LABORATORI DI CALCOLO NUMERICO

**LAB LEZIONE 6**

**SCHEDA 1**

**3. Mediante una sequenza di istruzioni di assegnazione in Matlab:**

**– Calcolare il raggio di una sfera che ha un volume del 30% più grande di una sfera di raggio 5 cm.**

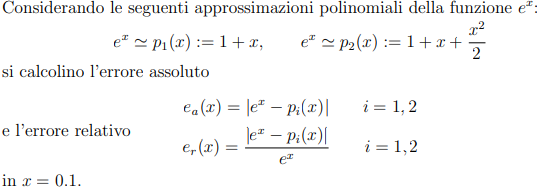
Comandi:

raggio= 5

volume\_sfera1= 4/3 \* pi \* raggio.^3

volume sfera2 = volume\_sfera1 /100 \* 30 \* volume\_sfera1

raggio2= ( volume\_sfera2 / pi \* ¾) ^(⅓)



x= 0.1

p1 = 1 + x

p2 = 1 + x + (x^2)/2

Ea1= abs( exp(x) - p1 )

Ea2 = abs(exp(x) - p2 )

// p1 ha un errore più grande

Er1= abs( exp(x) - p1 ) / exp(x)

Er1= abs( exp(x) - p2 ) / exp(x)

//p2 ha errore più grande

**– Calcolare le radici delle equazioni:**

**2t^2 − 4t − 1 = 0 x^4 + 2x^2 − 3 = 0 .**

a=1

b=2

c= -3

t1=(-b-sqrt(b.^2-4\*a\*c))/(2\*a) // calcolo radice tramite il delta

t2=(-b+sqrt(b.^2-4\*a\*c))/(2\*a)

a\*t1.^2+b\*t1+c //controllo che t1 sia effetivamente la radice dell’equazione, se fa zero, allora è correto

x1= sqrt(t1)

x2= -sqrt(t1)

x3= sqrt(t2)

x4= - sqrt(t2)

**4. Generare il vettore riga e il vettore colonna y di elementi equidistanti 1, 2, ..., 10 e 10, 9, ..., 1 rispettivamente e farne il prodotto scalare. Generare inoltre il vettore colonna z costituito dai valori della funzione seno in 11 elementi equidistanti nell’intervallo [0, 1].**

x= [1:10]

y= [10:-1:1]'

x\*y //prodotto scalare

z= sin(linspace(0,1,11))’

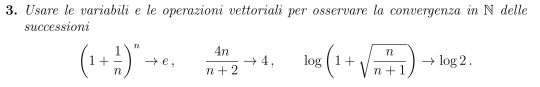
**5. Generare il vettore riga x e il vettore colonna y di elementi equidistanti 25, 28, 31, ..., 91 e 100, 98, 96, ..., 10 rispettivamente. Generare inoltre il vettore colonna z costituito da 33 elementi equidistanti nell’intervallo [−15, −10].**

x=[25:3:91]

y=[100:-2:10]’

z= linspace [-15,-10-33]

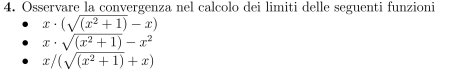
**SCHEDA 2**



x=[1:1000]

x.\*4./(x+2) // tende a 4

(1+1./x).^x // tende al numero di nepero



LEZIONE 7

GRAFICA:

X=linspace(0.2,pi)

Y= cos(x)

plot(X;Y)

axis ([0 6.3 -1 1])

axis square

xlabel (‘ascisse’)

ylabel(‘funzione coseno’)

title (‘grafico’)

grid on // griglia

plot(X;Y, ‘\*’) //evidenzio solo i punti

plot(X;Y, ‘\*-’) //evidenzio i punti con segmenti

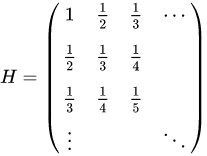
hold on // posso inserire unaltro grafico

legend (‘cos’, ‘sin’) //aggiunge una legenda specificando i nomi delle curve

subplot (1,2,1) // posso affiancare più grafici

LEZIONE:

hilb(15) //genera matrice di hilbert



H=hilb(15)\*ones(15,1)

x= hilb(15)/H //dovrebbe darmi un vettore di elementi uguale a 1 ma sono condizionati perche questa matrice (hilb) ha un numero di condizionamento molto elevato

rcond(x) // stima il numero di condizionamento della matrice x, se vicino a 1 ha basso condizionamento se vicino a eps ha alto condizionamento

LEZIONE 8

RISOLUZIONE SISTEMA LINEARE TRAMITE ELIMINAZIONE DI GAUSS E SOSTITUZIONE ALL’INDIETRO

A=[1 1/2 1/3; 1/2 1/3 1/4; 1/3 1/4 1/5]; //matrice (1)

b=[11/6;13/12;47/60];

