PRIMO ESONERO DI CALCOLO NUMERICO 21/22

Esercizio 1. Verificare che

$$\frac{3f(x) - 4f(x-h) + f(x-2h)}{2h} = f'(x) + O(h^2).$$

In questo caso applico la formula di Taylor ossia:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x)...$$

Siccome per verificare la funzione data necessito $f(x - h)$ e $f(x - 2h)$ applico taylor su esse:

- $f(x - h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) + O(h^3)$
- $f(x - 2h) = f(x) - hf'(x) + \frac{4h^2}{2}f''(x) + O(h^3)$

Adesso sostituisco all'interno della funzione data le funzioni appena trovate ottenendo così:

$$\frac{3f(x)-4f(x)+4hf'(x)-2h^2f''(x)+f(x)-2hf'(x)+2h^2f(x)}{2h}$$

Semplificando ottengo che:

$$\frac{2hf'(x)}{2h} = f'(x) + O(h^2)$$

Esecizio 2. Calcolare la precisione di macchina di un'aritmetica finita con arrotondamento alla quinta cifra della mantissa, utilizzante base 4. Se $fl(x)$ è un numero di macchina normalizzato, corrispondente ad $x \in \mathbb{R}$, quali sono il massimo errore relativo ed assoluto di rappresentazione?

Conoscendo la formula per la precisione di macchina:

- per arrotondamento $u = \frac{1}{2}b^{1-m}$
- per troncamento $u = b^{1-m}$

con b = base, m = mantissa

Utilizziamo quindi quella per arrotondamento ottenendo quindi $u = \frac{1}{2}4^{-4}$

Invece il massimo errore relativo ed assoluto sono:

- errore relativo massimo $\frac{|x-fl(x)|}{x} \leq u$
- errore assoluto massimo $|x - fl(x)| \leq u \cdot x$

Esercizio 3. Dimostrare che il metodo di Newton converge quadraticamente ad una radice semplice.

Quindi consideriamo una radice di f(x), ossia quando f(x) = 0.

La formula del metodo di Newton è: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ con n = 0,1,2,...

Supponendo che $x_n \to x^*$, $n \to \infty$, sia una radice semplice di f(x), allora $f'(x^*) \neq 0$ e per il teorema di permanenza del segno $f'(x) \neq 0$ in un intorno di x* supponendo $f \in C^2$. Definito l'errore al passo n come $e_n = x_n - x^*$, applicando lo sviluppo di Taylor con il resto di Peano:

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{f''(\xi_n)}{2}(x - x_n)^2$$ con $x \leq \xi \leq x_n$

Otteniamo:

$0 = f(x^*) = f(x_n) + f'(x_n)(x^* - x_n) + \frac{f''(\xi_n)}{2}(x^* - x_n)^2$ e raccolgo f'(x):

$= f'(x_n)(\frac{f(x_n)}{f'(x_n)} + x^* - x_n) + \frac{f''(\xi_n)}{2}(x^* - x_n)^2$ ottenendo così:

$= f'(x_n)(x^* - x_{n+1}) + \frac{f''(\xi_n)}{2}(x^* - x_n)^2$ sostituiamo adesso e_n definito prima:

$= f'(x_n)e_{n+1} + \frac{f''(\xi_n)}{2}e_n^2$ per qualche $\xi_n \in I(x_n, x^*)$.

Dunque deduciamo che se $x_n \to x^*$ allora $n \to \infty$ e $\xi_n \to x_n$ da cui otteniamo che:

$$\lim_{n \to \infty} \frac{e_{n+1}}{e_n^2} \to \frac{f''(x^*)}{2f'(x^*)} \equiv c$$

Pertanto si ha convergenza quadratica con costante dell'errore dato da c.

Esercizio 4. Definire la fattorizzazione LU di una matrice nonsingolare A. Dimostrare che, se esiste, la fattorizzazione è unica.

Sia $A \in R^{n \times n}$, $det(A) \neq 0$, diciamo che A è fattorizzabile LU se $A = L \cdot U$ con:

1. **L** - triangolare inferiore a diagonale unitaria
2. **U** - triangolare superiore

Siccome $det(A) \neq 0$, allora $det(A) = det(LU) = det(L)\, det(U)$, siccome L è a diagonale unitaria det(L) = 1.

Se $A = L \cdot U$ la sua fattorizzazione è unica. Lo dimostriamo definendo $A = L \cdot U = L_1 \cdot U_1$ come due fattorizzazioni LU di A, allora $L = L_1$, $U = U_1$ quindi $LU = L_1U_1$.

Adesso facciamo due operazioni:

- moltiplichiamo a destra e a sinistra per U_1^{-1} ottenendo quindi:

  $LUU_1^{-1} = L_1U_1U_1^{-1}$ semplificando le due U a destra abbiamo $LUU_1^{-1} = L_1$

- moltiplichiamo a destra e a sinistra per L^{-1} ottenendo quindi:

  $LUU_1^{-1}L^{-1} = L_1L^{-1}$ semplificando le due L a sinistra abbiamo $UU_1^{-1} = L_1L^{-1}$

Osserviamo che U_1, U, U_1^{-1} sono triangolari superiori, mentre L_1, L^{-1} sono triangolari inferiore a diagonale unitaria, quindi, UU_1^{-1}è triangolare superiore e L_1L^{-1} è triangolare inferiore a diagonale unitaria. Quindi $UU_1^{-1} = L_1L^{-1} = D$, matrice diagonale, poiché la diagonale è unitaria allora $D = I$ la matrice identità. Segue che l'unica cosa non uguale a zero è la diagonale. Quindi otteniamo che $L = L_1$ $U = U_1$ ossia la fattorizzazione è unica.

