SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Dejan Popović

Diplomski rad

PRIMJENA NUMERIČKOG RJEŠAVANJA DIFERENCIJALNIH JEDNADŽBI NA SUSTAV VEZANIH HARMONIČKIH OSCILATORA

Zagreb, 2006

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

SMJER: PROFESOR FIZIKE I INFORMATIKE

Dejan Popović

Diplomski rad

PRIMJENA NUMERIČKOG RJEŠAVANJA DIFERENCIJALNIH JEDNADŽBI NA SUSTAV VEZANIH HARMONIČKIH OSCILATORA

Voditelj diplomskog rada: Doc. dr. sc. Mirko Planinić

Ocjena diplomskog rada: _____

Povjerenstvo:

1. _____
2. _____
3. _____

Datum polaganja: _____

Zagreb, 2006

Sadržaj

Zahvaljujem svom mentoru Mirku Planiniću na pomoći oko izbora teme i rješavanju problema.

Hvala roditeljima koji su mi pomogli puno u životu i bili potpora na svakom koraku.

Uvod

Nikada nisam ni sanjao da će fizika biti moj izbor zanimanja u životu. Prvi susreti sa fizikom u osnovnoj školi su bili teški i nisam se najbolje snašao. Profesori mi nisu htjeli pomoći, a niti su se trudili da me zainteresiraju. Nakon što sam se razočarao u fiziku sve više pažnje sam pridavao matematici i informatici. Nakon osnovne škole odlučio sam se upisati u gimnaziju, a za smjer sam odabrao prirodoslovno-matematički. Tamo sam vratio osjećaj za fiziku pod vodstvom profesora Dubravka Heinricha i profesorice Tanje Božić. U srednjoj sam već priželjkivao da se upišem na PMF. Dvopredmetni studij mi se učinio čudnim, ali sve je postalo jasno kad sam shvatio da se u biti radi o interdisciplinarnosti ova dva smjera.

Za diplomski rad sam odabrao diferencijalne jednadžbe, jer su me zaintrigirale još na kolegiju "Diferencijalne jednadžbe i dinamički sustavi", gdje smo trebali primjeniti i znanje nekog programskog jezika kako bi napravili program koji bi rješavao neke od sustava. Za numeričke metode sam više dobio zanimanja na kolegiju "Numeričke metode". Spoj programiranja i fizike mi se čini idealnim pa sam se odlučio za diplomski rad u kojemu mogu upotrijebiti vještine programiranja. Za obradu podataka sam koristio Wolfram Mathematica simbolički jezik, tako da dobijem na neki način vizualizaciju fizikalnih problema koji su mi zadani u diplomskom radu.

1. Motivacija za rad

1.1 Diferencijalne jednadžbe u fizici – jednostavni primjeri

1.1.1 Kosi hitac

Newtonov drugi zakon gibanja

$$\boldsymbol{F} = m\boldsymbol{a}$$

koji matematički modelira gibanje čestice u polju sile $\boldsymbol{F}$, sustav je triju diferencijalnih jednadžbi. Naime, razložimo li vektor sile $\boldsymbol{F}$ na njegove Kartezijeve komponente:

$$F_x,\ F_y,\ F_z,$$

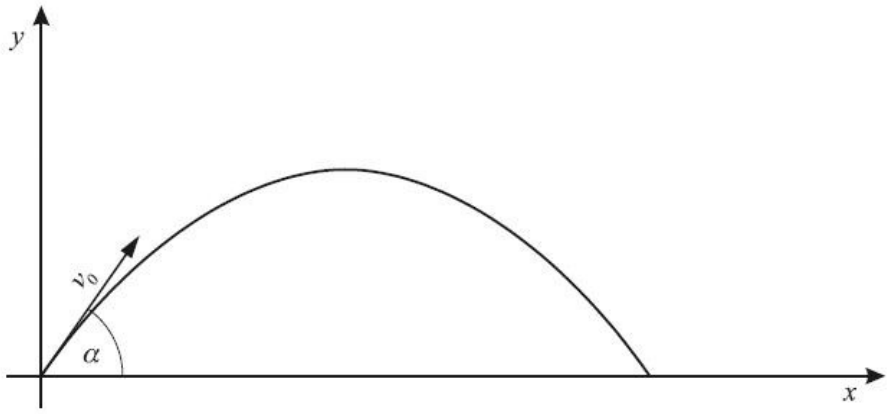
a vektor akceleracije na njegove Kartezijeve komponente:

$$a_x = \frac{d^2x}{dt^2},\ a_y = \frac{d^2y}{dt^2},\ a_z = \frac{d^2z}{dt^2},$$

onda iz $\boldsymbol{F} = m\boldsymbol{a}$ dobijemo tri diferencijalne jednadžbe:

$$m\frac{d^2x}{dt^2} = F_x,\ m\frac{d^2y}{dt^2} = F_y,\ m\frac{d^2z}{dt^2} = F_z,$$

koje opisuju kako položaj čestice *((x(t), y(t), z(t))* ovisi o vremenu *t* u polju sile $(F_x,\ F_y,\ F_z)$. Riješiti problem gibanja čestice u zadanom polju sile $\boldsymbol{F} = (F_x,\ F_y,\ F_z)$, znači riješiti gornji sustav od tri diferencijalne jednadžbe. Jednadžba kosog hica je dobro poznat primjer. Smjestimo koordinatni sustav tako da je os *y* paralelna sa silom teže *mg*, dok je os *x* vodoravna i s osi *y* tvori ravninu u kojoj se ispaljuje projektil (Slika 1.1).



Slika 1.1 Kosi hitac

Ako se on ispaljuje pod kutom α i s početnom brzinom v_0 (za $t = 0$), onda diferencijalne jednadžbe njegova gibanja izgledaju ovako:

$$m\frac{d^2x}{dt^2} = 0, \quad m\frac{d^2y}{dt^2} = -mg,$$

tj.

$$\frac{d^2x}{dt^2} = 0, \quad \frac{d^2y}{dt^2} = -g.$$

Rješenje tih jednadžbi

$$x = v_0 t\cos\alpha \quad \text{i} \quad y = -\frac{1}{2}g t^2 + v_0 t\sin\alpha,$$

koje zadovoljava uvjete $x(0) = 0,\ \dot{x}(0) = v_0\cos\alpha,\ y(0) = 0, \dot{y}(0) = v_0\sin\alpha,$ opisuje gibanje projektila. Početni uvjeti jednadžbe su proizvoljni, mi smo ih zadali ovako. Njegova putanja parametarski je određena tim jednadžbama. Izračunamo li parametar t iz prve jednadžbe i uvrstimo ga u drugu, dobit ćemo eksplicitnu jednadžbu putanje:

$$y = -\frac{g}{2v_0^2\cos^2\alpha}x^2 + (tg\,\alpha)x,$$

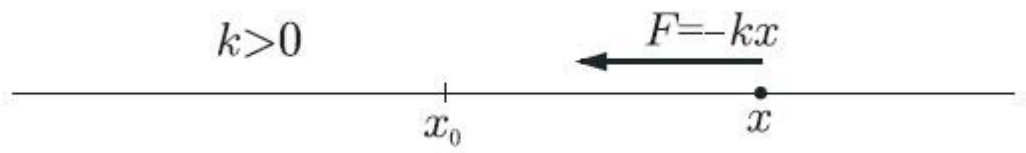
koja je oblika $y = ax^2 + b$ dakle parabola. Treća diferencijalna jednadžba za z nije se pojavila u prethodnom slučaju jer je gibanje ravninsko, pa smo koordinatne osi x, y i z mogli postaviti tako da se ravnina xy poklapa s ravninom gibanja, što znači da je $z(t) = 0$. Ako istražujemo pravocrtno gibanje, onda koordinatnu os x možemo postaviti u pravac gibanja, pa u tom slučaju samo jedna diferencijalna jednadžba:

$$m\frac{d^2x}{dt^2} = F$$

opisuje gibanje jer je tada $y(t) = z(t) = 0$. Ako intenzitet kojim sila djeluje na česticu ovisi samo o položaju čestice x, a ne ovisi o vremenu t, onda je F funkcija jedne varijable x.

1.1.2 Jednadžba harmonijskog oscilatora

Gibanje harmonijskog oscilatora (npr. opruge) jest gibanje pod utjecajem sile koja je proporcionalna otklonu od ravnotežnog položaja $x = 0$, a usmjerena je prema tom položaju (Slika 1.2).



Slika 1.2 Otklon iz ravnotežnog položaja i sila

Harmonijske oscilacije opisane su stoga jednadžbom:

$$m\frac{d^2x}{dt^2} = -kx \quad (k>0)$$

ili, uz uobičajenu pokratu $\omega = \sqrt{\frac{k}{m}}$, jednažbom:

$$\frac{d^2x}{dt^2} = -\omega^2 x.$$

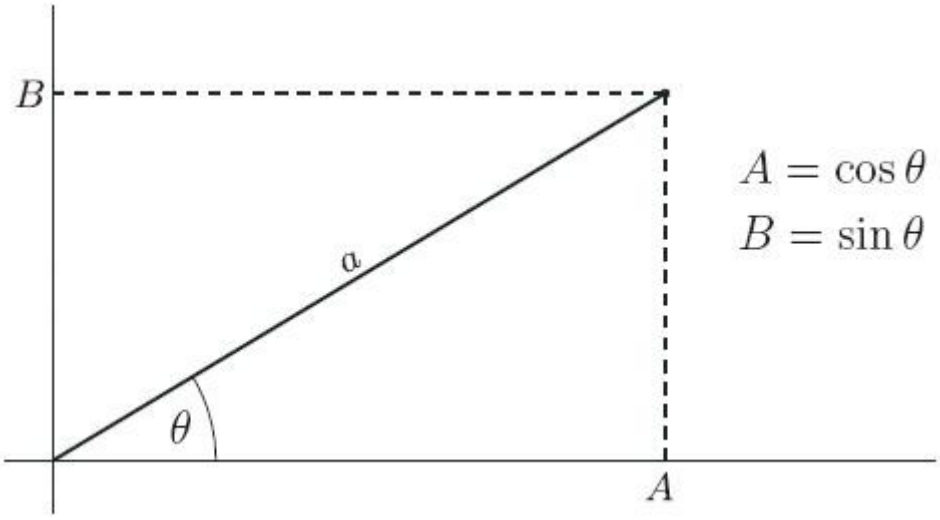
Poznato je da svako rješenje ove jednadžbe ima oblik:

$$x = A\cos\omega t + B\sin\omega t,$$

gdje su A i B proizvoljne konstante. Rješenja se mogu zapisati i u obliku:

$$x = a\cos(\omega t - \vartheta),$$

gdje su (a, ϑ) polarne koordinate od *(A, B)*, prikazano na slici (Slika 1.3).



Slika 1.3 Polarne koordinate

Uz početne uvjete $x(0) = x_0$ i $\frac{dx}{dt}(0) = v_0$ imamo jedinstveno rješenje

$$x = x_0\cos\omega t + \frac{v_0}{\omega}\sin\omega t.$$

Fizičari očekuju da je klasično gibanje čestice u zadanom polju sile potpuno determinirano ako su određeni početni položaj i početna brzina čestice. Jedinstvenost gornjeg rješenja pokazuje da je taj zahtjev zadovoljen. Diferencijalnu jednadžbu harmonijskog oscilatora možemo riješiti i na sljedeći način. Napišemo li jednadžbu $\frac{d^2x}{dt^2} = -\omega^2 x$ u obliku:

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

i pomnožimo je s $\frac{dx}{dt}$, dobit ćemo:

$$m\frac{dx}{dt}\frac{d^2x}{dt^2} + m\omega^2 x\frac{dx}{dt} = 0.$$

Uzmemo li u obzir da je

$$\frac{d}{dt}\left(\frac{dx}{dt}\right)^2 = 2\frac{dx}{dt}\frac{d^2x}{dt^2} \quad \text{i} \quad \frac{d}{dt}x^2 = 2x\frac{dx}{dt},$$

slijedi

$$\frac{d}{dt}\left(\frac{1}{2}m\left(\frac{dx}{dt}\right)^2 + \frac{1}{2}m\omega^2 x^2\right) = 0,$$

tj.

$$\frac{1}{2}m\left(\frac{dx}{dt}\right)^2 + \frac{1}{2}m\omega^2 x^2 = E,$$

gdje je E (kao zbroj kvadrata) pozitivna konstanta. Gornju jednadžbu, koju smo izveli iz jednadžbe harmonijskog oscilatora, zovemo jednadžbom energije harmonijskog oscilatora. Naime,

$$E_k = \frac{1}{2}mv^2 \quad \text{i} \quad E_p = \frac{1}{2}m\omega^2 x^2$$

jesu kinetička i potencijalna energija harmonijskog oscilatora, pa jednadžba energije znači da se ukupna energija harmonijskog oscilatora ne mijenja, $E_k + E_p = E = konstanta$. Zapamtimo da se jednadžba energije izvodi iz Newtonove jednadžbe gibanja množenjem sa $\frac{dx}{dt}$ te da se gledano matematički radi o svođenju jednadžbe drugog reda (u kojoj se pojavljuje druga derivacija) na jednostavniju jednadžbu prvog reda (u kojoj se pojavljuje samo prva derivacija). Jednadžbu energije, koja je prvoga reda, lako je riješiti. Iz nje slijedi:

$$\left(\frac{dx}{dt}\right)^2 = \frac{2E}{m} - \omega^2 x^2 = \omega^2\left(\frac{2E}{\omega^2 m} - x^2\right),$$

$$\frac{dx}{dt} = \omega\sqrt{\frac{2E}{\omega^2 m} - x^2},$$

$$\frac{dt}{dx} = \frac{1}{\omega}\frac{1}{\sqrt{a^2 - x^2}},$$

uz

$$a^2 = \frac{2E}{\omega^2 m}.$$

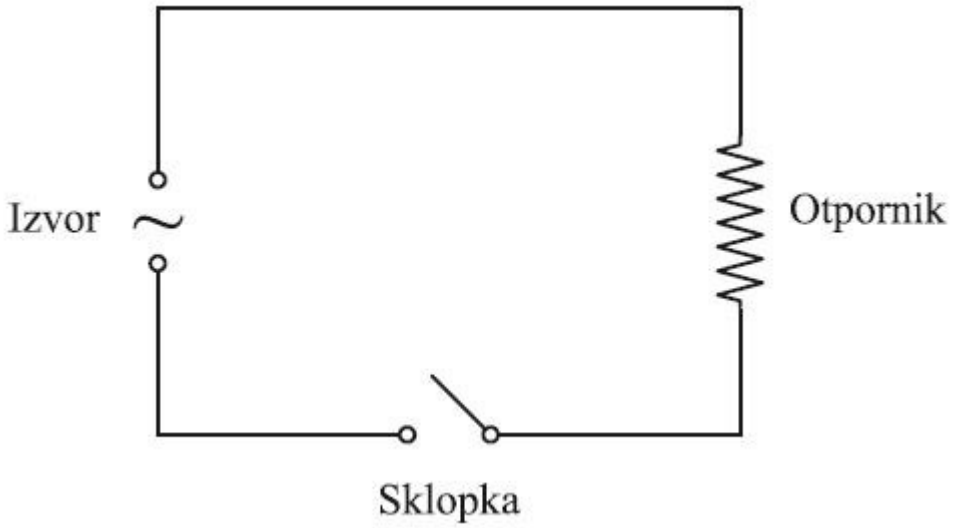
Dakle,

$$t = \frac{1}{\omega}\int \frac{dx}{\sqrt{a^2 - x^2}} = \frac{1}{\omega}\arcsin\frac{x}{a} + c,$$
$$\omega(t - c) = \arcsin\frac{x}{a},$$
$$\frac{x}{a} = \sin(\omega t - \omega c),$$
$$x = a\sin(\omega t - \gamma),$$

uz $\gamma = \omega c$ i $a^2 = \frac{2E}{\omega^2 m}$. Naravno, to je poznato rješenje $x = a\cos(\omega t - \vartheta)$, , uz $\gamma = \vartheta - \frac{\pi}{2}$. Uočimo da je kvadrat amplitude oscilacija a^2 proporcionalan energiji harmonijskog oscilatora *E*.

1.1.3 Jednadžba strujnog kruga

Najjednostavniji strujni krug (Slika 1.4) sadrži izvor električne energije, npr. generator ili bateriju te otpornik koji se koristi tom energijom, npr. žarulju.



Slika 1.4 Najjednostavniji strujni krug

Zatvorimo li sklopku, struja *J* poteći će otpornikom, što će uzrokovati pad napona (tj. električni potencijali na krajevima otpornika bit će različiti). Eksperimentalno se pokazuje da vrijedi Ohmov zakon:

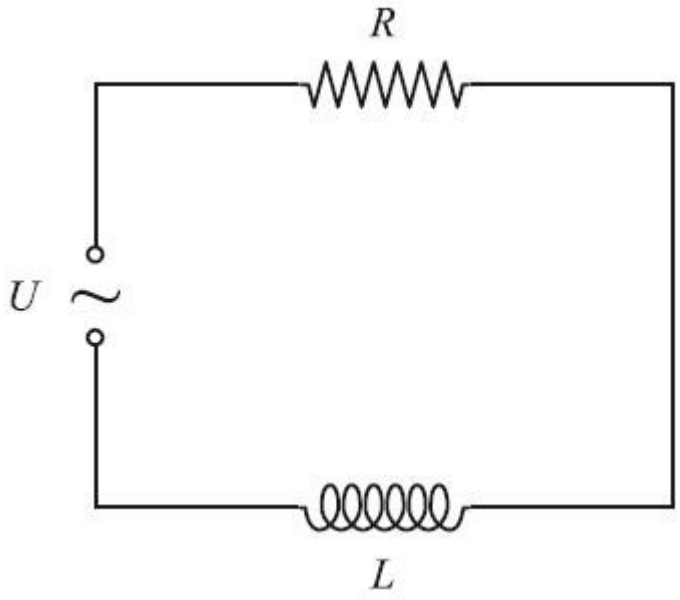
$$U_R = RJ,$$

tj. pad napona duž otpornika proporcionalan je trenutnoj struji *J*. Konstanta proporcionalnosti *R* zove se otporom otpornika. Važan element strujnog kruga može biti zavojnica. On se opire promjeni struje, analogno masi koja se opire promjeni gibanja. Eksperimentom se pokazuje da vrijedi sljedeći zakon:

$$U_L = L\frac{dJ}{dt},$$

tj. pad napona duž zavojnice proporcionalan je trenutnoj brzini promjene struje *J*. Konstanta proporcionalnosti *L* zove se indukcijom zavojnice. Drugi Kirchhoffov zakon (o naponima) ustvrđuje da je ukupni pad napona duž zatvorenog strujnog kruga jednak nuli, tj. da je napon izvora kojim se napaja strujni krug jednak zbroju padova napona po ostalim elementima strujnoga kruga (Slika 1.5). Dakle, tok struje *J* u strujnom krugu s otporom *R*, indukcijom *L* i izvorom konstantnog napona *U* određen je diferencijalnom jednadžbom:

$$U = L\frac{dJ}{dt} + RJ.$$



Slika 1.5 Strujni krug s otporom R i indukcijom L

Ta se diferencijalna jednadžba razlikuje od diferencijalne jednadžbe $\frac{dy}{dt} = ky$, ali uz supstituciju $h = RJ - U$ ona će, zbog $\frac{dh}{dt} = R\frac{dJ}{dt}$, prijeći u jednadžbu

$$\frac{dh}{dt} = -\frac{R}{L}h,$$

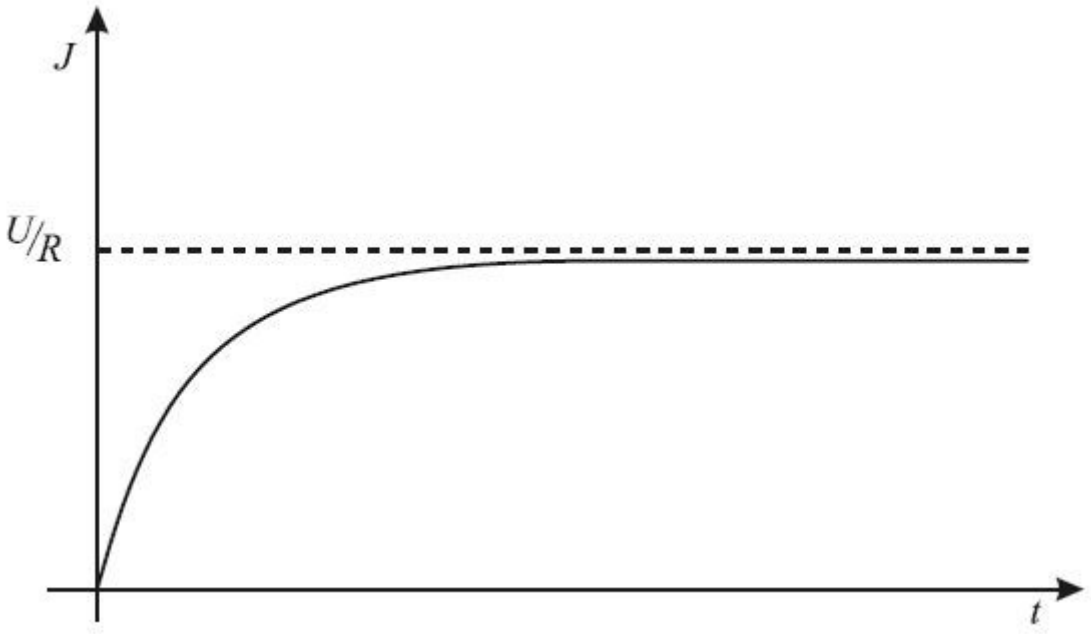
koja je oblika $\frac{dy}{dt} = ky$, uz $k = -\frac{R}{L}$. Njezino je rješenje oblika $y = y(t_o)e^{k(t - t_0)}$ ili $y = y(0)e^{kt}$ ako je $t_0 = 0$. Dakle,

$$h = h(0)e^{-\frac{R}{L}t}.$$

Ako strujni krug priključujemo na izvor u trenutku $t = 0$, onda je $J(0) = 0$, tj. $h(0) = -U$, pa je

$$h = -U e^{-\frac{R}{L}t}, \quad RJ - U = -Ue^{-\frac{R}{L}t}, \quad \text{tj.} \quad J = \frac{U}{R}\left(1 - e^{-\frac{R}{L}t}\right).$$

Graf funkcije *J(t)* prikazan je na slici (Slika 1.6).