%primo passo algoritmo di Gauss

M1=eye(3);

M1(2,1)=-1/2;

M1(3,1)=-1/3;

A1=M1\*A (seconda riga matrice 2)

b1=M1\*b (seconda riga matrice 2)

pause

%secondo passo algoritmo di Gauss

M2=eye(3);

M2(3,2)=-1;

A2=M2\*A1 (terza riga matrice 2)

b2=M2\*b1 (terza riga matrice 2)

%risoluzione sistema con matrice triangolare superiore

x=A2\b2

LEZIONE 10

SCHEDA 2

ES 5

v = ones(1, 9) \* (-2) //creo un vettore di 9 elementi uguali a -2

diag(v) //creo una matrice con diagonale usando il vettore creato

v2 = ones(1, 8) //creo un secondo vettore da 8 elementi tutti 1

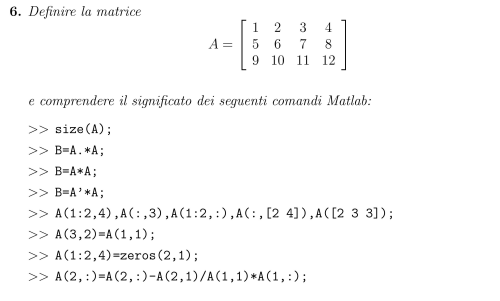
m = diag(v) + diag(v2, 1) + diag(v2, -1) // creo una matrice dove diagonale ho -2 e codiagonali ho 1

//scambio le righe della matrice

v3 = m(3, :); //salvo la 3 riga in una variabile di appoggio

m(3, :) = m(6, :); // copio la 6 riga al’interno della 3

m(6, :) = v3; // copio la 3 riga salvata nella variabile di appoggio all'interno della 6 riga



A = [1:4; 5:8; 9:12] //definiamo la matrice

size(A); //stampa la dimensione in righe e colonne

B = A.\*A; //moltiplica la matrice per se stessa elemento per elemento

B = A\*A // errore numero colonne righe non uguale

B=A'\*A;

%A(1:2,4) // stampa le prime due righe della 4 colonna

%A(:,3), // stampa tutte le righe della 3 colonna

%A(1:2,:), // stampa tutte le colonne delle prime due righe

%A(:,[2 4]), // stampa la 2 e la quarta colonna

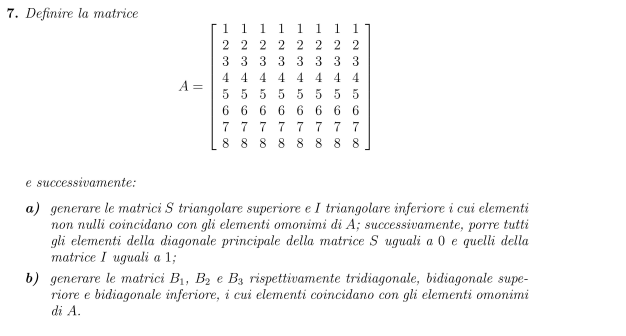
%A([2 3 3]) // stampa il secondo e due volte il terzo elemento

A(3,2)=A(1,1);

A(1:2,4)=zeros(2,1); // sostituisce con dei zeri

A(2,:)=A(2,:)-A(2,1)/A(1,1)\*A(1,:)

A(3,:)=A(3,:)-A(3,1)/A(1,1)\*A(1,:)



A=[1:8]'\*ones(1,8) //genera matrice

A)