Esercizio 5. Sotto quali condizioni esiste la fattorizzazione LU di una matrice? Dimostrare che una matrice simmetrica e definita positiva è fattorizzabile LU.

Sia $A \in R^{n\times n}$, $det(a) \neq 0$. Risolviamo con l'algoritmo di fattorizzazione LU di A.
$A = LU \Leftrightarrow det(A_k) \neq 0$ per ogni $k = 1,..., n$. Essendo $A_k \in R^{kxk}$ la sottomatrice principale di ordine k.
Quindi A è s.d.p. se:

- $A = A^T$ (simmetrica)
- per ogni $\underline{x} \in R^n$, $\underline{x} \neq \underline{0}$; $\underline{x}^T A\underline{x} > 0$ (definita positiva)

L'esistenza della fattorizzazione LU discende dal fatto che:

- (Lemma 1) se A è s.d.p., allora $det(A) \neq 0$. Quindi una matrice s.d.p. è non singolare.
- (Lemma 2) se A è s.d.p., allora, per ogni $k = 1,..., n$: A_k è s.d.p., cioè tutte le sottomatrici principali di una matrice s.d.p. sono s.d.p..

Dim (Lemma 1):
Se A fosse non singolare, allora $\exists\underline{x}\in R^n$, $\underline{x}\neq\underline{0}$, tale che $A\underline{x}=\underline{0}\Rightarrow\underline{x}^T A\underline{x} = 0$, assurdo CVD.
Dim (Lemma 2):

Sia A

A_k	B
C	D

con $A_k \in R^{k\times k}$ e $D \in R^{n-k\times n-k}$. Pertanto, poiché $A = A^T$, segue che $A_k = A_k^T$, $D = D^T$, $C = B^T$. Pertanto A_k è simmetrica.

Sia ora $\underline{y}\in R^k$, $\underline{y}\neq\underline{0}$ $\Rightarrow\underline{x}=$ $\begin{bmatrix} \underline{y} \\ \underline{0} \end{bmatrix} \in R^n$, sarà, a sua volta, non nullo.

Segue che:

$$0 < \underline{x}^T A \underline{x} = \begin{bmatrix} \underline{y} \\ \underline{0} \end{bmatrix}^T \begin{bmatrix} A_k & B \\ B^T & D \end{bmatrix} \begin{bmatrix} \underline{y} \\ \underline{0} \end{bmatrix}$$

$$= \begin{bmatrix} \underline{y}^T A_k & \underline{y}^T B \end{bmatrix} \begin{bmatrix} \underline{y} \\ \underline{0} \end{bmatrix} = \underline{y}^T A_k \underline{y} .$$

Esercizio 6. Definire il numero di condizione di una matrice e spiegarne il significato.

Sia $A \in R^{n\times n}$, $det(A) \neq 0$ risolvendo il sistema perturbato:

$(A + \Delta A)(x + \Delta x) = b + \Delta b$ otteniamo $\frac{||\Delta x||}{||x||} \leq ||A|| \cdot \left\|A^{-1}\right\|(\frac{||\Delta b||}{||b||} + \frac{||\Delta A||}{||A||})$

Dove si è considerata una qualunque norma su vettore e la corrispondente norma indotta su matrice. Pertanto, $K(A) = ||A|| \cdot \left\|A^{-1}\right\|$ definisce il numero di condizione di A.

Esercizio 7. Scrivere una *function* Matlab che, dato in ingresso un vettore, ne calcoli il corrispondente vettore di Householder, normalizzato in modo che la sua prima componente sia uguale a 1.

```
function v = house(x)
%
% v = house(x)    calcola il vettore
%                 di Householder v,
% relativo a x, normalizzato in
% modo che la sua prima
% componente sia 1.

v = x(:);
alfa = norm(v);
if alfa == 0, error('vettore x nullo'), end
if v(1) >= 0,
    v(1) = v(1) + alfa;
else
    v(1) = v(1) - alfa;
end
v = v / v(1);
return
end
```

Esercizio 8. Definire la fattorizzazione QR di una matrice $A \in \mathbb{R}^{m\times n}$, con $m > n = \text{rank}(A)$, e cosa si intende per soluzione del sistema lineare sovradeterminato

$$A\boldsymbol{x} = \boldsymbol{b}$$

nel senso dei minimi quadrati. Spiegare l'utilizzo della fattorizzazione QR per determinarla.

Vale il seguente teorema.

Sia $A \in R^{mxn}$, con m>n=rank(A), cioè con più equazioni che incognite dove la matrice dei coefficienti ha rango massimo.

