Test di Laboratorio di Calcolo Numerico, 25 Maggio 2020
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini25052020_ModII, contenente tutti i files creati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta. *Non occorre inserire le funzioni disponibili in Matlab di default.*

Considera l'equazione $x = e^{-x}$.

1. Implementa l'iterazione di punto fisso $x_{k+1} = e^{-x_k}$, partendo da $x_0 = 1$, usando come criterio solo $|x_{k+1} - x_k| < 10^{-9}$. Ad ogni iterazione riporta sulla finestra di Matlab i valori come nella seguente tabella:

k	x_{k+1}	$\lvert x_{k+1} - x_k \rvert$
$\vdots$	$\vdots$	$\vdots$

2. Per $\omega \in (0, 3)$, considera la seguente equazione equivalente,

$$x = \frac{\omega e^{-x} + x}{1 + \omega}$$

Adatta l'implementazione precedente alla presenza del parametro ω. Per 30 diversi valori di ω uniformemente distribuiti in $(0, 3)$, determina sperimentalmente il valore ottimale di ω, e fai in modo che tale valore venga riportato automaticamente sullo schermo al termine della sperimentazione nel seguente modo:

`...and the winner is omega = ...`

(tutto deve avvenire all'interno del codice, e non in modo manuale)
Riporta su uno stesso grafico tutte le storie della convergenza, in termini del residuo $|x_{k+1} - x_k|$, inserendo le etichette degli assi ed il titolo.
Al termine, riporta i risultati in una tabella come la seguente (k_* è il numero finale di iterazioni):

ω	# iter	x_{k_*}	$\lvert x_{k_*} - x_{k_*-1} \rvert$
$\vdots$	$\vdots$	$\vdots$	

Test di Laboratorio di Calcolo Numerico, 6 Luglio 2018
Laurea Triennale in Matematica

Al termine della prova, spedisci quanto fatto a `valeria.simoncini@unibo.it` (due files: una funzione matlab ed uno script contenente tutti i comandi degli script)

1. Dato un vettore $v \in \mathbb{R}^n$, proponi una implementazione (`miomax`) della funzione Matlab `[vmax,imax]=max(v)`, che determina il valore massimo in valore assoluto del vettore v, insieme all'indice di componente di tale valore.

2. Mediante script, usa la funzione `miomax` per determinare il massimo in modulo del vettore $v = [-1, 3, -5, 4, 10, -12]$.

3. È data la funzione $f(x) = x \sin(x)^2 \cos(x)$ per $x \in [0, \pi/2]$. Fai il grafico della funzione f. Usa la funzione Matlab `miomax` per approssimare il punto di massimo della funzione f sull'intervallo dato, mediante tabulazione dei valori della funzione su almeno 100 punti dell'intervallo. Riporta il punto $(x_{max}, f(x_{max}))$ sul grafico.

Test di Laboratorio di Calcolo Numerico, 16 Gennaio 2019
Laurea Triennale in Matematica
Test di fine Modulo

Al termine della prova, spedisci quanto fatto a `mat.calcolonumerico@unibo.it` come cartella chiamata col proprio nome e data (es. ValeriaSimoncini160119, contenente 3 files: due funzioni matlab ed uno script contenente tutti i comandi richiesti)

Crea uno script Matlab con il tuo cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento:

1. (2 punti) Crea una matrice A tridiagonale, con elementi diagonali uguali a 3.5 e non diagonali uguali ad 1.2 (sotto) e -1.2 (sopra), di dimensione n (il valore viene fissato successivamente). Crea un vettore b con elementi casuali (da una distribuzione normale), di norma Euclidea unitaria.

2. (8 punti) Crea la funzione `v=triang(d1,d2,y)` che risolve il sistema $Pv = y$ con P bidiagonale inferiore, le cui due diagonali nonzero sono memorizzate in `d1` (diagonale di P), e `d2`. (la funzione sarà usata al punto successivo)

3. (14 punti) Sia D la matrice diagonale di A, ed E (ed F) la parte triangolare strettamente inferiore (e superiore) di A. Per $\omega \neq 0$, considera lo splitting $A = (E + \omega D) - ((\omega - 1)D - F) =: P - N$. Implementa mediante funzione Matlab il metodo iterativo derivante da questo splitting – per la risoluzione del sistema lineare $Ax = b$ con A, b creati sopra – con la chiamata

   ```
   [x,iter,res]=iterazione(A,b,P,omega,x0,maxit,tol);
   ```

   (in output vengono forniti: la soluzione approssimata finale, il numero totale di iterazioni, il vettore della storia delle norme dei residui)
   Fai si che la funzione `iterazione` mostri ad ogni iterazione, sulla stessa riga,
   ω, n.iter, norma residuo

4. (8 punti) Valida la funzione implementata sui dati creati nello script per $n = 500$, `tol`= 10^{-6}, `maxit`= 100, `x0 = zeros(n,1);`, mediante un ciclo in $\omega \in \{1.2, 1.4, 1.6, 1.8\}$. Per ogni ω, calcola la norma Euclidea dell'errore relativo, rispetto alla soluzione $x^* = A \setminus b$ ottenuta con "backslash" di Matlab, e riporta la storia della norma del residuo (su uno stesso grafico, per tutti i valori di ω).

Test di Laboratorio di Calcolo Numerico, 15 Giugno 2020
Laurea Triennale in Matematica.
Test Modulo II

Crea uno script con il tuo nome e cognome, contenente tutto lo svolgimento.

Considera le due funzioni:

$$f_1(x) = \sin(2x)\cos(x),\ x \in [a,b] = [0,\pi], \qquad f_2(x) = x^{\frac{5}{2}},\ x \in [a,b] = [0,1].$$

1. Fai il grafico di f_i, $i = 1, 2$ su due figure distinte (risp. Figura 1 e Figura 2).
2. Per ognuna delle funzioni, riporta sul grafico corrispondente l'approssimazione della funzione mediante polinomi compositi $P_h^{(\ell)}$ di grado $\ell = 1, 2$, al variare del numero k di nodi, $k \in \{4, 8, 12, 16\}$ (h è la lunghezza del sottointervallo).
3. Per ognuna delle funzioni, fai un display come nella seguente tabella (qui riportato per f_1) per $k = 1, 2, 3, \ldots, 10$:

ℓ	k	$\|f_1 - P_h^{(\ell)}\|_\infty$	$\|f_1 - P_k^{(\ell)}\|_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

   dove con $\|f_1 - P_k^{(\ell)}\|_2$ si intende l'approssimazione numerica dell'errore

$$\left(\int_a^b (f_1(x) - P_k^{(\ell)}(x))^2 dx \right)^{\frac{1}{2}}$$

   mediante la formula composita dei trapezi, con un numero di nodi $n_* = 1000$.
4. Per $\ell = 2$ ed ognuna delle due funzioni, stima l'ordine di convergenza asintotica dell'approssimazione, fornendo un display come nella seguente tabella (qui riportato per f_1):

h	$\|f_1 - P_h^{(2)}\|_\infty$	p
0.5		
0.25		
$\vdots$		

   dove p è una stima dell'ordine di convergenza ed h è la lunghezza del sottointervallo nella decomposizione dell'intervallo $[a, b]$ in k sottointervalli, con $h \to 0$ (arriva fino a $h < 10^{-3}$).
5. Per $\ell = 2$, riporta su uno stesso grafico i valori (k, p_k) con $p = p_k$ presi dalla tabella precedente, per entrambe le funzioni. Aggiungi etichette, titolo e legenda. Commenta i risultati ottenuti usando un display nella command windows di Matlab.

Test di Laboratorio.

1. E' data la matrice $A \in \mathbf{R}^{n \times n}$ non singolare di tipo Hessenberg superiore.
   1. Implementa un algoritmo `[Q,R]=QR(A)` che determini le matrici $Q \in \mathbf{R}^{n \times n}$ ortogonale ed $R \in \mathbf{R}^{n \times n}$ triangolare superiore tale che $A = QR$, con costo computazionale $\mathcal{O}(n)$, ed allocazioni di memoria $\mathcal{O}(n)$.

2. Implementa un algoritmo `x=risolvo(Q,R,b);` per determinare la soluzione del sistema lineare $Ax = b$ con $b \neq 0$, che sfrutti la funzione `QR(A)`. Nota: A seconda dell'implementazione scelta per `QR`, gli input/output di queste due funzioni possono essere modificati.

Test di Laboratorio di Calcolo Numerico, 16 Gennaio 2019
Laurea Triennale in Matematica
Test di fine Modulo

Al termine della prova, spedisci quanto fatto a `mat.calcolonumerico@unibo.it` come cartella chiamata col proprio nome e data (es. ValeriaSimoncini160119, contenente 3 files: due funzioni matlab ed uno script contenente tutti i comandi richiesti)

La prova viene ritenuta sufficiente con un voto superiore a 18

Crea uno script Matlab con il tuo cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento:

1. (2 punti) Crea una matrice A tridiagonale, con elementi diagonali uguali a 3.5 e non diagonali uguali ad 1.2, di dimensione n (il valore viene fissato successivamente). Crea un vettore b con elementi casuali (da una distribuzione normale), di norma Euclidea unitaria.

2. (8 punti) Crea la funzione `v=triang(d1,d2,y)` che risolve il sistema $Pv = y$ con P bidiagonale superiore, le cui due diagonali nonzero sono memorizzate in `d1` (diagonale di P), e `d2`. (la funzione sarà usata al punto successivo)

3. (14 punti) Sia D la matrice diagonale di A, ed E la parte triangolare strettamente inferiore di A. Per $\omega \neq 0$, considera lo splitting $A = (E^T + \omega D) - ((\omega - 1)D - E) =: P - N$. Implementa mediante funzione Matlab il metodo iterativo derivante da questo splitting – per la risoluzione del sistema lineare $Ax = b$ con A, b creati sopra – con la chiamata

   ```
   [x,iter,res]=iterazione(A,b,P,omega,x0,maxit,tol);
   ```

   (in output vengono forniti: la soluzione approssimata finale, il numero totale di iterazioni, il vettore della storia delle norme dei residui)
   Fai si che la funzione `iterazione` mostri ad ogni iterazione, sulla stessa riga,
   n.iter, norma residuo, ω

4. (8 punti) Valida la funzione implementata sui dati creati nello script per $n = 500$, `tol`= 10^{-8}, `maxit`= 300, `x0 = ones(n,1);`, mediante un ciclo in $\omega \in \{1.3, 1.5, 1.7, 1.9\}$. Per ogni ω, calcola la norma Euclidea dell'errore relativo, rispetto alla soluzione $x^* = A \setminus b$ ottenuta con "backslash" di Matlab, e riporta la storia della norma del residuo (su uno stesso grafico, per tutti i valori di ω).

Test di Laboratorio di Calcolo Numerico, 23 Maggio 2019
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine della prova spedisci la cartella a

`mat.calcolonumerico@unibo.it`

Considera la funzione $f(x) = e^x \sin(x)$, per $x \in [a, b] = [-\pi, \pi]$.

1. Mediante `fplot`, fai il grafico di f nell'intervallo considerato;
2. Per $n = 5, 10, 15, 20$ determina il polinomio p_n di grado n interpolante di f in nodi equispaziati e riporta sullo stesso grafico di f, il grafico del polinomio al variare di n. Includi la legenda, le etichette sugli assi ed il titolo;
3. Per ogni $n \in \{1, 2, 3\}$ determina il polinomio composito $P_h^{(n)}$ al variare di $h = (b-a)/2^k$. In particolare, per $k = 2, 3, 4, 5, 6$ valuta l'errore ottenuto $\|f - P_h^{(n)}\|_\infty$ e riporta i valori in una tabella del tipo

n	h	$\|f - P_h^{(n)}\|_\infty$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

Inoltre, per ogni n riporta su un grafico (uno per ogni valore di n) la funzione f e la successione di polinomi $P_h^{(n)}$, inserendo legenda, etichette e titolo.

Test di Laboratorio di Calcolo Numerico, 08 Gennaio 2020
Laurea Triennale in Matematica
Test Modulo I, turno 3

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini080120 contenente i files che devono essere poi spediti a

`mat.calcolonumerico@unibo.it`

inserendo nel subject nome cognome e "Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data la matrice $A \in \mathbb{R}^{n\times n}$ simmetrica, definita positiva e tridiagonale.

1. Implementa un algoritmo `[alfa,beta]=intervallo(A)` che, mediante l'uso dei dischi di Gerschgorin, determini α, β tali che $0 \le \alpha \le \lambda \le \beta$, con λ autovalore di A.

2. Implementa un algoritmo per determinare una approssimazione dell'autocoppia relativa all'autovalore di A pi'ùvicino a zero, sfruttando α, β calcolati precedentemente, usando la chiamata

   ```
   [lambda,x,iter]=approssimo(A,x0,maxit,tol,alfa,beta);
   ```

   con `maxit=100, tol=1e-8` e `x0` scelto in modo opportuno. In output sono incluse le approssimazioni finali, ed il numero di iterazioni effettuate.
   La funzione deve prevedere un grafico di convergenza del residuo.

   Nota: L'implementazione proposta deve minimizzare il costo computazionale, evitando in particolare di rifare le stesse operazioni ad ogni iterazione. Tutte le funzioni usate all'interno di `approssimo.m` devono essere scritte ed incluse nella cartella.

3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento: Per $n = 100, 200, \ldots, 1000$,

   - Crea una matrice A tridiagonale del tipo sopra, con elementi da 1 ad n sulla diagonale, ed elementi uguali ad $1/3$ fuori dalla diagonale.
   - Usa le funzioni scritte sopra per risolvere il problema $Ax = \lambda x$ con λ minimo, e riporta in un display i seguenti dati (incolonnati come descritto in tabella)

n	α	β	n.iter	CPU time	$\frac{\|A\tilde{x} - \tilde{x}\tilde{\lambda}\|}{\|\tilde{\lambda}\|}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

   dove $(\lambda, \tilde{x})$ è la coppia determinata dalla funzione `approssimo`, e vengono visualizzati almeno 4 decimali significativi (in formato scientifico).
   - Per $n = 1000$ riporta su uno stesso grafico la storia della convergenza del residuo, e la convergenza attesa da risultati teorici noti. Includi le etichette e la legenda.

Test di Laboratorio di Calcolo Numerico, 08 Gennaio 2020
Laurea Triennale in Matematica
Test Modulo I, turno 2

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini080120 contenente i files che devono essere poi spediti a

`mat.calcolonumerico@unibo.it`

inserendo nel subject nome cognome e "Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

PROBLEMA: APPROSSIMAZIONE DI AUTOCOPPIA.

Sia $A \in \mathbb{R}^{n\times n}$ di tipo Hessenberg superiore che ammette fattorizzazione LU.

1. Implementa un algoritmo `[L,U]=fatt_Hess(A)` che determina la fattorizzazione LU di una matrice di tipo Hessenberg superiore.

2. Implementa un algoritmo per determinare una approssimazione dell'autocoppia relativa all'autovalore di A più vicino a zero, usando la chiamata

   ```
   [lambda,x]=approssimo(A,x0,maxit,tol,alfa,beta);
   ```

   con `maxit=100, tol=1e-8` e `x0` scelto in modo opportuno.
   L'implementazione proposta deve minimizzare il costo computazionale, evitando in particolare di rifare le stesse operazioni ad ogni iterazione. Tutte le funzioni usate all'interno di `approssimo.m` devono essere scritte ed incluse nella cartella.
   La funzione deve prevedere un grafico di convergenza del residuo.

3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento: Per $\alpha = 0.1, 0.2, 0.3, 0.4$,

   - Per $n = 100$, crea una matrice

   $$A = Q\Lambda Q^{-1}, \qquad \Lambda = \mathrm{diag}(1, 1+\alpha, 2, \ldots, n-1)$$

   e Q con elementi casuali presi da una distribuzione normale.
   - Crea la matrice `H=hess(A);` usando la corrispondente funzione Matlab.
   - Usa le funzioni scritte sopra per risolvere il problema $Ax = \lambda x$ con λ più vicino a zero e riporta in un display i seguenti dati (incolonnati come descritto in tabella)

α	n.iter	CPU time	$\tilde{\lambda}$	$\frac{\|A\tilde{x} - \tilde{x}\tilde{\lambda}\|}{\|\tilde{\lambda}\|}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

   dove $(\lambda, \tilde{x})$ è la coppia determinata dalla funzione `approssimo`, e vengono visualizzati almeno 4 decimali significativi (in formato scientifico).
   - Per $\alpha = 0.1$ e $\alpha = 0.4$, riporta su uno stesso grafico la storia della convergenza del residuo, e la convergenza attesa da risultati teorici noti. Includi le etichette e la legenda.

