

Código Parcial Final

German Stevan Rubiano.

Física Computacional.

Facultad de Ciencias Matemáticas y Naturales

Universidad Distrital Francisco José de Caldas

8 de abril de 2024

Contenido

1 Ecuaciones Base utilizadas

2 Código Parcial Final

$$\frac{dr}{dt} = \pm \sqrt{\frac{2}{\mu} \left(E - \frac{L^2}{2\mu r^2} + \frac{GmM}{r} \right)}$$

Figura 1: Cuadratura

$$\frac{d\theta}{dt} = \frac{L^2}{\mu r^2}$$

Figura 2: Cuadratura

$$\frac{d\theta}{dr} = \frac{\frac{L_{cm}}{\mu r^2}}{\sqrt{\frac{2}{\mu} \left(E_{cm} - \frac{L_{cm}^2}{2\mu r^2} + \frac{Gm_1m_2}{r} \right)}}$$

Figura 3: Cuadratura

Al despejar dr :

$$dr = \frac{\mu r^2}{L} \sqrt{\frac{2}{\mu} \left(E_{cm} - \frac{L^2}{2\mu r^2} + \frac{GMm}{r} \right)} d\theta$$

```
1 #include<iostream>
2 #include<cmath>
3 #include<fstream>
4
5 // Primero se definen los valores constantes y las
  igualdades.
6
7 const double epsilon=0.0167; // Valor determinado en las
  diapositivas del parcial.
8 const double R=1;
9 const double a=R; // El e es una forma de notacion para
  representar el 10^{x} valor en c++.
10 const double M_s=1.989e30; // Masa del sol en kg.
11 const double v_0=30300; // Valor de velocidad de la
  tierra en m/s.
12 const double M_t=5.972e24; // Masa de la tierra en kg.
13 const double G=6.67430e-11;
14 const double mu=(M_s*M_t)/(M_s+M_t); // El valor de mu
  se determina gracias a la ecuacion proporcionada por
  el ejercicio.
15 const double T = sqrt((pow(R,3)/(2*G*mu)));
```

```
1 const double E=(0.5)*v_0*mu+(G*M_s*mu)/R;  
2 const double R_max=1.017;  
3 const double R_min=0.983;  
4 const double L = R * mu * v_0;  
5 const double a_prima = (1-pow(epsilon,2));
```

```
1 double dr_theta(double L, double E, double r){
2     if(r==0){
3         return 0; // Se formula un if para evitar
        errores de divisi n sobre 0.
4     }
5
6     long double Multiplicador = (mu*pow(r,2))/L;
7     long double Discriminante = (2/mu)*(E-(pow(L,2)/(2*
        mu*pow(r,2)))+(G*M_s*M_t)/r);
8     double raiz;
9     if(Discriminante < 0){
10         return raiz=sqrt(abs(Discriminante)); //
        Nuevamente con un if se determina una forma para
        evitar errores,
11         // de raices negativas.
12     }
13     else{
14         raiz = sqrt(Discriminante);
15     }
16
17     return Multiplicador*raiz;
```

```
1 // Metodo de Simpson
2 long double simpsonr_theta(double a, double b, int N,
   double L, double E){
3     long double h=(b-a)/N; // Se determina el valor del
   paso, incluyendo los parametros de a y b, obtenidos
   mediante la integral del ejercicio.
4     double suma= dr_theta(a, L, E)+ dr_theta(b, L, E);
   //Se determina el primer y ultimo valor de la
   integral numerica.
5     for(int i=1;i<N;i++){ //Se determina el valor de los
   pasos o avance en el que se dividiran los valores de
   x.
6         double xi=a+(i*h);
7         if (i % 2 ==0){ //Se determina la diferencia de
   pares e impares gracias a la operaci n de residuo.
8             suma+= 2*dr_theta(xi, L, E);
9         }
```

