Preparazione alla II Prova in Itinere. Soluzioni

Esercizio 1. (Risoluzione di sistemi lineari. Metodi diretti)

1. Una matrice ammette fattorizzazione di Cholesky se è simmetrica e definita positiva. Il solo valore di ϵ per cui A è simmetrica è $\epsilon^* = \frac{1}{2}$. In corrispondenza di tale valore, A_{ϵ^*} è definita positiva, infatti i suoi autovalori sono tutti positivi

```
>> Astar=[1/2 1 1/2 0; 1/2 1 0 ; 0 0 1/2];
>> eig(Astar)
     1.909830056250526e-01
     5.000000000000000e-01
     1.309016994374947e+00
```

   Calcoliamo i fattori di Cholesky di A_{ϵ^*}, ovvero - secondo la notazione MATLAB - la matrice triangolare superiore R tale che $A = R^T R$:

```
>> R=chol(Astar)
```

   Calcoliamo quindi la soluzione del sistema $Ax = b$, risolvendo in cascata i sistemi $R^T y = b$ e successivamente $Rx = y$:

```
>> b=A*ones(3,1);
>> y=R'\b;
>> x=R\y
     9.999999999999997e-01
     1.000000000000000e+00
     9.999999999999999e-01
>> err=norm(x-ones(3,1))/norm(ones(3,1))
```

2. Una tecnica per calcolare l'inversa di A_{ϵ^*}, problema in generale malcondizionato, è risolvere 3 sistemi del tipo $Aa_j = e_j$, dove la soluzione a_j rappresenta la j-esima colonna della matrice inversa e il termine noto e_j rappresenta la j-esima colonna della matrice identità (ricordiamo che $A_{\epsilon^*} A_{\epsilon^*}^{-1} = I$). In tale situazione dobbiamo risolvere quindi 3 sistemi con la stessa matrice dei coefficienti A_{ϵ^*} e diverso termine noto. Poiché la matrice dei coefficienti non cambia possiamo utilizzare sempre la fattorizzazione di Cholesky calcolata una volta per tutte e conservata nella matrice R del punto precedente. Abbiamo quindi:

```
>> e1=[1 0 0]'; z=R'\e1; a1=R\z;
>> e2=[0 1 0]'; z=R'\e2; a2=R\z;
>> e3=[0 0 1]'; z=R'\e3; a3=R\z;
>> invAstar=[a1,a2,a3]
```

   Verifichiamo quindi che l'inversa calcolata sia tale da dare $A_{\epsilon^*} A_{\epsilon^*}^{-1} = I$

```
>> invAstar*Astar=
9.999999999999997e-01 2.220446049250313e-16
0
                    0 1.000000000000000e+00                            0
                    0                    0     9.999999999999999e-01
```

Il risultato è la matrice identità, a meno della precisione macchina.

Esercizio 2. (Risoluzione di sistemi lineari. Metodi iterativi)

1. Costruiamo in MATLAB la famiglia di matrici A_ϵ per $\epsilon = [0 : 0.1 : 0.5]$ e calcoliamo contemporaneamente il raggio spettrale delle matrici di iterazione B_J e B_{GS}

```
>> epsilon=[0:0.1:0.5]
>> for i=1:length(epsilon)
      A=(-2+epsilon(i))*diag(ones(5,1))+...
         diag(ones(4,1),-1)+diag(ones(4,1),1);
      D=diag(diag(A));
      E=-tril(A,-1);
      F=-triu(A,1);
      BJ=inv(D)*(E+F);
      BGS=inv(D-E)*F;
      rhoJ(i)=max(abs(eig(BJ)));
      rhoGS(i)=max(abs(eig(BGS)));
   end
```

Osserviamo che, ove il raggio spettrale è minore di 1, ovvero per $\epsilon < 0.26$, il raggio spettrale del metodo di Gauss-Seidel è inferiore a quello del metodo di Jacobi (si veda la Fig. 1). In particolare, si ha che $\rho_{GS} = \rho_J^2$ e dunque il metodo di Gauss-Seidel convergerà più rapidamente.

