Navadne diferencialne enačbe: Začetni problem

Miha Čančula

7. marec 2012

1 Struktura programa

Orodje za reševanje poljubnega sistema navadnih diferencialnih enačb sem napisal kot programsko knjižnico v jeziku `C++`. Podobno strukturo, kot smo jo omenjali na predavanjih, že vsebuje knjižnica `GSL`, zato sem uporabil njene element. Za reševanje sistema enačb potrebujemo:

- Sistem enačb v obliki funkcije, ki iz trenutnega stanja izračuna odvod
- Metodo, ki izvede en korak (`gsl_odeiv2_evolve`)
- Algoritem, ki dinamično prilagaja velikost koraka (`gsl_odeiv2_control`)
- Uporabniški vmesnik, ki dejansko izračuna rešitev (`gsl_odeiv2_evolve_apply`)

Za izvajanje korakov sem uporabil vgrajeno funkcijo, ki uporablja metodo Runge-Kutta 4, sistem enačb in prilagajanje velikosti koraka pa sem implementiral sam.

Nadzorni mehanizem je deloval tako, da če je bila razlika med enim korakom dolžine h in dvema korakoma dolžine $h/2$ večja od ε, sem korak prepolovil in preizkus ponovil z manjšim korakom. Po drugi strani pa sem korak podvojil, če je bila razlika med dvema korakoma dolžine h in enim dolžine $2h$ manjša od ε. To razliko sem določil kot največje relativno odstopanje ene izmed štirih komponent rešitve.

Primerjal sem tako hitrost kot natančnost računanja za različne vrednosti ε.

2 Preverjanje

Za preverjanje delovanja programa sem izračunal gibanje planeta po tirnici okrog sonca. To je znan problem, za katerega vemo, da se energija in vrtilna količina planeta ohranjata, poleg tega pa poznamo celo analitično rešitev za poljubne začetne pogoje.

Poljubno stanje sistem sem zapisal s štirim brezdimenzijskimi spremenljivkami (x in y kot komponenti položaja in u in v kot komponenti hitrosti) in časom. Enačbo gibanja sem za te spremenljivke zapisal kot

$$\dot{x} = u \tag{1}$$
$$\dot{y} = v \tag{2}$$
$$\dot{u} = -x/(\sqrt{x^2+y^2})^3 \tag{3}$$
$$\dot{v} = -y/(\sqrt{x^2+y^2})^3 \tag{4}$$
$$\tag{5}$$

Energija in vrtilna količina planeta se v tem zapisu glasita

$$E = \frac{u^2 + v^2}{2} + \frac{1}{\sqrt{x^2 + y^2}} \tag{6}$$

$$\Gamma = xv - yu \tag{7}$$

Simulacijo sem vedno začel z začetnimi pogoji $x = 1$, $y = 0$, $u = 0$. Preostali začetni pogoj, ki pove hitrost planeta v smeri y, pa sem spreminjal, saj je od njega odvisna oblika orbite. Račun sem vsakih izvedel za interval $t \in [0, 200]$, kar ustreza približno 30 revolucijam planeta.

