KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

Romas Meškauskas

ŠOKINĖJANČIO KAMUOLIUKO UŽDAVINIO SPRENDIMO METODAI

Magistro darbas

Vadovas
asist. Edita Šakytė

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

TVIRTINU
Katedros vedėjas
prof. habil.dr. V.Pekarskas
2013 06 02

ŠOKINĖJANČIO KAMUOLIUKO UŽDAVINIO SPRENDIMO METODAI

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
_____ asist. E. Šakytė
2013 06 01

Recenzentas
_____ doc. dr. R. Kregždytė
2013 06 01

Atliko
FMMM-1 gr. stud.
_____ R. Meškauskas
2013 05 30

KAUNAS, 2013

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Rimantas Rudzkis, profesorius (VU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Jonas Valantinas, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., direktoriaus pavaduotojas (UAB „Danet Baltic“)

Meškauskas R. Methods for solving bouncing ball problem: Master's work in applied mathematics / supervisor asist. E. Šakytė: Department of Mathematical Research in Systems, Faculty of Fundamental Science, Kaunas University of Technology. – Kaunas, 2013. – 54.

SUMMARY

The bouncing ball system has been the subject of intense studies both in experiments and theory during the past decades. It consists of a point-like particle bouncing on a vertically and sinusoidally oscillating table. This is quite a simple system however it reveals a wide range of dynamical behavior and has become one of the prototypical examples in nonlinear dynamics and chaos theory. It was shown, that a slight change of initial cases can cause extremely different movement of the ball.

This system is used for understanding more complex systems, such as probability machines, quantum billiard, chaos control, robotics and so on.

In this work there was created a program that simulates bouncing ball and table movement. Results of simulation were compared with results of Newton's method and High bounce approximation. This comparison revealed that both iterative methods give similar results, however the computation time of the simulation is a bit better generally, while the High bounce approximation is extremely fast, but gives inaccurate results except some special cases. Moreover, there was created programs for visualization of the chaotic maps and was made a comparison between the exact system and High bounce approximation standard maps.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Šokinėjančio kamuoliuko sistema pastaraisiais dešimtmečiais buvo intensyvių teorinių bei eksperimentinių tyrimų subjektas. Ji susideda iš kamuoliuko, vertikaliai šokinėjančio ant sinusoidiškai vibruojančio stalo. Tai sąlyginai paprasta sistema, tačiau atskleidžianti platų skirtingos dinaminės elgsenos spektrą ir yra vienas pagrindinių netiesinės dinamikos ir chaoso teorijos pavyzdžių.

Pirmą kartą ši sistema buvo paminėta E. Fermi 1949 metais, iškėlusio hipotezę apie didelės kosminių spindulių energijos prigimtį. Tačiau šokinėjančio kamuoliuko uždavinys plačiau pradėtas nagrinėti tik prieš tris dešimtmečius [24].

Šokinėjančio kamuoliuko sistema naudojama analizuojant daug sudėtingesnes sistemas, tokias kaip tikimybinės mašinos [4], chaoso kontrolė [6], [23], kvantinis biliardas [18], dalelių srautų fizika, robotika [10] ir kt. Vienas pavyzdžių- robotika, kur siekiama programavimo pagalba užtikrinti valdomą elektronikos ir mechanikos sričių robotų bendradarbiavimo ir veiklos procesą. Šiuo atveju šokinėjančio kamuoliuko ant vibruojančio stalo sistema pasitarnauja kuriant robotus, gebančius vaikščioti nestabiliu vibruojančiu paviršiumi. [10]

Šio darbo tikslas – sukurti šokinėjančio kamuoliuko judėjimą imituojančią programą bei atlikti simuliaciją, pritaikius teorinius šio uždavinio sprendimo metodus; palyginti įvairiais metodais gautuosius rezultatus bei palyginti metodus, įvertinant skaičiavimų trukmes. Taip pat sukurti programą chaotinio žemėlapio vizualizacijai bei palyginti standartinį žemėlapį Aukšto šuolio aproksimacijos bei tikslios sistemos atvejais.

1. BENDROJI DALIS

1.1. CHAOSO SĄVOKA

Chaosas – tai neperiodinė ilgalaikė deterministinės sistemos elgsena, parodanti stiprią priklausomybę nuo pradinių sąlygų. Šokinėjantis kamuoliukas yra vienas paprasčiausių chaotinės sistemos pavyzdžių [25]. Tai viena iš trijų pagrindinių literatūroje [12] išskiriamų smūginių sistemų klasių.

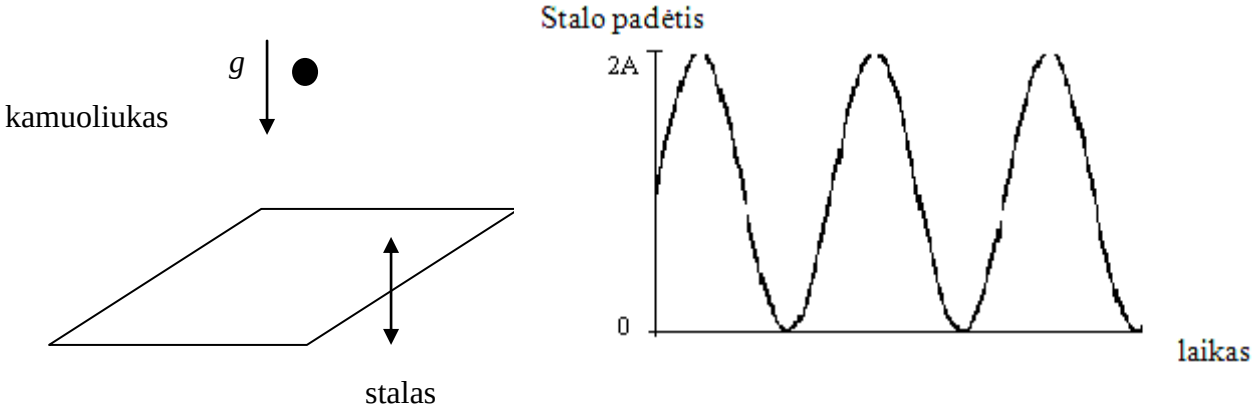
Chaosas fizikoje reiškia neabejotinai netiesinių dinaminių sistemų nagrinėjimą. Svarbus netiesinių lygčių sprendimo įrankis- sistemos vizualizacija taip vadinamoje fazinėje erdvėje. Matematikoje ir fizikoje, fazinė erdvė yra erdvė, kurioje yra atvaizduojamos visos galimos sistemos būsenos, kurios atitinka vienintelį tašką fazinėje erdvėje. Mechaninių sistemų fazinė erdvė yra sudaryta iš padėties ir impulso kintamųjų. Tokia, schema, kurioje yra padėtis ir momentas, dar vadinama faziniu portretu arba fazine diagrama.

Fazinis portretas – geometrinis dinaminių sistemų trajektorijų atvaizdavimas fazinėje plokštumoje. Kiekviena pradinė sąlyga atspindi skirtingą kreivę arba tašką. Faziniai portretai yra neįkainojami studijuojant dinamines sistemas. Jie sudaryti iš tipinių trajektorijų diagramų padėties erdvėje. Tai rodo informaciją apie tai, kaip vaizduojamas parametras, pvz., atraktorius. Fazinių portretų grafikas dinaminei sistemai brėžia sistemos trajektorijas (su rodyklėmis), stabilumo būsenas (su taškais) ir nestabilumo būsenas (su apskritimais). Ašys yra būsenų kintamieji.

Netampriai šokinėjantis kamuoliukas ant vibruojančios plokštumos yra paprastas eksperimentas, turintis sudėtingą dinaminę elgseną, demonstruojantis kai kurias pagrindines netiesinės dinamikos koncepcijas. Jeigu kamuoliuką laikytume dalele, tuomet tokia sistema implikuoja daug įdomių reiškinių, pavyzdžiui, Braziliško riešuto efektą (angl. *Brazil-Nut Effect*), Leidenfrost efektą. [25]

Dinaminių sistemų teorijos matematinis tikslas- suprasti dinaminio proceso asimptotinę elgseną. Asimptotinių sprendinių aibė vadinama atraktoriumi. Jų yra įvairių: fiksuoti taškai, ribiniai ciklai, keistieji atraktoriai ir t.t. [21] Atraktorius apibrėžiamas kaip mažiausias vienetas, kuris negali būti skaidomas į du ar dar daugiau atraktorių, turinčių skirtingas traukos sritis. Šis draudimas yra būtinas, nes dinaminė sistema gali turėti daug atraktorių, kurių kiekvienas turi savo atskirą traukos sritį. Konservatyvios sistemos, kur judėjimas yra periodinis, atraktorių neturi. [1]

1.2. ŠOKINĖJANČIO KAMUOLIUKO MODELIS



1.1 pav. Sistema

Nagrinėkime sistemą, sudarytą iš kamuoliuko, kurio masė *m*, bei stalo, kurio masė *M*. Stalas juda sinusoidiškai, o tuo tarpu kamuoliukas paleidžiamas kristi iš tam tikro aukščio x_0 virš stalo (žr. 1.1 pav.). Tam, kad palengvintume skaičiavimus, padarysime keletą prielaidų: pirma, kamuoliuką laikysime materialiuoju tašku, antra- stalo masė yra žymiai didesnė negu kamuoliuko, trečia, laikysime, jog tiek stalas, tiek kamuoliukas juda tiesiai vertikalia linija, ketvirta- laikysime, jog nėra oro pasipriešinimo. Tuomet tarp gretimų smūgių (su stalu), kamuoliuko judėjimas aprašomas remiantis Niutono dėsniais. Jo trajektorija yra tokia:

$$x(t) = x_k + v_k(t - t_k) - \frac{g}{2}(t - t_k)^2,\ t_k \le t \le t_{k+1}, \qquad \textbf{(1.1)}$$

čia *g*- laisvojo kritimo pagreitis $\left(g \approx 9{,}81\frac{m}{s^2}\right)$, t_k - *k*-tojo smūgio laiko momentas, x_k, v_k - atitinkamai kamuoliuko padėtis bei greitis *k*-tojo smūgio momentu.

Kadangi stalo masė yra didelė, palyginus su kamuoliuko mase, todėl jo judėjimas nėra įtakojamas kamuoliuko smūgių ir aprašomas taip:

$$s(t) = A(\sin(\omega t + \theta_0) + 1), \qquad \textbf{(1.2)}$$

jo greitis

$$v(t) = s'(t) = A\omega\cos(\omega t + \theta_0). \qquad \textbf{(1.3)}$$

Dydis $\omega = 2\pi f$ - kampinis dažnis, θ_0 – pradinė fazė. Atstumas tarp kamuoliuko ir stalo yra lygus

$$d(t) = x(t) - s(t). \qquad \textbf{(1.4)}$$

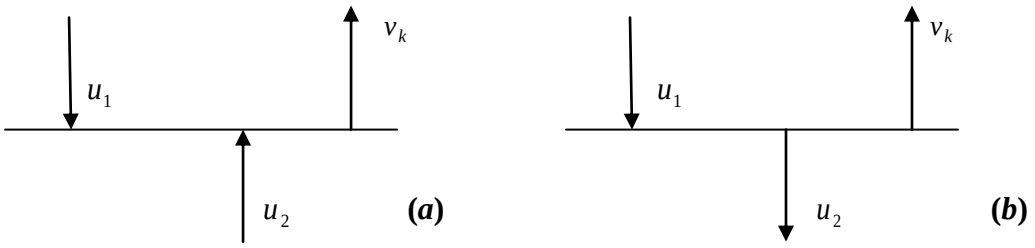
Pirmas laiko momentas $t > t_k$, kai $d(t) = 0$, yra sekantis susidūrimo momentas

$$0 = x_k + v_k(t_{k+1} - t_k) - \frac{g}{2}(t_{k+1} - t_k)^2 - A(\sin(\omega t_{k+1} + \theta_0) + 1)$$

Kadangi *k*-tojo smūgio metu $x_k = s(t_k)$, tai gauname:

$$f(t_{k+1}) = A\sin(\omega t_k + \theta_0) + v_k(t_{k+1} - t_k) - \frac{g}{2}(t_{k+1} - t_k)^2 - A\sin(\omega t_{k+1} + \theta_0) \equiv 0 \qquad \textbf{(1.5)}$$

Ši lygtis aprašo kamuoliuko padėtį (laboratorijos) grindų atžvilgiu. Čia v_k - nežinomas.



1.2 pav. Krintantis bei atšokantis kamuoliukas: (*a*) nuo vertikaliai aukštyn judančio stalo, (*b*) nuo vertikaliai žemyn judančio stalo

1.2 pav. iliustruoja kamuoliuką, krintantį greičiu u_1 vertikaliai žemyn į stalą ir atšokantį vertikaliai aukštyn greičiu v_k. 1.2 pav. (*a*) dalyje stalas juda vertikaliai aukštyn greičiu u_2, o 1.2 pav. (*b*) stalas juda vertikaliai žemyn, tačiau $u_1 > u_2$. Remiantis sąryšiu

|*kamuoliuko reliatyvus greitis prieš smūgį*| = α|*kamuoliuko reliatyvus greitis po smūgio*|, **(1.6)**

šiais dviem atvejais gauname:

$v_k - u_2 = \alpha(u_1 + u_2)$, $v_k + u_2 = \alpha(u_1 - u_2)$,

arba

$$v_k = \alpha \cdot u_1 + (1+\alpha)u_2, \; v_k = \alpha \cdot u_1 - (1+\alpha)u_2, \qquad \textbf{(1.7)}$$

čia α yra restitucijos parametras- kamuoliuko energijos praradimo kiekvieno smūgio metu matas, $0 \le \alpha \le 1$. Restitucijos parametro reikšmės gali varijuoti nuo 0,2 iki 0,8, naudojant skirtingas medžiagas gaminant kamuoliuką, pvz., medieną, plastiką, plieną ir t.t. [19]

Remiantis (1.7), jei v_k^* - kamuoliuko greitis, u_k - stalo greitis grindų atžvilgiu, gauname atšokančio kamuoliuko greitį v_k:

$$v_k = (1+\alpha)u_k - \alpha \cdot v_k^*. \quad \textbf{(1.8)}$$

Kamuoliuko greitis tarp smūgių (t.y. kai skrieja laisvai):

$$v(t) = \dot{x}(t) = v_k - g(t - t_k).$$

Iš čia gauname, kad kamuoliuko greitis prieš pat (k + 1)-ąjį smūgį:

$$v_{k+1} = v_k - g(t_{k+1} - t_k).$$

Laiko tarpas tarp smūgių randamas [14]:

$$\Delta t_k = \frac{\mathrm{v}_k + \sqrt{v_k^2 - 2\Delta x_k \cdot g}}{g}, \quad \textbf{(1.9)}$$

kur Δx_k reiškia kamuoliuko padėties pokytį tarp gretimų smūgių. Išvesime šią formulę.

Remiantis (1.1), kamuoliuko pozicija (k + 1)-ojo smūgio momentu:

$x_{k+1} = x_k + v_k(t_{k+1} - t_k) - \frac{g}{2}(t_{k+1} - t_k)^2$, pažymėjus $\Delta t_k = t_{k+1} - t_k$:

$x_{k+1} - x_k = v_k \Delta t_k - \frac{g}{2}\Delta t_k{}^2$, pažymėjus $\Delta x_k = x_{k+1} - x_k$:

$\Delta x_k = v_k \Delta t_k - \frac{g}{2}\Delta t_k{}^2$,

išsprendus šią lygtį Δt_k atžvilgiu ir paėmus didesniąją šaknį, gauname (1.9).

Atskiru atveju, kai stalo greitis $u_k = 0$, gauname laiko tarpą (periodą) tarp k-tojo (k – 1)-ojo smūgių (jį vadinsime ciklu) [25]:

$$\Delta T_k = \frac{2\mathrm{v}_0 \alpha^k}{g}, \quad \textbf{(1.10)}$$

kiekvieno k-tojo ciklo didžiausias aukštis

$$h_k = x_0 + \frac{\left(\mathrm{v}_0 \alpha^k\right)^2}{2g}. \quad \textbf{(1.11)}$$

1.3. ŠAKNŲ ATSKYRIMO UŽDAVINYS [17]

Intervalas (*a*; *b*) yra šaknies izoliacijos intervalas, jei lygtis $F(x) = 0$ tame intervale turi vienintelę šaknį.

Šaknų atskyrimo uždavinys formuluojamas taip: duota lygtis $F(x) = 0$; reikia rasti jos šaknų izoliacijos intervalus.

Dažniausiai taikomi šio uždavinio sprendimo būdai yra:

1. Grafinis būdas.
2. Fizikinis būdas.
3. Specialūs būdai.

Atliekant šokinėjančio kamuoliuko simuliaciją, *k*-tojo smūgio laiko momento t_k izoliacijos intervalui rasti buvo naudojamas fizikinis būdas. Galimi šie atvejai:

- kamuoliukas krenta iš aukštai $(x_{k-1} > 2A)$. t_k izoliacijos intervalas šiuo atveju $\left[t_{k-1} + \frac{v_{k-1}}{g} + \sqrt{\frac{v_{k-1}^2 + 2g(x_{k-1} - 2A)}{g}}; t_{k-1} + \frac{v_{k-1}}{g} + \sqrt{\frac{v_{k-1}^2 + 2gx_{k-1}}{g}} \right]$;
- kamuoliukas krenta iš žemai $(x_{k-1} \le 2A)$. t_k izoliacijos intervalas šiuo atveju $\left[t_{k-1} + \frac{v_{k-1}}{g} + \sqrt{\frac{v_{k-1}^2}{g}}; t_{k-1} + \frac{v_{k-1}}{g} + \sqrt{\frac{v_{k-1}^2 + 2gx_{k-1}}{g}} \right]$,

čia t_{k-1} - *k* -1-ojo smūgio laiko momentas, *g*- laisvojo kritimo pagreitis, v_{k-1} - kamuoliuko greitis *k* -1-ojo smūgio metu, *A*- stalo judėjimo amplitudė.

1.4. ŠOKINĖJANČIO KAMUOLIUKO UŽDAVINYS

Uždavinys yra toks: duoti t_k ir v_k. Rasti pirmąjį sprendinį t_{k+1}- sekančio smūgio laiko momentą, tokį, kad $t_{k+1} > t_k$. Šokinėjančio kamuoliuko ir stalo smūgių laiko momentai randami iš (1.5), šokinėjančio kamuoliuko dinamika simuliuojama kompiuteriu, naudojantis (1.5) bei (1.8) lygtimis. Skaičiuojant t_{k+1}, sprendžiame (1.5) lygtį. Ši lygtis negali būti išsprendžiama analiziškai, todėl jos sprendime naudojami skaitiniai metodai.

1.5. AUKŠTO ŠUOLIO APROKSIMACIJA

Kartais literatūroje [10], [7] naudojama smūgio laiko aproksimacija, dar vadinama Aukšto šuolio aproksimacija (angl. *High bounce approximation*), apibrėžiama:

$$t_{k+1} = \frac{2v_k}{g} + t_k \qquad \textbf{(1.12)}$$

Reiktų atkreipti dėmesį į tai, jog ši aproksimacija galioja tik tuomet, kai stalo aukščio pokytis yra daug mažesnis už didžiausią aukštį, į kurį atšoka kamuoliukas, t.y. kamuoliuko orbita yra simetrinė jo didžiausio pakilimo aukščio atžvilgiu. Šios aproksimacijos privalumas yra tas, jog skaičiavimai atliekami labai greitai (reikia 10^{-9} sekundžių atlikti šį skaičiavimą [10]).

1.6. SISTEMOS PARAMETRŲ TRANSFORMAVIMAS Į BEDIMENSIUS

Šokinėjančio kamuoliuko sistema turi nemažai parametrų- restitucijos parametrą, stalo amplitudę, kampinį dažnį, laisvojo kritimo pagreitį ir t.t. Literatūroje [3], [5], [6], [7], [8], [9], [11], [12], [13], [14], [24] siūloma nagrinėti transformuotą sistemą, kurios kintamieji- bedimensiai. Jie skaičiuojami pagal formules [7]:

$$\theta_k = \omega t_k + \theta_0\,, \qquad \textbf{(1.13)}$$

$$v_k = \frac{2\omega}{g} v_k\,. \qquad \textbf{(1.14)}$$

Taip pat įvedamas dar vienas parametras

$$\beta = \frac{2\omega^2(1+\alpha)A}{g}, \qquad \textbf{(1.15)}$$

Atliekant šokinėjančio kamuoliuko simuliaciją (skaičiavimuose), braižant chaotinį žemėlapį nagrinėjami bedimensiai dydžiai.

1.7. STANDARTINIS ŽEMĖLAPIS

Literatūroje [7] dinaminės sitemos analizei siūloma nagrinėti chaotinį (standartinį) žemėlapį. Chaotinis žemėlapis matematikoje yra žemėlapis, demonstruojantis tam tikrą nagrinėjamos sistemos chaotinės elgsenos rūšį. Standartinis žemėlapis (dar vadinamas Chirikov-Taylor žemėlapiu arba tiesiog Chirikov standartiniu žemėlapiu) yra chaotinis žemėlapis, kuris kvadratą $2\pi \times 2\pi$ atvaizduoja į patį save, išlaikant vienodą plotą. [16]

Aukšto šuolio aproksimacijos atveju chaotinis žemėlapis aprašomas [7] fazės lygtimi:

$$\theta_{k+1} = \theta_k + \nu_k \pmod{2\pi}, \qquad \textbf{(1.16a)}$$

greičio lygtimi:

$$\nu_{k+1} = \alpha\nu_k + \beta\cos(\theta_k + \nu_k), \qquad \textbf{(1.16b)}$$

kur α - restitucijos parametras, parametrai β, θ_k, ir ν_k - randami atitinkamai pagal (1.15), (1.13) ir (1.14) formules.

Šis žemėlapis turi fiksuotus taškus (θ,ν):

$$\left(\pm\arccos\left(\frac{2k\pi(1-\alpha)}{\beta}\right),2k\pi\right),\ k\in Z.$$

Paėmus $\alpha = 1$, gauname standartinį žemėlapį. Šiuo atveju fiksuoti taškai (θ,ν):

$$\left(\frac{\pi}{2},2k\pi\right),\left(\frac{3\pi}{2},2k\pi\right),\ k\in Z.$$

Analogiškai, tikslios bedimensės sistemos chaotinis žemėlapis aprašomas fazės lygtimi:

$$0 = \beta(\sin\theta_k - \sin\theta_{k+1}) + (1+\alpha)(\nu_k\ (\theta_{k+1} - \theta_k) - (\theta_{k+1} - \theta_k)^2), \qquad \textbf{(1.17a)}$$

bei greičio lygtimi:

$$\nu_{k+1} = \beta\cos\theta_{k+1} - \alpha(\nu_k - 2(\theta_{k+1} - \theta_k)). \qquad \textbf{(1.17b)}$$

1.8. KITI ŠOKINĖJANČIO KAMUOLIUKO UŽDAVINIO SPRENDIMO METODAI

Be jau 1.5 skyrelyje paminėtos Aukšto šuolio aproksimacijos, šokinėjančio kamuoliuko uždavinį galima spręsti kritinių taškų, žingsnelių laiko ašimi bei kitais iteraciniais metodais (Niutono [9], [22], pusiaukirtos ir kt.). Aptarsime šiuos metodus plačiau.

1.8.1. BENDRASIS ŠOKINĖJANČIO KAMUOLIUKO UŽDAVINIO SPRENDIMO ALGORITMAS

Šokinėjančio kamuoliuko uždavinio sprendimo algoritmas būtų toks:

1. Iki pirmo smūgio kamuoliukas juda vien veikiamas žemės traukos jėgos ir jo padėtis aprašoma (įsitikinti, kad tikrai taip yra, t.y. sudaryti dif. lygtį ir ją išspręsti):

$$x(t) = x_0 + v_0(t - t_0) - \frac{g}{2}(t - t_0)^2; \qquad \textbf{(1.18)}$$

$x_0 = x(t_0),\ v_0 = \dot{x}(t_0)$ - pradinės sąlygos.

Pastaba. Ši funkcija analogiška (1.1) funkcijai, tik čia įstatytos pradinės sąlygos, o (1.1) formulėje – smūgio sąlygos (iš esmės tai yra tas pats: galima laikyti, kad dalelės padėtis smūgio metu atitinka naujas pradines sąlygas).

2. Iteraciniu būdu (t.y. sekdami ar $d(t) \neq 0$ (žr. (1.4) formulę)) surandame pirmo smūgio:

- laiką t_1;
- $x_1 = x(t_1)$;
- greitį:
  - greitį prieš pat smūgį: $v_1^* = v_0 - g(t - t_0)$;
  - greitį smūgio metu (pagal (1.8) formulę): $v_1 = (1 + \alpha)u_1 - \alpha \cdot v_1^*$.

3. Visiems sekantiems smūgiams:

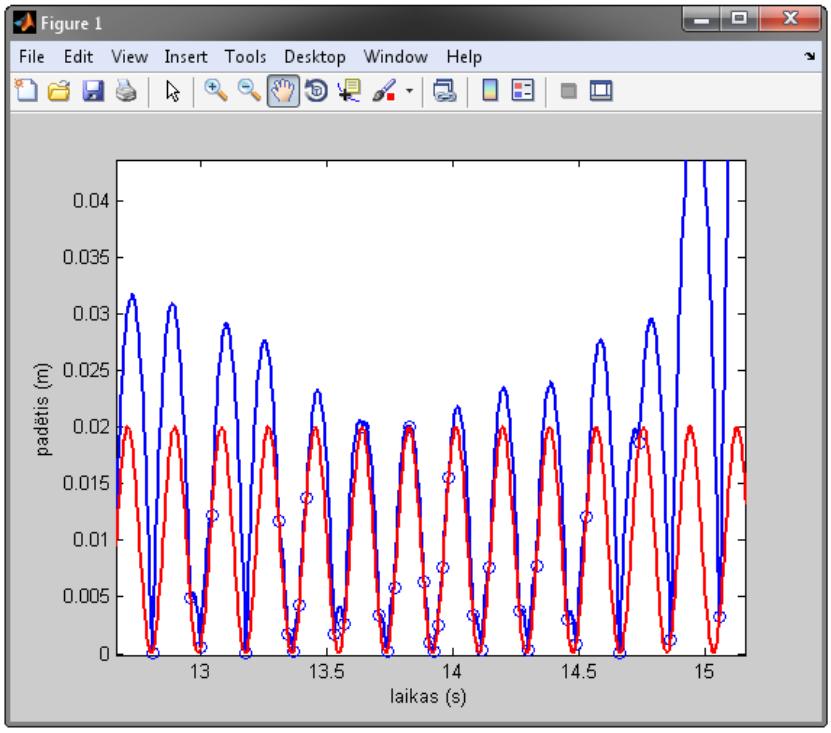
- sprendžiam (1.5) lygtį t_{k+1} atžvilgiu – surandame sekančio smūgio momentą t_{k+1};
- įstatom t_{k+1} į (1.8) formulę ir gauname kamuoliuko greitį sekančio smūgio metu;
- tarpiniais laiko momentais *t*, $t_k \le t \le t_{k+1}$ vizualizuojam pagal (1.1) formulę.

4. Kartojame 3 algoritmo žingsnį tiek kartų, kiek smūgių norime pavaizduoti.

Pastaba. Kadangi (1.1) ir (1.18) formulės analogiškos, tai ir 2 ir 3 žingsniai yra identiški: t.y. pirmo smūgio galima ne ieškoti iteraciniu būdu, o spręsti (1.5) lygtį. Taigi, galima 2 ir 3 žingsnius apjungti.

1.8.2. ŽINGSNELIŲ LAIKO AŠIMI METODAS

Sprendžiant šokinėjančio kamuoliuko uždavinį, galima pamanyti, jog pirmasis sprendinys t_{k+1} nesunkiai randamas parinkus kokį nors $t_g > t_k$ taip, kad būtų garantuojama, jog kamuoliukas atsitrenks į stalą iki laiko momento t_g. Tuomet ieškant (1.5) šaknų intervale $[t_k, t_g]$ galima pasinaudoti bet kuriuo šaknų radimo algoritmu, pvz., Niutono metodu, pusiaukirtos (angl. *bisection*) metodu. Tačiau gali būti du sprendiniai, esantys labai arti vienas šalia kito, taigi, jeigu parinktume t_g kiek dešiniau nuo antrojo sprendinio, nebūtume užtikrinti, jog nagrinėjamas šaknų radimo metodas duos teisingą sprendinį, žr. pavyzdį (1.3 pav., kuriame taškais pažymėta stalo bei kamuoliuko judėjimo trajektorijų sankirtos vietos) [9].



1.3 pav. Kamuoliuko bei stalo judėjimo dinamika

Taigi, kaip tiksliai pataikyti į pirmąją lygties šaknį? Yra keletas metodų šiam uždaviniui spręsti. Pirmasis- mažais žingsneliais keliauti laiko *t* ašimi, kol atstumas $d(t)$tarp kamuoliuko ir stalo pasidaro lygus 0, t.y. galioja (1.5). Šiuo atveju sprendžiamas toks uždavinys: reikia parinkti tokį žingsnį, kuris garantuotų ne daugiau kaip vieną sprendinį. Tufillaro [20] kiekvienoje iteracijoje parenka naują žingsnio dydį, kuris paremtas kamuoliuko skrydžio laiko įverčiu.

Literatūroje [10] pateikiamas dar vienas algoritmas, šokinėjančio kamuoliuko smūgio momentams rasti. Jo idėja būtų tokia: imamas laiko skaitliukas, dydis *t* (laikas) didinamas mažais, diskrečiais žingsneliais. Kiekviename žingsnyje atnaujinama kamuoliuko ir stalo padėtis ekrane bei ieškoma kada jų trajektorijos susikirs. Kadangi skaičiuojama dvejetainiu pagrindu, laiko žingsnelio dydį imant $\frac{1}{2^n}$ formoje, užtikrinamas rezultatų tikslumas, kuris galėtų sumažėti dėl apvalinimo paklaidų.

Tačiau tam, kad kamuoliuko ir stalo padėtis ekrane būtų tinkamai atnaujinama, minėtasis žingsnelio dydis privalo būti pakankamai didelis, vadinasi, kada randamas kamuoliuko smūgis į stalą, yra platus dydžio, kuris galėtų būti tikroji kamuoliuko padėtis smūgio metu, intervalas. Tai reiškia pakankamai didelį

tikslumo praradimą, kuomet kiekvieno smūgio metu atnaujinami faziniai portretai, perduodant vaizdą labai žema rezoliucija. Taigi, šis metodas nėra efektyvus, todėl nagrinėtini kiti metodai.

1.8.3. KRITINIŲ TAŠKŲ METODAS [9]

Antrasis metodas paremtas elementarių skaičiavimų idėjomis. Žinoma, jog nėra daugiau kaip vienos šaknies tarp dviejų gretimų funkcijos $f(t_{k+1})$ (5 formulė) kritinių taškų. Analogiškai, nėra daugiau kaip vieno funkcijos kritinio taško tarp dviejų gretimų funkcijos vingio taškų. Kadangi funkcijos $f(t_{k+1})$ vingio taškai yra jos antrosios išvestinės šaknys, tuomet jei galime rasti

$$f''(t_{k+1}) = \left(A\sin(\omega t_k + \theta_0) + v_k(t_{k+1} - t_k) - \frac{g}{2}(t_{k+1} - t_k)^2 - A\sin(\omega t_{k+1} + \theta_0) \right)\Big|_{t_{k+1}}^{\parallel} = \\ = \left(v_k - g(t_{k+1} - t_k) - A\omega\cos(\omega t_{k+1} + \theta_0)\right)\Big|_{t_{k+1}} = -g + A\omega^2 \sin(\omega t_{k+1} + \theta_0) \equiv 0 \quad \textbf{(1.19)}$$

sprendinius, galime $f(t_{k+1})$ kritinius taškus suskaidyti į intervalus. Pasinaudojus bet kuriuo šaknų radimo algoritmu šiems intervalams, randame

$$f'(t_{k+1}) = \left(A\sin(\omega t_k + \theta_0) + v_k(t_{k+1} - t_k) - \frac{g}{2}(t_{k+1} - t_k)^2 - A\sin(\omega t_{k+1} + \theta_0) \right)\Big|_{t_{k+1}}^{|} = v_k - \\ - g(t_{k+1} - t_k) - A\omega\cos(\omega t_{k+1} + \theta_0) \equiv 0 \quad \textbf{(1.20)}$$

šaknis, tokiu būdu nustatydami funkcijos kritinius taškus bei $f(t_{k+1})$ suskaidydami į intervalus. Šaknys būtų tokios:

$$t_{k+1(1)} = \frac{\arcsin\left(\frac{g}{A\omega^2}\right) + 2\pi n - \theta_0}{\omega},\ t_{k+1(2)} = \frac{\pi - \arcsin\left(\frac{g}{A\omega^2}\right) + 2\pi n - \theta_0}{\omega},\ n \in Z. \quad \textbf{(1.21)}$$

Tuomet ieškant minėtuose intervaluose galime surasti bet kurią norimą $f(t_{k+1})$ šaknį. Šio metodo algoritmas toks:

1. Naudojantis (1.21), randame visas $f''(t_{k+1}) = 0$ šaknis, esančias tarp t_k bei tam tikro $t_{\max}$. Rastieji sprendiniai yra $f(t_{k+1})$ vingio taškai.

2. Patikriname funkcijos $f'(t_{k+1})$ ženklą gretimuose perlinkio taškuose. Jei funkcija $f'(t_{k+1})$ keičia ženklą, tai tarp tų vingio taškų yra kritinis taškas. Randame jį pagal (1.20). Pažymėkime šį tašką t_{krit}^1.

3. Patikriname, ar funkcijos $f(t_{k+1})$ ženklai taškuose t_k ir t_{krit}^1 skirtingi. Jei taip, surandam funkcijos $f(t_{k+1})$ šaknį intervale $[t_k, t_{krit}^1]$. Tai bus sekančio susidūrimo momentas. Jei intervale $[t_k, t_{krit}^1]$ funkcija ženklo nekeičia, patikrinti, ar $f(t_{k+1})$ keičia ženklą intervale tarp pirmojo ir antrojo, antrojo ir trečiojo, trečiojo ir ketvirtojo kritinių taškų ir t.t. Ieškome tol, kol randame t_{k+1}.

1.8.4. NIUTONO (LIESTINIŲ) METODAS [17]

Sprendžiame tokį uždavinį: duota lygtis $f(\mathrm{x}) = 0$, šaknies izoliacijos intervalas $[a,b]$, šaknies apskaičiavimo tikslumas ε. Reikia rasti lygties $f(\mathrm{x}) = 0$ šaknį $s \in [a,b]$ tikslumu ε.

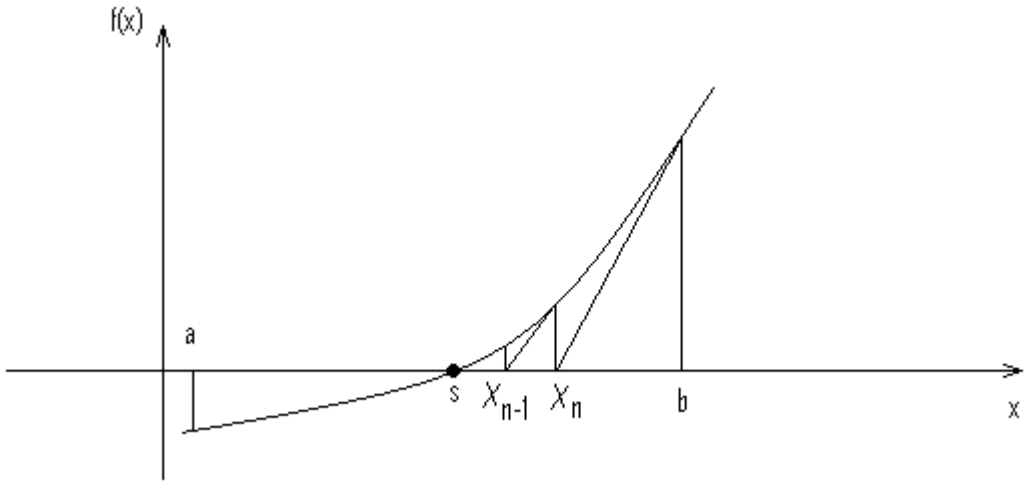
Funkcijos $\mathrm{y} = f(\mathrm{x})$, grafiko liestinės, einančios per tašką $(x_n, f(x_n))$, lygtis yra

$$y = f(x_n) + f'(x_n)(x - x_n).$$

Tašką, kuriame ši liestinė kerta Ox ašį, pažymėkime x_{n+1}:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \qquad \textbf{(1.22)}$$

Pasirinkus pradinį artinį x_0, pagal (1.22) formulę generuojama Niutono (liestinių) metodo iteracinė seka $\{x_n\}$. Geometrinė artinio x_{n+1} prasmė – funkcijos $\mathrm{y} = f(\mathrm{x})$ grafiko liestinės, einančios per tašką $(x_n, f(x_n))$, susikirtimo su *Ox* ašimi taškas.



1.4 pav. Niutono (liestinių) metodas

Algoritmo aprašymas:

1. Pradinio artinio pasirinkimas. Pasirenkame pradinį artinį $x_0 \in [a,b]$. Iteracijos numeris $n = 0$.

2. Naujo artinio apskaičiavimas: $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$.

3. Tikslumo sąlygos tikrinimas. Jei $|x_{n+1} - x_n| < \varepsilon$, tai $s = x_{n+1}$ yra apytikslis lygties sprendinys, apskaičiuotas norimu tikslumu ε, ir skaičiavimai baigiami; jei ne - vykdome naują iteraciją: $n = n+1$, grįžtame į 2 žingsnį.

Algoritmą galima taikyti tik tada, kada $f'(x)$ yra baigtinė ir nelygi nuliui. Jei Niutono metodas konverguoja, tai galioja įvertis $|x_{n+1} - x_n| < |x_n - x_{n-1}|$. Niutono metodo privalumas yra tas, jog jis greitai konverguoja.

1.8.5. PUSIAUKIRTOS METODAS [17]

Literatūroje [9], [24] pagrindinei lygčiai (1.5) spręsti naudojamas pusiaukirtos metodas. Šis metodas taikomas netiesinės lygties $f(x) = 0$ sprendimui, kai yra žinomas šaknų izoliacijos intervalas $[a, b]$, t.y. intervalas, kuriame $f(x)$ yra tolydi ir egzistuoja unikalus sprendinys, bei $f(a)$ ir $f(b)$ turi priešingus ženklus.

Algoritmas:

1. inicializuojame iteracijų skaitiklį $k = 0$;

2. tegu $m = \dfrac{1}{2}(a+b)$, jei $f(m) \approx 0$ ar $\dfrac{1}{2}(b-a) \approx 0$, stabdome iteracijas;

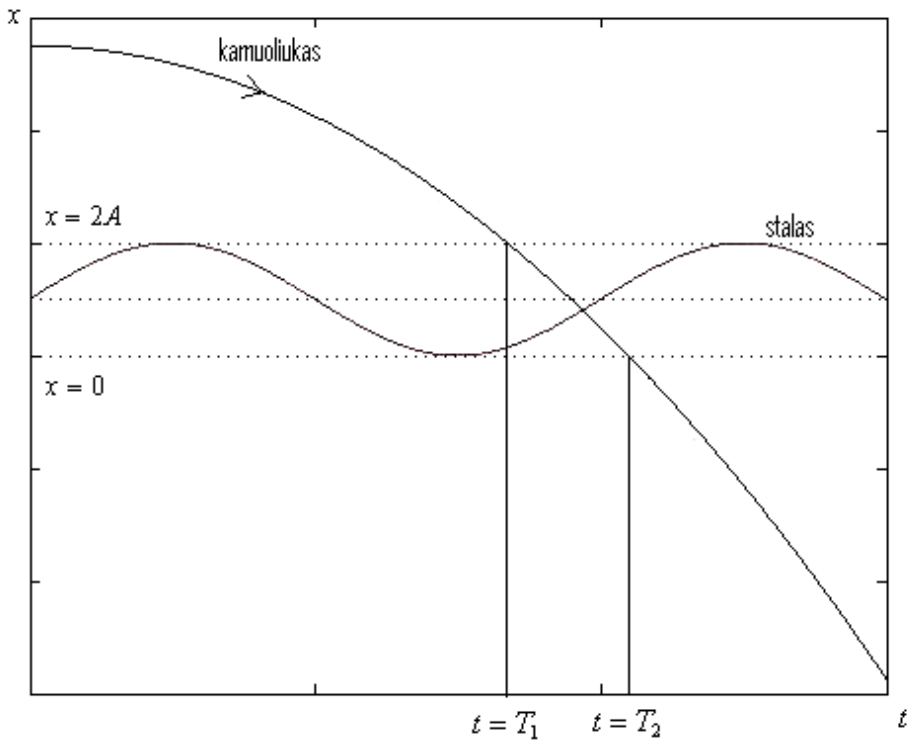
3. jei $f(a)\,f(m) > 0$, tada $a \leftarrow m$; kitu atveju, $b \leftarrow m$. Grįžti į 2 punktą.

Jei sprendinys egzistuoja intervale $[a, b]$, tada didžiausias atstumas nuo vidurinio taško $\dfrac{1}{2}(a+b)$ (apytikslio sprendinio) iki tikrojo sprendinio yra lygus pusei intervalo pločio – $\dfrac{1}{2}(b-a)$, o tai naudojama kaip paklaidos matas. Todėl, po kiekvienos iteracijos viršutinė paklaidos riba sumažėja per pusę.

Pusiaukirtos metodas mus ves prie sprendinio tik tada, jei sprendinys egzistuoja kažkokiame žinomame intervale.

Šiame darbe atliekant šokinėjančio kamuoliuko simuliaciją, buvo naudotas kiek modifikuotas pusiaukirtos metodas, padedantis nesunkiai nustatyti izoliacijos intervalą, kuriame garantuotai yra vienintelė lygties (1.5) (arba (1.17a) lygties, jei nagrinėjame bedimensius dydžius) šaknis. Kaip matyti iš

1.5 paveikslo, laisvai skriejantis kamuoliukas prieš susidurdamas su stalu kerta tiesę $x=2A$, o nesant susidūrimo, kamuoliuko trajektorija kerta tiesę $x=0$. Laiko momentai, atitinkantys lygčių $x_0+v_0(t-t_k)-\frac{g}{2}(t-t_0)^2=2A$ ir $x_0+v_0(t-t_k)-\frac{g}{2}(t-t_0)^2=0$ sprendinius, žymi pradinį izoliacijos intervalą $[T_1;T_2]$, kurį vėliau siauriname naudodami atkarpos dalijimo pusiau metodą.



1.5 pav. Modifikuoto pusiaukirtos metodo šaknies izoliacijos intervalo nustatymas

1.5 pav. punktyrinės linijos žymi aukščiausią $(x=2A)$, pusiausvirą $(x=A)$ ir žemiausią $(x=0)$ stalo padėtį skaičiuojant nuo viršaus į apačią. Juodos tiesės žymi pradinį izoliacijos intervalą $[T_1;T_2]$.

1.9. ŠOKINĖJANČIO KAMUOLIUKO UŽDAVINIO SPRENDINIŲ KLASIFIKACIJA

Šokinėjančio kamuoliuko uždavinio asimptotiniai sprendiniai skirstomi į:

- Periodinius sprendinius:
  - viengubo periodo orbitas;
  - dvigubo periodo orbitas;
  - ...
  - *N*-gubo periodo orbitas;

- Chaotines orbitas;
- Prilipusius sprendinius.

1.9.1. PERIODINIAI SPRENDINIAI [15]

Parodysime, jog egzistuoja stabilios periodinio kamuoliuko judėjimo būsenos. Pati paprasčiausia periodinio judėjimo būsena- vieno periodo, kuomet kiekvieno smūgio į stalą metu kamuoliuko fazė išlieka tokia pati. Šiuo atveju stalas kamuoliukui suteikia tiek pat kinetinės energijos, kiek jos praranda kamuoliukas kiekvieno smūgio metu. Jeigu apjungtume (1.3) ir (1.8) bei pažymėtume $\phi_k = \omega t_k + \theta_0$, gautume:

$$v_k = \alpha(A\omega\cos(\phi_k) - v_k^*) + A\omega\cos(\phi_k), \qquad \textbf{(1.23)}$$

kur mus domintų tiktai smūgio fazė, bet ne laiko momentas. Kadangi nagrinėjame vieno periodo judėjimą, kamuoliukas trenksis į stalą tokioje pat padėtyje, kokioje buvo atšokęs. Todėl galime užrašyti $v_k = -v_k^*$. Dėl to (1.23) virsta

$$v_k = \frac{1+\alpha}{1-\alpha} \cdot A\omega\cos(\phi_k). \qquad \textbf{(1.24)}$$

Reikalausime, jog kamuoliuko skrydžio laikas lygus bet kuriam sveikajam stalo judėjimo periodo skaičiui, vadinasi, $\Delta t_k = \frac{n}{f} = \frac{2\pi n}{\omega}$ ir $\Delta y = 0$. Iš (1.9) gauname, jog skrydžio laiko priklausomybė nuo v_k randama pagal:

$$\Delta t_k = \frac{2v_k}{g} \qquad \textbf{(1.25)}$$

Žinodami šį sąryšį, galime apibrėžti smūgio fazę, kuriai kamuoliuko judėjimas yra periodiškai stabilus:

$$\cos\phi^* = \frac{1-\alpha}{1+\alpha} \cdot \frac{\pi g n}{\omega^2 A}. \qquad \textbf{(1.26)}$$

Ši lygtis turi du sprendinius ϕ^*_1 bei ϕ^*_2, kurie abu yra fiksuoti taškai. Tačiau vienas jų stabilus, o kitas- nestabilus. Visiškai tampriai šokinėjančio kamuoliuko atveju $(\alpha = 1)$, nestabili (hiperbolinis

fiksuotas taškas) orbita paprastai atsiranda esant $\phi^*_1 = -\frac{\pi}{2}$, o tuo tarpu stabili orbita (elipsinis fiksuotas taškas) atsiranda esant $\phi^*_2 = \frac{\pi}{2}$. [22]

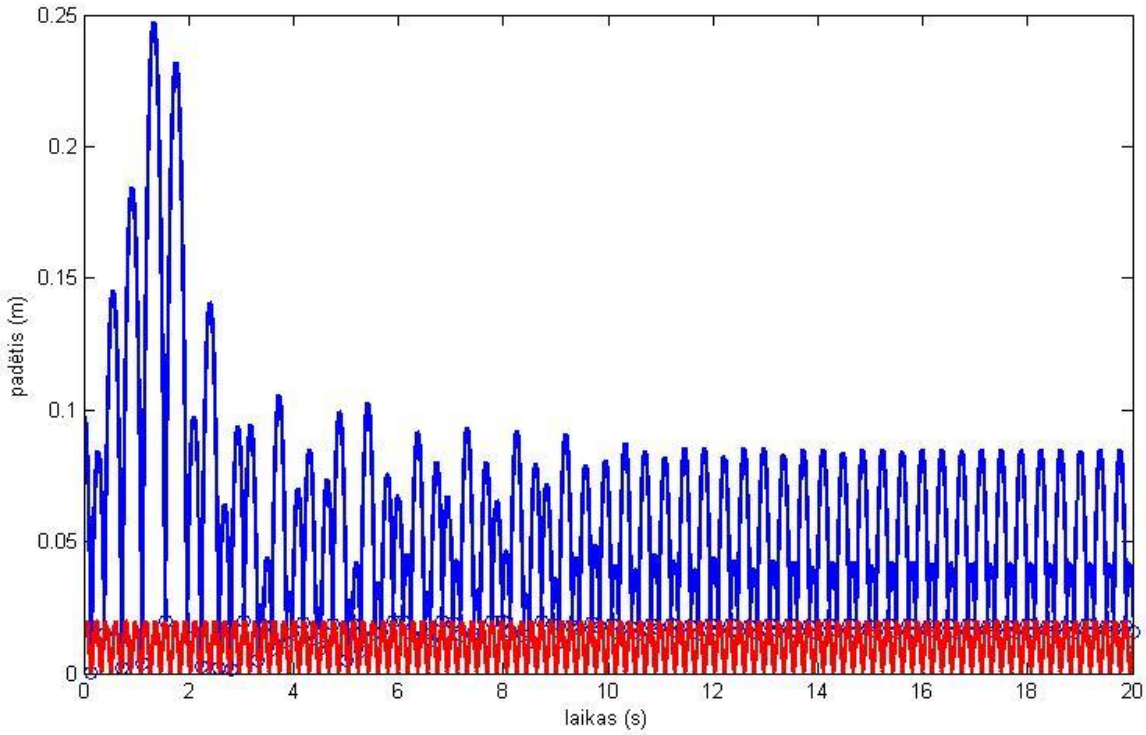
(1.26) taip pat parodo, jog egzistuoja tam tikras parametrų ω, A reikšmių intervalas, kuriame negalima vieno periodo būsena. Taip yra tada, kai $1 < \frac{1-\alpha}{1+\alpha} \cdot \frac{\pi g}{\omega^2 A}$.

Priartėjus prie šios sąlygos, du fiksuoti taškai susilieja ir išnyksta. Pavyzdžiui, jeigu keistume tik amplitudę, galime įvesti kritinės amplitudės dydį, apibrėžiamą:

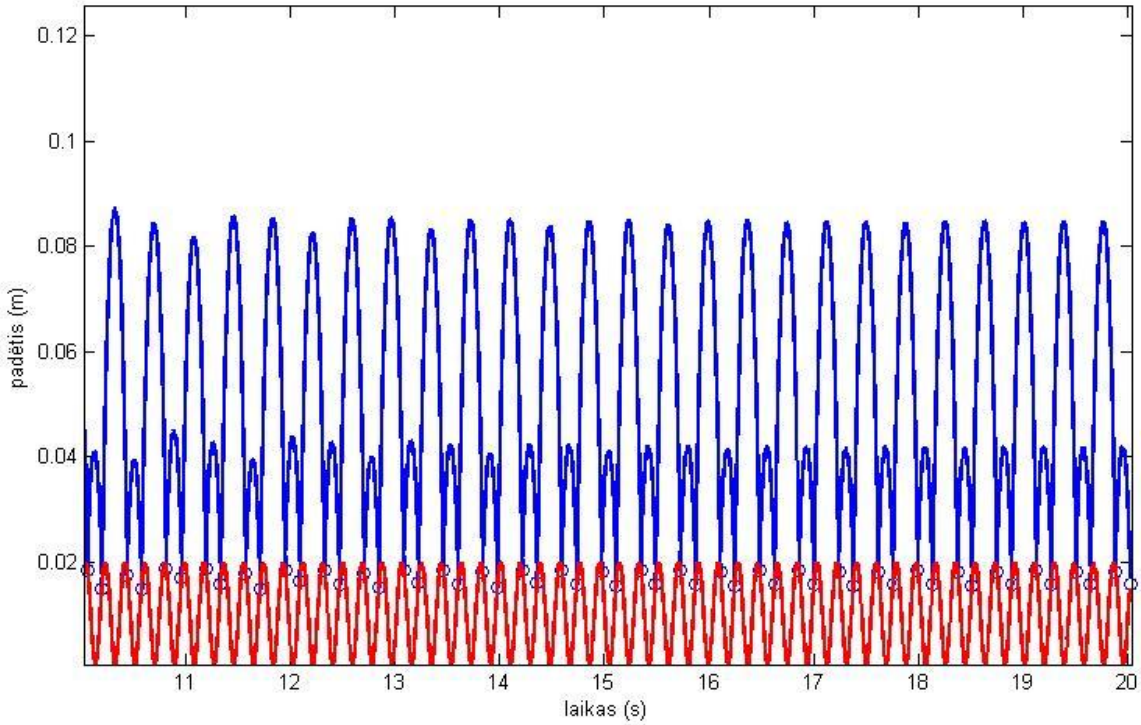
$$A_{krit} = \frac{1-\alpha}{1+\alpha} \cdot \frac{\pi g}{\omega^2}. \qquad \textbf{(1.27)}$$

Yra daug galimų periodinių sprendinių. Pavyzdžiui, kamuoliukas gali atšokti į fiksuotą aukštį visuose *n* stalo judėjimo cikluose. Jis taip pat gali atšokti į m skirtingų aukščių $n \times m$ stalo judėjimo cikluose prieš pradėdamas naują ciklą ir t.t. [23]

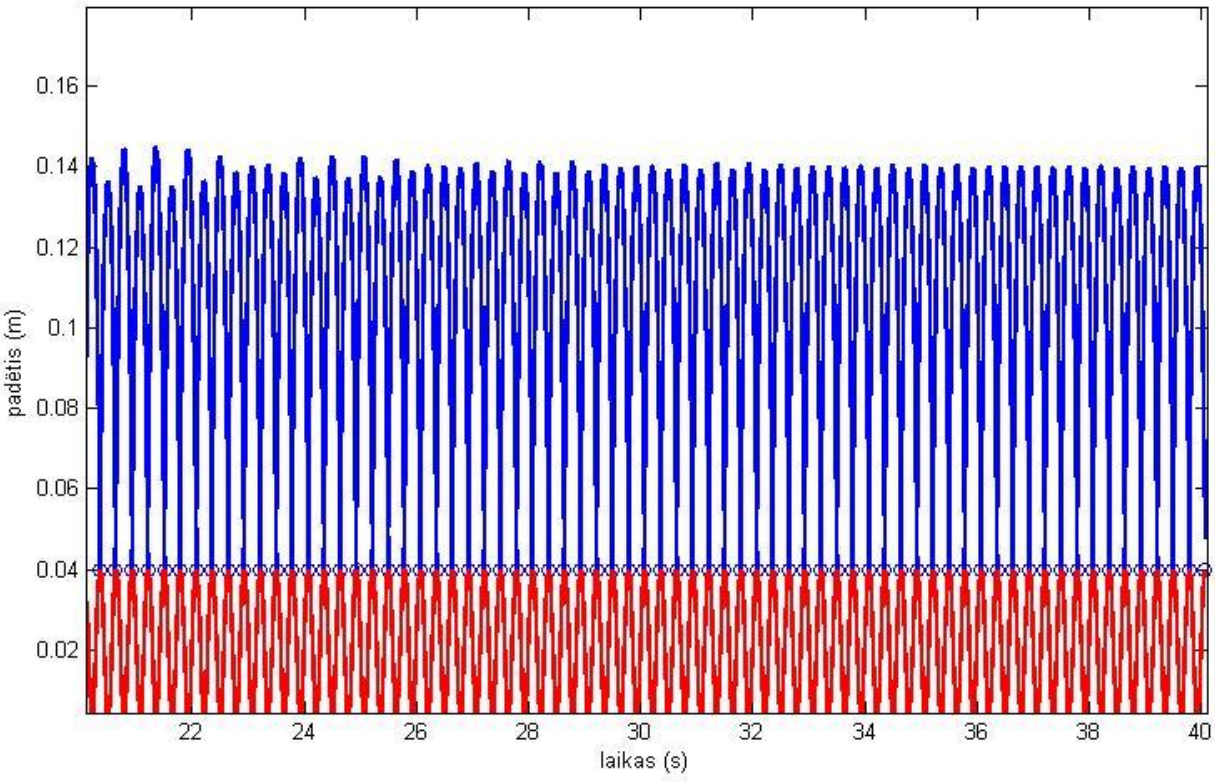
1.5 pav. pavaizduota dvigubo periodo orbita, 1.6 pav. pateikiamas padidintas dvigubo periodo orbitos vaizdas. Palyginimui, 1.7 pav. pateikiama viengubo periodo orbitos iliustracija.



1.6 pav. Dvigubo periodo orbita



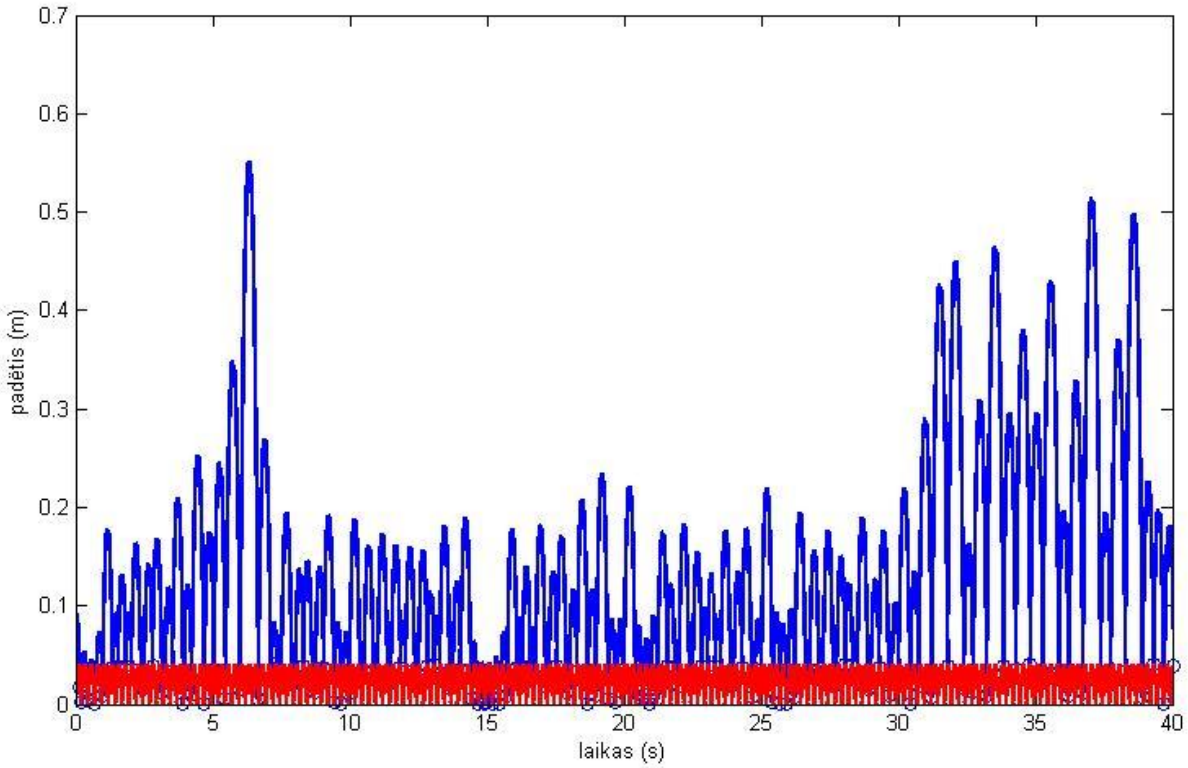
1.7 pav. Padidintas dvigubo periodo orbitos vaizdas



1.8 pav. Viengubo periodo orbita

1.9.2. CHAOTINIAI SPRENDINIAI

Orbitos, neturinčios jokių periodiškumo požymių, vadinamos chaotinėmis. [24] Chaotinės orbitos yra neprognozuojamo pobūdžio, taigi, negalime prognozuoti kamuoliuko padėties praėjus tam tikram laiko tarpui. 1.8 paveikslėlyje iliustruotas chaotinis kamuoliuko judėjimas.

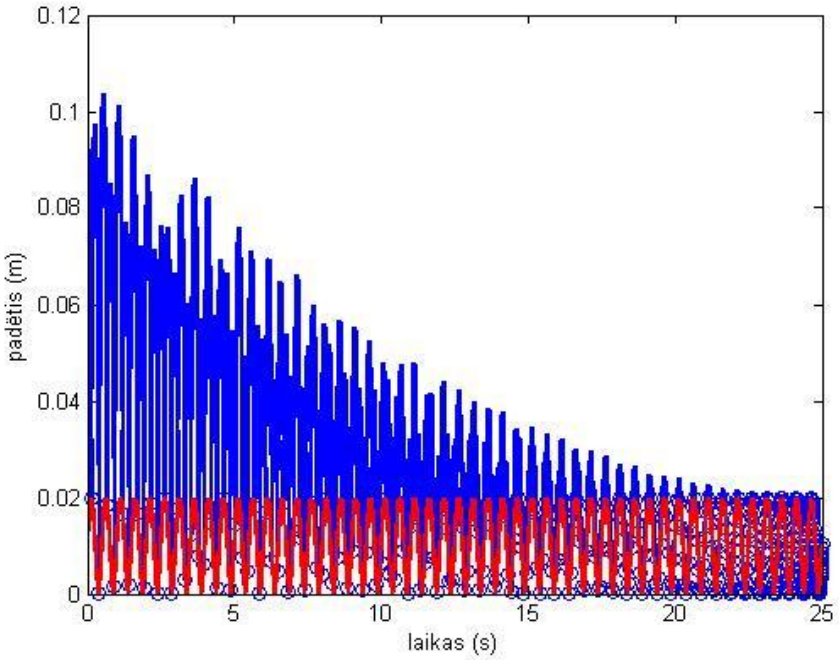


1.9 pav. Chaotinis kamuoliuko judėjimas

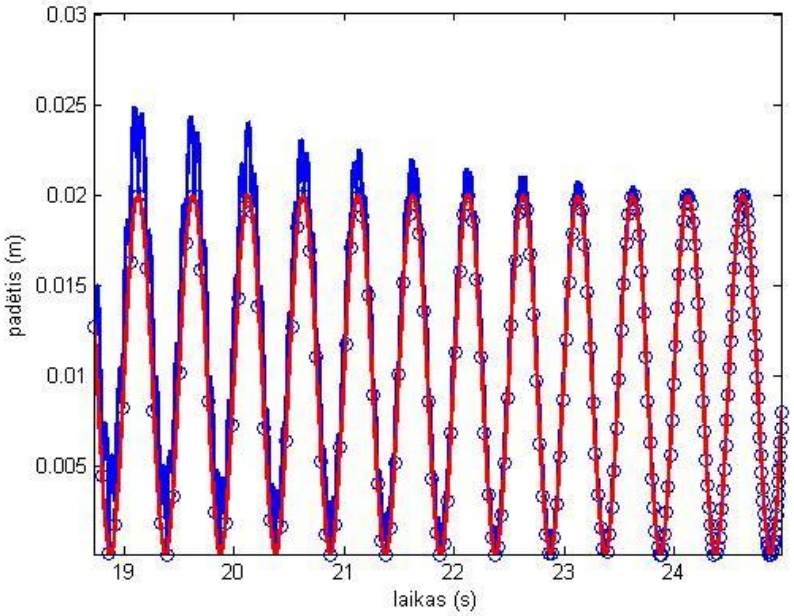
1.9.3. PRILIPĘ SPRENDINIAI

Jei stalo pagreitis, nukreiptas žemyn, yra mažesnis negu sunkio jėga, o artėjančio prie stalo kamuoliuko reliatyvus greitis yra pakankamai mažas, galima stebėti mažėjančios amplitudės kamuoliuko šuoliukus, kol galiausiai kamuoliukas galutinai prilips prie stalo. Įvykus prilipimui, galimi du atvejai- kamuoliukas prilips prie stalo visam laikui (kai $\frac{A\omega^2}{g} \le 1$), arba vėl atšoks nuo stalo taške $\phi = \frac{\arcsin\left(\frac{g}{A\omega^2}\right)}{2\pi}$, kur $\phi = \frac{\omega t + \theta_0}{2\pi} \bmod 1$ su nuliniu santykiniu greičiu (kai $\frac{A\omega^2}{g} > 1$). [24]

1.9 paveikslėlyje pateikiama kamuoliuko, prieš prilimpant prie stalo, judėjimo trajektorijos iliustracija. 1.10 paveikslėlyje pateikiama prilipusio kamuoliuko iliustracija.



1.10 pav. Kamuoliuko, prieš prilimpant prie stalo, judėjimo trajektorija



1.11 pav. Kamuoliukas, prilipęs prie stalo

2. TIRIAMOJI DALIS

2.1. PIRMASIS PAVYZDYS

Nagrinėkime tokį atvejį: kamuoliukas paleidžiamas iš 0,1 *m* aukščio virš stalo, stalo kampinis dažnis $2\pi \ \frac{rad}{s}$, pradinė fazė 0 *rad*. Pradinis kamuoliuko greitis $0\frac{m}{s}$, restitucijos parametro vertė 0,99, laisvojo kritimo pagreitis lygus $9{,}81\frac{m}{s^2}$.

Palyginsime rezultatus, gautus:

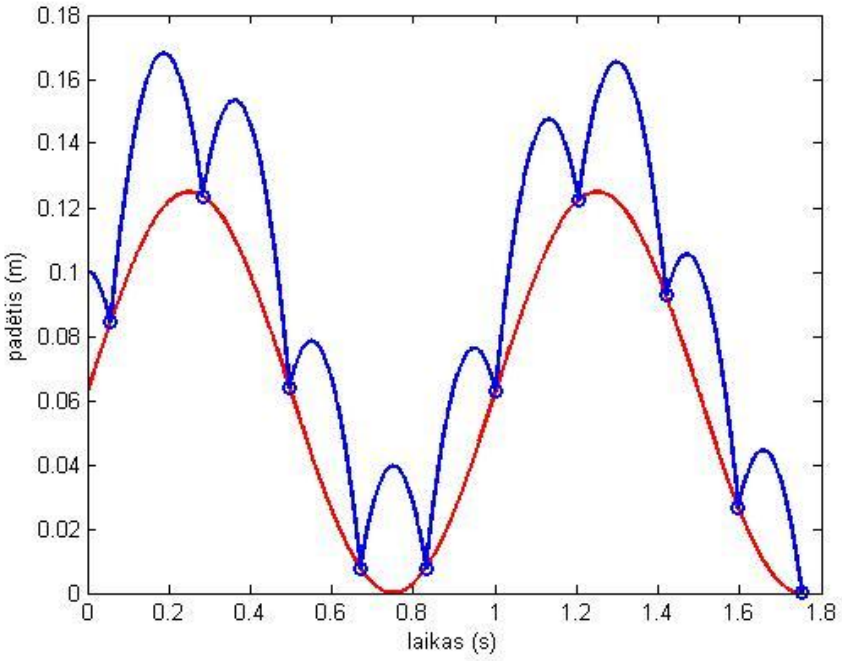
1. Atliekant šokinėjančio kamuoliuko simuliaciją;

2. Naudojant kitą iteracinį šio uždavinio sprendimo būdą. Netiesinei lygčiai spręsti naudosime Niutono metodą. Šaknų izoliacijos intervalą nustatysime grafiniu būdu.

Metodus palyginsime remiantis jų skaičiavimų trukme.

1. Simuliacijai buvo sukurta programa MATLAB aplinkoje (žr. 1 priedą), imituojanti šokinėjančio kamuoliuko judėjimą.

Atliekant šokinėjančio kamuoliuko simuliaciją, gauti rezultatai pateikti 2 priede. Sistemos dinamika pateikiama 2.1 paveikslėlyje. Stalo judėjimo amplitudė



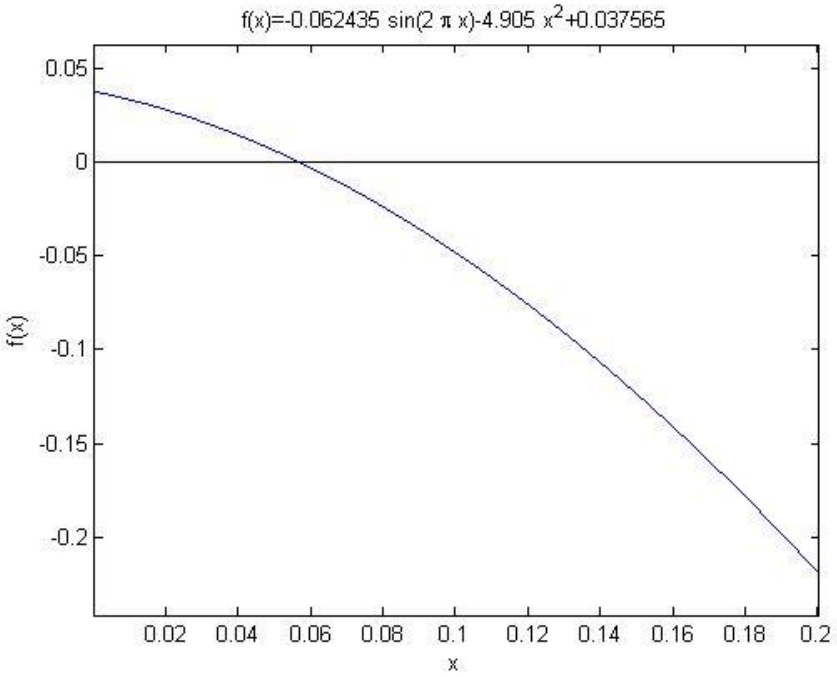
2.1 pav. Sistemos dinamika pirmojo pavyzdžio atveju

Atliekant simuliaciją, skaičiavimai truko 0,0234 *s*. Stalo judėjimo amplitudė, randama iš (1.15), kur $\beta = 1,$ lygi 0,0624 *m*.

2. Surasime pirmojo smūgio laiko momentą, kamuoliuko padėtį smūgio metu bei greitį. Įstatę pradines sąlygas į (1.5), gauname:

$$-0,06\sin(2\pi t_1) - 4,91t_1^2 + 0,04 = 0. \qquad \textbf{(2.1)}$$

(2.1) lygtis yra netiesinė. Jai spręsti naudosime Niutono metodą. Visų pirma reikia nustatyti lygties šaknies izoliacijos intervalą.



2.2 pav. Lygties (2.1) grafinis sprendimas

Šaknies izoliacijos intervalas $[0; 0,1]$. Pritaikome Niutono metodą, parinkę didžiausią iteracijų skaičių, lygų 25, šaknies apskaičiavimo minimalų tikslumą, lygų 10^{-5}. Pradinį artinį parenkame tokį: $t_0 = 0,1$. MATLAB terpėje buvo sukurta programa, realizuojanti šį metodą (2.1) lygties šakniai rasti (žr. 3 priedą). Gauti rezultatai pateikiami 2.1 lentelėje.

2.1 lentelė

Niutono (liestinių) metodo rezultatai pirmojo pavyzdžio pirmojo smūgio atveju

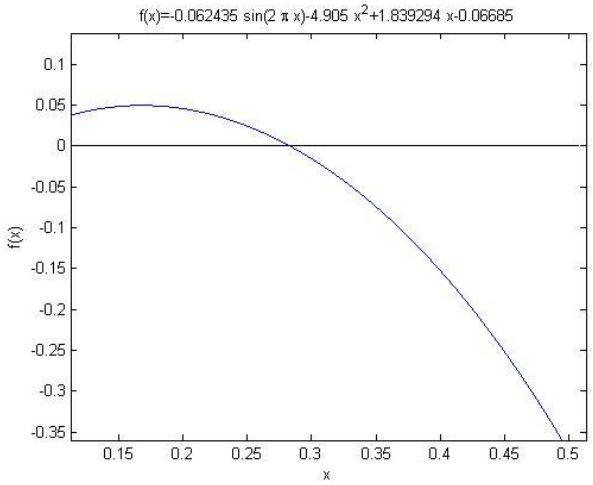
Iteracija	t_1, s
0	0,1000
1	0,0577
2	0,0565
3	0,0568
4	0,0567
5	0,0567
6	0,0567

Reikėjo 6 iteracijų. Paklaida lygi $4{,}6 \cdot 10^{-6} s$. Gavome, jog pirmojo smūgio laiko momentas $t_1 = 0{,}0567$ s. Pagal (1.1), randame kamuoliuko padėtį smūgio metu: $x(t_1) = 0{,}0842$ *m*, pagal (1.8) randame kamuoliuko greitį: $v_1 = (1 + 0{,}99)u_1 - 0{,}99 \cdot v_1^* = 1{,}2825 \frac{m}{s}$. Skaičiavimai truko 0,0021 *s*.

Turint pirmojo smūgio duomenis, galime surasti antrojo smūgio laiko momentą, kamuoliuko padėtį smūgio metu bei greitį. Įstatę turimus duomenis į (1.5), gauname:

$$-0{,}07 + 1{,}84t_2 - 4{,}91t_2^2 - 0{,}06\sin(2\pi t_2) = 0. \qquad \textbf{(2.2)}$$

Sprendžiant (2.2) Niutono metodu, pirmiausia nustatome lygties šaknies izoliacijos intervalą.



2.3 pav. Lygties (2.2) grafinis sprendimas

Šaknies t_2 izoliacijos intervalas $[0{,}25; 0{,}35]$. Pritaikome Niutono metodą, parinkę didžiausią iteracijų skaičių, lygų 25, šaknies apskaičiavimo minimalų tikslumą, lygų 10^{-5}. Pradinį artinį parenkame tokį: $t_0 = 0{,}35$. Šiek tiek modifikavus jau sukurtą programą MATLAB terpėje, randamos (2.2) lygties šaknys. Gauti rezultatai pateikiami 2.2 lentelėje.

2.2 lentelė

Niutono (liestinių) metodo rezultatai pirmojo pavyzdžio antrojo smūgio atveju

Iteracija	t_2, s
0	0,3500
1	0,2996
2	0,2847
3	0,2828
4	0,2827
5	0,2827

Prireikė 5 iteracijų. Paklaida lygi $4{,}76 \cdot 10^{-6} s$. Gavome, jog antrojo smūgio laiko momentas $t_2 = 0{,}2082$ s. Pagal (1.1), randame kamuoliuko padėtį smūgio metu: $x(t_2) = 0{,}1236$ *m*, pagal (1.8) randame kamuoliuko greitį: $v_2 = (1 + 0{,}99)u_2 - 0{,}99 \cdot v_2^* = 0{,}7657 \frac{m}{s}$. Skaičiavimai truko 0,0020 *s*.

Analogiškai, turint antrojo smūgio duomenis, galime surasti trečiojo smūgio laiko momentą, kamuoliuko padėtį smūgio metu bei greitį. Turint trečiojo smūgio duomenis, galima surasti ketvirtojo laiko momentą, kamuoliuko padėtį smūgio metu bei greitį ir t.t.

Apibendrinti rezultatai pateikiami 2.3 lentelėje.

2.3 lentelė

Apibendrinti Niutono (liestinių) metodo rezultatai pirmojo pavyzdžio atveju

k	*k*-tojo smūgio laiko momentas t_k, s	paklaida ε, s	Kamuoliuko padėtis *k*-tojo smūgio momentu $x(t_k)$, m	Kamuoliuko greitis v_k, $\frac{m}{s}$	Skaičiavimų trukmė, *s*
1	0,0567	$4{,}6 \cdot 10^{-6}$	0,0842	1,2825	0,0021

2	0,2827	$4{,}76 \cdot 10^{-6}$	0,1236	0,7657	0,0020
3	0,4957	$2{,}61 \cdot 10^{-6}$	0,0641	0,5302	0,0021
4	0,6698	$3{,}61 \cdot 10^{-6}$	0,0078	0,7888	0,0020
5	0,8305	$6{,}22 \cdot 10^{-6}$	0,0079	1,1586	0,0021
6	1,0010	$3{,}41 \cdot 10^{-6}$	0,0628	1,2888	0,0022
7	1,2040	$1{,}16 \cdot 10^{-6}$	0,1223	0,9185	0,0020
8	1,4193	$6{,}2 \cdot 10^{-6}$	0,0928	0,4987	0,0020
9	1,5968	$8{,}05 \cdot 10^{-6}$	0,0268	0,5891	0,0021
10	1,7528	$5{,}85 \cdot 10^{-7}$	$9{,}66 \cdot 10^{-6}$	0,9355	0,0021

Bendra skaičiavimų trukmė 0,0207 *s* (Niutono metodo) yra šiek tiek mažesnė negu simuliacijos (0,0234 *s*).

2.4 lentelė

Skirtumo tarp metodų rezultatų lentelė pirmojo pavyzdžio atveju

k	*k*-tojo smūgio laiko momento t_k skirtumas, *s*	Kamuoliuko padėties *k*-tojo smūgio momentu $x(t_k)$ skirtumas, *m*	Kamuoliuko greičio v_k skirtumas, $\frac{m}{s}$
1	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$
2	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$
3	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$
4	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
5	$<1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$
6	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$
7	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$
8	$1 \cdot 10^{-4}$	$<1 \cdot 10^{-4}$	$2 \cdot 10^{-4}$

9	$1\cdot10^{-4}$	$<1\cdot10^{-4}$	$5\cdot10^{-4}$
10	$9\cdot10^{-4}$	$<1\cdot10^{-4}$	$1{,}7\cdot10^{-3}$

2.2. ANTRASIS PAVYZDYS

Nagrinėkime tokį atvejį: kamuoliukas paleidžiamas iš 1,1 *m* aukščio virš stalo, kurio kampinis dažnis $2\pi\ \frac{rad}{s}$, pradinė fazė 0 *rad*. Pradinis kamuoliuko greitis $0\frac{m}{s}$, restitucijos parametro vertė 0,999, laisvojo kritimo pagreitis lygus $9{,}81\frac{m}{s^2}$.

Palyginsime rezultatus, gautus:

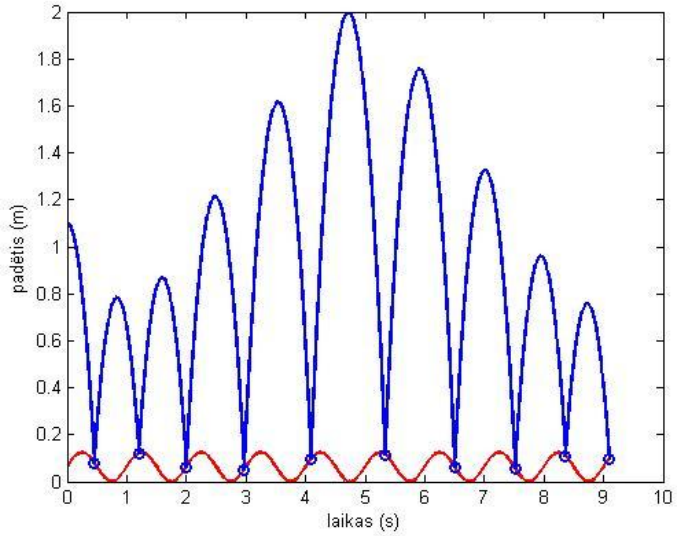
1. Atliekant šokinėjančio kamuoliuko simuliaciją;

2. Naudojant kitą iteracinį šio uždavinio sprendimo būdą. Netiesinei lygčiai spręsti naudosime Niutono metodą. Šaknų izoliacijos intervalą nustatysime grafiniu būdu.

3. Naudojant Aukšto šuolio aproksimaciją.

1. Simuliacijai buvo sukurta programa MATLAB aplinkoje (žr. 1 priedą), imituojanti šokinėjančio kamuoliuko judėjimą.

Atliekant šokinėjančio kamuoliuko simuliaciją, gauti rezultatai pateikti 4 priede. Sistemos dinamika pateikiama 2.4 paveikslėlyje.



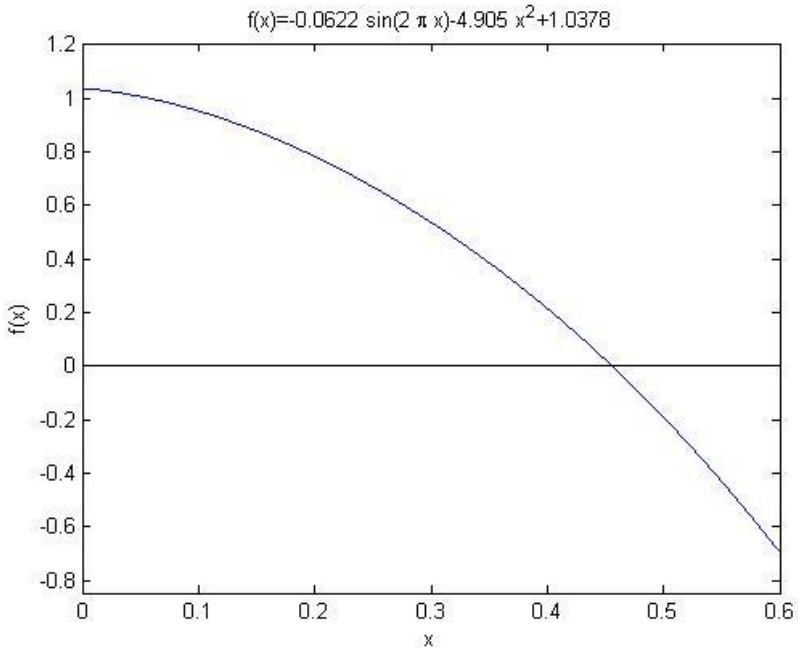
2.4 pav. Sistemos dinamika antrojo pavyzdžio atveju

Atliekant simuliaciją, skaičiavimai truko 0,0189 *s*. Stalo judėjimo amplitudė, randama iš (1.15), kur $\beta = 1,$ lygi 0,0622 *m*.

2. Surasime pirmojo smūgio laiko momentą, kamuoliuko padėtį smūgio metu bei greitį. Įstatę pradines sąlygas į (1.5), gauname:

$$-0{,}06\sin(2\pi t_1) - 4{,}91t_1^2 + 1{,}04 = 0. \qquad \textbf{(2.3)}$$

(2.3) lygtį sprendžiant Niutono metodu, visų pirma reikia nustatyti lygties šaknies izoliacijos intervalą.



2.5 pav. Lygties (2.3) grafinis sprendimas

Šaknies izoliacijos intervalas $[0{,}4;\, 0{,}6]$. Pritaikome Niutono metodą, parinkę didžiausią iteracijų skaičių, lygų 25, šaknies apskaičiavimo minimalų tikslumą, lygų 10^{-5}. Pradinį artinį parenkame tokį: $t_0 = 0{,}6$. MATLAB terpėje buvo sukurta programa, realizuojanti šį metodą (2.3) lygties šakniai rasti (žr. 3 priedą). Gauti rezultatai pateikiami 2.5 lentelėje.

2.5 lentelė

Niutono (liestinių) metodo rezultatai antrojo pavyzdžio pirmojo smūgio atveju

Iteracija	t_2, *s*
0	0,6000
1	0,4759

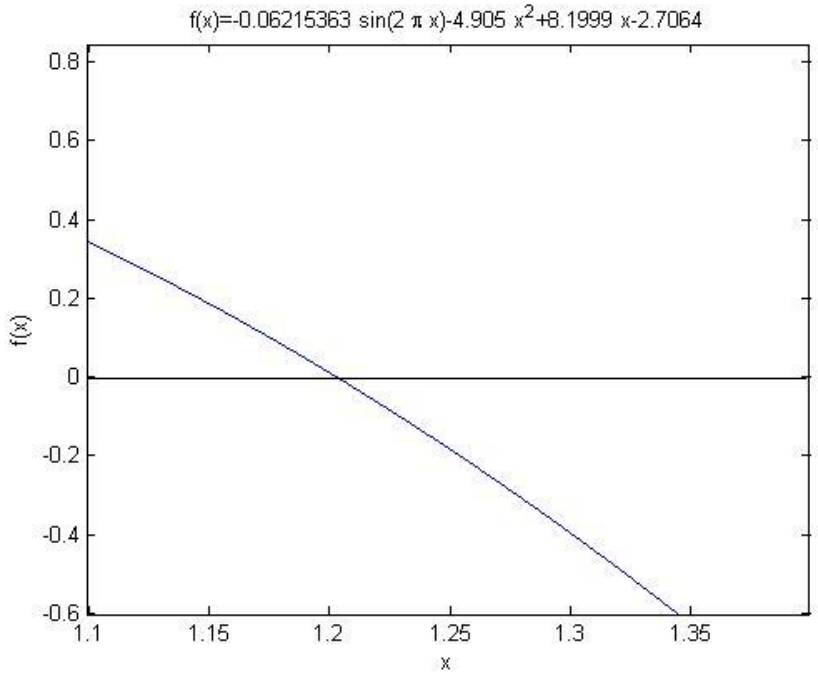
2	0,4567
3	0,4562
4	0,4562

Prireikė 4 iteracijų. Paklaida lygi $1{,}97 \cdot 10^{-7} s$. Gavome, jog pirmojo smūgio laiko momentas $t_1 = 0{,}4562$ s. Pagal (1.1), randame kamuoliuko padėtį smūgio metu: $x(t_1) = 0{,}0790$ *m*, pagal (1.8) randame kamuoliuko greitį: $v_1 = (1 + 0{,}999)u_1 - 0{,}999 \cdot v_1^* = 3{,}7198 \frac{m}{s}$. Skaičiavimai truko 0,0021 *s*.

Turint pirmojo smūgio duomenis, galime surasti antrojo smūgio laiko momentą, kamuoliuko padėtį smūgio metu bei greitį. Įstatę turimus duomenis į (1.5), gauname:

$$-2{,}70 + 8{,}20t_2 - 4{,}91t_2^2 - 0{,}06\sin(2\pi t_2) = 0. \qquad \textbf{(2.4)}$$

(2.4) lygtį sprendžiame Niutono metodu: pirmiausia nustatome lygties šaknies izoliacijos intervalą.



2.6 pav. Lygties (2.4) grafinis sprendimas

Šaknies t_2 izoliacijos intervalas $[1{,}15; 1{,}25]$. Pritaikome Niutono metodą, parinkę didžiausią iteracijų skaičių, lygų 25, šaknies apskaičiavimo minimalų tikslumą, lygų 10^{-5}. Pradinį artinį parenkame tokį: $t_0 = 1{,}25$. Šiek tiek modifikavus jau sukurtą programą MATLAB terpėje, randamos (2.4) lygties šaknys. Gauti rezultatai pateikiami 2.6 lentelėje.

2.6 lentelė

Niutono (liestinių) metodo rezultatai antrojo pavyzdžio antrojo smūgio atveju

Iteracija	t_2, s
0	1,2500
1	1,2050
2	1,2030
3	1,2030

Prireikė 3 iteracijų. Paklaida lygi $4{,}01\cdot 10^{-6}\,s$. Gavome, jog antrojo smūgio laiko momentas $t_2 = 1{,}2030$ s. Pagal (1.1), randame kamuoliuko padėtį smūgio metu: $x(t_2) = 0{,}1216\ m$, pagal (1.8) randame kamuoliuko greitį: $v_2 = (1+0{,}999)u_2 - 0{,}999\cdot v_2^* = 3{,}8296\frac{m}{s}$. Skaičiavimai truko 0,0021 *s*.

Analogiškai, turint antrojo smūgio duomenis, galime surasti trečiojo smūgio laiko momentą, kamuoliuko padėtį smūgio metu bei greitį. Turint trečiojo smūgio duomenis, galima surasti ketvirtojo laiko momentą, kamuoliuko padėtį smūgio metu bei greitį ir t.t.

Apibendrinti rezultatai pateikiami 2.7 lentelėje.

2.7 lentelė

Apibendrinti Niutono (liestinių) metodo rezultatai antrojo pavyzdžio atveju

k	*k*-tojo smūgio laiko momentas t_k, s	paklaida ε, s	Kamuoliuko padėtis *k*-tojo smūgio momentu $x(t_k)$, m	Kamuoliuko greitis v_k, $\frac{m}{s}$	Skaičiavimų trukmė, *s*
1	0,4562	$1{,}97\cdot 10^{-7}$	0,0790	3,7198	0,0021
2	1,2030	$4{,}01\cdot 10^{-6}$	0,1216	3, 8296	0,0021
3	1,9990	$7{,}01\cdot 10^{-6}$	0,0618	4,7565	0,0022
4	2,9710	$5{,}87\cdot 10^{-7}$	0,0509	5,5418	0,0020
5	4,0926	$1{,}15\cdot 10^{-9}$	0,0963	6,1076	0,0019
6	5,3346	$4{,}9\cdot 10^{-6}$	0,1157	5,6746	0,0019

7	6,5008	$4{,}17\cdot10^{-6}$	0,0618	4,9805	0,0020
8	7,5174	$1{,}39\cdot10^{-6}$	0,0554	4,2124	0,0019
9	8,3632	$2\cdot10^{-6}$	0,1093	3,5711	0,0019
10	9,0954	$8{,}3\cdot10^{-6}$	0,0972	4,2515	0,0019

Bendra skaičiavimų trukmė 0,0199 s

3. Naudojant Aukšto šuolio aproksimaciją, šokinėjančio kamuoliuko smūgio laikas randamas pagal (1.12). Skaičiuojant antrojo smūgio laiko momentą, pasinaudosime pirmojo smūgio duomenimis, gautais atliekant simuliaciją: laiko momentu t_1 bei greičiu v_1. Gauname: $t_2 = \frac{2v_1}{g} + t_1 = 1{,}2146$ (s).

Kadangi kamuoliuko padėtis smūgio metu sutampa su stalo padėtimi, antrojo smūgio padėtis randama pagal (1.2): $x(t_2) = 0{,}1228$ *m*, pagal (1.8) randame kamuoliuko greitį: $v_2 = (1+0{,}999)u_2 - 0{,}999\cdot v_2^* = 3{,}8886\frac{m}{s}$.

Turint antrojo smūgio duomenis, rasime trečiojo smūgio laiko momentą $t_3 = \frac{2v_2}{g} + t_2 = 2{,}0074$ (s).

Kamuoliuko padėtis trečiojo smūgio metu: $x(t_3) = 0{,}0650$ *m*, greitis

$v_3 = (1+0{,}999)u_3 - 0{,}999\cdot v_3^* = 4{,}6647\frac{m}{s}$.

Analogiškai randami sekančių smūgių laiko momentai, kamuoliuko greitis bei padėtis. 2.8 lentelėje pateikiami Aukšto šuolio aproksimacijos rezultatai.

2.8 lentelė

Aukšto šuolio aproksimacijos rezultatai

k	*k*-tojo smūgio laiko momentas t_k, *s*	Kamuoliuko padėtis *k*-tojo smūgio momentu $x(t_k)$, *m*	Kamuoliuko greitis v_k, $\frac{m}{s}$
1	0,4562	0,0790	3,7198
2	1,2146	0,1228	3,8886

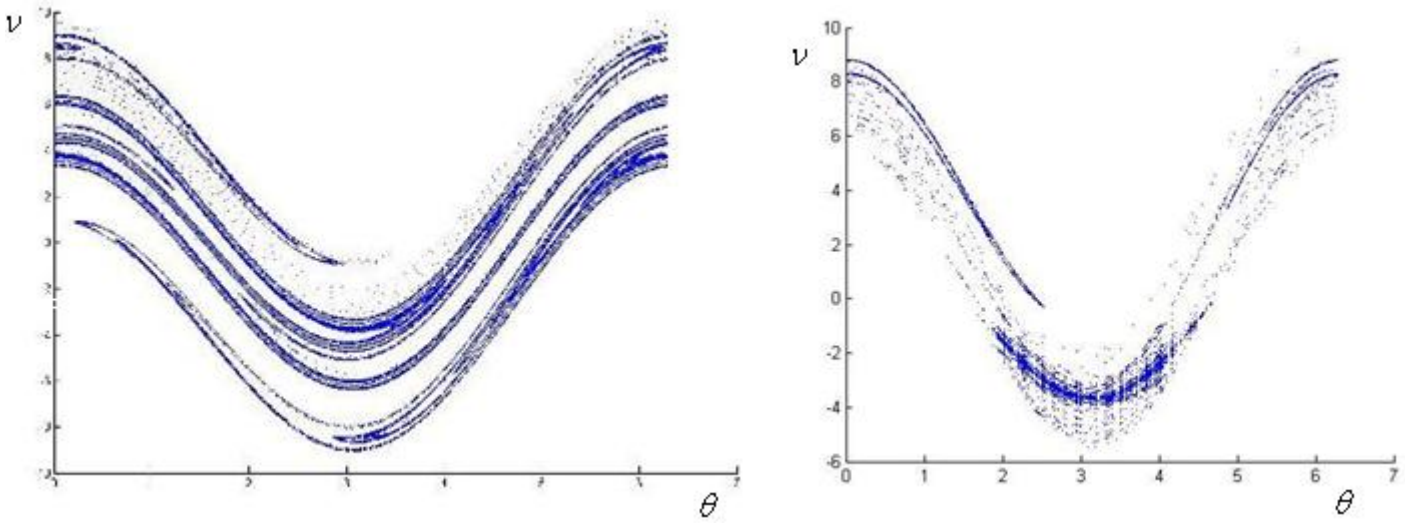
3	2,0074	0,0650	4,6647
4	2,9584	0,0461	5,4141
5	4,0622	0,0858	6,1306
6	5,3121	0,1196	5,8279
7	6,5002	0,0621	5,0409
8	7,5279	0,0513	4,2671
9	8,3979	0,0994	3,6379
10	9,1396	0,1099	4,1337

2.9 lentelė

Skirtumo tarp metodų rezultatų lentelė antrojo pavyzdžio atveju

k	**Niutono metodo lyginant su simuliacija**			**Aukšto šuolio aproksimacijos lyginant su simuliacija**		
	k-tojo smūgio laiko momento t_k skirtumas, s	Kamuoliuko padėties k-tojo smūgio momentu $x(t_k)$ skirtumas, m	Kamuoliuko greičio v_k skirtumas, $\frac{m}{s}$	k-tojo smūgio laiko momento t_k skirtumas, s	Kamuoliuko padėties k-tojo smūgio momentu $x(t_k)$ skirtumas, m	Kamuoliuko greičio v_k skirtumas, $\frac{m}{s}$
1	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	0	0	0
2	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$1\cdot10^{-4}$	$1{,}16\cdot10^{-2}$	$1{,}2\cdot10^{-3}$	$5{,}91\cdot10^{-2}$
3	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$8{,}4\cdot10^{-3}$	$3{,}2\cdot10^{-3}$	$9{,}18\cdot10^{-2}$
4	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$1{,}26\cdot10^{-2}$	$4{,}8\cdot10^{-3}$	$1{,}28\cdot10^{-1}$
5	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$3{,}04\cdot10^{-2}$	$1{,}05\cdot10^{-2}$	$2{,}3\cdot10^{-2}$
6	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$<1\cdot10^{-4}$	$2{,}25\cdot10^{-2}$	$3{,}9\cdot10^{-3}$	$1{,}53\cdot10^{-1}$
7	$1\cdot10^{-4}$	$<1\cdot10^{-4}$	$1\cdot10^{-4}$	$7\cdot10^{-4}$	$3\cdot10^{-4}$	$6{,}03\cdot10^{-2}$
8	$1{,}6\cdot10^{-4}$	$1\cdot10^{-4}$	$2\cdot10^{-4}$	$1{,}03\cdot10^{-2}$	$4\cdot10^{-3}$	$5{,}45\cdot10^{-2}$
9	$3\cdot10^{-4}$	$1\cdot10^{-4}$	$7\cdot10^{-4}$	$3{,}44\cdot10^{-2}$	$9{,}8\cdot10^{-3}$	$6{,}75\cdot10^{-2}$
10	$5{,}8\cdot10^{-4}$	$1{,}9\cdot10^{-4}$	$5{,}2\cdot10^{-3}$	$4{,}48\cdot10^{-2}$	$1{,}29\cdot10^{-2}$	$1{,}13\cdot10^{-1}$

2.3. CHAOTINIO ŽEMĖLAPIO TIKSLIOS SISTEMOS IR AUKŠTO ŠUOLIO APROKSIMACIJOS ATVEJAIS PALYGINIMAS

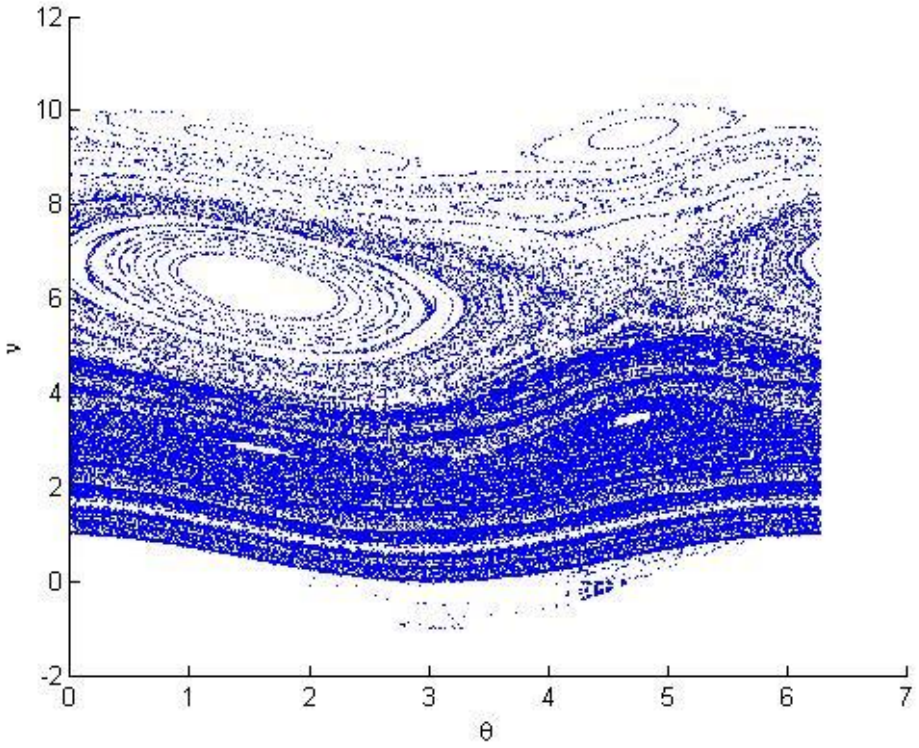


2.7 pav. Keistojo atraktoriaus iliustracija

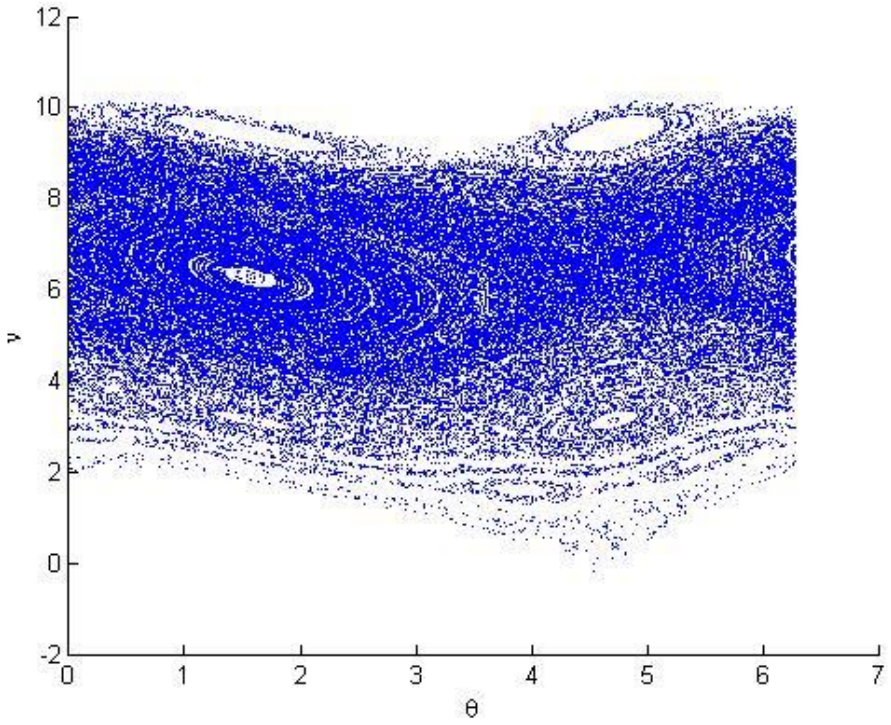
2.7 pav. pateikiamas keistojo atraktoriaus iliustracija Aukšto šuolio aproksimacijos (kairėje) bei tikslios sistemos (dešinėje) atvejais. Parametrai buvo parinkti tokie: restitucijos parametras $\alpha = 0{,}5$, bedimensis parametras $\beta = 5{,}5$, laisvojo kritimo pagreitis $g = 9{,}81\frac{\text{m}}{\text{s}^2}$, pradinė fazė $\theta_0 = 0$, kampinis dažnis $\omega = 2\pi$.

2.9 pav. pateikiamas Aukšto šuolio aproksimacijos standartinis žemėlapis bei 2.8 pav.- tikslios sistemos standartinis žemėlapis. Parametrai buvo parinkti tokie: restitucijos parametras $\alpha = 1$, bedimensis parametras $\beta = 1$, laisvojo kritimo pagreitis $g = 9{,}81\frac{\text{m}}{\text{s}^2}$, pradinė fazė $\theta_0 = 0$, kampinis dažnis $\omega = 2\pi$.

5 Priede pateikiamas standartinis žemėlapis abiem atvejais, keičiant parametro α reikšmes: $\alpha = 0{,}9$, $\alpha = 0{,}99$ bei $\alpha = 0{,}999$.



2.8 pav. Tikslios sistemos standartinis žemėlapis



2.9 pav. Aukšto šuolio aproksimacijos standartinis žemėlapis

3. PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

Šokinėjančio kamuoliuko simuliacijai, standartiniui žemėlapiui bei Niutono metodo realizavimui buvo sukurtos programos MATLAB aplinkoje. MATLAB (iš žodžių *MATrix LABoratory*) yra daugiaplatformė MathWorks programinė įranga, skirta įvairių mokslo šakų problemoms spręsti, ypač matematinėms. Tai didžiulis galingas paketas, turintis savitą lengvai perprantamą programavimo kalbą.

Nors simbolinėms matematinėms manipuliacijoms geriau tinka Maple ir Mathematica produktai, su MathCad pradedantis vartotojas gali greičiau gauti pirmuosius rezultatus, MATLAB kaip programavimo kalba turi didesnį lankstumą, kuris naudingas įvairiems fizikos, biologijos bei kitiems matematiniams modeliams kurti ir tirti.

3.1 lentelė

Sukurtos programos ir jų paskirtis

Programa	Paskirtis
pirmas_sm.m	funkcija, aprašanti kamuoliuko padėtį smūgio metu
niutono_metodas.m	Niutono metodu apskaičiuoti smūgio laiko momentą
zemelapis_supaprastintas_atvejis.m	nubraižyti chaotinį žemėlapį Aukšto šuolio aproksimacijos atveju
auksti_atsokimai_vaizdavimas.m	Aukšto šuolio aproksimacijos vaizdavimas
zemelapis_tikslus_atvejis.m	nubraižyti tikslios sistemos chaotinį žemėlapį
laisvas_skrydis.m	funkcija, aprašanti kamuoliuko judėjimą tarp smūgių
tikslus_modelis_vaizdavimas.m	tikslios sistemos simuliacija, rezultatų išvedimas
bisection.m	Šaknies radimas pusiaukirtos metodu

Pastaba. Sukurtose programose keičiant parametrų *x0* (pradinis kamuoliuko aukštis), *v0* (pradinis kamuoliuko greitis), *teta0* (pradinė (bedimensė) fazė), *omega* (kampinis dažnis (bedimensis)), *alfa* (restitucijos parametras), *beta* (bedimensis parametras) reikšmes, galima gauti skirtingas kamuoliuko judėjimo trajektorijas; keičiant *eps* (šaknies paieškos tikslumas), *N* (vaizduojamų susidūrimų skaičius) reikšmes, galima gauti norimo tikslumo *eps* norimo skaičiaus *N* kamuoliuko smūgių koordinates.

IŠVADOS

1. Buvo sukurta programa, vaizduojanti šokinėjančio kamuoliuko virš sinusoidiškai vibruojančio stalo judėjimo trajektoriją. Sprendžiant šokinėjančio kamuoliuko uždavinį, dėl paprastumo ir sprendinių pastovumo buvo pasinaudota modifikuotu pusiaukirtos algoritmu.
2. Gauta, jog šis algoritmas veikia gerai, kai turime besileidžiančio $(v_0 < 0)$ ir iš aukštai paleisto $(x_0 > 2A)$ kamuoliuko pradines sąlygas. Taigi, reikia atsižvelgti ir į sąlygas, kai turime iš žemai paleistą ir (arba) kylantį kamuoliuką.
3. Buvo nustatyta, jog iš žemai paleisto kamuoliuko atveju izoliacijos intervalo pradžią atitinka taškas $t = t_0$. Kylančio kamuoliuko atveju surandamas kamuoliuko aukščiausio pakilimo taškas, nuo kurio jis pradeda leistis ir apsiribojama besileidžiančio kamuoliuko nagrinėjimu.
4. Buvo nustatyta, jog tarp rezultatų, gautų atlikus šokinėjančio kamuoliuko simuliaciją bei rezultatų, gautų naudojant Niutono (liestinių) metodą, skirtumas nėra žymus, skirtumas dėl paklaidų kiek padidėja tik skaičiuojant paskutiniuosius smūgius. Labiausiai skyrėsi Aukšto šuolio aproksimacijos rezultatai lyginant su kitais metodais. Todėl ši aproksimacija taikytina tik išimtinais atvejais.
5. Palyginus metodus pagal skaičiavimo trukmę, nors pats greičiausias yra Aukšto šuolio aproksimacija, tačiau kokybiškus rezultatus davusio Niutono metodo skaičiavimų trukmė bendruoju atveju šiek tiek nusileidžia pusiaukirtos algoritmo skaičiavimų trukmei.
6. Šokinėjančio kamuoliuko uždavinio sprendimo rezultatai (kamuoliuko orbita) itin jautūs pradinių sąlygų pakeitimams; nedidelis stalo dažnio, pradinio kamuoliuko aukščio pakeitimas gali lemti visiškai skirtingo pobūdžio rezultatus.
7. Keistojo atraktoriaus egzistavimas įrodė kamuoliuko judėjimo chaotiškumą tam tikrais atvejais.
8. Palyginus standartinį žemėlapį, Aukšto šuolio aproksimacijos bei tikslios sistemos atvejais gauti tie patys fiksuoti taškai $\left(\frac{\pi}{2}, 2k\pi\right), \left(\frac{3\pi}{2}, 2k\pi\right),\ k \in Z.$

LITERATŪRA

1. Attractor / Wolfram MathWorld [interaktyvus] [žiūrėta 2013 05 11]. Prieiga per internetą: http://mathworld.wolfram.com/Attractor.html
2. Fzero / MathWorks documentation center [interaktyvus] [žiūrėta 2013 05 11]. Prieiga per internetą: http://www.mathworks.se/help/matlab/ref/fzero.html
3. Gray P. The completely inelastic bouncing ball. School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA, 2011.
4. Hansen L. U. W., Christensen M., Mosekilde E. Deterministic analysis of the probability machine. Journal Physica Scripta, 51, 35, 1995.
5. Holmes P. J. The dynamics of repeated impacts with a sinusoidally vibrating table. Journal of Sound and Vibration, 84(2), 1982, p. 173-189.
6. Kamath A. K., Singh N. M., Pasumarthy R. Dynamics and control of bouncing ball. World academy of science, engineering and technology, 41, 2008.
7. Lind C. Bouncing ball system. AMATH 575 final project, 2005, [interaktyvus] [žiūrėta 2013 05 11]. Prieiga per internetą: http://christinelindcole.com/public/Bouncing BallHandout.pdf
8. Macau E. E. N., Carneiro M. V., Barroso J. J. Bouncing ball problem: numerical behavior characterization. XI Latin American workshop on nonlinear phenomena. Journal of physics:conference series, 246, 2010, 012003.
9. Miller A. J. Bouncing balls and bifurcations: examining a discontinuous iterated map, 1996, [interaktyvus] [žiūrėta 2013 05 11]. Prieiga per internetą: http://jan.ucc.nau.edu/~ns46/student/1996/Andrew_Milller.pdf
10. Morrison D. R. Two methods for determining impact time in the Bouncing ball system, 2008, [interaktyvus] [žiūrėta 2013 05 11]. Prieiga per internetą: http://www.math.hmc.edu/~levy/181_web/Morrison_web.pdf
11. Okniński A., Radziszewski B. An analytical and numerical study of chaotic dynamics in a simple bouncing ball model. Journal Acta Mechanica Sinica, DOI 10.1007/s10409-011-0406-3, 2011.
12. Okniński A., Radziszewski B. Bouncing ball dynamics: simple model of motion of the table and sinusoidal motion. Publikuota eprint arXiv:1302.0369, 2013.

13. Okniński A., Radziszewski B. Simple model of bouncing ball dynamics: displacement of the limiter assumed as a cubic function of time. Journal Differential Equations Dynamical Systems, DOI 10.1007/s12591-012-0137-3, 2012.
14. Okniński A., Radziszewski B. Simple model of bouncing ball dynamics: displacement of the table assumed as a quadratic function of time. Journal Nonlinear Dynamics, 67, DOI 10.1007/s11071-011-0055-x, 2012, p. 1115-1122.
15. Østfeldt C., Dideriksen K. M. Bouncing ball. Chaotic mechanics, project no. 2012-27, 2012.
16. Ott E. Chaos in Dynamical Systems. Cambridge University Press, New York, 2002.
17. Plukas K. Skaitiniai metodai ir algoritmai: vadovėlis aukštųjų mokyklų studentams, Kaunas: Naujasis lankas, 2001. - 548 p.
18. Stöckmann H. J. Quantum chaos: an introduction. Cambridge University Press, Cambridge, 2007.
19. Tufillaro N. B. Comment on „Bouncing ball with finite restitution: chattering, locking, and chaos“, 1994.
20. Tufillaro N. B., Abbott T., Reilly J. An experimental approach to nonlinear dynamics and chaos. Addison – Wesley publishing company, Redwood city, CA, 1992.
21. Tufillaro N. B., Albano A. M. Chaotic dynamics of a bouncing ball, 1985.
22. Tufillaro N. B., Mello T. M., Choi Y. M., Albano A. M. Period doubling boundaries of a bouncing ball. Journal Physique, 47, 1986, p. 1477–1482.
23. Vincent T. L. Chaotic control systems. Nonlinear dynamics and systems theory, 1(2), 2001, p. 205-218.
24. Vogel S., Linz S. J. Regular and chaotic dynamics in bouncing ball models. International Journal of Bifurcation and Chaos, Vol. 21, No. 3, 2011, p. 869-884.
25. Wahyoedi S. A., Viridi S. One- dimensional bouncing- ball system on a sinusoidal vibrating plate. Prosiding simposium nasional inovasi pembelajaran dan sains 2011, Bandung, Indonesia, 2011 06 22-23.

PRIEDAI

1 PRIEDAS. Programos *„tikslus_modelis_vaizdavimas.m“* kodas

```
clear all;
close all;

x0 = 1.1;
v0 = 0;
t0 = 0;
teta0 = 0;
omega = 2*pi;
g = 9.81;
alfa = 0.999;
beta = 1;
A = beta*g/(2*omega*omega*(1+alfa));

eps = 0.00000001; % saknies paieskos tikslumas
nt = 200;

% pirmo smugio koordinates:
[t1,x1,v1] = bisection(t0,x0,v0,alfa,g,omega,teta0,A,eps);
t = t1; x = x1; v = v1;

ttt = (t0:(t1-t0)/nt:t1)';
[xxx1,vvv1] = laisvas_skrydis(ttt,x0,v0,t0,g,omega,teta0,A);
xxx = xxx1; vvv = vvv1;
tic;
N = 10; % vaizduojamu susidurimu skaicius
for (i=2:N)
    %"dimensiniai" dydziai:
    t0 = t1; x0 = x1; v0 = v1;
    [t1,x1,v1] = bisection(t0,x0,v0,alfa,g,omega,teta0,A,eps);
    t = [t; t1]; x = [x; x1]; v = [v; v1];
    ttt1 = (t0:(t1-t0)/nt:t1)';
    [xxx1,vvv1] = laisvas_skrydis(ttt1,x0,v0,t0,g,omega,teta0,A);
    ttt = [ttt; ttt1]; xxx = [xxx; xxx1]; vvv = [vvv; vvv1];

    % Bedimensiai dydziai:
%     teta = mod(omega*t+teta0,2*pi);
    teta = mod(omega*real(t)+teta0,2*pi);
    ni = 2*omega*v/g;
end
tElapsed = toc;
disp(' smugio laikas (s): padetis (m): kamuoliuko greitis (m/s): ');
disp([t x v]);
disp('laikas, reikalingas skaiciavimams,s:');
disp(toc);
figure;
plot(teta,ni,'.b');
figure;
sss = A*sin(omega*ttt+teta0)+A;
plot(ttt,sss,'-r',ttt,xxx,'-b',t,x,'o','linewidth',2);
```

```
xlabel('laikas (s)')
ylabel('padetis (m)')
```

2 PRIEDAS. Pirmojo pavyzdžio simuliacijos rezultatai

k	k-tojo smūgio laiko momentas t_k, s	Kamuoliuko padėtis k-tojo smūgio momentu $x(t_k)$, m	Kamuoliuko greitis v_k, $\frac{m}{s}$
1	0,0567	0,0842	1,2825
2	0,2827	0,1236	0,7657
3	0,4957	0,0641	0,5302
4	0,6698	0,0078	0,7889
5	0,8305	0,0078	1,1586
6	1,0010	0,0628	1,2888
7	1,2040	0,1223	0,9185
8	1,4192	0,0928	0,4985
9	1,5967	0,0268	0,5886
10	1,7519	0,0000	0,9338

3 PRIEDAS. Programos „*niutono_metodas.m*“ kodas

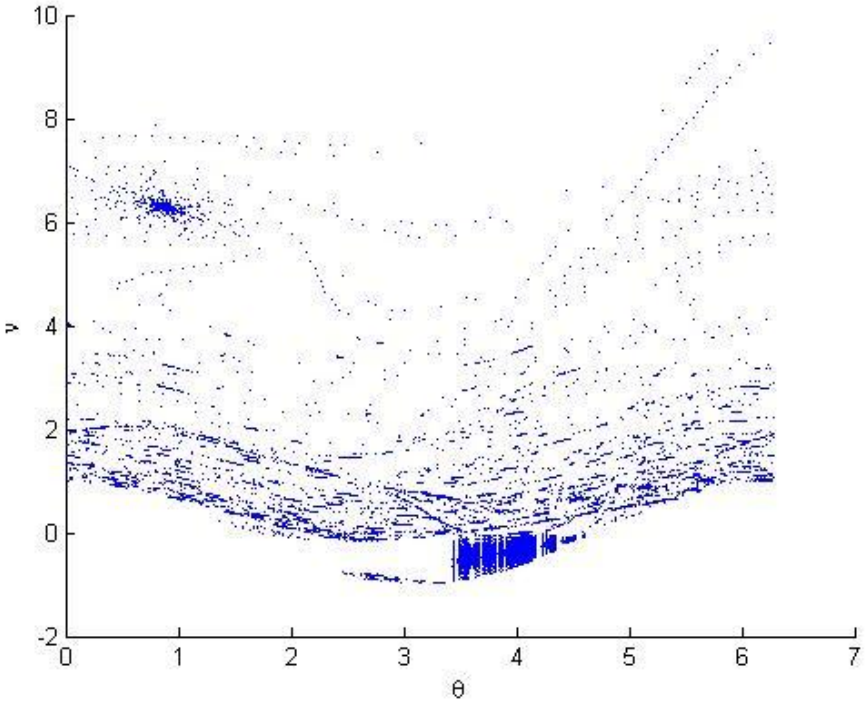
```
x=0.1;                                              %pradinis taskas
nmax=25;                                            %iteraciju skaicius
eps=1;                                              %paklaidos riba eps
xvals=x;                                            %iteraciju rodykle
n=0;                                                %iteraciju skaitliukas
tic;
while eps>=1e-5&n<=nmax
    y=x-(-0.062435*sin(2*pi*x)-4.905*x^2+0.037565)/(-0.062435*2*pi*cos(2*pi*x)-
9.81*x);
    xvals=[xvals;y];
    eps=abs(y-x);                                   %skaiciuojama paklaida
    x=y;n=n+1;                                      %atnaujinami x ir n
end
tElapsed =toc;
disp(toc);                                          %skaiciavimu trukme
disp(xvals);
disp(vpa(x,6));
disp(eps);
disp(n);
```

4 PRIEDAS. Antrojo pavyzdžio simuliacijos rezultatai

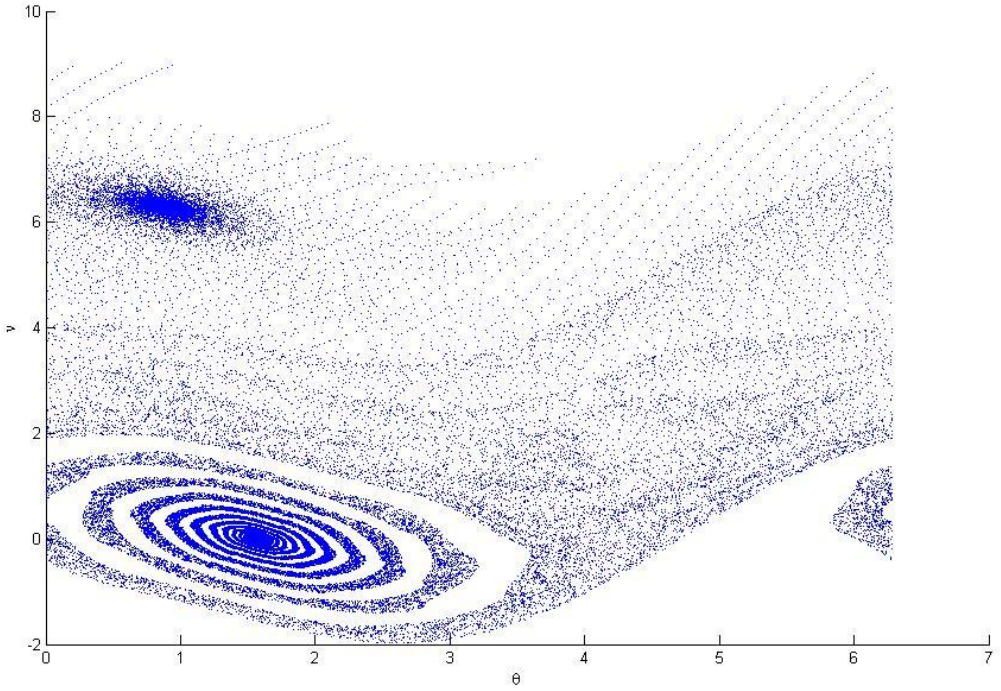
k	k-tojo smūgio laiko momentas t_k, s	Kamuoliuko padėtis k-tojo smūgio momentu $x(t_k)$, m	Kamuoliuko greitis v_k, $\frac{m}{s}$
1	0,4562	0,0790	3,7198
2	1,2030	0,1216	3,8295
3	1,9990	0,0618	4,7565
4	2,9710	0,0509	5,5418
5	4,0926	0,0963	6,1076
6	5,3346	0,1157	5,6746
7	6,5009	0,0618	4,9806
8	7,5176	0,0553	4,2126
9	8,3635	0,1092	3,5704
10	9,0948	0,0970	4,2463

5 PRIEDAS. Tikslios sistemos ir Aukšto šuolio aproksimacijos standartiniai žemėlapiai

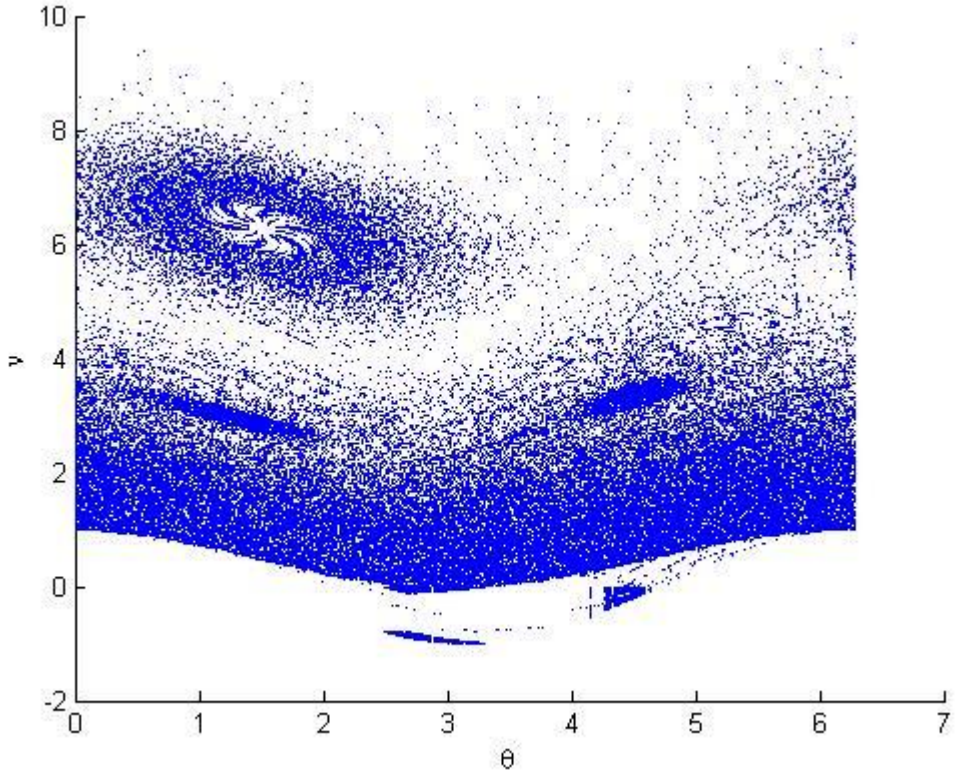
$\alpha = 0,9,$ tiksli sistema:



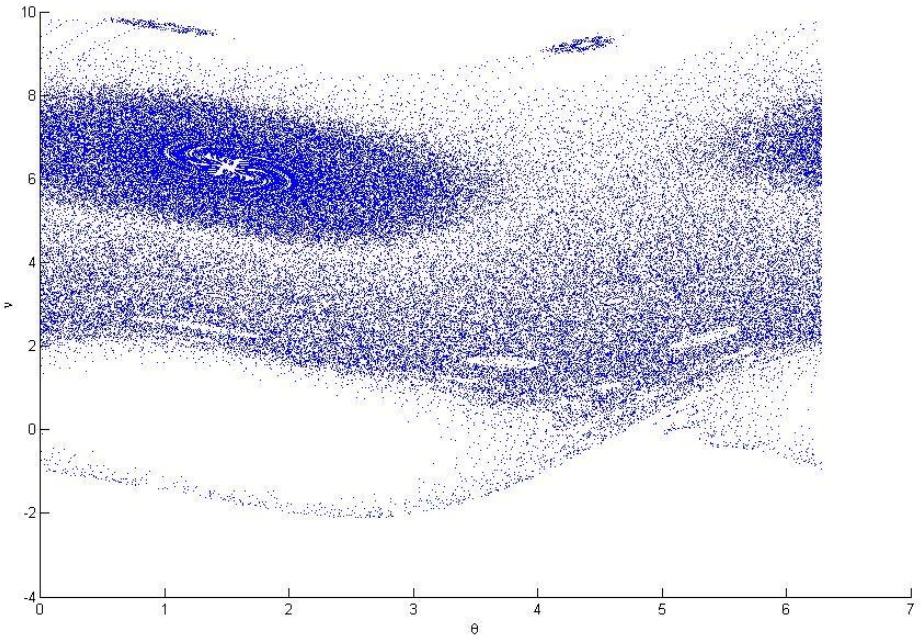
α = 0,9, Aukšto šuolio aproksimacija:



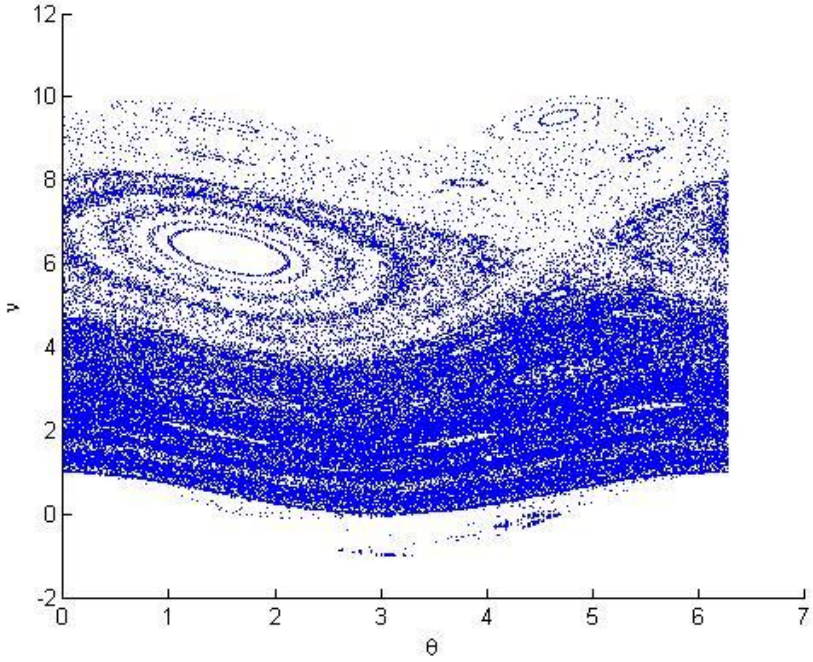
$\alpha = 0{,}99,$ tiksli sistema:



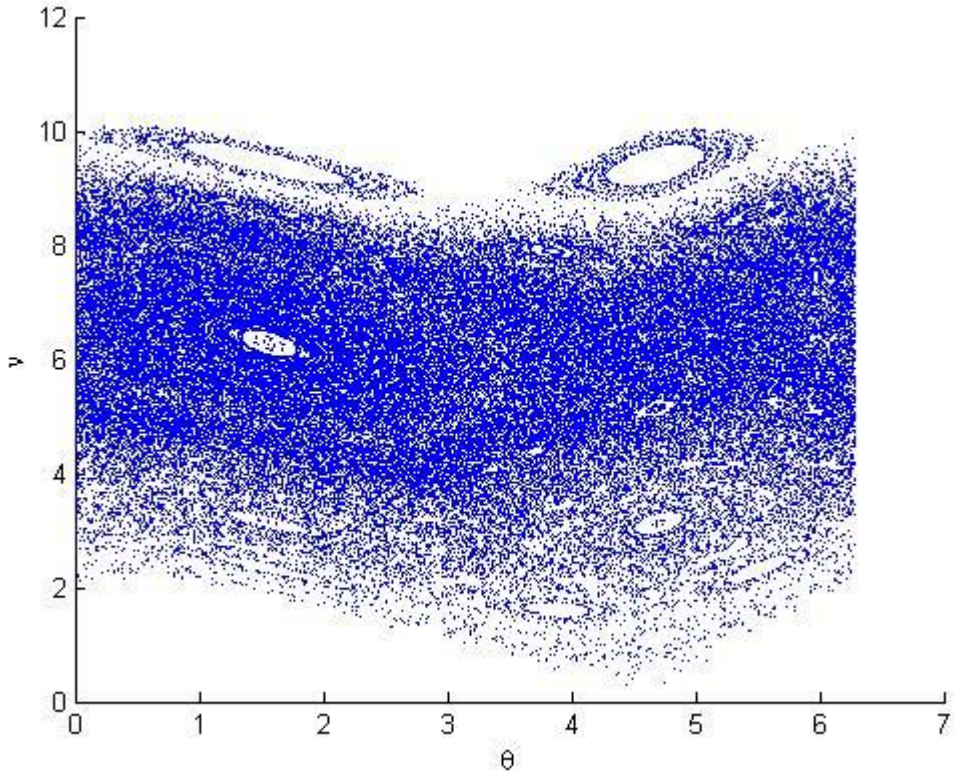
$\alpha = 0{,}99,$ Aukšto šuolio aproksimacija:



α = 0,999, tiksli sistema:



α = 0,999, Aukšto šuolio aproksimacija:



6 PRIEDAS. Kitų programų kodai

“*bisection.m*”

```
function [tk,xk,vk] = bisection(t0,x0,v0,alfa,g,omega,teta0,A,eps)
% Saknies radimas dalijimo pusiau metodu.
% t0,x0,v0 - pradines salygos: laikas, padetis, greitis
% alfa,g,omega,teta0,A - modelio parametrai
% eps - algoritmo sustojimo salyga (kai izoliacijos intervalo ilgis<=eps)

% Kadangi kamuoliukas negali isiskverbti i stala, tai izoliacijos
% intervala, kuriame kamuoliukas atsimusa i stala, nustatysim taip, kad
% intervalo pradzioje kamuoliukas butu virs stalo, o pabaigoje - po juo
% (hipotetine situacija, naudojama tik skaitiniam sprendimui)

% Nustatom pradini izoliacijos intervala (a1;a2):
a2 = 0;
if (v0<=eps) % -> kamuoliukas leidziasi (salyga turetu buti v0<0, leidziam
    % paklaidas, tai v0<eps)
    if (x0>2*A)
        % Kai kamuoliukas krenta is aukstai (x0>2A), izoliacijos intervalas:
        % tpr = arg(x=2A); tgal=arg(x=0)
        a1 = 2*A;
    else
        % Kai kamuoliukas krenta is zemai (x0<2A), izoliacijos intervalas:
        % tpr = arg(x=x0); tgal=arg(x=0).
        a1 = x0;
    end
else % -> kamuoliukas kyla: reikia padaryti kad leistusi, todel surandam
    % kamuoliuko laisvo skridimo trajektorijos (kuri yra paraboles formos)
    % virsunes koordinates. O nuo sio tasko kamuoliukas leidziasi zemyn.
    tmax = t0+v0/g;
    [xmax, vmax] = laisvas_skrydis(tmax,x0,v0,t0,g,omega,teta0,A); % vmax turetu buti
~=0
    d = xmax-A*(sin(omega*tmax+teta0)+1); % atstumas tarp kamuoliuko ir stalo
    if ((d>0)&&(xmax>2*A))
        a1 = 2*A;
        t0 = tmax; x0 = xmax; v0 = vmax;
    elseif ((d>0)&&(xmax<=2*A))
        t0 = tmax; x0 = xmax; v0 = vmax;
        a1 = x0;
    else  % "iseinam" is smugio:
        ttarp = t0+2*eps;
        [xt, vt] = laisvas_skrydis(ttarp,x0,v0,t0,g,omega,teta0,A);
        t0 = ttarp; x0 = xt; v0 = vt;
        a1 = x0;
        a2 = xmax;
    end
end

if (a2==0)
    tpr = t0+(v0/g)+sqrt((v0^2)+2*(x0-a1)*g)/g;
else
    tpr = t0;
end
tgal = t0+(v0/g)+sqrt((v0^2)+2*(x0-a2)*g)/g;
```

```
dpr = pirmas_sm(tpr,x0,v0,t0,g,omega,teta0,A);
dgal = pirmas_sm(tgal,x0,v0,t0,g,omega,teta0,A);

tvid = (tpr + tgal)/2;
dvid = pirmas_sm(tvid,x0,v0,t0,g,omega,teta0,A);

while (abs(dgal-dpr)>eps)
    if (dvid*dpr<0)
        tgal = tvid;
        dgal = pirmas_sm(tgal,x0,v0,t0,g,omega,teta0,A);
    else
        tpr = tvid;
        dpr = pirmas_sm(tpr,x0,v0,t0,g,omega,teta0,A);
    end
    tvid = (tpr + tgal)/2;
    dvid = pirmas_sm(tvid,x0,v0,t0,g,omega,teta0,A);
end

tk = tpr;
xk = A*(sin(omega*tpr+teta0)+1);
vk = (1+alfa)*A*omega*cos(omega*tk+teta0)-alfa*(v0-g*(tk-t0));
if (dpr<0)
    xk = A*(sin(omega*tpr+teta0)+1);
    vk = A*omega*cos(omega*tk+teta0);
end
```

„pirmas_sm.m“

```
function d = pirmas_sm(t,x0,v0,t0,g,omega,teta0,A)
d = x0+v0*(t-t0)-g*((t-t0).^2)/2-A*sin(omega*t+teta0)-A;
```

„auksti_atsokimai_vaizdavimas.m“

```
clear all;
close all;

x0 = 1;
v0 = 0;
t0 = 0;
teta0 = 0;
omega = 2*pi;
g = 10;
alfa = 1;
beta = 1;
A = beta*g/(2*omega*omega*(1+alfa));

eps = 0.000001; % saknies paieskos tikslumas

% pirmo smugio koordinates:
[t1,x1,v1] = bisection(t0,x0,v0,alfa,g,omega,teta0,A,eps);

ttt = (t0:(t1-t0)/400:t1)';
[xxx1,vvv1] = laisvas_skrydis(ttt,x0,v0,t0,g,omega,teta0,A);
```

```
xxx = xxx1; vvv = vvv1;

teta = mod(omega*t1+teta0,2*pi);
ni = 2*omega*v1/g;

N = 10; % vaizduojamu susidurimu skaicius
for (i=2:N)
    % Bedimensiai dydziai:
    teta = [teta; mod(teta(i-1)+ni(i-1),2*pi)];
    ni = [ni; alfa*ni(i-1)+beta*cos(teta(i-1)+ni(i-1))];

    %"dimensiniai" dydziai:
    t0 = t1; x0 = x1; v0 = v1;
    t1 = t0+2*v0/g;
    ttt1 = (t0:(t1-t0)/200:t1)';
    [xxx1,vvv1] = laisvas_skrydis(ttt1,x0,v0,t0,g,omega,teta0,A);
    ttt = [ttt; ttt1];
    xxx = [xxx; xxx1];
    vvv = [vvv; vvv1];

    x1 = x0+v0*(t1-t0)-g*((t1-t0)^2)/2;
    v1 = alfa*v0+(2*omega*beta/g)*cos(omega*t1+teta0+2*omega*v0/g);

end

figure;
plot(teta,ni,'.b','MarkerSize',0.8);

figure
sss = A*sin(omega*ttt+teta0)+A;
plot(ttt,sss,'-r',ttt,xxx,'-k');
```

„zemelapis_supaprastintas_atvejis.m“

```
t0 = 0;
teta0 = 0;
omega = 2*pi;
g = 9.81;
alfa = 1;
beta = 1;
A = beta*g/(2*omega*omega*(1+alfa));
eps = 0.000001; % saknies paieskos tikslumas
N = 200; % vaizduojamu tasku skaicius
xpr = 0.2; xgal = 2; nx = 40;
vpr = -4; vgal = 4; nv = 20;

figure;
xlabel('\theta'); ylabel('\nu');

for (i=0:nx)
    x0 = xpr+(xgal-xpr)*i/nx;
    for(j=0:nv)
        v0 = vpr+(vgal-vpr)*j/nv;
        [t1,x1,v1] = bisection(t0,x0,v0,alfa,g,omega,teta0,A,eps);
```

```
        teta = mod(omega*t1+teta0,2*pi);
        ni = 2*omega*v1/g;
        for (i=2:N)
            teta = [teta; mod(teta(i-1)+ni(i-1),2*pi)];
            ni = [ni; alfa*ni(i-1)+beta*cos(teta(i-1)+ni(i-1))];
        end
        hold on;
        plot(teta,ni,'.','MarkerSize',0.5);
    end
end
```

„zemelapis_tikslus_atvejis.m“

```
t0 = 0;
teta0 = 0;
omega = 2*pi;
g = 10;
alfa = 0.6;
beta = 1;
A = beta*g/(2*omega*omega*(1+alfa));
eps = 0.000001; % saknies paieskos tikslumas
N = 200; % vaizduojamu tasku skaicius
xpr = 0.2; xgal = 2; nx = 40;
vpr = -4; vgal = 4; nv = 20;

figure;
xlabel('\theta'); ylabel('\nu');

for (i=0:nx)
    x0 = xpr+(xgal-xpr)*i/nx;
    for(j=0:nv)
        v0 = vpr+(vgal-vpr)*j/nv;
        [t1,x1,v1] = bisection(t0,x0,v0,alfa,g,omega,teta0,A,eps);
        t = t1; v = v1;

        for (i=2:N)
            %"dimensiniai" dydziai:
            t0 = t1; x0 = x1; v0 = v1;
            % Surandam auksciausio pakilimo taska, t.y. nuo kada kamuoliukas
            % pradeda kristi zemyn:
            tmax = t0+v0/g;
            [xmax, vmax] = laisvas_skrydis(tmax,x0,v0,t0,g,omega,teta0,A); % vmax
turetu buti ~=0
            [t1,x1,v1] = bisection(tmax,xmax,vmax,alfa,g,omega,teta0,A,eps);
            t = [t; t1]; x = [x; x1]; v = [v; v1];
        end
        teta = mod(omega*t+teta0,2*pi);
        ni = 2*omega*v/g;
        hold on;
        plot(teta,ni,'.','MarkerSize',0.5);
    end
end
```

„laisvas_skrydis.m“

```
function [x,v] = laisvas_skrydis(t,x0,v0,t0,g,omega,teta0,A)
x = x0+v0*(t-t0)-g*((t-t0).^2)/2;
v = v0-g*(t-t0);
```