Bolygómozgás

Nagy Péter
m07ilf

2018.02.31.

Tartalomjegyzék

1. Bevezetés

Ebben a szimulációs kisérletben a bolygómozgások modellezésével foglalkoztam.

2. Elméleti áttekintés

A bolygó mozgásokat a Kepler törvények írják le:

$$\vec{F}_{12} = -G\frac{m_1 m_2}{r_{12}^3}\vec{r}_{12} \tag{1}$$

Az erőtér centrális és egy síkban történik. A nap és bolygók modelezése esetén a napot mozdulatlannak lehet tekinteni. Mivel ellipszis pályán keringenek ezért kifejezhető a távolság θ szög függvényében és meghatározható a sebesség is:

$$r(\theta) = \frac{a(1-\epsilon)}{1-\epsilon cos\theta} \tag{2}$$

$$b = a\sqrt{1-\epsilon^2} \tag{3}$$

$$v = \sqrt{G(m_1 m_2)(\frac{2}{r} - \frac{1}{a})} \tag{4}$$

A keringési időre a következőt kapjuk a Kepler törvényekből:

$$T = \sqrt{\frac{4\pi^2 a^3}{G m_1 m_2}} \tag{5}$$

3. Mérési feladatok

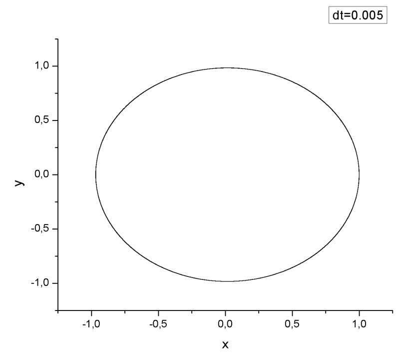
A mérés során a távolságot csillagászati egységben számoltuk és az időt évben mérjük.

3.1. A pályaellipszis nagytengelyének vizsgálata

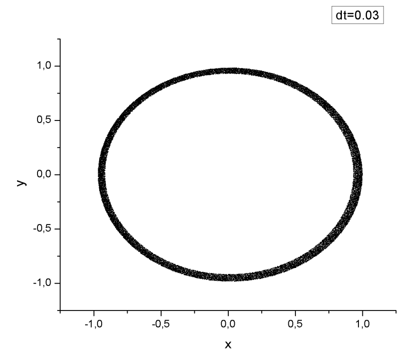
- aphélium távolság: 1 AU
- excentricitás:0.0167
- periodus szám: 1000
- adaptív pontosság: 0.00001

3.1.1. A lépéshossz hatása a nagytengely irányára és méretére

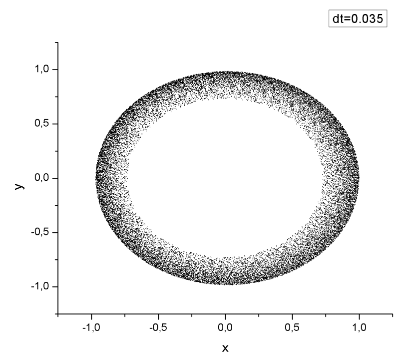
Ennél a feladatnál a fix lépésközös integrálási módszert használtam és fokozatosan növeltem a lépéshoszt és figyeltem, hogyan változik meg a pálya amit a bolygó bejárt.



