# Harmonikus oszcillátor

Tuhári Richárd

2018. március 11.

## Tartalomjegyzék

**1. Bevezető** **3**

**2. Elmélet** **3**
2.1. Analitikus tárgyalás . . . 3
2.1.1. A diffegyenletek . . . 3
2.1.2. A tehetetlenségi tenzor . . . 4
2.1.3. Fizikai inga . . . 5
2.2. Numerikus tárgyalás . . . 5
2.2.1. A bemeneti paraméterek: . . . 5
2.2.2. A kimeneti paraméterek: . . . 6

**3. Kiértékelés** **6**
3.1. Kitérés-, sebesség-, energia-idő diagrammok . . . 7
3.2. Fázistér diagram . . . 10

**4. A korábbi közelítések érzékenységének vizsgálata, különböző numerikus differenciálegyenlet megold**
4.1. Euler-módszer . . . 12
4.2. Euler-Cromer módszer . . . 12
4.3. Runge-kutta módszer . . . 13
4.4. Lépéshossz váktoztatással . . . 13

**5. Kettős inga** **15**

**6. Diszkusszió** **18**

**7. Hivatkozások** **18**

**8. Függelék** **18**
8.1. Adaptív Runge-Kutta . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

# 1. Bevezető

A szimuláció célja, az ingamozgás különböző közelítéseinek modellezése valamint az egyes kitérés-idő, sebesség-idő, energia-idő és kitérés-sebesség diagrammok ábrázolása, a közelítések különböző numerikus differenciálegyenletmegoldó módszerekre való érzékenységének vizsgálata. További feladat egy kettős inga szimulációja, a trajektóriák és fázisterek különböző beállítás-paraméterek melletti vizsgálata és az átbillenési jelenség fázisteréne rekonstrukciójára folytatott kísérlet.

# 2. Elmélet

## 2.1. Analitikus tárgyalás

### 2.1.1. A diffegyenletek

A Lagrange függvény:

$$L = \frac{m}{2}\ell^2\dot{\phi}^2 + mg\ell\cos\phi \tag{1}$$

Megoldva az Euler-Lagrange-t:

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\phi}} = \frac{\partial L}{\partial\phi} \;\rightarrow\; m\ell^2\ddot{\phi} = -mg\ell\sin\phi \;\rightarrow\; \ddot{\phi} = -\frac{g}{\ell}\sin\phi \tag{2}$$

Kicsi x-re sinx=x

$$\ddot{\phi} = -\frac{g}{\ell}\phi \tag{3}$$

Sűrű közere a Rayleigh függvény

$$R = \frac{\gamma}{2} \mid v \mid^2 = \frac{\gamma}{2}\ell^2 \mid \dot{\phi} \mid^2 \tag{4}$$

Megoldva az Lagrange-Rayleigh-t:

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\phi}} = \frac{\partial L}{\partial\phi} - \frac{\partial R}{\partial\dot{\phi}} \;\rightarrow\; m\ell^2\ddot{\phi} = -mg\ell\sin\phi - \gamma\ell^2\dot{\phi} \;\rightarrow\; \ddot{\phi} = -\frac{g}{\ell}\sin\phi - \frac{\gamma}{m}\dot{\phi} \tag{5}$$

Mindez harmonikus gerjesztéssel:

$$\ddot{\phi} = -\frac{g}{\ell}\sin\phi - \frac{\gamma}{m}\dot{\phi} + F_D\sin(\Omega_D t) \tag{6}$$

Oldjuk meg a differenciálegyenletet Green-függvény segítségével:

$$\ddot{\phi} + \frac{\gamma}{m}\dot{\phi} + \frac{g}{\ell}\phi = \frac{F(t)}{m} = f(t) \tag{7}$$

Legyen $\frac{\gamma}{m} = \alpha$ és $\frac{g}{\ell} = \omega_0^2$

$$\ddot{\phi} + \alpha\dot{\phi} + \omega_0^2\phi = f(t) \tag{8}$$

A Fourier transzformáltak:

$$\int_{-\infty}^{\infty} \omega_0^2\phi e^{-i\omega t}dt = \omega_0^2\widetilde{\phi}(\omega) \tag{9}$$

$$\int_{-\infty}^{\infty} \alpha\dot{\phi} e^{-i\omega t} dt = i\omega\alpha\widetilde{\phi}(\omega) \tag{10}$$

$$\int_{-\infty}^{\infty} \ddot{\phi} e^{-i\omega t} dt = -\omega^2\widetilde{\phi}(\omega) \tag{11}$$

Tehát:

$$(-\omega^2 + i\alpha\omega + \omega_0^2)\widetilde{\phi}(\omega) = \widetilde{f}(\omega) \tag{12}$$

f(t)=$\delta$(t)-re $\rightarrow$ $\widetilde{\delta}(\omega) = 1$ , $\widetilde{G}(\omega) = \frac{1}{-\omega^2+i\alpha\omega+\omega_0^2}$
ekkor a Green-függvény:

$$G(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \widetilde{G}(\omega) e^{i\omega t} d\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty} \frac{e^{i\omega t}}{-\omega^2 + i\alpha\omega + \omega_0^2} d\omega = * \tag{13}$$

A pólusok:

$$\omega_{1,2} = \frac{i\alpha \pm \sqrt{-\alpha^2 + 4\omega_0^2}}{-2} = -\frac{i\alpha}{2} \pm \sqrt{\omega_0^2 - (\frac{\alpha}{2})^2} \tag{14}$$

$\omega_0^2 - (\frac{\alpha}{2})^2$ legyen $= \omega_\alpha$

$$Res(f, \omega_1) = \lim_{\omega\to\omega_1} (\omega - \omega_1) \frac{e^{i\omega t}}{-(\omega - \omega_1)(\omega - \omega_2)} = \frac{e^{-\frac{\alpha}{2}t} e^{i\omega_\alpha}}{-2\omega_\alpha} \tag{15}$$

ugyan így számolva:

$$Res(f, \omega_2) = \frac{e^{-\frac{\alpha}{2}t} e^{-i\omega_\alpha}}{2\omega_\alpha} \tag{16}$$

$$* = \frac{1}{2\pi} 2\pi i \frac{e^{-\frac{\alpha}{2}t}}{2\omega\alpha} (e^{-i\omega_\alpha t} - e^{i\omega_\alpha t}) = \frac{e^{-\frac{\alpha}{2}}}{\omega_\alpha} \sin(\omega_\alpha t) \tag{17}$$

