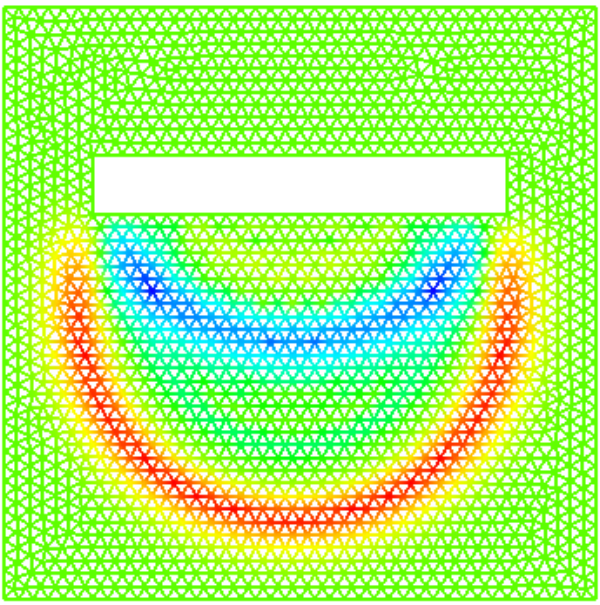
PDE Laboratory @ CRS4

Risoluzione numerica dell'equazione scalare delle onde mediante elementi finiti P1

Stefano Angioni s.angioni@infinito.it
Simone Locci smnlocci@email.it
Alessandro Pani supernandoz@libero.it



Center for Advanced Studies, Research and Development in Sardinia, F. Maggio
Dipartimento di Matematica e Informatica Università degli Studi di Cagliari, G. Rodriguez

4 luglio 2005

Abstract

This paper concerns the numerical resolution of the scalar wave equation using P1 finite elements. In the first section the numerical solution, for a given set of data, is compared with a known analytical solution. It is shown that the error depends on the dimension of the finite element grid and on the time step. Then, the same problem is solved again with an impulsive source.

L'oggetto di questo lavoro e' la risoluzione numerica dell'equazione scalare delle onde utilizzando elementi finiti di tipo P1. Nella prima parte del lavoro si verifica quanto la soluzione numerica per i dati assegnati si discosta da una soluzione analitica nota. Viene messo in evidenza come l'errore dipenda dalla dimensione della griglia degli elementi finiti e dall'intervallo di discretizzazione temporale. Nella seconda parte si affronta il problema in presenza di una sorgente impulsiva.

1 Introduzione

L'oggetto di questo lavoro è la risoluzione numerica dell'equazione scalare delle onde utilizzando elementi finiti di tipo P1. Il problema di partenza è il seguente:

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \Delta u + f \quad \text{in} \quad \Omega \times (0,T) \tag{1}$$

$$u = \phi \quad \text{su} \quad \Gamma_D \times (0,T) \tag{2}$$

$$\frac{\partial u}{\partial n} = \psi \quad \text{su} \quad \Gamma_N \times (0,T) \tag{3}$$

$$u\big|_{t=0} = u_0 \quad \text{in} \quad \Omega \tag{4}$$

$$\left.\frac{\partial u}{\partial t}\right|_{t=0} = u_1 \quad \text{in} \quad \Omega \tag{5}$$

per ogni $t \in (0,T)$ determinare $u(t) \in H^1(\Omega)$ tale che $u(t) = \phi(t)$ su Γ_D e dove c è la velocità di propagazione dell'onda, Ω è il dominio di integrazione, Γ_D è la frontiera di Dirichlet e Γ_N è la frontiera di Neumann. La formulazione variazionale è:

$$\int_\Omega \frac{1}{c^2}\ddot{u}v \, d\Omega + \int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f v \, d\Omega + \int_{\Gamma_N} \psi v \, d\Gamma \quad , \quad \forall v \in H_0^1(\Omega) \tag{6}$$

Per la risoluzione numerica della eq. (6) dobbiamo introdurre una discretizzazione del dominio Ω che consiste in una griglia di N_e triangoli non strutturata $\mathcal{F}_h$. In tal modo si può approssimare

numericamente ogni integrale della eq. (6) con la somma degli integrali valutati sugli N_e triangoli. A questo punto è utile definire gli spazi funzionali di dimensione finita usati per la discretizzazione ad elementi finiti. Per gli elementi triangolari essi sono:

$$X_h = \left\{ v \in C(\overline{\Omega}) \,:\, v|_{\Omega_e} \in P_1, \forall \Omega_e \in \mathcal{F}_h \right\}$$

$$X_{0h} = \{ v \in X_h \,:\, v|_{\Gamma_d} = 0 \}$$

dove P_1 é lo spazio dei polinomi lineari, si veda [1], paragrafo 7.3. Di conseguenza X_h è un sottospazio di $H^1(\Omega)$ e X_{0h} lo è di $H^1_0(\Omega)$. Indichiamo con $N_i(x, y)$ la i-esima funzione di X_{0h} che vale 1 sul grid-point i-esimo e si annulla su tutti gli altri grid-points. L'insieme delle $N_i(x, y)$ così definite, $i = 1, \ldots, N_n$ (numero dei nodi), costituisce una base per X_{0h}. La discretizzazione conduce al seguente sistema differenziale:

$$M\ddot{U} + KU = F \tag{7}$$

La matrice M è detta matrice di massa ed i suoi elementi sono pari a:

$$M_{ij} = \int_\Omega N_i N_j \, d\Omega \tag{8}$$

La matrice K è detta matrice di *stiffness* ed è definita nel seguente modo:

$$K_{ij} = \int_\Omega \nabla N_i \cdot \nabla N_j \, d\Omega \tag{9}$$

Vista l'importanza di tale matrice per la risoluzione del problema è riportata, in dettaglio, la sua costruzione. Considerazioni analoghe possono essere fatte per la matrice di massa.

1.1 Costruzione della matrice di stiffness

L'integrale dell'eq. (9) viene approssimato come somma degli integrali valutati sugli N_e triangoli:

$$\int_\Omega \nabla N_i \cdot \nabla N_j \, d\Omega = \sum_{e=1}^{N_e} \int_{\Omega_e} \nabla N_i \cdot \nabla N_j \, d\Omega = \sum_{e=1}^{N_e} \int_{\widehat{\Omega}} |J_e| \widehat{\nabla N_i} \cdot \widehat{\nabla N_j} \, d\widehat{\Omega} \tag{10}$$

dove $\widehat{\Omega}$ è un triangolo di riferimento, ad esempio il triangolo rettangolo di vertici $P_1 = (0, 0)$, $P_2 = (1, 0)$ e $P_3 = (0, 1)$. Il triangolo di riferimento viene mappato sul triangolo Ω_e tramite la trasformazione:

$$\begin{cases} x = & \alpha_1 \widehat{x} + \beta_1 \widehat{y} + \gamma_1 \\ y = & \alpha_2 \widehat{x} + \beta_2 \widehat{y} + \gamma_2 \end{cases} \tag{11}$$

caratterizzata dallo jacobiano $J_e = \alpha_1 \beta_2 - \alpha_2 \beta_1$. L'inverso della precedente trasformazione è:

$$\begin{aligned} \widehat{x} &= + \left(\beta_2 x - \beta_1 y + \beta_1 \gamma_2 - \beta_2 \gamma_1 \right) / J_e \\ \widehat{y} &= - \left(\alpha_2 x - \alpha_1 y + \alpha_1 \gamma_2 - \alpha_2 \gamma_1 \right) / J_e \end{aligned}$$