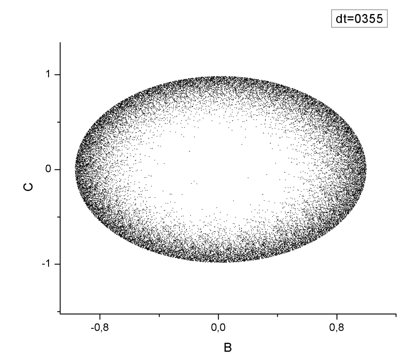
(a) dt=0.005



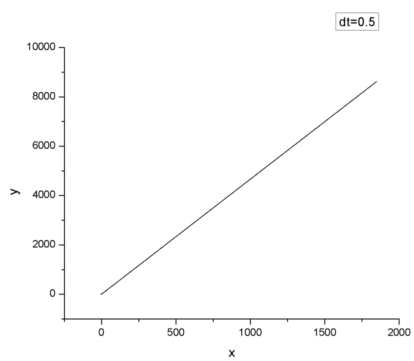
(b) dt=0.03



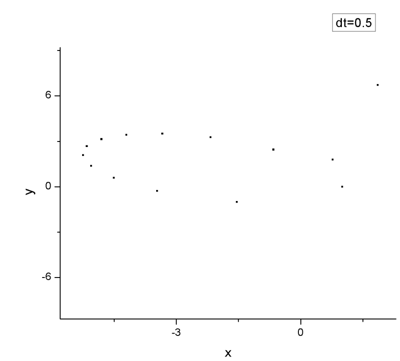
(c) dt=0.035



(d) dt=0.0355



(e) dt=0.5

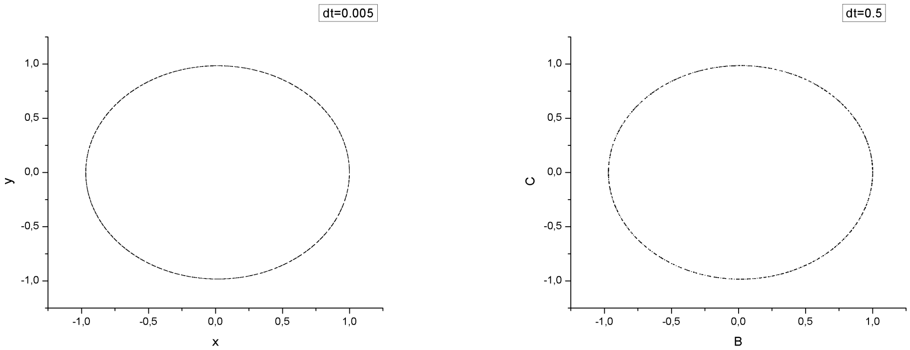


(f) dt=0.5 a 0 körül

1. ábra. A rendszer megváltozása a lépéshosszok modosításval

3.1.2. Az integrálási módszer hatása a nagytengely irányára és méretére

Ennél a feladatnál azonosan változtattam a lépésközöket, mint az elöző feladatnál, de adaptív lépésközzel integráltam. Ebben az esetben is megvizsgáltam az összes lépésközre a problémát, de mivel lényegi változást nem produkált a pálya alakja ennél az integrálási technikánál így csak a legkisebb valamint a legnagyobb lépésközre vizsgált esetet mutatom be.

