Corso di Laurea in Informatica – Università di Ferrara
Calcolo Numerico
Prova scritta e pratica del 19/01/2023 – A.A. 2022–2023
Tempo a disposizione: 4 ore

Istruzioni

Gli esercizi sono di due tipologie:

- tipologia (T): esercizio che riguarda gli argomenti teorici del corso. **lo svolgimento di tale esercizio va effettuato sul foglio, che verrà consegnato a parte**;
- tipologia (M): esercizio di programmazione da svolgere in ambiente Matlab. Il codice di svolgimento di tale esercizio va scritto sulla propria postazione PC e verrà consegnato inserendolo nella cartella di lavoro dedicata, già predisposta nella postazione.

  N.B.: tutti gli M-files consegnati devono contenere, all'inizio, due righe di commento con l'indicazione del proprio cognome (nella prima riga) e del proprio numero di matricola (nella seconda riga), pena l'esclusione dalla prova scritta.

Gli esercizi denotati con "(M + T)" riguardano argomenti sia teorici, che di programmazione Matlab: la parte teorica dovrà essere svolta sul foglio, mentre la parte di programmazione Matlab verrà consegnata insieme ai sorgenti degli altri esercizi.

Esercizio 1

1) (T) (2 punti) Determinare la rappresentazione nel formato ANSI standard IEEE a 32 bit del numero reale $\alpha = (-70407.050505)_{10} \in \mathbb{R}$, mostrando tutti i calcoli della conversione.
2) (M) (2 punti) Siano dati i seguenti coefficienti:

$$c_0 = -\sqrt{-\ln\big(3/10^5\big)}\,, \qquad c_1 = 9.0534\,, \qquad c_2 = \min\Big\{\sin(5.7),\, 71.6\mathrm{e}^{-4},\, 2.3/\pi^2\Big\}\,,$$
$$c_4 = \frac{1}{4}\max\Big\{\mathrm{e}^{\pi},\, \pi^{\mathrm{e}}\Big\}\,, \qquad c_6 = \log_2\Big(\arccos\big(1.64 - \mathrm{e}^{0.7}\big)\Big)\,, \qquad c_7 = \Big|\tan\big(4.2\pi - 2.8\big)\Big|,$$

del polinomio $p_7(x) = c_7x^7 + \ldots + c_1x + c_0$. Realizzare un M-script file Matlab che, usando la M-function `ruffiniHorner` in allegato, calcoli e stampi il valore del polinomio $p_7(x)$ e delle sue derivate prima $p_7'(x)$ e seconda $p_7''(x)$ in un prefissato punto x_0 accettato in input. Lo script visualizzi anche il grafico del polinomio nell'intervallo $[-2.5,\, 1.1]$. Provare lo script con il valore $x_0 = -0.97$, riportando nel foglio delle risposte i valori ottenuti.

Esercizio 2

1) (T) (3 punti) Determinare l'insieme $\Omega_\varphi \subseteq \mathbb{R}$ di definizione, l'errore inerente e l'errore algoritmico nel calcolo dell'espressione:
$$\varphi(x) = x\ln(\pi) - \sqrt{(2+x)/5} \qquad x \in \Omega_\varphi$$
Valutare inoltre l'indice di condizionamento e l'indice algoritmico. Determinare gli eventuali valori di x per i quali il problema è mal condizionato. Determinare inoltre eventuali valori di x per i quali il calcolo è instabile.
2) (M) (3 punti) Realizzare un M-function file per la valutazione della funzione $\varphi_{\alpha,\beta,\gamma,\delta}(x) = \alpha x + \beta\sqrt{(\gamma+x)/\delta}$, generalizzazione della $\varphi(x)$ del punto precedente, nella quale $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ sono fissati dall'utente. Programmare la M-function in modo che riceva tutti questi valori usando un unico parametro formale di ingresso, oltre ad `x`. Per ogni fissato vettore $\boldsymbol{x}$ di valori, la M-function esegua in modo efficiente il calcolo del valore della funzione per tutte le componenti x_i di $\boldsymbol{x}$ e restituisca in output il vettore $\boldsymbol{y}$ con i risultati delle valutazioni di $\varphi_{\alpha,\beta,\gamma,\delta}(\boldsymbol{x})$. Prevedere opportuni controlli (ingressi non vuoti, possibili problemi numerici, ...).
In un M-script di prova, si consideri un vettore $\boldsymbol{x}$ di N componenti equispaziate nell'intervallo $[a, b]$. Lo script invochi la funzione in un ciclo per ciascuna componente x_i di $\boldsymbol{x}$, salvando i risultati nel vettore `fx`. Poi invochi nuovamente la M-function passandole l'intero vettore $\boldsymbol{x}$, salvando i risultati nel vettore `z`. Il ciclo e la

seconda invocazione della M-function siano temporizzati separatamente. Lo script visualizzi a console i due tempi di calcolo `tempoCiclo` e `tempoVett`, insieme alla loro differenza relativa rispetto al tempo del ciclo e alla percentuale di `tempoVett` rispetto a `tempoCiclo`. Per le visualizzazioni, si utilizzino opportune istruzioni di stampa formattata. Lo script disegni infine il grafico di $\varphi_{\alpha,\beta,\gamma,\delta}(x)$ nell'intervallo $[a, b]$ usando le ascisse $\boldsymbol{x}$ ed i valori in `z`.
Provare lo script con $[a, b] = [-1.8,\ 10]$, $\alpha = 0.23$, $\beta = 9.45$, $\gamma = 3$, $\delta = 2$ ed $N = 200\,001$ (duecentomilauno).

Esercizio 3

1) (T) (4 punti) Risolvere con il metodo di eliminazione di Gauss con pivoting parziale il seguente sistema lineare:

$$\begin{cases} 2x_1 + x_4 = 0 \\ -x_1 + (1/2)x_2 = -2 \\ (3/10)x_1 + (3/5)x_3 = -1 \\ -3x_1 + (1/5)x_2 - (1/10)x_3 + (1/2)x_4 = 0 \end{cases}$$

Esplicitare tutte le matrici coinvolte nella fattorizzazione $PA = LR$ usata per risolvere il sistema. Calcolare il determinante di A, usando la fattorizzazione ottenuta. Verificare se il sistema può essere risolto anche con la strategia diagonale. [*Suggerimento: conviene mantenere i valori in forma di frazione.*]
Scrivere almeno una delle formule con le quali si può calcolare una stima del numero di condizionamento in norma infinito della matrice del sistema.

2) (M) (3 punti) Realizzare un M-script file che controlli la soluzione ottenuta nel punto precedente, sia utilizzando le M-function `gauss2`, `rtrisol` e `ltrisol` in allegato, sia usando l'opportuna funzione predefinita di Matlab. Stampare a video soluzione e residuo normalizzato (in norma infinito). Calcolare poi la matrice $B = AA^T$ e verificare se essa è definita positiva, eseguendo la fattorizzazione di Cholesky mediante la M-function `chol` di Matlab. Stampare a video il risultato. Nel caso B sia definita positiva, risolvere il sistema $B\boldsymbol{x} = \boldsymbol{c}$, con $\boldsymbol{c} = (8/3,\ -1/5,\ 3/10,\ -9/5)^T$, usando il fattore di Cholesky e le M-function in allegato.

Esercizio 4

Si consideri la seguente matrice:

$$A = \begin{pmatrix} 8/3 & -1 & 1/2 \\ 2/5 & 7/3 & 1/2 \\ 3/5 & -1/3 & 4 \end{pmatrix}$$

1) (T) (3 punti) Disegnare i dischi di Gershgorin per righe e per colonne della matrice A e dedurre da questi se essa è invertibile oppure no, motivando la risposta. Determinare la riducibilità o meno di A. Se possibile, stabilire *a priori* la convergenza dei metodi iterativi di Jacobi e di Gauss-Seidel su un sistema lineare avente A come matrice dei coefficienti. Determinare poi le matrici di iterazione del metodo di Jacobi e del metodo di Gauss-Seidel. Scrivere l'espressione (in forma di prodotto di matrici) della matrice di iterazione per il metodo SOR relativa ad A per un generico parametro di rilassamento ω.

   Opzionale. [1 punto] Calcolare i polinomi caratteristici delle matrici di iterazione del metodo di Jacobi e del metodo di Gauss-Seidel.

2) (M) (3 punti) Realizzare un M-script file che controlli i risultati ottenuti al punto precedente, determinando spettro, raggio spettrale e, se possibile, velocità asintotica di convergenza di entrambi i metodi di Jacobi e di Gauss-seidel. Visualizzare a monitor i risultati con istruzioni di stampa formattate. Successivamente, considerato il sistema lineare $A\boldsymbol{x} = \boldsymbol{b}$ con $\boldsymbol{b} = (1,\ -5,\ -2)^T$, lo script determini un'approssimazione $\boldsymbol{x}^{(\mathrm{J})}$ della soluzione $\boldsymbol{x}^*$ utilizzando la M-function `jacobi` (in allegato), con una tolleranza di arresto `tol` $= 10^{-5}$, un numero massimo di iterazioni pari a `maxit` $= 50$ e vettore iniziale $\boldsymbol{x}^{(0)} = (1,\ -2,\ 1)^T$. Lo script visualizzi a monitor il numero di iterazioni compiute e l'approssimazione calcolata. Infine, lo script costruisca la matrice di iterazione del metodo SOR utilizzando $\omega = 1.27$ come parametro di rilassamento e visualizzi a console la seconda iterazione, partendo da $\boldsymbol{x}^{(0)} = (1,\ -2,\ 1)^T$.

Esercizio 5

Si consideri la funzione: $f(x) = \sqrt{x+4} - (\pi/6)\sin(2x)$, $x \in [a, b]$, $a > -4$.

1) (T) (4 punti) Ricavare l'espressione analitica del polinomio $p_3(x)$ di grado al più 3 di interpolazione di $f(x)$ in $[a, b] = [0, 4]$ su nodi di Chebychev, mediante il polinomio di Newton. Scrivere l'espressione dell'errore di interpolazione. Approssimare l'errore valutando tutte le quantità dell'espressione in $x^* = 1.5$. Confrontare tale valore con la differenza $f(x^*) - p_3(x^*)$.
   Scrivere l'espressione dei polinomi di Lagrange su 3 punti equispaziati nell'intervallo $[a, b]$ dato sopra e la formula che fornisce il polinomio interpolante $p_2(x)$ di grado al più 2 su tali nodi nella forma di Lagrange.
2) (M) (3 punti) Scrivere un M-script file che calcoli il polinomio $p_n(x)$ di grado n interpolante la funzione $f(x)$ in $[a, b]$ sugli $n+1$ nodi di Chebychev, utilizzando a tale scopo la function `polyNewton` in allegato. Lo script generi poi in uno stesso grafico le due curve $f(x)$ e $p_n(x)$, evidenziando nodi e punti di interpolazione e dotando il grafico di titolo, etichette degli assi e legenda. Lo script valuti infine l'errore relativo percentuale $e_n(x) = |p_n(x) - f(x)|/|f(x)|$ negli N punti di un campionamento uniforme dell'intervallo $[a, b]$, disegnando il corrispondente grafico in una seconda finestra grafica. Utilizzare un campionamento di $[a, b]$ con $N \geq 201$. Evidenziare opportunamente in tale grafico i nodi di interpolazione.
   Provare lo script con $[a, b] = [-2, 5]$, $n + 1 = 6$ ed $N = 201$.

Esercizio 6

1) (T) (3 punti) Si consideri la funzione $g(x) = 2.2 - f(x)$ nell'intervallo $[a, b] = [0, 3]$, dove $f(x)$ è la funzione della parte teorica dell'esercizio 5. Dire se sono soddisfatte le condizioni necessarie per l'esistenza di almeno uno zero di $g(x)$ in $[a, b]$. Scrivere le formule generali di aggiornamento dell'approssimazione z_k di uno zero di $g(x)$ per i metodi di bisezione, delle secanti e di Newton. Per ciascun metodo, calcolare esattamente due passi, partendo dagli estremi dell'intervallo o dal suo punto medio (a seconda di quanti dati iniziali richiede il metodo). Nel caso siano verificate le condizioni di esistenza di almeno uno zero $z^* \in [a, b]$ di $g(x)$, determinare quante iterazioni del metodo di bisezione sono necessarie per raggiungere un'accuratezza di almeno 10^{-4}.
2) (M) (4 punti) Implementare un M-script file che utilizzi il metodo delle equazioni normali per trovare il polinomio algebrico $p_3^*(x)$ di grado al più 3 che meglio approssima nel senso dei minimi quadrati i valori della funzione $h(x) = \rho f(x) + \delta$ in $10N$ punti nell'intervallo $[a, b] \subset \mathbb{R}$, dove $f(x)$ è la funzione della parte teorica dell'esercizio 5, $\rho \in \mathbb{R}$ è un parametro fissato e, per ogni $x \in \mathbb{R}$, il valore δ è un numero casuale in $[-\rho/2, \rho/2]$. Per un dato vettore $\boldsymbol{x}$ di ascisse, costruire esplicitamente la matrice A dei coefficienti ed il corrispondente sistema delle equazioni normali, calcolandone la soluzione $\boldsymbol{z}^*$ in modo efficiente in Matlab. Confrontare poi tale soluzione con quella che si ottiene usando direttamente l'operatore "\" (`mldivide`) di Matlab sul sistema $A\boldsymbol{z} = \boldsymbol{y}$. Disegnare su un grafico i punti $y_i = f(x_i)$ e la curva del polinomio $p_3^*(x)$ di migliore approssimazione.
   Provare lo script con $[a, b] = [0, 3]$, $\rho = 1.2$, $N = 3$ ed il vettore $\boldsymbol{x}$ generato casualmente in $[a, b]$ da una distribuzione uniforme con seme impostato a 5.

