Appunti del corso di esercitazioni di Calcolo Scientifico a.a. 2014-2015 "Problemi di Vibrazioni"

Dario A. Bini

9 novembre 2014

1 Introduzione

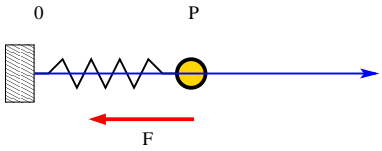
Lo studio delle vibrazioni di sistemi di particelle è un tipico problema in cui il ruolo giocato dagli autovalori e dagli autovettori è particolarmente rilevante. In questo contesto i metodi numerici per il calcolo di autovalori e autovettori hanno una importanza determinante.

Molte applicazioni del calcolo scientifico si riconducono allo studio di problemi di vibrazioni. Giusto per fare qualche esempio: minimizzare le vibrazioni delle rotaie nei treni ad alta velocità, minimizzare il rumore che si genera dentro l'abitacolo delle auto a causa delle vibrazioni del motore e degli effetti aerodinamici, la progettazione dei ponti in modo che le frequenze a cui vengono sollecitati (cammino dei pedoni, traffico stradale, raffiche di vento) non coincidano con frequenze di risonanza del ponte stesso.

È significativo il caso del Millennium Bridge di Londra che fu chiuso a tre soli giorni dall'apertura poiché oscillava in modo pericoloso nel momento in cui le persone vi camminavano sopra, vedi:
`http://en.wikipedia.org/wiki/Millennium_Bridge_(London)`
Infatti le frequenze laterali di oscillazione del ponte erano vicine alla frequenza del passo umano. Fu successivamente modificato inserendo degli ammortizzatori che aggiungevano forze resistenti. Per maggiori dettagli sulle applicazioni dei problemi di autovalori si veda il lavoro di Tisseur e Meerbergen "The quadratic eigenvalue problem"
`http://eprints.ma.man.ac.uk/466/01/covered/MIMS_ep2006_256.pdf`
dove vengono toccati sistemi dinamici discreti, meccanica dei fluidi, acustica, trattamento dei segnali digitali. Applicazioni allo studio di vibrazione delle rotaie sono mostrati in
`http://math.cts.nthu.edu.tw/Mathematics/preprints/prep2007-1-005.pdf`

2 Il caso di un punto materiale

Studiamo il problema nel caso di una sola particella, cioè di un punto materiale P di massa m che si può muovere lungo una retta ed è soggetto a una forza elastica $F = -kOP$ di centro O e costante elastica $k > 0$, e a una forza resistente $R = -\theta v$, di attrito, proporzionale alla velocità v del punto con costante $\theta \geq 0$.



L'equazione di moto per questo semplice modello, ricavata dalla equazione generale $ma = F$, è

$$ma = -kOP - \theta v, \tag{1}$$

dove a è l'accelerazione del punto P Indicando con $x = x(t)$ l'ascissa del punto P al tempo t si ottiene l'equazione differenziale

$$mx'' + \theta x' + kx = 0 \tag{2}$$

che, completata con le condizioni iniziali $x(0) = x_0$, $x'(0) = v_0$, individua una soluzione che costituisce l'espressione del moto del punto materiale p. Nell'ipotesi in cui θ e k siano costanti rispetto a t, l'insieme delle soluzioni dell'equazione differenziale si può esprimere in modo semplice mediante le soluzioni di un'equazione algebrica.

Cerchiamo soluzioni del tipo $x(t) = e^{\lambda t}$. Vale $x'(t) = \lambda e^{\lambda t}$, $x''(t) = \lambda^2 e^{\lambda t}$. Per cui sostituendo nell'equazione (2) si ottiene

$$m\lambda^2 e^{\lambda t} + \theta\lambda e^{\lambda t} + ke^{\lambda t} = 0,$$

da cui, essendo $e^{\lambda t} \neq 0$ si ottiene che una soluzione del tipo $e^{\lambda t}$ esiste se e solo se λ è soluzione dell'equazione algebrica

$$m\lambda^2 + \theta\lambda + k = 0.$$

Si hanno tre casi:

1. Se l'equazione ha due soluzioni distinte λ_1, λ_2 allora tutte le soluzioni di (2) si possono scrivere come

$$x(t) = \gamma_1 e^{\lambda_1 t} + \gamma_2 e^{\lambda_2 t}.$$

2. Se l'equazione ha una soluzione doppia $\lambda_1 = \lambda_2$ allora tutte le soluzioni di (2) si possono scrivere come

$$x(t) = (\gamma_1 + \gamma_2 t)e^{\lambda_1 t}.$$

3. Se l'equazione ha una coppia di soluzioni complesse coniugate $\lambda_{1/2} = \alpha \pm \beta \boldsymbol{i}$ allora $e^{\lambda_{1/2} t} = e^{t(\alpha \pm \beta \boldsymbol{i})} = e^{\alpha t}(\cos(\beta t) \pm \boldsymbol{i} \sin(\beta t))$. Per cui prendendo somma e differenza delle due soluzioni otteniamo una nuova base reale per lo spazio lineare delle soluzioni formata da $e^{\alpha t}\cos(\beta t)$ e $e^{\alpha t}\sin(\beta t)$. Otteniamo quindi per la soluzione generale l'espressione:

$$x(t) = e^{\alpha t}(\gamma_1 \cos(\beta t) + \gamma_2 \sin(\beta t)).$$

cioè il punto oscilla attorno ad O con frequenza $\beta/(2\pi)$ e ampiezza $e^{\alpha t}$. Le oscillazioni si smorzano se $\alpha < 0$, si amplificano se $\alpha > 0$.

In ogni caso l'insieme delle soluzioni costituisce uno spazio lineare di dimensione 2 e le costanti γ_1, γ_2 vengono determinate imponendo le due condizioni iniziali.

Poiché nel nostro caso è

$$\lambda_{1/2} = (-\theta \pm \sqrt{\theta^2 - 4km})/(2m)$$

si ha che, se $\theta^2 - 4km > 0$, cioè se la forza resistente domina la forza elastica, le soluzioni dell'equazione algebrica sono reali e negative. Siamo quindi nel caso 1. In questo caso il punto P converge a O senza oscillare.

Se invece $\theta^2 - 4km < 0$, cioè la forza resistente è dominata dalla forza della molla, le soluzioni sono oscillanti attorno ad O ma l'ampiezza delle oscillazioni converge a zero (oscillazioni smorzate) essendo $\alpha < 0$. La frequenza delle oscillazioni è data da $\frac{1}{2\pi}\sqrt{k/m - \theta^2/(4m^2)}$. Siamo quindi nel caso 3).

Il caso 2) si incontra quando $\theta^2 - 4km = 0$ e vale $\lambda_{1/2} = -\theta/(2m)$ e le soluzioni sono del tipo $e^{-t\frac{\theta}{2m}}(\gamma_1 + \gamma_2 t)$.

Nel caso di un'equazione non omogenea

$$mx'' + \theta x' + kx + f = 0$$

dove f è una funzione di t, l'insieme di tutte le soluzioni si ottiene sommando una qualsiasi soluzione della non omogenea con tutte le soluzioni dell'omogenea. Ad esempio, se f è costante allora una soluzione della non omogenea è data da $-f/k$.

Possiamo simulare il suono generato dalle vibrazioni di un punto materiale come quello descritto, registrando i valori numerici della funzione $e^{\alpha t}(\gamma_1 \cos \beta t + \gamma_2 \sin \beta t)$ per particolari valori delle costanti che intervengono e per un intervallo di tempo fissato. Il file con i valori numerici deve poi essere trasformato in un file di tipo cdr "suonabile" col comando "play" disponibile nelle *release* di Linux.

Nel listing 1 riportiamo un programma scritto da Sergio Steffé che trasforma due file numerici di nome `sx` e `dx` con i valori delle ampiezze di due oscillazioni istante per istante in un file di tipo cdr, quindi suonabile con il comando play, di nome `suono.cdr`. Il programma crea un file "stereo" a partire da due file poichè prevede di trattare due canali audio, uno destro con i dati numerici nel file `dx` e uno sinistro con i dati numerici nel file `sx`. In assenza di due sorgenti diverse di oscillazioni noi possiamo scegliere `sx=dx`. Nel programma si assume che i file `sx`

e `dx` contengano nell'ordine: il numero di valori con cui è campionato il suono, i valori numerici di ciascun campione ognuno su una riga diversa del file. I valori sono campionati alla distanza temporale di 1/44.100 secondi per cui un secondo di suono è ottenuto con 44.100 campioni.

Nel programma è usato il comando `MVBITS` del Fortran 90 per la manipolazione dei bit e sono usate alcune specifiche del Fortran 90 relative all'istruzione `OPEN` di apertura di file per la lettura/scrittura di dati quali `REPLACE`, `UNFORMATTED`, `DIRECT`, `RECL`. In particolare, l'apertura del file di scrittura viene fatta col comando

```
OPEN(unit=10,file='suono.cdr',status='REPLACE',  &
       form='UNFORMATTED',access='DIRECT',RECL=4)
```

dove il significato delle varie opzioni è il seguente:

```
REPLACE: se il file esiste gia' lo sovrascrive, altrimenti lo crea
UNFORMATTED: viene usato un formato interno per salvare i dati
DIRECT:  usa l'accesso ad un qualsiasi record
RECL:  e' la lunghezza di ogni record (4 bytes)
```

Per maggiori dettagli a riguardo si rimanda a un manuale del Fortran 90.

Nel caso in cui il punto materiale sia vincolato a stare su un piano, e quindi P sia individuato da due coordinate (x, y), l'equazione (1) riscritta nelle due componenti x e y dà origine a due equazioni differenziali

$$mx'' + \alpha x' + kx = 0,$$
$$my'' + \alpha y' + ky = 0.$$

Analogamente nel caso di un punto libero di muoversi nello spazio abbiamo tre equazioni differenziali. Le equazioni sono indipendenti nel senso che possono essere risolte separatamente e le loro soluzioni determinano le coordinate al tempo t del punto materiale.

In questi casi è facile dare una espressione esplicita della soluzione esprimendo separatamente le componenti in temini di esponenziali e di funzioni trigonometriche. Il problema diventa più complesso per sistemi formati da più punti materiali.

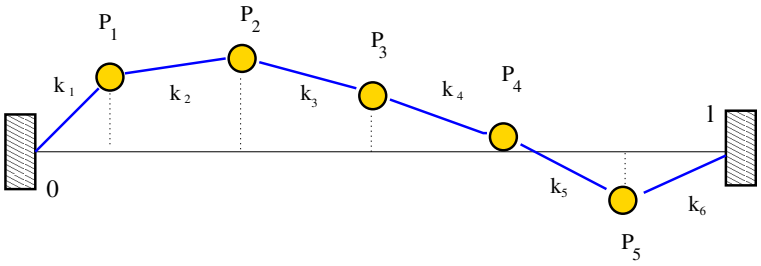
3 Un sistema formato da più particelle

Supponiamo ora di avere n punti materiali $P_1, P_2, \ldots, P_n$ di massa $m_1, m_2, \ldots, m_n$, che possono muoversi liberamente nel piano cartesiano. Assegnamo le coordinate ad ogni punto $P_i = (x_i, y_i)$ e definiamo $P_0 = (0, 0)$, $P_{n+1} = (\ell, 0)$ indipendenti dal tempo. Supponiamo che ogni punto sia soggetto ad una forza elastica e ad una forza resistente. Più precisamente il punto P_j è soggetto alla forza

Listing 1: Programma creasuono

```
! Sergio Steffe' - Laboratorio Sperimentale di Matematica Computazionale
! Dipartimento di Matematica - Universita' di Pisa - AA 2002/2003
! Modificato da Dario Bini

PROGRAM creasuono
! creasuono.f90
! crea un suono nel file suono.cdr, leggendo i dati dal file dx per
! il canale destro e sx per il canale sinistro
! 16 bit stereo 44.100Hz big-endian formato raw

  IMPLICIT NONE
  INTEGER, PARAMETER :: iduemax=32767
  INTEGER :: i, ntot !numero dei records
  REAL soundds,soundsn !canali ds e sn
  INTEGER(KIND=2) ii,iids,iisn !interi di due bytes

!si apre il file di scrittura
  OPEN(unit=10,file='suono.cdr',status='REPLACE', &
       form='UNFORMATTED',access='DIRECT',recl=4)

!si aprono i file di lettura
  OPEN(unit=71,file="dx")
  OPEN(unit=72,file="sx")
  READ(71,*)ntot
  READ(72,*)ntot

!inizia il ciclo
  DO i=1,ntot
     READ(71,*)soundds
     READ(72,*)soundsn
     soundds=soundds*iduemax
     soundsn=soundsn*iduemax
     ii=nint(soundds)
     CALL mvbits(ii,0,8,iids,8)
     CALL mvbits(ii,8,8,iids,0)
     ii=nint(soundsn)
     CALL mvbits(ii,0,8,iisn,8)
     CALL mvbits(ii,8,8,iisn,0)
     WRITE(unit=10,rec=i) iids,iisn
  END DO
END PROGRAM creasuono
```

elastica $F_j^{(1)} = -k_j(P_j - P_{j-1})$ esercitata dal punto P_{j-1} e alla forza elastica $F_j^{(2)} = -k_{j+1}(P_j - P_{j+1})$ esercitata dal punto P_{j+1}, e alla forza resistente $R_j = -\theta_j v_j$, dove $\theta_j \geq 0$ e v_j è la velocità di P_j. Inoltre i punti P_1 e P_n sono soggetti alle forze elastiche rispettivamente $-k_1(P_1 - P_0)$ e $-k_{n+1}(P_n - P_{n+1})$. La figura seguente mostra la situazione con $n = 5$:



In questo caso, posto a_j l'accelerazione di P_j, le equazioni di moto del sistema di punti materiali sono

$$a_j m_j = -k_j(P_j - P_{j-1}) - k_{j+1}(P_j - P_{j+1}) - \theta v_j, \quad j = 1, \ldots, n.$$

Proiettando le equazioni lungo l'asse delle x e delle y si ottengono i sistemi

$$\begin{aligned} &m_j x_j'' + \theta x_j' - k_j x_{j-1} + (k_j + k_{j+1}) x_j - k_{j+1} x_{j+1} = 0, \\ &m_j y_j'' + \theta y_j' - k_j y_{j-1} + (k_j + k_{j+1}) y_j - k_{j+1} y_{j+1} = 0, \end{aligned} \qquad j = 1, \ldots, n.$$

Questi sistemi possono essere riscritti in forma matriciale nel seguente modo. Posto $M = \text{diag}(m_1, \ldots, m_n)$, K tale che $(K)_{i,i} = k_i + k_{i+1}$, $i = 1, \ldots, n$, $(K)_{i,i+1} = (K)_{i+1,i} = -k_{i+1}$, $i = 1, \ldots, n-1$, R =diag$(\theta_1, \ldots, \theta_n)$, $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{y} = (y_1, \ldots, y_n)$, vale

$$M\boldsymbol{y}'' + R\boldsymbol{y}' + K\boldsymbol{y} = 0 \tag{3}$$

con le condizioni iniziali

$$\boldsymbol{y}(0) = \boldsymbol{y}_0, \quad \boldsymbol{y}'(0) = \boldsymbol{v}_0,$$

e le condizioni ai bordi $y_0(t) = y_{n+1}(t) = 0$ per ogni t. Analogamente per l'altro sistema

$$M\boldsymbol{x}'' + R\boldsymbol{x}' + K\boldsymbol{x} = 0 \tag{4}$$

con le condizioni iniziali

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \quad \boldsymbol{x}'(0) = \boldsymbol{x}_0,$$

e le condizioni ai bordi $x_0(t) = 0$, $x_{n+1}(t) = \ell$ per ogni t. Nel caso volessimo introdurre la forza peso allora le equazioni per le y_i cambiano in

$$M\boldsymbol{y}'' + R\boldsymbol{y}' + K\boldsymbol{y} + \boldsymbol{g} = 0$$

dove $\boldsymbol{g}$ è il vettore di componenti $(-gm_i)$, g accelerazione di gravità.

Si osservi che K è matrice tridiagonale, inoltre se le costanti elastiche k_i sono non nulle la matrice è irriducibile.

Si osservi che, analogamente al caso monodimensionale, possiamo cercare soluzioni del tipo $\boldsymbol{y}(t) = \boldsymbol{w}e^{\lambda t}$, dell'equazione omogenea dove però stavolta $\boldsymbol{w} \in \mathbb{R}^n$ è un vettore di costanti. Poichè $\boldsymbol{y}'(t) = \lambda \boldsymbol{w}e^{\lambda t}$, $y''(t) = \lambda^2 \boldsymbol{w}e^{\lambda t}$, sostituendo nell'equazione differenziale si ottiene

$$(\lambda^2 M + \lambda R + K)\boldsymbol{w} = 0.$$

Esistono allora soluzioni del tipo ipotizzato se solo se λ risolve l'equazione $\det(\lambda^2 M + \lambda R + K) = 0$ e $\boldsymbol{w}$ sta nel nucleo di tale matrice. Il problema del calcolo delle soluzioni $\lambda, \boldsymbol{w}$ è noto come problema quadratico agli autovalori ed è ampiamente studiato in letteratura, si veda ad esempio l'articolo di Tisseur e Meerbergen già citato.

Un modo di risolverlo è quello di ricondurlo ad un problema standard agli autovalori. Noi faremo questa trasformazione direttamente sull'equazione differenziale (3) in modo da trasformarla in una equazione del primo ordine.

3.1 Riduzione di un sistema dal secondo ordine al primo ordine.

È possibile trasformare il sistema di equazioni differenziali del secondo ordine (3) in un sistema di equazioni differenziali del primo ordine nel modo seguente.

Sia $\boldsymbol{w}^T = (\boldsymbol{y}^T, \boldsymbol{y}'^T)$ il vettore di $2n$ componenti ottenuto giustapponendo il vettore delle velocità $\boldsymbol{y}'$ al vettore degli spostamenti $\boldsymbol{y}$, per cui $\boldsymbol{w}'^T = (\boldsymbol{y}'^T, \boldsymbol{y}''^T)$. Vale allora

$$\boldsymbol{w}' = A\boldsymbol{w}, \quad A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}R \end{bmatrix} \tag{5}$$

e le condizioni iniziali diventano

$$\boldsymbol{w}(0) = \boldsymbol{w}_0, \quad \boldsymbol{w}_0^T = (\boldsymbol{y}_0^T, \boldsymbol{y}_0'^T).$$

Il sistema di equazioni differenziali scritto nella forma (5) può essere facilmente risolto se siamo in grado di calcolare gli autovalori e gli autovettori della matrice A. Supponiamo per semplicità che A sia simile ad una matrice diagonale D, cioè $A = SDS^{-1}$, dove $D = \operatorname{diag}(d_1, \ldots, d_n)$ e S ha per colonne gli autovettori $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ corrispondenti agli autovalori $d_1, \ldots, d_n$ di A.

Il sistema allora diventa $\boldsymbol{w}' = SDS^{-1}\boldsymbol{w}$, e posto $\boldsymbol{u} = S^{-1}\boldsymbol{w}$ si ottiene $\boldsymbol{u}' = D\boldsymbol{u}$. Cioè, in componenti si ha $u_j' = d_j u_j$, $j = 1, \ldots, n$, da cui $u_j(t) = e^{td_j} + c_j$. La soluzione risulta allora

$$\boldsymbol{w} = S\boldsymbol{u} = S\operatorname{diag}(e^{td_i})\boldsymbol{\gamma}, \quad \boldsymbol{\gamma} = (\gamma_i) \tag{6}$$

dove $\gamma_1, \ldots, \gamma_{2n}$ sono costanti che vengono determinate dalle condizioni iniziali

$$\boldsymbol{w}(0) = \begin{bmatrix} \boldsymbol{y}(0) \\ \boldsymbol{y}'(0) \end{bmatrix},$$

cioè

$$\boldsymbol{\gamma} = S^{-1} \left[\begin{array}{c} \boldsymbol{y}(0) \\ \boldsymbol{y}'(0) \end{array} \right]. \tag{7}$$

L'algoritmo di risoluzione diventa il seguente:

1. leggi i dati n, k_i, m_i, θ_i, $i = 1, \ldots, n$;
2. leggi le condizioni iniziali;
3. costruisci la matrice K in base alla distribuzione delle forze elastiche;
4. costruisci la matrice A;
5. calcola gli autovalori $\boldsymbol{d} = (d_i)$ e la matrice S degli autovettori di A;
6. calcola $\boldsymbol{\gamma}$ tramite (7) imponendo le condizioni iniziali;
7. salva su un file le ampiezze di oscillazione di una particella (ad esempio quella centrale) in quantità corrispondente a tre secondi di suono alla frequenza di campionamento di 44.100 campioni al secondo mediante la (6);
8. salva su un altro file analoghi valori per un'altra particella del sistema di particelle.

Il programma che implementa l'algoritmo è mostrato nel listing 2. I dati non vengono letti da file ma sono costruiti all'interno del programma stesso. Nel programma prendiamo valori uguali per le masse, costanti elastiche e forze di attrito.

Il programma utilizza due sottoprogrammi per la risoluzione di un sistema con matrice complessa, e per il calcolo degli autovalori e autovettori. Questi due programmi sono riportati di seguito e utilizzano due subroutine della libreria LAPACK Linear Algebra PACKage `www.netlib.org/lapack`. Tale libreria, di pubblico dominio, contiene l'implementazione degli algoritmi più efficienti per risolvere problemi di algebra lineare numerica. I due sottopragrammi di interfaccia sono riportati nei listing 4 e 17.

4 Sperimentazione

Riportiamo ora i risultati di alcune sperimentazioni fatte col nostro modello matematico. Consideriamo il caso riguarda un corda discreta costituita da $n = 40$ masse uguali tra di loro con il valore $1/(1000n)$ con costanti elastiche uguali tra di loro pari a 4.0×10^4 e coefficienti di attrito pari a 10^{-4}. Si studiano due condizioni iniziali:

a) le velocità sono nulle e gli spostamenti sono $y_i = i/q$, $i = 1, \ldots, q$, $y_i = 1 - (i - q)/(n + 1 - q)$, per $i = q + 1, \ldots n$, dove q è un intero. Questa è la simulazione di una corda pizzicata nel punto q come nel caso del clavicembalo.

Listing 2: Programma vibrazioni

```
program vibrazioni
    implicit none
    integer, parameter :: dp=kind(0.d0)
    real(dp),dimension(:),allocatable :: k, a, m
    real(dp),dimension(:,:),allocatable :: mat
    complex(dp),dimension(:),allocatable :: f, eigval, g
    complex(dp),dimension(:,:),allocatable :: eigvec, aux
    integer :: n, i, cont, kk
    integer :: frame, maxframe
    real(dp) :: t
    real :: dh, p, sample=1.0/44100

! Dati
    write(*,*)"assegna il numero n di punti (sugg. n=40)"
    read(*,*)n
    allocate(k(1:n+1), a(1:n), m(1:n), f(1:2*n))
    m = 1.d-3/n ! valori delle masse
    a = 1.d-4 ! forze di attrito
    k = 4.d4 ! costanti elastiche

!calcolo delle condizioni iniziali: corda pizzicata nel punto kk
   kk=n/2
   do i=1,kk
       f(i) = (i*1.d0)/kk
   end do
   do i=kk+1,n
       f(i) = 1-(i-kk)*1.d0/(n+1.d0-kk)
   end do

! linearizzazione del problema differenziale: costruisce matrice 2n x 2n
   allocate(mat(2*n,2*n))
   mat=0
   do i=1,n
       mat(i,n+i) = 1.d0;
       mat(n+i,n+i) = -a(i)/m(i)
       mat(n+i,i) = -(k(i)+k(i+1))/m(i)
   end do
   do i=1,n-1
       mat(n+i+1,i) = k(i+1)/m(i)
       mat(n+i,i+1) = k(i+1)/m(i+1)
   end do

! calcolo autovalori/autovettori
   allocate(aux(2*n,2*n), eigvec(2*n,2*n), eigval(2*n),g(2*n))
   call autovalori(2*n,mat,eigval,eigvec)
   aux=eigvec
   call sistema_lineare(2*n,aux,f)
```

Listing 3: Programma vibrazioni (continua)

```
! calcolo situazione al trascorrere del tempo
! salvo il file per creare un suono: nel canale destro metto la
! vibrazione del secondo punto della corda e nel canale sinistro metto
! la vibrazione del punto di mezzo
   open(unit=10,file='dx')
   open(unit=11,file='sx')
   maxframe=120000; ! max numero valori salvati: corrisponde a ~ 3 sec
   dh=1.d0/(n+1)
   write(10,*) maxframe
   write(11,*) maxframe
   do frame = 1, maxframe
      t = sample*frame
      g = f*exp(t*eigval)
      p = dot_product(eigvec(2,:),g)
      write(10,*) p
      p = dot_product(eigvec(n/2,:),g)
      write(11,*) p
   end do
end program vibrazioni
```

Listing 4: Programma sistema_lineare

```
SUBROUTINE sistema_lineare(n,a,f)
! subroutine di interfaccia con Lapack per risolvere il sistema lineare
    Ax=f
! in input: a e' la matrice e f il termine noto
! in output f e' la soluzione del sistema
   implicit none
   INTEGER, PARAMETER :: dp = KIND(0.d0)
   INTEGER :: n
   COMPLEX(dp), DIMENSION(n) :: f
   COMPLEX(dp), DIMENSION(n,n) :: a
! variabili locali
   INTEGER :: info
   COMPLEX(dp), DIMENSION(N,N) :: aux
   INTEGER, DIMENSION(N) :: ipiv

   aux = a
   CALL zgesv( n, 1, aux, n, ipiv, f, v, info )
END SUBROUTINE sistema_lineare
```

Listing 5: Programma autovalori

```
SUBROUTINE autovalori(n,a,eigval,eigvec)
! Subroutine di interfaccia con lapack per il calcolo di autovalori
! e autovettori di una matrice complessa. Usa la subroutine ZGEEV di
    lapack
   INTEGER,PARAMETER :: dp = KIND(0.d0)
   INTEGER :: n
   COMPLEX(dp), DIMENSION(n) :: eigval
   REAL(dp), DIMENSION(n,n) :: a
   COMPLEX(dp), DIMENSION(n,n) :: eigvec
! variabili locali
   COMPLEX(dp), DIMENSION(1,n) :: vl
   COMPLEX(dp), DIMENSION(2*n) :: work
   COMPLEX(dp), DIMENSION(n,n) :: aux
   INTEGER :: info, i
   COMPLEX(DP), DIMENSION(2*N) :: rwork
   CHARACTER :: jobvl='N',jobvr='V'

   aux=a
   CALL zgeev( jobvl, jobvr, n, aux, n, eigval, vl, 1, &
               eigvec, n, work, 2*n, rwork, info )
END SUBROUTINE autovalori
```

b) Si considera invece il caso iniziale di spostamenti nulli e di velocità date da $y'_i = 10^4 * \exp(-0.1 * (i-q)^2)$ che simula il caso della corda colpita da un martelletto nel punto q come nel caso del pianoforte.

