SZÁMÍTÓGÉPES SZIMULÁCIÓK

Húr

jegyzőkönyv

Jegyzőkönyvet készítette:
Koroknai Botond (AT5M0G)

Jegyzőkönyv leadásának időpontja:
2024.04.20

1. A szimuláció megvalósítása

Első lépésben beálítottam a húr paramétereit, melyek a következők:

1. kód. A húr paraméterei

```
int LENGTH = 1.0;
double c = 1.0;
double dt = 0.01;
int num_steps = 10000;
int num_points = 100;
double dx = 0.01;
int cdash = 1.0;
```

A húr hosszát és sebességét az egyszerűség kedvéért egységnyinek vettem. A eddig elvégzettt szimulációkban tapasztaltak alapján a dt időlépés értékének egy kis számot választottam, hogy stabil szimulációt kapjunk. Hasonló céllal a num_steps paraméter értékét nagynak válaszottam meg, ezzel is a stabil szimuláció hosszú fennállását vizsgálva. A dx paraméter értékét a LENGTH / num_points, valamint cdash értékét a dx / dt hányadosok adták.

Ezt követte az időléptető leapfrog algoritmus megvalósítása:

2. kód. Leapfrog

```
void leapfrog(std::vector<std::vector<double>> &y, double c, double dt, int num_steps)
{

    int n = y.size();


    for (int step = 2; step < num_steps; ++step)
    {

        for (int i = 1; i < n - 1; ++i)
        {
             y_new = 2 * y[i][step - 1] - y[i][step - 2] + (c * c) / (cdash * cdash) *
             (y[i + 1][step - 1] + y[i - 1][step - 1] - 2 * y[i][step - 1]);
            y[i][step] = y_new;

        }

    }
}
```

Működése:

- A függvény első bemeneti paramétere egy 2D-s vektor lesz, ami a húr diszkretizált pontjainak kitérését tárolja az idő függvényében. A további paraméterket már korábban definiáltam.
- Első lépésben létrehoznuk egy n változót, melyben a 2D-s vektor első dimenziójának méretét, azaz a térbeli pontok számát fogjuk tárolni.
- Ezt követően két egymásba ágyazott for ciklussal végig lépked az idő pillanatokon, valamint a húr a diszkrét pontjain. Minden iteráció során kiszámítjuk a pontok kitéréseit, és ezeket az y változóba mentjük az aktuális időlépshez tartozó indexnek megfelelően.

Végül pedig a main függvény kerül meghívásra, ahol elvégezzük a végső simításokat:

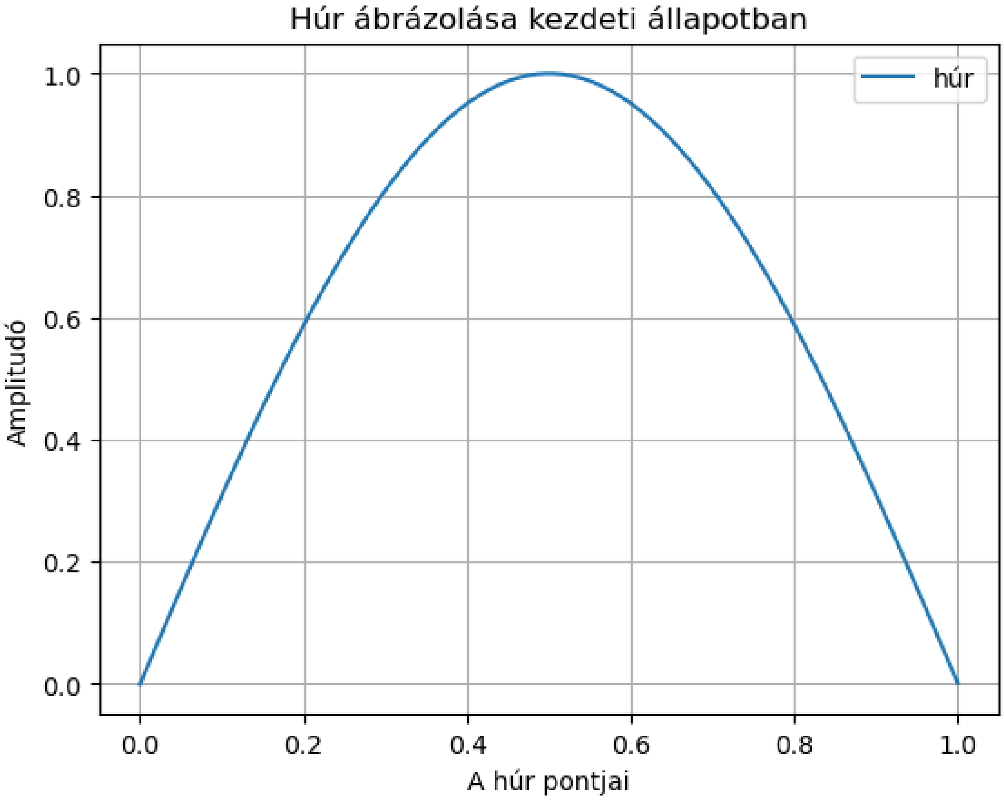
- Inicializálunk egy üres 2D-s vektort (minden értéke nulla), majd egy for ciklus segítségével az első két időponthoz tartozó adatokat feltöltjük egy szinusz hullám fél periódusának értékeivel, ezzel megadva a húr kezdeti állapotát.
- Utána lefuttatjuk magát a szimulációt.
- Végül az így kapott 2D-s y vektor értékeit kiíratjuk egy .data file-ba, lehetővé téve ezzel a későbbi pythonos elemzéseket.

3. kód. Main függvény

```
int main()
{
std::vector<std::vector<double>> y(num_points, std::vector<double>(num_steps, 0.0));

    for (int i = 0; i < num_points; ++i)
    {
        double x = double(i * LENGTH) / (num_points - 1);
        y[i][0] = sin(3.14 * x / LENGTH);
        y[i][1] = sin(3.14 * x / LENGTH);

    }

    leapfrog(y, c, dt, num_steps);


    std::ofstream outputFile(”output.data”);
    if (outputFile.is_open())
    {
        for (int i = 0; i < num_points; ++i)
        {
            for (int step = 0; step < num_steps; ++step)
            {
                outputFile << y[i][step] << ” ”;
            }
            outputFile << std::endl;
        }
        outputFile.close();
        std::cout << ”A kimeneti adatok sikeresen ki lettek
        irva a 'output.data' fajlba.” << std::endl;
    }
    else
    {
        std::cerr << ”Hiba: Nem sikerult megnyitni a kimeneti fajlt.” << std::endl;
        return 1;
    }

    return 0;
}
```