Konvolválva f(t)-t G(t)-vel, megkapjuk $\phi$(t)-t:

(18)

#### 2.1.2. A tehetetlenségi tenzor

Steiner tétel:

$$\Theta_{ij} = \sum_k m_k (r_{(k)}^2 \delta_{ij} - r_{(k)i} r_{(k)j}) \tag{19}$$

A tenzor számítása:

$$\Theta_{ij} = \int d^3r (r^2\delta_{ij} - r_i r_j) \tag{20}$$

Hogy megkapjuk a tehetelenségi tenzort a rúd végén először ki kell számolnunk a rúdra középen:

$$\int_m z^2 dm = \rho \int_V z^2 dV = \rho A \int_{-\frac{L}{2}}^{\frac{L}{2}} z^2 dz = \rho A [\frac{z^3}{3}]_{-\frac{L}{2}}^{\frac{L}{2}} = \frac{1}{12}(\rho A L)L^2 = \frac{1}{12} m h^2 \tag{21}$$

Állított rúd tkp-ja $\frac{L}{2}$-nél van tehát legyen $\underline{a}$:

$$\underline{a} = \begin{pmatrix} \frac{L}{2} \\ 0 \\ 0 \end{pmatrix} \tag{22}$$

$$\underline{a} \circ \underline{a} = \begin{pmatrix} \frac{L^2}{4} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{23}$$

$$\mathbb{1}\underline{a}^2 = \begin{pmatrix} \frac{L^2}{4} & 0 & 0 \\ 0 & \frac{L^2}{4} & 0 \\ 0 & 0 & \frac{L^2}{4} \end{pmatrix} , \tag{24}$$

Ezek után a rúd végére a Steiner-tétellel:

$$\Theta = \frac{ML^2}{12}\mathbb{1} + M(\mathbb{1}\underline{a}^2 - \underline{a} \circ \underline{a}) = ML^2(\frac{1}{12} + \frac{1}{4}) = \frac{ML^2}{3} \tag{25}$$

#### 2.1.3. Fizikai inga

Található olyan matematikai inga, melynek lengésideje megegyezik egy fizikai ingáéval. Ebben az inga redukált hossza lesz a segítségünkre.

$$\ell_0 = \frac{\Theta}{mL} , \tag{26}$$

ahol $\Theta$ a merev test adott tengelyre vonatkozó tehetetlenségi nyomatéka. Ez merev rúd végpontjára $\frac{m\ell^2}{3}$

Tehát a programokban általánosan használt (8)-as hossz, redukált értéke: $\ell_0 = \frac{8}{3}$

### 2.2. Numerikus tárgyalás

A tárgy honlapjáról letöltött mintaprogram alapesetben adaptív Runge-Kutta módszerrel dolgozik, melynek hibája $\mathcal{O}(\tau^5)$.

#### 2.2.1. A bemeneti paraméterek:

```
cout<< " Length of pendulum L: ";
    cin >> L;
    cout<< " Enter damping coefficient q: ";
    cin >> q;
    cout << " Enter driving frequencey Omega_D: ";
    cin >> Omega_D;
    cout << " Enter driving amplitude F_D: ";
    cin >> F_D;
```

```
cout << " Enter theta(0) and omega(0): ";
double theta, omega, tMax;
cin >> theta >> omega;
cout << " Enter integration time t_max: ";
cin >> tMax;
```

Illetve még, hogy a rendszer lineáris-e, vagy sem.

#### 2.2.2. A kimeneti paraméterek:

Minimális átalakítás után, már az energiákat is tartalmazva:

```
dataFile << t << '\t' << theta << '\t' << omega << '\t' << e <<'\n';
```

Ezek egy data fájlba íródnak ki, azokkal dolgozhatunk tovább.

