* Kayan Noktalı Sayılar -> X=işaret.m.b(işaret)E ,(b=2 yani taban değeri(binary)) , +7x2-3  bir örnek
* Gerçek Değer -> A
* Mutlak Hata(Absolute Error) -> Et=|A – a|
* Bağıl Hata(Relative Error) -> et=|A-a| / |A|
* Yaklaşık Bağıl Hata ->

ea=|En iyi Tahmin – yaklaşık değer| / |En iyi tahmin|

* Scarborough kriteri-> ea < 0.5x10-mise sonuç m’nin en küçük basamağı için doğrudur.
* exp(x) -> ex, abs(x)-> mutlak değer

**Kodlar**

clear all; close all; clc

**% Hata yöntemlerinin gösterimi.**

x=1;

toplam=0;

pi=4\*atan(1);

for n=1:130

isaret=(-1)^(n+1);

pay=x^(2\*n-1);

payda=2\*n-1;

sontoplam=toplam+4\*isaret\*pay/payda;

truehata=abs(pi-sontoplam)/abs(pi);

yakhata=abs(sontoplam-toplam)/abs(sontoplam);

plot(n,yakhata,'--r\*',n,truehata,'--b+');

hold on

xlabel('n terim sayısı');

ylabel('hata');

toplam=sontoplam;

end

text(30,0.6,'+doğru bağıl hata');

text(30,0.5,'\*yaklaşık bağıl hata');

**Grafik Yöntemi**

clear all; close all; clc;

% f(x) = x^3 + 2\*x + 1 fonksiyonunun kökünü grafik ile bulma

for t=-2:0.1:2

ft=t^3+2\*t+1;

plot(t,ft,"b\*");

hold on;

end

grid on;

xlabel("t(sec)");

ylabel("ft");

**Newton Raphson yöntemi**

clear all; close all; clc;

% Newton Raphson yöntemi, f(x)=sqrt(x)+ln(x)-2\*sin(x/2) denkleminin kök bul

x0=1.1;

tolerans=1.0E-6;

for i=1:100

fx0=sqrt(x0)+log(x0)-2\*sin(x0/2);

fdx0=1/(2\*sqrt(x0))+1/x0-cos(x0/2); %sqrt(x)+ln(x)-2\*sin(x/2)’in türevi

x1=x0-fx0/fdx0;

fprintf("%4.1f %7.4f %7.4f %7.4f \n",i,x0,x1,abs(x1-x0));

if abs(x1-x0)<tolerans

break;

end

x0=x1;

end

disp("Yaklaşık Kök= ");

x0

**Bisection - İkiye Bölme Yöntemi**

clear all; close all; clc;

% Bisection kullanarak f(x)=x^3-4 denkleminin köklerini bulma

a=-1;

b=2;

tolerans=1E-6;

for i=1:100

fonka=a^3-4;

fonkb=b^3-4;

xm=0.5\*(a+b);

fonkxm=xm^3-4;

if fonka\*fonkxm < 0

b=xm;

else

a=xm;

end

if abs(a-b) < tolerans

break

end

end

disp("iterasyon sayısı");

i

disp("denklemin kökü");

xm

disp("Fonksiyonun kökdeki değeri");

fonkxm

**Hepsi Aynı Yöntem: Regula Falsi – False Position – Yer Değiştirme – Doğrusal Interpolasyon**

clear all; close all; clc;

% Not: Bu metoda 2 farklı şekilde son verilebilir

% 1) f(c)=0 olunca. Kök c dir. xr=c, c=xr demek

% 2) |Eb|<e ise işleme son verilir.