I parametri α, β e γ della trasformazione per il triangolo e-esimo, sono calcolati nel eguente modo:

$$\begin{cases} \alpha_1 = x_B - x_A \\ \beta_1 = x_C - x_A \\ \gamma_1 = x_A \end{cases} \qquad \begin{cases} \alpha_2 = y_B - y_A \\ \beta_2 = y_C - y_A \\ \gamma_2 = y_A \end{cases}$$

avendo denotato con A, B, C i tre vertici del triangolo e avendo supposto di mettere in relazione P_1 con A, P_2 con B e P_3 con C. I gradienti nelle variabili $(\widehat{x}, \widehat{y})$ si calcolano mediante la *chain rule*:

$$\begin{aligned} \frac{\partial}{\partial x} &= \frac{\partial}{\partial \widehat{x}}\frac{\partial \widehat{x}}{\partial x} + \frac{\partial}{\partial \widehat{y}}\frac{\partial \widehat{y}}{\partial x} \\ \frac{\partial}{\partial y} &= \frac{\partial}{\partial \widehat{x}}\frac{\partial \widehat{x}}{\partial y} + \frac{\partial}{\partial \widehat{y}}\frac{\partial \widehat{y}}{\partial y} \end{aligned}$$

Le funzioni $\widehat{N}$ nel triangolo di riferimento $\widehat{\Omega}$ valgono:

$$\widehat{N}(\widehat{x}, \widehat{y}) = \begin{cases} 1 - \widehat{x} - \widehat{y} & per\, la\, funzione\, centrata\, in\, (0,0) \\ \widehat{x} & per\, la\, funzione\, centrata\, in\, (1,0) \\ \widehat{y} & per\, la\, funzione\, centrata\, in\, (0,1) \end{cases}$$

Di conseguenza il gradiente di una generica funzione $\widehat{N}$ sarà:

$$\begin{aligned} \widehat{\nabla N} &= \frac{\partial \widehat{N}}{\partial x}\mathbf{i}_x + \frac{\partial \widehat{N}}{\partial y}\mathbf{i}_y = \\ &= \left(\frac{\partial \widehat{N}}{\partial \widehat{x}} \cdot \frac{\partial \widehat{x}}{\partial x} + \frac{\partial \widehat{N}}{\partial \widehat{y}} \cdot \frac{\partial \widehat{y}}{\partial x}\right)\mathbf{i}_x + \left(\frac{\partial \widehat{N}}{\partial \widehat{y}} \cdot \frac{\partial \widehat{y}}{\partial y} + \frac{\partial \widehat{N}}{\partial \widehat{x}} \cdot \frac{\partial \widehat{x}}{\partial y}\right)\mathbf{i}_y = \\ &= \frac{1}{|J_e|}\left[\left(\frac{\partial \widehat{N}}{\partial \widehat{x}} \cdot \beta_2 - \frac{\partial \widehat{N}}{\partial \widehat{y}} \cdot \alpha_2\right)\mathbf{i}_x + \left(\frac{\partial \widehat{N}}{\partial \widehat{y}} \cdot \alpha_1 - \frac{\partial \widehat{N}}{\partial \widehat{x}} \cdot \beta_1\right)\mathbf{i}_y\right] \end{aligned}$$

Vista la forma delle funzioni $\widehat{N}$ i gradienti sono delle costanti, per cui l'integrale in eq. (10) diventa:

$$\sum_{e=1}^{N_e} |J_e| \left(\widehat{\nabla N}_i \cdot \widehat{\nabla N}_j\right)_e \int_{\widehat{\Omega}_e} d\widehat{\Omega} = \sum_{e=1}^{N_e} \frac{|J_e|}{2} \left(\widehat{\nabla N}_i \cdot \widehat{\nabla N}_j\right)_e = K_{ij}$$

La sommatoria ha solo i contributi dei triangoli i cui vertici coincidono con i nodi i e j.

1.2 Discretizzazione temporale

Si introduce la seguente discretizzazione temporale, basata sulle differenze finite:

$$t_j = jk,\ j = 0, 1, \ldots, m, \quad k = \frac{T}{m}$$

La derivata seconda temporale è stata approssimata nel seguente modo:

$$\ddot{U}^n = \frac{U^{n+1} - 2U^n + U^{n-1}}{k^2} + \mathcal{O}\left(k^2\right)$$

Il sistema (7) diviene:

$$MU^{n+1} = c^2k^2\left(F^n + KU^n\right) + H^n \ , \quad n = 0, \ldots, m-1$$

con

$$H = M\left(2U^n - U^{n-1}\right) \tag{12}$$

La matrice M, a seconda della formula di quadratura usata, è diagonale con elementi strettamente positivi, o comunque può essere diagonalizzata. Nel nostro caso la matrice ha già forma diagonale. Il sistema (12) diviene:

$$U^{n+1} = M^{-1}\left[c^2k^2\left(F^n + KU^n\right) + H^n\right] \ , \quad n = 1, \ldots, m-1$$

che non richiede la risoluzione di un sistema lineare. Poiché il metodo richiede, ad ogni passo, la conoscenza della U nei due step precedenti, assegneremo ai primi due valori della U la soluzione analitica calcolata agli istanti 0 e dt.

2 Problema 1

Lo scopo è verificare quanto la soluzione numerica per i dati assegnati si discosta dalla soluzione analitica, che è la seguente:

$$u(x, y, t) = \sin(3\pi a(x - x_0))\sin(2\pi b(y - y_0))\sin(\omega t)$$

nel dominio rettangolare $\Omega = (x_0, x_1)\times(y_0, y_1)$, con $a = 1/(x_1 - x_0)$, $b = 1/(y_1 - y_0)$, $\omega = \pi/(2T)$, $x_0 = y_0 = 0$, $x_1 = y_1 = 1$, $T = 0.5s$. In particolare, si vuol mettere in evidenza come l'errore dipenda dalla dimensione degli elementi finiti e dall'intervallo di discretizzazione temporale.

2.1 Sviluppo dell'algoritmo

Il programma a cui fare riferimento per il primo problema è `wave_analytic`. La prima parte del listato comprende la lettura della griglia realizzata con il programma *Easymesh* [1] e l'allocazione di tutti i parametri associati. Nella notazione degli elementi finiti, il tipico formato di una griglia di triangoli è riportato in tabella 1. In generale, il dominio bidimensionale Ω è considerato immerso in uno spazio 3D, e quindi le coordinate dei nodi della griglia (*grid-points*) hanno tre componenti. E' stato supposto, nel problema considerato, $z_i = 0$, $i = 1 \ldots n_{nod}$. Le tre *connettività* del generico elemento rappresentano gli indici dei nodi che giacciono sui vertici dell'elemento stesso. Ovviamente il numero delle connettività dipende dalla tipologia dell'elemento: per i nostri triangoli le connettività sono tre. La *proprietà* permette di associare al generico elemento una o più grandezze fisiche, ipotizzate costanti nell'elemento stesso: ad

n_{nod}				numero di nodi
n_{el}				numero di elementi
1	x_1	y_1	z_1	coordinate del grid point 1
2	x_2	y_2	z_2	coordinate del grid point 2
...	...	...	...	...
n_{nod}	$x_{n_{nod}}$	$y_{n_{nod}}$	$z_{n_{nod}}$	coordinate del grid point n_{nod}
$i_1^{(1)}$	$i_1^{(2)}$	$i_1^{(3)}$	P_1	Connettività e proprietà dell'elemento 1
$i_2^{(1)}$	$i_2^{(2)}$	$i_2^{(3)}$	P_2	Connettività e proprietà dell'elemento 2
...	...	...	...	...
$i_{n_{el}}^{(1)}$	$i_{n_{el}}^{(2)}$	$i_{n_{el}}^{(3)}$	$P_{n_{el}}$	Connettività e proprietà dell'elemento n_{nod}
n_{Dir}				numero di nodi di Dirichlet
d_1	d_2	...	$d_{n_{Dir}}$	Nodi di Dirichlet

Tabella 1: Formato della griglia

esempio, le caratteristiche meccaniche del materiale di cui è costituita la regione descritta dal dominio Ω.

Nella seconda parte vengono calcolati i coefficienti della trasformazione lineare degli elementi finiti nello spazio di riferimento di eq. (11) e la matrice di massa M.

```
Build M
!------------------------------------------------------------------

      allocate (mm(nnod))

      do i = 1,nnod
        mm(i) = 0
      enddo

      do ie = 1,nelem
        do i = 1,3
          irig = con(ie,i)
          mm(irig) = mm(irig) + jac(ie)/6
        enddo
      enddo
```

Poiché la matrice M risulta diagonale, allochiamo solo un vettore `mm` di numeri reali, di dimensione pari al numero di nodi, che viene inizializzato a zero. Considerando la eq. (8), l'integrale

viene risolto numericamente mediante formula di quadratura la quale, per ogni elemento, valuta il prodotto delle funzioni N_i ed N_j nei tre vertici ed esegue la somma. Lo jacobiano è dovuto al cambio del sistema di riferimento.

