UNIVERSIDAD NACIONAL DE INGENIERIA
FACULTAD DE CIENCIAS

Tema:
Problema de 2 cuerpos, ecuaciones de Kepler del movimiento

Apellidos: Moreno Vera
Nombres: Felipe Adrian
Código: 20120354I
Curso: Física Computacional
Codigo Curso: CC063

2016-II

Problema de los 2 cuerpos

Realizar la simulación de 2 cuerpos usando la mecánica de Newton y ecuaciones lagrangianas para el cálculo de las variables.

Se utilizó como base el programa de la simulación de la interacción de N Cuerpos realizado en el curso de Algoritmos Paralelos (CC301) UNI en el ciclo 2015-I

Preliminares:

Se decició implementar una mini librería llamada nbody en el lenguaje de programación C.
Que se copone del header nbody.h y la implementación nbody.c, el cual es llamado como #include “nbody.h”.

como compilar:
gcc TwoBodyProb.c nbody.c -lm -o TwoBody

o solo ejecutar bash make.sh
y saldrá por defecto el menú de ayuda:

```
Problema de 2 cuerpos:
El ingreso de datos es de la siguiente manera:
./TwoBodyProb [command]
                -- donde [command] puede ser:
[]              --  ejecuta la simulacion con el metodo de euler y 1000 iteraciones.
[num]           --  ejecuta la simulacion con el metodo de euler y [num] iteraciones.
[method]        --  ejecuta la simulacion con el metodo [method] y 1000 iteraciones.
[num] [method]  --  ejecuta la simulacion con el metodo [method] y [num] iteraciones.
```

Simulación:

Se simulará usando euler y leapfrog como se muestra en el siguiente código.

```
void particleInteraction(Particle *a, int metodo,FILE *f){
  // fuerza ejercida por b  que actua sobre a
  double ax=0,ay=0,az=0,r,vx1_2,vy1_2,vz1_2;
  int i,j;
  for(i=0;i<NBODIES;i++){
    // iniciando variables para euler y leapfrog
    // v1_2(t) = v(t)+a(t)*dt/2
    vx1_2 = a[i].vx + a[i].ax*dt/2.;
    vy1_2 = a[i].vy + a[i].ay*dt/2.;
    vz1_2 = a[i].vz + a[i].az*dt/2.;
    ax=ay=az=0;
    for(j=0;j<NBODIES;j++){
      if(i!=j){
        r = distance(a[j],a[i]);
        ax += G*a[j].masa*(a[j].x-a[i].x)/r;
        ay += G*a[j].masa*(a[j].y-a[i].y)/r;
```

```
      az += G*a[j].masa*(a[j].z-a[i].z)/r;
     }
    }
    a[i].ax=ax;
    a[i].ay=ay;
    a[i].az=az;
    if(metodo==1){ // metodo de euler
     // calculo de la velocidad
     // v(t+1) = v(t) + a(t)*dt
     a[i].vx = a[i].vx + ax*dt;
     a[i].vy = a[i].vy + ay*dt;
     a[i].vz = a[i].vz + az*dt;
     // calculo de la posicion
     // r(t+1) = r(t) + v(t)*dt
     a[i].x = a[i].x + a[i].vx*dt;
     a[i].y = a[i].y + a[i].vy*dt;
     a[i].z = a[i].z + a[i].vz*dt;
    } else if(metodo==2){ // metodo de leapfrog
     // calculo de la posicion
     // r(t+1) = r(t) + v1_2(t)*dt
     a[i].x = a[i].x + vx1_2*dt;
     a[i].y = a[i].y + vy1_2*dt;
     a[i].z = a[i].z + vz1_2*dt;
     // calculo de la velocidad
     // v(t+1) = v1_2(t) + a(t+1)*dt/2
     a[i].vx = vx1_2 + ax*dt/2.;
     a[i].vy = vy1_2 + ay*dt/2.;
     a[i].vz = vz1_2 + az*dt/2.;
    }
   }
   fprintf(f,"%lf\t%lf\t%lf\n",a[1].x,a[1].y,a[1].z);
   return ;
}
```

Para el método de leapfrog se modeló:

$v_{1/2}=v_0+a(x_0)\frac{h}{2}$, con $v_0\, y\, a(x_0)$ iniciales, para lueog utilizar la recursión:

$$v_{n+1/2}=v_n+a(x_n)\frac{h}{2} \quad , \quad x_{n+1}=x_n+v_{n+1/2}h \quad \text{y} \quad v_{n+1}=v_{n+1/2}+a(x_{n+1})\frac{h}{2}$$

a diferencia de euler que solo es:

$$v_{n+1}=v_n+a_nh \quad \text{y} \quad x_{n+1}=x_n+v_nh$$

Se ve que leapfrog:

Ventajas:

* Describe el comportamiento del sistema mediante aproximación intercalada.
* Es un método de orden 2.
* Usa las ecuaciones de velocidad de Verlet para aproximar las ecuaciones de derivadas, las cuáles simplifican el cálculo.