In entrambi i casi nel canale dx si mette la vibrazione del secondo punto della corda, nel canale sx si mette la vibrazione del punto centrale.

I valori delle frequenze di vibrazione (parte immaginaria degli autovalori diviso 2π) calcolati dal programma sono

```
12723.054712966446 26.088644264238155
12695.038600756858 26.03119718084567
12648.390768763584 25.93554572590614
12583.179679112636 25.80183028111967
12499.501038070426 25.630247092340305
12397.477655581537 25.421047981558022
12277.259265028317 25.174539977315465
12139.022303476508 24.89108486410127
11982.969652729344 24.57109865138106
11809.330341570667 24.21505096304648
11618.359209633607 23.82346434817738
11410.336533388087 23.396913514128595
11185.567614796659 22.936024483068064
10944.382333241627 22.441473673202662
```

```
10687.134661381559 21.913986906041224
10414.202145647607 21.35433834115138
10125.985352141903 20.76334933997333
9822.907278751756 20.141887260359066
9505.412734341568 19.49086418360437
9173.967685934569 18.81123557584392
8829.058574841923 18.10399888577292
8471.191602742732 17.37019208075298
8100.891988763158 16.610892123451656
7718.703198644738 15.827213391250849
7325.186147133079 15.020306040743518
6920.9183747576535 14.191354319719327
6506.4932002108435 13.3415748291165
6082.518849569934 12.472214737490111
5649.617563640088 11.584549950617415
5208.424684727946 10.679883238925635
4759.587724185709 9.759542325489589
4303.765412093086 8.824877937402999
3841.626730469931 7.877261823379432
3373.849931434018 6.918084740483228
2901.12154173489 5.948754412924554
2424.135355098419 4.970693465860291
1943.5914137823022 3.9853373370719316
1460.194980548174 2.994132168992735
974.6555011160505 1.998532681218886
487.68554563822164 1.
```

dove si riporta (secondo numero) anche il rapporto con la frequenza più bassa. Si osservi che tali rapporti sono spesso vicini a numeri interi.

Caso a) Con $q = n/2$. I grafici dei due segnali relativi al primo quarantesimo di secondo (1000 campioni). sono riportati nella figura 1

Si noti come nella oscillazione del punto vicino all'estremità (figura di destra) siano evidenti componenti in alta frequenza che non compaiono nel caso della oscillazione del punto centrale.

Con $q = 2$ I grafici dei due segnali relativi al primo quarantesimo di secondo (1000 campioni). sono riportati nella figura 2

Osservate la presenza di una maggior quantità di componenti in alta frequenza sia nella figura di destra che di sinistra: pizzicare un corda in prossimità di un suo estremo dà un suono più argentino, sia se questo viene rilevato in prossimità di un estremo della corda (figura di destra) sia che venga rilevato nel centro (figura a sinistra).

Caso b) Con $q = n/2$ i grafici dei due canali sono riportati nella figura 3

Si noti in questo caso l'assenza di componenti in alta frequenza sui due canali. La corda non è pizzicata ma è idealmente percossa da un martelletto nella parte centrale

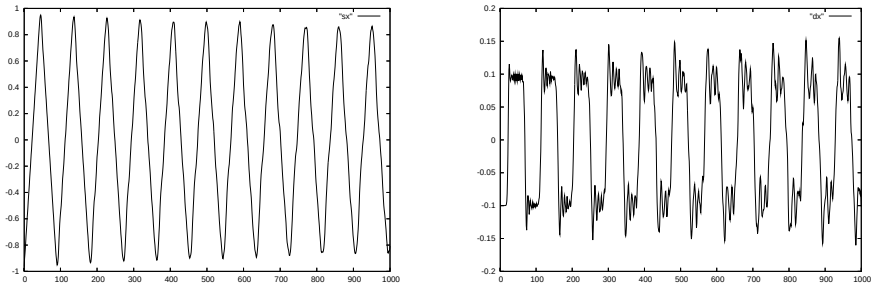


Figura 1: Grafici del primo quarantesimo di secondo della vibrazione di una corda pizzicata al centro. A sinistra oscillazioni rilevate al centro della corda; a destra oscillazioni vicino ad un estremo

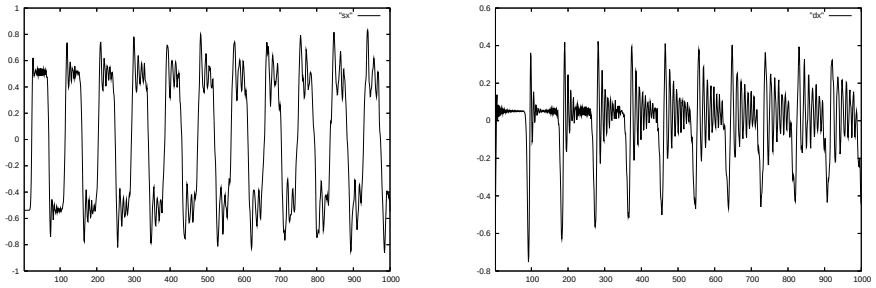


Figura 2: Grafici del primo quarantesimo di secondo della vibrazione di una corda pizzicata vicino ad un bordo. A sinistra oscillazioni rilevate al centro della corda; a destra oscillazioni vicino ad un estremo

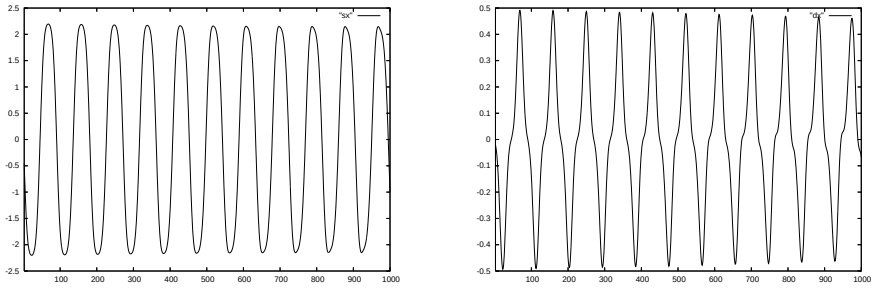


Figura 3: Simulazione della corda colpita da un martelletto al centro. A sinistra oscillazioni rilevate al centro della corda; a destra oscillazioni vicino ad un estremo

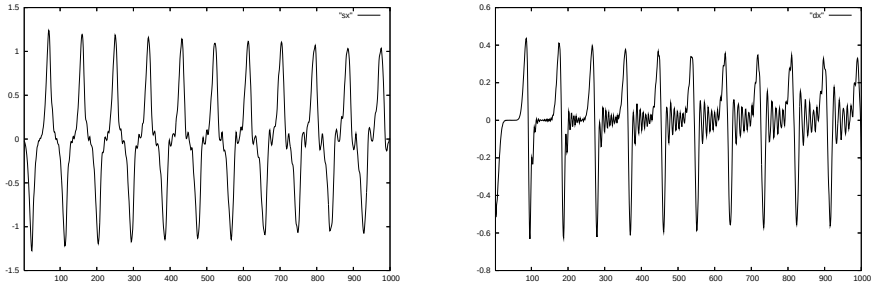


Figura 4: Simulazione della corda colpita da un martelletto vicino a un bordo. A sinistra oscillazioni rilevate al centro della corda; a destra oscillazioni vicino ad un estremo

Con $q = 2$, cioè se il punto di percussione è vicino al bordo, i grafici dei due canali sono riportati nella figura

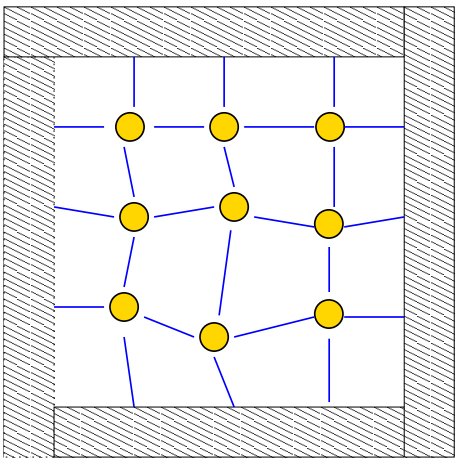
Anche in questo caso ci sono maggiori componenti in alta frequenza.

Vediamo ora alcuni esercizi.

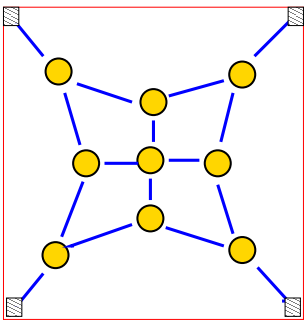
Esercizio 1. La simulazione fatta con le condizioni iniziali descrive il comportamento di una corda pizzicata (clavicembalo). Si provino altre condizioni iniziali, ad esempio, per simulare la vibrazione di una corda colpita da un martelletto (tipo pianoforte) si ponga $\mathbf{y_0} = \mathbf{0}$ e $\mathbf{y_0}'$ di componenti $1000 * \exp(-(i - n/2)^2)$. Si provi a vedere come suona una corda in cui un punto subisce una forza resistente superiore rispetto agli altri punti. Si provi a simulare una corda con una distribuzione di massa non uniforme

Esercizio 2. Si adatti il programma al caso del calcolo dei x_i. L'unica differemza rispetto al caso delle y_i è che le condizioni al bordo non sono omogenee essendo $x_{n+1} = \ell$. Si provi a registrare, nei file dx e sx anziché i valori di un particolare punto y_k, la media dei valori in un intorno di un particolare punto.

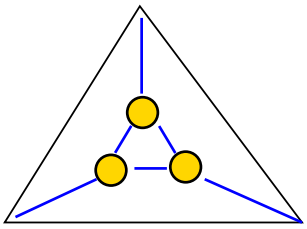
Esercizio 3. Si provino a costruire le matrici di elasticità K nel caso di un sistema di punti n^2 $p_{i,j}$, $i, j = 1, \ldots, n$ liberi di muoversi nel piano in cui il punto $P_{i,j}$ di massa $m_{i,j}$ è soggetto alle quattro forze elastiche esercitate dai punti "contigui" $p_{i,j\pm1}$, $p_{i\pm1,j}$, come in figura.
(Suggerimento: si definiscano le matrici $X = (x_{i,j})$, $Y = (y_{i,j})$ in cui $(x_{i,j}, y_{i,j})$ sono le coordinate del punto $P_{i,j}$; si definiscano anche le matrici K_x e K_y delle costanti elastiche delle forze "orizzontali" e "verticali").



Esercizio 4. Si consideri un problema analogo al prcedente dove solo i punti sui vertici sono soggetti a forze elastiche dirette verso i vertici di un quadrato.
(Suggerimento: modificare il problema precedente ponendo uguali a zero alcuni elementi delle matrici K_x e K_y)

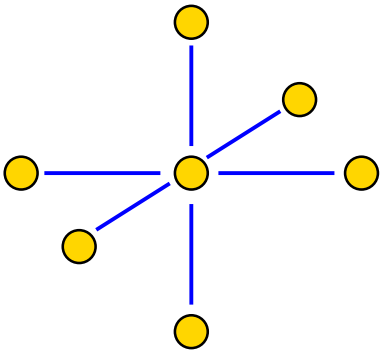


Esercizio 5. Si consideri il caso in figura



Esercizio 6. Si scrivano le equazioni di moto per un sistema come quelli dei tre casi precedenti dove i punti sono liberi di muoversi nello spazio.

Esercizio 7. Si scrivano le equazioni di moto di un sistema di n^3 punti $P_{i,j,k}$ liberi di muoversi nello spazio in cui il punto $P_{i,j,k}$ è soggetto alle 6 forze elastiche esercitate dai punti contigui $P_{i\pm1,j,k}$, $p_{i,j\pm1,k}$, $p_{i,j,k\pm1}$.



5 Un approccio diverso: l'esponenziale di matrice

Per risolvere il problema $\boldsymbol{w}' = A\boldsymbol{w}$ si consideri la seguente funzione di matrice

$$\exp(A) = \sum_{i=0}^{\infty} \frac{A^i}{i!}$$

che estende la funzione esponenziale al caso di matrici. Si osservi che la funzione è ben definita poichè la serie è convergente per ogni matrice quadrata A. Vale inoltre

$$\frac{d\exp(tA)}{dt} = A\exp(tA)$$

per cui possiamo descrivere le soluzioni dell'equazione differenziale $\boldsymbol{w}' = A\boldsymbol{w}$ come

$$\boldsymbol{w}(t) = \exp(tA)\boldsymbol{w}_0$$

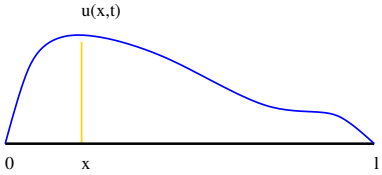
dove $\boldsymbol{w}(0) = \boldsymbol{w}_0$.

La simulazione delle vibrazioni di una corda basate sul calcolo dell'esponenziale di una matrice è più onerosa poiché per ogni valore di t occorre calcolare l'esponenziale. Ciò ha un costo maggiore che non calcolare il valore di y_k una volta noti gli autovalori e autovettori di A.

6 Sistemi continui: l'equazione delle onde

I problemi trattati nelle sezioni precedenti hanno riguardato lo studio di vibrazioni di sistemi formati da un numero finito di punti materiali. Nel caso si considerino sistemi continui quali una corda vincolata a due estremità o una membrana elastica vincolata a un certo bordo dato da una linea chiusa, il modello matematico da considerare è formato da equazioni differenziali.

Consideriamo il problema di una corda elastica vibrante sottesa tra due estremi di coordinate $(0, 0)$ e $(\ell, 0)$ descritta come descritta in figura



dove denotiamo con $u(x, t)$ l'ordinata del suo punto di ascissa x al tempo t. Il moto dei punti di questa corda è governato dalla seguente equazione differenziale nota come *equazione delle onde*

$$\frac{\partial^2}{\partial t^2}u(x,t) - \sigma\frac{\partial^2}{\partial x^2}u(x,t) = 0 \tag{8}$$

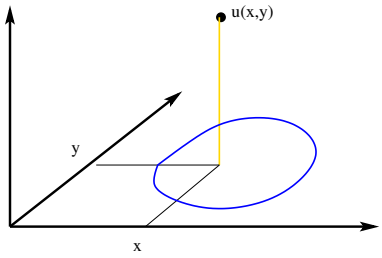
con $\sigma = Tg/w$, dove T è la tensione della corda, g è l'accelerazione di gravità e w è il peso per unità di lunghezza (C. Gerald, P.O. Wheatly, "Applied Numerical Analysis", Pearson, Addison-Wesley, 2004).

Tale equazione viene complementata con le seguenti condizioni aggiuntive

$$\begin{array}{ll} u(x,0) = p(x), \quad u'(x,0) = v(x), \quad x \in [0,\ell] & \text{Condizioni iniziali} \\ u(0,t) = 0, \quad u(\ell,t) = 0 & \text{Condizioni al bordo} \end{array} \tag{9}$$

La corda viene quindi abbandonata al tempo $t = 0$ in posizione $p(x)$ con velocità iniziale $v(x)$.

Nel caso bidimensionale di una membrana elastica vincolata a stare su un contorno chiuso Γ che racchiude un insieme aperto $\Omega \subset \mathbb{R}^2$, come mostrato in figura



l'equazione diventa

$$\frac{\partial^2}{\partial t^2}u(x,y,t) - \sigma(\frac{\partial^2}{\partial x^2}u(x,y,t) + \frac{\partial^2}{\partial y^2}u(x,y,t)) = 0, \quad (x,y) \in \Omega \tag{10}$$

con le condizioni aggiuntive

$$\begin{array}{ll} u(x,y,0) = p(x,y), \quad u'(x,y,0) = v(x), \quad (x,y) \in \Omega & \text{Condizioni iniziali} \\ u(x,y,t) = 0, \quad (x,y) \in \Gamma & \text{Condizioni al bordo} \end{array} \tag{11}$$

Siamo interessati alle soluzioni in cui tutti i punti oscillano con la stessa frequenza, cioè soluzioni del tipo $u(x,t) = v(x)\cos\omega t$, $v(0) = v(\ell) = 0$. Sostituendo questa espressione in (8), poiché $(\cos\omega t)'' = -\omega^2\cos\omega t$, si ottiene

$$\sigma v''(x) = -\lambda v(x), \quad \lambda = \omega^2/\sigma \tag{12}$$

Esistono diversi modi di approssimare la soluzione dell'equazione delle onde. Uno di questi si basa sul metodo delle differenze finite in cui si approssima l'operatore derivata seconda con opportuni rapporti incrementali. Per questo occorre assumere che la soluzione $v(x)$ sia di classe $C^4[0,\ell]$.

Si fissi un intero $n > 0$, si ponga $h = \ell/(n+1)$ e si definiscano i punti $x_i = ih$, $i = 0, 1, \ldots, n+1$. Si considerino i seguenti sviluppi in serie:

$$\begin{aligned} v(x_{i+1}) &= v(x_i) + hv'(x_i) + \tfrac{h^2}{2}v''(x_i) + \tfrac{h^3}{3!}v'''(x_i) + \tfrac{h^3}{4!}v^{(4)}(\xi_i) \\ v(x_{i-1}) &= v(x_i) - hv'(x_i) + \tfrac{h^2}{2}v''(x_i) - \tfrac{h^3}{3!}v'''(x_i) + \tfrac{h^3}{4!}v^{(4)}(\eta_i) \end{aligned}$$

dove $\xi_i \in (x_i, x_{i+1})$, $\eta_i \in (x_{i-1}, x_i)$. Sommando entrambi i membri delle due equazioni e ricavando $v''(x_i)$ si ha

$$v''(x_i) = \frac{1}{h^2}(v_{i-1} - 2v_i + v_{i+1}) - h^2\tau_i$$

dove si è posto $v_i = v(x_i)$, $\tau_i = \frac{h^3}{4!}v^{(4)}(\xi_i) + \frac{h^3}{4!}v^{(4)}(\eta_i)$. Si osservi che, poiché $v \in C^4[0,\ell]$, risulta $|v^{(4)}(x)| \leq M$ per una costante $M > 0$, per cui $|\tau_i| \leq \frac{1}{12}M$.

Possiamo allora restringere il nostro interesse al calcolo dei soli v_i. Abbiamo la seguente equazione

$$\frac{1}{h^2}(v_{i-1} - 2v_i + v_{i+1}) = -\lambda v_i + h^2\tau_i, \quad i = 1, 2, \ldots, n. \tag{13}$$

In forma matriciale abbiamo

$$\frac{1}{h^2}A\boldsymbol{v} = \lambda\boldsymbol{v} - h^2\boldsymbol{\tau} \tag{14}$$

con $A = \text{tridiag}(-1, 2, -1)$, $\boldsymbol{v} = (v_i)$, $\boldsymbol{\tau} = (\tau_i)$.

Poiché non conosciamo i valori di τ_i e poiché $h \to 0$ è ragionevole considerare il problema agli autovalori

$$A\boldsymbol{w} = \mu\boldsymbol{w}. \tag{15}$$

Dobbiamo però dimostrare che le soluzioni di (15) sono buone approssimazioni delle soluzioni del problema originale (14).

Per questo ci è di aiuto il teorema di Bauer-Fike.

Riscriviamo la (14) nella forma

$$(\frac{1}{h^2}A + \frac{h^2}{||\boldsymbol{v}||_2^2}\boldsymbol{\tau}\boldsymbol{v}^T)\boldsymbol{v} = \lambda\boldsymbol{v} \tag{16}$$

e applichiamo il teorma di Bauer-Fike alle matrici $B = \frac{1}{h^2}A$ e $B + F$ con $F = \frac{h^2}{||\boldsymbol{v}||_2^2}\boldsymbol{\tau}\boldsymbol{v}^T$.

Teorema 1 (Bauer-Fike) *Siano H, B, F matrici $n \times n$ tali che $H = B + F$. Supponiamo che B sia diagonalizzabile, cioè $B = SDS^{-1}$ con D matrice diagonale. Sia inoltre $\|\cdot\|$ una norma assoluta. Allora, per ogni autovalore λ di H esiste un autovalore μ di B tale che*

$$|\lambda - \mu| \leq \|F\| \cdot \|S\| \cdot \|S^{-1}\|$$

Un ulteriore analisi ci permette di determinare una maggiorazione dell'errore sugli autovettori.

Teorema 2 *Nelle ipotesi del teorema di Bauer-Fike, sia $\boldsymbol{u} \in \mathbb{R}^n$ tale che, $(B + F)\boldsymbol{u} = \lambda\boldsymbol{u}$. Allora esistono un autovettore $\boldsymbol{v}$ e un autovalore μ di B tali che $B\boldsymbol{v} = \mu\boldsymbol{v}$ e*

$$\frac{\|\boldsymbol{v} - \boldsymbol{u}\|}{\|\boldsymbol{u}\|} \leq \|(B - \mu I)^+\| \cdot (\|F\| + |\lambda - \mu|) \leq \|(B - \mu I)^+\| \cdot \|F\|(1 + \|S\| \cdot \|S^{-1}\|)$$

dove $(B - \mu I)^+$ è l'inversa generalizzata di $B - \mu I$ e $S^{-1}BS = D$ è diagonale. In particolare, se B è simmetrica risulta

$$\frac{\|\boldsymbol{v} - \boldsymbol{u}\|_2}{\|\boldsymbol{u}\|_2} \leq \frac{2}{\min_{t \in \sigma(B), t \neq \mu} |t - \mu|}\|F\|_2.$$

dove $\sigma(B)$ è l'insieme degli autovalori di B.

Dim. Vale

$$(B + F - \lambda I)\boldsymbol{u} = 0$$
$$B\boldsymbol{v} = \mu\boldsymbol{v}$$

Sottraendo entrambi i membri delle precedenti equazioni si ottiene

$$B(\boldsymbol{u}-\boldsymbol{v})+F\boldsymbol{u}+(\lambda-\mu)\boldsymbol{u}-\mu(\boldsymbol{u}-\boldsymbol{v})=0.$$

Da cui

$$(B-\mu I)(\boldsymbol{u}-\boldsymbol{v})=-F\boldsymbol{u}-(\lambda-\mu)\boldsymbol{u}$$

Quindi $F\boldsymbol{u}+(\lambda-\mu)\boldsymbol{u}$ sta nell'immagine di $B-\mu I$ ed esiste una soluzione $\boldsymbol{u}-\boldsymbol{v}$ di minima norma tale che

$$\boldsymbol{u}-\boldsymbol{v}=-(B-\mu I)^{+}(F+(\lambda-\mu)I)\boldsymbol{u}$$

inoltre vale

$$\|\boldsymbol{u}-\boldsymbol{v}\|\leq\|(B-\mu I)^{+}\|(\|F\|+|\lambda-\mu|)\|\boldsymbol{u}\|.$$

Da cui la prima parte della tesi. Se B è simmetrica, con autovalori β_i, si ha $\|S\|_2\cdot\|S^{-1}\|_2=1$, per cui la tesi discende dal fatto che i valori singolari di $(B-\mu I)^{+}$, per definizione di inversa generalizzata, sono $1/|\beta_i-\mu|$ se $\beta_i-\mu\neq 0$ e 0 altrimenti. □

Nel caso in cui la matrice B è $(1/h^2)\mathrm{trid}(-1,2,-1)$, e $F=\boldsymbol{\tau}\boldsymbol{v}^T$, risulta $||F||=\frac{h^2}{||\boldsymbol{v}||^2}||\boldsymbol{\tau}||\cdot||\boldsymbol{v}||=h^2||\boldsymbol{\tau}||/||\boldsymbol{v}||$, per cui

$$|\lambda-\mu|\leq h^2\frac{||\boldsymbol{\tau}||}{||\boldsymbol{v}||}. \tag{17}$$

Si osserva che il quoziente $\frac{||\boldsymbol{\tau}||}{||\boldsymbol{v}||}$ per $n\to\infty$ converge a

$$\int_0^\ell 2(v''(x))^2dx/\int_0^\ell v(x)^2dx$$

che ha un valore finito.

