Harmonikus oszcillátor

Tuhári Richárd

2018. február 25.

Tartalomjegyzék

1. Bevezető

A jegyzőköny a harmonikus oszcillátor szimulációját hivatott tárgyalni. Az alábbiakban egy rövid elméleti bevezető után ábrázoljuk és tárgyaljuk többek között az oszcillátor kitérésének sebesség és időfüggését, az energiákat Euler-Cromer illetve Euler algoritmussal és elemezzük a futási idő függését a lépések számától.

2. Elméleti tárgyalás

2.1. A harmonikus oszcillátor

Írjuk fel a harmonikus oszcillátor Lagrange függvényét: (itt legyen x már rögtön az x-$x_{egyensulyi}$)

$$L = \frac{m}{2}\dot{x} - \frac{k}{2}(x)^2 \tag{1}$$

Az Euler-Lagrange egyenlet által megkapjuk a mozgásegyenletet.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} = \frac{\partial L}{\partial q} \tag{2}$$

$$m\ddot{x} = -kx \tag{3}$$

legyen $\omega^2 = \frac{k}{m}$, ekkor

$$\ddot{x} = -\omega^2 x \tag{4}$$

beszorozva a differenciálegyenletet $2\dot{x}$-el:

$$2\dot{x}\ddot{x} = 2\dot{x}(-\omega^2 x) \tag{5}$$

amit írhatunk ebben az alakban is:

$$(\dot{x}^2)^{\cdot} = -\omega^2(x^2)^{\cdot} \tag{6}$$

$$\dot{x}^2 = -\omega^2 x^2 + \hat{c}^2 \tag{7}$$

a fenti átalakítások után ezt az inhomogén diffegyenletet kapjuk.

$$\dot{x}^2 + \omega^2 x^2 = \hat{c}^2 \tag{8}$$

A homogén egyenlet megoldása:

$$\dot{x}^2 + \omega^2 x^2 = 0 \tag{9}$$

$$\dot{x}^2 = -\omega^2 x^2 \tag{10}$$

$$\dot{x} = \pm i\omega x \tag{11}$$

$$\frac{x}{dt} = \pm i\omega x \tag{12}$$

$$\frac{dx}{x} = \pm i\omega dt \tag{13}$$

$$ln(x) = \pm i\omega t + ln(c) \tag{14}$$

$$\mathbf{x_H = ce^{\pm i\omega t}} \qquad\qquad c \in \mathbb{R} \tag{15}$$

Az inhomogén egyenlet partikuláris megoldása:

$$u_P = c(t)e^{\pm i\omega t} \qquad\qquad \dot{x_P} = \dot{c}(t)e^{\pm i\omega t} + c(t)(\pm)i\omega e^{\pm i\omega t} \tag{16}$$

mivel konstans legyen c $\rightarrow \frac{c}{\sqrt{2}}$

$$(\dot{c}(t)e^{\pm i\omega t} \pm c(t)i\omega e^{i\omega t}\frac{1}{\sqrt{2}})(\dot{c}(t)e^{\pm i\omega t} \pm c(t)i\omega e^{\pm i\omega t}\frac{1}{\sqrt{2}}) + \omega^2 c^2(t)e^{\pm 2i\omega t} = \hat{c}^2 \tag{17}$$

(+ - és - +)-ra kiesnek a közös tagok és a másodikak négyzetei.
ha a hattványban ugyan ez "találkozik" akkor ugye e^+e^- szintén kiesnek
az első tag négyzete egyértelmű
a másodiknál marad az azonos előjelek szorzata, hogy ez kiessen majd a helyettesítésnél azért jött a $\sqrt{2}$ tag.Így marad a következő:

$$\dot{c}(t)e^{\pm 2i\omega t} = \hat{c}^2 \tag{18}$$

$$\dot{c}(t)e^{\pm i\omega t} = \pm\hat{c} \tag{19}$$

$$\frac{dc}{dt} = \pm\hat{c}e^{\mp i\omega t} \tag{20}$$

$$\mathbf{c(t) = \pm\hat{c}\frac{e^{\mp i\omega t}}{\mp i\omega} = \frac{\hat{c}}{\omega}\frac{e^{i\omega t} - e^{-i\omega t}}{i\omega}} \tag{21}$$

ahol $\hat{c} = \frac{v_0}{2}$

Így a diffegyenlet megoldása a homogén és a partikuláris megoldásából, a konstnsokat megfelelően választva:

$$\mathbf{x = x_0\frac{e^{i\omega t} + e^{-i\omega t}}{2} + \frac{v_0}{\omega}\frac{e^{i\omega t} - e^{-i\omega t}}{2i} = x_0\cos(\omega t) + \frac{v_0}{\omega}\sin(\omega t)} \tag{22}$$

2.2. Numerikus tárgyalás

2.2.1. Az Euler-Cromer formula

Felbontván a másodrendű differenciálegyenletet (amelyből fentebb kiindultunk) két elsőrenűre, alkalmazhatjuk az Euler-Cromer formulát:

$$\frac{dx}{dt} = v \qquad\qquad \ddot{x} = \frac{dv}{dt} = a = -\omega^2 x \tag{23}$$

$$v(t+dt) = v(t) + a(t)dt \tag{24}$$

$$x(t+dt) = x(t) + v(t+dt)dt \tag{25}$$

illetve az Euler algoritmussal szemben itt megmaradó energia:

$$E = \frac{1}{2}mv^2 + \frac{1}{2}m\omega^2 x^2 \tag{26}$$

2.2.2. A program bemenete

```
void getInput ( ) {
    cout << "Enter omega: ";
    cin >> omega;
    cout << "Enter x(0) and v(0): ";
    cin >> x >> v;
    cout << "Enter number of periods: ";
    cin >> periods;
    cout << "Enter steps per period: ";
    cin >> stepsPerPeriod;
    cout << "Enter output file name: ";
    cin >> fileName;
}
```

Látható, hogy a bemeneti paraméterek a fent megoldott diffegyenlet tagjai, illetve a periódusok száma és a felosztás.

2.2.3. A program kimenete

```
cout << "Period = " << p << "\tt = " << t
     << "\tx = " << x << "\tv = " << v
     << "\tenergy = " << energy() << endl;
```

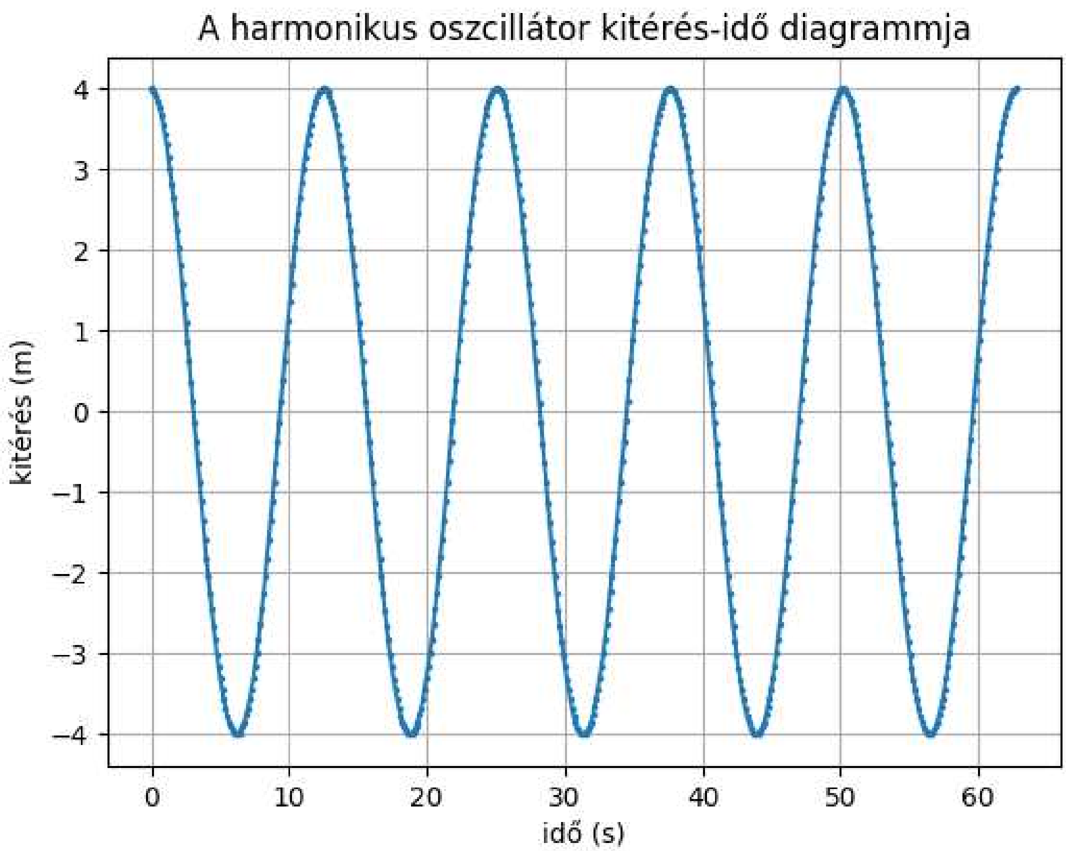
A data fájlba is szintén ezek íródnak a tárgy honlapján lévő programkód egy csekély módosítása után.