Appendice: codici forniti

```
function [r, q] = ruffiniHorner(p, a)
% RUFFINIHORNER - Schema di Ruffini-Horner
%  Calcola il valore di un polinomio p(x) nel punto x = a e i coefficienti
%  del polinomio quoziente q(x) = p(x) / (x - a)
% SYNOPSIS
%   [r, q] = ruffiniHorner(p, a)
% INPUT
%   p (double array) - Vettore dei coefficienti del polinomio, ordinati
%                      da quello di grado piu' alto a quello di grado zero
%   a (double)       - Punto (numero reale) in cui calcolare il polinomio
% OUTPUT
%   r (double)       - Valore del polinomio nel punto x = a
%   q (double array) - Vettore dei coefficienti del polinomio quoziente
%                      q(x) = p(x) / (x - a)
%
r = [];
if ( isempty(p) )
    q = [];
```

```
    warning('Il vettore p dei coefficienti e'' vuoto');
    return
elseif ( isempty(a) )
    q = p;
    warning('Il punto ''a'' in cui valutare il polinomio e'' vuoto');
    return
else
    n = numel(p) - 1;  % grado del polinomio
    q = zeros(n, 1);
    q(1) = p(1);
    for i = 2 : n+1
        q(i) = q(i-1)*a + p(i);
    end
    r = q(n+1);
    q = q(1:n);
end
end % fine della function ruffiniHorner
```

```
function [x] = ltrisol(L, b)
% LTRISOL - Soluzione di sistemi triang. inf. (per colonne)
  n = length(b);
  x = b;
  x(1) = x(1)/L(1,1);
  for j = 2:n
     % SAXPY
     x(j:n) = x(j:n) - L(j:n, j-1)*x(j-1);
     x(j)   = x(j) / L(j,j);
  end
end
```

```
function [x] = rtrisol(R,b)
% RTRISOL - Soluzione di sistema triang. sup. (per colonne)
  n = length(b);
  x = b;
  x(n) = x(n)/R(n,n);
  for j = n-1 : -1 : 1
     % SAXPY - BLAS1
     x(1:j) = x(1:j) - R(1:j, j+1)*x(j+1);
    x(j) = x(j)/R(j,j);
  end
end
```

```
function [L, R, p, deter] = gauss2(A)
% GAUSS2 - Fattorizzazione di Gauss con pivoting parziale (versione 2)
%  Esegue la fattorizzazione PA = LR con l'algoritmo di eliminazione di
%  Gauss con pivoting parziale (con aggiornamento mediante diadi)
% SYNOPSIS
%   [L, R, p, deter] = gauss2(A)
% INPUT
%   A (double array) - Matrice m x n
% OUTPUT
%   L (double array) - Matrice triangolare inferiore a diagonale unitaria
%   R (double array) - Matrice triangolare/trapezoidale superiore
%   p (double array) - Vettore delle permutazioni di righe
%   deter (double)   - determinante (della parte quadrata) di A
%
[m, n] = size(A); deter = 1;
temp = zeros(1, n); p = 1 : n;
tol = eps * norm(A, inf);  % tolleranza verso lo zero (trascurabilita')
for j = 1 : min(m-1,n)
   [amax, ind] = max( abs(A(j:n, j)) );
   ind = ind + j - 1;
   if (ind ~= j)  % pivot non in posizione diagonale: occorre scambiare righe
      % scambio di indici
```

```
      aux = p(j);  p(j) = p(ind);  p(ind) = aux;
      deter = -deter;
      % scambio di righe
      temp = A(ind,:); A(ind,:) = A(j,:); A(j,:) = temp;
   end
   deter = deter * A(j,j);
   if ( abs(A(j,j)) > tol )
      A(j+1:end, j) = A(j+1:end, j) / A(j,j);
      % operazione di base: aggiornamento mediante diadi
      A(j+1:end, j+1:end) = A(j+1:end, j+1:end) - A(j+1:end, j)*A(j, j+1:end);
   end
end
deter = deter * A(n,n);
R = triu(A);
L = eye(n) + tril(A(1:n,1:n), -1);
end % fine della function gauss2
```

```
function [x, k] = jacobi(A, b, x, maxit, tol)
%JACOBI - Metodo iterativo di Jacobi per sistemi lineari
n = size(A,1);
if ( issparse(A) )
  d = spdiags(A, 0);
else
  d = diag( A );
end
if ( any( abs(d) < eps*norm(d, inf) ) )
  error('Elementi diagonali troppo piccoli.');
end
b = b ./ d;
k = 0; stop = 0;
while ( ~stop )
   k = k + 1;
   xtemp = x;
   x = x - (A*x)./d + b; % istruzione vettoriale
   stop = ( norm(xtemp - x, inf) < tol*norm(x, inf) ) ...
          || ( k == maxit );
end
if ( k == maxit )
   warning('Raggiunto il numero massimo di iterazioni maxit = %d', ...
           maxit);
end
end % fine della M-function jacobi.m
```

```
function [z, d] = polyNewton(x, y, xx)
% POLYNEWTON - Valuta in xx il polinomio di Newton interpolante su (x, y)
%  Valuta nei punti xx il polinomio interpolante dei dati y sui nodi x
%  nella forma di Newton, secondo lo schema di Horner generalizzato.
% SYNOPSIS
%  [z, d] = polyNewton(x, y, xx)
% INPUT
%  x  (double array) - vettore dei nodi o punti di osservazione
%  y  (double array) - vettore delle osservazioni
%  xx (double array) - vettore dei punti in cui calcolare il polinomio
%                      di Newton
% OUTPUT
%  z  (double)       - valore del polinomio nei punti xx
%  d  (double array) - coefficienti del polinomio di Newton
%
  x = x(:); y = y(:);
  n = length(x);
  d = tabDiff(x,y);
  z = d(n)*ones(size(xx));
  for i = n-1 : -1 : 1
    z = z .* (xx - x(i)) + d(i);
  end
```

```
end

% Funzione tabDiff per il calcolo degli elementi della diagonale della tabella
% delle differenze divise

function [d] = tabDiff(x, y)
% TABDIFF - Tabella delle differenze divise sui nodi x e i valori y
% INPUT
%   x  (double array) - vettore dei nodi o punti di osservazione
%   y  (double array) - vettore delle osservazioni
% OUTPUT
%   d  (double array) - coefficienti del polinomio di Newton (ordinati
%                       per grado crescente del termine corrispondente)
%
  x = x(:); y = y(:);
  n = length(x); % numero dei punti di interpolazione ( = grado polinomio + 1 )
  d = y;
  for k = 2 : n
    d(k:n) = ( d(k:n) - d(k-1 : n-1) ) ./ ( x(k:n) - x(1 : n-k+1) );
  end
end
```

Soluzioni e commenti

Nel seguito sono riportate le soluzioni degli esercizi del compito, includendo commenti che aiutino il più possibile la comprensione della soluzione. Qualora uno stesso esercizio possa eventualmente essere risolto in più modi, in alcuni casi si riportano le descrizioni anche di qualche modo alternativo. In generale, questo rende il testo delle soluzioni degli esercizi inevitabilmente molto più lungo di quanto non sia effettivamente richiesto a stedentesse e studenti nel momento della prova scritta, pertanto **la lunghezza delle soluzioni qui riportate è da considerarsi NON RAPPRESENTATIVA della lunghezza del compito**. Si invitano studentesse e studenti a contattare il docente per eventuali dubbi o chiarimenti.

Nella trascrizione delle soluzioni è stata posta la massima cura. Tuttavia, nel caso si rilevino sviste e/o errori di vario genere, si invitano studentesse e studenti a comunicarlo via e-mail al docente.

Soluzione esercizio 1

Teoria

70407	/2
35203	1
17601	1
8800	1
4400	0
2200	0
1100	0
550	0
275	0
137	1
68	1
34	0
17	0
8	1
4	0
2	0
1	0
0	1

0.050505	×2
0.101010	0
0.202020	0
0.404040	0
0.808080	0
0.616160	1
0.232320	1
0.464640	0
⋮	⋮

$$
\begin{aligned}
y &= (-70407.050505)_{10} \\
&= (10001001100000111.0000110\ldots)_2 \\
&= (1.00010011000001110000110\ldots)_2 \cdot 2^{(+10000)_2}
\end{aligned}
$$

$y < 0 \Rightarrow s = 1$

$p = (+10000)_2 = (+16)_{10}$

$\Rightarrow \widetilde{p} = p + bias = 16 + 127 = 143 = 128 + 8 + 4 + 2 + 1 = (10001111)_2$

$m = (1.00010011000001110000110)_2$

$\Rightarrow \widetilde{m} = 00010011000001110000110$ (attenzione al troncamento della mantissa alla 24-esima cifra binaria!)

32 bit = 4 byte

1° byte	2° byte	3° byte	4° byte

$$\text{IEEE}_{32}(y) = \underbrace{1}_{s}\underbrace{10000111}_{\widetilde{p}}\underbrace{00100110000011100001 10}_{\widetilde{m}}$$

Matlab

Contenuto dell'M-script file `esercizio1.m`:

```
% Cognome Nome
% Matricola
%----------------------------
%  esercizio 1 - 19/01/2023
%----------------------------
close all; clear all; clc;
disp('Esecizio 1');

p7    = zeros(8,1);
p7(1) = abs( tan( 4.2*pi - 2.8 ) );                 % c7
p7(2) = log2( acos( 1.64 - exp(0.7) ) );            % c6
p7(4) = 0.25*max( exp(pi), pi^(exp(1)) );           % c4
p7(6) = min( [sin(5.7); 71.6*exp(-4); 2.3/pi^2] ); % c2
p7(7) = 9.0534;                                     % c1
p7(8) = -sqrt( -log( 3.0e-5 ) );                    % c0

x0 = input('Inserire il punto nel quale valutare il polinomio (double): x0 = ');

[r   , q ] = ruffiniHorner( p7, x0 );
[derp, q1] = ruffiniHorner( q,  x0 );
[ders, q2] = ruffiniHorner( q1, x0 );
fprintf('\nValore del polinomio in x0 = %g: p(x0) = %g', x0, r);
fprintf('\nDerivata prima del polinomio in x0: p''(%g) = %g', x0, derp);
fprintf('\nDerivata seconda del polinomio in x0: p''''(%g) = %g\n\n', x0, 2*ders);
ph = fplot( @(x)(polyval(p7,x)), [-2.5,1.1] );
```

Eseguendo lo script si ottengono i seguenti risultati in output e una figura simile alla seguente (quella presentata qui è stata leggermente migliorata per esigenze di stampa):

Punto di valutazione: $x_0 = -0.97$
Valori calcolati in x_0:
$p_7(x_0) \approx -7.7794$
$p_7'(x_0) \approx -7.4703$
$p_7''(x_0) \approx 37.2634$

Valori approssimati dei coefficienti del polinomio:

$$p_7(x) \approx \begin{pmatrix} 1.4589 \\ 0.9663 \\ 0 \\ 5.7852 \\ 0 \\ -0.5507 \\ 9.0534 \\ -3.2271 \end{pmatrix}$$

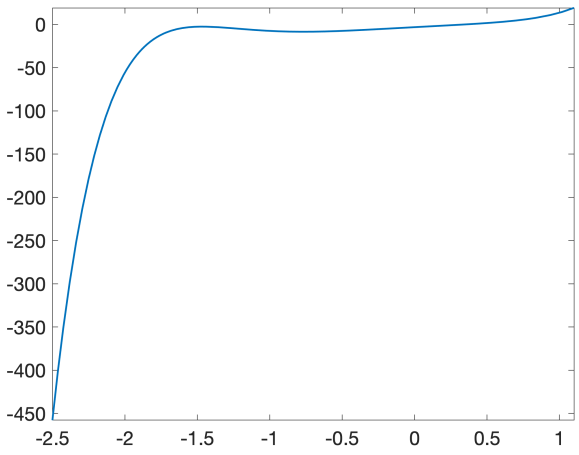


Figura 1: grafico del polinomio $p_7(x)$ dell'esercizio 1 nell'intervallo $[-2.5, 1.1]$.

Nota. Si richiama l'attenzione sul fatto che nei grafici di Matlab sia sempre MOLTO importante tenere presente in quale scala vengono visualizzati gli assi. In particolare, il motore grafico di Matlab è impostato di default nella modalità che prevede di scalare automaticamente gli assi delle ascisse e delle ordinate per far sì che l'immagine del grafico "si adatti" alle dimensioni della finestra grafica, che sono generalmente rettangolari, quasi quadrate.

Quando la scala degli assi è molto diversa, come nel caso in esame, si potrebbe essere tratti in inganno dalla figura, che risulta **apparentemente** incoerente con i valori numerici delle derivate (non dei valori della funzione).

Soluzione esercizio 2

Teoria

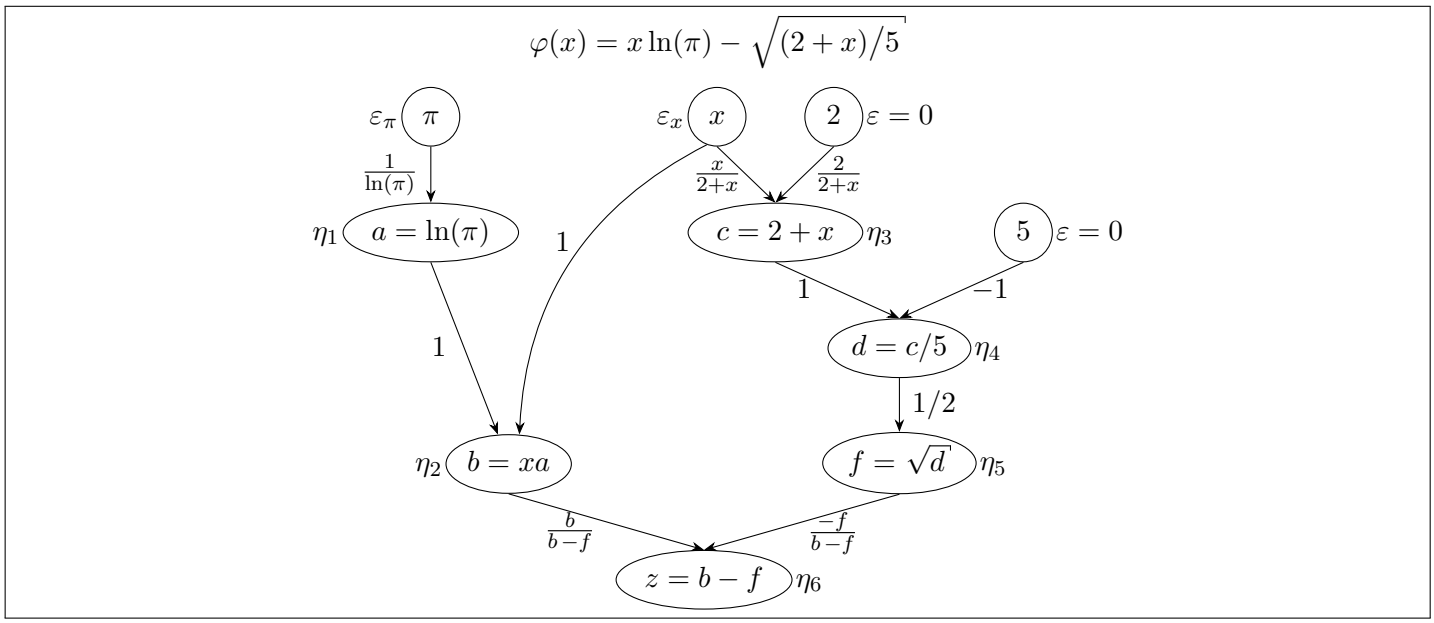


Figura 2: grafo della funzione $\varphi(x)$ dell'esercizio 2.

