione esatta del problema di Poisson.*

Essendo $\mu = 1$, iniziamo verificando che $-\Delta u_{ex}(\boldsymbol{x}) = f(\boldsymbol{x})$ per ogni $\boldsymbol{x} \in \Omega$:

$$\frac{\partial u_{ex}}{\partial x} = \pi \cos(\pi x)\sin(2\pi y) \rightarrow \frac{\partial^2 u_{ex}}{\partial x^2} = -\pi^2 \sin(\pi x)\sin(2\pi y)$$
$$\frac{\partial u_{ex}}{\partial y} = 2\pi \sin(\pi x)\cos(2\pi y) \rightarrow \frac{\partial^2 u_{ex}}{\partial y^2} = -4\pi^2 \sin(\pi x)\sin(2\pi y)$$

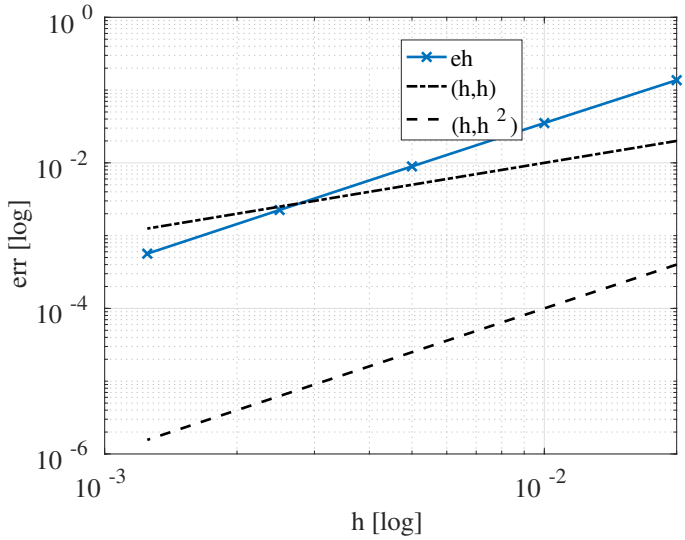

Figura 16: Esercizio 3.1, punto 5. e_h vs. h, scala logaritmica su entrambi gli assi; schema alle differenze finite centrate con differenze finite all'indietro per $u'(x_{N+1})$ con ordine di accuratezza 2

pertanto risulta

$$-\left(\frac{\partial^2 u_{ex}}{\partial x^2}+\frac{\partial^2 u_{ex}}{\partial y^2}\right)=5\pi^2\sin(\pi x)\sin(2\pi y)=f(x,y),\quad \forall x\in\Omega.$$

Le condizioni al contorno inoltre risultano verificate poichè

$$\begin{aligned} u_{ex}(0,y)&=0 \quad \forall (x,y)\in\partial\Omega,\\ u_{ex}(1,y)&=0 \quad \forall (x,y)\in\partial\Omega,\\ u_{ex}(x,0)&=0 \quad \forall (x,y)\in\partial\Omega,\\ u_{ex}(x,1)&=0 \quad \forall (x,y)\in\partial\Omega. \end{aligned}$$

2. *Si rappresenti in Matlab® la soluzione esatta del problema (si utilizzi opportunamente il comando* ***surf****) su una griglia con* $h=h_x=h_y=0.1$.

Consideriamo i seguenti comandi Matlab® per ottenere il risultato di Figura 17.

```
a = 0;   b = 1;   c = 0;   d = 1;

uex= @(x, y) sin(pi*x).*sin(2*pi*y);
h = 1e-1;
xvect = a:h:b;
yvect = c:h:d;
[Xplot, Yplot] = meshgrid(xvect, yvect);
surf(Xplot, Yplot, uex(Xplot, Yplot), 'Lines','no')
xlabel('x')
ylabel('y')
zlabel('u_{ex}(x, y)')
```