Successivamente vengono richiesti all'utente il tempo finale della simulazione e l'intervallo Δt (dt) e valutate le due condizioni iniziali necessarie all'algoritmo iterativo, a partire dalla soluzione analitica fornita.

```
!     Build uu0, uu1 condizioni iniziali
!--------------------------------------------------------------------

      allocate(uu0(nnod), uu(nnod))

      do ii= 1,nnod
         uu0(ii) = sol_an(xx(ii), yy(ii), zero)
         uu (ii) = sol_an(xx(ii), yy(ii), dt)
      enddo
```

Vengono allocati i vettori per il termine f (`rhs`), la matrice H (`hh`) ed il prodotto matrice vettore KU^n. Quest'ultimo viene effettuato senza la memorizzazione della matrice: tale metodo, implementato dalla subroutine `mat_vec_free`, viene chiamato approccio *matrix-free* ed è praticamente obbligatorio per il trattamento di problemi di grande scala, le cui matrici sono talmente grandi da non poter essere immagazzinate in memoria, neanche usando la loro sparsità. Infine viene inizializzato il loop temporale che permette di valutare tali matrici e la soluzione U nei vari istanti di tempo.

```
!--------------------------------------------------------------------

      allocate (hh(nnod))
      allocate (rhs(nnod))
      allocate (KUn(nnod))

      t = dt
      k2 = dt*dt

      write(*,*) 'k2,t = ',k2,t

      do it = 2,m
```

```
!     Build F = RHS:
!---------------------------------------------------------------------
      rhs = 0
      do ie=1,nelem
      cost=jac(ie)/6.d0
        do i=1,3
          irig=con(ie,i)
          term=cost*effe_an(xx(irig),yy(irig),t)
          rhs(irig)=rhs(irig)+term
        enddo
      enddo

      t = t + dt

!     Build H = M(2 Un - Un-1)
!---------------------------------------------------------------------

      do i = 1,nnod
         hh(i) = (2*uu(i) - uu0(i))
      enddo

!---------------------------------------------------------------------
!     Build K*Un
!---------------------------------------------------------------------

      call mat_vec_free(nnod,nelem,nbar,alfa1,beta1,gamma1,alfa2,beta2,%
      gamma2,con,con_edge,uu,KUn)
```

Il minimo numero di vettori necessari per allocare le iterate della soluzione U è pari a 2: essi sono stati indicati con `uu` e `uu0` ed aggiornati come segue:

```
!     sostituzione di uu0 con uu
!---------------------------------------------------------------------
      do i = 1,nnod
        uu0(i) = uu(i)
      enddo
```

```
!---------------------------------------------------------------------

!     Calcolo uu(t) n+1
!---------------------------------------------------------------------

      do i = 1,nnod
        uu(i) = k2*(rhs(i) - KUn(i))/mm(i) + hh(i)
      enddo
```

Sui punti della frontiera di Dirichlet viene imposta la soluzione analitica del problema.

```
!     Correction for Dirichlet nodes
      do ie=1,nbar
        do i=1,2
          irig=con_edge(ie,i)
          uu(irig)  = sol_an(xx(irig),yy(irig),t)
          uu0(irig) = sol_an(xx(irig),yy(irig),t-dt)
        enddo
      enddo
```

La valutazione dell'errore in norma infinito è svolta dal codice seguente:

```
!     Error evaluation; numerical and exact solutions are written in 'sol.d'
!---------------------------------------------------------------------

      allocate (uu_an(nnod))
      do i=1,nnod
        uu_an(i)=sol_an(xx(i),yy(i),t_fin)
      enddo

      num=0.d0; den=0.d0
      do i=1,nnod
        diff=dabs(uu(i)-uu_an(i))
        if (diff.gt.num) then
          num=diff
```

```
          imax=i
        endif
        if (dabs(uu_an(i)).gt.den) den=dabs(uu_an(i))
      enddo

      write(*,*)
      write(*,*)'Num, den, Relative error, uu = ',num,den, num/den,uu(imax)
      write(*,*) 'corresponding to node ',imax

      open(23,file='sol.d')
      do i=1,nnod
        write(23,*)i,uu0(i),uu_an(i)
      enddo
      close(23)
```

L'ultima parte del codice, anch'essa standard, consente la creazione del file che permette la visualizzazione della soluzione numerica. A tale scopo è stato utilizzato il programma *MayaVI* [2] , che consente la rappresentazione delle griglie e dei campi d'onda valutati. Infine sono riportate le subroutine richiamate nel codice precedente. Esse sono:

- `sol_an`: è la soluzione analitica del problema;
- `effe_an`: è la funzione f di eq. 2. Per determinarla abbiamo risolto analiticamente la eq. 2;
- `mat_vec_free`, che effettua il prodotto matrice vettore senza allocare la matrice. Di quest'ultima riportiamo il codice.

```
      subroutine mat_vec_free(nnn,nelem,nbar,alfa1,beta1,gamma1, &
      alfa2,beta2,gamma2,con,con_edge,xxx,bbb)

!     matrix-free matrix-vector product
!     A*xxx = b

      implicit none

      integer(4) :: nnn,nelem,nbar,ie,i,j,l,irig,icol
      integer(4), dimension(nelem,3) :: con
      integer(4), dimension(nbar,2) :: con_edge
```

```
      real(8) :: jac, cost, term
      real(8), dimension(*) :: alfa1,beta1,gamma1,alfa2,beta2,gamma2
      real(8), dimension(*) :: xxx,bbb
      real(8), dimension(3) :: ennex, enney

      ennex(1)=-1.d0; ennex(2)=1.d0; ennex(3)=0.d0
      enney(1)=-1.d0; enney(2)=0.d0; enney(3)=1.d0

      do i=1,nnn
        bbb(i)=0.d0
      enddo

      do ie=1,nelem
        jac=dabs(alfa1(ie)*beta2(ie)-alfa2(ie)*beta1(ie))
        cost=0.5d0/jac
        do i=1,3
          irig=con(ie,i)
          do j=1,3
            icol=con(ie,j)
             term=(beta2(ie)*ennex(i)-alfa2(ie)*enney(i))* &
                  (beta2(ie)*ennex(j)-alfa2(ie)*enney(j))
             term=term+(alfa1(ie)*enney(i)-beta1(ie)*ennex(i))* &
                  (alfa1(ie)*enney(j)-beta1(ie)*ennex(j))
             bbb(irig)=bbb(irig)+cost*term*xxx(icol)
          enddo
        enddo
      enddo

!     Correction for Dirichlet nodes
!     (the corresponding matrix row is 0 everywhere except for the diagonal
!     which takes the value 1.)
      do ie=1,nbar
        do i=1,2
          irig=con_edge(ie,i)
         bbb(irig)=xxx(irig)
        enddo
      enddo
```

```
return

end
```

2.2 Risultati numerici

Dal momento che si vuole mettere in evidenza la dipendenza dell'errore dai parametri h e k, abbiamo eseguito due serie di test, variando solo un parametro alla volta. Nel primo test abbiamo fissato il parametro h, quindi la griglia, al valore $h = 0.05$ e, posto un tempo finale pari a $T_{fin} = 1.1s$, si è fatto variare Δt da $0.001s$ a $0.05s$. Il metodo utilizzato è solo condizionatamente stabile (cfr. [4])al variare di Δt. Il metodo è stabile solo se:

$$\Delta t < \gamma \cdot \min\left(\frac{h}{c}\right) = \gamma \cdot \Delta t_{CFL}, \quad 0 < \gamma < 1$$

dove γ è un fattore di sicurezza. Nel nostro caso $h = 0.05$, $c = 1$, $t_0 = 0.05s$ e $\beta = 100s^{-1/2}$, per cui $\Delta t_{CFL} = 0.05s$. In tabella 2 sono riportati i risultati delle diverse simulazioni, mentre in figura 2.2 e 1 sono rappresentati, rispettivamente, l'andamento dell'errore in scala logaritmica ed un ingrandimento nella zona di stabilità numerica. Si può notare come la soluzione numerica diverga per $\Delta t > 0.03s$, corrispondente a $\gamma = 0.6$.