Slika 1.6 Graf funkcije J(t)

Ako je strujni krug priključen na izvor izmjeničnog napona $U = U_0 \cos \omega t$, onda *J(t)* zadovoljava diferencijalnu jednadžbu:

$$L\frac{dJ}{dt} + RJ = U_0 \cos \omega t.$$

Ona je nešto složenija, ali se može riješiti upotrebom kompleksnih funkcija realnog argumenta. Naime, stvarni napon $U_0 \cos \omega t$ možemo shvatiti kao realni dio kompleksnog "napona":

$$\bar{U} = U_0 e^{i\omega t} = U_0 \cos \omega t + i U_0 \sin \omega t,$$

a stvarna struja *J* kao realni dio kompleksne struje

$$\bar{J} = J + iH.$$

Ako je $\bar{J}$ rješenje kompleksne diferencijane jednadžbe

$$L\frac{d\bar{J}}{dt} + R\bar{J} = \bar{U},$$

onda je

$$L\frac{d(J + iH)}{dt} + R(J + iH) = U_0\cos\omega t + iU_0\sin\omega t,$$
$$\left(L\frac{dJ}{dt} + RJ\right) + i\left(L\frac{dH}{dt} + RH\right) = U_0\cos\omega t + iU_0\sin\omega t,$$

odakle slijedi

$$L\frac{dJ}{dt} + RJ = U_0\cos\omega t.$$

Realni dio rješenja kompleksne diferencijalne jednadžbe jest rješenje naše početne realne diferencijalne jednadžbe. Dakle, riješimo li kompleksnu diferencijalnu jednadžbu, riješili smo naš problem. No, lako vidimo da je funkcija $\bar{J} = J_0 e^{i(\omega t + \beta)}$ (s još neodređenim realnim parametrima J_0 i β) jedno rješenje naše kompleksne diferencijalne jednadžbe ako je

$$Li\omega\bar{J} + R\bar{J} = \bar{U}, \text{ tj. ako je } \bar{J} = \frac{\bar{U}}{R + iL\omega}.$$

To će biti ispunjeno ako je

$$|\bar{J}| = \left|\frac{\bar{U}}{R + iL\omega}\right| = \frac{|\bar{U}|}{|R + iL\omega|}, \text{ tj. } J_0 = \frac{U_0}{\sqrt{R^2 + L^2\omega^2}}$$

i ako je

$$arg\,\bar{J} = arg\frac{\bar{U}}{R + iL\omega} = arg\,\bar{U} - arg(R + iL\omega),$$

tj.

$$\omega t + \beta = \omega t - arctg\frac{L\omega}{R}, \quad \beta = -arctg\frac{L\omega}{R}.$$

Dakle, jedno kompleksno rješenje naše kompleksne diferencijalne jednadžbe je

$$\bar{J} = \frac{U_0}{\sqrt{R^2 + L^2\omega^2}} e^{i\left(\omega t - arctg\frac{L\omega}{R}\right)},$$

odakle slijedi da je i jedno rješenje polazne realne diferencijalne jednadžbe njegov realni dio

$$J = \frac{U_0}{\sqrt{R^2 + L^2\omega^2}}\cos\left(\omega t - arctg\frac{L\omega}{R}\right).$$

1.1.4 Zaključak o diferencijalnim jednadžbama u fizici

Na ovim jednostavnim primjerima smo vidjeli neke od primjena diferencijalnih jednadžbi u fizici. Navedeni primjeri su bili analitički rješivi. Za slučajeve kada analitički ne možemo riješiti problem, tada upotrebljavamo numeričke metode kako bismo vidjeli ponašanje sustava koji nam je zadan. U ovom diplomskom radu ćemo razmatrati numeričke metode.

1.2 Uvod u obične diferencijalne jednadžbe

Rješavanje diferencijalnih jednadžbi je problem koji se često javlja u raznim primjenama. Dok je nekim jednadžbama rješenje moguće eksplicitno izraziti pomoću poznatih funkcija, daleko su brojnije one jednadžbe za koje ne možemo napisati egzaktno rješenje. Stoga takve jednadžbe rješavamo numerički. Ponekad je čak brže i jednostavnije izračunati rješenje numeričkim putem umjesto dugotrajnim analitičkim postupkom.

Opisat ćemo nekoliko najčešćih numeričkih metoda za rješavanje običnih diferencijalnih jednadžbi (skraćeno ODJ) oblika

$$y'(x) = f(x, y(x)), \quad y \in (a, b) \tag{1}$$

uz zadani početni uvjet $y(a) = y_0$ ili uz zadani rubni uvjet $r(y(a), y(b)) = 0$, gdje je r neka zadana funkcija. Dakle, obična diferencijalna jednadžba prvog reda je jednadžba koja sadrži samo jednu nezavisnu varijablu t, jednu zavisnu varijablu y (koja je u stvari funkcija od t) te derivaciju zavisne varijable po nezavisnoj $\frac{dy}{dt}$. Matematičkim rječnikom rečeno:

$$F\left(t, y, \frac{dy}{dt}\right)$$

Tu implicitnu jednadžbu možemo u principu riješiti po $\frac{dy}{dt}$ i dobiti je u eksplicitnom obliku:

$$\frac{dy}{dt} = f(t, y)$$

Sustav običnih diferencijalnih jednadžbi je općenitiji problem:

$$y_1' = f_1(x, y_1, \ldots, y_n).$$

$$y_2' = f_2(x, y_1, \ldots, y_n),$$

$$\vdots$$

$$y_n' = f_n(x, y_1, \ldots, y_n).$$

Međutim, koristeći vektorsku notaciju

$$\boldsymbol{y} = \left[y_1, \ldots, y_n\right]^T$$
$$\boldsymbol{f} = \left[f_1, \ldots, f_n\right]^T$$

sustav pišemo u obliku analognom jednadžbi (1):

$$\boldsymbol{y}'(x) = \boldsymbol{f}(x, \boldsymbol{y}(x)),$$

te primijenjujemo iste numeričke metode kao za rješavanje diferencijalne jednadžbe (1) vodeći računa o tome da se umjesto skalarnih funkcija y i t javljaju vektorske funkcije $\boldsymbol{y}$ i $\boldsymbol{f}$.

Diferencijalne jednadžbe višeg reda

$$y^{(n)} = f(x, y, y', y'', \ldots, y^{(n-1)})$$

supstitucijama

$$y_1 = y, \quad y_2 = y'', \quad \ldots, y_n = y^{(n-1)}$$

svodimo na sustav jednadžbi prvog reda:

$$y_1' = y' = y_2$$
$$y_2' = y'' = y_3$$
$$\vdots$$
$$y_{n-1}' = y^{(n-1)} = y_n$$
$$y_n' = y^{(n)} = f(x, y, y', y'', \ldots, y^{(n-1)}) = f(x, y_1, y_2, y_3, \ldots, y_n)$$

te i u ovom slučaju koristiti metode razvijene za diferencijalnu jednadžbu (1).

I za sustav diferencijalnih jednadžbi i za jednadžbu višeg reda razlikujemo inicijalni (početni ili Cauchyjev) problem i rubni problem.

1.3 Inicijalni problem za običnu diferencijalnu jednadžbu – Eulerova metoda

Eulerova metoda je zasigurno najjednostavnija metoda za rješavanje inicijalnog problema za ODJ oblika

$$y' = f(x, y), \quad y(a) = y_0.$$

Metoda se zasniva na ideji da se y' u gornjoj jednadžbi zamijeni s podijeljenom razlikom

$$y'(x) = \frac{y(x + h) - y(x)}{h} + O(h),$$

pa rješenje diferencijalne jednadžbe zadovoljava

$$y(x + h) = y(x) + h y'(x) + O(h^2) = y(x) + h f(x, y(x)) + O(h^2).$$

Zanemarivanjem kvadratnog člana u gornjem razvoju dobivamo aproksimaciju

$$y(x + h) \approx y(x) + h f(x, y(x)). \qquad \textbf{(2)}$$

Ova formula je točnija što je h manji, tako da aproksimacija

$$y(b) \approx y(a) + (b - a) h f(a, y(a)) = y_0 + h f(a, y_0)$$

može biti jako neprecizna. Stoga interval $[a, b]$ podijelimo na n jednakih dijelova te stavimo

$$h = \frac{b-a}{n}, \quad x_i = a + ih, \quad i = 0, \ldots, n.$$

Korištenjem (2), prvo aproksimiramo rješenje u točki $x_1 = a + h$:

$$y(x_1) \approx y_1 = y_0 + h f(x_0, y_0).$$

Dobivenu aproksimaciju y_1 iskoristimo za računanje aproksimacije rješenja u točki $x_2 = x_1 + h$:

$$y_2 = y_1 + h f(x_1, y_1),$$

te postupak ponavljamo sve dok ne dođemo do kraja intervala $b = x_n$.

Opisani postupak nazivamo Eulerova metoda, i možemo ga kraće zapisati rekurzijom

$$y_{i+1} = y_i + h f(x_i, y_i), \quad i = 1, \ldots, n,$$

gdje je početni uvjet y_0 zadan kao inicijalni uvjet diferencijalne jednadžbe. Dobivene vrijednosti y_i su aproksimacije rješenja diferencijalne jednadžbe u točkama x_i .

1.4 Runge-Kuttine metode

Koristeći sličnu ideju kao u Eulerovoj metodi, diferencijalnu jednadžbu

$$y' = f(x, y), \quad y(a) = y_0 \tag{3}$$

na intervalu $[a, b]$, možemo rješavati tako da podijelimo interval $[a, b]$ na n jednakih podintervala, označivši

$$h = \frac{b-a}{n}, \quad x_1 = a + i h, \quad i = 0, \ldots, n.$$

Sada y_{i+1} aproksimaciju rješenja u točki x_{i+1} , računamo iz y_i korištenjem aproksimacije oblika

$$y(x+h) \approx y(x) + h\Phi(x, y(x), h; f), \tag{4}$$

te dobivamo rekurziju:

$$y_{i+1} = y_i + h\Phi(x_i, y_i, h; f), \quad i = 0, \ldots, n-1. \tag{5}$$

Funkciju Φ nazivamo **funkcija prirasta**, a različit izbor te funkcije definira različite metode. Uočimo da je funkcija f iz diferencijalne jednadžbe (3) parametar od Φ (tj. Φ ovisi o f). Tako je npr. u Eulerovoj metodi

$$\Phi(x, y, h; f) = f(x, y).$$

Metode oblika (5) zovemo **jednokoračne metode** (jer za aproksimaciju y_{i+1} koristimo samo vrijednosti y_i u prethodnoj točki x_i, tj. u jednom koraku dobijemo y_{i+1} iz y_i). Da bismo pojednostavili zapis, ubuduće ćemo f izostaviti kao argument funkcije Φ.

O odabiru funkcije Φ ovisi i točnost metode. Za očekivati je da ako izaberemo Φ tako da aproksimacija točnog rješenja $y(x+h)$ dana s (4) bude što točnija, da će točnija biti i aproksimacija y_i za $y(x_i)$ dana rekurzijom (5). Pogreška aproksimacije (4):

$$\tau(x; h) = \Delta(x; h) - \Phi(x, y(x), h), \qquad \textbf{(6)}$$

gdje je $y(x)$ točno rješenje diferencijalne jednadžbe (3) i

$$\Delta(x; h) = \frac{y(x+h) - y(x)}{h},$$

nazivamo **lokalna pogreška diskretizacije**.

Red metode odgovara njezinoj točnosti. Općenito, za jednokoračne metode kažemo da su reda p ako je

$$\tau(x; h) = O(h^p).$$

Što je veći p metoda je točnija, a to postižemo odabirom funkcije Φ.

Pod točnošću metode podrazumijevamo ponašanje pogreške $y(x_i) - y_i$. Da bismo pojednostavnili argumentaciju, promatrat ćemo pogrešku u fiksiranoj točki b. Ako je jednokoračna metoda reda p, tada je

$$y(b) - y_n = O(h^p).$$

Uočimo da je $h = \frac{b - a}{n}$ te da je y_n uvijek (za svaki n) aproksimacija za $y(b)$.

Najpoznatije jednokoračne metode su svakako Runge-Kuttine metode. Kod njih je funkcija Φ oblika

$$\Phi(x, y, h) = \sum_{j=1}^{r} \omega_j k_j(x, y, h),$$

a k_j su zadani s

$$k_j(x, y, h) = f\left(x + c_j h, y + h\sum_{l=1}^{r} a_{jl} k_l(x, y, h, f)\right), \quad j = 1, \ldots, r. \tag{7}$$

Broj r zovemo broj stadija Runge-Kuttine metode, i on označava koliko puta moramo računati funkciju f u svakom koraku.

Različit izbor koeficijenata ω_j, c_j i a_{jl} definira različite metode. Ovi koeficijenti se najčešće biraju tako da red metode bude što je moguće veći. Iz izraza (7) vidimo da se k_j nalazi na lijevoj i na desnoj strani jednadžbe, tj. zadan je implicitno te govorimo o **implicitnoj** Runge-Kuttinoj metodi. U praksi se najviše koriste metode gdje je $a_{jl} = 0$ za $l \geq j$. Tada k_j možemo izračunati preko $k_1, \ldots, k_{j-1}$, tj. funkcije k_j su zadane eksplicitno. Takve RK metode nazivamo **eksplicitnima**. Nadalje, obično se dodaje uvjet

$$\sum_{l=1}^{r} a_{jl} = c_j$$

Primjer odabira koeficijenata prikazat ćemo na RK metodi s dva stadija:

$$\begin{aligned}
&\Phi(x, y, h) = \omega_1 k_1(x, y, h) + \omega_2 k_2(x, y, h),\\
&k_1(x, y, h) = f(x, y),\\
&k_2(x, y, h) = f(x + ah, y + ahk_1)
\end{aligned}$$

Razvojem k_2 u Taylorov red po varijabli h dobivamo

$$k_2(x, y, h) = f + h(f_x a + f_y a f) + \frac{h^2}{2}(f_{xx} a^2 + 2 f_{xy} a^2 f + f_{yy} a^2 f^2) + O(h^3),$$

gdje su f_x i f_y prve parcijalne derivacije funkcije $f = f(x, y)$ po x, odnosno y, a f_{xx} , f_{xy} i f_{yy} odgovarajuće druge parcijalne derivacije. Razvoj rješenja diferencijalne jednadžbe $y(x)$ ima oblik

$$y(x+h) = y(x) + hf + \frac{h^2}{2}(f_x + f_y f) + \frac{h^3}{6}[f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y(f_x + f_y f)] + O(h^4).$$

Ovdje smo koristili da je $y(x)$ rješenje diferencijalne jednadžbe:

$$y'(x) = f(x,y) = f,$$

te pravila za deriviranje

$$y''(x) = f_x + f_y f,$$
$$y'''(x) = f_{xx} + 2f_{xy}fr + f_{yy}f^2 + f_y(f_x + f_y f).$$

Sada je lokalna pogreška diskretizacije jednaka

$$\frac{y(x+h) - y(x)}{h} - \Phi(x, y(x), h) =$$
$$= \frac{y(x+h) - y(x)}{h} - (\omega_1 k_1(x,y,h) + \omega_2 k_2(x,y,h))$$
$$= (1 - \omega_1 - \omega_2)f + h(f_x + f_y f)\left(\frac{1}{2} - \omega_2 a\right)$$
$$+ h^2\left[(f_{xx} + 2f_{xy}f + f_{yy}f^2)\cdot\left(\frac{1}{6} - \frac{\omega_2 a^2}{2}\right) + \frac{1}{6}f_y(f_x + f_y f)\right] + O(h^3).$$

Da bi metoda bila reda 1 koeficijente treba odabrati tako da se poništi prvi član u gornjem razvoju:

$$1 - \omega_1 - \omega_2 = 0.$$

Ukoliko je zadovoljeno i

$$\frac{1}{2} - \omega_2 a = 0$$

metoda će biti reda 2. Uvođenjem slobodnog koeficijenta t rješenje ove dvije jednadžbe možemo napisati u obliku:

$$\omega_2 = t \neq 0, \quad \omega_1 = 1 - t, \quad a = \frac{1}{2t}.$$

Uočimo da t ne možemo odabrati tako da poništimo i član uz h^2 tako da metoda bude reda 3. Ukoliko $\omega_2 = 0,$ radi se o metodi s jednim stadijem, i to upravo o Eulerovoj metodi.

Za $t = \frac{1}{2}$ dobivamo Heunovu metodu:

$$\begin{aligned} \Phi &= \frac{1}{2}(k_1 + k_2), \\ k_1 &= f(x, y), \\ k_2 &= f(x + h, y + hk_1), \end{aligned}$$

dok se za $t = 1$ dobiva modificirana Eulerova metoda:

$$\Phi = f\left(x + \frac{h}{2}, y + \frac{h}{2} f(x, y)\right).$$

Najrašireniye su metode sa četiri stadija. Odgovarajuće jednadžbe koje moraju zadovoljavati koeficijenti RK-4 metoda su:

$$\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1, \tag{8}$$

$$\omega_2 c_2 + \omega_3 c_3 + \omega_4 c_4 = \frac{1}{2}, \tag{9}$$

$$\omega_2 c_2^2 + \omega_3 c_3^2 + \omega_4 c_4^2 = \frac{1}{3}, \tag{10}$$

$$\omega_3 c_2 a_{32} + \omega_4 (c_2 a_{42} + c_3 a_{43}) = \frac{1}{6}, \tag{11}$$

$$\omega_2 c_2^3 + \omega_3 c_3^3 + \omega_4 c_4^3 = \frac{1}{4}, \tag{12}$$

$$\omega_3 c_2^2 a_{32} + \omega_4 (c_2^2 a_{42} + c_3^2 a_{43}) = \frac{1}{12}, \tag{13}$$

$$\omega_3 c_2 c_3 a_{32} + \omega_4 (c_2 a_{42} + c_3 a_{43}) c_4 = \frac{1}{8}, \tag{14}$$

$$\omega_4 c_2 a_{32} a_{43} = \frac{1}{24,} \tag{15}$$

gdje je

$$
\begin{aligned}
c_1 &= 0, \\
c_2 &= a_{21}, \\
c_3 &= a_{31} + a_{32}, \\
c_4 &= a_{41} + a_{42} + a_{43}.
\end{aligned}
$$

Uvjet (8) treba biti zadovoljen da bi metoda bila reda 1, uvjet (9) za red 2, uvjeti (10) do (11) za red 3, dok za red 4 trebaju biti ispunjeni i uvjeti (12) do (15). Ukupno imamo 10 koeficijenata i 8 jednadžbi ukoliko je metoda reda 4. Za metodu s tri stadija uvrštavanjem

$$
c_4 = a_{41} = a_{42} = a_{43} = \omega_4 = 0
$$

dobivamo 9 koeficijenata i 7 jednadžbi. Metoda s četiri stadija može postići najviše red četiri, tj. ne možemo dva stupnja slobode iz sustava jednadžbi koristiti da red metode podignemo na pet.

Općenito, za metode s jednim, dva, tri i četiri stadija najveći mogući red metode odgovara broju stadija. Za metode s 5, 6 i 7 stadija najveći mogući red je 4, 5 i 6, dok je za metode 8 i više stadija najveći mogući red barem za dva manji od broja stadija. To je razlog zašto su metode s četiri stadija najpopularnije. Red je 4, a da bismo postigli red 5 trebamo povećati broj stadija barem za dva, što povećava složenost metode.

Evo nekoliko primjeraRK-4 metoda. Najpopularnija je “klasična” Runge-Kutta metoda, koja se u literaturi najčešće naziva Runge-Kutta ili RK-4 metoda (iako je samo jedna u nizu Runge-Kutta metoda):

$$
\begin{aligned}
\Phi &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\
k_1 &= f(x, y), \\
k_2 &= f\left(x + \frac{h}{2}, y + \frac{h}{2}k_1\right), \\
k_3 &= f\left(x + \frac{h}{2}, y + \frac{h}{2}k_2\right), \\
k_4 &= f\left(x + h, y + hk_3\right).
\end{aligned}
$$

Spomenimo još i 3/8-sku metodu:

$$\begin{aligned}
\Phi &= \frac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4),\\
k_1 &= f(x,y),\\
k_2 &= f\left(x + \frac{h}{3}, y + \frac{h}{3}k_1\right),\\
k_3 &= f\left(x + \frac{2}{3}h, y - \frac{h}{3}k_1 + hk_2\right),\\
k_4 &= f\left(x + h, y + h(k_1 - k_2 + k_3)\right)
\end{aligned}$$

i Gillovu metodu:

$$\begin{aligned}
\Phi &= \frac{1}{6}(k_1 + (2 - \sqrt{2})k_2 + (2+\sqrt{2})k_3 + k_4),\\
k_1 &= f(x,y),\\
k_2 &= f\left(x + \frac{h}{2}, y + \frac{h}{2}k_1\right),\\
k_3 &= f\left(x + \frac{h}{2}, y + h\frac{\sqrt{2} - 1}{2}k_1 + h\frac{2 - \sqrt{2}}{2}k_2\right),\\
k_4 &= f\left(x + h, y - h\frac{\sqrt{2}}{2}k_2 + h\frac{2 + \sqrt{2}}{2}k_3\right).
\end{aligned}$$

1.5 Brzinski Verlet algoritam

Brzinski Verlet algoritam je jedan od najjednostavnijih algoritama za integriranje, zasnovao ga je L. Verlet u ranim danima molekularne simulacije. Jednadžbe algoritma su:

$$\begin{aligned}
\boldsymbol{r}_{n+1} &= \boldsymbol{r}_n + \boldsymbol{v}_n \Delta t + \frac{\boldsymbol{f}_n \Delta t^2}{2m} + O(\Delta t^3)\\
\boldsymbol{v}_{n+1} &= \boldsymbol{v}_n + \frac{\Delta t}{2m}\left[\boldsymbol{f}_{n+1} + \boldsymbol{f}_n\right] + O(\Delta t^3)
\end{aligned}$$

Iz jednadžbi se vidi da je u svakom trenutku brzinskim Verlet algoritmom dana brzina i položaj molekule. Ove dvije jednadžbe se mogu razbiti na tri jednadžbe:

$$\boldsymbol{v}_{n+\frac{1}{2}} = \boldsymbol{v}_n + \frac{\boldsymbol{f}_n \Delta t}{2m}$$

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \boldsymbol{v}_{n+\frac{1}{2}} \Delta t$$

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_{n+\frac{1}{2}} + \frac{\boldsymbol{f}_{n+1} \Delta t}{2m}$$

Zašto je bilo potrebno napraviti iz dvije jednadžbe tri? Zato što je za računanju brzine $\boldsymbol{v}_{n+1}$ potrebna sila $\boldsymbol{f}_{n+1}$. Ako znamo da je potencijal funkcija od položaja

$$V(\boldsymbol{r}) \ ,$$

a sila negativna parcijalna derivacija potencijala po položaju

$$\boldsymbol{f}_{n+1} = - \frac{\partial V(\boldsymbol{r}_{n+1})}{\partial \boldsymbol{r}_{n+1}} \ ,$$

tada lako dođemo do sile $\boldsymbol{f}_{n+1}$ i izračunamo brzinu $\boldsymbol{v}_{n+1}$. Pogreška u danom algoritmu je uračunata članom O koji je funkcija od Δt^3 .

