Mini progetto calcolo numerico

F. Bossio, F.S. Palumbo, P.I.L. Cortelazzo

A.A. 2020/2021

Sommario

In questo documento verrà studiato un problema di Dirichlet per un equazione alle derivate parziali iperbolica, con opportune condizioni iniziali e al contorno. Si vedrà una soluzione numerica al problema implementata tramite linguaggio MATLAB tramite metodo alle differenze finite esplicito, si vedrà poi un caso particolare del problema e si analizzerà infine l'andamento della soluzione trovata tramite metodi di calcolo numerico con quello della soluzione esatta, descrivendone convergenza e stabilità. Infine, verranno tratte alcune brevi conclusioni riguardo l'approssimazione numerica del problema.

1 Indice

Contents

1 Indice **1**

2 Consegna **2**

3 Introduzione **2**

4 Analisi del problema e metodi di calcolo **3**

5 Punto 1: metodo usato e implementazione in matlab **5**
5.1 Inserimento dei dati necessari all'utilizzo dei metodi alle differenze finite 5
5.2 Utilizzo dei metodi alle differenze finite 6
5.3 Calcolo degli errori 7
5.4 Plotting dei risultati 7

6 Punto 2: implementazione della soluzione esatta **10**
6.1 Risoluzione tramite metodi analitici 10
6.2 Rappresentazione grafica tramite matlab 11

7 Punto 3: paragone fra le funzioni **12**
7.1 Studio della stabilità 13
7.2 studio della convergenza 14

8 Codice MATLAB per il plotting dei grafici **17**

9 Conclusioni **19**

10 Bibliografia **19**

2 Consegna

Si consideri il seguente problema per una equazione alle derivate parziali iperbolica monodimensionale per l'incognita u(t, x)

$$\frac{\partial}{\partial t}u(t,x) + \frac{\partial}{\partial x}(\varepsilon u(t,x)) = f(t,x) \qquad 0 < t < 1, 0 < x < 1$$

Con la condizione al contorno di Dirichlet

$$\mathrm{u(t,\ 0) = 0;}$$

E la condizione iniziale

$$\mathrm{u(0,\ x) = \omega(x)}$$

Le richieste del mini progetto sono:

1. Si implementi in linguaggio MATLAB un solutore per il problema differenziale pro- posto basato su un appropriato metodo alle differenze finite esplicito, con passo di discretizzazione spaziale uniforme h e passo di integrazione in tempo costante δt.

2. Si ponga $\varepsilon = 10$ e si scelgano f(t, x), g(t) e ω(x) in modo che la soluzione esatta del problema differenziale proposto risulti

$$\mathrm{u(t,x)=x(\sin(\pi(x\text{-}\varepsilon t)))}$$

3. Si risolva il problema differenziale proposto con il codice implementato al primo punto e si confronti il risultato con la soluzione esatta per studiare la convergenza e la stabilità del metodo al variare di h e δt

3 Introduzione

$$\begin{cases} \frac{\partial}{\partial t}u(t,x) + \frac{\partial}{\partial x}(\varepsilon u(t,x)) = f(t,x) & 0 < t < 1,\ 0 < x < 1 \\ u(t,0) = 0 \\ u(0,x) = \omega(x) \end{cases}$$

Questo è un problema ai limiti evolutivi iperbolico del primo ordine, poichè oltre a venire fornita un equazione alle derivate parziali del primo ordine iperbolica vengono fornite anche la condizione iniziale e al contorno.
Il problema fornito è facilmente riconducibile ad un problema del trasporto scalare non omogeneo monodimensionale, dove "epsilon" è la velocità di propagazione.
Per risolvere agevolmente il problema del trasporto in modo analitico si potrebbe utilizzare il metodo delle caratteristiche.
Per il solutore verranno utilizzati metodi espliciti di approssimazione alle differenze finite, quali Lax-Friedrichs, Lax-Wendroff e Upwind. Se ne studieremo i funzionamenti, i pro e i contro nella scelta di ognuno di essi, la stabilità e la convergenza.

4 Analisi del problema e metodi di calcolo

Per poter risolvere una generica equazione differenziale alle derivate parziali sono disponibili tre gruppi di metodi di calcolo: Il metodo degli elementi finiti, il metodo alle differenze finite e il metodo dei volumi finiti.

Noi, come da richiesta, ci siamo concentrati suoi metodi alle differenze finite, in particolar modo quelli espliciti.

Per poter tuttavia procedere con l'introduzione dei metodi di calcolo è necessario definire il tipo di discretizzazione.È richiesta una discretizzazione omogenea sia spaziale che temporale. Va quindi diviso lo spazio in un numero j di segmenti di dimensione dx e il tempo in un numero n di intervalli finiti dt.

Fatto ciò si procede a calcolare il rapporto $\lambda = dt/dx$ che sarà di fondamentale importanza per i metodi di calcolo seguenti. A riprova di ciò ogni metodo alle differenze finite di tipo esplicito può essere generalizzato nella forma

$$u_j^{n+1} = u_j^n - \lambda(h_{j+1/2}^n - h_{j-1/2}^n)$$

Dove u è la soluzione al istante di tempo n e nella posizione j mentre h(,) è il flusso numerico che varia in base al metodo.

Andiamo quindi ora ad analizzare i 4 metodi di discretizzazione applicabili al nostro caso valutando quali possano essere applicati e quali no. Ci riferiremo spesso alla condizione di Courant, Friedrichs, Lewy (in breve CFL) per comprendere la stabilità di un metodo poiché quest'ultima risulta essere una condizione necessaria e sufficiente affinché un metodo alle differenze finite esplicito sia stabile.

Nelle seguenti formule ricordiamo a è la velocità dell'onda necessariamente reale e positiva.

1. Eulero in avanti/centrato

$$u_j^{n+1} = u_j^n - \frac{\lambda}{2}a(u_{j+1}^n - u_{j-1}^n)$$

   Il metodo di Eulero in avanti/centrato ha come flusso numerico:

$$h_{j+1/2}^n = \frac{1}{2}a(u_{j+1}^n + u_j^n)$$

   Il metodo di Eulero in avanti risulta il più intuitivo, tuttavia usando l'analisi della stabilità di Von Neumann è possibile dimostrare che quest'ultimo è incondizionatamente instabile per qualsiasi h>0 e Δt >0. L'instabilità lo esclude a priori dalla sua applicazione in quanto non porterà alcun risultato accettabile.

2. Lax-Friedrichs

$$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) - \frac{\lambda}{2}a(u_{j+1}^n - u_{j-1}^n)$$

   Il metodo di Lax-Friedrichs il cui flusso numerico è:

$$h_{j+1/2}^n = \frac{1}{2}[a(u_{j+1}^n + u_j^n) - \lambda^{-1}(u_{j+1}^n + u_j^n)]$$

   E risulta stabile secondo l'analisi della stabilità di Von Neumann a patto che la sua condizione CFL sia rispettata.

Il metodo presenta come errore di troncamento $\tau(h, \Delta x)$ pari a $O(\Delta x^2/\Delta t + \Delta t + \Delta x)$
Il metodo rientra nei metodi espliciti e può essere usato per risolvere il nostro problema se implementato correttamente.

3. Lax-Wendroff

$$u_j^{n+1} = u_j^n - \frac{\lambda}{2}a(u_{j+1}^n - u_{j-1}^n) + \frac{\lambda^2}{2}a^2(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

Il metodo di Lax-Wendroff il cui flusso numerico è:

$$h_{j+1/2}^n = \frac{1}{2}[a(u_{j+1}^n + u_j^n) - \lambda a^2(u_{j+1}^n + u_j^n)]$$

E risulta stabile per l'analisi di Von Neumann finché la condizione CFL è rispettata. Quest'ultimo presenta un errore di troncamento $\tau(h, \Delta x)$ pari a $\mathrm{O}(\Delta t^2 + \Delta x^2)$

Il metodo di Lax-Wendroff è spesso il più preciso per le giuste condizioni, ma produce un leggero sfasamento rispetto alla soluzione esatta che in questo caso è tollerabile.

4. Upwind (Eulero avanti/decentrato)

$$u_j^{n+1} = u_j^n - \frac{\lambda}{2}a(u_{j+1}^n - u_{j-1}^n) + \frac{\lambda}{2}|a|(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

Il metodo di Upwind (Eulero in avanti/decentrato) ha il seguente flusso numerico:

$$h_{j+1/2}^n = \frac{1}{2}[a(u_{j+1}^n + u_j^n) - |a|(u_{j+1}^n + u_j^n)]$$

Come i precedenti due metodi, anche il metodo di Upwind risulta stabile per l'analisi di Von Neumann e suo errore di troncamento $\tau(h, \Delta x)$ è $\mathrm{O}(\Delta t + \Delta x)$.