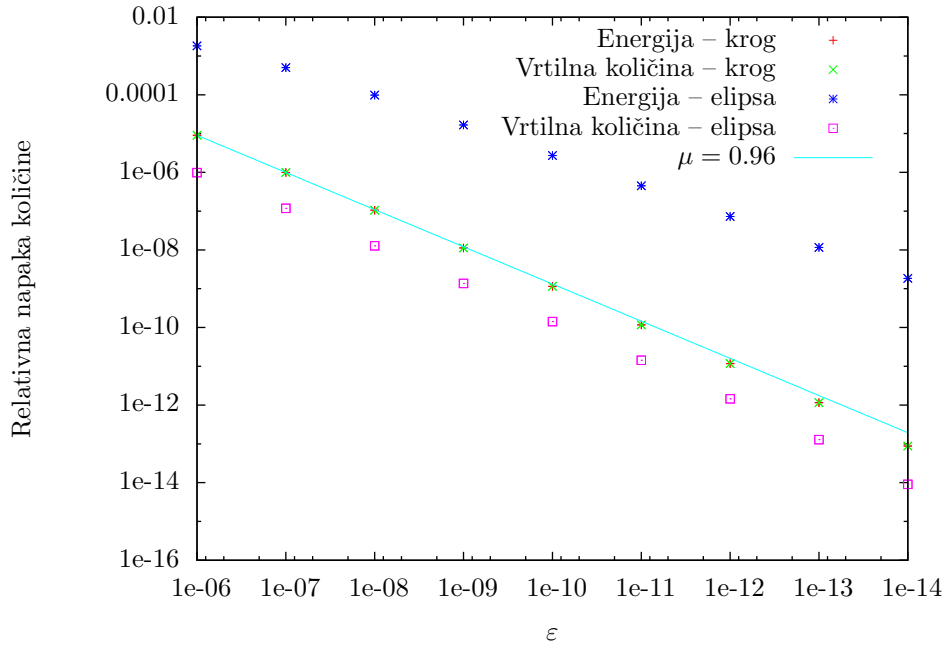
Rezultati kažejo, da je napaka metode σ potenčno odvisna od merila za natančnost koraka ε

$$\sigma = A \cdot \varepsilon^{\mu} \tag{8}$$

Konstanti A in μ sta seveda odvisni od tega, kako definiramo σ, torej napako katere količine merimo. Metodo sem preveril s štirimi kriteriji in za vsakega izračunal koeficient μ.

2.1 Energija in vrtilna količina

Najprej sem za vsako simulacijo izračunal največje odstopanje od začetnih vrednosti. Preizkus sem izvedel z različnimi začetnimi pogoji, ampak najbolj zanimiv je tisti, kjer planet prileti v neposredno bližino sonca, torej pri nizki začetni hitrosti. Rezultati prezkusa z $u(0) = 0,1$ so na sliki 1.

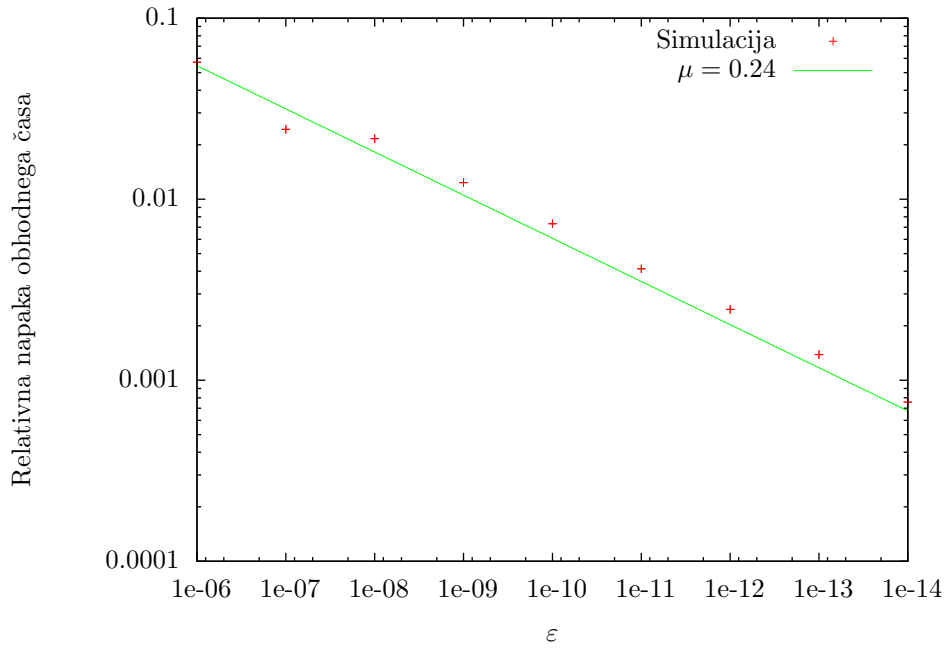


Slika 1: Odstopanja izračunanih količin, ki bi se morale ohranjati

Vidimo, da je relativna napaka obeh količin približno linearno odvisna od največje dovoljene napake spremenljivk ε, saj je μ blizu 1. Če torej zahtevamo manjšo napako posameznega koraka, bomo dosegli za enak faktor manjšo napako obeh konstant gibanja.

