**Valovna enačba**

projekt pri predmetu Matematično modeliranje

**Manja Cafuta, Jan Hartman, Žan Jelen, Žiga Kokelj**

Mentor: Žiga Lesar

6. 6. 2016

1. **Naloga**

Naloga od nas zahteva, da v realnem času simuliramo valovanje 2D ploskve. Pri reševanju lahko uporabljamo različne metoda (Eulerjevo, Runge Kutta, Leapfrog, …). Rezultate različnih metod pa lahko kasneje primerjamo in ugotovimo katera daje najbolj realne simulacije valovanja.

Po implementaciji pa je treba nastaviti še primerne parametre(dušenje, hitrost valovanja, velikost mreže), da bo simulacija tekoča. Zanimivo je tudi obnašanje točk na robovih. V naravi sta najbolj očitna dva primera. Fiksni robovi simulirajo valovanje membrane pritrjene na kvadratni okvir. Prosti robovi pa simulirajo valovanje vode kvadratnem bazenu.

1. **Ideja**

Valovno enačbo lahko obravnavamo analitično ali numerično. Ideja je, da parcialno diferencialno enačbo prevedemo na sistem navadnih diferencialnih enačb. Zanima nas rešitev na nekem kvadratu, zato to območje diskretiziramo. Dobimo mrežo enakomerno razporejenih točk velikosti n\*n in na teh točkah iščemo približne rešitve. Ob tem je treba paziti, da lahko enačbe še vedno rešujemo dovolj hitro, da animacija poteka tekoče. Hitrost je odvisna predvsem od n-ja ter metode z katero rešujemo diferencialno enačbo.  Za bolj realen prikaz v enačbo vpeljemo še dušenje.

1. **Matematična podlaga**

Valovna enačba je homogena linearna parcialna diferencialna enačba 2. reda hiperboličnega tipa, ki v splošnem opisuje različna valovanja, kot so zvočno valovanje, svetlobno valovanje in vodno valovanje.

Splošna enačba se glasi:

V enačbi predstavlja funkcijo časa t in koordinat x in y, c predstavlja hitrost valovanja, ∆ pa Laplaceov operator. V dveh dimenzijah tako dobimo pri začetnih pogojih:

sledečo enačbo:   
 kjer računamo parcialna odvoda v obeh smereh.

Enačba v splošnem analitično ni rešljiva, zato smo diferencialno enačbo prevedli na sistem navadnih diferencialnih enačb in iskali približke na diskretizirani mreži točk.

Spremembo lege posamezne točke izračunamo z Laplaceovim operatorjem funkcije , pri čemer predstavlja matriko vrednosti ob času t. Laplacov operator se na diskretni mreži izraza kot:

Iz tega dobimo sistem enačb za pospešek posameznih točk:

kjer c predstavlja hitrost valovanja. Reševanja tega sistema se lotimo kot ponavadi pri diferencialnih enačbah 2. reda, tako da ga prevedemo na sistem 1. reda z 2x več enačbami, za novo spremenljivko pa vzamemo polje s hitrostmi:

Za reševanje sistema smo uporabili Eulerjevo metodo in metodo Runge Kutta 4. reda. Eulerjeva metoda je relativno preprosta, a tudi dokaj nestabilna sploh pri večjih korakih, medtem ko je RK4 računsko zahtevnejša a zato tudi bolj stabilna.

Splošna enačba Eulerjeve metode je:

kjer h predstavlja iteracijski korak.

Če pa uporabljamo Runge Kutta 4. reda pa prav tako aproksimiramo navadne diferencialne enačbe, le da je postopek računsko zahtevnejši, porabimo več prostora (shranjujemo več matrik), a so zato tudi rezultati bolj stabilni.