5 Punto 1: metodo usato e implementazione in matlab

5.1 Inserimento dei dati necessari all'utilizzo dei metodi alle differenze finite

Per la risoluzione del problema sono stati utilizzati i metodi di Lax-Friedrichs, Lax-Wendroff e Upwind. Il metodo di Eulero in avanti non è stato considerato per via della sua instabilità, come descritto nelle pagine precedenti.
Come primo passo è necessario inizializzare tutti i parametri richiesti dal punto 1:

```
%% intervalli spaziali e temporali utilizzati
tspan = [0,1];
xspan = [0,1];
%% parametri da passare al file metodi_dfe.m
cfl = 0.8; %condizione di stabilit  cfl, per i metodi espliciti deve essere minore di 1
dt = 1.e-3; %passo temporale
dx = 1.e-3;%passo spaziale
xx= xspan(1):dx:xspan(2);
tt= tspan(1):dt:tspan(2);
%ricaviamo un vettore xx contenente tutti i passi spaziali all'interno del
%nostro span spaziale, e facciamo la stessa cosa con tt per lo span
%temporale
%% parametri utili per il plotting
t=tt(500); %istante temporale
x_x= 500; %punto spaziale
%% condizioni imposte al problema
u0 = @(x) sin(pi.*x);   %condizione iniziale, per t=0
ul = @(t) 0*t;          %condizione al contorno, per x=0
```

Si passa poi a creare una funzione "originale" di test che rispetti le condizioni imposte dal problema e che consenta di verificare l'efficacia del solutore e la sua precisione nell'approssimazione

```
%% funzione originale di cui cerchiamo l'approssimazione, utile per il confronto e ...
    l'analisi degli errori di approssimazione dei vari metodi
uex_t = @(x,t) (t+1).*sin(pi.*t.*x);
```

A questo punto viene inserita la funzione f(x, t), forzante dell'equazione differenziale, calcolata precedentemente. Questa funzione sarà essenziale per la risoluzione del problema in quanto andrà sommata ad ogni passo con le altre componenti nei vari metodi per garantire una migliore precisione dell'approssimazione

```
%% forzante, ricavata tramite metodi analitici, utile per l'approssimazione
%generazione del function handler
forzante=@(x,t) sin(pi.*x) + 10*pi.*(t+1).*cos(pi.*x);
%conversione da funzione a matrice uu_f
uu_f = zeros(size(xx,2), size(tt,2));
for n = 1:size(uu_f,2)-1
   for j = 2:size(uu_f,1)-1
    uu_f(j,n) = forzante(j*dx,n*dt);
   end
end
```

Il function handler della forzante è stato poi convertito in una matrice per poterne più facilmente studiare i valori.
È stato necessario introdurre un rescale della forzante, poichè senza di esso il suo contributo avrebbe influenzato in modo eccessivo i metodi di approssimazione, invalidandoli

```
%% rescale della funzione uu_f
%necessario alla comparazione e alla rappresentazione
uu_f(1:end,:)=uu_f(1:end,:)./60;
```

5.2 Utilizzo dei metodi alle differenze finite

Viene ora fatta una chiamata alla funzione all'interno del file **metodi dfe**, il cui scopo è quello di ricevere in ingresso i valori del problema dal file main, applicare i metodi delle differenze finite espliciti e fornire in output delle matrici relative alle funzioni approssimate:

```
%% chiamata della funzione metodi_dfe e soluzione con i 3 metodi espliciti
[ uu_lf] = metodi_dfe(tspan, xspan, u0, ul, 1, cfl, dx, dt, xx, tt, uu_f);  % metodo di ...
    Lax Friedrichs
[ uu_lw] = metodi_dfe(tspan, xspan, u0, ul, 2, cfl, dx, dt, xx, tt, uu_f); % metodo di ...
    Lax Wendroff
[ uu_up] = metodi_dfe(tspan, xspan, u0, ul, 3, cfl, dx, dt, xx, tt, uu_f);  % metodo di ...
    Upwind
```

All'interno di metodi dfe vengono generati il rapporto λ e la velocità ϵ, inoltre viene creata la matrice uu a cui vengono inizialmente applicate le condizioni iniziali e di bordo del problema

```
% parametri lambda ed epsilon
lambda= dt/dx;
epsilon=cfl*lambda^(-1);
%ricaviamo epsilon ( velocit  (utilizzata nei metodi delle differenze
%finite espliciti)

%% inizializzazione uu
uu= zeros(size(xx,2), size(tt,2));
%inizializziamo la matrice uu, che rappresenta la funzione u(t,x), tramite
%una serie di zeri di dimensioni pari a quelle di xx per quelle di tt

%% condizioni iniziali (u0) e al contorno (ul)
uu(:,1) = u0(xx);
uu(1,:)= ul(tt);
```

Qui si possono vedere nel dettaglio i metodi utilizzati, utilizzando la matrice **uu** inizializzata in precedenza :

```

 if scheme == 1 % Lax Friedrichs
    for n = 1:size(uu,2)-1
      for j = 2:size(uu,1)-1
            uu(j,n+1) = 0.5*(uu(j+1,n)+uu(j-1,n))- ...
                0.5*lambda*epsilon*(uu(j+1,n)-uu(j-1,n))+uu_f(n,j+1).*dt;

      end
    end

  elseif scheme == 2 % Lax Wendroff
    for n = 1:size(uu,2)-1
      for j = 2:size(uu,1)-1
        uu(j,n+1) = uu(j,n) - 0.5*lambda*epsilon*(uu(j+1,n)-uu(j-1,n)) + ...
            0.5*lambda^2*epsilon^2*(uu(j+1,n)-2*uu(j,n) + uu(j-1,n))+uu_f(n,j+1).*dt;

      end
      end

  elseif scheme == 3 % Upwind

    for n = 1:size(uu,2)-1
      for j = 2:size(uu,1)-1
        uu(j,n+1) = uu(j,n) - 0.5*lambda*epsilon*(uu(j+1,n)-uu(j-1,n)) + ...
            0.5*lambda*abs(epsilon)*(uu(j+1,n)-2*uu(j,n)+uu(j-1,n))+uu_f(n,j+1).*dt;

      end
    end
 end
```

5.3 Calcolo degli errori

Qui si può vedere il calcolo degli errori assoluti tra la funzione originale e le sue approssimazioni:

```
%% -------- errore nell'applicare i metodi ----------

%errori assoluti spaziali di approssimazione per un istante t fissato:
for i=1:size(xx,2)
    err_lf_sp(i,1)=abs(uex_t(i,round(t/dt))-uu_lf(i,round(t/dt)));
end
for i=1:size(xx,2)
    err_lw_sp(i,1)=abs(uex_t(i,round(t/dt))-uu_lw(i,round(t/dt)));
end
for i=1:size(xx,2)
    err_up_sp(i,1)=abs(uex_t(i,round(t/dt))-uu_up(i,round(t/dt)));
end
%errori assoluti di approssimazione temporali in un determinato punto x_x:
  err_lf_temp = abs(uex_t(xx(x_x), tt)-uu_lf(x_x,:));
  err_lw_temp = abs(uex_t(xx(x_x), tt)-uu_lw(x_x,:));
  err_up_temp = abs(uex_t(xx(x_x), tt)-uu_up(x_x,:));
```

