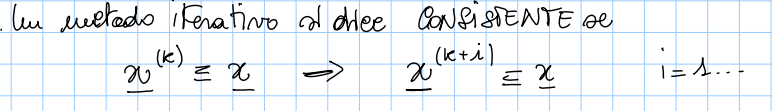
**DOMANDE TEORIA:**

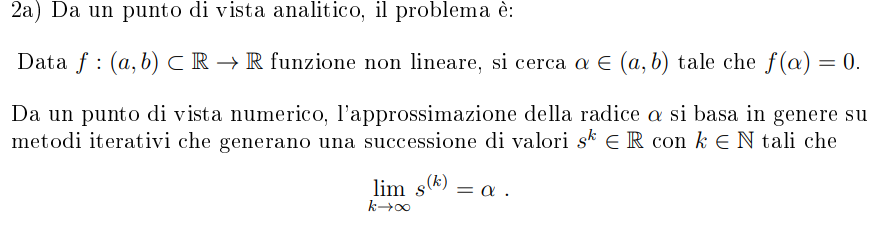
**Definizione di metodi diretti e metodi iterativi.**

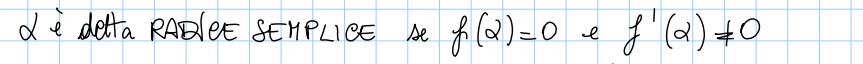
**i metodi diretti in aritmetica esatta forniscono la soluzione del sistema lineare in un numero finito di passi mentre i metodi iterativi forniscono la soluzione tramite una successione di approssimazioni della soluzione tendente all’infinito.**

**Metodo iterativo consistente**

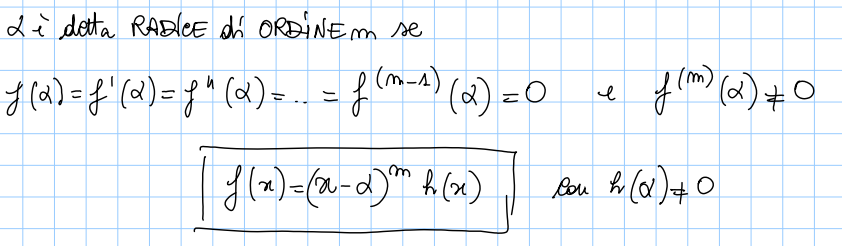
****

**Ricerca di radici di equazioni non lineari.**

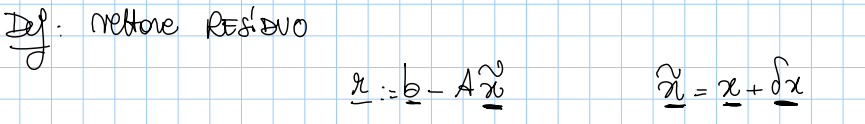
**Definizione di radice semplice**

****

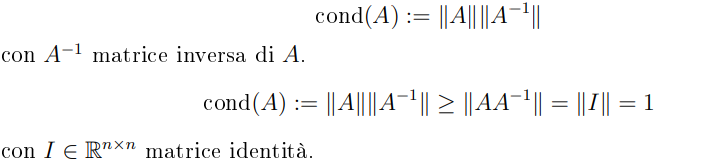
**radice di ordine m di equazioni non lineari:**

****

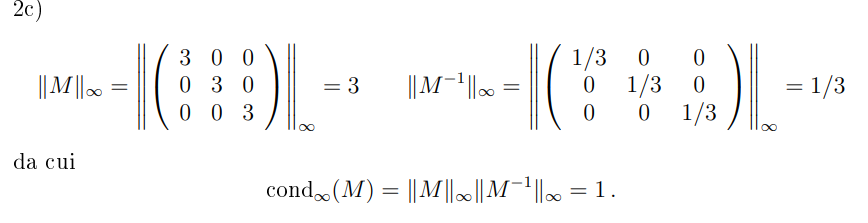
**DEFINIZIONE DI VETTORE RESIDUO**

****

**CONDIZIONAMENTO**

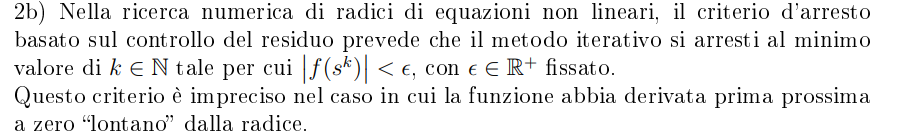


**esempio di calcolo di condizionamento di una matrice:**



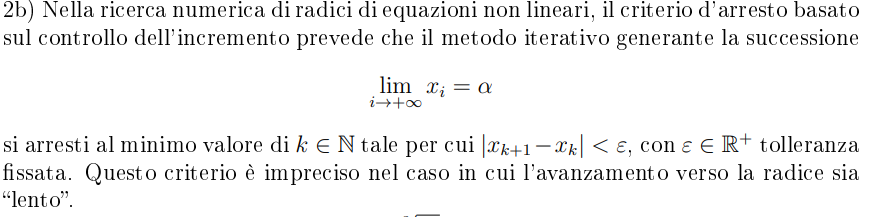
**CRITERI D’ARRESTO:**

per cicli iterativi:

* il criterio d'arresto basato sul controllo del **residuo**

**|f(xk)|<tolleranza**

* Criterio di arresto basato sull’ **incremento**



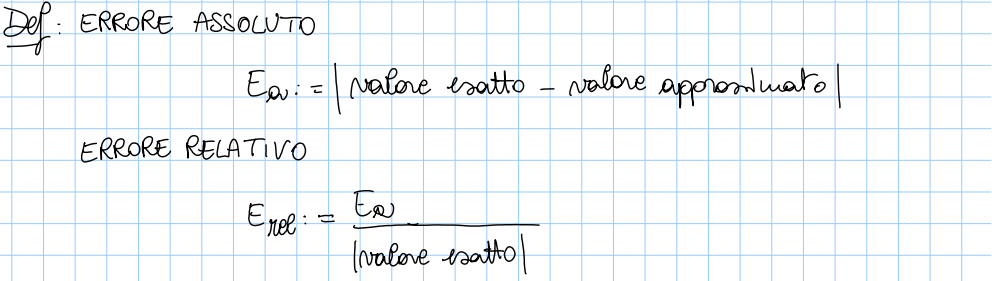
**|f(Xk+1)-f(Xk)|<toll**

PER MATRICI:

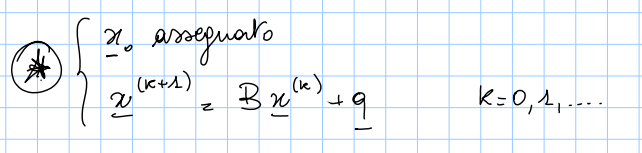
**residuo= norm(b-A\*x) < toll**

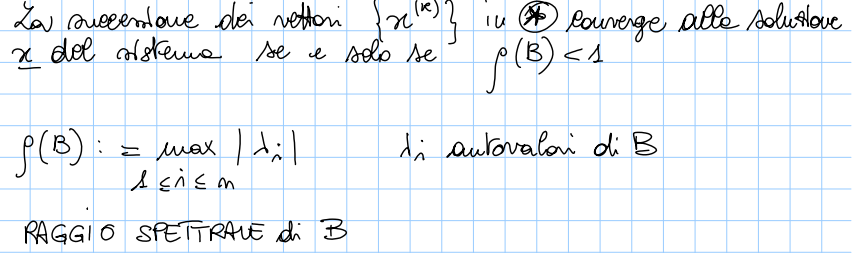
**incremento= norm(x-xo)<toll**

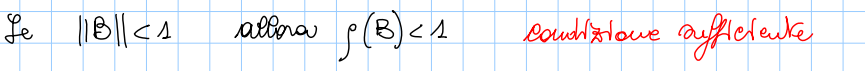
**ERRORI:**

****

**RAGGIO SPETTRALE DI B**

****

****

****

**MATLAB:**

**SISTEMA FLOATING POINT**

L’insieme dei numeri macchina è chiamato **SISTEMA FLOATING POINT**

**REALMIN E REALMAX:**

-**REALMAX** è il più grande numero positivo memorizzabile al di quale sopra di esso fa overflow (da come risultato INF)

-**REALMIN** è il minimo numero maggiore di 0, al di sotto di esso da 0 come risultato

alcuni numeri come **EPS** possono essere rappresentati tra 0 e REALMIN rinunciando alla rappresentazione IEEE

**L’EPSILON MACCHINA**

è il più piccolo numero macchina positivo x tale che (1+x) > 1

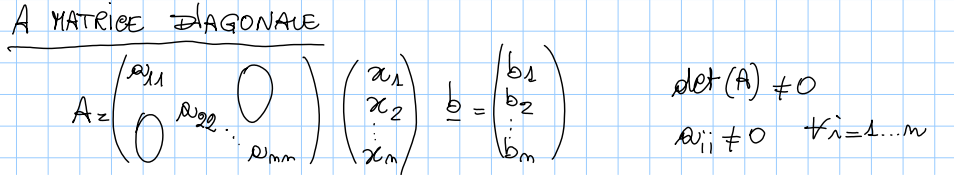
**Cancellazione numerica**

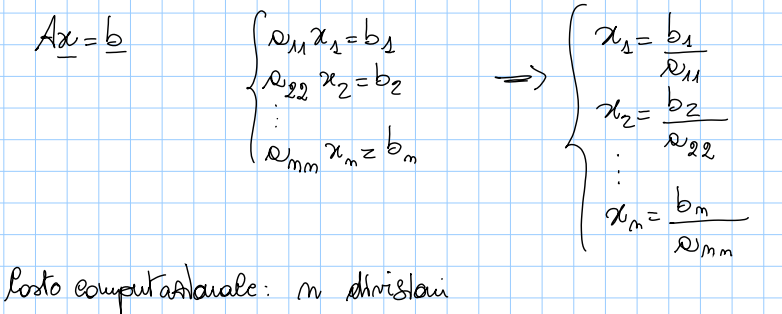
si ha quando si lavora con sottrazioni o somme ripetute di un numero, quindi ha un **cattivo condizionamento**

**Perdita di memorizzazione**

Si ha quando si lavora con un numero **molto piccolo**

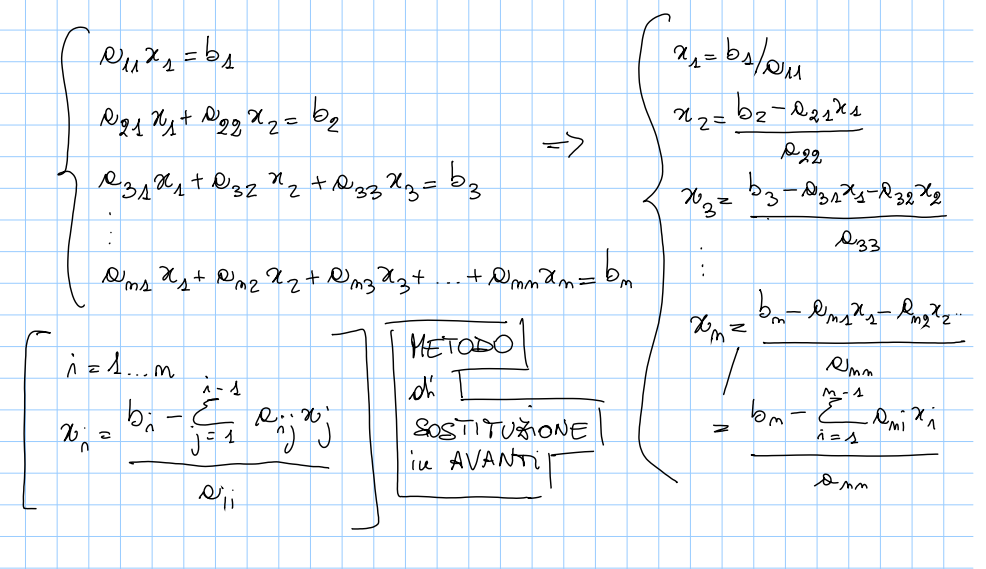
**METODI DIRETTI**

**MATRICE DIAGONALE**

****

**- Sostituzione in avanti**

Vogliamo risolvere il sistema lineare Ly = b con L **triangolare inferiore**



COSTO:

O(N^2)

IN MATLAB:

%Metodo di sostituzione in avanti

function x=SostituzioneAvanti(A,b)

%Calcolo la grandezza della matrice

[n,m]=size(A);

%Passo 1

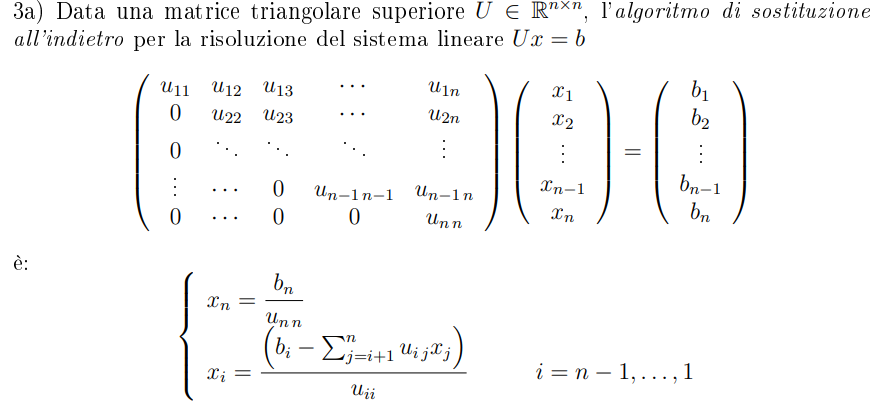
x(1)=b(1)/A(1,1);