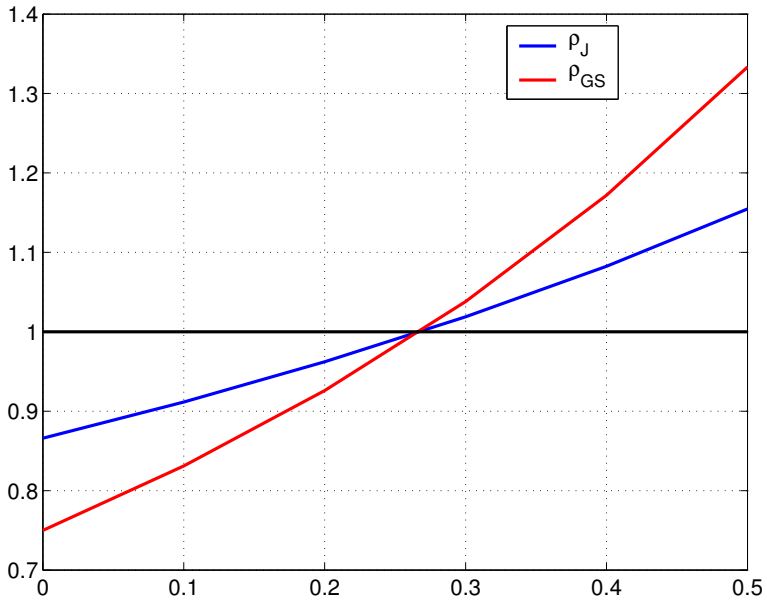

Figura 1: Raggio spettrale di B_J e B_{GS} in funzione di ϵ.

2. Verifichiamo i risultati del punto precedente:

```
>> epsilon=0.15; tol=1e-6; nmax=1000; b=A*ones(5,1);
```

```
>> A=(-2+epsilon)*diag(ones(5,1))+diag(ones(4,1),-1)+...
      diag(ones(4,1),1);
>> [x,iter] = itermeth(A,b,zeros(5,1),nmax,tol,'J')
>> [x,iter] = itermeth(A,b,zeros(5,1),nmax,tol,'G')
```

Entrambi i metodi convergono, Jacobi in 200 iterazioni, Gauss-Seidel in 95.

```
>> epsilon=0.15; tol=1e-6; nmax=1000; b=A*ones(5,1);
>> A=(-2+epsilon)*diag(ones(5,1))+diag(ones(4,1),-1)+...
      diag(ones(4,1),1);
>> [x,iter] = itermeth(A,b,zeros(5,1),nmax,tol,'J')
>> [x,iter] = itermeth(A,b,zeros(5,1),nmax,tol,'G')
```

Entrambi i metodi convergono, ma più lentamente, infatti il raggio spettrale si sta avvicinando a 1. Il metodo di Jacobi necessita di 340 iterazioni, Gauss-Seidel di 155.
Infine, se fissiamo `epsilon=0.3`, nessuno dei due metodi converge, nemmeno prendendo un numero molto elevato di iterazioni (ad ex. `nmax=5000`).

Esercizio 3. (Risoluzione dei sistemi lineari. Metodi iterativi)

1. Un metodo iterativo della forma

$$x^{(k+1)} = Bx^{(k)} + f,\ k \geq 0$$

si dirà consistente se la soluzione esatta del problema continuo, x, verifica l'equazione

$$x = Bx + f \rightarrow (I - B)x = f.$$

Quindi, se la relazione $g(\theta) = (I - B(\theta))x$ è soddifatta allora il metodo è consistente. La soluzione esatta del sistema $Ax = b$ è $x = (1, 1)^T$, da cui

$$[I - B(\theta)]\, x = \begin{bmatrix} 1 - (2\theta^2 + 2\theta + 1)/4 & -(-2\theta^2 + 2\theta + 1)/4 \\ -(-2\theta^2 + 2\theta + 1)/4 & 1 - (2\theta^2 + 2\theta + 1)/4 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{bmatrix} \frac{1}{2} - \theta \\ \frac{1}{2} - \theta \end{bmatrix} \equiv g(\theta)$$

Il metodo iterativo proposto è quindi (fortemente) consistente $\forall\theta$.