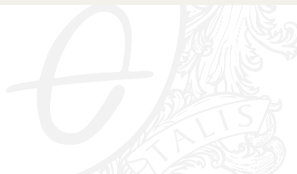


```
1      else{  
2          suma+=4*dr_theta(xi, L, E);  
3      }  
4  }  
5  return (h/3)*suma;  
6 }
```

```
1 // Metodo del Trapecio.
2
3 double Trapecior_theta(double a, double b, int N, double
   L, double E){
4     double h=(b-a)/N;
5     long double suma= dr_theta(a, L, E)+ dr_theta(b, L,
   E);
6     for(int i=1;i<N;i++){
7         double xi=a+(i*h);
8         suma+= 2*dr_theta(xi, L, E);
9     }
10    return (h/2)*suma;
11 }
```

```
1 // Cuadratura rectangular.
2
3 double Cuadraturar_theta(double a, double b, int N,
4     double L, double E){
5     long double h=(b-a)/N;
6     double suma=0.983;
7     for(int i=0;i<N;i++){
8         double xi=a+i*h;
9         suma+=dr_theta(xi, L, E);
10    }
11    return h*suma;
12 }
```

UBI
VERITAS
IBI
LIBERTAS



```
1 // Regla de Gauss.
2
3 double Gaussr_theta(double a, double b, double r, double
  L, double E){
4
5     long double x[6]={-0.93246951, -0.66120938,
6     -0.23861918, 0.23861918, 0.66120938, 0.93246951};
7     long double w[6]={0.17132449, 0.36076157,
8     0.46791393, 0.46791393, 0.36076157, 0.17132449};
9 // Se determinan los valores de xi y los valores de peso
10    w como vectores, para asi llamarlos posteriormente.
11    long double suma;
12    for(int i=0; i<6; ++i){
13        long double xi=0.017*x[i]+1; //Se determina el
14        cambio de la integral donde los valores son el
15        resultado de
```

```
1 //Aplicar la division y restas correspondientes tal y  
  como induca la formula general de Gauss.  
2     suma=dr_theta(xi, L, E)*w[i];  
3     } // Tanto el x[i], como el w[i], se usan dentro del  
      bucle for para asi obtener que se multipliquen  
4 // Todos los componentes que los conforman en la suma.  
5     return ((b-a)/2)*suma;  
6 }
```