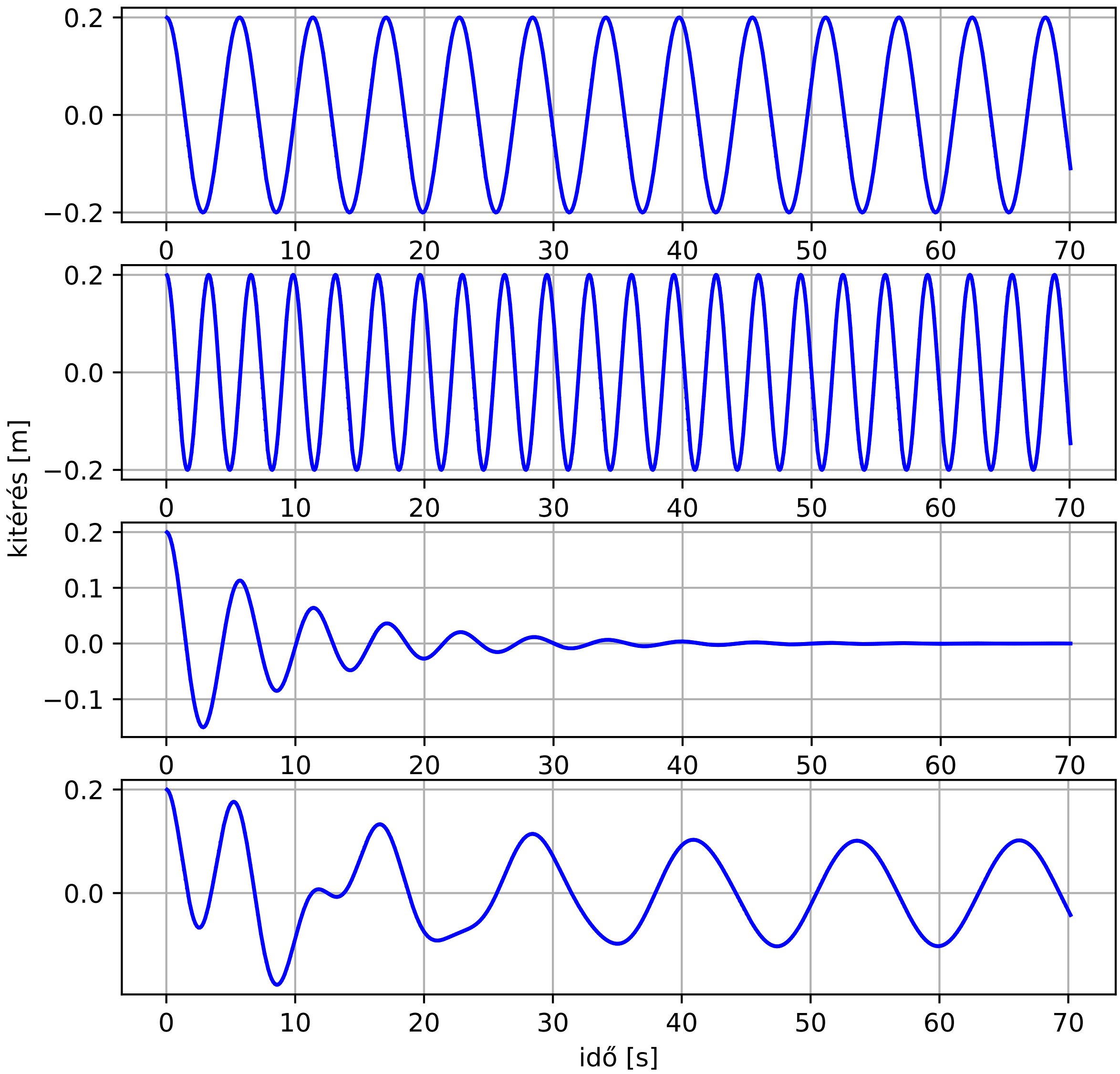
## 3. Kiértékelés

A következőkben csillapítás nélküli, csillapított, gerjesztett-csillapított matematikai inga és egy csillapítás nélküli fizikai inga kitérés-idő, sebesség idő és energia-idő diagrammját láthatjuk.

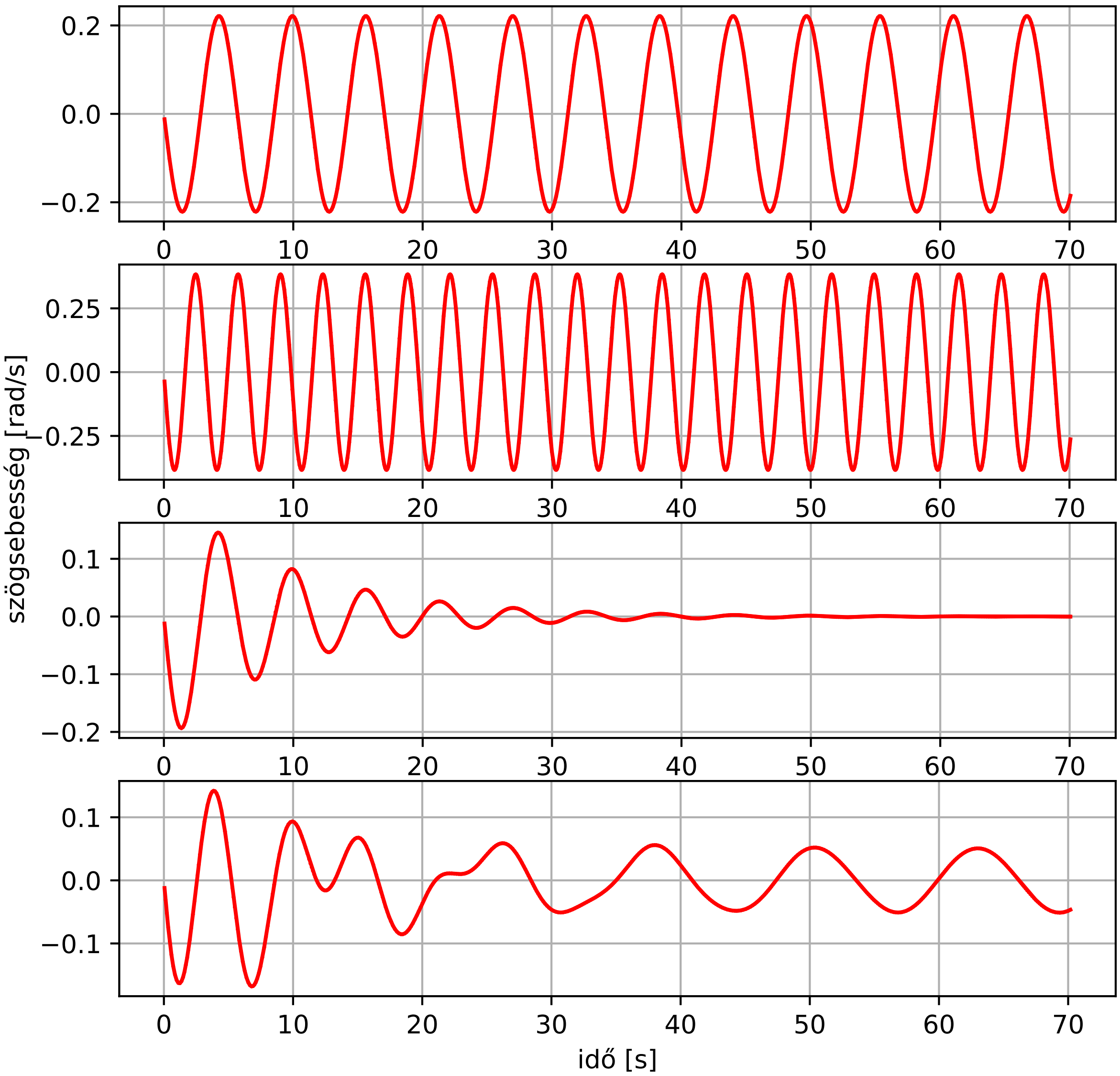
Az általam megadott paraméterek (matematikai inga/csillapított/gerjesztett-csillapított/fizikai inga):

```
Enter linear or nonlinear: linear/nonlinear
Length of pendulum L: 8/8/8/2.6666
Enter damping coefficient q: 0/0.2/0.2/0
Enter driving frequencey Omega_D: 0/0/0.5/0
Enter driving amplitude F_D: 0/0/0.2/0
Enter theta(0) and omega(0): 0.2/0.2/0.2/0.2
Enter integration time t_max: 70/70/70/70
```