Desventajas:

* No se puede inicializar le método, requiere del valor de $v_{1/2}$.

Se ve que euler:

Ventajas:

* Es un método sencillo de implementar, pues requiere de valores iniciales en una iteración.
* Es un método de orden 2, basado en la aproximación de Taylor.
* Puede aproxima los valores de las derivadas mediante taylor.

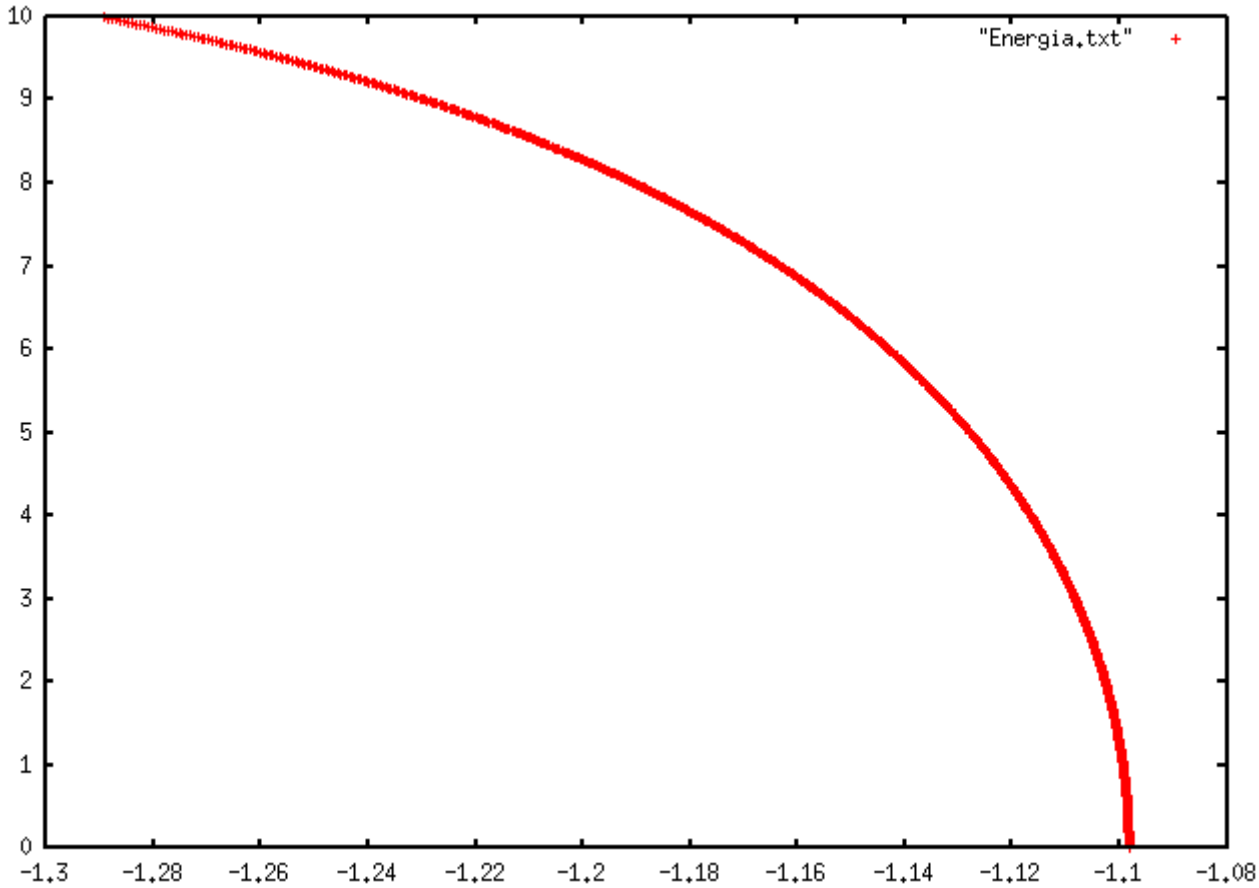
Desventajas:

* Tiene mayor error de truncamiento o de redondeo que leapfrog.

