cn - lab. scheda 2

Esercizio 1

1. a) Sia $x = [-3, 5, 8, 0, 1, 5, -2, 4]$

- imporre il 6° elemento uguale a 100
- imporre il 1°, 2°, 3° elemento uguali rispettivamente a 5, 6, 7
- togliere il 4° elemento
- togliere con un solo comando dal 4° al 7° elemento compresi
- aggiungere in testa 1, 2, 3
- aggiungere in coda 10, 11, 12.

b) Sia A la matrice identità di dimensione 4x4

- sostituire all'elemento (1,1) l'elemento (3,4)
- aggiungere una colonna di elementi uguali ad 1 in testa
- aggiungere una colonna di elementi uguali ad 1 in coda
- aggiungere una riga di elementi uguali ad 4 in testa
- aggiungere una riga di elementi uguali ad 4 in coda
- togliere la 3a riga
- togliere la 3a colonna

1.a

```
%creo il vettore x
>> x=[-3,5,8,0,1,5,-2,4];

%impongo il sesto elemento uguale a 100
>> x(6)=100

%estraggo i primi tre elementi del vettore
>> x([1,2,3])

%estraggo e modifico i primi tre elementi del vettore
>> x([1,2,3]) = [5,6,7]

%elimino il quarto elemento del vettore
>> x(4)=[]

%concateno il vettore 1,2,3 con il vettore x
>> x=[1,2,3,x]
```

```
%concateno il vettore x con il vettore 10,11,12
>> x=[x,10,11,12]
```

- **modifica di un elemento del vettore**:
  `x(index)`
  utilizzato per accedere o modificare un elemento specifico;
- **estrazione di elementi specifici**:
  `x[(index1, index2, ...)]`
  per estrarre più elementi contemporaneamente, mentre per fare un'assegnazione contemporanea di più indici:
  `x[(index1, index2, ...)]=[elem1, elem2, ...]`
- **eliminazione di un elemento**:
  `x(index)=[]`
  elimina un elemento nella posizione specificata, riducendo cosi la lunghezza del vettore.
- **concatenazione di vettori**:
  `x=[elem1, elem2, ..., x]`
  `x=[x, ...,elem1, ...]`

1.b

```
>> A=eye(4)

A =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> A(1,1)=A(3,4)

A =

     0     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> c = ones(4,1)

c =

     1
     1
     1
     1

>> A=[c A]

A =

     1     0     0     0     0
```

```
     1     0     1     0     0
     1     0     0     1     0
     1     0     0     0     1

>>
>> A=[A c]

A =

     1     0     0     0     0     1
     1     0     1     0     0     1
     1     0     0     1     0     1
     1     0     0     0     1     1

>> b = 4 * ones(1, size(A,2))

b =

     4     4     4     4     4     4

>> A=[b;A]

A =

     4     4     4     4     4     4
     1     0     0     0     0     1
     1     0     1     0     0     1
     1     0     0     1     0     1
     1     0     0     0     1     1

>> A=[A;b]

A =

     4     4     4     4     4     4
     1     0     0     0     0     1
     1     0     1     0     0     1
     1     0     0     1     0     1
     1     0     0     0     1     1
     4     4     4     4     4     4

>> A(3,:)=[]

A =

     4     4     4     4     4     4
     1     0     0     0     0     1
     1     0     0     1     0     1
     1     0     0     0     1     1
     4     4     4     4     4     4

>> A(:,3)=[]

A =
```

```
4   4   4   4   4
1   0   0   0   1
1   0   1   0   1
1   0   0   1   1
4   4   4   4   4
```

- **creazione matrice identità**:
  `A=eye(n)`
  crea una matrice identità della dimensione specificata $n \times n$.
- **modifica di un elemento specifico**:
  `A(1,1)=A(3,4)`
  si assegna all'elemento in posizione `(1,1)` il valore dell'elemento in posizione `(3,4)`.
- **creazione di un vettore di 1**:
  `c=ones(m,n)`
  genera una matrice di dimensione $m \times n$ con tutti gli elementi uguali ad 1.
- **concatenazione di un vettore colonna a sinistra**
  `A=[c A]`
  concateniamo il vettore `c` come prima colonna della matrice `A`, la concatenazione è orizzontale e il numero di righe di `c` deve corrispondere a quello di `A`.
- **creazione di un vettore riga**
  `b = 4 * ones(1, size(A,2));`
  si crea un vettore riga di lunghezza pari al numero di colonne di `A`, usando `size(A,2)` per ottenere il numero di colonne di `A`. Ogni elemento del vettore è uguale a 4.
- **Aggiunta di una riga all'inizio e alla fine**:
  `A = [r; A];`
  `A = [A; r];`
  Si aggiunge il vettore riga `r` come prima riga (`[r; A]`) e come ultima riga (`[A; r]`). La concatenazione è verticale.
- **Eliminazione di una riga specifica**:
  `A(3, :) = [];`
  Si elimina la terza riga della matrice utilizzando `A(indice, :) = []`.

