%Write and execute the MATLAB code to solve 𝑑𝑦/𝑑𝑥= 3𝑥 + 𝑦/2, (0) = 1 by Runge Kutta method for 𝑦(0.5) taking h=0.1. Display output for each value of x upto 0.5.

clc;

f=@(x,y)3.\*x+(y./2);

x=0;

y=1;

h=0.1;

xn=0.5;

n=(xn-x)/h;

for i=1:n

fprintf("Iteration number =%f\n",i)

disp(" x y ")

z=[x y];

disp(z)

k1=h\*f(x,y)

k2=h\*f(x+h/2,y+k1/2)

k3=h\*f(x+h/2,y+k2/2)

k4=h\*f(x+h,y+k3)

y=y+(k1+2\*k2+2\*k3+k4)/6

x=x+h;

end

% Write and execute the MATLAB code to show that the divergence of the curl of the vector

% field V=(x, 2 y^2, 3 z^3) is 0

syms x y z

V=[x 2\*y^2 3\*z^3];

X=[x y z];

divCurl=divergence(curl(V,X),X)

% Write and execute the MATLAB code to compute the volume of the sphere lying in the

%positive octant with radius a.

clear all;

clc;

syms x y a

f=sqrt(a^2-x^2-y^2);

V=int(int(f,y,0,sqrt(a^2-x^2)),x,0,a);

fprintf('the required volume is given by')

V

%Write and execute the MATLAB code to evaluate ∫ √sin (x)

% 𝜋⁄2 0 dx with 6 subintervals by using Simpson s 3/8th Rule

clc;

f=@(x)(sqrt(sin(x)));

a = input('enter the lower limit a:');

b = input('enter the upper limit b:');

n = input('enter the number sub interval n:');

h=(b-a)/n;

if rem(n,2)==1

fprintf('\n enter the valid n!!');

n = input('\n enter the n as even number:');

end

K=1:1:n-1;

s=f(a+K.\*h);

out = (3\*h/8).\*(f(a)+f(b)+3.\*(s(1)+s(2)+s(4)+s(5))+2.\*(s(3)));

fprintf('the value of integration is %.4f',out);

% Write and execute the MATLAB code to evaluate ∫ √(sin(x) + cos(x))

% 1 0 dx with 6 sub intervals by using Simpson%s 1/3rd Rule

clc;

clear all;

f=@(x)(sqrt(sin(x)+(cos(x))));

a=input('enter the lower limit a:');

b=input('enter the lower limit b:');

n=input('enter the number of sub intervals n:');

h=(b-a)/n;

if rem(n,2)==1

fprintf('\n please enter valid number n!!');

n=input('\n enter n as even number:');

end

K=1:1:n-1;

s=f(a+K.\*h);

out=(h/3).\*(f(a)+f(b)+4.\*(s(1)+s(3)+s(5))+2.\*(s(2)+s(4)));

fprintf('the value of the integration is%.4f',out)

% Write and execute the MATLAB code that uses Newton's forward interpolation formula and

% find the approximate value of f (9) given the following data points (x,y) = (8,10), (10,19),

% (12,32.5), (14,54), (16,89.5) & (18,154).

clc;

clear all;

x=[8,10,12,14,16,18];

y=[10,19,32.5,54,89.5,154];

n=length(x);

X=9;

h=x(2)-x(1);

F=zeros(n,n);

F(:,1)=y;

for j =2:n

for i=j:n

F(i,j)=F(i,j-1)-F(i-1,j-1);

end

end

F

p=(X-x(1))/h;

d=F(1,1)+p\*F(2,2)+p\*(p-1)\*F(3,3)/factorial(2)+p\*(p-1)\*(p-2)\*F(4,4)/factorial(3)+p\*(p-1)\*(p-2)\*(p-3)\*F(5,5)/factorial(4)+p\*(p-1)\*(p-2)\*(p-3)\*(p-4)\*F(6,6)/factorial(5);

fprintf('f(%0.4f)=%0.4f\n',X,d);

% . Write and execute the MATLAB code that uses Newton's backward interpolation formula and

% find the approximate value of f(17) given the following data points (x,y) = (8,10), (10,19),

% (12,32.5), (14,54), (16,89.5) & (18,154)

clc;

clear all;

x=[8,10,12,14,16,18];

y=[10,19,32.5,54,89.5,154];

n=length(x);

X=17;

h=x(2)-x(1);

F=zeros(n,n);

F(:,1)=y;

for j =2:n

for i=j:n

F(i,j)=F(i,j-1)-F(i-1,j-1);

end

end

F

p=(X-x(n))/h;

d=F(6,1)+p\*F(6,2)+p\*(p+1)\*F(6,3)/factorial(2)+p\*(p+1)\*(p+2)\*F(6,4)/factorial(3)+p\*(p+1)\*(p+2)\*(p+3)\*F(6,5)/factorial(4)+p\*(p+1)\*(p+2)\*(p+3)\*(p+4)\*F(6,6)/factorial(5);

fprintf('f(%0.4f)=%0.4f\n',X,d);

