Kandidaatintutkielma
Fysikaalisten tieteiden kandiohjelma
Teoreettinen fysiikka

Liikeyhtälöiden numeerinen ratkaiseminen

Arttu Hyvönen

22.5.2020

Ohjaaja(t): Pauli Pihajoki

Tarkastaja(t): Pauli Pihajoki

Helsingin Yliopisto
Matemaattis-luonnontieteellinen tiedekunta

PL 64 (Gustaf Hällströmin katu 2a)
00014 Helsingin yliopisto

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Matemaattis-luonnontieteellinen tiedekunta	Fysikaalisten tieteiden kandiohjelma Teoreettinen fysiikka

Tekijä — Författare — Author

Arttu Hyvönen

Työn nimi — Arbetets titel — Title

Liikeyhtälöiden numeerinen ratkaiseminen

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Kandidaatintutkielma	22.5.2020	21

Tiivistelmä — Referat — Abstract

Tässä työssä tutkittiin kahta erilaista numeerista menetelmää, joilla ratkaistiin Hamiltonisen systeemin liikeyhtälöiden aikakehitys. Erityisesti haluttiin selvittää, miten menetelmän symplektisyys vaikuttaa saatuihin tuloksiin. Menetelmiksi valittiin ei-symplektinen Runge-Kutta menetelmä ja symplektinen loikkakeino. Ensin molemmat menetelmät johdettiin, minkä jälkeen niitä testattiin käytännössä. Testausta varten tehtiin Python-ohjelma, jolla laskettiin harmonisen oskillaattorin aikakehitys valittuja menetelmiä käyttäen. Saatuja tuloksia vertailtiin, minkä perusteella huomattiin, että lyhyellä ajanjaksolla Runge-Kutta menetelmä antoi tarkempia tuloksia, mutta kun aikaa kului enemmän symplektinen loikkakeino säilytti realistisemman kuvan systeemistä.

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1. Johdanto

Numeeriset menetelmät ovat työkaluja, joilla voidaan ratkaista ongelmia numeerisesti. Erilaisiin ongelmiin on lukematon määrä erilaisia menetelmiä. Tässä työssä keskitytään menetelmiin, jotka ratkaisevat differentiaaliyhtälöitä. Tarkemmin sanottuna vertaillaan eri tyyppisten menetelmien saamia vastauksia Hamiltonisen systeemin liikeyhtälöihin.

Ensin käydään läpi Hamiltonin mekaniikka ja sen antamat liikeyhtälöt, sekä esitellään systeemi, jota käytetään vertailussa. Erityisesti keskitytään Hamiltonisten systeemien symplektisyyteen ja siihen, mitä se tarkoittaa. Symplektisyys on näiden systeemien luontainen ominaisuus, minkä vuoksi sen säilyminen on haluttu ominaisuus myös numeerisille menetelmille (Ruth, 1983). Numeerisia menetelmiä, jotka säilyttävät systeemin symplektisyyden, kutsutaan niin ikään symplektisiksi.

Tämän työn tavoitteena onkin verrata kahta numeerista menetelmää, joista vain toinen on symplektinen, ja huomata niiden käyttäytymisessä eroja. Mielenkiinnon kohteena on etenkin symplektisen menetelmän ominaisuudet.

Vertailtaviksi menetelmiksi valittiin ei-symplektinen Runge-Kutta-menetelmä ja symplektinen loikkakeino. Runge-Kutta-menetelmät johdetaan yleisemmin omassa kappaleessaan, mutta vertailussa käytetään yleisintä menetelmää, jonka esitteli Kutta (1901). Myös loikkakeino johdetaan ja sen symplektisyys todistetaan.

Runge-Kutta-menetelmät sopivat käyttöön suuressa määrässä eri ongelmia, mutta vertailusta voidaan todeta, että symplektinen loikkakeino suorituu paremmin tietyillä osa-alueilla, kun ratkaistavana on Hamiltonisen systeemin liikeyhtälöt.

2. Liikeyhtälöt

Jos halutaan ratkaista fysikaalisen systeemin kehitys, ensin tarvitaan yhtälöt kuvaamaan tätä kehitystä. Monesti järkevin tapa systeemin liikeyhtälöiden selvittämiseen on käyttää Hamiltonin mekaniikkaa.

Hamiltonin mekaniikka voidaan johtaa Lagrangen mekaniikasta, joka puolestaan esittää Newtonin mekaniikan variaatiolaskennan avulla. Muista tavoista poiketen Hamiltonin mekaniikalla liikeyhtälöiksi saadaan ensimmäisen asteen differentiaaliyhtälöitä, mikä tekee numeerisesta laskennasta suoraviivaisempaa.