2. Condizione necessaria e sufficiente affinchè un metodo iterativo consistente converga per ogni $x^{(0)}$, è che $\rho(B) < 1$, B essendo la matrice di iterazione. Calcoliamo gli autovalori di $B(\theta)$, cioè le soluzioni dell'equazione: $\det(B(\theta) - \lambda I) = 0$.

$$\det(B(\theta) - \lambda I) = 0 \ \Leftrightarrow \ \begin{vmatrix} (2\theta^2 + 2\theta + 1)/4 - \lambda & (-2\theta^2 + 2\theta + 1)/4 \\ (-2\theta^2 + 2\theta + 1)/4 & (2\theta^2 + 2\theta + 1)/4 - \lambda \end{vmatrix} = 0 \ \Rightarrow$$

$$\left[\left(2\theta^2 + 2\theta + 1\right)/4 - \lambda\right]^2 - \left[\left(-2\theta^2 + 2\theta + 1\right)/4\right]^2 = 0 \Rightarrow \ \lambda_1 = \theta^2,\ \lambda_2 = \theta + 1/2$$

Il raggio spettrale è :

$$\rho(B(\theta)) = \max\left\{|\theta + 1/2|, \theta^2\right\}$$

$$\rho(B(\theta)) = \begin{cases} \theta^2, & \theta \in \left(-\infty,\ \frac{1-\sqrt{3}}{2}\right) \bigcup \left(\frac{1+\sqrt{3}}{2},\ +\infty\right) \\ \theta + \frac{1}{2}, & \theta \in \left[\frac{1-\sqrt{3}}{2},\ \frac{1+\sqrt{3}}{2}\right] \end{cases}$$

- Se $\theta \in \left(-\infty,\ \frac{1-\sqrt{3}}{2}\right) \bigcup \left(\frac{1+\sqrt{3}}{2},\ +\infty\right)$, allora $\rho(B(\theta)) < 1 \ \Leftrightarrow \ \theta^2 < 1 \ \Leftrightarrow$ $\theta \in (-1,1) \bigcap \left(\theta \in \left(-\infty,\ \frac{1-\sqrt{3}}{2}\right) \bigcup \left(\frac{1+\sqrt{3}}{2},\ +\infty\right)\right) \Leftrightarrow \theta \in \left(-1,\ \frac{1-\sqrt{3}}{2}\right)$
- Se $\theta \in \left[\frac{1-\sqrt{3}}{2},\ \frac{1+\sqrt{3}}{2}\right]$, allora $\rho(B(\theta)) < 1 \ \Leftrightarrow \ \theta + \frac{1}{2} < 1 \ \Leftrightarrow$ $\theta \in \left(-\infty,\ \frac{1}{2}\right) \bigcap \left[\frac{1-\sqrt{3}}{2},\ \frac{1+\sqrt{3}}{2}\right] \ \Leftrightarrow \ \theta \in \left[\frac{1-\sqrt{3}}{2},\ \frac{1}{2}\right)$.

Quindi se $\theta \in \left(-1,\ \frac{1-\sqrt{3}}{2}\right) \bigcup \left[\frac{1-\sqrt{3}}{2},\ \frac{1}{2}\right) \equiv \left(-1, \frac{1}{2}\right)$ allora il raggio spettrale $\rho(B(\theta)) < 1$ e il metodo iterativo è convergente. Possiamo anche stabilire le condizioni di convergenza del metodo iterativo in funzione di θ utilizzando Matlab e disegnando il grafico del raggio spettrale.