2. Primjena numeričkog rješavanja diferencijalnih jednadžbi na sustav vezanih harmoničkih oscilatora

2.1 Primjer sustava dva vezana harmonička oscilatora

Obična diferencijana jednadžba koju treba riješiti su dva harmonička oscilatora sa slabim harmoničkim vezanjem. Položaj dva oscilatora je određen sa dvije koordinate: x_1 i x_2 koje imaju jedinicu metar. Potencijalna energija je dana:

$$V(x_1, x_2) = \frac{1}{2}\alpha_1 x_1^2 + \frac{1}{2}\alpha_2 x_2^2 + \frac{1}{2}\alpha_{12}(x_1 - x_2)^2$$

gdje su α_1 , α_2 i α_{12} konstante elastičnosti. Za konstante elastičnosti uzmimo za naš primjer da su $\alpha_1 = 0.99\,J/m$, $\alpha_2 = 1.01\,J/m$ i $\alpha_{12} = 0.05\,J/m$. Masa oba harmonička oscilatora je 1 kg za svaki od njih. U početku (vrijeme t = 0), oba harmonička oscilatora su u stanju mirovanja sa koordinatama $x_1(0) = -4\,m$ i $x_2(0) = 0\,m$.

Slika 2.1 Sustav dvaju tijela i tri opruge

Na slici (Slika 2.1) prikazan je sustav dvaju tijela jednakih masa M, povezanih međusobno sa oprugom koja ima mnogo manju konstantu elastičnosti nego li opruge kojima su mase u interakciji sa zidovima.

2.1.1 Sile u sustavu vezanih harmoničkih oscilatora

2.1.1.1 Zakon očuvanja mehaničke energije

Rad *dW* sila polja teže koje djeluje na materijalnu točku, dajući joj kinetičku energiju, jednak je smanjenju potencijala

$$dV = -\frac{dE_p}{m}$$

ili potencijalne energije

$$dW = -dE_p \ , \qquad \textbf{(16)}$$

a povećanje dE_k kinetičke energije jednako je pozitivnoj promjeni rada

$$dW = dE_k \quad . \tag{17}$$

Imamo, dakle, izjednačivši desne strane jednadžbe (16) i (17),

$$dE_k = - dE_p$$

ili

$$dE_p + dE_k = 0 \quad . \tag{18}$$

Relacija (18) pokazuje da je – u konzervativnom sustavu, u kojem su sile derivacija potencijala – promjena zbroja potencijalne i kinetičke energije sustava jednaka nuli, odnosno da je zbroj kinetičke i potencijalne energije ovakva sustava konstantan.

Budući da se radi o sustavu u kojem su sile derivacije potencijala, to se jednadžbu (18) može integrirati od nekog početnog stanja 1 do nekog konačnog stanja 2, što daje

$$E_{p2} - E_{p1} + E_{k2} - E_{k1} = 0$$

ili

$$E_{p1} + E_{k1} = E_{p2} + E_{k2} \quad . \tag{19}$$

Relacija (19) , kao i relacija (18) pokazuje zakon očuvanja kinetičke i potencijalne energije ili, kratko rečeno, zakon očuvanja mehaničke energije.

Radi li se o sustavu koji je u dodiru s tijelima izvan sustava, a dodirom ne dolazi ni do sila trenja ni do vršenja rada, možemo zakon očuvanja mehaničke energije formulirati ovako:

> U sustavu u kojem su sile unutar sustava derivacije potencijala, a vanjske sile djeluju bez trenja i uravnotežene su silama unutar sustava, zbroj kinetičke i potencijalne energije je konstantan.

2.1.1.2 Pronalaženje sile iz potencijala

Dakle, dani sustav dvaju harmoničkih oscilatora je zatvoren, nema djelovanja vanjskih sila. Na njega ne djeluju sile koje bi nam dale do znanja da imamo gušeni harmonički oscilator ili tjerani harmonički oscilator. Dani sustav je konzervativan, tj. sile na dane oscilatore su derivacije potencijala.

Za prvi harmonički oscilator gledamo derivaciju potencijala $V(x_1, x_2)$ po koordinati x_1 . Pošto je potencijal funkcija od koordinata x_1 i x_2 možemo očekivati da će i sila biti funkcija od x_1 i x_2 , tj. $F(x_1, x_2)$. Dakle, deriviramo:

$$F_1(x_1, x_2) = -\frac{\partial V(x_1, x_2)}{\partial x_1} ,$$

gdje smo sa $F_1(x_1, x_2)$ naznačili da se radi o sili na prvi harmonički oscilator. Uvrstimo li u gornju formulu izraz za potencijal imamo:

$$F_1(x_1, x_2) = -\frac{\partial(\frac{1}{2}\alpha_1 x_1^2 + \frac{1}{2}\alpha_2 x_2^2 + \frac{1}{2}\alpha_{12}(x_1 - x_2)^2)}{\partial x_1} .$$

Rješavajući ovu parcijalnu derivaciju konačan izraz za silu na prvi harmonički oscilator je oblika:

$$F_1(x_1, x_2) = -\alpha_1 x_1 - \alpha_{12}(x_1 - x_2) .$$

Za drugi harmonički oscilator gledamo derivaciju potencijala $V(x_1, x_2)$ po koordinati x_2 . Pošto je potencijal funkcija od koordinata x_1 i x_2 možemo očekivati da će i sila biti funkcija od x_1 i x_2 , tj. $F(x_1, x_2)$. Dakle, deriviramo:

$$F_1(x_1, x_2) = -\frac{\partial V(x_1, x_2)}{\partial x_2} ,$$

gdje smo sa $F_2(x_1, x_2)$ naznačili da se radi o sili na drugi harmonički oscilator. Uvrstimo li u gornju formulu izraz za potencijal imamo:

$$F_2(x_1, x_2) = -\frac{\partial(\frac{1}{2}\alpha_1 x_1^2 + \frac{1}{2}\alpha_2 x_2^2 + \frac{1}{2}\alpha_{12}(x_1 - x_2)^2)}{\partial x_2}$$

Rješavajući ovu parcijalnu derivaciju konačan izraz za silu na drugi harmonički oscilator je oblika:

$$F_2(x_1, x_2) = -\alpha_2 x_2 + \alpha_{12}(x_1 - x_2) .$$

2.1.2 Rješavanje problema sustava dva vezana harmonička oscilatora u programskom jeziku C++

2.1.2.1 Primjer jednostavnog harmoničkog oscilatora

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

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

gdje je y zavisna varijabla, a ω parametar. Vidimo da jednadžba harmoničkog oscilatora ima samo jednu zavisnu varijablu y, ali čija druga derivacija je prisutna. Ovakve jednadžbe se zovu diferencijalne jednadžbe drugog reda. Sad ćemo pokazati kako se jednadžba harmoničkog oscilatora može napisati kao sustav dviju diferencijalnih jednadžbi prvog reda.

Označimo sa v brzinu od y, tj. $v = \frac{dy}{dt}$ i iskoristimo je u jednadžbi za harmonički oscilator. Dobijemo

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

što je diferencijalna jednadžba prvog reda, ali koja sadrži dvije zavisne varijable y i v. Dakle, vidimo da je sustav diferencijalnih jednadžbi prvog reda

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

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

koji sadrži dvije zavisne varijable y i v ekvivalentan jednadžbi harmoničkog oscilatora koja je diferencijalna jednadžba drugog reda koja sadrži jednu zavisnu varijablu y. Uočimo da tu nema nikakve koordinacije, jer koju god varijantu koristili potpuna specifikacija početnog uvjeta uključuje početne vrijednosti $y(t=0) = y_0$ i $v(t=0) = v_0$. Napomenimo da se (višestrukim) korištenjem ovog trika svaka diferencijalna jednadžba proizvoljnog reda n može napisati kao sustav od n diferencijalnih jednadžbi prvog reda. Reprezentacija modela sa sustavom diferencijalnih jednadžbi prvog reda puno je pogodnija za numeričke metode.

2.1.2.2 Eulerova metoda za sustave

Za dani sustav

$$\frac{dx}{dt} = f(t, x, y)$$
$$\frac{dy}{dt} = g(t, x, y) \quad ,$$

početni uvjet $(x(t_0), y(t_0)) = (x_0, y_0)$ i veličinu koraka Δt , Eulerovo približenje rješenja

$$(x(t_k), y(t_k)) = (x_k, y_k), \quad k = 0, 1, 2, \ldots$$

dobijemo rekurzivnim ponavljanjem slijedećeg računa:

$$t_{k+1} = t_k + \Delta t$$
$$x_{k+1} = x_k + f(t_k, x_k, y_k)\Delta t$$
$$y_{k+1} = y_k + g(t_k, x_k, y_k)\Delta t \quad .$$

2.1.2.3 Rješenje primjera jednostavnog harmoničkog oscilatora Eulerovom metodom za sustave

Jednadžba harmoničkog oscilatora je

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

a pripadajući sustav diferencijalnih jednadžbi prvog reda

$$\frac{dy}{dt} = v$$
$$\frac{dv}{dt} = -\omega^2 y \quad .$$

Primjenimo li Eulerovu metodu za sustave, tada imamo:

$$\frac{dy}{dt} = f(t, y, v) = v$$

$$\frac{dv}{dt} = g(t, y, v) = -\omega^2 y \quad .$$

Početni uvjet $(y(t_0), v(t_0)) = (y_0, v_0)$ i veličinu koraka Δt , Eulerovo približenje rješenja

$$(y(t_k), v(t_k)) = (y_k, v_k), \quad k = 0, 1, 2, \ldots$$

dobijemo rekurzivnim ponavljanjem slijedećeg računa:

$$t_{k+1} = t_k + \Delta t$$
$$y_{k+1} = y_k + f(t_k, y_k, v_k)\Delta t$$
$$v_{k+1} = v_k + g(t_k, y_k, v_k)\Delta t \quad .$$

Koja je inicijalna ideja Eulerove metode za sustave? Ako obratimo pažnju na dosadašnje formule, vidjet ćemo da na primjeru jednostavnog harmoničkog oscilatora možemo promatrati kako će se sustav ponašati s obzirom na početne uvjete (y_0, v_0) i parametar ω .

Ako pogledamo po koracima računanja vrlo lako ćemo shvatiti ovaj mehanizam. Dakle, na početku imamo (y_0, v_0) što koristimo da izračunamo (y_1, v_1) . Veličinu koraka Δt biramo po volji sa opaskom da o veličini koraka ovisi i točnost dobivenih rezultata, tj. što nam je vremenski korak Δt iznosom manji to su nam dobivena rješenja točnija. Za prvi korak imamo:

$$k = 0$$
$$t_0 = 0$$
$$t_1 = t_0 + \Delta t$$
$$y_1 = y_0 + f(t_0, y_0, v_0)\Delta t = y_0 + v_0\Delta t$$
$$v_1 = v_0 + g(t_0, y_0, v_0)\Delta t = v_0 - \omega^2 y_0\Delta t \quad .$$

Za drugi korak imamo:

$$k = 1$$
$$t_2 = t_1 + \Delta t$$
$$y_2 = y_1 + f(t_1, y_1, v_1)\Delta t = y_1 + v_1\Delta t$$
$$v_2 = v_1 + g(t_1, y_1, v_1)\Delta t = v_1 - \omega^2 y_1\Delta t \quad .$$

Ova numerička metoda se jednostavno može napisati u kodu bilo kojeg programskog jezika.

2.1.2.4 Rješenje problema sustava dva vezana harmonička oscilatora Eulerovom metodom za sustave

Uzmemo izraz za silu na prvi harmonički oscilator oblika:

$$F_1(x_1, x_2) = -\alpha_1 x_1 - \alpha_{12}(x_1 - x_2) \quad .$$

Prema drugom Newtonovom zakonu

$$\boldsymbol{F} = m \, d^2 \frac{\boldsymbol{r}}{dt^2}$$

što kad prilagodimo na jednodimenzionalni problem, imamo

$$F = m \, \frac{d^2 x}{dt^2} \quad .$$

Za prvi harmonički oscilator imamo:

$$F_1 = m_1 \, \frac{d^2 x_1}{dt^2} \quad \text{i} \quad F_1(x_1, x_2) = -\alpha_1 x_1 - \alpha_{12}(x_1 - x_2) \quad .$$

Uvrstimo li izraz za F_1 iz prve jednadžbe u drugu umjesto $F_1(x_1, x_2)$ dobijemo:

$$m_1 \, \frac{d^2 x_1}{dt^2} = -\alpha_1 x_1 - \alpha_{12}(x_1 - x_2) \quad .$$

Primjenom rastavljanja diferencijalne jednadžbe drugog reda na sustav dviju diferencijalnih jednadžbi prvog reda imamo:

$$\frac{dx_1}{dt} = v_1$$

$$\frac{dv_1}{dt} = -\frac{\alpha_1 x_1 + \alpha_{12}(x_1 - x_2)}{m_1} \quad .$$

Primjenimo li Eulerovu metodu za sustave, tada imamo:

$$\frac{dx_1}{dt} = f(t, x_1, v_1) = v_1$$
$$\frac{dv_1}{dt} = g(t, x_1, v_1) = -\frac{\alpha_1 x_1 + \alpha_{12}(x_1 - x_2)}{m_1} \quad .$$

Početni uvjet $(x_1(t_0), v_1(t_0)) = (x_{1\,0}, v_{1\,0})$ i veličinu koraka Δt, Eulerovo približenje rješenja

$$(x_1(t_k), v_1(t_k)) = (x_{1\,k}, v_{1\,k}), \quad k = 0, 1, 2, \ldots$$

dobijemo rekurzivnim ponavljanjem slijedećeg računa:

$$t_{k+1} = t_k + \Delta t$$
$$x_{1\,k+1} = x_{1\,k} + f(t_k, x_{1\,k}, v_{1\,k})\Delta t$$
$$v_{1\,k+1} = v_{1\,k} + g(t_k, x_{1\,k}, v_{1\,k})\Delta t \quad .$$

Prvi indeks u $x_{1\,k+1}$ i $v_{1\,k+1}$ nam govori da se radi o prvom harmoničkom oscilatoru, dok drugi indeks je broj koraka, analogno označavamo i za drugi harmonički oscilator. Ako pogledamo malo bolje izraz za silu na prvi harmonički oscilator, vidimo da se u izrazu nalazi i koordinata x_2 drugog harmoničkog oscilatora. Dakle, problem se malo zakomplicirao, jer ne gledamo samo prvi harmonički oscilator kao zaseban problem, već je usko vezan sa drugim harmoničkim oscilatorom.

Uzmemo izraz za silu na drugi harmonički oscilator oblika:

$$F_2(x_1, x_2) = -\alpha_2 x_2 + \alpha_{12}(x_1 - x_2) \quad .$$

Za drugi harmonički oscilator, prema drugom Newtonovom zakonu za jednodimenzionalan problem, imamo:

$$F_2 = m_2 \frac{d^2 x_2}{dt^2} \quad \text{i} \quad F_2(x_1, x_2) = -\alpha_2 x_2 + \alpha_{12}(x_1 - x_2) \quad .$$

Uvrstimo li izraz za F_2 iz prve jednadžbe u drugu umjesto $F_2(x_1, x_2)$ dobijemo:

$$m_2 \frac{d^2 x_2}{dt^2} = -\alpha_2 x_2 + \alpha_{12}(x_1 - x_2) \quad .$$

Primjenom rastavljanja diferencijalne jednadžbe drugog reda na sustav dviju diferencijalnih jednadžbi prvog reda imamo:

$$\frac{dx_2}{dt} = v_2$$
$$\frac{dv_2}{dt} = -\frac{\alpha_2 x_2 - \alpha_{12}(x_1 - x_2)}{m_2} \quad .$$

Primjenimo li Eulerovu metodu za sustave, tada imamo:

$$\frac{dx_2}{dt} = f(t, x_2, v_2) = v_2$$
$$\frac{dv_2}{dt} = g(t, x_2, v_2) = -\frac{\alpha_2 x_2 - \alpha_{12}(x_1 - x_2)}{m_2} \quad .$$

Početni uvjet $(x_2(t_0), v_2(t_0)) = (x_{2\,0}, v_{2\,0})$ i veličinu koraka Δt, Eulerovo približenje rješenja

$$(x_2(t_k), v_2(t_k)) = (x_{2\,k}, v_{2\,k}), \quad k = 0, 1, 2, \ldots$$

dobijemo rekurzivnim ponavljanjem slijedećeg računa:

$$t_{k+1} = t_k + \Delta t$$
$$x_{2\,k+1} = x_{2\,k} + f(t_k, x_{2\,k}, v_{2\,k})\Delta t$$
$$v_{2\,k+1} = v_{2\,k} + g(t_k, x_{2\,k}, v_{2\,k})\Delta t \quad .$$

Sada ćemo pogledati kako se numerički rješavaju prva dva koraka, da dobijemo ideju kako bi to mogli napisati u nekom programskom jeziku. Za prvi korak imamo:

$k = 0$

$t_0 = 0$

$t_1 = t_0 + \Delta t$

$$x_{1\,1} = x_{1\,0} + f(t_0, x_{1\,0}, v_{1\,0})\Delta t = x_{1\,0} + v_{1\,0}\Delta t$$

$$v_{1\,1} = v_{1\,0} + g(t_0, x_{1\,0}, v_{1\,0})\Delta t = v_{1\,0} - \frac{\alpha_1 x_{1\,0} + \alpha_{12}(x_{1\,0} - x_{2\,0})}{m_1}\Delta t$$

$$x_{2\,1} = x_{2\,0} + f(t_0, x_{2\,0}, v_{2\,0})\Delta t = x_{2\,0} + v_{2\,0}\Delta t$$

$$v_{2\,1} = v_{2\,0} + g(t_0, x_{2\,0}, v_{2\,0})\Delta t = v_{2\,0} - \frac{\alpha_2 x_{2\,0} - \alpha_{12}(x_{1\,0} - x_{2\,0})}{m_2}\Delta t \quad .$$

Za drugi korak imamo:

$$k = 1$$

$$t_2 = t_1 + \Delta t$$

$$x_{1\,2} = x_{1\,1} + f(t_1, x_{1\,1}, v_{1\,1})\Delta t = x_{1\,1} + v_{1\,1}\Delta t$$

$$v_{1\,2} = v_{1\,1} + g(t_1, x_{1\,1}, v_{1\,1})\Delta t = v_{1\,1} - \frac{\alpha_1 x_{1\,1} + \alpha_{12}(x_{1\,1} - x_{2\,1})}{m_1}\Delta t$$

$$x_{2\,2} = x_{2\,1} + f(t_1, x_{2\,1}, v_{2\,1})\Delta t = x_{2\,1} + v_{2\,1}\Delta t$$

$$v_{2\,2} = v_{2\,1} + g(t_1, x_{2\,1}, v_{2\,1})\Delta t = v_{2\,1} - \frac{\alpha_2 x_{2\,1} - \alpha_{12}(x_{1\,1} - x_{2\,1})}{m_2}\Delta t \quad .$$

Sada se možemo posvetiti pisanju koda u nekom programskom jeziku, pošto smo problem sustava vezanih hamoničkih oscilatora pojednostavnili i pripremili za rješavanje numeričkom metodom.

2.1.2.5 Rješenje problema sustava dva vezana harmonička oscilatora brzinskim Verlet algoritmom

Uzevši jednadžbe za brzinski Verlet algoritam:

$$\boldsymbol{v}_{n+\frac{1}{2}} = \boldsymbol{v}_n + \frac{\boldsymbol{f}_n \Delta t}{2m}$$

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \boldsymbol{v}_{n+\frac{1}{2}}\Delta t$$

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_{n+\frac{1}{2}} + \frac{\boldsymbol{f}_{n+1}\Delta t}{2m}$$

i jednadžbe za potencijal i sile oba harmonička oscilatora:

$$V(x_1, x_2) = \frac{1}{2}\alpha_1 x_1^2 + \frac{1}{2}\alpha_2 x_2^2 + \frac{1}{2}\alpha_{12}(x_1 - x_2)^2$$

$$F_1(x_1, x_2) = -\alpha_1 x_1 - \alpha_{12}(x_1 - x_2)$$

$$F_2(x_1, x_2) = -\alpha_2 x_2 + \alpha_{12}(x_1 - x_2)$$

jednostavno možemo raspisati razvoj po koracima. Dakle, za početni korak $n = 0$ prema brzinskom Verlet algoritmu za prvi hamonički oscilator imamo:

$$\boldsymbol{v}_{1\,\frac{1}{2}} = \boldsymbol{v}_{1\,0} + \left(-\alpha_1 x_{1\,0} - \alpha_{12}(x_{1\,0} - x_{2\,0}) \right) \times \frac{\Delta t}{2m_1}$$

$$x_{1\,1} = x_{1\,0} + \boldsymbol{v}_{1\,\frac{1}{2}} \Delta t \quad ,$$

gdje prvi indeks u $\boldsymbol{v}_{1\,\frac{1}{2}}$ i $x_{1\,0}$ označava prvi harmonički oscilatora a drugi indeks broj koraka, analogno i za drugi harmonički oscilator. Za drugi harmonički oscilator za početni korak $n = 0$ imamo:

$$\boldsymbol{v}_{2\,\frac{1}{2}} = \boldsymbol{v}_{2\,0} + \left(-\alpha_2 x_{2\,0} + \alpha_{12}(x_{1\,0} - x_{2\,0}) \right) \times \frac{\Delta t}{2m_2}$$

$$x_{2\,1} = x_{2\,0} + \boldsymbol{v}_{2\,\frac{1}{2}} \Delta t$$

Sada prije nego li izračunamo $\boldsymbol{v}_{1\,1}$ i $\boldsymbol{v}_{2\,1}$ uočimo da u izrazima za njihovo računanje se koriste $x_{1\,1}$ i $x_{2\,1}$ pa je stoga potrebno prije izračunati njihove vrijednosti. Brzine za prvi i drugi harmonički oscilator nalazimo:

$$\boldsymbol{v}_{1\,1} = \boldsymbol{v}_{1\,\frac{1}{2}} + \left(-\alpha_1 x_{1\,1} - \alpha_{12}(x_{1\,1} - x_{2\,1}) \right) \times \frac{\Delta t}{2m_1}$$

$$\boldsymbol{v}_{2\,1} = \boldsymbol{v}_{2\,\frac{1}{2}} + \left(-\alpha_1 x_{2\,1} + \alpha_{12}(x_{1\,1} - x_{2\,1}) \right) \times \frac{\Delta t}{2m_2} \quad .$$

