**QUESTÃO 9:**

Resolução:

#include <stdio.h>

#include <stdlib.h>

/\*

C = 0.01 farad;

e0 = 100 volts;

L = 0.02 henry;

R = 10 ohms;

\*/

float xR[10], yR[10], zR[10];

float t;

int n;

float funcF(float x, float y, float z){

float F;

F = -500\*x - 5000\*z + 5000;

return F;

}

float funcJ(float x, float y, float z){

float J;

J = -5\*y + 5000\*z;

return J;

}

float funcH(float x, float y, float z){

float H;

H = x-y;

return H;

}

void metodoEuler(float x0, float y0, float z0, float t, float n){

int i;

float h;

xR[1] = x0;

yR[1] = y0;

zR[1] = z0;

h = t/n;

printf("x[1]: %0.4f, y[1]: %0.4f, z[1]: %0.4f\n", xR[1],yR[1],zR[1]);

for(i=2;i<n;i++){

xR[i] = xR[i-1] + (h\*funcF(xR[i-1], yR[i-1], zR[i-1]));

yR[i] = yR[i-1] + (h\*funcJ(xR[i-1], yR[i-1], zR[i-1]));

zR[i] = zR[i-1] + (h\*funcH(xR[i-1], yR[i-1], zR[i-1]));

printf("x[%d]: %0.4f, y[%d]: %0.4f, z[%d]: %0.4f\n", i,xR[i],i,yR[i],i,zR[i]);

}

printf("\n");

}

void metodoBashforth(float x0, float y0, float z0, float t, float n){

int i;

float h;

xR[1] = x0;

yR[1] = y0;

zR[1] = z0;

h = t/n;

xR[2] = x0 + h\*funcF(x0,y0,z0);

yR[2] = y0 + h\*funcJ(x0,y0,z0);

zR[2] = z0 + h\*funcH(x0,y0,z0);

printf("x[1]: %0.4f, y[1]: %0.4f, z[1]: %0.4f\n", xR[1],yR[1],zR[1]);

printf("x[2]: %0.4f, y[2]: %0.4f, z[2]: %0.4f\n", xR[2],yR[2],zR[2]);

for(i=3;i<n;i++){

xR[i] = xR[i-1] + ((h/2)\*((3\*(funcF(xR[i-1],yR[i-1],zR[i-1]))) - funcF(xR[i-2],yR[i-2],zR[i-2])));

yR[i] = yR[i-1] + ((h/2)\*((3\*(funcJ(xR[i-1],yR[i-1],zR[i-1]))) - funcJ(xR[i-2],yR[i-2],zR[i-2])));

zR[i] = zR[i-1] + ((h/2)\*((3\*(funcH(xR[i-1],yR[i-1],zR[i-1]))) - funcH(xR[i-2],yR[i-2],zR[i-2])));

printf("x[%d]: %0.4f, y[%d]: %0.4f, z[%d]: %0.4f\n", i,xR[i],i,yR[i],i,zR[i]);

}

printf("\n");

}

void metodoRungeKutta(float x0, float y0, float z0, float t, float n){

int i;

float h;

float m1, k1, r1;

float m2, k2, r2;

float m3, k3, r3;

float m4, k4, r4;

xR[1] = x0;

yR[1] = y0;

zR[1] = z0;

h = t/n;

printf("x[1]: %0.4f, y[1]: %0.4f, z[1]: %0.4f\n", xR[1], yR[1], zR[1]);

for(i=2;i<n;i++){

m1 = funcF(xR[i-1], yR[i-1], zR[i-1]);

k1 = funcJ(xR[i-1], yR[i-1], zR[i-1]);

r1 = funcH(xR[i-1], yR[i-1], zR[i-1]);

m2 = funcF((xR[i-1]+(h/2)\*m1), (yR[i-1]+(h/2)\*k1), (zR[i-1]+(h/2)\*r1));

k2 = funcJ((xR[i-1]+(h/2)\*m1), (yR[i-1]+(h/2)\*k1), (zR[i-1]+(h/2)\*r1));

r2 = funcH((xR[i-1]+(h/2)\*m1), (yR[i-1]+(h/2)\*k1), (zR[i-1]+(h/2)\*r1));

m3 = funcF((xR[i-1]+(h/2)\*m2), (yR[i-1]+(h/2)\*k2), (zR[i-1]+(h/2)\*r2));

k3 = funcJ((xR[i-1]+(h/2)\*m2), (yR[i-1]+(h/2)\*k2), (zR[i-1]+(h/2)\*r2));

r3 = funcH((xR[i-1]+(h/2)\*m2), (yR[i-1]+(h/2)\*k2), (zR[i-1]+(h/2)\*r2));

m4 = funcF(xR[i-1] + h\*m3, yR[i-1] + h \* k3, zR[i-1] + h\*r3);

k4 = funcJ(xR[i-1] + h\*m3, yR[i-1] + h \* k3, zR[i-1] + h\*r3);

r4 = funcH(xR[i-1] + h\*m3, yR[i-1] + h \* k3, zR[i-1] + h\*r3);

xR[i] = xR[i-1] + (h/6)\*(m1 + 2\*(m2 + m3) + m4);

yR[i] = yR[i-1] + (h/6)\*(k1 + 2\*(k2 + k3) + k4);

zR[i] = zR[i-1] + (h/6)\*(r1 + 2\*(r2 + r3) + r4);

printf("x[%d]: %0.4f, y[%d]: %0.4f, z[%d]: %0.4f\n", i,xR[i],i,yR[i],i,zR[i]);

}

printf("\n");

}

int main(int argc, char \*argv[]){

int op, i;

//Sendo X(0)=e0/2R . [1 1 0]T

xR[0] = 5;

yR[0] = 5;

zR[0] = 0;

t = 0.008;

n = 10;

printf("---------------------------------------\n");

printf("Menu: \n1-Metodo Euler\n2-Metodo Bashforth\n");

printf("3-Metodo Runge Kutta 4\n4-Resolucao\n5-Sair\n");

printf("---------------------------------------\n");

while(op!=4){

printf("Digite sua opcao: ");

scanf("%d",&op);

switch(op){

case 1: printf("\n---------------------------------------\n");

printf(" METODO EULER \n\n");

metodoEuler(xR[0],yR[0],zR[0],t,n);

printf("\n---------------------------------------\n");

break;

case 2: printf("\n---------------------------------------\n");

printf(" METODO BASHFORTH\n\n");

metodoBashforth(xR[0],yR[0],zR[0],t,n);

printf("\n---------------------------------------\n");

break;

case 3: printf("\n---------------------------------------\n");

printf(" METODO RUNGE-KUTTA 4 ORDEM\n\n");

metodoRungeKutta(xR[0],yR[0],zR[0],t,n);

printf("\n---------------------------------------\n");

break;

default:break;

}

}

printf("\nSaindo...\n\n\n");

system("PAUSE");

return 0;

}

Obtemos:

