%% 2.1&&2.2

% x1=linspace(0,2\*pi,10);

% x2=linspace(0,2\*pi,100);

% title('graph of x from 0 to 2pi');

% xlabel('x-axis');

% ylabel('y-axis');

% plot(x1,cos(x1),'g-',x2,cos(x2),'r-.');

% legend('10a','100b');

% x=linspace(0,2\*pi,1000);

% y=sin(x);

% plot(x,y)

% clear all;

% close all;

% clc;

% x=linspace(0,2\*pi,10);

% y1=cos(x)/x;

% y2=cos(x)./x;

% a = linspace(1,50,50);

% tic

% sum(a)

% toc

%% Task1

n= 1:1:100;

% disp(n);

sum\_1=0;

tic;

for i=1:1:length(n)

sum\_1=sum\_1+n(i);

end

toc;

% disp(sum\_1);

tic;

S=sum(n);

toc;

if(S~=sum\_1)

disp("error");

end

% disp(S);

%% Task2

in="Please input the num:\n";

num=input(in);

n= rand(1,100) ;

% disp(n);

sum\_1=0;

tic;

for i=1:1:length(n)

sum\_1=sum\_1+n(i);

end

toc;

% disp(sum\_1);

tic;

S=sum(n);

toc;

% disp(S);

if(S~=sum\_1)

disp("error");

end

%% Task 3

prompt = "Please input a Number you want:\n";

a = input(prompt);

% b=((15-a^2)/2)^(1/3);

b=nthroot((15-a^2)/2,3);

flag=1;

num=0;

while(flag)

if(abs(a-b)>0.001)

a=b;

% b=((15-a^2)/2)^(1/3);

b=nthroot((15-a^2)/2,3);

num=num+1;

else

disp(b);

disp(num);

flag=0;

end

end

%% Task 4 Example on roots of quadratic equation

prompt="Please input a Number you want :\n";

a=input(prompt);

b=input(prompt);

c=input(prompt);

d=b^2-4\*a\*c;

if(d<0)

disp("no real roots");

elseif (d==0)

disp("equal roots x="+(-b/(2\*a)));

else

x1=(-b+d^(1/2))/(2\*a);

x2=(-b-d^(1/2))/(2\*a);

disp("roots are x1="+x1+" x2="+x2);

end

%% Task 5 better\_plot

x=linspace(0,2\*pi);

y=sin(x);

figure(1);

better\_plot(x,y,'',5,'x','y',24,1);

figure(2)

better\_plot(x,y,'-\*r',7,'x','sin(x)',18,0);

% fontsize(10,"points");

%% Task 6

% n=input("Please input n:\n");

% x=input("Please input x:\n");

n=100;

x=linspace(0,2\*pi,100);

f=zeros(1,100);

num\_error\_n=zeros(1,100);

for i=1:100

f(i)=my\_sin(n,x(i));

num\_error\_n(i)=abs(sin(x(i))-f(i));%n stable

disp("i="+i+" f="+f(i)+" sin="+sin(x(i))+" error="+num\_error\_n(i));

end

semilogy(num\_error\_n);

figure(1);

hold on;

num\_error\_x=zeros(1,1000);

x=linspace(0,2\*pi,100);

f=zeros(1,1000);

for n=1:1000

f(n)=my\_sin(n,pi/4);

num\_error\_x(n)=abs(sin(pi/4)-f(n));

disp("i="+n+" f="+f(n)+" sin="+sin(pi/4)+" error="+num\_error\_x(n));

end

figure(2);

semilogy(num\_error\_x);

%% Task 7

A=randi(10,6);

A1=diag(diag(A));

disp(A);

disp(A1);

A=zeros(3,2);

A=ones(2,4);

A=[1 -5 7;4 8 9;-3 0 2];

B=[1 -1 3;2 4 16;0 0 2];

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

%% Task 8 Finding roots (Bisection, MATLAB fzero and Newton s method)

% x=linspace(0,10,1000);

x=-2:0.1:2;

y=@(x) x.^4-2\*x-2;

figure(1);

plot(x,x.^4-2\*x-2);

grid on;

[x1,e1]=mybisect(y,-1,0,5);

[x2,e2]=mybisect(y,1,2,5);

disp("x1="+x1+" e1="+e1);

disp("x2="+x2+" e2="+e2);

% x2=zeros(1,50);

% e2=zeros(1,50);

% for k=1:50

% [x2(k),e2(k)]=mybisect(y,1,2,k);

% end

% n=1:50;

% figure(2);

% plot(n,e2);

% grid on;

%% Task 9 fzero

clear;

x=-2:0.1:2;

y=@(x) x.^4-2\*x-2;

figure(1);

grid on;

plot(x,x.^4-2\*x-2);

xzero=[];

k=0;

for i=1:length(x)

fzero(y,x(i));

end

zero1\_fz=fzero(y,-1);

zero2\_fz=fzero(y,1);

disp(zero1\_fz);

disp(zero2\_fz);

