Leonardo Rios E.

201112402

**ACELERADORES DE PARTICULAS Y SUS APLICACIONES**

**SIMULACION DE UN CICLOTRÓN**

Para la simulación de un acelerador ciclotrón escogí el método Runge-Kutta que resuelve las ecuaciones de movimiento radial dadas por:m\*a=q\*v\*B, donde ‘a’ es la segunda derivada de la posición con respecto al tiempo.

El método Runge-Kutta puede resolver ecuaciones de primer orden por lo que se debe pasar de un sistema de una ecuación de segundo orden a dos ecuaciones acopladas de primer orden (los comentarios en el código indican el proceso).

El código para la solución de las ecuaciones está hecho en C junto con las consideraciones de la fuerza eléctrica. Se utiliza un intervalo de tiempo ‘h’ para resolver numéricamente la ecuación ‘n\_points’-veces.

Para graficar utilizo una tabla exportada desde el código en C que un código en Python recibe como parámetro, el programa grafica en coordenadas cartesianas la trayectoria de la partícula en los diferentes puntos en que se resolvió la ecuación diferencial de movimiento.

Los resultados son los siguientes:

**Trayectoria de una partícula con posición inicial mayor a 0.**

**Trayectoria de una partícula con posición inicial menos a 0.**

**Trayectoria de una partícula con posición inicial 0.**

**Trayectoria de una partícula con velocidad inicial en ausencia de campo eléctrico.**

**Simulador en C**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <math.h>**

**#define q 1.602**

**#define m 1.672**

**#define B 0.03**

**#define d 6**

**#define R 800**

**#define a 20**

*//a viene definido por el campo electrico, la carga y la masa F=qE => a=qE/m*

**int** **main**() {

*//Archivo donde se guardaran los datos de la trayectoria*

**FILE** **\***export;

export **=** fopen("trayectoria.dat", "w");

**double** h;

**int** n\_points;

h **=** 1;

n\_points **=** 200;

*//Variables para guardar la informacion necesaria*

**double** **\***t;

**double** **\***Vel;

**double** **\***Pos;

**double** **\***theta;

**double** **\***Costheta;

**double** **\***Sentheta;

**double** **\***Posx;

**double** **\***Posy;

Vel **=** malloc(n\_points**\*sizeof**(**double**));

Pos **=** malloc(n\_points**\*sizeof**(**double**));

t **=** malloc(n\_points**\*sizeof**(**double**));

theta **=** malloc(n\_points**\*sizeof**(**double**));

Costheta **=** malloc(n\_points**\*sizeof**(**double**));

Sentheta **=** malloc(n\_points**\*sizeof**(**double**));

Posx**=** malloc(n\_points**\*sizeof**(**double**));

Posy**=** malloc(n\_points**\*sizeof**(**double**));

**if** (**!**Vel **||** **!**Pos **||** **!**t){

    printf("Error en memoria");

    exit(1);

}

*//condiciones iniciales del problema*

Pos[0] **=** 1.0; *//Se puede ajustar segun se desee*

t[0] **=** 0.0;

theta[0]**=**0.0;

**if**(Pos[0]**>**0){

Costheta[0]**=**0.1; *//Valor aleatorio donde solo importa que sea mayor a 0*

}

**else** **if**(Pos[0]**<**0){

Costheta[0]**=-**0.1; *//Valor aleatorio donde solo importa que sea menor a 0*

}

Sentheta[0]**=**sin(theta[0]);

*//Se ajusta la velocidad inicial que depende de la posicion, la distancia de separacion entre las D y el a previamente definido*

**if**(Costheta[0]**>**0){

Vel[0]**=**sqrt(2**\***d**\***a);

}

**else** **if**(Costheta[0]**<**0){

Vel[0]**=-**sqrt(2**\***d**\***(a));

}

*//Se imprime la informacion de la trayectoria, el formato es: Tiempo, Velocidad, Radio, Coseno del angulo, Seno del angulo*

  fprintf(export,"%f %f %f %f %f \n", t[0],Vel[0],Pos[0],Costheta[0], Sentheta[0]);

**int** i;

**for** (i **=** 1; i **<** n\_points; i **++**)

    {

      rungekutta(i,Vel,Pos,t,h);

      theta[i]**=**10**\***i**/**(57.2956);

      Costheta[i]**=**cos(theta[i]);