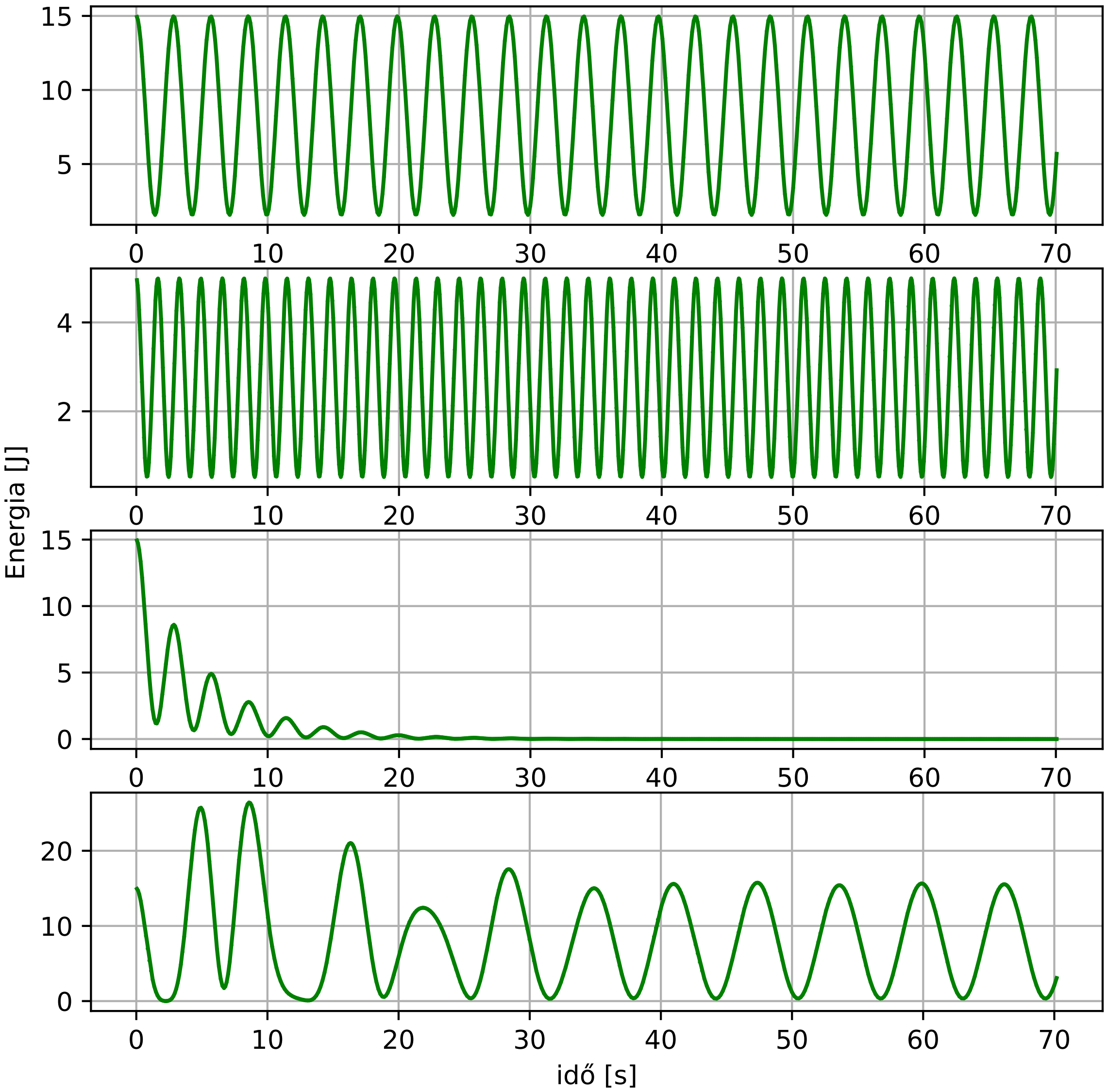
### 3.1. Kitérés-, sebesség-, energia-idő diagrammok