Za drugi korak analogno nalazimo:

$$\boldsymbol{v}_{1\,1+\frac{1}{2}} = \boldsymbol{v}_{1\,1} + \left(-\alpha_1 x_{1\,1} - \alpha_{12}(x_{1\,1} - x_{2\,1}) \right) \times \frac{\Delta t}{2m_1}$$

$$x_{1\,2} = x_{1\,1} + \boldsymbol{v}_{1\,1+\frac{1}{2}} \Delta t$$

$$v_{2\ 1+\frac{1}{2}} = v_{2\ 1} + (- \alpha_2 x_{2\ 1} + \alpha_{12}(x_{1\ 1} - x_{2\ 1})) \times \frac{\Delta t}{2m_2}$$

$$x_{2\ 2} = x_{2\ 1} + v_{2\ 1+\frac{1}{2}} \Delta t$$

$$v_{1\ 2} = v_{1\ 1+\frac{1}{2}} + (- \alpha_1 x_{1\ 2} - \alpha_{12}(x_{1\ 2} - x_{2\ 2})) \times \frac{\Delta t}{2m_1}$$

$$v_{2\ 2} = v_{2\ 1+\frac{1}{2}} + (- \alpha_2 x_{2\ 2} + \alpha_{12}(x_{1\ 2} - x_{2\ 2})) \times \frac{\Delta t}{2m_2} \quad .$$

2.1.2.6 Usporedba Eulerove metode i brzinskog Verlet algoritma

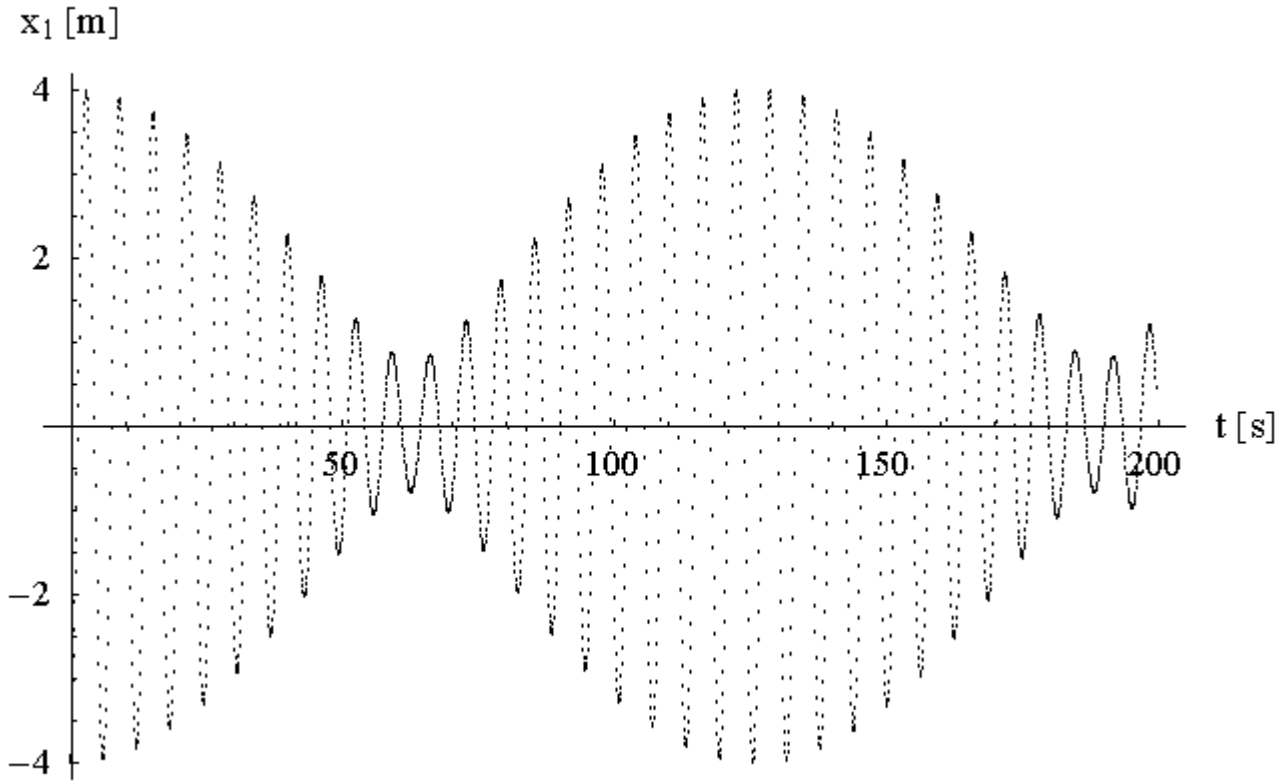
Usporedimo li Eulerovu metodu i brzinski Verlet algoritam možemo zaključiti sljedeće:

Vrijeme trajanja izračunavanja na računalu ovisi o kompleksnosti našeg fizikalnog problema i o vremenskom koraku *dt*. Štoviše, točnost našeg računa direktno ovisi o metodi koju koristimo i vremenskom koraku *dt*. Dok smo za Eulerovu metodu morali uzimati jako mali iznos vremenskog koraka da bi dobili što točnije rezultate, to za brzinski Verlet algoritam nije predstavljalo nikakav problem. Kad bismo za Eulerovu metodu uzeli vremenski korak kao i kod brzinskog Verlet algoritma, *dt = 0.01*, tada ne da bi naši računi bili sa velikom pogreškom, već bi bili potpuno netočni. Obje metode su jednostavne za primjenu, ali Eulerova metoda zasigurno ne bi bila povoljna kada bismo morali računati za sustav u kojem bismo imali N harmoničkih oscilatora. Jedan takav primjer je promatranje gibanja N čestica koje međudjeluju Lennard-Jonesovim potencijalom. Za takav sustav su pogreške Eulerove metode jednostavno nedopustive. Da bismo pokazali jednostavnost brzinskog Verlet algoritma kasnije ćemo promatrat sustav od N čestica koje međudjeluju Lennard-Jonesovim potencijalom. Brzinski Verlet algoritam je koristan alat u molekularnoj dinamici, ali samo kao pokazatelj ponašanja sustava, što ćemo kasnije i bogato opisati.

2.1.3 Grafovi dobiveni Eulerovom metodom i brzinskim Verlet algoritmom

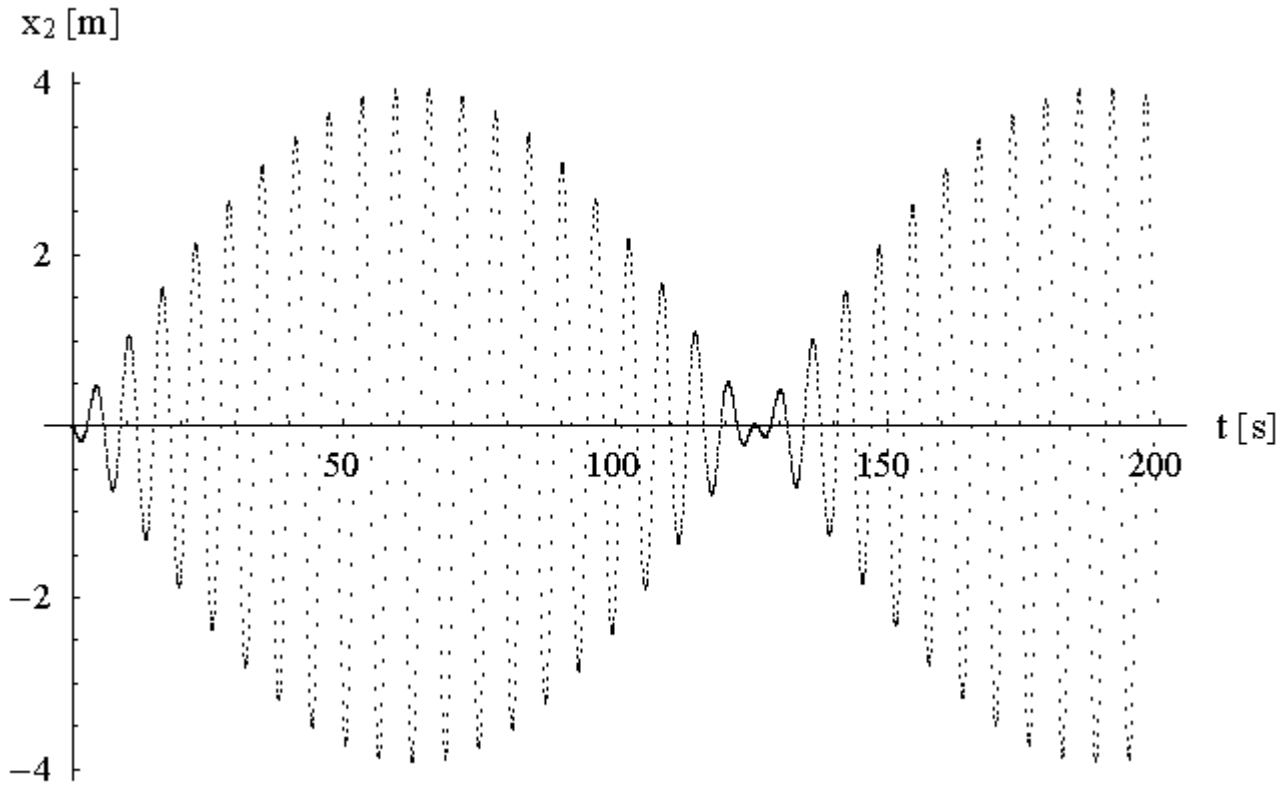
Dobiveni su podaci izvođenjem programa koji smo napisali u *C++* programskom jeziku i obrađeni u simboličkom jeziku *Mathematica*.

Kako bismo dobili osjećaj gibanja prvog tijela u vremenu pogledajmo graf (Slika 2.2).



Slika 2.2 Osciliranje prvog tijela

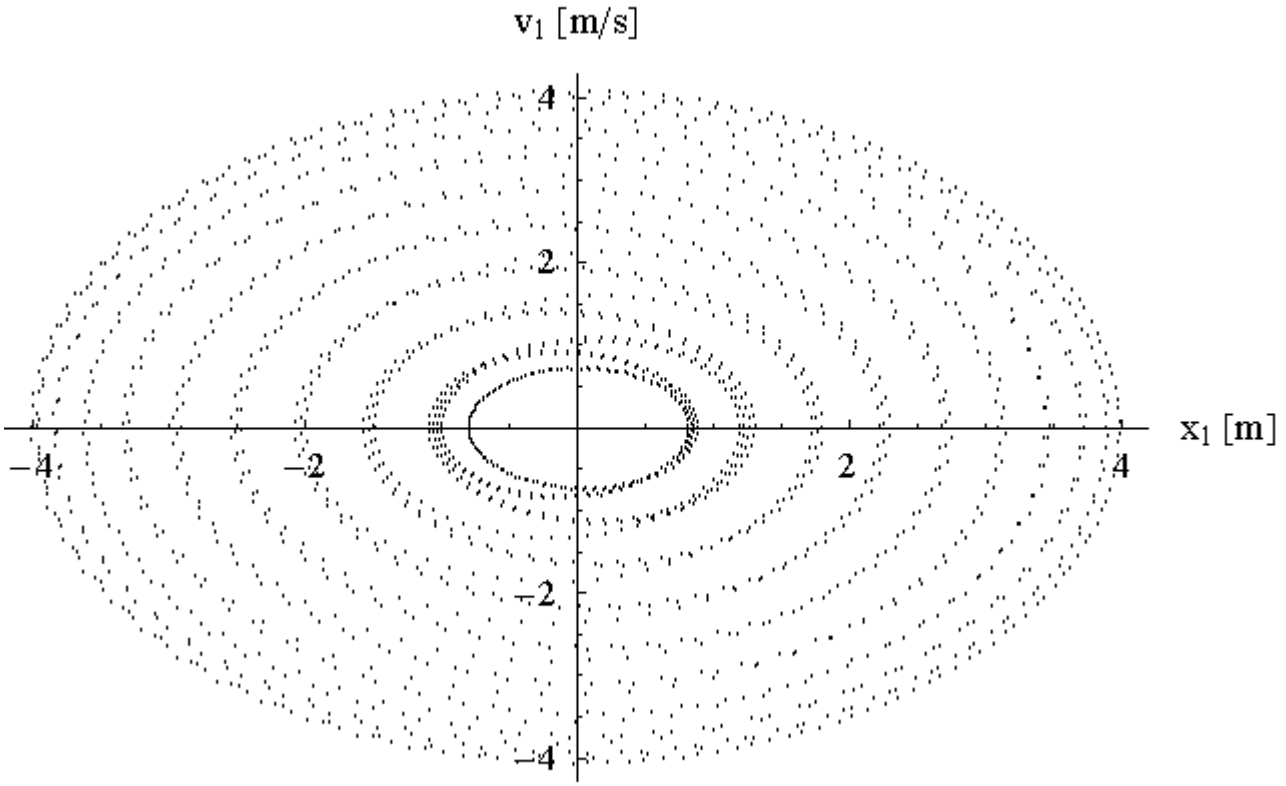
Iz priloženog grafa (Slika 2.2) možemo očitati maksimalnu amplitudu koja iznosi četiri metra, kao i pomak iz ravnotežnog položaja u $t = 0$. Promatrajući vremenski, vidimo da se maksimalna amplituda smanjuje do oko šesdesete sekunde od početka promatranja osciliranja. Nakon što se maksimalna amplituda smanjila lagano se povećava na četiri metra oko sto i tridesete sekunde od početka promatranja osciliranja. Periodički se ponavlja navedeni događaj. Pogledajmo što se događa sa gibanjem drugog tijela u vremenu na drugom grafu (Slika 2.3).



Slika 2.3 Osciliranje prvog tijela

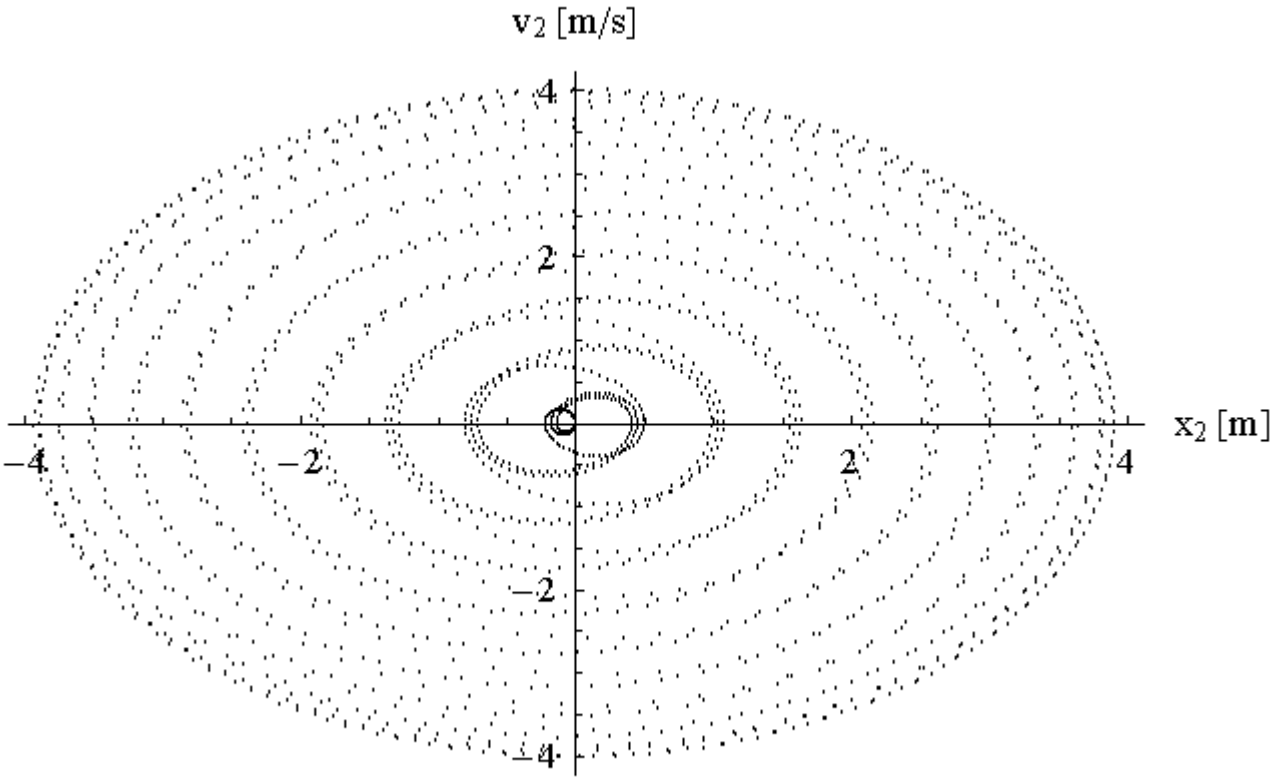
Kao i kod prvog tijela možemo očitati maksimalnu amplitudu od četiri metra. Vidimo da se u vremenu oko sto i tridesete sekunde od početka promatranja osciliranja tijelo vratilo u početni položaj koje je imalo na početku u $t = 0$.

Dakle, uzmimo sve ove dobivene rezultate i formirajmo zaključak o osciliranju ova dva tijela. Prvo tijelo je u $t = 0$ bilo pomaknuto iz ravnotežnog položaja u lijevu stranu za četiri metra, dok je drugo tijelo bilo u ravnotežnom položaju, tj. pomak je bio nula. Oba tijela su u $t = 0$ mirovala, tj. brzine su im bile nula. Kada prvo tijelo započne gibanje zbog povratne sile opruge, nalazi se u interakciji sa drugim tijelom preko opruge koja se nalazi između njih. Mehanička energija se sa prvog tijela postupno prenosi na drugo tijelo sve dok ono ne dostigne maksimalnu amplitudu četiri metra. Nakon što drugo tijelo dosegne svoju maksimalnu amplitudu vraća se u početni položaj koje je imalo u $t = 0$. Tako se ovaj događaj periodički ponavlja. Ovo dalje možemo potkrijepiti i prikazom promjene brzine u odnosu na položaj tijela.



Slika 2.4 Odnos brzine i pomaka za prvo tijelo

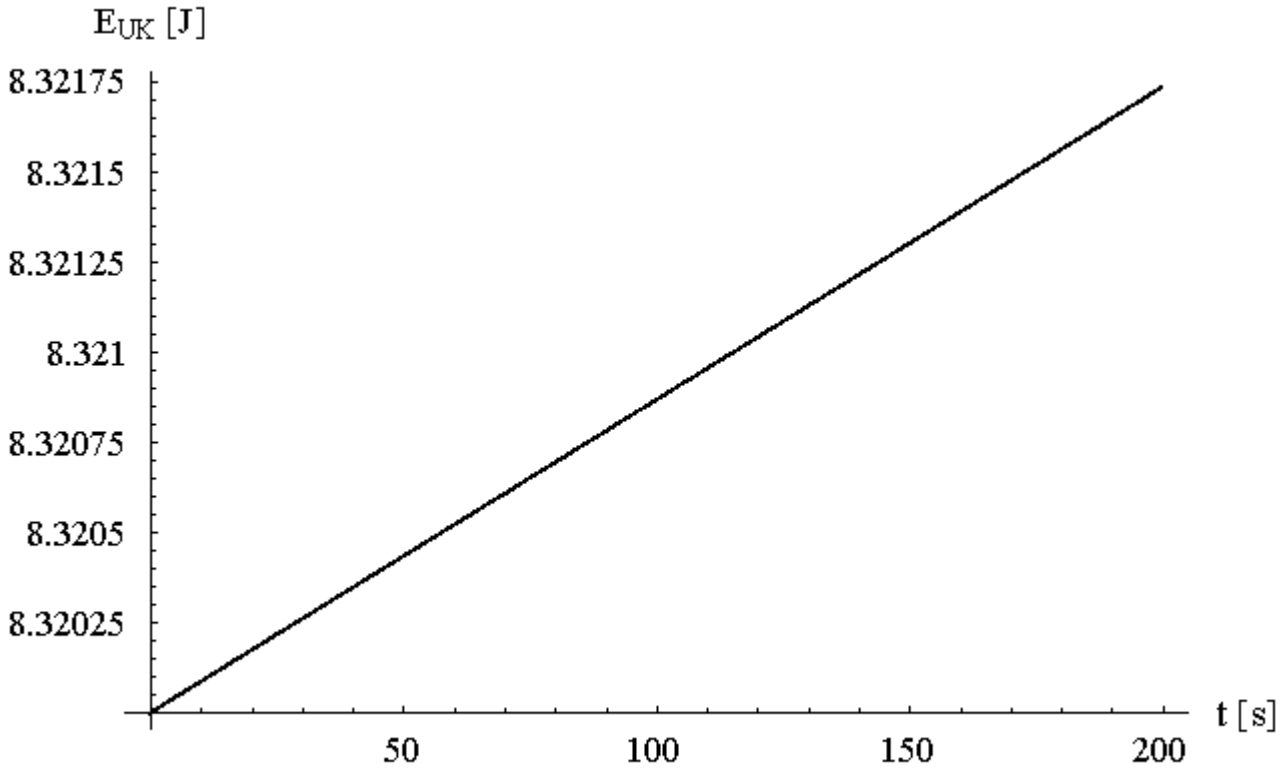
Pošto znamo za prvo tijelo (Slika 2.4) da je početna brzina bila nula, a pomak iz ravnoteže minus četiri metra, možemo na grafu pratiti kako se mijenja brzina u odnosu na položaj tijela. Dakle, od početka u $t = 0$ brzina tijela raste na nešto manje od četiri metra po sekundi dok je pomak iz ravnoteže jednak nula, tj. dok prvo tijelo prolazi kroz ravnotežni položaj ima maksimalnu brzinu. Maksimalna brzina se u određenim vremenskim periodima smanjuje sve do oko jedan i pol metara u sekundi, pa se oped periodički povećava. Pogledajmo sada za drugo tijelo graf (Slika 2.5).