Splošne enačbe metode Runge-Kutta 4.:

Na koncu smo dodali se dušenje in sicer tako da smo na koncu odšteli z hitrost pomnoženo s koeficientom dušenja in dobili končno formulo:

1. **Potek reševanja**
   * 1. **Začetek v Javi**

Z reševanjem smo po predlogu mentorja začeli v Javi. Sestavili smo grafični uporabniški vmesnik in sami naredili tudi vizualizacijo: razred Image, ki smo ga barvali »na roke«. Kljub precejšnji količini vloženega dela je program imel težavo: rešitev je zelo hitro postala nestabilna. Tudi asistenta po dolgem pregledu kode nista našla težave, zato smo se odločili za migracijo v Octave. To je bila dobra odločitev tudi zaradi kvalitete prikaza: Octavova funkcija surf vizualizira dosti bolje (in je enostavna za uporabo) kot bi mi lahko sami v Javi brez uporabe zunanjih knjižnic

* + 1. **Rešitev v Octave**

Naša končna rešitev teče v Octavu. Sestavljena je iz ene datoteke (mogoče bi se dalo bolje strukturirati, vendar je problem v ogromnem številu parametrov, ki jih rabiš povsod). Glavna funkcija je valovna (zacetniPogoj, okolica), katere parametra pomenita začetne valove in statične dele. Vse ostale parametre se nastavlja znotraj funkcije, po vrsti so:

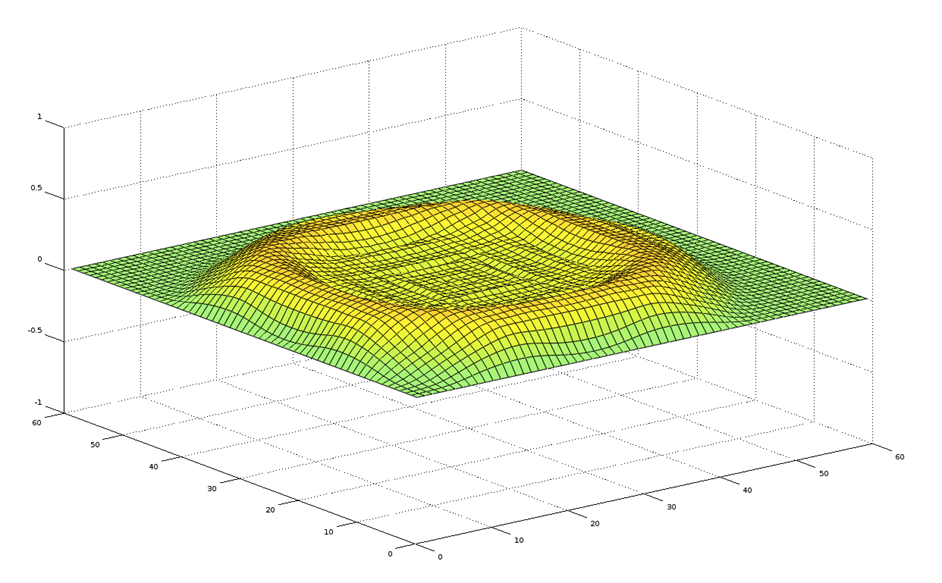
* n: velikost mreže (vpliva na hitrost procesiranja)
* v, u, w: matrike za hitrosti, pozicije in uteži delov mreže
* c: hitrost valovanja (vpliva na hitrost širjenja valov)
* h: razlike med točkami (diferenca)
* dt: časovni korak
* rob: pove, ali so robovi statični ali dinamični
* stIteracij: koliko korakov bomo izračunali
* metoda: izbira metode za reševanje diferencialnih enačb
* definicije začetnih pogojev in statičnih okolic

V glavni zanki gre postopek tako: po formuli izračunamo pospeške za točke in nato rešimo sistem diferencialnih enačb z izbrano metodo. Če so robovi fiksni, jih nastavimo na 0, upoštevamo pa tudi uteži točk (koliko se sploh premikajo, od 0 do 1). Nato nastavimo okolico za prikaz (ne računamo s tistimi vrednostmi, ker bi zmeraj dodajale energijo), izrišemo pa samo ob vsaki 15. iteraciji zanke (ker je izris časovno zahteven, za to uporabimo funkcijo surf). Spodaj sta še funkciji za izračun matrike pospeškov in energije (več o tem v naslednjih točkah).