Notiamo che la funzione $\varphi(x)$ è definita solo quando $(2+x)/5 \geq 0$, ossia solo in $\Omega = \{x \in \mathbb{R} \,|\, x \geq -2\} = [-2, +\infty[$. Con riferimento al grafo della figura 2, notiamo che la costante π è irrazionale: è dunque corretto considerare non nullo il corrispondente errore ε_π. L'errore inerente è quindi dato da:

$$\begin{aligned}
\epsilon_{\text{dati}} &= \left(\frac{1}{\ln(\pi)} \cdot 1 \cdot \frac{b}{b-f}\right)\varepsilon_\pi + \left(1 \cdot \frac{b}{b-f} + \frac{x}{2+x} \cdot 1 \cdot \frac{1}{2} \cdot \frac{-f}{b-f}\right)\varepsilon_x \\
&= \frac{1}{b-f}\left\{\left(\frac{b}{\ln(\pi)}\right)\varepsilon_\pi + \left(b + \frac{x\cdot(-f)}{2(2+x)}\right)\varepsilon_x\right\} \\
&= \frac{1}{x\ln(\pi) - \sqrt{(2+x)/5}}\left\{\left(\frac{x\cancel{\ln(\pi)}}{\cancel{\ln(\pi)}}\right)\varepsilon_\pi + \left(x\ln(\pi) + \frac{x\sqrt{(2+x)/5}}{2(2+x)}\right)\varepsilon_x\right\} \\
&= \frac{x}{x\ln(\pi) - \sqrt{(2+x)/5}}\left\{\varepsilon_\pi + \left(\ln(\pi) + \frac{1}{2\sqrt{5}\sqrt{2+x}}\right)\varepsilon_x\right\} = \theta_\pi\varepsilon_\pi + \theta_x\varepsilon_x
\end{aligned}$$

L'indice di condizionamento risulta:

$$I_{\text{cond}} = |\theta_\pi| + |\theta_x| = \left|\frac{x}{x\ln(\pi) - \sqrt{(2+x)/5}}\right|\left\{1 + \left|\ln(\pi) + \frac{1}{2\sqrt{5}\sqrt{2+x}}\right|\right\}$$

Vediamo subito che per $x \to -2^+$ il termine $x\ln(\pi)$ rimane limitato, mentre $(2+x) \to 0^+$: pertanto negli intorni destri di $x_0 = -2$ (che è l'estremo sinistro di Ω) si ha $I_{\text{cond}} \to +\infty$. Inoltre, dobbiamo vedere se e, nel caso, dove il denominatore del fattore fuori dalle parentesi graffe potrebbe annullarsi:

$$x\ln(\pi) - \sqrt{(2+x)/5} = 0 \quad \Leftrightarrow \quad x\ln(\pi) = \sqrt{(2+x)/5}$$

Osserviamo subito che, essendo $\ln(\pi) > 0$, questa equazione non lineare non ammette soluzioni in $[-2, 0[$, perché qui il primo membro è negativo, mentre il secondo è non negativo. Inoltre, è immediato accorgersi che anche $x = 0$ non può essere soluzione dell'equazione, perché $0 \neq \sqrt{2/5}$. Consideriamo pertanto solo $x > 0$: in tal caso, tutte le quantità dell'equazione sono non negative e il radicando è positivo. Elevando al quadrato entrambi i membri otteniamo

$$x\ln(\pi) = \sqrt{(2+x)/5} \quad \Leftrightarrow \quad 5x^2\ln^2(\pi) = 2 + x \quad \Leftrightarrow \quad \big(5\ln^2(\pi)\big)x^2 - x - 2 = 0$$

$$x_{1,2} = \frac{1 \mp \sqrt{1 + 4\cdot 2\cdot 5\ln^2(\pi)}}{10\ln^2(\pi)} \quad \Rightarrow \quad x_1 \approx -0.48143 < 0 \text{ non accettabile, } x_2 \approx 0.63405 > 0.$$

Pertanto, x_2 è l'unica radice positiva dell'equazione: la funzione è un polinomio quadratico con coefficiente positivo del termine di secondo grado (ossia, il grafico è una parabola convessa). Dato che il termine di I_{cond} entro parentesi graffe rimane limitato per $x \to x_2$, deduciamo che per $x > 0$ si ha $I_{\text{cond}} \to +\infty$ solo per $x \to x_2$. Non ci sono altri punti di mal condizionamento in Ω. Inoltre, per $x \to +\infty$ il termine fuori dalle graffe si comporta come $x^{-1/2}$ e lo stesso accade per quello dentro le graffe, pertanto I_{cond} rimane limitato e il condizionamento di $\varphi(x)$ rimane buono per $x \gg$. Concludiamo quindi che si ha mal condizionamento di $\varphi(x)$ in Ω solo quando $x \to -2^+$ oppure $x \to x_2 \approx 0.63405$.

Nota: per una verifica rapida del valore di x_2 si può utilizzare la funzione predefinita `fzero` di Matlab nel modo seguente:

```
>> funz = @(x)( x*log(pi) - sqrt((2 + x)/5) ); fzero( funz, [1.0E-2, 1.0] )
```

dove $\left[10^{-2}, 1\right]$ è un intervallo che contiene la radice dell'equazione. È facile avere un'idea del comportamento della funzione guardandone il grafico in un intervallo abbastanza ampio:

```
>> fplot( funz, [1.0E-2, 20] )
```

da cui si riscontra anche che x_2 è l'unica radice dell'equazione in $\mathbb{R}^+$.

L'errore algoritmico e l'indice algoritmico sono:

$$\begin{aligned}
\epsilon_{\text{alg}} &= \frac{b}{b-f}(\eta_1+\eta_2) + \frac{-f}{b-f}\left(\frac{1}{2}(\eta_3+\eta_4)+\eta_5\right) + \eta_6 = \frac{1}{b-f}\left\{b(\eta_1+\eta_2) - f\left(\frac{1}{2}(\eta_3+\eta_4)+\eta_5\right)\right\} + \eta_6 \\
&= \frac{1}{x\ln(\pi) + \sqrt{(2+x)/5}}\left\{x\ln(\pi)(\eta_1+\eta_2) - \sqrt{(2+x)/5}\left(\frac{1}{2}(\eta_3+\eta_4)+\eta_5\right)\right\} + \eta_6 \\
I_{\text{alg}} &= 2\frac{\left|x\ln(\pi)\right| + \sqrt{(2+x)/5}}{\left|x\ln(\pi) - \sqrt{(2+x)/5}\right|} + 1
\end{aligned}$$

Si vede che, come prima, i punti che richiedono attenzione sono solo $x_0 = -2$ e $x_2 \approx 0.63405$, nei pressi dei quali $I_{\text{alg}} \to +\infty$. Non ci sono altri punti di Ω che richiedano attenzione. Inoltre, l'indice algoritmico rimane limitato anche per $x \gg$, perché la frazione tende a 1 per $x \to +\infty$. Perciò, concludiamo che il calcolo di $\varphi(x)$ è stabile in Ω, tranne in un intorno destro di x_0 e nei pressi di x_2.

Matlab

Contenuto dell'M-function file `esercizio2.m`:

```
% Cognome Nome
% Matricola
function [y] = esercizio2( x, params )
%ESERCIZIO2 - Esercizio 2 prova scritta di Calcolo Numerico del 19/01/2023
%  Valuta la funzione
%    f(t; a,b,c,d) =  a*t + b*sqrt( (c + t)/d )
%  nel punto x, per fissati valori dei parametri reali 'a', 'b', 'c' e 'd'.
%  Versione con controlli sull'input e parametri potenzialmente vettoriali
% SYNOPSIS
%   y = esercizio2( x, params )
% INPUT
%   x       (double array) - Vettore di valori nei quali valutare la funzione f(t)
%   params  (double array) - Vettore dei parametri: params(1) = a,
%                            params(2) = b, params(3) = c, params(4) = d
% OUTPUT
%   y       (double array) - Vettore dei valori della funzione f(t) nei punti x

  % Controlli sui paramteri di input
  if ( isempty(x) )
      error('Il vettore delle ascisse deve essere non vuoto');
  elseif ( ~isnumeric(x) )
      error('Le ascisse devono essere numeri reali');
  end
  if ( isempty(params) )
      warning(sprintf('%s\n%s\n', ...
          'Parametri non forniti: si assumono valori di default:', ...
```

```
            'a = 2, b = 0.4, c = 3, d = -1.2'));
        params = [0.23, 9.45, 3, 2];
    elseif ( ~isnumeric(params) )
        error('Non sono ammessi parametri non numerici');
    elseif ( abs(params(4)) < eps )
        warning('Quarto parametro numericamente troppo piccolo.');
    end

    x = x(:); params = params(:);
    idxZeroDenom = find( params(3) + x < 0 );
    if ( ~isempty(idxZeroDenom) )
        warning('Presenza di radicandi negativi');
    end

    % Calcolo dei valori della funzione nelle ascisse x
    y = params(1)*x + params(2)*sqrt( (params(3) + x) / params(4) );

end
```

Contenuto dell'M-script file `testEsercizio2.m`:

```
% Cognome Nome
% Matricola
%-----------------------------
%  esercizio 2 - 19/01/2023
%-----------------------------
close all; clear all; clc;
disp('Test esercizio 2');

N = 200001; a = -1.8; b = 10;
parametri = [0.23; 9.45; 3; 2];
x = linspace(a,b,N)';
fx = zeros(N,1);
tic;
for i = 1:N
    fx(i) = esercizio2( x(i), parametri );
end
tempoCiclo = toc;
tic; z = esercizio2( x, parametri );
tempoVett = toc;
fprintf('\ntempo del ciclo: tempoCiclo = %g secondi', tempoCiclo);
fprintf('\ntempo dell''invocazione vettoriale: tempoVett = %g secondi', tempoVett);
fprintf('\ndifferenza relativa: |tempoCiclo - tempoVett| / tempoCiclo = %g', ...
        abs(tempoCiclo - tempoVett) / tempoCiclo );
fprintf('\nincidenza percentuale: tempoVett / tempoCiclo = %g%%', ...
        tempoVett*100/tempoCiclo );
fprintf('\n\n');
ph = plot(x,z);
```

L'esecuzione dell'M-script di prova `testEsercizio2.m` dovrebbe fornire risultati simili ai seguenti:

tempo del ciclo:	$t_{\text{ciclo}} \approx 0.1210$ secondi
tempo vettoriale:	$t_{\text{vett}} \approx 0.0055$ secondi
differenza relativa:	$\lvert t_{\text{vett}} - t_{\text{ciclo}}\rvert / \lvert t_{\text{ciclo}}\rvert \approx 0.9548$
incidenza percentuale:	$\lvert t_{\text{vett}}\rvert / \lvert t_{\text{ciclo}}\rvert \approx 4.5\%$

Ovviamente, i valori effettivi delle temporizzazioni dipendono dalla macchina e anche, sulla stessa macchina, dal livello di utilizzo del sistema sottostante: pertanto, possono risultare diversi (anche significativamente) da quelli riportati qui. Ciò che dovrebbe emergere in modo evidente è che il tempo per la valutazione con sintassi vettoriale richiede, grosso modo, meno di 5% del tempo richiesto dalle stesse valutazioni eseguite nel ciclo.

Il grafico generato dal codice è riportato in figura 3 (qui leggermente migliorato per esigenze di stampa).

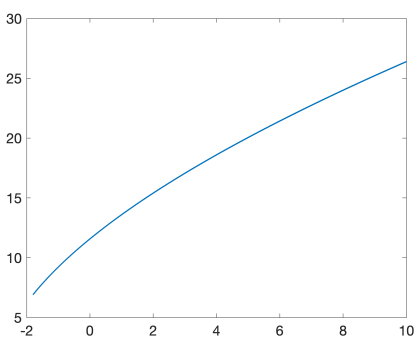


Figura 3: grafico generato dal codice dell'esercizio 2 con i valori richiesti.

Soluzione esercizio 3

Teoria

N.B. In ciò che segue, i numeri in trasparenza di ciascuna colonna rappresentano i moltiplicatori di quella stessa colonna, memorizzati nelle posizioni degli zeri sottodiagonali del fattore R finale. Questi valori **non entrano** nei calcoli successivi, ma vengono permutati insieme alle righe. Al termine, tutti i valori in trasparenza rappresentano la parte triangolare inferiore della matrice L della fattorizzazione di A.