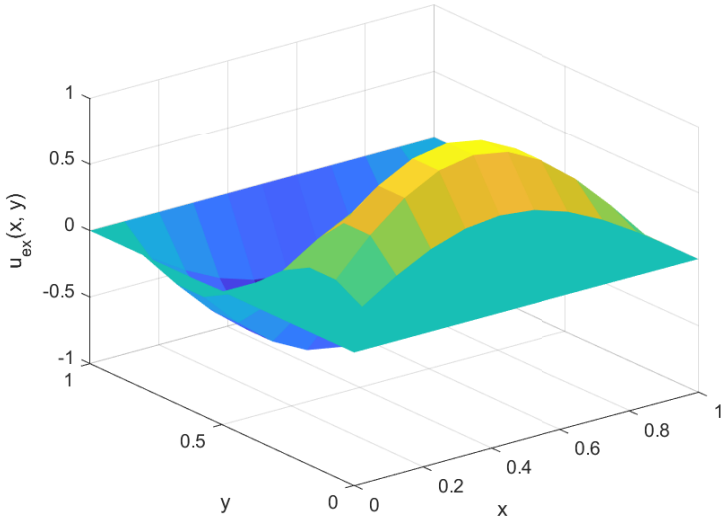

Figura 17: Esercizio 4.1, punto 2. Soluzione esatta del problema bidimensionale di Poisson

3. *Si approssimi il problema di Poisson con i dati precedenti tramite lo schema alle differenze finite centrate a cinque punti. In particolare, si utilizzi la funzione Matlab®* `Poisson_Dirichlet_diff_finite_5punti.m` *con* $h = h_x = h_y = 0.1$. *Si rappresenti la soluzione approssimata ottenuta.*

   Consideriamo i seguenti comandi Matlab® per ottenere il risultato di Figura 18.

```
mu = 1;
f = @(x, y) 5*pi^2*sin(pi*x).*sin(2*pi*y);
g = @(x, y) 0;
Omega = [a b c d];
hx = 1e-1;
hy = 1e-1;

[ X, Y, U ] = Poisson_Dirichlet_diff_finite_5punti( mu, f, g, Omega, hx, hy );
figure
surf(X, Y, U)
xlabel('x')
ylabel('y')
zlabel('u_{h}')
```

4. *Si risolva ora il problema con lo schema al punto 3) per*

$$h = h_x = h_y = 0.1, 0.05, 0.025, 0.0125, 0.00625.$$

   Si rappresenti graficamente, in funzione di $h = h_x = h_y$, *l'andamento dell'errore:*

$$e_h = \max_{i,j} |u(x_i, y_j) - u_{ij}|.$$

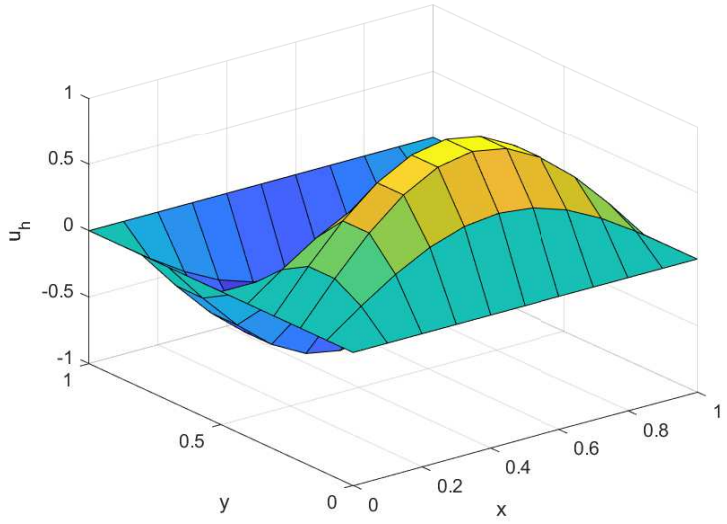

Figura 18: Esercizio 4.2, punto 3. Soluzione approsimata del problema bidimensionale di Poisson tramite metodo alle differenze finite centrate a cinque punti con $h_x = h_y = 0.1$

Si confronti il risultato ottenuto con quanto previsto dalla teoria.