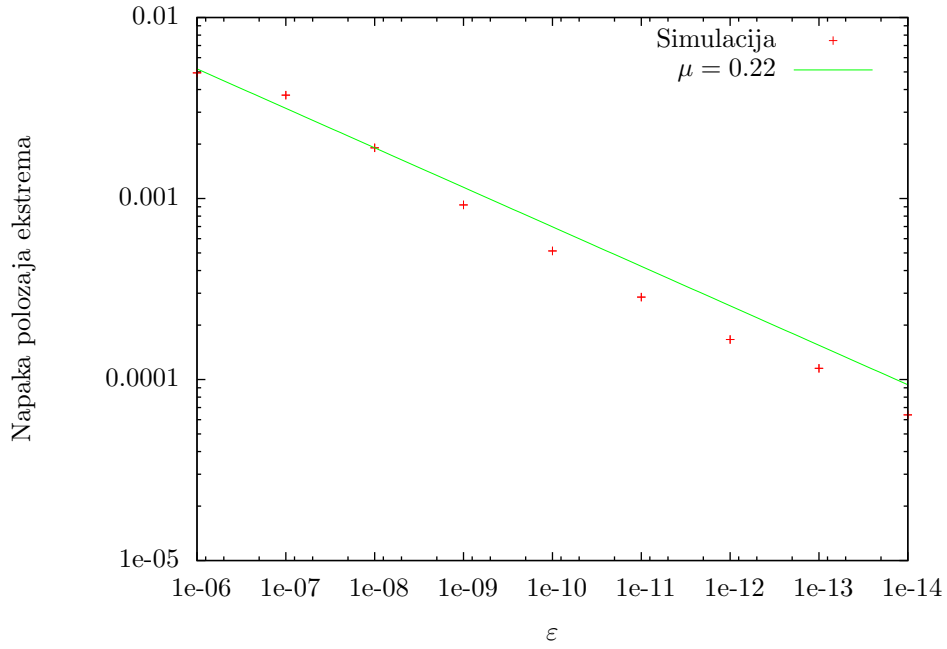
2.2 Obhodni čas in natančnost povratka

Pri krožnem tiru, ki ga dobimo pri pogoju $u(0) = 1$, lahko primerjamo dobljeni obhodni čas. Analitičen račun z izbranimi brezdimenzijskimi spremenljivkami napove obhodni čas 2π.



Slika 2: Odstopanje obhodnega časa od napovedi

Za tir v obliki male elipse pa lahko napovemo, da bodo vsi ekstremi oddaljenosti od sonca r na osi x. Napako povratka lahko vrednotimo kar kot največjo absolutno vrednost koordinate y