Nel seguito indicheremo con $\widehat{A}$ la matrice completa $\left[A^{(k)}|\boldsymbol{b}^{(k)}\right]$ della generica iterazione k-esima, omettendo in essa l'indice d'iterazione k per semplicità di notazione. Per il sistema dato si ha:

$$\left[A^{(1)}|\boldsymbol{b}^{(1)}\right] = \left(\begin{array}{cccc|c} 2 & 0 & 0 & 1 & 0 \\ -1 & 1/2 & 0 & 0 & -2 \\ 3/10 & 0 & 3/5 & 0 & -1 \\ -3 & 1/5 & -1/10 & 1/2 & 0 \end{array}\right), \quad \max_{i=1,\dots,4}\left|a_{i,1}^{(1)}\right| = 3\,,\ i_* = 4,\ \Rightarrow\ P_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow\ P_1\left[A^{(1)}|\boldsymbol{b}^{(1)}\right] = \left(\begin{array}{cccc|c} -3 & 1/5 & -1/10 & 1/2 & 0 \\ -1 & 1/2 & 0 & 0 & -2 \\ 3/10 & 0 & 3/5 & 0 & -1 \\ 2 & 0 & 0 & 1 & 0 \end{array}\right),\ \boldsymbol{m}^{(1)} = \begin{pmatrix} 0 \\ 1/3 \\ -1/10 \\ -2/3 \end{pmatrix},\ L_1 = I_4 - \boldsymbol{m}^{(1)}\mathbf{e}_1^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1/3 & 1 & 0 & 0 \\ 1/10 & 0 & 1 & 0 \\ 2/3 & 0 & 0 & 1 \end{pmatrix}$$

$$\xrightarrow[\substack{\widehat{A}_{3,*}\leftarrow\widehat{A}_{3,*}+(1/10)\widehat{A}_{1,*}\\ \widehat{A}_{4,*}\leftarrow\widehat{A}_{4,*}+(2/3)\widehat{A}_{1,*}}]{\widehat{A}_{2,*}\leftarrow\widehat{A}_{2,*}-(1/3)\widehat{A}_{1,*}} \left[A^{(2)}|\boldsymbol{b}^{(2)}\right] = L_1P_1\left[A^{(1)}|\boldsymbol{b}^{(1)}\right] = \left(\begin{array}{cccc|c} -3 & 1/5 & -1/10 & 1/2 & 0 \\ 1/3 & 13/30 & 1/30 & -1/6 & -2 \\ -1/10 & 1/50 & 59/100 & 1/20 & -1 \\ -2/3 & 2/15 & -1/15 & 4/3 & 0 \end{array}\right),$$

$$\max_{i=2,3,4}\left|a_{i,2}^{(2)}\right| = \frac{13}{30}\,,\ i_* = 2, \quad \Rightarrow \quad P_2 = I_4\,,$$

$$P_2\left[A^{(2)}|\boldsymbol{b}^{(2)}\right] = \left[A^{(2)}|\boldsymbol{b}^{(2)}\right], \quad \boldsymbol{m}^{(2)} = \begin{pmatrix} 0 \\ 0 \\ 3/65 \\ 4/13 \end{pmatrix}, \quad L_2 = I_4 - \boldsymbol{m}^{(2)}\mathbf{e}_2^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3/65 & 1 & 0 \\ 0 & -4/13 & 0 & 1 \end{pmatrix}$$

$$\xrightarrow[\widehat{A}_{4,*}\leftarrow\widehat{A}_{4,*}-(4/13)\widehat{A}_{2,*}]{\widehat{A}_{3,*}\leftarrow\widehat{A}_{3,*}-(3/65)\widehat{A}_{2,*}} \left[A^{(3)}|\boldsymbol{b}^{(3)}\right] = L_2P_2\left[A^{(2)}|\boldsymbol{b}^{(2)}\right] = \left(\begin{array}{cccc|c} -3 & 1/5 & -1/10 & 1/2 & 0 \\ 1/3 & 13/30 & 1/30 & -1/6 & -2 \\ -1/10 & 3/65 & 153/260 & 3/52 & -59/65 \\ -2/3 & 4/13 & -1/13 & 18/13 & 8/13 \end{array}\right),$$

$$\max_{i=3,4}\left|a_{i,3}^{(2)}\right| = \frac{153}{260}\,,\ i_* = 3, \quad \Rightarrow \quad P_3 = I_4\,,$$

$$P_3\left[A^{(3)}|\boldsymbol{b}^{(3)}\right] = \left[A^{(3)}|\boldsymbol{b}^{(3)}\right], \quad \boldsymbol{m}^{(3)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -20/153 \end{pmatrix}, \quad L_3 = I_4 - \boldsymbol{m}^{(3)}\mathbf{e}_3^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 20/153 & 1 \end{pmatrix}$$

$$\xrightarrow{\widehat{A}_{4,*}\leftarrow\widehat{A}_{4,*}+(20/153)\widehat{A}_{3,*}} \left[A^{(4)}|\boldsymbol{b}^{(4)}\right] = L_3P_3\left[A^{(3)}|\boldsymbol{b}^{(3)}\right] = \left(\begin{array}{cccc|c} -3 & 1/5 & -1/10 & 1/2 & 0 \\ 1/3 & 13/30 & 1/30 & -1/6 & -2 \\ -1/10 & 3/65 & 153/260 & 3/52 & -59/65 \\ -2/3 & 4/13 & -20/153 & 71/51 & 76/153 \end{array}\right) = [R|\boldsymbol{c}]$$

$$R = \begin{pmatrix} -3 & 1/5 & -1/10 & 1/2 \\ 0 & 13/30 & 1/30 & -1/6 \\ 0 & 0 & 153/260 & 3/52 \\ 0 & 0 & 0 & 71/51 \end{pmatrix} \approx \begin{pmatrix} -3.0000 & 0.2000 & -0.1000 & 0.5000 \\ 0 & 0.4333 & 0.0333 & -0.1667 \\ 0 & 0 & 0.5885 & 0.0577 \\ 0 & 0 & 0 & 1.3922 \end{pmatrix}$$

$$\boldsymbol{c} = L^{-1}P\boldsymbol{b} = \begin{pmatrix} 0 \\ -2 \\ -59/65 \\ 76/153 \end{pmatrix} \approx \begin{pmatrix} 0 \\ -2.0000 \\ -0.9077 \\ 0.4967 \end{pmatrix}, \qquad P = S_1 = P_3P_2P_1 = P_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

$$S_2 = P_3P_2 = I_4 \;\Rightarrow\; \widetilde{L}_1^{-1} = S_2L_1^{-1}S_2^T = L_1^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 1 & 0 & 0 \\ -1/10 & 0 & 1 & 0 \\ -2/3 & 0 & 0 & 1 \end{pmatrix} \approx \begin{pmatrix} 1.0000 & 0 & 0 & 0 \\ 0.3333 & 1.0000 & 0 & 0 \\ -0.1000 & 0 & 1.0000 & 0 \\ -0.6667 & 0 & 0 & 1.0000 \end{pmatrix}$$

$$S_3 = P_3 = I_4 \;\Rightarrow\; \widetilde{L}_2^{-1} = S_3L_2^{-1}S_3^T = L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 3/65 & 1 & 0 \\ 0 & 4/13 & 0 & 1 \end{pmatrix} \approx \begin{pmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0.0462 & 1.0000 & 0 \\ 0 & 0.3077 & 0 & 1.0000 \end{pmatrix}$$

$$L = \widetilde{L}_1^{-1}\widetilde{L}_2^{-1}L_3^{-1} = L_1^{-1}L_2^{-1}L_3^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 1 & 0 & 0 \\ -1/10 & 3/65 & 1 & 0 \\ -2/3 & 4/13 & -20/153 & 1 \end{pmatrix} \approx \begin{pmatrix} 1.0000 & 0 & 0 & 0 \\ 0.3333 & 1.0000 & 0 & 0 \\ -0.1000 & 0.0462 & 1.0000 & 0 \\ -0.6667 & 0.3077 & -0.1307 & 1.0000 \end{pmatrix}$$

$$\left.\begin{aligned} x_4^* &= \frac{c_4}{r_{4,4}} = \frac{76/153}{71/51} = \frac{76}{213} \\ x_3^* &= \frac{c_3 - r_{3,4}x_4^*}{r_{3,3}} = \frac{-59/65 - (3/52)(76/213)}{153/260} = -\frac{112}{71} \\ x_2^* &= \frac{c_2 - r_{2,3}x_3^* - r_{2,4}x_4^*}{r_{2,2}} = \frac{(-2) - (1/30)(-112/71) - (-1/6)(76/213)}{13/30} = -\frac{928}{213} \\ x_1^* &= \frac{c_1 - r_{1,2}x_2^* - r_{1,3}x_3^* - r_{1,4}x_4^*}{r_{1,1}} = \frac{0 - (1/5)(-928/213) - (-1/10)(-112/71) - (1/2)(76/213)}{-3} = -\frac{38}{213} \end{aligned}\right\} \Rightarrow$$

$$\Rightarrow\; \boldsymbol{x}^* = \frac{1}{213}\begin{pmatrix} -38 \\ -928 \\ -336 \\ 76 \end{pmatrix} \approx \begin{pmatrix} -0.1784 \\ -4.3568 \\ -1.5775 \\ 0.3568 \end{pmatrix}, \qquad \det(A) = \det(P)\det(R) = (-1)\cdot(-3)\frac{\cancel{13}}{\cancel{30}_{10}}\cdot\frac{\cancel{153}_{3}}{\cancel{260}_{20}}\cdot\frac{71}{\cancel{51}} = \frac{213}{200} \approx 1.0650\,.$$

Nota. Gli elementi sottodiagonali della matrice finale L sono esattamente gli elementi riportati in trasparenza nella parte strettamente triangolare inferiore nella matrice $A^{(4)}$: l'utilità della strategia di memorizzare i moltiplicatori al posto degli elementi resi nulli delle trasformazioni di Gauss è esattamente questa, cioè fornire direttamente la parte strettamente sottodiagonale della matrice triangolare inferiore finale L.

Si vede immediatamente che il sistema può essere risolto anche con strategia diagonale perché:

$$a_{1,1} = 2 \neq 0\,, \quad \begin{vmatrix} 2 & 0 \\ -1 & 1/2 \end{vmatrix} = 1 \neq 0\,, \quad \begin{vmatrix} 2 & 0 & 0 \\ -1 & 1/2 & 0 \\ 3/10 & 0 & 3/5 \end{vmatrix} = \frac{3}{5} \neq 0\,.$$

Per la stima del numero di condizionamento, scegliendo ad esempio la norma infinito, si possono usare le seguenti formule:

$$\mu_\infty(A) \approx \|A\|_\infty \frac{\|\boldsymbol{e}^*\|_\infty}{\|\boldsymbol{r}^*\|_\infty} \qquad \text{oppure} \qquad \mu_\infty(A) \approx 10^t \frac{\|\boldsymbol{e}^*\|_\infty}{\|\boldsymbol{w}^*\|_\infty}$$

dove $\boldsymbol{r}^* = \boldsymbol{b} - A\boldsymbol{x}^*$ è il residuo, $\boldsymbol{e}^*$ è la soluzione effettivamente calcolata di $A\boldsymbol{x} = \boldsymbol{r}^*$, $\boldsymbol{w}^* \approx \boldsymbol{x}^*$ è la soluzione effettivamente calcolata di $A\boldsymbol{x} = \boldsymbol{b}$ e, infine, t è il numero di cifre decimali in base 10 di accuratezza nell'aritmetica finita utilizzata per fare i calcoli. Le stesse formule valgono anche per le altre norme naturali.

Matlab

Contenuto dell'M-script file `esercizio3.m`:

```
% Cognome Nome
% Matricola
%----------------------------
%  esercizio 3 - 19/01/2023
%----------------------------
close all; clear all; clc;
disp('Esercizio 3');
```

```
A = [2 0 0 1; -1 0.5 0 0; 0.3 0 0.6 0; -3 0.2 -0.1 0.5]; b = [0, -2, -1, 0]';
xTeoria = [-38, -928, -336, 76]'/213;

% prima parte
[L, R, p, detA] = gauss2(A);
x1 = rtrisol( R, ltrisol(L, b(p)) )
% oppure:  y1 = sollower(L, b(p)); x1 = solupper(R, y1)
disp('Massima differenza in modulo dalla soluzione teorica:');
disp( max( abs( x1 - xTeoria ) ) );
disp('Residuo normalizzato (in norma infinito):');
disp( (b - A*x1) / norm(b, inf) );
x2 = A \ b  % soluzione calcolata dalla funzione standard di Matlab

% seconda parte
B = A * A';
disp('Fattorizzazione di Cholesky:');
[Lchol, flag] = chol(B, 'lower') % Lchol e' triangolare inferiore
if ( ~flag )
    fprintf('\nB e'' definita positiva\n');
    c  = [8/3, -0.2, 0.3, -9/5]';
    x3 = rtrisol( Lchol', ltrisol(Lchol, c) )
else
    error('\nB non e'' definita positiva\n');
end
```

Eseguendo lo script si ottiene il seguente output:

```
Esercizio 3

x1 =
   -0.1784
   -4.3568
   -1.5775
    0.3568

Massima differenza in modulo dalla soluzione teorica:
   5.5511e-17

Residuo normalizzato (in norma infinito):
   1.0e-16 *
    0.2776
         0
   -0.5551
    0.2776

x2 =
   -0.1784
   -4.3568
   -1.5775
    0.3568
```

```
Fattorizzazione di Cholesky:

Lchol =
    2.2361         0         0         0
   -0.8944    0.6708         0         0
    0.2683   -0.0894    0.6083         0
   -2.4597    1.3416   -0.2959    1.1672

flag =
     0

B e' definita positiva

x3 =
    1.1778
    2.7465
    0.0606
   -0.4063
```

Soluzione esercizio 4

Teoria

La matrice A è **non tridiagonale** e **non simmetrica**. Per la matrice A, i dischi di Gershgorin per righe sono (figura 4):

$$\mathcal{H}_1 = \left\{ x \in \mathbb{C} \,\middle|\, |x - 8/3| \leq 3/2 \right\}, \qquad \mathcal{H}_2 = \left\{ x \in \mathbb{C} \,\middle|\, |x - 7/3| \leq 9/10 \right\}, \qquad \mathcal{H}_3 = \left\{ x \in \mathbb{C} \,\middle|\, |x - 4| \leq 14/15 \right\},$$

mentre i dischi per colonne sono

$$\mathcal{K}_1 = \left\{ x \in \mathbb{C} \,\middle|\, |x - 8/3| \leq 1 \right\}, \qquad \mathcal{K}_2 = \left\{ x \in \mathbb{C} \,\middle|\, |x - 7/3| \leq 4/3 \right\}, \qquad \mathcal{K}_3 = \left\{ x \in \mathbb{C} \,\middle|\, |x - 4| \leq 1 \right\}.$$

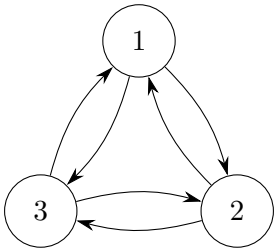
Pertanto (si veda la figura 4):

$$\mathcal{U}_r = \bigcup_{j=1}^{3} \mathcal{H}_j = \mathcal{H}_1 \cup \mathcal{H}_3\,, \quad \mathcal{U}_c = \bigcup_{j=1}^{3} \mathcal{K}_j = \mathcal{K}_2 \cup \mathcal{K}_3 \quad \Rightarrow \quad \mathcal{S} \text{ è descritta in dettaglio nella didascalia della fig. 4.}$$

Si vede immediatamente che A è una matrice strettamente diagonale dominante sia per righe, perché $|a_{i,i}| > \sum_{j\neq i} |a_{i,j}|\ \forall i = 1,2,3$, che per colonne, perché, $|a_{j,j}| > \sum_{i\neq j} |a_{i,j}|\ \forall j = 1,2,3$.