Consideriamo i seguenti comandi Matlab® per ottenere il risultato di Figura 19. Si noti che al fine di calcolare l'errore è necessario chiamare la function `max` due volte, in modo da calcolare il massimo lungo le righe e lungo le colonne della matrice `abs(U - uex(X, Y).`

```
hvect = 0.1*2.^[0:-1:-4];
errvect = zeros(size(hvect));

for i = 1:length(hvect)
h = hvect(i);
[ X, Y, U ] = Poisson_Dirichlet_diff_finite_5punti( mu, f, g, Omega, h, h );
errvect(i) = max(max((abs(U - uex(X, Y) ))));
end

figure
loglog(hvect, errvect, 'o-', 'LineWidth', 2, 'MarkerSize', 10)
hold on
loglog(hvect, hvect, 'k--', 'LineWidth', 2, 'MarkerSize', 10)
loglog(hvect, hvect.^2, 'k-.', 'LineWidth', 2, 'MarkerSize', 10)
legend('errore', 'lineare', 'quadratico', 'location', 'best')
grid on
xlabel( ' h [log]' );
ylabel( 'err [log]')
```

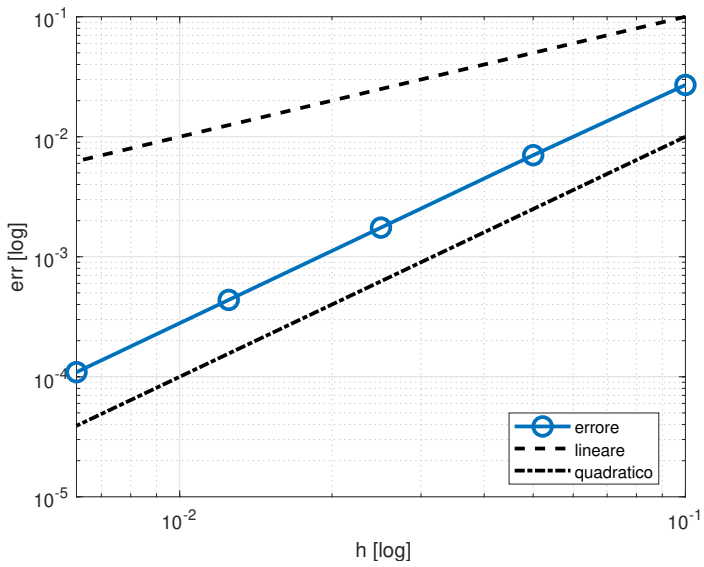

Figura 19: Esercizio 4.2, punto 4. e_h vs h, scala logaritmica su entrambi gli assi; schema alle differenze finite centrate a cinque punti

Si consideri il grafico in scala logaritmica su entrambi gli assi in Figura 19. Dato che l'errore e_h vs. h descrive una retta parallela a (h, h^2), deduciamo che il metodo converge con ordine 2. Il risultato ottenuto era atteso in quanto se $u \in C^4(\overline{\Omega})$, come effettivamente avviene in questo caso, vale $e_h \leq \tilde{C}h^2$.

Esercizio 5.1

1. *Si implementi la function Matlab® `[u, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, us, ud, g0, T, h, delta_t, theta)`*