Slika 2.5 Odnos brzine i pomaka za drugo tijelo

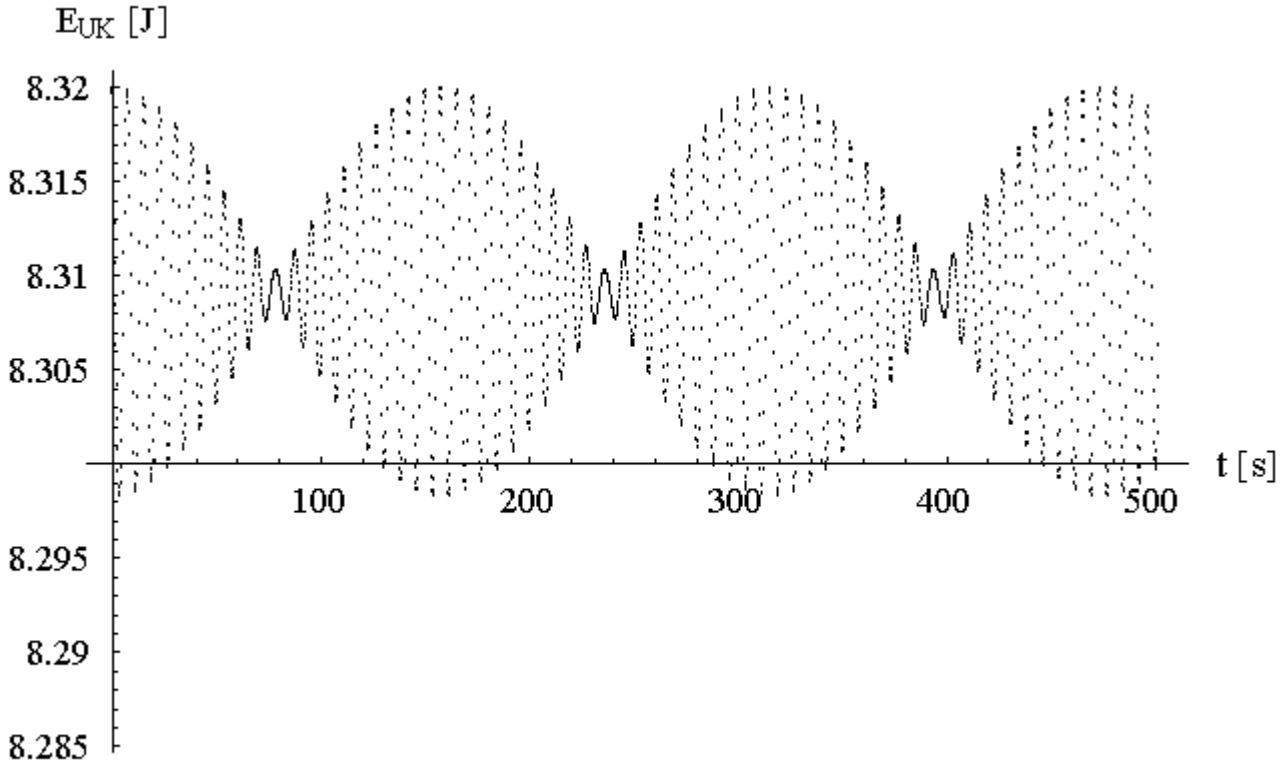
Za drugo tijelo (Slika 2.5) vidimo da se od početka promatranja maksimalna brzina periodički povećava do četiri metra u sekundi kada i maksimalna amplituda se periodički poveća na četiri metra. Nakon što dosegne maksimum, maksimalna brzina i amplituda padaju periodički sve do nule. Slike grafova, koje smo naveli u gornjim razmatranjima, su jednake za Eulerovu metodu i brzinski Verlet algoritam. Titranje koje smo promatrali za prvo tijelo do oko šezdesete sekunde je prigušeno titranje, gdje se smanjuje maksimalna amplituda i brzina, dok je u tom vremenskom periodu za drugo tijelo prisilno titranje. U periodu od šezdesete sekunde do oko sto i tridesete prvo tijelo prisilno titra, dok drugo tijelo prigušeno titra.

Govorili smo o tome kako je naš sustav konzervativan, tj. kako ukupna energija sustava treba biti očuvana. Pogledajmo za Eulerovu metodu graf (Slika 2.6) koji prikazuje iznos ukupne energije sustava u vremenu.



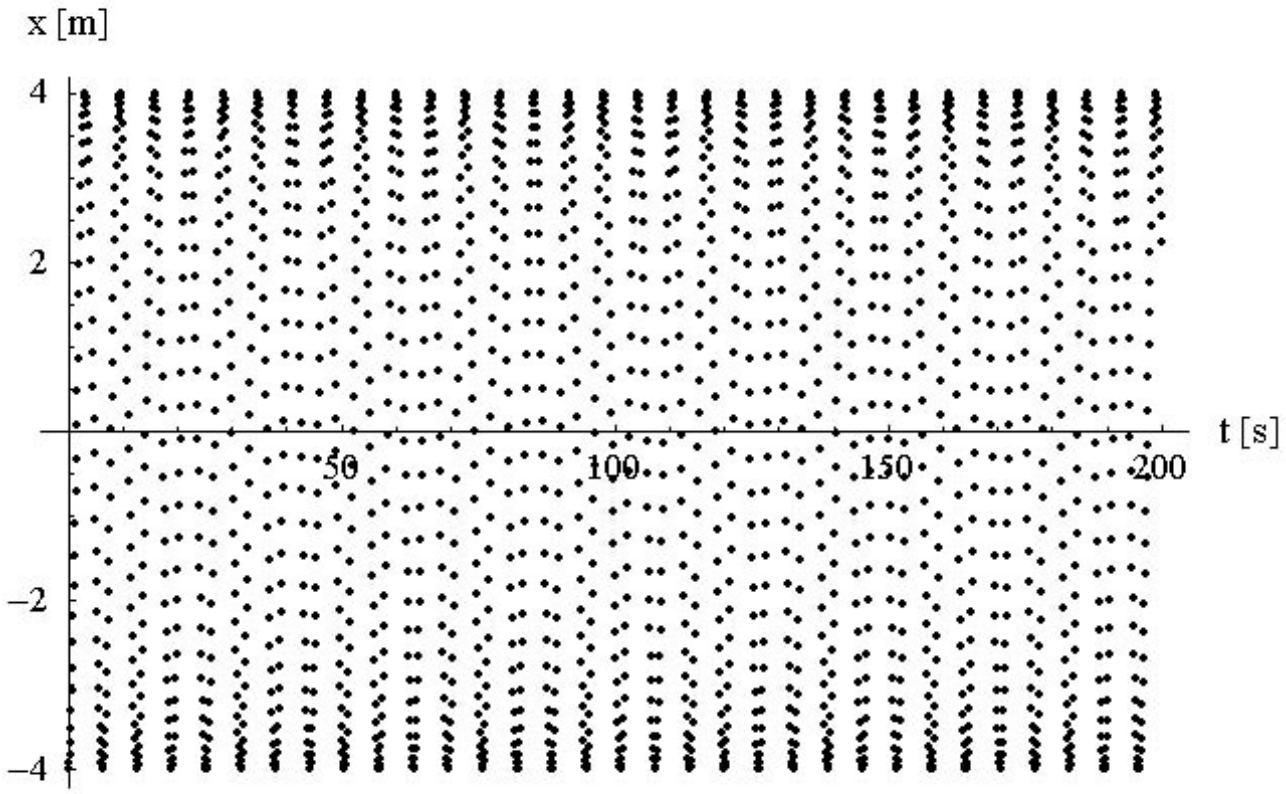
Slika 2.6 Ukupna energija u vremenu – Eulerova metoda

Iz grafa (Slika 2.6) vidi se da je ukupna energija sustava očuvana. Ono što je zanimljivo je to da smo morali uzeti jako malen vremenski korak (dt = 0.000001) da bismo dobili ovakvu točnost. Možemo primjetiti da se pogreška pri računanju ukupne energije linearno povećava. Za brzinski Verlet algoritam imamo graf (Slika 2.7) koji je potpuno drugačiji.



Slika 2.6 Ukupna energija u vremenu – brzinski Verlet algoritam

Za razliku od ukupne energije sustava dobivene Eulerovom metodom gdje se pogreška pri računanju linearno povećava, kod brzinskog Verlet algoritma pogreška oscilira oko točnog iznosa ukupne energije sustava na početku. Za brzinski Verlet algoritam nam je bio dovoljan vremenski korak veličine dt = 0.1. Osim bržeg izvođenja brzinskog Verlet algoritma, vidimo da i nakon dužeg vremenskog perioda promatranja sustava ukupna energija neće puno povećati zbog pogreške, jer je pogreška mala. Što se događa sa sustavom kada uzmemo za parametar $\alpha_{12} = 0$? Ako znamo da su oba tijela povezana oprugom konstante elastičnosti α_{12} , tada za $\alpha_{12} = 0$ možemo si zamisliti sustav u kojemu oba tijela se gibaju neovisno jedno o drugome. Sustav od dvaju tijela povezana sa oprugom će se podijeliti na dva sustava u kojima ćemo promatrati gibanje svakog od tih dvaju tijela zasebno. Za drugo tijelo je trivijalno rješenje, jer se u $t = 0$ nalazi u ravnotežnom položaju, a brzina u $t = 0$ mu je nula. Dakle, drugo tijelo će mirovati, neće biti zabilježen nikakav pomak iz ravnotežnog položaja od početka promatranja sustava, zato bi bilo suvišno prikazati to na grafu te ćemo ga izostaviti. Prvo tijelo će sada, kada više nije u interakciji sa drugim, periodički titrati oko ravnotežnog položaja. Maksimalna amplituda koju će imati prvo tijelo je četiri metra (Slika 2.7). Brzina u odnosu na pomak će se periodički ponavljati.



Slika 2.7 Osciliranje prvog tijela kada je $\alpha_{12} = 0$

3. Primjena numeričkog rješavanja sustava od N čestica koje međudjeluju Lennard-Jonesovim potencijalom

3.1 Uvod

U ovom poglavlju ćemo numeričku metodu, brzinski Verlet algoritam, iskoristi za rješavanje sustava N čestica koje međudjeluju Lennard-Jonesovim algoritmom.

Zamislimo plin, sustav od N čestica, gdje molekule međudjeluju Lennard-Jonesovim potencijalom. Kako bismo ograničili dimenzije promatranog sustava, zamislimo da se plin nalazi u kutiji stranice L. Uzeli smo samo dvije dimenzije kutije, da si olakšamo prikaz kretanja molekula u prostoru. Za duljinu stranice kutije smo uzeli četrdeset, znači L = 40.0 . Za promjer jedne molekule plina, uzeli smo jedan, znači $\sigma = 1.0$. Za energiju smo također uzeli jedan, znači $\epsilon = 1.0$. Ovi parametri određeni na ovaj način, bezdimenzionalno, se zovu reducirani parametri. Oblik Leenard-Jonesovog potencijala sa reduciranim parametrima ćemo razmatrati kasnije. Što kada molekula dođe do ruba kutije? Taj problem je riješen tako što zamislimo da se naša kutija ponavlja beskonačno u prostoru. Kada molekula, npr. izađe na lijevu stranu kutije, ona će biti transportirana na početak desne strane kutije. Ovim rješenjem smo zadržali molekulu u našem sustavu, ona je zadržala svoj smjer kretanja, a promijenila je samo položaj unutar kutije. Analogno učinimo kada molekula ima kontakt sa donjom i gornjom stranom kutije. Razmještaj molekula u kutiji može predstavljati problem, jer ne smijemo ih staviti preblizu, tj. udaljenost između dvije molekule ne smije biti manji od jedan, koliko je promjer molekule, tako riješimo problem kolizije na samom početku. Razmještaj ćemo obaviti pomoću funkcije *random* u C++ programskom jeziku. Za početni iznos brzine mmolekule uzmemo jedan, a komponente generiramo *random* tako da je iznos brzine molekule jedan. Kako bismo uštedjeli na vremenu pri računanju na računalu, udaljenost na kojoj Lennard-Jonesov potencijal djeluje smanjimo na pet. Dakle, računamo potencijal i silu samo za molekule za čiju udaljenost vrijedi $r \in \langle 0.8,\ 5.0 \rangle$. Time smo dozvolili da se događaju kolizije, ali samo do udaljenosti od r = 0.8, jer bi manja udaljenost mogla biti kobna za naše rezultate, što ćemo kasnije vidjeti kod razmatranja Lennard-Jonesovog potencijala. Za sustav ćemo bilježiti brzine molekula, putanje molekula, potencijalnu energiju, kinetičku energiju i ukupnu energiju konzervativnog sustava. Pošto koristimo reducirane parametre, možemo očekivati neprecizne rezultate. Glavni problem nam stvara kolizija molekula, što uvelike utječe na konačne rezultate energija.

3.2 Lennard-Jones potencijal

Promatranje međudjelovanja dviju molekula je matematički formulirano Lennard-Jonesovim potencijalom, opisan relacijom:

$$V(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{20}$$

U relaciji (20) parametri σ i ε su specifični Lennard-Jonesovi parametri, različiti za različite molekule u međudjelovanju. Vidimo da potencijal opisan relacijom (20) nam govori o tome što se događa kada se dvije molekule nađu na udaljenosti r. Kada su molekule jako blizu, tj. kada je iznos udaljenosti dviju molekula r je jako mala, tada dominira član $\frac{1}{r^{12}}$ u relaciji (20), tada je potencijal strogo pozitivan. Član $\frac{1}{r^{6}}$ u relaciji (20) dominira kada su molekule udaljene toliko daleko da su im elektronski oblaci odvojeni i tada se potencijal negativan, tj. opisuje energiju vezanja dviju molekula. Iako Leenard-Jones potencijal uključuje kvantnu mehaniku, za vrijeme kolizije pojedini atom nema dovoljnu energiju da promjeni elektronsku konfiguraciju drugog atoma. Dakle, atome možemo promatrati kao čestice, kuglice, a gibanje molekula možemo opisati drugim Newtonovim zakonom gibanja. Mi ćemo promatrati sustav sa Lennard-Jonesovim potencijalom oblika:

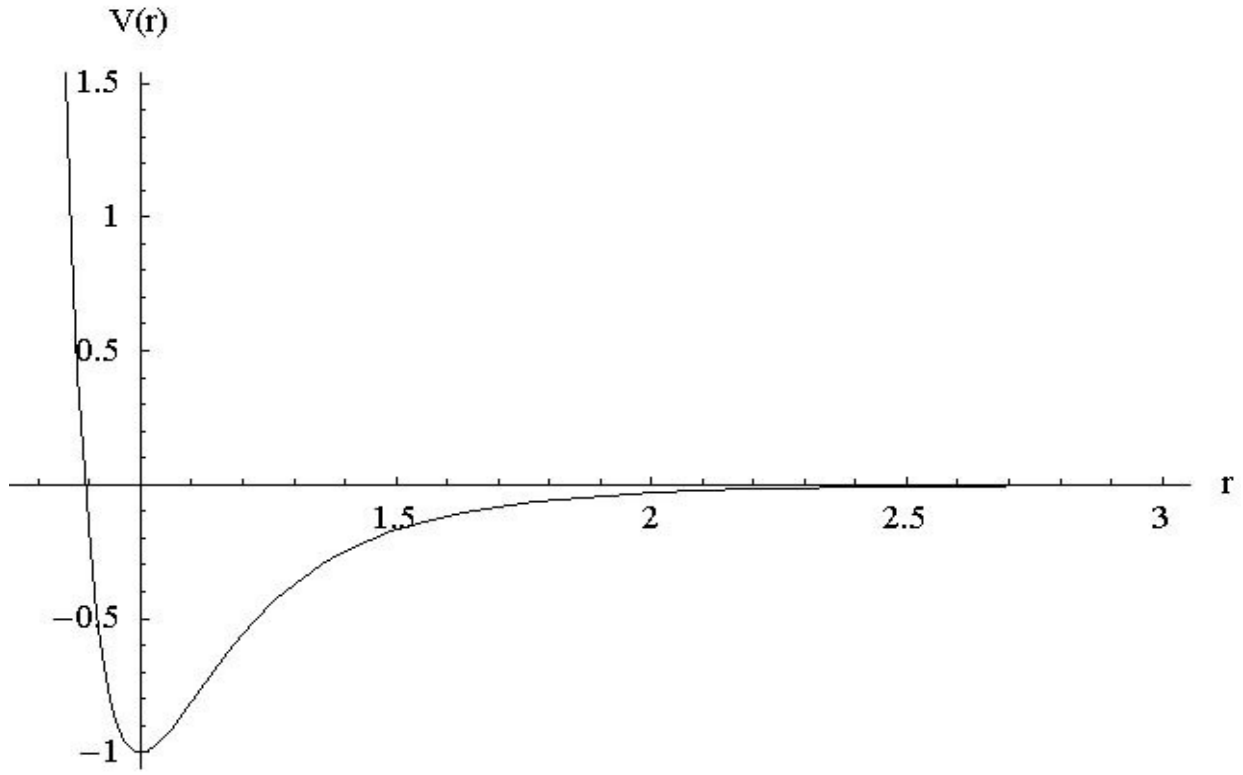
$$V(r) = \varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - 2\left(\frac{\sigma}{r}\right)^{6}\right] \tag{21}$$

Za naš oblik Lennard-Jonesovog potencijala imamo jednostavnu sliku (Slika 3.1) kojom ćemo dočarati ponašanje dviju molekula na udaljenosti r. Za naš sustav smo uzeli izbor parametara:

$$\varepsilon = 1$$
$$\sigma = 1.$$

Dakle, relaciju (21) možemo zapisati u obliku:

$$V(r) = \frac{1}{r^{12}} - 2\frac{1}{r^{6}}. \tag{22}$$



Slika 3.1 Lennard-Jones potencijal

Prema gornjoj slici (Slika 3.1) zaključujemo da na udaljenosti oko 2.5 počinje djelovati privlačna sila između dviju molekula. Kada molekule međusobno budu razmaknute na r = 1 tada je privlačna sila najveća. Počnu li se molekule još više približavati tada počinje prevladavati snažna odbojna energija. Dvije molekule ne mogu biti na istome mjestu u isto vrijeme. Naš je sustav konzervativan, tada sa lakoćom nalazimo silu kao parcijalnu derivaciju potencijala po pomaku, tj. pišemo kao prije:

$$\boldsymbol{F}(r) = -\frac{\partial V(r)}{\partial r}\hat{r}$$

te dobijemo:

$$\boldsymbol{F}(r) = \left(\frac{12}{r^{13}} - \frac{12}{r^{7}}\right)\hat{r}. \qquad \textbf{(23)}$$

Sada još samo da raspišemo jedinični vektor $\hat{r}$ u relaciji (23):

$$\boldsymbol{r} = r\,\hat{r}$$

$$\boldsymbol{r} = r_x\,\hat{x} + r_y\,\hat{y}$$

$$\hat{r} = \frac{\boldsymbol{r}}{r} = \frac{r_x\,\hat{x} + r_y\,\hat{y}}{r}$$

$$\hat{r} = \frac{r_x}{r}\,\hat{x} + \frac{r_y}{r}\,\hat{y}.$$

Ako primjenimo relaciju (23) na dvije molekule, recimo i-tu i j-tu molekulu tada imamo:

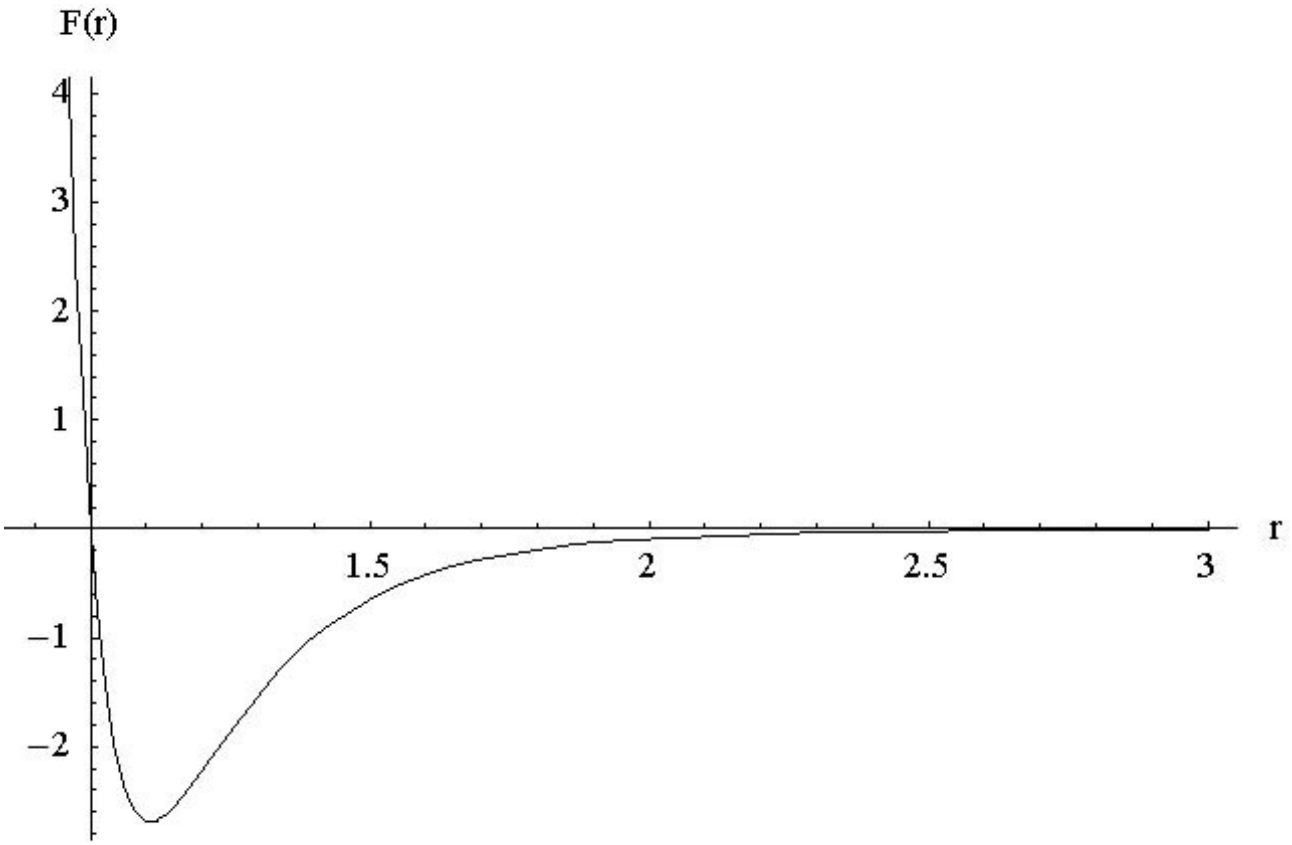
$$\boldsymbol{F_{ij}}(r) = \left(\frac{12}{r_{ij}^{13}} - \frac{12}{r_{ij}^{7}}\right)\left(\frac{r_{xj} - r_{xi}}{r_{ij}}\,\hat{x} + \frac{r_{yj} - r_{yi}}{r_{ij}}\,\hat{y}\right)$$

Iznos sile u odnosu na udaljenost dviju molekula prikazana je na slici (Slika 3.4). Promatrajući sliku (Slika 3.4) vidimo da na udaljenosti od 1.1 je privlačna sila najjača, a na udaljenosti manjoj od 1.1 prevladava odbojna sila koja ne dozvoljava da se molekule jako približe i da ne dođe do toga da se nađu na istome mjestu. Za molekule koje su udaljene između 4.0 i 5.0 potencijal tih parova molekula dodatno množimo sa funkcijom:

$$S(r) = \frac{(25-r^2)^2\ (25+2r^2-48)}{729}$$

Za udaljenosti između molekula koje su veće od 5.0 funkcija *S(r) = 0*. Što zapravo funkcija *S(r*) radi? Odlučili smo da ne računamo potencijal na udaljenostima molekula za *r > 5.0*. Tako smo dobili na brzini računanja na računalu. Potencijal i sila na udaljenosti *r > 5.0* nisu jednaki nuli, već su premaloga iznosa da bi za i-tu molekulu računali potencijal i silu za sve j-te molekule koje međudjeluju sa i-tom molekulom. Uvrstimo li u funkciju S(r) za r = 4.0, tj. za S(r = 4.0) dobijemo S(r = 4.0) = 1.0. Uvrstimo li u funkciju S(r) za r = 5.0, tj. za S(r = 5.0) dobijemo S(r = 5.0) = 0. Na udaljenosti među molekulama između četiri i pet, znači za $r \in \langle 4.0,\ 5.0\rangle$ imamo izraz za silu između dvije molekule:

$$\boldsymbol{F_{ij}}(r) = \left(\frac{4(-14375 + 2000\,r_{ij}^2 - 82\,r_{ij}^4 + 14376\,r_{ij}^6 - 1600\,r_{ij}^8 + 41\,r_{ij}^{10})}{243\,r_{ij}^{13}}\right)\left(\frac{r_{xj} - r_{xi}}{r_{ij}}\,\hat{x} + \frac{r_{yj} - r_{yi}}{r_{ij}}\,\hat{y}\right) .$$



Slika 3.4 Sila između dvije molekule

U našem sustavu od N molekula ćemo gledati interakciju između i-te molekule i j susjednih molekula što se nalaze oko nje, te sa njima čini j parova interakcija. Dakle, ukupan potencijal i-te molekule sa j susjednih molekula je opisan relacijom:

$$V(r) = \sum_{ij} \varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - 2 \left(\frac{\sigma}{r_{ij}} \right)^{6} \right].$$

Ako pogledamo slike (Slika 3.4) i (Slika 3.1) vidimo da možemo uštedjeti na vremenu kod računanja, jer na odaljenosti većoj od tri se i potencijal i sila svode na nulu. Moramo paziti kod računanja da ne uzimamo u sumu parove molekula koje smo prethodno izračunali. Koristit ćemo brzinski Verlet algoritam, jer je najefikasniji.