Pertanto, essendo verificata la condizione di stretta diagonale dominanza almeno per righe, i metodi Jacobi e di Gauss-Seidel sono entrambi convergenti e $\|\mathcal{G}\|_\infty < \|J\|_\infty$ (ma da questo non possiamo concludere che uno dei due sia asintoticamente più veloce dell'altro, in quanto la disuguaglianza delle norme **non** implica che $\rho(\mathcal{G}) < \rho(J)$). Inoltre, la matrice A **è invertibile**, in quanto strettamente diagonale dominante per righe (e dunque certamente 0 **non** appartiene alla regione $\mathcal{S}$).

La matrice è non tridiagonale, quindi per determinare la riducibilità o meno di A è necessario costruire il grafo della matrice che, non avendo A alcun elemento nullo, sarà strettamente connesso:



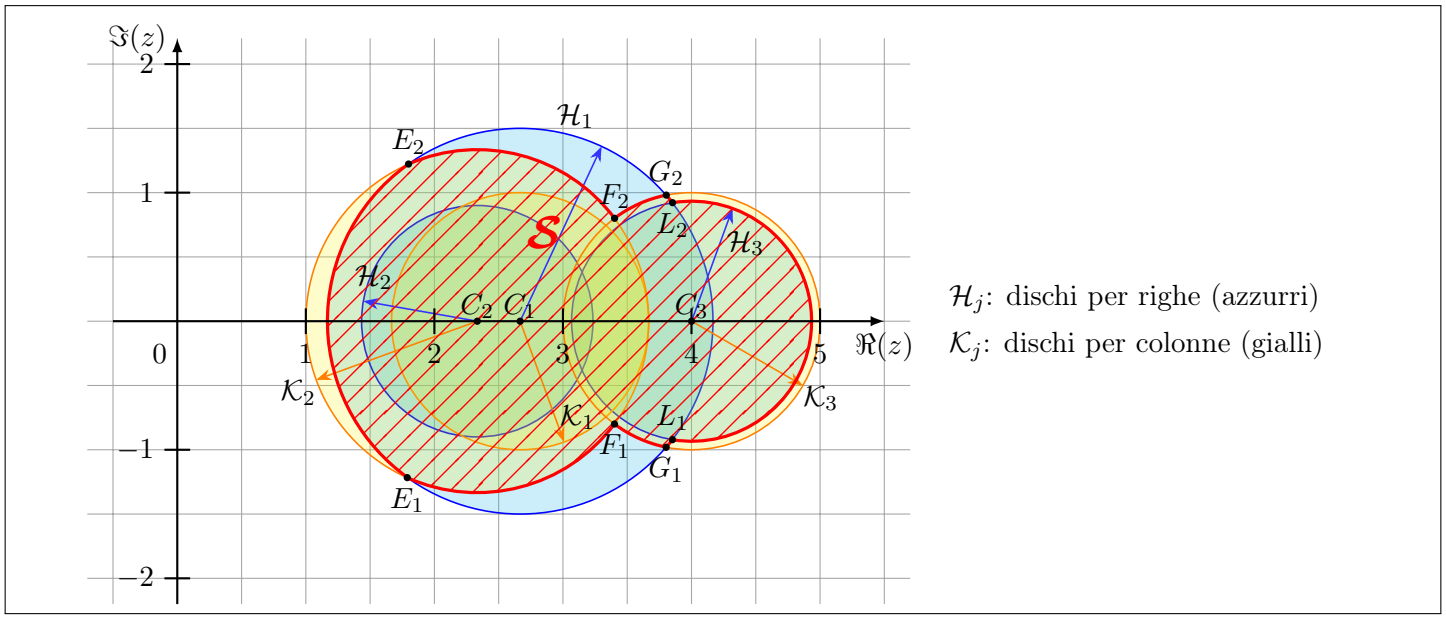


Figura 4: dischi di Ghershgorin per righe ($\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, in azzurro) e per colonne ($\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, in giallo) della matrice A dell'Esercizio 4. La regione tratteggiata $\mathcal{S}$ contiene *tutti* gli autovalori di A. Si noti che il bordo della regione $\mathcal{S}$ è composto da **otto diversi archi** di cerchio: i punti E_1 ed E_2 sono i punti di intersezione fra la circonferenza del disco $\mathcal{H}_1$ (di centro $C_1 = (8/3, 0) \approx (2.667, 0)$ e raggio $3/2 = 1.5$) e quella del disco $\mathcal{K}_2$ (di centro $C_2 = (7/3, 0) \approx (2.333, 0)$ e raggio $4/3 \approx 1.333$); i punti F_1 ed F_2 sono i punti di intersezione fra la circonferenza del disco $\mathcal{K}_2$ (di centro $C_2 = (7/3, 0)$ e raggio $4/3$) e quella del disco $\mathcal{K}_3$ (di centro $C_3 = (4, 0)$ e raggio 1); i punti G_1 e G_2 sono i punti di intersezione fra la circonferenza del disco $\mathcal{H}_1$ (di centro $C_1 = (8/3, 0)$ e raggio 1.5) e quella del disco $\mathcal{K}_3$ (di centro $C_3 = (4, 0)$ e raggio 1); i punti L_1 e L_2 sono i punti di intersezione fra la circonferenza del disco $\mathcal{H}_1$ (di centro $C_1 = (8/3, 0)$ e raggio 1.5) e quella del disco $\mathcal{H}_3$ (di centro $C_3 = (4, 0)$ e raggio $14/15 \approx 0.933$); Percorrendo dunque il bordo di $\mathcal{S}$ in senso orario a partire da E_1, l'arco di cerchio che va da E_1 a E_2 appartiene alla circonferenza che è la frontiera del disco $\mathcal{H}_1$, l'arco di cerchio che va da E_2 a F_2 appartiene alla circonferenza che è la frontiera del disco $\mathcal{K}_2$, l'arco di cerchio che va da F_2 a G_2 appartiene alla circonferenza che è la frontiera del disco $\mathcal{K}_3$, l'arco di cerchio che va da G_2 a L_2 appartiene alla circonferenza che è la frontiera del disco $\mathcal{H}_1$, l'arco di cerchio che va da L_2 a L_1 appartiene alla circonferenza che è la frontiera del disco $\mathcal{H}_3$, l'arco di cerchio che va da L_1 a G_1 appartiene alla circonferenza che è la frontiera del disco $\mathcal{H}_1$, l'arco di cerchio che va da G_1 a F_1 appartiene alla circonferenza che è la frontiera del disco $\mathcal{K}_3$, infine, l'arco di cerchio che va da F_1 a E_1 appartiene alla circonferenza che è la frontiera del disco $\mathcal{K}_2$.

Si ha immediatamente conferma che il grafo di A **è strettamente connesso**, da cui discende l'irriducibilità di A.

La matrice A è **non simmetrica**, quindi **non** possiamo applicare i teoremi che legano i metodi di Jacobi e Gauss-Seidel alle proprietà di definita positività (di A e di $2D - A$, con $D = \mathrm{diag}(A)$).

Essendo A **non tridiagonale**, **non** possiamo utilizzare i risultati che legano la convergenza del metodo di Jacobi a quella del metodo SOR e, inoltre, **non** possiamo calcolare esplicitamente il parametro ottimale di rilassamento ω^* di SOR ed il corrispondente raggio spettrale $\rho\big(\mathcal{G}(\omega^*)\big)$.

Lo *splitting* della matrice A è

$$D = \begin{pmatrix} 8/3 & 0 & 0 \\ 0 & 7/3 & 0 \\ 0 & 0 & 4 \end{pmatrix}, \quad L = \begin{pmatrix} 0 & 0 & 0 \\ -2/5 & 0 & 0 \\ -3/5 & 1/3 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & 1 & -1/2 \\ 0 & 0 & -1/2 \\ 0 & 0 & 0 \end{pmatrix}$$

Determiniamo la matrice J di iterazione di Jacobi:

$$J = D^{-1}(L+U) = I_3 - D^{-1}A = \begin{pmatrix} 0 & 3/8 & -3/16 \\ -6/35 & 0 & -3/14 \\ -3/20 & 1/12 & 0 \end{pmatrix} \approx \begin{pmatrix} 0.0000 & 0.3750 & -0.1875 \\ -0.1714 & 0.0000 & -0.2143 \\ -0.1500 & 0.0833 & 0.0000 \end{pmatrix}.$$

Vediamo che **non vale** la condizione $J \geq 0$: ne discende che **non** sono verificate la ipotesi del Teorema di Stein-Rosenberg.

Calcoliamo ora la matrice di iterazione di Gauss-Seidel. Per prima cosa occorre determinare la matrice inversa di $D - L$. In questo caso l'applicazione dell'algoritmo di Gauss-Jordan è molto semplice e veloce:

$$\begin{aligned} T = \big[D - L \,\big|\, I_3\big] &= \left(\begin{array}{ccc|ccc} 8/3 & 0 & 0 & 1 & 0 & 0 \\ 2/5 & 7/3 & 0 & 0 & 1 & 0 \\ 3/5 & -1/3 & 4 & 0 & 0 & 1 \end{array}\right) \xrightarrow[T_{3,*} \leftarrow T_{3,*} - (9/40)T_{1,*}]{T_{2,*} \leftarrow T_{2,*} - (3/20)T_{1,*}} \left(\begin{array}{ccc|ccc} 8/3 & 0 & 0 & 1 & 0 & 0 \\ 0 & 7/3 & 0 & -3/20 & 1 & 0 \\ 0 & -1/3 & 4 & -9/40 & 0 & 1 \end{array}\right) \\ &\xrightarrow{T_{3,*} \leftarrow T_{3,*} + (1/7)T_{2,*}} \left(\begin{array}{ccc|ccc} 8/3 & 0 & 0 & 1 & 0 & 0 \\ 0 & 7/3 & 0 & -3/20 & 1 & 0 \\ 0 & 0 & 4 & -69/280 & 1/7 & 1 \end{array}\right) \\ &\xrightarrow[T_{3,*} \leftarrow (1/4)T_{3,*}]{T_{1,*} \leftarrow (3/8)T_{1,*}} \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 3/8 & 0 & 0 \\ 0 & 1 & 0 & -9/140 & 3/7 & 0 \\ 0 & 0 & 1 & -69/1120 & 1/28 & 1/4 \end{array}\right) = \big[I_3 \,\big|\, (D-L)^{-1}\big] \end{aligned}$$

$$\Rightarrow \quad (D-L)^{-1} = \begin{pmatrix} 3/8 & 0 & 0 \\ -9/140 & 3/7 & 0 \\ -69/1120 & 1/28 & 1/4 \end{pmatrix} \approx \begin{pmatrix} 0.3750 & 0.0000 & 0.0000 \\ -0.0643 & 0.4286 & 0.0000 \\ -0.0616 & 0.0357 & 0.2500 \end{pmatrix}$$

Possiamo ora calcolare la matrice d'iterazione di Gauss-Seidel:

$$\begin{aligned} \mathcal{G} = (D-L)^{-1}U &= \begin{pmatrix} 3/8 & 0 & 0 \\ -9/140 & 3/7 & 0 \\ -69/1120 & 1/28 & 1/4 \end{pmatrix} \begin{pmatrix} 0 & 1 & -1/2 \\ 0 & 0 & -1/2 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 3/8 & -3/16 \\ 0 & -9/140 & -51/280 \\ 0 & -69/1120 & 29/2240 \end{pmatrix} \\ &\approx \begin{pmatrix} 0.0000 & 0.3750 & -0.1875 \\ 0.0000 & -0.0643 & -0.1821 \\ 0.0000 & -0.0616 & 0.0129 \end{pmatrix}. \end{aligned}$$

Infine, esprimiamo la matrice di iterazione $\mathcal{G}(\omega)$ del metodo SOR come prodotto delle due matrici triangolari, per un generico valore di $\omega \in]0, 2[$:

$$\mathcal{G}(\omega) = (D - \omega L)^{-1}\big((1-\omega)D + \omega U\big) = \begin{pmatrix} 8/3 & 0 & 0 \\ (2/5)\omega & 7/3 & 0 \\ (3/5)\omega & (-1/3)\omega & 4 \end{pmatrix}^{-1} \begin{pmatrix} (8/3)(1-\omega) & \omega & (-1/2)\omega \\ 0 & (7/3)(1-\omega) & (-1/2)\omega \\ 0 & 0 & 4(1-\omega) \end{pmatrix}.$$

Opzionale. Determiniamo il polinomio caratteristico $p_3^{(J)}(\lambda)$ della matrice J:

$$\begin{aligned} p_3^{(J)}(\lambda) = \det\big(J - \lambda I_3\big) &= \begin{vmatrix} -\lambda & 3/8 & -3/16 \\ -6/35 & -\lambda & -3/14 \\ -3/20 & 1/12 & -\lambda \end{vmatrix} \\ &= -\lambda^3 + \left\{ \left(-\frac{\cancel{6}}{35}\right) \left(\frac{1}{\cancel{12}_2}\right) \left(-\frac{3}{16}\right) + \left(-\frac{3}{20}\right) \left(\frac{3}{8}\right) \left(-\frac{3}{14}\right) \right\} \end{aligned}$$

$$
\begin{aligned}
&\quad - \left\{ \left(-\frac{3}{20}\right)\left(-\frac{3}{16}\right) + \left(-\frac{6}{35}\right)\left(\frac{3}{8}\right) + \left(\frac{1}{\cancel{12}_4}\right)\left(-\frac{\cancel{3}}{14}\right) \right\} (-\lambda) \\
&= -\lambda^3 - \frac{1}{2240}\Big(121\lambda - 33\Big) \approx -\lambda^3 - 0.0540\lambda + 0.0147
\end{aligned}
$$