Test di Laboratorio di Calcolo Numerico, 20 Giugno 2019
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini200619, contenente tutti i files scritti. Al termine spedisci la cartella a
`mat.calcolonumerico@unibo.it`

Data $A \in \mathbb{R}^{n\times n}$ simmetrica definita positiva considera il problema agli autovalori

$$Ax = \lambda x, \quad x \in \mathbb{R}^n, \lambda \in \mathbb{R}.$$

1. (10 punti) Implementa l'algoritmo delle potenze inverse tralsate

   `[xk,lambdak,iter,normres]=potenze_inverse_shift(A,x0,shift,maxit,tol);`

   dove `x0` è un vettore iniziale di norma unitaria, `shift` è il numero reale usato per la traslazione, `maxit` è il numero massimo di iterazioni permesso, e `tol` è la tolleranza per il criterio di arresto per il residuo *relativo*, cioè

   $$\|Ax_k - x_k\lambda_k\|/|\lambda_k|.$$

   In output vengono restituiti i valori della autocoppia approssimata (`xk,lambdak`), il numero di iterazioni effettuate (`iter`), e la norma finale del residuo (`normres`).
   Nota: L'implementazione deve prevedere la risoluzione numerica del sistema con A ad ogni iterazione del metodo delle potenze, con un costo per iterazione di $O(n^2)$. Si possono usare le funzioni di Matlab per determinare la soluzione.
2. (5 punti) Fai il grafico della convergenza del residuo (vedi sotto).
3. (5 punti) Scrivi una funzione matlab `[lmin,lmax]=interv_spettr(A);` che determini una stima dal basso e dall'alto dell'intervallo spettrale di A;

4. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento:

   Per ogni $n \in \{10, 20, \ldots, 100\}$:
   i) (2 punti) Crea la matrice $A = (a_{i,j})_{i,j=1,n}$ con $a_{i,i} = i$, $i = 1, \ldots, n$, e $a_{i,i+1} = a_{i+1,i} = 1/3$, $i = 1, \ldots, n-1$.
   ii) (4 punti) Approssima gli estremi dell'intervallo spettrale di A con la funzione `interv_spettr`.
   Approssima la autocoppia di A più vicina all'origine mediante la funzione costruita, `potenze_inverse_shift`, usando l'estremo spettrale sinistro come `shift`.
   iii) (3 punti) Riporta con un display le seguenti informazioni (incolonnate come descritto in tabella):

n	num.iter.	CPU time	$\|Ax_k - x_k\lambda_k\|$	`lmin`	`lmax`
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

   iv) - (3 punti) Per ogni valore di n, riporta il grafico della convergenza del residuo **su una stessa figura**, includendo legenda, x e y labels, titolo (devono risultare varie curve nello stesso grafico).

Test di Laboratorio di Calcolo Numerico, 17 Aprile 2019
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini170419, contenente 3 files: due funzioni matlab ed uno script contenente tutti i comandi richiesti). Al termine della prova spedisci la cartella a
`mat.calcolonumerico@unibo.it`

Problema: risolvi il problema ai minimi quadrati $\min_x \|b - Ax\|$ con A avente la seguente forma

$$A = \begin{bmatrix} R \\ a^T \end{bmatrix} \in \mathbb{R}^{(n+1)\times n}$$

con $R \in \mathbb{R}^{n\times n}$ triangolare inferiore non singolare e $0 \neq a \in \mathbb{R}^n$.

1. (14 punti) Implementa un algoritmo che determini la soluzione del problema

   `[x]=minquad(R,a,b);`

   avente un costo computazionale di $O(n^2)$ e con una aggiunta di memoria di al più O(n); sfrutta anche la funzione `x=triang(M,y,tipo)` definita qui sotto.

2. (8 punti) Crea la funzione `x=triang(M,y,tipo);` che risolva il sistema $Mx = y$ con M triangolare superiore o inferiore (triang.inferiore per tipo=1, triang.superiore altrimenti).
3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento: Per $n = 10, 20, \ldots, 100$,

   - (2 punti) Crea una matrice A del tipo sopra, con R avente valori casuali nell'intervallo $(0, 1)$ e a vettore con elementi casuali da una distribuzione normale. Crea un vettore b con tutti elementi uguali ad uno.

   - (8 punti) Usa le funzioni `minquad` e `triang` per risolvere il problema ai minimi quadrati $\min_x \|b - Ax\|$, e riporta in un display i seguenti dati (incolonnati come descritto in tabella)

n	`cond`(A)	CPU time	$\|x - x_*\|$
⋮	⋮	⋮	⋮

   dove x_* è la soluzione ottenuta con "\" di matlab (la funzione "\" determina la soluzione ai minimi quadrati, per A alta).

Test di Laboratorio di Calcolo Numerico, 29 Aprile 2019
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini170419, contenente 3 files: due funzioni matlab ed uno script contenente tutti i comandi richiesti). Al termine della prova spedisci la cartella a
`mat.calcolonumerico@unibo.it`

Data $A \in \mathbb{R}^{n\times n}$ simmetrica considera il problema agli autovalori

$$Ax = \lambda x, \quad x \in \mathbb{R}^n, \lambda \in \mathbb{R}.$$

1. (14 punti) Implementa
   - L'algoritmo delle potenze inverse

     `[xk,lambdak,iter,normres]=potenze_inverse(A,x0,maxit,tol);`

     dove `x0` è un vettore iniziale di norma unitaria, `maxit` è il numero massimo di iterazioni permesso, e `tol` è la tolleranza per il criterio di arresto per il residuo *relativo*, cioè

     $$\|Ax_k - x_k\lambda_k\|/|\lambda_k|.$$

     In output vengono restituiti i valori della autocoppia approssimata (`xk,lambdak`), il numero di iterazioni effettuate (`iter`), e la norma finale del residuo (`normres`).
   - (4 punti) L'implementazione deve prevedere la risoluzione numerica del sistema con A ad ogni iterazione con un costo per iterazione di $O(n^2)$. Si possono usare le funzioni di Matlab per determinare la soluzione.
   - Fai il grafico della convergenza del residuo (vedi sotto).

2. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento:

   - (2 punti) Per $n = 20$:
   i) Crea una matrice $Q \in \mathbb{R}^{n\times n}$, ottenuta dalla ortogonalizzazione di una matrice di valori casuali da una distribuzione nell'intervallo (0,1). Questa matrice rimarrà invariata per tutto l'esperimento.
   ii) Crea una matrice $A = Q\Lambda Q^T$, dove per $\delta > 0$, vale $\Lambda = \mathrm{diag}(1, 2-\delta, 3, \ldots, n)$.

   - (8 punti) Per $\delta = 0, 0.1, \ldots, 0.9$, usa il codice `potenze_inverse` per determinare il più piccolo autovalore di A, e riporta in un display i seguenti dati (incolonnati come descritto in tabella)

δ	num.iter.	CPU time	$\|Ax_k - x_k\lambda_k\|$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

   - (4 punti) Per ogni valore di δ, fai il grafico della convergenza del residuo, usando la stessa figura per tutti i valori di δ, includendo legenda, x e y labels, titolo (devono risultare varie curve nello stesso grafico).

Test di Laboratorio di Calcolo Numerico, 23 Maggio 2019
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519, contenente tutti i files creati durante il test. Al termine della prova spedisci la cartella a

`mat.calcolonumerico@unibo.it`

Considera la risoluzione del seguente sistema lineare non singolare

$$Ax = b, \quad \text{con} \quad A = L + uv^T \in \mathbb{R}^{n\times n},\ u, v, b \in \mathbb{R}^n r, \tag{0.1}$$

e L bidiagonale superiore, non singolare.

1. Scrivi una funzione `y = bidiag(L,b);` che risolva il sistema $Ly = b$ con L bidiagonale superiore con il minimo costo computazionale;
2. Scrivi una funzione `x = ShermanMorrison(L,u,v,b);` che risolva il sistema (0.1) mediante la formula di Sherman-Morrison e la funzione `bidiag` costruita precedentemente;
3. Crea ora uno script. Per ogni $n = 100, 200, \ldots, 1000$
   - Crea i dati L (matrice bidiagonale, avente diagonale con valori da 1 ad n, e sopradiagonale uguale ad un vettore con valori casuali in $(0, 1)$), u, v vettori con valori casuali presi da una distribuzione normale. Fissata la soluzione esatta $x = \mathbf{1}$ (vettori di tutti uno), costruisci b come $b = Ax$ (senza costruire A esplicitamente).
   - Risolvi il sistema (0.1) usando le funzioni costruite precedentemente. Fai in modo che per ogni valore di n vengano stampati (sulla stessa riga) i seguenti valori

n	CPU time	Errore
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮

   dove per "Errore" si intende la norma Euclidea relativa dell'errore, rispetto alla soluzione esatta data.
   Fai inoltre un grafico che riporti in ascissa la dimensione del problema, ed in ordinata il tempo di CPU necessario. Inserisci legenda, etichette sugli assi e titolo.

Test di Laboratorio di Calcolo Numerico, 20 Giugno 2019
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine della prova spedisci la cartella a

`mat.calcolonumerico@unibo.it`

Considera il problema della stima dell'integrale $\mathcal{I} = \int_0^{\frac{\pi}{2}} \exp(x^2 - 1)\sin(x)\cos(x)dx$

1. (2 punti) Fai il grafico della funzione integranda nell'intervallo $[a, b]$ considerato mediante `fplot`;
2. (8 punti) Scrivi una funzione `I1=trapezi(f,a,b,mflag,m);` che implementi la formula di quadratura dei trapezi, standard o composita ad m intervalli (usa la variabile `mflag` per tale scelta);
3. (8 punti) Scrivi una funzione `I2=Simpson(f,a,b,mflag,m);` che implementi la formula di quadratura di Cavalieri-Simpson, standard o composita ad m intervalli (usa la variabile `mflag` per tale scelta);
4. (4 punti) Scrivi uno script che determini una approssimazione dell'integrale $\mathcal{I}$ mediante le funzioni costruite (versione standard). Fai un display dei risultati, come segue, sapendo che $\mathcal{I} \approx 0.454338162532260$,

I1	$\lvert$I1 $- \mathcal{I}\rvert$	I2	$\lvert$I2 $- \mathcal{I}\rvert$

5. (5 punti) Scrivi uno script che determini una approssimazione dell'integrale $\mathcal{I}$ mediante le funzioni costruite (versione composita). In particolare, per $m = 2^k$, $k = 1, 2, \ldots, 6$, riporta i valori come nella seguente tabella:

m	I1	$\lvert$I1 $- \mathcal{I}\rvert$	I2	$\lvert$I2 $- \mathcal{I}\rvert$

6. (5 punti) Verifica graficamente che la convergenza è dell'ordine atteso, rispetto ad $h = (b - a)/m$.

Test di Laboratorio di Calcolo Numerico, 29 Maggio 2019
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine della prova spedisci la cartella a

`mat.calcolonumerico@unibo.it`

Considera la funzione $f(x) = e^{1-x} - 1$, per $x \in [a, b] = [-1, 10]$.

1. Individua graficamente lo zero[1] x^* di f nell'intervallo considerato mediante `fplot`;
2. Scrivi una funzione `x=PuntoFisso(phi,x0,maxit,tola,tolr,tolf);` che implementi il metodo di punto fisso per una generica equazione di punto fisso $x = \varphi(x)$;
3. Scrivi una funzione `x=Newton(f,df,x0,maxit,tola,tolr,tolf);` che implementi il metodo di Newton per una generica `f` e la sua derivata `df`;
4. Scrivi uno script per determinare lo zero x^* della funzione data mediante i due algoritmi implementati. con
   $$maxit = 30,\ tola = 10^{-8},\ tolr = 10^{-8},\ tolf = 10^{-9}.$$
   e $\varphi(x) = x + e^{1-x} - 1$.
   In particolare,
   - Figure 1. Considerando $x_0 = 3.0$, in un solo grafico riporta la storia della convergenza del residuo per i due metodi, includendo etichette degli assi, titolo e legenda;
   - Figure 2. In un solo grafico riporta la storia della convergenza del residuo per il metodo `PuntoFisso` al variare di x_0, con $x_0 \in \{0.0, 2.0, 4.0, 6.0, 8.0\}$, includendo etichette degli assi, titolo e legenda;
   - Figure 3. Fai girare l'algoritmo di Newton per `tolf`$= 10^{-13}$ e prendi la soluzione ottenuta come se fosse x^*. Fai un nuovo run di Newton con `tolf`$= 10^{-8}$ e riporta su un grafico i valori di
     $$\frac{|x^* - x_k|}{|x^* - x_{k-1}|^2}, \quad k = 1, 2, \ldots, k_{final}$$
     includendo etichette degli assi e titolo. Questo esercizio può richiedere la modifica di input e/o output della chiamata alla funzione `Newton` precedentemente fatta.

[1] Cioè, x^* tale che $f(x^*) = 0$.

Test di Laboratorio (turno 2) di Calcolo Numerico, 12 Febbraio 2019
Laurea Triennale in Matematica
Test di fine Modulo

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini120219, contenente 3 files: due funzioni matlab ed uno script contenente tutti i comandi richiesti). Al termine della prova spedisci la cartella a

`mat.calcolonumerico@unibo.it`

Problema: risolvi il sistema lineare $Ax = b$ con A a banda.

1. (14 punti) Implementa il metodo di eliminazione di Gauss per matrici a banda (`beta1` banda inferiore, `beta2` banda superiore), che modifichi contemporaneamente A e b, mediante la chiamata: `[A1,b1]=gauss_banda(A,b,beta1,beta2);`
2. (8 punti) Crea la funzione `x=triang(U,beta3,y)` che risolve il sistema $Ux = y$ con U triangolare superiore con banda `beta3`.
3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento:
   - (2 punti) Per $n = 10, 20, \ldots, 100$, crea una matrice A **pentadiagonale** $n \times n$, con elementi diagonali uguali a 5 e non diagonali uguali ad 1.2 (sotto) e -1.2 (sopra). Crea un vettore b con elementi casuali (da una distribuzione normale), di norma Euclidea unitaria.
   - (8 punti) Per ogni n, usa le funzioni `gauss_banda` e `triang` per risolvere il sistema $Ax = b$, e riporta in un display i seguenti dati (incolonnati come descritto in tabella)

n	`cond`(A)	CPU time	$\|x - x_*\|$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

   dove x_* è la soluzione ottenuta con "\" di matlab.

Test di Laboratorio (turno 1) di Calcolo Numerico, 12 Febbraio 2019
Laurea Triennale in Matematica
Test di fine Modulo

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini120219, contenente 3 files: due funzioni matlab ed uno script contenente tutti i comandi richiesti). Al termine della prova spedisci la cartella a
`mat.calcolonumerico@unibo.it`

Problema: risolvi il problema ai minimi quadrati $\min_x \|b - Ax\|$ con A a banda.

1. (14 punti) Implementa la fattorizzazione QR mediante trasformazioni di Givens per matrici a banda (`beta1` banda inferiore, `beta2` banda superiore), che modifichi contemporaneamente A e b, mediante la chiamata:

   `[A1,b1]=givens_banda(A,b,beta1,beta2);`

2. (8 punti) Crea la funzione `x=triang(U,beta3,y)` che risolve il sistema $Ux = y$ con U triangolare superiore con banda `beta3`.
3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento: Per $n = 10, 20, \ldots, 100$,

   - (2 punti) Crea una matrice A **pentadiagonale** $(n + 1) \times n$, con elementi diagonali uguali a 5 e non diagonali uguali ad 1.2 (sotto) e -1.2 (sopra). Crea un vettore b con elementi casuali (da una distribuzione normale), di norma Euclidea unitaria.

   - (8 punti) Usa le funzioni `givens_banda` e `triang` per risolvere il problema ai minimi quadrati $\min_x \|b - Ax\|$, e riporta in un display i seguenti dati (incolonnati come descritto in tabella)

n	`cond`(A)	CPU time	$\|x - x_*\|$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

   dove x_* è la soluzione ottenuta con "\" di matlab (la funzione "\" determina la soluzione ai minimi quadrati, per A alta).

Test di Laboratorio di Calcolo Numerico, 08 Gennaio 2020
Laurea Triennale in Matematica
Test Modulo I, turno 1

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini080120 contenente i files che devono essere poi spediti a

`mat.calcolonumerico@unibo.it`

inserendo nel subject nome cognome e "Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

PROBLEMA: APPROSSIMAZIONE DI AUTOCOPPIA.

Sia $A \in \mathbb{R}^{n\times n}$ di tipo Hessenberg **inferiore** che ammette fattorizzazione LU.

1. Implementa un algoritmo `[L,U]=fatt_Hess(B)` che determina la fattorizzazione LU di una matrice di tipo Hessenberg **superiore**.

2. Implementa un algoritmo per determinare una approssimazione dell'autocoppia relativa all'autovalore di A più vicino a μ, usando la chiamata

   `[lambda,x]=approssimo(A,mu,x0,maxit,tol,alfa,beta);`

   con `maxit=100, tol=1e-8` e `x0` scelto in modo opportuno. La funzione deve prevedere un grafico di convergenza del residuo.

   Nota: L'implementazione proposta deve minimizzare il costo computazionale, evitando in particolare di rifare le stesse operazioni ad ogni iterazione (a tal fine usare `fatt_Hess` in modo opportuno). Tutte le funzioni usate all'interno di `approssimo.m` devono essere scritte ed incluse nella cartella.