Δt(s)	Errore rel.
0.001	0.11
0.002	0.12
0.003	0.13
0.008	0.17
0.01	0.19
0.02	0.27
0.03	0.31
0.035	$2 \cdot 10^3$
0.04	$4 \cdot 10^{11}$
0.05	$2 \cdot 10^{14}$

Tabella 2: Errore relativo al variare di Δt ($h = 0.05m$)

Nel secondo test abbiamo fissato il parametro $\Delta t = 10^{-3}s$ e, posto il tempo finale pari a $T_{fin} = 0.1s$, si è fatto variare h da $0.02m$ a $0.1m$. Con questa scelta di Δt l'errore associato alla discretizzazione temporale è trascurabile. In figura 3 è mostrato l'andamento dell'errore relativo al variare di h.

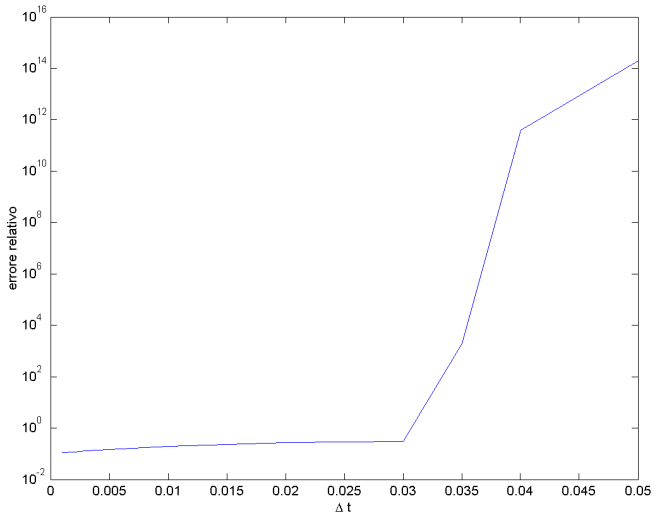


Figura 1: Andamento dell'errore al variare del parametro k. Si noti l'instabilità per $\Delta t > 0.03s$

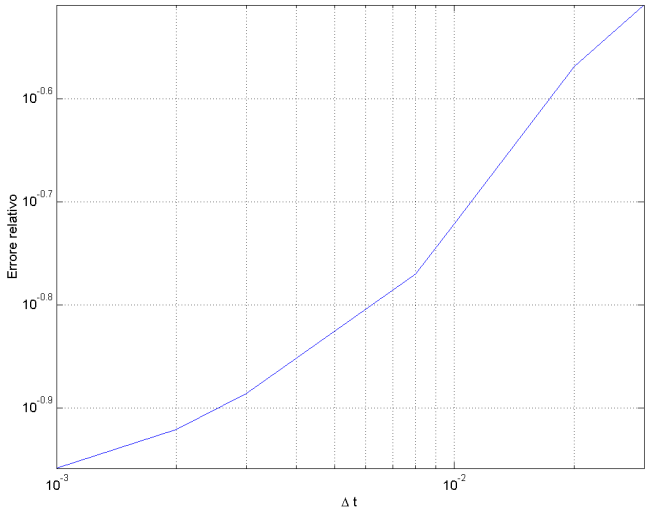


Figura 2: Andamento dell'errore al variare del parametro k (zoom)

3 Problema 2

In questo caso non viene fornita la soluzione analitica, ma viene data solo la sollecitazione f, non nulla solo nelle coordinate (0,0) e con il seguente andamento temporale:

$$h(t) = \left(1 - 2\beta(t - t_0)^2\right) \exp\left(-\beta(t - t_0)^2\right)$$

dove $t_0 = 0.05s$ e $\beta = 100s^{-1/2}$. La sua rappresentazione grafica è in figura 4. Il modulo della trasformata di Fourier, calcolata numericamente tramite *Matlab*, è riportata nel grafico di figura 5 ed ha un andamento a campana. Analiticamente, è possibile dimostrare che la trasformata

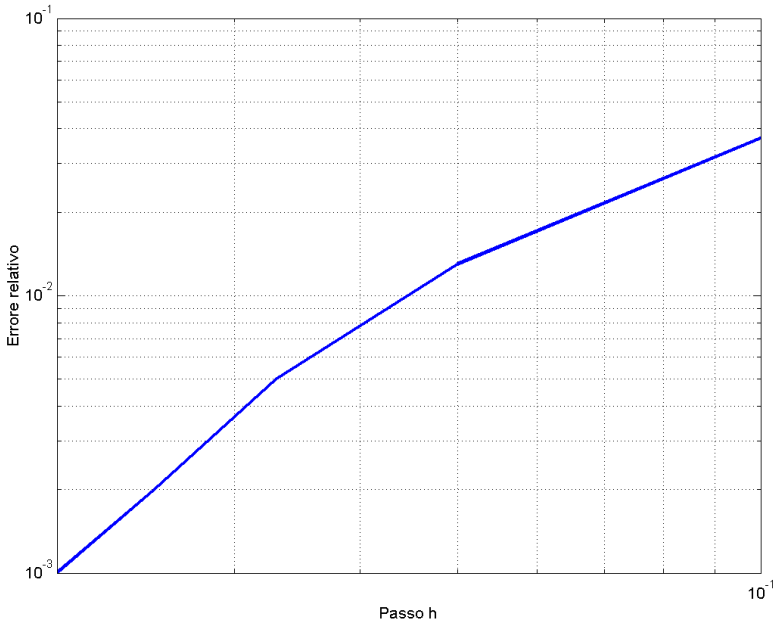


Figura 3: Andamento dell'errore relativo al variare di h

di Fourier di $h(t)$, è data da:

$$H(\omega) = \frac{\sqrt{\pi}}{2 \cdot \beta^{3/2}} \cdot \omega^2 \cdot \exp\left[-\frac{\omega}{(4 \cdot \beta)} \cdot \left(\frac{j}{t_0} + \omega\right)\right] \tag{13}$$

Il dominio computazionale è riportato in figura 6.

3.1 Sviluppo dell'algoritmo

Il programma a cui fare riferimento per il problema è `wave`. Vengono ora messe in evidenza solo le differenze rispetto al programma precedente. Dette x_s ed y_s le coordinate della sorgente, viene calcolato il nodo della griglia più vicino.

```
diff=1.d90
do i=1,nnod
  term=(xx(i)-xs)*(xx(i)-xs)+(yy(i)-ys)*(yy(i)-ys)
  if (term<diff) then
    diff=term
    is=i
  endif
enddo
```

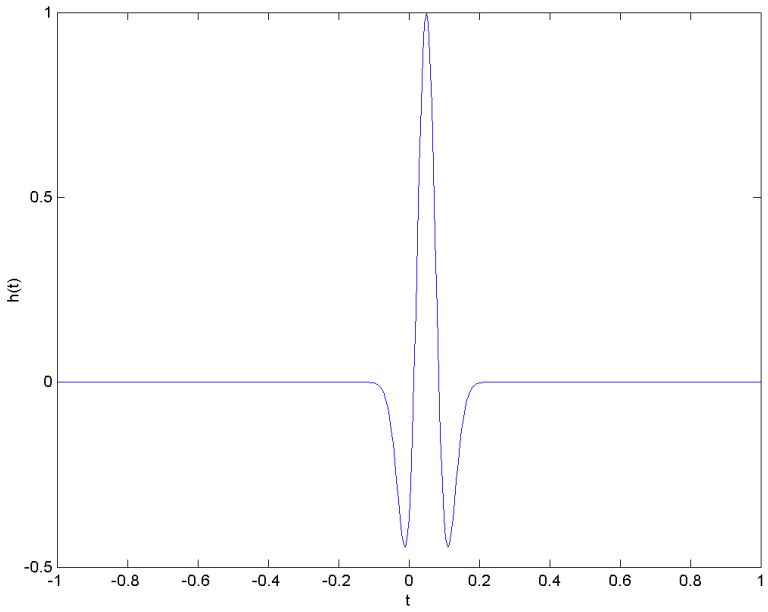


Figura 4: Grafico della funzione $h(t)$ (ondina di Ricker)