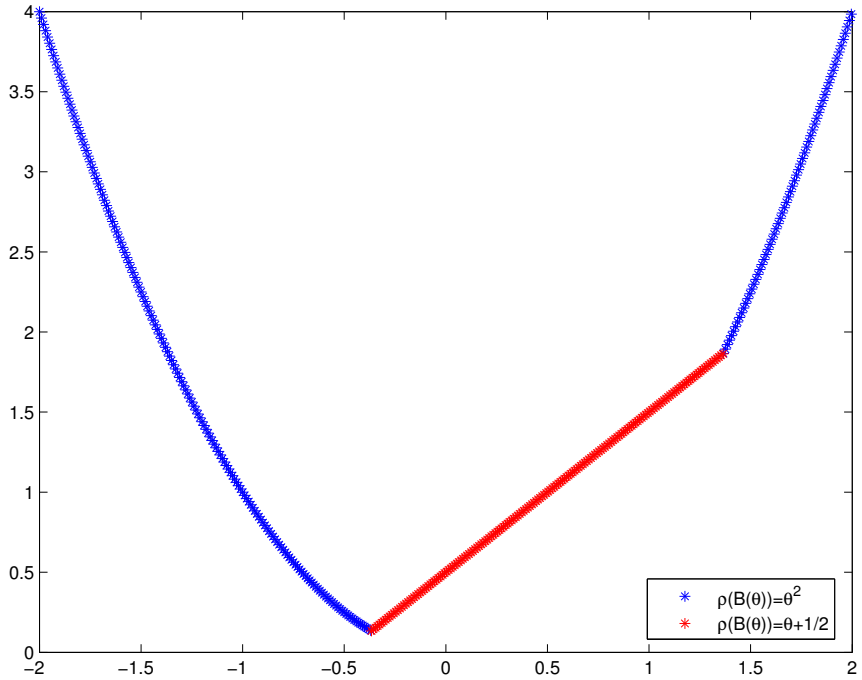

Figura 2: Grafico del raggio spettrale

3. Il valore ottimale del parametro θ deve essere scelto tale che il raggio spettrale sia minimo, nel nostro caso $\theta_{ott} = \frac{1-\sqrt{3}}{2}$.

Esercizio 4. (Approssimazione di problemi ai limiti)

1. Sia $x_0 = 0 < x_1 < \cdots < x_n = 1$ una discretizzazione dell'intervallo $(0, 1)$.
   Per ogni nodo $x_j,\ j = 1, \ldots n-1$, imponiamo che sia soddisfatto il problema ai limiti
   $$-u''(x_j) + \frac{k}{T}u(x_j) = \frac{1}{T}f(x_j)$$
   Per $u : [a, b] \to \mathbb{R}$, funzione sufficientemente regolare in un intorno di un generico punto $\overline{x} \in (a, b)$, la differenza finita centrata
   $$\delta^2 u(\overline{x}) = \frac{u(\overline{x} - h) - 2u(\overline{x}) + u(\overline{x} + h)}{h^2}$$
   fornisce un'approssimazione di $u''(\overline{x})$ di ordine 2 rispetto a h.
   In ciascun nodo $j = 1, \ldots n-1$ avremo la seguente relazione:
   $$\frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2} + \frac{k}{T}u(x_j) + \mathcal{O}(h^2) = \frac{1}{T}f(x_j)$$
2. Se ora scriviamo la relazione sopra per ogni nodo interno e transcuriamo il termine dell'errore, otteniamo un sistema lineare le cui incognite sono i valori degli spostamenti verticali (di una fune 1 metro, soggetta ad un carico trasversale di intensità $f(x)$ per unità di lunghezza) nei nodi interni. Il sistema è del tipo
   $$Au = b,$$
   dove $A \in \mathcal{M}_{n-1\times n-1},\ b \in \mathcal{M}_{n-1\times 1}$
   Per un n generico, la matrice dei coefficienti A e il termine noto b sono:
   $$A = \begin{pmatrix} 2 + \frac{k}{T}\cdot h^2 & -1 & & & & & \\ -1 & 2 + \frac{k}{T}\cdot h^2 & -1 & & & & \\ & \cdot & & & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & -1 & 2 + \frac{k}{T}\cdot h^2 & -1 & \\ & & & & -1 & 2 + \frac{k}{T}\cdot h^2 & \end{pmatrix},$$
   $$b = \begin{pmatrix} \frac{h^2}{T}f(x_1) + u(x_0) \\ \frac{h^2}{T}f(x_2) \\ \cdot \\ \cdot \\ \cdot \\ \frac{h^2}{T}f(x_{n-2}) \\ \frac{h^2}{T}f(x_{n-1}) + u(x_n) \end{pmatrix}$$

3. In Matlab:

```
function A=CostruiscoA(j,T,k); h=1/j; n=j; A=sparse(n-1,n-1);
A=-diag(ones(n-2,1),-1)+diag((2+k*h^2/T)*ones(n-1,1),0)-...
   diag(ones(n-2,1),+1);

function b=Costruiscob(j,f,k,T,u0,un); h=1/j; n=j; x=[h:h:1-h];
y=eval(f); b=zeros(n-1,1); b=y*h^2/T; b(1)=b(1)+u0;
b(n-1)=b(n-1)+un; b=b';
```

Usando le funzioni sopra con i valori $h = 1/j,\ j = 10, 20, 40$, costruiamo A e b; e con il comando \ risolviamo il sistema $Au = b$:

```
>> u0=0;
>> un=0;
>> f='1+sin(4*pi*x)';
>> T=1;
>> k=0.1;
>> j=10;
>> A10=CostruiscoA(j,T,k);
>> b10=Costruiscob(j,f,k,T,u0,un);
>> u10=A10\b10;
>> j=20;
>> A20=CostruiscoA(j,T,k);
>> b20=Costruiscob(j,f,k,T,u0,un);
>> u20=A20\b20;
>> j=40;
>> A40=CostruiscoA(j,T,k);
>> b40=Costruiscob(j,f,k,T,u0,un);
>> u40=A40\b40;
```