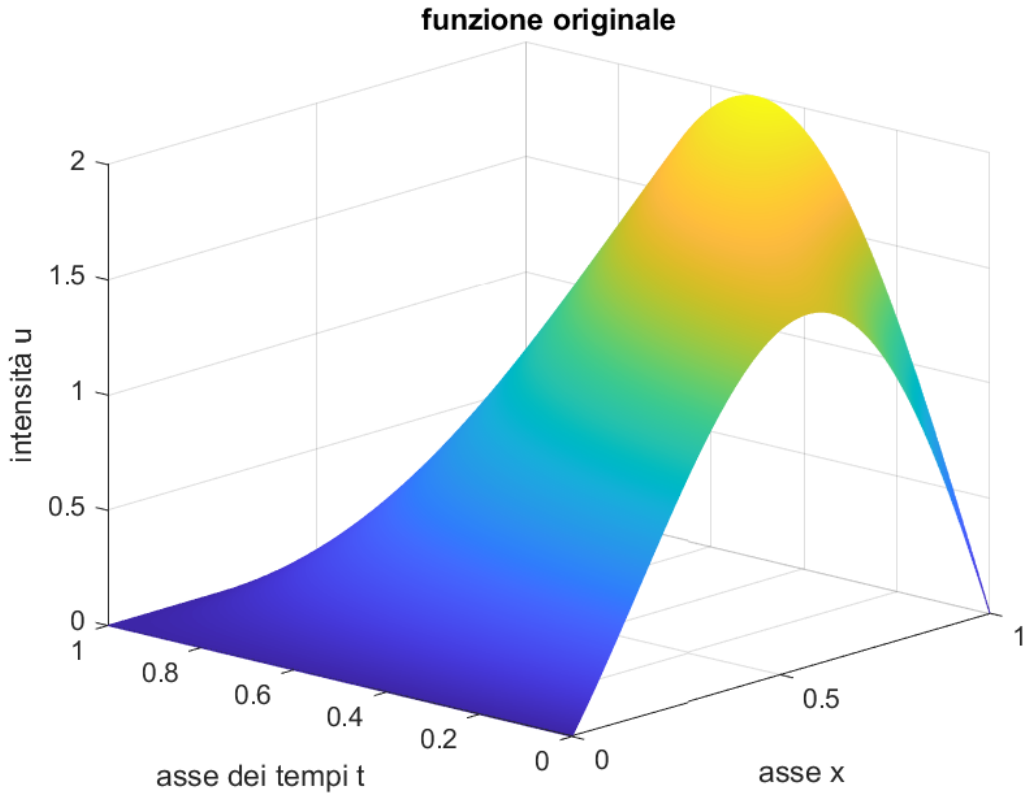
Si può notare come gli errori siano stati divisi in errori spaziali e temporali, per poter analizzare lo scostamento dell'approssimazione dalla funzione originale in entrambi gli assi.

La trattazione dell'errore verrà approfondita nel dettaglio nel punto 3.

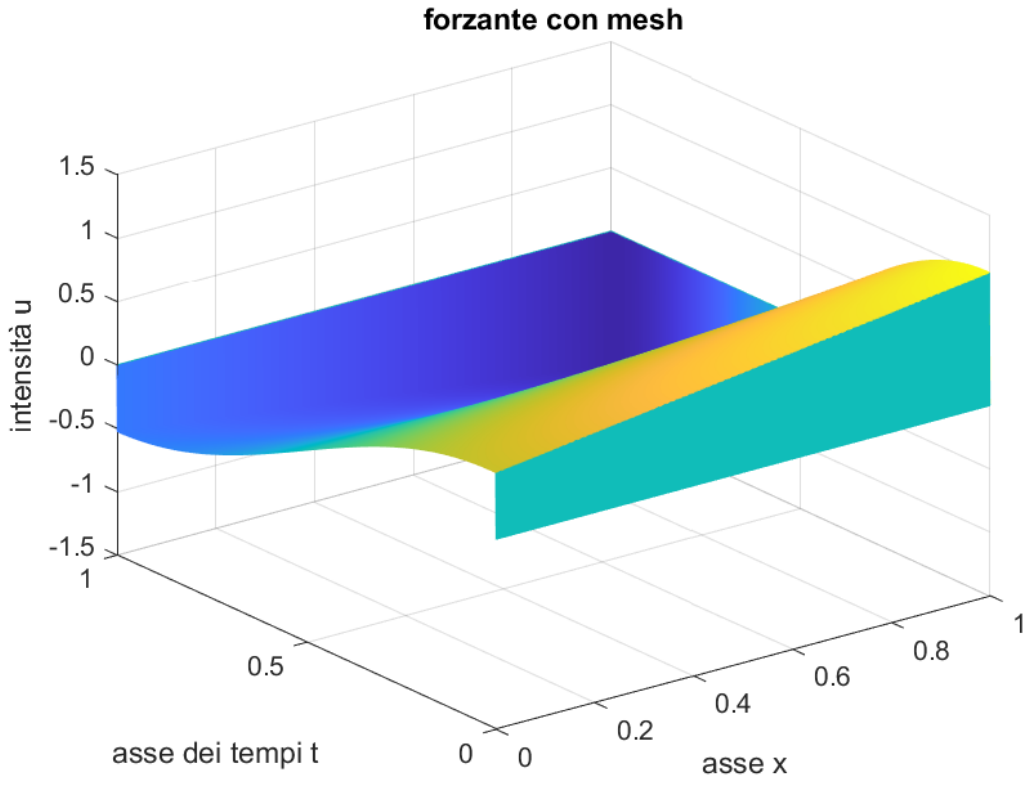
5.4 Plotting dei risultati

Una volta approssimata la funzione, generate le matrici relative alle approssimazioni e agli errori, si passa ad analizzarne i risultati tramite grafici.
Per via della quantità di righe e della sua scarsa utilità ai fini del primo punto, la parte di codice relativo al plotting dei grafici verrà posta in un appendice alla fine del documento. Qui verranno mostrati i risultati finali del plotting, con:

- Una funzione di prova pari a **(t+1)*sin(pi*t*x)**
- Una condizione **CFL=0.8**
- Un passo temporale pari a **dt=1*10^{-3}**
- Un passo temporale pari a **dx=1*10^{-3}**
- Istante temporale per il grafico dell'andamento spaziale della funzione **t=500*dt**

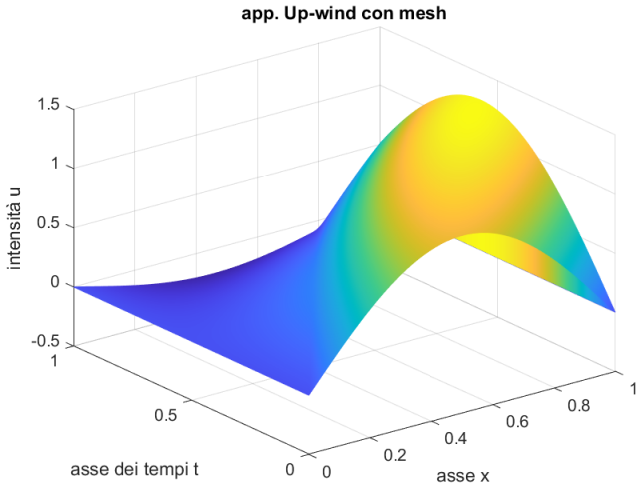


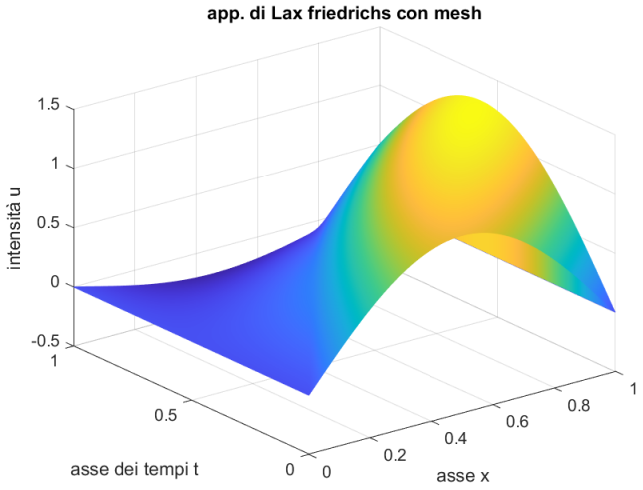
(t+1)*sin(pi*t*x)