Inoltre, poiché gli autovalori di B sono $\beta_i=(n+1)^2(2-2\cos\pi i/(n+1))=i^2\pi^2+O(h^2)$, se $\mu=\beta_j$ allora $1/\min_{i\neq j}|\beta_i-\mu|=1/(\pi^2(2i-1))+O(h^2)$ che non dipende da n. Per cui

$$\frac{||\boldsymbol{w}-\boldsymbol{v}||_2}{||\boldsymbol{v}||_2}\leq h^2\frac{2}{\pi^2(2i-1)}\int_0^\ell 2(v''(x))^2dx/\int_0^\ell v(x)^2dx+O(h^2)$$

6.1 Caso bidimensionale

Nel caso della vibrazione di una membrana vincolata al bordo di un quadrato di lato ℓ l'equazione (13) diventa

$$\frac{1}{h^2}(v_{i-1,j}+v_{i,j-1}-4v_{i,j}+v_{i,j+1}+v_{i+1,j})=-\lambda v_{i,j}+h^2\tau_{i,j},\quad i,j=1,2,\ldots,n, \tag{18}$$

dove $v_{i,j} = v(x_i, y_j)$, $x_i = \ell * i/(n+1)$, $y_j = \ell * j/(n+1)$. Il problema agli autovalori da risolvere è allora

$$-\frac{1}{h^2} A\boldsymbol{w} = \mu \boldsymbol{w}. \tag{19}$$

dove la matrice $-A$ è la matrice tridiagonale a blocchi $n \times n$ con blocchi sopra diagonali e sotto diagonali uguali a $-I$ e blocchi diagonali uguali a

$$\begin{bmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix}.$$

7 Calcolo dei modi di vibrazione

Nelle applicazioni hanno maggior interesse gli autovalori più piccoli ed è importante poter calcolare gli autovettori corrispondenti che approssimano le autofunzioni del problema continuo. Questi autovettori sono chiamati i modi di vibrazione. In questa sezione ci occupiamo di questo calcolo per quanto riguarda il problema bidimensionale (19).

Assumiamo di avere un dominio rettangolare discretizzato con un reticolo di $m \times n$ punti interni, di modo che la matrice in (19) sia $m \times m$ a blocchi con blocchi $n \times n$. Supponiamo che $h = 1/(\max(m, n) + 1)$ e occupiamoci del calcolo degli autovalori di A che alla fine verranno moltiplicati per $1/h^2$. Più precisamente, dato un intero k molto più piccolo di n e m, il nostro scopo è calcolare i più piccoli k autovalori di A. Si osserva che, poiché i metodi numerici disponibili, tipo il metodo delle potenze e le sue varianti, calcolano i k più grandi autovalori di A è necessario fare la seguente manipolazione.

Siano

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{mn}$$

gli autovalori di A. Si osservi che, per il teorema di Gerschgorin risulta $0 < \lambda_1$, $\lambda_{mn} < 8$, per cui la matrice $B = 8I - A$ ha autovalori

$$\mu_{mn} \leq \mu_{mn-1} \leq \cdots \leq \mu_1$$

dove $\mu_i = 8 - \lambda_i$. per cui, applicando il metodo delle potenze o le sue varianti a B si calcolano $\mu_1, \ldots, \mu_k$ e quindi $\lambda_1, \ldots, \lambda_k$.

Si osserva che la matrice B ha la stessa struttura tridiagonale a blocchi di A dove i blocchi sopra e sotto diagonali sono matrici identiche e il blocco diagonale è la matrice tridiagonale con elementi diagonali uguali a 4 e sopra sotto diagonali uguali a 1.

Si osserva ancora che se un vettore $\boldsymbol{v}$ di mn componenti viene organizzato come matrice $m \times n$ $V = (v_{i,j})$, allora il prodotto $\boldsymbol{u} = A\boldsymbol{v}$ è tale che

$$u_{i,j} = \frac{1}{h^2}(v_{i-1,j} + v_{i,j-1} + 4v_{i,j} + v_{i,j+1} + v_{i+1,j}) \tag{20}$$

7.1 Iterazione dei sottospazi (generalizzazione del metodo delle potenze)

In questa sezione assumiamo che il problema da risolvere sia quello di calcolare i k autovalori più grandi di una matrice A di dimensione $n \times n$ reale simmetrica, definita positiva.

Il metodo delle iterazioni dei sottospazi, detto anche delle iterazioni ortogonali, è una generalizzazione del metodo delle potenze in cui, anziché costruire una successione di vettori che converge all'autovettore dominante della matrice, si costruisce una successione di sottospazi, tutti di dimensione k che converge al sottospazio invariante di A generato dai primi k autovettori.

Ricordiamo prima il metodo delle potenze nella forma:

$$\begin{aligned} &\boldsymbol{y}^{(\nu)} = A\boldsymbol{x}^{(\nu)} \\ &\boldsymbol{x}^{(\nu+1)} = \boldsymbol{y}^{(\nu)}/||\boldsymbol{y}^{(\nu)}|| \qquad\qquad \nu = 0, 1, 2, \dots \\ &\sigma_\nu = y_i^{(\nu)}/x_i^{(\nu)}, \quad \text{dove } x_i^{(\nu)} \neq 0 \end{aligned}$$

Si dimostra facilmente che $|\sigma_\nu - \lambda_1| = O(|\lambda_2/\lambda_1|^\nu)$, dove λ_1 è l'autovalore di massimo modulo di A e si assume che $|\lambda_2| < |\lambda_1|$.

Nel caso in cui A sia simmetrica si ottiene una migliore convergenza con la seguente variante

$$\begin{aligned} &\boldsymbol{y}^{(\nu)} = A\boldsymbol{x}^{(\nu)} \\ &\boldsymbol{x}^{(\nu+1)} = \boldsymbol{y}^{(\nu)}/||\boldsymbol{y}^{(\nu)}||_2 \quad \nu = 0, 1, 2, \dots \\ &\sigma_\nu = {\boldsymbol{y}^{(\nu)}}^T \boldsymbol{x}^{(\nu)} \end{aligned}$$

Infatti si può dimostrare che $|\sigma_\nu - \lambda_1| = O(|\lambda_2/\lambda_1|^{2\nu})$.

Il metodo delle iterazioni dei sottospazi viene dato in due versioni che generalizzano, in qualche forma, le due versioni date del metodo delle potenze.

Data la matrice $n \times k$ $X^{(0)}$ con colonne ortogonali, la prima versione del metodo è la seguente

$$\begin{aligned} &Y^{(\nu)} = AX^{(\nu)} \\ &Y^{(\nu)} = Q^{(\nu)}R^{(\nu)} \quad \nu = 0, 1, 2, \dots \\ &X^{(\nu+1)} = Q^{(\nu)} \end{aligned}$$

dove $Y^{(\nu)} = Q^{(\nu)}R^{(\nu)}$ è la fattorizzazione QR della matrice $Y^{(\nu)}$. Si dimostra che per quasi ogni scelta di $X^{(0)}$, se $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > \lambda_{k+1}$ allora

$$|r_{i,i}^{(\nu)} - \lambda_i| \leq \gamma(|\lambda_{i+1}/\lambda_i|^\nu + |\lambda_i/\lambda_{i-1}|^\nu)$$

dove se $i = 1$ il termine $|\lambda_i/\lambda_{i-1}|^\nu$ non compare.

L'inconveniente di questo metodo è che per avere convergenza occorre la stretta separazione di tutti gli autovalori da λ_1 a λ_{k+1}.

Questo inconveniente viene rimosso se si adotta la seconda variante:

$$
\begin{aligned}
&Y^{(\nu)} = AX^{(\nu)} \\
&Y^{(\nu)} = Q^{(\nu)}R^{(\nu)} \\
&B^{(\nu)} = {Q^{(\nu)}}^T AQ^{(\nu)} \qquad \nu = 0, 1, 2, \ldots \\
&B^{(\nu)} = U^{(\nu)}D^{(\nu)}{U^{(\nu)}}^T \\
&X^{(\nu+1)} = Q^{(\nu)}U^{(\nu)}
\end{aligned} \tag{21}
$$

Si osserva che le colonne della matrice $X^{(\nu)}$ generata nelle due varianti generano lo stesso sosttospazio vettoriale, cioè quello generato dalle colonne di $A^\nu X^{(0)}$. Le due basi peò sono differenti, e nel caso della seconda variante si può dimostrare che per quasi ogni scelta di $X^{(0)}$ esiste una costante $\gamma > 0$ tale che

$$|d_{i,i}^{(\nu)} - \lambda_i| \leq \gamma(|\lambda_{k+1}/\lambda_i|^\nu).$$

Dalla relazione precedente segue che la convergenza della seconda variante è migliore e che è garantita dalla condizione più debole $\lambda_i > \lambda_{k+1}$.

7.2 Implementazione

L'implementazione del metodo delle iterazioni dei sottospazi nella forma (21) si realizza agevolmente una volta che abbiamo a disposizione una subroutine per calcolare il prodotto matrice vettore, una subroutine per calcolare la fattorizzazione QR e una subroutine per calcolare gli autovalori e gli autovettori di una matrice reale simmetrica.

Le subroutine per il calcolo della fattorizzazione QR e per il calcolo degli autovalori e autovettori utilizzeranno opportune subroutine della libreria LAPACK. Supponiamo quindi di aver scritto due subroutine del tipo

```
SUBROUTINE QR(m, n, A, Q, R)
! A matrice mxn in input
! Q matrice mxn in output, con colonne ortonormali
! R matrice nxn in output
.......
END SUBROUTINE QR

SUBROUTINE eig(n, A, eigval, eigvect)
! A matrice reale simmetrica nxn in input
! eigval vettore in output di n componenti contenente gli autovalori di A
! eigvect matrice nxn in output che per colonne ha gli autovettori di A
.......
END SUBROUTINE eig
```

che calcolano rispettivamente la fattorizzazione QR e gli autovalori della matrice reale simmetrica A. I listati di questi programmi interfaccia sono riportati in appendice.

Per il calcolo del prodotto matrice vettore $\boldsymbol{u} = A\boldsymbol{v}$ è utile rappresentare il vettore $\boldsymbol{v}$ con una variabile a due indici `V(0:m+1,0:n+1)` in accordo con la

Listing 6: Programma prodotto

```
SUBROUTINE prodotto(m,n,v,u)
! v vettore di input organizzato come matrice
! u=Av vettore di output organizzato come matrice
    IMPLICIT NONE
    INTEGER, PARAMETER :: dp=KIND(0.d0)
    INTEGER :: m,n,i,j
    REAL(dp),DIMENSION(0:m+1,0:n+1) :: u, v
    u=0
    DO i=1,m
        DO j=1,n
            u(i,j)=4*v(i,j)+v(i+1,j)+v(i-1,j)+v(i,j+1)+v(i,j-1)
        END DO
    END DO
END SUBROUTINE prodotto
```

Listing 7: Programma prodotto1

```
SUBROUTINE prodotto1(m,n,v,u)
! v vettore di input organizzato come matrice
! u=Av vettore di output organizzato come matrice
    IMPLICIT NONE
    INTEGER, PARAMETER :: dp=KIND(0.d0)
    INTEGER :: m,n
    REAL(dp),DIMENSION(0:m+1,0:n+1) :: u, v
    u=4.d0*v
    u(1:m,:) = u(1:m,:)+v(0:m-1,:)+v(2:m+1,:)
    u(:,1:n) = u(:,1:n)+v(:,0:n-1)+v(:,2:n+1)
END SUBROUTINE prodotto1
```

natura bidimensionale del problema, dove sono inclusi anche i valori del bordo `V(0,j), V(m+1,j), V(i,0), V(i,n+1)`. In questo modo le formule per calcolare il prodotto $\boldsymbol{u} = A\boldsymbol{v}$ diventano le (20). La subroutine che le implementa è riportata nel listing 6.