%Passo 2...n

for i=2:n

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

end

**- Sostituzione all’indietro**

CALCOLO DEL COSTO:

O(N^2)

IN MATLAB:

**%Metodo di sostituzione all'indietro**

**function x=SostituzioneIndietro(A,b)**

**%Calcolo la grandezza della matrice**

**[n,m]=size(A);**

**%Passo 1**

**x(n)=b(n)/A(n,n);**

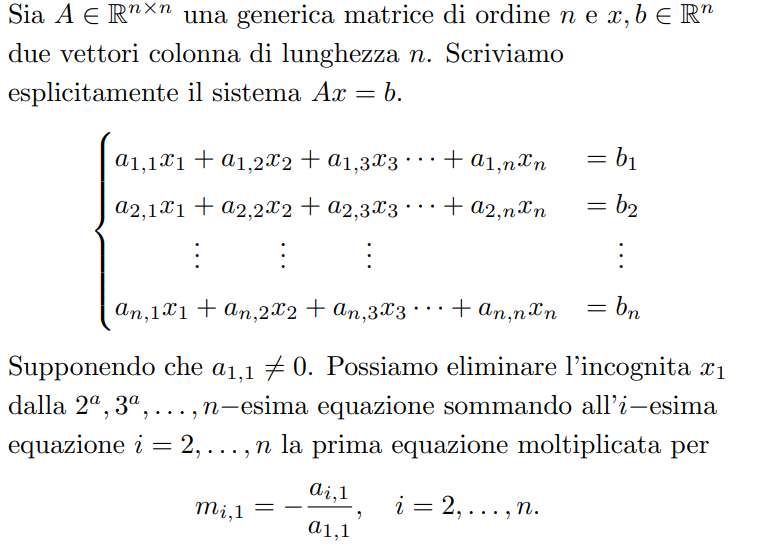
**%Passo n-1 1**

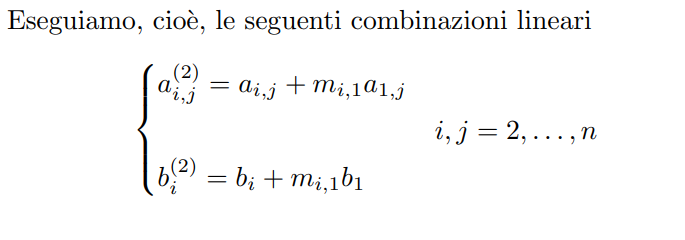
**for i=n-1:-1:1**

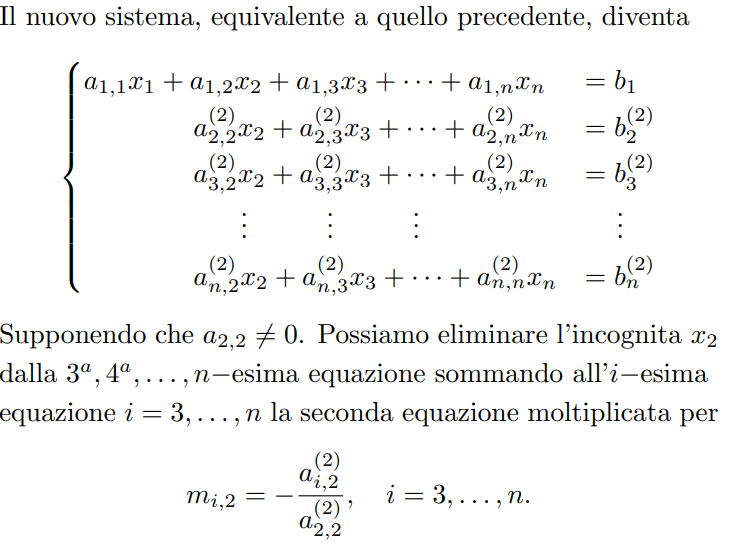
**x(i)=(b(i)-A(i,i+1:n)\*(x(i+1:n))')/A(i,i);**

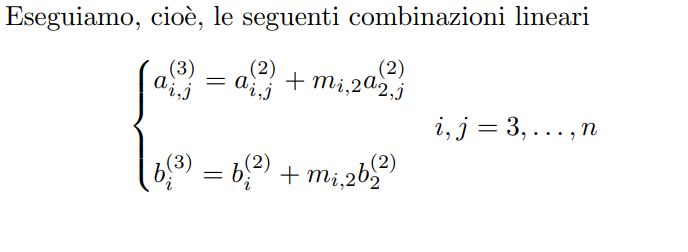
**end**

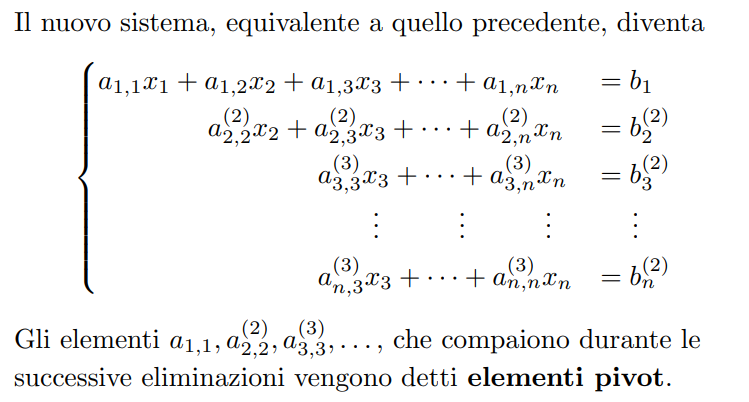
**METODO DI GAUSS (RIDUZIONE)**

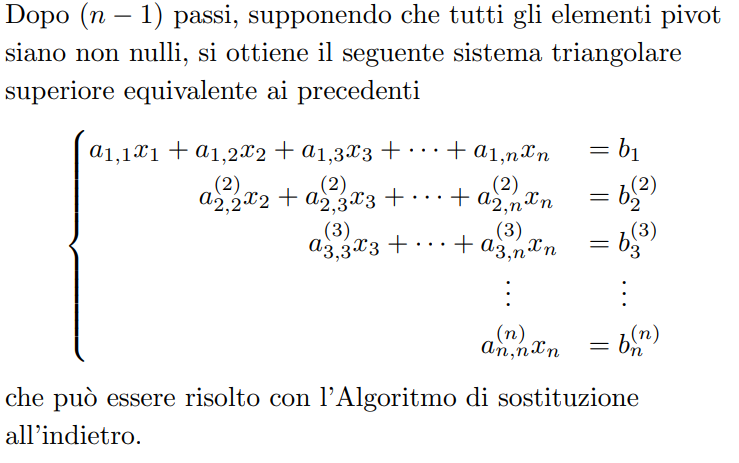
****

PASSO 1****

****

PASSO 2****





IN MATLAB:

%Risoluzione sistema lineare

clear

clc

close all

A=[1 1/2 1/3; 1/2 1/3 1/4; 1/3 1/4 1/5];

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

%primo passo algoritmo di Gauss

M1=eye(3);

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

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

A1=M1\*A

b1=M1\*b

%secondo passo algoritmo di Gauss

M2=eye(3);

M2(3,2)=-(A1(3,2)/A1(2,2));

A2=M2\*A1

b2=M2\*b1

%risoluzione sistema con matrice triangolare superiore

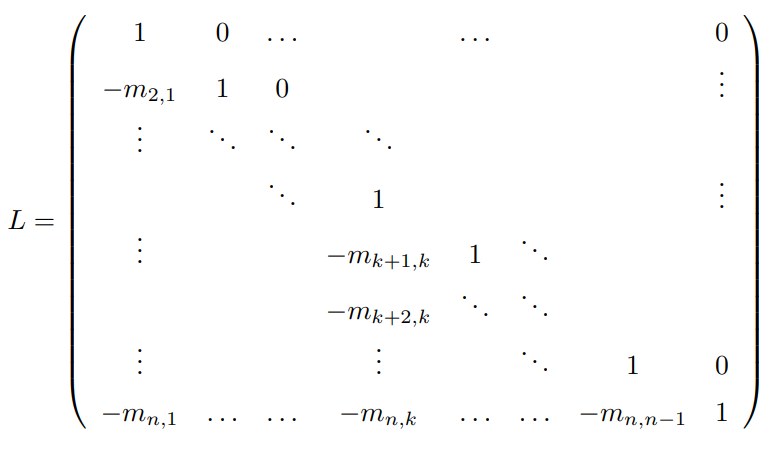
x=A2\b2

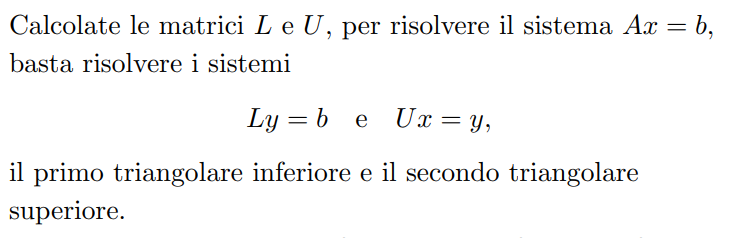
**FATTORIZZAZIONE LU**

La fattorizzazione LU indotta dal metodo di gauss esiste ed è unica se A è non singolare, cioè con DET(A) diverso da 0.

Il metodo di eliminazione di Gauss calcola esplicitamente la matrice U.

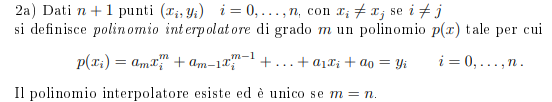
La matrice L è:

****

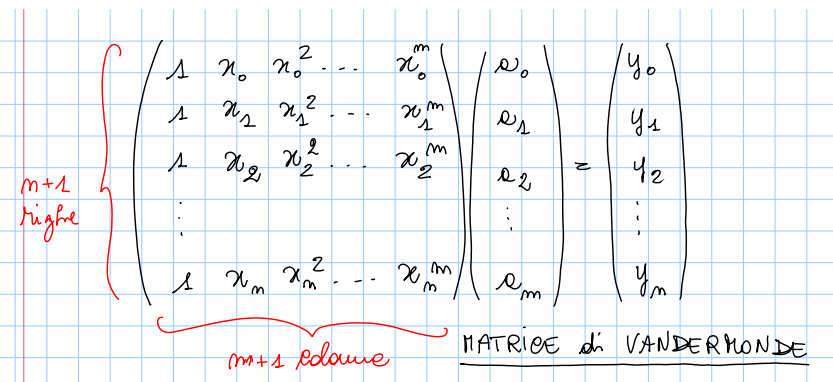
****

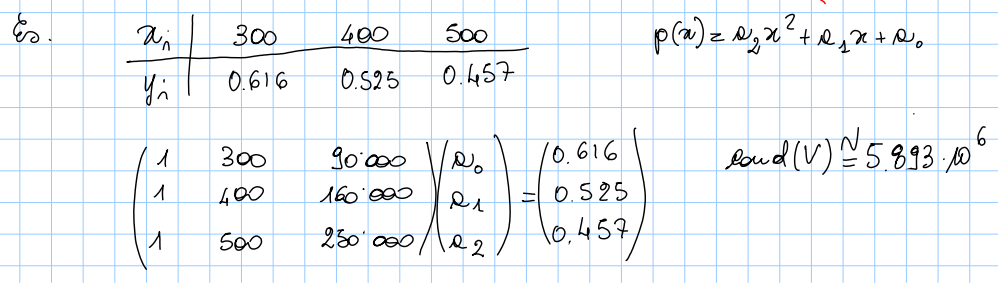
**PROBLEMA DELL’INTERPOLAZIONE POLINOMIALE**

**definizione di polinomio interpolatore:**

****

La matrice associata di questo sistema lineare si chiama matrice di Vandermonde.