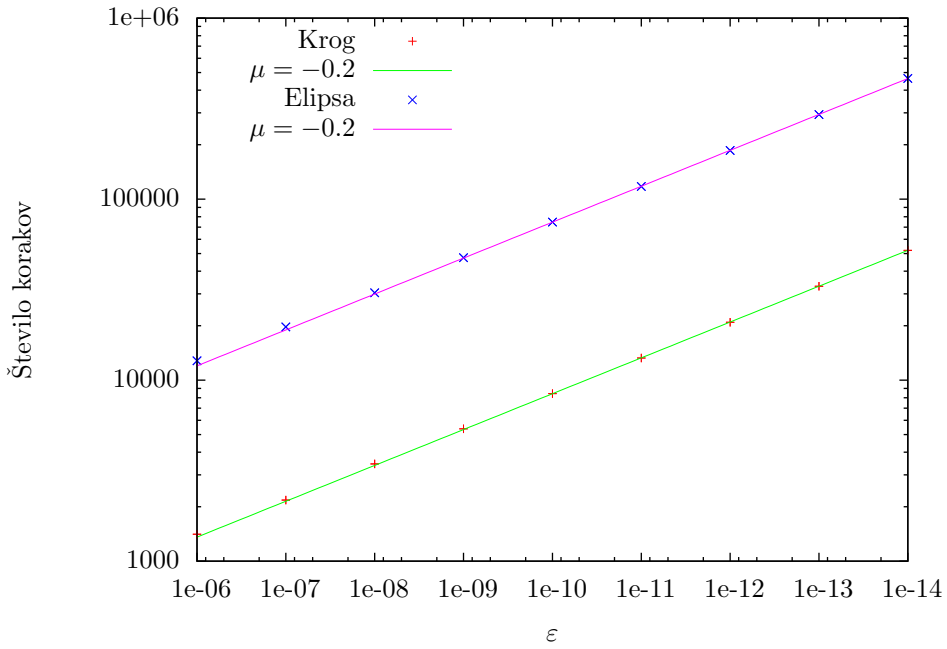


Slika 3: Odstopanje y koordinate ekstremov radija

V obeh primeri je μ občutno manjši od 1, nekje med četrtino in petino. Če želimo razpoloviti napako obhodnega časa ali položaja perihelija, moramo ε zmanjšati za faktor 20.

2.3 Hitrost

Odvisnost napak od količine ε nam samo po sebi ne pove dosti o uporabnosti algoritma, če ne poznamo tudi njegove časovne zahtevnosti. Neposredne odvisnost med časom računanja in ε ne poznamo, lahko pa preštejemo število korakov, potrebnih za doseg določene natančnosti. Če uporabimo prevzetek, da je čas računanja pogojen predvsem s številom evaluacij funkcije, ta pa je sorazmerna s številom korakov, je dobro merilo za hitrost algoritma kar število narejenih korakov.



Slika 4: Odvisnost števila korakov od zahtevane natančnosti

Po pričakovanju račun eliptične orbite pri isti natančnosti zahteva več korakov, saj zaradi bližini sonca postane sila na planet večja. Odvisnost števila korakov od ε pa je v obeh primerih enaka $\mu = 0,2$.

Pomemben podatek je, da število korakov narašča počasneje kot padajo vse merjene računske napake. Podvojitev števila korakov (in s tem časa računanja) nam torej prinese več kot dvakratno izboljšanje natančnosti vseh štirih merjenih količin.

3 Mimolet zvezde

Z istim algoritmom sem obravnaval tudi trk planetarnega sistema z zunanjo zvezdo. Vrtenje planeta sem usmeril tako, da se ujema s smerjo prihajajoče zvezde.