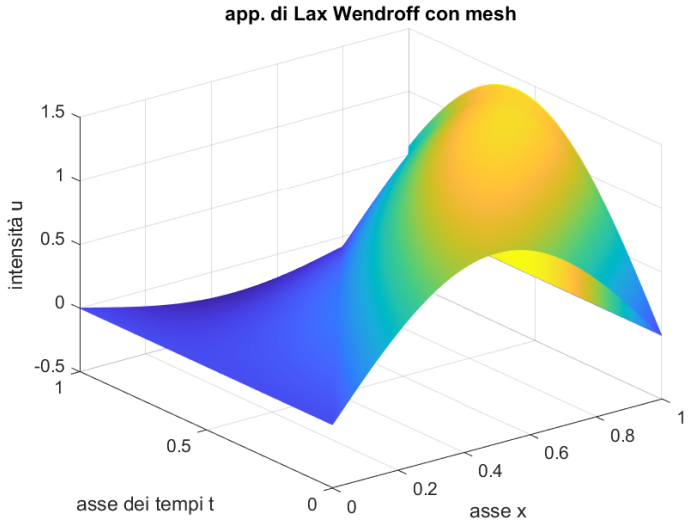


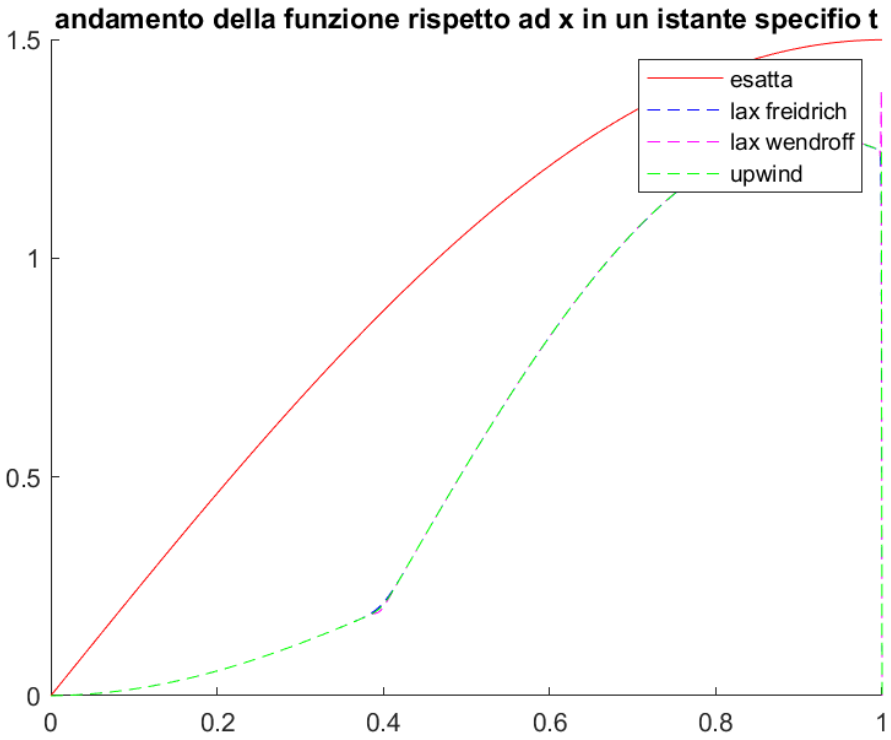
sin(pi*x)+ 10*pi*(t+1)*cos(pi*x)

Si può notare come le approssimazioni abbiano una perdita di ampiezza ed uno sfasamento rispetto alla funzione originale









6 Punto 2: implementazione della soluzione esatta

6.1 Risoluzione tramite metodi analitici

Andiamo ora a calcolare f(t,x), g(t) e w(x) affinché la u(t,x) sia la soluzione dell'equazione differenziale alle derivate parziali iperboliche in esame.

Si parte da

$$F(t,x) = \frac{d}{dt}(u(t,x)) + \varepsilon * \frac{d}{dx}(u(t,x))$$

In questa si impone il vincolo sulla velocità ε=10 ottenendo

$$F(t,x) = \frac{d}{dt}(u(t,x)) + 10 * \frac{d}{dx}(u(t,x))$$

Arrivati a questo punto la prima cosa da ricavare è la forzante f(t,x) e per fare ciò si può usare la soluzione fornita.

Si sostituisce ora nell' ultima equazione ricavata la seguente funzione:

$$u(t,x) = xsin(\pi(x - 10t))$$

E si procede a derivare nel tempo e nello spazio ottenendo:

$$f(t,x) = \cancel{-10\pi xcos(\pi(x - 10t))} + 10sin(\pi(x - 10t)) + \cancel{10\pi xcos(\pi(x - 10t))}$$

Che adeguatamente semplificata porta al valore della forzante f(t,x) che è:

$$f(t,x) = 10sin(\pi(x - 10t))$$

Trovata la forzante si prosegue calcolando w(x) utilizzando la condizione iniziale U(x,0)=w(x) Sostituendo la condizione nell'equazione iniziale e al posto di u(t,x) e imponendo t=0, si otterrà un equazione composta unicamente da w(x) e l'incognita x:

$$\frac{\partial}{\partial t}w(x) + \varepsilon\frac{\partial}{\partial x}w(x) = f(0,x)$$

Procedendo a calcolare derivata facendo attenzione che la derivata nel tempo di w(x) è nulla visto che w dipende unicamente da x. Otteniamo quindi:

$$10\frac{\partial}{\partial x}w(x) = f(0,x)$$

Da qui per poter ottenere w(x) basta integrare f(0,x) in dx e dividerlo per la velocità ottenendo:

$$w(x) = \int f(0,x)dx = \frac{-cos(\pi x)}{\pi}$$

In modo analogo per calcolare g(t). Si riprende l'equazione (1.1) e al posto di u(t,x), si inserisce g(t) e si impone x=0. Come nel caso precedente g(t) è indipendente da x allora la sua derivata in x sarà nulla ottenendo:

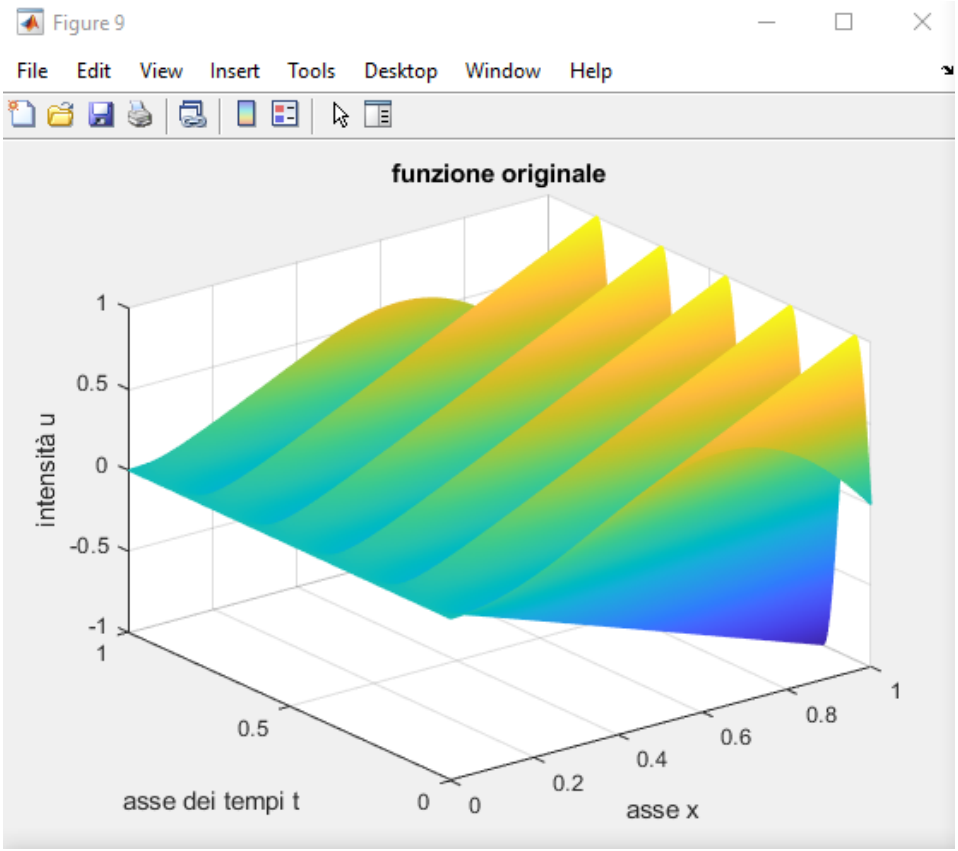
$$\frac{\partial}{\partial t}g(t) = f(t,0)$$

Quindi integrando f(t,0) in dt otteniamo che g(t)