3. A szimuláció vizsgálata

Az alábbiak pythonban lettek plotolva.

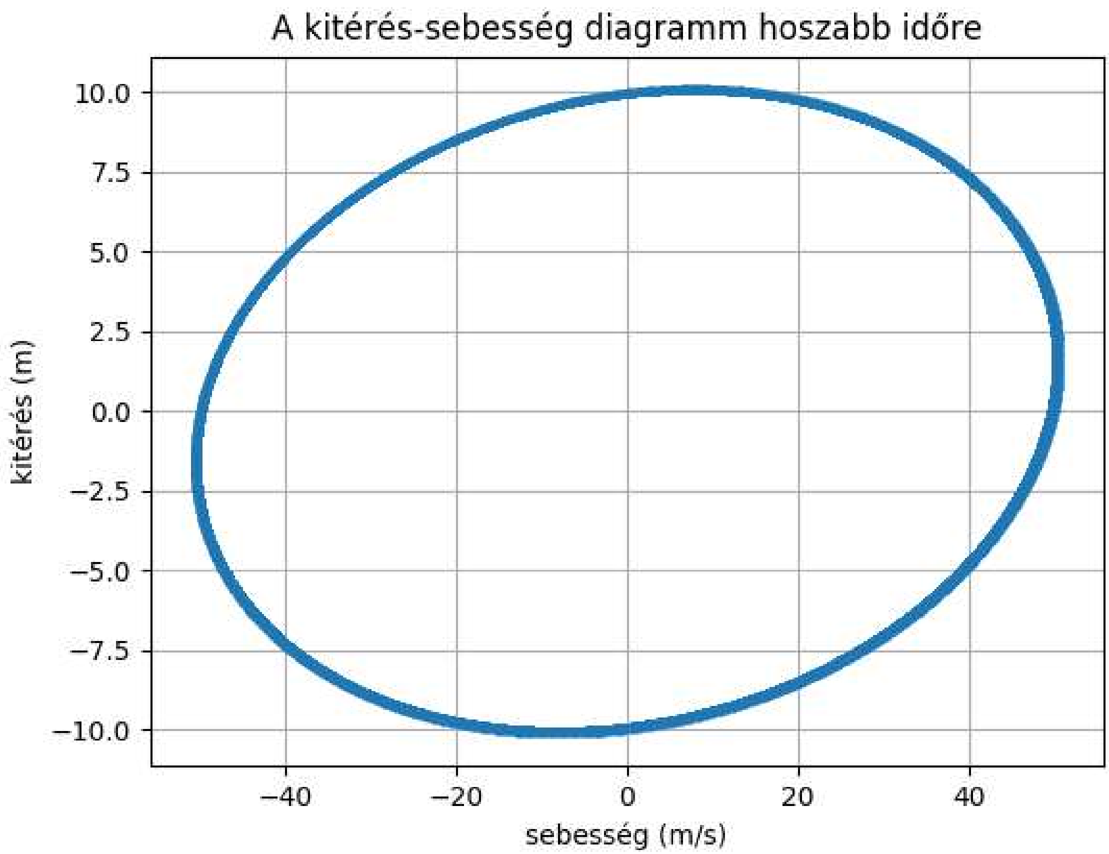
3.1. Kitérés-idő diagramm

A diagrammon látható, hogy a numerikusan visszakapott eredmény valóban harmonikus és periodikus:



3.2. Kitérés-sebesség diagramm

A fázitéren egy ellipszist várunk, mely főtengelyei párhuzamosak a koordinátarendszerünk tengelyeivel. Hoszabb időre az alábbi ábra ábrázolja a kimenetelt:



3.3. Energiamegmaradás

3.3.1. A tapasztalat

Az Euler-Cramer algoritmus esetén, oszcilláló energiaértékeket tapasztalhatunk, melyek a felosztás növelésével (minden más értéket azonosnak választva, a lenti ábrán 50,250,2500-as felosztásokkal) egy konstanshoz tartanak. Az Euler esetében, picit kellet csak változtani a kódon. Itt exponenciálisan elszáll az energia.