Allora $\exists Q \in R^{m\times m}$, $Q^T Q = I\ (ortogonale)$,

e R=

$$\begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix} \in R^{m\times n}$$, con $\widehat{R} \in R^{nxn}$ triangolare superiore e non singolare, tale che: A=QR.

Il sistema lineare A<u>x</u>=<u>b</u>, con A come sopra, non ammette in genere soluzione, poiché $dim(range(A)) = rank(A) = n$, mentre <u>b</u>$\in R^m$, con m>n. Pertanto, si ricerca <u>x</u>$\in R^n$ t.c. il vettore residuo <u>r</u>=A<u>x</u>-<u>b</u>, sia tale che $||\underline{r}||_2^2$ = min!

La norma 2 è scelta per il fatto che essa è invariante per moltiplicazione di un dato vettore <u>v</u> per una matrice ortogonale. Infatti:

$||Q\underline{v}||_2^2 = (Q\underline{v})^T(Q\underline{v}) = \underline{v}^T Q^T Q\, \underline{v} = \underline{v}^T \underline{v} = I$, ovvero la matrice è ortogonale.

Si ottiene:

$$\begin{aligned} \|\mathrm{r}\|_2^2 &= \|A\mathrm{x} - \mathrm{b}\|_2^2 = \|QR\mathrm{x} - \mathrm{b}\|_2^2 \\ &= \|Q(R\mathrm{x} - Q^T\mathrm{b})\|_2^2 = \|Q(R\mathrm{x} - \mathrm{g})\|_2^2 \\ &= \|R\mathrm{x} - \mathrm{g}\|_2^2 \equiv (*). \end{aligned}$$

con $\mathrm{g} = \begin{pmatrix} \mathrm{g}_1 \\ \mathrm{g}_2 \end{pmatrix}, \quad \mathrm{g}_1 \in \mathbb{R}^n$

$$\begin{aligned} (*) &= \left\| \begin{pmatrix} \hat{R} \\ O \end{pmatrix} \mathrm{x} - \begin{pmatrix} \mathrm{g}_1 \\ \mathrm{g}_2 \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \hat{R}\mathrm{x} - \mathrm{g}_1 \\ -\mathrm{g}_2 \end{pmatrix} \right\|_2^2 \\ &= \|\hat{R}\mathrm{x} - \mathrm{g}_1\|_2^2 + \|\mathrm{g}_2\|_2^2 = \|\mathrm{g}_2\|_2^2, \end{aligned}$$

= min!

scegliendo <u>x</u> come soluzione del sistema lineare $\widehat{R}$<u>x</u>=<u>g1</u>, in modo tale che $||\widehat{R}\underline{x} - \underline{g1}||_2^2 = 0$.

Questo vettore è detto soluzione del sistema lineare nel senso dei minimi quadrati, avendosi $||\underline{r}||_2^2 = \sum_{i=1}^{m} r_i^2$.

Da quanto esposto, si deduce che <u>x</u> è unica e, inoltre, il fattore Q è richiesto solo per calcolare il vettore <u>g</u>=Q^T<u>b</u>.

PRIMO ESONERO DI CALCOLO NUMERICO 22/23

1. Come si definisce la precisione di macchina di un'aritmetica finita? Quanto vale la precisione di macchina della doppia precisione IEEE?

Se un aritmetica finita in base b utilizza m cifre per la mantissa di un numero di macchina normalizzato, allora la precisione di macchina n è definita come:

- per arrotondamento $u = \frac{1}{2}b^{1-m}$
- per troncamento $u = b^{1-m}$

Essa fornisce una maggiorazione uniforme per l'errore di rappresentazione, per i numeri di macchina normalizzati.
La doppia precisione IEEE utilizza:

- base b = 2
- mantissa m = 53 cifre
- per arrotondamento

Pertanto applicando la formula $u = \frac{1}{2}2^{1-53} = 2^{-53} \approx 10^{-16}$quindi 16 cifre decimali

N.B. se si parla invece di singola precisione IEEE si utilizza:

- base b = 2
- mantissa m = 24 cifre
- per arrotondamento

Pertanto applicando la formula $u = \frac{1}{2}2^{1-24} = 2^{-24} \approx 10^{-7}$quindi 7 cifre decimali

2. Cosa è il fenomeno della cancellazione numerica?

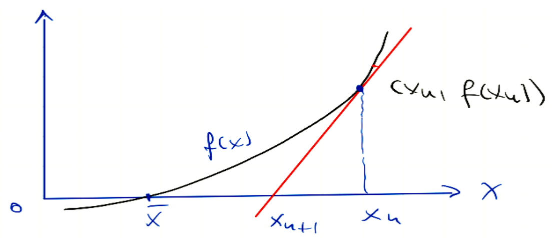
La cancellazione numerica è la perdita di cifre significative nella somma, in aritmetica finita, di numeri quasi opposti. Questo fenomeno è dovuto al mal condizionamento della somma algebrica quando due numeri da sommare sono di segno discorde. Infatti in questo caso il numero di condizione della somma x+y che è dato da:

$$k = \frac{|x|+|y|}{|x+y|}$$

non è limitato superiormente se $x \approx - y$.

3. Derivare il metodo di Newton per la ricerca della radice di una funzione, e dimostrare che esso converge quadraticamente a radici semplici.