****

****

**in matlab:**

%%%%Interpolazione

**n=5; % Grado del polinomio**

**a=0; b=2; %intervallo**

**x=linspace(a,b,n+1);% Nodi**

**f=@(x) exp(x).\*sin(2\*x); %funzione**

**y=f(x); %valori nei nodi**

**p=polyfit(x,y,n); %coefficienti polinomio interpolatore**

**plot(x,y,'\*')**

**hold on**

**g=linspace(a,b);**

**h=polyval(p,g); %polinomio interpolante**

**plot(g,h)**

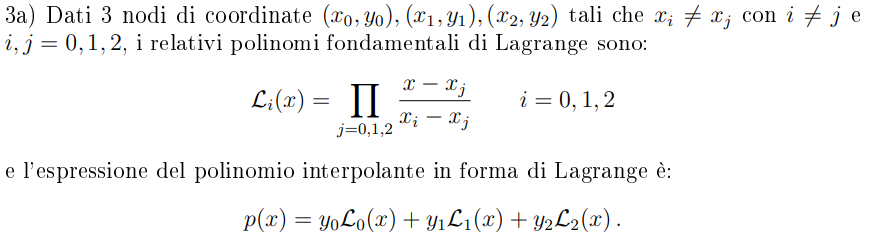
**y=f(g);**

**plot(g,y)**

**legend('punti interpolazione','polinomio interpolante','funzione')**

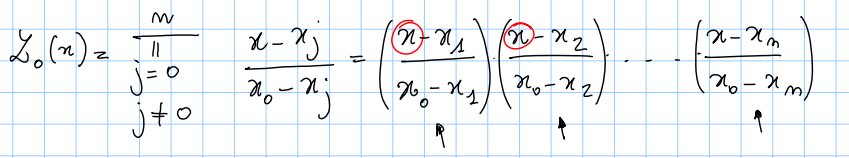
**grid on**

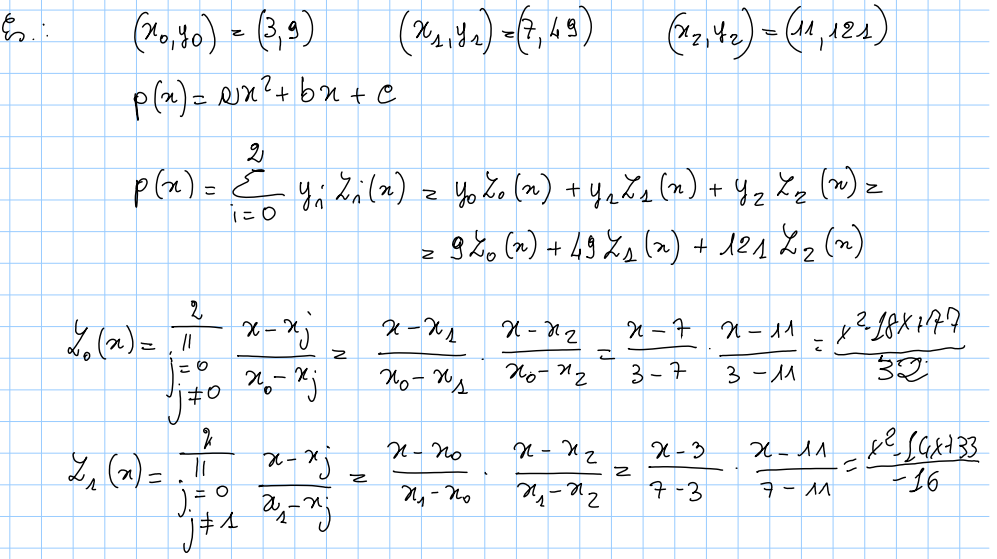
**POLINOMI FONDAMENTALI DI LAGRANGE**

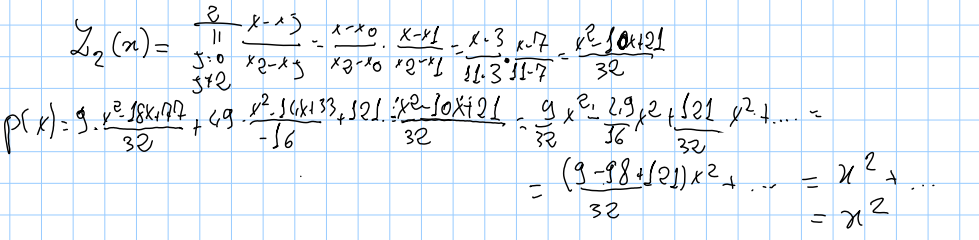


**con i diverso da J!!!**

**es:**

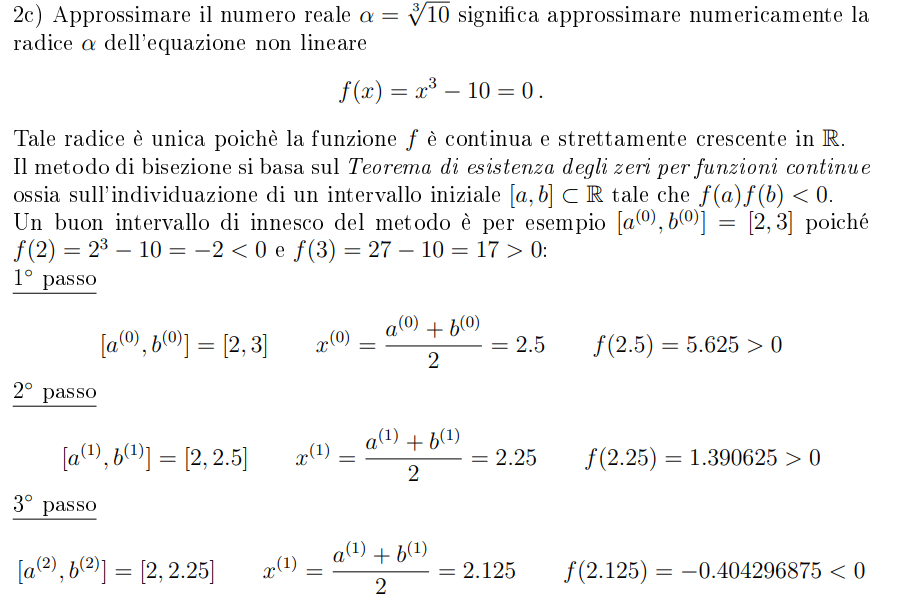
****

****

****

**METODI ITERATIVI**

**METODO DI BISEZIONE**



in matlab:

function x=Bisezione(a,b,toll)

i=1;

err=1;

while err<toll

x=(a+b)/2;

if (a^8-2)\*(x^8-2)<0

b=x;

elseif (b^8-2)\*(x^8-2)<0

return;

else

a=x;

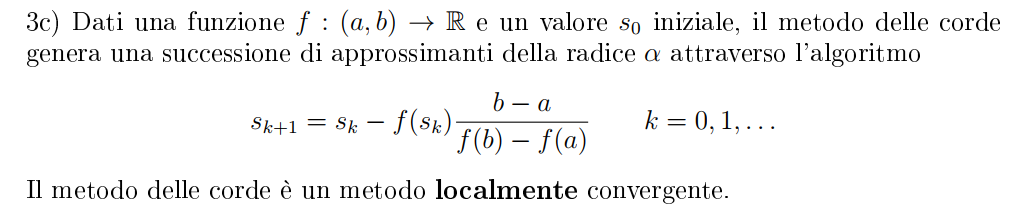
end

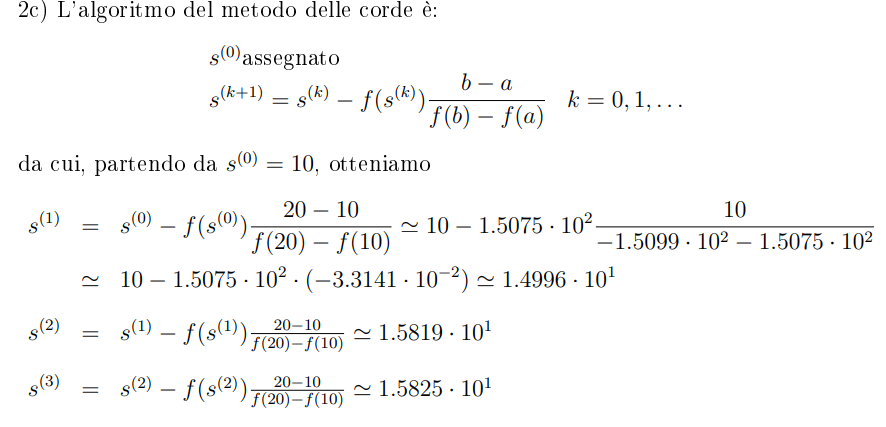
err=abs(b-a);

i=i+1;