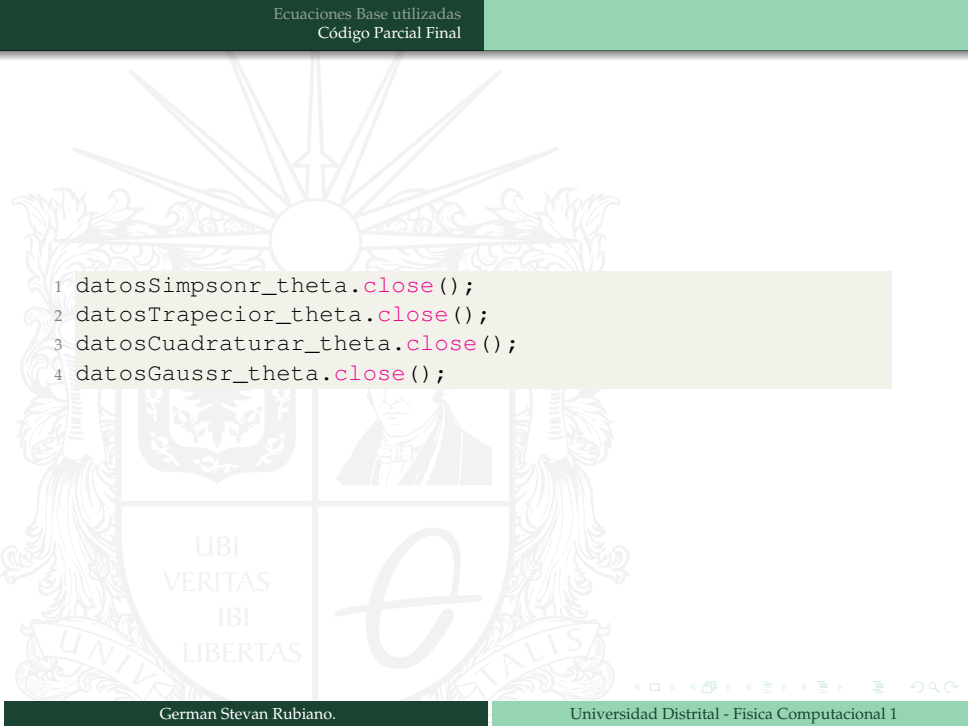
```
1 int main(){// Se crean los documentos necesarios para
  poder guardar los datos generados por los metodos.
2   std::ofstream datosSimpsonr_theta("
  Resultados_Metodo_Simpson_radio_theta.dat");
3   std::ofstream datosTrapezior_theta("
  Resultados_Metodo_Trapezio_radio_theta.dat");
4   std::ofstream datosCuadraturar_theta("
  Resultados_Metodo_Cuadratura_radio_theta.dat");
5   std::ofstream datosGaussr_theta("
  Resultados_Metodo_Gauss_radio_theta.dat");
6 //Se determina una forma para obtener datos de Theta,
  conociendo que estos datos pueden variar de 0 a 360
7   const int num_puntos_theta = 1000;
8
9   // Rango de theta
10  const double theta_min = 0.0;
11  const double theta_max = 360.0;
```

```
1 // Calcula el paso angular
2 const double deltaTheta = (theta_max - theta_min) /
3 (num_puntos_theta - 1);
4
5 // Genera los valores de theta
6 double theta[num_puntos_theta];
7 for (int i = 0; i < num_puntos_theta; ++i) {
8     theta[i] = i * deltaTheta;
9 }
```

```
1  for (int N = 1; N < 1000; ++N) { for (int i = 0; i <
2      num_puntos_theta; ++i) {
3      long double r_theta1 = simpsonr_theta(R_min,
4      R_max, N, L, E);
5      long double r_theta2 = Trapecior_theta(R_min
6      , R_max, N, L, E);
7      long double r_theta3 = Cuadraturar_theta(
8      R_min, R_max, N, L, E);
9      long double r_theta4 = Gaussr_theta(R_min,
10     R_max, N, L, E);
```

UBI
VERITAS
IBI
LIBERTAS θ


```
1 // Generamos los valores en modo polar.
2     long double x1=r_theta1*cos(theta[i]);
3     long double x2=r_theta2*cos(theta[i]);
4     long double x3=r_theta3*cos(theta[i]);
5     long double x4=r_theta4*cos(theta[i]);
6
7     long double y1=r_theta1*sin(theta[i]);
8     long double y2=r_theta2*sin(theta[i]);
9     long double y3=r_theta3*sin(theta[i]);
10    long double y4=r_theta4*sin(theta[i]);
11
12    datosSimpsonr_theta<<x1<<" "<<y1<<std::endl;
13    datosTrapecior_theta<<x2<<" "<<y2<<std::endl;
14    datosCuadraturar_theta<<x3<<" "<<y3<<std::endl;
15    datosGaussr_theta<<x4<<" "<<y4<<std::endl;
16
17 }
18 }
```



```
1 datosSimpsonr_theta.close();  
2 datosTrapezior_theta.close();  
3 datosCuadraturar_theta.close();  
4 datosGaussr_theta.close();
```

```
1 std::ofstream Grafica_Simpson_Radio_Theta("  
    Grafico_Simpson_Radio_theta.gp");  
2 Grafica_Simpson_Radio_Theta<<"set output '  
    Grafico_Simpson_Radio_theta.png'\n";  
3 Grafica_Simpson_Radio_Theta<<"set xlabel 'Theta'\n";  
4 Grafica_Simpson_Radio_Theta<<"set ylabel 'Radio'\n";  
5 Grafica_Simpson_Radio_Theta<<"plot '  
    Resultados_Metodo_Simpson_radio_theta.dat' u 1:2  
    title 'Trayectoria de la Tierra'\n";  
6 Grafica_Simpson_Radio_Theta.close();
```