Dopo aver definito il vettore $x = [1 : -0.1 : 0]$ *spiegare il significato dei seguenti comandi Matlab:*

```
>> x([1 4 3]);
>> x([1:2:7 10])=zeros(1,5);
>> x([1 2 5])=[0.5*ones(1,2) -0.3];
>> y=x(end:-1:1);
```

Script:

```
clear
clc
x=[1:-0.1:0]

x([1 4 3])
% seleziona gli elementi di x nelle posizioni 1, 4 e 3

x([1:2:7 10])=zeros(1,5)
% da 1 a 7 con passo due + un altro indice 10 alla fine;
```

```
% questi indici vengono poi sostituiti da 0 => ci serviranno 5 zeri perciò creiamo un
vettore zeros(1,5)

x([1 2 5])=[0.5*ones(1,2) -0.3]
% [1 2 5] seleziona gli elementi di x nelle posizioni 1, 2 e 5
% 0.5*ones(1,2) crea un vettore [0.5 0.5] che viene inserito nelle posizioni 1 e 2 di
x
% [-0.3] viene assegnato alla posizione 5

y = x(end:-1:1)
% end:-1:1 cea un vettore che parte dall'ultimo elemento di x (end) e arriva al primo
invertendolo (con passo -1)
```

3. *Usare le variabili e le operazioni vettoriali per osservare la convergenza in $\mathbb{N}$ delle successioni*

$$\left(1+\frac{1}{n}\right)^n \to e\,, \qquad \frac{4n}{n+2} \to 4\,, \qquad \log\left(1+\sqrt{\frac{n}{n+1}}\right) \to \log 2\,.$$

Script

```
clear
clc

n = linspace(0,10^8, 10)'

%definizione della prima successione
s1=(1 + 1./n).^n;

% verifichiamo che la successione converga al numero di nepero,
%ci aspettiamo che questa differenza tenda a 0
abs(exp(1)-s1); %s1 converge al numero di nepero

%definizione della seconda successione
s2=4*n./(n+2);

%verifica
abs(4-s2); %s2 converge a 4

%definizione della terza successione
s3=log(1+sqrt(n./(n+1)))

%verifica
abs(log(2)-s3) %converge a log(2)
```

4. Osservare la convergenza nel calcolo dei limiti delle seguenti funzioni

- $x \cdot (\sqrt{(x^2+1)} - x)$
- $x \cdot \sqrt{(x^2+1)} - x^2$
- $x/(\sqrt{(x^2+1)} + x)$

Script:

```
clear
clc

x=linspace(0, 10^10, 10)'

%l1 e l2 a causa di problemi numerici si comportano in modo diverso rispetto a l3

l1 = x.*(sqrt(x.^2+1)-x);
%quando x è molto grande, (sqrt(x.^2+1) è quasi uguale a x, quindi la
%sottrazione (sqrt(x.^2+1)-x comporta cancellazione numerica. Ciò accade
%perchè i numeri molto vicino tra loro perdono precisione nelle operazioni
%di sottrazione

l2 = x.*sqrt(x.^2+1)-x.^2;
%anche l2 soffre di cancellazione numerica per valori molto grandi di x.

l3 = x./(sqrt(x.^2+1)+x);
%questa è la riscrittura delle funzioni l1 e l2 senza sottrazioni problematiche,
%non c'è cancellazione numerica perché il numeratore e il denominatore
%sono dello stesso ordine di grandezza e vengono trattati in modo più
%stabile dal punto di vista numerico
```

5. *Utilizzare il comando* `diag` *per generare la matrice tridiagonale* A *di dimensione* 9×9 *i cui elementi della diagonale principale coincidono con* -2 *e quelli delle codiagonali con* 1. *Successivamente scambiare in* A *dapprima le righe* 3 *e* 6, *e di seguito, le colonne* 1 *e* 4.

Script:

```
clear
clc

d = -2 * ones(9, 1); %vettore di 9 righe e 1 colonna di elementi pari -2

codiagonale = ones(8,1);

A = diag(d) + diag(codiagonale, 1) + diag(codiagonale, -1)
%la matrice A è formata da:
% - diag(d) crea una matrice diagonale (9x9) con il vettore d sulla diagonale
principale
% - diag(codiagonale,1) posiziona il vettore codiagonale sulla codiagonale superiore
(offset =1)
% - diag(codiagonale,-1) posiziona lo stesso vettore sulla codiagonale inferiore
(offset =1)
%la somma di questi crea una matrice tridiangolare con -2 sulla diagonale principale
e 1 sulle codiagonali

%scambiamo la terza e la sesta riga della matrice
A([3 6], :) = A([6 3], :)
%seleziona le righe 3 e 6, con le rispettive colonne
```

```
%scambiamo la prima e la quarta colonna
A(: , [1 4]) = A(: , [4 1])
```

6. *Definire la matrice*

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

e comprendere il significato dei seguenti comandi Matlab:

```
>> size(A);
>> B=A.*A;
>> B=A*A;
>> B=A'*A;
>> A(1:2,4),A(:,3),A(1:2,:),A(:,[2 4]),A([2 3 3]);
>> A(3,2)=A(1,1);
>> A(1:2,4)=zeros(2,1);
>> A(2,:)=A(2,:)-A(2,1)/A(1,1)*A(1,:);
```