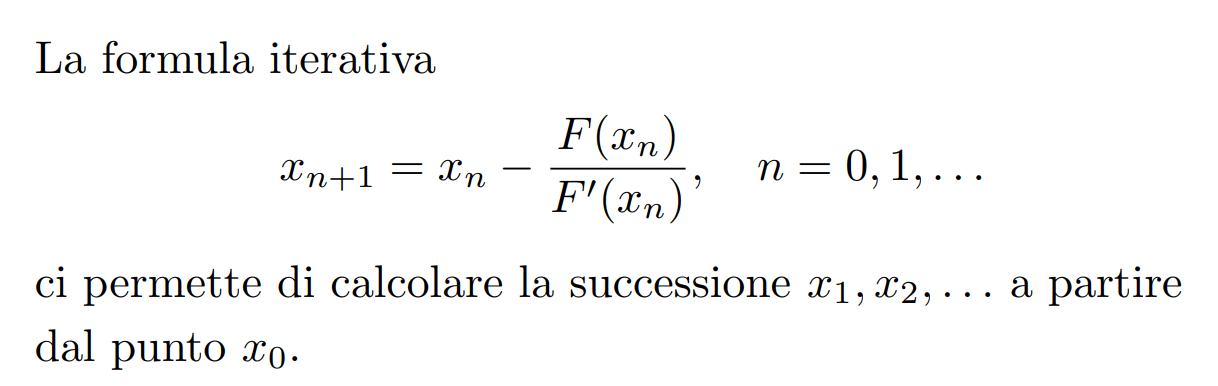
end

**METODO DELLE CORDE**

ESEMPIO: Eseguire analiticamente tre passi del metodo delle corde per la ricerca di una radice nell'intervallo (**a; b) = (10; 20)** della funzione **f(s) = sin(4s) - s2 + 250** partendo da **s(0) = 10.**

****

**METODO DI NEWTON (TANGENTI)**

****

IN MATLAB:

**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**

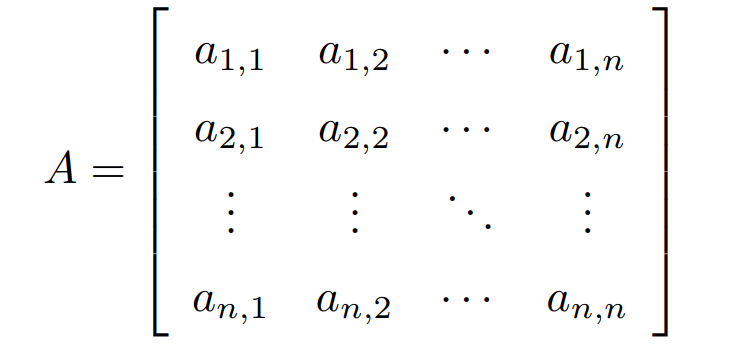
**crit\_arr = abs(fun(x\_next)); %criterio residuo**

**x = x\_next;**

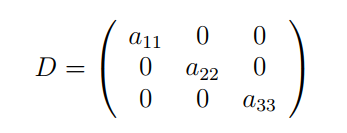
**end**

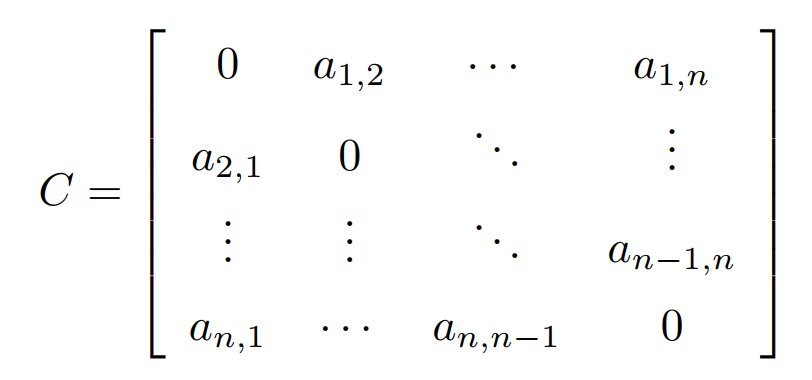
**METODO DI JACOBI**

Il metodo di Jacobi si basa sullo splitting della matrice A nella somma di due matrici A = D + C tali che:

****

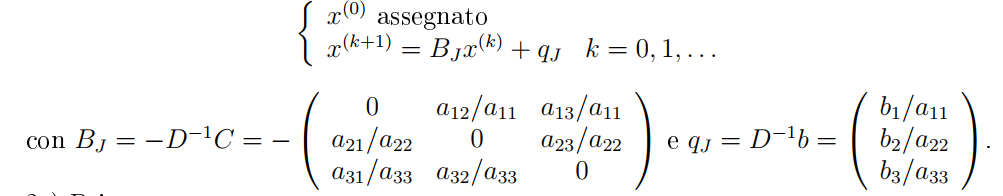
DIVENTA:



****

con aii ̸= 0 per i = 1; 2; 3.