3.3.2. A kód

Euler-Cramer

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

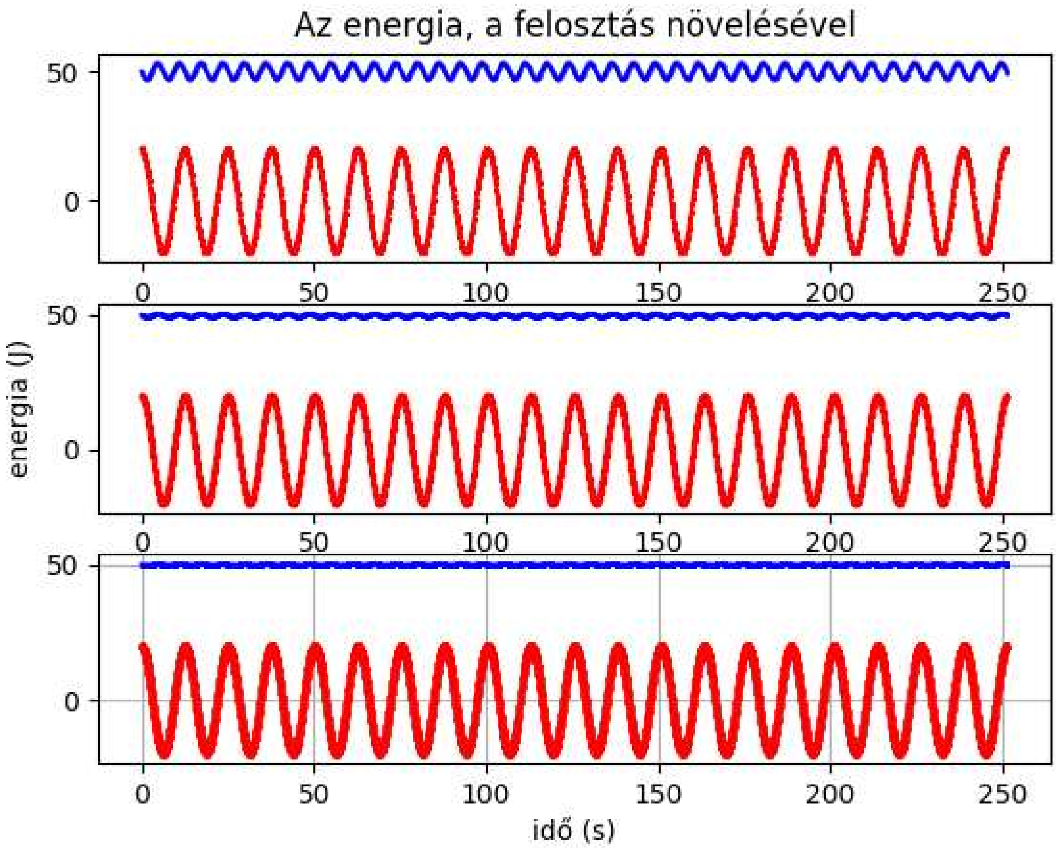
```
    x += v * dt;
}
}
```

Euler