Una versione che evita l'uso del ciclo `do` è riportata nel listing 7

La subroutine che esegue un passo del metodo delle iterazioni dei sottospazi può quindi essere facilmente costruita. Prima di descrivere questa subroutine occorre osservare che, in base alle nostre notazioni la matrice X che ha per colonne le approssimazioni dei primi k autovettori sarà rappresentata da una variabile a tre indici `X(0:m+1,0:n+1,1:k)`. Di questa matrice occorrerà calcolare la fattorizzazione QR. Il programma che calcola la fattorizzazione QR, così come lo abbiamo scritto, prende però in input una matrice a due indici. Occorre quindi trasformare i dati contenuti nella variabile a tre indici `X` in una variabile a due indici che denoteremo con `VX` (dove `V` sta per "vettore") in cui la generica

colonna j di `VX` è ottenuta incolonnando una sull'altra le colonne di `X(:,:,j)`. Questa trasformazione è realizzata da una funzione intrinseca del Fortran 90 che si chiama `RESHAPE` nel modo seguente

```
VX=RESHAPE(X(1:m,1:n,:), (/m*n, k/) )
```

Con questo comando, la variabile `VX` di dimensione $mn \times k$ conterrà gli elementi di `X(1:m,1:n,:)` organizzati come matrice $mn \times k$. L'operazione inversa si realizza col comando

```
X(1:m,1:m,:)=RESHAPE(VX, (/m, n, k/) )
```

Si ricorda che la notazione `(/ a, b, c, d /)` denota il vettore di componenti (a, b, c, d).

La subroutine che implementa un passo del metodo di iterazione dei sottospazi è quindi:

Il programma principale che calcola i primi k autovalori e i primi k autovettori è riportato di seguito

Il programma esegue al più 10000 iterazioni e arresta le iterazioni quando la massima differenza fra due approssimazioni consecutive degli autovalori è in valore assoluto più piccola di `eps=1.0e-10`.

Per quanto riguarda l'istruzione di scrittura `WRITE(10+it,*)`, si osserva che poiché non è stato attribuito nessu file all'unita di scrittura `10+it`, l'output verra scritto per default nel file `fort.xx` dove xx è il valore numerico di `10+it`. Per cui alla fine dell'esecuzione ritroveremo i risultati relativi al primo autovettore nel file `fort.11`, quelli del secondo autovettore nel file `fort.12` ecc.

Lanciando il programma con $m = n = 40$ e con $k = 7$ l'iterazione si arresta dopo 3164 iterazioni e si ottengono i seguenti autovalori

```
19.7295528405405
49.2659916707359        49.2659916707195
78.8024305057777
98.3007991729601        98.3007613432024
127.837200283465
```

Si provi a dare una spiegazione del perché il secondo e terzo autovalore, così come il quinto e il sesto sembrano essere uguali. Per questo può essere di aiuto il fatto che che gli autovalori semplici λ_1 e λ_3 hanno autofunzioni dotati di forti simmetrie (vedi figura 1).

Un'altra osservazione è che dividendo i valori precedenti per π^2 is ottengono i valori

```
1.99902164653761
4.99168858939251        4.99168858939085
7.98435553273846
9.95994949223623        9.95995332519206
12.9526164462433
```

Listing 8: Subroutine sottospazi

```
SUBROUTINE sottospazi(m,n,k,x,y,eigenval)
  ! notazioni: le variabili il cui nome inizia per "v" sono
  ! le espressioni "vettorizzate" delle variabili di
  ! dimensione mxn
  ! x: variabile di input
  ! y: variabile di output

  IMPLICIT NONE
  INTEGER, PARAMETER :: dp=KIND(0.d0)
  INTEGER :: m,n,i,j,k
  REAL(dp), DIMENSION(0:m+1,0:n+1,1:k) :: x,y
  REAL(dp), DIMENSION(k) :: eigenval
  REAL(dp), DIMENSION(k,k) :: eigenvec,b
  REAL(dp), DIMENSION(0:m+1,0:n+1,1:k) :: aq, q
  REAL(dp), DIMENSION(1:k,1:k) :: r
  REAL(dp), DIMENSION(n*m,k) :: vq, vy, vaq

  ! prodotto y=Ax
  DO i = 1, k
     CALL prodotto(m, n, x(:,:,i), y(:,:,i))
  END DO

  ! fattorizzazione QR di y
  vy = RESHAPE(y(1:n,1:m,1:k), (/n*m, k/))
  CALL QR(m*n, k, vy, vq, r)
  q = 0
  q(1:m,1:n,1:k) = RESHAPE(vq, (/m, n, k/))

  ! proietto A su Q
  DO i = 1, k
     CALL prodotto(m, n, q(:,:,i), aq(:,:,i))
  END DO
  vaq = RESHAPE(aq(1:m,1:n,1:k), (/n*m, k/))
  b = MATMUL(TRANSPOSE(vq), vaq)

  ! calcolo autovalori/vettori di b
  CALL autovalori(b, k, eigenvec, eigenval)

  ! aggiorno
  vy = MATMUL(vq, eigenvec)
  y = 0
  y(1:m,1:n,1:k) = RESHAPE(vy, (/m,n,k/))
END SUBROUTINE sottospazi
```

Listing 9: Programma main

```
PROGRAM main
  IMPLICIT NONE
  INTEGER, PARAMETER :: dp=KIND(0.d0)
  INTEGER :: n, m, k, i, j, it, h
  REAL(dp) :: rnd, eps = 1.E-10, err
  REAL(dp), DIMENSION(:,:,:), ALLOCATABLE :: x, y
  REAL(dp), DIMENSION(:), ALLOCATABLE :: eig, eig1
  REAL(dp), DIMENSION(:,:), ALLOCATABLE :: r
  REAL(dp), DIMENSION(:,:), ALLOCATABLE :: vx,vq
  WRITE(*,*) "assegna le dimensioni m,n della membrana"
  READ(*,*) m, n
  WRITE(*,*) "assegna il numero k di autovalori da calcolare"
  READ(*,*) k

  ALLOCATE(x(0:m+1,0:n+1,k), y(0:m+1,0:n+1,k))
  ALLOCATE(eig(k), eig1(k), r(k,k), vx(m*n,k), vq(m*n,k))
! genero il punto iniziale a caso
  call random_number(x)
  x = x-0.5d0

! ortogonalizzo il punto iniziale
  vx = RESHAPE(x(1:n,1:m,1:k), (/n*m, k/))
  CALL QR(m*n, k, vx, vq, r)
  x = 0
  x(1:m,1:n,1:k) = RESHAPE(vq, (/m, n, k/))

! Eseguo AL PIU' 10000 iterazioni
  eig1 = 0.d0
  DO it=1,10000
     CALL sottospazi(m, n, k, x, y, eig)
     x=y
     err = MAXVAL(ABS(eig-eig1))
     WRITE(*,*)"it=",it,"eig=",err
     IF(err<eps) THEN
         EXIT
     END IF
     eig1 = eig
  END DO

! salvo i risultati
  OPEN(UNIT = 2, FILE='autovalori.txt')
  WRITE(2,*) (8.d0 - eig)*(1+max(m,n))**2 ! ho normalizzato
  DO h=1,k
     DO i=0,m+1
        WRITE(10+h,*)x(:,i,h)
     END DO
  END DO
END PROGRAM main
```

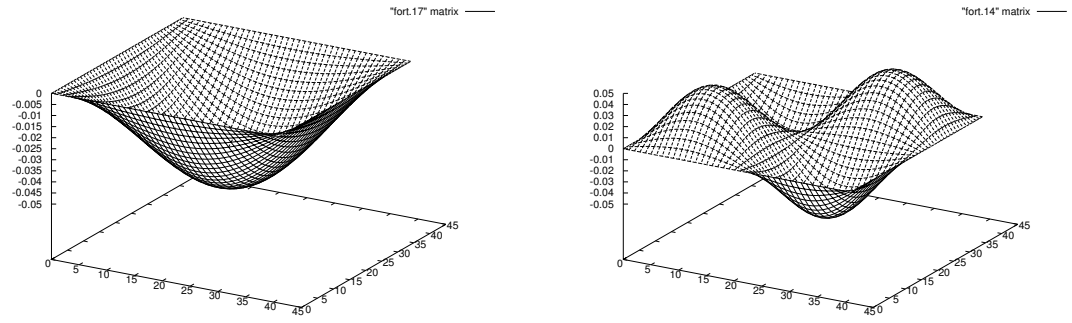


Figura 5: Modi 1 e 4

mentre con $m = n = 200$ si ottengono i valori

```
1.99995942022081
4.99966733106443        4.99966393771013
7.99936227650493
9.99841962236914        9.99833435843634
12.9991609109434
```

che sembrano approssimare numeri interi. Sapreste dare una spiegazione?

Per fare il calcolo con $n = m = 200$ sono state necessarie 16104 iterazioni. Lanciando il programma con valori più alti di n si scopre una grande lentezza nella convergenza. Sapreste spiegare il perché?

I corrispondenti modi di vibrazione (autovettori) sono riportati nelle figure 1,2, 3 e 4. Essi sono stati tracciati con `gnuplot` usando il comando

```
splot ‘‘fort.xx’’ matrix with lines
```

per cancellare le linee nascoste abbiamo dato in precedenza il comando di gnuplot

```
set hidden
```

.

Per creare un file di tipo “eps” (encapsulated postscript) con il disegno del grafico abbiamo dato i comandi di gnuplot

```
set terminal postscript
set output "nomefile.eps"
```

Lanciando il programma con varie dimensioni accade che le forme di alcune autofunzioni non si mantengono inalterate passando da una dimensione ad un’altra. Sapreste spiegare perché?

Un esercizio interessante è quello di mettere in evidenza le linee modali di ogni autofunzione, cioè quelle linee formate dai punti nulli. Nel caso discreto

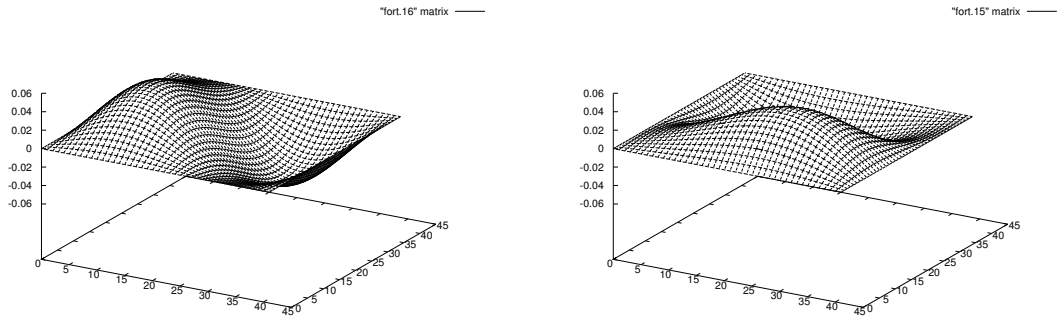


Figura 6: Modi 2 e 3

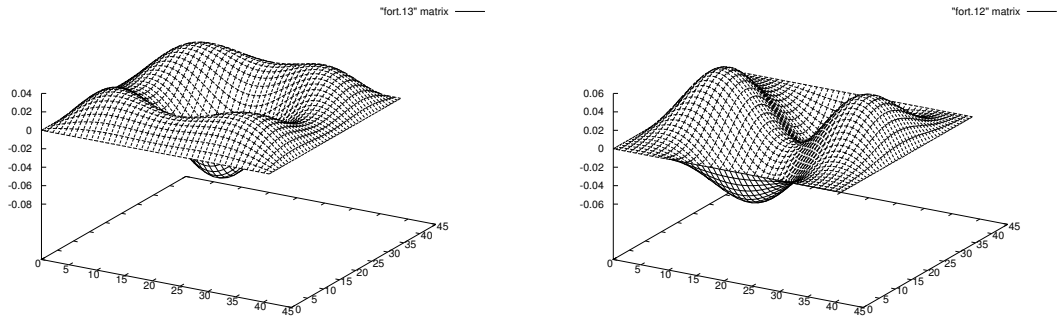


Figura 7: Modi 5 e 6

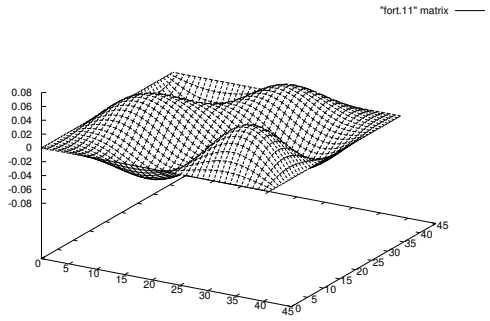


Figura 8: Modo 7

occorre individuare le coppie di punti $(i,j),(i+1,j)$ oppure $(i,j),(i,j+1)$ in cui il valore dell'autovettore cambia segno.

Per questo si può scrivere un programmino (si veda il listing 10) che costruisce una matrice `F(0:m+1,0:n+1)` di dimensione $(2m+3)\times(2n+3)$ tale che $F(i,j)=0$ se $x(i,j)=0$ oppure $x(i,j)x(i+1,j)<0$ oppure $x(i,j)x(i,j+1)<0$, e vale 255 nei punti rimanenti.