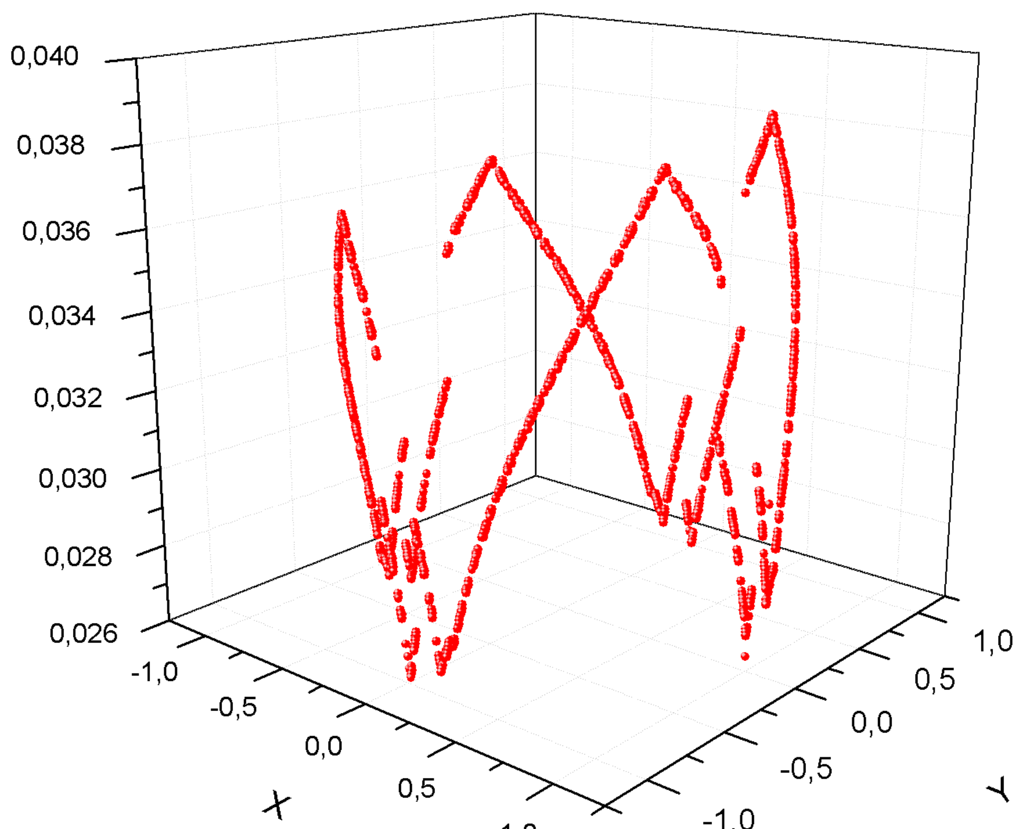


(a) dt=0.005 (b) dt=0.5

2. ábra. A rendszer megváltozása a lépéshosszok modosításval és adaptív lépésközzel

3.2. Adaptív lépéshossz változása a pálya mentén

Ennél a feladatnál azt vizsgáltam, hogy a lépéshossz abban az esetben amikor adaptívan változik, milyen mértékben teszi ezt és hol.



3. ábra. Az lépéshosszok változása a pálya mentén

Azt figyelhettük meg az első kettő mérési feladatban, hogy a lépés szám nagyban befolyásolja a fix lépésközös integrálást, mivel hirtelen változásoknál könnyen elöfordulhat, hogy nem vételez elég mintát és így a mi esetünkben például a bolygók elfognak repülni. Az adaptív lépésközös módszert ez a probléma nem érintette, mivel ahol nagyobb mértékü változás történt ott besürüsödött a lépések száma és ez véget hiába választottam nagy lépés közt a program ahol érzékelte, hogy nagy mértékű változás történik ott besűrűsítette a lépések számát.

3.3. Merkúr perihélium processziója

A Merkúr pályájának az elfordulása az általános relativítás elmélet ad magyarázatot.