1. ábra. Kitérés-idő diagramm

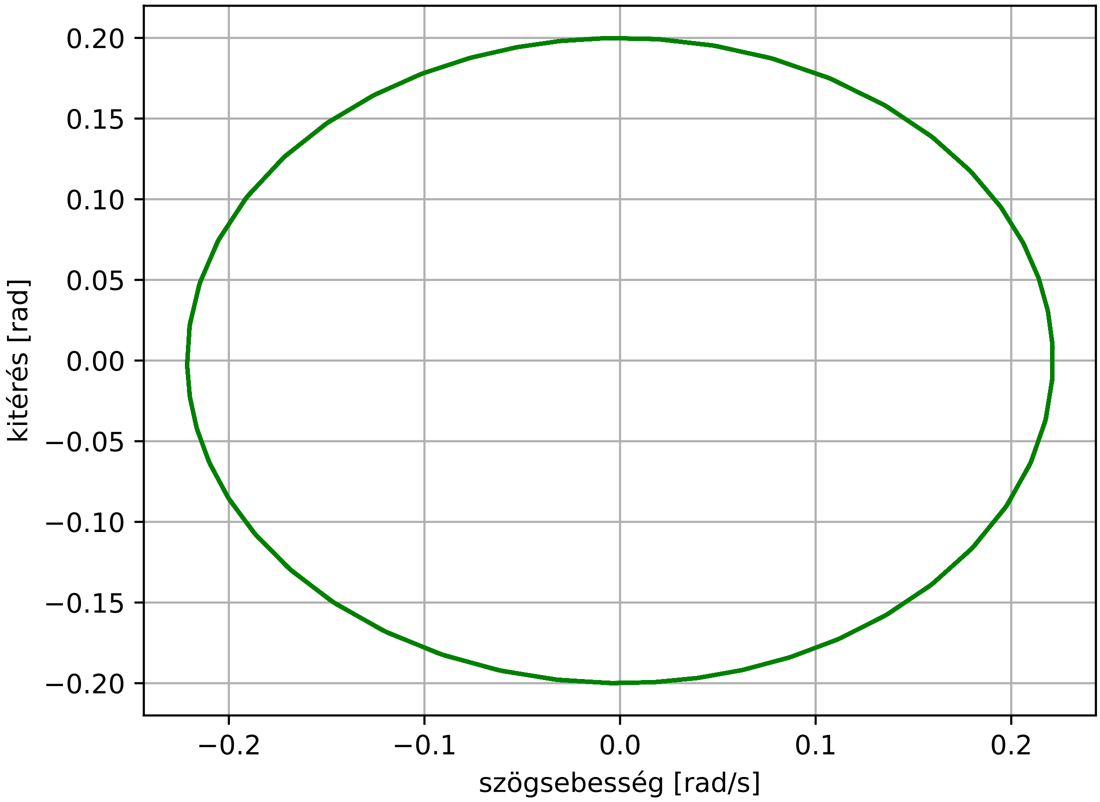



2. ábra. Sebeeség-idő diagramm

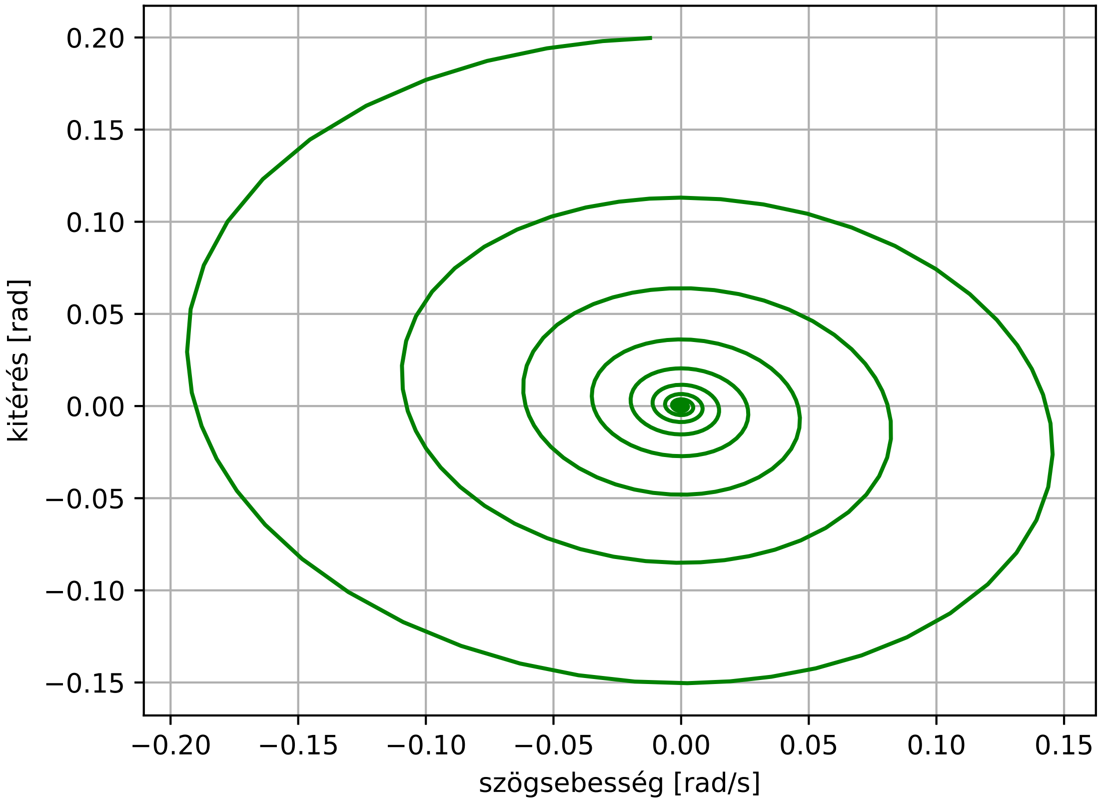



3. ábra. Energia-idő diagramm

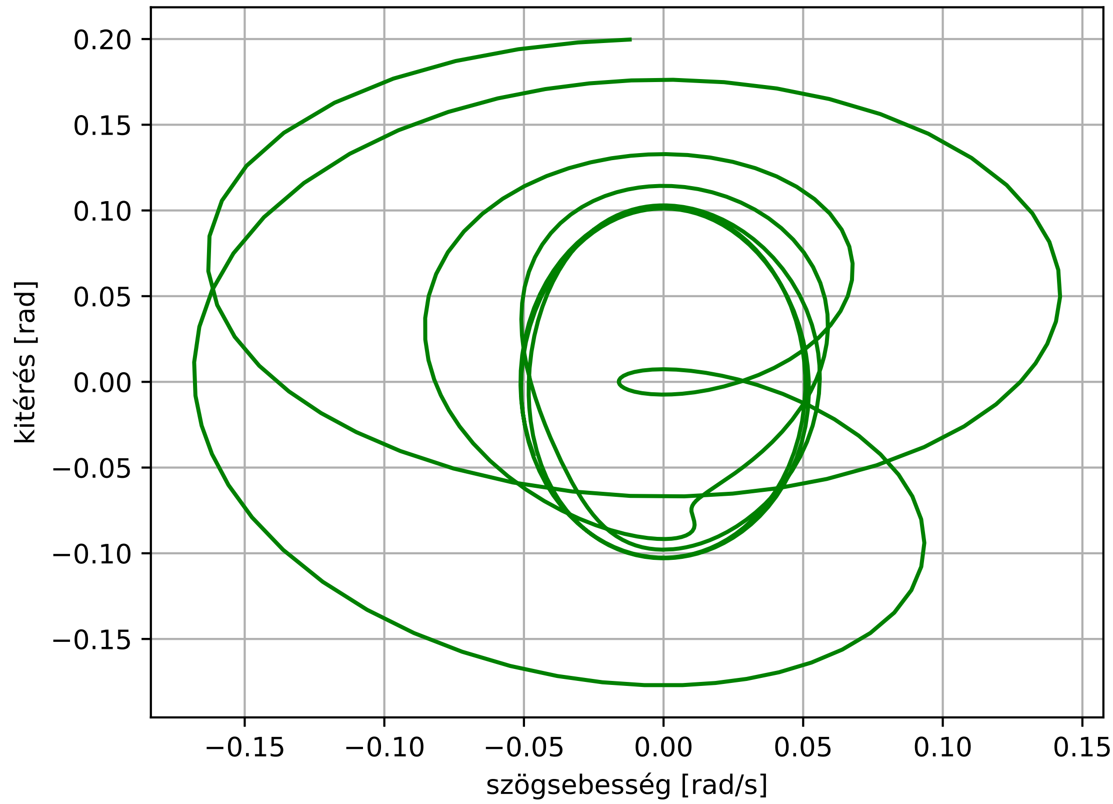
### 3.2. Fázistér diagram




4. ábra. Kitérés-sebesség diagramm csillapítatlan inga esetén




5. ábra. Kitérés-sebesség diagramm csillapított inga esetén




6. ábra. Kitérés-sebesség diagramm gerjesztett-csillapított inga esetén

# 4. A korábbi közelítések érzékenységének vizsgálata, különböző numerikus differenciálegyenlet megoldó módszerekre

## 4.1. Euler-módszer

```
void Euler (double dt) {