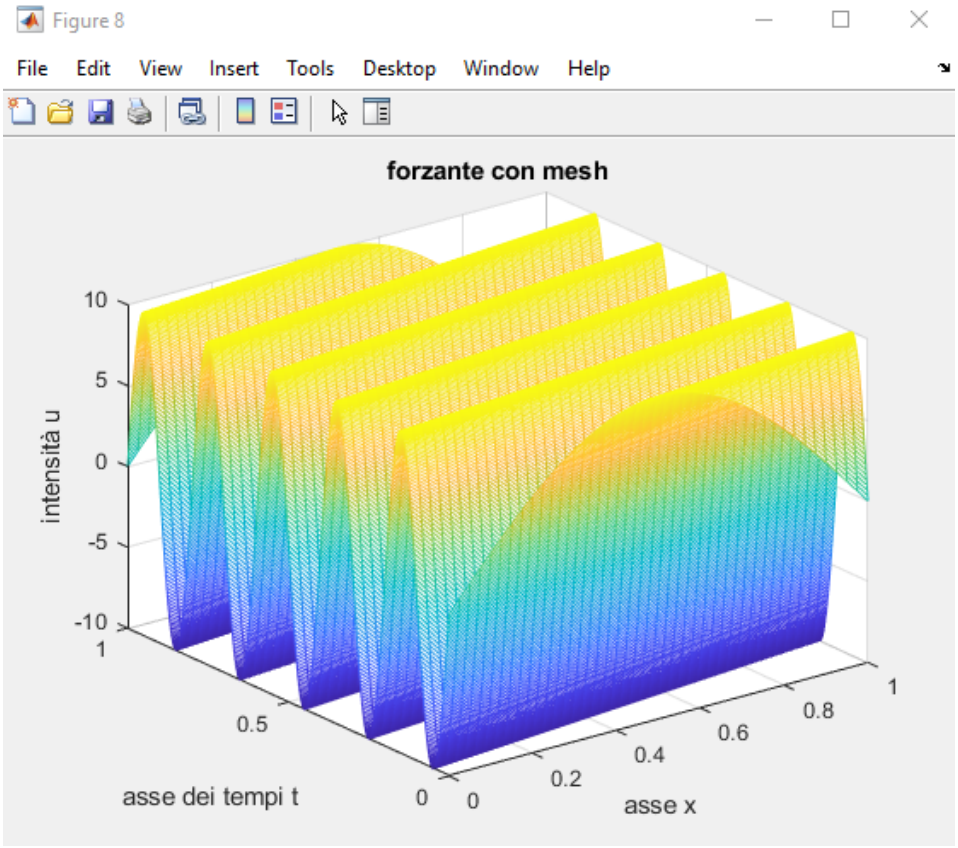
$$g(t) = \int f(t,0)dt = \frac{cos(10\pi t)}{\pi}$$

6.2 Rappresentazione grafica tramite matlab

È possibile utilizzare il linguaggio Matlab per rappresentare graficamente sia la funzione esatta



Sia l'andamento della forzante calcolata



Così da riuscire ad avere una idea della forma sia della soluzione che della forzante trovata.

7 Punto 3: paragone fra le funzioni

Come illustrato in precedenza, per le equazioni differenziali alle derivate parziali iperboliche si hanno a disposizione 3 metodi espliciti stabili. Non essendo questi particolarmente complessi da implementare, sono stati utilizzati tutti.

Sono stati quindi inseriti i valori forniti e calcolati nel punto 2 dentro il codice matlab:

sono state inserite le seguenti condizioni iniziali ed al contorno

```
u0= @(x)-(cos(pi.*x)./pi);
ul= @(t)(cos(pi.*t)./pi);
```

Sono poi state inserite la forzante e la funzione originale uex t:

```
forzante= @(x,t)10.*(sin(pi.*(x-10*t)));
uex_t= @(x,t)x.*(sin(pi.*(x-10*t)));
```

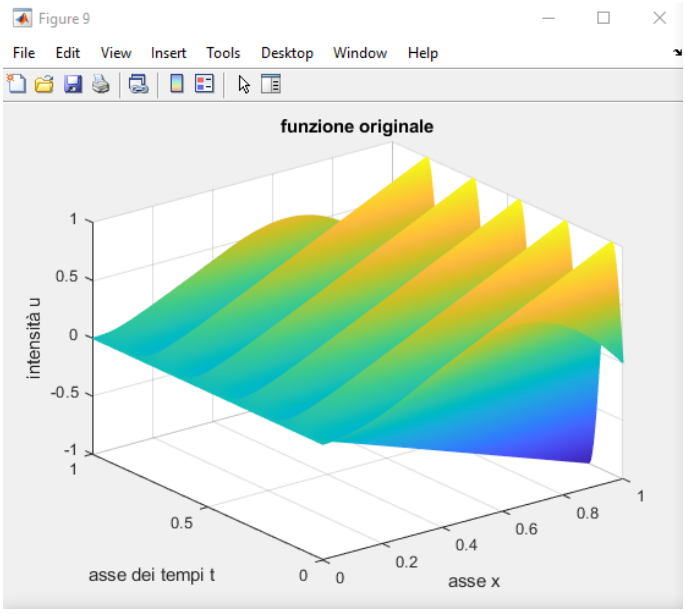
Infine sono stati determinati i parametri necessari affinchè i metodi siano stabili e sia applicata una discretizzazione omogenea sia nello spazio che nel tempo:

```
tspan=[0,1];
xspan=[0,1];
cfl=0,8;
dt=4.e-4;
dx=50.e-4;
```

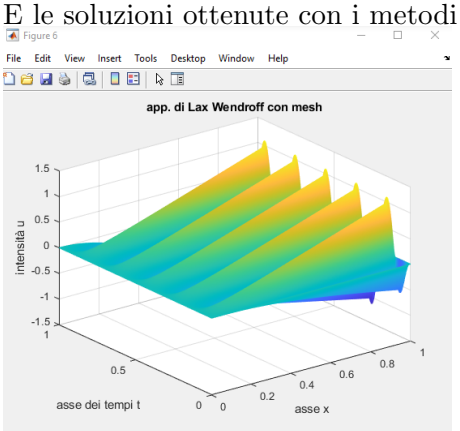
I primi parametri impostati sono stati il dominio della funzione e la condizione CFL. Avendo deciso di scegliere manualmente i parametri per soddisfare la condizione CFL sono stati calcolati degli intervalli di discretizzazione che rispettano i vincoli sulla velocità:

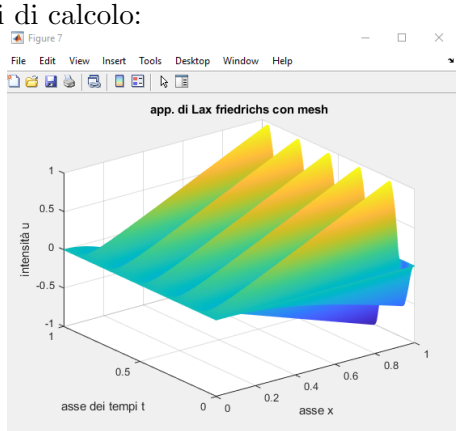
```
lambda= dt/dx;
epsilon=cfl*lambda^(-1);
```

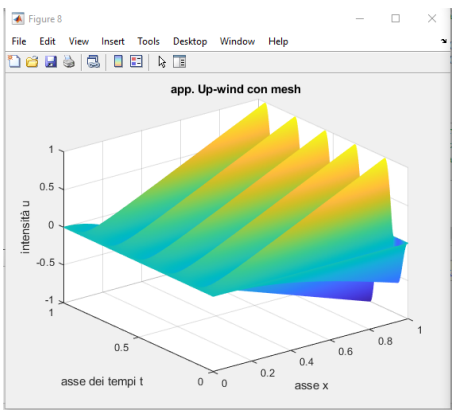
Così impostato il codice fornisce la soluzione esatta:



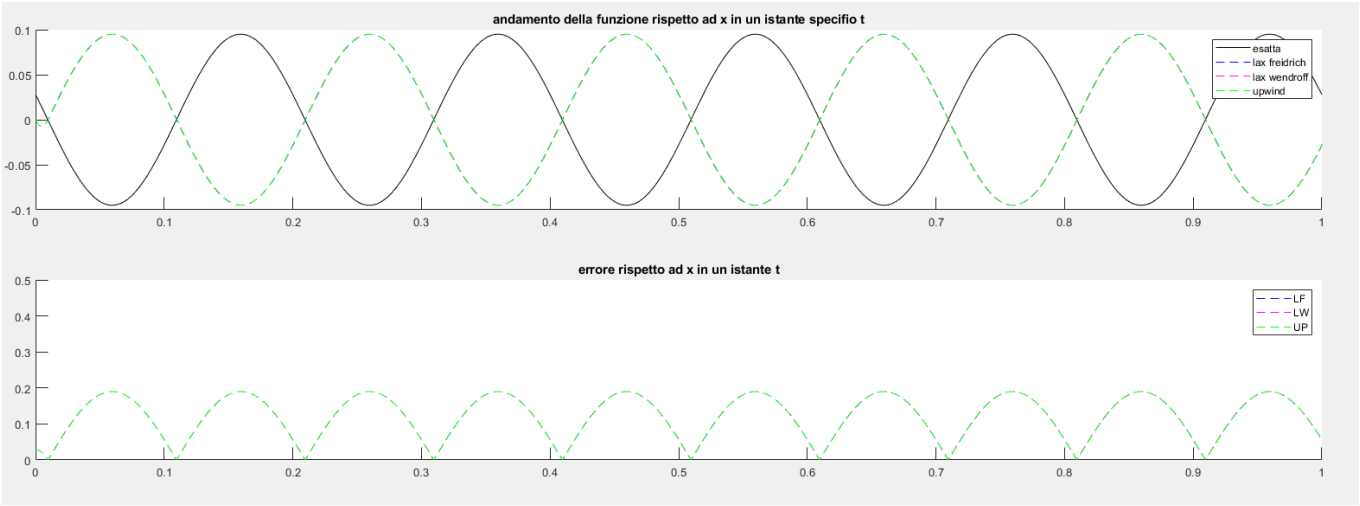
E le soluzioni ottenute con i metodi di calcolo:







È possibile notare che indipendentemente dal metodo di calcolo, tutti e 3 soffrono nel medesimo picco di errore iniziale e ciò causa uno sfasamento netto tra il metodo di calcolo e la funzione esatta. Siccome la soluzione esatta è periodica, avere uno sfasamento notevole come quello rilevato causa dei picchi periodici e di uguale intensità nell'errore di approssimazione.
La causa dello sfasamento è da attribuirsi al metodo di calcolo stesso che necessita di alcune iterazioni prima di entrare "a regime" e quindi di produrre dei risultati coerenti e soddisfacenti.



Con l'ausilio di alcuni grafici 2D è possibile farsi un'idea del comportamento e dell'andamento periodico dell'errore. Si nota immediatamente come tutti e tre i metodi presentino uno sfasamento rispetto alla soluzione originale e di come quest'ultimo causi un errore periodico per via della natura periodica della funzione.

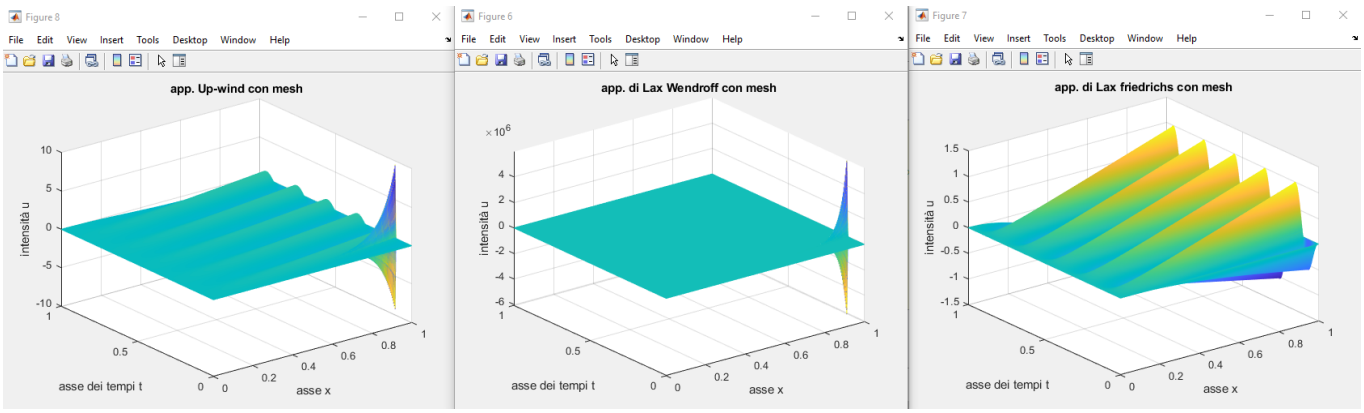
Siccome la funzione esatta e quella stimata sono quasi in controfase è possibile osservare dal grafico soprastante come l'errore di approssimazione rispecchi sia la periodicità che lo sfasamento. Si avrà infatti un picco di errore quando la funzione reale raggiunge un massimo o un minimo, in quanto la versione approssimata al medesimo istante indicherà il contrario. Infine l'errore si azzera ogni volta che il metodo numerico e la soluzione esatta si sovrappongono per poi tornare ad aumentare.

7.1 Studio della stabilità

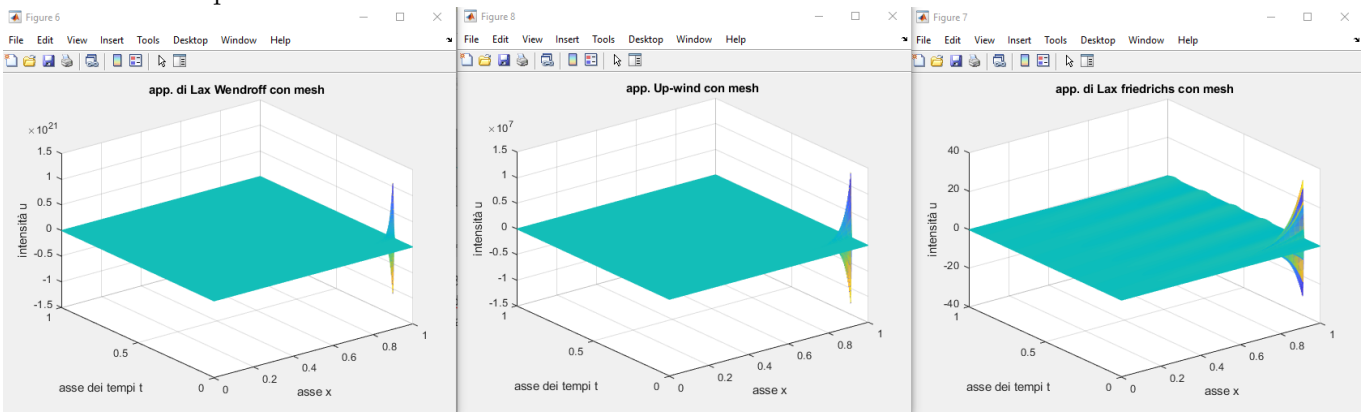
È possibile dimostrare che se la condizione CFL è rispettata tutti e tre i metodi di calcolo risultano stabili per velocità positive, altrimenti i metodi risultano instabili e non più idonei all'approssimazione. Per poter osservare il comportamento instabile dei metodi è necessaria una piccola modifica al codice Matlab così da non rispettare più la condizione CFL

```
dt=4.e-4;
dx=39.e-4;
cfl=10*(dt./dx);
```

Così facendo la CFL=1,025 e i metodi non sono più stabili ottenendo:



Dei tre metodi quello che diverge per primo risulta essere Lax-sendroff, seguito da Upwind e da Lax-Friedrichs che nonostante la CFL non sia rispettata riesce ancora ad approssimare la funzione. Tuttavia basta portare la CFL a 1,052 impostando dx a 38e-4 per vedere come anche Lax-Friedrichs necessiti della condizione CFL per mantenere la sua stabilità.