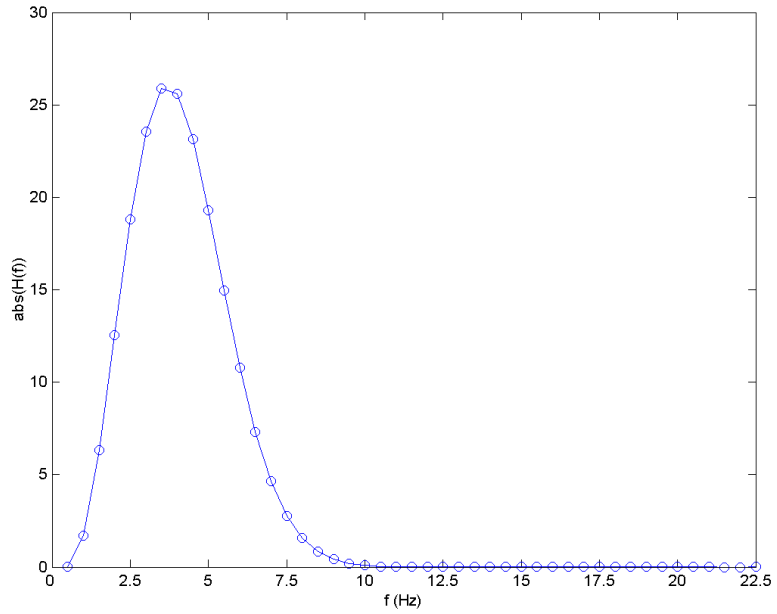


Figura 5: Grafico di $|H(f)|$

Le condizioni iniziali sono poste a zero:

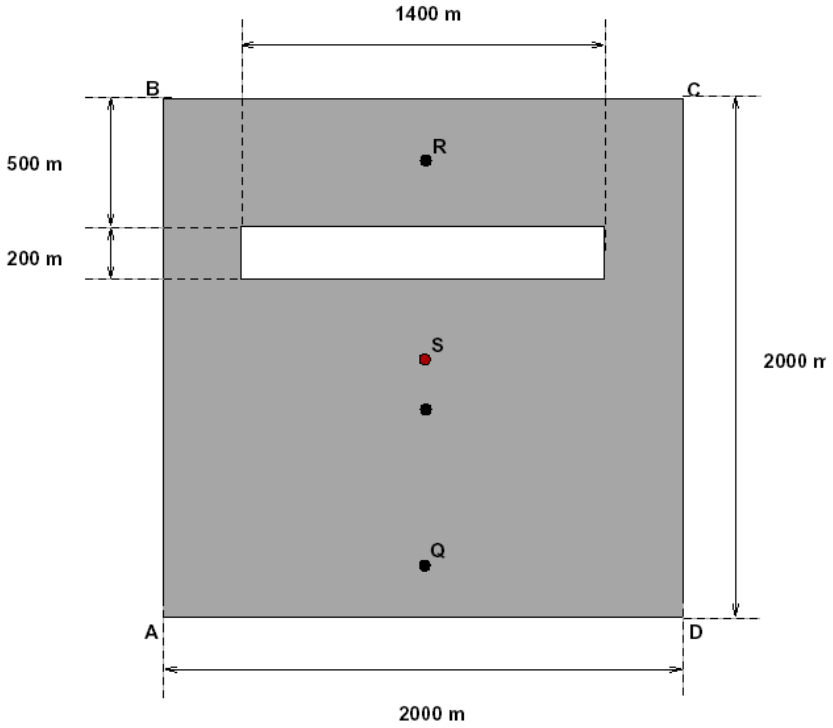


Figura 6: Dominio computazionale del problema 2

```
do ii= 1,nnod
    uu0(ii) = zero
    uu (ii) = zero
 enddo
```

La sollecitazione f (`rhs`) è posta pari alla funzione $h(t)$. Poiché la velocità di propagazione dell'onda non è più unitaria, ma pari a $c = 1800m/s$, il codice che determina la soluzione numerica è stato modificato:

```
!     Calcolo uu(t) n+1
!--------------------------------------------------------------------

      do i = 1,nnod
        uu(i) = k2*ccc2*(rhs(i) - KUn(i))/mm(i) + hh(i)
      enddo
```

La soluzione sulla frontiera è stata posta a zero perché le condizioni al contorno di Dirichlet sono identicamente nulle.

3.1.1 Sismogrammi

L'utilizzo dei sismogrammi permette di visualizzare l'evoluzione temporale della soluzione in punti definiti della griglia.

Per implementare i sismogrammi si fa utilizzo di un vettore nel quale, al termine di ogni iterazione, viene aggiunto il valore della soluzione nel punto determinato all'inizio dell'elaborazione. Si viene così a costruire un array che al suo interno è strutturato come una successione di valori $[u(x_P, y_P, t_0), u(x_P, y_P, t_1), \ldots, u(x_P, y_P, t_m)]$ che rappresenta l'evoluzione temporale della funzione $u(x, y, t)$ calcolata nel punto $P(x_P, y_P)$.

Poiché però in generale il punto scelto per il sismografo non corrisponderà ad un nodo della griglia di discretizzazione, è necessario scegliere un nodo a cui fare riferimento. Si può scegliere di utilizzare il nodo della griglia più vicino a quello indicato come sismografo; dal punto di vista pratico si calcola la distanza tra le coordinate desiderate e quelle di tutti i punti, si sceglie la distanza minore e si imposta quel particolare nodo come punto in cui registrare il sismogramma.

3.2 Risultati numerici

Le griglie generate con EasyMesh hanno passo h pari a $20m$, $50m$ e $100m$. Nella tabella 3 sono riportate le coordinate desiderate e quelle effettive dei sismografi. I grafici nelle figure 7, 8 e 9

Punto	Coord. des.	Coord. eff.
P	(0,-200)	(0,-219)
Q	(0,-800)	(0,-826)
R	(0,750)	(2,752)

Tabella 3: Coordinate dei sismografi

raffigurano i sismogrammi per i punti P, Q ed R. Nelle figure 10, 11 e 12 sono rappresentati gli snapshot dei campi d'onda a $T = 0.5s$ e $\beta = 100s^{-1/2}$, corrispondenti alle tre griglie create. La soluzione relativa alla griglia con $h = 20m$ è peggiore (cioè i fronti d'onda sono meno netti) rispetto alla griglia con $h = 50m$, mentre ci aspetteremmo il contrario. Questo è spiegabile se si visualizzano le diverse griglie (figure 13, 14 e 15). Solo le griglie relative ad $h = 100m$ e $h = 50m$ hanno buona qualità (cioè la dimensione degli elementi finiti è pressoché uniforme) invece la griglia relativa ad $h = 20m$ presenta una forte disomogeneità nella dimensione degli elementi finiti.