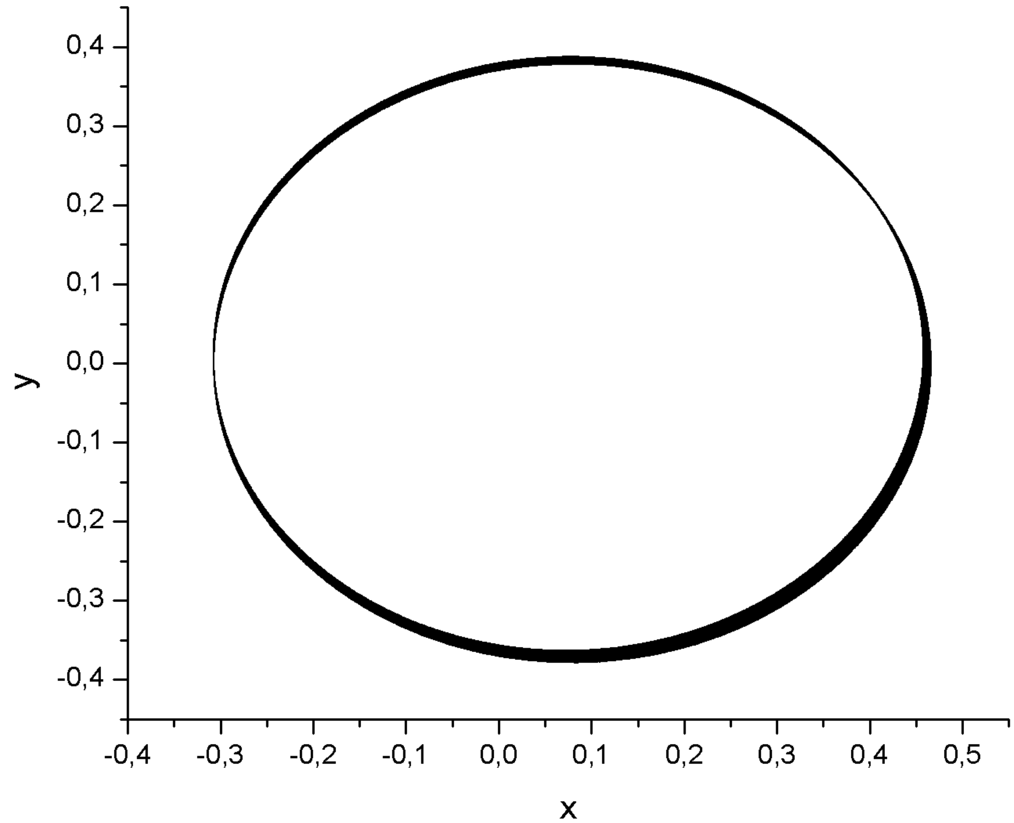
- aphélium távolság: 0.466 AU
- excentricitás:0.2056
- periodus szám: 1000
- adaptív pontosság: 0.00001

Így a következő képpen kell átalakítani a szimuláció egyenletét:

$$F_{12} = -G\frac{m_1 m_2}{r_{12}^2}(1 + \frac{a}{r^2}) \tag{6}$$

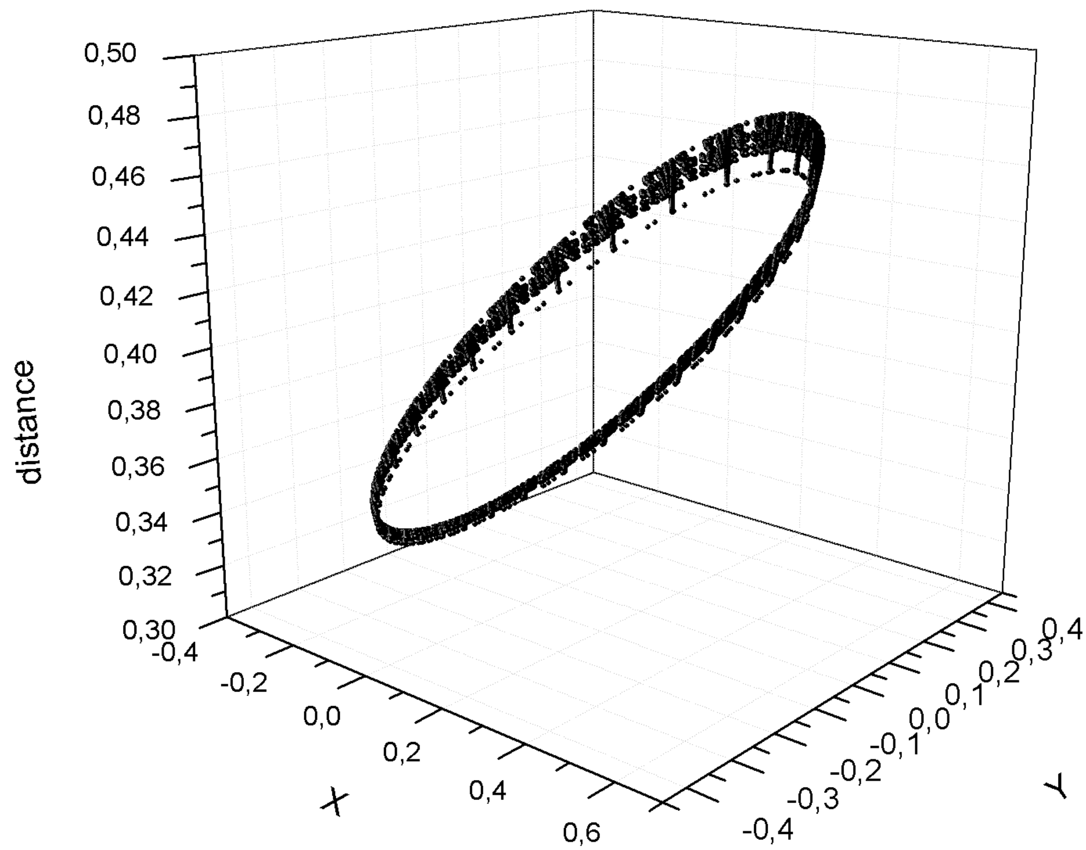
Ezt a program kódban a következő képpen írtam be:
$f[3] = -GmPlusM * r_x/rCubed * (1 + (1.1e - 8/rSquared));$
$f[4] = -GmPlusM * r_y/rCubed * (1 + (1.1e - 8/rSquared));$