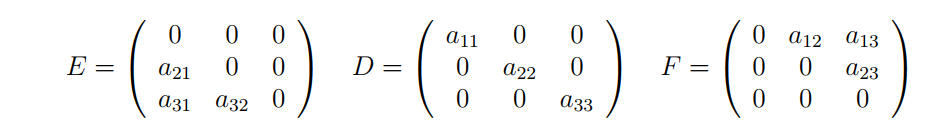
Nel caso in cui aii = 0 per qualche valore di i, prima si effettua un opportuno scambio di righe e/o colonne. L'algoritmo si può rappresentare nella seguente forma matriciale:



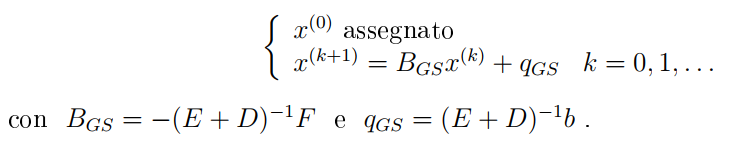
La matrice A deve essere strettamente diagonale dominante per convergere alla soluzione

**METODO DI GAUSS-SIDEL**

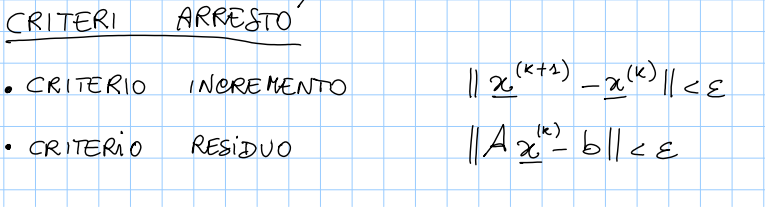
Il metodo di Gauss-Seidel si basa sullo splitting della matrice A nella somma di tre matrici A = D + E + F tali che:



L'algoritmo si può rappresentare nella seguente forma matriciale:



OSS: La matrice A deve essere strettamente diagonale dominante per convergere alla soluzione



IN MATLAB:

itMax = 1000;

toll = 10^-5;

D=diag(diag(A))

E=tril(A,-1)

F=triu(A,1)

dinv=inv(E+D)

bj=-dinv\*F

qj=dinv\*b

r=norm(eig(bj),"inf") %raggio spettrale

i=0;

x=ones(3,1);

while i<itMax

x0=x;

x=bj\*x+qj;

if norm(x0-x)<toll

break

end

err=norm([5 4 7]'-x);

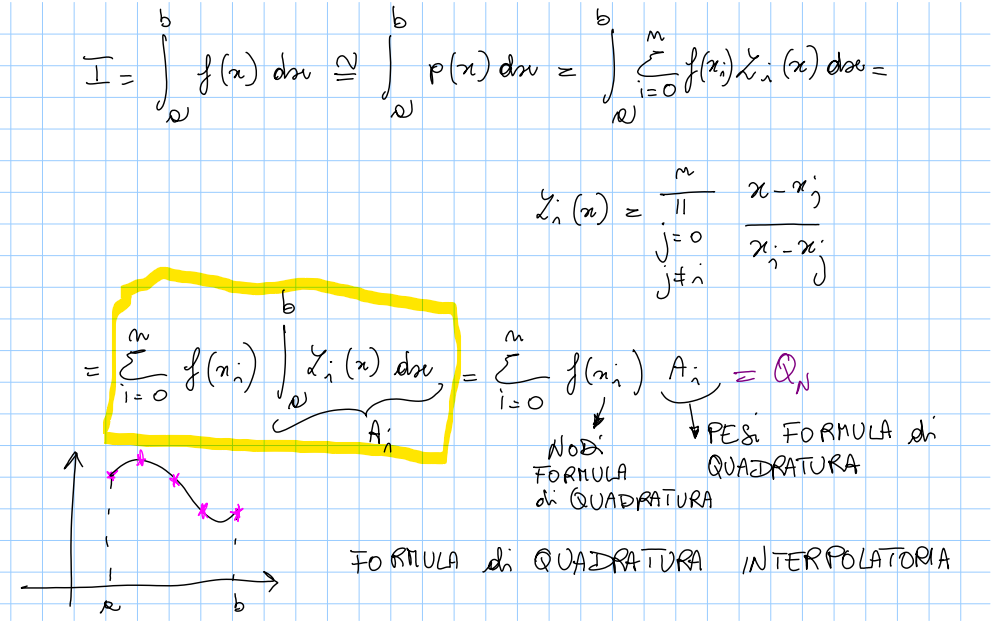
i=i+1;

end

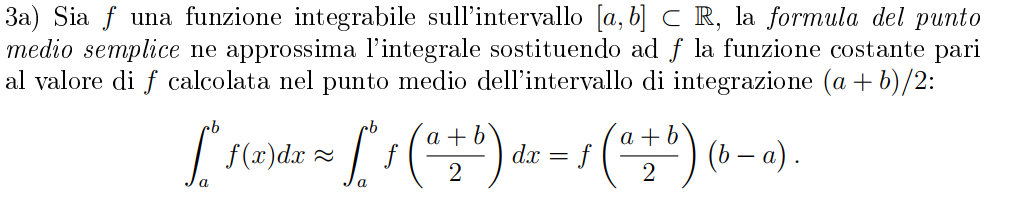
**INTEGRAZIONE NUMERICA**

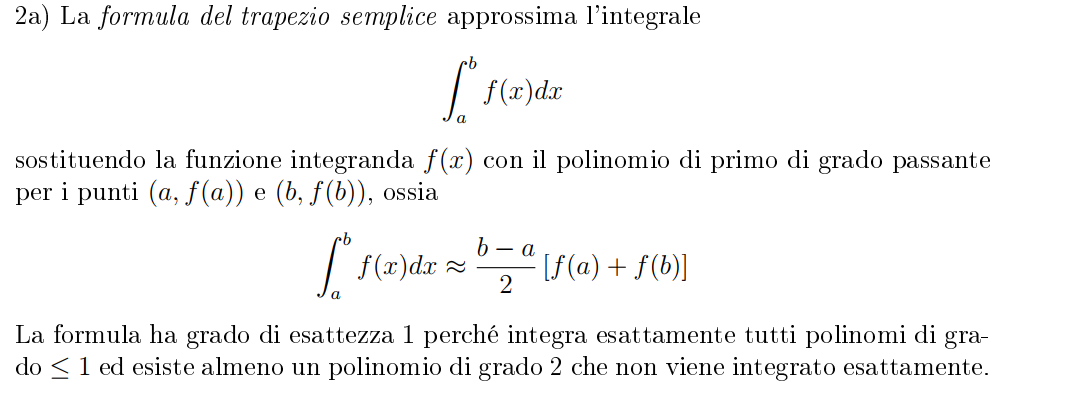
**FORMULA DELLA QUADRATURA INTERPOLATORIA**