UBI
VERITAS
IBI
LIBERTAS



```
1 //Ejecutar todos los archivos generados de Gnuplot.  
2  
3 system("gnuplot Grafico_Simpson_Radio_theta.gp");  
4 system("gnuplot Grafico_Trapezio_Radio_theta.gp");  
5 system("gnuplot Grafico_Cuadratura_Radio_theta.gp");  
6 system("gnuplot Grafico_Gauss_Radio_theta.gp");  
7 system("gnuplot Grafica_Analitica.gp");  
8  
9 return 0;  
10 }
```

UBI
VERITAS
IBI
LIBERTAS



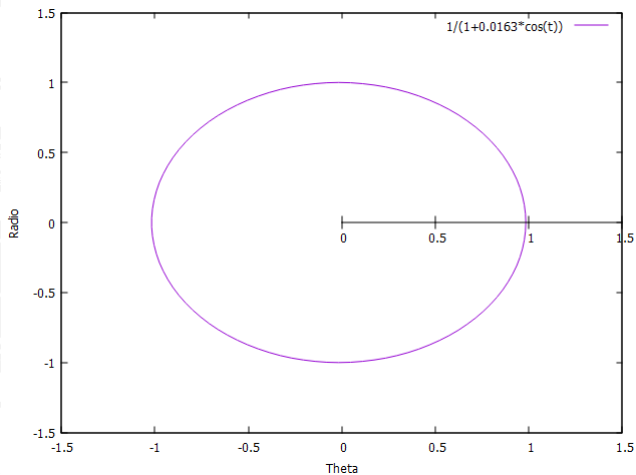


Figura 4: Analítica

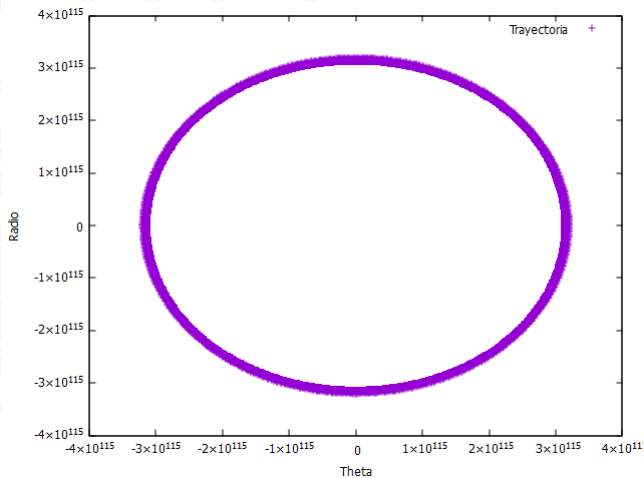