Come soluzione "esatta", consideriamo una soluzione approssimata relativa ad una griglia estremamente fitta, ponendo per esempio $h = 1/1000$.

```
>> j=1000;
>> A=CostruiscoA(j,T,k);
>> b=Costruiscob(j,f,k,T,u0,un);
>> u=A\b;
```

Nel vettore `u10` in posizione 1 è immagazzinato il valore della soluzione calcolata per $x = 0.1$, mentre nella prima posizione del vettore `u` vi è la soluzione per $x = 0.001$. Per trovare l'errore commesso nei nodi comuni, dobbiamo quindi percorrere il vettore `u` con passo 100, partendo dalla posizione 100 del vettore `u`.

```
>>N=length(u);
>>err10=max(abs(u10-u(100:100:N)))
>>err20=max(abs(u20-u(50:50:N)))
>>err40=max(abs(u40-u(25:25:N)))
```

Osserviamo che dimezzando h, l'errore si divide per 4, a conferma dell'ordine 2.

4. Per esempio, consideriamo il caso in cui $h = 1/10$. Con MATLAB, si verfica che $A10$ è simmetrica:

   ```
   >>sim=isequal(A10, A10')
   ```

   e definita positiva (ha tutti gli autovalori positivi)

   ```
   >>eigMIN=min(eig(A10))
   >>eigMAX=max(eig(A10))
   ```

   Quindi è verificata una condizione sufficiente perchè il metodo di Gauss-Seidel sia convergente. Utilizziamo la function `itermeth` per risolvere il sistema `A10*u10=b10` con il metodo di Gauss-Seidel:

   ```
   >>x0=zeros(1,9);
   >>tol=1e-6;
   >>nmax=100;
   >>[u10,iterGS,resGS]=itermeth(A10,b10,x0,nmax,tol,'G');
   ```

Esercizio 5. (Risoluzione di equazioni differenziali ordinarie)

1. La soluzione esatta $y = y(t)$ del problema di Cauchy si ottiene per separazione delle variabili.

   $\frac{dy}{dt} = -te^{-y} \;\Rightarrow\; \int e^y dy = \int -tdt \;\Leftrightarrow\; e^y = -\frac{t^2}{2} + \mathcal{C} \;\Leftrightarrow y(t) = \log\left(-\frac{t^2}{2} + \mathcal{C}\right)$
   Il valore della costante $\mathcal{C}$ si ottiene imponendo la condizione iniziale $y(0) = 0$, da cui $\log(\mathcal{C}) = 0 \;\Rightarrow\; \mathcal{C} = 1$. Quindi la soluzione del problema Cauchy è :

   $$y(t) = \log\left(1 - \frac{t^2}{2}\right)$$

2. Il problema Cauchy può essere scritto nella forma seguente:

   $$\begin{cases} y'(t) = f(t, y(t)) \\ y(0) = 0 \end{cases}$$

   dove $f(t, y(t)) = -te^{-y}$. Il metodo di Eulero implicito è :

   $$u_{n+1} = u_n + \Delta t \cdot f(t_{n+1}, u_{n+1}).$$

   Nel nostro caso, abbiamo:

   $$u_{n+1} = u_n + \Delta t \cdot \left(-t_{n+1} e^{-u_{n+1}}\right)$$

   e quindi dobbiamo risolvere ad ogni passo un'equazione nonlineare: $F(u_{n+1}) = 0$, dove $F(u_{n+1}) = u_{n+1} - u_n + \Delta t \cdot \left(t_{n+1} e^{-u_{n+1}}\right)$.
   Il codice MATLAB:

   ```
   y0=0; T=1;
   dt=[ 1e-1 1e-2 1e-3];
   g='log(1-t.^2/2)'; %la soluzione esatta
   ```

```
for i=1:length(dt),
     t=[0:dt(i):T];
     N=length(t);
     uEi=zeros(1,N);
     uEi(1)=y0;
     for n=1:N-1
         fun=strcat('x-',num2str(uEi(n),16),'+',...
            num2str(dt(i)*(n+1)*dt(i),16),'*exp(-x)');
         zero=fzero(fun,uEi(n));
         uEi(n+1)=zero;
     end
     y=eval(g);
     errEi(i)=max(abs(y-uEi));
end

plot(t,y,t,uEi);
legend('sol exacta','uEi');
```