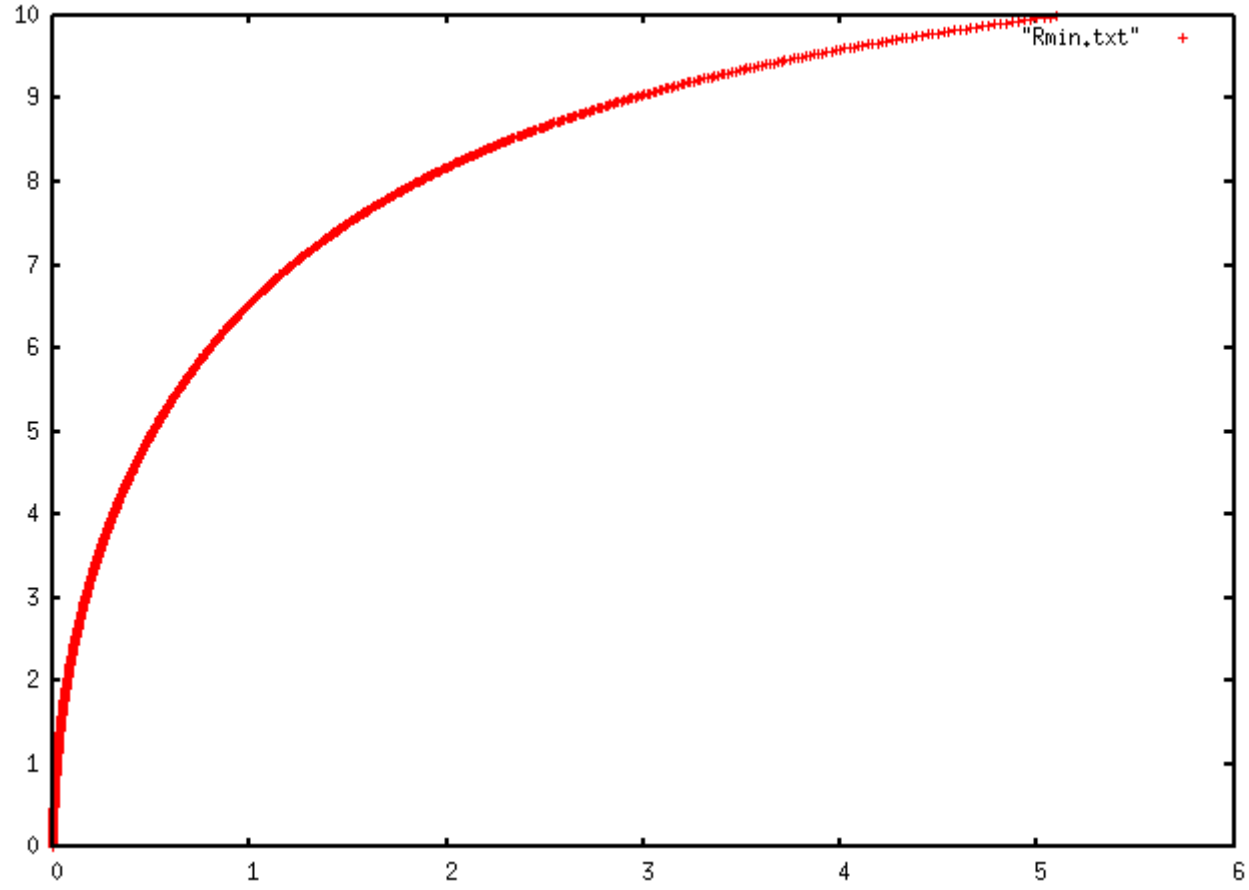
Gráficas:

Leapfrog:

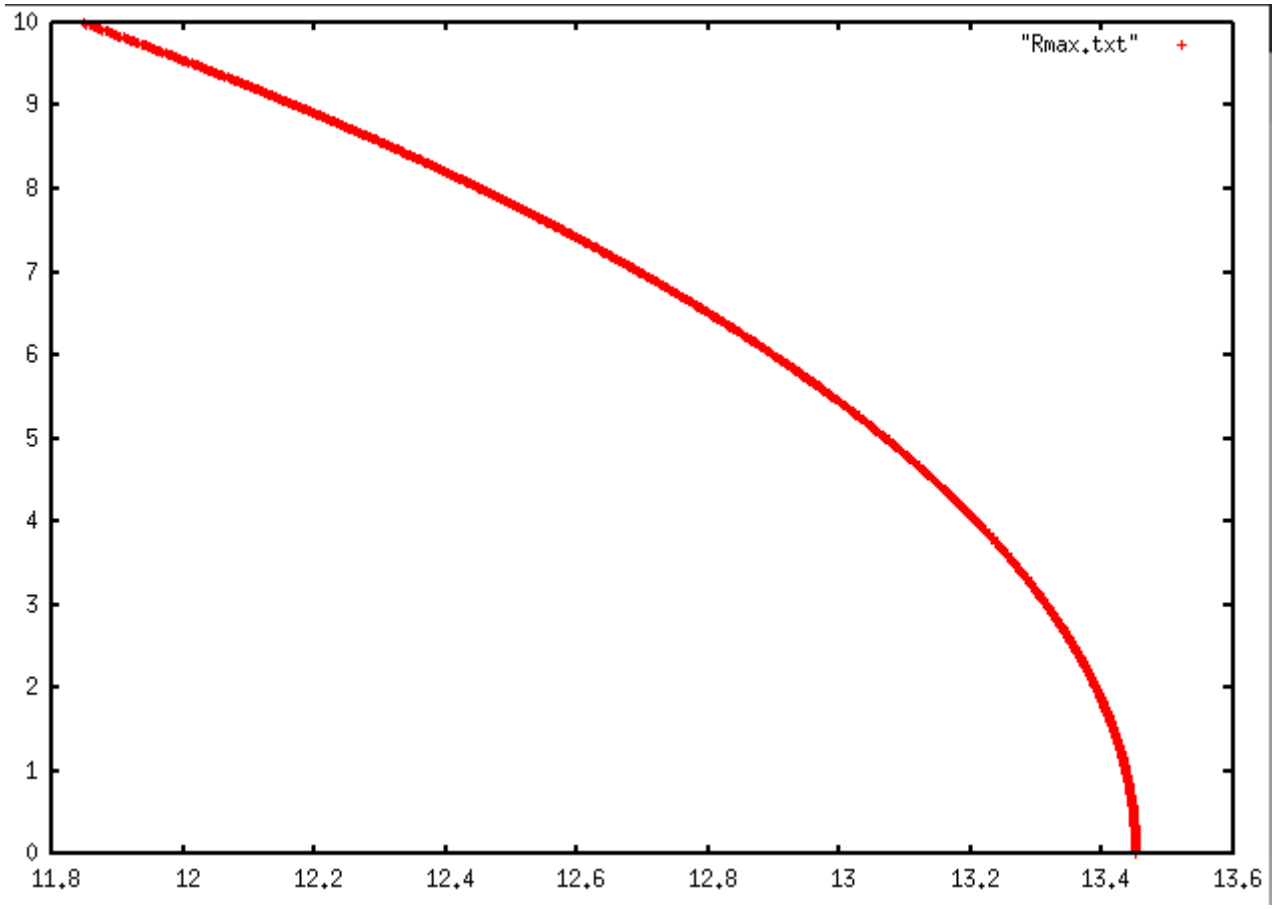
Energia:



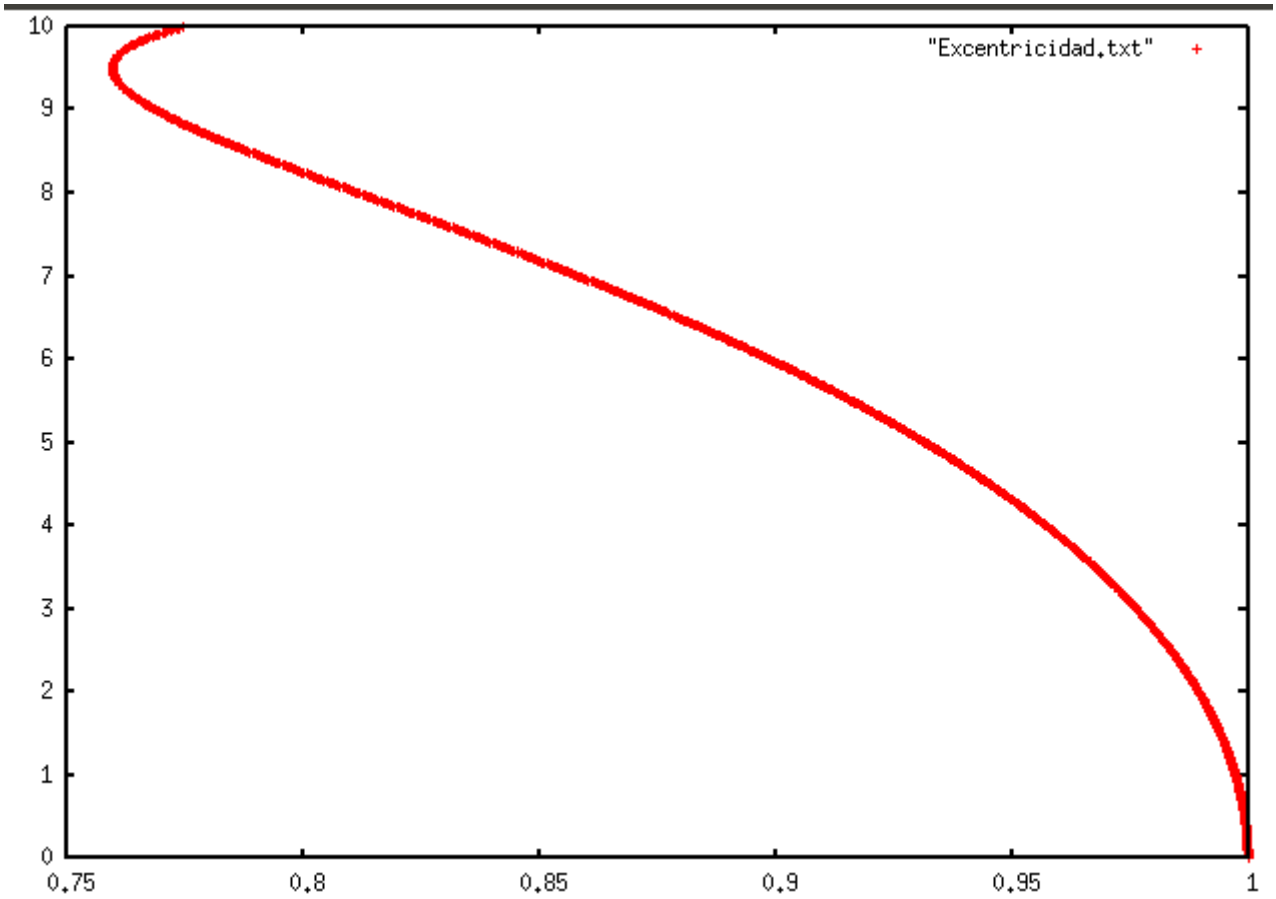
R mínimo:



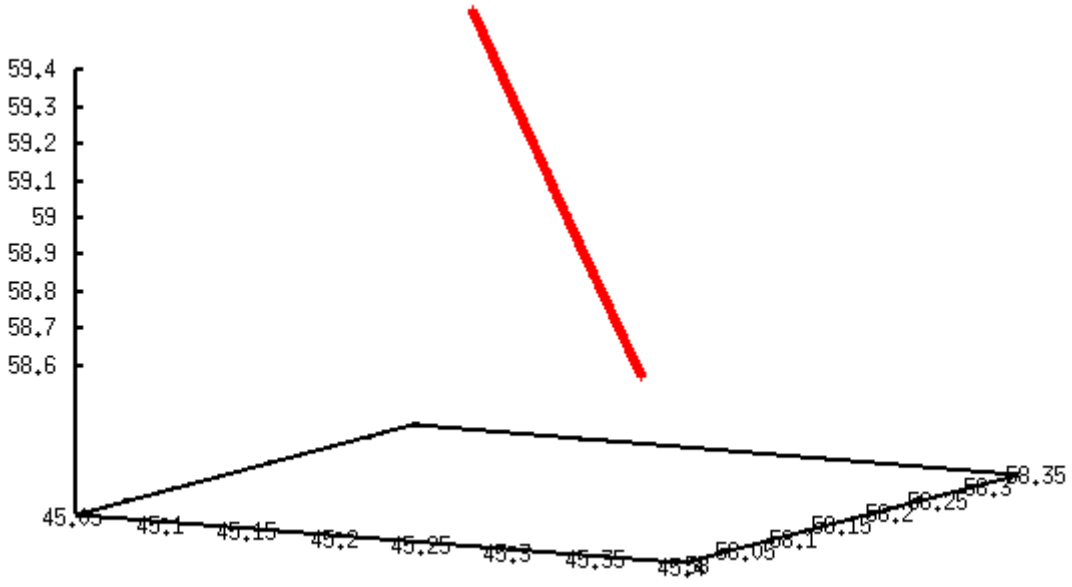
R máximo:



Excentricidad:

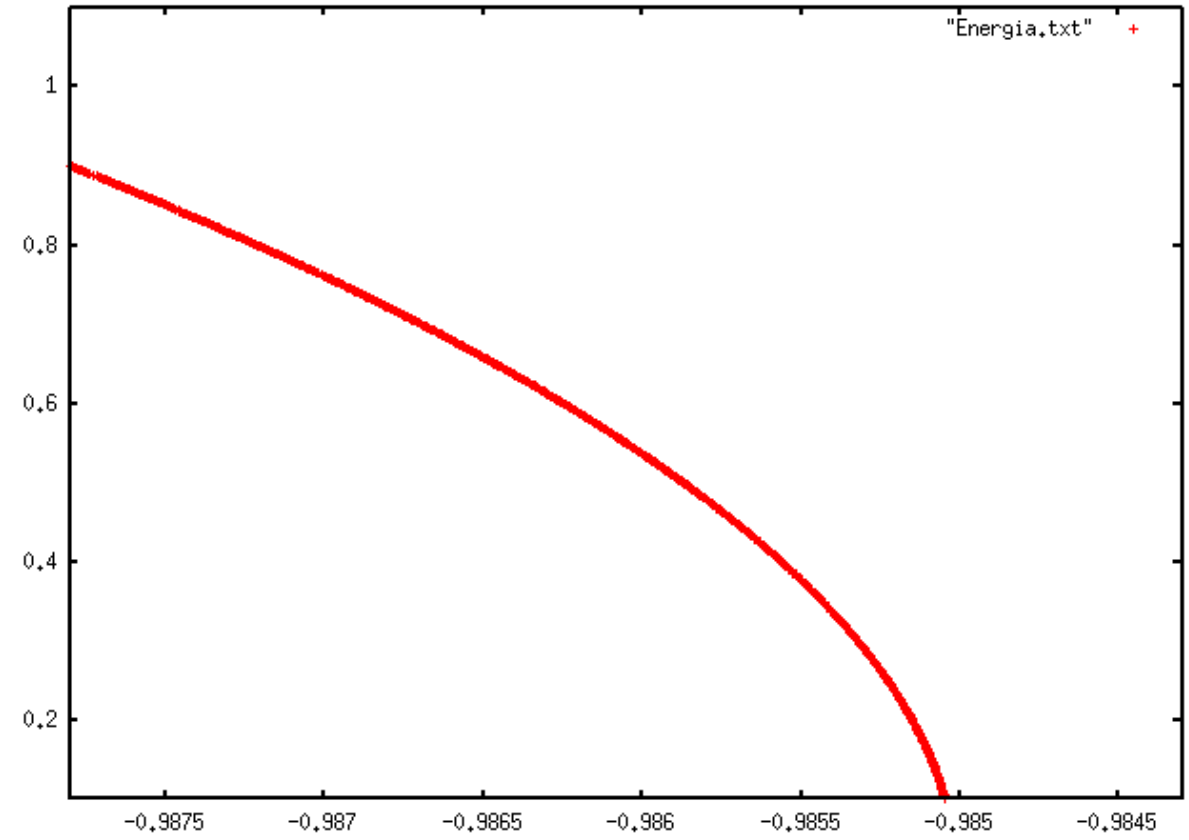


Movimiento:

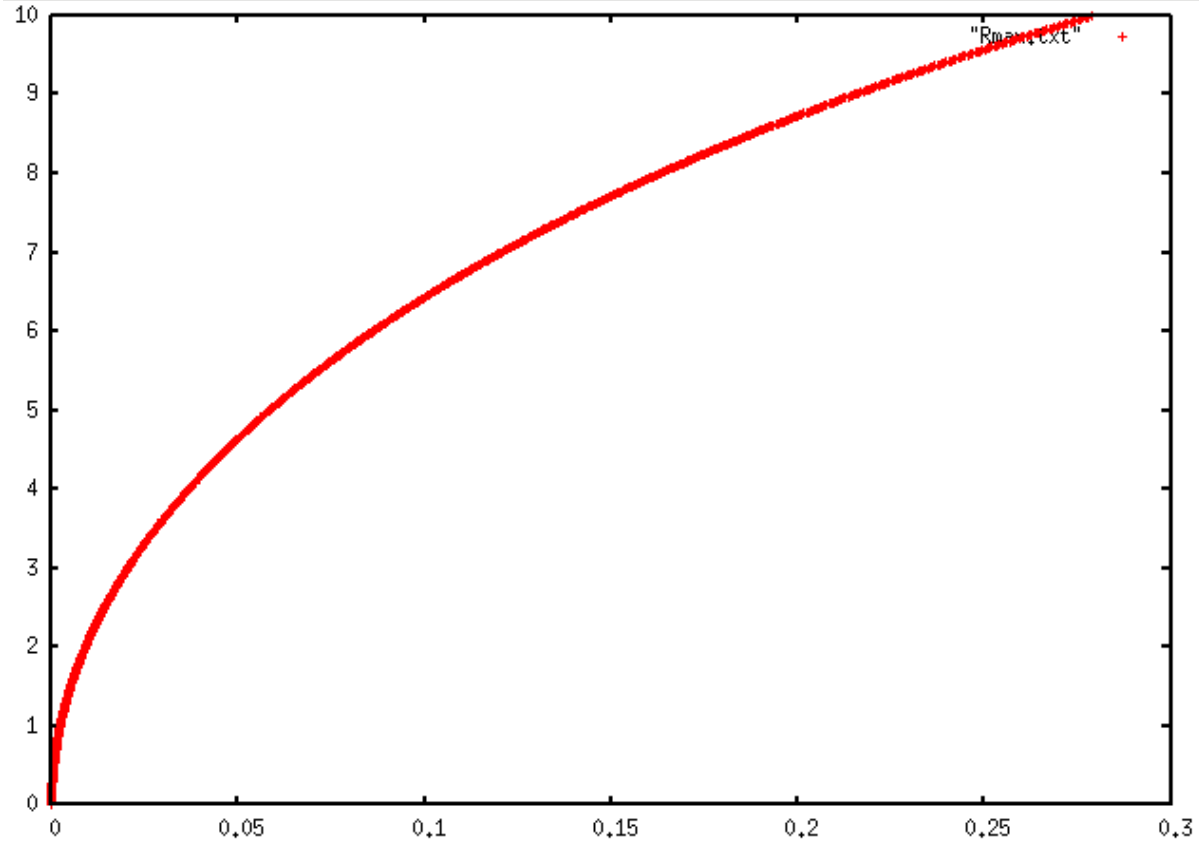


Euler:

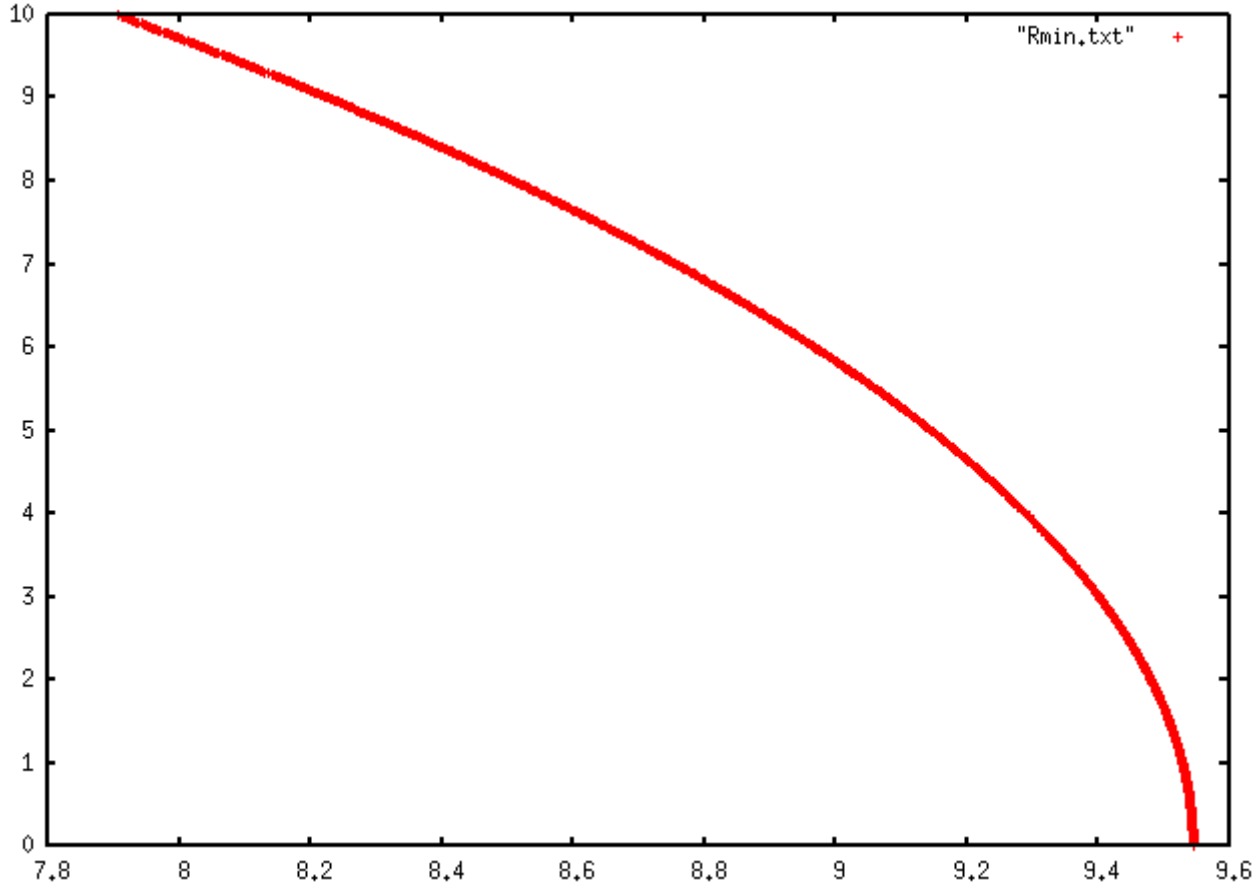
Energia:



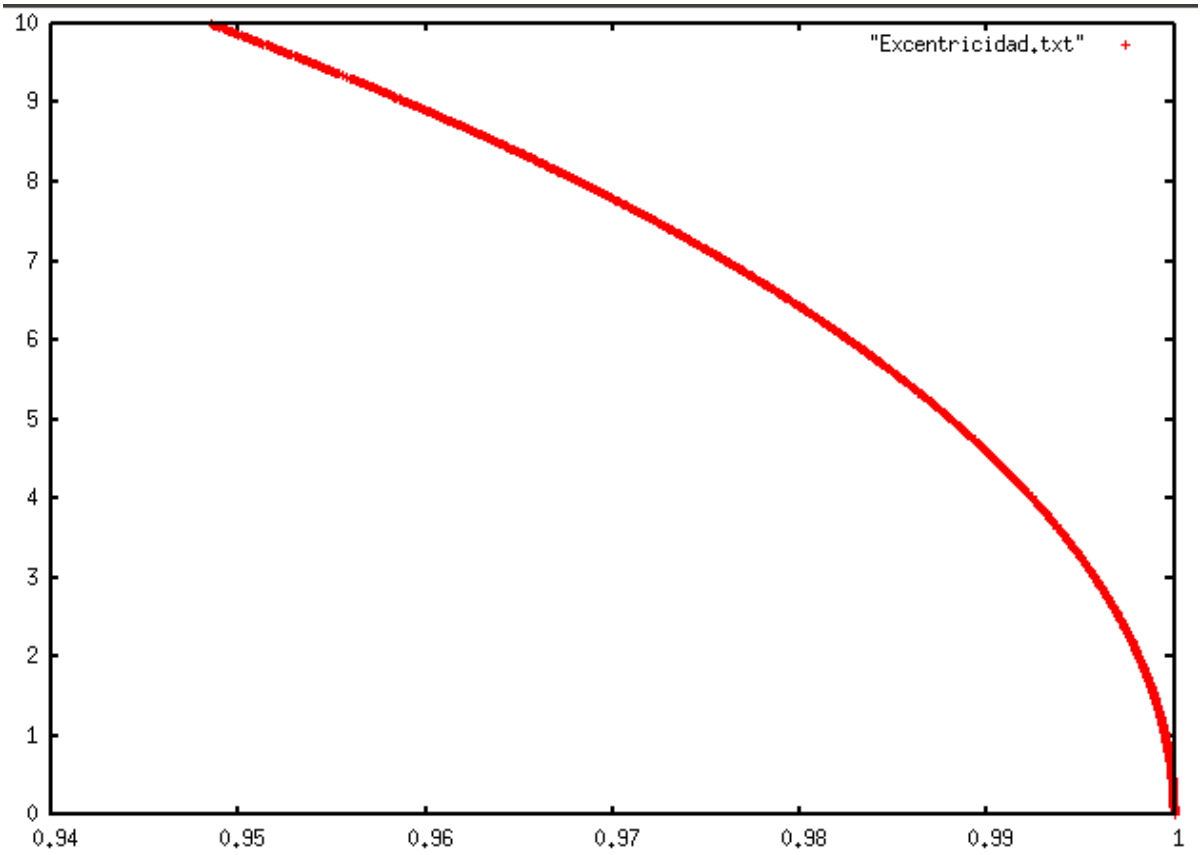
R mínimo:



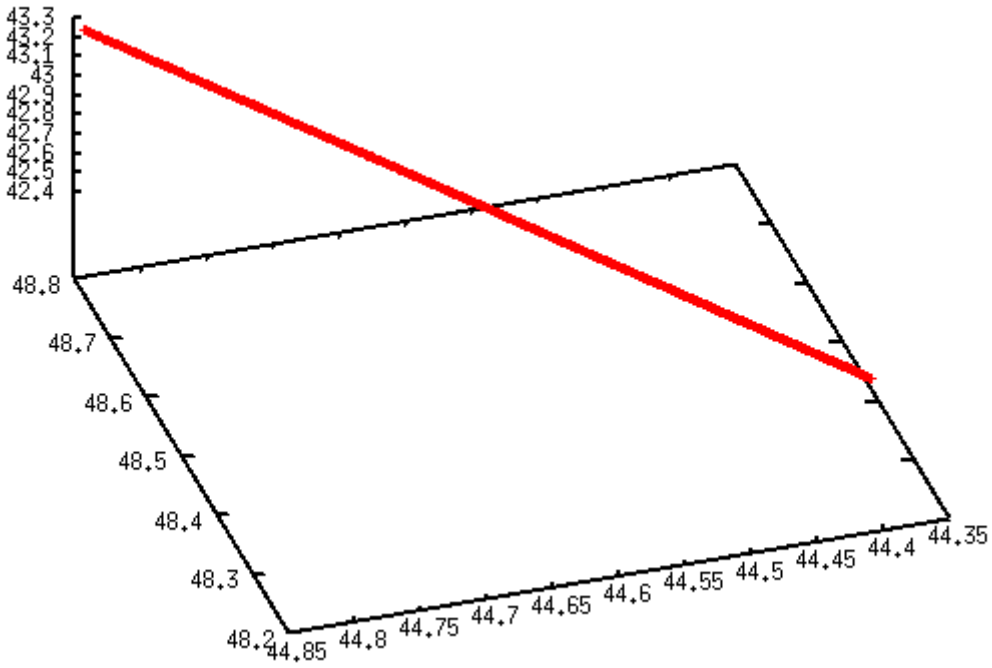
R máximo:



Excentricidad:



Movimiento:



Comparación:

Se ve que los métodos, la excentricidad calculada, en leapfrog, disminuye hasta 0.77, sin embargo, con euler hasta 0.99, y respecto a los radios, se ve que converban la misma relación (todo depende de los valores iniciales) y la Engería esta oscilando entre -0.9 y -1.1.

Probando para distintos valores de dt, se vió que leapfrog mantiene su margen de error y convergencia, sin embargo, euler rebotada, es decir, estaba en 0.00000023564 y luego saltaba a 2468995233.0000. el cual sucede cuando ocurre overflow.

La conservación se mantiene hasta dt = 0.01 y iteraciones = 10000, si aumentamos las iteraciones, ocurre lo comentado.

Observación:

Las gráficas de excentricidad y radios, parece que disminuye (pero es porque se realizó rmin x t,en vez de t x rmin).

Referencias:

https://en.wikipedia.org/wiki/Dot_product
https://en.wikipedia.org/wiki/Laplace%E2%80%93Runge%E2%80%93Lenz_vector
https://es.wikipedia.org/wiki/Vector_de_Runge-Lenz
https://es.wikipedia.org/wiki/Momento_angular
http://www.fisica.edu.uy/~sbruzzone/FlexPaper_1.4.2_flash/defensa.pdf