All'aumentare di β aumenta la frequenza corrispondente al massimo della trasformata di Fourier di $h(t)$, con conseguente riduzione della lunghezza d'onda. Tale aumento della frequenza di propagazione causa un peggioramento della soluzione numerica, in quanto si mostra (come vedremo successivamente) una maggiore presenza di rumore (quindi soluzioni con fronti d'onda

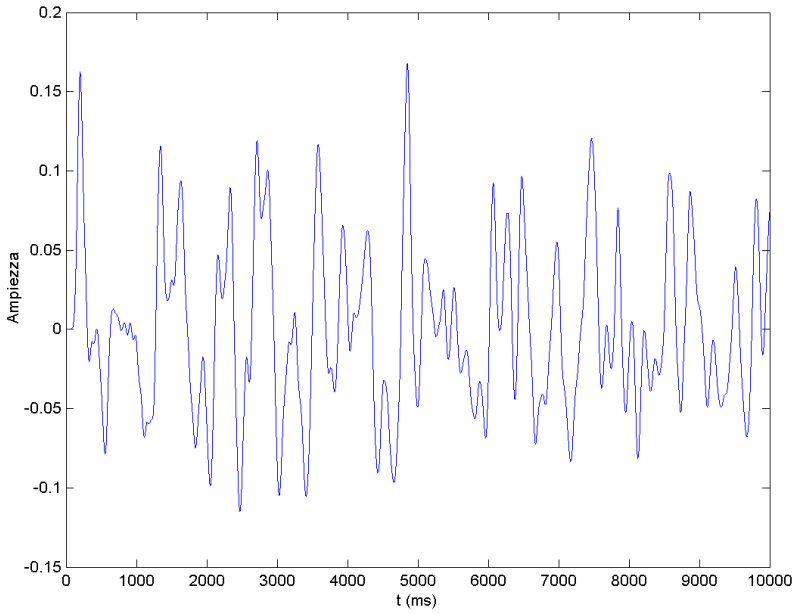


Figura 7: Sismogramma nel punto P

meno netti). Le figure 16, 17, 18, 19 e 20 mostrano gli snapshot, con tempo finale pari a $0.5s$ e griglia con $h = 50$, relativi a β rispettivamente pari a 100, 125, 150, 250 e 400 $s^{-1/2}$. Si mette in evidenza come già per $\beta = 100s^{-1/2}$ la soluzione presenti un po' di rumore, rappresentato dal debole fronte d'onda giallo. Inoltre si mette in evidenza come all'aumentare di β il rumore aumenti sempre di più, cioè sarà sempre maggiore la presenza di fronti d'onda che in realtà non esistono.

Per determinare λ è stato calcolato il massimo del modulo della funzione $H(\omega)$ di eq. (13), annullandone la derivata. Si ottiene $f = \sqrt{\beta}/\pi = 3.18Hz$. Questo vuole dire che $\lambda = c/f = 1800/3.18 = 566m$. Questo valore di λ è accettabile in quanto per la componente a $3.18Hz$ il periodo corrispondente è pari a $0.31s$ quindi in $0.45s$ (che è l'intervallo di tempo in cui si propaga l'onda) dovrebbe essere percorso un tratto pari a 1.45λ. Se si controllano gli snapshot si nota come il fronte dell'onda che si propaga ha percorso circa $850m$, cioè un numero di lunghezze d'onda pari a $850/\lambda = 850/566 = 1.5$.

Sia $G = \lambda/h$ il rapporto fra la lunghezza d'onda ed il passo della griglia. Dalla letteratura, si sa che per elementi finiti lineari ci si deve aspettare $G = 10$ per avere un segnale pulito. Dalle simulazioni effettuate è emerso che, per la griglia con $h = 50m$ (cioè quella con il miglior compromesso fra omogeneità degli elementi e dimensione del passo), il valore minimo del rapporto è pari a $G = 566/50 = 11.3$.

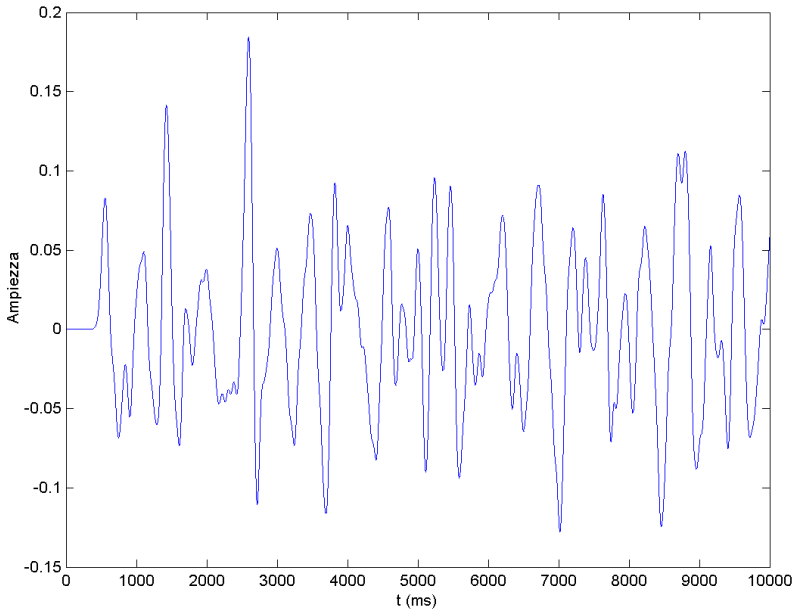


Figura 8: Sismogramma nel punto Q

Bibliografia

[1] Quarteroni A., R. Sacco, F. Saleri, *Matematica numerica*, 2a edizione, Springer-Verlag Italia, 2000

[2] EasyMesh Web Page,
http://www.dinma.univ.trieste.it/nirftc/research/easymesh/Default.htm

[3] The MayaVi Data visualizer Web Page
http://mayavi.sourceforge.net/

[4] Maggio F., 2005, *Laboratorio di metodi numerici per la risoluzione delle equazioni alle derivate parziali*

[5] Maggio F. and A. Quarteroni, 1994, *Acoustic wave simulation by spectral methods*, East-West J. Num. Math., 2, 2, 129-150