Il metodo di Newton è un metodo iterativo per risolvere $f(x) = 0$, con $f: R \rightarrow R$, basata su una linearizzazione locale della funzione $f(x)$ nell'approssimazione corrente x_n:



Data la retta tangente il grafico di $f(x)$ nel punto $(x_n,\ f(x_n))$, r: $y - f(x_n) = f'(x_n)(x - x_n)$, x_{n+1} è ricavata come l'ascissa per cui $y = 0$:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \text{ con n = 0,1,2...}$$

Per la convergenza vedi esercizio 3 prova 21/22.

4. Derivare il metodo di accelerazione di Aitken.

Il metodo di accelerazione di Aitken serve a ripristinare la convergenza quadratica del metodo di Newton verso radici multiple, per cui la convergenza è solo lineare. In dettaglio detto $e_n = x_n - \overline{x}$ l' errore al passo n, si avrà asintoticamente: $e_n \approx c\, e_{n-1}$ e $e_{n+1} \approx c\, e_n$ con c costante ignota dell'errore. Dividendo membro a membro si ottiene: $\frac{e_{n+1}}{e_n} \approx \frac{e_n}{e_{n-1}}$. Ponendo l'uguaglianza e detta $\overline{x_n} \approx \overline{x}$ il valore che la soddisfa, si ottiene:

$$\frac{\overline{x_n} - x_{n+1}}{\overline{x_n} - x_n} = \frac{\overline{x_n} - x_n}{\overline{x_n} - x_{n-1}}, \text{ da cui } (\overline{x_n} - x_{n+1})(\overline{x_n} - x_{n-1}) = (\overline{x_n} - x_n)^2.$$

Si ottiene quindi, $x_n = \frac{x_n^2 - x_{n+1}x_{n-1}}{2x_n - x_{n+1} - x_{n-1}}$

Ripartendo da questa approssimazione con due passi di Newton si ottiene una nuova successione $\{\overline{x_n}\}$, che converge quadraticamente a $\overline{x}$. Il metodo di accelerazione di Aitken è da preferirsi a Newton modificato, quando la molteplicità della radice non è nota.

5. Scrivere in modo "professionale" una function Matlab che risolva efficientemente un sistema triangolare superiore.

```
function x= sollu(U,b)
%
% function x= sollu(U,b)
%
% Risoluzione del sistema triangolare
% superiore Ux=b.
%
% Input:
%   U - matrice dei coefficienti (di
%       cui si utilizza la sola porzione
%       triangolare superiore);
%   b - vettore dei termini noti.
%
% Output:
%   x - vettore soluzione.
%
%                          Rel. 2023-01-26.
%
[m,n] = size(U);
if m~=n || n~= length(b), error, end
x = b(:);
for i= n:-1:1
    if U(i,i)==0, error, end
    x(i) = x(i)/U(i,i);
    x(1:i-1) = x(1:i-1)-x(i)*U(1:i-1,i);
end
return
```

6. Sotto quali condizioni esiste la fattorizzazione LU di una matrice non-singolare? Definire cosa si intende per matrice a *diagonale dominante*. Dimostrare che una matrice diagonale dominante è fattorizzabile LU.

Sia data una matrice non singolare $A \in R^{n\times n}$. La fattorizzazione LU di A esiste se e solo se detta, A_k la sottomatrice principale di ordine k di A, risulta:

$det(A_k) \neq 0$ con k =1,...,n. La matrice $A(a_{ij}) \in R^{n\times n}$ si dice diagonale dominante:

- per righe, se per ogni i=1,..n: $|a_{ii}| > \sum_{j=1,\, j\neq i}^{n} |a_{ij}|$
- per colonne, se per ogni j=1,..n: $|a_{jj}| > \sum_{i=1,\, i\neq j}^{n} |a_{ij}|$

Una matrice diagonale dominante è fattorizzabile LU, in quanto, A diagonale dominante per righe o/e colonne se e solo se per ogni k=1,..,n: A_k è diagonale dominante per righe e/o colonne. Basta quindi dimostrare che se A è diagonale dominante allora A è non singolare. Considerato che vale la seguente proprietà: A è diagonale dominante per righe/colonne se A^T è diagonale dominante per colonne/righe, è sufficiente dimostrare che A è diagonale dominante per righe se e solo se A è non singolare: infatti ragionando per assurdo e supponendo che $det(A) = 0$ raggiungiamo un assurdo logico.

7. Definire cosa è il numero di condizionamento di una matrice. Spiegarne il significato.

Vedi esercizio 6 prova 21/22.

8. Definire la soluzione nel senso dei minimi quadrati di un sistema lineare sovra-determinato a rango pieno. Dimostrarne l'esistenza ed unicità.

Vedi esercizio 8 prova 21/22. (guarda la soluzione di questo sul pdf, in quanto è spiegato meglio)

ESERCIZI MANCANTI DELLE ESERCITAZIONI

Qual'è il range di rappresentazione per gli interi, in un aritmetica finita in base 8 con 5 cifre?

Il range di rappresentazione è in generale:

$$\{-b^N, \dots, 1-b^N\}$$

in questo caso otteniamo il range di rappresentazione pari a: $\{-8^5, \dots, 1-8^5\}$.