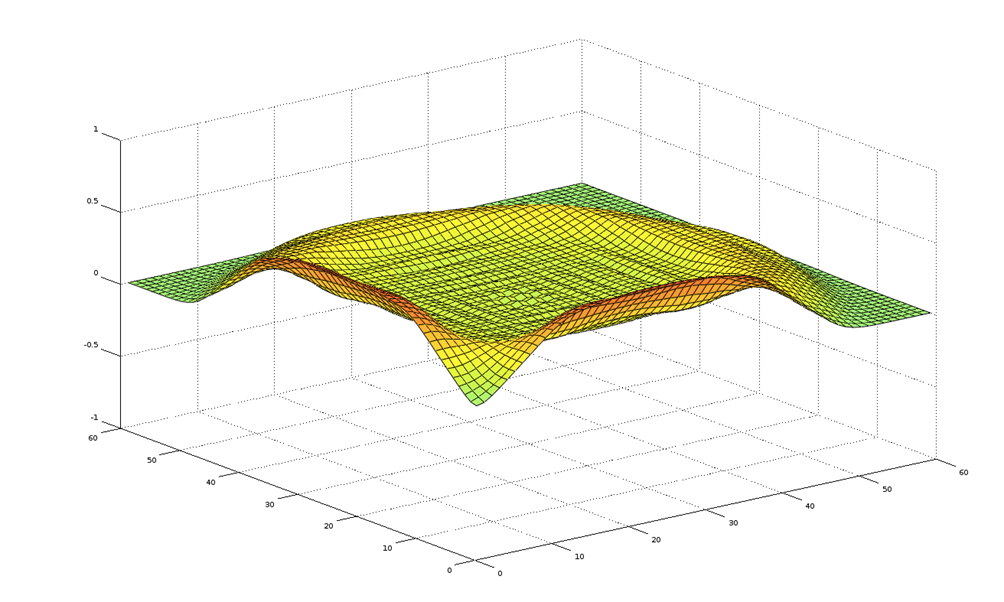
V končni fazi se je Octave izkazal za dosti bolj primernega za našo nalogo, saj podpira matrično računanje, ki zaradi optimizirane implementacije računanje pohitri. Vsa koda je tudi pokomentirana.

1. **Rezultati**

Program smo testirali na različnih nastavitvah. Možno je nastaviti dva različna načina obravnave robov, da so robovi fiksni ali pa dinamični in se spreminjajo po vzoru vode v bazenu. Možno je nastavljati tudi okolico, kjer lahko nastavimo različne konfiguracije prednastavljenih matrik uteži, ki kasneje delujejo kot statične točke v naši matriki. Nastaviti pa je možno tudi nekaj že vnaprej skonfiguriranih začetnih pogojev. Spodaj pa je predstavljenih nekaj zanimivejših rezultatov.

****

* + - 1. Osnovni val s fiksnimi robovi

****

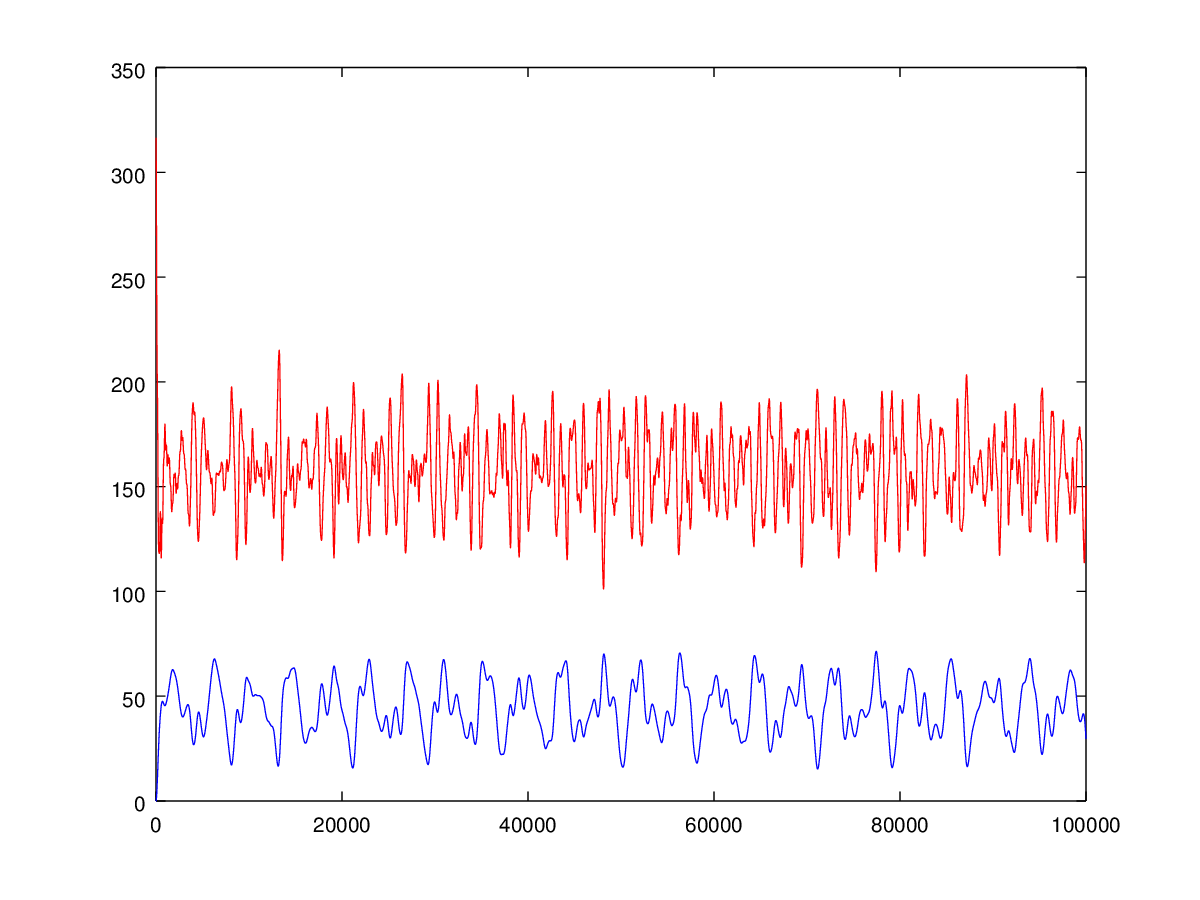
* + - 1. Osnovni val z dinamičnimi robovi



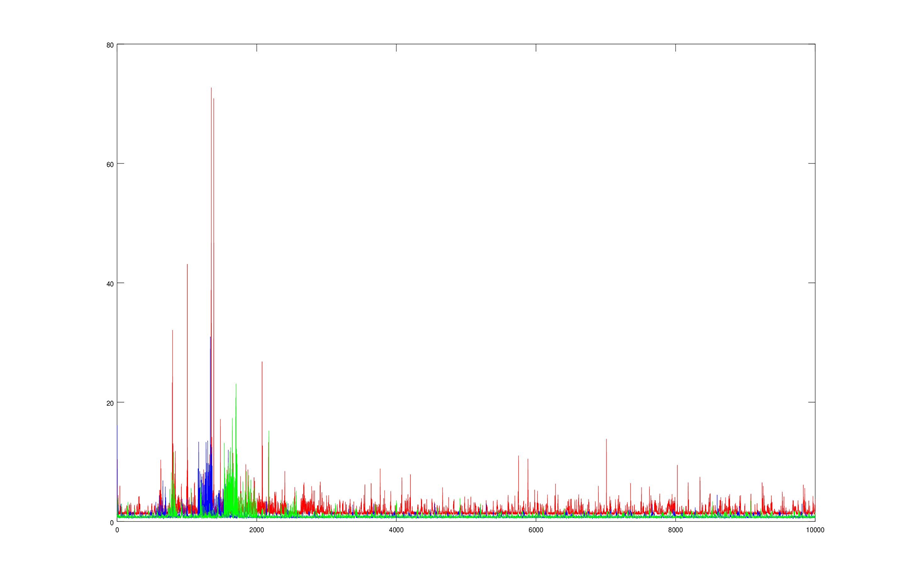
* + - 1. Zid z dvema luknjama, kjer prihaja do interference valov

1. **Grafi**

Na spodnjem grafu lahko opazimo spreminjanje kinetične energije (rdeči graf) in nihajne energije (modri graf) na primeru velikem 100000 iteracij.

****

Na spodnjem grafu je prikazana časovna zahtevnost iteracij na primeru velikem 10000 iteracij za vse tri uporabljene metode. Eulerjeva metoda (rdeča) je v začetnih iteracijah, kot tudi v nadalnjih, časovno najzahtevnejša metoda med tremi, medtem ko sta si RK4 in Leapfrog po zahtevnosti enakovredni, le da pri metodi Leapfrog kritične odseke algoritem najzahtevnejše iteracije poračuna nekoliko prej kot RK4.



1. **Zaključek**
2. **Viri**

<https://en.wikipedia.org/wiki/Euler_method>

<https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods>

<https://en.wikipedia.org/wiki/Wave_equation>