   Una possibile implementazione della funzione richiesta è la seguente:

```
function [u, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, us, ud, g0, ...
T, h, delta_t, theta)
% [u, x, t] = EqCalore_DiffFin_Theta(f, a, b, us, ud, g0,
%   T, h, delta_t, theta)
% Soluzione del seguente problema per l'equazione del calore:
%   du/dt - mu d2u/dx2 = f       x in (a, b), t in (0, T]
%   u(0,x) = g0(x)            x in (a, b)
%   u(t,a) = us(t)            t in (0, T]
%   u(t,b) = ud(t)            t in (0, T]
%
% INPUT:
%   - mu: coefficiente (costante)
%   - f(x,t): forzante
%   - a, b: estremi del dominio
%   - us(t), ud(t): dati di Dirichlet rispettivamente in x = a e x = b
%   - g0(x): dato iniziale
%   - T: tempo finale
```

```
%   - h: passo di discretizzazione spaziale
%   - delta_t: passo di discretizzazione temporale
%   - theta: parametro del theta-metodo
% OUTPUT:
%   - u: soluzione numerica; il primo indice determina il nodo
%        nello spazio, il secondo indice l'istante temporale
%        (u(i,j) = soluzione approssimata al nodo x_i, al tempo t_j)
%   - x: vettore dei nodi di discretizzazione
%   - t: vettore dei tempi


% Vettore dei nodi x_j.
Nx = (b - a) / h - 1;
x = linspace(a, b, Nx + 2);

% Vettore dei tempi t^(k).
t = 0:delta_t:T;
Nt = length(t);

% Matrice A.
A = mu / h^2 * (sparse(1:Nx, 1:Nx, 2, Nx, Nx) ...
+ sparse(1:Nx-1, 2:Nx, -1, Nx, Nx) ...
+ sparse(2:Nx, 1:Nx-1, -1, Nx, Nx));

% Matrice A_theta.
A_theta = speye(Nx) + delta_t * theta * A;

% Soluzione approssimata.
u = zeros(length(x), length(t));
u(:,1) = g0(x);

for k = 1:Nt-1
Fk = f(x(2:end-1), t(k))';
Fk(1) = Fk(1) + mu * us(t(k)) / h^2;
Fk(end) = Fk(end) + mu * ud(t(k)) / h^2;

Fkp1 = f(x(2:end-1), t(k+1))';
Fkp1(1) = Fkp1(1) + mu * us(t(k+1)) / h^2;
Fkp1(end) = Fkp1(end) + mu * ud(t(k+1)) / h^2;

bkp1 = (speye(Nx) - delta_t * (1 - theta) * A) * u(2:end-1,k) ...
+ delta_t * theta * Fkp1 ...
+ delta_t * (1 - theta) * Fk;

u(2:end-1,k+1) = A_theta \ bkp1;
u(1,k+1) = us(t(k+1));
u(end,k+1) = ud(t(k+1));
end

end
```

2. *Si risolva il problema con la funzione appena implementata utilizzando il metodo di Eulero implicito ($\theta = 1$), $h = 0.01$, $\Delta t = 0.1$, e si rappresentino in un grafico la soluzione esatta e la soluzione numerica al tempo finale T.*

   Il codice seguente risolve il problema come richiesto, producendo il grafico in figura 20:

```
% Estremi dell'intervallo.
a = 0;
b = 1;

% Tempo finale.
T = 1;

% Dati funzionali.
mu = 1;
f = @(x,t) (-sin(t) + 0.25 * cos(t)) * sin(0.5*x); % Termine forzante.
u_ex = @(x,t) sin(0.5*x) * cos(t); % Soluzione esatta.
u_s = @(t) u_ex(a, t); % Dato di Dirichlet in x = a.
u_d = @(t) u_ex(b, t); % Dato di Dirichlet in x = b.
g_0 = @(x) u_ex(x, 0); % Dato iniziale.

theta = 1; % Parametro del theta-metodo.
h = 0.01; % Passo di discretizzazione in spazio.
delta_t = 0.1; % Passo di discretizzazione in tempo.

% Risolvo il problema con la funzione EqCalore_DiffFin_Theta
[u, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, u_s, u_d, g_0, T, h, delta_t, theta);

% Plot della soluzione esatta e di quella numerica all'istante finale.
figure;
plot(x, u(:,end), '-', x, u_ex(x,T), '--', 'LineWidth', 2);
legend('Soluzione numerica', 'Soluzione esatta', 'Location', 'SouthEast');
xlabel('x');
ylabel('u(x)');
title(['Eulero implicito, h = ' num2str(h) ', \Deltat = ' num2str(delta_t)]);
```