% 2a) Eb=(sondeger - biröncekideger)/sondeger= (x1-x0)/x1

% Regula Falsi kullanarak f(x)=x^3-2 denkleminin köklerini bulma

a=1;

b=2;

tolerans=0.00001;

fprintf('\n iter a b xr y(xr)\n');

for i=1:50

fonka=a^3-2;

fonkb=b^3-2;

xr=(a\*fonkb-b\*fonka)/(fonka-fonkb);

fonkxr=xr^3-2;

fprintf('%4.1f %7.4f %7.4f %7.4f %7.4f \n',i,a,b,xr,fonkxr);

if abs(fonkxr)< tolerans

break

end

if fonka\*fonkxr<0

b=xr;

fonkb=fonkxr;

else

a=xr;

fonka=fonkxr;

end

end

disp("iterasyon sayısı");

i

disp("denklem kökü");

xr

disp("Fonksiyonun kökdeki değeri");

fonkxr

**FİXED POİNT – Sabit Nokta iterasyonu**

clear all; close all; clc;

% Fixed Point kullanılarak f(x)=x-2^(-x) denkleminin köklerini bulma

x1=0;

tolerans=0.1;

for x1=0:0.01:1

y=x1;

yy=2^(-x1);

plot(x1,y,'r\*',x1,yy,'b.');

hold on;

grid on;

xlabel('x');

ylabel('y');

text(0.5,0.8,'\*y=x');

text(0.5,0.75,'+y=2^(-x)');

end

x1=0;

fprintf("Iter x1 x2 Ea ear \n");

for i=1:50

x2=2^(-x1);

Ea=abs(x2-x1);

ear=Ea/abs(x2);

fprintf("%4.1f %7.4f %7.4f %7.4f %7.4f \n",i,x1,x2,Ea,ear);

if abs(x2-x1)<tolerans

break;

else

x1=x2;

end

end

disp("Denklemin Kökü");

disp([x2]);

**Secant – kiriş yöntemi**

clear all; close all; clc;

% Secant yöntemi ile f(x)=x-0.17/sqrt(15/(x^0.3)+2) denk. kök bulma

x1=1.0;

x0=5.0;

tolerans=1.0E-5;

fprintf("Iter x2 abs(x2-x1) \n");

for i=1:100

fx1=x1-0.17/sqrt(15/(x1^0.3)+2);

fx0=x0-0.17/sqrt(15/(x0^0.3)+2);

x2=x1-(fx1\*(x1-x0))/(fx1-fx0);

fprintf("%4.1f %7.4f %7.4f \n",i,x2,abs(x2-x1));

if abs(x2-x1)<tolerans

break;

end

x0=x1;

x1=x2;

end

disp("Kök= ");

x2

**Cramer Yöntemi**

clear all; close all; clc;

% Cramer Yöntemi ile,

% 2x + y + z = 3

% x - y - z = 0

% x + 2y + z = 0

% denklemlerinin köklerini bulunuz

% Ax matrisini oluştururken, ilk sütuna; Ay matrisi için 2. sütuna,

% Az için 3. sütuna 3 0 0 değerlerini ekleyerek düzenliyoruz

% Ax(Matrisi) = | 3 1 1 | Ay(Matrisi) = | 2 3 1 |

% | 0 -1 -1 | | 1 0 -1 |

% | 0 2 1 | | 1 0 1 |

b = [3 ; 0 ; 0];

A = [2 1 1 ; 1 -1 -1 ; 1 2 1];

Ax = [3 1 1 ; 0 -1 -1 ; 0 2 1];

Ay = [2 3 1 ; 1 0 -1 ; 1 0 1];

Az = [2 1 3 ; 1 -1 0 ; 1 2 0];

x=det(Ax)/det(A);

y=det(Ay)/det(A);

z=det(Az)/det(A);

**Gauss Elimination**

clear all; close all; clc;

%Gauss Elimination ile 2x+8y+2z=14,x+6y-z=13,2x-y+2z=5 doğr. denk. kökleri

A=[2 8 2 ; 1 6 -1 ; 2 -1 2];

b=[14;13;5];

[n,~]=size(A);

x=zeros(n,1);