Figura 5: Rectangular

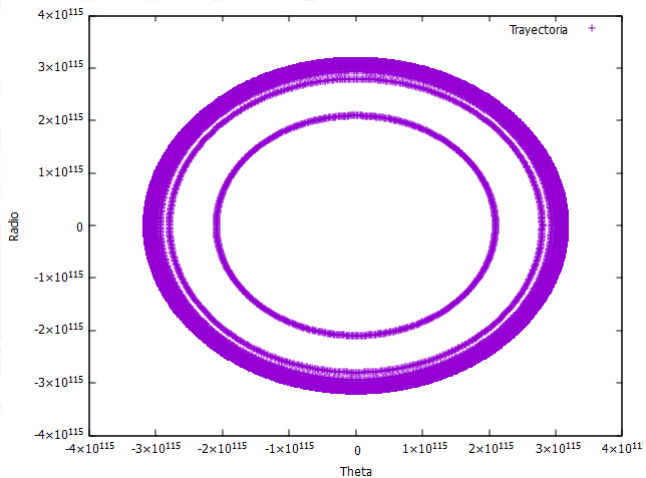


Figura 6: Simpson

El error puede deberse al bucle for de N dentro del main, esto debido a que los valores de la función r_{theta1} se pueden estar comparando con diversos N al mismo tiempo y no de una forma inyectiva. Obteniendo así para un valor de r distintos valores de N.

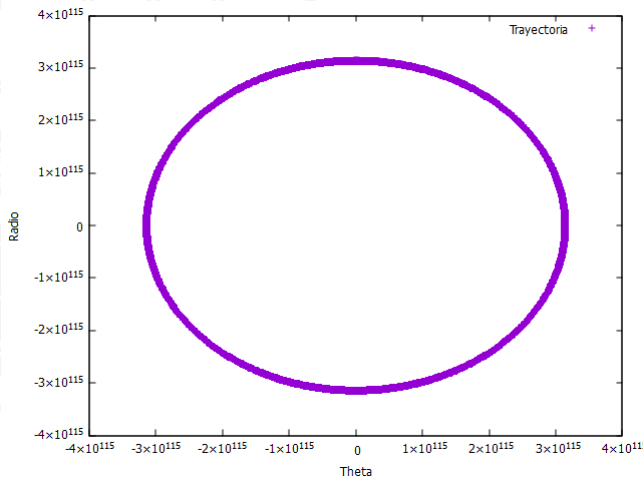


Figura 7: Trapecio

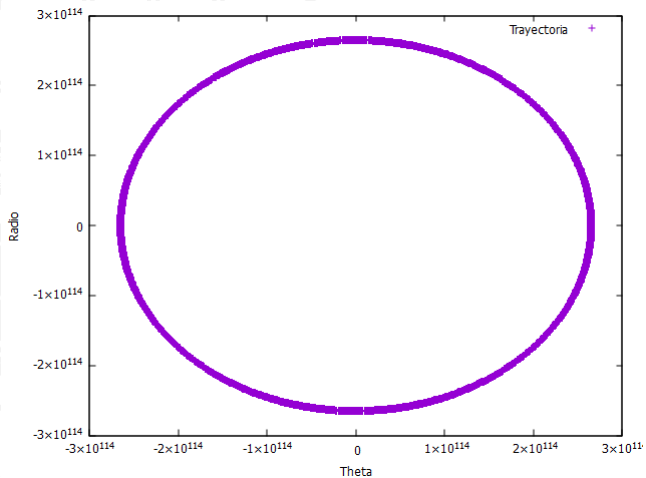


Figura 8: Gauss

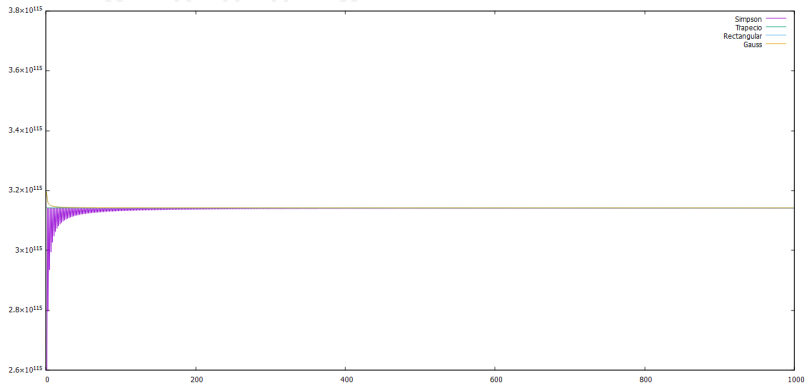


Figura 9: Error