3. *Si risolva il problema utilizzando il metodo di Eulero esplicito ($\theta = 0$) prima con $h = 0.01$ e $\Delta t = 0.1$ e, poi, con $h = 0.1$ e $\Delta t = 0.001$. Si commenti il risultato ottenuto.*

   Il codice seguente produce i grafici in Figura 21:

```
% (a)
theta = 0;
h = 0.01;
delta_t = 0.1;
[u, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, u_s, u_d, g_0, T, h, delta_t, theta);

% Plot della soluzione esatta e di quella numerica all'istante finale.
figure;
plot(x, u(:,end), '-', x, u_ex(x,T), '--', 'LineWidth', 2);
legend('Soluzione numerica', 'Soluzione esatta', 'Location', 'SouthEast');
xlabel('x');
ylabel('u_h');
title(['Eulero esplicito, h = ' num2str(h) ', \Deltat = ' num2str(delta_t)]);
```

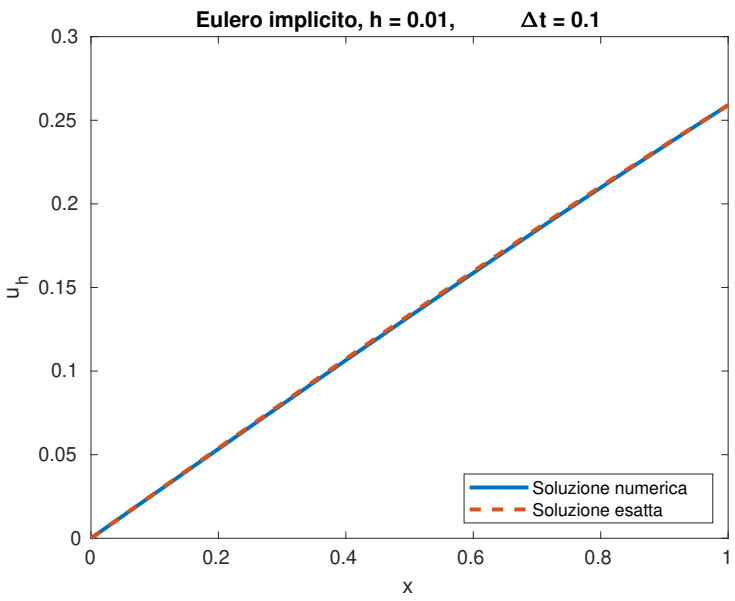

Figura 20: Esercizio 5.1, punto 2. Soluzione numerica e soluzione esatta al tempo $t = T$ del problema per l'equazione del calore, ottenuta con le differenze finite centrate e il metodo di Eulero implicito, utilizzando $h = 0.01$, $\Delta t = 0.1$.

```
% (b)
theta = 0;
h = 0.1;
delta_t = 0.001;
[u, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, u_s, u_d, g_0, T, h, delta_t, theta);

% Plot della soluzione esatta e di quella numerica all'istante finale.
figure;
plot(x, u(:,end), '-', x, u_ex(x,T), '--', 'LineWidth', 2);
legend('Soluzione numerica', 'Soluzione esatta', 'Location', 'SouthEast');
xlabel('x');
ylabel('u_h');
title(['Eulero esplicito, h = ' num2str(h) ', \Deltat = ' num2str(delta_t)]);
```

Si osserva che scegliendo $h = 0.01$, $\Delta t = 0.1$ il metodo di Eulero esplicito produce una soluzione numerica non assolutamente stabile. Ciò è dovuto al fatto che il metodo di Eulero esplicito è condizionatamente assolutamente stabile, ovvero se

$$\Delta t < \frac{h^2}{2\mu} = \frac{h^2}{2}, \tag{1}$$

mentre con questa scelta di parametri di discretizzazione si ha $\Delta t = 0.1 > 0.00005 = \frac{h^2}{2}$. Scegliendo invece $h = 0.1$, $\Delta t = 0.001$, si ha $\frac{h^2}{2} = 0.005 > 0.001 = \Delta t$, per cui il metodo è assolutamente stabile.