Determiniamo il polinomio caratteristico $p_3^{(\mathcal{G})}(\lambda)$ della matrice $\mathcal{G}$:

$$
\begin{aligned}
p_3^{(\mathcal{G})}(\lambda) = \det\big(\mathcal{G} - \lambda I_3\big) &= \begin{vmatrix} -\lambda & 3/8 & -3/16 \\ 0 & (-9/140) - \lambda & -51/280 \\ 0 & -69/1120 & (29/2240) - \lambda \end{vmatrix} \\
&= -\lambda\left(-\frac{9}{140} - \lambda\right)\left(\frac{29}{2240} - \lambda\right) - (-\lambda)\left(-\frac{51}{280}\right)\left(-\frac{69}{1120}\right) \\
&= -\lambda\left(\lambda^2 + \frac{115}{2240}\lambda - \frac{261}{140\cdot 2240} - \frac{3519}{280\cdot 1120}\right) \\
&= -\lambda\left(\lambda^2 + \frac{115}{2240}\lambda - \frac{\cancel{3780}_{27}}{\cancel{140}\cdot 2240}\right) \\
&= -\frac{\lambda}{2240}\left(2240\lambda^2 + 115\lambda - 27\right) \approx -\lambda\left(\lambda^2 + 0.0513\lambda - 0.0121\right)
\end{aligned}
$$

Nota (non richiesto dat testo dell'esercizio). È immediato calcolare le radici di $p_3^{(\mathcal{G})}(\lambda)$: $\lambda_1(\mathcal{G}) = 0$,

$$
\lambda_{2,3}(\mathcal{G}) = \frac{-115 \mp \sqrt{115^2 - 4\cdot 2240\cdot(-27)}}{2\cdot 2240} = \frac{-115 \mp \sqrt{255145}}{4480} = \frac{-115 \mp 505.1188}{4480}
$$

$$
\Rightarrow \qquad \lambda_2(\mathcal{G}) = \frac{-115 - 505.1188}{4480} \approx -0.1384, \quad \lambda_3(\mathcal{G}) = \frac{-115 + 505.1188}{4480} \approx 0.0871
$$

da cui otteniamo che

$$
\rho(\mathcal{G}) = \max_{j=1,2,3}\big|\lambda_j(\mathcal{G})\big| = \big|\lambda_2(\mathcal{G})\big| \approx 0.1384 < 1 \quad \Rightarrow \quad R_\infty(\mathcal{G}) = -\ln\big(\rho(\mathcal{G})\big) \approx -\ln\big(0.0871\big) \approx 1.98
$$

che confermano la convergenza del metodo di Gauss-Seidel. D'altro canto, non è immediato calcolare le radici di $p_3^{(J)}(\lambda)$. Utilizzando la funzione `roots` di Matlab possiamo, in due righe di codice, calcolare radici, raggio spettrale e velocità asintotica di convergenza per il metodo di Jacobi:

```
>> p3J = [-1, 0, -121/2240, 33/2240], eigsJ = roots( p3J )
>> rhoJ = max( abs( eigsJ ) ), RinftyJ = -log( rhoJ )
```

che danno tre autovalori (di cui due complessi coniugati), $\rho(J) \approx 0.2906 < 1$ e $R_\infty(J) \approx 1.24$, confermando la convergenza del metodo di Jacobi e la sua minor velocità asintotica rispetto al metodo di Gauss-Seidel. Volendo, è immediato controllare anche i risultati ottenuti sopra per il metodo di Gauss-Seidel con due righe di codice Matlab analoghe alla precedenti:

```
>> p3GS = [-1, -115/2240, 27/2240, 0], eigsGS = roots( p3GS )
>> rhoGS = max( abs( eigsGS ) ), RinftyGS = -log( rhoGS )
```

I risultati coincidono con quelli trovati sopra.

Matlab

Contenuto dell'M-script file `esercizio4.m`:

```
% Cognome Nome
% Matricola
%----------------------------
%  esercizio 4 - 19/01/2023
%----------------------------
close all; clear all; clc;
disp('Esercizio 4');

A = [8/3 -1 1/2; 2/5 7/3 1/2; 3/5 -1/3 4]; b = [1, -5, -2]';

d = diag( A );
```

```
if any( ~d ), error('presenza di elementi diagonali nulli in A'); end
invD = diag( 1 ./ d );
J    = -invD * (tril(A,-1) + triu(A,1));
cJ   = b ./ d;
GS   = -tril(A) \ triu(A,1);

eigsJ = eig( J ); rhoJ =  max( abs(eigsJ) ); velJ = -log( rhoJ );
eigsGS = eig( GS ); rhoGS =  max( abs(eigsGS) ); velGS = -log( rhoGS );

fprintf('\nAutovalori della matrice J di Jacobi:\n'); disp(eigsJ);
fprintf('Raggio spettrale di J e velocita'' asintotica di convergenza:\n');
fprintf('\trhoJ = %g,\t\tRinftyJ = %g\n', rhoJ, velJ);

fprintf('\nAutovalori della matrice GS di Gauss-Seidel:\n'); disp(eigsGS);
fprintf('Raggio spettrale di GS e velocita'' asintotica di convergenza:\n');
fprintf('\trhoGS = %g,\t\tRinftyGS = %g\n', rhoGS, velGS);

x0 = [1; -2; 1]; maxit = 50; tol = 1.0e-5;

% Soluzione con il metodo di Jacobi, utilizzando la M-function fornita
fprintf('\nApproxs. xJ calcolata con il metodo di Jacobi e numero di iterazioni:')
[xJ, iterJ] = jacobi(A, b, x0, maxit, tol)

% Metodo SOR con omega = 1.27
omega = 1.27;
SOR = (diag(d) + omega*tril(A, -1)) \ [((1 - omega)*diag(d) - omega*triu(A, 1)), b];
cSOR = SOR(:,end); SOR(:, end) = [];
fprintf('\nMatrice del metodo SOR con omega = %g:\n', omega); disp(SOR);
for k = 1 : 2
    xSOR = SOR*x0 + cSOR;
    x0 = xSOR;
end
fprintf('\nSecondo iterato del metodo SOR:\n')
disp(xSOR)
```

Eseguendo lo script, si ottengono le conferme numeriche dei valori calcolati nella parte teorica:

```
Esercizio 4

Autovalori della matrice J di Jacobi:
  -0.0872 + 0.2772i
  -0.0872 - 0.2772i
   0.1744 + 0.0000i

Raggio spettrale di J e velocita' asintotica di convergenza:
        rhoJ = 0.290603,               RinftyJ = 1.2358

Autovalori della matrice GS di Gauss-Seidel:
         0
   -0.1384
    0.0871

Raggio spettrale di GS e velocita' asintotica di convergenza:
        rhoGS = 0.138419,                RinftyGS = 1.97747

Approxs. xJ calcolata con il metodo di Jacobi e numero di iterazioni:
xJ =
   -0.2448
   -1.9665
   -0.6271

iterJ =
    11

Matrice del metodo SOR con omega = 1.27:
   -0.2700    0.4763   -0.2381
    0.0588   -0.3737   -0.2203
    0.0577   -0.1303   -0.2480
```

```
Secondo iterato del metodo SOR:
   -0.1797
   -1.5520
   -0.4981
```

Il metodo di Jacobi si arresta dopo 11 iterazioni con un'approssimazione della soluzione data da $\boldsymbol{x}_{\text{J}} \approx (-0.2448, -1.9665, -0.6271)^T$, che è un'approssimazione a quattro cifre decimali della soluzione $\boldsymbol{x}^* = (-95/388, -763/388, -365/582)^T$ del sistema $A\boldsymbol{x} = \boldsymbol{b}$ (questa può facilmente essere calcolata da linea di comando con `xs = A\b` e impostando il formato di visualizzazione a numeri razionali mediante `format rat`).

D'altro canto, l'approssimazione raggiunta dal metodo SOR con $\omega = 1.27$ dopo 2 iterazioni partendo dallo stesso punto iniziale $\boldsymbol{x}^{(0)} = (1, -2, 1)^T$ è $\boldsymbol{x}^{(2)}_{\text{SOR}} \approx (0.0631, -1.5137, -0.4733)^T$. Si osserva che le componenti di quest'approssimazione si stanno avvicinando a quelle della soluzione $\boldsymbol{x}^*$, ma la loro accuratezza è ancora non buona: l'errore relativo percentuale in norma euclidea per questa approssimazione è infatti ancora $\|\boldsymbol{x}^{(2)}_{\text{SOR}} - \boldsymbol{x}^*\|_2 / \|\boldsymbol{x}^*\|_2 \approx 27.4\%$.

Soluzione esercizio 5

Teoria

Notiamo innanzitutto che la funzione è ben definita nell'intervallo $[a, b] = [0, 4]$ perchè il radicando è sempre positivo. Dato che si sta cercando un polinomio interpolante di grado al più $n = 3$, i punti di interpolazione devono essere $n + 1 = 4$. Costruiamo il polinomio interpolante per la funzione $f(x)$ sull'intero intervallo $[0, 4]$. Siano t_j i nodi di Chebychev in $[-1, 1]$ e w_j i nodi di Chebychev in $[0, 4]$, $j = 0, 1, 2, 3$, ottenuti usando la trasformazione lineare che mappa $[-1, 1]$ in $[a, b]$:

$$t_{3-j} = \cos\left(\frac{2j+1}{2(n+1)}\pi\right) \quad \forall j = 3, 2, 1, 0$$

$$\Rightarrow \quad t_0 = \cos(7\pi/8) \approx -0.9239\,,\ t_1 = \cos(5\pi/8) \approx -0.3827\,,\ t_2 = \cos(3\pi/8) = -\cos(5\pi/8) \approx 0.3827\,,$$
$$t_3 = \cos(\pi/8) = -\cos(7\pi/8) \approx 0.9239\,,$$

$$w_j = \frac{a+b}{2} + \frac{b-a}{2} t_j = 2 + 2t_j = 2(1 + t_j) \quad \forall j = 0, 1, 2, 3 \quad \Rightarrow \quad \boldsymbol{w} \approx (0.1522,\ 1.2346,\ 2.7654,\ 3.8478)^T\,.$$

Calcoliamo ora per questi quattro punti di interpolazione la relativa tabella delle differenze divise:

w_k	$f(w_k)$	$f[w_k, w_{k+1}]$	$f[w_0, w_1, w_2]$	$f[w_0, w_1, w_2, w_3]$
0.1522	1.8807			
1.2346	1.9618	0.0749		
2.7654	2.9589	0.6514	0.2206	
3.8478	2.2843	−0.6232	−0.4877	−0.1917

Dalla tabella ricaviamo l'espressione del polinomio di interpolazione nella forma di Newton:

$$\begin{aligned} p_3^{\text{C}}(x) &\approx 1.8807 + 0.0749(x - w_0) + 0.2206(x - w_0)(x - w_1) + (-0.1917)(x - w_0)(x - w_1)(x - w_2) \\ &\approx 1.8807 + 0.0749(x - 0.1522) + 0.2206(x - 0.1522)(x - 1.2346) \\ &\quad - 0.1917(x - 0.1522)(x - 1.2346)(x - 2.7654)\,. \end{aligned}$$

Svolgendo i calcoli, si arriva molto rapidamente alla forma canonica del polinomio interpolante sui nodi di Chebychev:

$$p_3^{\text{C}}(x) \approx -0.1917x^3 + 1.0165x^2 - 1.0021x + 2.0104\,.$$

Nota. La tabella delle differenze divise e i coefficienti del polinomio $p_3^{\text{C}}(x)$ si ottengono facilmente con poche righe di Matlab. In un ciclo di n iterazioni, con n il grado del polinomio interpolante, ad ogni k-esima iterazione si generano la colonna $(k+2)$-esima della tabella, il prodotto dei polinomi $(x - x_0) \cdots (x - x_k)$ e la forma canonica del polinomio $p_k(x)$ di grado k di interpolazione sui primi $k + 1$ nodi $x_0, \ldots, x_k$. Per il prodotto dei polinomi si usa la funzione predefinita `conv` (*convoluzione*, facilmente rintracciabile nella documentazione di Matlab mediante `help poly`). Sia dunque `y` il vettore colonna che contiene gli $n + 1$ valori della funzione da interpolare (ossia `y` è la seconda colonna della tabella delle differenze divise), partendo da `y(1)` $= f(x_0)$. Allora:

```
d = y; s = [1]; p = zeros(1, (n+1)); p(end) = d(1);
for k = 1 : n
    d((k+1) : end) = diff( d(k : end) ) ./ ( x((k+1) : end) - x(1 : (end-k)) )
    s = conv(s, [1, -x(k)]);
```

```
    p((end-k) : end) = p((end-k) : end) + d(k+1)*s;
end
```

Al termine, il vettore colonna `d` contiene gli elementi diagonali della tabella delle differenze divise (iniziando con `d(1)` $= f(x_0)$), il vettore riga `s` contiene i coefficienti della forma canonica del polinomio $\omega_n(x) = (x - x_0)(x - x_1)\cdots(x - x_{n-1})$ e il vettore riga `p` contiene esattamente i coefficienti della forma canonica del polinomio interpolante $p_n(x)$ di grado n su tutti i nodi $x_0, x_1, \ldots, x_n$.

Evitando di concludere con il punto e virgola la prima riga del ciclo (ed inserendo eventualmente un `pause` al termine del corpo del ciclo), nelle componenti dalla $(k+1)$-esima all'ultima del vettore `d` si leggono i valori da inserire nella $(k+2)$-esima colonna della tabella.

Nel caso del presente esercizio, essendo $n = 3$, per il polinomio interpolante $p_3^{\mathrm{C}}(x)$ i vettori `y`, `d` e `p` hanno quattro componenti e il ciclo effettua tre sole iterazioni. Alla prima iterazione, i valori in `d(2:4)` sono quelli della terza colonna della tabella delle differenze divise, alla seconda iterazione in `d(3:4)` si leggono i due elementi della quarta colonna della tabella e alla terza (e ultima) iterazione il valore in `d(4)` contiene l'unico valore diverso da zero nella quinta (e ultima) colonna della tabella.