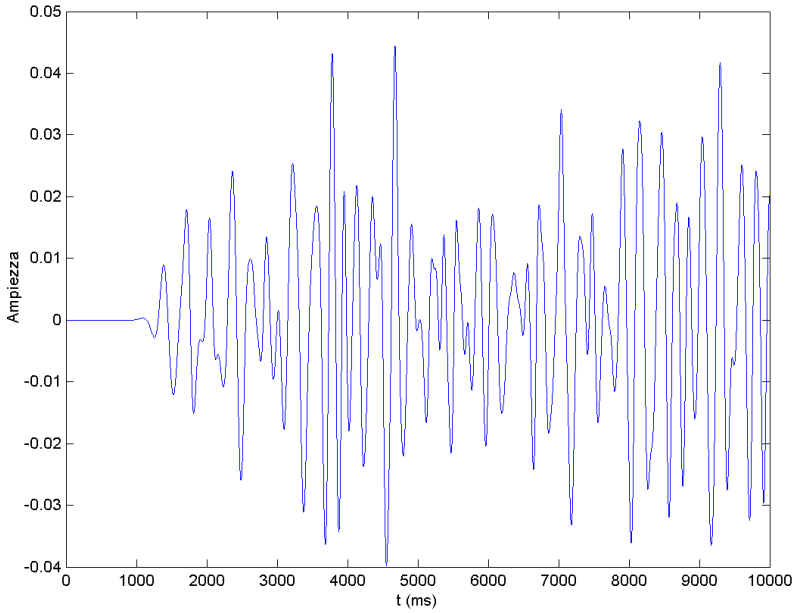


Figura 9: Sismogramma nel punto R

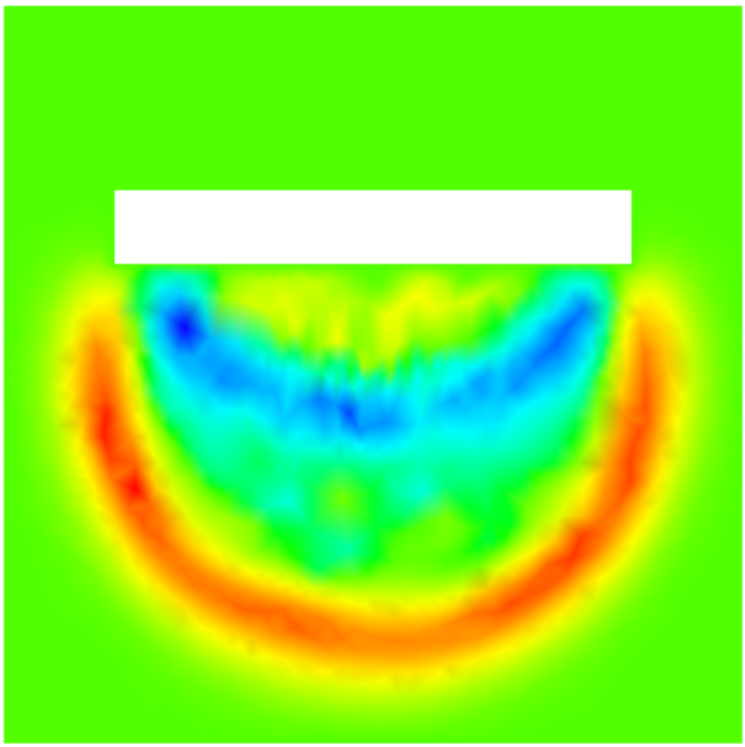


Figura 10: Campo d'onda per $h = 20m$

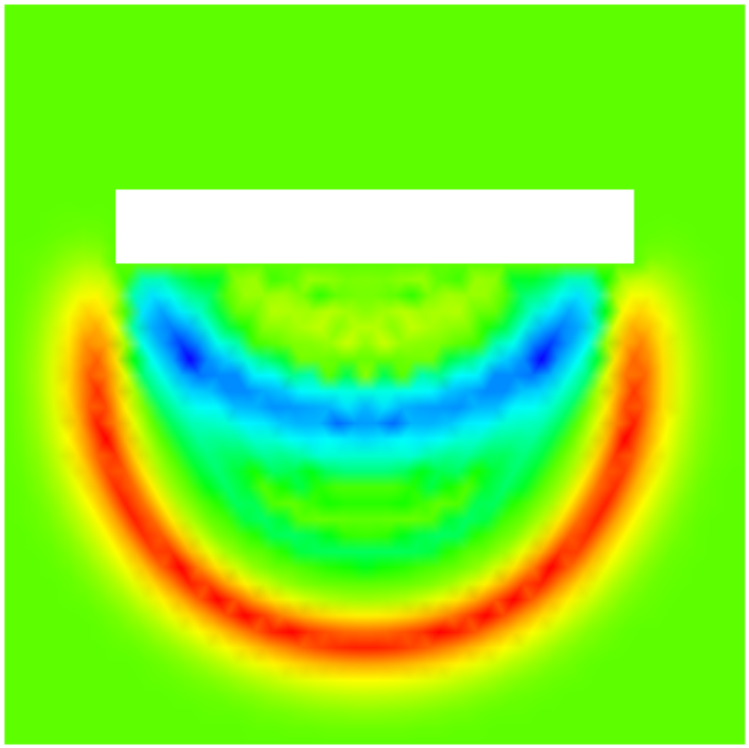


Figura 11: Campo d'onda per $h = 50m$

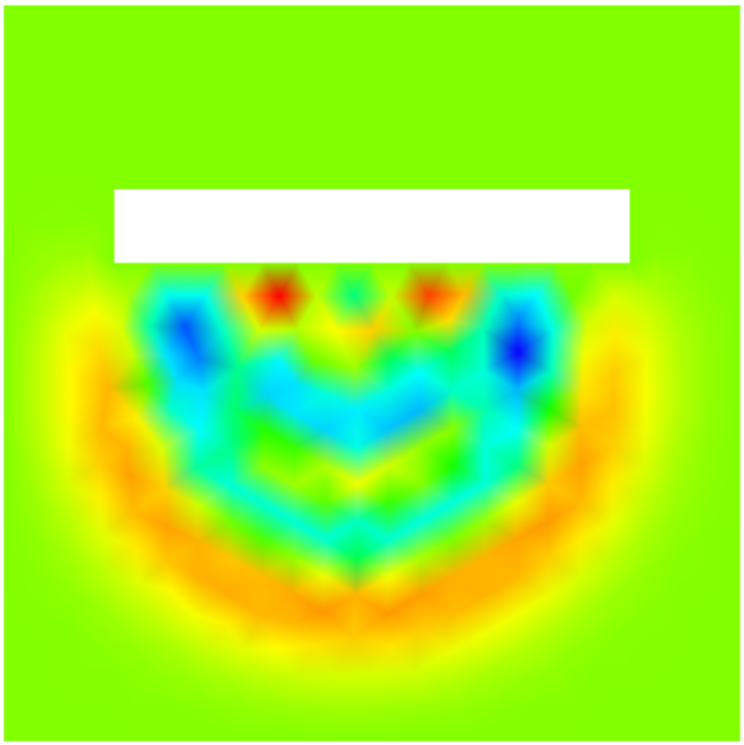


Figura 12: Campo d'onda per $h = 100m$

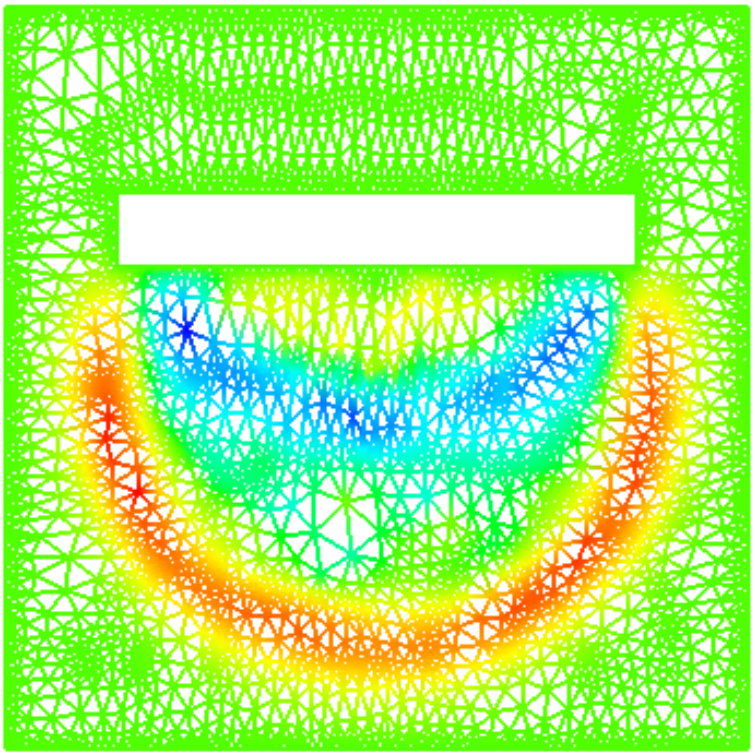


Figura 13: Griglia per $h = 20m$

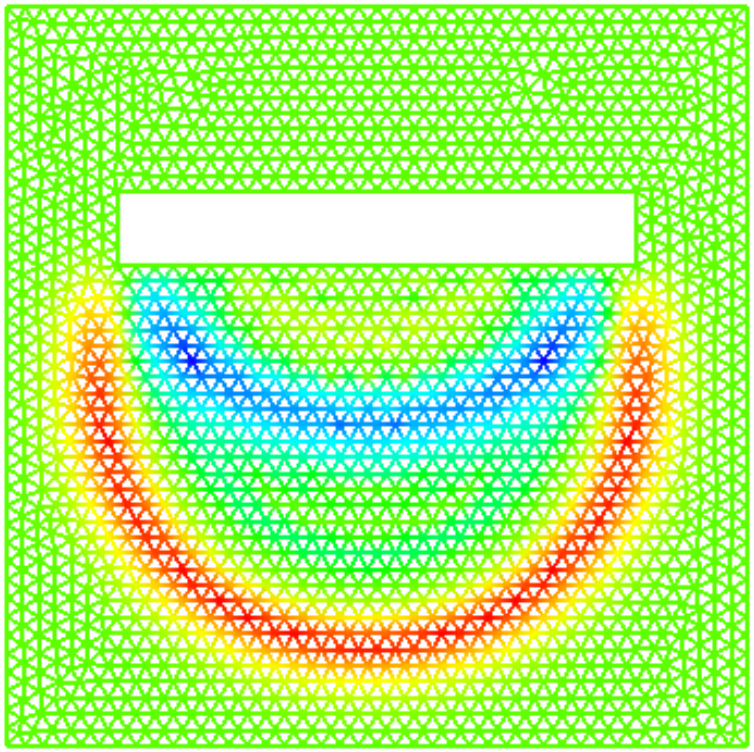


Figura 14: Griglia per $h = 50m$

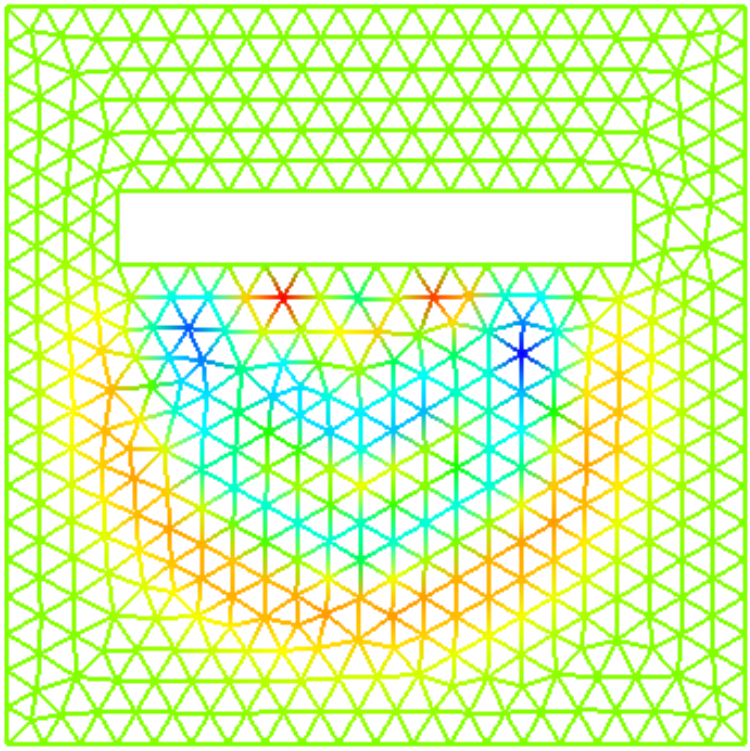


Figura 15: Griglia per $h = 100m$

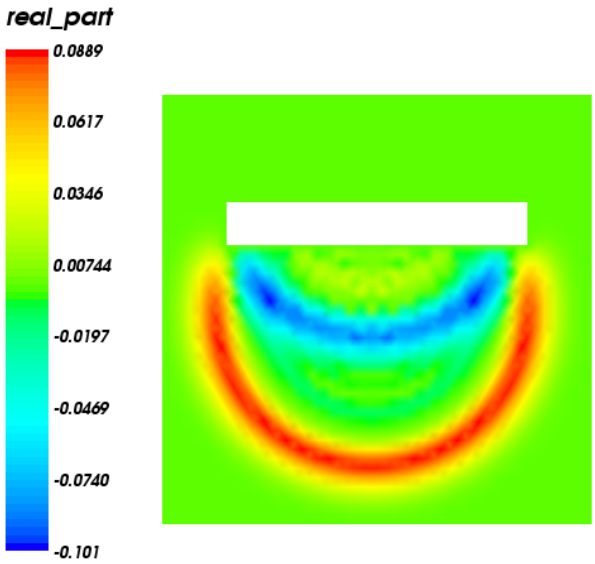