1.1. A húr ábrázolása

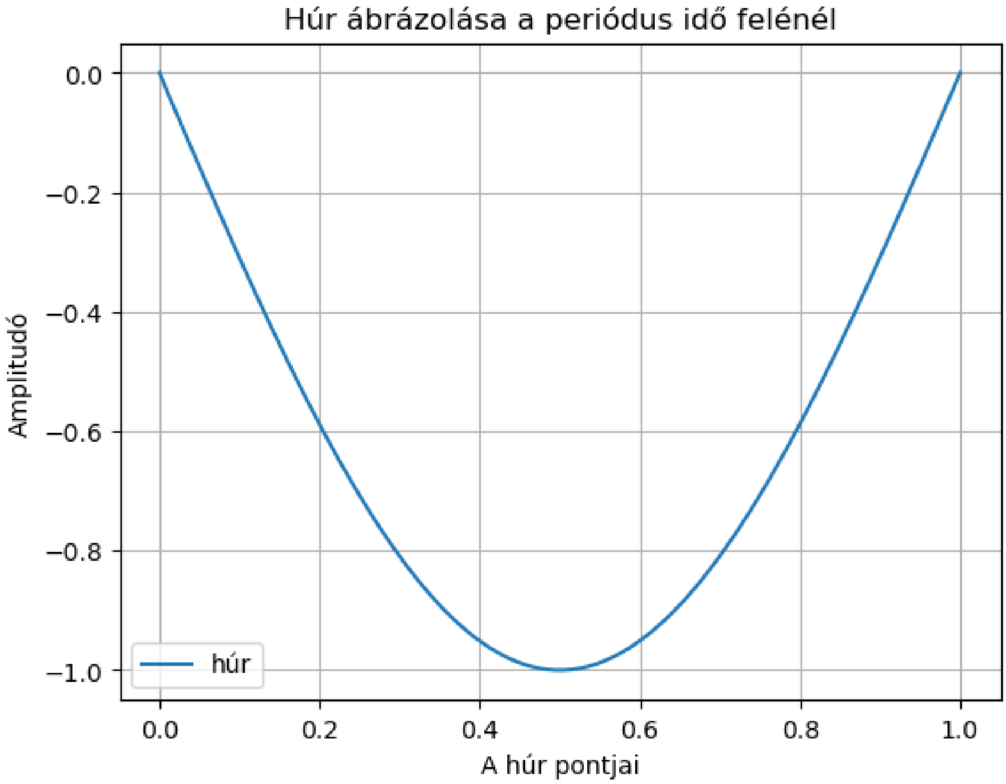
A húr mozgásáról 3 ábrát készítettem, amik egy fél periódus mozgását követik végig. Az első a kezdeti állapotot mutatja.



A második a negyed periódusban lévő állapotot ábrázolja.

Azt várnánk, hogy az inga minden pontja 0 értéket vesz fel. Ezeket a 8 tízezredes kis hibákat a diszkretizálásból adódó numerikus hibának könyvelem el.

A harmadik ábra pedig a fél periódusnál lévő állapotot demonstrálja.



A húr mozgásáról készült animáció a task1 mappában található, hur_animacio_task1.gif néven.

2. Analitkus megoldás:

A munkát ugyanazzal az n = 0 normál módusnak megfelelő kezdeti értékkel folytatom, így a hullámszám:

$$k_0 = \frac{\pi(0+1)}{L} = \frac{\pi}{L}$$

$$\omega_0 = c \cdot k_0 = c \cdot \frac{\pi}{L}$$

A kezdeti feltétel alapján

$$B_0 = 1$$

A húr mozgása így a következő képpen írható fel:

$$y(x,t) = B_0 \cdot sin\left(k_0 \cdot x\right) \cdot cos\left(\omega_0 \cdot t\right)$$

Behelyettesítve az adatokat:

$$y(x,t) = \sin\left(\frac{\pi x}{L}\right)\cos\left(\frac{\pi c t}{L}\right)$$

Ennek megfelelően pythonban definiáltam a következő függvényt:

```
def y_analytical(x, c, L,t):
    return np.sin(np.pi * x / L) * np.cos(np.pi * c * t / L)
```

4. kód. Analitikus megoldás

Kezdeti paramétereknek a következőket adtam meg az analitikus esetben:

- x : egy linspace 0-1 -ig 100 ponton mintavételezve
- c = 1, mint a diszkrét esetben.
- L = 1
- t = 1 s, mivel a diszkrét esetben pont 100 db kis 0.01 dt kis lépésre van szükség a fél periódus megtételéhez.

Az így kapott két húr szimulációt egy közös animáción ábrázoltam, ami megtalálható a task2 mappában compare_task2.gif néven. Azt láthatjuk, hogy a két megoldás nagyon szépen hasonlít egymásra.