3. Si ricorda che il metodo di Crank-Nicholson è :

$$u_{n+1} = u_n + \frac{\Delta t}{2} \left[f(t_n, u_n) + f(t_{n+1}, u_{n+1}) \right].$$

Nel nostro caso, abbiamo:

$$u_{n+1} = u_n - \frac{\Delta t}{2} \left[t_n e^{-y_n} + t_{n+1} e^{-y_{n+1}} \right]$$

e quindi dobbiamo risolvere un'equazione nonlineare: $F(u_{n+1}) = 0$,
dove $F(u_{n+1}) = u_{n+1} - u_n + \frac{\Delta t}{2} \left[t_n e^{-y_n} + t_{n+1} e^{-y_{n+1}} \right]$.
Il codice MATLAB:

```
for i=1:length(dt),
     t=[0:dt(i):T];
     N=length(t);
     uCN=zeros(1,N);
     uCN(1)=y0;
     for n=1:N-1
         fun=strcat('x+exp(-x)*',num2str(dt(i)/2*(n+1)*dt(i),16),'+',...
   num2str(dt(i)/2*n*dt(i)*exp(-uCN(n)),16),'-',num2str(uCN(n),16));
         zero=fzero(fun,uCN(n));
         uCN(n+1)=zero;
     end
     y=eval(g);
     errCN(i)=max(abs(y-uCN));
end
```

Esercizio 6. (Risoluzione di equazioni differenziali ordinarie)

1. Abbiamo che

$$\begin{aligned}
&\theta = 0 \rightarrow \quad u_{n+1} = u_n + hf(t_n, u_n) && \rightarrow \quad \text{Eulero esplicito}\\
&\theta = 1 \rightarrow \quad u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}) && \rightarrow \quad \text{Eulero implicito}\\
&\theta = \tfrac{1}{2} \rightarrow \quad u_{n+1} = u_n + \tfrac{1}{2}h(f(t_n, u_n) + f(t_{n+1}, u_{n+1})) && \rightarrow \quad \text{Crank-Nicolson}
\end{aligned}$$

2. (a) Se applichiamo il θ-metodo alla discretizzazione del problema modello, troviamo

$$u_{n+1} = u_n + \theta h\lambda u_{n+1} + (1-\theta)h\lambda u_n,$$

relazione che possiamo riscrivere come

$$(1 - \theta h\lambda)u_{n+1} = (1 + (1-\theta)h\lambda)u_n$$

e dunque

$$u_{n+1} = G(h\lambda;\theta)u_n = (G(h\lambda;\theta))^{n+1}y_0$$

con $G(h\lambda;\theta) = \dfrac{(1 + (1-\theta)h\lambda)}{(1-\theta h\lambda)}$

(b) Il θ-metodo è assolutamente stabile se

$$|G(h\lambda;\theta)| < 1,$$

ovvero

$$\begin{cases} 1 + \frac{h\lambda}{(1-\theta)h\lambda} < 1 \\ 1 + \frac{h\lambda}{(1-\theta)h\lambda} > -1 \end{cases}$$

Poiché $\lambda < 0$, la prima condizione è sempre soddisfatta; per quanto riguarda la seconda, abbiamo

$$\begin{cases} 1 + \frac{h\lambda}{(1-\theta)h\lambda} > -1 \rightarrow 2 + \frac{h\lambda}{(1-\theta)h\lambda} > 0 \rightarrow \\ h\lambda(1-2\theta) + 2 > 0 \rightarrow h|\lambda|(2\theta - 1) + 2 > 0. \end{cases}$$

Se $\theta \geq \frac{1}{2}$, abbiamo $h|\lambda|(2\theta-1)+2 > 0$ indipendentemente dal valore di h; invece, se $\theta < \frac{1}{2}$ dobbiamo imporre la seguente condizione su h:

$$h < \frac{2}{(1-2\theta)|\lambda|}$$

(c) Per la stabilità dello schema di Eulero esplicito, abbiamo la condizione

$$h < \frac{2}{|\lambda|},$$

risultato che quindi ritroviamo se prendiamo $\theta = 0$ nell'ultima relazione del punto precedente. Per gli schemi di Eulero implicito ($\theta = 1$) e Crank-Nicolson ($\theta = \frac{1}{2}$) ritroviamo la proprietà di incondizionata assoluta stabilità.