**SOSTITUIAMO AD F IL SUO POLINOMIO INTERPOLATORIO DI LAGRANGE:**

****

**PUNTO MEDIO (O DEL RETTANGOLO) SEMPLICE**

****

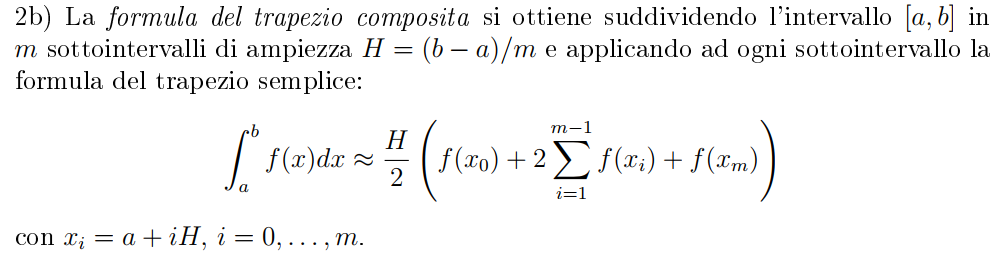
**TRAPEZIO SEMPLICE**

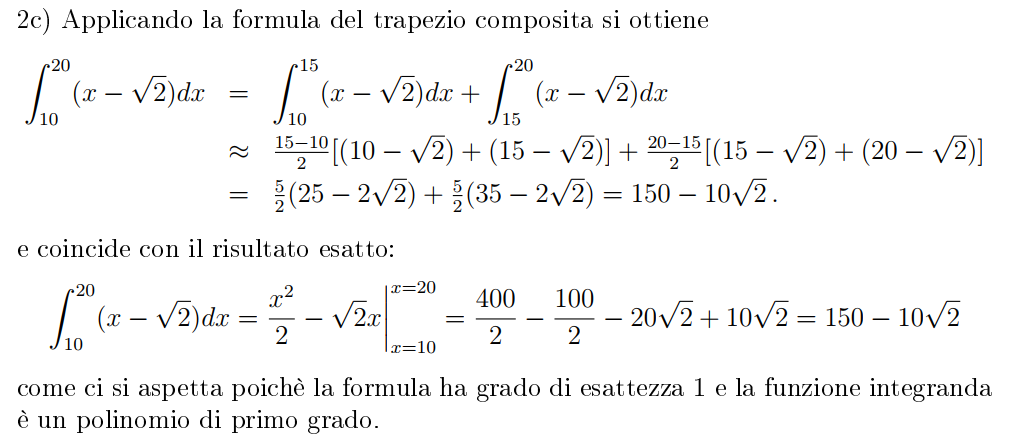
**IN MATLAB:**

**% formula trapezi semplice**

**I\_ts=(f(a)+f(b))\*(b-a)/2**

**TRAPEZIO COMPOSITA**

****



IN MATLAB:

**t = 5;**

**for indice = 1:t**

**n = 10^indice;**

**step=(b-a)\n;**

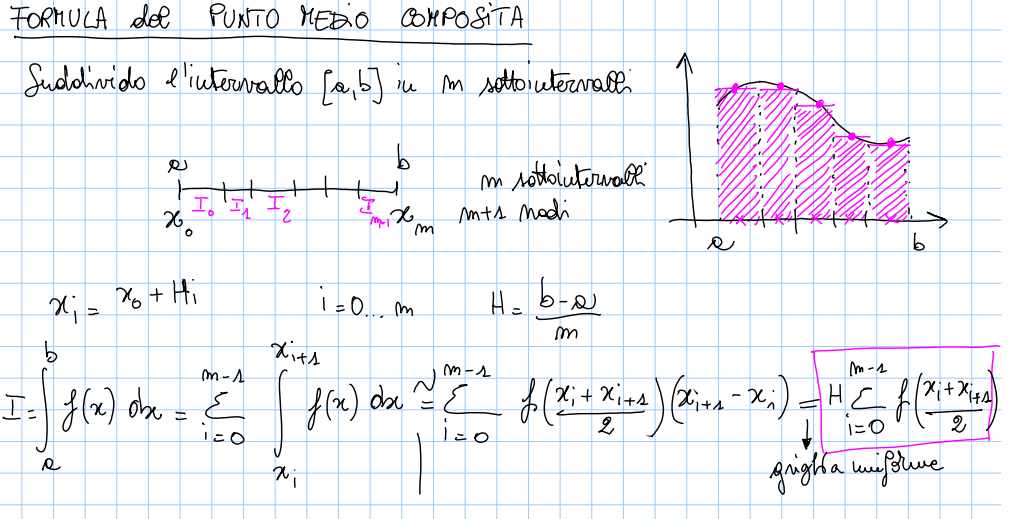
**nodi = linspace(a,b,n+1);**

**i = 1:n;**

**%formula del punto medio composita**

**res\_trapz = step/2 \* (f(nodi(1)) + f(nodi(end)) + 2\*sum(f(nodi([2:end-1]))));**

**PUNTO MEDIO COMPOSITA**

****

**IN MATLAB:**

**t = 5;**

**for indice = 1:t**

**n = 10^indice;**

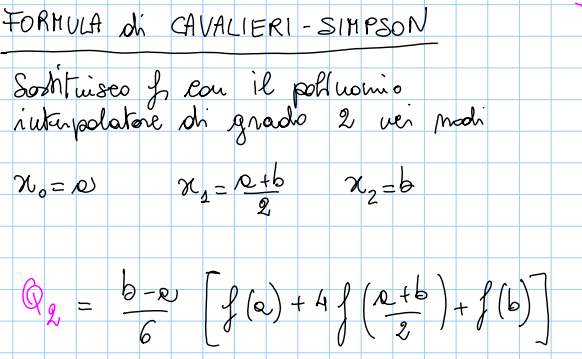
**nodi = linspace(a,b,n+1);**

**i = 1:n;**

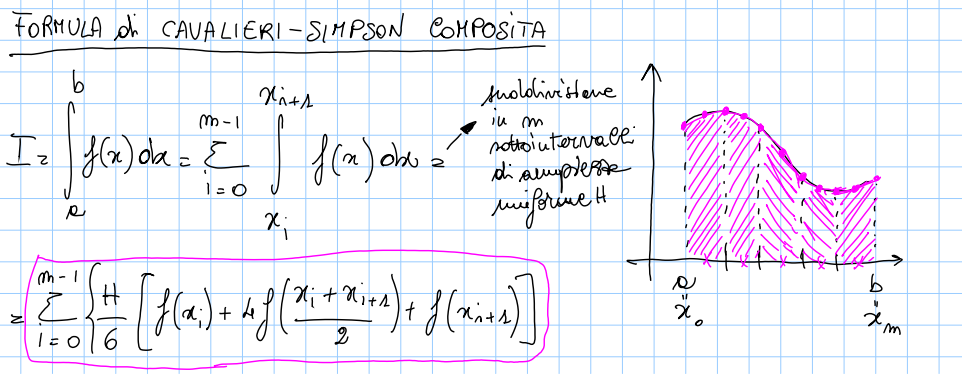
**%formula del punto medio composita**

**res = ((b-a)/n)\*sum(f((nodi(i)+nodi(i+1))/2));**

**CAVALIERI - SIMPSON**

****

**CAVALIERI - SIMPSON COMPOSITA**

****