Scrivendo in un file `lineemodali.pgm` i valori di questa matrice per righe dopo aver inserito nelle prime tre righe del file i caratteri

```
P2
n+2 m+2
maxval
```

dove `n` e `m` sono i valori di n e di m, e `maxval`$= 255$ è la massima intensità luminosa possibile, è possibile vedere il disegno delle righe col comando `display lineemodali.pgm`.

In questa visualizzazione ci si basa sulla possibilità di descrivere una immagine digitale in scala di grigi. Una immagine digitale costituita da un insieme di $m\times n$ punti luminosi detti pixel (contrazione di picture-element) può essere descritta mediante una matrice di $m\times n$ numeri che determinano il grado di luminosità di ciascun pixel. Nel formato PGM (portable Gray Map) questa matrice è costituita da numeri interi compresi tra 0 e `maxval`< 65536, con la convenzione che 0 corrisponde a nero mentre `maxval` corrisponde a bianco e i valori intermedi corrispondono a valori di grigio. Questa matrice viene memorizzata in un file per righe, partendo dall'angolo in alto a sinistra. La lista degli elementi di questa matrice è preceduta da una parte iniziale, detta header, che come si è già detto contiene informazioni sul file. Diamo la descrizione dello header nel caso più generale di formato PNM (Portable aNy Map) che include lo standard PGM e PPM (portable Pix Map) e PBM (portable Bit Map) che assieme al formato PAM costituiscono i *netpbm formats*. Per maggiori informazioni si veda il sito `netpbm.sourceforge.net/doc`.

Sulla prima riga del file sta il "numero magico" che determina la tipologia del file

- `P2` file in scala di grigi, formato ASCII
- `P5` file in scala di grigi, formato RAW
- `P3` file a colori, standard RGB, formato ASCII
- `P6` file a colori, standard RGB, formato RAW

dove per formato ASCII si intende che la parte del file contenente i singoli numeri che danno la luminosità dei pixel sono scritti in formato ASCII; ad esempio il numero 123 è rappresentato mediante tre cifre e separato dal successivo da uno o più "ritorni a capo" o da uno o più spazi. Nel formato RAW la parte del file contenente i valori numerici dei pixel è rappresentata in modo binario, cioè ogni singolo numero compreso tra 0 e 255 è memorizzato col singolo byte corrispondente. Con la codifica `P2` e `P5` l'immagine è intesa in bianco-nero (scala di grigi) dove 0 corrisponde a nero e `maxval` a bianco. Con la codifica `P3` e `P6` l'immagine è a colori e ciascun pixel è rappresentato da una terna di interi tra 0 e `maxval` che indicano nell'ordine la quantità di rosso, verde e blu del pixel.

Per questo motivo si parla di standard RGB (Red Green Blue). Nella codifica ASCII le componenti della terna possono essere separate da uno o più spazi o ritorni a capo. In questo modo ogni numero può essere messo su una riga diversa, oppure è possibile mettere ciascuna terna su ogni riga o più terne sulla stessa riga. Su una riga non possono stare più di 70 caratteri.

Successivamente al numero magico, nello header vanno messe le dimensioni dell'immagine, prima il numero di colonne poi il numero di righe, infine va messo l'intero `maxval`, solitamente 255, che dà il valore della massima intensità luminosa.

Nello header possono stare commenti di qualsiasi tipo che vanno preceduti dal carattere #.

7.3 Altri domini

Per poter implementare il metodo dei sottospazi relativamente ad altri domini si può procedere nel modo seguente. Supponiamo di inserire il nostro dominio dentro un rettangolo che lo include. Introduciamo un reticolo di punti sul rettangolo e per definire il dominio e la sua frontiera costruiamo una matrice che chiamiamo `Mask` in cui l'elemento di posto (i, j) vale 1 se il punto è interno al dominio e vale 0 altrimenti. Ad esempio, la matrice

```
      00000000000000000000
      01111111111111111110
      00111111111111111100
      00011111111111111000
      00001111111111110000
mask= 00000111111111100000
      00000011111111000000
      00000001111110000000
      00000000111100000000
      00000000011000000000
      00000000010000000000
      00000000000000000000
```

definisce un dominio triangolare. A questo punto, poiché i valori dei punti esterni al dominio in ogni autovettore sono forzati a zero, il programma per il calcolo del prodotto matrice vettore andrà modificato con l'istruzione

Listing 10: Programma linee_modali

```
PROGRAM linee_modali
  IMPLICIT NONE
  CHARACTER(len=8) :: nome
  INTEGER :: m, n, i, j
  REAL(KIND(0.d0)), ALLOCATABLE :: a(:,:)
  INTEGER, ALLOCATABLE :: F(:,:)

  WRITE(*,*)"assegna le dimensioni m e n"
  READ(*,*)m, n
  ALLOCATE(a(0:m+1,0:n+1), F(0:m+1,0:n+1))
  WRITE(*,*)"assegna il nome del file"
  READ(*,*)nome
  OPEN(unit=10, file=nome)
  DO i=0,m
     READ(10,*) a(i,:)
  ENDDO
  F=255
  F(0,:)=0;F(:,0)=0;F(m+1,:)=0;F(:,n+1)=0
  DO i=1,m
     DO j=1,n
        IF( a(i,j)*a(i+1,j)< 0 .OR. a(i,j)*a(i,j+1)<0) THEN
           F(i,j)=0
        ENDIF
     ENDDO
  ENDDO
  OPEN(file='linee_modali.pgm', unit=20)
  WRITE(20,'(A)')'P2'
  WRITE(20,*)n+2,m+2
  WRITE(20,*)255
  DO i=0,m+1
     DO j=0,n+1
        WRITE(20,*)F(i,j)
     ENDDO
  ENDDO
END PROGRAM linee_modali
```

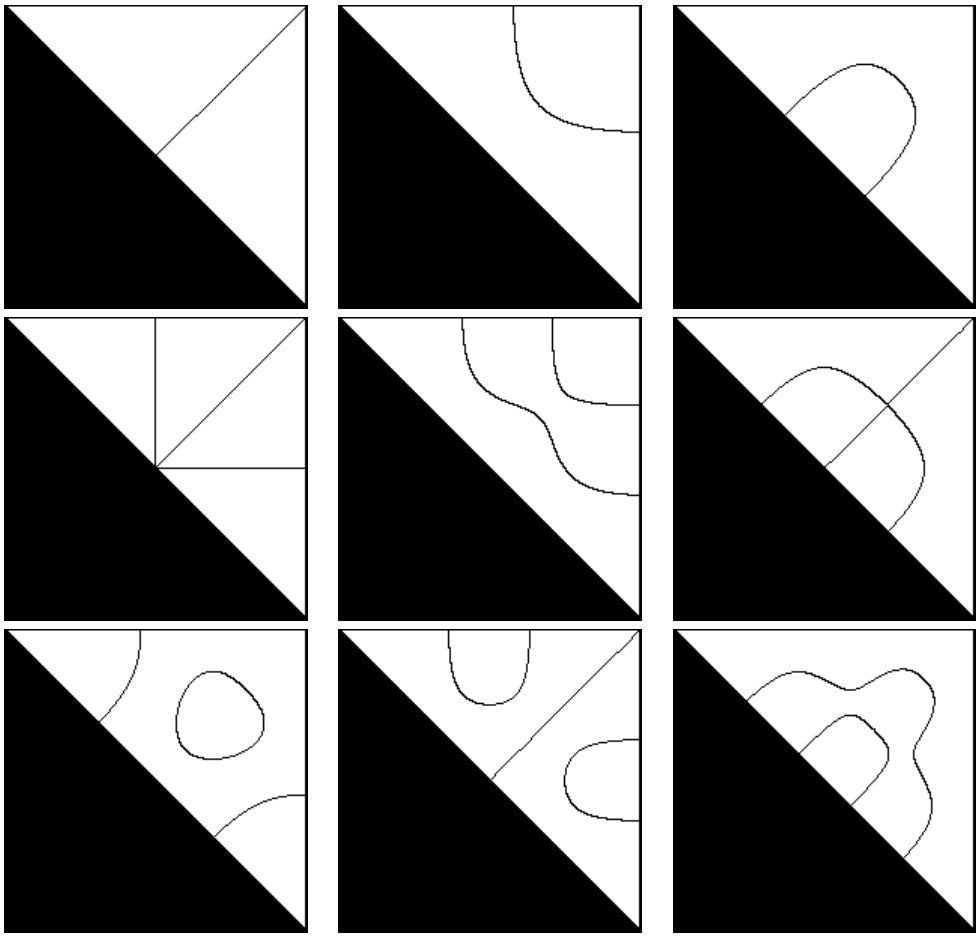


Figura 9: Prime 9 linee modali di un dominio triangolare

```
IF(mask(i,j) /= 0) THEN
v(i,j) = 4*u(i,j) + u(i+1,j) + u(i-1,j) + u(i,j+1) + u(i,j-1)
END IF
```

Nella figura 7.3 si riportano le prime 9 linee modali di un dominio dato da un triangolo rettangolo isoscele. Le linee sono state calcolate ponendo $n = 200$ e applicando 1000 passi del metodo di Lanczos descritto nel prossimo paragrafo.

8 Il metodo di Lanczos

Un metodo poco costoso, adatto per calcolare efficientemente gli autovalori estremi dello spettro è il metodo di Lanczos. Esso genera una successione di matrici $k \times k$ tridiagonali T_k i cui autovalori più grandi e più piccoli convergono

rapidamente agli autovalori più grandi e più piccoli di A. L'idea è quella di costruire una matrice ortogonale Q che tridiagonalizzi la matrice A. Cioè sia tale che

$$AQ = QT$$

dove T è la matrice tridiagonale simmetrica con elementi diagonali α_i, $i = 1, \ldots, n$ e sopra diagonali β_i, $i = 1, 2, \ldots, n-1$. Scrivendo la relazione di sopra in funzione delle colonne $\boldsymbol{q}_i$ di Q si ottiene

$$\begin{aligned} A\boldsymbol{q}_1 &= \alpha_1 \boldsymbol{q}_1 + \beta_1 \boldsymbol{q}_2 \\ A\boldsymbol{q}_i &= \beta_{i-1}\boldsymbol{q}_{i-1} + \alpha_i \boldsymbol{q}_i + \beta_i \boldsymbol{q}_{i+1}, \quad i = 2, \ldots, n-1 \\ A\boldsymbol{q}_n &= \beta_{n-1}\boldsymbol{q}_{n-1} + \alpha_n \boldsymbol{q}_n \end{aligned}$$

Il calcolo dei vettori $\boldsymbol{q}_i$ e degli scalari α_i, β_i procede sfruttando l'ortogonalità dei $\boldsymbol{q}_i$ con le seguenti formule

$$\begin{aligned} &\alpha_1 = \boldsymbol{q}_1^T A \boldsymbol{q}_1, \quad \beta_1 = ||(A - \alpha_1 I)\boldsymbol{q}_1||_2, \quad \boldsymbol{q}_2 = \tfrac{1}{\beta_1}(A - \alpha_1 I)\boldsymbol{q}_1 \\ &\alpha_i = \boldsymbol{q}_i^T A \boldsymbol{q}_i, \quad \boldsymbol{y}_i = (A - \alpha_i I)\boldsymbol{q}_i - \beta_{i-1}\boldsymbol{q}_{i-1}, \quad \beta_i = ||\boldsymbol{y}_i||_2, \quad \boldsymbol{q}_{i+1} = \tfrac{1}{\beta_i}\boldsymbol{y}_i \end{aligned}$$

assumendo di partire con un vettore $\boldsymbol{q}_1$ tale che $||\boldsymbol{q}_1||_2 = 1$. Il metodo presenta problemi di stabilità numerica nel caso in cui si presenta cancellazione (β_i piccolo) nel calcolo di β_i e di $\boldsymbol{q}_{i+1}$.

Per dare una implementazione del metodo è conveniente utilizzare una subroutine di LAPACK che calcola gli autovalori di una matrice tridiagonale reale e simmetrica, ad esempio `dsetqr` basata sull'iterazione QR. Un interfaccia con questa subroutine di LAPACK è riportata nel listing 11.

Una implementazione del metodo di Lanczos è riportata nei listings 12 (programma principale) e 13 (metodo di Lanczos)

Nel programma 12 vengono fissate le dimensioni del reticolo e viene chiesto all'utente di assegnare il numero di iterazioni del metodo di Lanczos. Il programma poi invoca la subroutine `lanczos` che calcola autovalori e autovettori selezionando quelli che hanno raggiunto la precisione di `1.e-9`. Il programma infine salva gli autovettori (modi di vibrazione) nei file `fort.101`, `fort.102`,..., corrispondenti agli autovalori calcolati.