L'espressione generale dell'errore di interpolazione di un polinomio interpolante di grado al più n su $n+1$ punti distinti x_j, $j = 0, \ldots, n$, di un generico intervallo $[a, b]$ per una funzione $f \in C^{n+1}\big([a, b]\big)$ si scrive come

$$R_n(x) = \omega_{n+1}(x)\frac{f^{(n+1)}(\xi_x)}{(n+1)!} \qquad \text{con} \quad \xi_x \in]a, b[\quad \text{e} \quad \omega_{n+1}(x) = (x - x_0)(x - x_1)\cdots(x - x_n)\,.$$

Nel caso in esame si ha:

$$f'(x) = \frac{1}{2}(x+4)^{-1/2} - \frac{\pi}{3}\cos(2x)\,, \quad f''(x) = -\frac{1}{4}(x+4)^{-3/2} + \frac{2}{3}\pi\sin(2x)\,,$$
$$f'''(x) = \frac{3}{8}(x+4)^{-5/2} + \frac{4}{3}\pi\cos(2x)\,, \quad f^{(4)}(x) = -\frac{15}{16}(x+4)^{-7/2} - \frac{8}{3}\pi\sin(2x)$$
$$\Rightarrow \; R_3(x) = (x - 0.1522)(x - 1.2346)(x - 2.7654)(x - 3.8478)\frac{1}{4!}\left(-\frac{15}{16}(\xi_x+4)^{-7/2} - \frac{8}{3}\pi\sin(2\xi_x)\right)\,,\ \xi_x \in]0,\,4[\,.$$

I valori in $x^* = 1.5$ della funzione $f(x)$, di $p_3^{\mathrm{C}}(x)$ e della loro differenza sono:

$$f\big(x^*\big) = f(1.5) \approx 2.2713\,,\ p_3^{\mathrm{C}}\big(x^*\big) = p_3^{\mathrm{C}}(1.5) \approx 2.1474 \;\Rightarrow\; f\big(x^*\big) - p_3^{\mathrm{C}}\big(x^*\big) = 2.2713 - 2.1474 \approx 0.1240\,.$$

Il valore dell'espressione dell'errore valutata prendendo $\xi_x = x = x^*$ è ≈ 0.0524

N.B. **Non** possiamo scrivere $R_3\big(x^*\big) \approx 0.0524$ perchè, com'è ben noto, il punto ξ_{x^*} nell'argomento della derivata **è non noto** e, pur dipendendo da x^*, **non è lecito supporre** che sia $\xi_{x^*} = x^*$.

Come si vede, il valore 0.0524 è minore della metà dell'errore vero 0.1240. Tuttavia, per quanto detto nella nota, per stimare correttamente l'errore dovremmo determinare una maggiorazione di $|R_3(x)|$ in $[0, 4]$, determinando una maggiorazione per $\big|f^{(4)}(x)\big|$.

Non richiesto. Al solo scopo di mostrare il corretto modo di procedere, determiniamo questa maggiorazione di $|R_3(x)|$ in $[0, 4]$, che nel caso in esame può essere facilmente calcolata in quanto i nodi di interpolazione **sono i nodi di Chebychev** e quindi conosciamo una maggiorazione anche di $|\omega_4(x)|$.

Dato che $|\sin(2x)| \le 1$, il modulo del secondo addendo di $f^{(4)}(x)$ si maggiora con $8\pi/3 \approx 8.3776$. D'altro canto, dato che $1/\sqrt{x+4}$ è una funzione non negativa e monotona strettamente decrescente in $[0, 4]$, possiamo maggiorarne il modulo con il valore che assume nell'estremo sinistro, cioè in $a = 0$, che è $1/2$. Inoltre, $15/16 \lessapprox 1$. Pertanto, possiamo facilmente maggiorare il modulo del primo addendo di $f^{(4)}(x)$: $(15/16)(x+4)^{-7/2} \le (1/2)^7 = 1/128 \approx 0.0078 \approx 7.8125 \cdot 10^{-3}$. Quindi

$$\big|f^{(4)}(x)\big| \le \left|\frac{15}{16}(x+4)^{-7/2}\right| + \left|\frac{8}{3}\pi\sin(2x)\right| \le \frac{1}{128} + \frac{8}{3}\pi \approx 8.3854 \quad \forall x \in [0, 4]\,.$$

D'altro canto, avendo considerato i nodi di Chebychev come nodi di interpolazione, sappiamo che $\|\omega_{n+1}(x)\|_\infty = \max_{x\in[a,b]}\big|\omega_{n+1}(x)\big| = 2(b-a)^{n+1}/4^{n+1}$: quindi, nel caso in esame, $\|\omega_4(x)\|_\infty = \max_{x\in[0,4]}\big|\omega_4(x)\big| = 2(4)^4/4^4 = 2$. Concludiamo allora che

$$|R_3(x)| \le \|\omega_4(x)\|_\infty \frac{\max_{x\in[0,4]}\big|f^{(4)}(x)\big|}{4!} \le 2\frac{1/128 + 8\pi/3}{24} \approx 2 \cdot 3.4939 \cdot 10^{-1} \approx 0.6988 \quad \forall x \in [0, 4]$$

che è effettivamente una maggiorazione dell'errore vero $R_3(x^*) = 0.1240$ in $x^* = 1.5 \in [0, 4]$.

Infine, gli $N = 3$ nodi equispaziati di interpolazione in $[a, b]$ e i corrispondenti valori di f e sono:

$$h = \frac{b-a}{N-1} = \frac{4-0}{2} = 2\,, \quad x_j = a + jh \quad \forall j = 0, 1, 2$$
$$\Rightarrow \quad \boldsymbol{x} = \big(a,\, (a+b)/2,\, b\big)^T = \big(0,\, 2,\, 4\big)^T \quad \Rightarrow \quad f(\boldsymbol{x}) \approx (2.0000,\, 2.8458,\, 2.3104)^T$$

Le espressioni dei tre polinomi di Lagrange per questi punti sono:

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-2)(x-4)}{(0-2)(0-4)} = \frac{1}{8}\big(x^2 - 6x + 8\big) = \frac{1}{8}x^2 - \frac{3}{4}x + 1\,,$$
$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_0-x_2)} = \frac{(x-0)(x-4)}{(2-0)(2-4)} = -\frac{1}{4}\big(x^2 - 4x\big) = -\frac{1}{4}x^2 + x\,,$$
$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-0)(x-2)}{(4-0)(4-2)} = \frac{1}{8}\big(x^2 - 2x\big) = \frac{1}{8}x^2 - \frac{1}{4}x\,.$$

La formula del polinomio interpolante $p_2(x)$ sui tre punti equistaziati mediante i polinomi di Lagrange è:

$$\begin{aligned} p_2(x) &= \sum_{i=0}^{2} f(x_i) L_i(x) \\ &\approx 2\Big((1/8)x^2 - (3/4)x + 1\Big) + 2.8458\Big((-1/4)x^2 + x\Big) + 2.3104\Big((1/8)x^2 - (1/4)x\Big)\,. \end{aligned}$$

Non richiesto dal testo dell'esercizio: nel caso in esame, è molto facile svolgere i calcoli per i polinomi di Lagrange, ottenendo l'espressione canonica del polinomio interpolante $p_2(x)$ sui 3 nodi equispaziati:

$$L_0(x) = \frac{1}{8}x^2 - \frac{3}{4}x + 1 = 0.125x^2 - 0.75x + 1\,,$$
$$L_1(x) = -\frac{1}{4}x^2 + x = -0.25x^2 + x\,,$$
$$L_2(x) = \frac{1}{8}x^2 - \frac{1}{4}x = 0.125x^2 - 0.25x\,,$$
$$\begin{aligned} p_2(x) &= \sum_{i=0}^{2} f(x_i) L_i(x) \approx 0.25x^2 - 1.5x + 2 - 0.7114x^2 + 2.8458x + 0.2888x^2 - 0.5776x \\ &\approx -0.1726x^2 + 0.7682x + 2\,. \end{aligned}$$

Possiamo controllare molto rapidamente questo risultato in due righe di codice Matlab, sfruttando la funzione predefinita `polyfit`:

```
>> funz = @(x)( sqrt(x + 4) - (pi/6)*sin(2*x) ); xEq = linspace(0,4,3); yEq = funz( xEq );
>> p2Eq = polyfit(xEq, yEq, 2)
```

che fornisce come risultato `p2Eq =  -0.1726  0.7682  2.0000`.

Matlab

Contenuto dell'M-script file `esercizio5.m`:

```
% Cognome Nome
% Matricola
%----------------------------
%  esercizio 5 - 19/01/2023
%----------------------------
close all; clear all; clc;
disp('Esercizio 5');

a = -2; b = 5; n = 5; npiu1 = n + 1; N = 201;
cost = pi/6;
funz = @(x)( sqrt(x + 4) - cost*sin(2*x) );
nodiChebychev = cos( (2*[n : -1 : 0]+1)'*pi / (2*n+2) );
puntiChebychev = 0.5*( nodiChebychev*(b - a) + (a + b) );
fPuntiChebychev = funz( puntiChebychev );
xx = linspace(a, b, N);
```

```
ff = funz( xx );

[yy, coeffNewton] = polyNewton( puntiChebychev, fPuntiChebychev, xx );

errRelativo = abs(yy - ff) ./ abs(ff);

f1 = figure;
ph1 = plot(xx, ff, '-b', xx, yy, '-m', 'Linewidth', 1.5);
hold on
ph2 = plot( [a, b]', [0, 0]', '-k', ... % asse delle ascisse
            puntiChebychev, zeros(size(puntiChebychev)), '.k', ...
            puntiChebychev, fPuntiChebychev, '.r',  'MarkerSize', 16 );
lh = line([puntiChebychev, puntiChebychev]',[zeros(1,npiu1); fPuntiChebychev'], ...
        'LineStyle', '--', 'Color', 'k'); % non richiesto
hold off
set(gca, 'YLim', [-0.5, 3.5]); % non indispensabile
th(1) = title(sprintf('Interpolazione di grado %d su punti di Chebychev', n));
xh(1) = xlabel('x');
yh(1) = ylabel('y = f(x)');
lh = legend(ph1, {'funzione f(x)', 'polinomio interpolante'}, ...
                 'Location', 'northwest');

f2 = figure;
ph3 = semilogy(xx, errRelativo, '-r');
th(2) = title(sprintf(...
        'Errore relativo dell''interpolante di grado %d su punti di Chebychev', n));
xh(2) = xlabel('x');
yh(2) = ylabel('errore rel. = |pn(x) - f(x)| / |f(x)|');
```

L'esecuzione dell'M-script genera i grafici in fig. 5 (qui leggermente migliorati per esigenze di stampa).

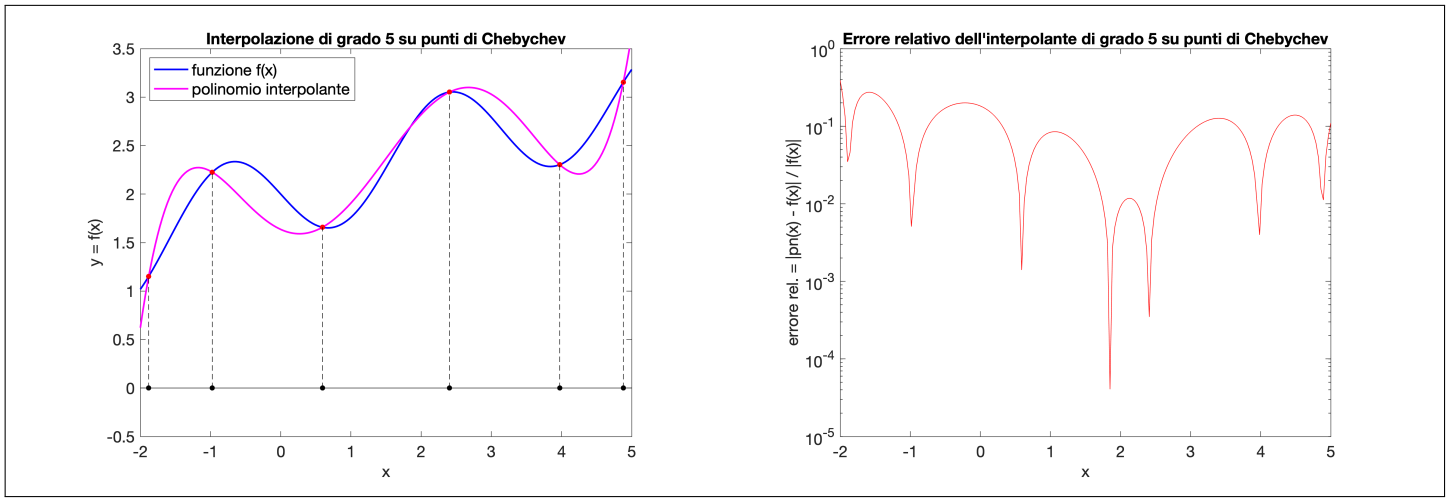


Figura 5: grafici generati dall'M-script `esercizio5.m` (figure leggermente migliorate per esigenze di stampa).

Soluzione esercizio 6

Teoria

Consideriamo dunque la funzione $g(x) = 2.2 - f(x) = 2.2 - \sqrt{x+4} + (\pi/6)\sin(2x)$ nell'intervallo $[a, b] = [0, 3]$. Le condizioni da verificare sono quelle del Teorema degli zeri, ossia che $g(x)$ sia una funzione continua su un intervallo chiuso e limitato, in cui assume valori di segno discorde agli estremi. La funzione $g(x)$ è certamente continua in $[0, 3]$, come composta e somma di funzioni ivi continue. Inoltre, l'intervallo $[0, 3]$ è chiuso e limitato. Agli estremi dell'intervallo si ha:

$$g(a) = g(0) = 2.2 - \sqrt{4} + (\pi/6)\sin(0) = 0.2 > 0 \qquad g(b) = g(3) = 2.2 - \sqrt{7} + (\pi/6)\sin(6) \approx -0.5921 < 0\,.$$

Dunque le ipotesi per l'esistenza di almeno uno zero di $g(x)$ in $[a.b]$ sono verificate. Possiamo facilmente avere un'idea del comportamento qualitativo di $g(x)$ in $[0, 3]$ disegnando il grafico in Matlab con la funzione predefinita `fplot` (fig. 6a):