3.3 Maxwellova raspodjela molekula prema brzinama

Promatramo plin sastavljen od jednakih molekula u stanju termičke ravnoteže na temperaturi T. Tražimo vjerojatnost da x-komponenta translacijske brzine molekule bude između v_x i $v_x + dv_x$, y-komponente između v_y i $v_y + dv_y$, a z-komponenta između v_z i $v_z + dv_z$. Uzimamo dvije pretpostavke:

1. Prema prvoj pretpostavci vjerojatnost da molekula ima brzinu između v_x i $v_x + dv_x$ ovisi samo o v_x. Analogne uvjete ćemo postaviti za y- i z-komponentu brzine. Prema toj pretpostavci vjerojatnost da projekcija brzine na neku koordinatnu os ima određenu vjerojatnost ne ovisi o projekcijama brzinena preostale dvije osi.

2. Druga pretpostavka zahtjeva da vjerojatnost ovisi samo o iznosu brzine, a ne i o njezinom predznaku. Brzine v_x i $-v_x$ moraju imati istu vjerojatnost pojavljivanja.

Funkcija raspodjele brzina u smjeru osi x:

$$f(v_x^2) = \frac{e^{-\frac{mv_x^2}{2kT}}}{\int_{-\infty}^{+\infty} e^{-\frac{mv_x^2}{2kT}} dv_x},$$

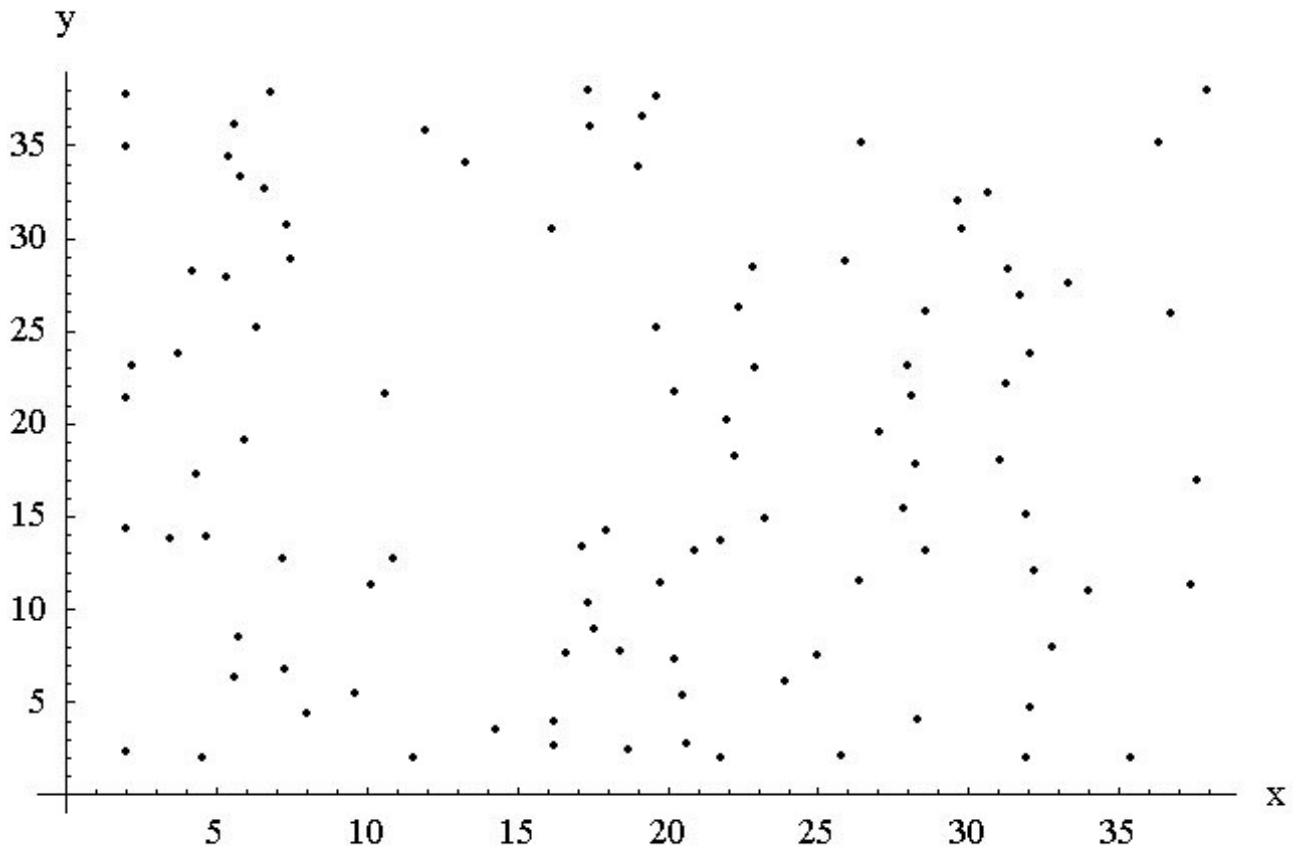
gdje je sa m određena masa molekule, T je temperatura sustava (izražena u Kelvinima), k je Boltzmanova konstanta i iznosi $k = 1.38\times10^{-23} J/K$. U nazivniku integriramo po v_x u granicama $-\infty$ do $+\infty$. Zamjenjujući v_x sa v_x i v_x dobivamo funkciju raspodjele za preostala dva smjera. To je Gaussova funkcija. Funkcija Maxwellove raspodjele:

$$P(v^2) = 4\pi\left(\frac{m}{2\pi kT}\right)^{\frac{3}{2}} v^2 e^{-\frac{mv^2}{2kT}},$$

koja određuje raspodjelu brzina molekula na teperaturi T. Sa v je označen iznos brzine molekula. U području malih brzina funkcija raspodjele raste s porastom brzine, postiže maksimum i na velikim brzinama opada prema nuli.

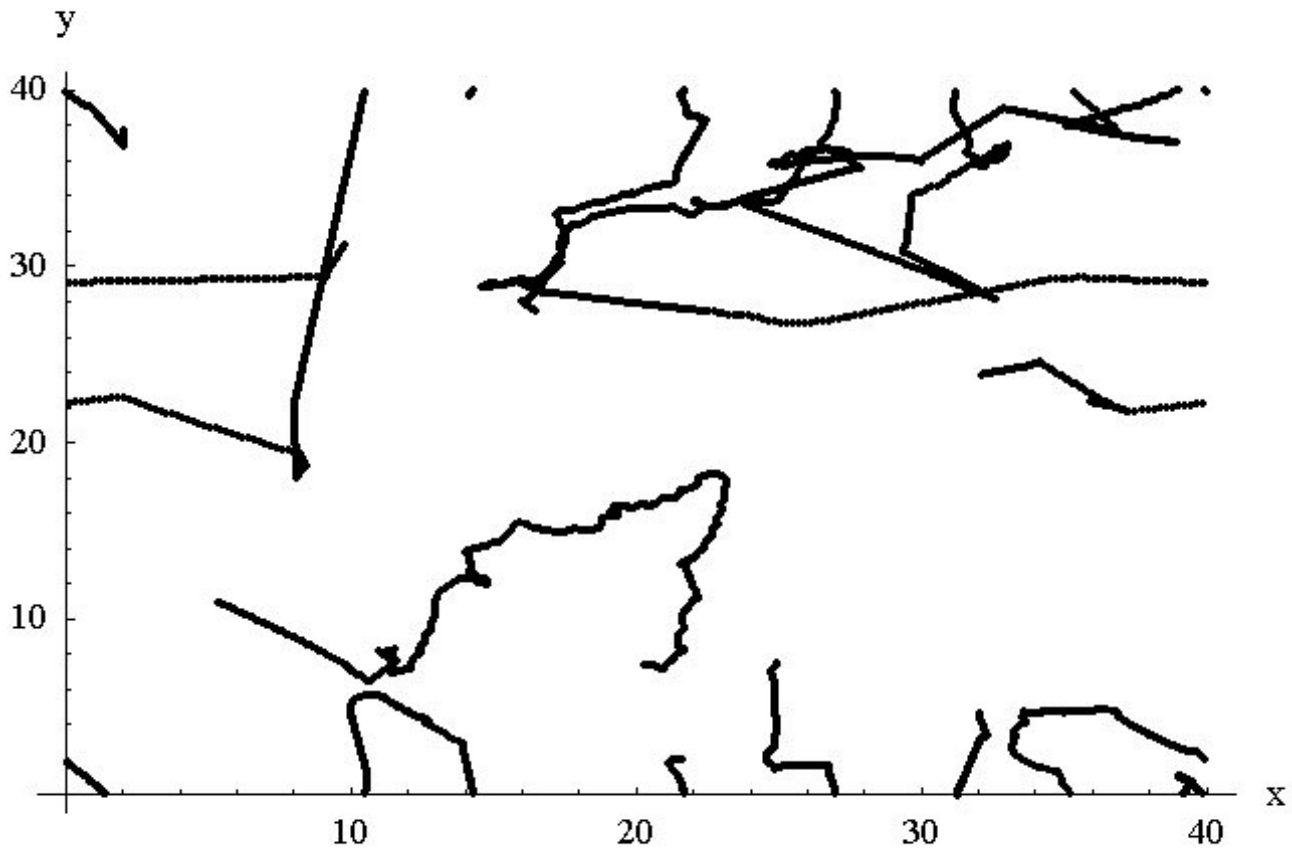
3.4 Rezultati

Za početak ćemo izvesti simulaciju za sto molekula u kutiji (Slika 3.5). Iznos brzine za svaku molekulu će biti jedan na početku simulacije, a komponenta u x i y smjeru moramo izgenerirati u skladu sa iznosom.



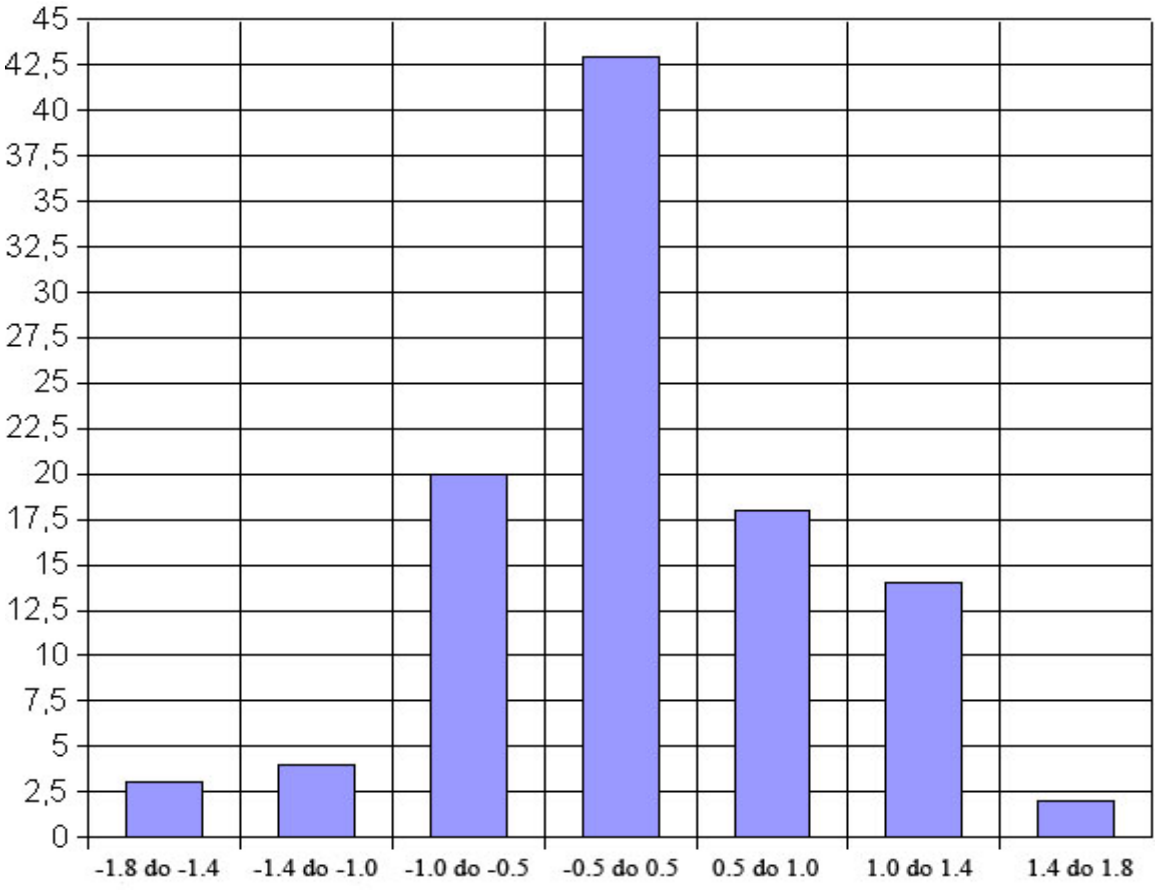
Slika 3.5 Razmještaj 100 molekula u kutiji

Nakon nekog vremenskog perioda uzeli smo putanje za šest molekula od sto, koliko ih ima u kutiji, pa njihovo gibanje izgleda ovako na slici (Slika 3.6).

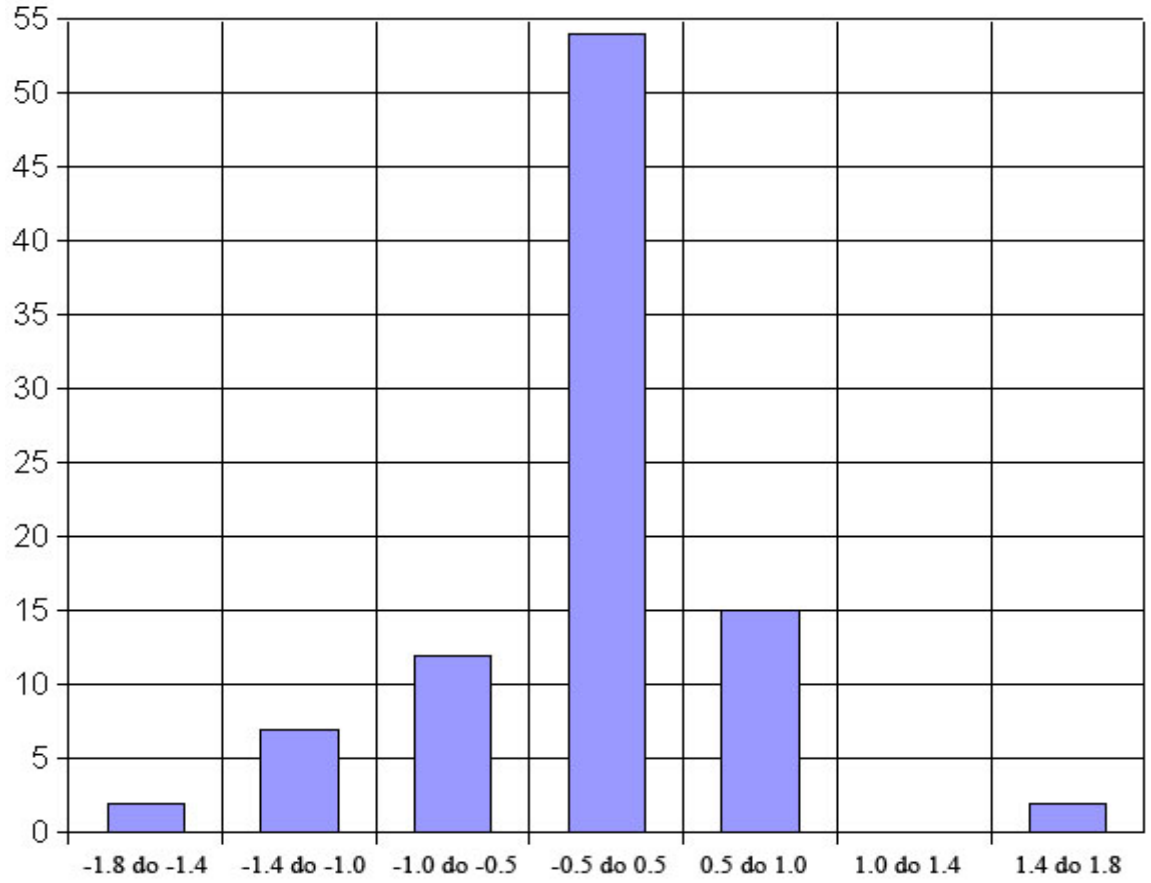


Slika 3.6 Putanje šest molekula

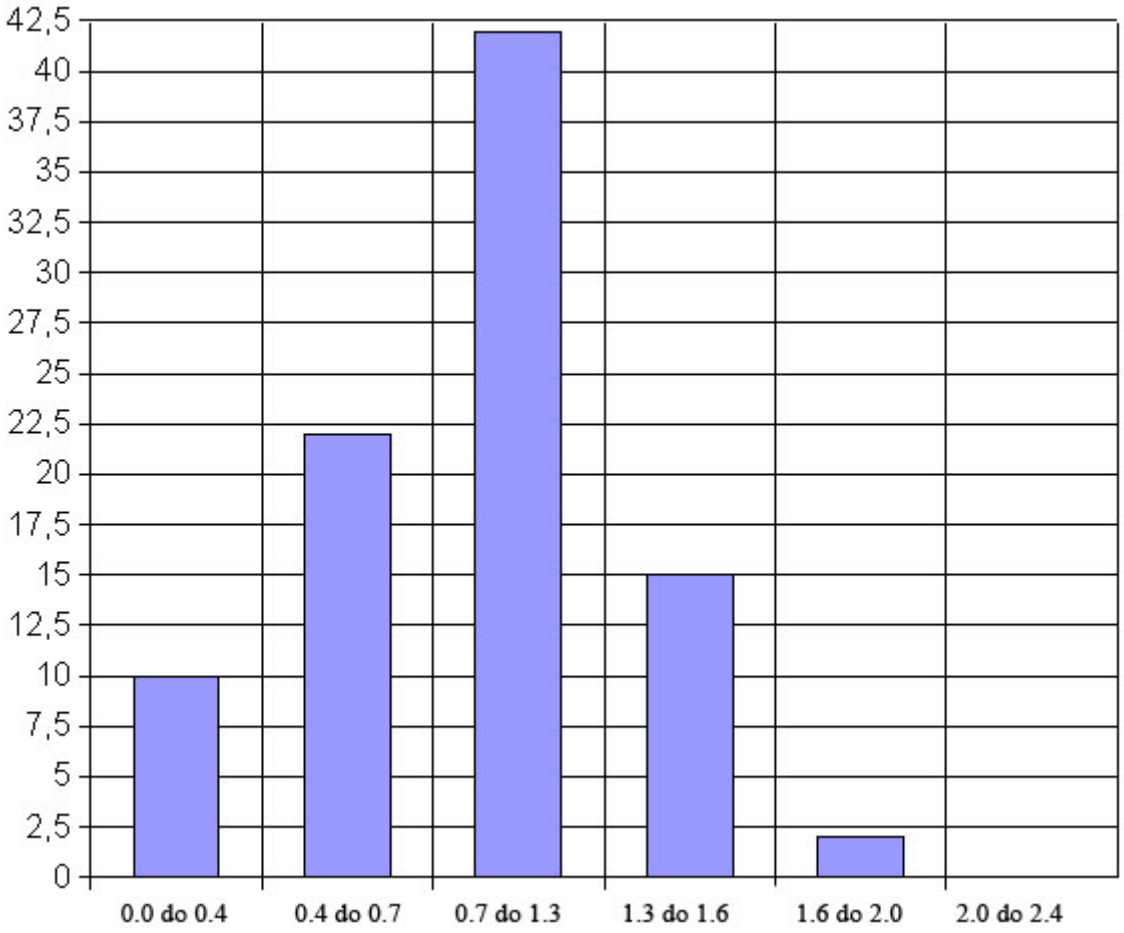
Pogledamo li sliku (Slika 3.6) vidimo kako putanja gibanja molekula nije pravilna, već određena gibanjem drugih molekula. Iste rezultate možemo promatrati i kod kutija sa dvjesto, tristo, četiristo, petsto i šesto molekula. Ono što nam je najbitnije, očuvanje ukupne energije, je ovaj put nešto što moramo uzeti zdravo za gotovo. Naime, radimo sa reduciranim parametrima, štoviše bezdimenzionalni su nam parametri pa su takvi i rezultati. Ove simulacije koristimo kao alat kojim možemo samo promatrati kako bi se sustav ponašao u sklopu molekularne dinamike. Nadalje, pogledamo li raspodjelu brzina za v_x na slici (Slika 3.7), za v_y na slici (Slika 3.8), za v_z na slici (Slika 3.9), vidjet ćemo da se nakon nekog vremena izvođenja simulacije raspodjela brzina molekula raspodjelila prema Maxwellovoj raspodjeli. Na početku simulacije, u $t = 0$, sve molekule su imale iznos brzine jedan, a po komponentama im je iznos bio slučajan tako da iznos brzine molekule bude jedan.