3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento: Per $\mu = 2 + \delta$ con $\delta = 0.1, 0.2, \ldots, 0.9$,

   - Per $n = 100$, crea una matrice $A0 = Q\Lambda Q^{-1}$ con $\Lambda = \mathrm{diag}(1, 2, 3, \ldots, n)$ e Q con elementi casuali presi da una distribuzione uniforme nell'intervallo $(0, 1)$.
   - Crea la matrice `A=(hess(A0))';` usando la corrispondente funzione Matlab (attenzione alla trasposizione!)
   - Usa la funzione `approssimo` per risolvere il problema $Ax = \lambda x$ con λ più vicino a μ e riporta in un display i seguenti dati (incolonnati come descritto in tabella)

μ	n.iter	CPU time	$\tilde{\lambda}$	$\frac{\|A\tilde{x} - \tilde{x}\tilde{\lambda}\|}{\|\tilde{\lambda}\|}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

   dove $(\lambda, \tilde{x})$ è la coppia determinata dalla funzione `approssimo`, e vengono visualizzati almeno 4 decimali significativi (in formato scientifico).

Test di Laboratorio di Calcolo Numerico, 03 Settembre 2019
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini200619, contenente tutti i files scritti. Al termine spedisci la cartella a
`mat.calcolonumerico@unibo.it`

Data $A \in \mathbb{R}^{n\times n}$ tridiagonale simmetrica definita positiva considera il problema della risoluzione del sistema lineare

$$Ax = f, \quad f \in \mathbb{R}^n,$$

con $A = \text{tridiag(b, a, b)}$, con $a \in \mathbb{R}^n$, $b \in \mathbb{R}^{n-1}$.

1. (10 punti) Implementa l'algoritmo di Thomas in modo da ottenere la fattorizzazione $A = LL^T$, $L = \text{bidiag}(\ell_1, \ell_2)$, con $\ell_1 \in \mathbb{R}^n$ e $\ell_2 \in \mathbb{R}^{n-1}$,

   `[`ℓ_1, ℓ_2`]=thomas_spd(a,b);`

   Risolvi quindi il sistema dato implementando la funzione `x = risolvo(`ℓ_1, ℓ_2`, f)`.

2. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento:

   Sia a il vettore costante di valori uguali a 2, e b un vettore di valori casuali da una distribuzione uniforme nell'intervallo $(0, 1)$. Sia f un vettore di valori casuali da una distribuzione normale.
   Per ogni $n \in \{100, 200, \ldots, 1000\}$:

   (a) Usa i comandi `tic-toc` per valutare il tempo di CPU della prima funzione, della seconda funzione, ed il tempo totale, riportando questi valori con display come nella seguente tabella:

n	norma res	CPU time f1	CPU time f2	CPU time tot
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

   Nella tabella riporta anche la norma del residuo con il valore di x ottenuto.

   (b) Memorizza il tempo della risoluzione t_{sol} (uso della funzione `risolvo`) ed il tempo totale di calcolo t_{tot} per ogni n, cosi da poter fare un grafico dei tempi t_{sol} e t_{tot} al variare di n.

   Fai il grafico del tempo totale e del tempo di risoluzione al variare di n (due curve nello stesso grafico), inserendo le etichette degli assi, la legenda ed il titolo.

Test di Laboratorio di Calcolo Numerico, 08 Gennaio 2020
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Considera la funzione $f(x) = x\cos(2\pi x^3)$, per $x \in [a, b]$.

1. Mediante `fplot`, fai il grafico di f nell'intervallo $[a, b] = [-1, 1]$.

2. Sia $[a, b] = [-1, 1]$. Per ogni $n \in \{1, 2, 3, 4\}$ determina il polinomio di grado n interpolante composito $P_h^{(n)}$ di f al variare di $h = (b-a)/2^k$. In particolare, per $k = 3, \ldots, 7$ valuta l'errore ottenuto $\|f - P_h^{(n)}\|_\infty$ e riporta i valori in una tabella del tipo

n	h	$\|f - P_h^{(n)}\|_\infty$
$\vdots$	$\vdots$	$\vdots$

   Per ogni n riporta su uno stesso grafico la funzione f e la successione di polinomi $P_h^{(n)}$ al variare di k, inserendo legenda, etichette e titolo.

   Verifica graficamente che la convergenza è dell'ordine atteso, rispetto ad $h = (b-a)/m$.

3. Scrivi la funzione Matlab `simpcomp.m` che determina la formula di quadratura di Simpson composita per l'approssimazione dell'integrale definito di una data funzione, su un dato intervallo $[a, b]$.
   Siano f la funzione definita all'inizio del testo, e $[a, b] = [-1/2, 1]$. Scrivi uno script che determini una approssimazione `I2` dell'integrale

$$\mathcal{I} = \int_{-\frac{1}{2}}^{1} f(x)dx \approx -0.052002226153888,$$

   mediante la funzione `simpcomp`. In particolare, per $m = 2^k$, $k = 3, 2, \ldots, 10$, riporta i valori come nella seguente tabella:

m	`I2`	$\|\texttt{I2} - \mathcal{I}\|$
$\vdots$	$\vdots$	$\vdots$

   Verifica graficamente che la convergenza è dell'ordine atteso, rispetto ad $h = (b-a)/m$.

Test di Laboratorio di Calcolo Numerico, 29 Maggio 2019
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine della prova spedisci la cartella a

`mat.calcolonumerico@unibo.it`

Considera la funzione $f(x) = \frac{1}{2}\sin(x)\cos(x) - x$, per $x \in [a, b] = [-1, 1]$.

1. Individua graficamente lo zero[1] x^* di f nell'intervallo considerato mediante `fplot`;
2. Scrivi una funzione `x=PuntoFisso(phi,x0,maxit,tola,tolr,tolf);` che implementi il metodo di punto fisso per una generica equazione di punto fisso $x = \varphi(x)$;
3. Scrivi una funzione `x=secanti(f,xm1,x0,maxit,tola,tolr,tolf);` che implementi il metodo delle secanti per una generica `f` e dati iniziali x_{-1}, x_0;
4. Scrivi uno script per determinare lo zero x^* della funzione data mediante i due algoritmi implementati, con
$$maxit = 200,\ tola = 10^{-8},\ tolr = 10^{-8},\ tolf = 10^{-9}.$$
e $\varphi(x) = \frac{1}{2}\sin(x)\cos(x)$.
In particolare,
   - Figure 1. Considerando $x_{-1} = -1.0, x_0 = -1/2$, in un solo grafico riporta la storia della convergenza del residuo per i due metodi, includendo etichette degli assi, titolo e legenda;
   - Figure 2. In un solo grafico riporta la storia della convergenza del residuo per il metodo `PuntoFisso` al variare di x_0, con $x_0 \in \{1e-4, 5e-4, 1e-3, 5e-3, ..., 5e-1\}$, includendo etichette degli assi, titolo e legenda;
   - Figure 3. Sapendo che la soluzione esatta è $x^* = 0$, stima l'ordine di convergenza del metodo delle secanti, cioè stima p in
$$\frac{|x^* - x_k|}{|x^* - x_{k-1}|^p}, \quad k = 1, 2, \ldots, k_{final}$$
facendo il display del valore di p al crescere di k, riportato in una tabella appropriata. Questo esercizio può richiedere la modifica di input e/o output del codice precedente.

[1] Cioè, x^* tale che $f(x^*) = 0$.

Test di Laboratorio di Calcolo Numerico, 10 Dicembre 2019
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini200619, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo I".

Data $A \in \mathbb{R}^{n\times m}$, $n \geq m$ e $0 \neq b \in \mathbb{R}^n$ considera il problema

$$\min_{x\in\mathbb{R}^m} \|b - Ax\| \tag{0.1}$$

con A avente rango massimo.

1. Crea la funzione `[Q1,R1]=gram(A);` che implementa l'algoritmo di Gram-Schmidt per determinare la fattorizzazione QR ridotta $A = Q_1R_1$ con $Q_1 \in \mathbb{R}^{n\times m}$ avente colonne ortonormali, ed $R_1 \in \mathbb{R}^{m\times m}$ triangolare superiore;
2. Sia $A_p = [A, a_p]$ la matrice ottenuta aggiungendo una nuova colonna ad A. Crea una funzione `[Qp,Rp]=gram_update(Q1,R1,ap);` che **aggiorna** la fattorizzazione QR ottenuta sopra, senza ricalcolare `Q1, R1`.
3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento:

   Siano

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & -11 & 12 \\ 13 & 14 & -15 & 16 \\ 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 34 \end{bmatrix}, \quad b_j = j \cdot (-1)^j.$$

   (a) Fai un display delle dimensioni di A e del suo rango. Includi nel display i nomi delle quantità riportate, come se fosse una tabella.
   (b) Mediante fattorizzazione QR di Gram-Schmidt, determina la soluzione del problema (0.1) con A, b dati. (Il sistema triangolare può essere risolto come si preferisce)
   (c) Fai un display della soluzione ottenuta x, a fianco di quella ottenuta con backslash di Matlab. Fai anche un display della norma dell'errore assoluta e relativa, e della norma del residuo assoluto e relativo. Includi nel display i nomi delle quantità riportate, come se fosse una tabella.
   (d) Sia $(a_p)_j = (-1)^j$, $j = 1, \ldots, n$. Usa la funzione creata in precedenza per risolvere il problema

$$\min_{x\in\mathbb{R}^{m+1}} \|b - [A, a_p]x\|$$

   Procedi come nel punto precedente, facendo un display di tutte le quantità coinvolte.

Test di Laboratorio di Calcolo Numerico, 10 Dicembre 2019
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II"

Considera la funzione $f_\alpha(x) = e^{-x/2}\cos(x^\alpha)$, per $x \in [a,b] = [-\pi/4, \pi/4]$ e $\alpha \in \{1,2,3\}$.

1. Mediante `fplot`, fai il grafico di f_α nell'intervallo considerato, uno per ogni valore di α considerato (una figura distinta per ogni valore di α).

2. Per $n = 4, 8, 12, 16$ determina il polinomio p_n di grado n interpolante di f_α in nodi equispaziati in $[a,b]$ e riporta il grafico del polinomio al variare di n **sullo stesso grafico di f_α (un grafico per ogni α)**. Includi la legenda, le etichette sugli assi ed il titolo. Fai inoltre un display del seguente tipo

α	n	$\|f_\alpha - p_n\|_\infty$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

3. Al variare di α, per ogni $n \in \{1,2,3\}$ determina il polinomio composito $P_h^{(n)}$ al variare di $h = (b-a)/2^k$. In particolare, per $k = 2,3,4,5,6$ valuta l'errore ottenuto $\|f_\alpha - P_h^{(n)}\|_\infty$ e riporta i valori in una tabella del tipo

α	n	h	$\|f_\alpha - P_h^{(n)}\|_\infty$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

   Inoltre, per $\alpha = 2$, per ogni n riporta su un grafico (uno per ogni valore di n) la funzione f_α e la successione di polinomi $P_h^{(n)}$ al variare di k, inserendo legenda, etichette e titolo.

Test di Laboratorio di Calcolo Numerico, 28 Gennaio 2020
Laurea Triennale in Matematica
Test Modulo I, turno 2

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini280120 contenente i files che devono essere poi spediti a

`mat.calcolonumerico@unibo.it`

inserendo nel subject nome cognome, "Modulo I, turno 2".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data la matrice $A \in \mathbb{R}^{n\times n}$ non singolare di tipo Hessenberg inferiore.

1. Implementa un algoritmo `[Q,L]=LQ(A)` che determini le matrici $Q \in \mathbb{R}^{n\times n}$ ortogonale ed $L \in \mathbb{R}^{n\times n}$ triangolare inferiore tale che $A = LQ$, con costo computazionale $\mathcal{O}(n)$, e possibilmente allocazioni di memoria $\mathcal{O}(n)$.

2. Implementa un algoritmo `x=risolvo(Q,L,b);` per determinare la soluzione del sistema lineare $Ax = b$ con $b \neq 0$, che sfrutti la funzione `LQ(A)`.

   Nota: A seconda dell'implementazione scelta per `LQ`, gli input/output di queste due funzioni possono essere modificati.

3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento: Per $n = 10, 12, 14, \ldots, 30$,

   - Crea la matrice $A = (a_{i,j})$, $i, j = 1, \ldots, n$ di tipo Hessenberg inferiore con $a_{i,i+1} = 2^i$, $a_{i,j}$ con $i \geq j$ uguali a valori reali casuali presi da una distribuzione uniforme in $(0, 1)$, e gli altri valori uguali a zero. Crea il vettore b con valori reali presi da una distribuzione normale.
   - Usa le funzioni create sopra per risolvere il sistema $Ax = b$, facendo il seguente display (coi dati incolonnati come descritto in tabella) al variare di n:

n	cond(A)	CPU time	$\frac{\|b-Ax\|}{\|b\|}$	$\frac{\|x-x_\star\|}{\|x_\star\|}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

   dove x è la soluzione ottenuta con la procedura creata, e $x_\star$ è la soluzione ottenuta con \ (backslash) di Matlab.
   - Su un grafico riporta l'errore relativo ottenuto, in funzione del numero di condizionamento della matrice, includendo titolo, etichette e legenda.

Test di Laboratorio di Calcolo Numerico, 28 Gennaio 2020
Laurea Triennale in Matematica
Test Modulo I, turno 1

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini280120 contenente i files che devono essere poi spediti a

`mat.calcolonumerico@unibo.it`

inserendo nel subject nome cognome, "Modulo I, turno 1".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data la matrice $A \in \mathbb{R}^{n\times n}$ non singolare di tipo Hessenberg superiore.

1. Implementa un algoritmo `[Q,R]=QR(A)` che determini le matrici $Q \in \mathbb{R}^{n\times n}$ ortogonale ed $R \in \mathbb{R}^{n\times n}$ triangolare superiore tale che $A = QR$, con costo computazionale $\mathcal{O}(n)$, ed allocazioni di memoria $\mathcal{O}(n)$.

2. Implementa un algoritmo `x=risolvo(Q,R,b);` per determinare la soluzione del sistema lineare $Ax = b$ con $b \neq 0$, che sfrutti la funzione `QR(A)`.

   Nota: A seconda dell'implementazione scelta per `QR`, gli input/output di queste due funzioni possono essere modificati.

3. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento: Per $n = 100$ e ϵ_k in $\{10^{-5}, 5\cdot10^{-5}, 10^{-4}, 5\cdot10^{-4}, \ldots, 10^{-2}\}$,

   - Crea la matrice $A = (a_{i,j})$, $i, j = 1, \ldots, n$ di tipo Hessenberg superiore con $a_{i+1,i} = 2^i$, $a_{i,j}$ con $i \leq j$ uguali a valori reali casuali presi da una distribuzione uniforme in $(0, 1)$, e gli altri valori uguali a zero. Crea il vettore b_0 con valori reali presi da una distribuzione normale.
   - Usa le funzioni create sopra per risolvere il sistema

   $$Ax = b_k, \qquad b_k = b_0 + \epsilon_k v,$$

   con v vettore di norma euclidea unitaria, e valori casuali da distribuzione uniforme in $(0, 1)$. Al crescere di ϵ_k, fai il seguente display (coi dati incolonnati come descritto in tabella):

ϵ_k	cond(A)	CPU time	$\frac{\|b_0 - Ax_k\|}{\|b_0\|}$	$\frac{\|x_k - x_\star\|}{\|x_\star\|}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

   dove x_k è la soluzione ottenuta con la procedura creata per il problema $Ax = b_k$, e $x_\star$ è la soluzione ottenuta con \ (backslash) di Matlab per il problema $Ax = b_0$.
   - Su un grafico riporta l'errore relativo ottenuto, in funzione del numero di condizionamento della matrice, includendo titolo, etichette e legenda.

Test di Laboratorio di Calcolo Numerico, 09 Gennaio 2020
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini080120 contenente i files che devono essere poi spediti a

`mat.calcolonumerico@unibo.it`

inserendo nel subject nome cognome e "Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

PROBLEMA: RISOLUZIONE DEL PROBLEMA AI MINIMI QUADRATI $\min_x \|b - Ax\|$ CON $A \in \mathbb{R}^{(n+1)\times n}$ TRIDIAGONALE.

1. Implementa un algoritmo che determini la soluzione del problema

   `[x]=minquad(A,b);`

   mediante i seguenti passi:
   - Scrivi la procedura per annullare l'elemento $(n+1, n)$ di A nel problema ai minimi quadrati;
   - Risolvi il sistema lineare rimanente con matrice tridiagonale, fornendo tutti i passaggi algoritmici, implementando una funzione chiamata `risolvi`.

2. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento: Per $n = 10, 20, \ldots, 100$,

   - Crea una matrice A tridiagonale del tipo sopra, avente valori casuali nonzero presi da una distribuzione uniforme nell'intervallo $(0, 1)$, e crea un vettore b con tutti elementi uguali ad uno.

   - Usa le funzioni `minquad` e `risolvi` per risolvere il problema ai minimi quadrati $\min_x \|b - Ax\|$, e riporta in un display i seguenti dati (incolonnati come descritto in tabella)

n	$\kappa(A)$	CPU time	$\|x - x_*\|$	$\|b - Ax\|$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

   dove $\kappa(A)$ è il numero di condizionamento di A (si può usare la stessa funzione matlab del caso quadrato), e x_* è la soluzione ottenuta con "\" di matlab (la funzione "\" determina la soluzione ai minimi quadrati, per A alta).

Test di Laboratorio di Calcolo Numerico, 28 Gennaio 2020
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Considera la funzione $f(x) = x\cos(2\pi x^3)$, per $x \in [-1, 1]$.

