Simulacija gibanja zvrka pomoću programskog jezika Python

Ivanjek, Jasnik

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:284478

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Jasnik Ivanjek

SIMULACIJA GIBANJA ZVRKA POMOĆU PROGRAMSKOG JEZIKA PYTHON

Diplomski rad

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA I INFORMATIKA, SMJER: NASTAVNIČKI

Jasnik Ivanjek

Diplomski rad

Simulacija gibanja zvrka pomoću programskog jezika Python

Voditelj diplomskog rada: prof. dr. sc. Tamara Nikšić

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2019.

Sažetak

U ovom radu razmatramo ulogu računalnih simulacija kao nadopunu ili alternativu izvođenju pokusa u nastavi fizike. Postupak izrade računalne simulacije ilustriran je na primjeru zvrka. Jednadžbe gibanja zvrka postavljene su u sustavu zvrka (Eulerove jednadžbe) što vodi na sustav običnih vezanih diferencijalnih jednadžbi koje najčešće možemo riješiti samo numerički. Pritom smo se, radi jednostavnosti, ograničili na problem zvrka koji rotira oko fiksirane točke, odnosno eliminirali smo mogućnost translacija. Za izradu simulacije koristili smo programski jezik Python 3 i njegove biblioteke numpy i scipy za numeričko rješavanje diferencijalnih jednadžbi te paket VPython za vizualizaciju gibanja zvrka. Izrađene simulacije obuhvaćaju slobodni simetrični i asimetrični zvrk, te simetrični, asimetrični i Kovaleskayin zvrk u polju sile teže. Na primjerima pokazujemo kako oblik zvrka, vanjska sila i početni uvjeti utječu na gibanje zvrka. Ove i slične simulacije mogle bi pomoći u nastavi fizike kada nemamo uvjete za izvođenje pokusa ili izvedeni pokus želimo nadopuniti simulacijom kako bismo učenicima dodatno približili teorijsku pozadinu određenog fizikalnog problema.

Ključne riječi: simulacija, zvrk, Python, Eulerove jednadžbe, kruto tijelo

Spinning top simulations using Python

Abstract

In this thesis, we consider the role of computer simulations as a supplement or an alternative to performing experiments in teaching physics. The computer simulation process is illustrated by the example of the spinning top. The spinning top equations of motion are derived in the moving system attached to the top (Euler's equations) leading to a system of coupled ordinary differential equations that one usually solves numerically. For simplicity, we restrict the problem to rotation around the fixed point thus eliminating the translational degrees of freedom. In order to solve the Euler's equations, we used the Python 3 programming language together with numpy and scipy packages. Spinning top visualization is performed with VPython. Simulations include free symmetric and asymmetric top, as well as the heavy symmetric, asymmetric and Kovaleskaya top. We illustrate the influence of the shape of the spinning top, the external force and the initial conditions on the motion of the top. These and similar simulations could be useful in teaching physics when we cannot perform the experiment or when the teacher wants to use the simulation as the supplement to the experiment in order to further explain the theoretical background of a particular physical problem.

Keywords: simulation, spinning top, Python, Euler's equations, rigid body

Sadržaj

1. Uvod ... 1

1.1. Simulacije ... 1

1.2. Zvrk ... 2

1.3. Povijest zvrka ... 2

2. Opis gibanja ... 3

2.1. Eulerove jednadžbe ... 3

2.2. Slobodni simetrični zvrk ... 4

2.3. Slobodni asimetrični zvrk ... 5

2.4. Rekonstrukcija gibanja zvrka iz Eulerovi jednadžbi ... 7

2.5. Gibanje zvrka u polju sile teže ... 8

3. Simulacija slobodnog simetričnog zvrka ... 9

3.1. Korišteni programski paketi ... 9

3.2. Ulaz ... 9

3.3. Izlaz ... 10

3.4. Tijek programa ... 11

3.5. Provjera simulacija ... 12

4. Simulacija slobodnog asimetričnog zvrka ... 13

4.1. Ulaz ... 13

4.2. Izlaz ... 13

4.3. Tijek programa ... 15

4.4. Provjera simulacije ... 16

5. Simulacija zvrka u polju sile teže ... 17

5.1. Ulaz ... 17

5.2. Izlaz ... 17

5.3. Tijek programa ... 20

5.4. Opis gibanja simetričnog zvrka u polju sile teže – precesija i nutacija ... 21

5.5. Primjer kaotičnog sustava – asimetrični zvrk u polju sile teže ... 25

6. Zaključak ... 27

Dodaci ... 29

A. Usporedba rješenja za slobodni simetrični zvrk ... 29

B. Usporedba rješenja za slobodni asimetrični zvrk ... 31

C. Simulacija simetričnog slobodnog zvrka ... 33

D. Simulacija asimetričnog slobodnog zvrka ... 36

E. Simulacija simetričnog zvrka u gravitacijskom polju 39
F. Simulacija asimetričnog zvrka u gravitacijskom polju 42
G. Simulacija Kovalevskaya zvrka 45
Literatura 49

1. Uvod

1.1. Simulacije

Izvođenje demonstracijskih pokusa u učionici ili samostalni učenički pokusi zasigurno predstavljaju najbolji način učenja fizike, ali i ostalih prirodoslovnih predmeta. Međutim, pokusi mogu biti teško izvodljivi s dostupnom tehnologijom, preskupi ili vremenski prezahtjevni. Iako nikada ne mogu u potpunosti nadomjestiti pokus, pažljivo oblikovane računalne simulacije mogu biti prihvatljiva alternativa. Pomoću računala možemo programirati zakone fizike i stvoriti razne objekte, te vizualizirati odvijanje samog pokusa. Neke simulacije su jednostavne i mogu se napraviti pomoću jednostavnih i svima dostupnih programskih jezika i različitih računala, dok napredne simulacije zahtijevaju posebne programske jezike i mogu se pokrenuti samo na super-računalima. U načelu možemo simulirati sve pokuse, ali u praksi smo ograničeni mogućnostima programskih jezika i snagom računala. Primjerice, simulacije kretanja galaksija sa svim nebeskim tijelima ili kretanja svih atoma u trkaćem automobilu je nemoguće izvesti, ali zato možemo pojednostavniti te probleme i izostaviti dijelove koji daju zanemarive doprinose odvijanju simulacije. Na primjer, možemo izbaciti sva tijela manja od Mjeseca ili pak sustave zamijeniti zvijezdom malo veće mase. U pravilu, tražimo najjednostavniju simulaciju koja daje dobre rezultate. Ukoliko naš model gdje su sustavi zamijenjeni zvijezdama ne daje zadovoljavajuće rezultate, morali bismo uključiti planete, pa po potrebi i satelite, komete i ostala tijela. Dakle, prirodno je da se ide od jednostavnog prema složenom.

Vodeći se tom idejom, i u ovom radu krenut ćemo od jednostavnijih prema složenijim simulacijama. Točnije, simulirat ćemo jednostavan pokus koji lako možemo rekonstruirati i vidjeti poklapaju li se rezultati simulacije s rezultatima pokusa. Ukoliko je simulacija dobra, možemo je koristiti kao dio kompliciranijih simulacija. Odabrali smo simulaciju gibanja zvrka, pošto su se svi u svom životu susreli sa zvrkom. Kada smo odabrali pokus koji želimo simulirati, moramo odabrati i programski jezik u kojem ćemo tu simulaciju programirati. Odabrali smo programski jezik Python 3, koji je jednostavan, besplatan i nudi veliki izbor biblioteka za simulacije. Alternativno, mogli bismo koristiti i druge programske jezike, kao na primjer Mathematicu ili Javu. Dio Mathematice „Wolfram demonstrations[*]“ je pogodan za simulacije, ali za korištenje Mathematice potrebno je imati skupu licencu koja zasigurno

[*] Više na https://demonstrations.wolfram.com/

neće biti dostupna većini učenika i nastavnika. U programskom jeziku Java dostupan je „Open source physics[†]“ projekt koji je besplatan i svima dostupan, ali ima nešto težu sintaksu u usporedbi s Pythonom, pa je nastavniku teže programirati i održavati kod ili objasniti ga svojim učenicima.

1.2. Zvrk

Pod pojmom zvrk u fizici podrazumijevamo općenito kruto tijelo. Zvrkovi mogu biti igračke za djecu ili ukrasi, ali ponekad se koriste i u druge svrhe. Iako pod zvrkove spadaju i predmeti koji se ne vrte na podu, poput igračke yo-yo ili mažoret-štapa, mi ćemo se ograničiti na primjere zvrkova kod kojih je jedna točka fiksirana. Kada pripremamo pokuse sa zvrkovima, dobro je odabrati zvrk sa što nižim centrom mase, većom tromosti i manjim trenjem s podlogom – takav zvrk može rotirati nekoliko minuta, pa čak i sati. Trenutni rekord u Guinnessovoj knjizi rekorda je 7 sati, 1 minutu i 14 sekundi [1]. Zvrk možemo pokrenuti direktno rukom, ili može imati dodatni mehanizam. Tradicionalni mehanizmi najčešće uključuju vrpcu omotanu oko zvrka, iako se moderne igračke pokreću pomoću nazubljenog plastičnog štapa. Postoje i neki posebni mehanizmi, poput biča kojim se udara zvrk u istočnoj Aziji. Naša simulacija neće uključivati mehanizam kojim se pokreće zvrk, nego će krenuti od trenutka kada se je zvrk već zavrtio. Također, kako u našoj simulaciji nema trenja, energija zvrka će biti sačuvana.

1.3. Povijest zvrka

Zvrk je jedna od najstarijih igračaka. Najstariji pronađen zvrk je glineni zvrk iz sumerskog grada Ura i star je više od 5500 godina [6]. Pretpostavlja se da su zvrkovi postojali i prije, te da su se prije koristili prirodni zvrkovi, poput žira ili školjke. I onda su ljudi znali da se zvrk duže vrti ako ima veći moment tromosti (iako ne takvom formulacijom), pa su školjke punili voskom ili pijeskom. Zvrk nije izumljen jednom, nego mnogo puta na gotovo svim dijelovima svijeta [7]. Drevni zvrkovi pronađeni su u Egiptu, Grčkoj, Kini, Indiji, Africi, jugoistočnoj Aziji, Pacifičkim otocima i južnoj Americi. Najstariji pisani spomen zvrka je u Homerovoj Ilijadi, u osmom stoljeću prije Krista. Najjednostavniji zvrk koji su ljudi diljem svijeta radili je voće probijeno štapom. Zvrkovi su u različitim dijelovima svijeta napredovali u različitim smjerovima. Zvrkovi u jugoistočnoj

[†] Dostupno na https://www.compadre.org/osp/

Aziji postajali su sve veći (mogu težiti i do 8 kilograma) i bili dio organizirane igre odraslih, u Grčkoj i Rimu koristili su se za prorokovanje, gatanje i klađenje (i bili su mali), u Japanu su školjke vrtili kao dio igre, a na Pacifičkim otocima i u duhovne svrhe. Unatoč tolikoj rasprostranjenosti zvrkova, prvi put su znanstveno opisani tek u 18. stoljeću dolaskom Eulera i Lagrangea, koji su fizički opisali gibanje zvrka kao čvrstog tijela u klasičnoj mehanici. Pomoću zvrka, znanstvenici su naučili mnogo o fizici, posebno o integrabilnim i neintegrabilnim sustavima, kao i prijelazu u kaos. Kod suvremenih zvrkova koristimo nove znanstvene spoznaje da bismo ih unaprijedili, pa se primjerice koriste pokretni mehanički dijelovi, magnetizam i napredni materijali. Iako zvrkovi zabavljaju djecu već tisućljećima, oni nisu zastarjeli. Naprotiv, i danas, u malo novijim izdanjima, uspijevaju zadobiti njihovu pažnju, pa tako imamo Beyblade, Infinity Nado i Lego Spinjitzu, kao i razne vrste tradicionalnih i modernih zvrkova.

2. Opis gibanja

2.1. Eulerove jednadžbe

Kako bi simulacija na kojoj radimo bila što bolja, potrebno je dobro proučiti fizikalni problem prije njene izrade. Problem zvrka može se proučavati iz dva sustava: fiksnog sustava izvan zvrka te iz sustava samog zvrka. Budući da je složenije primjere zvrkova jednostavnije riješiti u sustavu zvrka, sve primjere ćemo sustavno proučavati iz sustava zvrka. Do rješenja ćemo doći koristeći Eulerove jednadžbe. Odabrat ćemo sustav vezan uz zvrk takav da se osi sustava podudaraju s glavnim osima sustava. Jednadžbe gibanja zvrka u odnosu na vanjski sustav su sljedeće:

$$\frac{d\vec{P}}{dt} = \frac{d'\vec{P}}{dt} + \vec{\Omega} \times \vec{P} = \vec{F}, \tag{2.1}$$

$$\frac{d\vec{M}}{dt} = \frac{d'\vec{M}}{dt} + \vec{\Omega} \times \vec{M} = \vec{N}. \tag{2.2}$$

U prethodnim jednadžbama operator d'/dt označava derivaciju u pomičnom sustavu. $\vec{P} = \mu\vec{V}$ odgovara količini gibanja, a $\vec{M} = I\vec{\Omega}$ momentu količine gibanja zvrka. Jednadžbu (2.1) raspisat ćemo po komponentama:

$$\frac{d'V_1}{dt} + \Omega_2 V_3 - \Omega_3 V_2 = \frac{F_1}{\mu}, \tag{2.3}$$

$$\frac{d'V_2}{dt} + \Omega_3 V_1 - \Omega_1 V_3 = \frac{F_2}{\mu}, \tag{2.4}$$

$$\frac{d'V_3}{dt} + \Omega_1 V_2 - \Omega_2 V_1 = \frac{F_3}{\mu}. \tag{2.5}$$

Budući da smo odabrali sustav takav da se glavne osi zvrka podudaraju s osima sustava, vrijedi:

$$\vec{M} = I_1\Omega_1\hat{x} + I_2\Omega_2\hat{y} + I_3\Omega_3\hat{z}, \tag{2.6}$$

pa jedn. (2.2) raspisana po komponentama glasi:

$$I_1\frac{d'\Omega_1}{dt} + \Omega_2\Omega_3(I_3 - I_2) = N_1, \tag{2.7}$$

$$I_2\frac{d'\Omega_2}{dt} + \Omega_1\Omega_3(I_1 - I_3) = N_2, \tag{2.8}$$

$$I_3\frac{d'\Omega_3}{dt} + \Omega_1\Omega_2(I_2 - I_1) = N_3. \tag{2.9}$$

Jednadžbe (2.3-2.5) i (2.7-2.9) su Eulerove jednadžbe, gdje prve tri opisuju gibanje točke vanjskog sustava u odnosu na zvrk, a druge tri vrtnju okoline u odnosu na zvrk. U ovom radu promatrat ćemo samo primjere zvrkova koji imaju jednu točku fiksiranu i u toj točki bit će ishodišta pomičnog i nepomičnog sustava. Stoga jedn (2.3)-(2.5) koje opisuju translaciju zvrka postaju nebitne, dok su relevantne jednadžbe koje opisuju rotaciju (2.7)-(2.9).

2.2. Slobodni simetrični zvrk

Simetrični zvrk je onaj kojemu su dva dijagonalna elementa tenzora tromosti jednaka, dakle vrijedi:

$$I \equiv I_1 = I_2. \tag{2.10}$$

Slobodan zvrk je onaj na koji ne djeluje nikakav moment sile, odnosno:

$$\dot{\vec{M}} = \vec{N} = 0 \Rightarrow \vec{M} = konst. \tag{2.11}$$

Sada trebamo riješiti Eulerove jednadžbe, koje su u ovom slučaju jednostavne

$$I\frac{d\Omega_1}{dt} + (I_3 - I)\Omega_2\Omega_3 = 0, \tag{2.12}$$

$$I\frac{d\Omega_2}{dt} + (I - I_3)\Omega_3\Omega_1 = 0, \tag{2.13}$$

$$I_3 \frac{d\Omega_3}{dt} = 0. \tag{2.14}$$

Rješenje jedn. (2.14) nije vezano s jedn. (2.12) i (2.13) i jednako je konstanti

$$\Omega_3 = \Omega_{30} = konst. \tag{2.15}$$

Dakle, kutna brzina oko glavne osi, tj. osi simetrije zvrka je konstantna, odnosno jednaka Ω_{30}. Sada ćemo definirati još jednu konstantu iz konstanti koje već imamo

$$\omega \equiv \Omega_{30} \frac{I_3 - I}{I}. \tag{2.16}$$

Dvije jednadžbe (2.12) i (2.13) možemo zapisati kao

$$\dot{\Omega}_1 = -\omega\Omega_2, \tag{2.17}$$

$$\dot{\Omega}_2 = \omega\Omega_1. \tag{2.18}$$

Sada možemo derivirati jedn. (2.17) po vremenu i uvrstiti $\dot{\Omega}_2$ iz jedn. (2.18) te dobijemo

$$\ddot{\Omega}_1 + \omega^2\Omega_1 = 0, \tag{2.19}$$

što prepoznajemo kao jednadžbu harmoničkog oscilatora čija rješenja znamo

$$\Omega_1 = \Omega_0 \cos \omega t, \tag{2.20}$$

$$\Omega_2 = \Omega_0 \sin \omega t. \tag{2.21}$$

Rješenje Eulerovih jednadžbi za slobodni simetrični zvrk odgovaraju jednoj konstantnoj te dvjema oscilirajućim komponentama vektora kutne brzine $\vec{\Omega}$. Ako pogledamo vektorski zbroj oscilirajućih komponenti

$$\vec{\Omega}_0 = \Omega_1\hat{x} + \Omega_2\hat{y} = \Omega_0[\cos(\omega t)\,\hat{x} + \sin(\omega t)\hat{y}], \tag{2.22}$$

možemo primijetiti da je veličina tog vektora $\Omega_0 = \sqrt{\Omega_1^2 + \Omega_2^2}$ stalna. Dakle, vektor $\vec{\Omega}_0$ rotira kutnom brzinom ω iz jedn. (2.16).