Come si definisce l'ordine di convergenza di un metodo per la ricerca degli zeri di una funzione?

Sia $x_{n+1} = \Phi(x_n)$ con n = 0,1,..., denota un generico metodo iterativo per la ricerca di una radice $\bar{x}$ dell'equazione $f(x) = 0$. Si supponga che $x_n \to \bar{x}$, per $n \to \infty$, e si denota con $e_n = |x_n - \bar{x}|$ il corrispondente errore al passo n. Il metodo si dice convergere con ordine $p \geq 1$ alla radice, se p è il più grande valore reale per cui $\lim_{n\to\infty} \frac{e_{n+1}}{e_n^p} = c < \infty$.

La costante c prende il nome di costante asintotica dell'errore, questo significa che per n>>1, $e_{n+1} \approx c \cdot e_n^p$.

Qual è il numero massimo di iterazioni che richiederà il metodo di bisezione per determinare la radice di una funzione assegnata con tolleranza (assoluta) 10^{-3}, se l'intervallo di confidenza iniziale è $[33, 37]$?

Il metodo di bisezione dimezza ad ogni iterazione l'ampiezza dell'intervallo di confidenza. Pertanto l'approssimazione al passo n, avrà un accuratezza:

$$2^{-n}(b-a) \leq tol$$

Essendo b-a l'ampiezza dell'intervallo di confidenza iniziale. Nel nostro caso, b-a= 33-37=4, per cui si ottiene $2^{-n}4 \leq 10^{-3}$, semplificando abbiamo $2^{2-n} \leq 10^{-3}$. Si ottiene quindi il numero massimo di iterazioni. Osservando che $10^{-3} \approx 2^{-10}$, si ottiene quindi $2^{2-n} \leq 2^{-10}$ quindi $2 - n \leq -10$ quindi $n \geq 12$ cioè $n = 12$.

Calcolare il numero di condizionamento della radice nulla di

$$f(x) = 3e^x - 2\cos x - 1.$$

Il numero di condizione di una radice nulla è dato da $\frac{1}{f'(\bar{x})}$, se $\bar{x}$ è la radice.Nel nostro caso $f'(x) = 3e^x + 2\,sinx$ e, quindi, $f'(0) = 3$. Pertanto il numero di condizione della radice nulla di $f(x)$ vale $\frac{1}{3}$.

Definire la molteplicità di una radice. Calcolare la molteplicità della radice nulla di

$$f(x) = e^{x^2} - 1.$$

Perché il calcolo di una radice nulla è un problema malcondizionato?

La radice $\bar{x}$ di f(x)=0 ha molteplicità m se:

$$f(\bar{x}) = f'(\bar{x}) = \ldots\ldots = f^{m-1}(\bar{x}) = 0 \text{ e se } f^{m}(\bar{x}) \neq 0.$$

Se m=1 la radice si dice semplice, se m>1 la radice si dice multipla.
La determinazione di una radice multipla è un problema mal condizionato, perché il numero di condizione della radice, $\frac{1}{f'(\bar{x})}$, è infinito, essendo $f'(\bar{x})$=0.

Se $f(x) = e^{x^2} - 1$, allora f(0)=0. Inoltre, $f'(x) = 2xe^{x^2} \Rightarrow f'(0) = 0$.

Ancora, $f''(x) = 2e^{x^2} + 4x^2e^{x^2} \Rightarrow f''(0) = 2 \neq 0$. Pertanto la radice ha molteplicità 3.

Sapendo che, se A è sdp, A è fattorizzabile LU, dImostrare che A=LDL^T, con D matrice diagonale ad elementi positivi.

Sia A sdp allora

1) $A = LU = LD\hat{U}$, con:
   -D matrice diagonale contenente gli elementi diagonali di U;
   -$\hat{U}$ triangolare superiore a diagonale unitaria.
   [ricordiamo che L è triangolare inferiore a diagonale unitaria].

Poiché $A = A^T$ (ortogonale), dalla 1) ricaviamo che

2) $A = (LD\hat{U})^T = \hat{U}^T D^T L^T = \hat{U}^T DL^T$ in quanto D^T diagonale.

Poiché:

- $\hat{U}^T$ è triangolare inferiore a diagonale unitaria,
- DL^T è triangolare superiore,
- la fattorizzazione LU di una matrice (non singolare) è unica,

dalle 1) e 2) si deduce che L=$\hat{U}^T$ *e, pertanto*:

$$A = LDL^T.$$

Se $D = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}$, abbiamo che, per un generico i∈{1,....,n}:
$d_i = e_i^T De_i$, con $\underline{e_i} \in R^n$, l'i-esimo versore della base canonica. Inoltre, essendo L^Tnon singolare, il sistema $L^T\underline{x}=\underline{e_i}$ è risolvibile e $\underline{x} \neq \underline{0}$. Pertanto, segue che:
$d_i = e_i^T De_i = (L^T\underline{x})^T DL^T\underline{x} = \underline{x}^T(LDL^T)\underline{x} = \underline{x}^T A\underline{x} > 0.$

Definire cosa si intende per norma indotta su matrice. Dare qualche esempio di qualche norma.