% a) Write and execute the MATLAB code to evaluate ∫ ∫ xe−x^2/y dx/dy

clear all;

clc;

syms x y

f= x\*exp(-x^2/y);

fprintf('the value of the given double integral is');

int(int(f,x,y,inf),y,0,inf)

% Write and execute the MATLAB code to find the dimension a n d b a s i s of subspace spanned

% by the vectors (1, 2, 1, -1),(3,1,0,5) and (0,5,3,-8).

clear all;

clc;

v1=[1;2;1;-1];v2=[3;1;0;5];v3=[0;5;3;-8];

v=[v1 v2 v3];

[r,basiccol]=rref(v);

r\_1=rank(rref(v));

fprintf('dimension of subspace spanned by the given vectors is given by');

r\_1

fprintf('basis of subspace is given by');

b=v(:,basiccol)

%. Write and execute the MATLAB code to verify the rank-nullity theorem for the linear

% transformation T: R3 → R3 defined by T(x,y,z) = (x + 4y + 7z,2x + 5y + 8z,3x + 6y + 9z).

clear all;

clc;

fprintf('matrix of linear transformatransformation is given by');

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

rref(A);

fprintf('echelon form of A is given by');

rref(A)

R=rank(rref(A));

fprintf('rank of linear transformation is given by');

R

null(A);

fprintf('null space is given by');

null(A)

N=size(null(A),2);

fprintf('dimension of null space is given by');

N

if R+N==3

fprintf('rank nullity theorem holds');

else

fprintf('rank nullity theorem holds');

end

% Write and execute the MATLAB code to find the gradient vector of f(x,y)=xe^(x^2-y^2)

% with respect to vector [x,y] and also plot the contour lines and vectors

x=-2:0.2:2;

y=x';

z=x.\*exp(-x.^2-y.^2);

[px,py]=gradient(z);

figure

contour(x,y,z)

hold on

quiver(x,y,px,py)

hold off

% Write and execute the MATLAB code to find the root of y=cos(x) near 1 with tolerance

% 0.0001. Carry out four iterations by Newton- Raphson method

f=@(x)(cos(x));

fd=@(x)(-sin(x));

x0=input('Enter initial approximation :');

n=input('Enter no of iteration,n: ');

tol=input('Enter tolerance,tol :');

i=1;

while i<=n

d=f(x0)/fd(x0);

x0=x0-d

if abs(d)<tol

fprintf('\n Approximate solution xn= %0.4f \n\n',x0)

break;

else

i=i+1;

end

end

%.. Write and execute the MATLAB code to find the root of y=sin(x)+cos(x)+exp(x)-8 at 2 and

% 3 with tolerable error 0.00001 by Regula-Falsi method

clc

syms x;

y = input('Enter non-linear equations: ');

a = input('Enter first guess: ');

b = input('Enter second guess: ');

e =0.00001;

fa = eval(subs(y,x,a));

fb = eval(subs(y,x,b));

if fa\*fb > 0

disp('Given initial values do not bracket the root.');

else

c = a - (a-b) \* fa/(fa-fb);

fc = eval(subs(y,x,c));

fprintf('\n\na\t\t\tb\t\t\tc\t\t\tf(c)\n');

while abs(fc)>e

fprintf('%f\t%f\t%f\t%f\n',a,b,c,fc);

if fa\*fc< 0

b =c;

fb = eval(subs(y,x,b));

else

a =c;

fa = eval(subs(y,x,a));

end

c = a - (a-b) \* fa/(fa-fb);

fc = eval(subs(y,x,c));

end

fprintf('\nRoot is: %f\n', c);

end

%Write and execute the MATLAB code to solve 𝑑𝑦/𝑑𝑥= 𝑙𝑜𝑔(𝑥 + 𝑦), 𝑦(1) = 2 by Modified

%Euler%s method for (1.4)taking h=0.1. Display output for each value of x upto 1.4. Perform

% 4 modifications at every step.

clear all

clc

f=@(x,y)log(x+y);

x0=1;

y0=2;

h=0.1;

xn=1.4;

n=(xn-x0)/h;

for i=1:n+1

fprintf("Iteration number=% f\n",i)

y1E=y0+h\*f(x0,y0)

for j=1:3

y1=y0+(h/2)\*(f(x0,y0)+f(x0+h,y1E))

y1E=y1;

j=j+1;

end

y0=y1;

x0=x0+h;

n=n+1;

end

x=[x0(:)];

y=[y0(:)];

disp(" x y")

z=[x y];

disp(z)