

Código Parcial Final

German Stevan Rubiano.

Física Computacional.

Facultad de Ciencias Matemáticas y Naturales

Universidad Distrital Francisco José de Caldas

5 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

UBI
VERITAS
IBI
LIBERTAS



```

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)));
16 const double E=(0.5)*pi*0*mu*(G*M_s*mu)/R;

```

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

```
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         }
10        else{
11            suma+=4*dr_theta(xi, L, E);
12        }
13    }
14    return (h/2)*suma;
```

```
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 }
12
13 // Cuadratura rectangular.
14
15 double Cuadraturar_theta(double a, double b, int N,
    double L, double E){
16     long double h=(b-a)/N;
17     double suma=0.983;
18     for(int i=0;i<N;i++){
19         double xi=a+i*h;
```



```
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
    w como vectores, para asi llamarlos posteriormente.
10    long double suma;
11    for(int i=0; i<6; ++i){
12        long double xi=0.017*x[i]+1; //Se determina el
13        cambio de la integral donde los valores son el
14        resultado de
15 //Aplicar la division y restas correspondientes tal y
16 como induca la formula general de Gauss.
17        suma=dr_theta(xi, L, E)*w[i];
18    } // Tanto el x[i], como el w[i], se usan dentro del
19    bucle for para asi obtener que se multipliquen
20 // Todas las componentes que los conforman en la suma
```

```
1 double r_real(double L, double E, double theta){
2     long double C=G*M_s*M_t; //Se determina la siguiente
    igualdad para reducir escritura.
3     long double arriba=(pow(L,2))/(mu*C);
4     long double Determinante1 = 1+((2*E*pow(L,2))/(mu*
    pow(C,2))*sin(theta));
5     long double raiz;
6     if(Determinante1<0){
7         raiz=sqrt(abs(Determinante1));
8     }//Al igual que en anteriores casos se determina por
    medio de un if una forma de evitar valores de
9 // Raices negativas.
10    else{
11        raiz=sqrt(Determinante1);
12    }
13    long double r= arriba/(1-raiz);
14    return r;
15 }
```

```
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   std::ofstream datosRealesr_theta("
  Resultado_Solucion_Analitica.dat");
7 //Se determina una forma para obtener datos de Theta,
  conociendo que estos datos pueden variar de 0 a 360
8   const int num_puntos_theta = 1000;
9
10  // Rango de theta
11  const double theta_min = 0.0;
12  const double theta_max = 360.0;
13
14  // Calcula el paso angular
15  const double deltaTheta = (theta_max - theta_min) /
```

```
1  for (int N = 1; N < 1000; ++N) { for (int i = 0; i <
   num_puntos_theta; ++i) {
2
3      long double r_theta1 = simpsonr_theta(R_min,
      R_max, N, L, E);
4      long double r_theta2 = Trapecior_theta(R_min
      , R_max, N, L, E);
5      long double r_theta3 = Cuadraturar_theta(
      R_min, R_max, N, L, E);
6      long double r_theta4 = Gaussr_theta(R_min,
      R_max, N, L, E);
7      long double r_realtheta = r_real(L, E, theta
      [i]);
8
9
10 // Generamos los valores en modo polar.
11     long double x1=r_theta1*cos(theta[i]);
12     long double x2=r_theta2*cos(theta[i]);
13     long double x3=r_theta3*cos(theta[i]);
14     long double x4=r_theta4*cos(theta[i]);
15     long double x5=r_realtheta*cos(theta[i]);
```

```
1      long double y1=r_theta1*sin(theta[i]);  
2      long double y2=r_theta2*sin(theta[i]);  
3      long double y3=r_theta3*sin(theta[i]);  
4      long double y4=r_theta4*sin(theta[i]);  
5      long double y5=r_realtheta*sin(theta[i]);  
6      datosSimpsonr_theta<<x1<<" "<<y1<<std::endl;  
7      datosTrapecior_theta<<x2<<" "<<y2<<std::endl;  
8      datosCuadraturar_theta<<x3<<" "<<y3<<std::endl;  
9      datosGaussr_theta<<x4<<" "<<y4<<std::endl;  
10     datosRealesr_theta<<x5<<" "<<y5<<std::endl;  
11 }  
12 }
```