A precesszió megfigyelhető a pályénak a két dimenziós ábráján is a vonalak kiszélesedésén is keresztül.



4. ábra. A Merkúr precessziója

Szemléletesebben a következő ábrán láthatjuk a jelenséget ahol a naptól vett távolságot ábrázoltam a helytől függően.



5. ábra. A Merkúr pályájának a függvényében a távolsága a naptól

3.4. Háromtest probléma

Ebben a feladatban a Jupitert és a Földet közösen probáltam meg szimulálni a naprendszerben.
A következő differenciál egyenletekkel írhatjuk le a rendszert:

$$\ddot{x_1} = -Gm_2 \frac{x_1 - x_2}{|x_1 - x_2|^3} - Gm_3 \frac{x_1 - x_3}{|x_1 - x_3|^3} \tag{7}$$

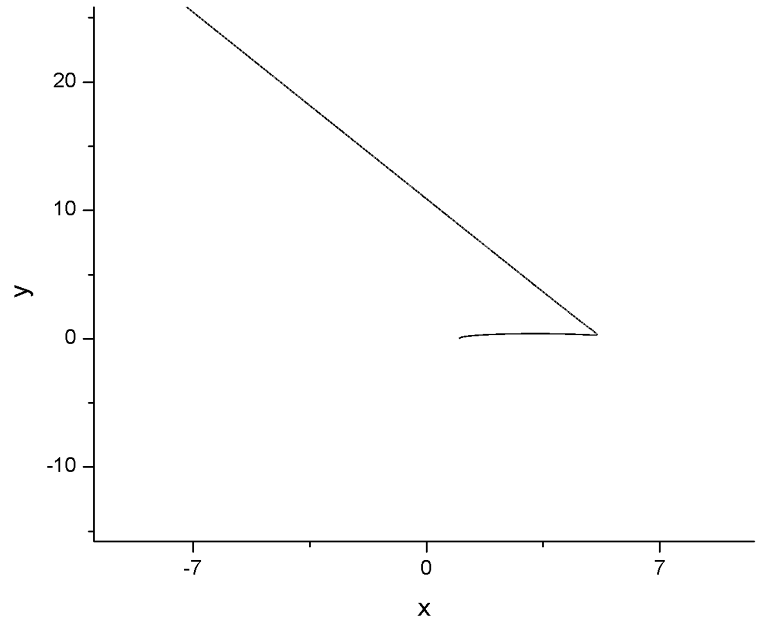
$$\ddot{x_2} = -Gm_3 \frac{x_2 - x_3}{|x_2 - x_3|^3} - Gm_1 \frac{x_2 - x_1}{|x_2 - x_1|^3} \tag{8}$$

$$\ddot{x_3} = -Gm_1 \frac{x_3 - x_1}{|x_3 - x_1|^3} - Gm_2 \frac{x_3 - x_2}{|x_3 - x_2|^3} \tag{9}$$