Slika 3.7 Raspodjela brzina po x-komponenti



Slika 3.8 Raspodjela brzina po y-komponenti



Slika 3.9 Raspodjela iznosa brzina

4. Nastava fizike: Jednostavno harmonijsko titranje

Škola: Gimnazija
Razred: 3.
Nastavni predmet: Fizika
Nastavna cjelina: Harmonijsko titranje
Nastavna jedinica: Titranje u mehanici – tijelo ovješeno na elastičnu oprugu
Trajanje: 1 školski sat
Mjesto održavanja nastave: Učionica
Cilj: Postići razumijevanje jednostavnog harmonijskog titranja
Zadaci: Usvojiti pojmove: ravnotežni položaj, amplituda, jednostavno harmonijsko titranje, elastična sila opruge, ukupna mehanička energija sustava jednostavnog harmonijskog titranja
Nastavna pomagala: Ploča, projektor, opruga, uteg i stalak za oprugu
Nastavne metode: Diskusija sa učenicima o pokusu prije samog pokusa. Pokus sa oprugom. Diskusija o pokusu sa oprugom.
Nastavni proces:

Nastavna jedinica "Tijelo ovješeno na elastičnu oprugu" je izvedena iz nastavne cjeline "Titranje u mehanici". Ovo je prvo sustretanje učenika sa harmonijskim titranjem, no očekujemo predznanje o zakonu očuvanja energije, kinetička energija, Newtonov prvi i drugi zakon gibanja.

Prije samog pokusa pokažemo rekvizite za pokus učenicima i složimo ih. Na stol stavimo stalak na koji ovjesimo oprugu. Zatim na oprugu ovjesimo tijelo.

Pitanje učenicima:

Što će se dogoditi sa oprugom kada tijelo ovješeno na oprugi potegnemo prema dolje?

Očekivani odgovor:

Opruga će se izdužiti.

Pitanje učenicima:

Što će se dogoditi kada prestanemo držati tijelo ovješeno na oprugi?

Očekivani odgovor:

Opruga će povući tijelo prema gore, pri tome će se opruga stisnuti.

Pitanje učenicima:

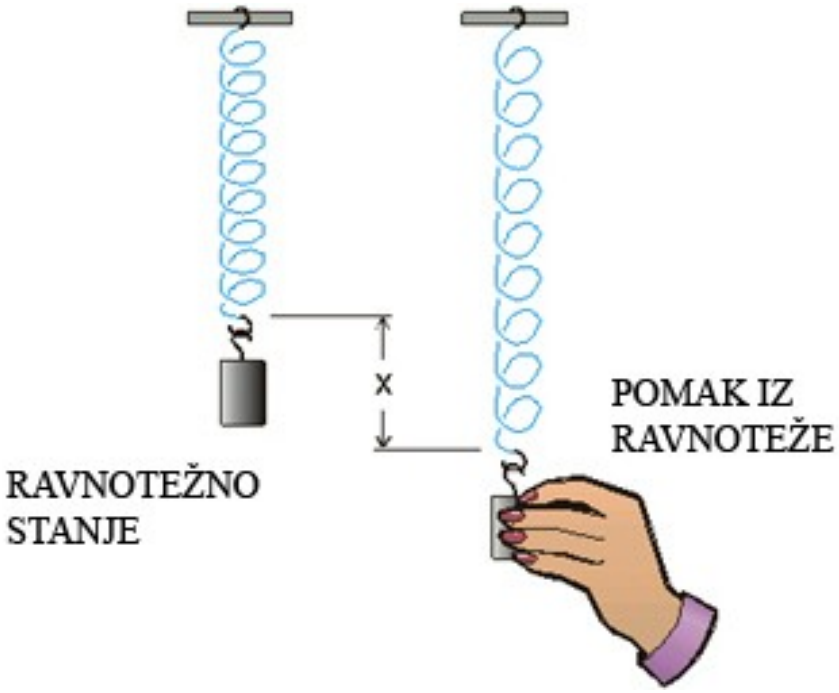
Što kada tijelo ode prema gore zajedno sa oprugom? Ostat će opruga stisnuta?

Očekivani odgovor:

Tijelo će početi padati prema dolje, zajedno sa njime i opruga, pri čemu će se opruga opet izdužiti.

Kako bismo najbolje provjerili ove pretpostavke, bilo bi poželjno pozvati jednog od učenika da izvede pokus i još jednom objasni što se događa sa tijelom i oprugom. Ako učenik ne uspije riješiti zadatak, tada mu pomoći, no prije toga upitati još nekoga od učenika da objasni pokus umjesto učenika koji nije znao. Na taj način možemo prodiskutirati pokus i uključiti učenike u nastavni proces.

Kada smo završili sa pokusom, pred učenicima uzimam rekvizite za pokus i objašnjavam im pojmove kao što su ravnotežni položaj i amplituda. Za uvođenje pojma elastične sile opruge postavimo sljedeće probleme:



Pitanje učenicima:

Zašto se u pokusu tijelo ovješeno o oprugu i izvučeno za x iz ravnotežnog položaja uopće giba kad ga ispustimo? Za koliko se produži opruga kada tijelo pomaknemo za x iz ravnotežnog položaja?

Očekivani odgovor:

Ako tijelo izvučemo za pomak x iz ravnotežnog položaja, opruga se produži također za pomak x.

Tijelo se giba jer na njega djeluje elastična sila opruge ($F = -kx$) u smjeru suprotnom od pomaka x, tj. prema ravnotežnom položaju.

Pitanje učenicima:

Je li gibanje titrajućeg tijela između dva krajnja položaja jednoliko, jednoliko akcelerirano (ubrzano ili usporeno), ili akcelerirano sa promjenjivom akceleracijom?

Očekivani odgovor:

Na krajnjim položajima je brzina tijela ovješenog na oprugu jednaka nuli. Između dva krajnja položaja je gibanje akcelerirano i to s promjenjivom akceleracijom.

Dopuna odgovora:

Na tijelo djeluje sila F = -kx koja nije konstantna, već za različite vrijednosti od x poprima različite vrijednosti F. Iz drugog Newtonowog zakona, F = ma, izlazi ma = -kx, pa otuda da i akceleracija a ovisi o pomaku x.

Pitanje učenicima:

Kad se tijelo vrati u ravnotežni položaj (x = 0), elastična sila opruge također je nula. Zašto se ipak tijelo nastavlja gibati i preko ravnotežnog položaja? Objasnite pomoću prvog i drugog Newtonovog zakona.

Očekivani odgovor:

Tijelo stigne u ravnotežni položaj određenom brzinom, pa zbog inercije (tromosti) nastavlja gibanje u istom smjeru.

Pitanje na koje ja dajem odgovor:

Je li ukupna mehanička energija u sustavu određena ako su zadani konstanta elastičnosti k opruge i amplituda A titranja?

Objašnjenje dano učenicima:

Prema zakonu očuvanja energije zbroj kinetičke energije tijela i elastične potencijalne energije opruge je konstanta, tj.

$$E = E_k + E_p$$
$$E = konst.$$

Proizlazi:

$$\frac{m v^2}{2} + \frac{k x^2}{2} = konst.$$

i to za svaki x. Ako u posebnom slučaju za x odabremo amplitudu, x = A, za ukupnu energiju dobivamo:

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

jer je u amplitudnom položaju brzina v tijela nula (v = 0), pa prvi član u prethodnoj jednadžbi iščezava. Uočavamo proporcionalnost $E \propto A^2$.

Pitanje učenicima:

U kojem je položaju, prilikom titranja, brzina tijela maksimalna? Pogledajmo jednadžbu:

$$\frac{m v^2}{2} + \frac{k x^2}{2} = \frac{k A^2}{2}$$

Očekivani odgovor:

Brzina će biti maksimalna kad je kinetička energija maksimalna, a to se ostvaruje u onom položaju tijela kad je potencijalna energija nula, tj. za x = 0. Dakle, tijelo ima maksimalnu brzinu kad prolazi kroz ravnotežni položaj.

Pitanje učenicima:

Možete li sada iz gornje jednadžbe izvesti izraz za maksimalnu brzinu?

Ostaviti malo vremena da učenici izvedu traženu veličinu te pozvati jednog od učenika da izvod napiše na ploču. Ukoliko je krivo ispraviti učenika ili pozvati još jednog učenika da napiše svoj točan izvod i objasni kako je došao do njega.

Očekivani odgovor:

$$\frac{m v_{maks}^2}{2} = \frac{k A^2}{2}$$

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

Zaključak:

Raspravom o problemima i izvodeći pokus i komentirajući ga učenici su usvojili pojmove kao što su: ravnotežni položaj, amolituda, jednostavno harmonijsko titranje, elastična sila opruge, ukupna mehanička energija sustava jednostavnog harmonijskog titranja, te znaju objasniti jednostavno harmonijsko titranje.

Korištene aplikacije

Pri izradi diplomskog rada koristio sam software:

1. Microsoft Visual Studio 6.0 (Visual C++)
2. Wolfram Mathematica 5.0
3. OpenOffice.org 2.0 BETA
4. Adobe Photoshop CS2

Svi gore navedeni programi su izvršavani pod operativnim sustavom Windows XP Proffesional 2002 SP2.

Zaključak

U diplomskom radu smo primjenili dvije numeričke metode rješavanja difrencijanih jednadžbi. Usporedili smo Eulerovu metodu i brzinski Verlet algoritam. Vidjeli smo da je Eulerova metoda dovoljna za rješavanje jednostavnih primjera iz fizike, no za molekularnu dinamiku nije ekonomična, tu je bolje primjeniti brzinski Verlet algoritam. Naš model molekularne dinamike je bio opisan kutijom u kojoj se nalaze čestice koje međudjeluju Lennard-Jonesovim potencijalom. Na početku smo svim molekulama odredili isti iznos brzine, da bi nakon nekog vremena raspodjela brzina molekula bila slična Maxwellovoj. Time smo dokazali da je naš model primjenjiv na sustav u molekularnoj dinamici.

Literatura

Fizika

1. Uvod u statističku fiziku, Prof. dr. Vladimir Šips, Školska knjiga, Zagreb, 1990.

Matematika

1. Numerička matematika, Ivan Ivanšić, Element, Zagreb, 1998.
2. Numerička analiza, grupa autora, PMF – matematički odjel, Zagreb, 2003.

Informatika

1. Demistificirani C++, Julijan Šribar i Boris Motik, Element, Zagreb, 2001.
2. The C++ programming language, Bjarne Stroustrup, AT&T, 1997
3. Computational Physics, N.J. Giordano, H. Nakanishi, 2006, Pearson Education

Prilog

C++ naredbe i funkcije:

#include <iostream>

Naredba kojom se od prevoditelja zahtijeva da u program uključi biblioteku "iostream". "#include" je pretprocesorska naredba kojom se prevoditelj pri prevođenju koda zaustavlja, skače u datoteku "iostream", prevede ju i potom se vraća na prevođenje koda iza naredbe "#include". "iostream" je primjer datoteke zaglavlja. U takvim datotekama se nalaze deklaracije funkcija sadržanih u odgovarajućim bibliotekama. "iostream" pripada kategoriji ulazno-izlaznih komponenti. Ulazni i izlazni tokovi osiguravaju vezu između programa i vanjskih jedinica na računalu: tipkovnice, zaslona, disketne jedinice, diska, CD-jedinice. Ovu biblioteku koristimo kako bi sa određenim naredbama spremili podatke na disk u određenu datoteku koju smo zadali da se kreira na disku.

#include <cmath>

Naredba kojom se od prevoditelja zahtijeva da u program uključi biblioteku "cmath". Ta biblioteka pripada kategoriji numeričkih komponenti. Ovu biblioteku koristimo, jer nam je potrebna za matematičku operaciju potenciranja koju koristimo u kodu za rješavanje zadatka.

using namespace std;

"using" i "namespace" su ključne riječi jezika C++ kojima se aktivira određeno područje imena (*imenik*, engl. *Namespace*), a "std" je naziv imenika u kojem su obuhvaćene sve standardne funkcije, uključujući i funkcije iz gore opisane biblioteke "iostream".

int main(){ . . . }

"main" je naziv za glavnu funkciju u svakom C++ programu. Izvođenje svakog programa počinje naredbama koje se nalaze u njoj. Pritom valja uočiti da je to samo simboličko ime koje daje do znanja prevoditelju koji se dio programa treba prvo početi izvoditi. Riječ "int" ispred oznake glavne funkcije ukazuje na to da će "main()" po svršetku izvođenja naredbi i funkcija sadržanih u njoj kao rezultat tog izvođenja vratiti cijeli broj ("int" dolazi od engleske riječi "integer" koja znači "cijeli broj"). Budući da se glavni program pokreće iz operacijskog sustava, rezultat glavnog programa se vraća operacijskom sustavu. Najčešće je to kod koji signalizira pogrešku nastalu tijekom izvođenja programa ili obavijest o uspješnom izvođenju.

int, double

U jeziku C++ ugrađena su u suštini dva osnovna tipa brojeva: cijeli brojevi (engl. integers) i realni brojevi (tzv. brojevi s pomičnom decimalnom točkom, engl. floating-point). Najjednostavniji tip brojeva su cijeli brojevi. Cjelobrojna varijabla deklarira se riječju int i njena će vrijednost u memoriji računala zauzeti dva bajta (engl. byte), tj. 16 bitova. Prvi bit je rezerviran za predznak, tako da preostaje 15 bitova za pohranu vrijednosti. Stoga se varijablom tipa int mogu obuhvatiti svi brojevi od najmanjeg broja (najvećeg negativnog broja) $-2^{15} = -32768$ do najvećeg broja

$2^{15} - 1 = 32767$. Brojevi tipa double pokriva opseg vrijednosti od 1.7×10^{-308} do 1.7×10^{308} . Varijabla tip double u memoriji računala će zauzeti 8 bajtova ili 64 bitova. Znači, deklaraciju provodimo pišući, na primjeru zadatka, "***double dt;"*** za realne brojeve i ***"int i;"*** za cijele brojeve.

FILE *objekt;

Definiramo objekt tipa FILE, na primjeru zadatka "***FILE *x1_time;"***.

FILE fopen(const char *imeDatoteke, const char *mod);

Na primjeru zadatka: ***"x1_time = fopen("x1_time.txt", "w");"***. Funkcija za rad sa datotekama, fopen, kao prvi argumente prima ime datoteke, iz primjera ta datoteka je **x1_time.txt**. Datoteka će biti stvorena u istom direktoriju gdje se nalazi i kompajlirani zadatak, ali ne kao *.cpp* datoteka, već kao *.exe* datoteka (izvršna datoteka). Drugi argument je mod u kojem se otvara datoteka, na primjeru zadatka je taj mod **w**, što znači da se otvara prazna datoteka za pisanja, ukoliko postoji navedena datoteka **x1_time.txt**, njen sadržaj se briše. Ime datoteke i oznaka moda dolaze pod navodnicima, jer su tipa **char**, a dolaze u znakovnom nizu. Funkcija **fopen** kao rezultat vraća pokazivač na objek tipa **FILE** u kojem će biti pohranjeni podaci o stanju toka. U slučaju neuspješnog otvaranja, funkcija će vratiti nul-pokazivač. Na primjeru zadatka to je pokazivač na objekt **x1_time**.

***** Pokazivač, na primjeru zadatk **FILE *x1_time;** je definiran objekt **x1_time** kao pokazivač na sadržaj tipa **FILE**.

Jednodimenzionalna polja

Najjednostavniji oblik polja su jednodimenzionalna polja kod kojih se članovi dohvaćaju preko samo jednog indeksa. Članovi polja složeni su linearnim slijedom, a indeks pojedinog člana odgovara njegovoj udaljenosti od početnog člana. Na primjeru zadatka ***double x1[2];*** deklarirali smo jednodimenzionalno polje x1 koje će sadržavati 2 realna broja tipa double. Pri odabiru imena za polje treba voditi računa da se ime polja ne smije podudarati s imenom neke varijable. Prvi član u polju ima indeks 0, a posljednji član indeks za 1 manji od duljine polja. Jedno od ograničenja pri korištenju polja jest da duljina polja mora biti specificirana i poznata u trenutku prevođenja koda. Duljina jednom deklariranog polja se ne može mijenjati tijekom izvođenja programa. Navedete li preveliki ili negativni indeks, prevoditelj neće javiti pogrešku i pristupit će memorijskoj adresi koja je izvan područja rezerviranog za polje.

while

Jedna od tri petlje kojima jezik C++ raspolaže. Ona se koristi uglavnom za ponavljanje segmenta koda kod kojeg broj ponavljanja nije unaprijed poznat. Sintaksa while bloka je

while (uvjet_izvođenja)

//blok naredbi

uvjet_izvođenja je izraz čiji je rezultat tipa **bool**, znači ili je rezultat *true* ili *false*. Na primjeru

zadatka blok naredbi će se izvoditi sve dok je uvjet ispunjen, a to je ***i < 200000000***.

if

Naredba *if* omogućava uvjetno grananje toka programa ovisno o tome da li je ili nije zadovoljen uvjet naveden iza ključne riječi *if*. Najjednostavniji oblik naredbe za uvjetno grananje je:

if (logički_izraz)

// blok naredbi

Ako je vrijednost izraza iza riječi *if* logička istina (*true*), izvodi se blok naredbi koje slijede iza izraza. U protivnom se taj blok preskače i izvođenje nastavlja od prve naredbe iza bloka.

int fprintf(FILE *tok, const char *format [, argument] ...);

Funkcija *fprintf()* omogućava formatirani upis podataka u datoteku vezanu uz tok. U slučaju uspješnog zapisa, funkcija vraća broj upisanih znakova, u slučaju pogreške pri upisu vraćaju negativan broj. Na primjeru zadatka imamo **fprintf(x1_time, "%f \t\t %f\n", t, x1[0]);** , gdje *x1_time* je tok na koji je vezana datoteka "x1_time.txt", a upisujemo varijable *t* i *x1[0]*.

int fclose(FILE *tok);

Funkcija *fclose()* omogućava brisanje iz memorije podataka vezanih uz tok, dok podaci u datoteci ostaju upisani. Na primjeru zadatka **fclose(x1_time);** se briše memorija vezana uz tok *x1_time*.

<u>Prvi primjer rješen Eulerovom metodom:</u>

```
#include <iostream>
#include <cmath>

using namespace std;

int main(){
        double dt;
        double x1[2];
        double x2[2];
        double a1, a2, a12;
        double v1[2];
        double v2[2];
        double uk;
        double t;
```

```
double m1, m2;

FILE *x1_time;
FILE *x2_time;
FILE *uk_time;
FILE *x1_v1;
FILE *x2_v2;

x1_time = fopen("x1_time.txt", "w");
x2_time = fopen("x2_time.txt", "w");
uk_time = fopen("uk_time.txt", "w");
x1_v1 = fopen("x1_v1.txt", "w");
x2_v2 = fopen("x2_v2.txt", "w");

dt = 0.000001;
t = 0.0;
x1[0] = -4.0;
x2[0] = 0.0;
m1 = 1.0;
m2 = 1.0;

a1 = 0.99;
a2 = 1.01;
a12 = 0.05;

v1[0] = 0.0;
v2[0] = 0.0;

int i;
i = 0;
while(i < 200000000){
        t = t + dt;

        x1[1] = x1[0] + v1[0] * dt;
        x2[1] = x2[0] + v2[0] * dt;
```

```
		v1[1] = v1[0] + dt * (-1 * a1 * x1[0] - a12 * (x1[0] - x2[0]))/m1;
		v2[1] = v2[0] + dt * (-1 * a2 * x2[0] + a12 * (x1[0] - x2[0]))/m2;

		x1[0] = x1[1];
		v1[0] = v1[1];

		x2[0] = x2[1];
		v2[0] = v2[1];

		uk = 0.5 * pow(v1[0],2) * m1 + 0.5 * pow(v2[0],2) * m2 + 0.5 * a1 * pow(x1[0],2) +
0.5 * a2 * pow(x2[0],2) + 0.5 * a12 * pow((x1[0] - x2[0]),2);

		if(i % 100000 == 0){
			fprintf(x1_time, "%f \t\t %f\n", t, x1[0]);
			fprintf(x2_time, "%f \t\t %f\n", t, x2[0]);
			fprintf(x1_v1, "%f \t\t %f\n", x1[0], v1[0]);
			fprintf(x2_v2, "%f \t\t %f\n", x2[0], v2[0]);
			fprintf(uk_time, "%f \t\t %f\n", t, uk);
		}
		i++;
	}

	fclose(x1_time);
	fclose(x2_time);
	fclose(x1_v1);
	fclose(x2_v2);
	fclose(uk_time);

	return 989;
}
```

Prvi primjer rješen brzinskim Verlet algoritmom:

```
#include <iostream>
```

```
#include <cmath>

using namespace std;

int main(){

	double m;
	double dt;
	double x1[2000];
	double x2[2000];
	double f1[2000];
	double f2[2000];
	double a1, a2, a3;
	double v1[2000];
	double v2[2000];
	double pkv1[2000];
	double pkv2[2000];
	double ke[2000];
	double pe[2000];
	double uk[2000];

	FILE *x1_time;
	FILE *x2_time;
	FILE *v1_x1;
	FILE *uk_time;


	x1_time = fopen("x1_time.txt", "w");
	x2_time = fopen("x2_time.txt", "w");
	v1_x1 = fopen("v1_x1.txt", "w");
	uk_time = fopen("uk_time.txt", "w");

	m = 1.0;
	dt = 0.1;
	x1[0] = -4.0;
```

```
x2[0] = 0.0;

// konstante sile
a1 = 0.99;
a2 = 1.01;
a3 = 0.05;

pe[0] = 0.5 * a1 * pow(x1[0], 2) + 0.5 * a2 * pow(x2[0], 2) + 0.5 * a3 * pow((x1[0] -
x2[0]),2);
f1[0] = -1.0 * x1[0] * a1 - (x1[0] - x2[0]) * a3;
f2[0] = -1.0 * x2[0] * a2 + (x1[0] - x2[0]) * a3;
v1[0] = 0.0;
v2[0] = 0.0;
uk[0] = pe[0];

int i, j;
for(i = 1; i <= 2000; i++){
        j = i;
        // računamo polukorak brzine u t + dt/2 za 1. HO i položaj 1. HO u t + dt
        pkv1[i] = v1[j-1] + (f1[j-1] * dt)/(2 * m);
        x1[i] = x1[j-1] + v1[j-1] * dt + (f1[j-1] * pow(dt,2))/(2 * m);

        // računamo polukorak brzine u t + dt/2 za 2. HO i položaj 2. HO u t + dt
        pkv2[i] = v2[j-1] + (f2[j-1] * dt)/(2 * m);
        x2[i] = x2[j-1] + v2[j-1] * dt + (f2[j-1] * pow(dt,2))/(2 * m);

        f1[i] = -1.0 * x1[i] * a1 - (x1[i] - x2[i]) * a3; // sila za 1. HO u t+ dt
        f2[i] = -1.0 * x2[i] * a2 + (x1[i] - x2[i]) * a3; // sila za 2. HO u t+ dt

        v1[i] = pkv1[i] + (f1[i] * dt)/(2 * m); //brzina za 1. HO u t + dt
        v2[i] = pkv2[i] + (f2[i] * dt)/(2 * m); //brzina za 2. HO u t + dt

        ke[i] = 0.5 * m * (pow(v2[i], 2) + pow(v1[i],2));

        pe[i] = 0.5 * a1 * pow(x1[i], 2) + 0.5 * a2 * pow(x2[i], 2) + 0.5 * a3 * pow((x1[i] -
x2[i]), 2); // potencijalna energija u t + dt
```

```
            uk[i] = ke[i] + pe[i];
        }
        float t;
        t = .0;
        for(i = 0; i <=2000; i++){

            fprintf(x1_time, "%f \t\t %f\n", t, x1[i]);
            fprintf(x2_time, "%f \t\t %f\n", t, x2[i]);
            fprintf(v1_x1, "%f \t %f\n", x1[i], v1[i]);
            fprintf(uk_time, "%f \t %f\n", t, uk[i]);
            t = t + dt;
        }

        fclose(x1_time);
        fclose(x2_time);
        fclose(v1_x1);
        fclose(uk_time);

        return 989;
}
```