Script:

```
clear
clc

A = [1 2 3 4; 5 6 7 8; 9 10 11 12]

size(A);
%dimensioni della matrice; stampa 3 4 (righe colonne)

B=A.*A
%prodotto elemento per elemento, in cui ogni elementi di B è il quadrato
%del corrispondente elemento di A

%B=A*A
%dà un messaggio di errore, in quanto questa operazione è possibile solo
% se A è quadrata o se il numero di colonne di A corrisponde al numero di
% righe della seconda matrice

B=A'*A
%moltiplicazione tra la trasposta di A e A

B=A*A'
%moltiplicazione tra A e la sua trasposta --> output diverso

A(1:2,4)
%accede agli elementi della prima e seconda riga della quarta colonna

A(:,3)
%accede a tutti gli elementi della terza colonna

A(1:2,:)
%accede alle prime due righe di tutte le colonne
```

```
A(:,[2 4])
%accede a tutte le righe delle colonne 2 e 4

A([2 3 3])
%crea un sottoinsieme utilizzando elementi selezionati

A(3,2)=A(1,1)
%sostituisce il valore dell'elemento alla terza riga e seconda colonna
%con il valore di A(1,1)

A(1:2,4) = zeros(2,1)
%gli elementi della prima e seconda riga nella quarta colonna vengono
%impostati a 0

A(2,:) = A(2,:) - A(2,1)/A(1,1)*A(1,:)
%modifica la seconda riga della matrice eliminando la dipendenza lineare
% dalla prima riga. È una tipica operazione utilizzata nei metodi di
% eliminazione, come quello di Gauss.
```

7. *Definire la matrice*

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\ 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \end{bmatrix}$$

e successivamente:

a) *generare le matrici S triangolare superiore e I triangolare inferiore i cui elementi non nulli coincidano con gli elementi omonimi di A; successivamente, porre tutti gli elementi della diagonale principale della matrice S uguali a 0 e quelli della matrice I uguali a 1;*

b) *generare le matrici B_1, B_2 e B_3 rispettivamente tridiagonale, bidiagonale superiore e bidiagonale inferiore, i cui elementi coincidano con gli elementi omonimi di A.*

Script:

```
clear
clc

A1=(1:8)' * ones(1,8)
% Il prodotto `(1:8)' * ones(1,8)` crea una matrice `A1` di dimensioni 8×8, in cui
% ogni riga è formata dalla moltiplicazione di ciascun elemento del vettore colonna
% `(1:8)` con il vettore riga `ones(1,8)`
A2=(1:100)' * ones(1,100)

% la funzione triu(A1) estrae la parte triangolare superiore della matrice A1
```

```
% (inclusa la diagonale principale), impostando a zero tutti gli elementi al di
% sotto della diagonale
S = triu(A1)

% la funzione tril(A1) estrae la parte triangolare inferiore della matrice A1
% (inclusa la diagonale principale), impostando a zero tutti gli elementi al di
% sopra della diagonale
I = tril(A1)

% 2 metodi per azzerrare la diagonale di S:

% 1 metodo:
% estraggo la diagonale della matrice S sottoforma di vettore colonna
sD=diag(S);
% trasforma il vettore diagonale in una matrice diagonale
s=diag(sD);
% sottraggo s da S in modo da azzerare gli elementi della diagonale principale di S
S=S-s

% 2 metodo:
% estraggo la parte triangolare superiore della matrice A1, escludendo la diagonale
% principale. Gli elementi della diagonale principale e quelli sotto di essa
% vengono impostati a 0
S=triu(A1,1)

% aggiungere 1 alla diagonale principale di I:
I = tril(A1, -1) + eye(size(A1))
% estraggo la parte triangolare inferiore di A1, escludendo la diagonale principale
% genero una matrice identità delle stesse dimensioni di A1, con 1 sulla diagonale
% principale. Sommando le due matrici otteniamo una matrice triangolare inferiore
% con 1 sulla diagonale principale.
```

8. Al variare del parametro $p = 10^{\alpha}$, con $\alpha = 1 : 10$, calcolare mediante le note formula risolutive, le radici dell' equazione di quarto grado

$$x^4 - b\,x^2 + 1 = 0,$$

con $b = \frac{1+p^2}{p}$. In seguito, tradurre tali formule in istruzioni di assegnazione Matlab in una funzione matlab che ha α come parametro di input e le 4 soluzioni come output. Predisporre una tabella con gli errori relativi commessi da Matlab nel calcolo numerico delle radici dell'equazione assegnata. Motivare i risultati ottenuti.

Script:

```
clear

clc

alfa = 10;

p = 10^alfa;

b=(1+p^2)/p;
```

```
%risolvo: t^2 - b*t +1 = 0

t1=(b + sqrt(b^2-4))/2;

t2=(b - sqrt(b^2-4))/2;

x1=sqrt(t1)

x2=-sqrt(t1)

x3=sqrt(t2)

x4=-sqrt(t2)

[x1 x2 x3 x4]

sqrt(p)

sqrt(1/p)
```

DA FARE