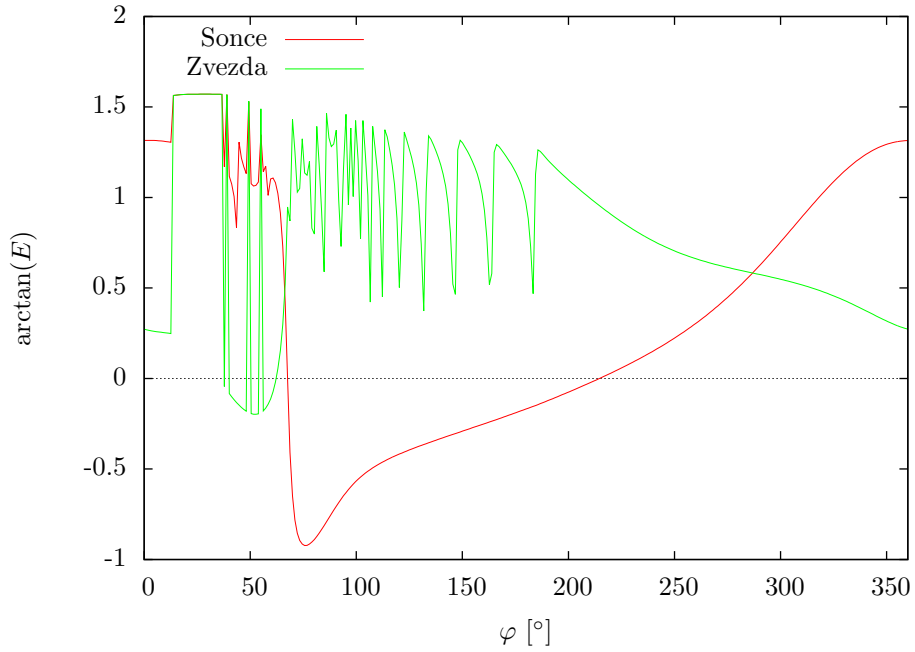
Srečanje sem simuliral dlje časa kot piše v navodili, saj sem na ta način lažje preveril, ali druga zvezda ujame planet ali planet odleti stran od obeh zvezd. Na začetku je bila zvezda od sonca oddaljena 20 astronomskih enot, na koncu pa 40. Relativno natančnost koraka ε sem postavil na 10^{-14}, tako da je simulacijo še vedna trajala manj kot minuto.

Končni rezultat takšnega srečanja je ena izmed naslednjih možnosti:

1. Planet se naprej kroži okrog sonca, le da je sedaj njegova orbita drugačna. To je najbolj pogost pojav.
2. Planet prileti tako blizu druge zvezde, da ga odnese v daljavo.
3. Druga zvezda ujame planet, tako da sedaj kroži okrog nje.

Drugi in tretji primer so zgodita, če je ob mimoletu druge zvezde planet na isti strani, tako da je sila med njima najmočnejša. V koordinatnem sistemu, kot smo ga vpeljali pri predavanjih, je to takrat, ko je kot φ blizu 0.

Končno stanje lahko obravnavamo tudi bolj kvantitativno. Za različne φ sem izračunal kakšno energijo ima planet na koncu glede na obe zvezdi. Če je kakšna izmed teh dveh energij negativna, je planet ujet v njeno gravitacijsko polje.



Slika 5: Skupna energija planeta po trku

V nekaterih primerih je energija zelo velika, dopuščati pa moramo tudi možnost, da je negativno, zato sem za prikaz narisal njen inverzni tangens. Vidimo, da planet ostane v orbiti svojega sonca, če je φ med 70 in 200 stopinjami, v ostalih primerih pa pobegne v vesolje. Možnost, da konča v orbiti druge zvezde je zelo majhna, to se zgodi le na ozkem intervalu, ko je φ okrog 50 stopinjami.

Predvsem v energiji glede na drugo zvezdo opazimo močna nihanja tudi pri majhni spremembi kota, kar lahko pripišemo numeričnim napakam algoritma. Pri majhnih kotih planet prileti zelo blizu zvezde, kar je najverjetneje razlog za zelo visoke izhodne energije v tistem delu. Vseeno pa se dobro vidi interval, kjer se planet ujame v zvezdino gravitacijsko polje.

Za različne kote φ sem naredil animacijo, ki prikazuje gibanje vseh treh teles. Poročilu prilagam po eno animacijo vsakega izmed treh primerov. Tri datoteke (`odleti`, `ujame` in `nic`) pripadajo

po vrsti kotom 40, 51 in 115°. Ker sem želel prikazati čim daljše obdobje, razmerje med višino in širino slike ni pravo, zato orbite izgledajo pokvarjene. Pri predstavi pomaga podatek, da se planet na začetku giblje po krožnici.