1. Mediante `fplot`, fai il grafico di f nell'intervallo $[-1, 1]$.
2. Per $n \in \{10, 16, 24\}$, determina la spline cubica interpolante "not-a-knot", e sovrapponi il grafico a quello di f. Includi etichette, titolo e legenda.
3. Verifica sperimentalmente mediante display, anche per n più grande, che l'ordine di convergenza è $\mathcal{O}(h^p)$ con p opportuno, dove h è il massimo della lunghezza degli intervalli considerati:

n	p
$\vdots$	$\vdots$

4. Fai la stessa verifica per la convergenza della derivata prima.
5. Calcola a mano le derivate di f agli estremi. Per $n \in \{10, 16, 24\}$, determina la spline cubica interpolante "completa", e sovrapponi il grafico a quello di f, in una nuova figura. Includi etichette, titolo e legenda.

Test di Laboratorio di Calcolo Numerico, 17 Luglio 2018
Laurea Triennale in Matematica

Al termine della prova, spedisci quanto fatto a `valeria.simoncini@unibo.it` (due files: una funzione matlab ed uno script contenente tutti i comandi degli script)

1. Data una matrice $A \in \mathbb{R}^{n\times n}$ ed un vettore $v \in \mathbb{R}^n$, proponi una implementazione del metodo di Jacobi per risolvere il sistema lineare $Ax = b$, mediante chiamata
   `[xk,iter,h_res]=Jacobi(A,b,x0,maxit,tol);`
   dove in input vengono dati, oltre ad A e b, anche l'approssimazione iniziale $x0$, il numero massimo di iterazioni e la tolleranza, mentre in output viene data la soluzione approssimata, il numero di iterazioni effettuate, e la storia della norma relativa del residuo durante tutte le iterazioni fatte.

2. Mediante script, usa la funzione `Jacobi` per risolvere il sistema lineare con
   $$A = \begin{bmatrix} 3 & 1 & 0 & -1 & 0 \\ -1 & 4 & 0 & 1 & 1 \\ 1 & 0 & 6 & -1 & 1 \\ 0 & 0 & 1 & 4 & -1 \\ -1 & 0 & 0 & 2 & 5 \end{bmatrix}, \qquad b = \mathbf{1},$$
   con `tol`=10^{-9} e `maxit`=100 e $x0$ scelto a piacere.

3. Nello script, inserisci poi i comandi per visualizzare in un grafico la storia del residuo. Nello stesso grafico, riporta la stima teorica ρ^k, al crescere dell'iterazione k, per confrontarla con la convergenza effettiva. Qui ρ è il raggio spettrale della matrice di iterazione. Inserisci una legenda per le due curve, le etichette degli assi ed il titolo del grafico.

Test di Laboratorio di Calcolo Numerico, 11 Settembre 2018
Laurea Triennale in Matematica

Al termine della prova, spedisci quanto fatto a `valeria.simoncini@unibo.it` (due files: una funzione matlab ed uno script contenente tutti i comandi degli script)

1. È data la funzione $f(x) = x^2 \sin(x)$ per $x \in [0, 2\pi]$. Fai il grafico della funzione f.

2. Implementa il metodo delle secanti per determinare gli zeri di una funzione non lineare f, includendo opportuni criteri d'arresto, ed in modo che ad ogni iterazione la funzione stampi su video sulla stessa riga: n. iterazione, residuo corrente, stima corrente dello zero della funzione f.
   In output, la funzione deve fornire il vettore di tutte le approssimazioni x_k.

3. Mediante uno script, applica la funzione Matlab/Octave cosi ottenuta alla funzione f data al punto 1., per l'approssimazione del suo **primo zero positivo**, nei seguenti casi:
   i) $x_{-1} = 3$, $x_0 = 4$
   ii) $x_{-1} = 2$, $x_0 = 5$
   iii) $x_{-1} = 5$, $x_0 = 6$
   Commenta e spiega i risultati nello script.

4. Riporta in uno stesso grafico, per tutti e tre i casi, **il valore dell'errore** durante le iterazioni, includendo *labels*, *titolo* e *legend*.

Test di Laboratorio di Calcolo Numerico, 8 Luglio 2020
Laurea Triennale in Matematica.
Test Modulo II

Crea uno script con il tuo nome e cognome, `NomeCognome.m`, contenente tutto lo svolgimento.

Dato l'integrale $\mathcal{I}(f) = \int_{-1}^{1} f(t)dt$, considera la formula di quadratura

$$\widetilde{\mathcal{I}}(f) := \alpha_1 \int_{-1}^{-\frac{1}{2}} f(t)dt + \alpha_2 f\left(-\frac{1}{4}\right) + \alpha_3 f\left(\frac{1}{4}\right) + \alpha_4 \int_{\frac{1}{2}}^{1} f(t)dt.$$

Nota: eccetto il caso di funzione integranda costante, tutti gli integrali devono essere calcolati con le formule di quadratura richieste.

1. (*Necessario per la sufficienza*) Usando la formula composita dei rettangoli con 40 sottointervalli dell'intervallo di integrazione, determina α_j, $j = 1, .., 4$ in modo che il grado di esattezza sia 3.

2. (*Necessario per la sufficienza*) Per $f(x) = x^3 \sin(x\pi)\cos(x\pi)$, mostra nella command window di Matlab i valori come in tabella,

$\mathcal{I}(f)$	$\widetilde{\mathcal{I}}(f)$	$E_m(f)$
$\cdots$	$\cdots$	$\cdots$

   dove gli integrali presenti nelle formule sono stimati mediante la formula composita di Cavalieri-Simpson con $m = 30$ sottointervalli. L'errore E_m è definito in (*).

3. Usando ora la formula composita di Cavalieri-Simpson con m sottointervalli per stimare gli integrali presenti nelle formule, verifica che il grado di esattezza della formula trovata non è uguale a 4, mostrando nella command window di Matlab come varia l'errore

   $$(*) \qquad E_m(f) = |\mathcal{I}(f) - \widetilde{\mathcal{I}}(f)|,$$

   per $f = p_4$ all'aumentare m del numero di sottointervalli (p_4 è un opportuno polinomio di grado 4), secondo la seguente tabella:

m	E_m
$\vdots$	$\vdots$

   Fai un grafico dell'errore al crescere di $m \in \{5, \ldots, 50\}$, includendo etichette, titolo e legenda.

Test di Laboratorio di Calcolo Numerico, 8 Luglio 2020
Laurea Triennale in Matematica
Test Modulo I

È dato il sistema lineare $Ax = b$, con $A \in \mathbb{R}^{m\times m}$ generata mediante la funzione `A=crea_dati(n);`, e b vettore con elementi presi da una distribuzione casuale uniforme in $(0, 1)$; Si ottiene $m = (n + 1)^2$.

1. Sia P la parte tridiagonale della matrice A, e considera lo splitting

$$A = P - N.$$

Scrivi una funzione `[x,its,vecres,flag]=risolvo(A,b,P,x0,maxit,tol);` per la risoluzione iterativa del sistema considerato mediante lo splitting proposto. Le variabili in input rappresentano i dati, l'approssimazione iniziale (`x0`), il numero massimo di iterazioni (`maxit`) e la tolleranza richiesta (`tol`). **Le eventuali risoluzioni con P devono prevedere l'uso di una propria funzione, adatta alla situazione. No backslash di Matlab.**
Oltre ai soliti criteri d'arresto, prevedi l'uscita anche per stagnazione, cioè nel caso in cui

$$\frac{\|x_k - x_{k-1}\|}{\|x_k\|} < 10^{-4}.$$

Le variabili in output sono la soluzione approssimata finale, il numero di iterazioni effettivo, il vettore (`vecres`) contenente la storia della norma del residuo durante l'iterazione, ed il tipo di uscita (`flag`): convergenza, stagnazione, max iterazioni.
Durante l'esecuzione, mostra nella command window di Matlab i dati come in tabella:

m	k	$\|b - Ax_k\|/\|b\|$	$\|x_k - x_{k-1}\|/\|x_k\|$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

dove k è l'iterazione corrente e x_k l'approssimazione all'iterazione k.

2. Scrivi uno script (`NomeCognome.m`) che, per $n = 5, 10, 15, 20$, crei i dati e risolva il sistema con la funzione `risolvo.m` (`maxit`=300, `tol`=10^{-6}). Per ogni valore di n riporta su uno stesso grafico la storia della convergenza, inserendo etichette, titolo e legenda, in modo che sia chiaro a quale convergenza si riferisce ogni curva.
Al termine della risoluzione per ogni n, mostra nella command window le seguenti frasi, inserendo i valori corrispondenti in modo automatico:

```
Il metodo ha terminato in:  ...  iterazioni con norma del residuo:....
                    Il metodo ha terminato per:
                                ....
```

Inserisci quindi il motivo della terminazione:

`convergenza/numero massimo di iterazioni/stagnazione`.

```
---------------------------------------
function A=crea_dati(n)

n=n+1;
h=1/(n-1);
x=0:h:1;

T=1/h*( -diag(exp(x(2:n)),-1)+2*diag(exp(x(1:n)))-diag(exp(x(1:n-1)),1) );
T=sparse(T);
A=kron(speye(n),T)+kron(T,speye(n));
```

Test di Laboratorio di Calcolo Numerico, 02 Settembre 2020
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini200619, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo I".

Considera la successione di matrici $\{A_m\}$, $m = 2, 3, \ldots$ con $A_m \in \mathbb{R}^{n\times m}$, $n \geq m$ e $0 \neq b \in \mathbb{R}^n$ considera il problema

$$\min_{x\in\mathbb{R}^m} \|b - A_m x\| \tag{0.1}$$

con A_m avente rango massimo.

1. La successione $\{A_m\}$ è data da $A_m = [A_{m-1}, a_m]$, cioè la matrice A_m è ottenuta aggiungendo una nuova colonna ad A_m. Sapendo che $A_1 = Q_1 R_1$ con $Q_1 = A_1/\|A_1\|$ e $R_1 = \|A_1\|$, crea una funzione `[Qm1,Rm1]=gram_update(Qm,Rm,am);` che **aggiorna** la fattorizzazione di Gram-Schmidt $A_m = Q_m R_m$ del passo precedente, senza ricalcolare `Qm, Rm`.
2. Crea uno script Matlab con il tuo nome e cognome (es. `valeriasimoncini.m`) ed in esso riporta lo svolgimento:

   Siano $n = 10, m_{max} = 9$. Fissato $b(j) = j \cdot (-1)^j$, $j = 1, \ldots, n$, all'interno di un ciclo in $m = 2, \ldots, m_{max}$:

   (a) Crea $A_m = [A_{m-1}, a_m]$, con a_m vettore di elementi casuali da una distribuzione normale. (cioè A_{m-1} deve essere la stessa dell'iterazione precedente!)
   (b) Mediante la funzione `gram_update` aggiorna la fattorizzazione QR di A_m e risolvi il problema (0.1), con x_m la soluzione determinata (includi la funzione per la risoluzione dell'eventuale sistema).

   (c) Fai un display delle dimensioni di A_m, del suo rango e del condizionamento `kappa` di $A_m^T A_m$ come nella seguente tabella (inclusa la riga di descrizione)

n	m	rango(A_m)	`kappa`	$\|x_m - x_*\|$	$\|b - A_m x_m\|$
:	:	:	:	:	:
:	:	:	:	:	:

   dove x_* è la soluzione ottenuta con backslash di Matlab.
   (d) Fai un grafico riportante la norma dell'errore e quella del residuo, al variare di m. Includi etichette, titolo e legenda.

Test di Laboratorio di Calcolo Numerico, 2 Settembre 2020
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini200619, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

Dati i *nodi* $\{x_i\}$, $i = 0, \dots n$, $x_i \in [a, b]$, scrivi la funzione matlab

```
[valmax]=maxomega(xx,tolr,tola,tolf,maxit);
```

che determina una stima di

$$\|\omega\|_\infty = \max_{x\in[a,b]} |\omega(x)|, \quad \text{dove} \quad \omega(x) = \prod_{i=0}^{n}(x - x_i),$$

mediante il metodo di Newton (`tolr,tola,tolf=1e-3, maxit=100`, `xx` vettore dei nodi). A tal fine, ricorda che

1. Sono possibili massimi locali, quindi è suggeribile valutare la presenza di estremanti per ogni intervallo $[x_i, x_{i+1}]$, $i = 0, \dots, n-1$, e poi determinare il massimo tra gli estremanti ottenuti, prendendo come dato iniziale per Newton il punto medio dell'intervallo $[x_i, x_{i+1}]$.
2. I massimi di $|\omega(x)|$ si ottengono a partire dagli zeri della funzione derivata $\omega'(x)$.
3. La derivata prima e seconda di $\omega(x)$ si possono approssimare mediante differenze finite,

   $$\omega'(x) \approx \frac{\omega(x+h) - \omega(x-h)}{2h}, \quad \omega''(x) \approx \frac{\omega(x-h) - 2\omega(x) + \omega(x+h)}{h^2},$$

   (Poni $h = 10^{-4}$).
4. I coefficienti del polinomio nodale $\omega(x)$ con radici nel vettore `xx` si possono determinare con la funzione Matlab `coef=poly(xx);`, ed il valore del polinomio di coefficienti `coef` nel punto `t` si trova usando `val=polyval(coef,t);`.

Crea uno script con il tuo nome e cognome, `NomeCognome.m`, contenente tutto lo svolgimento:

Per $n = 4, 5, \dots, 8$, considera $a = \frac{1}{4}\pi, b = \pi$, e la successione di nodi equispaziati $\{x_i\}$ in $[a, b]$. Approssima la funzione

$$f(x) = \frac{\sin(x)}{x^2}, \quad x \in [a, b],$$

mediante il polinomio di Lagrange P_n di grado n, e riporta un display con la seguente tabella,

n	$\|P_n - f\|_\infty$	E
:	:	:
:	:	:

dove E è la stima dell'errore calcolata sfruttando la stima di $\|\omega\|_\infty$ ottenuta sopra.

Test di Laboratorio di Calcolo Numerico, 20 Novembre 2020
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini201120, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

È data la funzione

$$f(x) = \int_0^x t^2 \sin(t) dt, \quad x \in [0, 2\pi].$$

1. Fai il grafico della funzione f ed individua un intervallo opportuno dove possa essere il suo punto di massimo.
2. Scrivi la funzione matlab

   ```
   [valmax]=maxf(f1,x0,tolr,tola,tolf,maxit);
   ```

   che approssima il valore di x^* tale che $x^* = \text{argmax}_{x\in[0,2\pi]} f(x)$ mediante il metodo di Newton (individua la funzione `f1` da inserire). Fai il display della storia della convergenza.
   Nel caso tu abbia bisogno di fare la derivata di una funzione g, ricorda che la derivata prima di g in un punto x può essere approssimata mediante differenze finite centrate,

   $$g'(x) \approx \frac{g(x+h) - g(x-h)}{2h}, \quad h > 0.$$

3. Verifica che il valore ottenuto $(x^*, f(x^*))$ corrisponda al valore cercato aggiungendolo al grafico di f.
4. Approssima la funzione f nell'intervallo dato mediante il polinomio composito di Lagrange $P_3^{(m)}$ di grado 3 su m sottointervalli, $m \in \{4, 6, 8, \dots 20\}$, e riporta un display con la seguente tabella,

m	$\|P_3^{(m)} - f\|_\infty$
:	:
:	:

Test di Laboratorio di Calcolo Numerico, 15 Giugno 2020
Laurea Triennale in Matematica
Test Modulo I

Crea uno script con il tuo nome e cognome, contenente tutto lo svolgimento.

È dato il sistema lineare triangolare a blocchi

$$\begin{bmatrix} A & B \\ 0 & A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{0.1}$$

con $A \in \mathbb{R}^{n\times n}$ di tipo Hessenberg superiore e non singolare, e $B \neq 0$.

1. Scrivi una funzione `[A1,A2]=factor(A);` con costo computazionale $\mathcal{O}(n^2)$ per fattorizzare A come $A = A_1A_2$ con A_2 triangolare superiore e A_1 bidiagonale inferiore (supponendo questo sia possibile);

2. Scrivi una funzione `[x,y]=risolvo(A1,A2,B,b1,b2);` che risolva il sistema lineare (0.1) tenendo conto della struttura dei dati;

3. Per n fissato, crea i vettori $b_1, b_2 \in \mathbb{R}^n$ con elementi presi da una distribuzione casuale normale. Crea inoltre $B \in \mathbb{R}^{n\times n}$ come matrice di elementi casuali presi da una distribuzione uniforme in $(0, 1)$. Infine, crea A come la parte Hessenberg superiore della matrice $A_0 = uv^T + I_n$, con $u, v \in \mathbb{R}^n$ vettori presi da una distribuzione casuale normale ed $I_n \in \mathbb{R}^{n\times n}$ la matrice identità.
   Infine, crea la soluzione "esatta" z_* del problema (0.1), ottenuta con "\" di Matlab.

4. Per $n = 100, 200, \ldots, 1000$, posto $z = [x; y]$, risolvi il sistema con le funzioni create, e fai il display come nella seguente tabella:

n	$\|z - z_*\|_2/\|z_*\|_2$	CPU time
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

   dove "CPU time" è il tempo completo di risoluzione per ogni n.

Test di Laboratorio di Calcolo Numerico, 19 Novembre 2020
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini191120, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo I".