A

for i=1:n-1

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

A(i+1:n,:)= A(i+1:n,:)-m\*A(i,:);

b(i+1:n,:)= b(i+1:n,:)-m\*b(i,:);

end

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

for i=n-1:-1:1

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

end

x

**Jacobi İterasyon Yöntemi**

clear all; close all; clc;

% Jacobi İterasyon Yöntemi ile x^2-2x-y=0.5 ve x^2+4y^2=4 denk. kökleri

% Birinci denklemde x yanlız bırakılacak

% İkinci denklemde y yanlız bırakılacak

% yani

% x^2-2x-y=0.5 == (x^2-y+0.5)/2

% x^2+4y^2=4 == sqrt((4-x^2)/4)

x0=0;

y0=0;

E=1.0E-4;

fprintf("i x1 y1 errorX errorY \n");

for i=1:100

x1 = (x0^2-y0+0.5)/2;

y1 = sqrt((4-x0^2)/4);

errorx=abs(x1-x0);

errory=abs(y1-y0);

fprintf("%4.1f %7.4f %7.4f %7.4f %7.4f\n",i,x1,y1,errorx,errory);

if errorx<E & errory<E

break;

else

x0=x1;

y0=y1;

end

end

disp("Denklemin Kökleri: ")

disp([x1,y1]);

**Gauss Seidel Yöntemi**

clear all; close all; clc;

% Gauss Seidel, 3x-0.1y-0.2z=7.85, 0.1x+7y-0.3z=-19.3, 0.3x-0.2y+10z=71.4

% Köklerini bulunuz. Bitirme Şartı x in hataoranı(errorx)<0.0.1 ise

% Birinci denklemde x yanlız bırak

% İkinci denklemde y yanlız bırak

% Üçüncü denklemde z yanlız bırak

i=1;

y(i)=0;

z(i)=0;

x(i)=0;

errorx=9999;

while errorx(i)>=0.01

x(i+1)=(7.85+0.1\*y(i)+0.2\*z(i))/3;

y(i+1)=(-19.3-0.1\*x(i+1)+0.3\*z(i))/7;

z(i+1)=(71.4-0.3\*x(i+1)+0.2\*y(i+1))/10;

errorx(i+1)=abs(x(i+1)-x(i))/x(i+1)\*100;

errory(i+1)=abs(y(i+1)-y(i))/y(i+1)\*100;

errorz(i+1)=abs(z(i+1)-z(i))/z(i+1)\*100;

i=i+1;

end

disp(" x error(%)");

disp([x',errorx']);

disp(" y error(%)");

disp([y',errory']);

disp(" z error(%)");

disp([z',errorz']);

**Matris**

clear all; close all; clc;format("long","g");

Matris=[ 4 -2 6 ; 1 8 4 ; -3 -1 5 ];

transpoz=Matris';

determinant=det(Matris);

tersi=inv(Matris);

transpoz

determinant

tersi

**LU Ayrışım Yöntemi**

clear all; close all; clc;format("long","g");

%LU Ayrışımı Yöntemi

A=[ 3 -0.1 -0.2 ; 0.1 7 -0.3 ; 0.3 -0.2 10];

b=[7.85; -19.3 ; 71.4];

[L U]=lu(A);

d=L\b;

x=U\d;

A

b

L

U

**Newton İnterpolasyonu**

clear all; close all; clc;

x=[3 1 5 6];

y=[1 -3 2 4];

p=4

n = length(x);

a(1) = y(1);

for k = 1 : n - 1

d(k, 1) = (y(k+1) - y(k))/(x(k+1) - x(k));

end

for j = 2 : n - 1

for k = 1 : n - j

d(k, j) = (d(k+1, j - 1) - d(k, j - 1))/(x(k+j) - x(k));

end

end

d

for j = 2 : n

a(j) = d(1, j-1);

end

Df(1) = 1;

c(1) = a(1);

for j = 2 : n

Df(j)=(p - x(j-1)) .\* Df(j-1);

c(j) = a(j) .\* Df(j);

end

fp=sum(c);

fp

**Lagrange İnterpolasyonu**

clear all; close all; clc;

