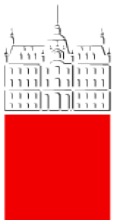
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Newtonov zakon

7. naloga pri Matematično-fizikalnem praktikumu

Avtor: Marko Urbanč (28191096)
Predavatelj: prof. dr. Borut Paul Kerševan

4.12.2021

Kazalo

1 Uvod

Simplektični integrator je v matematiki shema za numerično integracijo Hamiltonianskih sistemov. Pogosto se uporabljajo v nelinearni dinamiki, fiziki plazme, kvantni fiziki in pri reševanju problemov orbitalne dinamike. Uporabni so v vsakršnjem področju, kjer se energija sistema ohranja. Ustvarjene so za numerično reševanje Hamiltonovih enačb

$$\dot{p} = -\frac{\partial H}{\partial q} \,, \qquad \dot{q} = \frac{\partial H}{\partial p} \tag{1}$$

kjer so q prostorske koordinte, p koordinate gibalnih količin in H Hamiltonian sistema. Z pomočjo *simplektičnih metod* lahko, za enačbe, kjer je f le funkcija koordinat $f(x)$, rešimo enačbo (2) kar direktno.

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} = f(x) \tag{2}$$

Najbolj znana metoda je Verlet/Störmer/Encke metoda, ki je globalno natančna do drugega reda in ki točno ohranja tudi vrtilno količino sistema. Enako kot za npr. Eulerjevo metodo rešujemo za vsak diskretni korak velikosti h. Druge znane metode vključujejo še "leapfrog"metodo, kjer s pomočjo dodatnih vmesnih točk preskakujemo med lego in hitrostjo z zamikom $\frac{h}{2}$. Za metode višjih redov sta na voljo Forest-Ruth in Position Extended Forest-Ruth (PEFRL) metoda, ki sta obe globalno četrtega reda.

Gibanje masne točke v polju sil v eni dimenziji opišemo z Newtonovim zakonom, ki je diferencialna enačba drugega reda:

$$m \frac{\mathrm{d}^2 x}{\mathrm{d}t^2} = F \,.$$

Ta enačba je seveda enakovredna sistemu dveh enačb prvega reda, ki sta podobni kot (1):

$$m \frac{\mathrm{d}x}{\mathrm{d}t} = p \,, \qquad \frac{\mathrm{d}p}{\mathrm{d}t} = F$$

in tako jo lahko tudi rešujemo. Torej kot sistem dveh enačb prvega reda za katerega morajo biti na voljo tudi ustrezni začetni pogoji:

$$x(t=0) = x_0 \,, \qquad \frac{\mathrm{d}x}{\mathrm{d}t} = v(t=0) = v_0 \,.$$

V splošnem tu rešujemo sistem diferencialnih enačb drugega reda, ki ga lahko prevedemo na sistem enačb prvega reda z uvedbo novih spremenljivk v slogu gibalne količine pri Newtonovi enačbi. To pomeni, da metode iz prejšnje naloge (Euler, Midpoint, Runge-Kuta 4, ...) delujejo neposredno za reševanje takšnih sistemov enačb, kar naj bi v principu zadovoljiljo večino naših potreb. Problem, ki ga imajo te metode je, da ne nujno ohranjajo energijo, kar vodi do napačne rešitve predvsem daleč od izhodišča. Tu so zato uporabne prej omenjene simplektične metode.

2 Naloga

Naloga od nas zahteva, da rešimo problem matematičnega nihala pri začetnih vrednostih $x_0 = 1$ in $\dot{x}(0) = 0$. Zanima nas, kakšno velikost koraka h moramo izbrati, da dobimo natančnost na 3 decimalke. Primerjajmo tudi periodično stabilnost shem in opazujemo kako se amplitude nihajev sistematično kvarijo. Zraven narišemo tudi ustrezne fazne portrete.

Rešujemo torej enačbo:

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} = -\sin x \tag{3}$$

oziroma:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v \;, \qquad \frac{\mathrm{d}v}{\mathrm{d}t} = -\sin x \tag{4}$$

Za obravnavo si pomagamo z izračunom energije, za katero velja:

$$E \propto 1 - \cos x + \frac{\dot{x}^2}{2\omega_0^2} \;, \tag{5}$$

kjer je frekvenca $\omega_0^2 = \frac{g}{l}$ in je g gravitacijski pospešek in l dolžina nihala. Točna rešitev matematičnega nihala je:

$$x(t) = 2 \arcsin \left[\sin \frac{x_0}{2} \, \mathrm{sn} \left\{ \mathrm{K} \left(\sin^2 \frac{x_0}{2} \right) - \omega_0 t \;;\; \sin^2 \frac{x_0}{2} \right\} \right] . \tag{6}$$

3 Opis reševanja

Kot je že tradicija sem se naloge lotil v Pythonu, kjer sem uporabljal `matplotlib` za vso risanje in `NumPy` za vse numerične postopke. Primerjal sem tudi vgrajeno rutino `scipy.integrate.odeint()`. Kot pri prejšnji nalogi, je profesor priložile že nekaj spisanih rutin za numerično reševanje v datoteki `diffeq2.py`.

Prav tako kot zadnjič sem imel težave z uprabo profesorjevih priloženih metod, zato sem sam naredil implementacije raznih metod po najdeni psevdokodi in s pomočjo tega, kar je bilo napisano v diffeq2.py. Smiselno se mi je zdelo napisati postopke, ker so preprosti za implementacijo.

3.1 Verlet integrator

Recept za Verlet/Störmer/Encke metodo je preprost. Za vsak diskretni korak n velikosti h imam recept za korake:

$$\begin{aligned} x_{n+1} &= x_n + h \cdot v_n + \frac{h^2}{2} \cdot f(x_n) \\ v_{n+1} &= v_n + \frac{h}{2} \cdot [f(x_n) + f(x_{n+1})] \,. \end{aligned}$$

3.2 Leapfrog integrator

Alternativno lahko to shemo zapišemo s pomočjo dodatnih vmesnih točk in preskakujemo med lego in hitrostjo z zamikom $\frac{h}{2}$. Tako se recept glasi:

$$\begin{aligned} v_{n+\frac{1}{2}} &= v_n + \frac{h}{2} f(x_n) \\ x_{n+1} &= h v_{n+\frac{1}{2}} \\ v_{n+1} &= v_{n+\frac{1}{2}} + \frac{h}{2} f(x_n)\,. \end{aligned}$$

3.3 Yoshida integrator

Po odkritju Haruo Yoshida [1] se izkaže, da se lahko preprosto "leapfrogšhemo spremeni v metodo višjega reda tako, da se izračuna hitrosti in pozicije ob različnih časih. Če pravilno izberemo vmesne čase se napake pokrajšajo in dobimo preprosti recept za metodo četrtega reda. Recept se tako glasi:

$$\begin{aligned} x_i^1 &= x_i + c_1 v_i h \\ v_i^1 &= v_i + d_1 f(x_i^1) h \\ x_i^2 &= x_i^1 + c_2 v_i^1 h \\ v_i^2 &= v_i^1 + d_2 f(x_i^2) h \\ x_i^3 &= x_i^2 + c_3 v_i^2 h \\ v_i^3 &= v_i^2 + d_3 f(x_i^3) h \\ x_{i+1} &= x_i^3 + c_4 v_i^3 h \\ v_{i+1} &= v_i^3 \end{aligned}$$

kjer so x_i^k,v_i^k vmesne pozicije v vmesnem koraku k. Za koeficiente velja:

$$w_0 = -\frac{2^{\frac{1}{3}}}{2 - 2^{\frac{1}{3}}}$$

$$w_1 = \frac{1}{2 - 2^{\frac{1}{3}}}$$

$$c_1 = c_4 = \frac{w_1}{2} \qquad c_2 = c_3 = \frac{w_0 + w_1}{2}$$

$$d_1 = d_3 = w_1\,, \quad d_2 = w_0$$

3.4 Position extended Forest-Ruth Like algoritem

PEFRL je ena od najboljših simplektičnih metod na voljo. Globalno je četrtega reda in potrebuje štiri vrednotenja funkcije na korak. Algoritem je prav tako zelo preprost in simetričen:

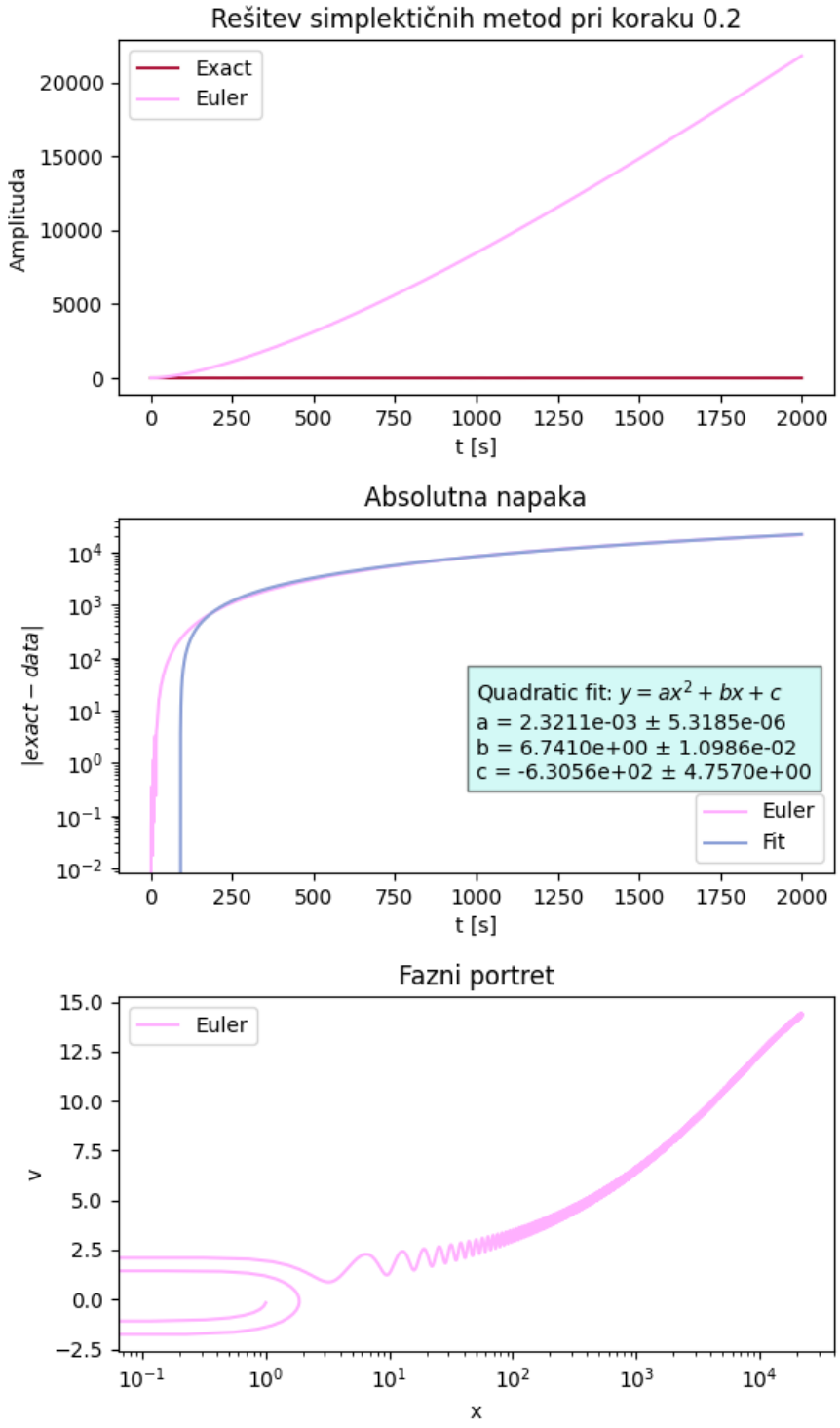
$$\begin{aligned} x &= x + \xi h v \\ v &= v + (1 - 2\lambda)\frac{h}{2} f(x) \\ x &= x + \chi h v \\ v &= v + \lambda h f(x) \\ x &= x + (1 - 2(\chi + \xi)) h v \\ v &= v + \lambda h f(x) \\ x &= x + \chi h v \\ v &= v + (1 - 2\lambda)\frac{h}{2} f(x) \\ x &= x + \xi h v \end{aligned}$$

kjer so koeficienti:

$$\begin{aligned} \xi &= 0.1786178958448091 \\ \lambda &= -0.2123418310626054 \\ \chi &= -0.6626458266981849 \cdot 10^{-1} \end{aligned}$$

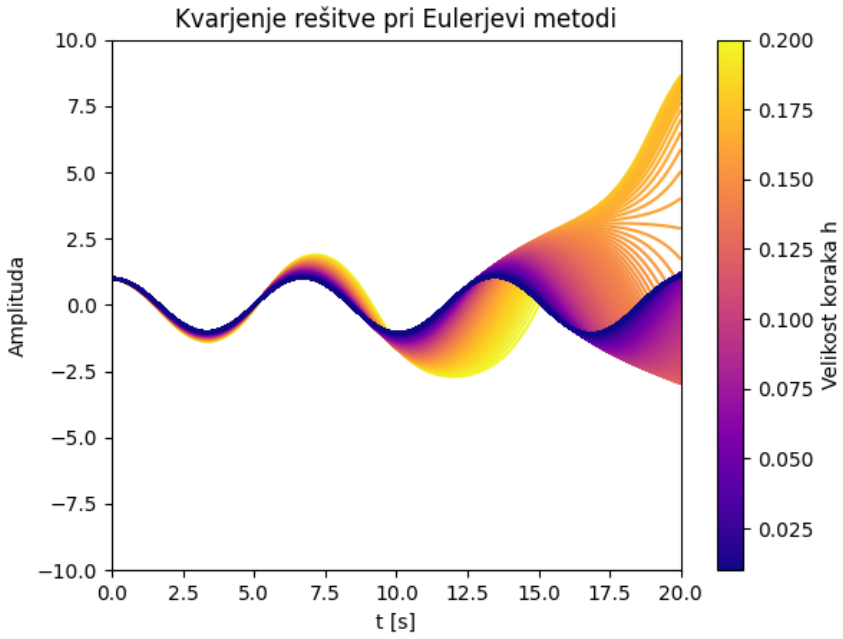
3.5 Sistematsko kvarjenje nihajev in velikosti koraka

Naloga od nas med drugim zahteva, da si pogledamo tudi kaj se dogaja po nekaj 10 nihajih. Tu se zelo jasno vidi slabost nesimplektičnih metod, ki že po parih nihajih precej zgrešijo točno rešitev. To je še bolj opazno pri manjših korakih seveda. Zgodi se lahko tudi to, da se napake ustrezno seštejejo in funkcija "pobegne". Primer tega sem si pogledal pri Eulerjevi metodi, ki je najbolj preprosta. Jasno se vidi, da absolutno odstopanje narašča z kvadratom časa, kar res pomeni, da zelo hitro zgubimo pravo rešitev. Tudi ob pogledu na fazni portret je to precej očitno.
Do podobnega kvarjenja lahko pride tudi pri simplektičnih metodah, ampak so te napake manj izrazite.



Slika 1: Pobegla rešitev pri Eulerjevi metodi

Opazujemo lahko tudi bližje, kaj se godi, pri spreminjanju velikosti koraka.



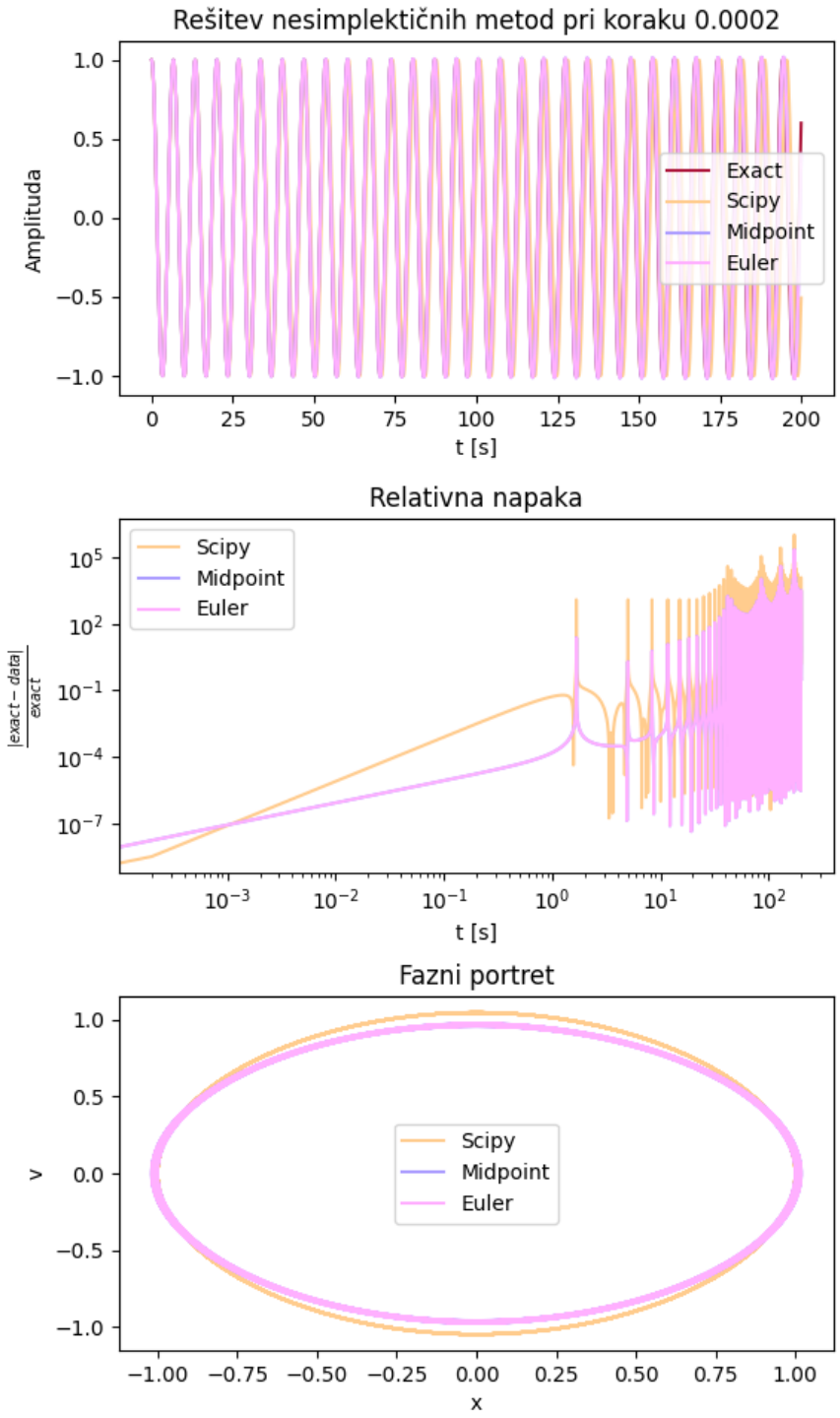
Slika 2: Kvarjenje rešitev pri Eulerjevi metodi

Kot rečeno je natančnost vseh teh metod zelo občutljiva na velikost koraka h. Naloga od nas zahteva, da poiščemo tisti h, ki nam bo zagotovoli natančnost 10^{-3}. Potrebna velikost koraka za željeno natančnost je zelo odvisna od metode, ki jo uporabljamo. Potrebni redi velikosti korakov za natančnost 10^{-3} pri $t = 0$ za različne metode so zbrani v tabeli.

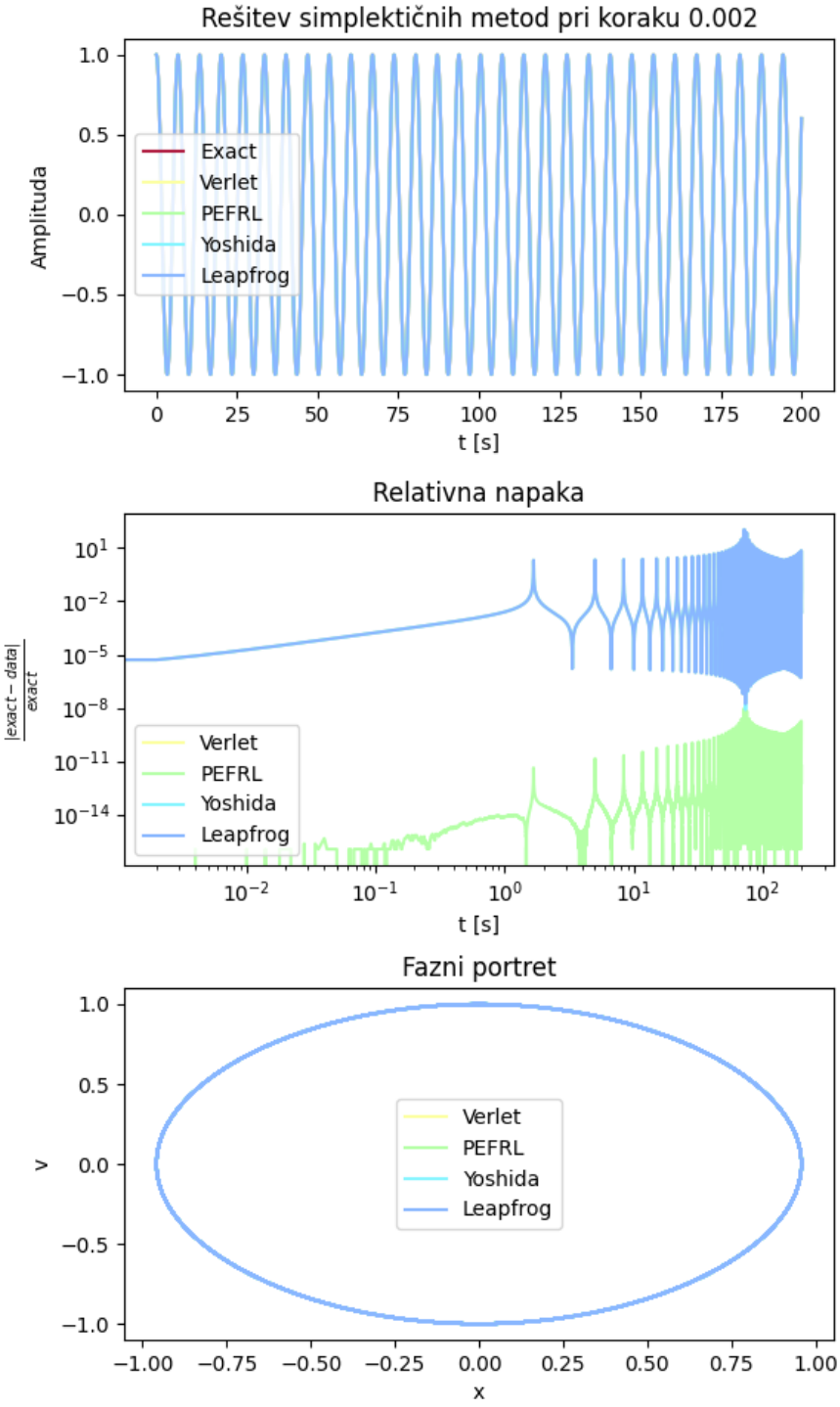
Metoda	Euler	Midpoint	Verlet	Leapfrog	Yoshida	PEFRL
Red potrebnega korak	10^{-4}	10^{-4}	10^{-3}	10^{-3}	10^{-3}	10^{-1}

4 Rezultati

Prvo sem si pogledal rešitve na zelo širokem intervalu. Vidi se, kako pri nesimplektičnih metodah "pušča" energija in so elipse v faznem portretu razmazane. To se zgodi tudi v neki meri pri simplektičnih metodah, ki pravzaprav zares le približno ohranjajo energijo. Popolnoma očitno je, da je za ta problem PEFRL algoritem najbolj primeren. Nudi res vrhunsko natančnost ob relativno majhni numerični zahtevnosti in zelo preprosti implementaciji.



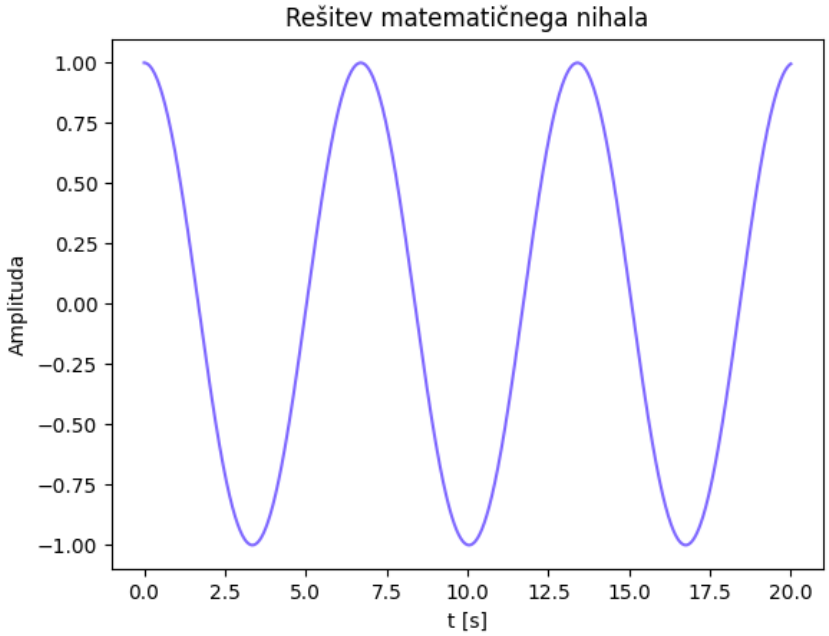
Slika 3: Rešitve nesimplektičnih metod



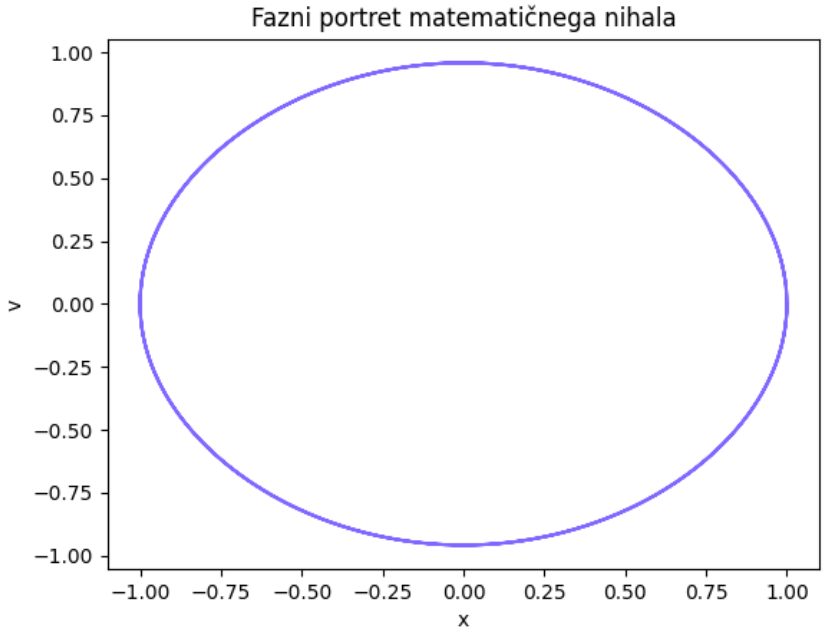
Slika 4: Rešitve simplektičnih metod

4.1 Končna rešitev

Zdelo se mi je smiselno predstaviti rešitev še v nekakšni končni obliki, kljub temu, da sem jo že prej prikazal. Tako sem naredil dva čisto osnovna grafa z PEFRL metodo, kjer se vidi obnašanje in res dobro definirana elipsa v faznem prostoru.



Slika 5: Rešitev matematičnega nihala z PEFRL



Slika 6: Fazni portret matematičnega nihala z PEFRL

5 Komentarji in izboljšave

Kar se tiče komentarjev glede rezultatov in obdelave se mi zdi, da sem vse sproti ustrezno pokomentiral.
Precej sem obupan in razočaran glede svojega izdelka. Zdi se mi, da je to najmanj "inspired" mafijski praktikum, ki sem ga oddal. Želel sem si, da bi lahko pogledal še kaj v zvezi z izrazom za energijo in da bi lahko dejansko napravil animacije teh rešitev. Imel sem tudi idejo rešiti kakšno bolj kompliciran nihalo, ampak seveda mi to ni uspelo. Manjka mu nek posben "spice", ki ga pa ni bilo od nikjer.. Tudi glede pozne oddaje sem žalosten. Upal sem, da mi bo uspelo oddati takoj med vikendom (že z zamudo), ampak sem imel nekoliko težave z zdravjem, ki so mi to preprečile. Dodatno se pa počutim potem še krivo, da se zgovarjam na te svoje težave kot nekakšen izgovor. Utrujen sem..

Slika 7: *Doge funny*

Literatura

[1] Haruo Yoshida. Construction of higher order symplectic integrators. *Physics Letters A*, 150(5-7):262–268, November 1990.