Sia $||\cdot||$ una assegnata norma sul vettore. Per esempio, se $x \in R^n$:

- $||x||_\infty = max_i |x_i|$
- $||x||_1 = \sum_{i=1}^{n} |x_i|$
- $||x||_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} = x^T x$

Si definisce, se $A \in R^{m\times n}$, norma indotta della corrispondente norma su vettore, $||A|| = max_{||x||=1} ||Ax||$. Ad esempio, delle precedenti norme su vettore, se $A = (a_{ij})$, allora:

- $||A||_\infty = max_{i=1,..,m} \sum_{j=1}^{n} |a_{ij}|$
- $||A||_1 = max_{j=1,..,n} \sum_{i=1}^{n} |a_{ij}|$
- $||A||_2 = \sqrt{\varrho(A^T A)} \equiv \sqrt{\varrho(AA^T)}$

Essendo ϱ il raggio spettrale della matrice in argomento, ovvero il massimo dei moduli dei suoi autovalori.

Definire il metodo di Newton per sistemi non lineari e dettagliarne l'implementazione.

Sia $f: R^n \to R^n$, con $f_1,..., f_n: R^n \to R$ le sue funzioni componenti.

Definiamo

$$F(\underline{x}) = \begin{bmatrix} \frac{\partial f_1(\underline{x})}{\partial x_1}, \dots, \frac{\partial f_1(\underline{x})}{\partial x_n} \\ \vdots \\ \frac{\partial f_n(x)}{\partial x_1}, \dots, \frac{\partial f_n}{\partial x_n}(\underline{x}) \end{bmatrix} \in \mathbb{R}^{n\times n}$$

la matrice Jacobiana di $f(x)$. Il metodo di Newton per risolvere $f(x) = 0$ è definito dall'iterazione $x_{n+1} = x_n - F(x_n)^{-1} f(x_n)$, n = 0,1...

Nella pratica si risolve il sistema lineare $F(x_n)\delta x_n = - f(x_n)$ e quindi $x_{n+1} = x_n + \delta x_n$, n = 0,1,...

Costruire la matrice di Householder relativa al vettore $z = \begin{bmatrix} 1 \\ -2 \\ -2 \end{bmatrix}$. Quanto vale H_z?

Ricerchiamo H, matrice ortogonale tale che:

$H_z = \begin{bmatrix} \alpha \\ 0 \\ 0 \end{bmatrix} \equiv \alpha e_1$, $e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$. Noto subito che:

- $\alpha^2 = ||z||_2^2 = 9 \Rightarrow \alpha = \pm 3$
- $H = I - \frac{2}{V^T V} V V^T$

con $v = z - \alpha e_1$ il corrispondente vettore di Householder v = $\begin{bmatrix} 1-\alpha \\ -2 \\ -2 \end{bmatrix}$.

Per ottenere la somma ben condizionata, il segno di α deve essere scelto opposto a quello della prima componente di z, pertanto $\alpha = -3$. Pertanto otteniamo:

v = $\begin{bmatrix} 4 \\ -2 \\ -2 \end{bmatrix}$ e $H_z = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix}$

Proprietà delle norme

Su vettore

Una funzione definita su uno spazio vettoriale V

$$||\cdot||: V \to R$$

definisce una norma in V, se soddisfa le seguenti tre proprietà caratteristiche:
N1: per ogni $x \in V$: $||x|| \geq 0$, inoltre $||x|| = 0 \Leftrightarrow x = 0$;
N2: per ogni scalare $\boldsymbol{\alpha}$ e per ogni $x \in V$: $||\boldsymbol{\alpha} x|| = |\boldsymbol{\alpha}| \cdot ||x||$;
N3: per ogni $x, y \in V$: $||x + y|| \leq ||x|| + ||y||$.

Formula della norma p di un vettore $x = (x1,...., xn)^T$:

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}, \ p \geq 1$$

Formula della norma ∞:

$$||x||_\infty \equiv max_{i=1,...,n}|x_i| = \lim_{p \to \infty} ||x||_p$$

Su matrici

Se $A \in R^{m \times n}$ e $||\cdot||$ è una norma su vettore, definiamo

$$||A|| = max_{||x||=1}||Ax||$$

la norma su matrice A. Notare che $x \in R^n$, $Ax \in R^m$. Vi sono le seguenti proprietà:

N4: $||Ax|| = ||A|| \cdot ||x||$, ovvero la norma indotta su matrice è compatibile con la corrispondente norma su vettore;

N5: $||AB|| \leq ||A|| \cdot ||B||$;

N6: se B è una qualunque sottomatrice di A, allora $||B|| \leq ||A||$;

N7: se A è $n \times n$, allora $\boldsymbol{\rho}(A) \leq ||A||$;

N8: $||I|| = 1$.