2.3. Slobodni asimetrični zvrk

Kod asimetričnog zvrka, jedn. (2.10) više ne vrijedi, ali bez gubitka općenitosti možemo pretpostaviti sljedeći odnos momenata tromosti

$$I_3 > I_2 > I_1. \tag{2.23}$$

Kako problem promatramo u sustavu zvrka, trebamo provjeriti zakone očuvanja. Krećemo od sačuvanja momenta količine gibanja u vanjskom sustavu

$$\frac{d\vec{M}}{dt} = \frac{d'\vec{M}}{dt} + \vec{\Omega} \times \vec{M} = \vec{N} = 0. \tag{2.24}$$

Lako se možemo uvjeriti da je u sustavu zvrka očuvan samo iznos momenta količine gibanja. Možemo pomnožiti jedn. (2.24) skalarno s $\vec{M}$

$$\vec{M} \cdot \frac{d'\vec{M}}{dt} + \vec{M} \cdot \left(\vec{\Omega} \times \vec{M}\right) = 0 => \frac{1}{2}\frac{d'\vec{M}^2}{dt} = 0. \tag{2.25}$$

Ako su M_1, M_2 i M_3 komponente momenta količine gibanja u sustavu zvrka, vrijedi

$$M_1^2 + M_2^2 + M_3^2 \equiv \vec{M}^2 = konst. \tag{2.26}$$

Dakle, iznos momenta količine gibanja je očuvan. Da bismo pokazali sačuvanje energije, množimo jedn. (2.24) skalarno s $\vec{\Omega}$

$$\vec{\Omega} \cdot \frac{d'\vec{M}}{dt} = \vec{\Omega}_1 \cdot \frac{d'\vec{M}_1}{dt} + \vec{\Omega}_2 \cdot \frac{d'\vec{M}_2}{dt} + \vec{\Omega}_3 \cdot \frac{d'\vec{M}_3}{dt} = 0. \tag{2.27}$$

Znamo da vrijedi $M_i = I_i\Omega_i$ pa jedn. (2.27) možemo zapisati kao

$$\frac{d'}{dt}\left(\frac{M_1^2}{I_1} + \frac{M_2^2}{I_2} + \frac{M_3^2}{I_3}\right) = 0, \tag{2.28}$$

gdje je vrijednost u zagradi

$$\frac{M_1^2}{I_1} + \frac{M_2^2}{I_2} + \frac{M_3^2}{I_3} = 2E = konst. \tag{2.29}$$

te je i energija sačuvana. Ukratko, došli smo do dva uvjeta na komponente momenta količine gibanja M_1, M_2 i M_3 od kojih prvi predstavlja jednadžbu sfere, a drugi jednadžbu elipsoida:

$$M_1^2 + M_2^2 + M_3^2 \equiv M^2, \tag{2.30}$$

$$\frac{M_1^2}{2I_1E} + \frac{M_2^2}{2I_2E} + \frac{M_3^2}{2I_3E} = 1. \tag{2.31}$$

Sjecište sfere iz jedn. (2.30) i elipsoida iz jedn. (2.31) postoji samo ako vrijedi

$$2EI_1 \leq M^2 \leq 2EI_3. \tag{2.32}$$

Eulerove jednadžbe za slobodni asimetrični zvrk, koje koristimo u programu su sljedeće:

$$\dot{\Omega}_1 = \frac{I_2 - I_3}{I_1}\Omega_2\Omega_3, \tag{2.33}$$

$$\dot{\Omega}_2 = \frac{I_3 - I_1}{I_2}\Omega_3\Omega_{1,} \tag{2.34}$$

$$\dot{\Omega}_3 = \frac{I_1 - I_2}{I_3}\Omega_1\Omega_2. \tag{2.35}$$

Najjednostavnija rješenja Eulerovih jednadžbu za slobodni asimetrični zvrk su vrtnje oko glavnih osi:

$$\Omega_{10} \neq 0, \qquad \Omega_{20} = \Omega_{30} = 0, \tag{2.36}$$

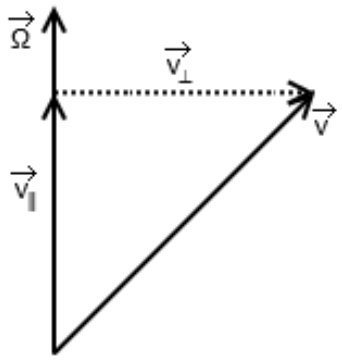
$$\Omega_{20} \neq 0, \qquad \Omega_{30} = \Omega_{10} = 0, \tag{2.37}$$

$$\Omega_{30} \neq 0, \qquad \Omega_{10} = \Omega_{20} = 0. \tag{2.38}$$

Može se pokazati da su rješenja (2.36) i (2.37) stabilna, dok je rješenje (2.38) nestabilno [3]. Eulerove jednadžbe za slobodan asimetrični zvrk mogu se riješiti analitički, a rješenja pojedinih komponenti kutne brzine odgovaraju Jacobijevim eliptičkim funkcijama. Budući da je izvod složen, ovdje ga ne navodimo, iako ćemo analitička rješenja kasnije u radu koristiti za provjeru numeričkih rješenja. Detalji izvoda mogu se pronaći u literaturi [3].

2.4. Rekonstrukcija gibanja zvrka iz Eulerovih jednadžbi

Rješenja Euleorovih jednadžbi su komponente kutne brzine, Ω_1, Ω_2 i Ω_3. Kada znamo njih, možemo rekonstruirati gibanje zvrka. Vektor kutne brzine $\vec{\Omega}$ jednak je u fiksnom sustavu i u sustavu zvrka, te pomoću njega možemo zarotirati bilo koji vektor. Kao primjer, uzmimo vektor $\vec{v}$, koji možemo rastaviti na komponente $\vec{v}_\perp$ i $\vec{v}_\parallel$ u odnosu na vektor $\vec{\Omega}$, kao na slici 2.1. Za njih tada vrijedi:



Slika 2.1. Rastavljanje vektora $\vec{v}$ na komponente

$$\vec{v} = \vec{v}_\perp + \vec{v}_\parallel = (\vec{v} \cdot \hat{\Omega})\hat{\Omega} + \vec{v} - (\vec{v} \cdot \hat{\Omega})\hat{\Omega} = (\vec{v} \cdot \hat{\Omega})\hat{\Omega} - \hat{\Omega} \times (\hat{\Omega} \times \vec{v}), \tag{2.39}$$

gdje je $\hat{\Omega} = \frac{\vec{\Omega}}{|\vec{\Omega}|}$ jedinični vektor. Nadalje

$$\vec{v}_{\parallel,rot} = \vec{v}_\parallel, \tag{2.40}$$

$$\vec{v}_{\perp,rot} = \cos\delta\, \vec{v}_\perp + \sin\theta\, \hat{\Omega} \times \vec{v}_\perp = \cos\delta\, \vec{v}_\perp + \sin\delta\, \hat{\Omega} \times \vec{v}, \tag{2.41}$$

$$\vec{v}_{rot} = \vec{v}_\parallel + \cos\delta\, (\vec{v} - \vec{v}_\parallel) + \sin\delta\, \hat{\Omega} \times \vec{v}, \tag{2.42}$$

$$\vec{v}_{rot} = \vec{v}\cos\delta + \left(\vec{v}\cdot\hat{\Omega}\right)\hat{\Omega}(1-\cos\delta) + \sin\delta\,\hat{\Omega}\times\vec{v}. \quad (2.43)$$

Sada smo dobili rotaciju vektora $\vec{v}$ pod uvjetom da je $\vec{\Omega}$ konstantni vektor. No, ako vrijeme podijelimo na kratke intervale, u svakom kratkom intervalu $\vec{\Omega}$ je približno konstantan, pa možemo koristiti formulu iz jedn. (2.43). Tu formulu zovemo Rodriguesova formula. Primjenom na jedinične vektore sustava vezanog uz zvrk, dobit ćemo:

$$\hat{x}_{rot} = \hat{x}\cos\delta + \left(\hat{x}\cdot\hat{\Omega}\right)\hat{\Omega}(1-\cos\delta) + \sin\delta\,\hat{\Omega}\times\hat{x}, \quad (2.44)$$

$$\hat{y}_{rot} = \hat{y}\cos\delta + \left(\hat{y}\cdot\hat{\Omega}\right)\hat{\Omega}(1-\cos\delta) + \sin\delta\,\hat{\Omega}\times\hat{y}, \quad (2.45)$$

$$\hat{z}_{rot} = \hat{z}\cos\delta + \left(\hat{z}\cdot\hat{\Omega}\right)\hat{\Omega}(1-\cos\delta) + \sin\delta\,\hat{\Omega}\times\hat{z}. \quad (2.46)$$

Orijentaciju pomičnog ustava naspram fiksnom obično opisujemo Eulerovim kutovima θ, φ i ψ. Ovdje nećemo ulaziti u detalje, budući da su dostupni u literaturi [5]. Ovisnost Eulerovog kuta θ o vremenu možemo dobiti jednostavno:

$$\cos\theta = \widehat{z'}\cdot\hat{z}_{rot}. \quad (2.47)$$

2.5. Gibanje zvrka u polju sile teže

Do sada smo promatrali slobodne zvrkove bez utjecaja vanjskih sila, odnosno momenata sila. Naravno, u realističnom slučaju će na zvrk djelovati sila teža pa se pojavljuje i moment sile s obzirom na fiksiranu točku zvrka, odnosno ishodište pomičnog sustava $\vec{N} = \vec{d}\times m\vec{g}$, gdje je $\vec{d}$ vektor koji ima hvatište u ishodištu sustava, a kraj u centru mase zvrka. m je masa zvrka, a $\vec{g}$ je ubrzanje sile teže. Kada to uzmemo u obzir, Euleorve jednadžbe postaju:

$$\dot{\Omega}_1 = \frac{I_2 - I_3}{I_1}\Omega_2\Omega_3 + \frac{mdg_2}{I_1}, \quad (2.48)$$

$$\dot{\Omega}_2 = \frac{I_3 - I_1}{I_2}\Omega_3\Omega_1 - \frac{mdg_1}{I_2}, \quad (2.49)$$

$$\dot{\Omega}_3 = \frac{I_1 - I_2}{I_3}\Omega_1\Omega_2. \quad (2.50)$$

Jedn (2.50) nema dodatnog člana jer je d_3 paralelan s g_3 pa ta komponenta momenta sile iščezava. Međutim, jedn. (2.48)-(2.50) ne čine zatvoreni sustav jednadžbi, jer se vektor $\vec{g}$ mijenja u vremenu. Stoga sustavu (2.48)-(2.50) moramo dodati još 3 jednadžbe:

$$\dot{g}_1 = \Omega_3 g_2 - \Omega_2 g_3, \quad (2.51)$$

$$\dot{g}_2 = \Omega_1 g_3 - \Omega_3 g_1, \quad (2.52)$$

$$\dot{g}_3 = \Omega_2 g_1 - \Omega_1 g_2, \quad (2.53)$$

čime smo dobili sustav od 6 vezanih diferencijalnih jednadžbi. Pri izradi simulacije prikladno je koristiti bezdimenzionalne veličine:

$$t = \frac{t}{\sqrt{\frac{d}{g}}}, \quad \tilde{I}_i = \frac{I_i}{md^2}, \quad \tilde{\Omega}_i = \frac{\Omega_i}{\sqrt{\frac{g}{d}}}. \tag{2.54}$$

Uvodimo i vektora $\vec{\gamma}$; $\vec{g} = -g\vec{\gamma}$, pa jedn. (2.48-2.50) prelaze u

$$\dot{\tilde{\Omega}}_1 = \frac{\tilde{I}_2 - \tilde{I}_3}{\tilde{I}_1}\tilde{\Omega}_2\tilde{\Omega}_3 + \frac{\gamma_2}{\tilde{I}_1}, \tag{2.55}$$

$$\dot{\tilde{\Omega}}_2 = \frac{\tilde{I}_3 - \tilde{I}_1}{\tilde{I}_2}\tilde{\Omega}_3\tilde{\Omega}_1 + \frac{\gamma_1}{\tilde{I}_2} \tag{2.56}$$

$$\dot{\tilde{\Omega}}_3 = \frac{\tilde{I}_1 - \tilde{I}_2}{\tilde{I}_3}\tilde{\Omega}_1\tilde{\Omega}_2. \tag{2.57}$$

3. Simulacija slobodnog simetričnog zvrka

3.1. Korišteni programski paketi

Simulacije predstavljene u ovom radu napravljene su pomoću programskog jezika Python[‡] (verzija 3.5.3). Korišteni su paketi Numpy[§] (verzija 1.10.4), Scipy[**] (verzija 0.17.0) i VPython[††] (verzija 7.4.5). Simulacija se otvara u web pregledniku. Sve je rađeno na 64-bitnom Windows 7 operativnom sustavu. Koristili smo Python 3 zato što ima bolju podršku na novijim sustavima u usporedbi s Python 2. Paketi Numpy i Scipy bili su nam potrebni za matematičke operacije, a VPython za grafički prikaz simulacije i grafove. Prozor u kojem se odvija simulacija prilagođen je za FullHD (1920x1080) rezoluciju, iako se može prilagoditi bilo kojoj rezoluciji za vrijeme trajanja programa.

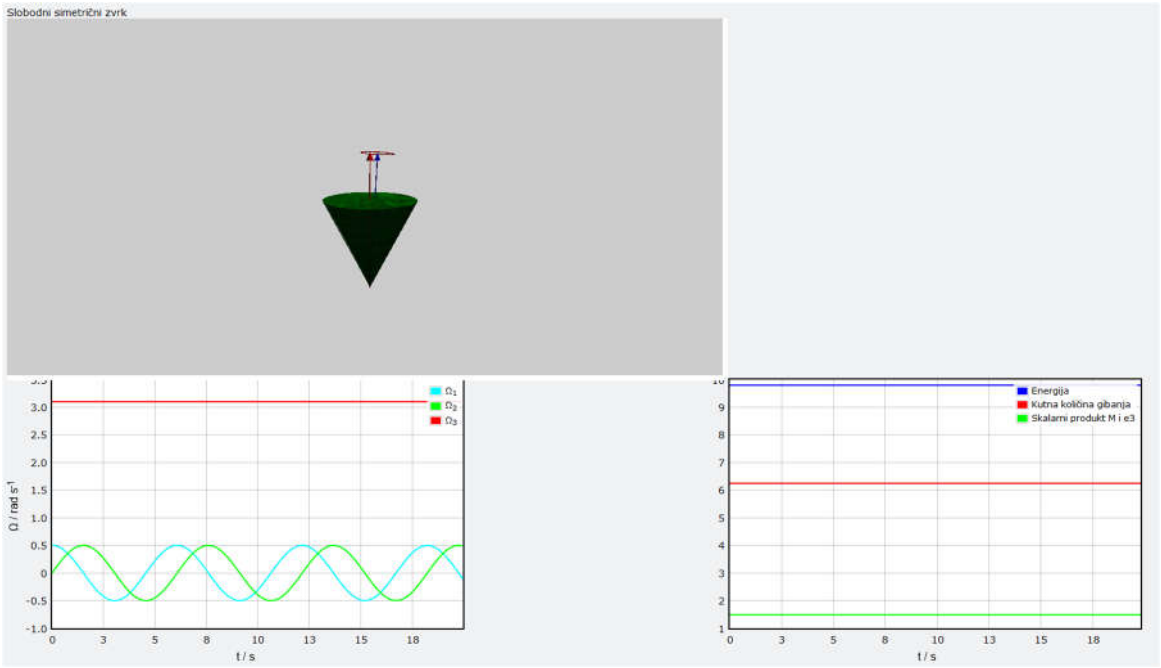
3.2. Ulaz

Program ne traži unos, nego se automatski pokreće sa zadanim parametrima. Unutar programa mogu se mijenjati početni uvjeti: dijagonalni elementi tenzora tromosti (od kojih su dva jednaka zbog simetričnosti), početnu kutnu brzinu (odnosno projekcije na glavne osi), te početne kutove θ, φ i ψ.

[‡] https://www.python.org/
[§] http://www.numpy.org/
[**] https://www.scipy.org/
[††] https://vpython.org/

3.3. Izlaz

Nakon pokretanja programa, prvo se ispisuju početna energija i moment količine gibanja, te provjera kutnih brzina dobivena integracijom unatrag[‡‡]. Ukoliko su vrijednosti iste (ili približno iste), program je dobro izračunao kutne brzine. Nakon nekoliko sekundi, u zadanom web pregledniku otvara se nova kartica u kojoj se prikazuje simulacija koja traje 20 sekundi. Primjer simulacije možemo vidjeti na slici 3.1. U prvom planu je simulacija gibanja samog zvrka, koji je u svim simulacijama radi jednostavnosti prikazan kao stožac, iako VPython podržava i razne druge oblike. Za vrijeme trajanja simulacije, a i kasnije, možemo mijenjati pogled na zvrk, zumirati i mijenjati veličinu prozora simulacije. Os simetrije zvrka ostavlja trag tako da možemo vidjeti kako se kreće. Osim samog zvrka, nacrtan je i vektor momenta količine gibanja koji je konstantan budući da je zvrk slobodan.



Slika 3.1. Prikaz simulacije slobodnog simetričnog zvrka.

Ispod prozora sa simulacijom nalaze se dva grafa koji se iscrtavaju kako se simulacija odvija. Prvi graf prikazuje ovisnost komponenata kutne brzine u sustavu zvrka o vremenu, dok drugi graf prikazuje konstante gibanja: energiju i moment količine gibanja. Ako preko grafova prijeđemo mišem, pokazat će se vrijednosti nacrtanih veličina u danom vremenskom trenutku.

[‡‡] Integracija unatrag objašnjena je u sljedećem poglavlju.

3.4. Tijek programa

Program možemo podijeliti na nekoliko dijelova. Prvi dio programa je postavljanje početnih uvjeta i računanje veličina koje možemo direktno dobiti iz početnih uvjeta (energija i moment količine gibanja).

U drugom dijelu numeričkim rješavanjem sustava diferencijalne jednadžbe dolazimo do polja koje sadrži sve kutne brzine koje su nam potrebne za simulaciju. Prije rješavanja samog sustava diferencijalnih jednadžbi, definiramo vrijeme simulacije (20 sekundi) i broj koraka (40000). Povećanjem broja koraka možemo poboljšati simulaciju, ali pritom trebamo osigurati i više računalnih resursa. Idealno, koristimo najveći broj za koji se simulacija odvija glatko. Ako postavimo 40000, to znači da će se simulacija računati 2000 puta u sekundi. Nakon postavljanja vremena i koraka, numerički rješavamo sustav diferencijalnih jednadžbi (koji je u ovom slučaju jednostavan) iz početnih uvjeta i pospremamo rješenja u memoriju. U tu svrhu koristimo metodu *odeint* iz Scipy paketa. Metoda *odeint* kao argumente uzima funkciju *f (y , x)* u kojoj su definirani međusobni odnosi varijabli i njihovih derivacija, polje početnih uvijeta y0 i polje t vrijednosti za koje želimo rješenja. Rezultat te metode je dvodimenzijalno polje y koje u sebi sadrži sve vrijednosti svih varijabli, tako da se u nultom retku (y[:,0]) nalaze sva rješenja prve varijable, a u nultom stupcu (y[0,:]) nalaze rješenja svih varijabli u prvom koraku. Nakon što smo dobili rezultate, provjeravamo ih integracijom unatrag. Za integraciju unatrag koristili smo istu metodu *odeint*, ali smo krenuli od kraja prema početku. Ako se početna vrijednost i vrijednost dobivena integracijom unatrag bitno razlikuju, treba povećati preciznost numeričkog algoritma rješavanja diferencijalnih jednadžbi.

