Harmonikus oszcillátor

Szőke Kálmán Benjamin SZKRADT.ELTE

2012. február 28.

1. Bevezetés

A jegyzőkönyv célja a harmónikus oszcillátor szimulációja volt. A program forráskódját a labor honlapjáról lehetett elérni, és ezt módosítottam a feladatokhoz. Több feladaton keresztűl a modell viselkedésének tanulmányozása volt a cél. Az kitérés-idő diagrammot, fázisteret, futási időt és a energiamegmaradást vizsgáltam.

2. Elméleti leírás

A szimulálást úgy tudjuk véghez vinni, ha a harmonikus oszcillátor modell mozgásegyenletét megadjuk, és ezt a differeneciál egyenletet megoldjuk numerikusan.

$$m\ddot{x} = -m\omega^2 x$$

Ennek a differenciálegyenletnek az analatikusan a megoldása a következő:

$$x_{(t)} = x_0 \cos \omega t + \frac{v_0}{\omega} \sin \omega t$$

Numerikusan kezelve a másodredű differenciálegyenletet felbontjuk két csatolt elsőrendű differenciálegyenletre, amiket az Euler-Cromer algoritmus segítségével oldunk meg, és ábrázoljuk ezeket az eredményeket.

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = \omega^2 x = a$$

$$v\left(t + dt\right) = v\left(t\right) + a\left(t\right) dt$$

$$x\left(t+dt\right)=x\left(t\right)+v\left(t\right)dt$$

Az energiát a következőképpen számoljuk az energiamegmaradás ellenőrzésénél.

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

2.1. A program bemeneti paraméterei

```
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;
```

2.2. A program kimeneti paraméterei fájlba

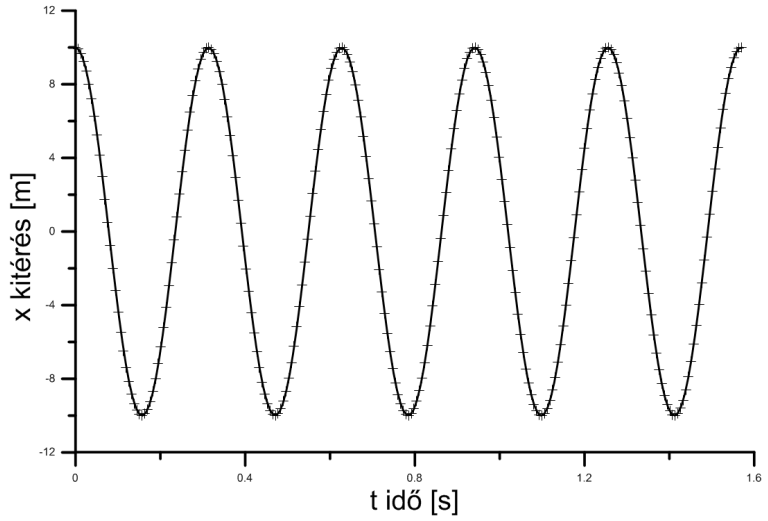
```
file<< t << '\t' << x << '\t' << v << '\t' << energy() << '\n';
```

3. Eredmények

3.1. Kitérés-idő diagramm

Ennél a feladatnál a kiszámolt x és t változókat ábrázoltuk. Láthatjuk hogy tetszőleges kezdőfeltétel mellett indítva mindig, periodikus, szinuszos görbét kapunk megoldásnak.

```
Enter omega: 20
Enter x(0) and v(0): 10 10
Enter number of periods: 5
Enter steps per period: 50
```



1. ábra. Kitérés-idő diagramm

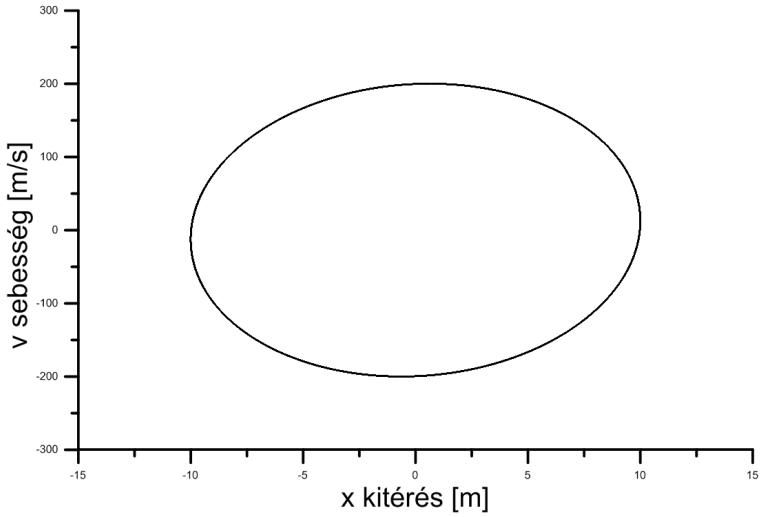
3.2. A kitérés-sebesség diagramm

A kitérés-sebesség diagrammhoz vagyis hogy megkapjuk a fázisteret a v és t változókat kellett ábrázolni, vagyis a sebesség időfüggését. Eredményeűl egy ellipszist kellett kapnunk, melynek főtengelyeia koordinátarendszer tengelyeivel párhuzamosak.

```
Enter omega: 20
Enter x(0) and v(0): 10 10
Enter number of periods: 150
Enter steps per period: 50
```