Formule:

$$\begin{aligned}
\|A\|_1 &= \max_{j=1,\dots,n} \sum_{i=1}^{m} |a_{ij}|, \\
\|A\|_2 &= \sqrt{\rho(A^* A)} \equiv \sqrt{\rho(A\,A^*)}, \\
\|A\|_\infty &= \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|.
\end{aligned}$$

Scrivere una function Matlab che risolva efficientemente un sistema triangolare superiore, in modo vettoriale, e per colonne.

```
function x= triu( U, b)
%
%  x= triu( U, b)
%
%  Risolve un sistema triangolare superiore.
%
%  Input:
%    U - matrice dei coefficienti
%    b - vettore dei termini noti
%
%  Output:
%    x - soluzione del sistema.
%
[m,n] = size(U);
if m ~= n || n ~= length(b)
    error('sistema non compatibile');
end
x= b(:);
for i= n:-1:1
    if U(i,i)==0, error('matrice singolare'), end
    x(i)= x(i)/U(i,i);
    x(1:i-1) = x(1:i-1) - x(i)* U(1:i-1,i);
end
return
```

Scrivere una **`function`** Matlab che implementi efficientemente il metodo di Newton.

```
function  x = newton ( f , f1 , xφ, tol, itmax)
%
%  x = newton ( f , f1 , xφ, tol, itmax)
%
%  Metodo di Newton per la ricerca della radice
%  di una funzione.
%
%  Input:
%    f   - function che implementa f(x);
%    f1  - function che implementa f'(x);
%    xφ  - punto iniziale;
%    tol - tolleranza richiesta (default 1e-12);
%   itmax - numero massimo di iterazioni
%             (default 1000);
%
%  Output:
%     x  - soluzione approssimata.
%
if nargin < 5
    maxit = 1000;
else
    if maxit < 1, error ('maxit errato'); end
end
if nargin < 4
    tol = 1e-12;
else
    if tol < 0, error(' tolleranza negativa'); end
    tol = max (tol, 10*eps);
end
if nargin < 3
    error(' numero argomenti di ingresso errato');
end
x = xo;
for i = 1 : maxit
    xold = x;
    fx = feval ( f, x);
    f1x = feval ( f1, x);
    if f1x == 0, error('il metodo non converge');end
    x = x - fx / f1x;
    err = abs(x-xold);
    if err <= tol , break , end
end
if err > tol, warning('tolleranza richiesta
        non soddisfatta');
end
return
```

Scrivere una function Matlab che implementi efficientemente il metodo delle secanti.

```
function   xstar = secanti( fun, x0, x1, tol, itmax)
%
%   xstar = secanti( f , x0, x1, tol , itmax)
%
%   Calcola una approssimazione della
%   radice di f(x) con tolleranza tol.
%
%   Input:
%
%     f     - identificatore della function che
%             implementa f(x);
%   x0,x1   - punti iniziali;
%    tol    - accuratezza richiesta (default = 10^-6);
%   itmax   - numero massimo di iterazioni
%               (default = 1000).
%
%   Output:
%
%     xstar - approssimazione della soluzione.
%
if nargin < 3
    error(' numero di argomenti in ingresso errato')
else if nargin ==3
    tol = 1e-6 ; itmax = 1000;
else if nargin == 4
    itmax = 1000 ;
end
if tol <= 0, error('tolleranza errata'); end
if itmax <= 0, error(' itmax errate'); end

f0 = feval ( f, x0) ;
f1 = feval ( f, x1) ;

for i = 1: itmax
```

```
    if fϕ == f1 && f1 ~= 0
        error('il metodo non converge');
    end
    xstar = (xϕ*f1 - x1*fϕ)/(f1-fϕ);
    delta = abs(xstar-x1);
    if delta <= tol(1+abs(xstar))
        break
    elseif it < itmax
        xϕ = x1; xstar = x1;
        fϕ = f1; f1 = feval(f, x1);
    end
end
if delta > tol(1+abs(xstar)),
    warning('accuratezza richiesta non
             raggiunta');
end
return
```

Codici Matlab

1)

Algoritmo di **risoluzione per sistemi triangolari** (caso inferiore):

Alg1) Accesso per riga

Alg2) Accesso per colonna

a(i,j) , x(i) , b(j)

Algoritmo 1:

```
for i= 1:n
    x(i)= b(i)
    for j= 1:i-1
        x(i) = x(i) - a(i,j) * x(j)
    end
    x(i)= x(i) / a(i,i)
end
```

Algoritmo 2: $\underline{x} \leftarrow \underline{b}$

```
for j= 1:n
    x(j) = x(j) / a(j,j)
    for i= j+1:n
        x(i) = x(i) - a(i,j) * x(j)
    end
end
```

Quindi, la complessità per il numero di operazioni è n^2, quindi quadratica.

Riguardo all'operazione di memoria, questa è $\frac{n(n+1)}{2} + n = O(\frac{n^2}{2})$.

2)

Algoritmo di **fattorizzazione LU di una matrice**:

```
for i = 1:n-1
    if A(i,i)==0
       error('matrice non fattorizzabile LU')
    end
    A(i+1:n,i) = A(i+1:n,i)/A(i,i);
    A(i+1:n,i+1:n) = A(i+1:n,i+1:n) -A(i+1:n,i)*A(i,i+1:n);
end
```

Risolvere per i fattori L ed U ha un costo di circa $2n^2$ flops.