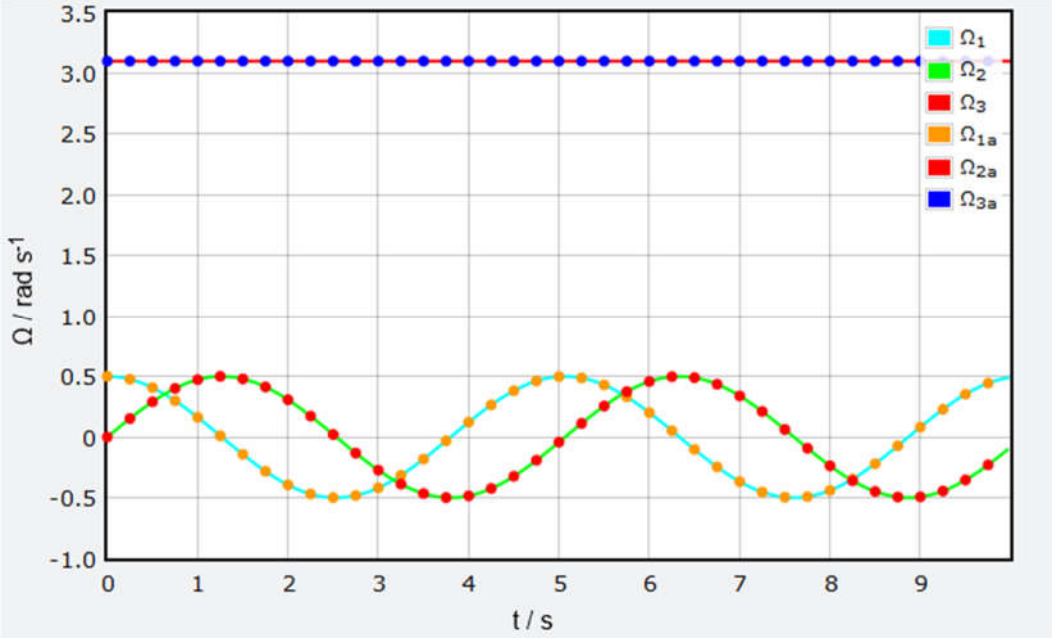
U trećem dijelu programa stvaramo prostor na kojemu će se izvoditi simulacija. Ovdje odabiremo oblik zvrka, boje, okvire grafova i slične varijable. Ovaj dio pretežno se svodi na postavljanje VPython-a. Prvo definiramo prostor na kojem će VPython prikazivati rezultate. To radimo metodom *canvas* iz VPython paketa kojoj kao argumente dodajemo naslov, dimenzije i boju pozadine. Zatim definiramo jedinične vektore $\hat{e}_1$, $\hat{e}_2$ i $\hat{e}_3$, vektor osi simetrije zvrka, te vektor momenta količine gibanja koje koristimo kasnije u programu. Slijedi stvaranje zvrka metodom *cone* iz paketa VPython, kojoj kao argumente prosljeđujemo teksturu, položaj zvrka kao vektor kojem ima početak u središtu baze a kraj u vrhu stošca, veličinu zvrka, boju, transparentnost i ostavljanje traga vektora. Zatim prikazujemo os simetrije zvrka metodom *arrow* paketa VPython, kojoj kao argumente

pridružujemo teksturu, položaj, duljinu, boju, orijentaciju, prozirnost i širinu vrha strelice, te konačno istom metodom i vektor momenta količine gibanja.

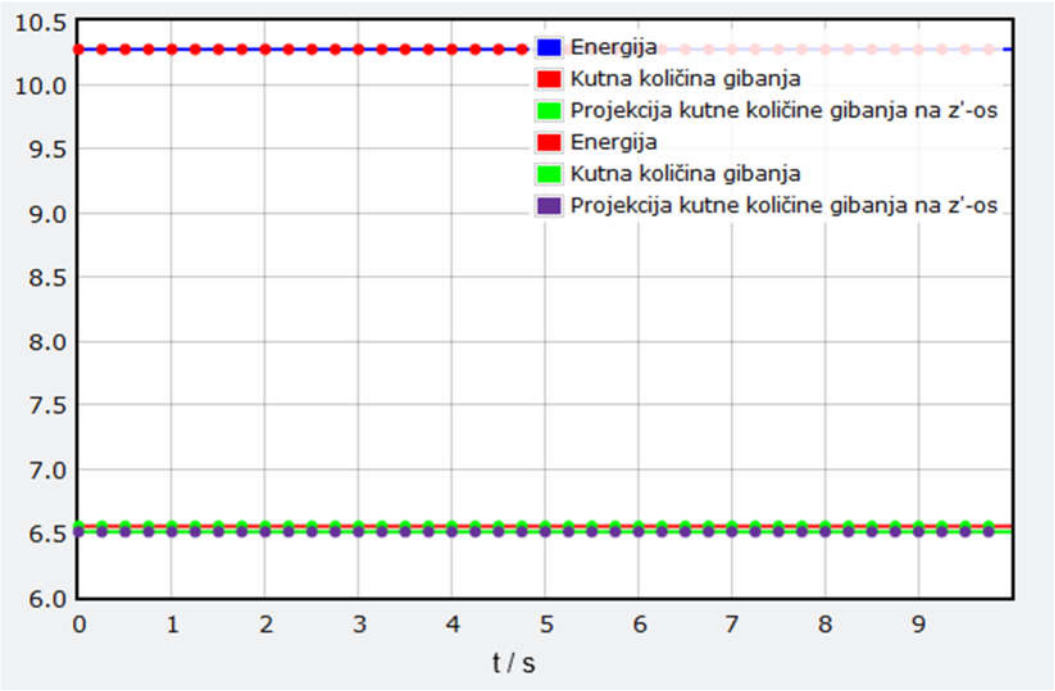
Četvrti, ujedno i zadnji dio programa zadužen je za prikaz simulacije. Koristeći ranije izračunate kutne brzine, okrećemo zvrk i prikazujemo podatke u tablicu. Sve se odvija u jednoj for petlji, a na početku te petlje ograničimo broj operacija naredbom *rate* iz paketa VPython, tako da se u jednoj sekundi simulira jedna sekunda gibanja. Da bismo mogli zarotirati zvrk, prvo trebamo odrediti trenutni vektor kutne brzine $\vec{\Omega}$, kut rotacije, te trenutni vektor momenta količine gibanja. Naredbom *rotate* iz istog paketa okrećemo zvrk gdje su argumenti iznos kuta za koji rotiramo i os oko koje rotiramo, a naredbom *plot* istog paketa dodajemo točke na grafovima. Naredbi *plot* kao argumente pridružujemo koordinate točke koju prikazujemo.

3.5. Provjera simulacija

Da se uvjerimo da su naši rezultati dobri, usporediti ćemo ih s analitičkim rješenjima. Usporediti ćemo kutne brzine i konstante gibanja, koje se u slučaju slobodnog simetričnog zvrka energija, kutna količina gibanja i projekcija kutne količine gibanja na z-os sustava zvrka. Na slikama 3.2. i 3.3. možemo vidjeti kutne brzine, odnosno konstante. Numeričke vrijednosti prikazane su kao linije, a analitički rezultati kao točke. Možemo primijetiti da se sve točke nalaze na linijama, pa zaključujemo da se simulirani rezultati slažu sa očekivanima.



Slika 3.2. Usporedba kutnih brzina



Slika 3.3. Usporedba konstanta gibanja

4. Simulacija slobodnog asimetričnog zvrka

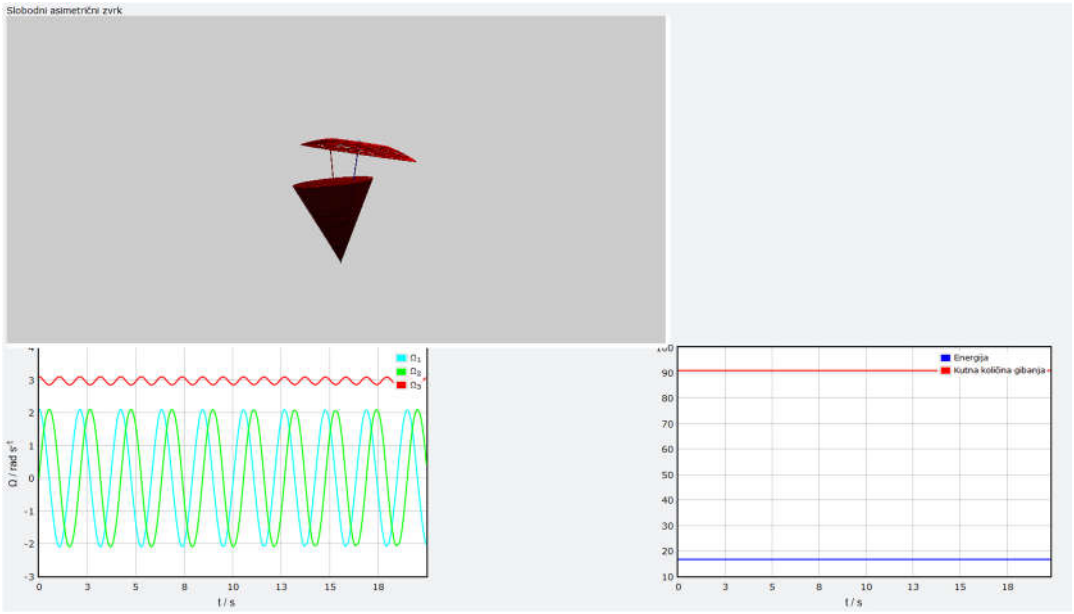
4.1. Ulaz

Kao i kod simulacije slobodnog simetričnog zvrka, program ne traži ulaz. Postoji mogućnost mijenjanja početnih uvjeta: vrijednosti dijagonalnih elementa tenzora tromosti, komponente početne kutne brzine te početnih Eulerovih kutova θ, φ i ψ.

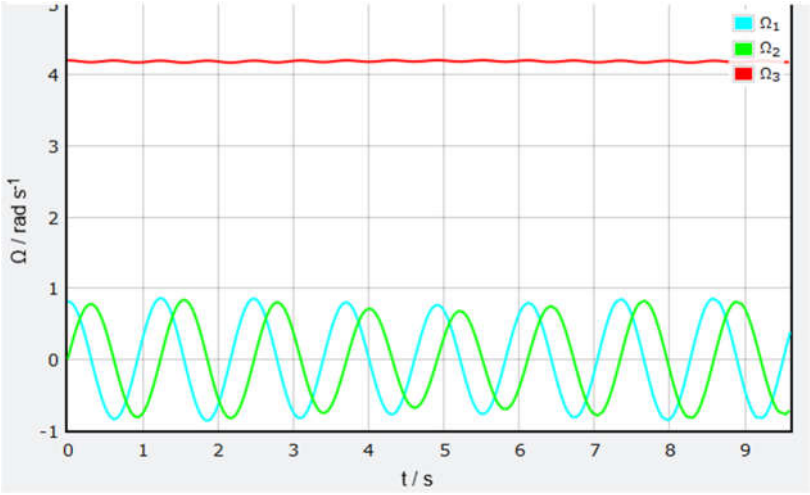
4.2. Izlaz

Sučelje ovog programa organizirano je kao i ono prošlog. Nakon pokretanja, prvo se ispisuju početna vrijednost energije i momenta količine gibanja. Nakon toga, ispisuju se i početne vrijednosti komponenata kutne brzine zajedno sa vrijednostima dobivenim integracijom unazad. Što su vrijednosti bliže, to su naša rješenja preciznija. U prosjeku, vrijednost su jednake do na sedmu decimalu. Nakon toga u web pregledniku otvara se simulacija, kao na slici 4.1. U gornjem lijevom kutu nalazi se simulacija gibanja slobodnog asimetričnog zvrka. Pošto je zvrk asimetričan, prikazan je kao eliptični stožac. Simulacija traje 20 sekundi, u kojima se zvrk vrti i ostavlja trag osi rotacije. Osim zvrka, prikazan je i vektor momenta količine gibanja. Prostor simulacije u bilo kojem trenutku možemo poveća

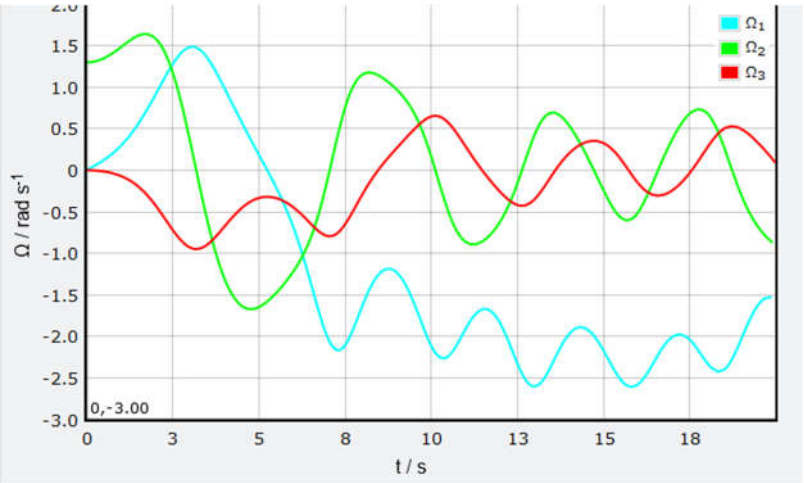
ili smanjiti, zumirati ili mijenjati kut pod kojim promatramo. Ispod prostora za simulaciju, nalaze se dva grafa. Lijevi graf prikazuje vrijednosti komponenta kutne brzine zvrka kroz vrijeme, i puni se paralelno s izvođenjem simulacije. Desni graf se također puni paralelno s



Slika 4.1. Prikaz simulacije asimetričnog zvrka



Slika 4.2. Stabilno gibanje slobodno asimetričnog zvrka



Slika 4.3. Nestabilno gibanje slobodno asimetričnog zvrka

izvođenjem simulacija, a prikazuje konstante gibanja: energiju i vrijednost kutne količine gibanja. Ovisno o početnim uvjetima, ovdje možemo razaznati dva tipa rješenja: stabilno kao na slici 4.2. te nestabilno kao na slici 4.3 [2].

4.3. Tijek programa

Ovaj program, analogno prošlom, podijeliti ćemo na četiri dijela.

U prvom dijelu, definiramo početne uvijete, što uključuje dijagonalne vrijednosti tenzora tromosti, početne kutne brzine te početne kutove θ, φ i ψ. Vrijednosti tenzora tromosti moramo izabrati sukladno jedn. (2.23), odnosno I3 > I2 > I1. O20 postavljamo na nulu sukladno teoriji iz [3]. Kutove možemo mijenjati kako želimo. Zatim računamo komponente vektora γ, početnu energiju E pomoću jednadžbe

$$E = \frac{1}{2}(I_1\Omega_{10}^2 + I_2\Omega_{20}^2 + I_3\Omega_{30}^2), \tag{4.1}$$

i vrijednost početnog momenta količine gibanja M pomoću jednadžbe

$$M = \sqrt{(I_1\Omega_{10})^2 + (I_2\Omega_{20})^2 + (I_3\Omega_{30})^2}, \tag{4.2}$$

te ih ispisujemo.

Drugi dio odnosi se na rješavanje diferencijalnih jednadžbi. Prvi korak je, kao i u prošlom programu, da odredimo vrijeme i broj koraka simulacije. Iz tih informacija određujemo diferencijal vremena dt, te stvaramo polje koje će sadržavati sve trenutke simulacije. Za rješavanje diferencijalne funkcije opet ćemo koristiti metodu *odeint* iz paketa

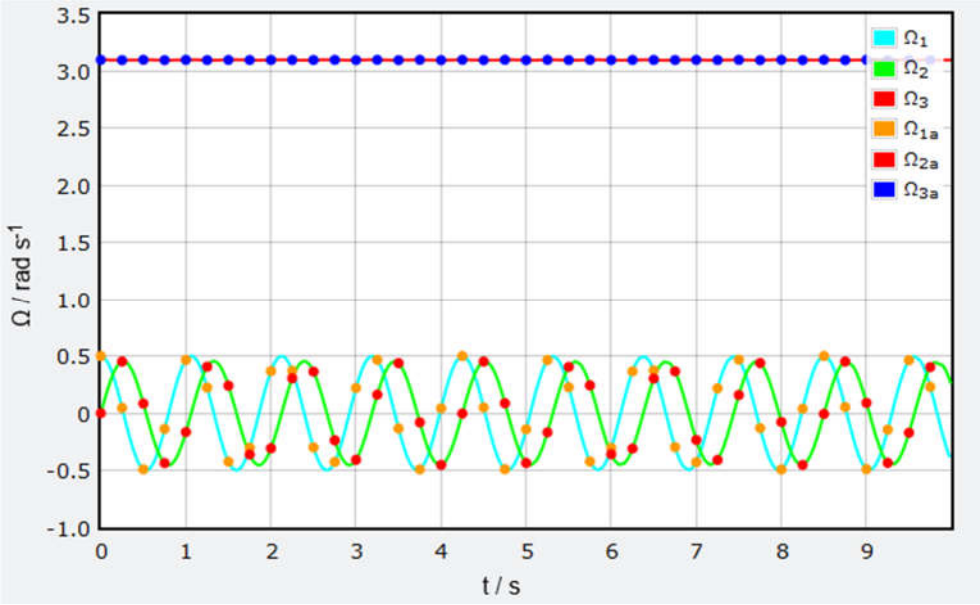
Scipy. U ovom slučaju imamo tri diferencijalne jednadžbe. Umjesto dvije varijable O1 i O2, sada imamo 3 varijable: O1, O2 i O3, pa nam je stoga funkcija $f(y, x)$ promijenjena. Glavna razlika u odnosu na primjer simetričnog slobodnog zvrka je da treća komponenta kutne brzine više neće biti konstantna. Nakon rješavanja diferencijalnih jednadžbi iz poglavlja 2.2 (jedn. 2.33-2.35), provjeravamo rješenja integracijom unatrag, koja je opisana u poglavlju 3.4.

Treći dio ima samo nekoliko izmjena u odnosu na simetrični zvrk. Promijenili smo naslov, boju zvrka u crvenu da ih lakše razaznajemo i oblik zvrka u eliptični stožac.

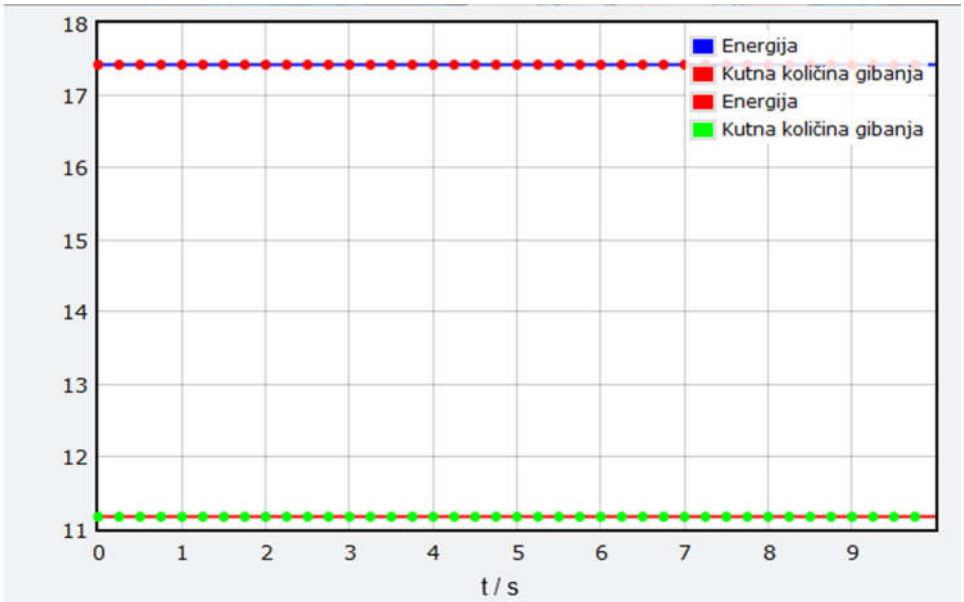
Konačno, u četvrtom dijelu promijenili smo jednadžbe za energiju i kutnu količinu gibanja sukladno jednadžbama (4.1) i (4.2).

4.4. Provjera simulacije

Kao i kod simetričnog zvrka, prvo ćemo provjeriti odgovaraju li naša numerička rješenja očekivanim, analitičkim rješenjima. Usporediti ćemo kutne brzine i konstante. Na slikama 4.4. i 4.5. numerički dobiveni rezultati prikazani su punim linijama, a analitički točkama. Sve točke se nalaze na linijama, što potvrđuje naše rezultate.



Slika 4.4. Usporedba kutnih brzina asimetričnog zvrka



Slika 4.2. Usporedba konstanti asimetričnog zvrka

5. Simulacija zvrka u polju sile teže

5.1. Ulaz

Kod simulacija u polju sile teže, moramo koristiti nešto drugačije početne uvijete. Komponente tenzora tromosti ostaju, kao i Eulerovi kutovi: φ, θ i ψ, ali sada imamo i njihove derivacije: $\dot{\varphi}$, $\dot{\theta}$ i $\dot{\psi}$, koje su bezdimenzionalne. To vrijedi za sva tri slučaja: simetrični zvrk, asimetrični zvrk, te Kovalevskayin zvrk.

5.2. Izlaz

Položaj komponenti programa ostao je kao i prije. Program prvo ispisuje konstante i provjeru integracije unatrag, a zatim prikazuje simulaciju zvrka, graf sa kutnim brzinama i graf sa konstantama. Općenite konstante su energija i projekcija kutne količine gibanja na z-os vanjskog sustava, a u slučaju simetričnog i Kovalevskaya zvrka postoji još po jedna dodatna konstanta gibanja. Kod simetričnog zvrka dodatna konstantna odgovara projekciji kutne količine gibanja na z-os sustava zvrka:

$$M_3 = I_3\Omega_3. \tag{5.1}$$

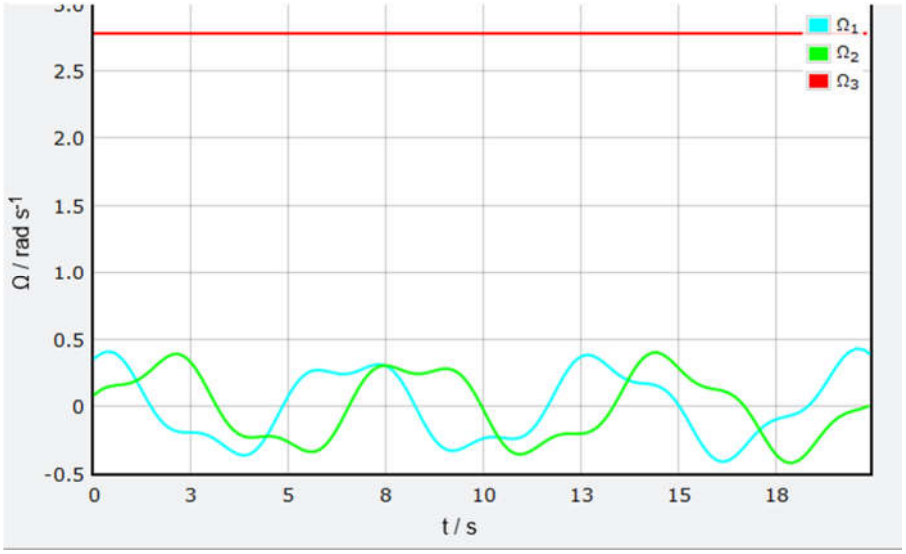
Kovalevskayin zvrk ima nešto složeniju dodatnu konstantu koju zovemo Kovalevskayina konstanta [4]:

$$K = [I_1(\Omega_3^2 - \Omega_2^2) - \gamma_3]^2 + [I_2\Omega_2\Omega_3 - \gamma_2]^2. \tag{5.2}$$

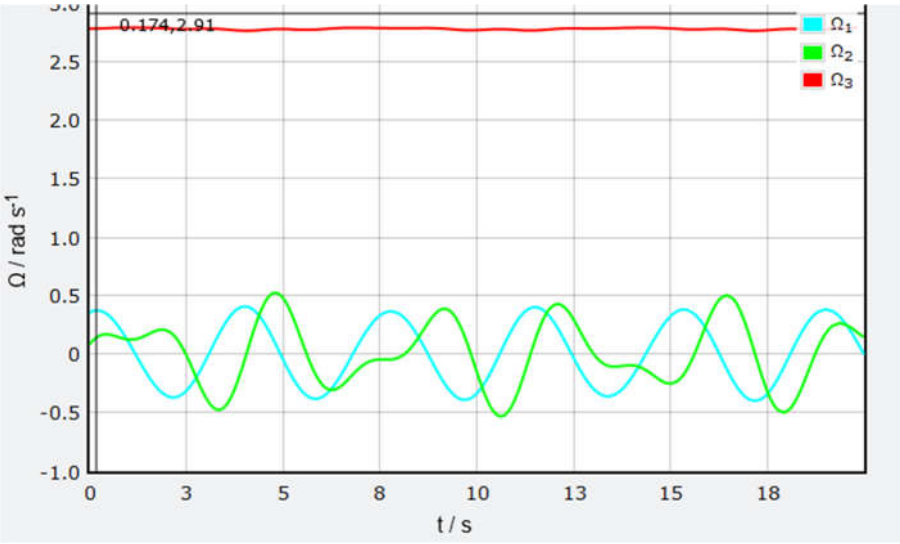
Boju zvrka također smo promijenili, radi lakšeg raspoznavanja. Simetrični zvrk je žuti, asimetrični je svjetlo plavi, a Kovalevskayin narančasti.

Sada možemo i usporediti gibanja različitih zvrkova: pri istim početnim uvjetima (osim, naravno, izmjeni kod komponenta tenzora tromosti) usporediti ćemo kutne brzine i putanje zvrkova. U sljedećim primjerima koristili smo sljedeće početne uvijete: $\varphi = \frac{90\pi}{180}$, $\theta = \frac{33\pi}{180}$, $\psi = \frac{16\pi}{180}$, $\dot{\varphi} = \frac{18\pi}{180}$, $\dot{\theta} = \frac{18\pi}{180}$ i $\dot{\psi} = \frac{144\pi}{180}$. Komponente tenzora tromosti bile su $I_1 = 1.5$, $I_2 = 1.5$ i $I_3 = 2.1$ za simetrični, $I_1 = 1.5$, $I_2 = 1.0$ i $I_3 = 2.1$ za asimetrični i $I_1 = 1.0$, $I_2 = 2.0$ i $I_3 = 2.0$ za Kovalevskaya zvrk. Kutne brzine možemo vidjeti na slikama 5.1.-5.3., a putanje na slikama 5.4.-5.6.

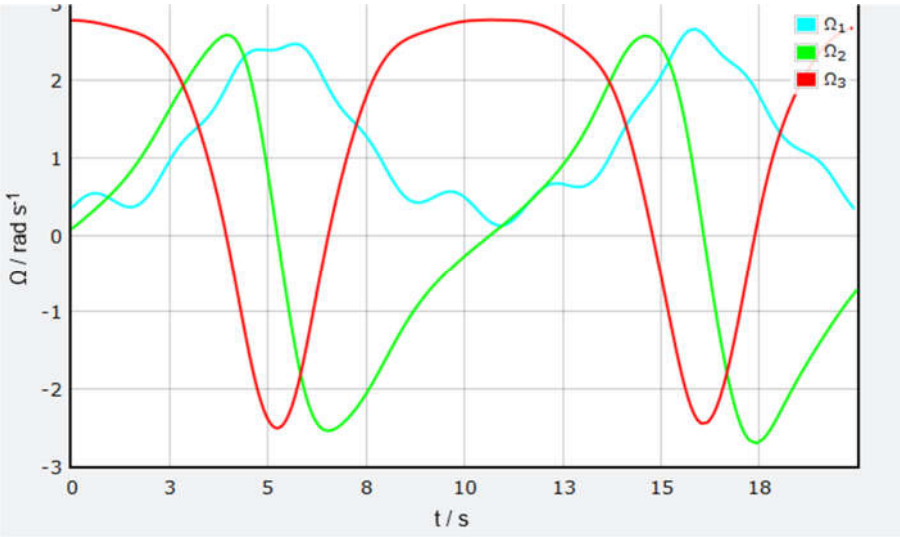
Kod simetričnog zvrka možemo primijetiti da su Ω_1 i Ω_2 jednake do na pomak u fazi. Kod putanje možemo primijetiti simetričnosti. Za asimetrični zvrk Ω_1 i Ω_2 nisu jednake, niti slične jedna drugoj, iako obje osciliraju oko nule. Kovalevskaya zvrk giba se potpuno drugačije. Ω_3 sada ima velike oscilacije oko nule, kao i Ω_2, dok je Ω_1 sada uvijek



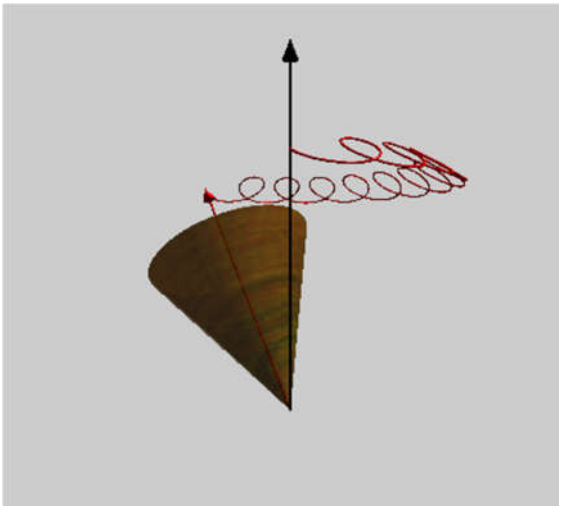
Slika 5.1. Kutne brzine simetričnog zvrka u polju sile teže

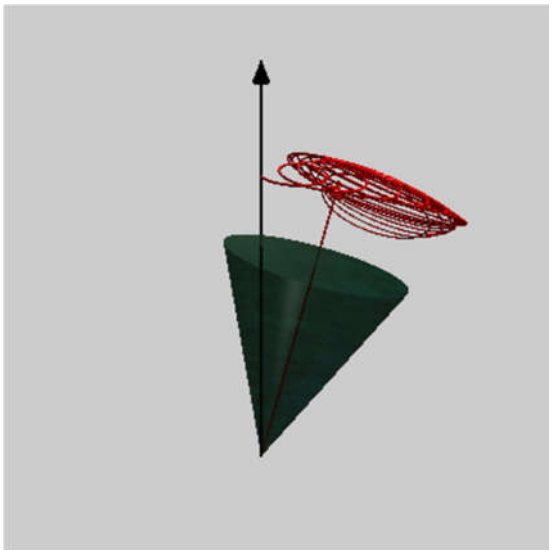


Slika 5.2. Kutne brzine asimetričnog zvrka u polju sile teže

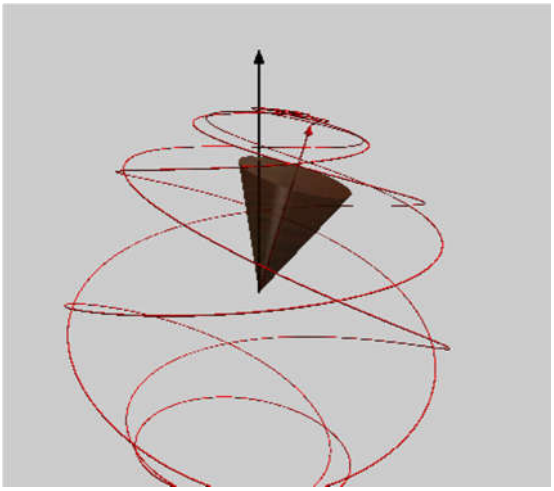


Slika 5.3. Kutne brzine Kovalevskaya zvrka


Slika 5.4. Putanja simetričnog zvrka u polju sile teže



Slika 5.5. Putanja asimetričnog zvrka u polju sile teže



Slika 5.6. Putanja Kovalevskaya zvrka

pozitivna. Sve komponente su gotovo periodične, i to sa sličnom periodom. Putanja je gotovo jednaka u sve 3 dimenzije.

5.3. Tijek programa

U prvom dijelu programa, kao i prije, definiramo početne uvijete, kojih ima više u odnosu na slobodne zvrkove, kao što smo diskutirali u poglavlju 5.1. Što se tiče tenzora tromosti, za simetričan zvrk vrijedi $I_1 = I_2 < I_3$, asimetričan $I_2 < I_1 < I_3$, a za Kovalevskaya $2I_1 = I_2 = I_3$. Komponente kutne brzine više nisu početni uvjet, nego ih računamo iz početnih kutova i njihovih derivacija:

$$\Omega_{10} = \dot{\theta}_0 \cos\psi_0 + \dot{\varphi}_0 \sin\theta_0 \sin\psi_0 \,, \tag{5.3}$$

$$\Omega_{20} = -\dot{\theta}_0 \sin\psi_0 + \dot{\varphi}_0 \sin\theta_0 \cos\psi_0 \,, \tag{5.4}$$

$$\Omega_{30} = \dot{\psi}_0 + \dot{\varphi}_0 \cos\theta_0. \tag{5.5}$$

Tako dobivene komponente kutne brzine su bezdimenzionalne, što nam olakšava jednadžbe kasnije u programu. Komponente γ računamo kao i prije. Jednadžbe za konstante ostaju iste, uz dodatne konstante opisane u poglavlju 5.2.

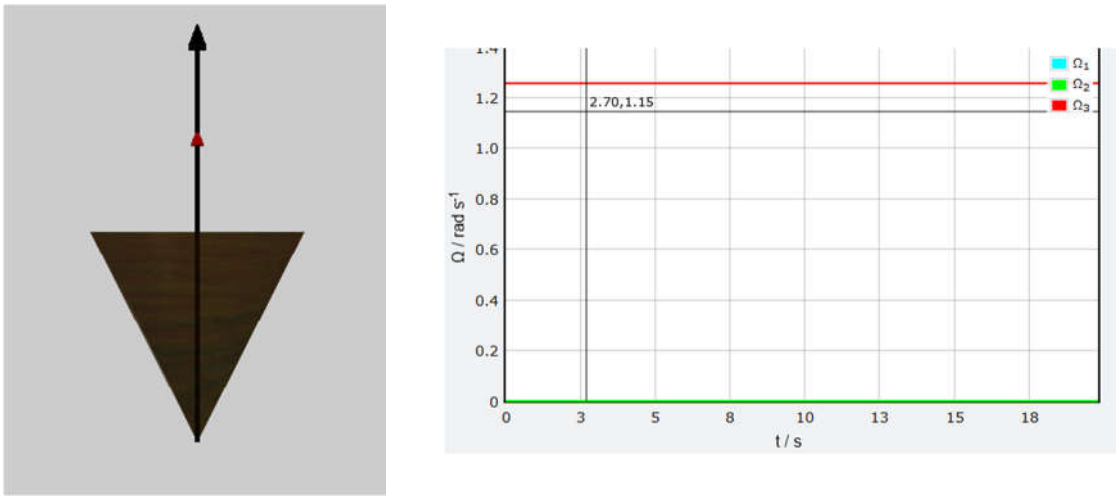
Drugi dio programa sastoji se od pripremanja i rješavanja diferencijalnih jednadžbi. Sada imamo sustav od 6 jednadžbi, jer nam kutne brzine ovise i o vektoru γ, kako je opisano u poglavlju 2.5. Nakon rješavanja diferencijalnih jednadžbi, integracijom unatrag provjeravamo točnost rješenja.

Treći dio ostaje jednak kao i prije, do na promjenu naslova, te boje i oblika zvrka. Definiramo prostor na kojemu će se odvijati simulacija i stvaramo zvrk.

U konačnom, četvrtom dijelu, simuliramo gibanje zvrka, i ispunjavamo grafove analogno programima za slobodne zvrkove. Koje konstante pratimo ovisi o vrsti zvrka.

5.4. Opis gibanja simetričnog zvrka u polju sile teže – precesija i nutacija

Gibanje zvrka sastoji se od tri različita gibanja (pod uvjetom da smo mu fiksirali vrh): vrtnje oko vlastite osi, precesije i nutacije. *Vrtnja oko vlastite osi* je kružno gibanje koje očekujemo od zvrka: rotacije oko glavne z-osi, koja je ujedno i os simetrije. To gibanje povezujemo sa trećim Eulerovim kutom, ψ, točnije s njegovom promjenom $\dot{\psi}$. Što je veća promjena $\dot{\psi}$, brže se zvrk okreće oko sebe, odnosno vrijeme da se zvrk okrene oko sebe je manje. Prvi Eulerov kut φ, odnosno njegovu promjenu $\dot{\varphi}$ povezujemo sa sljedećim gibanjem, koje nazivamo precesija. *Precesija* je rotacija glavne z'-osi zvrka oko osi-z fiksnog sustava. Drugim riječima, osim što se zvrk vrti oko sebe, on se vrti i oko z-osi fiksnog sustava. Veća precesija znači da je zvrku potrebno manje vremena da napravi krug oko z-osi fiksnog sustava. Konačno, drugi Eulerov kut θ, odnosno njegovu promjenu $\dot{\theta}$ povezujemo sa trećim gibanjem, *nutacijom*. Glavna z'-os zvrka se za vrijeme vrtnje može približiti ili udaljiti osi z fiksnog sustava. Zvrk sa velikom nutacijom jako će se „njihati“. No, moramo uzeti u obzir da kutne brzine Ω_1, Ω_2 i Ω_3 nisu jednake promjenama Eulerovih kutova $\dot{\varphi}$, $\dot{\theta}$ i $\dot{\psi}$, nego imaju nekoliko komponenti. Uzmimo primjer 1 u kojem su naši početni uvjeti sljedeći: $\varphi = \frac{90\pi}{180}$, $\theta = 0$, $\psi = 0$, $\dot{\varphi} = \frac{72\pi}{180}$, $\dot{\theta} = \frac{0\pi}{180}$ i $\dot{\psi} = \frac{0\pi}{180}$. Rezultat možemo vidjeti na slici 5.7.



Slika 5.7. Rezultat primjera 1

Iako bi očekivali precesiju, pošto je $\dot{\varphi} > 0$, imamo samo rotaciju, zato što je precesija s kutom $\theta = 0$ zapravo vrtnja oko samog zvrka. Kutne brzine dobivamo iz sljedećih relacija:

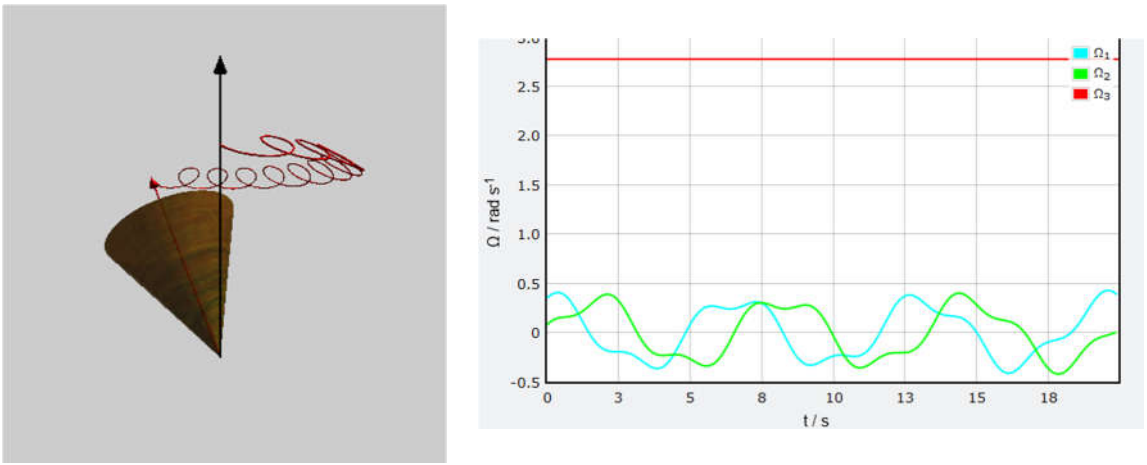
$$\Omega_1 = \dot{\theta}\cos\psi + \dot{\varphi}\sin\theta\sin\psi, \tag{5.6}$$

$$\Omega_2 = -\dot{\theta}\sin\psi + \dot{\varphi}\sin\theta\cos\psi, \tag{5.7}$$

$$\Omega_3 = \dot{\psi} + \dot{\varphi}\cos\theta. \tag{5.8}$$

Sada ćemo na sljedećem primjeru 2 objasniti pojedina gibanja. Početni uvjeti su sljedeći: $I_1 = I_2 = 2.0, I_3 = 3.1, \varphi = \frac{90\pi}{180}, \theta = \frac{33\pi}{180}, \psi = \frac{16\pi}{180}, \dot{\varphi} = \frac{18\pi}{180}, \dot{\theta} = \frac{18\pi}{180}$ i $\dot{\psi} = \frac{144\pi}{180}$.

Rezultat možemo vidjeti na slici 5.8. Rotaciju direktno očitavamo iz grafa, kao Ω_3. Ako nam je ukupno vrijeme

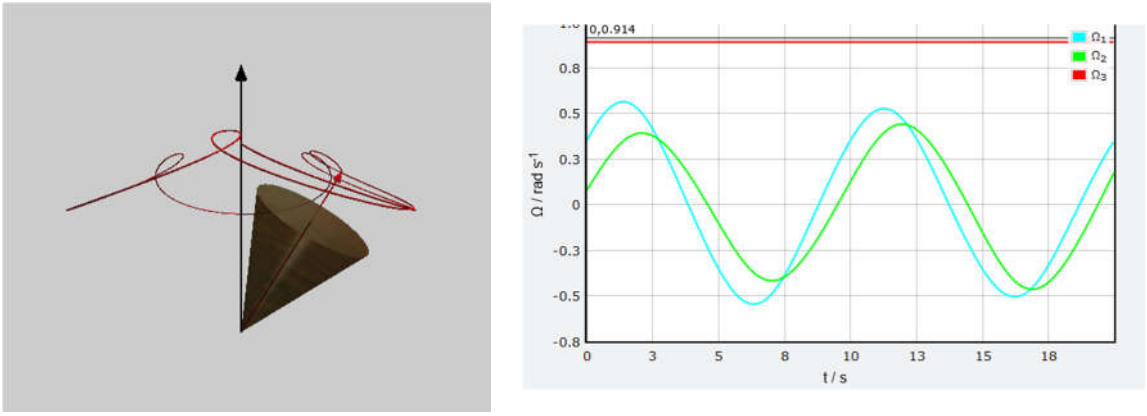


Slika 5.8. Rezultat primjera 2

konstantno, precesiju možemo očitati sa slike, koliko je krugova je napravila os simetrije zvrka oko momenta količine gibanja. Na ovom primjeru, možemo vidjeti da je zvrk

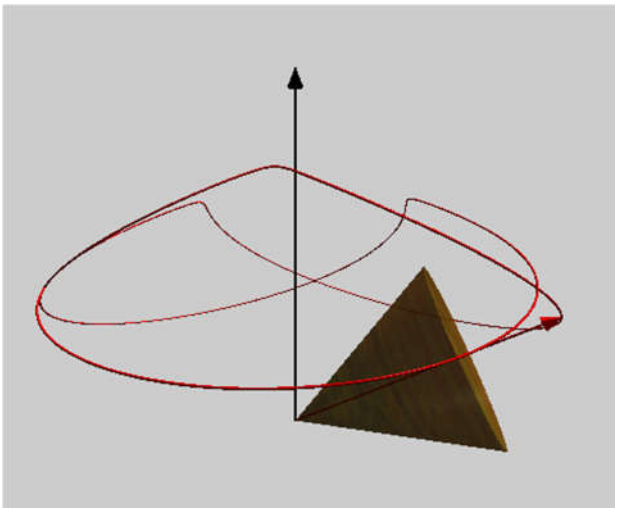
napravio nešto više od pola kruga. Konačno, nutaciju možemo također očitati sa slike: koliki je visinski razmak između brjegova i dolova traga središta zvrka, odnosno koliko su putanje „razmazane“.

Sljedeći primjer 3 ima početne uvjete: $I_1 = I_2 = 1.5$, $I_3 = 3.1$, $\varphi = \frac{90\pi}{180}$, $\theta = \frac{33\pi}{180}$, $\psi = \frac{16\pi}{180}$, $\dot{\varphi} = \frac{18\pi}{180}$, $\dot{\theta} = \frac{18\pi}{180}$ i $\dot{\psi} = \frac{36\pi}{180}$. Na slici 5.9 je prikazano gibanje. Krugovi su veći, što ukazuje na veću nutaciju, a os simetrije je napravila cijeli krug (zvrk kreće iz pozicije prema nama) što ukazuje na veću precesiju. Primijetimo, da kako smo smanjili $\dot{\psi}$, rotacija se je smanjila.



Slika 5.9. Rezultat primjera 3

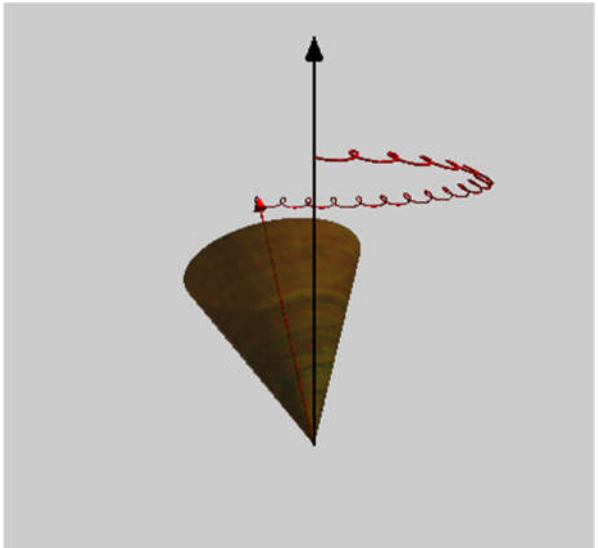
S obzirom na to kakav je odnos precesije, nutacije i kuta θ, razlikujemo tri vrste gibanja: monotona precesija, nemonotona precesija i granična precesija [5]. Iako svi u nazivu imaju samo precesija, ovise o raznim veličinama. Kako nam rotacija nije važna, sada ćemo se fokusirati samo na druga dva gibanja, odnosno na putanju osi simetrije.



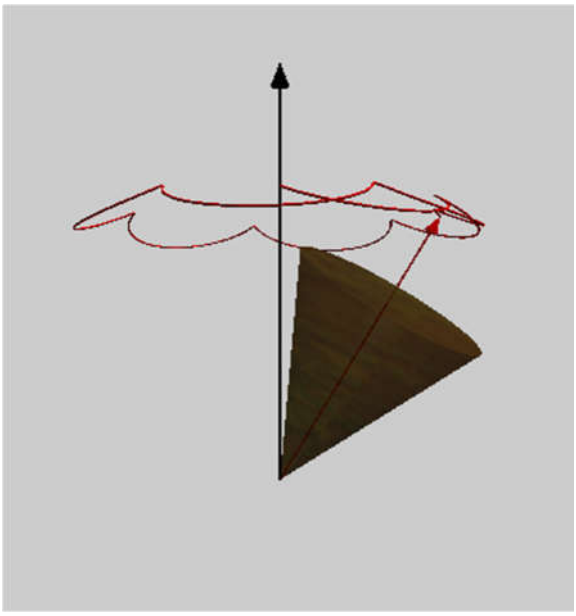
Slika 5.10. Rezultat primjera 4, monotona precesija

Primjer 4 ima za početne uvijete $I_1 = I_2 = 1.0, I_3 = 2.1,\ \varphi = \frac{90\pi}{180}, \theta = \frac{33\pi}{180}, \psi = \frac{16\pi}{180}, \dot{\varphi} = \frac{18\pi}{180}, \dot{\theta} = \frac{9\pi}{180}$ i $\dot{\psi} = \frac{32\pi}{180}$, i predstavlja monotonu precesiju, koju možemo vidjeti na slici 5.10. Zvrk se cijelo vrijeme rotira u istom smjeru oko osi z. U ovom slučaju, dobili smo u potpunosti periodičko gibanje. Nakon nekog vremena t (koje je u ovom slučaju jednako vremenu jedne precesije), zvrk se vraća u početni položaj sa istim uvjetima, tako da će svaki krug imati identičnu putanju.

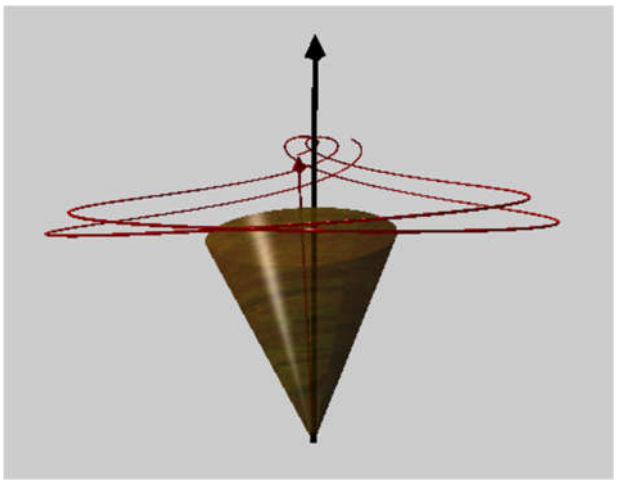
Primjer 5 ima za početne uvijete $I_1 = I_2 = 1.5, I_3 = 3.1,\ \varphi = \frac{90\pi}{180}, \theta = \frac{33\pi}{180}, \psi = \frac{16\pi}{180}, \dot{\varphi} = \frac{18\pi}{180}, \dot{\theta} = \frac{18\pi}{180}$ i $\dot{\psi} = \frac{144\pi}{180}$, a rezultat je nemonotona precesija koju možemo vidjeti na slici 5.11. Smjer rotacije oko z-osi se mijenja u vremenu. To možemo vidjeti po „petljama“ koje stvara putanja osi simetrije. Granični slučaj je onaj kada rotacija oko z osi privremeno prestaje, odnosno pada na nulu, ali je uvijek u istom smjeru. Možemo ga vidjeti na slici 5.12 primjera 6, s početnim uvjetima $I_1 = I_2 = 1.0, I_3 = 2.1, \varphi = \frac{90\pi}{180}, \theta = \frac{33\pi}{180}, \psi = \frac{16\pi}{180}, \dot{\varphi} = \frac{18\pi}{180}, \dot{\theta} = \frac{9\pi}{180}$ i $\dot{\psi} = \frac{62\pi}{180}$. Ovdje možemo vidjeti „špice“ na putanji osi simetrije zvrka. Konačno, vrsta precesije ovisi i o početnom kutu θ. Primjer 7 ima početene uvjete $I_1 = I_2 = 1.0, I_3 = 2.1, \varphi = \frac{90\pi}{180}, \theta = \frac{8\pi}{180}, \psi = \frac{16\pi}{180}, \dot{\varphi} = \frac{18\pi}{180}, \dot{\theta} = \frac{9\pi}{180}$ i $\dot{\psi} = \frac{32\pi}{180}$, što je slično primjeru 4 s monotonom precesijom, ali sada zbog manjeg kuta imamo nemonotonu precesiju, kao što možemo vidjeti na slici 5.13. U pravilu, za manji kut θ gibanje ide prema nemonotonoj precesiji.



Slika 5.11. Rezultat primjera 5, nemonotona precesija



Slika 5.12. Rezultat primjera 6, granična precesija

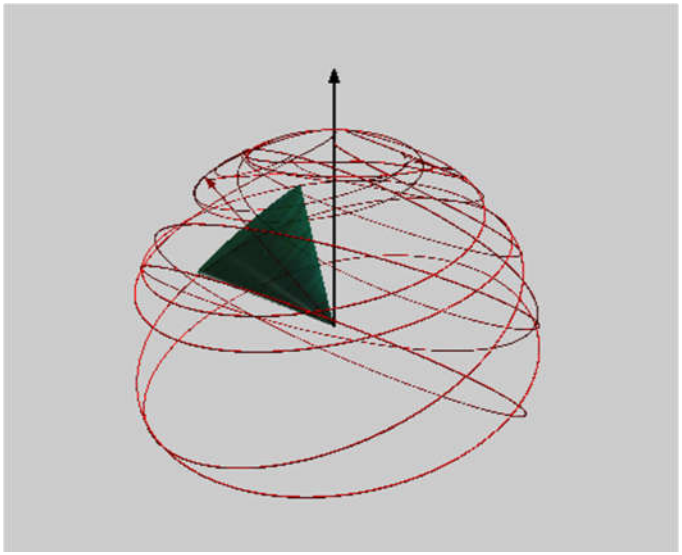


Slika 5.13. Rezultat primjera 7, granična precesija

5.5. Primjer kaotičnog sustava – asimetrični zvrk u polju sile teže

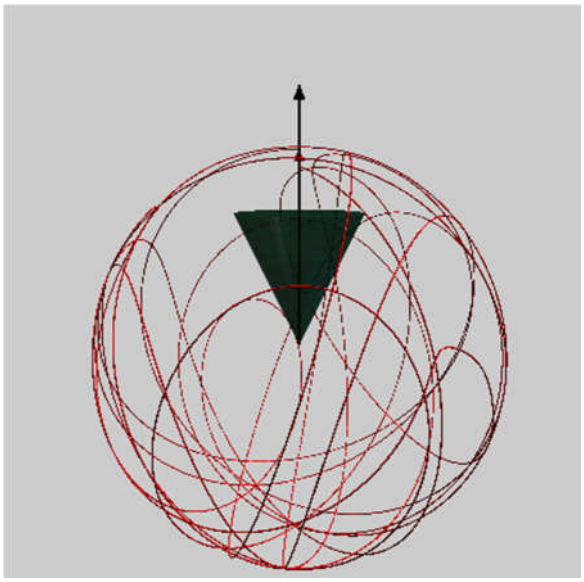
Simetričan zvrk u polju sile teže predstavlja integrabilni sustav jer ima dovoljan broj konstanti gibanja, a isto vrijedi i za Kovaleskayin zvrk. S druge strane, asimetrični zvrk u polju sile teže ima jednu konstantu manje te se stoga radi o neintegrabilnom sustavu. Jedna od glavnih karakteristika neintegrabilnih tj kaotičnih sustava je njihova osjetljivost na početne uvjete [4]. Kod integrabilnog sustava će dva bliska početna uvjeta rezultirati bliskim putanjama i u kasnijim vremenskim trenutcima, dok se kod neintegrabilnog sustava putanje čak i za bliske početne uvjete mogu u kasnijim trenutcima posve razlikovati. Uzimajući u obzir konačnu numeričku preciznost rješavanja jednadžbi gibanja zvrka, jasno je da će neintegrabilni sustav u praksi biti nemoguće pratiti na imalo duljim vremenskim skalama, iako se radi o deterministčkom sustavu. Takvo, nepredvidljivo

gibanje dinamičnog sustava nazivamo *deterministčkim kaosom,* ili samo *kaosom*. Simulirati ćemo dva asimetrična zvrka, i promatrati njihove putanje kroz duži vremenski period (90 sekundi). U prvom slučaju, početni uvjeti su sljedeći [4]: $I_1 = 2.55$, $I_2 = 2.0$, $I_3 = 3.0$, $\varphi = \frac{90\pi}{180}$, $\theta = \frac{28\pi}{180}$, $\psi = \frac{32\pi}{180}$, $\dot{\varphi} = \frac{18\pi}{180}$, $\dot{\theta} = \frac{18\pi}{180}$ i $\dot{\psi} = \frac{44\pi}{180}$. Putanju možemo vidjeti na



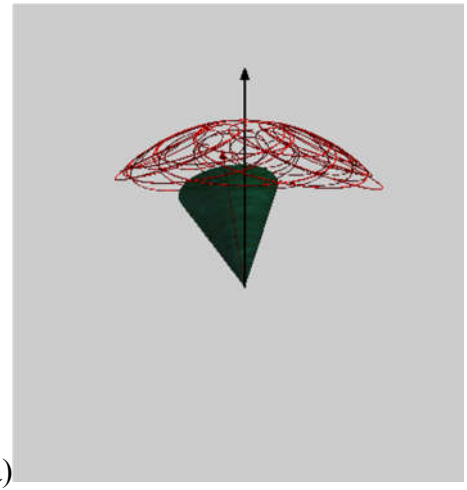
Slika 5.12. Putanja središta zvrka s malim kaosom

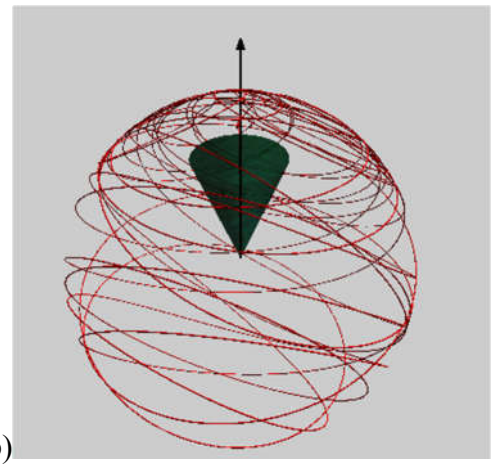
slici 5.12. Iako ne možemo predvidjeti gibanje zvrka, postoje položaji u kojima bi zvrk mogao postojati, ali do njih ne dolazi (donji dio kugle). U drugom slučaju, početni uvjeti bili su: $I_1 = 2.5$, $I_2 = 1.5$, $I_3 = 2.0$, $\varphi = \frac{90\pi}{180}$, $\theta = \frac{32\pi}{180}$, $\psi = \frac{32\pi}{180}$, $\dot{\varphi} = \frac{42\pi}{180}$, $\dot{\theta} = \frac{18\pi}{180}$ i $\dot{\psi} = \frac{-32\pi}{180}$. Na slici 5.13. možemo vidjeti gibanje takovog zvrka. Primijetimo da sada zvrk prolazi svim dijelovima kugle. U ovom slučaju kaos je velik. Kada bi ga pustili dovoljno dugo, najvjerojatnije bi dobili cijelu površinu sfere. Iako ova dva primjera imaju znatno različite početne uvijete, to ne znači da promjena mora biti velika da bi primijetili kaos. Naime, čak i ako napravimo najmanju



Slika 5.13. Putanja središta zvrka s velikim kaosom

promjenu, rezultati mogu biti potpuno drugačiji. Na primjer, u sljedeća dva primjera ima iste početne uvijete: $I_1 = 1.5$, $I_2 = 1.0$, $\varphi = \frac{90\pi}{180}$, $\theta = \frac{33\pi}{180}$, $\psi = \frac{16\pi}{180}$, $\dot{\varphi} = \frac{36\pi}{180}$, $\dot{\theta} = \frac{36\pi}{180}$ i $\dot{\psi} = \frac{72\pi}{180}$. Jedina razlika je što je u prvom primjeru $I_3 = 1.8$, a u drugom je $I_3 = 1.7$.

a) 

b) 

Slika 5.14. Trag središta zvrka s a) jako malim i b) velikim kaosom

Dva slučaja možemo vidjeti na slici 5.14. Zbog kaosa, može doći do slučaja da na zvrk markerom nacrtamo krug oko glavne z-osi i u istim početnim uvjetima dobijemo sasvim drugačije gibanje.

6. Zaključak

Fizikalni pokusi jedna su od glavnih metoda poučavanja fizike u sklopu istraživačke nastave, ali nije ih uvijek moguće izvesti. U takvim slučajevima, pokuse možemo zamijeniti

računalnim simulacijama. U ovom radu bavili smo se simulacijama zvrka u programskom jeziku Python 3 pomoću paketa VPython. Zvrk smo odabrali jer se s jedne strane radi o povijesno važnom fizikalnom problemu, a s druge strane se radi o igrački koja je poznata svim učenicima. Izbor programskog jezika Python motiviran je zbog njegove dostupnosti, raširenosti i jednostavnosti sintakse. Pri odabiru konkretnih simulacija išli smo od jednostavnijih (slobodni zvrkovi) prema složenijim sustavima (zvrkovi u polju sile teže). U svakoj simulaciji smo morali riješiti sustav vezanih običnih diferencijalnih jednadžbi pri čemu smo koristili rutinu odeint iz paketa scipy. Osim same simulacije gibanja zvrka, na grafovima smo prikazali vrijednosti komponenata kutne brzine, kao i konstante gibanja. Za slučaj simetričnog zvrka u polju sile teže ilustrirali smo precesiju i nutaciju zvrka, kod Kovaleskayinog zvrka smo ilustrirali postojanje dodatne netrivijalne konstante gibanja, dok smo kod asimetričnog zvrka u polju sile teže ilustrirali pojam neintegrabilnog sustava i s time povezanog determinističkog kaosa.

Iako se naše simulacije podudaraju s analitičkim rješenjima (kada ona postoje) i vjerodostojno opisuju gibanje zvrka, one ipak ne mogu u potpunosti nadomjestiti pokus niti su same po sebi dovoljne da se učenicima prenese znanje. Simulacije mogu pomoći kada pokusi nisu mogući, no treba ih pažljivo oblikovati da bi postigle željeni efekt kod učenika.

Dodaci

A. Usporedba rješenja za slobodni simetrični zvrk

```
### Za diplomski rad
### Usporedba rješenja za simetrični slobodni zvrk
### Mentor: Tamara Nikšić
### Student: Jasnik Ivanjek
### Prirodoslovno matematički fakultet u Zagrebu
### Odsjek: Fizika
### Smjer: Fizika i Informatika, nastavnički

import numpy as np
import scipy.integrate as si
import vpython as v


### Početni uvjeti
I1 = 1.5     # I1 = I2 = 1.5, I3 = 3.6
I2 = I1      # Simetrični zvrk
I3 = 2.1     # I3 > I1 = I2

O30 = 3.1    # O = Omega
O10 = 0.5    # O30 = 3.1, O10 = 0.5, O20 = 0.0
O20 = 0.0    # Odabrali smo sustav u kojem ovo vrijedi

t0 = 33*np.pi/180     # Theta_0 = 13*pi/180
p0 = 16*np.pi/180     # Psi_0 = 16*pi/180

### Računamo iz početnih uvjeta
w = O30*(I3-I1)/I1       # w = omega

### Računamo početnu energiju (konstanta)
L = 1./2. * (I1*O10**2 + I2*O20**2 + I3*O30**2)
print ('L =', L)

### Računamo projekciju početne kutne količine gibanja na
### z-os (konstanta)
M30 = I3*O30
print ('Mz =', M30)

### Računamo projekciju početne kutne količine gibanja na z-os
### sustava zvrka (konstanta zbog simetrije)
M30 = I3*O30
print ('M3 =', M30)
### Određujemo vrijeme izvođenja i broj koraka
tmax = 10.           #20
steps = 2000.       #40000
dt = tmax/steps
t = np.arange(0.,tmax,dt)

### Analitička rješenja
o1 = np.zeros(int(steps))
o2 = np.zeros(int(steps))
o3 = np.zeros(int(steps))

for i in range(int(steps)):
    o1[i] = O10 * np.cos(w*t[i])
```

```
for i in range(int(steps)):
    o2[i] = O10 * np.sin(w*t[i])

for i in range(int(steps)):
    o3[i] = O30

### Rješavamo diferencijalnu jednadžbu
def f(y,x):
    O1 = y[0]
    O2 = y[1]

    dtO1 = -w*O2
    dtO2 = w*O1

    return [dtO1,dtO2]

y0 = [O10, O20]
y=si.odeint(f,y0,t)

O1 = y[:,0]
O2 = y[:,1]

### Provjera rješenja integracijom unatrag
tback=np.arange(tmax-dt,-dt,-dt)
y0=[O1[steps-1],O2[steps-1]]
yback=si.odeint(f,y0,tback)

print ('O1 = ',O10, yback[steps-1,0]) # Uspoređujemo početne
print ('O2 = ',O20, yback[steps-1,1]) # vrijednosti s
                                     # integriranima

### Definiramo prostor na kojem se prikazuje simulacija
prostor = v.canvas(title='Usporedba rješenja slobodnog simetričnog \
                  zvrka', width = 1000, height = 500,
                  background = v.vector(0.8,0.8,0.8))

### Stvaramo grafove
graph = v.graph(xtitle='t / s', ytitle='\u03A9 / rad s<sup>-1</sup>',
                align="left")
graph1 = v.gcurve(color=v.color.cyan, label = "\u03A9<sub>1</sub>")
graph2 = v.gcurve(color=v.color.green, label = "\u03A9<sub>2</sub>")
graph3 = v.gcurve(color=v.color.red, label = "\u03A9<sub>3</sub>")
graph1a = v.gdots(color=v.color.orange, label = "\u03A9<sub>1a</sub>",
                  interval=50)
graph2a = v.gdots(color=v.color.red, label = "\u03A9<sub>2a</sub>",
                  interval=50)
graph3a = v.gdots(color=v.color.blue, label = "\u03A9<sub>3a</sub>",
                  interval=50)
konst = v.graph(xtitle='t / s', align = "right", legend = True)
konst1 = v.gcurve(color=v.color.blue, label = "Energija")
konst2 = v.gcurve(color=v.color.red, label = "Kutna količina gibanja")
konst3 = v.gcurve(color=v.color.green,
                  label = "Projekcija kutne količine gibanja na z'-os")
konst1a = v.gdots(color=v.color.red, label = "Energija", interval=50)
konst2a = v.gdots(color=v.color.green, label = "Kutna količina gibanja",
                  interval=50)
konst3a = v.gdots(color=v.color.purple,
                  label = "Projekcija kutne količine gibanja na z'-os",
                  interval=50)
```

```
### Simuliramo
for i in range (int(steps)):
    v.rate(int(steps/tmax))

    graph1a.plot(t[i],o1[i])  # omega 1 - svjetlo plavo
    graph2a.plot(t[i],o2[i])  # omega 2 - zeleno
    graph3a.plot(t[i],o3[i])  # omega 3 - crveno
    graph1.plot(t[i],O1[i])  # omega 1 - svjetlo plavo
    graph2.plot(t[i],O2[i])  # omega 2 - zeleno
    graph3.plot(t[i],O30)  # omega 3 - crveno
    konst1a.plot(t[i], 1./2. * (I1*o1[i]**2 + I2*o2[i]**2 + I3*o3[i]**2))
                          # Numerički izračunata energija
    konst2a.plot(t[i],
                 ((I1*o1[i])**2 + (I2*o2[i])**2 + (I3*o3[i])**2)**(1/2))
                # Numerički izračunata kutna količina gibanja
    konst3a.plot(t[i], (I3*o3[i]))
                # Numerički izračunata projekcija kutne količine gibanja
    konst1.plot(t[i], 1./2. * (I1*O1[i]**2 + I2*O2[i]**2 + I3*O30**2))
                # Numerički izračunata energija
    konst2.plot(t[i], ((I1*O1[i])**2 + (I2*O2[i])**2 + \
                   (I3*O30)**2)**(1/2))
                # Numerički izračunata kutna količina gibanja
    konst3.plot(t[i], (I3*O30))
                # Numerički izračunata projekcija kutne količine gibanja
```

B. Usporedba rješenja za slobodni asimetrični zvrk

```
### Za diplomski rad
### Usporedba rješenja za asimetrični slobodni zvrk
### Mentor: Tamara Nikšić
### Student: Jasnik Ivanjek
### Prirodoslovno matematički fakultet u Zagrebu
### Odsjek: Fizika
### Smjer: Fizika i Informatika, nastavnički

import numpy as np
import scipy.integrate as si
import scipy.special as ss
import vpython as v


### Početni uvjeti
I1 = 1.0    # I1 = 1.0, I2 = 1.5, I3 = 3.6
I2 = 1.5     # Asimetrični zvrk
I3 = 3.6    # I3 > I2 > I1

O30 = 3.1   # O = Omega
O10 = 0.5   # O30 = 3.1, O10 = 0.5, O20 = 0.0
O20 = 0.0   # Odabrali smo sustav u kojem ovo vrijedi

t0 = 33*np.pi/180     # Theta_0 = 13*pi/180
p0 = 16*np.pi/180     # Psi_0 = 16*pi/180

### Računamo
O0 = (O10**2 + O20**2 + O30**2)**(1/2)

### Računamo početnu energiju (konstanta)
L = 1./2. * (I1*O10**2 + I2*O20**2 + I3*O30**2)
```

```
print ('L =', L)

### Računamo projekciju početne kutne količine gibanja na z-os
### (konstanta)
M30 = ((I1*O10)**2 + (I2*O20)**2 + (I3*O30)**2)**(1/2)
print ('M =', M30)

### Određujemo vrijeme izvođenja i broj koraka
tmax = 10.          #20
steps = 2000.      #40000
dt = tmax/steps
t = np.arange(0.,tmax,dt)

th = (((I3-I2)*(M30**2-2*L*I1))/(I3*I2*I1))**(1/2) * t
k2 = ((I2-I1)/(I3-I2))*((2*L*I3-M30**2)/(M30**2-2*L*I1))

### Analitička rješenja
o1 = np.zeros(int(steps))
o2 = np.zeros(int(steps))
o3 = np.zeros(int(steps))

for i in range(int(steps)):
    o1[i] = ((2*L*I3-M30**2)/(I1*(I3-I1)))**(1/2)* ss.ellipj(th[i],k2)[1]

for i in range(int(steps)):
    o2[i] = ((2*L*I3-M30**2)/(I2*(I3-I2)))**(1/2)* ss.ellipj(th[i],k2)[0]

for i in range(int(steps)):
    o3[i] = ((M30**2-2*L*I1)/(I3*(I3-I1)))**(1/2)* ss.ellipj(th[i],k2)[2]

### Rješavamo diferencijalnu jednadžbu
def f(y,x):
    O1 = y[0]
    O2 = y[1]
    O3 = y[2]

    dtO1 = (I2-I3)*O2*O3/I1
    dtO2 = (I3-I1)*O3*O1/I2
    dtO3 = (I1-I2)*O1*O2/I3

    return [dtO1,dtO2,dtO3]

y0 = [O10, O20, O30]

y=si.odeint(f,y0,t)

O1 = y[:,0]
O2 = y[:,1]
O3 = y[:,2]

### Provjera rješenja integracijom unatrag
tback=np.arange(tmax-dt,-dt,-dt)
y0=[O1[steps-1],O2[steps-1],O3[steps-1]]
yback=si.odeint(f,y0,tback)

print ('O1 = ',O10, yback[steps-1,0]) # Uspoređujemo početne vrijednosti
print ('O2 = ',O20, yback[steps-1,1]) # s integriranima
print ('O3 = ',O30, yback[steps-1,2])

### Definiramo prostor na kojem se prikazuje simulacija
prostor = v.canvas(title='Usporedba rješenja slobodnog simetričnog \
```

```
                    zvrka', width = 1000, height = 500,
                    background = v.vector(0.8,0.8,0.8))


### Stvaramo grafove
graph = v.graph(xtitle='t / s', ytitle='\u03A9 / rad s<sup>-1</sup>',
                align="left")
graph1 = v.gcurve(color=v.color.cyan, label = "\u03A9<sub>1</sub>")
graph2 = v.gcurve(color=v.color.green, label = "\u03A9<sub>2</sub>")
graph3 = v.gcurve(color=v.color.red, label = "\u03A9<sub>3</sub>")
graph1a = v.gdots(color=v.color.orange, label = "\u03A9<sub>1a</sub>",
                  interval=50)
graph2a = v.gdots(color=v.color.red, label = "\u03A9<sub>2a</sub>",
                  interval=50)
graph3a = v.gdots(color=v.color.blue, label = "\u03A9<sub>3a</sub>",
                  interval=50)
konst = v.graph(xtitle='t / s', align = "right", legend = True)
konst1 = v.gcurve(color=v.color.blue, label = "Energija")
konst2 = v.gcurve(color=v.color.red, label = "Kutna količina gibanja")
konst1a = v.gdots(color=v.color.red, label = "Energija", interval=50)
konst2a = v.gdots(color=v.color.green, label = "Kutna količina gibanja",
                  interval=50)

### Simuliramo
for i in range (int(steps)):
    v.rate(int(steps/tmax))

    graph1a.plot(t[i],o1[i])  # omega 1 - svjetlo plavo
    graph2a.plot(t[i],o2[i])  # omega 2 - zeleno
    graph3a.plot(t[i],o3[i])  # omega 3 - crveno
    graph1.plot(t[i],O1[i])   # omega 1 - svjetlo plavo
    graph2.plot(t[i],O2[i])   # omega 2 - zeleno
    graph3.plot(t[i],O3[i])   # omega 3 - crveno
    konst1a.plot(t[i], 1./2. * (I1*o1[i]**2 + I2*o2[i]**2 + I3*o3[i]**2))
                              # Numerički izračunata energija
    konst2a.plot(t[i], ((I1*o1[i])**2 + (I2*o2[i])**2 + \
                        (I3*o3[i])**2)**(1/2))
                              # Numerički izračunata kutna količina gibanja
    konst1.plot(t[i], 1./2. * (I1*O1[i]**2 + I2*O2[i]**2 + I3*O3[i]**2))
                              # Numerički izračunata energija
    konst2.plot(t[i], ((I1*O1[i])**2 + (I2*O2[i])**2 + \
                       (I3*O3[i])**2)**(1/2))
                              # Numerički izračunata kutna količina gibanja
```

C. *Simulacija simetričnog slobodnog zvrka*

```
### Za diplomski rad
### Simulacija simetričnog slobodnog zvrka
### Mentor: Tamara Nikšić
### Student: Jasnik Ivanjek
### Prirodoslovno matematički fakultet u Zagrebu
### Odsjek: Fizika
### Smjer: Fizika i Informatika, nastavnički

import numpy as np
import scipy.integrate as si
import vpython as v
```

```
### Početni uvijeti
I1 = 1.5     # I1 = I2 = 1.5, I3 = 3.6
I2 = I1      # Simetrični zvrk
I3 = 3.6     # I3 > I1 = I2

O30 = 3.1    # O = Omega
O10 = 0.5    # O30 = 3.1, O10 = 0.5, O20 = 0.0
O20 = 0.0    # Odabrali smo sustav u kojem ovo vrijedi

t0 = 33*np.pi/180     # 33*pi/180
p0 = 16*np.pi/180     # 6*pi/180
psi_0 = 45*np.pi/180 # 45*pi/180

### Računamo iz početnih uvjeta
w = O30*(I3-I1)/I1       # w = omega

### Računamo početnu energiju (konstanta)
E = 1./2. * (I1*O10**2 + I2*O20**2 + I3*O30**2)
print ('E =', E)

### Računamo početni moment količine gibanja (konstanta)
M0 = np.sqrt((I1*O10)**2+(I2*O20)**2+(I3*O30)**2)
print ('Moment kolicine gibanja =', M0)


### Određujemo vrijeme izvođenja i broj koraka
tmax = 20.           #20
steps = 20000.       #40000
dt = tmax/steps
t = np.arange(0.,tmax,dt)


### Rješavamo diferencijalnu jednadžbu
def f(y,x):
    O1 = y[0]
    O2 = y[1]
    O3 = y[2]

    dtO1 = -w*O2
    dtO2 = w*O1
    dtO3 = 0.0

    return [dtO1,dtO2,dtO3]

y0 = [O10, O20,O30]
y=si.odeint(f,y0,t)

O1 = y[:,0]
O2 = y[:,1]
O3 = y[:,2]


### Provjera rješenja integracijom unatrag
tback=np.arange(tmax-dt,-dt,-dt)
y0=[O1[steps-1],O2[steps-1],O3[steps-1]]
yback=si.odeint(f,y0,tback)


print ('O1 = ',O10, yback[steps-1,0]) # Uspoređujemo početne vrijednosti
print ('O2 = ',O20, yback[steps-1,1]) # s integriranima
print ('O3 = ',O30, yback[steps-1,2])
```

```
### Definiramo prostor na kojem se prikazuje simulacija
prostor = v.canvas(title='Slobodni simetrični zvrk', width = 1000,
                   height = 500, forward = v.vector(-1,0,0),
                   up=v.vector(0,0,1), background = \
                   v.vector(0.8,0.8,0.8), range=2,autoscale=False)


### Jedinični vektori fiksnog sustava
x_prime = v.vector(1,0,0)
y_prime = v.vector(0,1,0)
z_prime = v.vector(0,0,1)


### Jedinični vektori pomičnog sustava
e1 = v.vector(np.cos(phi_0)*np.cos(psi_0)-np.sin(phi_0)*\
              np.cos(theta_0)*np.sin(psi_0), np.sin(phi_0)*\
              np.cos(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.sin(psi_0), np.sin(theta_0)*np.sin(psi_0))
e2 = v.vector(-(np.cos(phi_0)*np.sin(psi_0)+np.sin(phi_0)*\
              np.cos(theta_0)*np.cos(psi_0)), -np.sin(phi_0)*\
              np.sin(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.cos(psi_0),np.sin(theta_0)*np.cos(psi_0))
e3 = v.vector(np.sin(phi_0)*np.sin(theta_0),-np.cos(phi_0)\
              *np.sin(theta_0),np.cos(theta_0))

#os simetrije zvrka
os = -v.vector(np.sin(phi_0)*np.sin(theta_0),-np.cos(phi_0)\
               *np.sin(theta_0),np.cos(theta_0))

M1 = I1*O10*e1
M2 = I2*O20*e2
M3 = I3*O30*e3
M_vektor = M1+M2+M3
M_vektor = 1.5*M_vektor/M_vektor.mag


### Ubacujemo zvrk (stožac)
zvrk=v.cone(texture={'file':v.textures.wood_old,'bumpmaps': \
            v.bumpmaps.wood_old}, pos=v.vector(0,0,0),\
            axis=os, size=v.vector(1,1,1), color=v.color.green,
            opacity=0.9)
    # Teksturu stavljamo da lakše uočimo rotacije.
os_zvrka=v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
                 v.bumpmaps.stucco}, pos=os, length=1.5, color =
v.color.red,\
                 axis=e3, shaftwidth = 0.01, make_trail=True)
                    # Pokazujemo os zvrka

M = v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
            v.bumpmaps.stucco}, pos=os, length=1.5,\
            color = v.color.blue, axis=M_vektor,
            shaftwidth = 0.01,) #Moment količine gibanja

putanja=v.curve(radius=0.005,color=v.color.red)
putanja.append(pos=os+1.5*e3)


### Stvaramo grafove
```

```
graph = v.graph(xtitle='t / s', ytitle='\u03A9 / rad s<sup>-1</sup>',
                align="left")
graph1 = v.gcurve(color=v.color.cyan, label = "\u03A9<sub>1</sub>")
graph2 = v.gcurve(color=v.color.green, label = "\u03A9<sub>2</sub>")
graph3 = v.gcurve(color=v.color.red, label = "\u03A9<sub>3</sub>")
konst = v.graph(xtitle='t / s', align = "right", legend = True)
konst1 = v.gcurve(color=v.color.blue, label = "Energija")
konst2 = v.gcurve(color=v.color.red, label = "Kutna količina gibanja")
konst3 = v.gcurve(color=v.color.green, label = "Projekcija kutne \
                  količine gibanja na z'-os")

### Simuliramo
for i in range(int(steps)):
    v.rate(int(steps/tmax))     # Ograničimo količinu operacija u sekundi
                                # (usporimo simulaciju)
    Omega = O1[i]*e1 + O2[i]*e2 + O3[i]*e3 # Ovo je trenutni Omega
    os_rotacije = Omega/Omega.mag
    kut_rotacije = Omega.mag*dt
    M_vektor = I1*O1[i]*e1 + I2*O2[i]*e2 + I3*O3[i]*e3
    M_vektor = 1.5*M_vektor/M_vektor.mag
        # Prikladna duljina vektora M za sliku

    e1 = e1.rotate(angle=kut_rotacije,axis=os_rotacije)
    e2 = e2.rotate(angle=kut_rotacije,axis=os_rotacije)
    e3 = e3.rotate(angle=kut_rotacije,axis=os_rotacije)

    zvrk.rotate(angle=kut_rotacije, axis = os_rotacije, origin =os)
            # Rotiramo zvrk oko odgovarajuće osi
    os_zvrka.rotate(angle=kut_rotacije, axis = os_rotacije, origin =os)
    putanja.append(pos=os+1.5*e3)
    M.axis=M_vektor

    graph1.plot(t[i],O1[i]) # omega 1 - svjetlo plavo
    graph2.plot(t[i],O2[i]) # omega 2 - zeleno
    graph3.plot(t[i],O3[i]) # omega 3 - crveno
    konst1.plot(t[i], 1./2. * (I1*O1[i]**2 + I2*O2[i]**2 + I3*O3[i]**2))
            # Numerički izračunata energija
    konst2.plot(t[i], ((I1*O1[i])**2 + (I2*O2[i])**2 + \
                       (I3*O3[i])**2)**(1/2))
            # Numerički izračunata kutna količina gibanja
    konst3.plot(t[i], v.dot(M_vektor,e3)) # cos(theta)
```

D. *Simulacija asimetričnog slobodnog zvrka*

```
### Za diplomski rad
### Simulacija asimetričnog slobodnog zvrka
### Mentor: Tamara Nikšić
### Student: Jasnik Ivanjek
### Prirodoslovno matematički fakultet u Zagrebu
### Odsjek: Fizika
### Smjer: Fizika i Informatika, nastavnički

import numpy as np
import scipy.integrate as si
import vpython as v


### Početni uvijeti
I1 = 1.5
```

```
I2 = 2.5
I3 = 3.1

# Komponente kutne brzine Omega (mjeri se u rad/s)
O30 = 2.1    # O = Omega
O10 = 0.5    # O30 = 3.1, O10 = 0.5, O20 = 0.0
O20 = 0.0    # Odabrali smo sustav u kojem ovo vrijedi

theta_0 = 3*np.pi/180    # 33*pi/180
phi_0 = 16*np.pi/180     # 16*pi/180
psi_0 = 45*np.pi/180     # 45*pi/180

### Računamo iz početnih uvijeta
w = O30*(I3-I1)/I1       # w = omega

### Računamo početnu energiju (konstanta)
E = 1./2. * (I1*O10**2 + I2*O20**2 + I3*O30**2)
print ('Energija =', E)

### Računamo pocetni moment kolicine gibanja (konstanta)
M0 = np.sqrt((I1*O10)**2+(I2*O20)**2+(I3*O30)**2)
print ('Moment kolicine gibanja =', M0)

### Određujemo vrijeme izvođenja i broj koraka
tmax = 20.           #20
steps = 20000      #40000
dt = tmax/steps
t = np.arange(0.,tmax,dt)

### Rješavamo diferencijalnu jednadžbu
def f(y,x):
    O1 = y[0]
    O2 = y[1]
    O3 = y[2]

    dtO1 = -(I3-I2)*O2*O3/I1
    dtO2 = -(I1-I3)*O1*O3/I2
    dtO3 = -(I2-I1)*O2*O1/I3

    return [dtO1,dtO2,dtO3]

y0 = [O10,O20,O30]
y=si.odeint(f,y0,t)

O1 = y[:,0]
O2 = y[:,1]
O3 = y[:,2]

### Provjera rješenja integracijom unatrag
tback=np.arange(tmax-dt,-dt,-dt)
y0=[O1[steps-1],O2[steps-1],O3[steps-1]]
yback=si.odeint(f,y0,tback)

print ('O1 = ',O10, yback[steps-1,0]) # Uspoređujemo početne
print ('O2 = ',O20, yback[steps-1,1]) # vrijednosti s integriranima
print ('O3 = ',O30, yback[steps-1,2])

### Definiramo prostor na kojem se prikazuje simulacija
prostor = v.canvas(title='Slobodni asimetrični zvrk', width = 1000,
                   height = 500, forward = v.vector(-1,0,0),
                   up=v.vector(0,0,1), background =\
```

```
                    v.vector(0.8,0.8,0.8), range=2,autoscale=False)

### Jedinični vektori fiksnog sustava
x_prime = v.vector(1,0,0)
y_prime = v.vector(0,1,0)
z_prime = v.vector(0,0,1)

### Jedinični vektori pomičnog sustava
e1 = v.vector(np.cos(phi_0)*np.cos(psi_0)-np.sin(phi_0)*\
              np.cos(theta_0)*np.sin(psi_0), np.sin(phi_0)*\
              np.cos(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.sin(psi_0), np.sin(theta_0)*np.sin(psi_0))
e2 = v.vector(-(np.cos(phi_0)*np.sin(psi_0)+np.sin(phi_0)*\
              np.cos(theta_0)*np.cos(psi_0)), -np.sin(phi_0)*\
              np.sin(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.cos(psi_0),np.sin(theta_0)*np.cos(psi_0))
e3 = v.vector(np.sin(phi_0)*np.sin(theta_0),-np.cos(phi_0)\
              *np.sin(theta_0),np.cos(theta_0))

# Os simetrije zvrka
os = -v.vector(np.sin(phi_0)*np.sin(theta_0),-np.cos(phi_0)\
               *np.sin(theta_0),np.cos(theta_0))

M1 = I1*O10*e1
M2 = I2*O20*e2
M3 = I3*O30*e3
M_vektor = M1+M2+M3
M_vektor = 1.5*M_vektor/M_vektor.mag

### Ubacujemo zvrk (stožac)
zvrk=v.cone(texture={'file':v.textures.wood_old,'bumpmaps': \
            v.bumpmaps.wood_old}, pos=v.vector(0,0,0),\
            axis=os, size=v.vector(1,1,0.9), color=v.color.red,
            opacity=0.9)
    # Teksturu stavljamo da lakše uočimo rotacije.
os_zvrka=v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
                 v.bumpmaps.stucco}, pos=os, length=1.5, color =
v.color.red,\
                 axis=e3, shaftwidth = 0.01)
                    # Pokazujemo os zvrka

M = v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
            v.bumpmaps.stucco}, pos=os, length=1.5,\
            color = v.color.blue, axis=M_vektor,
            shaftwidth = 0.01,) #Moment količine gibanja

putanja=v.curve(radius=0.005,color=v.color.red)
putanja.append(pos=os+1.5*e3)

### Stvaramo grafove
graph = v.graph(xtitle='t / s', ytitle='\u03A9 / rad s<sup>-1</sup>',
                align="left")
graph1 = v.gcurve(color=v.color.cyan, label = "\u03A9<sub>1</sub>")
graph2 = v.gcurve(color=v.color.green, label = "\u03A9<sub>2</sub>")
graph3 = v.gcurve(color=v.color.red, label = "\u03A9<sub>3</sub>")
konst = v.graph(xtitle='t / s', align = "right", legend = True)
konst1 = v.gcurve(color=v.color.blue, label = "Energija")
konst2 = v.gcurve(color=v.color.red, label = "Kutna količina gibanja")


### Simuliramo
```

```
for i in range(int(steps)):
    v.rate(int(steps/tmax))     # Ograničimo količinu operacija u sekundi
                                # (usporimo simulaciju)
    Omega = O1[i]*e1 + O2[i]*e2 + O3[i]*e3 # Ovo je trenutni Omega
    os_rotacije = Omega/Omega.mag
    kut_rotacije = Omega.mag*dt
    M_vektor = I1*O1[i]*e1 + I2*O2[i]*e2 + I3*O3[i]*e3
    M_vektor = 1.5*M_vektor/M_vektor.mag
        # Prikladna duljina vektora M za sliku

    e1 = e1.rotate(angle=kut_rotacije,axis=os_rotacije)
    e2 = e2.rotate(angle=kut_rotacije,axis=os_rotacije)
    e3 = e3.rotate(angle=kut_rotacije,axis=os_rotacije)

    zvrk.rotate(angle=kut_rotacije, axis = os_rotacije, origin =os)
            # Rotiramo zvrk oko odgovarajuće osi
    os_zvrka.rotate(angle=kut_rotacije, axis = os_rotacije, origin =os)
    putanja.append(pos=os+1.5*e3)
    M.axis=M_vektor

    graph1.plot(t[i],O1[i]) # omega 1 - svjetlo plavo
    graph2.plot(t[i],O2[i]) # omega 2 - zeleno
    graph3.plot(t[i],O3[i]) # omega 3 - crveno
    konst1.plot(t[i], 1./2. * (I1*O1[i]**2 + I2*O2[i]**2 + I3*O3[i]**2))
            # Numerički izračunata energija
    konst2.plot(t[i], ((I1*O1[i])**2 + (I2*O2[i])**2 + \
                       (I3*O3[i])**2)**(1/2))
            # Numerički izračunata kutna količina gibanja
```

E. Simulacija simetričnog zvrka u gravitacijskom polju

```
### Za diplomski rad
### Simulacija simetričnog zvrka u gravitacijskom polju
### Mentor: Tamara Nikšić
### Student: Jasnik Ivanjek
### Prirodoslovno matematički fakultet u Zagrebu
### Odsjek: Fizika
### Smjer: Fizika i Informatika, nastavnički

import numpy as np
import scipy.integrate as si
import vpython as v


### Početni uvijeti
I1 = 1.0      # I1 = I2 = 1.5, I3 = 2.1
I2 = I1       # Simetrični zvrk
I3 = 2.1      # I3 > I1 = I2

t0 = 33.0*np.pi/180.        # theta_0  | 33
p0 = 16.0*np.pi/180.        # psi_0    | 16
f0 = 90.0*np.pi/180.        # phi_0    | 90
vt0 = 9.0*np.pi/180.        # vtheta_0 | 18
vp0 = 32.0*np.pi/180.       # vpsi_0   | 144
vf0 = 18.0*np.pi/180.       # vphi_0   | 18

theta_0 = t0
phi_0 = f0
```

```
psi_0 = p0

O10 = vt0*np.cos(p0)+vf0*np.sin(t0)*np.sin(p0)
O20 = -vt0*np.sin(p0)+vf0*np.sin(t0)*np.cos(p0)
O30 = vp0+vf0*np.cos(t0)

### Računamo iz početnih uvijeta
g10 = np.sin(t0)*np.sin(p0)      # koordinate
g20 = np.sin(t0)*np.cos(p0)      # g = gamma
g30 = np.cos(t0)

### Računamo početnu energiju (konstanta)
E = (I1*O10**2 + I2*O20**2 + I3*O30**2)/2+g30
print ('E =', E)

### Početni moment količine gibanja
Mz0 = g10*I1*O10 + g20*I2*O20 + g30*I3*O30
print ('M_z =', Mz0)

### Računamo projeciju početne kutne količine gibanja na
    #  z-os sustava zvrka (konstanta zbog simetrije)
M30 = I3*O30
print ('M_3 =', M30)

### Određujemo vrijeme izvođenja i broj koraka
tmax = 60.          #20
steps = 10000      #40000
dt = tmax/steps
t = np.arange(0.,tmax,dt)

### Rješavamo diferencijalnu jednadžbu
def f(y,x):
    O1 = y[0]
    O2 = y[1]
    O3 = y[2]

    g1 = y[3]
    g2 = y[4]
    g3 = y[5]

    dtO1 = (I2-I3)*O2*O3/I1 + g2/I1
    dtO2 = (I3-I1)*O3*O1/I2 - g1/I2
    dtO3 = (I1-I2)*O1*O2/I3

    dtg1 = O3*g2 - O2*g3
    dtg2 = O1*g3 - O3*g1
    dtg3 = O2*g1 - O1*g2

    return [dtO1,dtO2,dtO3,dtg1,dtg2,dtg3]

y0 = [O10, O20, O30, g10, g20, g30]

y=si.odeint(f,y0,t)

O1 = y[:,0]
O2 = y[:,1]
O3 = y[:,2]
g1 = y[:,3]
g2 = y[:,4]
g3 = y[:,5]
```

```
### Provjera rješenja integracijom unatrag
tback=np.arange(tmax-dt,-dt,-dt)
y0=[O1[steps-1],O2[steps-1],O3[steps-1], g1[steps-1],
    g2[steps-1], g3[steps-1]]
yback=si.odeint(f,y0,tback)

print ('O1 = ',O10, yback[steps-1,0]) # Uspoređujemo početne
print ('O2 = ',O20, yback[steps-1,1]) # vrijednosti s integriranima
print ('O3 = ',O30, yback[steps-1,2])


### Definiramo prostor na kojem se prikazuje simulacija
prostor = v.canvas(title='Simetrični zvrk u gravitacijskom polju', width
= 1000,
                   height = 500, forward = v.vector(-1,0,0),
                   up=v.vector(0,0,1), range=1.5,
                   background = v.vector(0.8,0.8,0.8),autoscale=False)

### Definiramo osi
#jedinicni vektori fiksnog sustava
x_prime = v.vector(1,0,0)
y_prime = v.vector(0,1,0)
z_prime = v.vector(0,0,1)


#jedinicni vektori pomicnog sustava
### Jedinični vektori pomičnog sustava
e1 = v.vector(np.cos(phi_0)*np.cos(psi_0)-np.sin(phi_0)*\
              np.cos(theta_0)*np.sin(psi_0), np.sin(phi_0)*\
              np.cos(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.sin(psi_0), np.sin(theta_0)*np.sin(psi_0))
e2 = v.vector(-(np.cos(phi_0)*np.sin(psi_0)+np.sin(phi_0)*\
              np.cos(theta_0)*np.cos(psi_0)), -np.sin(phi_0)*\
              np.sin(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.cos(psi_0),np.sin(theta_0)*np.cos(psi_0))
e3 = v.vector(np.sin(phi_0)*np.sin(theta_0),-np.cos(phi_0)\
              *np.sin(theta_0),np.cos(theta_0))

g = v.vector (-g10,-g20,-g30)

### Ubacujemo zvrk (stožac)
os = -e3
zvrk=v.cone(texture={'file':v.textures.wood_old,'bumpmaps': \
            v.bumpmaps.wood_old}, pos=v.vector(0,0,0),axis=os,\
            size=v.vector(1,1,1), color=v.color.yellow, opacity=0.7)
            # Teksturu stavljamo da lakše uočimo rotacije

os_zvrka=v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
                 v.bumpmaps.stucco}, pos=os,length=1.5, \
                 color = v.color.red,axis=e3, shaftwidth = 0.01)

os_z_vanjski = v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
                       v.bumpmaps.stucco}, pos=os,length=2.0, color = \
                       v.color.black, axis=z_prime, shaftwidth = 0.02)

putanja=v.curve(radius=0.005,color=v.color.red)
putanja.append(pos=os+1.5*e3)

### Stvaramo grafove
graph = v.graph(xtitle='t / s', ytitle='\u03A9 / rad s<sup>-1</sup>',
                align="left")
```

```
graph1 = v.gcurve(color=v.color.cyan, label = "\u03A9<sub>1</sub>")
graph2 = v.gcurve(color=v.color.green, label = "\u03A9<sub>2</sub>")
graph3 = v.gcurve(color=v.color.red, label = "\u03A9<sub>3</sub>")
konst = v.graph(xtitle='t / s', align = "right", legend = True)
konst1 = v.gcurve(color=v.color.blue, label = "Energija")
konst2 = v.gcurve(color=v.color.red, label = "Kutne količina gibanja")
konst3 = v.gcurve(color=v.color.green,
                  label = "Projekcija kutne količine gibanja na z-os")

### Simuliramo
for i in range(int(steps)):
    v.rate(int(steps/tmax))
        # Ograničimo količinu operacija u sekundi (usporimo simulaciju)

    Omega = O1[i]*e1+O2[i]*e2+O3[i]*e3 # Ovo je trenutni Omega
    os_rotacije = Omega/Omega.mag
    kut_rotacije = Omega.mag*dt

    e1 = e1.rotate(angle=kut_rotacije,axis=os_rotacije)
    e2 = e2.rotate(angle=kut_rotacije,axis=os_rotacije)
    e3 = e3.rotate(angle=kut_rotacije,axis=os_rotacije)

    g.rotate(angle=kut_rotacije,axis=os_rotacije)

    zvrk.rotate(angle=kut_rotacije, axis = os_rotacije, origin = os)
        # Rotiramo zvrk oko odgovarajuće osi
    os_zvrka.rotate(angle=kut_rotacije, axis = os_rotacije, origin =os)
    putanja.append(pos=os+1.5*e3)

    graph1.plot(t[i],O1[i]) # omega 1 - svjetlo plavo
    graph2.plot(t[i],O2[i]) # omega 2 - zeleno
    graph3.plot(t[i],O3[i]) # omega 3 - crveno
    konst1.plot(t[i], ((I1*O1[i]**2 + I2*O2[i]**2 + I3*O3[i]**2)/2\
                       + g3[i])) # Numerički izračunata energija
    konst2.plot(t[i], (g1[i]*I1*O1[i]+g2[i]*I2*O2[i]+g3[i]*I3*O3[i]))
            # Kutna količina gibanja
    konst3.plot(t[i], (I3*O3[i])) # Projekcija na os z sustava zvrka
```

F. Simulacija asimetričnog zvrka u gravitacijskom polju

```
### Za diplomski rad
### Simulacija asimetričnog zvrka u gravitacijskom polju
### Mentor: Tamara Nikšić
### Student: Jasnik Ivanjek
### Prirodoslovno matematički fakultet u Zagrebu
### Odsjek: Fizika
### Smjer: Fizika i Informatika, nastavnički

import numpy as np
import scipy.integrate as si
import vpython as v


### Početni uvijeti
I1 = 1.5
I2 = 1.0
I3 = 3.1

t0 = 33.0*np.pi/180.        # theta_0  | 33
```

```
p0 = 16.0*np.pi/180.         # psi_0     | 16
f0 = 90.0*np.pi/180.         # phi_0     | 90
vt0 = 18.0*np.pi/180.         # vtheta_0 | 18
vp0 = 144.0*np.pi/180.        # vpsi_0   | 144
vf0 = 18.0*np.pi/180.        # vphi_0    | 18

theta_0 = t0
phi 0 = f0
psi_0 = p0

O10 = vt0*np.cos(p0)+vf0*np.sin(t0)*np.sin(p0)
O20 = -vt0*np.sin(p0)+vf0*np.sin(t0)*np.cos(p0)
O30 = vp0+vf0*np.cos(t0)

### Računamo iz početnih uvijeta
g10 = np.sin(t0)*np.sin(p0)      # koordinate
g20 = np.sin(t0)*np.cos(p0)      # g = gamma
g30 = np.cos(t0)

### Računamo početnu energiju (konstanta)
E = (I1*O10**2 + I2*O20**2 + I3*O30**2)/2+g30
print ('E =', E)

### Računamo projeciju početne kutne količine gibanja na
    #  z-os sustava zvrka (konstanta zbog simetrije)
M30 = I3*O30
print ('M_3 =', M30)

### Određujemo vrijeme izvođenja i broj koraka
tmax = 20.           #20
steps = 20000       #40000
dt = tmax/steps
t = np.arange(0.,tmax,dt)

### Rješavamo diferencijalnu jednadžbu
def f(y,x):
    O1 = y[0]
    O2 = y[1]
    O3 = y[2]

    g1 = y[3]
    g2 = y[4]
    g3 = y[5]

    dtO1 = (I2-I3)*O2*O3/I1 + g2/I1
    dtO2 = (I3-I1)*O3*O1/I2 - g1/I2
    dtO3 = (I1-I2)*O1*O2/I3

    dtg1 = O3*g2 - O2*g3
    dtg2 = O1*g3 - O3*g1
    dtg3 = O2*g1 - O1*g2

    return [dtO1,dtO2,dtO3,dtg1,dtg2,dtg3]

y0 = [O10, O20, O30, g10, g20, g30]

y=si.odeint(f,y0,t)

O1 = y[:,0]
O2 = y[:,1]
O3 = y[:,2]
```

```
g1 = y[:,3]
g2 = y[:,4]
g3 = y[:,5]

### Provjera rješenja integracijom unatrag
tback=np.arange(tmax-dt,-dt,-dt)
y0=[O1[steps-1],O2[steps-1],O3[steps-1], g1[steps-1],
    g2[steps-1], g3[steps-1]]
yback=si.odeint(f,y0,tback)

print ('O1 = ',O10, yback[steps-1,0]) # Uspoređujemo početne
print ('O2 = ',O20, yback[steps-1,1]) # vrijednosti s integriranima
print ('O3 = ',O30, yback[steps-1,2])


### Definiramo prostor na kojem se prikazuje simulacija
prostor = v.canvas(title='Asimetrični zvrk u gravitacijskom polju', width
= 1000,
                   height = 500, forward = v.vector(-1,0,0),
                   up=v.vector(0,0,1), range=1.5,
                   background = v.vector(0.8,0.8,0.8),autoscale=False)

### Definiramo osi
#jedinicni vektori fiksnog sustava
x_prime = v.vector(1,0,0)
y_prime = v.vector(0,1,0)
z_prime = v.vector(0,0,1)


#jedinicni vektori pomicnog sustava
### Jedinični vektori pomičnog sustava
e1 = v.vector(np.cos(phi_0)*np.cos(psi_0)-np.sin(phi_0)*\
              np.cos(theta_0)*np.sin(psi_0), np.sin(phi_0)*\
              np.cos(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.sin(psi_0), np.sin(theta_0)*np.sin(psi_0))
e2 = v.vector(-(np.cos(phi_0)*np.sin(psi_0)+np.sin(phi_0)*\
              np.cos(theta_0)*np.cos(psi_0)), -np.sin(phi_0)*\
              np.sin(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.cos(psi_0),np.sin(theta_0)*np.cos(psi_0))
e3 = v.vector(np.sin(phi_0)*np.sin(theta_0),-np.cos(phi_0)\
              *np.sin(theta_0),np.cos(theta_0))

g = v.vector (-g10,-g20,-g30)

### Ubacujemo zvrk (stožac)
os = -e3
zvrk=v.cone(texture={'file':v.textures.wood_old,'bumpmaps': \
            v.bumpmaps.wood_old}, pos=v.vector(0,0,0),axis=os,\
            size=v.vector(1,1,1), color=v.color.cyan, opacity=0.7)
            # Teksturu stavljamo da lakše uočimo rotacije

os_zvrka=v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
                 v.bumpmaps.stucco}, pos=os,length=1.5, \
                 color = v.color.red,axis=e3, shaftwidth = 0.01)

os_z_vanjski = v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
                       v.bumpmaps.stucco}, pos=os,length=2.0, color = \
                       v.color.black, axis=z prime, shaftwidth = 0.02)

putanja=v.curve(radius=0.005,color=v.color.red)
putanja.append(pos=os+1.5*e3)
```

```
### Stvaramo grafove
graph = v.graph(xtitle='t / s', ytitle='\u03A9 / rad s<sup>-1</sup>',
                align="left")
graph1 = v.gcurve(color=v.color.cyan, label = "\u03A9<sub>1</sub>")
graph2 = v.gcurve(color=v.color.green, label = "\u03A9<sub>2</sub>")
graph3 = v.gcurve(color=v.color.red, label = "\u03A9<sub>3</sub>")
konst = v.graph(xtitle='t / s', align = "right", legend = True)
konst1 = v.gcurve(color=v.color.blue, label = "Energija")
konst2 = v.gcurve(color=v.color.red, label = "Kutna količina gibanja")


### Simuliramo
for i in range(int(steps)):
    v.rate(int(steps/tmax))
        # Ograničimo količinu operacija u sekundi (usporimo simulaciju)

    Omega = O1[i]*e1+O2[i]*e2+O3[i]*e3 # Ovo je trenutni Omega
    os_rotacije = Omega/Omega.mag
    kut_rotacije = Omega.mag*dt

    e1 = e1.rotate(angle=kut_rotacije,axis=os_rotacije)
    e2 = e2.rotate(angle=kut_rotacije,axis=os_rotacije)
    e3 = e3.rotate(angle=kut_rotacije,axis=os_rotacije)

    g.rotate(angle=kut_rotacije,axis=os_rotacije)

    zvrk.rotate(angle=kut_rotacije, axis = os_rotacije, origin = os)
        # Rotiramo zvrk oko odgovarajuće osi
    os_zvrka.rotate(angle=kut_rotacije, axis = os_rotacije, origin =os)
    putanja.append(pos=os+1.5*e3)

    graph1.plot(t[i],O1[i]) # omega 1 - svjetlo plavo
    graph2.plot(t[i],O2[i]) # omega 2 - zeleno
    graph3.plot(t[i],O3[i]) # omega 3 - crveno
    konst1.plot(t[i], ((I1*O1[i]**2 + I2*O2[i]**2 + I3*O3[i]**2)/2\
                       + g3[i])) # Numerički izračunata energija
    konst2.plot(t[i], (g1[i]*I1*O1[i]+g2[i]*I2*O2[i]+g3[i]*I3*O3[i]))
            # Kutna količina gibanja
```

G. *Simulacija Kovalevskaya zvrka*

```
### Za diplomski rad
### Simulacija Kovalevskaya zvrka
### Mentor: Tamara Nikšić
### Student: Jasnik Ivanjek
### Prirodoslovno matematički fakultet u Zagrebu
### Odsjek: Fizika
### Smjer: Fizika i Informatika, nastavnički

import numpy as np
import scipy.integrate as si
import vpython as v


### Početni uvijeti
I1 = 1.0
I2 = 2*I1
```

```
I3 = 2*I1

t0 = 33.0*np.pi/180.          # theta_0   | 33
p0 = 16.0*np.pi/180.          # psi_0     | 16
f0 = 90.0*np.pi/180.          # phi_0     | 90
vt0 = 9.0*np.pi/180.          # vtheta_0  | 18
vp0 = 32.0*np.pi/180.         # vpsi_0    | 144
vf0 = 18.0*np.pi/180.         # vphi_0    | 18

theta_0 = t0
phi_0 = f0
psi_0 = p0

O10 = vt0*np.cos(p0)+vf0*np.sin(t0)*np.sin(p0)
O20 = -vt0*np.sin(p0)+vf0*np.sin(t0)*np.cos(p0)
O30 = vp0+vf0*np.cos(t0)

### Računamo iz početnih uvijeta
g10 = np.sin(t0)*np.sin(p0)      # koordinate
g20 = np.sin(t0)*np.cos(p0)      # g = gamma
g30 = np.cos(t0)

### Računamo početnu energiju (konstanta)
E = (I1*O10**2 + I2*O20**2 + I3*O30**2)/2+g30
print ('E =', E)

### Računamo Kovaleskayinu konstantu
K_0 = (I1*(O30**2-O20**2)-g30)**2 + (I2*O20*O30-g20)**2
print ('K =', K_0)

### Računamo projeciju početne kutne količine gibanja na
    #  z-os sustava zvrka (konstanta zbog simetrije)
M30 = I3*O30
print ('M_3 =', M30)

### Određujemo vrijeme izvođenja i broj koraka
tmax = 60.           #20
steps = 10000       #40000
dt = tmax/steps
t = np.arange(0.,tmax,dt)

### Rješavamo diferencijalnu jednadžbu
def f(y,x):
    O1 = y[0]
    O2 = y[1]
    O3 = y[2]

    g1 = y[3]
    g2 = y[4]
    g3 = y[5]

    dtO1 = (I2-I3)*O2*O3/I1 + g2/I1
    dtO2 = (I3-I1)*O3*O1/I2 - g1/I2
    dtO3 = (I1-I2)*O1*O2/I3

    dtg1 = O3*g2 - O2*g3
    dtg2 = O1*g3 - O3*g1
    dtg3 = O2*g1 - O1*g2

    return [dtO1,dtO2,dtO3,dtg1,dtg2,dtg3]
```

```
y0 = [O10, O20, O30, g10, g20, g30]

y=si.odeint(f,y0,t)

O1 = y[:,0]
O2 = y[:,1]
O3 = y[:,2]
g1 = y[:,3]
g2 = y[:,4]
g3 = y[:,5]

K = (I1*(O3**2-O2**2)-g3)**2 + (I2*O2*O3-g2)**2

### Provjera rješenja integracijom unatrag
tback=np.arange(tmax-dt,-dt,-dt)
y0=[O1[steps-1],O2[steps-1],O3[steps-1], g1[steps-1],
    g2[steps-1], g3[steps-1]]
yback=si.odeint(f,y0,tback)

print ('O1 = ',O10, yback[steps-1,0]) # Uspoređujemo početne
print ('O2 = ',O20, yback[steps-1,1]) # vrijednosti s integriranima
print ('O3 = ',O30, yback[steps-1,2])


### Definiramo prostor na kojem se prikazuje simulacija
prostor = v.canvas(title='Kovalevskayin zvrk', width = 1000,
                   height = 500, forward = v.vector(-1,0,0),
                   up=v.vector(0,0,1), range=1.5,
                   background = v.vector(0.8,0.8,0.8),autoscale=False)

### Definiramo osi
#jedinicni vektori fiksnog sustava
x_prime = v.vector(1,0,0)
y_prime = v.vector(0,1,0)
z_prime = v.vector(0,0,1)


#jedinicni vektori pomicnog sustava
### Jedinični vektori pomičnog sustava
e1 = v.vector(np.cos(phi_0)*np.cos(psi_0)-np.sin(phi_0)*\
              np.cos(theta_0)*np.sin(psi_0), np.sin(phi_0)*\
              np.cos(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.sin(psi_0), np.sin(theta_0)*np.sin(psi_0))
e2 = v.vector(-(np.cos(phi_0)*np.sin(psi_0)+np.sin(phi_0)*\
              np.cos(theta_0)*np.cos(psi_0)), -np.sin(phi_0)*\
              np.sin(psi_0)+np.cos(phi_0)*np.cos(theta_0)*\
              np.cos(psi_0),np.sin(theta_0)*np.cos(psi_0))
e3 = v.vector(np.sin(phi_0)*np.sin(theta_0),-np.cos(phi_0)\
              *np.sin(theta_0),np.cos(theta_0))

g = v.vector (-g10,-g20,-g30)

### Ubacujemo zvrk (stožac)
os = -e3
zvrk=v.cone(texture={'file':v.textures.wood_old,'bumpmaps': \
            v.bumpmaps.wood_old}, pos=v.vector(0,0,0),axis=os,\
            size=v.vector(1,1,1), color=v.color.orange, opacity=0.7)
            # Teksturu stavljamo da lakše uočimo rotacije

os_zvrka=v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
                 v.bumpmaps.stucco}, pos=os,length=1.5, \
```

```
                color = v.color.red,axis=e3, shaftwidth = 0.01)

os_z_vanjski = v.arrow(texture={'file':v.textures.stucco,'bumpmaps': \
                        v.bumpmaps.stucco}, pos=os,length=2.0, color = \
                        v.color.black, axis=z_prime, shaftwidth = 0.02)

putanja=v.curve(radius=0.005,color=v.color.red)
putanja.append(pos=os+1.5*e3)

### Stvaramo grafove
graph = v.graph(xtitle='t / s', ytitle='\u03A9 / rad s<sup>-1</sup>',
                align="left")
graph1 = v.gcurve(color=v.color.cyan, label = "\u03A9<sub>1</sub>")
graph2 = v.gcurve(color=v.color.green, label = "\u03A9<sub>2</sub>")
graph3 = v.gcurve(color=v.color.red, label = "\u03A9<sub>3</sub>")
konst = v.graph(xtitle='t / s', align = "right", legend = True)
konst1 = v.gcurve(color=v.color.blue, label = "Energija")
konst2 = v.gcurve(color=v.color.red, label = "Kutne količina gibanja")
konst3 = v.gcurve(color=v.color.green, label = "Kovaleskayina konstanta")

### Simuliramo
for i in range(int(steps)):
    v.rate(int(steps/tmax))
        # Ograničimo količinu operacija u sekundi (usporimo simulaciju)

    Omega = O1[i]*e1+O2[i]*e2+O3[i]*e3 # Ovo je trenutni Omega
    os_rotacije = Omega/Omega.mag
    kut_rotacije = Omega.mag*dt

    e1 = e1.rotate(angle=kut_rotacije,axis=os_rotacije)
    e2 = e2.rotate(angle=kut_rotacije,axis=os_rotacije)
    e3 = e3.rotate(angle=kut_rotacije,axis=os_rotacije)

    g.rotate(angle=kut_rotacije,axis=os_rotacije)

    zvrk.rotate(angle=kut_rotacije, axis = os_rotacije, origin = os)
        # Rotiramo zvrk oko odgovarajuće osi
    os_zvrka.rotate(angle=kut_rotacije, axis = os_rotacije, origin =os)
    putanja.append(pos=os+1.5*e3)

    graph1.plot(t[i],O1[i]) # omega 1 - svjetlo plavo
    graph2.plot(t[i],O2[i]) # omega 2 - zeleno
    graph3.plot(t[i],O3[i]) # omega 3 - crveno
    konst1.plot(t[i], ((I1*O1[i]**2 + I2*O2[i]**2 + I3*O3[i]**2)/2\
                        + g3[i])) # Numerički izračunata energija
    konst2.plot(t[i], (g1[i]*I1*O1[i]+g2[i]*I2*O2[i]+g3[i]*I3*O3[i]))
            # Kutna količina gibanja
    konst3.plot(t[i], (K[i])) # Kovaleskayina konstanta
```

Literatura

[1] Spinning top – duration, (18.11.2006), Guinness World Records, http://www.guinnessworldrecords.com/world-records/spinning-top-duration/, 11.2.2019.

[2] Goldstein, H.; Poole, C.; Safko, J. Classical Mechanics. Third edition; Addison-Wesley, 2002

[3] Landau, L. D.; Lifshitz, E. M. Mechanics. Third edition: Butterworth-Heinenann, 2000

[4] Tél, T.; Gruitz, M. Chaotic Dynamics: an introduction based on classical mechanics. First edition, Cambridge University Press, 2006

[5] Arnold, V. I. Mathematical Methods of Classical Mechanics. Second edition, Springer-Verlag, 1989

[6] History of Spinning Tops, (21.11.2016), Art of Play, https://www.artofplay.com/blogs/articles/history-of-spinning-tops/, 11.2.2019.

[7] History of the Spin Top, (), Museum of Yo-yo history, http://www.yoyomuseum.com/museum_view.php?action=profiles&subaction=spin top/, 11.2.2019.