UBI
VERITAS
IBI
LIBERTAS



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

```
1 // Generación de graficas a base de Gnuplot.
2
3 // Graficas para Radio vs Theta con los 4 metodos.
4 std::ofstream Resultado_Solucion_Analitica("
    Resultado_Solucion_Analitica.gp");
5 Resultado_Solucion_Analitica<<"set term png\n";
6 Resultado_Solucion_Analitica<<"set output '
    Resultado_Solucion_Analitica.png'\n";
7 Resultado_Solucion_Analitica<<"set xlabel 'Theta'\n";
8 Resultado_Solucion_Analitica<<"set ylabel 'Radio'\n";
9 Resultado_Solucion_Analitica<<"set logscale x\n";
10 Resultado_Solucion_Analitica<<"plot '
    Resultado_Solucion_Analitica.dat' u 1:2 w l title '
    Trayectoria de la Tierra'\n";
11 Resultado_Solucion_Analitica.close();
```

```
1 //Ejecutar todos los archivos generados de Gnuplot.  
2  
3 system("gnuplot Grafico_Simpson_Radio_theta.gp");  
4 system("gnuplot Grafico_Trapecio_Radio_theta.gp");  
5 system("gnuplot Grafico_Cuadratura_Radio_theta.gp");  
6 system("gnuplot Grafico_Gauss_Radio_theta.gp");  
7 system("gnuplot Resultado_Solucion_Analitica.gp");  
8  
9  
10 return 0;  
11 }
```

UBI
VERITAS
IBI
LIBERTAS





Figura 4: Analítica

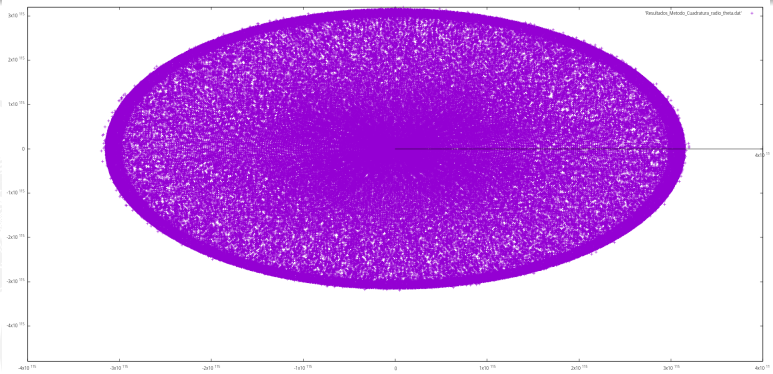


Figura 5: Cuadratura

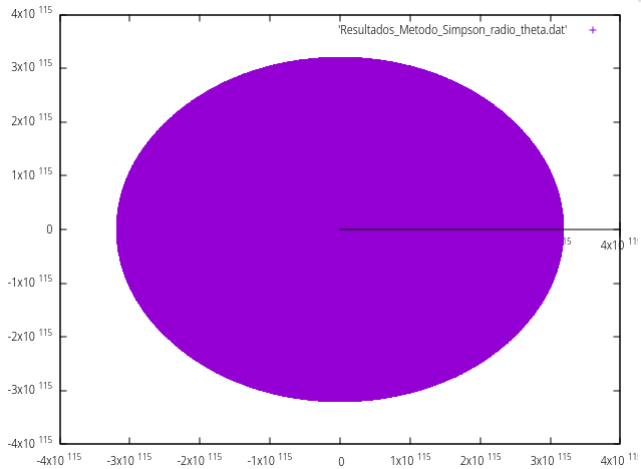


Figura 6: Simpson

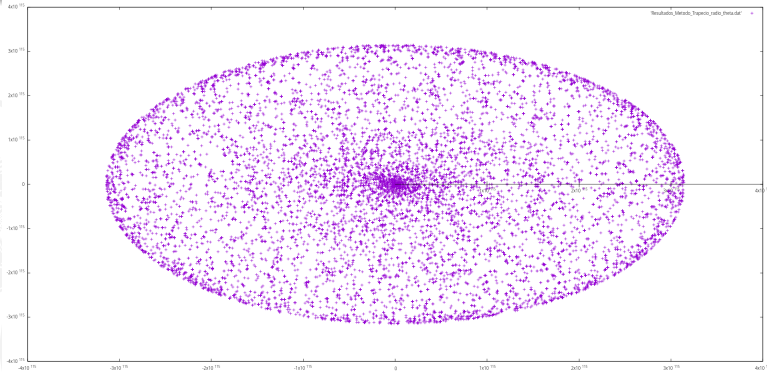


Figura 7: Trapecio

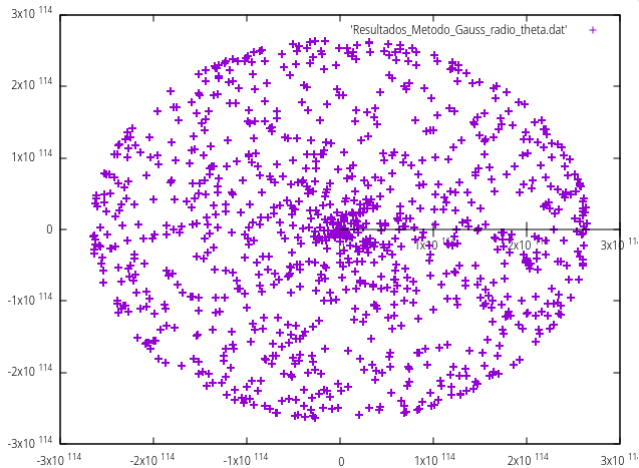


Figura 8: Gauss