Siamo interessati alla risoluzione del sistema lineare

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad b = \mathbf{1} \tag{0.1}$$

mediante un metodo iterativo stazionario. Per $n = n_0^2$ la matrice A ha la seguente struttura,

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n_0} \\ A_{2,1} & A_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_{n_0 1} & \dots & \dots & A_{n_0,n_0} \end{bmatrix},$$

con i blocchi diagonali $A_{i,i} \in \mathbb{R}^{n_0\times n_0}$, $i = 1, \dots n_0$ non singolari, definita in Matlab come

```
n0=10;
T=diag(2*ones(n0,1))+diag(-ones(n0-1,1),1)+diag(-ones(n0-1,1),-1);
A=kron(diag(rand(n0,1)),T)+kron(T,eye(n0));
```

1. Crea la matrice D diagonale a blocchi, avente come blocchi diagonali le matrici $A_{i,i}$, $i = 1, \dots n_0$. Per un $\omega \in \mathbb{R}$ fissato, mediante una funzione

   `y=risolviD(D+omega I,z,n0)`,

   implementa un algoritmo per la risoluzione del sistema lineare $(D + \omega I)y = z$. L'algoritmo deve sfruttare la struttura a blocchi e non usare risolutori predefiniti di Matlab (es. \).
2. Sfruttando lo splitting $A = P - N := (D + \omega I) - (E + F + \omega I)$ con D come sopra, risolvi il sistema (0.1) per 10 valori di ω equispaziati nell'intervallo $[2, 3]$ mediante l'implementazione della funzione

   `[x,iter,res]=iterazione(A,b,P,x0,maxit,tol);`

   con `x0=0, maxit=300, tol=1e-4`, dove `iter` è il numero di iterazioni del metodo, ed il vettore `res` contiene la storia della norma relativa del residuo.
3. Riporta su uno stesso grafico la storia della convergenza del metodo (in termini della norma relativa del residuo) al variare di ω. Includi etichette degli assi, titolo e legenda.
4. Riporta su Command Window i risultati ottenuti, come segue

ω	# iter k	$\|x_k - x_*\|$	$\|b - Ax_k\|/\|b - Ax_0\|$
:	:	:	:
:	:	:	:

   dove x_* è la soluzione ottenuta con backslash di Matlab.

Turno 2 - Test di Laboratorio di Calcolo Numerico, 19 Novembre 2020
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini191120, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo I".

Siamo interessati alla risoluzione del sistema lineare

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad b = \mathbf{1} \tag{0.1}$$

mediante un metodo iterativo stazionario. Per $n = n_0^2$ la matrice A ha la seguente struttura,

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n_0} \\ A_{2,1} & A_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_{n_0 1} & \dots & \dots & A_{n_0,n_0} \end{bmatrix},$$

con i blocchi diagonali $A_{i,i} \in \mathbb{R}^{n_0\times n_0}$, $i = 1, \dots n_0$ non singolari, definita in Matlab come

```
T=sparse(diag(4*ones(n0,1))+diag(-ones(n0-1,1),1)+diag(-ones(n0-1,1),-1));
d=rand(n0,1);
A=kron(sparse(diag(d)),T)+kron(T,speye(n0));
```

1. Crea la matrice D diagonale a blocchi, avente come blocchi diagonali le matrici $A_{i,i} + n_0 I$, $i = 1, \dots n_0$. Mediante una funzione

   `y=risolviD(D,z,n0)`,

   implementa un algoritmo per la risoluzione del sistema lineare $Dy = z$. L'algoritmo deve sfruttare la struttura a blocchi e non usare risolutori predefiniti di Matlab (es. \).
2. Sfruttando lo splitting $A = P - N := D - (E + F)$ con D come sopra, risolvi il sistema (0.1) per 10 valori di n_0 equispaziati nell'intervallo $[10, 100]$ mediante l'implementazione della funzione

   `[x,iter,res]=iterazione(A,b,P,x0,maxit,tol);`

   con `x0=0, maxit=1000, tol=1e-4`, dove `iter` è il numero di iterazioni del metodo, ed il vettore `res` contiene la storia della norma relativa del residuo.
3. Riporta su uno stesso grafico la storia della convergenza del metodo (in termini della norma relativa del residuo) al variare di n_0. Includi etichette degli assi, titolo e legenda.
4. Riporta su Command Window i risultati ottenuti, come segue

n_0	# iter k	$\|x_k - x_*\|$	$\|b - Ax_k\|/\|b - Ax_0\|$
:	:	:	:
:	:	:	:

   dove x_* è la soluzione ottenuta con backslash di Matlab.

Turno 2 - Test di Laboratorio di Calcolo Numerico, 20 Novembre 2020
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini201120, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

È data la funzione

$$f(x) = \int_0^x (t - \cos(t))dt, \quad x \in [-\pi/4, \pi/4].$$

1. Fai il grafico della funzione f ed individua un intervallo opportuno dove possa essere il suo punto di minimo.
2. Scrivi la funzione matlab

   `[valm]=minf(f1,x0,tolr,tola,tolf,maxit);`

   che approssima il punto critico x^* di f tale che $x^* = \text{argmax}_{x\in[-\pi/4,\pi/4]} f(x)$ mediante una opportuna procedura di punto fisso (individua la funzione `f1` da inserire). Fai il display della storia della convergenza.
3. Verifica che il valore ottenuto $(x^*, f(x^*))$ corrisponda al valore cercato aggiungendolo al grafico di f.
4. Approssima la funzione f nell'intervallo dato mediante il polinomio composito di Lagrange $P_3^{(m)}$ di grado 3 su m sottointervalli, $m \in \{4, 6, 8, \dots 20\}$, e riporta un display con la seguente tabella,

m	$\|P_3^{(m)} - f\|_\infty$
:	:
:	:

Turno 1 - Test di Laboratorio di Calcolo Numerico, 21 Dicembre 2020
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini211220, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo I".

Siamo interessati alla risoluzione del problema ai minimi quadrati

$$\min_{x\in\mathbb{R}^n} \|b - Ax\| \quad A \in \mathbb{R}^{(n+1)\times n}, \quad b = [1, 1, \ldots, 1]^T \in \mathbb{R}^{n+1}$$

con A tridiagonale, di rango massimo.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Lo script chiama la funzione `LSgivens.m`, creata da te, che risolve il problema ai minimi quadrati mediante l'uso delle rotazioni di Givens, e tiene conto della struttura a banda della matrice. La funzione deve riportare in output tutte le quantità ottenute durante la fattorizzazione, oltre alla soluzione x.
   Al termine lo script mostra su Command Window in modo descrittivo la norma del residuo relativo, il tempo di CPU impiegato, la norma dell'errore rispetto alla soluzione calcolata con $x^* = A \setminus b$.

2. Lo script crea poi una modifica del problema dato,

$$\min_{x\in\mathbb{R}^{n+1}} \|b - \widetilde{A}x\| \quad \widetilde{A} = \begin{bmatrix} A & a \\ 0^T & \alpha_2 \end{bmatrix} \in \mathbb{R}^{(n+2)\times(n+1)}, \quad \widetilde{b} = [1, 1, \ldots, 1]^T \in \mathbb{R}^{n+2}$$

   con $\widetilde{A}$ ancora tridiagonale, cosicchè $a = \alpha_0 e_n + \alpha_1 e_{n+1}$.

3. Infine, sfruttando quanto calcolato nella procedura al punto (1), lo script calcola la nuova soluzione $x \in \mathbb{R}^{n+1}$. Al termine lo script mostra su Command Window in modo descrittivo la norma del residuo relativo, il tempo di CPU impiegato, la norma dell'errore rispetto alla soluzione calcolata con $x^* = \widetilde{A} \setminus \widetilde{b}$.

- Verifica la correttezza dello script con i seguenti dati:

```
n=10;
A=diag(randn(n+1,1))+diag(randn(n,1),1)+diag(randn(n,1),-1);
A=A(1:n+1,1:n);
a=[0;...; 0; 1.2;-4.3];  alpha2 = 1.4;
```

Turno 2 - Test di Laboratorio di Calcolo Numerico, 21 Dicembre 2020
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini211220, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo I".

Siamo interessati alla risoluzione del problema ai minimi quadrati

$$\min_{x\in\mathbb{R}^n} \|b - Ax\| \quad b = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^{n+1}, \; A = \begin{bmatrix} \times & \times & 0 & & & \\ \times & \times & \times & 0 & & \\ \times & \times & \times & \times & 0 & \\ 0 & \times & \times & \times & \times & 0 \\ & 0 & \times & \times & \times & \times \\ & & 0 & \times & \times & \times \\ & & & 0 & \times & \times \end{bmatrix} \in \mathbb{R}^{(n+1)\times n}$$

con A di rango massimo avente banda inferiore 2 e banda superiore 1 (vedi esempio sopra).

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Lo script chiama la funzione `LSgivens.m`, creata da te, che risolve il problema ai minimi quadrati mediante l'uso delle rotazioni di Givens, e tiene conto della struttura a banda della matrice. La funzione deve riportare in output tutte le quantità ottenute durante la fattorizzazione, oltre alla soluzione x_G.
   Al termine lo script mostra su Command Window in modo descrittivo la norma del residuo relativo, il tempo di CPU impiegato, la norma dell'errore rispetto alla soluzione calcolata con $x^* = A \setminus b$.

2. Usando l'output della funzione `LSgivens.m`, lo script determina quindi la matrice Q_1 avente colonne ortogonali tale che $\text{Range}(Q_1) = \text{Range}(A)$ e mostra su Command Window in modo descrittivo le quantità corrispondenti a
   $$\|b - Ax\|_2, \qquad \|(I - Q_1Q_1^T)b\|_2$$
   (la matrice $Q_1Q_1^T$ non deve essere calcolata esplicitamente)

3. Sfruttando la matrice triangolare superiore a banda ottenuta nel punto (1), lo script determina la soluzione del problema dato x_N mediante la risoluzione dell'equazione normale. Su Command Window vengono riportate le quantità corrispondenti a
   $$\|x_N - x^*\|_2, \qquad \|x_G - x^*\|_2$$

- Verifica la correttezza dello script con i seguenti dati:

```
n=10;
A=diag(randn(n+1,1))+diag(randn(n,1),1)+diag(randn(n,1),-1)+diag(randn(n-1,1),-2);
A=A(1:n+1,1:n);
```

Aula - Test di Laboratorio di Calcolo Numerico, 21 Dicembre 2020
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini211220, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo I".

Siamo interessati alla risoluzione del sistema lineare

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad b = [1, 1, \ldots, 1]^T \tag{0.1}$$

mediante un metodo iterativo stazionario di tipo splitting, con $A = P - N$ e P invertibile e triangolare superiore.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Lo script chiama la funzione `calcolarho.m`, creata da te, che determina in modo accurato il raggio spettrale ρ della matrice di iterazione del metodo iterativo, minimizzando il costo computazionale e la memoria usata (la matrice di iterazione **non** deve essere costruita esplicitamente).

2. Lo script chiede all'utente se continuare o meno, a seconda del valore di ρ (uso del comando `input`). Nel caso l'utente decida di continuare, lo script informa su Command Window della continuazione dell'algoritmo; altrimenti lo script termina previo avviso sulla Command Window che il metodo è stato bloccato.

3. Nel caso la procedura continui, lo script chiama la funzione `iterazione.m`, creata da te, che implementa l'iterazione considerata con soglia di arresto 10^{-8}; usa le tue funzioni per risolvere il sistema ad ogni iterazione, non \.
   Riporta su Command Window l'andamento del metodo in modo descrittivo, ed al termine mostra il grafico della storia della convergenza (norma relativa del residuo), includendo etichette e titolo.

- Verifica la correttezza dello script con i seguenti dati: P la parte triangolare superiore di A, ed A ottenuta come

```
n0=30;
T=diag(3*ones(n0,1))+diag(-ones(n0-1,1),1)+diag(-ones(n0-1,1),-1);
T1=diag(2*ones(n0,1))+diag(-ones(n0-1,1),1)+diag(-ones(n0-1,1),-1);
A=sparse(kron(eye(n0),T1)+kron(T,eye(n0)));
```

Test di Laboratorio di Calcolo Numerico, 07 Gennaio 2021
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

Considera la funzione $f(x) = \int_0^x t^2 \cos(t)\sin(t)\mathrm{dt}$, per $x \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.
Nota: Approssima f mediante una formula di quadratura composita di grado 2, con almeno 10 intervalli.

1. Fai il grafico di f nell'intervallo considerato.
2. Per $n+1$ nodi con $n \in \{6, 10, 14\}$, determina la spline cubica interpolante "not-a-knot" e quella "completa", e sovrapponi il loro grafico a quello di f. Includi etichette, titolo e legenda.
3. Verifica sperimentalmente mediante display come nella tabella sottostante, anche per n più grande, che l'ordine di convergenza è $\mathcal{O}(h^p)$ con p opportuno, dove h è il massimo della lunghezza degli intervalli considerati:

	spline not-a-knot	spine completa
n	p	p
$\vdots$	$\vdots$	$\vdots$

Test di Laboratorio di Calcolo Numerico, 07 Gennaio 2021
Laurea Triennale in Matematica.
Test Modulo II - Turno 2

Crea una cartella chiamata col proprio nome e data (es. ValeriaSimoncini230519ModII, contenente tutti i files creati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

Considera la funzione $f(x) = x^2 \cos(x) \sin(x)$, per $x \in [-\frac{\pi}{3}, \frac{\pi}{3}]$.

1. Fai il grafico di f nell'intervallo considerato.
2. Fissata l'approssimazione iniziale $x_0 \in [-\frac{\pi}{3}, \frac{\pi}{3}]$, implementa il metodo di Newton per approssimare lo zero $x^* = 0$ di f, con tolleranza assoluta e relativa uguale a 10^{-10} e con un massimo di 1000 iterazioni.
   Nota: La derivata deve essere approssimata mediante differenze finite centrate, con passo $h = 10^{-4}$.
   Su un grafico (Figura 1) riporta la storia della convergenza sia di $|f(x_k)|$ che dell'errore $|x^* - x_k|$. Includi etichette, titolo e legenda.
3. Per 20 valori di x_0 equispaziati in $[-\frac{\pi}{3}, \frac{\pi}{3}]$ riporta su un grafico (Figura 2) come varia il numero di iterazioni per la convergenza, inserendo sull'asse delle ascisse il valore di x_0, su quello delle ordinate il numero di iterazioni.
4. Sullo stesso grafico (Figura 2), riporta anche come varia il numero di iterazioni del metodo delle secanti. Includi etichette, titolo e legenda.

Test di Laboratorio di Calcolo Numerico, 22 Gennaio 2021
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini201120, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data la funzione

$$f(x) = \int_0^x \exp(-t)(t-1)dt, \quad x \in [a,b] = [0,5].$$

1. Fai il grafico della funzione f (Figura 1) ed individua un intervallo opportuno dove possa essere il suo punto di minimo. Stima il valore di f mediante una **tua** formula di quadratura composita, con un numero di sottointervalli superiore a 10.

2. Scrivi la funzione matlab

   `[valmax]=minf(f1,x0,tolr,tola,tolf,maxit);`

   che approssima il punto critico x^* di f tale che $x^* = \text{argmin}_{x\in[a,b]} f(x)$ mediante il metodo di Newton, con parametri `tolf=tola=tolr=1e-10, maxit=100` (usa questi parametri per tutta la prova, determina la funzione `f1` adeguata). Fai il display descrittivo della storia della convergenza (num.iterazione, valore del residuo, valore dell'iterata x_k) e riportala su un grafico (Figura 2), inserendo etichette e titolo.
   Nel caso tu abbia bisogno di fare la derivata di una funzione g, ricorda che la derivata prima di g in un punto x può essere approssimata mediante differenze finite centrate,

$$g'(x) \approx \frac{g(x+h) - g(x-h)}{2h}, \quad h > 0.$$

3. Verifica che il valore ottenuto $(x_{\bar{k}}, f(x_{\bar{k}}))$ corrisponda al valore cercato aggiungendolo al grafico di f in Figura 1.

4. Riporta su un grafico (Figura 3) la storia della convergenza del metodo di Newton per una scelta di x_0 minore di $1/2$ e per un valore di x_0 maggiore di 3. Includi etichette, titolo, legenda.

5. Confronta il metodo di Newton con quello delle secanti: riporta in Figura 2 la storia della convergenza del metodo delle secanti, insieme a quella di Newton, avendo usato stesso punto iniziale del metodo di Newton. Come secondo punto iniziale usa il risultato di un paio di iterazioni del metodo di bisezione.