Drugi primjer rješen brzinskim Verlet algoritmom:

```
#include <iostream>
#include <cmath>
#include <fstream>
#include <cstdlib>

using namespace std;

fstream kineticenergy("ekinetic.dat", ios_base::in | ios_base::out | ios_base::trunc);
fstream potentialenergy("epoten.dat", ios_base::in | ios_base::out | ios_base::trunc);
fstream totalenergy("etotal.dat", ios_base::in | ios_base::out | ios_base::trunc);
fstream moltrace("moltrace.dat", ios_base::in | ios_base::out | ios_base::trunc);
```

```
struct vector2d
{
	double coordxy[2][2][600];
} force, molecula, velocity, separation;

struct scalar
{
	double energy;
} potential, kinetic, total;

class lj_vv_md
{
	private:
		vector2d force, molecula, velocity;
		scalar potential, kinetic, total;
		int Np;
		double r;
		double dt;
		double t;
		double cm_x, cm_y;

	public:
		void parameters_initial(char choose)
		{
			fstream place2d;
			fstream veloc2d;

			if(choose == '1')
			{
				Np = 100;
				dt = 0.01;
				place2d.open("place100.dat", ios_base::in);
				veloc2d.open("veloc100.dat", ios_base::in);
			}
```

```
if(choose == '2')
{
        Np = 200;
        dt = 0.01;
        place2d.open("place200.dat", ios_base::in);
        veloc2d.open("veloc200.dat", ios_base::in);
}

if(choose == '3')
{
        Np = 300;
        dt = 0.01;
        place2d.open("place300.dat", ios_base::in);
        veloc2d.open("veloc300.dat", ios_base::in);
}

if(choose == '4')
{
        Np = 400;
        dt = 0.01;
        place2d.open("place400.dat", ios_base::in);
        veloc2d.open("veloc400.dat", ios_base::in);
}

if(choose == '5')
{
        Np = 500;
        dt = 0.01;
        place2d.open("place500.dat", ios_base::in);
        veloc2d.open("veloc500.dat", ios_base::in);
}

if(choose == '6')
{
```

```
        Np = 600;
        dt = 0.01;
        place2d.open("place600.dat", ios_base::in);
        veloc2d.open("veloc600.dat", ios_base::in);
}


place2d.seekg(0);
veloc2d.seekg(0);
for(int i = 0; i < 2; i++)
{
        for(int j = 0; j < Np; j++)
        {
                molecula.coordxy[0][i][j] = 0.0;
                velocity.coordxy[0][i][j] = 0.0;
                force.coordxy[0][i][j] = 0.0;
        }
}



for( i = 0; i < Np; i++)
{
        place2d >> molecula.coordxy[0][0][i] >> molecula.coordxy[0][1][i];
}

for(i = 0; i < Np; i++)
{
        veloc2d >> velocity.coordxy[0][0][i] >> velocity.coordxy[0][1][i];
}

cm_x = 0.0;
cm_y = 0.0;

for( i = 0; i < Np; i++)
```

```
{
        cm_x += velocity.coordxy[0][0][i];
        cm_y += velocity.coordxy[0][1][i];
}

for(i = 0; i < Np; i++)
{
        velocity.coordxy[0][0][i] -= cm_x / static_cast<double>(Np);
        velocity.coordxy[0][1][i] -= cm_y / static_cast<double>(Np);
}

potential.energy = 0.0;

for( i = 0; i < Np-1; i++)
{
        for(int j = i + 1; j < Np; j++)
        {
                for(int k = 0; k < 2; k++)
                {
                        separation.coordxy[0][k][1] = 0.0;
                        separation.coordxy[0][k][1] =
molecula.coordxy[0][k][i] - molecula.coordxy[0][k][j];

                        if(separation.coordxy[0][k][1] >= 20.0)
                                separation.coordxy[0][k][1] -= 40.0;
                        if(separation.coordxy[0][k][1] < -20.0)
                                separation.coordxy[0][k][1] += 40.0;
                }

                        r = sqrt(pow(separation.coordxy[0][0][1], 2.0) +
pow(separation.coordxy[0][1][1], 2.0));
                        if(r >= 0.8 &&  r <= 4.0)
                        {
                                potential.energy += (pow((1.0 / r), 12.0) - 2.0 *
pow((1.0 / r), 6.0));
```

```
                                                force.coordxy[0][0][i] += 12.0 * (pow((1.0 / r),
14.0) - pow((1.0 / r), 8.0)) * separation.coordxy[0][0][1];
                                                force.coordxy[0][1][i] += 12.0 * (pow((1.0 / r),
14.0) - pow((1.0 / r), 8.0)) * separation.coordxy[0][1][1];
                                                force.coordxy[0][0][j] -= 12.0 * (pow((1.0 / r),
14.0) - pow((1.0 / r), 8.0)) * separation.coordxy[0][0][1];
                                                force.coordxy[0][1][j] -= 12.0 * (pow((1.0 / r),
14.0) - pow((1.0 / r), 8.0)) * separation.coordxy[0][1][1];
                                        }


                                        if(r > 4.0 && r < 5.0)
                                        {
                                                potential.energy += ((pow((1.0 / r), 12) - 2.0 *
pow((1.0 / r), 6)) * (pow((25.0 - pow(r, 2)), 2) * (25.0 + 2 * pow(r, 2) - 48.0)) / pow(9.0, 3));
                                                force.coordxy[0][0][i] += (( 4 * (-14375.0 +
2000.0 * pow(r,2) - 82.0 * pow(r,4) + 14376 * pow(r,6) - 1600.0 * pow(r,8) + 41.0 * pow(r,10))) /
(243.0 * pow(r , 13))) * (separation.coordxy[0][0][1] / r);
                                                force.coordxy[0][1][i] += (( 4 * (-14375.0 +
2000.0 * pow(r,2) - 82.0 * pow(r,4) + 14376 * pow(r,6) - 1600.0 * pow(r,8) + 41.0 * pow(r,10))) /
(243.0 * pow(r , 13))) * (separation.coordxy[0][1][1] / r);
                                                force.coordxy[0][0][j] -= (( 4 * (-14375.0 +
2000.0 * pow(r,2) - 82.0 * pow(r,4) + 14376 * pow(r,6) - 1600.0 * pow(r,8) + 41.0 * pow(r,10))) /
(243.0 * pow(r , 13))) * (separation.coordxy[0][0][1] / r);
                                                force.coordxy[0][1][j] -= (( 4 * (-14375.0 +
2000.0 * pow(r,2) - 82.0 * pow(r,4) + 14376 * pow(r,6) - 1600.0 * pow(r,8) + 41.0 * pow(r,10))) /
(243.0 * pow(r , 13))) * (separation.coordxy[0][1][1] / r);


                                        }
                                }
                        }


                        kinetic.energy = 0.0;
                        total.energy = 0.0;


                        for( i = 0; i < Np; i++)
                        {
                                kinetic.energy += 0.5 * (pow(velocity.coordxy[0][0][i], 2.0) +
pow(velocity.coordxy[0][1][i], 2.0));
                        }
```

```
total.energy = potential.energy + kinetic.energy;

cout.width(6);
cout.setf(ios_base::left, ios_base::adjustfield);
cout << "TIME";

cout.width(10);
cout.setf(ios_base::right, ios_base::adjustfield);
cout << "KINETIC E.";

cout.width(15);
cout.setf(ios_base::right, ios_base::adjustfield);
cout << "POTENTIAL";

cout.width(10);
cout.setf(ios_base::right, ios_base::adjustfield);
cout << "TOTAL E." << endl;

cout << "----------------------------------------" << endl;


cout.width(6);
cout.setf(ios_base::right, ios_base::adjustfield);
cout << "0";

cout.width(10);
cout.setf(ios_base::right, ios_base::adjustfield);
cout << kinetic.energy;

cout.width(15);
cout.setf(ios_base::right, ios_base::adjustfield);
cout << potential.energy;

cout.width(10);
```

```
        cout.setf(ios_base::right, ios_base::adjustfield);
        cout << total.energy << endl;
}

void remove_cm()
{
        cm_x = 0.0;
        cm_y = 0.0;

        for(int i = 0; i < Np; i++)
        {
                cm_x += velocity.coordxy[1][0][i] / static_cast<double>(Np);
                cm_y += velocity.coordxy[1][1][i] / static_cast<double>(Np);
        }

        for(i = 0; i < Np; i++)
        {
                velocity.coordxy[1][0][i] -= cm_x;
                velocity.coordxy[1][1][i] -= cm_y;
        }
}

void new_positions()
{
        for(int i = 0; i < 2; i++)
        {
                for(int j = 0; j < Np; j++)
                {
                        molecula.coordxy[1][i][j] = 0.0;
                }
        }

        for(i = 0; i < Np; i++)
        {
                molecula.coordxy[1][0][i] = molecula.coordxy[0][0][i] +
```

```
velocity.coordxy[0][0][i] * dt + 0.5 * force.coordxy[0][0][i] * pow(dt, 2.0);
                        molecula.coordxy[1][1][i] = molecula.coordxy[0][1][i] +
velocity.coordxy[0][1][i] * dt + 0.5 * force.coordxy[0][1][i] * pow(dt, 2.0);

                        if(molecula.coordxy[1][0][i] < 0.0)
                                molecula.coordxy[1][0][i] = molecula.coordxy[1][0][i] + 40.0;

                        if(molecula.coordxy[1][0][i] > 40.0)
                                molecula.coordxy[1][0][i] = molecula.coordxy[1][0][i] - 40.0;

                        if(molecula.coordxy[1][1][i] < 0.0)
                                molecula.coordxy[1][1][i] = molecula.coordxy[1][1][i] + 40.0;

                        if(molecula.coordxy[1][1][i] > 40.0)
                                molecula.coordxy[1][1][i] = molecula.coordxy[1][1][i] - 40.0;
                }
        }

        void new_force()
        {
                for(int i = 0; i < Np; i++)
                {
                        force.coordxy[1][0][i] = 0.0;
                        force.coordxy[1][1][i] = 0.0;
                }

                for( i = 0; i < Np-1; i++)
                {
                        for(int j = i + 1; j < Np; j++)
                        {
                                for(int k = 0; k < 2; k++)
                                {
                                        separation.coordxy[0][k][1] = 0.0;
                                        separation.coordxy[0][k][1] =
molecula.coordxy[1][k][i] - molecula.coordxy[1][k][j];
```

```
                        if(separation.coordxy[0][k][1] >= 20.0)
                                separation.coordxy[0][k][1] -= 40.0;
                        if(separation.coordxy[0][k][1] < -20.0 )
                                separation.coordxy[0][k][1] += 40.0;
                }

                        r = sqrt(pow(separation.coordxy[0][0][1], 2.0) +
pow(separation.coordxy[0][1][1], 2.0));

                        if(r >= 0.0 && r <=4)
                        {
                                force.coordxy[1][0][i] += 12.0 * (pow((1.0 / r),
14.0) - pow((1.0 / r), 8.0)) * separation.coordxy[0][0][1];
                                force.coordxy[1][1][i] += 12.0 * (pow((1.0 / r),
14.0) - pow((1.0 / r), 8.0)) * separation.coordxy[0][1][1];
                                force.coordxy[1][0][j] -= 12.0 * (pow((1.0 / r),
14.0) - pow((1.0 / r), 8.0)) * separation.coordxy[0][0][1];
                                force.coordxy[1][1][j] -= 12.0 * (pow((1.0 / r),
14.0) - pow((1.0 / r), 8.0)) * separation.coordxy[0][1][1];
                        }

                        if(r > 4.0 && r < 5.0)
                        {
                                force.coordxy[1][0][i] += (( 4 * (-14375.0 +
2000.0 * pow(r,2) - 82.0 * pow(r,4) + 14376 * pow(r,6) - 1600.0 * pow(r,8) + 41.0 * pow(r,10))) /
(243.0 * pow(r , 13))) * (separation.coordxy[0][0][1] / r);
                                force.coordxy[1][1][i] += (( 4 * (-14375.0 +
2000.0 * pow(r,2) - 82.0 * pow(r,4) + 14376 * pow(r,6) - 1600.0 * pow(r,8) + 41.0 * pow(r,10))) /
(243.0 * pow(r , 13))) * (separation.coordxy[0][1][1] / r);
                                force.coordxy[1][0][j] -= (( 4 * (-14375.0 +
2000.0 * pow(r,2) - 82.0 * pow(r,4) + 14376 * pow(r,6) - 1600.0 * pow(r,8) + 41.0 * pow(r,10))) /
(243.0 * pow(r , 13))) * (separation.coordxy[0][0][1] / r);
                                force.coordxy[1][1][j] -= (( 4 * (-14375.0 +
2000.0 * pow(r,2) - 82.0 * pow(r,4) + 14376 * pow(r,6) - 1600.0 * pow(r,8) + 41.0 * pow(r,10))) /
(243.0 * pow(r , 13))) * (separation.coordxy[0][1][1] / r);

                        }
                }
        }
```

```
}

void new_velocity()
{
	for(int i = 0; i < 2; i++)
	{
		for(int j = 0; j < Np; j++)
		{
			velocity.coordxy[1][i][j] = 0.0;
		}
	}

	for(i = 0; i < Np; i++)
	{
		velocity.coordxy[1][0][i] = velocity.coordxy[0][0][i] + 0.5 *
(force.coordxy[1][0][i] + force.coordxy[0][0][i]) * dt;
		velocity.coordxy[1][1][i] = velocity.coordxy[0][1][i] + 0.5 *
(force.coordxy[1][1][i] + force.coordxy[0][1][i]) * dt;
	}
}

void new_total_energy(int step)
{
	kinetic.energy = 0.0;
	potential.energy = 0.0;
	total.energy = 0.0;

	for(int i = 0; i < Np; i++)
	{
		kinetic.energy += 0.5 * (pow(velocity.coordxy[1][0][i], 2.0) +
pow(velocity.coordxy[1][1][i], 2.0));
	}

	for(i = 0; i < Np-1; i++)
	{
		for(int j = i + 1; j < Np; j++)
```

```
                {
                        for(int k = 0; k < 2; k++)
                        {
                                separation.coordxy[0][k][1] = 0.0;
                                separation.coordxy[0][k][1] =
molecula.coordxy[1][k][i] - molecula.coordxy[1][k][j];

                                if(abs(separation.coordxy[0][k][1]) > 20.0)
                                {
                                        if(separation.coordxy[0][k][1] >= 20.0)
                                                separation.coordxy[0][k][1] -= 40.0;
                                        if(separation.coordxy[0][k][1] < -20.0 )
                                                separation.coordxy[0][k][1] += 40.0;
                                }

                                r = sqrt(pow(separation.coordxy[0][0][1], 2.0) +
pow(separation.coordxy[0][1][1], 2.0));
                                if(r >= 0.8 && r <=4)
                                        potential.energy += (pow((1.0 / r), 12.0) - 2.0 *
pow((1.0 / r), 6.0));
                                if(r > 4.0 && r < 5.0)
                                        potential.energy += ((pow((1.0 / r), 12) - 2.0 *
pow((1.0 / r), 6)) * (pow((25.0 - pow(r, 2)), 2) * (25.0 + 2 * pow(r, 2) - 48.0)) / pow(9.0, 3));
                        }
                }
        }

        total.energy = potential.energy + kinetic.energy;
}

void rearrange_parameters(int step)
{
        for(int i = 0; i < Np; i++)
        {
                velocity.coordxy[0][0][i] = velocity.coordxy[1][0][i];
                velocity.coordxy[0][1][i] = velocity.coordxy[1][1][i];
```

```
                molecula.coordxy[0][0][i] = molecula.coordxy[1][0][i];
                molecula.coordxy[0][1][i] = molecula.coordxy[1][1][i];
                force.coordxy[0][0][i] = force.coordxy[1][0][i];
                force.coordxy[0][1][i] = force.coordxy[1][1][i];
        }
    }

    void put_in_file(double step)
    {
        kineticenergy << step * dt << "\t" << kinetic.energy << endl;
        potentialenergy << step * dt << "\t" << potential.energy << endl;
        totalenergy << step * dt << "\t" << total.energy << endl;

        cout.width(6);
        cout.setf(ios_base::right, ios_base::adjustfield);
        cout << step * dt;

        cout.width(10);
        cout.setf(ios_base::right, ios_base::adjustfield);
        cout << kinetic.energy;

        cout.width(15);
        cout.setf(ios_base::right, ios_base::adjustfield);
        cout << potential.energy;

        cout.width(10);
        cout.setf(ios_base::right, ios_base::adjustfield);
        cout << total.energy << endl;

        for(int i = 70; i < 76; i++)
            moltrace << molecula.coordxy[0][0][i] << "\t" <<
molecula.coordxy[0][1][i] << endl;
    }

    void put_into_bins()
```

```
{
	int bin[2][20];
	int bin2[6];

	for(int i = 0; i < 2; i++)
		for(int j = 0; j < 7; j++)
			bin[i][j] = 0;

	for(i = 0; i < 6; i++)
		bin2[i] = 0;

	for(i = 0; i < Np; i++)
	{
		if(velocity.coordxy[0][0][i] > -1.8 && velocity.coordxy[0][0][i] <=
-1.4)
			bin[0][0]++;
		if(velocity.coordxy[0][0][i] > -1.4 && velocity.coordxy[0][0][i] <=
-1.0)
			bin[0][1]++;
		if(velocity.coordxy[0][0][i] > -1.0 && velocity.coordxy[0][0][i] <=
-0.5)
			bin[0][2]++;
		if(velocity.coordxy[0][0][i] > -0.5 && velocity.coordxy[0][0][i] <=
0.5)
			bin[0][3]++;
		if(velocity.coordxy[0][0][i] > 0.5 && velocity.coordxy[0][0][i] <=
1.0)
			bin[0][4]++;
		if(velocity.coordxy[0][0][i] > 1.0 && velocity.coordxy[0][0][i] <=
1.4)
			bin[0][5]++;
		if(velocity.coordxy[0][0][i] > 1.4 && velocity.coordxy[0][0][i] <=
1.8)
			bin[0][6]++;

		if(velocity.coordxy[0][1][i] > -1.8 && velocity.coordxy[0][1][i] <=
-1.4)
			bin[1][0]++;
```

```
                    if(velocity.coordxy[0][1][i] > -1.4 && velocity.coordxy[0][1][i] <=
-1.0)
                        bin[1][1]++;
                    if(velocity.coordxy[0][1][i] > -1.0 && velocity.coordxy[0][1][i] <=
-0.5)
                        bin[1][2]++;
                    if(velocity.coordxy[0][1][i] > -0.5 && velocity.coordxy[0][1][i] <=
0.5)
                        bin[1][3]++;
                    if(velocity.coordxy[0][1][i] > 0.5 && velocity.coordxy[0][1][i] <=
1.0)
                        bin[1][4]++;
                    if(velocity.coordxy[0][1][i] > 1.0 && velocity.coordxy[0][1][i] <=
1.4)
                        bin[0][5]++;
                    if(velocity.coordxy[0][1][i] > 1.4 && velocity.coordxy[0][1][i] <=
1.8)
                        bin[1][6]++;



                }


                for(i = 0; i < Np; i++)
                {
                    if(sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) >= 0 &&
sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) < 0.4)
                        bin2[0]++;
                    if(sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) >= 0.4 &&
sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) < 0.7)
                        bin2[1]++;
                    if(sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) >= 0.7 &&
sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) < 1.3)
                        bin2[2]++;
                    if(sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
```

```
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) >= 1.3 &&
sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) < 1.6)
                                        bin2[3]++;
                                if(sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) >= 2.0 &&
sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) < 2.4)
                                        bin2[4]++;
                                if(sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) >= 2.4 &&
sqrt(velocity.coordxy[0][0][i]*velocity.coordxy[0][0][i] +
velocity.coordxy[0][1][i]*velocity.coordxy[0][1][i]) < 2.8)
                                        bin2[5]++;
                        }



                        for(i = 0; i < 7; i++)
                                cout << bin[0][i] << "\t" << bin[1][i] << endl;


                        for(i = 0; i < 6; i++)
                                cout << bin2[i] << " ";
                }
};



int main()
{
        lj_vv_md compute;


        char choose;
        int step;


a:      cout << "Odaberi koliko molekula ima u sustavu:\n";
        cout << "Biraj 1 za 100\nBiraj 2 za 200\nBiraj 3 za 300\nBiraj 4 za 400\nBiraj 5 za
500\nBiraj 6 za 600\nBiraj 7 za izlaz\n";
        cout << "Odabir: ";
        cin >> choose;
```

```
switch (choose)
{
	case '1' :
		compute.parameters_initial(choose);
		break;
	case '2' :
		compute.parameters_initial(choose);
		break;
	case '3' :
		compute.parameters_initial(choose);
		break;
	case '4' :
		compute.parameters_initial(choose);
		break;
	case '5' :
		compute.parameters_initial(choose);
		break;
	case '6' :
		compute.parameters_initial(choose);
		break;
	case '7' :
		goto b;
	default :
		cout << "Krivi odabir!\n\n";
		goto a;
}

for(step = 1; step <= 10000; step++)
{
	compute.new_positions();
	compute.new_force();
	compute.new_velocity();
	compute.new_total_energy(step);
	compute.rearrange_parameters(step);
```

```
            if(step%100 == 0)
            {
                compute.remove_cm();
            }
            if(step%10 == 0)
                compute.put_in_file(static_cast<double>(step));

        }
        compute.put_into_bins();

b:      return 9;
}
```