Per valutare l'errore a posteriori si usa il seguente risultato

Teorema 3 *Sia A una matrice normale $n \times n$, $v \in \mathbb{C}^n$ un vettore non nullo e $\sigma \in \mathbb{C}$. Esiste un autovalore λ di A tale che $|\sigma - \lambda| \leq \|(A - \sigma I)v\|_2 / \|v\|_2$. Se A è invertibile esiste un autovalore λ di A tale che $|(\sigma - \lambda)/\lambda| \leq \|(\sigma A^{-1} - I)v\|_2 / \|v\|_2$.*

Dim. Dimostriamo più in generale che se $f(x)$ è una funzione definita nello spettro di A allora esiste un autovalore λ di A tale che $|f(\lambda)| \leq \|f(A)v\|_2 / \|v\|_2$. Per funzione di matrice $f(A)$ si intende la matrice $f(A) = Qf(D)Q^H$ dove $A = QDQ^H$. La tesi poi segue scegliendo $f(x) = x - \sigma$ e $f(x) = (x - \sigma)/x$. Poiché A è normale esiste Q unitaria tale che $D = Q^H A Q$ è diagonale. Allora vale $f(A) = Qf(D)Q^H$ per cui $\|f(A)v\|_2 = \|f(D)Q^H v\|_2$. Inoltre, poiché

Listing 11: Subroutine di interfaccia con la subroutine `dsteqr` di Lapack per il calcolo degli autovalori e autovettori di una matrice tridiagonale simmetrica

```
subroutine autovalori_tridiag(n, a, b, lambda, v)
! subroutine di interfaccia per il calcolo degli aurovalori e autovettori
! di una matrice tridiagonale simmetrica reale usando dsteqr di lapack
! n: dimensione
! a, b: elementi diagonali e sopradiagonali
! lambda: autovalori
! v: autovettori

  implicit none
  integer, parameter :: dp=kind(0.d0)
  character :: compz
  integer :: n, ldz, info
  real(dp) :: a(n), d(n), b(n-1), e(n-1), v(n,n), work(2*n-2), lambda(n)
  compz='I'
  ldz=n
  d=a
  e=b
  call dsteqr(compz, n, d, e, v, ldz, work, info)
  lambda=d
end subroutine autovalori_tridiag
```

Listing 12: Programma per il calcolo dei modi di vibrazione col metodo di Lanczos

```
PROGRAM autovalori_lanczos
  ! Calcola i modi di vibrazione di una membrana a supporto rettangolare
  ! col metodo di Lanczos
  IMPLICIT NONE
  INTEGER, PARAMETER :: dp=KIND(0.d0)
  INTEGER :: m, n, k, i, h
  REAL(dp), ALLOCATABLE :: a(:), b(:), eig(:), v(:,:,:)
  !
  m=20;n=20;
  WRITE(*,*)"assegna il numero di iterazioni del metodo di Lanczos"
  READ(*,*)k
  ALLOCATE(a(k),b(k),eig(k),v(0:m+1,0:n+1,k))
  CALL lanczos(m,n,k,a,b,eig,v)
  DO h=1,k
     DO i=0,m+1
        WRITE(100+h,*) v(i,:,h)
     END DO
  ENDDO
END PROGRAM autovalori_lanczos
```

Listing 13: Subroutine che implementa il metodo di Lanczos

```
SUBROUTINE lanczos(m, n, maxit, a, b, eig, u)
  ! implementa il metodo di Lanczos per calcolare autovalori e
      autovettori
  ! di una matrice reale simmetrica assegnata attraverso la subroutine
  ! "prodotto" che svolge il prodotto matrice vettore
  IMPLICIT NONE
  INTEGER, PARAMETER :: dp=KIND(0.d0)
  INTEGER :: m, n, maxit
  REAL(dp) :: a(maxit), b(maxit)
  !
  INTEGER :: k, cont
  REAL(dp), DIMENSION(0:m+1,0:n+1) :: q1, q2, q3, y, aq
  REAL(dp) :: u(0:m+1,0:n+1,maxit), v(maxit,maxit), eig(maxit)
  REAL(dp) :: rr((n+2)*(m+2),maxit), err
  q1=0; q2=0; q3=0
  aq=0
  y=0
  u=0
  v=0

  CALL RANDOM_NUMBER(q1(1:m,1:n))
  q1(1:m,1:n)=q1(1:m,1:n)-0.5d0
  q1=q1/SQRT(SUM(q1*q1))
  CALL prodotto(m, n, q1, aq)
  a(1)=SUM(aq*q1)
  y(1:m,1:n)=aq(1:m,1:n)-a(1)*q1(1:m,1:n)
  b(1)=SQRT(SUM(y*y))
  q2=y/b(1)
  u(:,:,1)=q1
  DO k=2,maxit
     CALL prodotto(m, n, q2, aq)
     u(:,:,k)=q2
     a(k)=SUM(q2*aq)
     y(1:m,1:n)=aq(1:m,1:n)-a(k)*q2(1:m,1:n)-b(k-1)*q1(1:m,1:n)
     b(k)=SQRT(SUM(y*y))
     q3=y/b(k)
     q1=q2
     q2=q3
  ENDDO
  ! calcolo autovalori e autovettori
  CALL autovalori_tridiag(maxit, a, b, eig, v)
  rr=RESHAPE(u,(/(m+2)*(n+2),maxit /))
  u=RESHAPE(MATMUL(rr,v),(/m+2,n+2,maxit/))
  ! stimo l'errore
  cont=0
  DO k=1,maxit
     CALL prodotto(m,n,u(:,:,k),aq);
     aq=aq-eig(k)*u(:,:,k)
     err=SQRT(SUM(aq*aq))/SQRT(SUM(u(:,:,k)**2))
     IF (err<1.d-8) THEN
        cont=cont+1
        u(:,:,cont)=u(:,:,k)
        WRITE(*,*)cont,err,eig(k)
     ENDIF
  END DO
  maxit=cont
END SUBROUTINE lanczos
```

$\|Q^H v\|_2 = \|v\|_2$ ne segue che $\|f(A)v\|_2/\|v\|_2 = \|f(D)w\|_2/\|w\|_2$, con $w = Q^H v$. Quindi, $\|f(A)v\|_2/\|v\|_2 = (\sum_{i=1}^n |f(\lambda_i)^2 w_i^2|)^{1/2}/(\sum_{i=1}^n |w_i|^2)^{1/2} \geq \min |f(\lambda_i)|$. □

Se σ è una approssimazione di un autovalore e v è una approssimazione dell'autovettore corrispondente allora la quantità $\|(A-\sigma I)v\|_2$, presumibilmente piccola, fornisce una stima a posteriori dell'errore di approssimazione. Questo risultato viene applicato dentro la subroutine `lanczos` per selezionare quegli autovalori che hanno raggiunto la precisione voluta.

Riferimenti bibliografici

Peter Arbenz, Daniel Kressner, "Lecture Notes on Solving Large Scale Eigenvalue Problems", `http://people.inf.ethz.ch/arbenz/ewp/Lnotes/`

D. Bini, M. Capovani, O. Menchi, "Metodi Numerici per l'Algebra Lineare", Zanichelli, Bologna.

B. N. Parlett, The Symmetric Eigenvalue Problem, Prentice Hall, Englewood Cliffs, NJ, 1980. (Republished by SIAM, Philadelphia, 1998.).

Appendice

Si riportano alcuni programmi di interfaccia per l'uso di subroutine LAPACK.

Il programma riportato nel listing 14 funge da interfaccia per l'uso della subroutine di LAPACK `DGEQRF`. Il programma è stato scritto dallo studente S. Brugiapaglia.

Il programma riportato nel listing 15 azzera gli elementi nella parte triangolare inferiore di una matrice.

Il programma riportato nel listing 16 costruisce esplicitamente gli elementi della matrice Q conoscendo le trasformazioni di Givens che compongono Q.

La subroutine riportata nel listing 17 funge da interfaccia per l'uso della subroutine di LAPACK `DSYEV` per il calcolo di autovalori e autovettori di una matrice reale. Il programma è stato scritto dallo studente L. Robol.

Listing 14: Subroutine QR

```
subroutine QR(m, n, A, Q, R)
  ! Di S.Brugiapaglia 2009
  !Subroutine di interfaccia con Lapack che opera la fattorizzazione
  !A=QR di una matrice A di dimensione M x N, dove M>=N
  !
  !Input: A è la matrice da fattorizzare, M, N il numero di righe e
  ! colonne
  !
  !Output: Q è il fattore ortogonale MxN,
  ! R la triangolare superiore NxN

  IMPLICIT NONE
  integer, parameter :: dp=kind(0.d0)
  real(dp), dimension(:), allocatable :: TAU
  real(dp), dimension(m,n) :: AUX, A
  real(dp),dimension(n,n) :: R
  real(dp), dimension(m,n) :: Q
  real(dp), dimension(n) :: WORK
  integer :: info, dim_tau, i, j, m, n, min_mn

  !in DGEQRF A verrebbe modificata, quindi la copio in AUX
  AUX=A

  min_mn=min(m,n)
  allocate (TAU(min_mn))

  call DGEQRF( m, n, AUX, m, TAU, WORK, n, info )
  if (info==0) then
     !la fattorizzazione è avvenuta con successo
     call elabora_Q(Q,m,n,AUX,TAU,min_mn)
     call elabora_R(R,m,n,AUX)
  else
     write(*,*) "Errore negli argomenti di input."
  end if
end subroutine QR
```

Listing 15: Subroutine elabora_R

```
subroutine elabora_R(R,m,n,A)
  implicit none
  integer, parameter :: dp=kind(0.d0)
  real(dp), dimension(m,n) :: A
  real(dp),dimension(n,n) :: R
  integer :: i, j, m, n
  !copio A in R, poi metterò gli zeri opportunamente
  R=A(1:n,1:n);
  do i=2,n
     do j=1,i-1
        r(i,j)=0
     end do
  end do
end subroutine elabora_R
```

Listing 16: Subroutine elabora_Q

```
subroutine elabora_Q(Q,m,n,A,TAU,min_mn)
! Costruisce esplicitamente Q date le trasformazioni di Householder
! fornite dalla subroutine DGEQRF di Lapack
  implicit none
  integer, parameter :: dp=kind(0.d0)
  real(dp), dimension(m) :: v
  real(dp), dimension(n) :: w
  real(dp), dimension(min_mn) :: TAU
  real(dp), dimension(m,n) :: A
  real(dp), dimension(m,n) :: Q
! real(dp), dimension(m,m) :: H
  integer :: i, j, k, m, n, min_mn

  !creo v(n)
  v=0
  v(n)=1
  v(n+1:m)=A(n+1:m,n)
! do i=n+1,m
! v(i)=A(i,n)
! end do
  !Copio H(1) in Q
  do i=1,m
     do j=1,n
        Q(i,j)=-TAU(n)*v(i)*v(j)
     end do
  end do
  !modifico gli el sulla diagonale del tipo 1-tau(1)*v(i)*v(i)
  do i=1,n
     Q(i,i)=1+Q(i,i)
  end do
  !creo Q = H(1)*...*H(min_mn)
  ! mediante Q=H(k)*Q
  do k=n-1,1,-1
     !creo v(k)
     v=0
     v(k)=1
     do i=k+1,m
        v(i)=A(i,k)
     end do
     w=matmul(transpose(q),v)
     do i=k,n
        q(:,i)=q(:,i)-v*Tau(k)*w(i)
     end do
  end do
end subroutine elabora_Q
```

Listing 17: Subroutine eig

```
SUBROUTINE eig(A,n,eigenvec,eigenval)
! Programma di Leonardo Robol
! Novembre 2009
! SUBROUTINE per calcolare autovettori ed autovalori di una matrice
! reale simmetrica
!
! Prende in input la matrice A reale simmetrica e la sua dimensione n,
! e restituisce la matrice eigenvec con gli autovettori e il vettore
! eigenval con gli autovalori
!

  IMPLICIT NONE;

  ! Dichiarazioni
  INTEGER, PARAMETER :: dp = KIND(0.d0);
  INTEGER, INTENT(IN) :: n;
  REAL(dp), DIMENSION(n,n) :: eigenvec, A;
  REAL(dp), DIMENSION(n) :: eigenval;
  INTEGER :: LWORK;
  REAL(dp), DIMENSION(:), ALLOCATABLE :: WORK;
  INTEGER :: INFO;

  ! Creiamo lo spazio di lavoro
  LWORK = n*n + 3;
  ALLOCATE(WORK(LWORK));

  ! Preserviamo la matrice A
  eigenvec = A;

  ! Chiamiamo la funzione LAPACK
  CALL DSYEV( 'V', 'U', n, eigenvec, n, eigenval, WORK, LWORK, INFO )

END SUBROUTINE
```