Homework 2: Kepler Orbits

Zangarini Riccardo

November 23, 2021

1 Code

1.1 Description

The aim of the project is to solve the trajectory equations for an object with elliptical motion. This motion is described by the following parametric equations:

$$x(E) = a(\cos(E) - e), \tag{1}$$

$$y(E) = a\sqrt{1-e^2}\sin(E), \tag{2}$$

where E is a time-dependent parameter:

$$E = \omega t + e\sin(E). \tag{3}$$

To solve (3) it is necessary to find the roots of the function

$$f(E) = \omega t + e\sin(E) - E, \tag{4}$$

called *anomaly* in the code. This needs to be done for N time intervals from $t = 0$ to $t = T$, where T is the orbit period.

The procedure is the following: a constant time step is defined as:

$$\Delta t = \frac{T}{N} \tag{5}$$

for each time step the `Newton` function finds the root of the (3).
This value is then used to solve (1) and (2), providing the trajectory on the xy plane.

The root-finder function Newton is defined as *double* because it returns the value of the root itself. This is useful to define the parameter E in the line 33.

Furthermore, Newton does not require a preliminary bracketing because the function (3) is a monotonous decreasing function, with a quite smooth profile[1].

1.2 Code

```cpp
#include "my_header.h"
#include <fstream>

double anomaly(double t, double E);

double dv_anomaly(double t, double E);

double x_func(double t, double E, double a);

double y_func(double t, double E, double a);

const double e = 0.55;

int main(){
  int i, N = 100;
  double T = 1., a = 1., sx = -1., dx = 6., tol = 1.e-7, E;
  double dt = T / (double)N;
  static double t = 0.;
  ofstream fdata;

  fdata << setiosflags(ios::scientific);

  fdata.open("kepler.txt", std::ofstream::out);
  fdata << "t" << "\t \t" << "x(t)" << "\t \t" << "y(t)" << "\n";

  for(i = 0; i < N; i++){
    E = Newton(anomaly, dv_anomaly, sx, dx, tol, t);

    fdata << std::setprecision(6) << t << "\t" << x_func(t, E, a);
    fdata << "\t" << y_func(t, E, a) << "\n";

    t += dt;
  }

  fdata.close();

  return 0;
}

double anomaly(double t, double E){
  return (2 * M_PI) * t + (e * sin(E)) - E;  //since T = 1., T is not included
                                             //in the definition
}

double dv_anomaly(double t, double E){
  return e * cos(E) - 1.;
}

double x_func(double t, double E, double a){
  return a * (cos(E) - e);
}

double y_func(double t, double E, double a){
  return a * sqrt(1. - e * e) * sin(E);
}
```

2 Plot

The final goal of the project is to generate a plot of the trajectory on the xy plane, namely the time evolution of the coordinates of the object during a period T.
The resulting plot is the following:

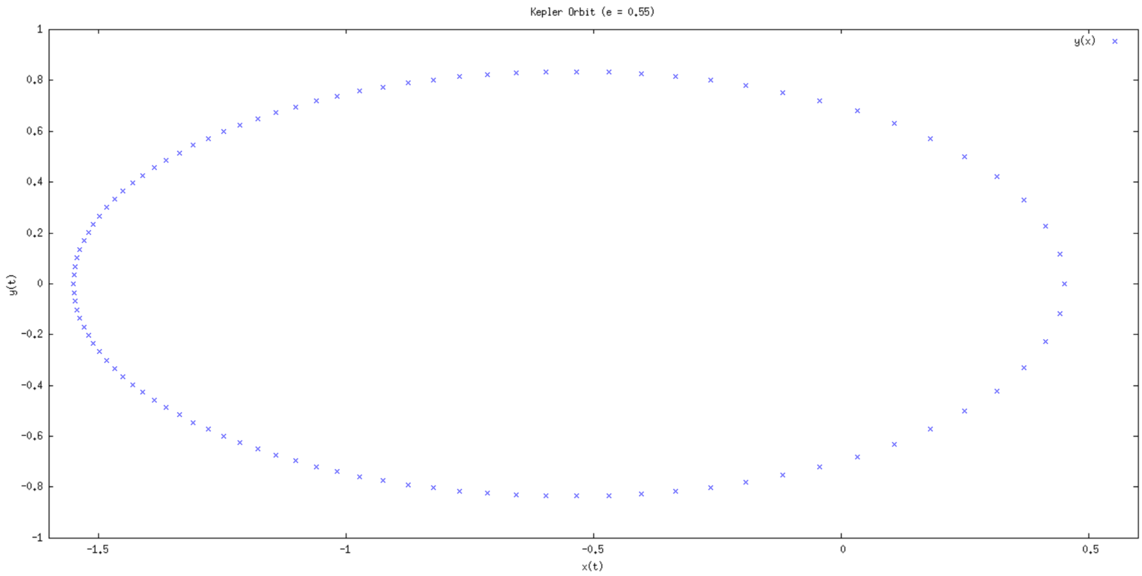

Figure 1: plot of the trajectory $y(x)$, using $e = 0.55$ and $T = 1$.

Notes

1:

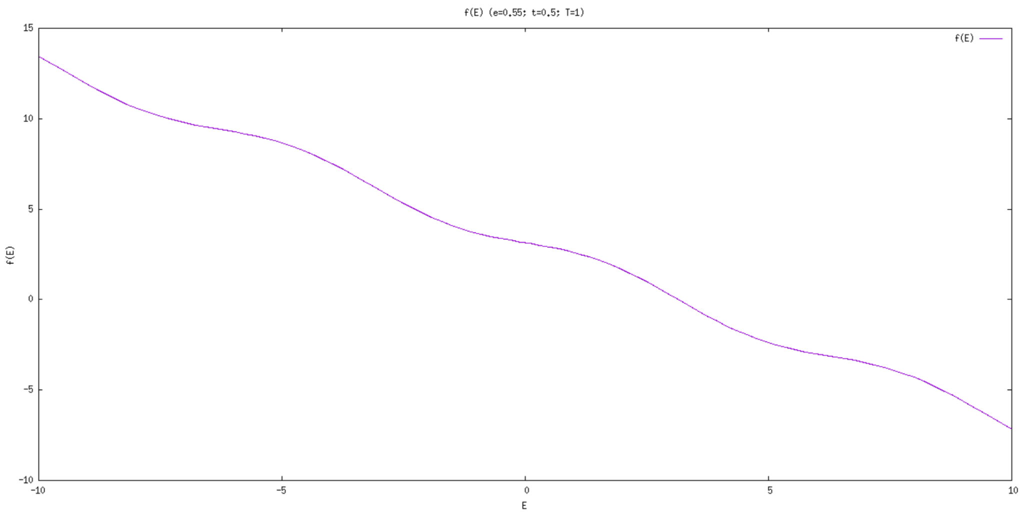

Figure 2: Function (3) using $e = 0.55$, $t = 0.5$ and $T = 1$.