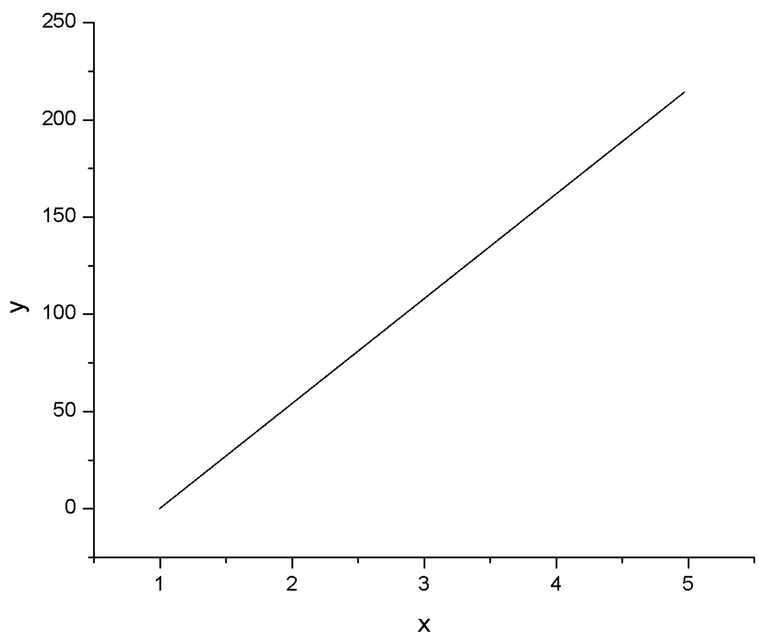
$$\tag{10}$$

A rendszer egyszerübbé tételéhez tételezzük fel hogy a nap tömege annyival nagyobb a Jupite és a Föld tömegénél, hogy fix pontként tekinthetjük ami körül fog keringeni a másik két test.
A Jupiter adatai:

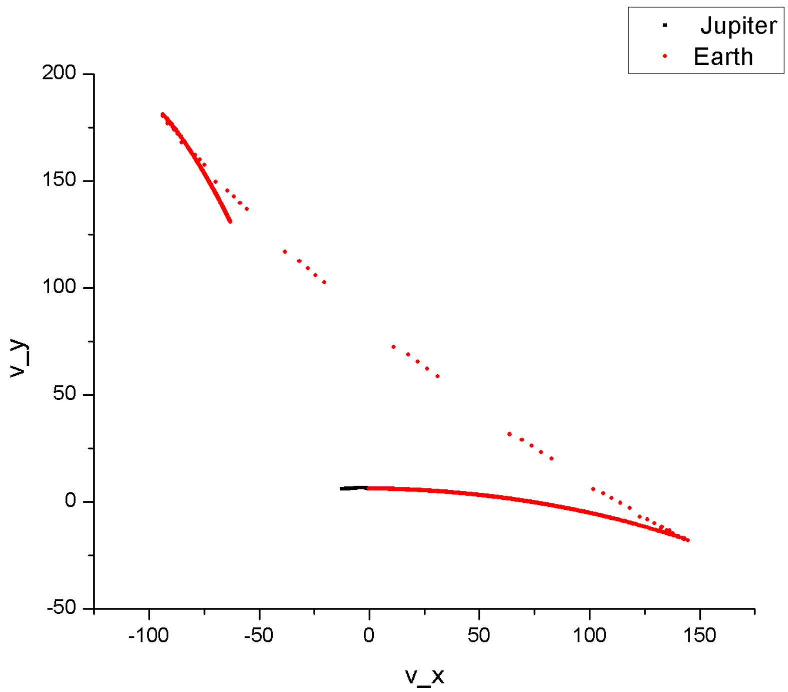
- aphélium távolság: 5,5 AU
- excentricitás:0.05
- tömeg a föld tömegének számszorosával kifejezve: 318