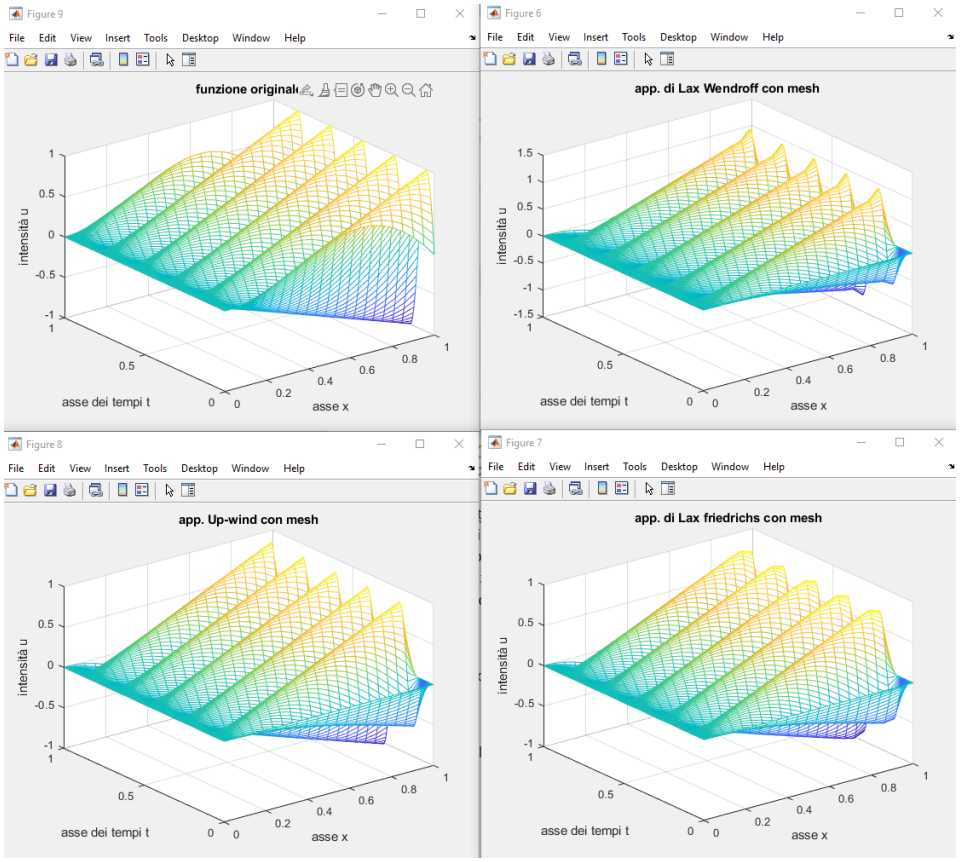


7.2 studio della convergenza

Per studiare invece la convergenza del metodo è necessario applicare

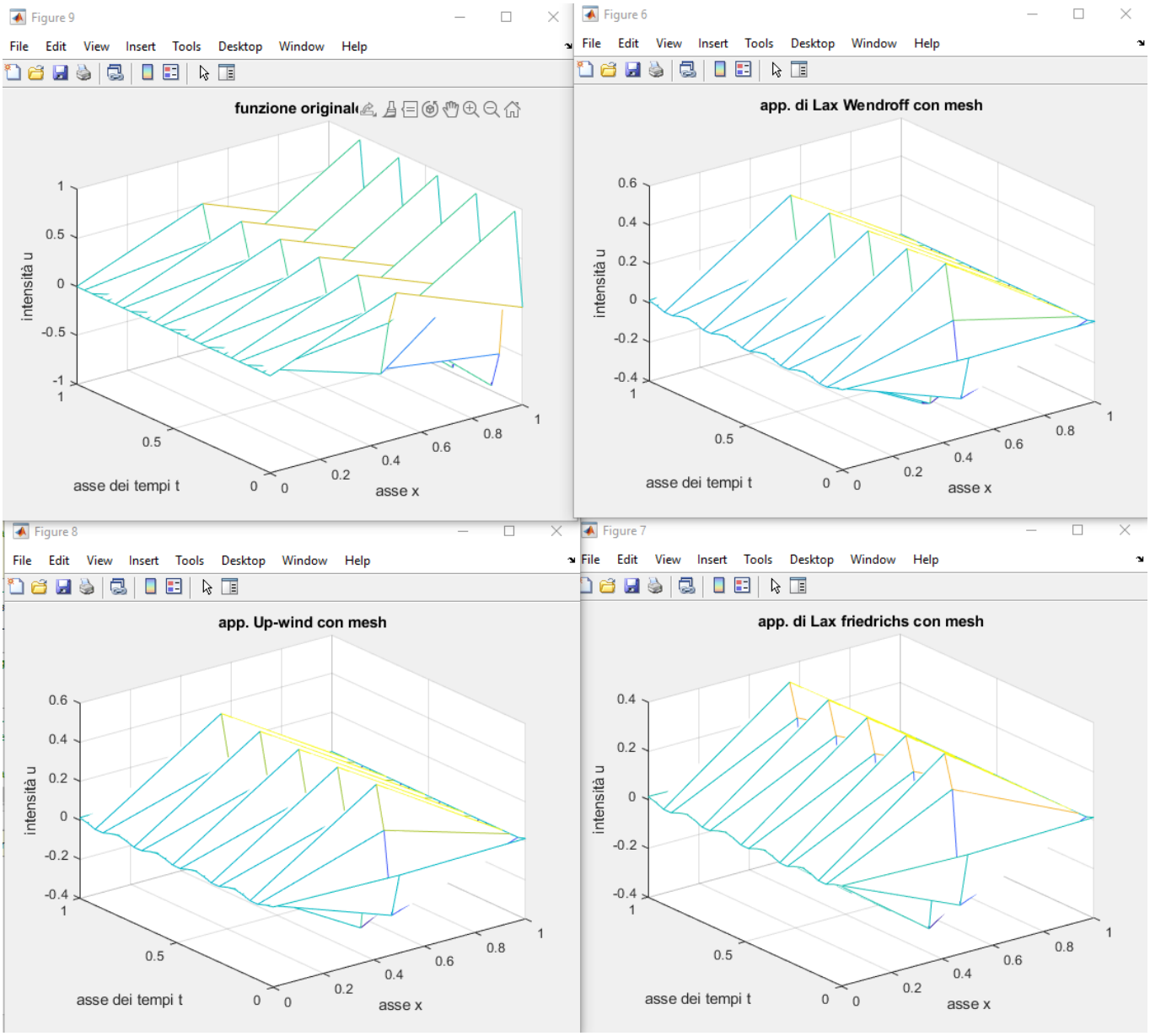
$$lim_{\Delta t,\Delta x \to 0} max_{j,n} |u(x_j, t^n) - u_j^n| = 0$$

Dove u(x_j,t_n) è il valore esatto mentre u_j^n è quello ottenuto dai metodi di calcolo. È evidente come questa richiesta sia impossibile da rispettare se non in linea teorica. In una visione più pratica è possibile dire che il metodo converge se non si discosta in modo eccessivo dal valore reale. Risulta logico che con passi di discretizzazione sempre più piccoli e prossimi allo zero si riesca ad ottenere una approssimazione sempre più precisa aumentando di conseguenza il quantitativo di calcoli da effettuare. Per osservare ciò basta modificare i passi nel codice matlab, in questo caso moltiplicando i passi di discretizzazione di un fattore 10 portandoli da dt=4*10^{-4} e dx=50*10^{-4} rispettivamente a dt=4*10^{-3} e dx=50*10^{-3}. Il risultato di ciò è visibile nei grafici sottostanti della medesima funzione.

