

Kettősinga szimuláció felhasználói dokumentációja

Nógrádi Zsófia

May 26, 2019

Contents

1	A feladat	3
2	A fizikai leírása	3
3	Negyedrendű Runge-Kutta-módszer	4
4	Fejlesztői környezet	4
4.1	Összefoglaló	4
4.2	Microsoft Visual Studio Telepítése Windows rendszerre	4
4.3	Microsoft Visual Studio problémák	5
5	Kezdetek	5
6	A program használata	5
7	Akadályok	6

1 A feladat

A kettősinga kaotikus mozgásának modellezésére íródott a program. A kettős inga két inga egymáshoz csatolásával keletkező egyszerű rendszer, melynek mozgását a kezdeti feltételek erősen befolyásolják. Ez az objektum a legelterjedtebb demonstrációs eszköze a kaotikus mozgásnak.

2 A fizikai leírása

Az alap fizikai ismereteinket használjuk fel arra, hogy felírjuk a Newton-féle erőttörvényeket a rendszerre. A továbbiakban használjuk a következő jelöléseket:

x : azingatömegvízszinteshelyzete

y : azingatömegfüggőlegeshelyzete

θ : azingasöge

L : akötélhossza

F : akötélerő

m : azingatömege

g : nehézségigyorsulásértéke

Pusztán geometriai tudáunkat elővéve felírjuk az ingatömegek helyzetét:

$$x_1 = L_1 \sin \theta_1$$

$$y_1 = -L_1 \cos \theta_1$$

$$x_2 = x_1 + L_2 \sin \theta_2$$

$$y_2 = y_1 - L_2 \cos \theta_2$$

Ezeket az egyenleteket kétszer deriválva megkapjuk a gyorsulás értékét, majd azt felhasználva felírhatjuk a dinamika alaptörvényét, ami alapján $F = m \cdot a$. Az így kapott egyenleteket θ'' -re rendezve adódnak azon differenciálegyenletek, amiket a programunk Runge-Kutta-módszerrel oldja meg. A továbbiakban elsősorban jelölés szempontjából bevezetjük a szögsebességet ω ami éppen θ első deriváltja. Így a Runge-Kuttának már négy elsőrendű differenciálegyenletet tudunk átadni, amit az meg tud oldani. Ezek a következők:

$$\theta'_1 = \omega_1$$

$$\theta'_2 = \omega_2$$

$$\omega'_1 = \frac{-g(2m_1 + m_2) \sin \theta_1 - m_2 \cdot g \cdot \sin(\theta_1 - 2\theta_2) - 2 \sin(\theta_1 - \theta_2) m_2 (\omega_2^2 L_2 + \omega_1^2 L_1 \cos(\theta_1 - \theta_2))}{L_1(2m_1 + m_2 - m_2 \cdot \cos(2\theta_1 - 2\theta_2))}$$

$$\omega'_2 = \frac{2 \sin(\theta_1 - \theta_2) (\omega_1^2 \cdot L_1 (m_1 + m_2) + g(m_1 + m_2) \cos \theta_1 + \omega_2^2 \cdot L_2 \cdot m_2 \cos(\theta_1 - \theta_2))}{L_2(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2))}$$

3 Negyedrendű Runge-Kutta-módszer

A módszer differenciálegyenletrendszer megoldására alkalmas. Ahogy azt a neve is sugallja, ez egy negyedrendű közelítő módszer, ahol egy választott kicsi lépésközhez n -edik (ahol n tipikus nagy) lépésben oldja meg a kezdetiértékproblémát.

4 Fejlesztői környezet

4.1 Összefoglaló

A program C++ program nyelven íródott, ami jól fordul CodeBlocks-ban Cmake, vagy C/C++ fordítással, és VisualStudióban is.

4.2 Microsoft Visual Studio Telepítése Windows rendszerre

A Visual Studio pár éve ingyenesen letölthető például <https://code.visualstudio.com/> A telepítés egyszerűen vezetett telepítés, majd a környezet megnyitásakor

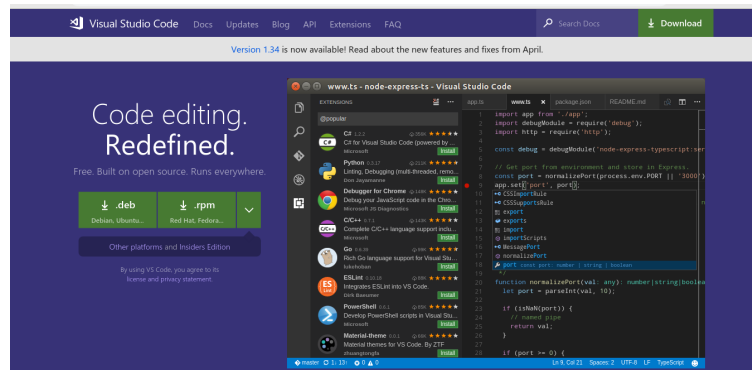


Figure 1: Telepítő felület

még telepítjük rá a CMake-t, a CMake Tools-t és a C/C++-t. Majd Scan for Kits-nél jó eséllyel megakadunk, vagy olyan elavult fordítót találunk, aminek a megléte teljesen felesleges. Szóval még egy fordítót sem árt telepítenünk, ami például lehet: Build Tools for Visual Studio 2019, ami elérhető a <https://visualstudio.microsoft.com/downloads/> címről, ha lejjebb görgetünk, akkor azonnal megtaláljuk a fentebb említett fordítót, amit szintén egyszerűen tudunk telepíteni. Ha ez megvan újra rámegyünk a Scan for Kitsre és kivlasszuk a telepített fordítót, ha sikerült azt a path-ban elhelyezni, hanem, akkor hibüzenetet kapunk, aminek megoldása az, hogy megkeressük a path mappát és áthelyezzük oda a fordítót.