Test di Laboratorio di Calcolo Numerico, 22 marzo 2021
Laurea Triennale in Matematica
Test Modulo I - turno 1

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini211220, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Siamo interessati all'approssimazione mediante un metodo iterativo di tipo splitting della soluzione del sistema lineare

$$Ax = f, \qquad A \in \mathbb{R}^{n\times n}, f \in \mathbb{R}^n, \quad A = P - N,$$

con $P = \text{tridiag}(b, \underline{a}, c)$ avente diagonale $a = [3.5, 3.5, \ldots, 3.5]$ e valori non diagonali uguali a 1, ed $N = (N_0 + N_0^T)/2$ ed N_0 pentadiagonale avente valori non nulli presi da una distribuzione casuale uniforme in $(0, 1)$.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

Per $n = 1000, 2000, 3000$,

1. Lo script chiama la funzione `x = stimarho(a,b,c,N,x0,maxit,tol);` che determina una stima accurata del raggio spettrale ρ della matrice di iterazione, minimizzando il costo computazionale e la memoria usata (la matrice di iterazione $P^{-1}N$ **non** deve essere costruita esplicitamente). (`tol`= 10^{-3}, `maxit`= 1000, `x0 = rand(n,1);`)

2. Lo script chiama la funzione `x = iterazione(a,b,c,A,N,f,x0,maxit,tol);` che approssima la soluzione x usando lo splitting dato. All'interno, la funzione *iterazione* chiama a sua volta la funzione `v=tridiag(a,b,c,y)` per risolvere il sistema $Pv = y$ con $P = \text{tridiag}(b, \underline{a}, c)$ tridiagonale. (`tol`= 10^{-6}, `maxit`= 100, `x0 = zeros(n,1);`)

3. Usando una figura diversa per ogni valore di n, lo script riporta su un grafico la storia della convergenza di `iterazione`, in termini del residuo, e sovrappone la storia della convergenza prevista dall'analisi asintotica con ρ. Vengono inclusi etichette, titolo e legenda.

Test di Laboratorio di Calcolo Numerico, 22 marzo 2021
Laurea Triennale in Matematica
Test Modulo I - turno 2

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini211220, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Siamo interessati all'approssimazione mediante un metodo iterativo dell'autovalore più vicino a zero della matrice $A \in \mathbb{R}^{n\times n}$ definita come $A = P - N$, dove $P = \text{tridiag}(b, \underline{a}, c)$ avente diagonale $a = [3.5, 3.5, \ldots, 3.5]$ e valori non diagonali uguali a 1, ed $N = (N_0 + N_0^T)/2$ ed N_0 pentadiagonale avente valori non nulli presi da una distribuzione casuale uniforme in $(0, 1)$.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

Per $n = 500, 1000, 1500$,

1. Lo script chiama la funzione `[x,e] = approx_eig(a,b,c,A,N,x0,maxit,tol);` che approssima l'autocoppia con autovalore più vicino a zero. (`tol`= 10^{-4}, `maxit`= 1000, `x0 = rand(n,1);`)

2. Al suo interno, la funzione `approx_eig` deve risolvere un sistema lineare del tipo $Av = f$. A tal fine, usa un metodo iterativo stazionario che sfrutta lo splitting dato. Lo script chiama quindi la funzione `v = iterazione(a,b,c,A,N,f,x0,maxit,tol);` che approssima la soluzione v. All'interno, la funzione *iterazione* chiama a sua volta la funzione `v=tridiag(a,b,c,y)` per risolvere il sistema $Pv = y$ con $P = \text{tridiag}(b, \underline{a}, c)$ tridiagonale. (`tol`= 10^{-9}, `maxit`= 100, `x0 = zeros(n,1);`)

3. Lo script fa il grafico della storia della convergenza del metodo `approx_eig` al variare di n, includendo etichette, titolo e legenda.

Test di Laboratorio di Calcolo Numerico, 23 marzo 2021
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini201120, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data la funzione

$$f(x) = x\sin(x)^2\cos(x), \quad x \in [0, 2\pi].$$

1. Per $n = 8$, nella stessa figura (Figura 1) riporta il grafico dei polinomi caratteristici di Lagrange $\{L_i\}$, $i = 0, \dots n$ per $n + 1$ nodi $\{x_i\}$, $i = 0, \dots n$ equispaziati in $[0, 2\pi]$ (*sugg. costruisci uno handle per ogni* L_i).
2. Fai il grafico della funzione f (Figura 2)
3. Per $n = 4, 7, 10, 13$ determina il polinomio p_n di grado n interpolante f in nodi equispaziati in $[0, 2\pi]$ nella forma di Lagrange, e riporta il grafico in Figura 2.
4. Fai il grafico (Figura 3) della funzione errore $E(x) = f(x) - p_n(x)$ per $n = 4, 7, 10, 13$ e $x \in [0, 2\pi]$.
5. Dopo aver verificato graficamente che f ha un solo zero nell'intervallo $[0.5, 2]$, approssima tale zero x_f di f mediante il metodo delle secanti, riportando in una tabella la storia della convergenza (riporta iterazione, residuo, differenza tra iterate successive). In modo analogo, approssima lo zero x_p di p_n in $[0.5, 2]$ ($n = 10, 13$), calcola e visualizza in modo esplicativo l'errore relativo tra i due valori approssimati $\tilde{x}_f$, $\tilde{x}_p$ ottenuti, per entrambi i valori di n.

Test di Laboratorio di Calcolo Numerico, 28 maggio 2021
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini280521, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data l'equazione

$$f(x) = 0, \quad x \in (0, 2], \quad \text{con} \quad f(x) := x^2 - \alpha \int_0^x e^{-\sqrt{t}} dt,$$

ed $\alpha \in \{0.1, 0.5, 1, 5, 10\}$.

Nota: La funzione integrale deve essere approssimata mediante uno dei metodi sviluppati durante il corso, con parametri opportuni.

Crea lo script `nomecognome.m` che svolge quanto riportato in seguito.

1. Per $\alpha = 1$ fai il grafico della funzione f nell'intervallo considerato (Figura 1).

2. Implementa una funzione `itera.m` che determina una approssimazione del primo zero positivo dell'equazione $f(x) = 0$ mediante l'iterazione: per x_0 fissato,

$$x_{k+1} = \left(\int_0^{x_k} e^{-\sqrt{t}} dt \right)^{1/2}, \quad k = 0, 1, 2, \ldots,$$

   Per ogni valore di α, riporta sulla command window la storia della convergenza (sulla stessa riga, alfa, n. iter., x_k, residuo), con `tola=tolr=1e-6, tolf=1e-9, maxit=100`. Riporta su uno stesso grafico (Figura 2) la storia della convergenza del residuo dell'iterazione al variare di α, inserendo etichette degli assi, titolo e legenda.

Aula - Test di Laboratorio di Calcolo Numerico, 22 Gennaio 2021
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini211220, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Siamo interessati all'approssimazione dell'autovalore più piccolo in modulo della matrice

$$A = A_0 + A_0^T, \quad A_0 = \texttt{randn(n,n)}; \tag{0.1}$$

con $n = 16$, mediante un metodo iterativo.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Lo script chiama la funzione `H=miaHess(A);`, **creata da te**, che determina la matrice di Hessenberg superiore H tale che $H = Q^T A Q$, con Q matrice ortogonale (**Attenzione all'effettiva struttura di H!!**).
   La funzione inoltre misura l'accuratezza della decomposizione, riportando la quantità $\|A - QHQ^T\|$ con un display descrittivo.

2. Lo script chiama la funzione `approssimo.m`, **creata da te**, che approssima l'autovalore più piccolo in modulo di H (che corrisponde a quello di A). Fornisci ogni funzione necessaria per l'eventuale risoluzione di sistemi lineari (non usare \ di Matlab!) Riporta su Command Window l'andamento del metodo in modo descrittivo (iterazione, norma del residuo, autovalore approssimante), ed al termine mostra il grafico della storia della convergenza (norma relativa del residuo), includendo etichette e titolo.

Aula - Test di Laboratorio di Calcolo Numerico, 01 Giugno 2021
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini010621, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Dati: Per un n fissato, è data la matrice simmetrica e definita positiva $A \in \mathbb{R}^{n\times n}$, determinata dai seguenti comandi:

```
R=sprand(n,n,0.3);
A=R+R';
A=diag(A*ones(n,1))-A;
A=n*speye(n)-A;
```

Il più grande autovalore di A è uguale a $\lambda_1 = n$, ed il corrispondente autovettore x_1 ha tutte componenti uguali, con norma Euclidea unitaria, cioè $\|x_1\| = 1$.

Svolgimento: Per ogni $n \in \{100, 200, 300, 400, 500\}$, scrivi uno script che esegue i seguenti steps (usa `tol=1e-6` e `maxit=500` per i metodi iterativi):

1. Verifica computazionalmente che x_1 sia autovettore di A;
2. Chiama una funzione `eigmax2` che approssima il secondo autovalore più grande di A, con il corrispondente autovettore. A tal fine, ricorda che tale autovalore è il più grande autovalore della matrice
$$(I - x_1x_1^T)A(I - x_1x_1^T)$$
IMPORTANTE: questa matrice non deve essere costruita esplicitamente, e neppure la matrice $I - x_1x_1^T$.
Lo script riporta su di uno stesso grafico (Figura 1) la storia della convergenza del metodo iterativo utilizzato, per tutti i valori di n; include legenda, titolo, etichette degli assi;
3. Chiama una funzione `eigmin` che approssima λ_n, l'autovalore più vicino a zero di A, con il corrispondente autovettore. Lo script riporta su di uno stesso grafico (Figura 2) la storia della convergenza del metodo iterativo utilizzato, per tutti i valori di n; include legenda, titolo, etichette degli assi;
4. Approssima il numero di condizionamento di A, $\kappa(A)$.

Al termine della procedura lo script mostra sulla command window di Matlab la seguente tabella, coi valori corrispondenti

n	λ_1	$\|Ax_1 - x_1\lambda_1\|$	λ_2	$\|Ax_2 - x_2\lambda_2\|$	λ_n	$\|Ax_n - x_n\lambda_n\|$	$\kappa(A)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Test di Laboratorio di Calcolo Numerico, 22 giugno 2021
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini220621, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data l'equazione

$$F(x) = 0, \quad \text{con} \quad F(x) := \int_0^x \sin(t)\frac{t^2-1}{t^2+1}dt, \quad x \in [0, \pi]$$

Crea lo script `nomecognome.m` che svolge quanto riportato in seguito.

1. Approssima il grafico di F, per $x \in [0, \pi]$, dove l'integrale è approssimato mediante la formula $\mathcal{I}_{0,m}(x)$ (rettangoli composita) e la formula $\mathcal{I}_{2,m}(x)$ (Cavalieri-Simpson composita). Riporta i due grafici nella stessa Figura 1 (con legenda, etichette assi, titolo). Usa $m = 4$ ed $m = 8$, per un totale di 4 figure sullo stesso grafico.

2. In Figura 2, riporta il grafico della funzione errore $E_i(x) = F(x) - \mathcal{I}_{i,m}(x)$, $i = 0, 2$. Usa $m = 4$ ed $m = 8$, per un totale di 4 figure sullo stesso grafico. La funzione "esatta" $F(x)$ può essere calcolata usando $\mathcal{I}_{2,m}(x)$ con $m = 50$, per esempio.

3. Con $m = 8$ e $\mathcal{I}_{2,m}$, usa il metodo di bisezione per determinare una approssimazione del primo zero positivo di F. Mostra su display una tabella riportante l'iterazione, il valore approssimato, il residuo, la lunghezza dell'intervallo, man mano che il metodo procede. In Figura 3 mostra la storia della convergenza del metodo, in termini del residuo con legenda, etichette assi, titolo.

Aula - Test di Laboratorio di Calcolo Numerico, 22 Giugno 2021
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini220621, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Dati: È data la matrice simmetrica e definita positiva $A \in \mathbb{R}^{n\times n}$ con $n = 10\,000$, determinata dai seguenti comandi:

```
n=1e4;
e=ones(n,1);
A=spdiags([-e,2*e,-e],-1:1,n,n);
```

Svolgimento: Crea lo script `nomecognome.m` che svolge quanto riportato in seguito.

1. Determina una approssimazione accurata (residuo relativo minore di 10^{-10}) del più piccolo autovalore di A e relativo autovettore, (λ_n, v_n), mediante un metodo iterativo - gli autovalori sono ordinati in modo decrescente.
   Fai il grafico della storia del residuo e dell'errore $|\tilde{\lambda} - \lambda_n|$ (Figura 1, cioè due grafici), e della storia dell'approssimazione di $\tilde{\lambda}$ a λ_n (Figura 2, curva di $\tilde{\lambda}$ e linea corrispondente a λ_n, cioè due curve[1]), includendo legenda, etichette assi e titolo.

2. Ricordando che il più grande autovalore di

   $$(I - x_n x_n^T)A^{-1}(I - x_n x_n^T)$$

   coincide con $1/\lambda_{n-1}$, determina quindi una approssimazione accurata di (λ_{n-1}, v_{n-1}). Riporta gli stessi grafici come nel punto precedente, rispettivamente nelle Figura 3 e Figura 4.

[1] Sull'asse delle ascisse viene riportato il numero dell'iterazione, sull'asse delle ordinate il valore dell'autovalore e della sua approssimazione.

Test di Laboratorio di Calcolo Numerico, 05 luglio 2021
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini050721, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data la funzione $f : [0, 2\pi] \to \mathbb{R}$, $f(x) = \sqrt{1 + x^2}(\sin(x))^3$.

Crea lo script `nomecognome.m` che svolge quanto riportato in seguito.

1. Approssima f nell'intervallo considerato mediante splines cubiche s_3 di tipo *not-a-knot* su m sottointervalli, con $m \in \{4, 6, 8, \ldots, 16\}$.
   Figura 1 mostra il grafico di f e di s_3 al crescere di m.
   Figura 2 mostra il grafico della funzione errore $|f(x) - s_3(x)|$, $x \in [0, 2\pi]$, per $m = 6, 8$.
   Figura 3 mostra l'errore $\max_{[0,2\pi]} |f(x) - s_3(x)|$ al crescere di m.
2. Usando s_3 al posto di f, lo script usa il metodo di Newton per l'approssimazione di $x^* = \arg\max_{[0,2\pi]} f(x)$ come punto critico di f (aiutarsi con i grafici di Fig.1 per individuare un intervallo opportuno contenente x^* ed un valore basso di m per la spline).
   Figura 4 mostra la storia della convergenza del metodo iterativo (`tol`=1e-6).

Aula - Test di Laboratorio di Calcolo Numerico, 05 luglio 2021
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini050721, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Dati: Per $n \in \{100, 200, 300, 400, 500\}$, è dato il problema ai minimi quadrati

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|, \qquad A \in \mathbb{R}^{(n+1)\times n},\ b \in \mathbb{R}^{n+1} \tag{0.1}$$

con `A = A=[triu(rand(n,n));zeros(1,n)]+spdiags(2*ones(n,1),-1,n+1,n);` e b con elementi presi da una distribuzione casuale normale.

Scrivi uno script che faccia quanto segue:

1. Al variare di n, risolve l'equazione normale associata a (0.1) mediante un metodo iterativo (`tol=1e-6, maxit=1000`). Per tutti i valori di n, Figura 1 mostra la storia della convergenza (norma relativa del residuo all'aumentare del numero di iterazioni), per un totale di 5 curve. Lo script fornisce anche una tabella come la seguente:

n	cond(A)	# iter	CPU time
:	:	:	:

2. Al variare di n, risolve (0.1) mediante un metodo di tipo QR opportuno. Figura 2 mostra il costo computazionale (CPU time) all'aumentare di n;

3. Figura 3 mostra la norma relativa della differenza tra le soluzioni ottenute coi due diversi metodi, all'aumentare di n.

Lo script usa solo metodi risolutivi inclusi nella cartella da spedire, e discussi durante il corso (in particolare, lo script **non** usa funzioni ufficiali Matlab(R) predefinite).

Aula - Test di Laboratorio di Calcolo Numerico, 02 settembre 2021
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini020921, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Dati: Per $n \in \{100, 200, 300, 400, 500\}$, è dato il problema ai minimi quadrati

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|, \qquad A \in \mathbb{R}^{(n+2)\times n},\ b \in \mathbb{R}^{n+2} \tag{0.1}$$

con A avente elementi presi da una dstribuzione casuale uniforme nell'intervallo $(0, 1)$, e b con elementi presi da una distribuzione casuale normale.

Scrivi uno script che faccia quanto segue:

1. Al variare di n, determina i fattori Q ed R della fattorizzazione QR di A. Lo script fornisce anche una tabella come la seguente:

n	cond(A)	CPU time
:	:	:

2. Per ogni n, e per $a \in \mathbb{R}^{n+2}$ preso da una distribuzione uniforme come sopra, lo script riprende la fattorizzazione al punto precedente e la modifica, in modo da risolvere il problema
$$\min_{x \in \mathbb{R}^{n+1}} \|b - [A, a]x\|,$$
col minimo costo computazionale, supponendo di avere già ottenuto la fattorizzazione QR di A.
Lo script può anche eventualmente prevedere la memorizzazione dei fattori Q ed R del punto precedente, in modo da non ricalcolarli.
Figura 1 mostra il costo computazionale (CPU time) all'aumentare di n (con etichette e titolo);

3. Figura 2 mostra la norma relativa del residuo all'aumentare di n (con etichette e titolo).

Lo script usa solo metodi risolutivi inclusi nella cartella da spedire, e discussi durante il corso (in particolare, lo script **non** usa funzioni ufficiali Matlab(R) predefinite, quali per esempio "\").

Test di Laboratorio di Calcolo Numerico, 02 settembre 2021
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini020921, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data la funzione $f_\alpha : [a, b] \to \mathbb{R}$, $f_\alpha(x) = x^\alpha + x^2 - 1$.

Crea lo script `nomecognome.m` che svolge quanto riportato in seguito.