    double a = - omega * omega * x;
    v += a * dt;
    x += v * dt + x;
}
}
```

## 4.2. Euler-Cromer módszer

```
void EulerCromer (double dt) {
```

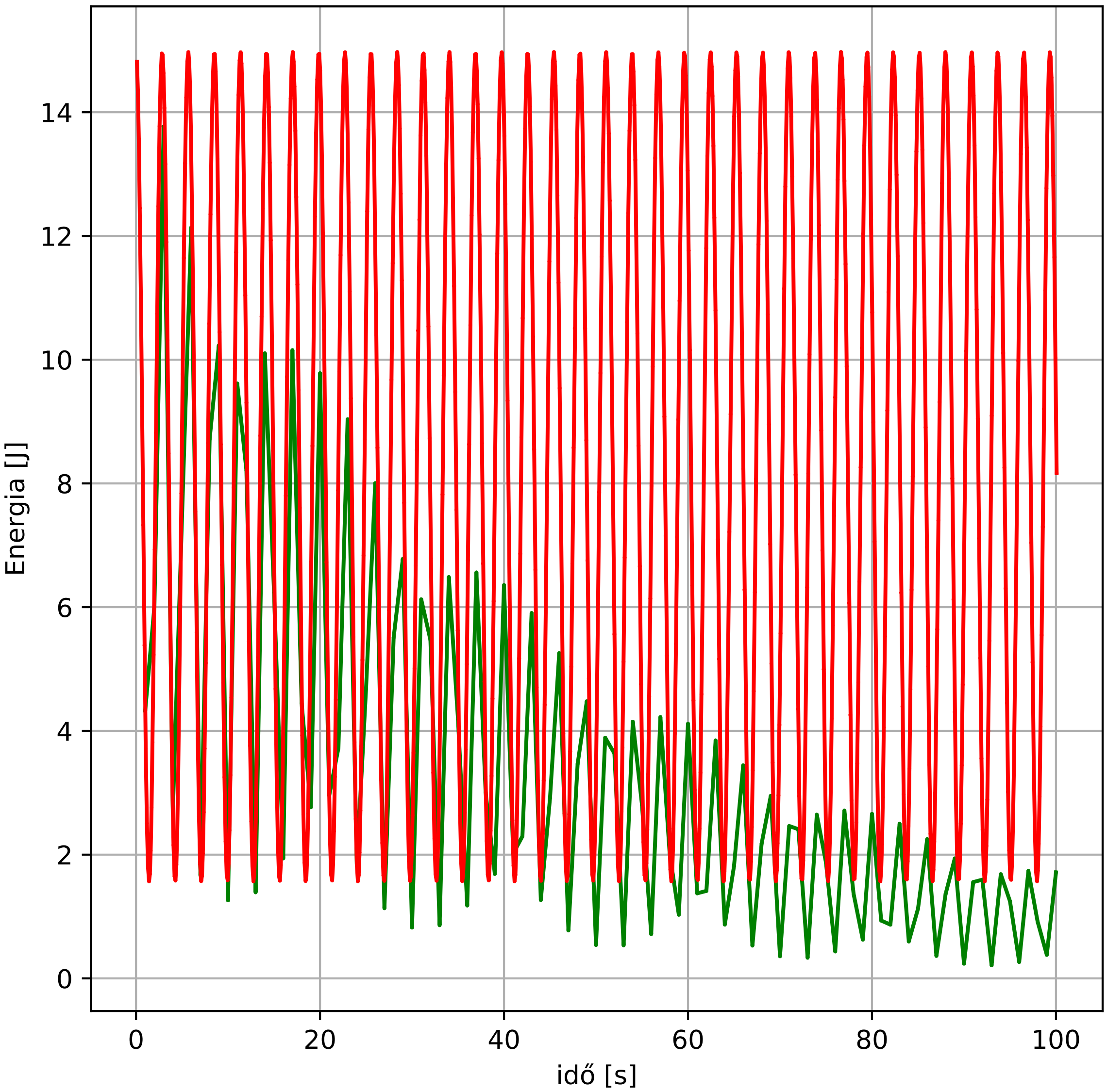
```
    double a = - omega * omega * x;
    v += a * dt;
    x += v * dt;
}
}
```

### 4.3. Runge-kutta módszer

```
void RK4Step(Vector& x, double tau,
             Vector derivs(const Vector&))
{
    Vector k1 = tau * derivs(x);
    Vector k2 = tau * derivs(x + 0.5 * k1);
    Vector k3 = tau * derivs(x + 0.5 * k2);
    Vector k4 = tau * derivs(x + k3);
    x += (k1 + 2 * k2 + 2 * k3 + k4) / 6.0;
}
```