S=triu(A) //triangolare superiore

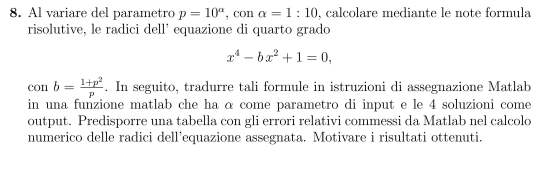
I=tril(A) //triangolare inferiore

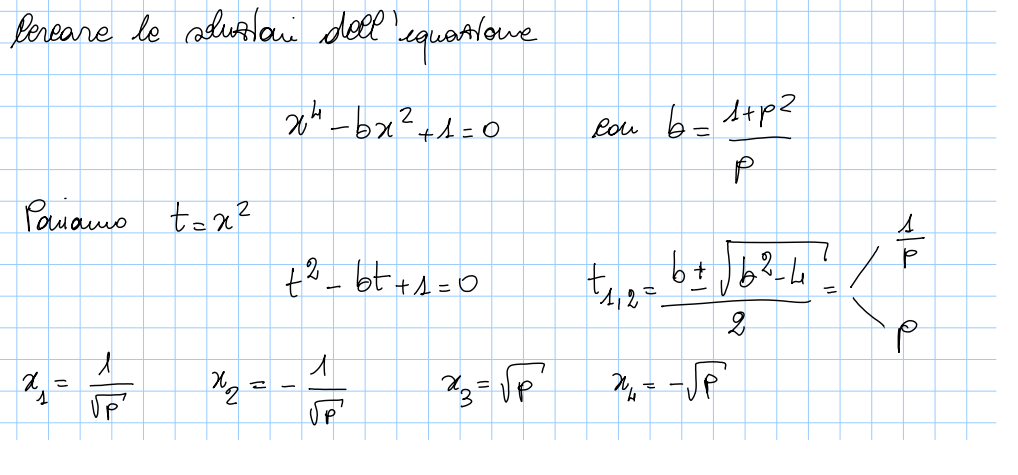
S=S-diag(diag(S)) //

I=I-diag(diag(I))+diag(ones(1,8))

B)

B1=diag(diag(A))+diag(diag(A,1),1)+diag(diag(A,-1),-1)





for a=1:10

p=10^a;

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

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

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

t1e=1/p;

t2e=p;

x1=sqrt(t1);

x2=-sqrt(t1);

x3=sqrt(t2);

x4=-sqrt(t2);

x1e=1/sqrt(p);

x2e=-1/sqrt(p);

x3e=sqrt(p);

x4e=-sqrt(p);

x=[x1,x2,x3,x4]';

xe=[x1e,x2e,x3e,x4e]';

a

abs(x-xe)

pause

end

LEZIONE 12

X=[-6:6];

C=[1 -6 11 -6];

r1=roots(C);

%plot(X,polyval(C,X))

z1=fzero(@(x) x.^3-6\*x.^2+11\*x-6,1.6);

z2=fzero(@(x) (x.^3-6\*x.^2+11\*x-6)./(x-z1),1.6);

z3=fzero(@(x) (x.^3-6\*x.^2+11\*x-6)./((x-z2).\*(x-z1)),1.6);

x=1;

p2=x.^2-2\*x+1;

%z4=fzero(@(x) x.^2-2\*x+1,2)

C2=[1 -2 1];

z4=roots(C2)

C3=[1 -7 15 -13 4];

z5=roots(C3)

polyval(C3,1)

[C4,R]=deconv(C3,[1 -z5(1)])

roots(C4)

**LEZIONE 14**

**%calcolo di radici attraverso il metodo**

**%di Newton**

close all

clear

clc

fun = @(x) exp(-x)-10.^-9;

dfun = @(x) -exp(-x);

x0 = 0; //INNESCO

max\_iter = 100; //CRITERIO DI ARRESTO PER NUM ITERAZIONI

toll = 10^(-3); //CRITERIO DI ARRESTO PER TOLLERANZA

[approx n\_iter] = Newton(x0, fun, dfun, toll, max\_iter); //GUARDARE FUNZIONE NEWTON

disp("Num iterazioni: " + n\_iter);

approx

%GRAFICO

x\_plot=linspace(19, 21);

plot(x\_plot, fun(x\_plot));

grid on;

radice = -log(10.^-9)

err = abs(radice - approx);

figure;

semilogy([1:n\_iter+1], err, '-\*');

**FUNZIONE NEWTON**

function [approx, num\_iter] = Newton(x0, fun, dfun, toll, max\_iter)

%algoritmo di Newton

x = x0;

approx = [x];

num\_iter = 0;

crit\_arr = 1;

while(num\_iter < max\_iter && crit\_arr > toll)

num\_iter = num\_iter + 1;

x\_next = x - (fun(x)/dfun(x));

approx = [approx; x\_next];

%crit\_arr = abs(x\_next - x); %criterio incremento

crit\_arr = abs(fun(x\_next)); %criterio residuo

x = x\_next;

end

**METODO BISEZIONE**

%calcolo di radici attraverso il metodo

%di bisezione

clear

clc

close all

tolleranza=10^-5;

a=0;

b=1.5;

err=1;

i=1;

while err>tolleranza

x=(a+b)/2;

if (cos(2\*a)^2-a^2)\*(cos(2\*x)^2-x^2)<0

b=x;

elseif (cos(2\*a)^2-a^2)\*(cos(2\*x)^2-x^2)==0

disp('x � radice');

return;

else

a=x;

end

err=abs(b-a);

e(i)=err;

i=i+1;

end

semilogy(e);

x

**LEZIONE 15**

**2. Scrivere in Matlab gli algoritmi per la risoluzione di sistemi triangolari inferiore e superiore.**

%Metodo di sostituzione in avanti

function x=SostituzioneAvanti(A,b)

%Calcolo la grandezza della matrice

[n,m]=size(A);

%Passo 1

x(1)=b(1)/A(1,1);

%Passo 2...n

for i=2:n

x(i)=(b(i)-A(i,1:i-1)\*(x(1:i-1))')/A(i,i);

end