3.3. Energiamegmaradás

Ennél a feladatnál az elérhető forráskódon egy apró változtatást kellett megtenni, hogy a kimeneti fájlba is szerepeljenek a kiszámolt energiaértékek. Megfigyelhető az ábrán, hogy ha alacsonymintavételezéssel vizsgáljuk, akkor nagyon ingadozik, de azonban ha ezt növeljük az energia egy konstans függvényhez tart, és ez mutatka hogy teljesül az energiamegmaradás. A feladat része volt még, hogy vizsgáljuk meg az energia alakulását ha a sima Euler algoritmust használunk a differenciálegyenlet numerikus megoldásához. Az Euler-Cramernél oszcilláló jelleget vehettünk észre, míg a sima Eulernél exponenciálisan növekvőt.



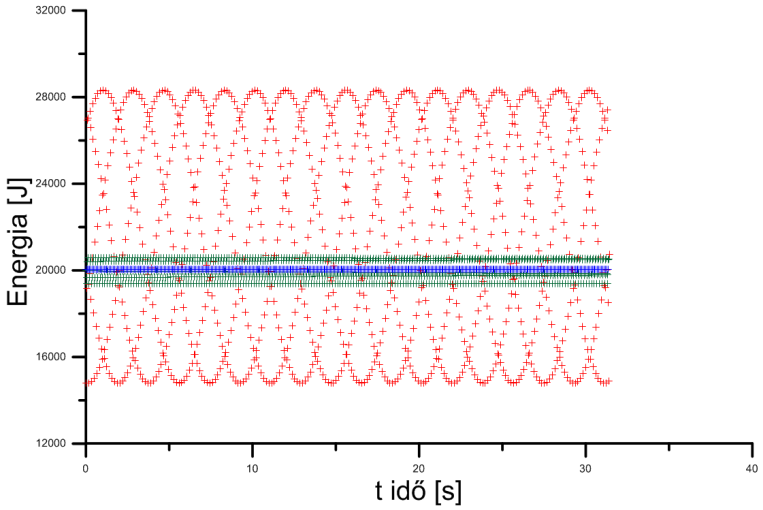
2. ábra. A kitérés-sebesség diagramm

3.3.1. Euler-Cramer algoritmus

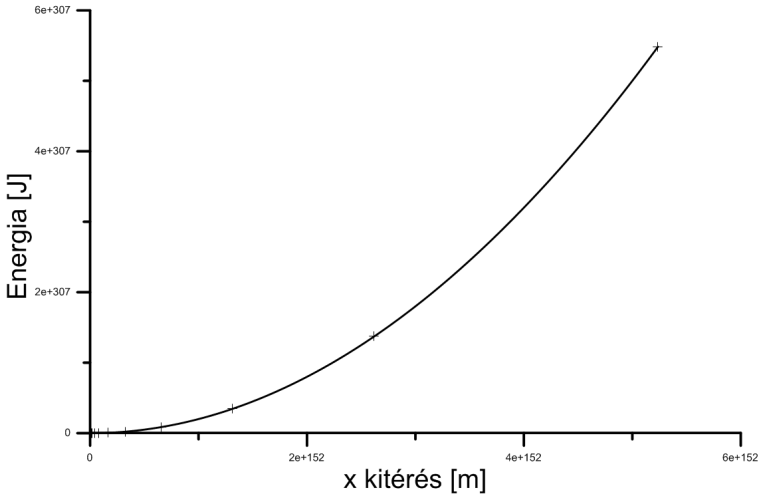
```
void EulerCromer (double dt)
{
double a = - omega * omega * x;
v += a * dt;
x += v * dt;
}
```

3.3.2. Euler algoritmus

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



3. ábra. Energiamegmaradás



4. ábra. Sima Euler algoritmus

3.4. Futási idő

A program futási idejének mérését a clock() függvény segítségével mértem. Az ábrán eredményűl látszik, hogy a felbontással lineárisan nőtt a számításhoz szükséges idő.

```
clock_t start = clock();        //idomérés start
for (int p = 1; p <= periods; p++) {
```

```
for (int s = 0; s < stepsPerPeriod; s++) {
EulerCromer(dt);
t += dt;
}
}
clock_t stop = clock();      //idomérés stop

double T = (double)(((stop-start)));
cout << stepsPerPeriod << '\t' << T << '\n';
file << stepsPerPeriod << '\t' << T << '\n';
```

5. ábra. Futási idő

Tartalomjegyzék

Hivatkozások

[1] Jegyzet
http://complex.elte.hu/∼csabai/szamszim/

[2] C++ forráskód
http://complex.elte.hu/∼csabai/szamszim/sho.cpp