%Lagrange İnterpolasyonu

x=[2 3 5];

y=[5 7 8];

n=length(x);

f=zeros(n,n);

for i=1:n

L=1;

for j=1:n

if i~=j

L=conv(L,poly(x(j))/(x(i)-x(j)));

end

end

f(i,:)=L\*y(i);

end

f

P=sum(f)

**Sayısal Türev**

clear all; close all; clc;

% Sayısal Türev Alma

syms x t

diff(sin(2\*x\*t),t) % diff komutu sin(2xt)'nin t'ye göre türevini alır

**Sayısal Türev**

clear all; close all; clc;

% İleri, Geri, Merkezi Farklar

arr = [

1.0 0.7651977; %Xi0, Δy0

1.3 0.6200860; %Xi1, Δy1

1.6 0.4554022; %Xi2, Δy2

1.9 0.2818186; %Xi3, Δy3

2.2 0.1103623; %Xi4, Δy4

];

% İleri Farklar Yöntemi

for inx = 1:4 % Matlabda indexler 1 den başlar

y = (arr(inx+1,2) - arr(inx,2))/(arr(inx+1,1) - arr(inx,1));

fprintf('idy%d = %d \n\n',(inx-1),y);

end

% Geri Farklar Yöntemi

for inx = 2:5 % Matlabda indexler 1 den başlar

y = (arr(inx,2) - arr(inx-1,2))/(arr(inx,1) - arr(inx-1,1));

fprintf('gdy%d = %d \n\n',(inx-1),y);

end

% Merkezi Farklar Yöntemi

for inx = 2:4 % Matlabda indexler 1 den başlar

y = (arr(inx+1,2) - arr(inx-1,2))/(arr(inx+1,1) - arr(inx-1,1));

fprintf('mdy%d = %d \n\n',(inx-1),y);

end

**Sayısal İntegral**

clear all; close all; clc;

% Dikdörtgenler Yöntemi

% 1-8 integral (x+2)dx/(x^2+2) n=6 için

n=6;

x0=1;

xn=8;

h=(xn-x0)/n;

xk=x0;

arrFxk=[];

while xk<xn

fxdx =(xk+2)/(xk^2+2);

arrFxk(end+1) = fxdx;

xk=xk+h;

end

sonuc = h\*sum(arrFxk);

sonuc

clear all; close all; clc;

% Trapez ( Yamuk ) Yöntemi

% f(x)=(x^3+1) fonk. Simpson 1/3 ile çöz

% temel formül: (b-a)\*(f(a)+ f((a+b)/2)+ f(b))/6

a=-1;

b=3;

n=2;

h=(b-a)/n;

toplam=0;

for x0=a:h:b-h

x1=(x0+(x0+h))/2;

x2=x0+h;

fx0=(x0^3+1);

fx1=(x1^3+1);

fx2=(x2^3+1);

toplam=toplam + (h/6)\*(fx0+4\*fx1+fx2);

end

toplam

**Eğri Uydurma**

clear all; close all; clc;format long;

% Eğri Uydurma

x = [5 10 15 20 25 30 35 40 45 50];

y = [16 25 32 33 38 36 39 40 42 42];

a2=polyfit(x,y,2);% 2 . dereceden polinomum katsayıları bulunur

disp("a2 katsayıları"),disp(a2)

xi=linspace(5,50,101); % x aralığı

yi2=polyval(a2,xi); % polinomun aldığı değerler hesaplanır

plot(x,y,'ro','linewidth',2),hold on % datalar çizdirilir

plot(xi,yi2,'b','linewidth',2),grid on % uydurulan eğriler çizdirilir

xlabel('x'),ylabel('y'),title('y=-0.0155x^2+1.3458x+12.1667')