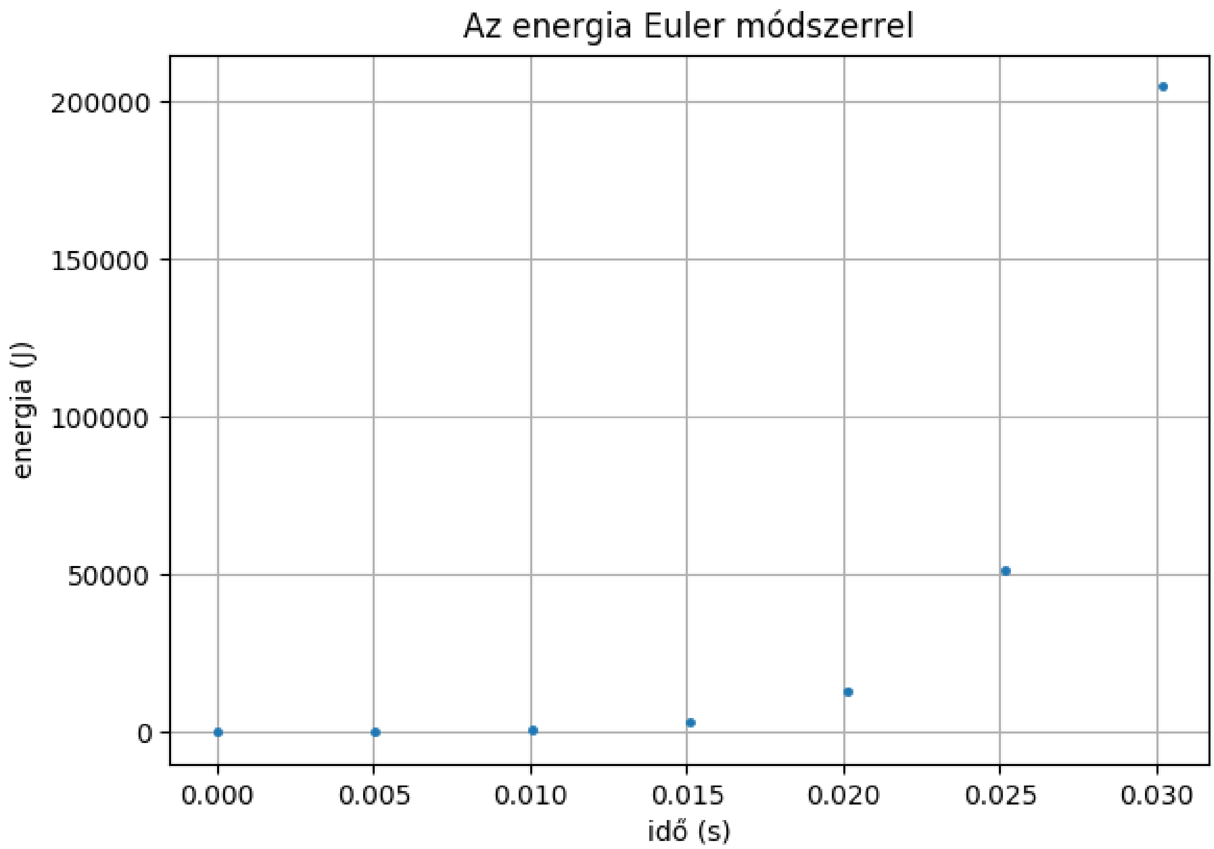
```
void EulerCromer (double dt) {
    double a = - omega * omega * x;
    v += a * dt;
    x += v * dt + x;
}
}
```