      Sentheta[i]**=**sin(theta[i]);

      fprintf(export,"%f %f %f %f %f \n",t[i],Vel[i],Pos[i],Costheta[i], Sentheta[i]);

    }

FuerzaElectrica(**\***Costheta, **\***Sentheta, n\_points, **\***Vel);

**return** 0;

  }

*//Consideracion de la fuerza electrica para la aceleracion*

**int** **FuerzaElectrica**(**double** **\***Costheta, **double** **\***Sentheta, **int** n\_points, **double** **\***Vel){

**int** k;

**for**(k**=**0; k**<**n\_points; k**++**){

**if**(Costheta[k]**>**0){

**if**(Sentheta[k]**<**0.0001){

Vel[k]**=**sqrt(Vel[k**-**1]**\***Vel[k**-**1]**+**2**\***d**\***a);

}

}

**else** **if**(Costheta[k]**<**0){

**if**(Sentheta[k]**<**0.0001){

Vel[k]**=**sqrt(Vel[k**-**1]**\***Vel[k**-**1]**+**2**\***d**\***(**-**a));

}

}

}

**return** 0;

}

*//Funciones que rigen el movimiento radial*

**double** **v\_prime**( **double** v, **double** Vel) {

**return** (q**\***v**\***B)**/**(m);

}

**double** **r\_prime**( **double** v, **double** Vel) {

**return** v;

}

*//Metodo rungekutta*

**int** **rungekutta**(**int** i, **double** **\***Vel,**double** **\***Pos,**double** **\***t, **double** h)

{

**double** kx1 **=** v\_prime(Vel[i**-**1],Pos[i**-**1]);

**double** ky1 **=** r\_prime(Vel[i**-**1],Pos[i**-**1]);

*// Paso1*

**double** t1 **=** t[i**-**1] **+** (h**/**2.0);

**double** Vel1 **=** Vel[i**-**1] **+** (h**/**2.0) **\*** kx1;

**double** Pos1 **=** Pos[i**-**1] **+** (h**/**2.0) **\*** ky1;

**double** kx2 **=** v\_prime(Vel1, Pos1);

**double** ky2 **=** r\_prime(Vel1, Pos1);

*// Paso2*

**double** t2 **=** t[i**-**1] **+** (h**/**2.0);

**double** Vel2 **=** Vel[i**-**1] **+** (h**/**2.0) **\*** kx2;

**double** Pos2 **=** Pos[i**-**1] **+** (h**/**2.0) **\*** ky2;

**double** kx3 **=** v\_prime(Vel2, Pos2);

**double** ky3 **=** r\_prime(Vel2, Pos2);

*// Paso3*

**double** t3 **=** t[i**-**1] **+** h;

**double** Vel3 **=** Vel[i**-**1] **+** h **\*** kx3;

**double** Pos3 **=** Pos[i**-**1] **+** h **\*** ky3;

**double** kx4 **=** v\_prime(Vel3, Pos3);

**double** ky4 **=** r\_prime(Vel3, Pos3);

*// Paso4*

**double** average\_kx **=** (1.0**/**6.0)**\***(kx1 **+** 2.0**\***kx2 **+** 2.0**\***kx3 **+** kx4);

**double** average\_ky **=** (1.0**/**6.0)**\***(ky1 **+** 2.0**\***ky2 **+** 2.0**\***ky3 **+** ky4);

*// Promedio de las variables*

  t[i] **=** t[i**-**1] **+** h;

  Vel[i] **=** Vel[i**-**1] **+** h **\*** average\_kx;

  Pos[i] **=** Pos[i**-**1] **+** h **\*** average\_ky;

**return** 0;

}

**Graficador en python**

**import** pylab

**import** numpy

**import** math

data **=** numpy**.**loadtxt("trayectoria.dat")

t **=** data[:,0]

Vel **=** data[:,1]

Pos**=** data[:,2]

Costheta **=** data [:,3]

Sentheta**=** data [:, 4]

ejex**=** Pos**\***Costheta

ejey**=** Pos**\***Sentheta

pylab**.**plot(ejex,ejey, 'm')

pylab**.**xlabel('t')

pylab**.**ylabel('x')

pylab**.**title('X-T')

pylab**.**savefig("trayectoria.png")

pylab**.**close()

**print** "La grafica fue generada y guardada"

**graficas con velocidad inicial 0 (circulo) posicion inicial 0, y velocidad dada por el campo electrico**