4.3 Microsoft Visual Studio problémák

A telepítés nem volt éppen egyszerű, hiszen az egyik operációs rendszerem, a Linux családba tartozó Endless csodálatos biztonsági rendszere csak a saját Appstore-ból való telepítést engedélyezte. Ezzel még nem is volt feltétlen baj, a VS Code itt is elérhető volt, viszont a fordítója egy őskövületnél megállt, így alkalmatlanná vált arra, hogy használni tudjam. Tehát a későbbiekben arra kényszerültem, hogy a Windows 8-cas rendszerű, kevésbé alkalmas gépemre telepítsem a VS Code-ot és a magát a fordítót. Ez sem volt leányálom, ugyanis a fordítót csak nem akarta elérhetővé tenni a Path-nak, majd sok újratöltést követően sikerült a path-ban elhelyezni a fordítót. Ami van két hétig jól viselte ott magát, majd a gép egy frissítése során nem ismerte fel a környezetet, így kezdődött az egész előlről. Jelenleg működőképes a program.

5 Kezdetek

Először egy Git repozitóriumot kellett nyitni a Githubon. Majd a Visual Stu-

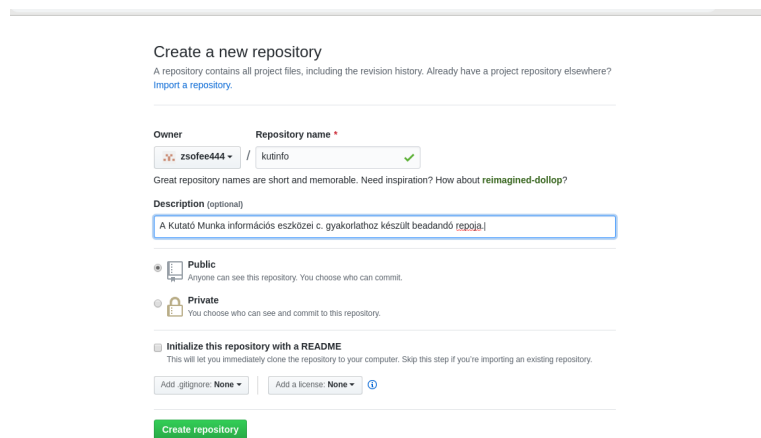


Figure 2: Git repo létrehozása

dióban klónozzuk a repo-t a Git-re, hogy a programírás minden módszere bekerüljön.

6 A program használata

Két fontos részből épül fel a projekt. Az egyik a "miniwindow.h" ez a tárgy oktatói által megírt ablakkezelő header, ami elérhető a nyilvános GitHub-ról <https://github.com/u235axe/miniwnd> címről. Ez a program teszi lehetővé a kettősinga kirajzolását és a mozgatását is. A másik rész az maga a "main.cpp", ahol igazából az érdemi része történik a programnak. Először egy 4 komponensű

pici vektorstruktúra megírása látszik, amiben csak azok a műveletek vannak definiálva, amire a differenciálegyenlet megoldása során szükségünk lesz, tehát a szorzár, összeadás illetve a skalárral való osztás.

Ezután következik a Runge-Kutta módszer definiálása, majd a rajzoltatás kezdeti fázisai az App struktúrában. Megadjuk a kezdeti értékeket, az ingatestek tömegét az ingák összát, a nehézségi gyorsulás értékét, a Runge-Kutta-módszer léptetésének hosszát, a kezdeti időt és a kezdeti értéket. A négyesvektor felépítése végig: $(\theta_1, \theta_2, \omega_1, \omega_2)$.

- `wnd.mouseHandler` az egérrel való műveletek elvégzését végzi. Erre most nincs szükségünk, így ezt nem kell babrálni.
- `wnd.resizeHandler` az ablak újraméretezésekor fellépő műveletek elvégzését végzi. Most ez is alapbeállításon marad.
- `wnd.idleHandler`-ben hívjuk meg a Runge-Kutta módszert, mostmár a négykomponensű vektorunkba beírva a differenciálegyenleteket, amiket az algoritmus megad.
- `wnd.renderHandler` akkor hívódik meg, amikor az ablak újrarajzolódik. Itt adjuk meg a pixelek alap színét, és a két ingatestből és két fonálból álló rendszer koordinátáit. És azok változását az időben.
- `wnd.open` pedig magát az ablakot formázza, megadja, hogy hol és mekkora méretben nyíljon meg, hogy mi legyen az ablak neve és hogy az ablak rendelkezzen a három alap funkcióval is.

Az így megírt App függvényt hívjuk be a main-be, hogy működjön a program.

7 Akadályok

A probléma az volt elsősorban, hogy a VS Code nem akarta az igazságot, így elkeseredésemben a CodeBlocksal és netes fordítókkal próbálkoztam. A netes fordítók azonban a "miniwindow.h" különböző alap package-ei hiányoztak, így azok hamar kiestek a próbálkozásból. Majd a sok próbálkozás után ugye a klónozást a VS Code-dal történt, ezért semmire nem jutottam a CodeBlockssal. De végül segítséggel a VS Code használhatóvá vált. A nehézséget főleg az ablakkezelő parancsok okozták, mivel ezek teljesen újak voltak, de sokat segített a tanárok által előre megírt Lotka-Volterra szimuláció, amit az egész alapjául vettem.