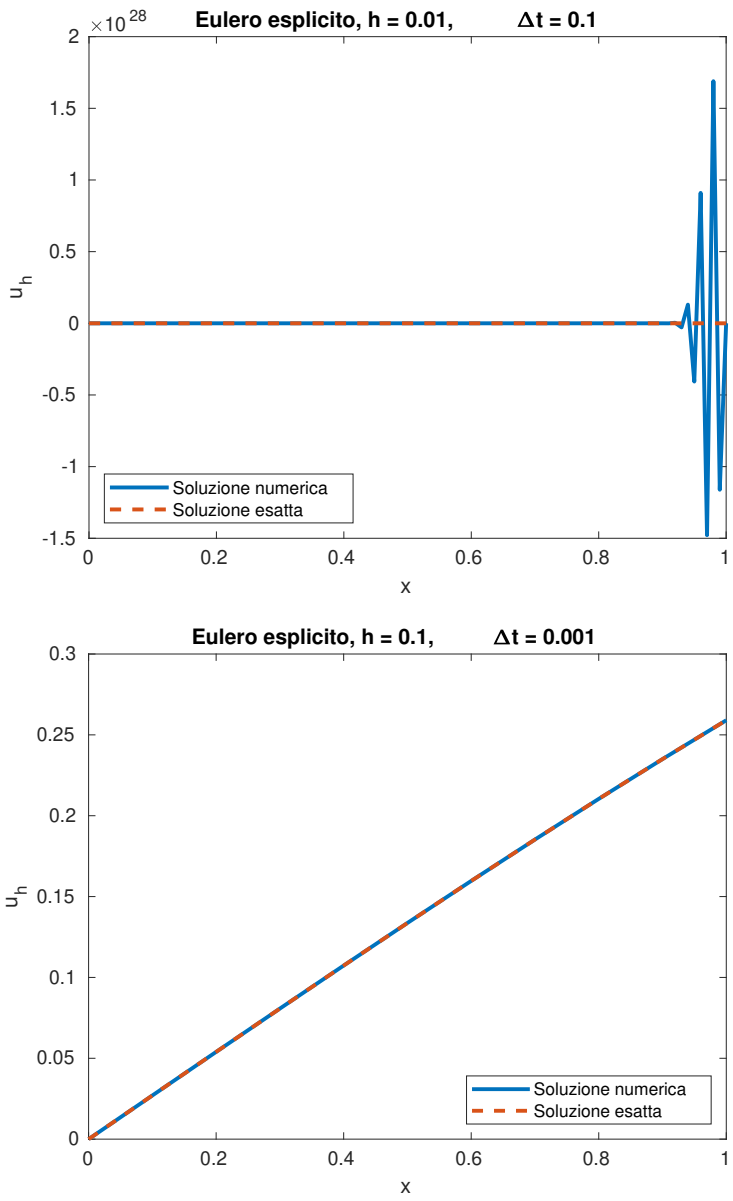

Figura 21: Esercizio 5.1, punto 3. Soluzioni numeriche e soluzione esatta al tempo $t = T$ del problema per l'equazione del calore, ottenute con le differenze finite centrate e il metodo di Eulero esplicito.