%% Task 10 Newton’s method

clear;

tol=1.0e-6;

f=@(x) x.^3+4\*x.^2-7\*x-20;

f1=@(x) 3\*x.^2+8\*x-7;

x=-5:0.1:5;

grid on;

plot(x,x.^3+4\*x.^2-7\*x-20);

figure(1);

x0=linspace(-6,6,1000);

x1=zeros(1,1000);

for i=1:length(x0)

[x1(i),~]=mynewtontol(f,f1,x0(i),tol);

end

x1\_round=round(x1,6);

x1\_final=unique(x1\_round);

disp(x1\_final);

%% Task 11 incremental search algorithm

clear;

tol=1.0e-6;

f=@(x) x.^3+4\*x.^2-7\*x-20;

f1=@(x) 3\*x.^2+8\*x-7;

left=-6;

right=6;

x=-6:0.1:6;

grid on;

plot(x,x.^3+4\*x.^2-7\*x-20);

figure(1);

x0=linspace(left,right,13);

zero=[];

for i=1:length(x0)-1

disp([f(x0(i)),f(x0(i+1)),x0(i),x0(i+1)]);

if(f(x0(i))\*f(x0(i+1))<0)

x\_mid=(x0(i)+x0(i+1))/2;

[x1,~]=mynewtontol(f,f1,x\_mid,tol);

zero=[zero,x1];

end

end

disp(zero);

%% Task 12

clear;

tol=1.0e-6;

f=@(x) x.^2-4\*x+4;

f1=@(x) 2\*x+4;

left=-6;

right=6;

x=-6:0.1:6;

figure(1);

plot(x, x.^2-4\*x+4);

grid on;

x0=linspace(left,right,13);

zero=[];

for i=1:length(x0)-1

% disp([f(x0(i)),f(x0(i+1)),x0(i),x0(i+1)]);

if(f(x0(i))\*f(x0(i+1))<=0)

x\_mid=(x0(i)+x0(i+1))/2;

[x1,~]=mynewtontol(f,f1,x\_mid,tol);

zero=[zero,x1];

end

end

disp(zero);

% function y=myfunction(x)

% y=x.^4-2\*x-2;

% end

function [x,e] = mybisect(f,a,b,n)

% function [x e] = mybisect(f,a,b,n)

% Does n iterations of the bisection method for a function f

% Inputs: f -- an inline function

% a,b -- left and right edges of the interval

% n -- the number of bisections to do.

% Outputs: x -- the estimated solution of f(x) = 0

% e -- an upper bound on the error

format long

c = f(a);

d = f(b);

if (c\*d > 0.0)

error('Function has same sign at both endpoints.')

end

disp(' x y ')

for i = 1:n

x = (a + b)/2;

y = f(x);

disp ([ x y])

if (y == 0.0) % solved the equation exactly

e = 0;

break % jumps out of the for loop

end

if (c\*y) < 0

b=x;

else

a=x;

end

end

x = (a + b)/2;

e = (b-a)/2;

end

function x = mynewton(f,f1,x0,n,tol)

% Solves f(x) = 0 by doing n steps of Newton’s method starting at x0.

% Inputs: f -- the function, input as an inline

% f1 -- it’s derivative, input as an inline

% x0 -- starting guess, a number

% tol -- desired tolerance, prints a warning if |f(x)|>tol

% Output: x -- the approximate solution

x = x0; % set x equal to the initial guess

for i=1:n % Do n times

x = x - f(x)/f1(x); % Newton;s formula

end

r = abs(f(x));

if (r > tol)

warning("The desired accuracy was not attained")

end

end

function [x no\_iterations]= mynewtontol(f,f1,x0,tol)

x = x0; % set x equal to the initial guess x0.

i=0; % set counter to zero

y = f(x);

while (abs(y) > tol && i < 1000)

% Do until the tolerance is reached or max iter.

x = x - y/f1(x); % Newton’s formula

y = f(x);

i = i+1;

end

no\_iterations=i;

end

function result=my\_sin(n,x)

f =x;

for i=1:1:n

myfct=(my\_factorial(2\*i+1));

num=(((-1)^i)\*((x^(2\*i-1))/myfct));

f= f+num;

end

result=double(f);

end

function myfct=my\_factorial(n)

% Establish a function to evaluate n!

f=1;

for i=1:n

f=(f\*i);

end

myfct=f;

end

function z=better\_plot(x,y,LineSpec,LineThickness,xlab,ylab,FontSize,enablegrid)

plot(x,y,LineSpec,"LineWidth",LineThickness);

xlabel(xlab);

ylabel(ylab);

if(enablegrid)

grid on;

else

grid off;

end

ax1 = gca;

ax1.FontSize=FontSize;

z=1;

end