3.3.3. Az ábra

Euler-Cramer



Euler



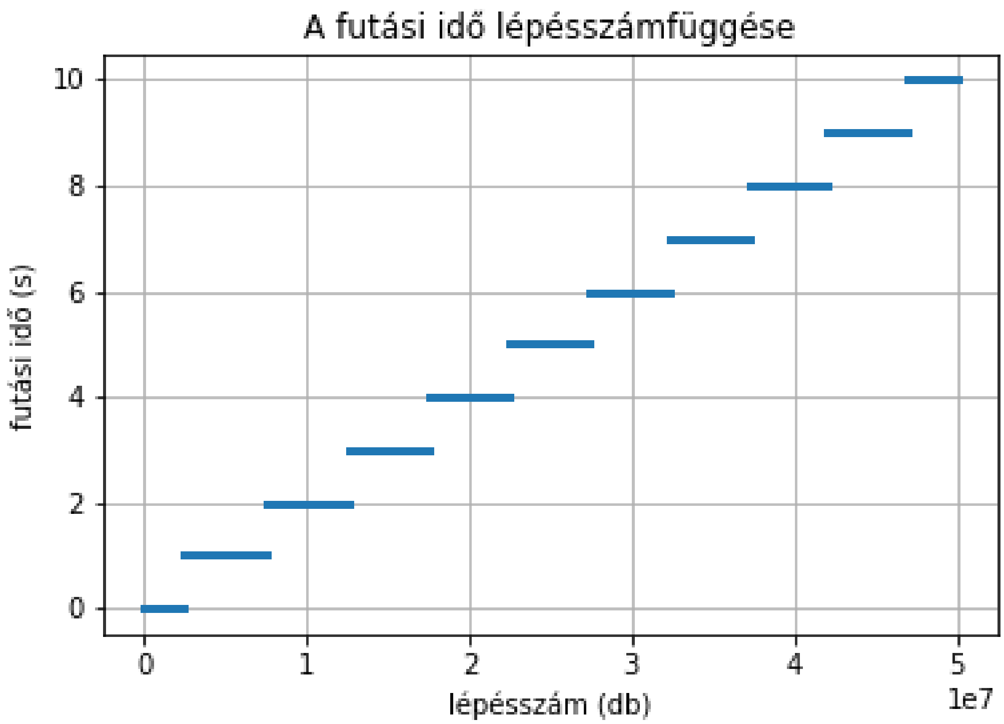
3.4. A futási idő

A futási időt az órán ajánlott time() parancs segítségével vizsgáltam. Itt fontosnak tartom megjegyezni, hogy a time parancs a valós idővel dolgozik, lényegileg különböző a clock()-tól, hiszen az egyik az aktuális időt nézi ami által ha a rendszer elfoglalt, vagy valami befolyásolja, teljesen más értékeket is adhat, mint a másik mely csak a programra szánt processzálási időt nézi. A kód maga így alakult:

```
time_t start = time(0);
for (int p = 1; p <= periods; p++) {
    for (int s = 0; s < stepsPerPeriod; s++) {
        EulerCromer(dt);
```

```
            t += dt;
            time_t currenttime = time(0);
            //cout << time(0) << '\n';
            file << s << '\t' << currenttime-start <<'\n';
        }
    }
```

Az ábrából láthatjuk, hogy a futási idő lineárisan függ a lépések számától:



4. Diszkusszió

Összességében elmondható hogy az Euler-Cromer algoritmus elég pontos és mindenképp ajánlottabb az alkalmazása a sima Euler helyett.

5. Hivatkozások

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