1. Per $\alpha = 0.1 : 0.1 : 2$, fa il grafico di f e lo riporta su figure distinte mediante l'uso di `subplot` (in Figura 1), includendo etichette degli assi e titolo riportante il valore di α;
2. Per ogni $\alpha = 0.1 : 0.1 : 2$, sia $x(\alpha)$ lo zero della funzione $f_\alpha(x)$ nell'intervallo $[0, 5]$. Lo script determina tale zero mediante il metodo di Newton, approssimando la derivata di f mediante differenze finite ($h = 10^{-4}$). Riporta i risultati finali della convergenza di Newton, come nella seguente tabella,

α	$x(\alpha)$	# iter	$f_\alpha(x)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

3. Lo script fa il grafico di $x(\alpha)$ al variare di α (Figura 2), per $\alpha = 0.1 : 0.1 : 2$, includendo etichette degli assi e titolo.

Test di Laboratorio di Calcolo Numerico, 16 novembre 2021
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini161121, contenente tutti i files scritti. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "Modulo II".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Sono date le funzioni

$$f : I \to \mathbb{R},\ f(x) = \cos(x)\sin(x), \quad g : I \to \mathbb{R},\ g(x) = (x+1)e^{-(x+1)},$$

con $I = [0, \pi]$.

Crea lo script `nomecognome.m` che svolge quanto riportato in seguito.

1. Fai il grafico delle due funzioni f e g nell'intervallo considerato, nella stessa figura (Figura 1).
2. Siano $A_1 = (x_1, y_1)$, $A_2 = (x_2, y_2)$ in due punti di intersezione dei grafici di f e g, con $x_1, x_2 \in I$. Approssima x_1, x_2, e quindi A_1 e A_2, mediante un metodo che abbia convergenza sperimentale di ordine superiore al primo. Riporta la convergenza in uno stesso grafico, per entrambe le approssimazioni (Figura 2).
   Nota: il metodo deve utilizzare solo valutazioni di f e g (eventuali derivate, integrali, od altro, devono essere approssimati).
3. Determina il punto medio M del segmento A_1A_2. Riporta in Figura 1 il segmento A_1A_2 ed il punto medio M, insieme ai due grafici.

Test di Laboratorio di Calcolo Numerico, 16 novembre 2021
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini161121, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Dati: Per $n \in \{100, 200, 300, 400, 500\}$, è dato il sistema lineare

$$\mathcal{A}u = b, \quad \Leftrightarrow \quad \begin{bmatrix} H & A \\ 0 & H^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \quad H, A \in \mathbb{R}^{n \times n}, f, g \in \mathbb{R}^n,$$

con H tridiagonale, $H_{i+1,i} = 1/2, H_{i,i} = (-1)^i 3, H_{i,i+1} = 3/2$, ed A piena con elementi presi da una distribuzione normale. I vettori f e g hanno entrambi elementi presi da una distribuzione uniforme nell'intervallo $(0, 1)$.

Scrivi uno script che faccia quanto segue:

1. Al variare di n, risolve il sistema lineare dato, sfruttando una sola fattorizzazione utilizzabile per H e la sua trasposta. Riporta i seguenti dati in tabella:

n	cond($\mathcal{A}$)	cond(H)	residuo	CPU time
:	:	:	:	:

2. In Figura 1 mostra il costo computazionale (CPU time) all'aumentare di n (con etichette e titolo);

3. In Figura 2 mostra la norma del residuo all'aumentare di n (con etichette e titolo).

Lo script usa solo metodi risolutivi inclusi nella cartella da spedire, e discussi durante il corso (in particolare, lo script **non** usa funzioni ufficiali Matlab(R) predefinite, quali per esempio "\").

Test di Laboratorio di Calcolo Numerico, 20 Dicembre 2021
Laurea Triennale in Matematica
Test Modulo I. n.3

Crea una cartella chiamata col tuo nome e data (es. nomecognome201221), contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nomecognome Modulo I".

Siamo interessati all'approssimazione dell'autovalore più vicino a zero, e corrispondente autovettore, della matrice

$$A = \begin{bmatrix} 0 & M \\ M^T & \alpha D \end{bmatrix},$$

con $M \in \mathbb{R}^{n\times n}$, $D \in \mathbb{R}^{n\times n}$ dove $M = \text{penta}(-1, -1, \underline{4}, -2, -1)$, D diagonale con elementi non zero presi da una distribuzione uniforme e $\alpha = 10^{-7}$.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Lo script crea i dati per $n = 100$.
2. Lo script chiama la funzione `approssima.m`, creata da te, che approssima l'autocoppia cercata mediante un metodo iterativo, con soglia di arresto 10^{-10} e maxit=1000; usa le tue funzioni per risolvere eventuali sistemi ad ogni iterazione, non \, sfruttando la presenza dei blocchi in modo opportuno.
   Al termine dell'iterazione lo script mostra il grafico della storia della convergenza (norma relativa del residuo), includendo etichette e titolo.
3. Per $\alpha = 10^{-7}, 10^{-6}, \ldots, 10^{-2}$, lo script crea i dati e risolve il problema, riportando in una tabella (display su Command Window) i seguenti valori:

α	# iter	CPU time	autoval approx	norma finale residuo
⋮	⋮	⋮	⋮	⋮

Test di Laboratorio di Calcolo Numerico, 20 Dicembre 2021
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. nomecognome201221), contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nomecognome Modulo I".

Siamo interessati alla risoluzione del sistema lineare

$$Ax = b, \quad A \in \mathbb{R}^{(n+m)\times(n+m)}, \quad b = [1, 1, \ldots, 1]^T$$

mediante un metodo iterativo stazionario di tipo splitting, $A = P - N$, con

$$A = \begin{bmatrix} T & \frac{1}{2}B^T \\ B & -D \end{bmatrix}, \quad P = \begin{bmatrix} T & 0 \\ B & -D \end{bmatrix}$$

dove $D = I + 4\widehat{D} \in \mathbb{R}^{m\times m}$ con $\widehat{D}$ diagonale avente elementi da distribuzione casuale uniforme, $B \in \mathbb{R}^{m\times n}$ con elementi da distribuzione casuale uniforme, e $T = \text{tridiag}(-1, \underline{3}, -1) \in \mathbb{R}^{n\times n}$.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Lo script crea i dati per $n = 10$, $m = 6$.

2. Lo script chiama la funzione `iterazione.m`, creata da te, che implementa l'iterazione di tipo splitting definita sopra, con soglia di arresto 10^{-8} e maxit=5000; usa le tue funzioni per risolvere il sistema ad ogni iterazione, non \, sfruttando la presenza dei blocchi in modo opportuno.
   Al termine dell'iterazione lo script mostra il grafico della storia della convergenza (norma relativa del residuo), includendo etichette e titolo.

3. Per $n = 100$ ed $m \in \{10, 20, \ldots, 100\}$, lo script risolve il problema e riporta in una tabella (display su Command Window) i seguenti valori:

m	# iter	CPU time	norma residuo
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Test di Laboratorio di Calcolo Numerico, 20 Dicembre 2021
Laurea Triennale in Matematica
Test Modulo I. n.2

Crea una cartella chiamata col tuo nome e data (es. nomecognome201221), contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nomecognome Modulo I".

Siamo interessati all'approssimazione dell'autovalore più vicino a zero, e corrispondente autovettore, della matrice

$$A = \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix},$$

con $M \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{(n-1)\times n}$ dove $M = \text{penta}(-1, -1, \underline{4}, -1, -1)$ e $B = \text{upperbidiag}(\underline{1}, -1)$. Si richiama alla memoria la seguente fattorizzazione di A:

$$A = \begin{bmatrix} M & 0 \\ B & I \end{bmatrix} \begin{bmatrix} I & W \\ 0 & -S \end{bmatrix}, \quad S = BM^{-1}B^T,\ W = M^{-1}B^T.$$

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Lo script crea i dati per $n = 100$, e **crea esplicitamente** S e W, sfruttando le funzioni scritte da te.

2. Lo script chiama la funzione `approssima.m`, creata da te, che approssima l'autocoppia cercata mediante un metodo iterativo, con soglia di arresto 10^{-8} e maxit=1000; usa le tue funzioni per risolvere eventuali sistemi ad ogni iterazione, non \, sfruttando la presenza dei blocchi in modo opportuno.
   Al termine dell'iterazione lo script mostra il grafico della storia della convergenza (norma relativa del residuo), includendo etichette e titolo.

3. Per $n = 50, 60, \ldots, 150$, lo script crea i dati e risolve il problema, riportando in una tabella (display su Command Window) i seguenti valori:

n	# iter	CPU time	autoval approx	norma finale residuo
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Test di Laboratorio di Calcolo Numerico, 11 gennaio 2022
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini110122, contenente tutti i files utili. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

È data la funzione

$$F : I \to \mathbb{R}, \quad F(x) = \int_0^x f(t)dt, \quad f(t) = e^{-t}\sin(t),$$

con $I = [0, \pi]$.

Crea lo script `nomecognome.m` che svolge quanto riportato in seguito.

1. Crea i dati.
2. Chiama la funzione `quadratura.m`. Per ogni fissato x, tale funzione determina in modo automatico i coefficienti $\alpha_0, \beta_0, \alpha_1, \beta_1$ che definiscono la seguente formula di quadratura

   $$\widetilde{F}(x) = \sum_{i=0}^{1} \alpha_i f(t_i) + \sum_{i=0}^{1} \beta_i f'(t_i),$$

   In modo che la formula abbia il massimo grado di esattezza, con $t_0 = 0$, $t_1 = x$. La derivata è approssimata mediante differenze finite con passo di discretizzazione $h = 10^{-4}$.
3. Fa il grafico della funzione errore $E(x) := F(x) - \widetilde{F}(x)$, $x \in [0, \pi]$, includendo etichette degli assi e titolo. La funzione F è approssimata mediante la formula di quadratura di Cavalieri-Simpson composita, con un minimo di 5 sottointervalli.
4. Determina una buona approssimazione di $\|E\|_\infty$ in $[0, \pi]$, usando almeno 40 valori di x uniformemente distribuiti nell'intervallo $[0, \pi]$. Lo script riporta il valore ottenuto su schermo, mediante il comando `fprintf`, in modo descrittivo.

Test di Laboratorio di Calcolo Numerico, 11 gennaio 2022
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini110122, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: È dato il problema ai minimi quadrati

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|, \qquad b = \mathbf{1} \in \mathbb{R}^{n+1}, \quad A \in \mathbb{R}^{(n+1)\times n},$$

con

$$A = LR, \quad L = \begin{bmatrix} L_{11} & & & & & \\ L_{21} & L_{22} & & & & \\ & L_{32} & L_{33} & & & \\ & & \ddots & \ddots & & \\ & & & L_{n,n-1} & L_{n,n} \\ & & & & L_{n+1,n} \end{bmatrix} \in \mathbb{R}^{(n+1)\times n}$$

(L è bidiagonale inferiore) e $R \in \mathbb{R}^{n\times n}$ triangolare superiore. Dove non sono zero, entrambe L ed R hanno valori random (da distribuzione uniforme in $(0, 1)$) ed hanno rango massimo.

La matrice A non deve essere costruita esplicitamente!.

Scrivi uno script che faccia quanto segue:

1. Eventualmente chiamando una funzione apposita `risolvi.m`, risolve i problema ai minimi quadrati per $n = 10, 15, 20, \ldots, 50$, mediante una procedura che sfrutta la forma già fattorizzata della matrice A. Riporta inoltre su display i seguenti dati,

n	cond(L)	cond(R)	norma residuo	CPU time
:	:	:	:	:

2. In Figura 1 riporta la norma del residuo all'aumentare di n, con etichette e titolo.

Lo script usa solo metodi risolutivi inclusi nella cartella da spedire, e sviluppati durante il corso anche dallo studente (in particolare, lo script **non** usa funzioni ufficiali Matlab(R) predefinite non di base, quali per esempio "\").

Test di Laboratorio di Calcolo Numerico, 31 gennaio 2022
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. valeriasimoncini310122) contenente tutti i files creati/usati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Considera la funzione $f(x) = \int_{-\pi/2}^{x} t\cos(t)\sin(t)\mathrm{dt}$, per $x \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.

Nota: Per le valutazioni di f, approssima l'integrale mediante una formula di quadratura composita di grado 2, con almeno 10 intervalli.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Fai il grafico di f (Figura 1) nell'intervallo considerato.
2. Per $n+1$ nodi con $n \in \{4, 6, 8\}$, determina la spline cubica interpolante con condizioni sulla derivata agli estremi, e sovrapponine il grafico a quello di f in Figura 1.
3. Verifica sperimentalmente e riporta mediante display come nella tabella sottostante, anche per n più grande, che l'ordine di convergenza è $\mathcal{O}(h^p)$ con p opportuno, dove h è il massimo della lunghezza degli intervalli considerati:

	spine completa
n	p
$\vdots$	$\vdots$

4. Per lo stesso numero di nodi del punto 2, scegliendo nodi di Chebyshev, determina il polinomio interpolatorio p_n approssimante f, ed aggiungilo alla Figura 1. Includi etichette, titolo e legenda per tutte le curve presenti.
   Fai il grafico (Figura 2) dell'errore $E(x) = p_n(x) - f(x)$ con $x \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. Includi etichette e titolo.

Test di Laboratorio di Calcolo Numerico, 31 gennaio 2022
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini310122, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: È dato il sistema lineare $AX = B$ con più vettori noti, dove $A = (a_{i,j}) \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times s}$, e

$$a_{k,k} = 3k, \quad a_{k+1,k} = \frac{1}{k}, \quad a_{k,k+1} = -\frac{1}{k}, \quad \forall k;$$

la matrice $B = [b_1, \ldots, b_s]$ ha elementi presi da una distribuzione casuale normale.

Crea uno script chiamato `nomecognome.m` che esegue quanto segue:

1. Chiama una funzione `sor.m` che risolve il sistema lineare dato mediante il metodo SOR(ω). In particolare:
   - Il metodo risolve per tutti i vettori noti simultaneamente, cioè SOR è applicato all'intera matrice B (lo stesso vale per i risolutori interni, creati appositamente - non usa "\");
   - Il criterio d'arresto usa la norma di Frobenius della matrice residuo $R_k = B - AX_k$ (vedi `help norm`) e la tolleranza 10^{-8};

2. Per $n = 1000$, risolve il sistema con almeno 50 valori di ω uniformemente scelti nell'intervallo $(0.5, 1.5)$, e riporta su un grafico (Figura 1) il numero di iterazioni necessarie per la convergenza (inserisce etichette e titolo). Riporta sul display, in modo descrittivo, la stima del valore ottimo ω_{opt} (per cui si ha min. numero di iterazioni) ottenuta sperimentalmente tra i valori usati.

3. Per il valore stimato di ω_{opt}, riporta su un grafico (Figura 2), la storia della convergenza, insieme alla storia della convergenza teorica ottenuta con il fattore asintotico di convergenza ρ (ρ *può essere ottenuto usando le funzioni predefinite di matlab*). Include etichette, legenda e titolo.

Test di Laboratorio di Calcolo Numerico, 31 gennaio 2022
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini310122, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: È dato il sistema lineare $AX = B$ con più vettori noti, dove $A = (a_{i,j}) \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times s}$, e

$$a_{k,k} = 3k, \quad a_{k+1,k} = \frac{1}{k}, \quad a_{k,k+1} = -\frac{1}{k}, \quad \forall k;$$

la matrice $B = [b_1, \ldots, b_s]$ ha elementi presi da una distribuzione casuale normale.

Crea uno script chiamato `nomecognome.m` che esegue quanto segue:

1. Chiama una funzione `itera.m` che risolve il sistema lineare dato mediante un metodo di tipo splitting, con $A = P(\omega) - N(\omega)$ e $P(\omega) = \frac{1}{\omega}D - F$, dove $-F$ è la parte strettamente **superiore** di A. In particolare:
   - Il metodo risolve per tutti i vettori noti simultaneamente, cioè SOR è applicato all'intera matrice B (lo stesso vale per i risolutori interni, creati appositamente - non usa "\");
   - Il criterio d'arresto usa la norma di Frobenius della matrice residuo $R_k = B - AX_k$ (vedi `help norm`) e la tolleranza 10^{-8};

2. Per $n = 1000$, risolve il sistema con almeno 50 valori di ω uniformemente scelti nell'intervallo $(0.5, 1.5)$, e riporta su un grafico (Figura 1) il numero di iterazioni necessarie per la convergenza (inserisce etichette e titolo). Riporta sul display, in modo descrittivo, la stima del valore ottimo ω_{opt} (per cui si ha min.numero di iterazioni) ottenuta sperimentalmente tra i valori usati.

3. Per il valore stimato di ω_{opt}, riporta su un grafico (Figura 2), la storia della convergenza, insieme alla storia della convergenza teorica ottenuta con il fattore asintotico di convergenza ρ (ρ *può essere ottenuto usando le funzioni predefinite di matlab*). Include etichette, legenda e titolo.

Test di Laboratorio di Calcolo Numerico, 4 aprile 2022
Laurea Triennale in Matematica
Test Modulo I - Turno 1

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini040422, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Data la matrice $A \in \mathbb{R}^{n\times n}$ di tipo Hessenberg superiore, approssima il suo autovalore più piccolo in valore assoluto ed il corrispondente autovettore, mediante la procedura descritta nel seguito. La matrice `A=creamat(n)`; è ottenuta richiamando la seguente funzione matlab

```
function A=creamat(n)

X=randn(n);
G=X*diag(rand(n,1))/X;
A=hess(G);
```

Nel seguito usa $n = 100$.