4. *Indicando con* $e_T = \max_{i=0,\dots,N+1} |u_{ex}(x_i, T) - u_i(T)|$ *l'errore di discretizzazione al tempo finale* T*, studiare l'andamento di* e_T *al variare di* Δt *per i metodi di Eulero implicito, Eulero esplicito, Crank-Nicolson* ($\theta = 1/2$)*.*

   I seguenti comandi producono i grafici in Figura 22:

```
% Eulero implicito, Crank-Nicolson.
err_ei = [];
err_cn = [];
err_cn_h0005 = [];

h = 0.1; % Passo di discretizzazione in spazio.

% Vettore dei passi di discretizzazione in tempo.
delta_t_vec = [0.1, 0.05, 0.025, 0.0125, 0.00625, 0.003125];

for delta_t = delta_t_vec
[u_ei, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, u_s, u_d, g_0, T, h, delta_t, 1);
err_ei(end+1) = max(abs(u_ei(:,end) - u_ex(x', T)));

[u_cn, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, u_s, u_d, g_0, T, h, delta_t, 0.5);
err_cn(end+1) = max(abs(u_cn(:,end) - u_ex(x', T)));

[u_cn, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, u_s, u_d, g_0, T, ...
0.005, delta_t, 0.5);
err_cn_h0005(end+1) = max(abs(u_cn(:,end) - u_ex(x', T)));
end

% Plot logaritmico di errore vs. delta_t
figure;
loglog(delta_t_vec, err_ei, 'o-', delta_t_vec, err_cn, 'o-', ...
delta_t_vec, err_cn_h0005, 'o-', ...
delta_t_vec, delta_t_vec, '--', delta_t_vec, delta_t_vec.^2, '--');
legend('Eulero implicito', 'Crank-Nicolson, h = 0.1', ...
'Crank-Nicolson, h = 0.005', '\Deltat', '\Deltat^2', ...
'Location', 'SouthEast');
title('Errore vs. \Deltat, Eulero implicito e Crank-Nicolson');
grid on
xlabel('\Delta t [log]')
ylabel('e_T [log]')

% Eulero esplicito.
delta_t_vec = [0.001, 0.0005, 0.00025, 0.000125, 0.0000625, 0.00003125];
err_ea = [];

for delta_t = delta_t_vec
[u_ea, x, t] = EqCalore_DiffFin_Theta(mu, f, a, b, u_s, u_d, g_0, T, h, delta_t, 0);
err_ea(end+1) = max(abs(u_ea(:,end) - u_ex(x', T)));
end

% Plot logaritmico di errore vs. delta_t
figure;
loglog(delta_t_vec, err_ea, 'o-', delta_t_vec, delta_t_vec, '--');
legend('Eulero esplicito, h = 0.1', '\Deltat', ...
'Location', 'SouthEast');
title('Errore vs. \Deltat, Eulero esplicito');
```

```
grid on
xlabel('\Delta t [log]')
ylabel('e_T [log]')
```

Ricordiamo che, se la soluzione $u(x,t)$ è "sufficientemente" accurata, allora l'errore si comporta come:

$$e_T = \max_{i=0,\dots,N+1} |u_{ex}(x_i, T) - u_i(T)| \leq C \left(h^2 + \Delta t^{p(\theta)}\right),$$

dove $p(\theta) = 1$ per $\theta \in [0, 1/2) \cup (1/2, 1]$ e $p(1/2) = 2$. Vi sono dunque due contributi dell'errore, uno legato alla discretizzazione spaziale e uno alla discretizzazione in tempo. Dunque, al fine di evidenziare l'ordine di convergenza rispetto ad h o Δt, è necessario che l'altra componente dell'errore sia "sufficientemente" piccola.

Osserviamo come per Eulero implicito ($\theta = 1$) l'andamento dell'errore vs. Δt nel grafico logaritmico è parallelo alla retta $(\Delta t, \Delta t)$: il metodo è convergente in tempo con ordine 1. Per il metodo di Crank-Nicolson ($\theta = 1/2$), ponendo $h = 0.1$, per Δt tendente a 0, si osserva che l'errore diminuisce più lentamente dell'andamento teorico previsto. Ciò è dovuto al fatto che il termine dominante nell'errore è quello dovuto alla discretizzazione in spazio. A conferma di questo, si osservi come l'errore per il metodo di Crank-Nicolson con $h = 0.005$ mostri l'ordine di convergenza previsto (ordine di convergenza 2).