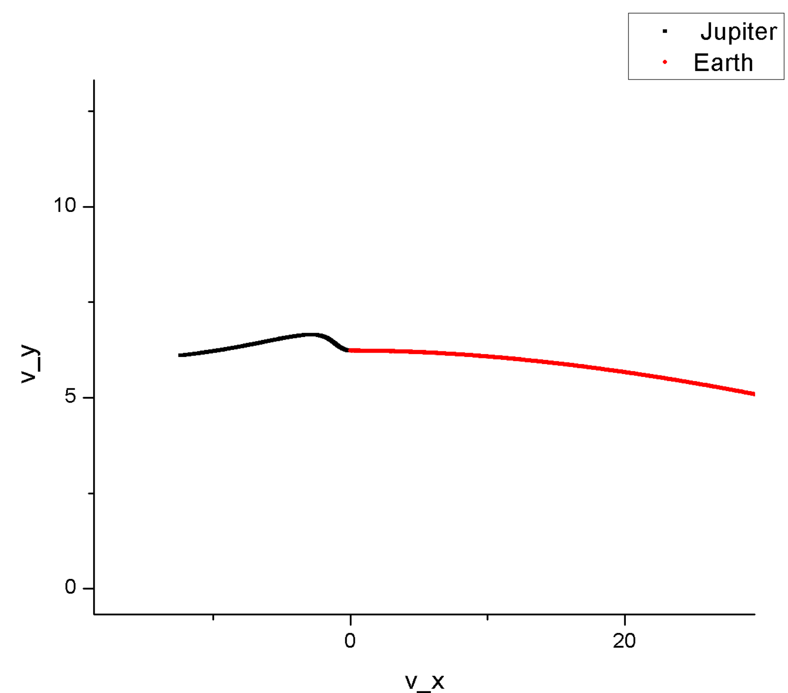
6. ábra. A Föld pályája a Jupiter jelenlétében