Crea uno script chiamato `nomecognome.m` che esegue quanto segue:

1. Crea la matrice;
2. Chiama una funzione `decomp.m` da te creata, che determina L bidiagonale e U triangolare superiore tali che $A = LU$. L'implementazione deve tenere conto della struttura di A;
3. Chiama una funzione `itera.m` da te creata, che approssima l'autocoppia desiderata (usa maxit=1000 e tol=10^{-6} per i criteri d'arresto) sfruttando in modo opportuno la decomposizione del punto precedente.
   Ad ogni iterazione, la funzione mostra su display in modo descrittivo il numero di iterazioni, l'autovalore approssimato corrente e la norma del residuo associato. Al termine, la funzione riporta in Figura 1, con etichette e titolo, la storia della norma del residuo, ed in Figura 2 la storia dell'autovalore approssimante, con etichette e titolo.

Test di Laboratorio di Calcolo Numerico, 4 aprile 2022
Laurea Triennale in Matematica
Test Modulo I - Turno 2

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini040422, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Data la matrice $A \in \mathbb{R}^{n\times n}$ simmetrica, approssima il suo autovalore più piccolo in valore assoluto ed il corrispondente autovettore, mediante la procedura descritta nel seguito. La matrice `A=creamat(n)`; è ottenuta richiamando la seguente funzione matlab

```
function A=creamat(n)

X=orth(randn(n));
A=X*diag(randn(n,1))*X';
```

Nel seguito usa $n = 100$.

Crea uno script chiamato `nomecognome.m` che esegue quanto segue:

1. Crea la matrice;
2. Chiama una funzione `decomp.m` da te creata, che determina Q ortogonale e T tridiagonale simmetrica tali che $A = QTQ^T$;
3. Chiama una funzione `itera.m` da te creata, che approssima l'autocoppia desiderata (usa maxit=1000 e tol=10^{-6} per i criteri d'arresto) sfruttando in modo opportuno la decomposizione del punto precedente.
   Ad ogni iterazione, la funzione mostra su display in modo descrittivo il numero di iterazioni, l'autovalore approssimato corrente e la norma del residuo associato. Al termine, la funzione riporta su una stessa figura, con etichette, legenda e titolo, la storia della norma del residuo e quella dell'errore relativo all'autovalore, considerando come esatto l'ultimo autovalore approssimato calcolato.

Test di Laboratorio di Calcolo Numerico, 4 aprile 2022
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. valeriasimoncini310122) contenente tutti i files creati/usati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Considera la funzione $f(x) = \int_0^x t^2 \cos(t) \sin(t) \mathrm{dt}$, per $x \in [0, \frac{\pi}{2}]$.

Problema: determinare una approssimazione di $x_\star$ tale che $f(x_\star) = 0$ nell'intervallo considerato.

Nota: Per le valutazioni di f, approssima l'integrale mediante una formula di quadratura composita di grado 2, con almeno 10 sottointervalli.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Fa il grafico di f (Figura 1) nell'intervallo considerato.
2. Chiama i metodi necessari (scritti da te e riportati in cartella) per l'approssimazione del valore $x_\star$ mediante i) il metodo di Newton, ii) il metodo di bisezione. Usa tolleranze per i criteri di arresto non superiori a 10^{-6}.
   Riporta su uno stesso grafico la storia della convergenza dei due metodi, includendo etichette, legenda e titolo.

Test di Laboratorio di Calcolo Numerico, 31 maggio 2022
Laurea Triennale in Matematica.
Turno 2 - Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. valeriasimoncini310522) contenente tutti i files creati/usati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Considera i seguenti dati

x	0.81	1.20	1.47	1.72	1.76	2.02	2.28
y	-0.55	-0.34	-0.09	0.14	0.18	0.40	0.54

Problema: determina una approssimazione della soluzione $f(x) = 0$, $x \in [\frac{1}{4}\pi, \frac{3}{4}\pi]$, per la cui funzione f sono noti i valori (x_i, y_i) ($y_i = f(x_i)$, in tabella)

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Fa il grafico delle coppie di punti (x_i, y_i). (Figura 1)
2. Approssima la curva passante per le coppie (x_i, y_i) mediante polinomio p di interpolazione su tutti i nodi, e riportane il grafico, includendo etichette, legenda e titolo in Figura 1, in modo che siano visibili ancora i valori del punto precedente.
3. Approssima il valore dello zero x^* mediante il metodo delle secanti applicato a p, usando come valori iniziali due opportuni valori $\{x_i\}$ tra quelli dati. Visualizza in modo descrittivo la storia della convergenza, riportando i valori come nella seguente tabella:

iter	iterata	residuo	diff. iterate
⋮	⋮	⋮	⋮

Per i criteri d'arresto, proponi dai valori dei parametri che permettano una buona approssimazione di x^*.
Riporta su un grafico (Figura 2) la storia della convergenza, includendo etichette, legenda e titolo.

Test di Laboratorio di Calcolo Numerico, 31 maggio 2022
Laurea Triennale in Matematica.
Turno 1 - Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. valeriasimoncini310522) contenente tutti i files creati/usati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Considera i seguenti dati

x	0.81	0.99	1.62	1.64	1.71	1.71	2.15
y	-0.55	-0.48	0.05	0.07	0.13	0.14	0.48

Problema: determina una approssimazione dell'integrale $\int_{\frac{1}{4}\pi}^{\frac{3}{4}\pi} f(x)dx$ per la cui funzione integranda sono noti i valori (x_i, y_i) ($y_i = f(x_i)$, in tabella)

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Fa il grafico delle coppie di punti (x_i, y_i). (Figura 1)
2. Approssima la curva passante per le coppie (x_i, y_i) mediante polinomio p di interpolazione su tutti i nodi, e riportane il grafico, includendo etichette, legenda e titolo in Figura 1, in modo che siano visibili ancora i valori del punto precedente.
3. Approssima il valore dell'integrale $\int_{\frac{1}{4}\pi}^{\frac{3}{4}\pi} f(x)dx \approx \int_{\frac{1}{4}\pi}^{\frac{3}{4}\pi} p(x)dx$ mediante una formula composita di grado due, usando i valori $\{x_i\}$ come nodi della suddivisione dell'intervallo $[\frac{1}{4}\pi, \frac{3}{4}\pi]$. Visualizza in modo descrittivo il valore ottenuto.

Test di Laboratorio di Calcolo Numerico, 31 maggio 2022
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini310522, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Risoluzione di due problemi ai minimi quadrati, a cascata.

Crea uno script chiamato `nomecognome.m` che esegue quanto segue:

1. Genera i dati $A \in \mathbb{R}^{n\times m}$, $b \in \mathbb{R}^n$, con $n = 10, m = 6$, con A avente come elementi numeri reali presi da una distribuzione casuale normale, e b vettore di valori costanti e norma unitaria.
2. Chiama una funzione `risolviLS.m` che risolve il problema ai minimi quadrati
$$\min_x \|b - Ax\|$$
riportando in output tutte le informazioni utili per il punto successivo.
3. Data $G \in \mathbb{R}^{n\times k}$, $k = 2$, con G avente come elementi numeri reali presi da una distribuzione casuale uniforme, chiama una funzione `aggiornaLS.m` che risolve il problema
$$\min_z \|b - [G, A]z\|$$
sfruttando al meglio quanto costruito nel punto precedente.
4. Data $G \in \mathbb{R}^{n\times k}$, $k = 2$, con $G = [g_0, a_1]$, g_0 con elementi da distribuzione normale, e a_1 uguale alla prima colonna di A, chiama la funzione `aggiornaLS1.m` che risolve in modo opportuno il problema
$$\min_z \|b - [G, A]z\|$$
sfruttando al meglio quanto costruito nei punti precedenti.

Test di Laboratorio di Calcolo Numerico, 24 giugno 2022
Laurea Triennale in Matematica.
Turno 2 - Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. valeriasimoncini310522) contenente tutti i files creati/usati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Approssima la funzione

$$F(x) = x - \int_{-1}^{x} t(\sin t)^2 e^{-t} dt$$

mediante polinomi compositi, e stima l'area sotto al grafico.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Riporta il grafico di F in Figura 1.
2. Determina il polinomio composito di grado $n = 3$ che appossima F nell'intervallo $[-2, 1]$, con $m = 10$ sottointervalli, e ne riporta in grafico in Figura 1, includendo etichette, legenda e titolo.
3. Approssima l'area della regione sotto al grafico della funzione F e compresa tra $\min F$ e $\max F$ nell'intervallo $[-2, 1]$. Riporta il risultato su display in modo descrittivo.

Test di Laboratorio di Calcolo Numerico, 24 giugno 2022
Laurea Triennale in Matematica.
Turno 1 - Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. valeriasimoncini310522) contenente tutti i files creati/usati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Approssima lo zero ($x_\star \approx -4.3251e-01$) dell'equazione non lineare

$$F(x) = 0 \qquad F(x) = x - \int_{-1}^{x} t(\sin t)^2 e^{-t} dt,$$

mediante l'iterazione

$$x_{k+1} = \Phi(x_k), \qquad \Phi(x) = \int_{-1}^{x} t(\sin t)^2 e^{-t} dt.$$

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Fa il grafico della funzione F per $x \in [-3, 3]$ (Figura 1), includendo etichette, legenda e titolo.
2. Chiama la funzione `puntofisso.m` che approssima lo zero $x_\star$ con un residuo inferiore a 10^{-8} usando l'iterazione sopra. Riporta su un grafico (Figura 2), la storia della convergenza (residuo e differenza tra iterate), includendo etichette, legenda e titolo.
3. Stima la costante asintotica di convergenza, riportando su schermo il risultato, in modo descrittivo.

Test di Laboratorio di Calcolo Numerico, 24 giugno 2022
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini240622, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Risoluzione approssimata di un sistema lineare $Ax = b$.

Crea uno script chiamato `nomecognome.m` che esegue quanto segue:

1. Genera i dati $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, con $n = 50$

   ```
   rng(1); A=randn(n,n)+10*eye(n); x_exact=ones(n,1); b=A*x_exact;
   ```

2. Chiama una funzione `itera.m` che approssima la soluzione mediante lo splitting $A = P - N$, dove P è la matrice con elementi $P_{ii} = A_{ii}, P_{1,i} = A_{1,i}, P_{i,1} = A_{i,1}$ e gli altri elementi zero.
   La funzione `itera.m` risolve i sistemi con P in modo da sfruttare la struttura "a freccia".
   Durante l'iterazione, la funzione mostra su display i seguenti dati come in una tabella,

   iter norma errore relativo norma residuo relativo

   ed al termine riporta su uno stesso grafico la storia della norma del residuo e dell'errore (relativi), includendo etichette, legenda e titolo. (usa opportuni parametri per i criteri d'arresto)

3. Chiama la funzione `stima.m` che approssima il raggio spettrale della matrice di iterazione $B = P^{-1}N$ (≈ 0.82), **senza creare B esplicitamente**.

Test di Laboratorio di Calcolo Numerico, 18 luglio 2022
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini180722, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Risoluzione approssimata del problema ai minimi quadrati $\min_x \|b - Ax\|$.

Crea uno script chiamato `nomecognome.m` che esegue quanto segue:

1. Genera i dati $A \in \mathbb{R}^{n\times m}$, $b \in \mathbb{R}^n$,

   `rng(1); A=randn(n,m); b=rand(n,1);`

   e costruisce l'equazione normale associata.

   Al variare di $n \in \{50, 100, 150, ..., 400\}$ e $m = n/2$:

2. Chiama una funzione `iteraJ.m` che approssima la soluzione dell'equazione normale mediante il metodo di Jacobi (tol=1e-6, maxit=1000).

3. Chiama una funzione `iteraGS.m` che approssima la soluzione dell'equazione normale mediante il metodo di Gauss-Seidel (tol=1e-6, maxit=1000).

4. Per entrambi i metodi, lo script prende nota del tempo impiegato. Al termine lo script riporta in Figura 1 il grafico dei tempi per i due metodi al variare di n, ed in Figura 2 il numero di iterazioni richieste dai due metodi al variare di n. I grafici includono etichette, legenda e titolo.

   Per ognuno dei due casi, lo script riporta anche su display in modo descrittivo, **la norma del residuo del problema ai minimi quadrati**, insieme al numero di iterazioni del metodo iterativo, come segue:

n	norma LS+J	iter J	norma LS+GS	iter GS
:	:	:	:	

Test di Laboratorio di Calcolo Numerico, 18 luglio 2022
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. valeriasimoncini180722) contenente tutti i files creati/usati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Determina una successione di approssimazioni dell'area $\mathcal{A}(f)$ della regione compresa tra il grafico di una funzione $f \geq 0$ e l'asse delle ascisse, con metodi di ordine 1 e 2.

È data la funzione $f(x) = x\sin(x)\cos(x)^2$, $x \in [0, \pi]$. Approssima l'area

$$\mathcal{A}(f) \approx \mathcal{A}_{i,m}(f), \qquad i = 1, 2$$

dove $\mathcal{A}_{i,m}(f)$ deriva da una formula composita con m sottointervalli, e di ordine i.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Fa il grafico della funzione f per $x \in [0, \pi]$ (Figura 1), includendo etichette e titolo.
2. Per $m = 2, 4, 6, ...$, chiama le funzioni `I1.m` e `I2.m` che approssimano il valore di $\mathcal{A}(f)$ al crescere di m. Sapendo che $\mathcal{A}(f) \approx 1.047197551196598$, usa come criterio d'arresto l'errore relativo (tol=1-8) per ognuno dei due metodi
3. Riporta su un grafico (Figura 2), la storia della convergenza (errore relativo) delle due approssimazioni al crescere di m, includendo etichette, legenda e titolo.

Test di Laboratorio di Calcolo Numerico, 1 settembre 2022
Laurea Triennale in Matematica
Test Modulo I

Crea una cartella chiamata col tuo nome e data (es. ValeriaSimoncini010922, contenente tutti i files che saranno utilizzati. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti per far girare il test) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "NOME COGNOME - Modulo I".

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Risoluzione mediante un metodo diretto di un sistema lineare strutturato

Crea uno script chiamato `nomecognome.m` che esegue quanto segue:

1. Per n fissato, crea i dati

   ```
   rng(1); A=tril(randn(n,n)); A=flip(A,2); D=diag(rand(n,1));
   b=rand(n,1); f=rand(n,1);
   ```

2. Al variare di $n \in \{10, 20, 30, 40, 50\}$, risolve il sistema

   $$Mq = g \quad \Leftrightarrow \quad \begin{bmatrix} 0 & A \\ A & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ f \end{bmatrix}$$

   con i dati precedentemente generati. A tal fine, lo script chiama la funzione

   `skewtriang.m`

   (da creare) che risolve il sistema lineare $Az = q$ con $A \in \mathbb{R}^{n \times n}$ avente la forma (*qui riportata per* $n = 5$)

   $$A = \begin{bmatrix} & & & & \times \\ & & & \times & \times \\ & & \times & \times & \times \\ & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}$$

3. Lo script prende nota del tempo impiegato. Al termine lo script riporta in Figura 1 il grafico dei tempi di calcolo al variare di n. Il grafico include etichette, legenda e titolo.

   Lo script riporta anche su display in modo descrittivo, **la norma del residuo del problema**, insieme al numero di condizionamento della matrice M, come segue:

n	norma residuo	cond(M)
:	:	:

Nota: lo script non usa funzioni avanzate predefinite di matlab, eccetto `cond`, che può essere usata.

Test di Laboratorio di Calcolo Numerico, 1 settembre 2022
Laurea Triennale in Matematica.
Test Modulo II

Crea una cartella chiamata col tuo nome e data (es. valeriasimoncini180722) contenente tutti i files creati/usati durante il test. Al termine spedisci il contenuto della cartella (cioè tutte le funzioni e script richiesti) a

`mat.calcolonumerico@unibo.it`

con subject del messaggio "nome cognome Modulo II"

ATTENZIONE: la cartella deve contenere tutte le funzioni utili a far funzionare lo script principale. Se lo script principale non funziona per mancanza di funzioni, la prova non verrà corretta.

Problema: Determina una successione di approssimazioni di una funzione f mediante polinomi interpolatori p_n con $n+1$ nodi equispaziati.

È data la funzione $f(x) = \sin(x)\cos(x)e^{-\frac{x}{4}}$, $x \in [0, 4\pi]$.

Scrivi uno script Matlab (chiamalo con il tuo `nomecognome.m`) che esegua quanto segue:

1. Fa il grafico della funzione f per $x \in [0, 4\pi]$ (Figura 1), includendo etichette e titolo.
2. Approssima la funzione mediante polinomio di Lagrange di grado $n \in \{4, 6, 8, 10, 12\}$.
3. Riporta sullo stesso grafico (Figura 1), i grafici dei polinomi p_n al variare di n.
4. Per ogni n, approssima il valore x^* nel quale il polinomio p_n assume il valore 0.25. Calcola il corrispondente valore x_f nel quale la funzione f assume il valore 0.25. Riporta quindi in una tabella su display le seguenti informazioni:

n	$\|f - p_n\|_\infty$	x^*	$\|x^* - x_f\|$
:	:	:	: