Relazione di Laboratorio computazionale

Enrico Polesel

3 novembre 2014

1 Problema diretto

Il problema diretto viene risolto con la funzione

```
function [Lambda,Y] = directSLP(q,L,N)
```

dove vogliamo risolvere il problema diretto con potenziale q sull'intervallo $[-L, L]$ discretizzando su una griglia di $N+1$ punti. La funzione ritorna il vettore degli autovalori Λ e matrice Y tale che la colonna i-esima rappresenta l'autofunzione corrispondente all'autovalore i-esimo calcolata nei punti griglia.

Il primo passo è riscalare il potenziale dall'intervallo $[-L, L]$ all'intervallo $[0, 2\pi]$ e poi calcolarlo su una griglia di $N+1$ punti equispaziati sull'intervallo.

```
v = @(xi) (L/pi)^2 * feval(q,(L/pi)*(xi - pi )) ;
V = feval(v, linspace (0, 2*pi, N+1) ) ;
```

L'ultimo ingrediente che ci manca per poter risolvere il problema è la matrice $D^{(2)}$ che, dati i valori di una funzione y nei punti griglia, ne resistuisce i valori di $y^{(2)}$ nei punti griglia. Questa matrice viene calcolata nella funzione

```
function D = directSLP_inner2(N)
```

usando le formule (11) e (12) dell'articolo.

A questo punto va scritta (e risolta) la matrice che discretizza il problema continuo agl'autovalori. Questo viene delegato[1] alla funzione

```
function [ E, Y] = directSLP_inner1(D,V)
```

che, data la matrice di differenziazione D (che nell'articolo viene chiamata $D^{(2)}$) e il potenziale calcolato nei punti V, ritorna gli autovalori E (**non** riscalati a $[-L, L]$) e il valore delle autofunzioni nei punti griglia Y.

La funzione directSLP_inner1, dopo aver escluso i valori ai bordi $0, 2\pi$ (perché usiamo condizioni al contorno di Dirichlet) costruisce la matrice M e ne calcola autovalori e autovettori utilizzando la funzione **eig** di MATLAB.

L'ultima operazione che rimane da fare su directSLP è riscalare autovalori.

[1] la scelta di dividere questo codice deriva dal fatto che viene riutilizzato nella funzione per risolvere il problema inverso

2 Problema inverso

Per risolvere il problema inverso aggiungiamo l'ipotesi che il potenziale, definito su $[-L, L]$ deve essere simmetrico rispetto a 0.

Prendiamo in ingresso M autovalori e ci proponiamo di risolvere il problema trovando il valore del potenziale su una griglia di $N+1 = 2M+2$ punti esclusi i punti estremi (per motivi che saranno chiariti in seguito).

Di nuovo rimappiamo il problema su $[0, 2\pi]$, fissiamo la griglia $\xi_0, \xi_1, ..., \xi_N$ con $\xi_i = \frac{2\pi}{N} i$ e chiamiamo $v(\xi)$ il potenziale rimappato.

Riscrivendo il problema diretto

$$D^{(2)} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_M \\ y_{M+1} \\ \vdots \\ y_{2M} \\ y_{2M+1} \end{pmatrix} + \begin{pmatrix} v(\xi_0) & & & & & & & & \\ & v(\xi_1) & & & & & & & \\ & & \ddots & & & & & & \\ & & & v(\xi_M) & & & & & \\ & & & & v(\xi_{M+1}) & & & & \\ & & & & & \ddots & & & \\ & & & & & & v(\xi_{2M}) & & \\ & & & & & & & v(\xi_{2M+1}) & \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_M \\ y_{M+1} \\ \vdots \\ y_{2M} \\ y_{2M+1} \end{pmatrix} = \lambda y$$

osseviamo che, essendo $y_0 = y_{2M+1} = 0$ (perché abbiamo fissato condizioni al contorno di Dirichlet), nella relazione non compaiono i valori di $v(\xi_0)$ e $v(\xi_{2M+1})$. Per simmetria abbiamo $v(\xi_i) = v(\xi_{2M+1-i})$, quindi il vettore incognito del nostro probelma è

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix} = \begin{pmatrix} v(\xi_1) \\ v(\xi_2) \\ \vdots \\ v(\xi_M) \end{pmatrix}$$

Per passare da questo vettore ridotto al vettore "completo" si può usare la matrice extender così definita

```
extender = [
              zeros(1,M)  ;
              eye(M)  ;
              flipud(eye(M))  ;
              zeros(1,M)
           ];
```

$$\text{extender} = \begin{pmatrix} 0 & & 0 \\ 1 & & 0 \\ & \ddots & \\ 0 & & 1 \\ 0 & & 1 \\ & \iddots & \\ 1 & & 0 \\ 0 & & 0 \end{pmatrix}$$

Vediamo ora come è stata implementata la funzione per risolvere il probelma inverso.

La funzione per calcolare il problema inverso è:

```
function q=inverseSLP(L,Lambda,Kmax,tol,v0)
```

dove L è il limite dell'intervallo, Lambda è il vettore dei primi M autovalori (ordinati in ordine decrescente), Kmax è il massimo numero di iterazioni da eseguire, tol è la tolleranza entro la quale si considera che la soluzione ha raggiunto un punto fisso. v0 è un parametro opzionale e rappresenta il potenziale iniziale da cui partire.

2.1 Implementazione: preliminari

Dopo aver incluso le librerie (regu e dmsuite), la funzione imposta (se non è stato passato come argomento) il potenziale iniziale a 0, riscala il problema da $[-L, L]$ trasformando gli autovalori e calcola i punti griglia.

Vengono precalcolate le seguenti matrici:

- La matrice di differenziazione $D^{(2)}$ che viene salvata in D
- La matrice extender che estende il potenziale per simmetria
- La matrice di penalizzazione PenMat

Le prime due matrici sono semplici da costruire, per la prima basta usare la funzione directSLP_inner2 e per la seconda la costruzione è già stata illustrata precedentemente.

Per la terza si utilizza la funzione hermite_differentation_matrix.

Prima di entrare nel ciclo vengongo inizializzate le variabili fra cui il vettore del potenziale candidato vk e la variabile di conteggio cicli.

2.2 Implementazione: hermite_differentation_matrix

La funzione

```
function D2 = hermite_differentation_matrix(N)
```

viene utilizzata per calcolare la matrice di differenziazione di Hermite del second'ordine.

Per prima cosa vengono calcolate le radici dei polinomi di Hermite calcolando gli autovalori (opportunamente riscalati di $\sqrt{2}$) della matrice

$$\begin{pmatrix} 0 & \sqrt{1} & 0 & \dots & 0 \\ \sqrt{1} & 0 & \sqrt{2} & \ddots & \vdots \\ 0 & \sqrt{2} & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & \sqrt{N} \\ 0 & \dots & 0 & \sqrt{N} & 0 \end{pmatrix}$$

Visto che nelle formule per le derivate compaiono i termini $t_i - t_j$ viene calcolata una matrice minus tale che

$$\text{minus}_{i,j} = t_i - t_j$$

Viene calcolata anche versione leggermente modificata minusplus, che differisce da minus perché gli zeri sulla diagonale di quest'ultima sono stati sostituiti da 1 in modo da poter calcolare più agevolmente i prodotti.

Per calcolare la matrice di differenziazione abbiamo bisogno di conoscere le derivate prime, seconde e terze dei polinomi di Lagrange centrati nelle radici di Hermite calcolate nei punti griglia.

Il calcolo della derivata prima è facile usando la formula:

$$\pi'(t_j) = \prod_{i \neq j} (t_j - t_i)$$

che si trasforma in un prodotto per righe di minusplus

La derivata seconda è data da:

$$\pi''(t_j) = 2 \sum_{i \neq j} \prod_{h \neq j, h \neq i} (t_j - t_h)$$

Il calcolo viene eseguito facendo variare $i \in 1, ..., N$ e sommando ogni volta al risultato il prodotto delle righe di minusline dove nella colonna i-esima è stato sostituito 1 in tutte le righe tranne la i-esima dove è stato sostituito 0.

```
for i = (1:N)
  mask = ones(N,1);
  mask(i) = 0;
  minusplusMOD = minusplus;
  minusplusMOD(:,i)= mask;
  pi2T = pi2T + 2*(prod((minusplusMOD)'))';
end
```

Per la derivata terza la formula peggiora:

$$\pi'''(t_k) = 3 \sum_{i \neq k} \sum_{j \neq k, j \neq i} \prod_{h \neq k, h \neq i, h \neq j} (t_k - t_h)$$

La soluzione adattata per il calcolo è analoga alla precedente

```
for i = (1:N)
  for j = (1:N)
    if i ~= j
      maski = ones(N,1);
      maski(i) = 0;
      maski(j) = 0;
      maskj = ones(N,1);
      maskj(i) = 0;
      maskj(j) = 0;
      minusplusMOD = minusplus;
      minusplusMOD(:,i)= maski;
      minusplusMOD(:,j)= maskj;
      pi3T = pi3T + 3*(prod((minusplusMOD)'))';
    end
  end
end
```

dove maski servono ad escludere dalla produttoria i fattori $(t_k - t_i)$ e $(t_k - t_j)$ e dalla sommatoria le righe i-esime e j-esime[2].

[2]In realtà impostare a zero le due righe sia su maski sia su maskj è superfluo, ne basta solo una delle due

Ora che conosciamo le derivate dei polinomi di Lagrange possiamo calcolare le matrici di differenziazione di primo e secondo grado utilizzando le formule

$$d_{j,k}^{(1)} = \begin{cases} \frac{\pi'(t_j)}{(t_j - t_k)\pi'(t_k)} & j \neq k \\ \frac{\pi''(t_k)}{2\pi'(t_k)} & j = k \end{cases}$$

$$d_{j,k}^{(2)} = \begin{cases} \frac{1}{t_j - t_k}\left[\frac{\pi''(t_j)}{\pi'(t_k)} - 2d_{j,k}^{(1)}\right] & j \neq k \\ \frac{\pi'''(t_k)}{3\pi'(t_k)} & j = k \end{cases}$$

2.3 Implementazione: ciclo principale: calcolo di $T(v_k)$ e dello jacobiano nel punto

Dopo aver esteso per simmetria il potenziale, risolviamo il problema diretto (con directSLP_inner1). A questo punto possiamo calcolare $T(v_k)$ come differenza tra i primi M autovalori calcolati e gli autovalori in ingresso.

Per applicare il metodo di Newton dobbiamo calcolare lo Jacobiano di T nel punto v_k. La formula può essere ricavata derivando l'equazione di Sturm Liouville, infatti dopo alcuni calcoli algebrici si arriva alla relazione

$$y_i^T \frac{\partial V}{\partial v_j} y_i = y_i^T \frac{\partial \lambda_i}{\partial v_j} y_i$$

dove y_i è l'i-esimo autovettore relativo all'i-esimo autovalore λ_i, v_j è il valore del potenziale nel punto ξ_j e V è la matrice che ha sulla diagonale il valore del potenziale calcolato nei punti di griglia esteso per simmetria.

Da questo si ricava che, utilizzando l'ortonormalità dei y_i, si ha

$$\frac{\partial \lambda_i}{\partial v_j} = 2((y_i)_j)^2$$

dove il fattore moltiplicativo 2 deriva dal fatto che v_j compare due volte nella matrice V (perché abbiamo esteso il potenziale con la simmetria).

Questa formula è implementata in

```
Ak = 2*(((Yk(2:M+1,1:M))').^2)
```

dove dobbiamo ricordarci che, per le condizioni al contorno di Dirichlet, dobbiamo ignorare i valori del potenziale in 0

2.4 Implementazione: ciclo principale: regolarizzazione e aggiornamento del potenziale

Visto il cattivo condizionamento dello Jacobiano A_k si utilizza il metodo di Tikhonov per regolarizzare il problema

$$\Delta v_k = A_k^{-1} T(v_k)$$

Per questo passaggio utilizziamo i Regularization Tools di Per Christian Hansen che si trovano alla pagina `http://www.imm.dtu.dk/~pcha/Regutools/regutools.html`

Per prima cosa calcoliamo la scomposizione in valori singolari generalizzata dello jacobiano e della matrice di penalizzazione con l'istruzione

```
[ WW, SigmaM, XX, VV] = cgsvd(Ak, DiffMat) ;
```

Ora possiamo cercare il parametro di ottimizzazione ottimale con l'istruzione

```
reg_corner = l_curve(WW, SigmaM, Tk, 'Tikh') ;
```

Da esperimenti su uno dei casi di prova si vede che questo metodo sceglie parametri di regolarizzazione troppo grandi e, quindi, regolarizza troppo. Per questo limitiamo la scelta del paramentro di regolarizzazione sostituendo la chiamata a l_curve con

```
reg_param = logspace(6,-4,600);
[x, rho, eta] = tikhonov(WW, SigmaM, XX, Tk, reg_param, vk);
reg_corner = l_corner(rho, eta, reg_param, WW, SigmaM, Tk);
```

Una volta ottenuto il parametro otteniamo la soluzione regolarizzata con il comando

```
Deltavk = tikhonov(WW, SigmaM, XX, Tk, reg_corner, vk) ;
```

che ha risolto il problema

$$\text{Deltavk} = \arg\min_x \left(\|\text{Ak}x - \text{Tk}\|^2 + \text{reg\_corner} \, \|\text{DiffMat}\,(x - \text{vk})\|^2 \right)$$

Ora che è disponibile Δv_k possiamo aggiornare il valore di vk per la prossima iterazione

```
vk = vk - Deltavk ;
```

2.5 Implemenatazione: fine ciclo e confezionamento dell'output

Il ciclo viene terminato quando viene raggiunto il numero massimo di iterazioni consentite Kmax oppure quando la variazione Deltavk è minore di tol $*$ vk, cioè ci stiamo spostando "poco" dal punto in cui siamo.

Infine il vettore del valore del potenziale nei punti griglia viene riscalato all'intervallo iniziale ed esteso per simmetria.

```
q = extender * ((pi/L)^2 * vk ) ;
```