2.1. A hullám sebességének becslése

Két végén rögzített húr esetén kialakuló állóhullámnak a hullámhossza felírható mint:

$$\lambda = \frac{2L}{n+1} = \frac{2L}{0+1} = 2L$$

A frekvencia így:

$$f = \frac{v}{\lambda} = \frac{v}{2L}$$

A szimuláció során látszik, hogy $T = 2$ azaz $f = \frac{1}{T} = 0.5$. Így

$$v = f \cdot \lambda = 0.5 \cdot 2 = 1$$

3. A rendszer stabilitásának vizsgálata

A stabilitást vizsgálatát két féle módon közelítettem meg, egyszer a c'-őt hagytam változatlanul 1-nek, és a c értékét változtattam: 0.5,0.25,0.1, valamint 0.01 -re, és az animációkat megnézve vizsgáltam a stabilitást. Az ilyesféle változtatások során minden esetben stabil maradt a szimulációm mint az látható a: c0_5..gif, c0_25..gif, c0_1..gif, és c0_01..gif fájlokban.

A másik vizsgálat során c értékét hagytam végig 1-en, és c' értékét változtattam, oly módon, hogy dx értékét növeltem: 0.05,0.5, és 5- re növeltem. Ebben az esetben is minden változat esetén stabil megoldásokat kaptam. Az eredmények a cdash_5.gif, cdash_50.gif, cdash_500.gif fájlokat lejátszva tekinthetőek meg.

4. Linearitás vizsgálata

Az első vizsgálat során két különböző kezdőeltételű húr rezgését vizsgáltam meg. A módosítást a kezdeti szinuszhullám amplitudóján végeztem a következő képpen:

5. kód. Kezdőállapotot inicializáló függvény

```
void initialize(std::vector<std::vector<double>> &y, double amplitude)
{
    for (int i = 0; i < num_points; ++i)
    {
        double x = double(i * LENGTH) / (num_points - 1);
        y[i][0] = amplitude * sin(3.14 * x / LENGTH);
        y[i][1] = amplitude * sin(3.14 * x / LENGTH);
    }
}
```

Az egyik húr amplitudóját 1 - nek választottam, míg a másikét 0.5 - nek. Mindkét szimulációrol animációt készítettem, és a vártnak megfelelően egy 1 illetve 0.5 amplitudójú rezgést kaptunk eredményül. A kombinált kezdőfeltételek eléréséhez:

6. kód. Kombinált kezdőfeltétel

```
std::vector<std::vector<double>> y_combined(num_points, std::vector<double>(num_steps, 0.0
    for (int i = 0; i < num_points; ++i)
    {
        for (int step = 0; step < num_steps; ++step)
        {
            double x = double(i * LENGTH) / (num_points - 1);
            y_combined[i][0] = 1 * sin(3.14 * x / LENGTH) + 0.5 * sin(3.14 * x / LENGTH);
            y_combined[i][1] = 1 * sin(3.14 * x / LENGTH) + 0.5 * sin(3.14 * x / LENGTH);
        }
    }
```

Ebben az esetben is a várt 1.5 amplitudós rezgést kaptam vissza eredményül.

A második vizsgált során két közeli normál módus kombinációjából előállított kezdeti feltételt kellett megvizsgálni. A normál módusokat a következő képpen határoztam meg, ismert a képlet a két végén rögzített állóhullám frekvenciájára:

$$f = \frac{nc}{2L}$$

Ennek megfelelően n = 1, és c = 1 ismeretében, valamint 2π-vel beszorozva, hogy körfrekvenciát kapjunk az első frekvencia értéke :

$$\omega_1 = \frac{1}{2L} \cdot 2\pi = \frac{\pi}{L}$$

A második frekvencia: n = 2

$$\omega_2 = \frac{2 \cdot 1}{2 \cdot L} \cdot 2\pi = \frac{2\pi}{L}$$

A 6.kód- ban ismertetett módon összekombináltam a kezdőfeltételeket. Itt a várt eredményt sikerül szimulálni, az első sajátfrekvencia hatására a húr középpontjának fel-le mozgását láthatjuk, miközben a második sajátfrekvencia hatására egy n = 2 paraméterű állóhullám karakterisztikus mozgását látjuk.

5. Források

- normal mode: https://en.wikipedia.org/wiki/Normal_mode
- standing wave: https://en.wikipedia.org/wiki/Standing_wave