Figura 16: Campo d'onda per $\beta = 100s^{-1/2}$ (griglia corrispondente a $h = 50m$)

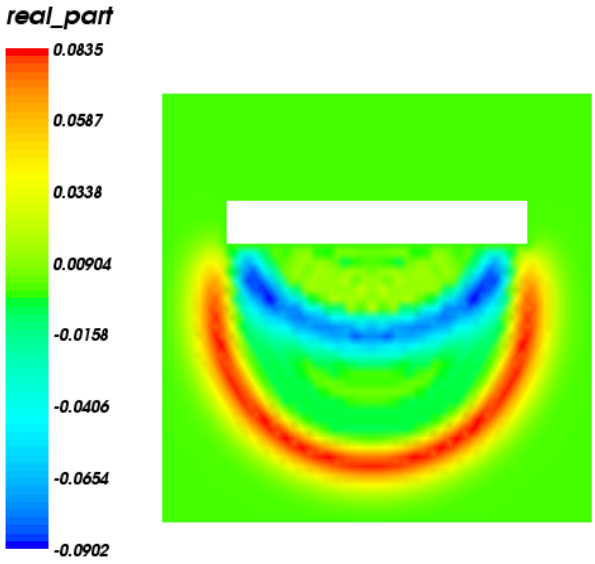


Figura 17: Campo d'onda per $\beta = 125 s^{-1/2}$ (griglia corrispondente a $h = 50m$)

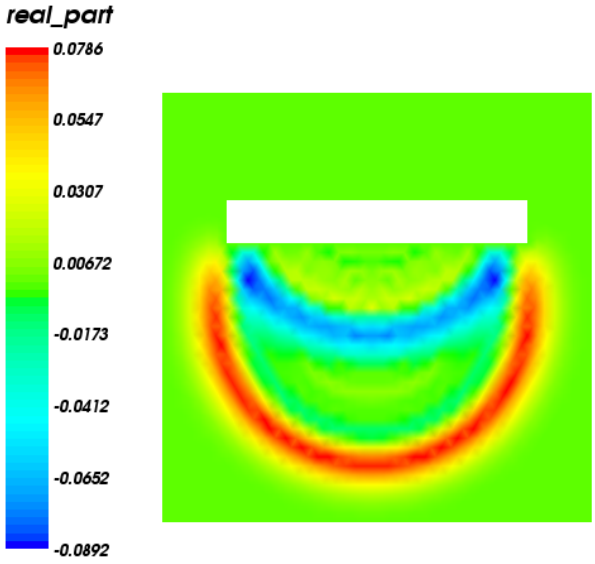


Figura 18: Campo d'onda per $\beta = 150 s^{-1/2}$ (griglia corrispondente a $h = 50m$)

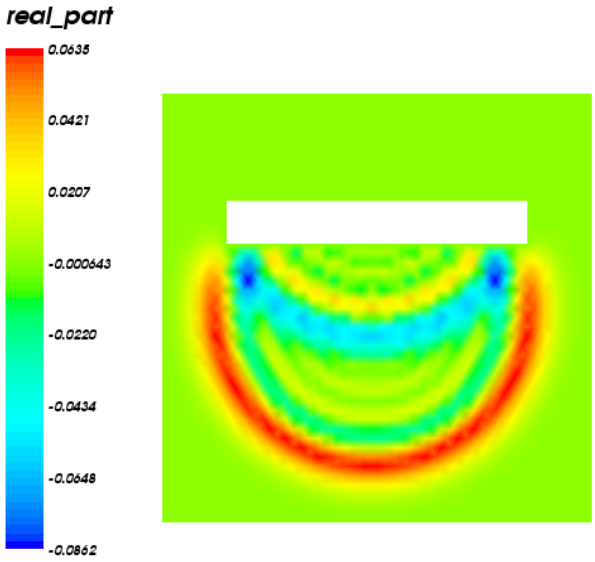


Figura 19: Campo d'onda per $\beta = 250 s^{-1/2}$ (griglia corrispondente a $h = 50m$)

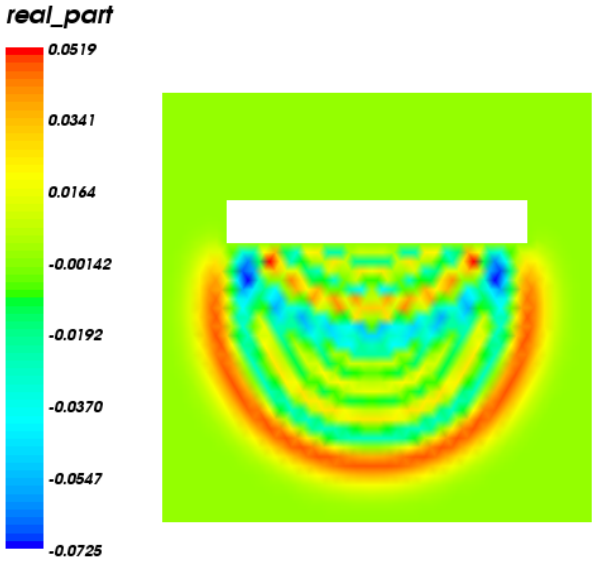


Figura 20: Campo d'onda per $\beta = 400 s^{-1/2}$ (griglia corrispondente a $h = 50m$)