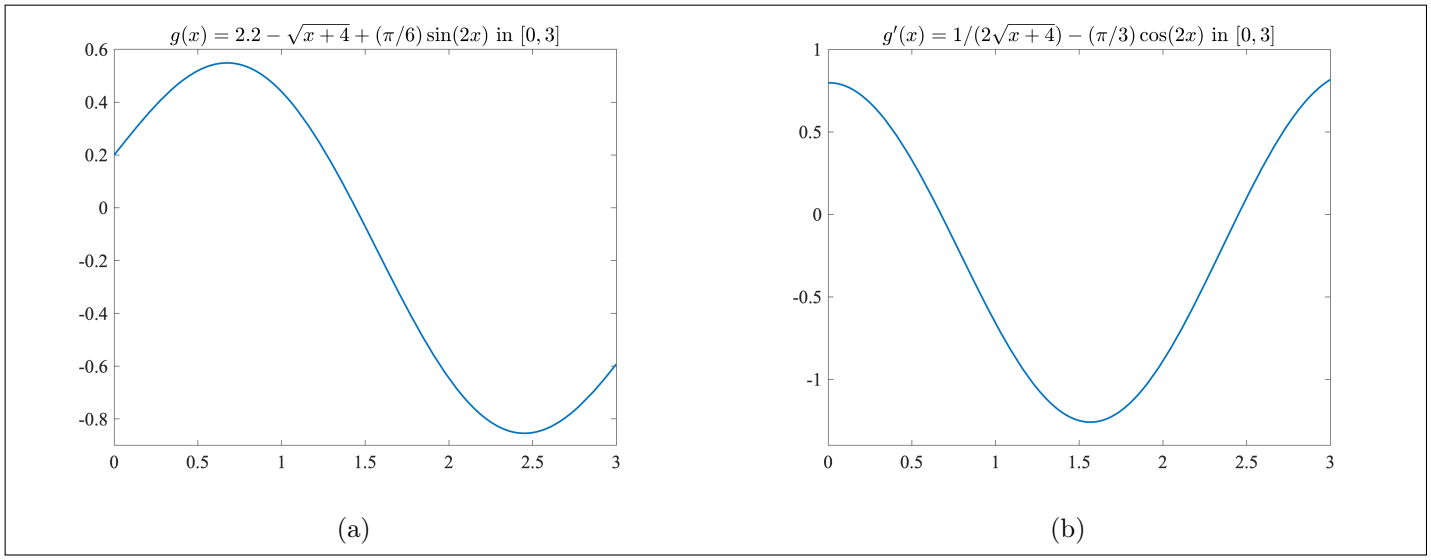


Figura 6: grafici in $[0,3]$ della funzione $g(x) = 2.2 - \sqrt{x+4} + (\pi/6)\sin(2x)$ (a sinistra, pannello (a)) dell'esercizio 6 e della sua derivata prima $g'(x) = -(1/2)(x+4)^{-1/2} + (\pi/3)\cos(2x)$ (a destra, pannello (b)).

```
>> g = @(x)( 2.2 - sqrt(x+4) + (pi/6)*sin(2*x) ); fplot(g, [0, 3]);
```

da cui si deduce facilmente che $g(x)$ ammette un unico zero z^* in $[0,3]$ e che esso si trova nei pressi del punto medio $\overline{x} = 1.5$ di $[a,b]$.

Nota. Possiamo avere molto facilmente una buona stima dello zero z^* di $g(x)$ in [0.3] usando la funzione predefinita `fzero` di Matlab: avendo già inserito la riga di codice che definisce il *function handle* di $g(x)$, le istruzioni `[zs, gzs] = fzero(g, 1.5)` restituiscono $z^* \approx 1.4425$ e $g(z^*) = 0$.

La formula di aggiornamento del metodo di bisezione è:

$$z_k = a_k + \frac{b_k - a_k}{2} \qquad [a_{k+1}, b_{k+1}] = \begin{cases} [a_k, z_k] & \text{se } \operatorname{sgn}\big(g(z_k)\big) = \operatorname{sgn}\big(g(b_k)\big) \\ [z_k, b_k] & \text{se } \operatorname{sgn}\big(g(z_k)\big) = \operatorname{sgn}\big(g(a_k)\big) \end{cases} \qquad k = 0, 1, \ldots$$

con $[a_0, b_0] = [a, b]$. Il numero minimo di iterazioni necessarie per ottenere un'approssimazione $\widetilde{z}$ di uno zero z^* di $g(x)$ in $[a,b]$ con tolleranza non maggiore di $\tau = 10^{-4}$ è

$$K = \left\lceil \log_2 \left(\frac{b-a}{\tau} \right) \right\rceil = \left\lceil \log_2 \left((3-0)10^4 \right) \right\rceil = 15\,.$$

La formula di aggiornamento per il metodo delle secanti è

$$z_{k+1} = z_k - \frac{(z_k - z_{k-1})g(z_k)}{g(z_k) - g(z_{k-1})}$$

sotto l'ipotesi che $g(z_k) \neq g(z_{k-1})$ $\forall k = 0, 1, \ldots$. La formula di aggiornamento per il metodo di Newton è

$$z_{k+1} = z_k - \frac{g(z_k)}{g'(z_k)}$$

sotto l'ipotesi che $g'(x) \neq 0$ $\forall x \in [a,b]$. Qui è $g'(x) = -f'(x) = -(1/2)(x+4)^{-1/2} + (\pi/3)\cos(2x)$. Per capire se $g'(x) \neq 0$ $\forall x \in [0,3]$, possiamo dare uno sguardo al suo grafico mediante la funzione predefinita `fplot` di Matlab: con

```
>> gprime = @(x)( (-0.5 ./ sqrt(x+4)) + (pi/3)*cos(2*x) );
>> fplot(gprime, [0,3]); gprime( [0, 1.5, 3]' )
```

ci accorgiamo immediatamente che $g'(x)$ cambia segno due volte in [0, 3], come mostra la figura 6b. Concludiamo, quindi, che il metodo di Newton è **non ben definito** per $g(x)$ in $[0,3]$.
Tuttavia, non è difficile mostrarlo formalmente: per $x \in [0,3]$, $g'(x)$ è continua e si ha $g'(a) = g'(0) \approx 0.7972 > 0$, $g'\big((a+b)/2\big) = g'(1.5) \approx -1.2499 < 0$, $g'(b) = g'(3) \approx 0.8165 > 0$. Pertanto, si conclude immediatamente che $g'(x)$ cambia segno almeno una volta in $[0, 1.5]$ e almeno un'altra volta in $[1.5, 3]$.

Per calcolare due iterate del metodo di bisezione, partiamo dagli estremi dell'intervallo, in cui abbiamo già i valori della funzione: $a_0 = a = 0$, $g(a_0) = g(a) = 0.2$, $b_0 = b = 3$, $g(b_0) = g(b) \approx -0.5921$

$$\begin{aligned} \text{bisezione :} \qquad z_0 &= a_0 + \frac{1}{2}(b_0 - a_0) = 3/2 = 1.5, \quad g(z_0) \approx -0.0713 \\ &\operatorname{sgn}\big(g(z_0)\big) = \operatorname{sgn}\big(g(b_0)\big) \quad \Rightarrow \quad a_1 = a_0,\ b_1 = z_0,\ g(a_1) = g(a_0),\ g(b_1) = g(z_0) \\ z_1 &= a_1 + \frac{1}{2}(b_1 - a_1) = 3/4 = 0.75, \quad g(z_1) \approx 0.5428 \end{aligned}$$

Per calcolare due iterate del metodo delle secanti, partiamo ancora dagli estremi dell'intervallo: $z_0 = a$, $g(z_0) = g(a)$, $z_1 = b$, $g(z_1) = g(b)$:

$$\begin{aligned} z_2 &= z_1 - \frac{g(z_1)(z_1 - z_0)}{g(z_1) - g(z_0)} \approx 0.7575\,, \qquad g(z_2) \approx 0.5416\,, \\ z_3 &= z_2 - \frac{g(z_2)(z_2 - z_1)}{g(z_2) - g(z_1)} \approx 1.8289\,, \qquad g(z_3) \approx -0.4727\,. \end{aligned}$$

Infine, per calcolare due iterate del metodo di Newton, partiamo dal punto medio dell'intervallo: $z_0 = (a+b)/2 = 3/2 = 1.5$, $g(z_0) \approx -0.0713$, $g'(z_0) \approx -1.2499$,

$$\begin{aligned} z_1 &= z_0 - \frac{g(z_0)}{g'(z_0)} \approx 1.5 - \frac{-0.0713}{-1.2499} \approx 1.4429\,, \qquad g(z_1) \approx -5.7718 \cdot 10^{-4}\,, \qquad g'(z_1) \approx -1.2275\,, \\ z_2 &= z_1 - \frac{g(z_1)}{g'(z_1)} \approx 1.4429 - \frac{-5.7718}{-1.2275} \cdot 10^{-4} \approx 1.4425\,, \qquad g(z_2) \approx -5.6458 \cdot 10^{-8}\,. \end{aligned}$$

Come si nota, nel metodo di Newton l'accuratezza nell'approssimazione di uno zero della funzione migliora molto velocemente e anche il corrispondente valore della funzione si avvicina a zero molto velocemente. In questo caso, infatti, si ha l'**evento fortunato** che il punto iniziale $z_0 = (a+b)/2 = 1.5$ sia già molto vicino all'unico zero $z^* \approx 1.4425$ della funzione in $[0, 3]$ e, inoltre, tale zero è uno zero semplice: infatti, $g'(x) < 0$ in un ampio intervallo che contiene z^* e la funzione $g'(x)$ è continua, pertanto, per il Teorema di permanenza del segno, anche $g'(z^*) < 0$, ossia $g'(z^*) \neq 0$. Siamo dunque nelle ipotesi del Teorema di convergenza quadratica locale del metodo di Newton che, infatti, osserviamo nel fatto che ad ogni singola iterazione si guadagnano (almeno) **due cifre** significative di accuratezza per la stima di z^*.

Nota. I valori riportati fin qui sono tutti approssimati alla quarta cifra decimale: z^* ritornato da `fzero` e z_2 del metodo di Newton coincidono nelle prime cinque cifre significative riportate qui e, di più, coincidono nelle prime otto cifre significative, ma differiscono dalla nona in poi: $z_2 - z^* \approx 4.6 \cdot 10^{-8}$. Per questo motivo, i rispettivi valori della funzione $g(x)$ sono diversi: $g(z_2) \approx -5.6458 \cdot 10^{-8} \neq 0 = g(z^*)$.

Matlab

Contenuto dell'M-script file `esercizio6.m`:

```
% Cognome Nome
% Matricola
%-----------------------------
%  esercizio 6 - 19/01/2023
%-----------------------------
close all; clear all; clc;
disp('Esercizio 6');

rng(5); N = 3; n = 3; nn = 2001;
a = 0; b = 3; myRho = 1.2;

NN = 10*N; n1 = n+1;
delta = myRho*(rand(NN,1) - 0.5);
cost = (pi/6);
f = @(x)( sqrt(x + 4) - cost*sin( 2*x ) );

x = sort(rand(NN,1)*(b - a) + a); y = myRho*f(x) + delta;
A = ones(NN, n1);
for k = n:-1:1
    A(:,k) = A(:,(k+1)) .* x;
```

```
end
B = A'*A; c = A'*y;    [LChol, flag] = chol(B, 'lower');
if ( ~flag )
    zstar = rtrisol( LChol', ltrisol(LChol, c) )
else
    error('Matrice B del sistema delle equazioni normali non definita positiva');
end
wstar = A \ y

xx = linspace(a, b, nn)';  yy = polyval(zstar, xx);  ph1 = plot(xx, yy, '-b');
hold on;
ph2 = plot([a;b],[0;0],'-k', x,zeros(size(x)),'+k', x,y,'.r','MarkerSize',16);
hold off
alh(1) = xlabel('Intervallo delle ascisse');
alh(2) = ylabel('Valori delle osservazioni e del polinomio');
tlh    = title(sprintf(...
         'Polinomio ai minimi quadrati di grado n = %d per le osservazioni', n));
```

Eseguendo lo script si ottiene un output simile a quello della figura 7 seguente (qui leggermente migliorata per esigenze di stampa).

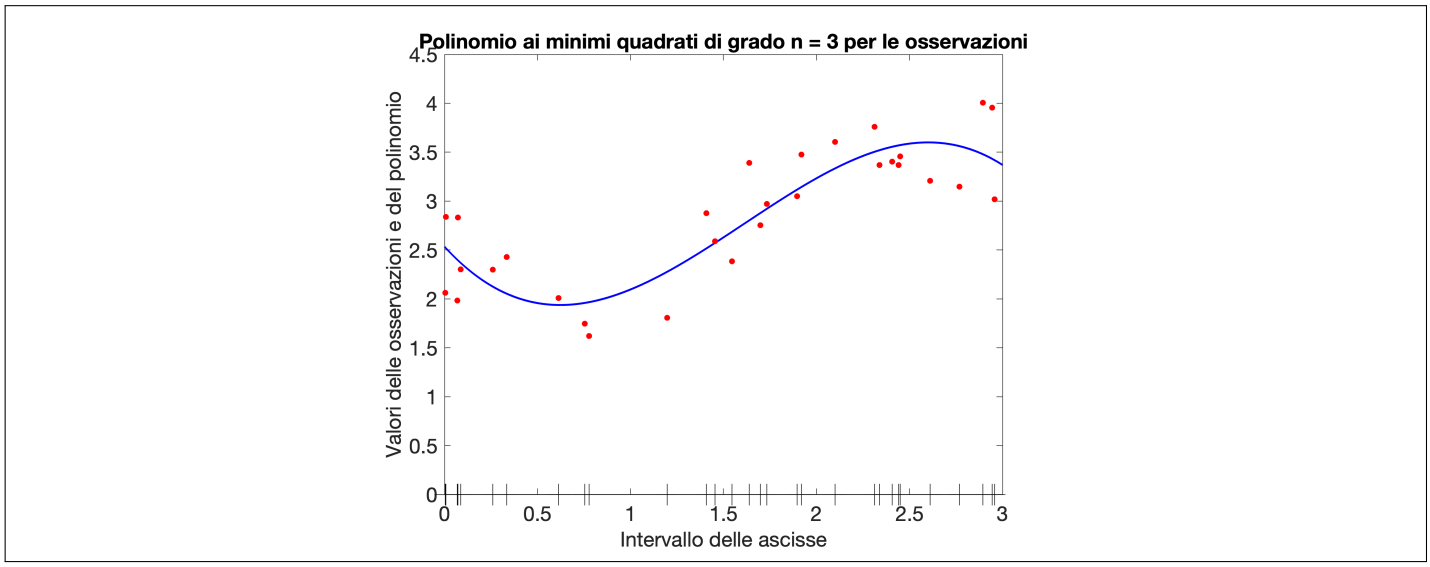


Figura 7: osservazioni casuali (in rosso) generate perturbando la funzione $h(x)$ dell'esercizio 6 e grafico del corrispondente polinomio $p_3(x)$ di grado $n = 3$ (in blu), approssimante le osservazioni nel senso dei minimi quadrati. Le ascisse (casuali) degli $N = 30$ valori campionati nell'intervallo $[a, b] = [0, 3]$ sono evidenziate dai segmenti verticali sull'asse x [la figura è stata leggermente migliorata per esigenze di stampa].