3)

Algoritmo di **fattorizzazione** LDL^T:

```
if A(1,1)<=0, error('la matrice non e'' sdp'), end
A(2:n,1) = A(2:n,1)/A(1,1);
for j = 2:n
    v = ( A(j,1:j-1).' ) .* diag(A(1:j-1,1:j-1));
    A(j,j) -= A(j,j) - A(j,1:j-1)*v;
    if A(j,j)<=0, error('la matrice non e'' sdp'), end
    A(j+1:n,j) = ( A(j+1:n,j) -A(j+1:n,1:j-1)*v )/A(j,j);
end
```

4)

Algoritmo di **fattorizzazione LU con pivoting parziale:**

```
function   x = palu ( A, b )
%
%     x = palu (A,b)
%
%     Risolve il sistema lineare A x=b
%     con A matrice nonsingolare.
%
%     Input:
%       A  -  matrice dei coefficienti;
%       b  -  vettore dei termini noti;
%
%     Output:
%       x  -  vettore soluzione.
%
%                                          Rel. 2023-12-05.
[m,n] = size (A);
if m~=n, error('matrice non quadrata'), end
k = length (b);
if k~=n, error('vettore dei termini noti errato'),end

% fattorizzazione della matrice.
p = 1:n;
for i=1:n
≈ 1/2 n^2 confronti  [mi, ki] = max ( abs( A(i:n,i) ) );  % calcolo del pivot
   if mi==0, error(' matrice singolare'), end
   ki= ki +i-1 ;
   if ki>i
      A([i,ki],:) = A([ki,i],:);
      p([i,ki]) = p([ki,i]);
   end

   A( i+1:n, i)= A(i+1:n,i)/ A(i,i);

   A(i+1:n,i+1:n)= A(i+1:n,i+1:n) - ...                 ≈ 2/3 n^3 flops

               A(i+1:n,i) * A(i,i+1:n);
end

x= b(p);   %  P·b
x= x(:);
 for i=2:n      % risolvo per L
     x(i:n) = x(i:n) - A(i:n, i-1) * x(i-1);            n^2 flops
 end

 for i= n:-1:1              % risolvo per U
     x(i) = x(i) / A(i,i);
     x(1:i-1) = x(1:i-1) - A(1:i-1, i) * x(i);          n^2 + n flops
 end

 return
```

5)
Algoritmo di **fattorizzazione QR di Householder**:

```
for i=1:n
  alfa = norm( A(i:m,i) );
  if alfa==0, error('la matrice A non ha rango massimo'), end
  if A(i,i)>=0, alfa = -alfa; end
  v1 = A(i,i) -alfa;
  A(i,i) = alfa; A(i+1:m,i) = A(i+1:m,i)/v1;
  beta   = -v1/alfa;
  A(i:m,i+1:n) = A(i:m,i+1:n) -(beta*[1; A(i+1:m,i)])*...
                          ([1 A(i+1:m,i)']*A(i:m,i+1:n));
end
```

Costo: $2(m-i)(n-i)$flops.

6)

Tratti da esami

1) Scrivere una function Matlab che, data in ingresso una matrice triangolare superiore U ed un vettore b, **calcoli efficientemente la soluzione del sistema lineare Ux = b**:

```
function x = Usolve(U,b)
% commenti appropriati
[m,n] = size(U); if m~=n || n~=length(b), error('dati errati'), end
x=b(:);
for i = n:-1:1
   % x(i) = x(i)   (barrato)
   if U(i,i)==0, error('matrice singolare'); end
   x(i) = x(i)/U(i,i);  x(1:i-1) = x(1:i-1) - x(i) * U(1:i-1, i);
end
return
```

2) **Metodo di bisezione**:

```
function [x,it] = bise (fun, a, b, tol)
%
% [x,it] = bise (fun, a, b, tol)
%
% Metodo di bisezione per la ricerca
% dello zero di una funzione.
%
% Input:
% fun : stringa con la function che
%       implementa la funzione;
% a,b : estremi intervallo di confidenza
%       iniziale;
% tol : accuratezza richiesta.
% Output:
% x  : soluzione ottenuta;
% it : numero iterazioni effettuate
%      (opzionale).
%
if b <= a, error ('intervallo iniziale non
                   appropriato'); end
nit=0;
fa= feval (fun, a); if fa ==0, x=a, return, end
fb= feval (fun, b); if fb ==0, x=b, return,
                                         end
if fa*fb > 0
   error ('metodo non applicabile');
end
if tol <=0, error ('tolleranza non corretta');
maxit = -log( tol /(b-a)) ;                end

for it= 1: maxit
      x= (a+b)/2;
      fx = feval (fun, x);
      f1= abs( fb-fa) / (b-a);
      if abs(fx) <= f1*tol, break
      elseif fa*fx <0
            b=x; fb= fx;
      else
            a= x; fa=fx;
      end
end

return
end
```

3) Dato in ingresso un vettore, ne calcoli il corrispondente vettore di **Householder, normalizzato in modo che la sua prima componente sia uguale a 1**:

```
function v = house(x)
%
% v = house(x)    calcola il vettore
%                 di Householder v,
% relativo a x, normalizzato in
% modo che la sua prima
% componente sia 1.

v = x(:);
alfa = norm(v);
if alfa == 0, error('vettore x nullo'), end
if v(1) >= 0,
    v(1) = v(1) + alfa;
else
    v(1) = v(1) - alfa;
end
v = v / v(1);
return
end
```