Anche l'errore per il metodo di Eulero esplicito ($\theta = 0$), per Δt tendente a 0, converge più lentamente dell'andamento teorico previsto (ordine di convergenza 1). Anche in questo caso, ciò è dovuto al fatto che per Δt piccolo l'errore dovuto alla discretizzazione in spazio diventa dominante rispetto a quello in tempo. Tuttavia in questo caso per poter ridurre h è necessario ridurre anche Δt rispettando la condizione (1).

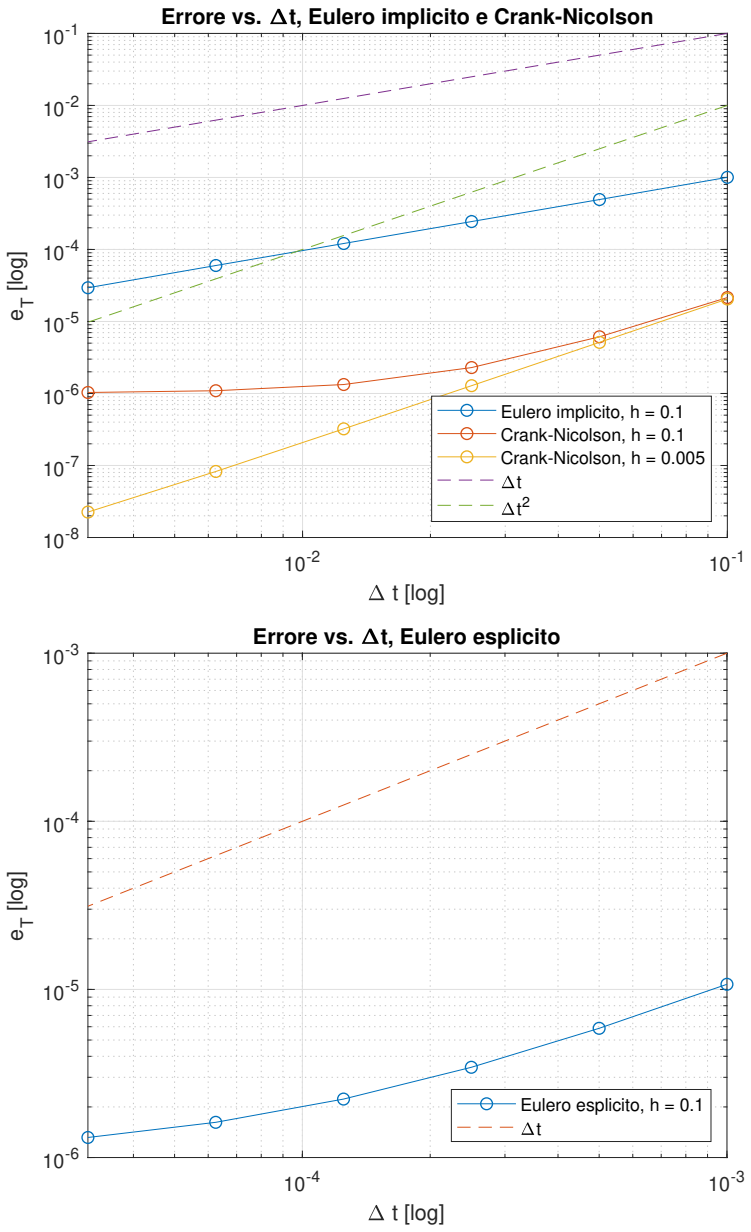

Figura 22: Esercizio 5.1, punto 4. Andamento dell'errore di discretizzazione al tempo finale per l'equazione del calore, al variare di Δt, per i metodi di Eulero implicito, Crank-Nicolson ed Eulero esplicito.