7. ábra. A Jupiter pályája a Föld jelenlétében



8. ábra. A Jupiter és a Föld sebességének alakulása



9. ábra. A Jupiter és a Föld sebességének alakulása a 0 körül

4. Függelék

4.1. Három test probléma kód

```
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iostream>
using namespace std;

#include "vector.hpp"
#include "odeint.hpp"
using namespace cpl;

const double pi = 4 * atan(1.0);
const double GmPlusM = 4 * pi * pi;

bool switch_t_with_y = false;    //  to interpolate to y = 0

//  1. testre az egyenlet
Vector f(const Vector& x) {
    double t = x[0], r_x = x[1], r_y = x[2], v_x = x[3], v_y = x[4], r_x2=x[5], r_y2=x[6], mass=x[9], ma
    double rSquared = r_x*r_x + r_y*r_y;
    double rCubed = rSquared * sqrt(rSquared);
    double rSquared2 = (r_x-r_x2)*(r_x-r_x2) + (r_y-r_y2)*(r_y-r_y2);
    double rCubed2 = rSquared2 * sqrt(rSquared2);
    Vector f(5);
    f[0] = 1;
    f[1] = v_x;
    f[2] = v_y;
    f[3] = - GmPlusM * r_x / rCubed- GmPlusM *mass2* (r_x-r_x2) / rCubed2;
    f[4] = - GmPlusM * r_y / rCubed- GmPlusM *mass2* (r_y-r_y2) / rCubed2;
    if (switch_t_with_y) {
        //  use y as independent variable
        for (int i = 0; i < 5; i++)
            f[i] /= v_y;
    }
    return f;
}
//  2. testre az egyenlet
Vector g(const Vector& x) {
    double t = x[0], r_x = x[5], r_y = x[6], v_x = x[7], v_y = x[8], r_x2=x[1], r_y2=x[2], mass=x[9], ma
    double rSquared = r_x*r_x + r_y*r_y;
    double rCubed = rSquared * sqrt(rSquared);
    double rSquared2 = (r_x-r_x2)*(r_x-r_x2) + (r_y-r_y2)*(r_y-r_y2);
    double rCubed2 = rSquared2 * sqrt(rSquared2);
    Vector g(5);
    g[0] = 1;
    g[1] = v_x;
    g[2] = v_y;
    g[3] = - GmPlusM * r_x / rCubed- GmPlusM *mass* (r_x-r_x2) / rCubed2;
    g[4] = - GmPlusM * r_y / rCubed- GmPlusM *mass* (r_y-r_y2) / rCubed2;
    if (switch_t_with_y) {
        //  use y as independent variable
        for (int i = 0; i < 5; i++)
            g[i] /= v_y;
```

```
    }
    return g;
}

//  Change independent variable from t to y and step back to y = 0
void interpolate_crossing(Vector x, int& crossing) {
    ++crossing;
    switch_t_with_y = true;
    RK4Step(x, -x[2], f);
    cout << " crossing " << crossing << "\t t = " << x[0]
         << "\t x = " << x[1] << endl;
    switch_t_with_y = false;
}

int main() {
    cout << " Kepler orbit comparing fixed and adaptive Runge-Kutta\n"
         << " -----------------------------------------------------\n"
         << " Enter aphelion distance in AU, and eccentricity and mass: ";
    double r_ap, eccentricity, a, T, v0, m;
    cin >> r_ap >> eccentricity >> m;
    a = r_ap / (1 + eccentricity);
    T = pow(a, 1.5);
    v0 = sqrt(GmPlusM * (2 / r_ap - 1 / a));
    cout << " Enter number of periods, step size, and adaptive accuracy: ";
    double periods, dt, accuracy;
    cin >> periods >> dt >> accuracy;
    cout<< "bonus body dist and ecc. and mass:";
    double x2, y2, v02, a2, m2;
    cin >> x2 >> y2>> m2;
    a2 = x2 / (1 + y2);
    v02 = sqrt(GmPlusM * (2 / r_ap - 1 / a2));
    Vector x0(11);
    x0[0] = 0;  x0[1] = r_ap;  x0[2] = 0;  x0[3] = 0;  x0[4] = v0;
    x0[5] = x2; x0[6] = 0; x0[7] =0; x0[8] = v02; x0[9] = m; x0[10] = m2;

    ofstream dataFile("fixed.data");
    Vector x = x0;
    Vector y =x0;
    int steps = 0, crossing = 0;
    cout << "\n Integrating with fixed step size" << endl;
    do {
        for (int i = 0; i < 5; i++)
            dataFile << x[i] << '\t';
        for (int i = 1; i < 5; i++)
            dataFile << y[i] << '\t';
        double q = x[2];
         double w = y[2];
        dataFile<<endl;
        RK4Step(x, dt, f);
        RK4Step(y, dt, g);
        ++steps;
        if (q * x[2] < 0)
            interpolate_crossing(x, crossing);
        if (w * y[2] < 0)
            interpolate_crossing(y, crossing);
```

```
    } while (x[0] < periods * T);
    cout << " number of fixed size steps = " << steps << endl;
    cout << " data in file fixed.data" << endl;
    dataFile.close();


}
```

Hivatkozások

[1] Jegyzet
$https://stegerjozsef.web.elte.hu/teaching/szamszim/bolygo.pdf$

[2] Forráskód
$https://stegerjozsef.web.elte.hu/teaching/szamszim/kepler.tgz$

[3] 3 test
$https://en.wikipedia.org/wiki/Three-body_problem$

[4] Lagrange pontok
$https://en.wikipedia.org/wiki/Lagrangian_point$