SVEUČILIŠTE U ZAGREBU
PRIROOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Mijo Dropuljić

Diplomski rad

IZRADA INTERAKTIVNIH ANIMACIJA ZA SIMULIRANJE HARMONIČKOG OSCILATORA

Zagreb, 2008.

SVEUČILIŠTE U ZAGREBU
PRIROOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

SMJER: PROF. FIZIKE I INFORMATIKE

Mijo Dropuljić

Diplomski rad

IZRADA INTERAKTIVNIH ANIMACIJA ZA SIMULIRANJE HARMONIČKOG OSCILATORA

Voditelj diplomskog rada: doc.dr.sc. Darko Androić

Ocjena diplomskog rada:_____

Povjerenstvo: 1._____

2._____

3._____

Datum polaganja:

Zagreb, 2008.

Zahvaljujem profesoru doc.dr.sc Darku Androiću na pomoći pri odabiru teme, kao i svim ostalim profesorima i asistentima na kvalitetno prenesenom znanju, koje sam usvojio tijekom studiranja.
Zahvaljujem svojim sestrama Sandri i Korneliji, a pogotovo sestri Sandri, s kojom sam dijelio dobro i zlo tijekom stanovanju u Zagrebu.
Zahvaljujem i cijeloj obitelji Pezo koja je vjerovala u mene.

I na kraju jedno veliko ***HVALA*** *mojim roditeljima na moralnoj i materijalnoj podršci u životu.*

Sadržaj

Uvod

Fiziku sam zavolio još u osnovnoj školi. To je bio novi nastavni predmet nakon šest godina školovanja i naravno meni jako zanimljiv. Nakon završene osnovne škole upisao sam prirodoslovno-matematičku gimnaziju, gdje mi je informatika bila obavezan predmet, te sam se tako posvetio i bolje upoznao i tu znanost. No, to je sve brzo prošlo i slijedio je upis na fakultet, gdje sam htio nastaviti proučavati ove dvije znanosti, što mi je ovaj smjer i omogućio. Moram priznati da na prvoj godini nisam imao velikih problema sa polaganjem ispita iz fizika i matematika, ali me namučilo učenje programiranja u raznim programskim jezicima. Kad sam na višim godinama studija savladao tehniku programiranja, bio sam u mogućnosti riješiti i neke fizikalne probleme, te ih i vizualno predočiti sa nekim programskim jezicima. Nisam mogao vjerovati kako se u vrlo kratkom vremenu može izraditi kvalitetna aplikacija.

Tijekom školovanja u sklopu informatičkih kolegija neprestano sam se susretao na Internetu sa aplikacijama u Javi koje simuliraju neki fizikalni problem. Primijetio sam da se neki fizikalni problemi mogu puno bolje razjasniti interaktivnom animacijom, nego eksperimentalno. Odabrao sam programirati u Javi jer se tijekom studiranja nismo upoznali s njom, ali i zbog toga jer se u njoj može jednostavnije definirati animacijske petlje.

Mentor mi je ponudio temu i ja sam prihvatio, iako do tada nisam imao nikakvoga znanja o Javi, ali sam se oslonio na znanja iz C, C++ i Pythona, nadajući se sličnosti u sintaksi.

Odabrao sam baš ovu temu jer smatram da je titranje jedno od bitnijih fizikalnih problema u fizici. Kod učenika postoji problem zašto tijelo obješeno na oprugu titra gore-dolje, te koje sile uzrokuju to gibanje. Pored toga jako im je nejasno zašto baš graf ovisnosti položaja tijela o vremenu izgleda kao sinusoida, odnosno kosinusoida. Sve to sam htio povezati u jednu lako razumljivu cjelinu. Ovim diplomskim radom želio sam povezati obje znanosti, fiziku i informatiku, u jednu cjelinu i time pokazati da se one međusobno isprepleću.

Diplomski rad sastoji se od četiri glavna poglavlja. U prvom i drugom poglavlju opisani su osnovni pojmovi harmoničkog osciliranja bez gušenja i sa gušenjem. Drugo poglavlje opisuje programski jezik Javu, te kako se koristi i od čega se sastoji applet koji sam izradio. U posljednjem dijelu opisan je način na koji bih ja izveo nastavni sat u srednjoj školi i implementacija appleta koji sam izradio.

1. Harmonički oscilator

1.1. Uvod

Harmonički oscilator je izuzetno važan primjer periodičkog gibanja, jer služi kao točan ili kao približan model za mnoge probleme u klasičnoj i kvantnoj fizici. Klasični sustavi, koje možemo prikazati harmoničkim oscilatorom, obuhvaćaju bilo koji stabilni sustav koji neznatno pomaknemo iz položaja ravnoteže, kao na primjer:

- tijelo na spiralnoj opruzi s malom amplitudom titranja,
- jednostavno njihalo kod malih kutova njihanja,
- električni krug sa zavojnicom i kondenzatorom kod struja i napona koji su dovoljno maleni da su elementi kruga linearni.

Kažemo da je element električnog ili mehaničkog kruga linearan, ako je njegov odziv razmjeran sili koja ga tjera. Većina pojava (ali ne i sve koje su nam zanimljive) u fizici linearne su odaberemo li dovoljno malo područje, kao što se i većina krivulja s kojima računamo mogu smatrati pravcima za dovoljno mala područja vrijednosti.
Najvažnija svojstva harmoničkog oscilatora su:

- Frekvencija gibanja je neovisna o amplitudi titranja dotle dok je unutar ograničenja linearnosti.
- Učinci nekoliko pogonskih sila linearno se superponiraju.

U sljedećim poglavljima razmatrati ćemo navedena svojstva harmoničkog oscilatora, promatrati ćemo i slobodno gibanje sa prigušenjem i bez njega.

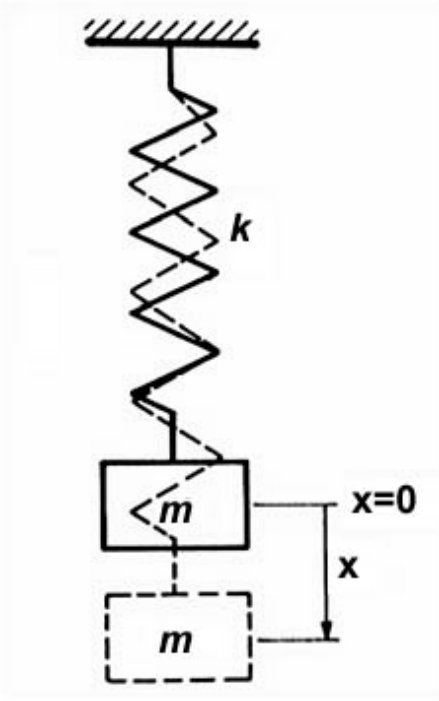
1.2. Elastična sila

Harmonijsko gibanje je ubrzano gibanje: i smjer i veličina brzine neprestano se mijenjaju. Prema tome kod tog gibanja mora djelovati sila. Pokazati ćemo da je ta sila promjenjiva. Svojstva te sile možemo kvalitativno proučiti na primjeru gibanja elastične opruge na koju je obješen uteg (Slika 1.1.).

Pomaknemo li uteg iz položaja ravnoteže, npr. prema dolje, on će se vratiti prema gore, proći kroz položaj ravnoteže, doseći maksimalnu elongaciju prema gore, zaustaviti se i vratiti dolje. Promotrimo sada djelovanje sile u svakom pojedinom trenutku. Kada smo uteg pomaknuli iz položaja ravnoteže prema dolje, sila je djelovala prema gore, dakle suprotno od pomaka. Kada je uteg prošao kroz položaj ravnoteže prema gore, sila je promijenila smjer i počela usporavati njegovo gibanje. Kada je uteg dosegnuo najvišu točku i počeo se vraćati prema položaju ravnoteže, sila je još uvijek djelovala prema položaju ravnoteže, tj. suprotno od smjera pomaka. Vidi se dakle da sila stalno djeluje u smjeru suprotnom od pomaka utega. Točnija analiza pokazala bi da je uvijek bila proporcionalna pomaku, ali suprotnog smjera. Sila ima oblik:

$$F = -kx \tag{1.1.}$$

Gdje je *k* pozitivna konstanta. Takva sila zove se *elastična sila*.



Slika 1.1. Titranje tijela na opruzi

1.3. Jednadžba harmoničkog titranja

Svaka sila tipa $F = -kx$ proizvodi harmonijsko titranje. Ako uteg mase *m* objesimo o elastičnu oprugu uteg se giba u smjeru osi *x*. Jednadžba gibanja u tom slučaju glasi:

$$F = -kx = ma = m\frac{d^2x}{dt^2} \tag{1.2.}$$

odnosno:

$$m\frac{d^2x}{dt^2} + kx = 0 \tag{1.3.}$$

Jednadžba (1.3.) je diferencijalna jednadžba koja određuje funkciju *x(t)* vremenske ovisnosti udaljenosti utega od položaja ravnoteže. Tu ćemo jednadžbu jednostavno tješiti napišemo li je u obliku:

$$\frac{d^2x}{dt^2} = -\frac{k}{m}x \tag{1.4.}$$

Tražimo dakle funkciju *x(t)*, čija je druga derivacija proporcionalna samoj funkciji, ali sa suprotnim predznakom. Takve su funkcije na primjer sinus ili kosinus. Pretpostavit ćemo dakle da rješenje diferencijalne jednadžbe (1.3.) ima oblik kosinus funkcije:

$$x = A\cos(\omega_0 t + \varphi) \tag{1.5.}$$

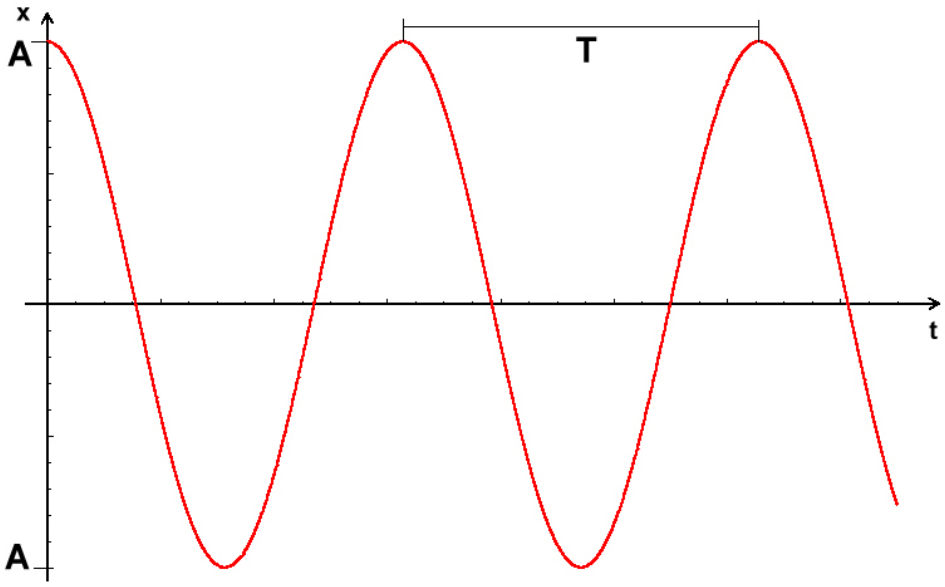
gdje je A neka konstantna duljina (*amplituda titranja*), a φ neki konstantni kut. Jednadžba (1.5.) je diferencijalna jednadžba drugog reda pa je razumljivo da njeno rješenje mora sadržavati dvije proizvoljne konstante. Fizikalno, da bismo potpuno zadali neko konkretno titranje moramo zadati dvije fizikalne veličine. Budući da se veličina $\omega_0 t$ pojavljuje kao neki kut i jer ω_0 ima dimenziju $\left[T^{-1}\right]$, onda ω_0 zovemo *kutna frekvencija* i mjerimo je u $\left(s^{-1}\right)$.

1.4. Fazni kut

Ovisnost o vremenu u jednadžbi (1.5.) dana je preko veličine $\omega_0 t + \varphi$ koju zovemo *fazni kut* ili jednostavnije *faza*. Fazni kut raste jednoliko s vremenom, međutim vrijednost *x*-eva bit će jednaka za sve fazne kutove koji se međusobno razlikuju za cijeli višekratnik od 2π. Dakle, harmoničko gibanje je periodičko i beskonačno se ponavlja u nizu identičnih ciklusa. Sve mjerljive fizikalne veličine harmoničkog oscilatora, kao što su pomak, brzina, smjer gibanja, ubrzanje itd., ponavljaju se kad god se fazni kut uveća za 2π, a to će se događati u vremenskim trenucima odvojenim za interval *T*, danim sa:

$$\omega_0 t = 2\pi \tag{1.6.}$$

Ovaj karakterističan vremenski interval *T* nazivamo *period* harmoničkog titranja.



Slika 1.2. Period harmoničkog titranja

Svaka vrijednost *fazne konstante* daje neko partikularno rješenje jednadžbe. Mijenjamo li samo iznos φ, slijed događaja u ciklusu ostati će isti, samo će se oni događati ranije ili kasnije za taj iznos vremena. Ako promijenimo φ za neki višekratnik od 2π neće doći do nikakve promjene. U trenutku *t=0* fazni kut je jednak φ, pa ga zato zovemo još i *početna faza*.

U toku jednog perioda, *x* poprima sve vrijednosti između *x(max)=A* i *x(min)=-A*, pa onda *A* zovemo *amplituda*.

1.5. Brzina tijela koje harmonički titra

Brzinu tijela mase *m* koje harmonički titra dobiti ćemo ako jednadžbu (1.5.) deriviramo po vremenu:

$$v(t) = \frac{dx(t)}{dt} = \frac{d}{dt}(A\cos(\omega_0 t + \varphi))$$

$$v(t) = -\omega_0 A\sin(\omega_0 t + \varphi) \tag{1.7.}$$

$$v(t) = -\omega_0 A\cos(\omega_0 t + \varphi + \frac{1}{2}\pi)$$

Odavde vidimo da se brzina također harmonički mijenja i to s istom frekvencijom kao i pomak iz ravnotežnog položaja *x*. Isto tako vidimo da brzina brza u fazi za $\frac{\pi}{2}$ pred pomakom. Ovo brzanje u fazi možemo izraziti brzanjem u vremenu. Naime, jednadžbu (1.5.) možemo napisati u obliku:

$$x(t) = A\cos\omega_0(t + \frac{\varphi}{\omega_0}) \tag{1.8.}$$

Veličina $\frac{\varphi}{\omega_0}$ ima dimenziju vremena pa možemo uvesti oznaku:

$$\frac{\varphi}{\omega_0} = t' \tag{1.9.}$$

Onda je:

$$x(t) = A\cos\omega_0(t + t') \tag{1.10.}$$

To znači da će tijelo poprimiti, tek nakon vremena t', fazu gibanja koju ima tijelo čije je gibanje opisano sa :

$$x = A\cos\omega_0 t \tag{1.11.}$$

Argument brzine možemo onda pisati kao:

$$\omega_0 t + \varphi + \frac{\pi}{2} = \omega_0 (t + \frac{\varphi}{\omega_0} + \frac{\pi}{2\omega_0} \cdot \frac{2}{2})$$

$$\omega_0 t + \varphi + \frac{\pi}{2} = \omega_0 (t + t' + \frac{T}{4}) \tag{1.12.}$$

Vidimo da brzina tijela koje harmonički titra brza za $\frac{1}{4}$ perioda ispred pomaka.
Maksimalna brzina koju masa postiže, tj. amplituda od *v* je iz jednadžbe (1.7.):

$$v_{\max} = \omega_0 A \tag{1.13.}$$

1.6. Akceleracija tijela koje harmonički titra

Akceleracija tijela koje harmonički titra je druga derivacija pomaka (1.5.) po vremenu:

$$a(t) = \frac{d^2 x(t)}{dt^2} = \frac{d^2}{dt^2}(A\cos(\omega_0 t + \varphi))$$

$$a(t) = -\omega_0^2 A\cos(\omega_0 t + \varphi) \tag{1.14.}$$

$$a(t) = \omega_0^2 A\cos(\omega_0 t + \varphi + \pi)$$

Vidimo da se akceleracija također mijenja harmonički i to s istom frekvencijom kao i pomak i brzina. Akceleracija brza u fazi ispred brzine za $\frac{\pi}{2}$, a to znači da je upravo u protufazi (brza za π) sa pomakom. Izrazimo sada brzanje u fazi preko brzanja u vremenu. Iz (1.14.) proizlazi da je:

$$a = -\omega_0^2 a$$

$$a = \omega_0^2 A\cos\omega_0 (t + \frac{\varphi}{\omega_0} + \frac{\pi}{\omega_0}) \tag{1.15.}$$

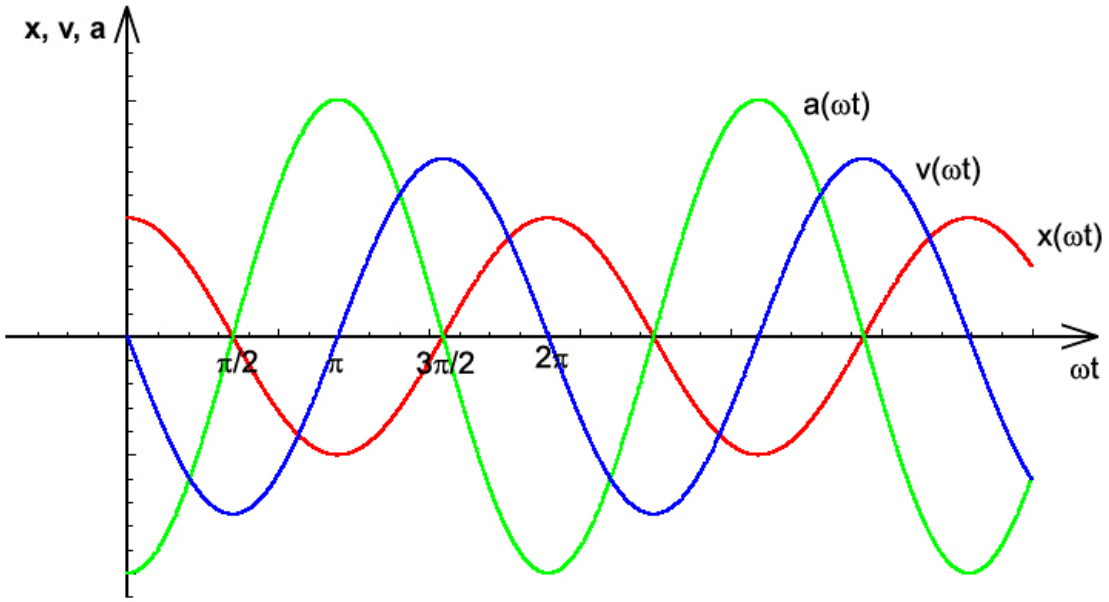
$$a = \omega_0^2 A\cos(t + t' + \frac{T}{2})$$

Dakle, akceleracija brza u vremenu za pola perioda pred pomakom, a za četvrtinu perioda za brzinom.

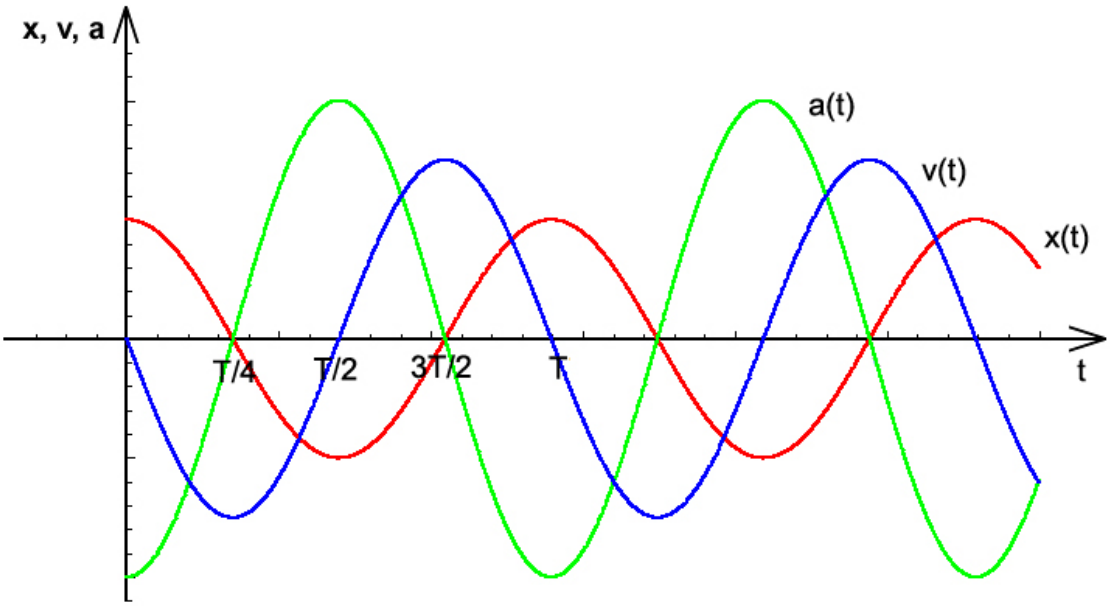
Maksimalna akceleracija, tj. amplituda od *a* je:

$$a_{\max} = \omega_0^2 A \tag{1.16.}$$

Brzanje u fazi akceleracije i brzine pred pomakom možemo prikazati grafički ne jednoj slici (Slika 1.3.), a isto tako brzanje u vremenu (Slika 1.4.).



Slika 1.3. Brzanje u fazi akceleracije i brzine pred pomakom



Slika 1.4. Brzanje u vremenu akceleracije i brzine pred pomakom

Budući da je sila (1.1.) harmonička i u vremenu oscilirajuća veličina, možemo je pisati u obliku:

$$F = -m\omega_0^2 x \tag{1.17.}$$

jer je ω_0^2 uvijek pozitivno, onda akceleracija $a = -\omega_0^2 x$ ima smjer uvijek suprotan pomaku *x*. To znači da sila (koja ima smjer akceleracije) djeluje uvijek suprotno pomaku tj. kao što smo već prije intuitivno zaključili, nastoji tijelo vratiti u ravnotežni položaj.

1.7. Frekvencija tijela koje harmonički titra

Broj titraja u jedinici vremena nazivamo *frekvencija*:

$$\nu_0 = \frac{1}{T} = \frac{\omega_0}{2\pi} \tag{1.18.}$$

Veličina ν_0 ima istu dimenziju kao i ω_0. Ako je vrijeme mjereno u sekundama, ν se izražava u hertzima (Hz).

1.8. Početni uvjeti

Jednadžba (1.5.) sadrži dvije proizvoljne konstante, amplitudu *A* i fazni kut φ. Bilo koji par vrijednosti amplitude i faznog kuta opisivati će neko specijalno titranje, koje može izvoditi tijelo na opruzi. Za konkretno titranje konstante *A* i φ određujemo iz *početnih* ili *rubnih uvjeta*.

Ako je masa na početku bila na nekoj udaljenosti *A* na desno od svog ravnotežnog položaja i onda puštena u trenutku *t=0*, možemo reći da je:

$$x(0) = A\cos(\omega_0 0 + \varphi) = A\cos\varphi = A_1 \tag{1.19.}$$

Deriviramo li *x(t)* po vremenu *t*, dobivamo za *t=0*:

$$x(0) = -\omega_0 A\sin(\omega_0 0 + \varphi) = -\omega_0 A\sin\varphi = 0 \tag{1.20.}$$

Iz jednadžbe (1.20.) slijedi da su za $\omega_0 \neq 0$ i $A \neq 0$, jedine moguće vrijednosti za φ, nula ili π. S druge strane jednadžba (1.19.) zahtjeva da je $\cos\varphi > 0$. Jedino rješenje koje zadovoljava oba uvjeta je $\varphi = 0$ (u tom slučaju je $\cos\varphi = 1$), pa je $A = A_1$.

Jednadžba (1.5.) uz ovakve početne uvjete poprima oblik:

$$x(t) = A_1\cos\omega_0 t \tag{1.21.}$$

Različiti početni uvjeti vode na različite vrijednosti A i φ.

1.9. Energija tijela koje harmonički titra

Kada se tijelo giba brzinom *v* njegova *kinetička energija* je:

$$E_k = \frac{1}{2}mv^2 \tag{1.22.}$$

Da bi smo vidjeli kako se ona mijenja s vremenom uvrstimo izraz (1.7.) za brzinu u jednadžbu (1.22.). Onda je:

$$E_k = \frac{1}{2}m\left[-\omega_0 A\sin(\omega_0 t+\varphi)\right]^2$$

$$E_k = \frac{1}{2}m\omega_0^2 A^2\sin^2(\omega_0 t+\varphi) \tag{1.23.}$$

Opruga rastegnuta ili stisnuta za duljinu *x* ima uskladištenu *potencijalnu energiju*:

$$E_p = \frac{1}{2}kx^2 \tag{1.24.}$$

Uvrstimo li u (1.24.) izraz za pomak, dobijemo promjenu potencijalne energije u vremenu:

$$E_p = \frac{1}{2}k\left[A\cos(\omega_0 t+\varphi)\right]^2$$

$$E_p = \frac{1}{2}kA^2\cos^2(\omega_0 t+\varphi) \tag{1.25.}$$

Ukupna energija sistema jednaka je zbroju kinetičke i potencijalne energije:

$$E = E_k + E_p = \frac{1}{2}mv^2 + \frac{1}{2}kx^2 \tag{1.26.}$$

tj. uvrstimo li izraze (1.23.) i (1.25.) dobijemo:

$$E = \frac{1}{2}m\omega_0^2 A^2\sin^2(\omega_0 t+\varphi) + \frac{1}{2}kA^2\cos^2(\omega_0 t+\varphi) \tag{1.27.}$$

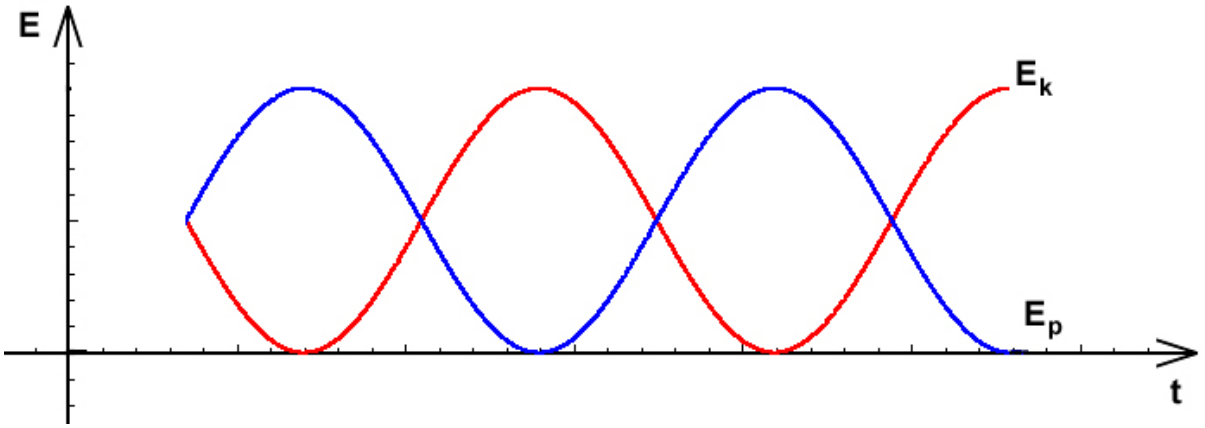
Iz jednadžbe $\omega_0^2 = \frac{k}{m}$ slijedi da je $k = \omega_0^2 m$, pa je:

$$E = \frac{1}{2} m \omega_0^2 A^2 \left[\sin^2(\omega_0 t + \varphi) + \cos^2(\omega_0 t + \varphi) \right]$$

$$E = \frac{1}{2} m \omega_0^2 A^2 \qquad (1.28.)$$

$$E = \frac{1}{2} k A^2$$

Odavde vidimo da je ukupna energija konstantna u vremenu, ako pretpostavimo da su sile kao npr. trenje zanemarive. Ona je proporcionalna kvadratu amplitude (Slika 1.5.).



Slika 1.5. Promjena kinetičke i potencijalne energije s vremenom

Titranje možemo shvatiti kao ponavljano prenošenje kinetičke energije tijela u elastičnu potencijalnu energiju opruge i obrnuto.
Udaljavanje tijela u horizontalnom smjeru iz položaja ravnoteže zahtjeva dovođenje rada sistemu, jer treba savladati harmoničku silu, a to znači povećanje potencijalne energije sistema. Kada vanjska sila prestane djelovati, harmonička sila sistema vraća tijelo prema položaju ravnoteže, pri tom ono dobiva kinetičku energiju. Kada se tijelo nalazi u ravnotežnom položaju, cijela potencijalna energija sistema, dovedena radom, pretvorena je u kinetičku energiju. Dakle, tijelo ovdje ima maksimalnu brzinu, pa samo prođe kroz položaj ravnoteže. Udaljavanjem tijela od položaja ravnoteže kinetička energija pretvara se postepeno u potencijalnu energiju, sve dok čitava kinetička energija nije iscrpljena, a to se događa onda kada se tijelo udaljilo od ravnotežnog položaja za upravo onu duljinu za koju je bilo udaljeno kada je vanjska sila prestala djelovati. Fizikalno, ona se nalazi u istoj situaciji kao na početku: sistem ima istu potencijalnu energiju koja se postepeno pretvara u kinetičku, masa se ponovo vraća prema ravnotežnom položaju, kojeg ponovo prijeđe i nakon toga dospijeva u prvobitni položaj. Sada sve počinje iz početka.

1.10. Primjer rješenja harmoničkog osciliranja

Harmonički oscilator je vrlo važan model, jer se gotovo obavezno pojavljuje kad modeliramo mala titranja oko ravnotežnog položaja nekog sistema.

Na svim grafovima na faznom portretu smjer gibanja je suprotan od smjera gibanja kazaljke na satu.

Na početku ćemo razmotriti jednadžbu za koju vrijedi i $k \neq 0$. Za ostale parametre uzmimo vrijednosti $m = 5$ i $k = 4$
Pogledajmo jednadžbu harmoničkog oscilatora:

$$m\frac{d^2y}{dt^2} + ky = 0$$

ili

$$\frac{d^2y}{dt^2} + \omega^2 y = 0$$

U jednadžbi postoje sljedeće veličine:

m – masa tijela (parametar)
y – pomak iz ravnotežnog položaja (zavisna varijabla)
t – vrijeme (nezavisna varijabla)
k – konstanta opruge (parametar)

Označimo sa v (zavisna varijabla) brzinu od y , tj. $v = \frac{dy}{dt}$ i dobijemo:

$$\frac{dv}{dt} + \omega^2 y = 0$$

što je diferencijalna jednadžba prvog reda, koja sadrži dvije zavisne varijable v i y,

gdje je ($\omega -$ *parametar*)

$$\omega^2 = \frac{k}{m} = \frac{4}{5}$$

Znači jednadžbu možemo napisati kao sustav dvije vezane autonomne diferencijalne jednadžbe prvog reda:

$$\frac{dy}{dt} = v$$

$$\frac{dv}{dt} = -\omega^2 y$$

U ovom slučaju jednadžbe imaju oblik:

$$\frac{dy}{dt} = v$$

$$\frac{dv}{dt} = -0.8y$$

Za rješavanje ćemo koristiti numerički postupak, npr. Eulerovu metodu:

$$y_{k+1} = y_k + v_k \Delta t$$

$$v_{k+1} = v_k - \omega^2 y_k \Delta t$$

Uzet ćemo tri različita početna uvjeta i pogledati što se događa s rješenjima.

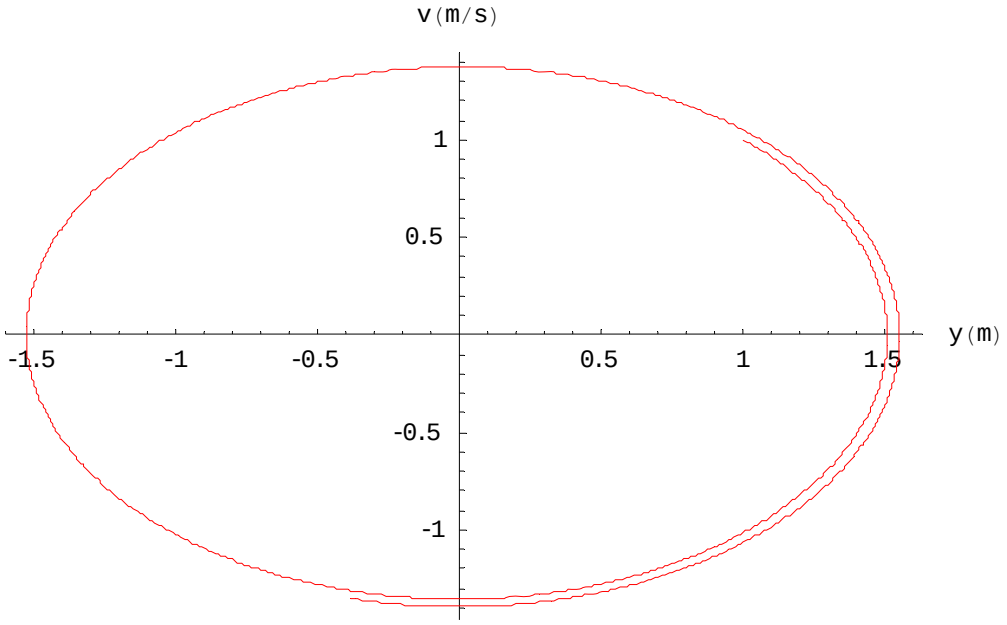
1. Slučaj

a) Uzeti ćemo početne uvjete: $y(0) = 1$ korak $\Delta t = 0.01$
$v(0) = 1$ broj koraka = 1000

Dobivamo sljedeće rezultate (u tablici su prikazani su samo početni koraci):

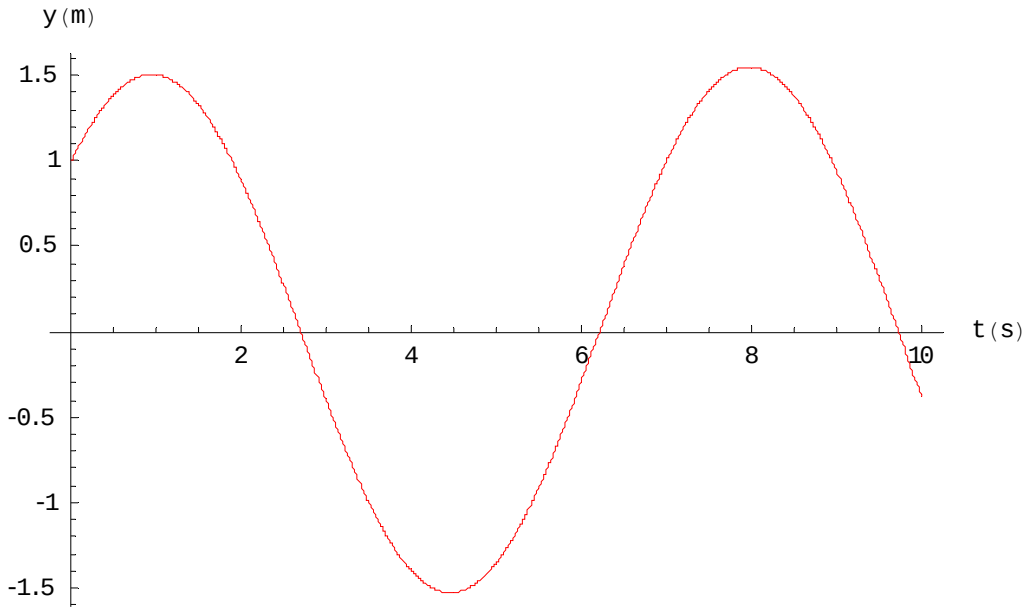
i	t(i)	v(i)	y(i)
0	0	1	1
1	0.01	0.992	1.01
2	0.02	0.98392	1.01992
3	0.03	0.975761	1.029759
4	0.04	0.967523	1.039571
5	0.05	0.959206	1.049192
6	0.06	0.950813	1.058784
7	0.07	0.942343	1.068292
8	0.08	0.933796	1.077716
9	0.09	0.925175	1.087054
10	0.1	0.916478	1.096305
11	0.11	0.907708	1.10547
12	0.12	0.898864	1.114547
13	0.13	0.889948	1.123536
14	0.14	0.880959	1.132435
15	0.15	0.8719	1.141245

Ako vrijednosti v(i) i y(i) prikažemo u faznoj ravnini (Slika 1.7.), za korak $\Delta t = 0.01$ i Broj koraka = 1000 dobivamo sljedeću sliku:

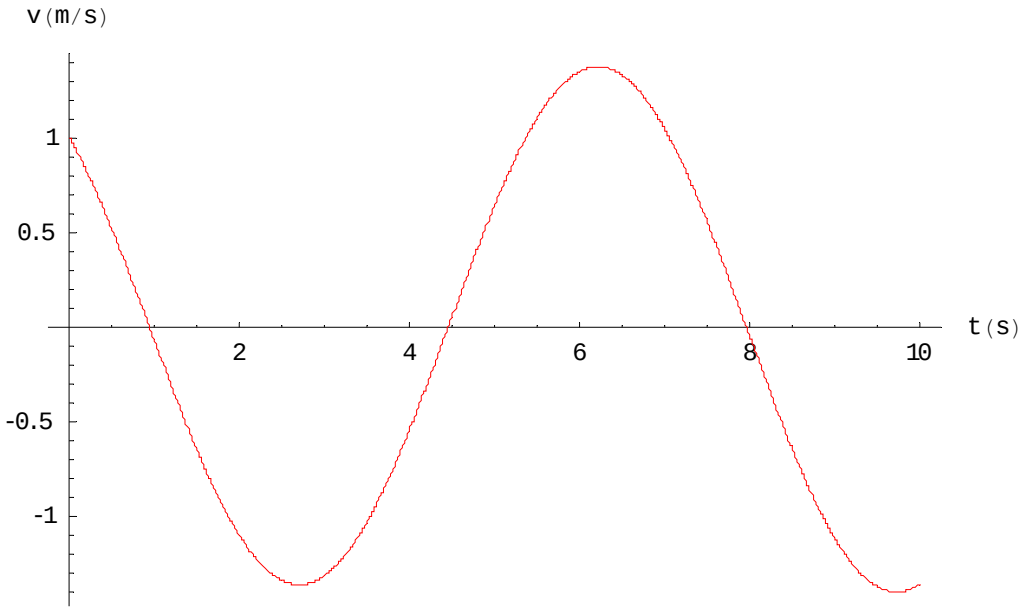


Slika 1.7. Fazni portret sustava sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001, broj koraka = 1000

Ako dobivene vrijednosti prikažemo kao grafove *y(t)* i *v(t)* dobivamo sljedeće slike (Slika 1.8.) i (Slika 1.9.)



Slika 1.8. y(t) graf rješenja sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001, broj koraka = 1000

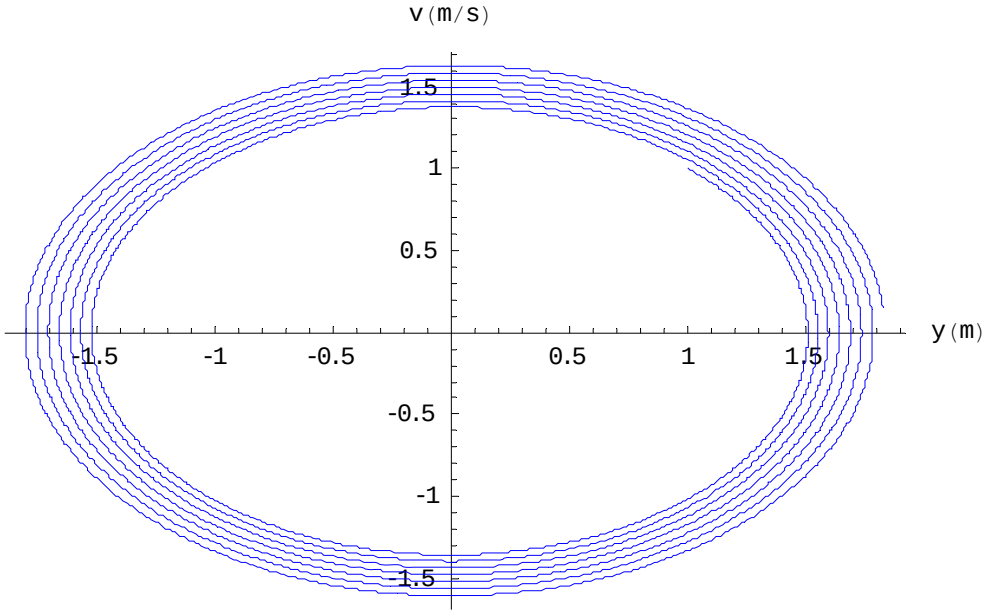


Slika 1.9. v(t) graf rješenja sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001, broj koraka = 1000

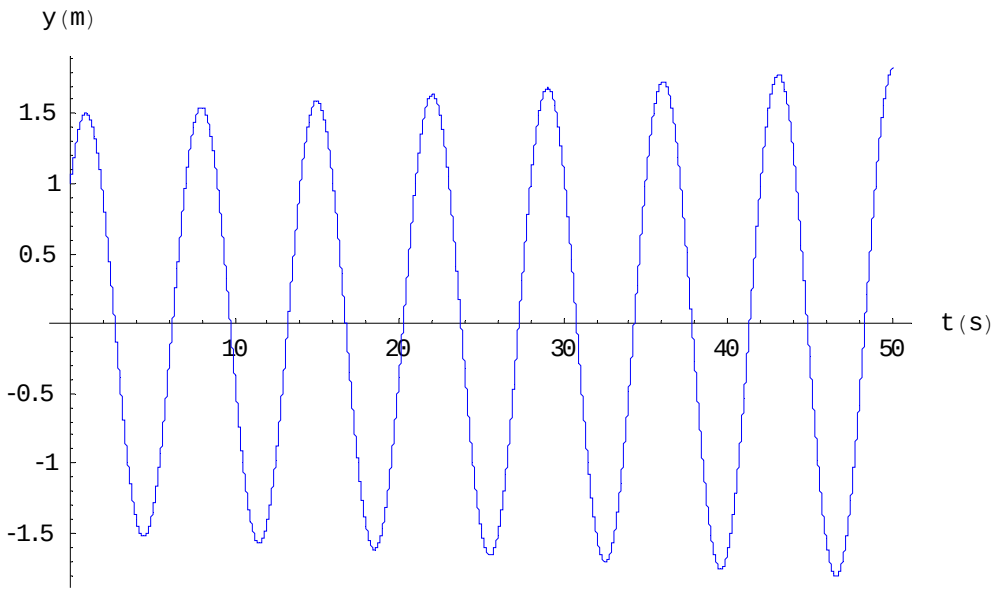
Iz slika bi se moglo zaključiti da se amplituda linearno povećava s vremenom, ali trebalo bi nam više koraka da bi bili sigurni.

b) Uzeti ćemo početne uvjete: $y(0) = 1$ korak $\Delta t = 0.01$
$v(0) = 1$ broj koraka = 5000

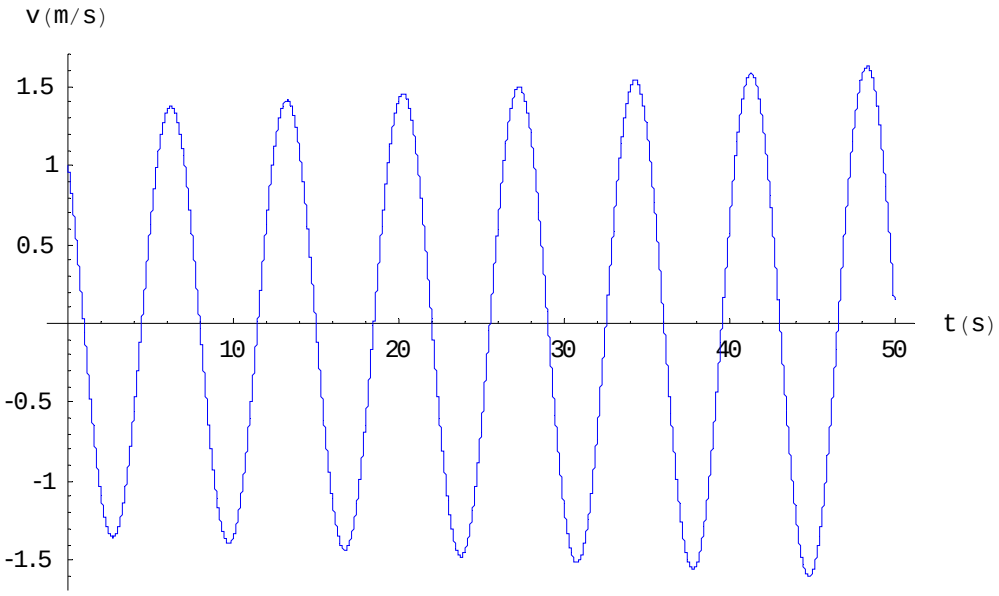
Znači povećali smo samo broj koraka i dobili sljedeće slike:



Slika 1.10. Fazni portret sustava sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001, broj koraka = 5000



Slika 1.11. y(t) graf rješenja sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001, broj koraka = 5000

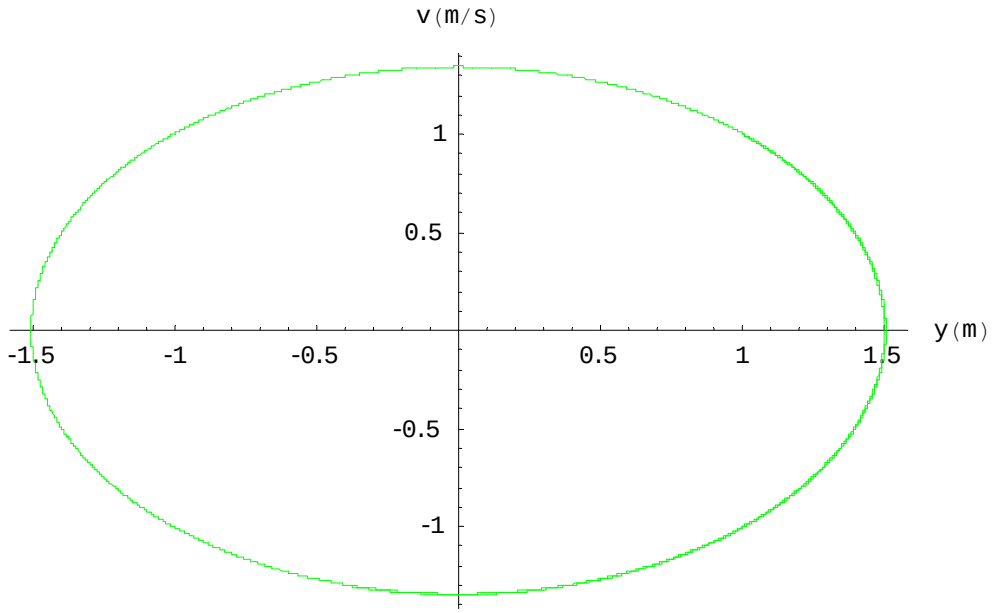


Slika 1.12. v(t) graf rješenja sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001, broj koraka = 5000

Gledajući slike (Slika 1.10., Slika 1.11., Slika 1.12.) zaključili bismo da se amplituda stvarno povećava s vremenom i da će neograničeno rasti, ali pošto znamo da je amplituda harmoničkog oscilatora konstanta (za velike pomake, tj. amplitude, jednadžba harmoničkog oscilatora ne bi više ni vrijedila), naš zaključak je da je greška u numerici tj. da je korak koji smo uzeli premali.

c) Ponoviti ćemo račun za iste početne uvjete: y(0) = 1, v(0) = 1, s manjim korakom $\Delta t = 0.001$ i broj koraka = 10000

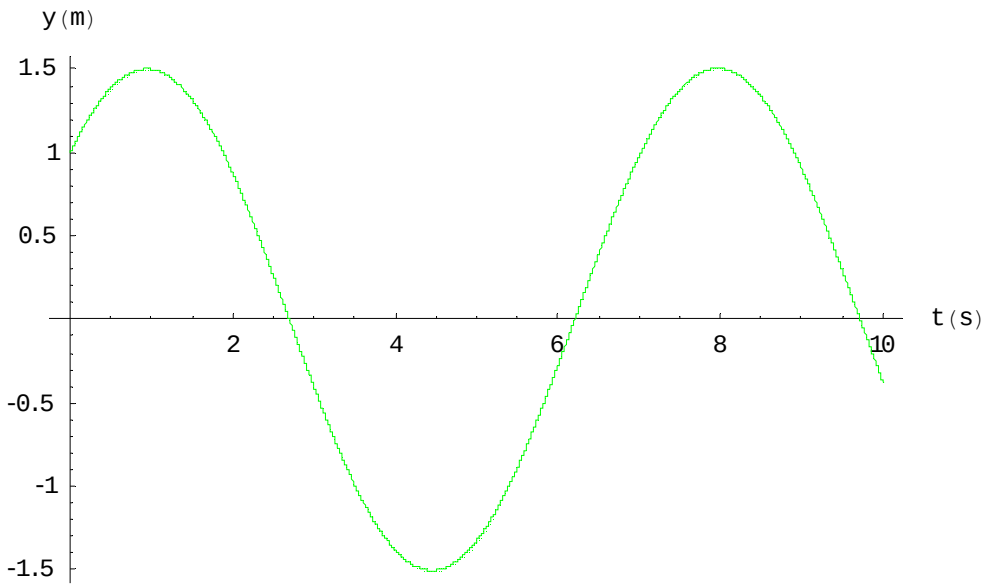
Dobiti ćemo sljedeće slike



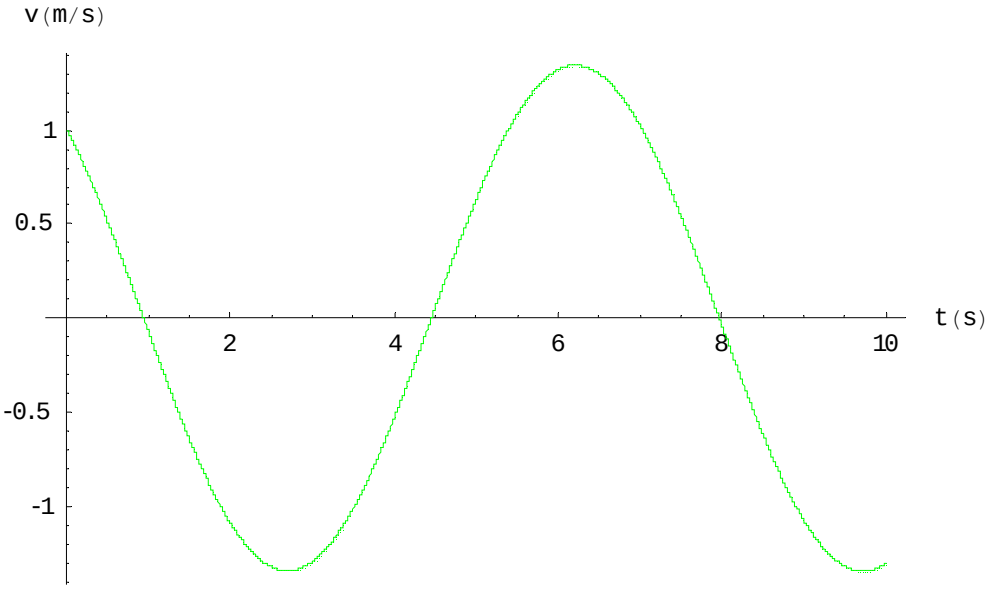
Slika 1.13. Fazni portret sustava sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001, broj koraka = 10000

Vidimo da u faznoj ravnini (Slika 1.13.) zapravo dobivamo zatvorenu krivulju, elipsu. Otprije nam je poznato rješenje da za simetričan slučaj dobivamo zatvorenu krivulju – kružnicu, a s obzirom da imamo slučaj koji nije simetričan u odnosu na v i y jer je $\omega^2 = \frac{k}{m} = \frac{4}{5}$, zaključujemo da je sadašnja veličina koraka dovoljno mala i da je rješenje dobro.

Iz krivulje rješenja u faznoj ravnini (Slika 1.13.) zaključujemo da je rješenje periodično (to već znamo otprije), a iz grafova y(t) (Slika 1.14) i v(t) (Slika 1.15.) možemo očitati period.



Slika 1.14. y(t) graf rješenja sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001 , broj koraka = 10000



Slika 1.15. v(t) graf rješenja sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.001 , broj koraka = 10000

Vidi se da *y(t)* i *v(t)* imaju jednak period $T \approx 7$.

Fizikalno objašnjenje početnih uvjeta:

Počinjemo sa pozitivnim pomakom y_0 (koji će se dalje povećavati) i pozitivnom brzinom v_0 (koja će se dalje smanjivati). To znači da smo počeli mjerenje u trenutku kad je masa

otklonjena iz ravnotežnog položaja, ali još nije dosegla maksimalni otklon. Otklon će se povećavati do maksimuma, a brzina će se smanjivati zbog povratne sile, dok ne postane nula. Općeniti opis gibanja vidi u slučaju 2. i 3.

Procjena pogreške

Razmotrit ćemo pitanje pogreške u Eulerovoj metodi. Pošto računamo numeričko približenje stvarnog rješenja, za očekivati je da postoje odstupanja, tj. pogreška. Također je za očekivati da će pogreška biti manja ako uzmemo manju veličinu koraka Δt.

Uzeti ćemo za početni uvjet (y(0), v(0)) = (1,1) u trenutku t = 1
Računamo numeričko približenje rješenja (y(1),v(1))
Za početak ćemo odabrati veličinu koraka $\Delta t^{(0)} = 0.002$ i broj koraka n = 500
Dobivamo rezultat (y(1), v(1)) = (1.49906 , -0.07161).

Zatim uzmemo dvostruko manji korak $\Delta t^{(1)} = 0.001$ i broj koraka n = 1000
Dobijemo rezultat (y(1),v(1)) = (1.49846 , -0.071582)

Razlika između ta dva rezultata $(\Delta y^{(1)}, \Delta v^{(1)})$ = (-0.0006 , 0.000028) je ocjena pogreške numeričkog približenja sa korakom $\Delta t^{(1)} = 0.001$.

Za provjeru izračunamo (y(1) , v(1)) za ponovo prepolovljeni broj koraka $\Delta t^{(2)} = 0.0005$ i brojem koraka n = 2000 Dobijemo rezultat (y(1),v(1)) = (1.49817 , -0.071568).
Razlika u odnosu na rezultat $\Delta t^{(1)} = 0.001$ je: $(\Delta y^{(2)}, \Delta v^{(2)})$ = (-0.000299 , 0.000014)

Usporedbom sa $(\Delta y^{(1)}, \Delta v^{(1)})$ dobijemo približno:

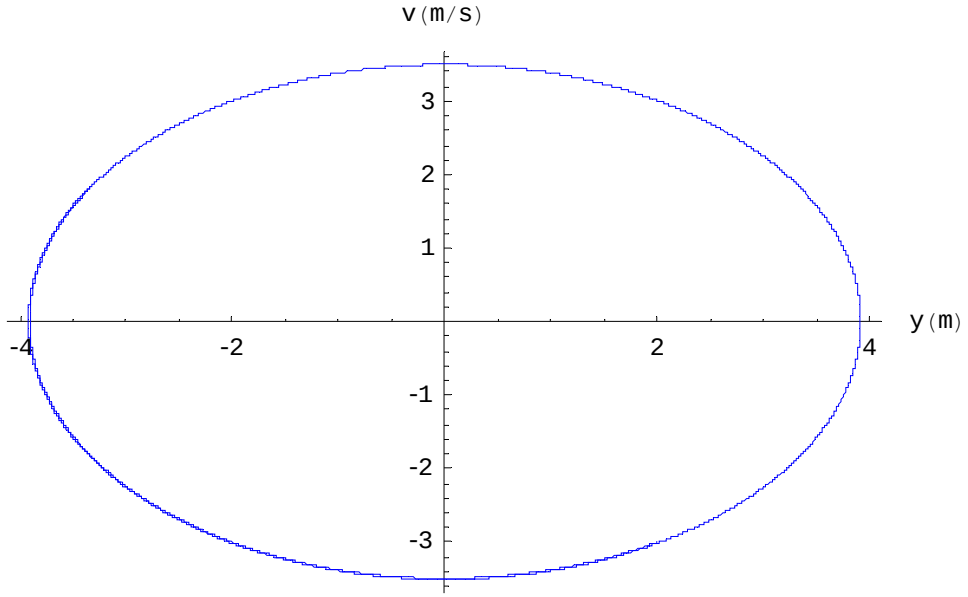
$$(\Delta y^{(2)}, \Delta v^{(2)}) \approx \frac{1}{2}(\Delta y^{(1)}, \Delta v^{(1)})$$

te se može pokazati da je to općenito svojstvo Eulerove metode.

2. Slučaj

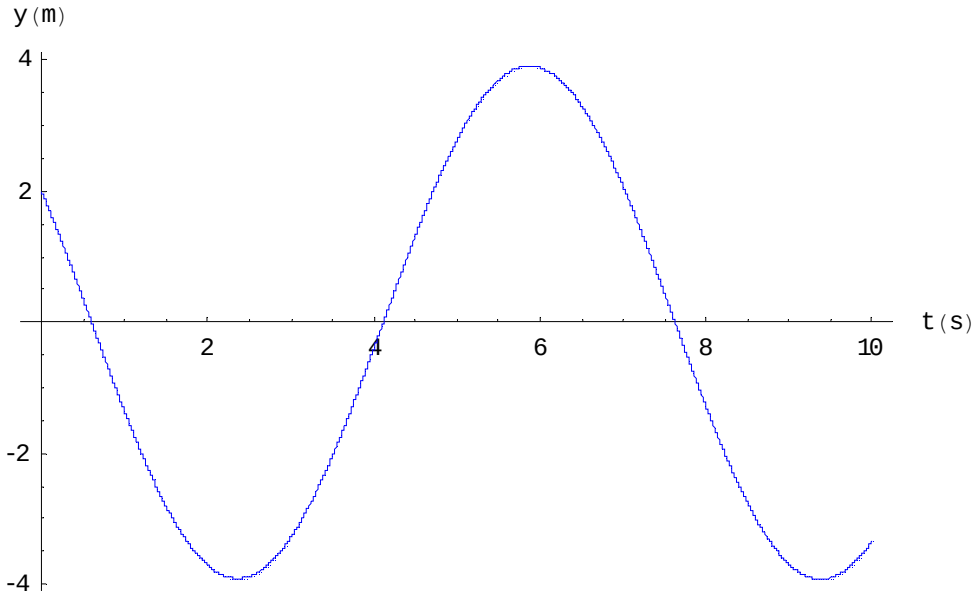
Uzeti ćemo početne uvjete: y(0) = 2 korak $\Delta t = 0.001$
v(0) = -3 broj koraka = 10000

Za ove početne uvjete dobivamo sljedeće slike:

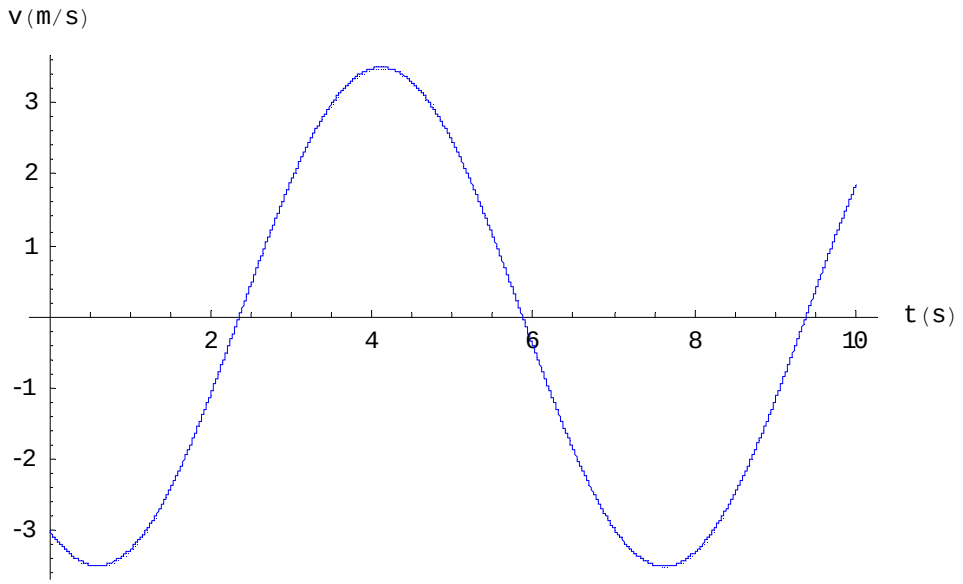


Slika 1.16. Fazni portret sustava sa početnim uvjetom y(0) = 2, v(0) = -3, Δt=0.001, broj koraka = 10000

Vidimo da rješenje opisuje elipsu u faznoj ravnini (Slika 1.16.), ali nešto veću nego u prethodnom slučaju (tj. za prethodne početne uvjete).
Period je jednak onom u prethodnom slučaju $T \approx 7$.



Slika 1.17. y(t) graf rješenja sa početnim uvjetom y(0) = 2, v(0) = -3, Δt=0.001 , broj koraka = 10000



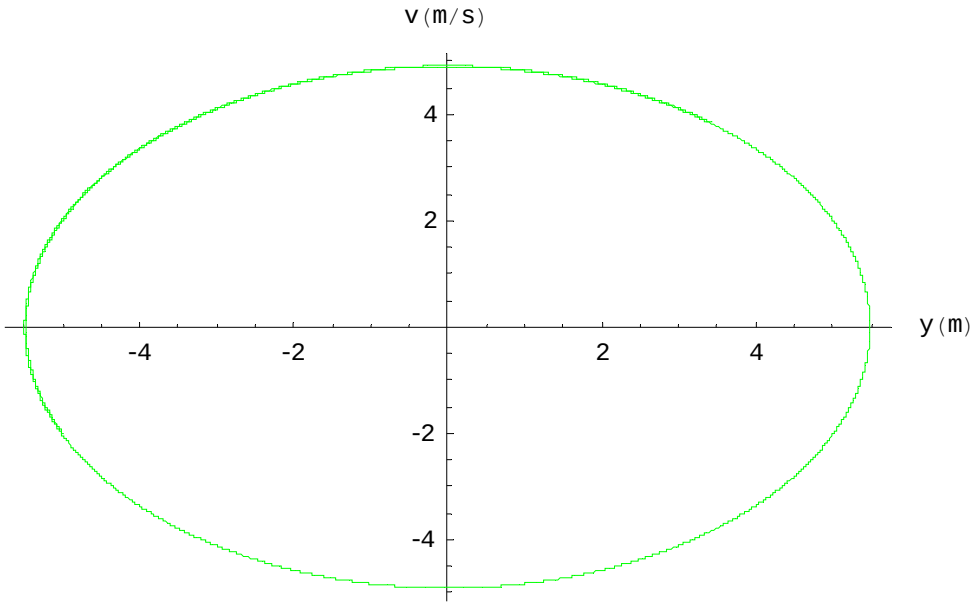
Slika 1.18. v(t) graf rješenja sa početnim uvjetom y(0) = 2, v(0) = -3, Δt=0.001 , broj koraka = 10000

Gornji početni uvjeti (Slika 1.17.) i (Slika1.18.) znače da smo počeli mjeriti u trenutku kad je otklon y_0 pozitivan (ali se dalje smanjuje), a brzina v_0 „negativna“ (i dalje se povećava u istom smjeru – postaje „negativnija“). Ti uvjeti opisuju trenutak u kojem se masa vraća iz maksimalnog otklona, ali još nije dosegla ravnotežni položaj. Brzina će se još povećavati u negativnom smjeru osi y, dok ne dosegne maksimum u trenutku kad masa prolazi kroz ravnotežni položaj.

3. Slučaj

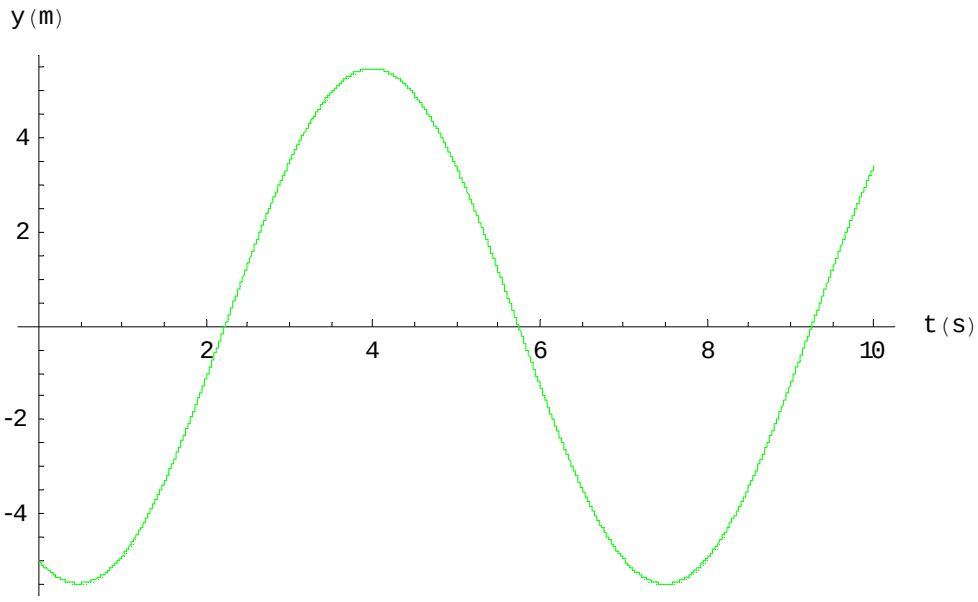
Uzeti ćemo početne uvjete: y(0) = -5 korak $\Delta t = 0.001$
v(0) = -2 broj koraka = 10000

Za ove početne uvjete dobivamo sljedeće slike:

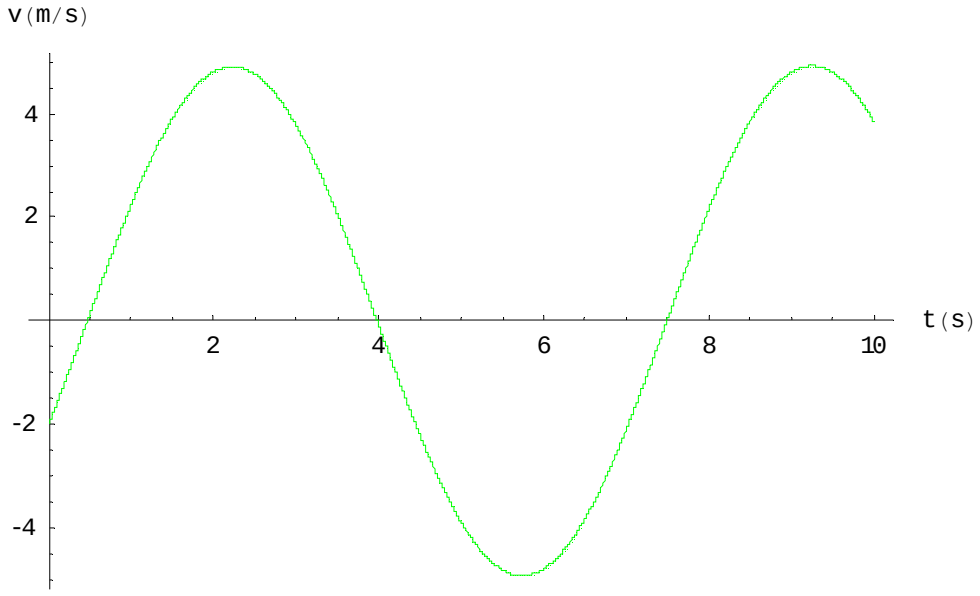


Slika 1.19. Fazni portret sustava sa početnim uvjetom y(0) = -5, v(0) = -2, Δt=0.001, broj koraka = 10000

Za ove početne uvjete vidimo da se elipsa u faznoj ravnini (Slika 1.19.) još povećala.



Slika 1.20. y(t) graf rješenja sa početnim uvjetom y(0) = -5, v(0) = -2, Δt=0.001 , broj koraka = 10000



Slika 1.21. v(t) graf rješenja sa početnim uvjetom y(0) = -5, v(0) = -2, Δt=0.001, broj koraka = 10000

Period opet očitamo iz grafa *y(t)* (Slika 1.20.) i *v(t)* (Slika 1.21.) i on se nije promijenio, te je opet $T \approx 7$.

Ovdje smo počeli mjeriti u trenutku kad je masa vrlo blizu (negativnog) maksimalnog otklona, a brzina skoro jednaka nuli. Dalje će se otklon još malo povećati do maksimalnog negativnog otklona, a zatim smanjivati do ravnotežnog položaja. Brzina će se ponovo smanjivati na nulu, a zatim će promijeniti smjer te povećavati, a u ravnotežnom položaju će biti maksimalna.

Fizikalno značenje početnih uvjeta $y(o) = y_0$ i $v(0) = v_0$ za sva tri slučaja je sljedeće:

y(t) opisuje pomak mase na opruzi od ravnotežnog položaja u određenom trenutku, te je pozitivan za pomak u smjeru osi *y*, a negativan za pomak u smjeru *–y* (sami po volji izaberemo koji je pozitivan smjer osi y). Početni uvjet $y(o) = y_0$ opisuje u kojem je položaju bila masa u trenutku u kojem smo počeli mjerenje.

v(t) opisuje brzinu mase u određenom trenutku i pozitivan je za brzinu u smjeru *y*, a negativan za brzinu u smjeru *–y*. Početni uvjet $v(0) = v_0$ opisuje brzinu mase u početnom trenutku mjerenja.

Sa grafa *y(t)* i *v(t)* da su *y(t)* i *v(t)* pomaknuti u fazi tako da kad *y(t)* poprima maksimalne (apsolutne) vrijednosti, *v(t)* je nula i obrnuto. To odgovara ponašanju harmoničkog oscilatora tj. Tijela na opruzi. Kad je pomak *y(t)* pozitivan i brzina *v(t)* nula, to je trenutak u kojem je pomak maksimalan. Zbog povratne sile javlja se ubrzanje koje je proporcionalno pomaku, ali suprotnog predznaka, tako da vraća tijelo prema ravnotežnom položaju u kojem je pomak nula, a brzina negativna i maksimalna. U ravnotežnom položaju je i slia nula (proporcionalna je pomaku), ali s obzirom da je brzina negativna i maksimalna, zbog inercije se tijelo nastavlja

gibati dok pomak ne postane dovoljno velik i negativan da sila postane dovoljno velika i pozitivna da „vrati“ brzinu na nulu. Tada je opet pomak maksimalan i sila opet djeluje tako da vraća masu prema ravnotežnom položaju.
Taj ciklus se ponavlja: povratna sila „vraća“ pomak na nulu, pritom se pojavi brzina, a inercija „čuva“ brzinu te uzrokuje da se pomak opet poveća, te tako sustav oscilira.

Možemo reći da kad su pomak i brzina istog predznaka, masa se udaljava od ravnotežnog položaja, a kad su suprotnog, masa se kreće prema ravnotežnom položaju.

Za harmonički oscilator vrijedi da je prosječna vrijednost kinetičke energije jednaka prosječnoj vrijednosti potencijalne energije, a ukupna energija je konstanta gibanja i jednaka je prosjeku ukupne energije, tj. dvostrukoj vrijednosti prosječne kinetičke ili potencijalne energije.

Jednadžbu $\frac{d^2y}{dt^2} + \omega^2 y = 0$ možemo analitički riješiti tako da riješimo pripadnu karakterističnu jednadžbu $r^2 + \omega^2 = 0$. Dobijemo kompleksna rješenja $r = \pm i\omega$, što znači da rješenja diferencijalne jednadžbe imaju oblik $y(t) = C_1 \cos \omega t + C_2 \sin \omega t$. Kad uvrstimo $\omega^2 = 0.8$, te početne uvjete za prvi slučaj y(0) = 1, v(0) = 1 dobijemo integracijske konstante $C_1 = 1$ i $C_2 = \frac{1}{\omega} \approx 1.118$.

Za period harmoničkog oscilatora vrijedi $T = \frac{2\pi}{\omega}$, pa dobijemo $T = 7.025$ što se izvrsno slaže s našim očitavanjem iz grafa $T \approx 7$.

2. Harmonički oscilator s gušenjem

2.1. Uvod

Do sada smo promatrali oscilacije sistema uz pretpostavku da se potencijalna energija potpuno pretvarala u kinetičku energiju i obrnuto. Sistem sa ovakvim svojstvima, jednom pobuđen, oscilirao bi beskonačno dugo. Međutim, iz iskustva znamo, da slobodne oscilacije nekog realnog fizikalnog sistema zamiru nakon nekog vremena. To se događa zato što pretvaranje potencijalne energije opruge u kinetičku energiju tijela i obrnuto, nije potpuno. Jedan dio mehaničke energije pretvara se u toplinu, zbog trenja i zbog deformacije dijelova u gibanju, kao npr. zavoja kod opruge. Raspoloživa mehanička energija tako postaje sve manja dok konačno ne padne na nulu. Tada oscilacije prestaju.

Kažemo da je harmoničko titranje *gušeno*, ako mu amplituda monotono opada s vremenom.

U slučaju kada sila trenja nije prevelika tj. kada brzina nije prevelika, vanjska sila koja djeluje na masu i guši titranje, proporcionalna je brzini:

$$F = -bv \tag{2.1.}$$

Ovakva sila otpora ima smjer suprotan od smjera brzine.

2.2. Jednadžba harmoničkog titranja s gušenjem

U ovom slučaju drugi Newtonov zakon možemo pisati u obliku:

$$m\frac{d^2x}{dt^2} = -kx - bv \tag{2.2.}$$

Jednadžba gibanja je onda:

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0 \tag{2.3.}$$

ili ako uvedemo oznake:

$$\gamma = \frac{b}{m} \quad , \quad \omega_0^2 = \frac{k}{m} \tag{2.4.}$$

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \omega_0^2 x = 0 \tag{2.5.}$$

Rješimo sada ovu jednadžbu gibanja. Pretpostavimo da je pomak x realni dio rotirajućeg vektora z koji zadovoljava sličnu jednadžbu:

$$\frac{d^2z}{dt^2} + \gamma\frac{dz}{dt} + \omega_0^2 z = 0 \tag{2.6.}$$

Pretpostavimo da je rješenje oblika:

$$z = Ae^{i(pt+\alpha)} \tag{2.7.}$$

I ovdje kao i kod negušenog harmoničkog oscilatora postoje dvije konstante A i α koje su određene početnim uvjetima.

Uvrstimo li (2.7.) u jednadžbu (2.6.) slijedi:

$$-Ap^2e^{i(pt+\alpha)} + \gamma Aipe^{i(pt+\alpha)} + \omega_0^2 Ae^{i(pt+\alpha)} = 0 \tag{2.8.}$$

odnosno:

$$(-p^2 + ip\gamma + \omega_0^2)Ae^{i(pt+\alpha)} = 0 \tag{2.9.}$$

Ako je ova jednadžba zadovoljena za sve vrijednosti od t, onda mora biti:

$$-p^2 + ip\gamma + \omega_0^2 = 0 \tag{2.10.}$$

Budući da ovaj uvjet sadrži kompleksne brojeve, možemo smatrati da su to u stvari dva uvjeta, po jedan za realne i kompleksne komponente odvojeno. Uvjet (2.10.) ne može biti zadovoljen ako je p realna veličina (jer bi onda član $ip\gamma$ bio jedini imaginarni član koji se nema čime poništiti). Zato stavimo da je:

$$p = n + is \tag{2.11.}$$

gdje su n i s realne veličine. Onda je:

$$p^2 = n^2 + 2ins - s^2 \tag{2.12.}$$

Uvrstimo li ovo u (2.10.), dobivamo:

$$-n^2 - 2ins + s^2 + in\gamma - s\gamma + \omega^2 = 0 \tag{2.13.}$$

Odvojimo li realne komponente od imaginarnih, slijede dva uvjeta:

$$\begin{aligned} &\text{i)} \quad -n^2 + s^2 - s\gamma + \omega_0^2 = 0 \\ &\text{ii)} \quad -2ns + n\gamma = 0 \end{aligned} \tag{2.14.}$$

Iz (2.14. ii) dobivamo:

$$s = \frac{\gamma}{2} \tag{2.15.}$$

uvrstimo li to u (2.13. i) slijedi:

$$n^2 = \omega_0^2 - \frac{\gamma^2}{4} \tag{2.16.}$$

Pišemo li u (2.7.) $n + is$ umjesto p, onda slijedi:

$$z = Ae^{i(nt+ist+\alpha)}$$

$$z = Ae^{-st}e^{i(nt+\alpha)} \tag{2.17.}$$

Budući da je x realna komponenta od z, onda je:

$$x = Ae^{-st}\cos(nt + \alpha) \tag{2.18.}$$

Uvrstimo li sada u (2.18.) dobivene rezultate za n i s, dobivamo:

$$x = Ae^{\frac{-\gamma t}{2}}\cos(\omega t + \alpha) \tag{2.19.}$$

gdje je:

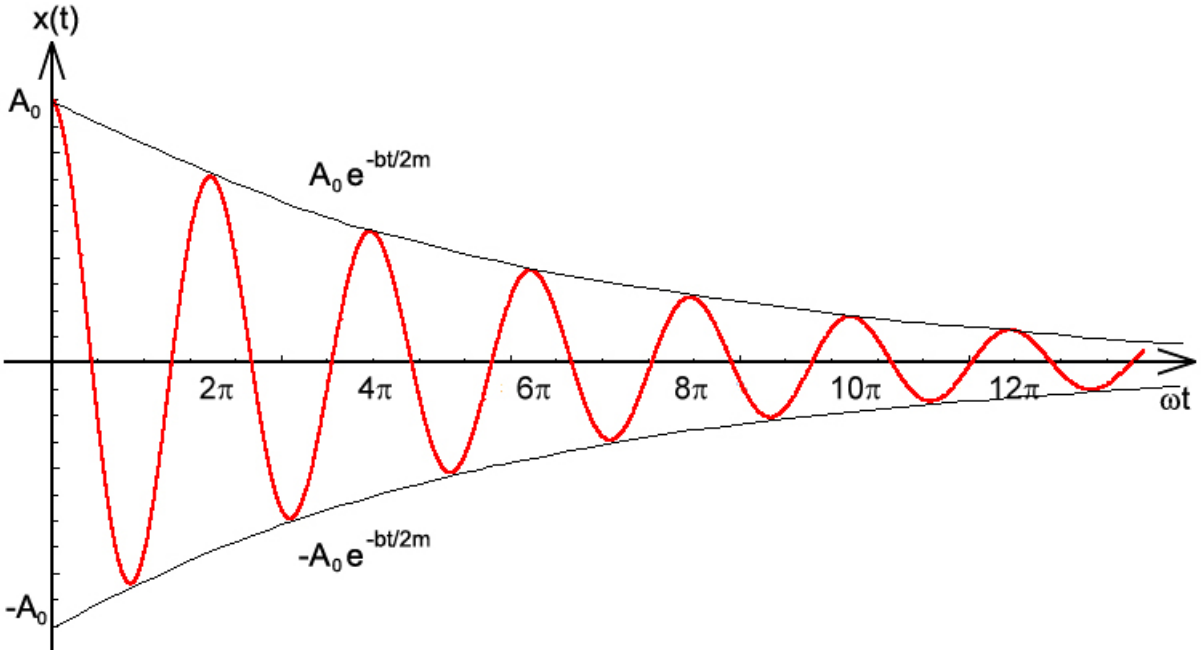
$$\omega^2 = n^2 = \omega_0^2 - \frac{\gamma^2}{4} = \frac{k}{m} - \frac{b^2}{4m^2} \tag{2.20.}$$

Svedimo rješenje (2.19.) na oblik rješenja kao kod negušenog harmoničkog oscilatora:

$$x = A(t)\cos(\omega t + \alpha) \tag{2.21.}$$

Onda je za $\gamma << \omega$:

$$A(t) = A_0 e^{\frac{-\gamma t}{2}} \tag{2.22.}$$



Slika 2.1. Slučaj kada je α=0

Na slici 2.1. prikazana je jednadžba (2.19.) u slučaju kada je $\alpha = 0$. Razmak između nula krivulje je stalan i iznosi $\omega\Delta t = \pi$. Envelopa amplituda ima dvije grane koje se asimptotski približavaju nuli. Iako gušenje povećava period u odnosu na negušeni oscilator ipak je interval vremena između dva prolaza kroz ishodište, u istom smjeru i ovdje stalan. Taj interval vremena obično zovemo *period gušenih oscilacija*. On dakle, ostaje konstantan za neko titranje iako se amplituda stalno smanjuje.

2.3. Energija

Ukupna mehanička energija harmoničkog oscilatora dana je jednadžbom (1.28.):

$$E = \frac{1}{2}kA^2$$

Uvrstimo li u taj izraz amplitudu (2.22.) gušenih harmoničkih oscilacija, onda je:

$$E(t) = \frac{1}{2}kA_0^2 e^{-\gamma t} \tag{2.23.}$$

tj.

$$E(t) = E_0 e^{-\gamma t} \tag{2.24.}$$

Ako smo stavili da je:

$$E_0 = \frac{1}{2} k A_0^2$$

Vidimo da ukupna energija gušenog harmoničkog oscilatora eksponencijalno trne s vremenom.

2.4. Q-faktor

Gušene oscilacije, kao što smo vidjeli, karakterizirane su sa dva parametra ω_0 i $\gamma = \frac{b}{m}$. Konstanta ω_0 je kružna frekvencija negušenih oscilacija, a γ veličina recipročna vremenu potrebnom da energija padne na e-ti dio svoje početne vrijednosti. To su ω_0 i γ veličine istih dimenzija.

Definirajmo pomoću ovih parametara još jedan parametar, koji ćemo zvati *Q-faktor*:

$$Q = \frac{\omega_0}{\gamma} \tag{2.25.}$$

Q je bezdimenzionalna veličina, čija je vrijednost usporediva s jedinicom za oscilatorne sisteme, sa malim rasipanjem energija.

Jednadžba (2.20.) onda postaje:

$$\omega^2 = \omega_0^2 (1 - \frac{1}{4Q^2}) \tag{2.26.}$$

Ako je $\omega \approx \omega_0$ gibanje oscilatora vrlo dobro je opisano sa:

$$x = A_0 e^{-\frac{\omega_0 t}{2Q}} \cos(\omega_0 t + \alpha) \tag{2.27.}$$

Faktor Q povezan je sa brojem oscilacija nakon kojeg amplituda pada ne e-ti dio svoje početne vrijednosti. Jednadžba (2.22.) onda postaje:

$$A(t) = A_0 e^{-\frac{\omega_0 t}{2Q}} \tag{2.28.}$$

Ako vrijeme t mjerimo kao funkciju broja kompleksnih ciklusa, n, onda u aproksimaciji možemo staviti:

$$t \approx \frac{2\pi n}{\omega_0} \tag{2.29.}$$

pa umjesto (2.28.) imamo:

$$A(n) \approx A_0 e^{-\frac{n\pi}{Q}} \tag{2.30.}$$

tj. amplituda pada na svoj e-ti dio početne vrijednosti u otprilike $\frac{Q}{\pi}$ -tom ciklusu.

I jednadžbu gibanja (2.5.) sada možemo pisati preko veličina ω_0 i Q:

$$\frac{d^2x}{dt^2} + \frac{\omega_0}{Q}\frac{dx}{dt} + \omega_0^2 x = 0 \tag{2.31.}$$

Za mnoge fizikalne sisteme ovaj oblik jednadžbe gibanja biti će pogodniji od onog osnovnog za slobodne oscilacije kada još moramo uračunati i gušenje.

2.5. Primjer rješenja harmoničkog osciliranja s gušenjem

U ovom primjeru proučavati ćemo jednadžbu:

$$m\frac{d^2y}{dt^2} + b\frac{dy}{dt} + ky = 0$$

U jednadžbi postoje sljedeće veličine:

m – masa tijela (parametar)
y – pomak iz ravnotežnog položaja (zavisna varijabla)
t – vrijeme (nezavisna varijabla)
b – faktor gušenja (parametar)
k – konstanta opruge (parametar)

Također ćemo uzeti da je m = 5, k = 4 i b = 4, što znači da je u sustavu prisutno gušenje, koje je proporcionalno brzini.

Jednadžbu možemo napisati kao sustav dvije vezane autonomne diferencijalne jednadžbe prvog reda:

$$\frac{dy}{dt} = v$$

$$\frac{dv}{dt} = -\omega^2 y - \lambda v$$

gdje je:

$$\omega^2 = \frac{k}{m} = \frac{4}{5}, \text{ a } \lambda = \frac{b}{m} = \frac{4}{5}$$

U ovom slučaju jednadžbe imaju oblik:

$$\frac{dy}{dt} = v$$

$$\frac{dv}{dt} = -0.8y - \frac{4}{5}v$$

Iz jednadžbi vidimo da ubrzanje djeluje tako da je suprotno brzini, te je usporava.

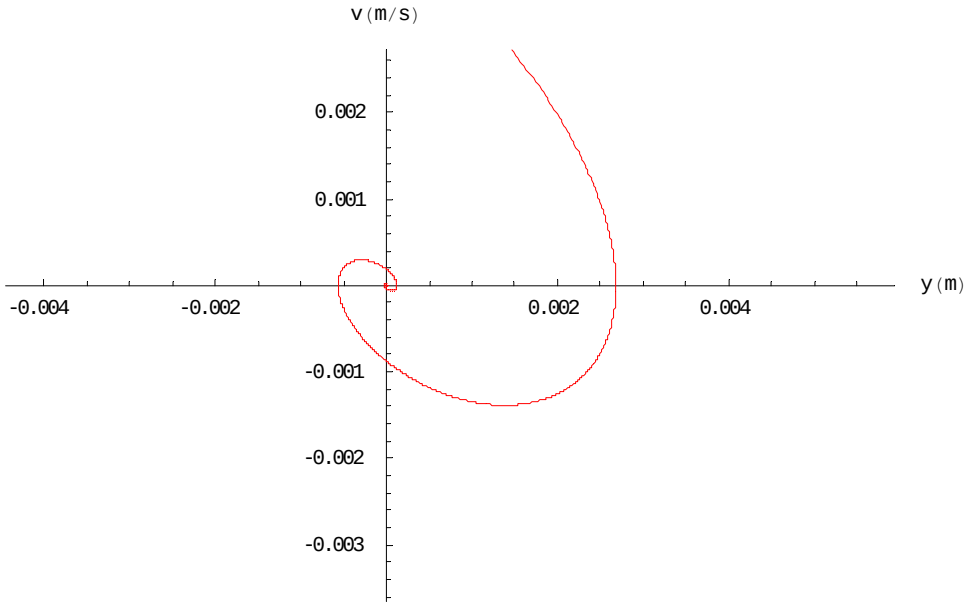
Za rješavanje ćemo koristiti numerički postupak, npr. Eulerovu metodu.

Uzimati ćemo iste početne uvjete kao u prvom primjeru.

1. slučaj

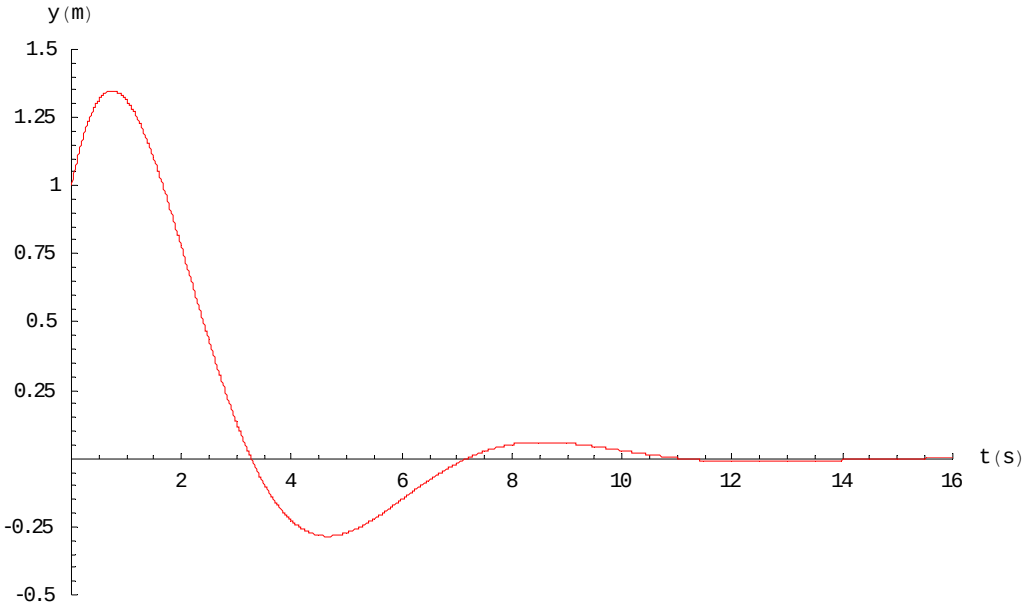
Uzeti ćemo početne uvjete: y(0) = 1 korak $\Delta t = 0.01$
v(0) = 1 broj koraka = 5000

Za ove početne uvjete dobivamo sljedeće slike:

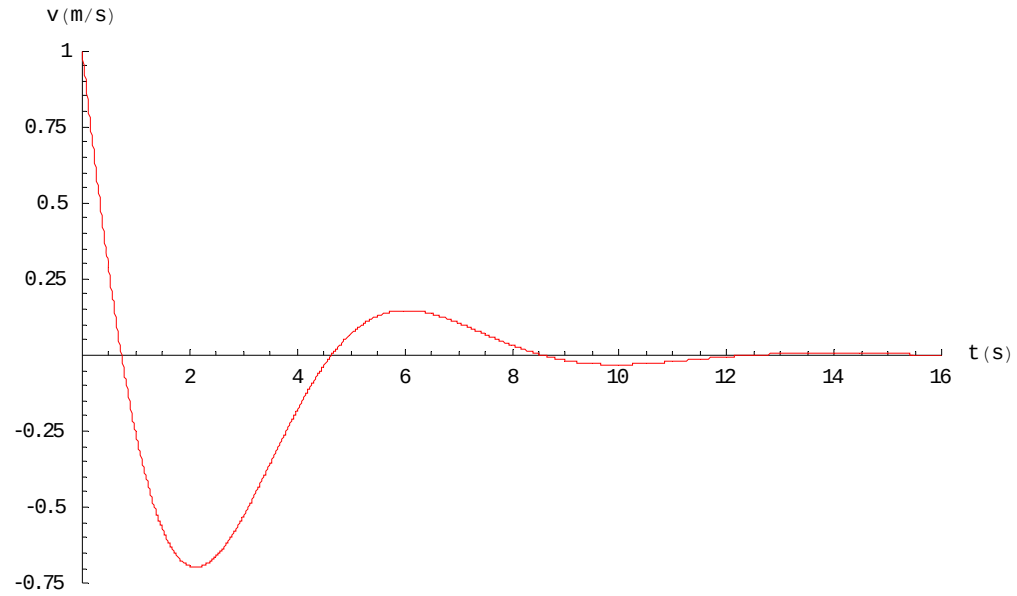


Slika 2.3. Fazni portret sustava sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.01, broj koraka = 5000

Iz faznog portreta (Slika 2.3.) vidimo da se rješenje asimptotski približava nuli. Koristili smo manji korak jer smo i za njega dobili dobre rezultate.



Slika 2.4. y(t) graf rješenja sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.01, broj koraka = 5000



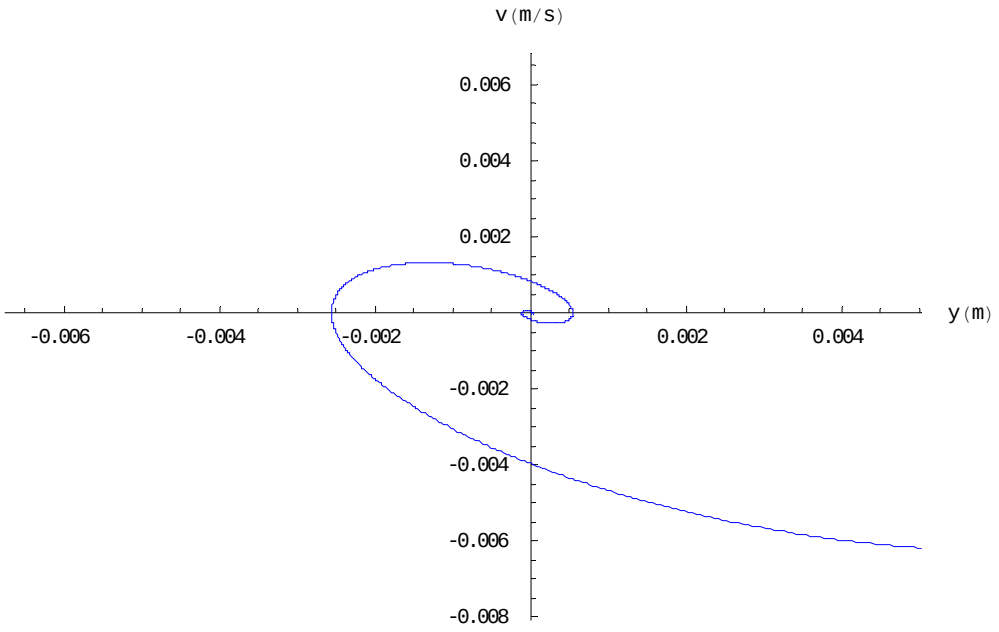
Slika 2.5. v(t) graf rješenja sa početnim uvjetom y(0) = 1, v(0) = 1, Δt=0.01, broj koraka = 5000

Iz gornjih grafova (Slika 2.4.) i (Slika 2.5.) vidimo da se amplituda brzo smanjuje i teži nuli. Period oscilacija se čini da je približno isti kao u slučaju bez gušenja, te iznosi $T \approx 7.5$

2. slučaj

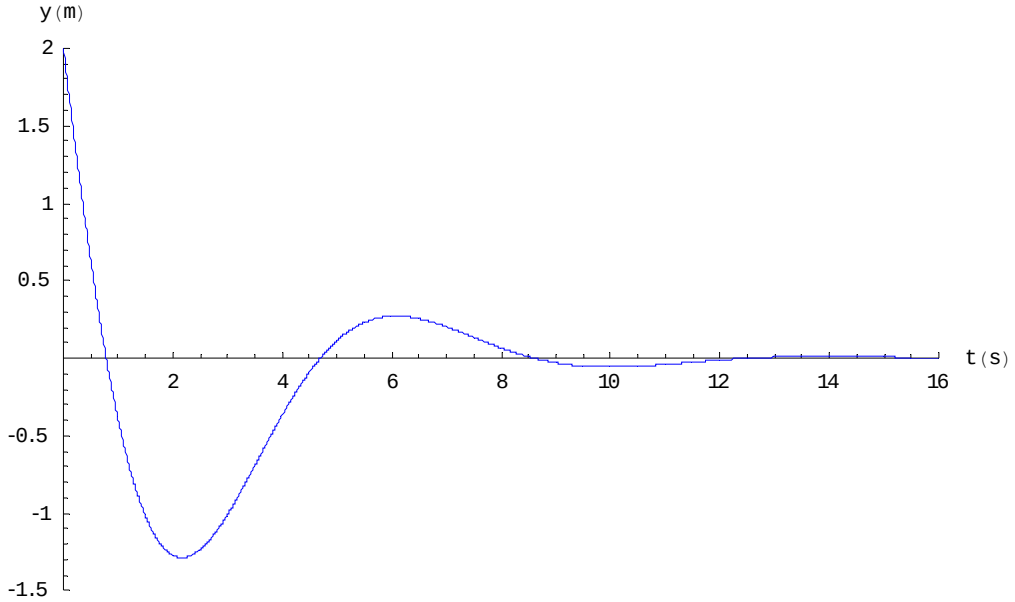
Uzeti ćemo početne uvjete: y(0) = 2 korak $\Delta t = 0.01$
v(0) = -3 broj koraka = 5000

Za ove početne uvjete dobivamo sljedeće slike:

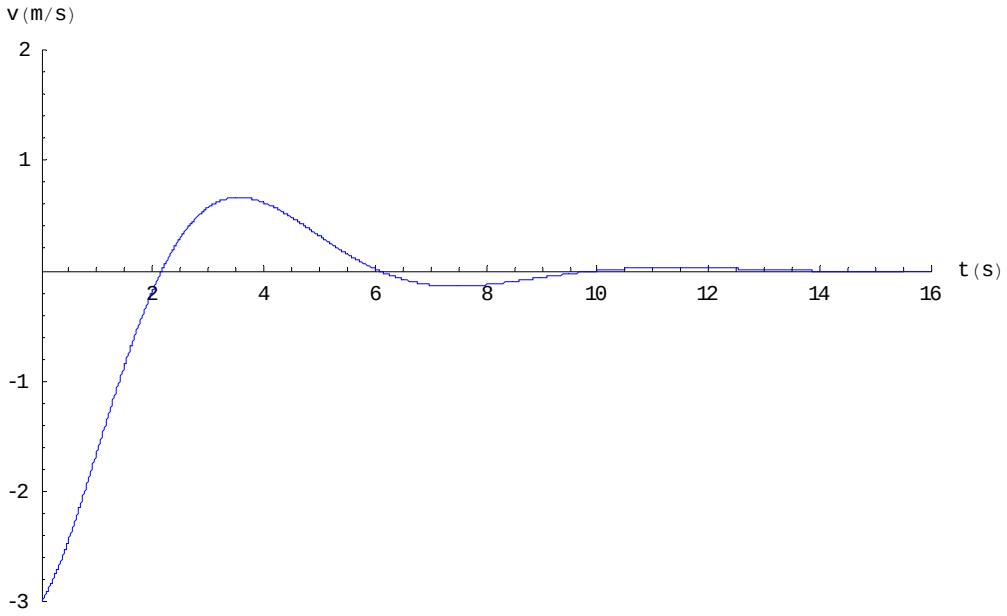


Slika 2.6. Fazni portret sustava sa početnim uvjetom y(0) = 2, v(0) = -3, Δt=0.01, broj koraka = 5000

Ovaj slučaj (Slika 2.6.) počinje sa drugog mjesta u faznoj ravnini, ali također brzo ide u nulu.



Slika 2.7. y(t) graf rješenja sa početnim uvjetom y(0) = 2, v(0) = -3, Δt=0.01, broj koraka = 5000



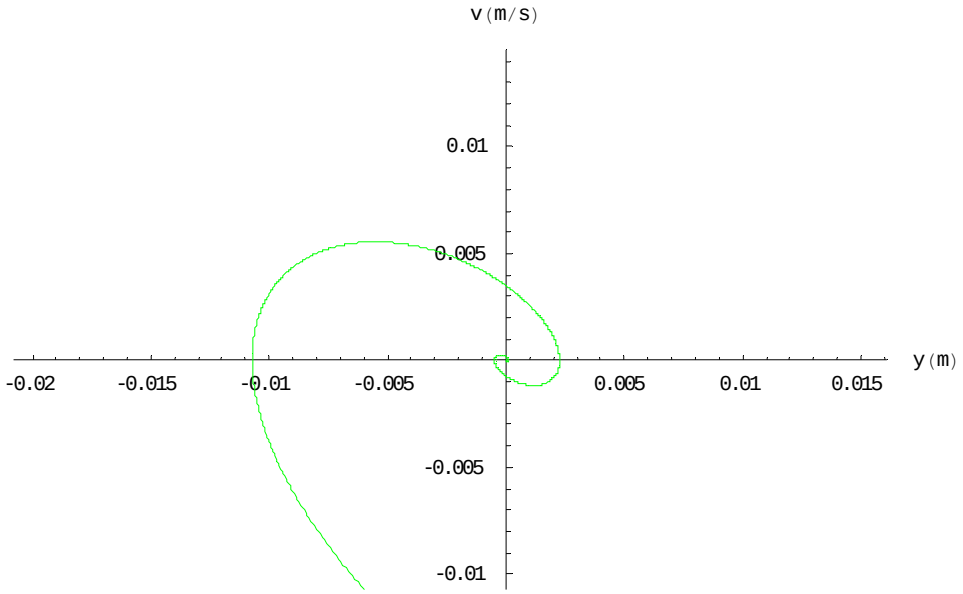
Slika 2.8. v(t) graf rješenja sa početnim uvjetom y(0) = 2, v(0) = -3, Δt=0.01, broj koraka = 5000

Druga dva grafa (Slika 2.7.) i (Slika 2.8.) nam potvrđuju da se amplituda brzo smanjuje ka nuli, kao i u prvom slučaju, iako su početni uvjeti drukčiji.

3. slucaj

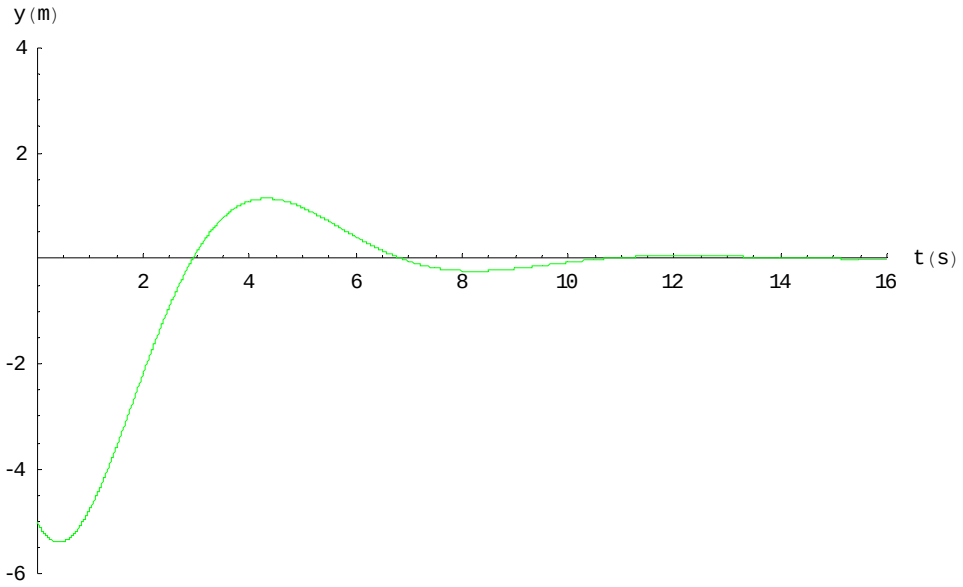
Uzeti ćemo početne uvjete: y(0) = -5 korak $\Delta t = 0.01$
v(0) = -2 broj koraka = 5000

Za ove početne uvjete dobivamo sljedeće slike:

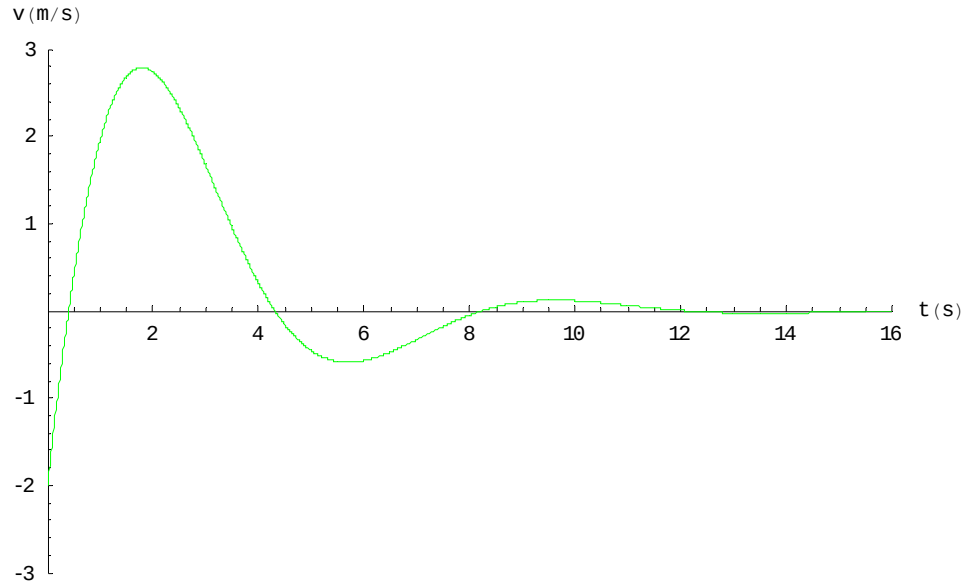


Slika 2.9. Fazni portret sustava sa početnim uvjetom y(0) = -5, v(0) = -2, Δt=0.01, broj koraka = 5000

Za treći slučaj vidimo isto ponašanje kao za prva dva (Slika 2.9.), samo što opet kreće s drugog mjesta u faznom portretu.



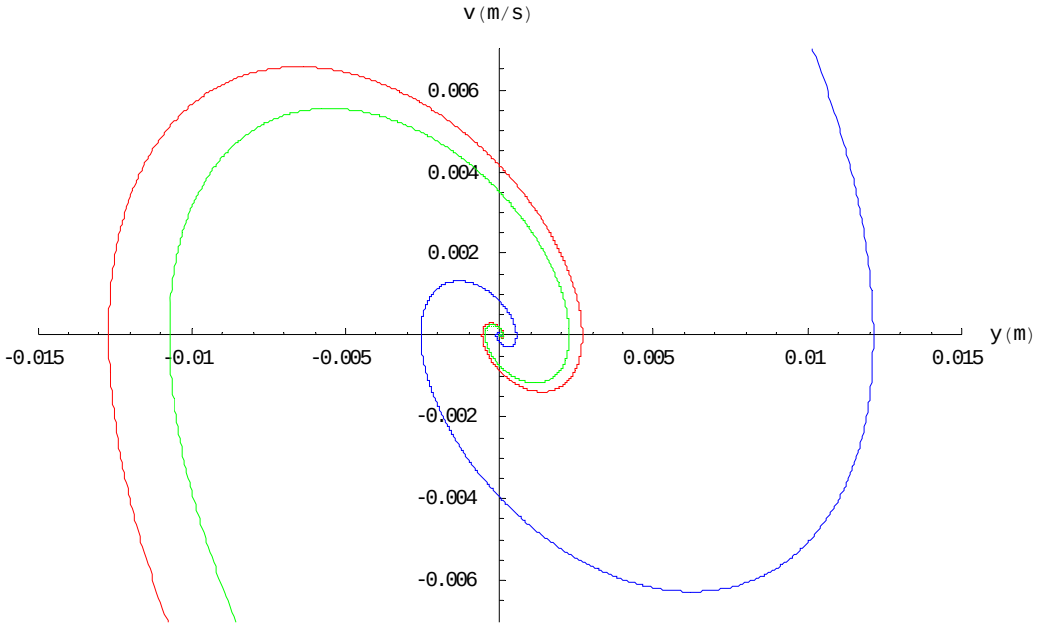
Slika 2.10. y(t) graf rješenja sa početnim uvjetom y(0) = -5, v(0) = -2, Δt=0.01, broj koraka = 5000



Slika 2.11. v(t) graf rješenja sa početnim uvjetom y(0) = -5, v(0) = -2, Δt=0.01, broj koraka = 5000

Druga dva grafa (Slika 2.10.) i (Slika 2.11.) nam potvrđuju da se amplituda također brzo smanjuje ka nuli, kao u prethodna dva slučaja, iako su početni uvjeti drukčiji.

Ako prikažemo sva tri slučaja na jednom faznom portretu (Slika 2.12.) , točno se vidi od kuda se počinju kretati:



Slika 2.12. Fazni portret prvog, drugog i trećeg slučaja

Možemo zaključiti da sva tri slučaja opisuju natkritično gušenje harmoničkog oscilatora, tj. slučaj kad na masu koja se giba djeluje sila koja ga usporava (trenje).

Jednadžba $m\frac{d^2y}{dt^2}+b\frac{dy}{dt}+ky=0$ se može analitički riješiti tako da riješimo karakterističnu jednadžbu $mr^2+br+k=0$. Za naš slučaj dobivamo rješenja: $r_{1,2}=-0.2\pm 0.4i$, a rješenje ima oblik $y(t)=e^{-0.2t}(C_1\sin 0.4t+C_2\cos 0.4t)$. Eksponencijalni član na početku opisuje gušenje, tj. kako se amplituda i brzina smanjuju s vremenom. Sada se vidi zašto je to smanjenje eksponencijalno. Konstante C_1 i C_2 možemo dobiti iz početnih uvjeta i različite su za različite početne uvjete.

3. Programski jezik Java

3.1. Uvod

Zamislite da ste programer aplikacija. Programski jezik u kojem programirate je C ili C++. U posljednjih nekoliko godina u svijetu računalstva se pojavi mnoštvo inkompatibilnih računalskih arhitektura i još više operativnih sustava. Zadaća Vam je da razvijete aplikaciju koja će se izvršavati u svim tim okruženjima. Shvatili ste da programski jezici C i C++ koje imate na raspolaganju i nisu baš od neke koristi u ovoj situaciji. Kažete sami sebi: „postoji li bolji način?“ I zaista postoji bolji način – to je Java.

Krajem 1990. skupina programera iz tvrtke Sun Microsystem pod vodstvom Jamesa Goslinga i Billa Joya krenula je na kreiranje jedinstvenog programskog jezika. Cilj im je bio napraviti programski jezik čije će se aplikacije moći izvršavati na računalima različitih proizvođača i pod različitim operativnim sustavima. To znači da će se aplikacija napisana u tom programskom jeziku moći izvršavati na PC-u, Mac-u i pod različitim operativnim sistemima kao što su Unix, Linux, Windows itd. Nakon pet uspjeli su ostvariti svoj cilj, kreiran je novi programski jezik pod imenom Java. I sami smo svjedoci da im je u tome uspjelo. Java aplikacije se danas na mobilnim uređajima, Internetu itd.

3.2. Karakteristike Jave

Kako bi smo se upoznali sa karakteristikama Jave nabrojiti ćemo neke od njenih glavnih karakteristika koje će pokazati da Java jest „bolji način“. Pa krenimo redom:

- **Sličnost i jednostavnost**

Ove dvije karakteristike možete i osobno primijeniti ako se bez ikakvog predznanja upustite programirati u Javi. Ono što ćete odmah zaključiti je da je Java vrlo jednostavna, ukoliko ste prije programirali u C i C++, uočiti ćete sličnost u sintaksi Jave i C ili C++.

- **Robusnost**

Osobina koja čini Javu robusnom je nepostojanje pokazivača, na taj način je onemogućeno pretrpavanje memorije ili slučajno ulaženje u tuđi memorijski prostor. Naime, izvedbeni dio Jave se brine o memoriji npr. oslobađanje memorije se vrši automatski, ta tehnika se naziva „automatic garbage collection“.

- **Nezavisnost strojnoj platformi**

Aplikacija napisana u Javi prenosiva je među raznim strojnim platformama. Dovoljno je jednom napisati programski kod aplikacije i možete ga izvoditi bez ikakvih izmjena u kodu na raznim strojnim platformama i pod raznim operativnim sustavima.

• **Performanse i višenitnost**

Java podržava višenitnost i time omogućava brže izvođenje paralelnih aktivnosti u aplikaciji (npr. pomicanje nekog objekta i istovremeno sviranje glazbe na web pregledniku). Također postoji mogućnost pisanja dijelova aplikacije u C-u čime se dobiva na brzini. Performansama pridonosi već prije spomenuta tehnika „automatic garbage collection“, koja omogućava upravljanje memorijom kada je to aplikaciji potrebno.

• **Sigurnost**

Java nudi sigurnost zato jer ima ugrađene mehanizme za zaštitu od virusa i drugih bugova. Neki od mehanizama su:

- računalna memorija nije izravno dostupna ni Java programu
- provjera bajt-kodova da bi se ustanovile neželjene izmjene

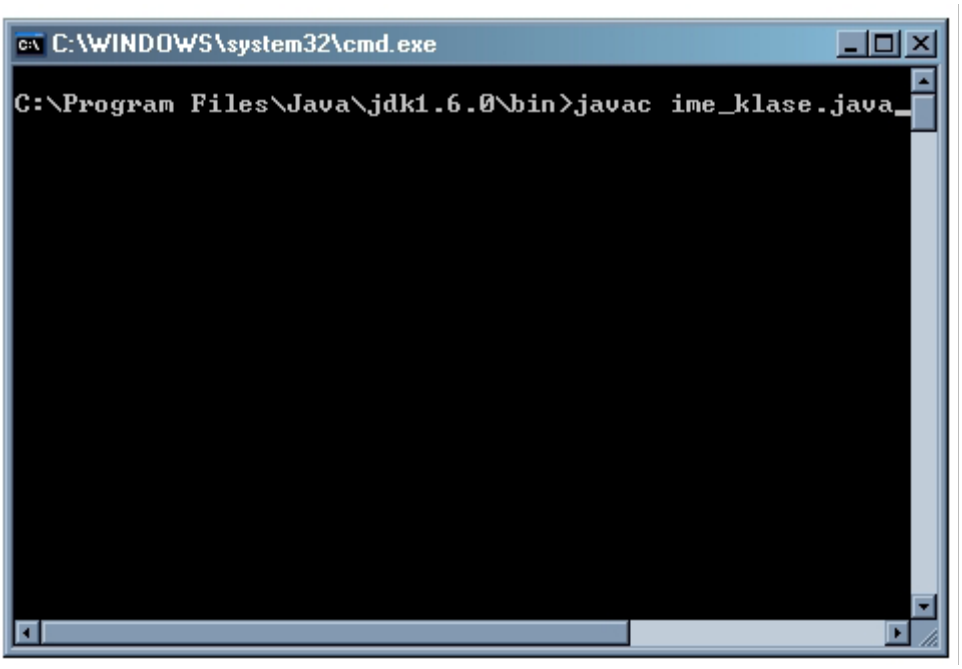
3.3. Izvršavanje programa

Već smo prije napomenuli da su programi napisani u Javi prenosivi među raznim strojnim platformama. To je ostvarivo zato što se Java programi prevode pa interpretiraju.
Objasnimo sada postupak prevođenje-interpretiranje u Javi. Izvorni kod (source code) se prvo prevodi (kompajlira) u tzv. *bajt-kod* (bytcode) zatim slijedi interpretacija koju provodi Java virtualna mašina (Java VM). Java VM se nalazi u sklopu *Java Runtime Enviromenta* (Java izvedbena okolina-JRE). Za svaki operativni sistem postoji različita Java izvedbena okolina (npr. za Linux, Windows, Solaris itd.). Java izvedbena okolina provjerava primljeni bajt-kod te ga zatim Java VM interpretira. Takav princip (prevođenje-interpretiranje) omogućava Java aplikacijama neovisnost o strojnoj patformi. Izvorni kod u Javi možemo prevesti u bajt-kod npr. pod Unixom, a zatim taj bajt-kod interpretirati pod Windowsima.
Kod ostalih jezika npr. C, C++ izvorni kod se direktno prevodi u strojni jezik karakterističan za dani operativni sistem. Izvorni kod napisan u C ili C++ je djelomično prenosiv, pogotovo ako se u kodu programiraju mrežni sistemi. Dok su Java izvorni kodovi potpuno prenosivi, dakle: „Napiši jednom, vrti svugdje“.

3.4. Java development Kit (JDK)

Da bi smo mogli pokrenuti pod našim operativnim sustavom aplikaciju napisanu u Javi, moramo imati instaliran JRE (Java Runtime Enviroment). JRE u sebi sadrži Java VM koja omogućava interpretaciju bajt-koda.
Ukoliko pak želimo prevoditi izvorne kodove napisane u Javi, potrebno nam je instalirati JDK. To je razvojno okruženje koje u sebi uključuje alate koji nam omogućavaju razvoj, prevođenje i interpretiranje Java aplikacija. Razvojni alati se nalaze u poddirektoriju bin i pokreću se iz komandne linije (komandnog promta), Slika 3.1.
Nabrojimo neke od njih:

1) javac.exe-služi za prevođenje izvornog koda u Javi u bajt-kod. Izvorni kod u Javi je oblika *ime_klase.java*, a kad se prevede u bajt-kod dobiva ekstenziju class, dakle *ime_klase.class*



Slika 3.1. Prevođenje Java programa u komandnom promtu

2) java.exe-služi za interpretaciju Java bajt-koda, izvršava bajt-kod aplikacije.
3) appletviewer.exe-služi za izvršavanje bajt-koda apleta.

3.5. Objektno-orijentirano programiranje

Java je objektno orijentiran programski jezik. Temelj objektno-orijentiranog programiranja su *klase* i *objekti*. Svugdje oko nas su objekti npr. automobili, zgrade, drveća itd. Ako promatramo automobile, misleći pritom na sve automobile, a ne na neki konkretni automobil, tada nam ti automobili predstavljaju *klasu*. Suzimo sada naš izbor na Hondu Accord. Ako saznamo sve detalje o tom automobilu (godina proizvodnje, jačina motora, vrsta motora, ime vlasnika) znamo o kojem se konkretno automobilu radi, te nam to vozilo tada predstavlja *objekt*. Dakle, *objekt nam predstavlja konkretizaciju klase, a klasa je skup mnogo objekata iste vrste*. Klase mogu imati neka *svojstva* npr. kod klase automobil svojstva bi bila: marka, tip, jačina motora, maksimalna brzina itd. Klase također mogu imate i *metode*, tj. radnje koje će se nad njima izvršiti, npr. za klasu automobila: dodaj gas, promijeni brzinu, promijeni ulje itd. Sljedeći važan pojam unutar klase je *konstruktor*. Konstruktor je metoda unutar klase koja ima isto ime kao i klasa. Zadaća mu je incijaliziranje svojstva objekta. Navedimo još jednu važnu karakteristiku, a to je *nasljeđivanje klasa*. Nasljeđivanje klasa se koristi koko bi se izbjeglo višestruko pisanje istih dijelova koda.

3.6. Java applet

Java applet je java program koji se izvršava unutar Internet preglednika. Naravno da bi mogli izvršiti applet na računalu mora biti instaliran JRE koji u sebi sadrži Java VM-interpreter, koji služi za interpretaciju bajt-koda.

3.6.1. Prevođenje Java appleta

Prevođenje Java appleta se vrši kao i prevođenje drugih java programa. Na računalu mora biti instaliran JDK koji u sebi sadrži razvojne alate za izradu java programa. Da bi smo preveli neki izvorni kod u Javi potrebno je u komandnom promtu napisati naredbu:

```
Javac ime_appleta.java
```

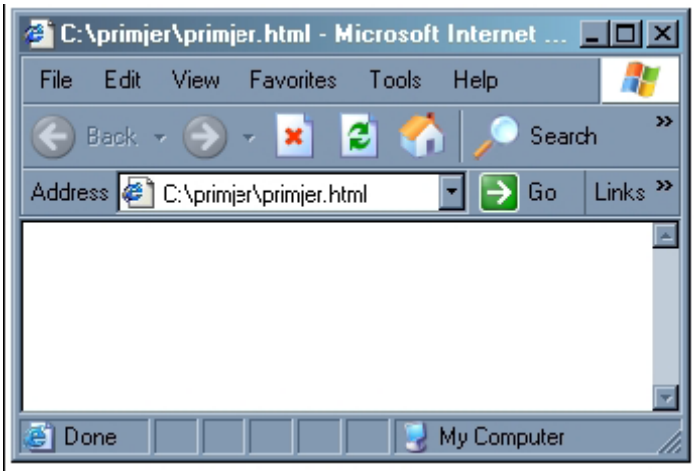
Javac.exe je Java prevodilac koji prevodi izvorni kod u Javi u bajt.kod. Prevođenjem izvornog koda *ime_appleta.java* dobijemo bajt-kod *ime_appleta.class*

3.6.2. Pokretanje Java appleta

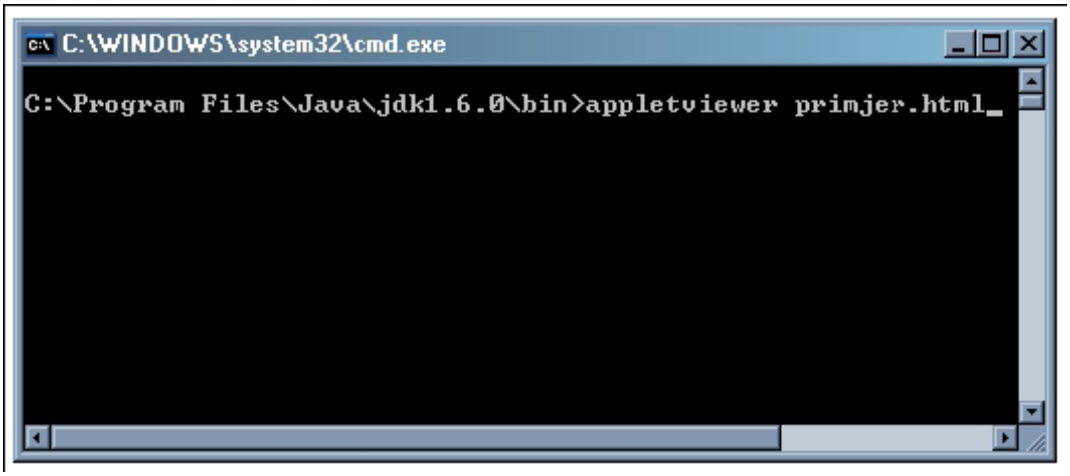
Da bi smo pokrenuli applet potrebno je napisati HTML datoteku iz koje se pokreće Java applet. Primjer HTML datoteke sa Java appletom:

```
<html>
<body>
<applet code=ime_appleta.class width“304“ height=“304“>
</applet>
</body>
</html>
```

Spremimo sada HTML datoteku pod imenom *primjer.html*. Pokretanje appleta se može vršiti na dva načina:
1) pomoći Internet preglednika (Slika 3.2.)
2) korištenjem *appletviewer.exe* koji se nalazi u sklopu JDK razvojnih alata (Slika 3.3.)



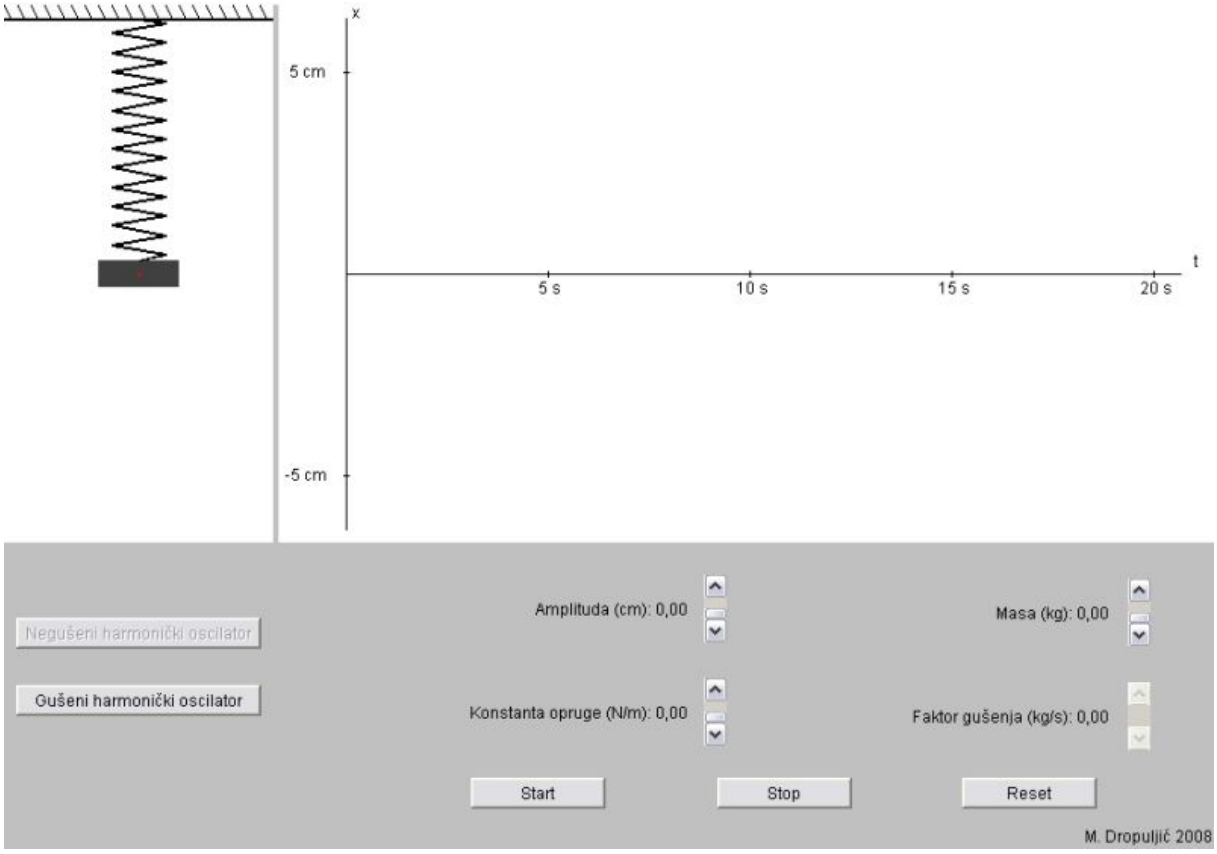
Slika 3.2. Pokretanje appleta iz Internet preglednika



Slika 3.3. Pokretanje Java appleta iz komandnog promta

3.7. Upute za korištenje appleta

U ovom dijelu govorit ćemo o načinu korištenja appleta. Applet se sastoji od tri dijela (Slika 3.4.). Prvi dio predstavlja samu animaciju gibanja tijela na opruzi, zatim slijedi područje na kojem se odvija simulacija (crtanje grafa) i na kraju područje gdje biramo parametre, sa kontrolama koje omogućavaju interakciju appleta i korisnika.



Slika 3.4. Dijelovi appleta

Na appletu se mogu odabrati dvije vrste harmoničkog osciliranja: negušeno harmoničko osciliranje i gušeno harmoničko osciliranje.
Odabirom negušenog harmoničkog oscilatora možemo po želji odabrati vrijednosti za amplitudu (pomak iz položaja ravnoteže), konstantu opruge i masu tijela, ali ne možemo odabrati faktor gušenja.
Kod gušenog harmoničkog oscilatora također možemo odabrati vrijednosti za amplitudu, konstantu opruge, masu tijela, ali i faktor gušenja.

NEGUŠENI HARMONIČKI OSCILATOR

Negušeni harmonički oscilator odabiremo klikom na tipku 

, nakon čega će nam lijevi donji ugao appleta izgledati ovako:

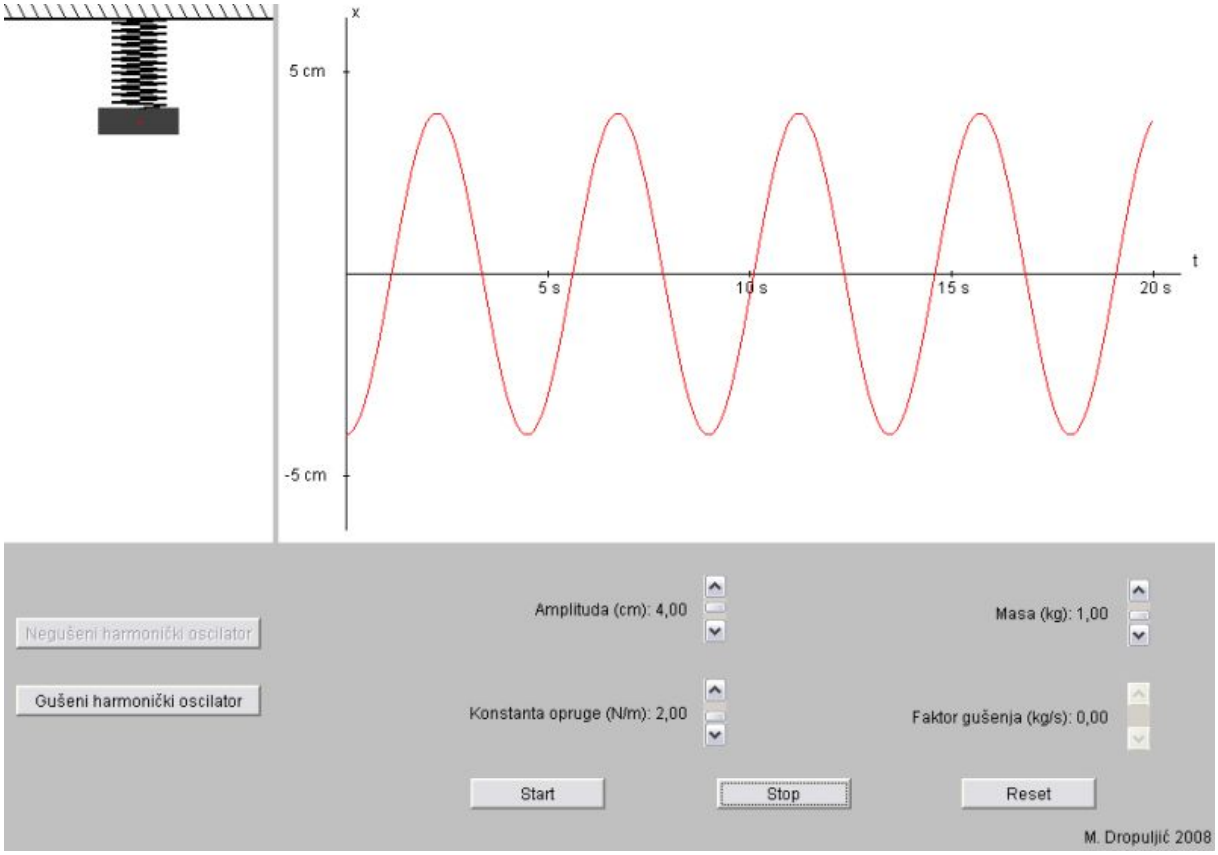
Slika 3.5. Odabir vrste osciliranja

Nakon toga odabiremo ostale parametre po želji (amplituda, konstanta opruge, masa tijela), osim faktora gušenja. Uzeti ćemo da je amplituda: 4cm, konstanta opruge: 2 N/m i masa: 1 kg, kao što je prikazano na Slici 3.6. (odabirom vrijednosti za amplitudu tijelo na opruzi se automatski pomiče iz položaja ravnoteže za isti iznos).

Slika 3.6. Odabir vrijednosti parametara

Kada smo odabrali parametre kliknemo na tipku 

i applet se pokrene, nakon čega slijedi gibanje tijela na opruzi u lijevom gornjem uglu i crtanje grafa gibanja tijela u desnom gornjem uglu. Naš applet tada izgleda ovako (Slika 3.7.)

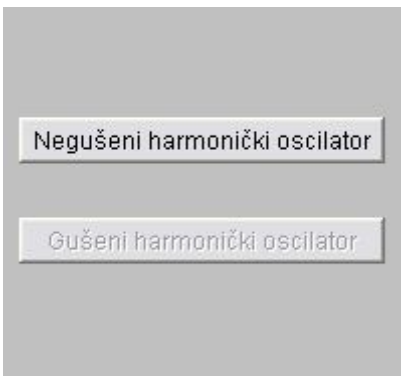


Slika 3.7. Izgled appleta nakon odabranih parametara

Ako kliknemo na tipku Stop možemo zaustaviti u bilo kojem trenutku i ponovo je pokrenuti sa tipkom Start. Kliknemo li na tipku Reset sve se poništava i možemo odabrati nove parametre ili novu vrstu harmoničkog osciliranja.

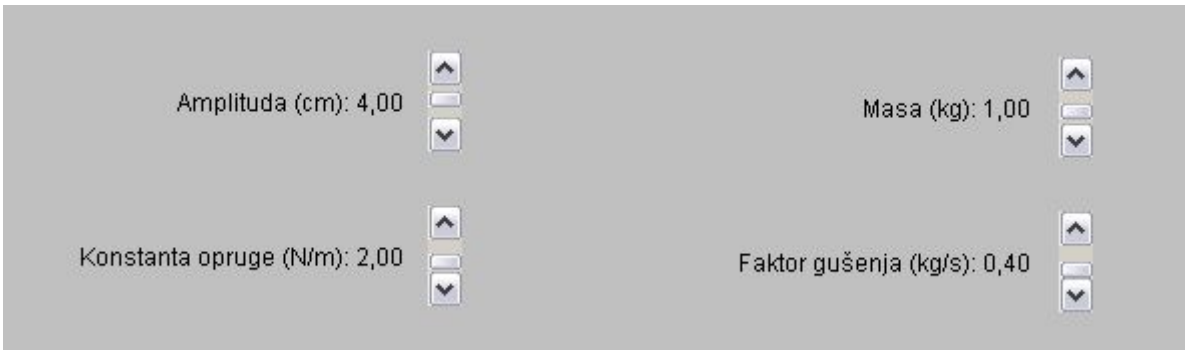
GUŠENI HARMONIČKI OSCILATOR

Gušeni harmonički oscilator odabiremo klikom na tipku Gušeni harmonički oscilator, nakon čega će nam lijevi donji ugao appleta izgledati ovako:



Slika 3.8. Odabir gušenog harmoničkog osciliranja

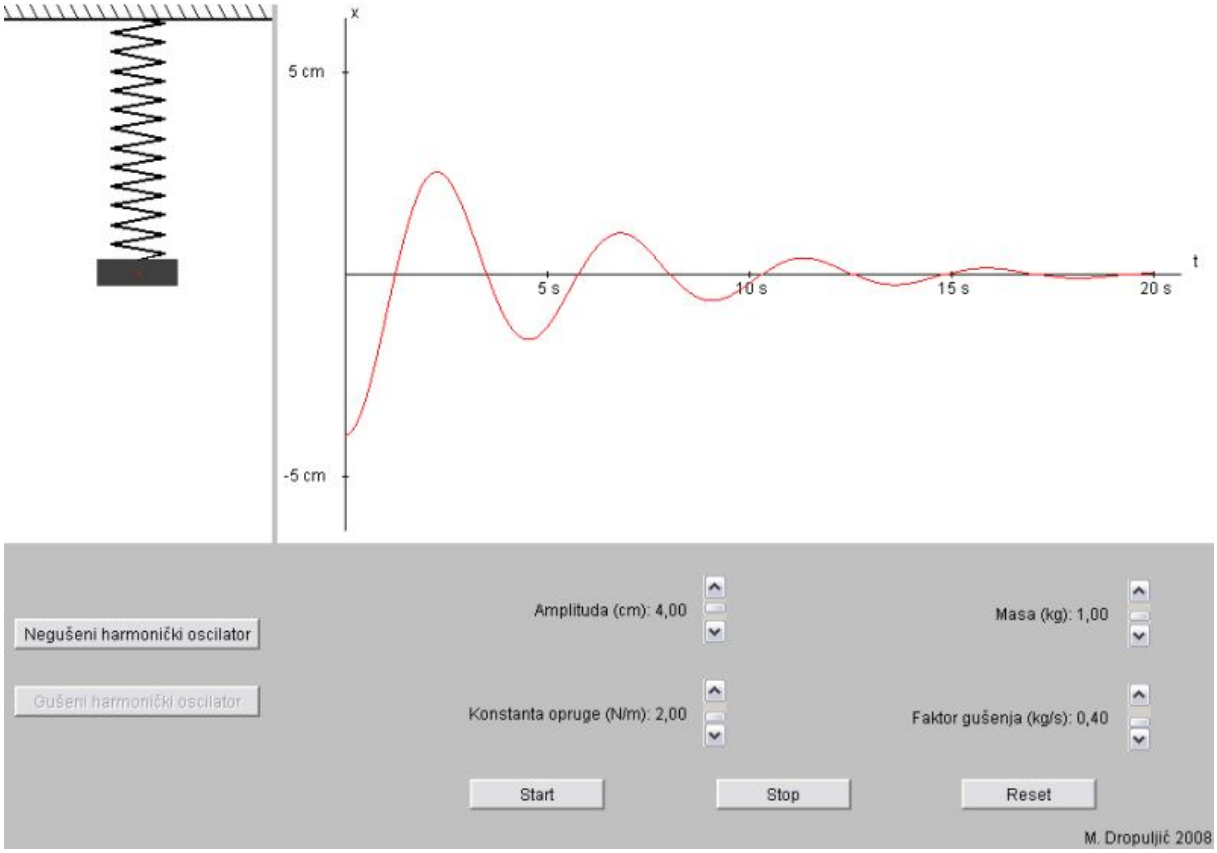
Uzeti ćemo iste vrijednosti parametara kao kod negušenog harmoničkog osciliranja, samo ovaj put moramo odabrati i vrijednosti faktora gušenja (Slika 3.9.),jer u protivnom tijelo bi osciliralo kao kod negušenog harmoničkog osciliranja. Faktor gušenja ćemo odabrati da je 0,4 kg/s.



Slika 3.9. Odabir vrijednosti svih parametara u appletu

Nakon toga kliknemo ponovo na tipku 

i tijelo će početi titrati s gušenjem (Slika 3.10.):



Slika 3.10. Izgled appleta nakon odabranih svih parametara

4. Implementacija appleta u nastavi fizike

Škola: Gimnazija
Razred: 3
Nastavni predmet: Fizika
Nastavna cjelina: Titranje
Nastavna jedinica: Harmonijsko titranje
Mjesto održavanja nastave: Računalna učionica uz pribor potreban za izradu pokusa
Cilj: Uočiti rezultate harmonijskog titranja za različite parametre i uvjete
Nastavna pomagala: Računalo sa instaliranom Javom, projektor, oprema za pokus (flomaster, papir, 2 opruge, uteg, pluteni čep)
Nastavna sredstva: Java applet, školska ploča.
Nastavne metode: Rad na računalu. Po dva učenika za jednim računalom. Metoda razgovora (diskusija). Eksperiment.
Nastavni proces:

Nastavna jedinica Harmonijsko titranje je izvedena iz nastavne cjeline Titranje. Podrazumijeva se da su učenici upoznati sa osnovnim terminima titranja i gdje su u prirodi vidjeli i upoznali neku vrstu titranja.

Eksperimentalni prikaz harmonijskog titranja

POKUS 1
Pribor: Opruga, uteg (0,1kg)
Na oprugu konstante k objesimo uteg mase m, upoznamo učenike sa dijelovima pribora pokusa. Prije nego pobudimo uteg na titranje postavimo učenicima pitanje:
Što će se dogoditi kada uteg izvučemo iz položaja ravnoteže, te pustimo? Možemo li prije izvođenja pokusa predvidjeti gibanje?
Većina učenika će reći da uteg titra oko položaja ravnoteže.
Izvedemo pokus nekoliko puta i vidimo da li su naša predviđanja bila ispravna. Nakon pokusa biti će lakše odgovoriti na sljedeća pitanja:
Što se događa s tijelom kada ga izvučemo iz položaja ravnoteže? Što su parametri ovog sustava? Što je pomak iz ravnotežnog položaja, je li on konstantan za dani sustav? Ako maksimalan pomak zovemo amplituda, da li se on nakon duljeg vremena ponavljanja gibanja promijenio?
Iz pokusa je vidljivo da tijelo titra oko položaja ravnoteže. Kako se sustav sastoji od opruge i utega, tada su parametri ovog sustava određeni njihovim karakteristikama. Nije moguće pronaći još koji parametar osim konstante opruge k i mase utega m, pa zaključujemo da su to parametri koji karakteriziraju dani sustav.
Ako pustimo tijelo da dovoljno dugo titra vidimo da se amplituda počinje smanjivati, tj. postoji sila koja uzrokuje smanjenje amplitude, a to je otpor sredstva. Kako mi možemo na početku izvući tijelo za bilo koji pomak, on nije konstantan, već se mijenja s vremenom. Smanjenje amplitude nastaje uslijed gušenja titranja. Sada ćemo proučavati idealnu situaciju gdje nema gušenja, tj. promatrati ćemo *jednostavno harmoničko titranje*.
(Napišemo naslov na ploču: „Jednostavno harmoničko titranje“)
Pokušajmo sada razmatrati kako i zašto se tijelo na opruzi giba. To ćemo razmatranje provoditi serijom pitanja. Nakon svakog postavljenog pitanja pokus možemo ponoviti (za to nam ne treba previše vremena), kako bi učenicima pitanja bila što jasnija, a time odgovori još jednostavniji. Ako nakon postavljenog pitanja vidimo po reakciji učenika da nemaju odgovor,

pitanje možemo preformulirati, ali nikako sami dati odgovor, već navesti učenike da oni dođu do njega.
Zašto se tijelo izvučeno iz položaja ravnoteže giba kad ga ispustimo? Kakvo gibanje izvodi tijelo, ako postoji akceleracija, je li ona konstantna ili promjenjiva? Kolika je sila u položaju ravnoteže?
Prvo moramo raspraviti koje sve sile ovdje postoje i kako one utječu na titranje tijela. Neki učenici bi možda rekli da se tijelo giba zbog sile teže, ali znamo da sila teža jednako utječe na tijelo i u položaju ravnoteže i kad tijelo izvučemo iz položaja ravnoteže za neki pomak. Primjedba koja bi mogla slijediti je da se g mijenja s visinom, međutim ovdje se radi o malim pomacima, tako da je ta promjena od g zanemariva. Sila koja je usko vezana uz oprugu je *elastična sila*. Ako se opruga produlji za pomak x, djeluje elastična sila opruge $F = -kx$ koja nastoji vratiti tijelo u položaj ravnoteže. Zbog postojanja elastične sile koja ovisi o x i zbog 2. Newtonovog zakona $am=-kx$, vidi se da je akceleracija promjenjiva. Kako je $F =-kx$, za $x=0$ sila je također jednaka nuli.
Sada dolazimo do sljedećih pitanja:
Zašto tijelo prolazi kroz položaj ravnoteže? Koji zakon očuvanja ovdje vrijedi, te kako on glasi? Koliki je iznos ukupne energije, ako za x izaberemo amplitudu, tj. $x=A$? Kada je brzina maksimalna i koliki je njezin iznos?
Tijelo u položaju ravnoteže ima određenu brzinu, pa zbog tromosti nastavlja gibanje u istom smjeru. Kada smo tijelo izvukli za pomak x, ono ima određenu potencijalnu energiju. U položaju ravnoteže potencijalna energija jednaka je nuli, a kako se energija ne može izgubiti već samo pretvoriti iz jednog oblika u drugi, u položaju ravnoteže tijelo ima samo kinetičku energiju. Zakon očuvanja mehaničke energije glasi:

$$E = E_k + E_p = const.$$

2

Prisjetimo se sada iznosa kinetičke i potencijalne energije za sustav koji ima elastičnost opruge k, masu utega m i amplituda mu je A:

$$E_k = \frac{mv^2}{2} \quad \text{i} \quad E_p = \frac{kx^2}{2}$$

Sada je:

$$E = \frac{mv^2}{2} + \frac{kx^2}{2}$$

Kako je brzina tijela u amplitudnom položaju jednaka nuli, tada je:

$$E = \frac{kA^2}{2}$$

pa zakon očuvanja energije glasi:

$$\frac{mv^2}{2} + \frac{kx^2}{2} = \frac{kA^2}{2}$$

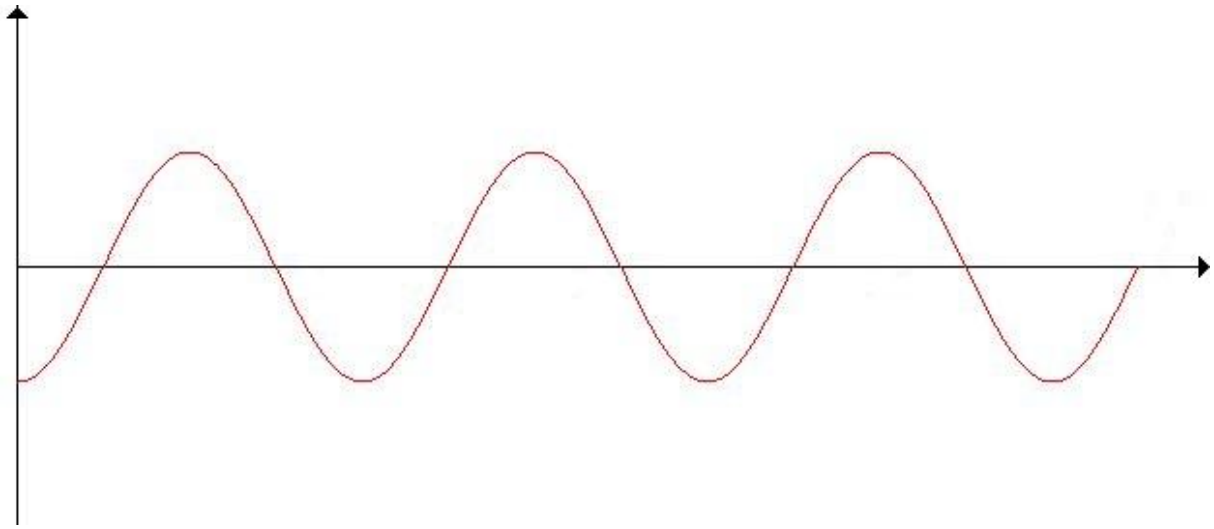
Iz ove jednakosti se vidi kada je brzina maksimalna. To je kada je potencijalna energija maksimalna tj. za *x=0*, odnosno kada tijelo prolazi kroz ravnotežni položaj.

$$\frac{mv_{\max}^2}{2} = \frac{kA^2}{2}, \qquad v_{\max}^2 = \frac{kA^2}{m}$$

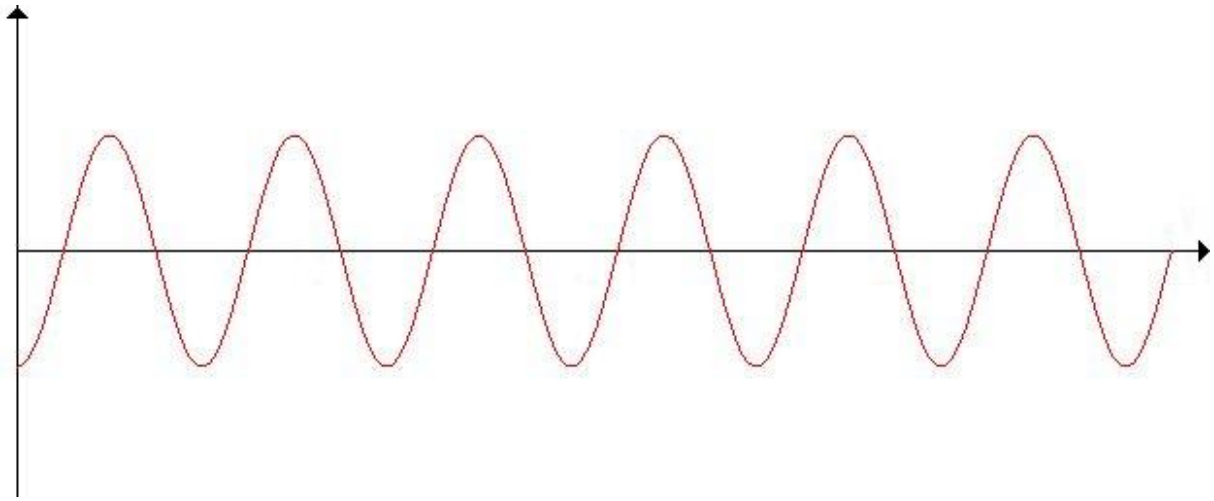
$$v_{\max} = A\sqrt{\frac{k}{m}}$$

Nakon dobivenih svih ovih odgovora pozovemo jednog učenika koji će kratko uz demonstraciju pokusa sve ovo ponoviti, kako bi učenici mogli nadopuniti svoje bilješke u bilježnici.
Pošto znamo da na tijelo djeluje sila, moramo odrediti položaj tijela u ovisnosti o vremenu. To ćemo učiniti pomoću sljedećeg pokusa. U pokusu ćemo koristiti flomaster učvršćen pomoću plutenog čepa između dviju opruga, zatim papir i dva ravnala.
Čime je određen položaj tijela? Kako bi pomoću ovog pribora izveli pokus kojim bi odredili tu ovisnost?
Kako je čep s flomasterom učvršćen između dviju opruga, on će harmonički titrati. Položaj tijela jedino može biti određen pomakom iz položaja ravnoteže. Učenici će sigurno reći da opruge i flomaster postavimo tako da kad sustav titra, flomaster ostavlja trag na papiru. To možemo izvesti i vidjeti ćemo da je trag flomastera crta koja se stalno iscrtava na jednom mjestu. Iz tog zapisa ne možemo odrediti vremensku ovisnost. Ona je vidljiva kad papir ispod flomastera vučemo stalnom brzinom.
Pripremimo sada taj pokus.

POKUS 2.
Pribor: Dvije jednake elastične opruge (*k = 20 N/m*), pluteni čep, flomaster, nekoliko papira, dva ravnala kao vodilice.
Položaj flomastera namjestimo tako da mu vrh prislanja uz papir. Zapis nam pokazuje kako se položaj čepa mijenja s vremenom. Bitno je da se papir giba u pravcu što postižemo pomoću ravnala koja nam predstavljaju vodilice. Čep pomaknemo iz ravnotežnog položaja i pustimo ga. On će titrati, pri čemu će na mirnom papiru ostaviti trag u obliku crte. Da bi dobili zapis vremenske osi papir pomičemo ispod flomastera jednolikom brzinom. Prije izvođenja pokusa pitamo učenike; što misle kakav će trag ostaviti flomaster, ako papir vučemo stalnom brzinom. Odgovor će najvjerojatnije biti valovita crta; možda će netko reći da je to baš sinusoida.
Demonstrirajmo sada pokus. Pokus ponovimo nekoliko puta pri čemu papir pomičemo s drugom stalnom brzinom. Svaki put koristimo novi papir. Promotrimo dobivene rezultate (Slika 4.1. i Slika 4.2.odgovoriti na sljedeća pitanja:



Slika 4.1. Dobiveni rezultati s većom stalnom brzinom



Slika 4.2 Dobiveni rezultati s manjom stalnom brzinom

Po čemu se ove krivulje mogu međusobno razlikovati? Da li možete prepoznati ovu krivulju? O čemu ovisi je li krivulja sinusoida odnosno kosinusoida? Ako je na početku pomak jednak amplitudi, što je tada naša krivulja?
Iz dobivenih zapisa na papirima da se zaključiti da se krivulje međusobno razlikuju po brzini kojom smo vukli papir. Nije teško prepoznati ovu krivulju . To je *sinusoida*, odnosno *kosinusoida*, ovisno o tome gdje izaberemo početak krivulje. Ako je na početku pomak jednak amplitudi, tada je krivulja kosinusoida. Nacrtamo kosinusoidu na ploču.
U kojem obliku možemo napisati rješenje?
To specijalno rješenje očigledno je u obliku:

$$x = A\cos bt$$

Zbog čega smo u argument funkcije kosinus uz vrijeme t stavili i nepoznat faktor b? Koji je period kosinus funkcije i kako bi pomoću toga odredili b?

Kako je argument funkcije kosinus bezdimenzionalni broj, vrijeme *t* u argumentu funkcije kosinus mora biti pomnoženo s nekim faktorom koji ima dimenziju recipročnog vremena. Proizlazi da jedinica od *b* mora biti recipročna sekunda. Period kosinus funkcije je 2π , dakle ako argument *bt* povećamo za 2π , ili ako *t* povećamo za vremenski period *T*, dobit ćemo jednaku vrijednost za funkciju kosinus:

$$\cos(bt + 2\pi) = \cos b(t + T)$$

$$bt + 2\pi = bt + bT$$

$$b = \frac{2\pi}{T} = 2\gamma = \omega$$

Sada vidimo da je faktor *b* jednak kružnoj frekvenciji ω .
Neka sada jedan učenik ponovi: kakvu krivulju predstavlja zapis, tj. kakva je ovisnost pomaka o vremenu.

Provjera pretpostavki korištenjem appleta

Na svakom računalu nalazi se applet kojeg učenici trebaju učitati. Objasniti učenicima kako se pokreće simulacija, koju vrstu harmoničkog osciliranja odabrati i kako mijenjati parametre.
Zadaci za učenike
Svaki od učenika neka po želji odabere konstantu opruge , masu i amplitudu, te pokrenu simulaciju. U drugom slučaju neka odaberu isti iznos za konstantu opruge i za masu, ali drugačiji iznos za amplitudu.
Postavljamo im pitanja: Kakva je razlika između ova dva titranja?
Očekivani odgovor od učenika: Uteg u istim vremenskim razmacima prolazi kroz ravnotežni položaj. Ova dva slučaja razlikuju se u amplitudi titranja.

Za domaći uradak zadamo ima da naprave mali seminar tako da će napisati sve moguće razlike titranja, za razne slučajeve kada odaberemo različite parametre.

Korištene aplikacije

Pri izradi ovog diplomskog koristio sam sljedeći software:

- MS WORD 2000
- IrfanView 3.98
- JDK 5.0 i JRE 5.0
- Eclipse SDK 3.1.2
- MS Excel 2000
- Mathematica 5.0

Svi gore navedeni programi su izvršavani pod operativnim sustavom Windows XP Professional Version 2002 SP2

Literatura

Fizika

1. M. Paić: „Gibanja, sile, valovi“, Školska knjiga, Zagreb, 1997.
2. N. Cindro: „Fizika 1.“, Školska knjiga, Zagreb, 1975.
3. C. Kittel
   W.D. Knighl
   M.A. Rudermann: „Mehanika“- svezak 1, udžbenik Sveučilišta u Berkelyu, Tehnička knjiga, Zagreb, 1981.
4. R. Krsnik, B. Mikuličić: „Međudjelovanja relativnosti, titranja i zvuk“, Školska knjiga, Zagreb, 1996.

Informatika

5. P. Niemeyer, J. Knudsen, O'Reilly: „Learning Java“ Second Edition, Sebastpol, 2002.
6. M Campione, K. Walrath: „The Java Tutorial“ Second Edition, Sun Microsystem, Palo Alto, 1998.
7. T. Čukman: „Java“, Alfej, Zagreb, 1997.
8. G. Muić: „Računalni praktikum 2., Matematički odsjek, Zagreb, 2005. http://www.math.hr/nastava/rp2
9. P. Brođanac: „Java i objektno orijentirano programiranje“, Zagreb, 2005.
10. Jack's Page: „Basic Physics and Optics“, Jack Ord, http://www.kw.igs.net/~jackord/
11. W Fendt: „Java Applets on Physics“ http://www.walter-fendt.de/ph14e
12. J. Gosling, H. McGilton: „The Language Environment white paper“, 1996. http://java.sun.com/docs/white/langenv/

Prilog

```
package hr.phy.oscilator;

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;

public class GrafCanvas extends Canvas {

	private static final long serialVersionUID = 8320140611405277104L;

	private Image offScreenImage;
	private OscillatorGraph oscillatorGraph;

	private int it = 0;

	public GrafCanvas(Image image, OscillatorGraph oscillatorGraph)
	{
		setBackground(Color.white);
		this.offScreenImage = image;
		this.oscillatorGraph = oscillatorGraph;
	  repaint();
	}

    public void update(Graphics g)
    {
        Graphics offg = this.offScreenImage.getGraphics();
        paint(offg);
        g.drawImage(this.offScreenImage, 0, 0, this);
    }

    public void drawCoordinates(Graphics g)
    {
        g.setFont(new Font("Helvetica", 0, 12));
        g.setColor(Color.white);
        g.fillRect(0, 0, 696, 400); // pobrisi cijeli graf
        g.setColor(Color.black);
        g.drawLine(50, 200, 670, 200); // x os
        g.drawLine(200, 198, 200, 202); // crtica na 5 s
        g.drawLine(350, 198, 350, 202); // crtica na 10 s
        g.drawLine(500, 198, 500, 202); // crtica na 15 s
        g.drawLine(650, 198, 650, 202); // crtica na 20 s
        g.drawString(" 5 s", 190, 215);
        g.drawString("10 s", 340, 215);
        g.drawString("15 s", 490, 215);
        g.drawString("20 s", 640, 215);
        g.drawString("t", 680, 195);
        g.drawLine(50, 10, 50, 390); // y os
        g.drawLine(48, 50, 52, 50); // crtica na 5 cm
        g.drawLine(48, 350, 52, 350); // crtica na -5 cm
        g.drawString(" 5 cm", 5, 54);
        g.drawString("-5 cm", 5, 354);
        g.drawString("x", 55, 10);
    }

    public void paint(Graphics g)
    {
```

```
      if (this.oscillatorGraph.resetMe || it == 0)
      {
            g.setColor(Color.white);
            g.fillRect(0, 0, 500, 300);
            drawCoordinates(g);
            it = 0;
            this.oscillatorGraph.resetMe = false;
      }

      if (this.oscillatorGraph.isRunning && it < 600) {
      double t = (double) it * .033;
      this.oscillatorGraph.x = it + 50;

      if (this.oscillatorGraph.omega0 > this.oscillatorGraph.beta) {
      double omega = Math.sqrt(this.oscillatorGraph.omega0 *
this.oscillatorGraph.omega0
        - this.oscillatorGraph.beta * this.oscillatorGraph.beta);
      this.oscillatorGraph.y = 200 + (int) ((double)
(this.oscillatorGraph.y0 - 200)
      * (Math.cos(omega * t) + (Math.sin(omega * t) *
this.oscillatorGraph.beta)
      / omega) * Math.exp(-this.oscillatorGraph.beta * t));
      } else {
      double omega = Math.sqrt(-this.oscillatorGraph.omega0 *
this.oscillatorGraph.omega0
        + this.oscillatorGraph.beta * this.oscillatorGraph.beta);
      if (omega < 0.0001D) {
      this.oscillatorGraph.y = 200 + (int) ((double)
(this.oscillatorGraph.y0 - 200)
      * Math.exp(-this.oscillatorGraph.beta * t) * (1.0D +
this.oscillatorGraph.beta * t));
      } else {
      double A = (0.5D * (omega - this.oscillatorGraph.beta)) / omega;
      double B = (0.5D * (omega + this.oscillatorGraph.beta)) / omega;
      this.oscillatorGraph.y = 200 + (int) ((double)
(this.oscillatorGraph.y0 - 200) * (A
      * Math.exp(-(omega + this.oscillatorGraph.beta) * t) + B
      * Math.exp(-(this.oscillatorGraph.beta - omega) * t)));
      }
      }

      if (it > 0)
      {
      g.setColor(Color.red);
      g.drawLine(this.oscillatorGraph.x, this.oscillatorGraph.y,
this.oscillatorGraph.xCrtano,
        this.oscillatorGraph.yCrtano);
      }
      this.oscillatorGraph.xCrtano = this.oscillatorGraph.x;
      this.oscillatorGraph.yCrtano = this.oscillatorGraph.y;
      it++;
      }
            if (it == 600) {
                  this.oscillatorGraph.isRunning = false;
                  this.oscillatorGraph.enableComponentsOnStop();
            }

    }


}
```

```
package hr.phy.oscilator;

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;

public class OprugaCanvas extends Canvas {

      private static final long serialVersionUID = 3635721956877624230L;

      private Image offScreenImage;
      private OscillatorGraph oscillatorGraph;

      public OprugaCanvas(Image image, OscillatorGraph oscillatorGraph)
      {
            setBackground(Color.white);
            this.offScreenImage = image;
            this.oscillatorGraph = oscillatorGraph;
        repaint();
      }

    public void update(Graphics g)
    {
        Graphics offg = this.offScreenImage.getGraphics();
        paint(offg);
        g.drawImage(this.offScreenImage, 0, 0, this);
    }

    public void crtanjeMase(Graphics g)
    {
        g.setColor(Color.white);
        g.fillRect(0, 0, 200, 400); // ocisti opruga canvas
        g.setColor(Color.darkGray);
        g.fillRect(70, this.oscillatorGraph.y - 10, 60, 20); // crta masu
        g.setColor(Color.red);
        g.fillOval(99, this.oscillatorGraph.y -1, 3, 3); // crta tockicu u
sredini
        g.setColor(Color.black);
        for (int i = 0; i < 200; i+=10) {
                g.drawLine(i, 0, i+5, 10); // srafira strop
            }
        g.drawLine(0, 10, 200, 10); // linija ispod srafiranja
        g.drawLine(0, 11, 200, 11); // podebljanje linije
    }

    public void crtanjeOpruge(int nt, int yhigh, int ylow, Graphics g)
    {
        double denom = 2D * (double)nt + 1.0D;
        g.drawLine(100, ylow, 120, (int)((double)ylow + (0.5D *
(double)(yhigh - ylow)) / denom));
        for(int i = 1; i < 2 * nt; i += 2)
        {
            g.drawLine(120, (int)((double)ylow + (((double)i - 0.5D) *
(double)(yhigh - ylow))
            / denom), 80, (int)((double)ylow + (((double)i + 0.5D) *
(double)(yhigh - ylow)) / denom));
            g.drawLine(120, (int)((double)ylow + (((double)i + 1.5D) *
(double)(yhigh - ylow)) / denom),
```

```
            80, (int)((double)ylow + (((double)i + 0.5D) * (double)(yhigh -
ylow)) / denom));
        }

        g.drawLine(120, (int)((double)ylow + ((2D * (double)nt + 0.5D) *
(double)(yhigh - ylow)) / denom),
        100, this.oscillatorGraph.y - 10);
    }

    @Override
    public void paint(Graphics g) {
      super.paint(g);
        crtanjeMase(g);
        crtanjeOpruge(12, this.oscillatorGraph.y - 11, 10, g);
        crtanjeOpruge(12, this.oscillatorGraph.y - 10, 11, g);
    }
}
```

```
package hr.phy.oscilator;

import java.applet.Applet;
import java.awt.Dimension;
import java.awt.Color;
import java.awt.Button;
import java.awt.Event;
import java.awt.Rectangle;
import java.awt.Label;
import java.awt.Point;
import java.awt.Scrollbar;
import java.text.DecimalFormat;

public class OscillatorGraph extends Applet implements Runnable {

      private static final long serialVersionUID = 1L;
      private Button hoNeguseniButton = null;
      private Button hoGuseniButton = null;
      private Button startButton = null;
      private Button stopButton = null;
      private Button resetButton = null;
      private Label amplitudaLabel = null;
      private Scrollbar amplitudaScrollbar = null;
      private Label masaLabel = null;
      private Scrollbar masaScrollbar = null;
      private Label konstantaLabel = null;
      private Scrollbar konstantaScrollbar = null;
      private Label gusenjeLabel = null;
      private Scrollbar gusenjeScrollbar = null;

      private OprugaCanvas oprugaCanvas;
      private GrafCanvas grafCanvas;

      private boolean isGuseni = false;

      boolean isRunning = false;

      boolean resetMe = true;

      private double k;
      private double m;
      private double A;
      private double mi;
```

```
	double omega0;
	double beta;

	int y0;

	int x;
	int y;
	int xCrtano;
	int yCrtano;

	private Thread moveThread;  //  @jve:decl-index=0:
	private Label potpisLabel = null;


	/**
	 * This is the default constructor
	 */
	public OscillatorGraph() {
		super();
	}

	public void start() {
		if (isRunning && moveThread == null) {
			moveThread = new Thread(this);
			moveThread.start();
		}
	}

	public void stop() {
		if (moveThread != null) {
			moveThread.stop();
			try {
				moveThread.join();
			} catch (InterruptedException _ex) {
			}
		}
		moveThread = null;
	}

	public void run() {
		do {
			try {
				Thread.sleep(33L);
			} catch (InterruptedException _ex) {
			}
			grafCanvas.repaint();
			oprugaCanvas.repaint();
		} while (true);
	}

	/**
	 * This method initializes this
	 *
	 * @return void
	 */
	public void init() {

		initVars();

		potpisLabel = new Label();
```

```
        potpisLabel.setBounds(new Rectangle(737, 607, 163, 23));
        potpisLabel.setAlignment(Label.RIGHT);
        potpisLabel.setText("M. Dropuljiæ 2008");
        oprugaCanvas = new OprugaCanvas(createImage(200, 400), this);
        this.add(oprugaCanvas, null);
        oprugaCanvas.setBounds(0, 0, 200, 400);

        grafCanvas = new GrafCanvas(createImage(696, 400), this);
        add(grafCanvas, null);
        grafCanvas.setBounds(204, 0, 696, 400);

        gusenjeLabel = new Label();
        gusenjeLabel.setBounds(new Rectangle(629, 518, 194, 23));
        gusenjeLabel.setAlignment(Label.RIGHT);
        gusenjeLabel.setText("Faktor gušenja (kg/s): 0,00");
        konstantaLabel = new Label();
        konstantaLabel.setBounds(new Rectangle(316, 515, 194, 23));
        konstantaLabel.setAlignment(Label.RIGHT);
        konstantaLabel.setText("Konstanta opruge (N/m): 4,00");
        masaLabel = new Label();
        masaLabel.setBounds(new Rectangle(629, 441, 194, 23));
        masaLabel.setAlignment(Label.RIGHT);
        masaLabel.setText("Masa (kg): 1,00");
        amplitudaLabel = new Label();
        amplitudaLabel.setBounds(new Rectangle(316, 438, 194, 23));
        amplitudaLabel.setAlignment(Label.RIGHT);
        amplitudaLabel.setText("Amplituda (cm): 5,00");
        this.setLayout(null);
        this.setSize(900, 630);

        this.setBackground(Color.lightGray);
        this.add(getHoNeguseniButton(), null);
        this.add(getHoGuseniButton(), null);
        this.add(getStartButton(), null);
        this.add(getStopButton(), null);
        this.add(getResetButton(), null);
        this.add(amplitudaLabel, null);
        this.add(getAmplitudaScrollbar(), null);
        this.add(masaLabel, null);
        this.add(getMasaScrollbar(), null);
        this.add(konstantaLabel, null);
        this.add(getKonstantaScrollbar(), null);
        this.add(gusenjeLabel, null);
        this.add(getGusenjeScrollbar(), null);
        this.add(potpisLabel, null);

        gusenjeScrollbar.setValue(800);
        amplitudaScrollbar.setValue(100);
        masaScrollbar.setValue(300);
        konstantaScrollbar.setValue(1200);
    }

    private void initVars() {
        x = 250;

        m = 1d;
        k = 4d;
        A = 5d;

        y0 = 200 + (int) (A / 5 * 150);
        y = y0;
```

```
            omega0 = Math.sqrt(k / m);
            beta = mi / 2.0D / m;
      }

      /**
       * This method initializes hoNeguseniButton
       *
       * @return java.awt.Button
       */
      private Button getHoNeguseniButton() {
            if (hoNeguseniButton == null) {
            hoNeguseniButton = new Button();
            hoNeguseniButton.setLabel("Negušeni harmonièki oscilator");
            hoNeguseniButton.setEnabled(false);
            hoNeguseniButton.setBounds(new Rectangle(9, 456, 182, 23));
            hoNeguseniButton
            .addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent e) {
            isGuseni = false;
            hoGuseniButton.setEnabled(true);
            hoNeguseniButton.setEnabled(false);
            gusenjeScrollbar.setEnabled(false);
            gusenjeScrollbar.setValue(gusenjeScrollbar.getMaximum());

      gusenjeScrollbar.getAdjustmentListeners()[0].adjustmentValueChanged(n
ull);
            gusenjeScrollbar.repaint();
            }
            });
            }
            return hoNeguseniButton;
      }

      /**
       * This method initializes hoGuseniButton
       *
       * @return java.awt.Button
       */
      private Button getHoGuseniButton() {
            if (hoGuseniButton == null) {
            hoGuseniButton = new Button();
            hoGuseniButton.setLabel("Gušeni harmonièki oscilator");
            hoGuseniButton.setLocation(new Point(9, 506));
            hoGuseniButton.setSize(new Dimension(182, 23));
            hoGuseniButton
            .addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent e) {
            isGuseni = true;
            hoNeguseniButton.setEnabled(true);
            hoGuseniButton.setEnabled(false);
            gusenjeScrollbar.setEnabled(true);
                                    }
                              });
            }
            return hoGuseniButton;
      }

      /**
       * This method initializes startButton
       *
```

```
	 * @return java.awt.Button
	 */
	private Button getStartButton() {
		if (startButton == null) {
			startButton = new Button();
			startButton.setLabel("Start");
			startButton.setSize(new Dimension(100, 23));
			startButton.setLocation(new Point(347, 575));
		}
		return startButton;
	}

	public boolean handleEvent(Event evt) {
		if ((evt.target == startButton) && evt.id == 1001) {
			isRunning = true;
			disableComponentsWhenRun();
			if (moveThread == null) {
				moveThread = new Thread(this);
				moveThread.start();
			}
			return true;
		} else {
			return false;
		}
	}

	/**
	 * This method initializes stopButton
	 *
	 * @return java.awt.Button
	 */
	private Button getStopButton() {
		if (stopButton == null) {
		stopButton = new Button();
		stopButton.setLabel("Stop");
		stopButton.setSize(new Dimension(100, 23));
		stopButton.setLocation(new Point(530, 575));
		stopButton.addActionListener(new
java.awt.event.ActionListener() {
		public void actionPerformed(java.awt.event.ActionEvent e) {
		isRunning = false;
		enableComponentsOnStop();
		if (moveThread != null) {
		moveThread.stop();
		try {
		moveThread.join();
		} catch (InterruptedException _ex) {
		}
		}
		moveThread = null;
		}
		});
		}
		return stopButton;
	}

	/**
	 * This method initializes resetButton
	 *
	 * @return java.awt.Button
	 */
```

```
	private Button getResetButton() {
		if (resetButton == null) {
		resetButton = new Button();
		resetButton.setLabel("Reset");
		resetButton.setSize(new Dimension(100, 23));
		resetButton.setLocation(new Point(712, 575));
		resetButton.addActionListener(new
java.awt.event.ActionListener() {
		public void actionPerformed(java.awt.event.ActionEvent e) {
		if (moveThread != null) {
		moveThread.stop();
		try {
		moveThread.join();
		} catch (InterruptedException _ex) {
		}
		}
		moveThread = null;
		resetMe = true;
		y = y0;
		oprugaCanvas.repaint();
		grafCanvas.repaint();
		}
		});
		}
		return resetButton;
	}

	/**
	 * This method initializes amplitudaScrollbar
	 *
	 * @return java.awt.Scrollbar
	 */
	private Scrollbar getAmplitudaScrollbar() {
		if (amplitudaScrollbar == null) {
		amplitudaScrollbar = new Scrollbar();
		amplitudaScrollbar.setBounds(new Rectangle(520, 424, 17, 50));
		amplitudaScrollbar.setVisibleAmount(1);
		amplitudaScrollbar.setMaximum(601);
		amplitudaScrollbar
		.addAdjustmentListener(new java.awt.event.AdjustmentListener()
{
		public void adjustmentValueChanged(
		java.awt.event.AdjustmentEvent e) {
		A = getVariableValue("Amplituda (cm): ",
		amplitudaLabel, amplitudaScrollbar);
		y0 = 200 + (int) (A / 5 * 150);
		y = y0;
		oprugaCanvas.repaint();
		}
		});
		}
		return amplitudaScrollbar;
	}

	/**
	 * This method initializes masaScrollbar
	 *
	 * @return java.awt.Scrollbar
	 */
	private Scrollbar getMasaScrollbar() {
		if (masaScrollbar == null) {
```

```
            masaScrollbar = new Scrollbar();
            masaScrollbar.setBounds(new Rectangle(835, 427, 17, 50));
            masaScrollbar.setVisibleAmount(1);
            masaScrollbar.setMaximum(401);
            masaScrollbar
            .addAdjustmentListener(new java.awt.event.AdjustmentListener()
{
            public void adjustmentValueChanged(
            java.awt.event.AdjustmentEvent e) {
            m = getVariableValue("Masa (kg): ", masaLabel,
            masaScrollbar);
            omega0 = Math.sqrt(k / m);
            beta = mi / 2.0D / m;
                                    }
                              });
            }
            return masaScrollbar;
      }

      /**
       * This method initializes konstantaScrollbar
       *
       * @return java.awt.Scrollbar
       */
      private Scrollbar getKonstantaScrollbar() {
            if (konstantaScrollbar == null) {
            konstantaScrollbar = new Scrollbar();
            konstantaScrollbar.setBounds(new Rectangle(520, 501, 17, 50));
            konstantaScrollbar.setVisibleAmount(1);
            konstantaScrollbar.setMaximum(1601);
            konstantaScrollbar
            .addAdjustmentListener(new java.awt.event.AdjustmentListener()
{
            public void adjustmentValueChanged(
            java.awt.event.AdjustmentEvent e) {
            k = getVariableValue("Konstanta opruge (N/m): ",
            konstantaLabel, konstantaScrollbar);
            omega0 = Math.sqrt(k / m);
            }
            });
            }
            return konstantaScrollbar;
      }

      /**
       * This method initializes gusenjeScrollbar
       *
       * @return java.awt.Scrollbar
       */
      private Scrollbar getGusenjeScrollbar() {
            if (gusenjeScrollbar == null) {
            gusenjeScrollbar = new Scrollbar();
            gusenjeScrollbar.setBounds(new Rectangle(835, 504, 17, 50));
            gusenjeScrollbar.setEnabled(false);
            gusenjeScrollbar.setVisibleAmount(1);
            gusenjeScrollbar.setMaximum(801);
            gusenjeScrollbar
            .addAdjustmentListener(new java.awt.event.AdjustmentListener()
{
            public void adjustmentValueChanged(
            java.awt.event.AdjustmentEvent e) {
```

```
            mi = getVariableValue("Faktor gušenja (kg/s): ",
            gusenjeLabel, gusenjeScrollbar);
            beta = mi / 2.0D / m;
            }
            });
            }
            return gusenjeScrollbar;
      }

      private void disableComponentsWhenRun() {
            hoGuseniButton.setEnabled(false);
            hoNeguseniButton.setEnabled(false);
            amplitudaScrollbar.setEnabled(false);
            masaScrollbar.setEnabled(false);
            konstantaScrollbar.setEnabled(false);
            gusenjeScrollbar.setEnabled(false);
            startButton.setEnabled(false);
            resetButton.setEnabled(false);
      }

      public void enableComponentsOnStop() {
            amplitudaScrollbar.setEnabled(true);
            masaScrollbar.setEnabled(true);
            konstantaScrollbar.setEnabled(true);
            startButton.setEnabled(true);
            resetButton.setEnabled(true);
            if (isGuseni) {
                  gusenjeScrollbar.setEnabled(true);
                  hoNeguseniButton.setEnabled(true);
            } else {
                  hoGuseniButton.setEnabled(true);
            }
      }

      private double getVariableValue(String text, Label label,
                  Scrollbar scrollbar) {
            DecimalFormat formatter = new DecimalFormat("0.00");

            double result = (scrollbar.getMaximum() - scrollbar.getValue()
- 1) / 100d;

            label.setText(text + formatter.format(result));

            return result;
      }
}
```