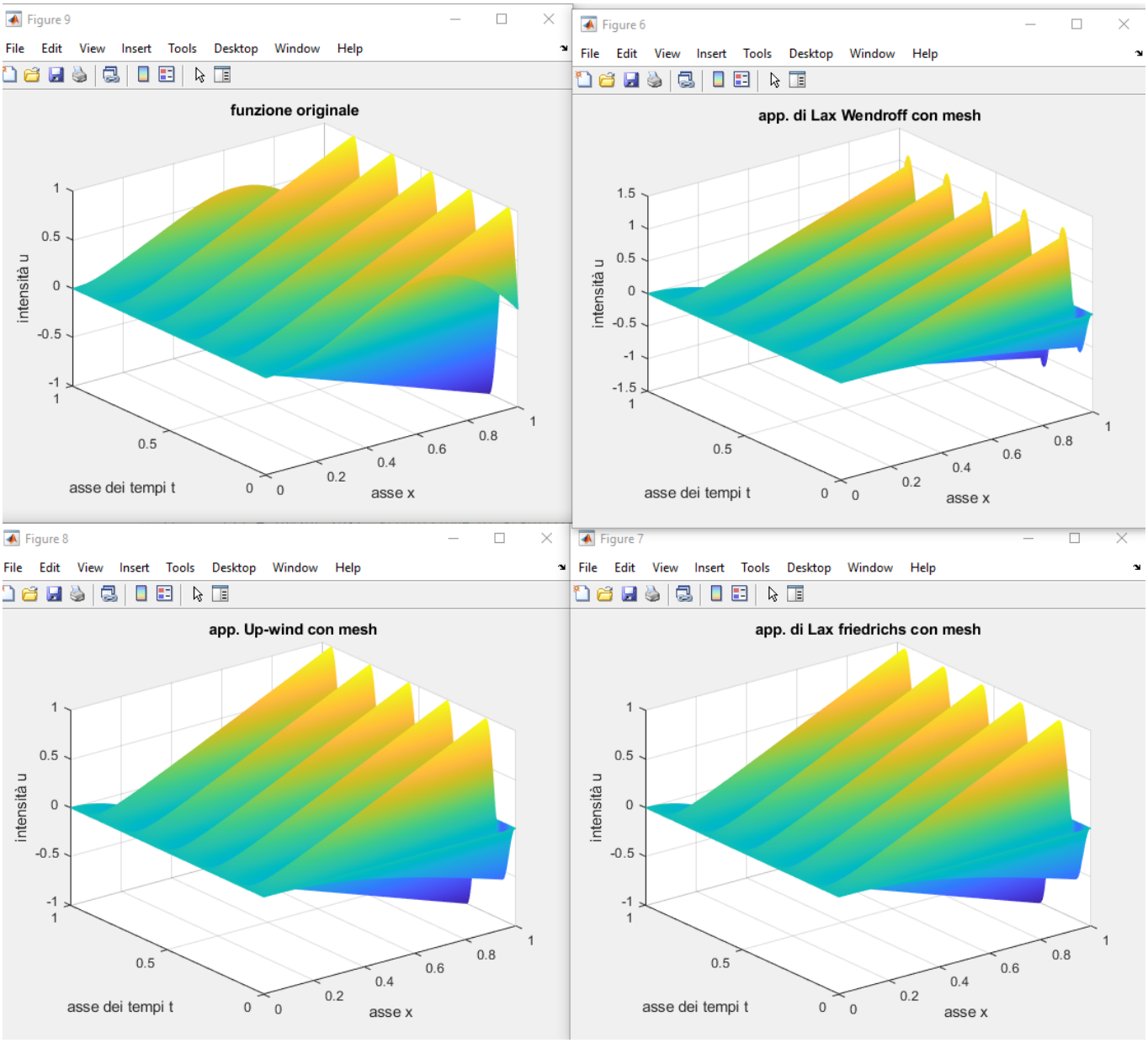


Con questa semplice riscaldatura è già evidente come i metodi di calcolo fatichino ad approssimare correttamente la funzione e ciò è anche visibile da una riduzione in ampiezza massima più o meno evidente su tutti e tre i metodi.

Moltiplicando i passi ulteriormente per un fattore 10 ricaviamo che dt= $4*10^{-2}$ e dx=$50*10^{-2}$ ed otteniamo delle funzioni drasticamente diverse dalla funzione esatta al medesimo passo di discretizzazione.



Se invece di moltiplicare per un fattore 10 volessimo dividere i passi di discretizzazione per il medesimo fattore arrivando ad avere dt=4*10^{-5} e dx=50*10^{-5}, otteniamo seguenti grafici.



Tuttavia questo incremento in precisione non è minimamente apprezzabile se non guardando da vicino una zona ridotta della funzione.

8 Codice MATLAB per il plotting dei grafici

```
%% plotting dei risultati: soluzione esatta e approssimazioni

% figura 1: rappresentazione della funzione originale e delle sue
% approssimazioni coi vari metodi lungo tutto l'asse x in un determinato
% istante t
figure(1)
hold on
  plot(xx, uex_t(xx,t), 'r');
  plot(xx, uu_lf(:,round(t/dt)),  'b--');
  plot(xx, uu_lw(:,round(t/dt)), 'm--');
  plot(xx, uu_up(:,round(t/dt)), 'g--');
  legend('esatta','lax freidrich','lax wendroff','upwind')
  title('andamento della funzione rispetto ad x in un istante specifio t')
 hold off;

 %figura 2: rappresentazione degli errori assoluti nel tempo tra la funzione
 %originale e i vari metodi di approssimazione ad un determinato punto x_x fissato

 figure(2)
  hold on
  plot(tt, err_lf_temp,  'b--');
  plot(tt, err_lw_temp, 'm--');
  plot(tt, err_up_temp, 'g--');
  ylim([0 1])
  title('andamento errore in un istante t rispetto ad x')
  hold off
%figura 3: rappresentazione a confronto dei valori delle approssimazioni, dei loro ...
    errori assoluti
% nel tempo in un determinato punto x_x fissato

  figure(3)
  subplot(3,1,1);
  hold on;
  plot(tt, uex_t(xx(x_x), tt), 'k');
  plot(tt, uu_lf(x_x,:),  'b--');
  plot(tt, uu_lw(x_x,:), 'm--');
  plot(tt, uu_up(x_x,:), 'g--');
  legend('esatta','lax freidrich','lax wendroff','upwind')
  title('andamento della funzione rispetto ad x in un istante specifio t')
  hold off;

  subplot(3,1,2);
  hold on
  plot(tt, err_lf_temp,  'b--');
  plot(tt, err_lw_temp, 'm--');
  plot(tt, err_up_temp, 'g--');
  legend('LF','LW','UP')
  title('errore rispetto ad x in un istante t')
  ylim([0 0.5])


 %figura 4: rappresentazione degli errori assoluti nello spazio tra la funzione
 %originale e i vari metodi di approssimazione ad un determinato istante t fissato
  figure(4)
  hold on
  plot(xx, err_lf_sp,  'b--');
  plot(xx, err_lw_sp, 'm--');
  plot(xx, err_up_sp, 'g--');
  ylim([0 0.5])
  hold off

%figura 5: rappresentazione a confronto dei valori delle approssimazioni, dei loro ...
    errori assoluti
% nello spazio in un determinato istante t fissato
```

```
63
64    figure(5)
65    subplot(3,1,1);
66    hold on;
67    plot(xx, uex_t(xx,t), 'r');
68    plot(xx, uu_lf(:,round(t/dt)),  'b--');
69    plot(xx, uu_lw(:,round(t/dt)), 'm--');
70    plot(xx, uu_up(:,round(t/dt)), 'g--');
71    legend('esatta','lax freidrich','lax wendroff','upwind')
72    hold off;
73
74    subplot(3,1,2);
75    hold on
76    plot(xx, err_lf_sp,  'b--');
77    plot(xx, err_lw_sp, 'm--');
78    plot(xx, err_up_sp, 'g--');
79    legend('LF','LW','UP')
80    title('errore')
81    ylim([0 0.5])
82
83    %% --------- grafici 3d -----------
84
85    %generazione della mesh per il plotting dei grafici 3d
86    [X,Y] = meshgrid(xx(1:end),tt);
87
88    %figura 6: rappresentazione 3d dell'approssimazione di Lax Wendroff
89    figure(6)
90    mesh(X,Y,(uu_lw(1:end,:))')
91    %axis([0 1 0 1 0 50])
92    title('app. di Lax Wendroff con mesh')
93    xlabel('asse x');
94    ylabel('asse dei tempi t');
95    zlabel('intensit  u');
96
97    %figura 7: rappresentazione 3d dell'approssimazione di Lax Friedrichs
98    figure(7)
99    mesh(X,Y,((uu_lf(1:end,:)))')
100   title('app. di Lax friedrichs con mesh')
101    xlabel('asse x');
102   ylabel('asse dei tempi t');
103   zlabel('intensit  u');
104
105   %figura 8: rappresentazione 3d dell'approssimazione Con metodo di upwind
106   figure(8)
107   mesh(X,Y,(uu_up(1:end,:))')
108   title('app. Up-wind con mesh')
109   xlabel('asse x');
110   ylabel('asse dei tempi t');
111   zlabel('intensit  u');
112
113   %figura 9: rappresentazione 3d della funzione originale
114   figure(9);
115   mesh(X,Y,uex_t(X,Y))
116   xticks([0 0.2 0.4 0.6 0.8 1])
117   xticklabels({'1', '0.8' , '0.6',' 0.4', '0.2', ' 0'})
118   title('funzione originale')
119   ylabel('asse x');
120   xlabel('asse dei tempi t');
121   zlabel('intensit  u');
122
123   %figura 10: rappresentazione 3d della forzante calcolata analiticamente
124   figure(10);
125   mesh(X,Y,uu_f(1:end,:))
126   title('forzante con mesh')
127   xlabel('asse x');
128   ylabel('asse dei tempi t');
129   zlabel('intensit  u');
```

9 Conclusioni

L'elaborato mette in luce come una equazione differenziale alle derivate iperboliche non omogenea non differisce in modo drastico dal caso omogeneo. I metodi di calcolo alle differenze finite espliciti che sono basati sulla forma omogenea sono perfettamente validi, ma è necessaria l'aggiunta punto per punto del valore della forzante che gioca un ruolo fondamentale nella elaborazione di una corretta approssimazione numerica. La nostra inesperienza nel utilizzo degli strumenti di calcolo ci ha portato ad avere un pesante sfasamento tra la funzione esatta quella approssimata. È tuttavia osservabile come lo sfasamento sia costante e quindi compensabile mediante opportuni accorgimenti.

10 Bibliografia

References

[1] Quarteroni, Saleri, and Gervasio. *Calcolo Scientifico: Esercizi e problemi risolti con MATLAB e Octave*. UNITEXT. Springer Milan, 2012.

[2] Justin Hudson *Numerical Techniques for Conservation Laws with Source Terms*

[3] Tobias Oetiker, Hubert Partl, Irene Hyna, Elisabeth Schlegl *The Not So Short Introduction to LATEX 2 ε*. Version 6.4, March 09, 2021