### 4.4. Lépéshossz váktoztatással

Ezen kód, a hosszúsága miatt, a függelékben megtalálható: 8.1




7. ábra. Az adaptív Runge-Kutta (piros)viszonya a simáéhoz (zöld)

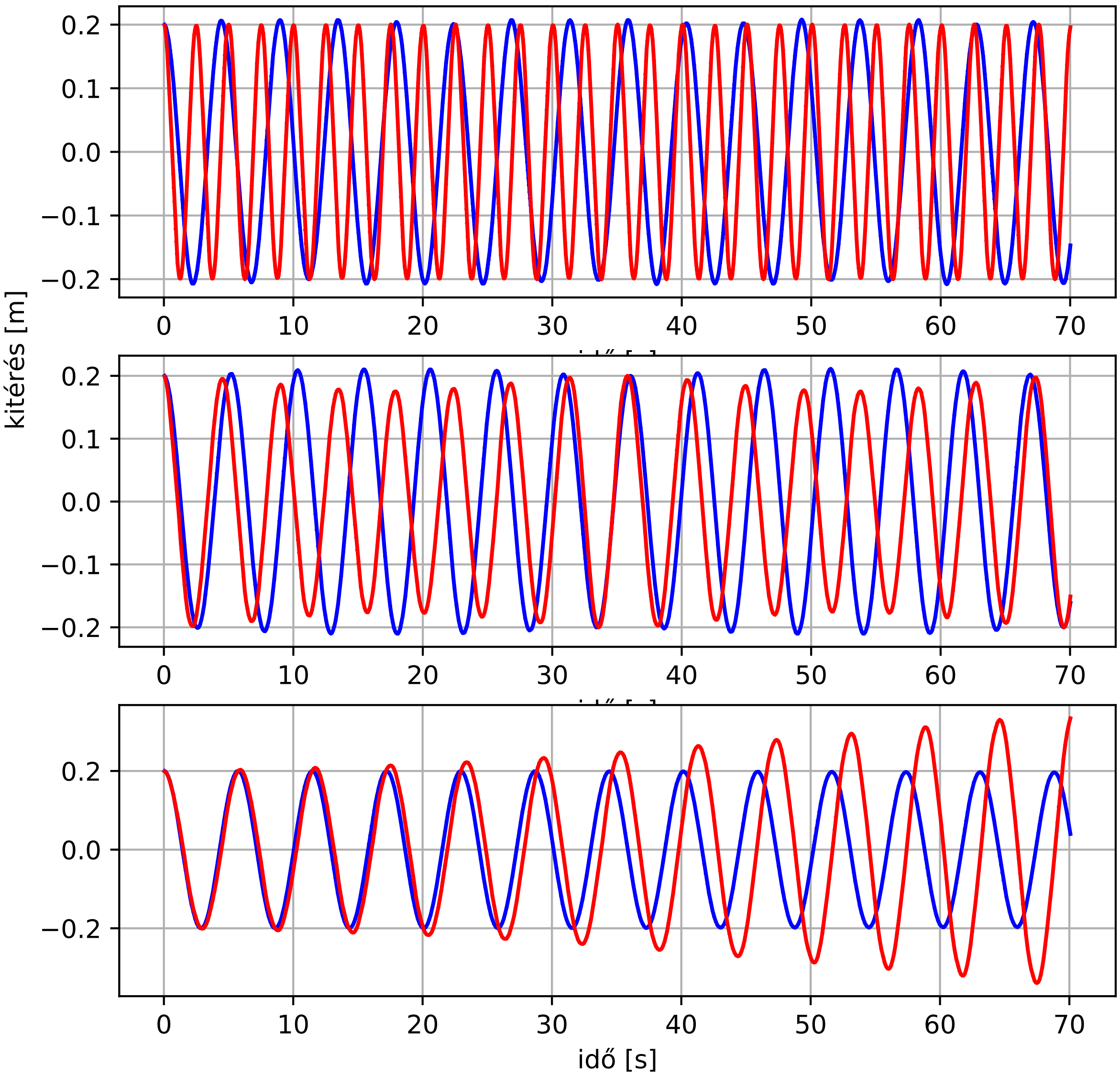
## 5. Kettős inga

Itt a kettős ingát próbáltam meg szimulálni, kiegészítve a laborbéli programot, a mozgásegyenlet átírásával (kettőre) és az eddig is használ változók bevezetésére, az új ingára is. Mivel itt is figyelembe lehetett venni a redukált hosszt, nem lett bonyolult az átalakítás. Példának okául, ha a két ingát megegyezőnek vesszük, akkor a diffegyenletek, persze később ahol lehet, élve a sinx=x közelítéssel:

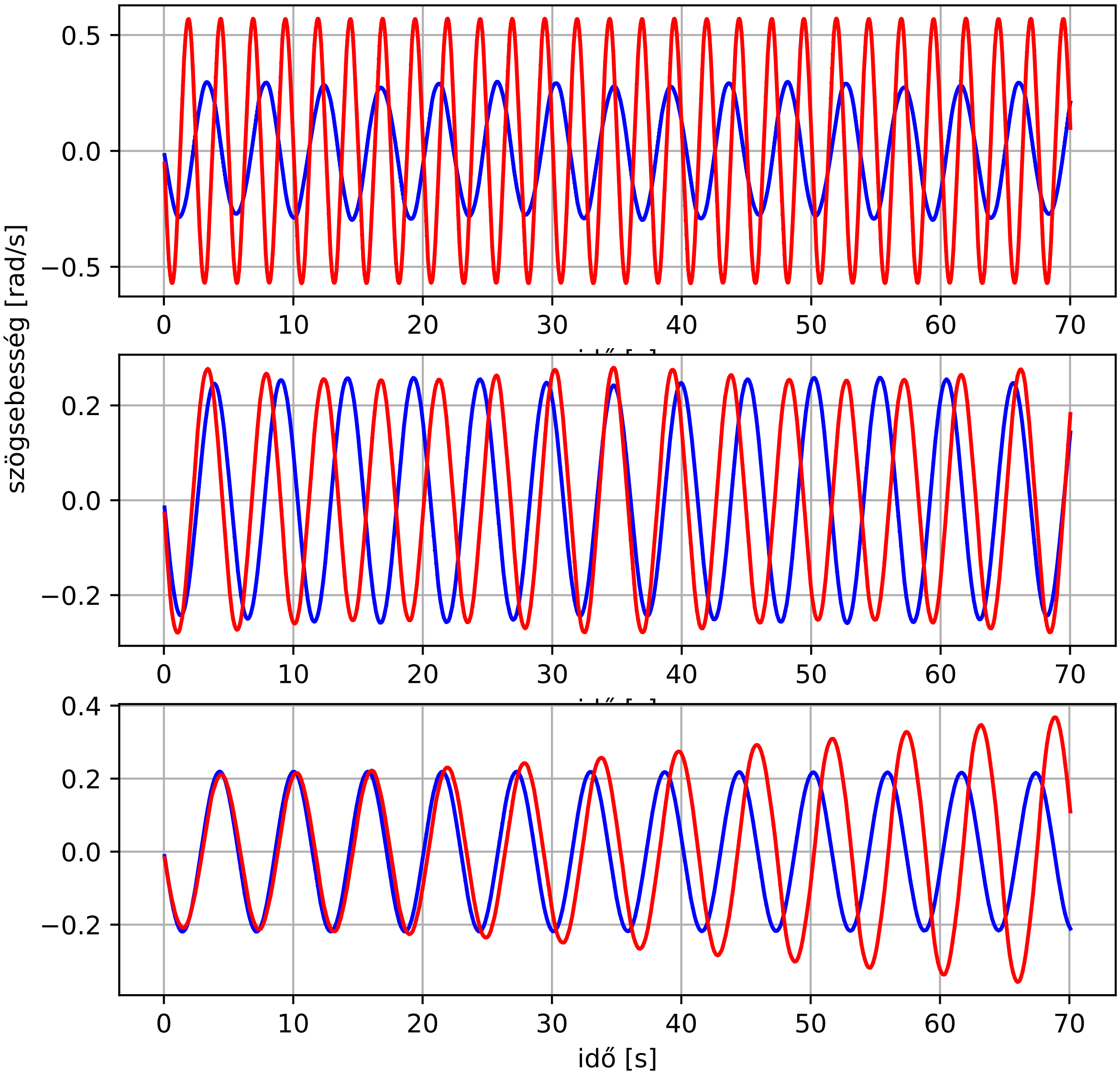
```
    f[2]=(-2*l2*l2*omega1*omega2*sin(alpha-beta)-3*(g)*(l1+2*l2)*sin(alpha)+
-2*l2*l2*(f[5]*cos(\alpha-beta)-omega2*sin(alpha-beta)(omega1-omega2)))/(5*l1*l1);

    f[5] = (2*omega1*omega2*sin(alpha-beta)-(g/l2)*sin(beta)+
-2*(f[2]*cos(alpha-beta)-omega1*sin(alpha-beta)(omega1-omega2)));
```

Három különböző esetet vizsgáltam:

- Az első rövid, a csatolt hosszú
- Mindkettő ugyanannyira hosszú
- Az első hosszú, a csatolt rövid




8. ábra. A kettős inga kitérés-idő diagrammja (1. kék, 2.piros)




9. ábra. A kettős inga sebesség-idő diagrammja (1. kék, 2.piros)

# 6. Diszkusszió

Tekintsük előszöt a 3-as ábrát. Látható, hogy csillapított rezgésnél nagyon szépen eltűnik az energia, és gerjesztett-csillapítottnál is gyönyörűen látszik , hogy a rendszer egy idő után elfelejti a kiindulási állapotát (habár ez a kitérés időnél még jobban :1). Az, hogy külső behatás nélküli rendszeben, a matematikai és a fizikai ingánál (első kettő) miért oszcillál és nem konstans, az valószínűleg a numerikus módszerből adódik. A numerikus módszerek összehasonlításánál jól látszik 9, hogy elég nagy(vagy megfelelően kicsi) lépésköznél a sima Runge-Kutta már nem tartja az energiát. A kettős ingánál észrevehető, hogy elég könnyen kaotikussá tud válni a rendszer, viszont az is, hogy ha az első ingát kicsinek vesszük a csatolthoz képest, akkor az rendes ingaként fog működni, amit ha jobban rápillantunk a diffegyenleteire amúgy várunk is.

# 7. Hivatkozások


https://stegerjozsef.web.elte.hu/teaching/szamszim/index.php


# 8. Függelék

## 8.1. Adaptív Runge-Kutta

```
void adaptiveRK4Step(Vector& x, double& tau, double accuracy,
                     Vector derivs(const Vector&))
{
    const double SAFETY = 0.9, PGROW = -0.2, PSHRINK = -0.25,
                 ERRCON = 1.89E-4, TINY = 1.0e-30;
    int n = x.dimension();
    Vector x_half(n), x_full(n), Delta(n);
    Vector scale = derivs(x);
    for (int i = 0; i < n; i++)
        scale[i] = abs(x[i]) + abs(scale[i] * tau) + TINY;
    double err_max;
    while (true) {
        // take two half steps
        double tau_half = tau / 2;
        x_half = x;
        RK4Step(x_half, tau_half, derivs);
        RK4Step(x_half, tau_half, derivs);
        // take full step
        x_full = x;
        RK4Step(x_full, tau, derivs);
        // estimate error
        Delta = x_half - x_full;
        err_max = 0;
        for (int i = 0; i < n; i++)
            err_max = max(err_max, abs(Delta[i]) / scale[i]);
        err_max /= accuracy;
        if (err_max <= 1.0)
```

```
                break;
            double tau_temp = SAFETY * tau * pow(err_max, PSHRINK);
            if (tau >= 0.0)
                tau = max(tau_temp, 0.1 * tau);
            else
                tau = min(tau_temp, 0.1 * tau);
            if (abs(tau) == 0.0) {
                cerr << "adaptiveRK4Step: step size underflow\naborting ..."
                     << endl;
                exit(EXIT_FAILURE);
            }
        }
        tau *= (err_max > ERRCON ? SAFETY * pow(err_max, PGROW) : 5.0);
        x = x_half + Delta / 15.0;
}
```