2.1 Lagrangen mekaniikka

Fysikaaliset systeemit useimmiten kehittyvät ajassa seuraten pienimmän vaikutuksen periaatetta[†]. Eli toisin sanoen systeemin, jolla on N vapausastetta, kehitys saadaan selvittämällä rata $\mathbf{q}(t)$, jolla funktionaali

$$S[\mathbf{q}(t)] = \int_{t_0}^{t_1} L(\mathbf{q}(t), \dot{\mathbf{q}}(t), t)\, dt \tag{2.1}$$

saavuttaa ääriarvonsa. Tässä Lagrangen funktio L on kineettisen ja potentiaalienergian erotus, $\mathbf{q}$ on yleistettyjen koordinaattien muodostama N-ulotteinen vektori ja $\dot{\mathbf{q}}$ vastaava nopeus. Koordinaattivektoreiden komponentteja tullaan merkitsemään alaindeksillä i.

Funktionaalin (2.1) ääriarvon tuottavalle radalle voidaan johtaa Eulerin-Lagrangen yhtälöt

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0, \tag{2.2}$$

jotka ovat N kappaletta toisen asteen differentiaaliyhtälöitä systeemin kehitykselle. (Arnold, 1989)

2.2 Hamiltonin mekaniikka

Lagrangen mekaniikan avulla saadaan N kappaletta toisen asteen differentiaaliyhtälöitä. Hamiltonin mekaniikalla saman systeemin kehitys kuvataan ensimmäisen asteen differen-

[†]Engl. principle of least action

tiaaliyhtälöillä, joita on $2N$ kappaletta.

Tekemällä Legendren muunnos Lagrangen funktiolle $L(\mathbf{q}, \dot{\mathbf{q}}, t)$ muuttujien $\dot{q}_i$ suhteen saadaan Hamiltonin funktio

$$H(q_i, p_i, t) = \sum_{i=1}^{N} p_i \dot{q}_i - L(q_i, \dot{q}_i, t), \tag{2.3}$$

josta voidaan eliminoida $\dot{q}_i$ yleistettyjen liikemäärien

$$p_i = \frac{\partial L}{\partial \dot{q}_i} \tag{2.4}$$

avulla.

Hamiltonin funktion kokonaisderivaatta voidaan muodostaa kahdella eri tavalla. Käyttämällä määritelmää (2.3) saadaan

$$dH = \sum_{i=1}^{N} \left(p_i d\dot{q}_i + \dot{q}_i dp_i - \frac{\partial L}{\partial q_i} dq_i - \frac{\partial L}{\partial \dot{q}_i} d\dot{q}_i - \frac{\partial L}{\partial t} dt \right), \tag{2.5}$$

joka voidaan muuttaa yhtälöiden (2.2) ja (2.4) avulla muotoon

$$dH = \sum_{i=1}^{N} (\dot{q}_i dp_i - \dot{p}_i dq_i) - \frac{\partial L}{\partial t} dt. \tag{2.6}$$

Toisaalta käyttämällä tietoa, että $H = H(q_i, p_i, t)$ saadaan

$$dH = \sum_{i=1}^{N} \left(\frac{\partial H}{\partial p_i} dp_i + \frac{\partial H}{\partial q_i} dq_i \right) + \frac{\partial H}{\partial t} dt. \tag{2.7}$$

Vertaamalla yhtälöitä (2.6) ja (2.7) päädytään Hamiltonin yhtälöihin

$$\dot{p}_i = -\frac{\partial H}{\partial q_i} \tag{2.8}$$

$$\dot{q}_i = \frac{\partial H}{\partial p_i}, \tag{2.9}$$

jotka ovat systeemin liikeyhtälöt. Nämä yhtälöt ovat ekvivalentteja yhtälöiden 2.2 kanssa. (Tuominen, 2017)

Hamiltonin mekaniikassa kuvataan siis systeemin tila vektoreiden $\mathbf{q}$ ja $\mathbf{p}$ avulla. Voidaan myös ajatella, että systeemin tila on piste paikka- ja liikemääräkoordinaattien määrittelemässä $2N$-ulotteisessa avaruudessa. Tätä avaruutta kutsutaan faasiavaruudeksi. (Nolte, 2015)

Lisäksi jos systeemin kineettinen energia on tavallista muotoa $T = \frac{1}{2} \sum_i m_i \dot{q}_i^2$ ja potentiaalienergia $V = V(\mathbf{q})$ saa Lagrangen funktio muodon

$$L = \frac{1}{2} \sum_i m_i \dot{q}_i^2 - V(\mathbf{q}). \tag{2.10}$$

Kun tämä sijoitetaan Hamiltonin funktion määritelmään (2.3) systeemin liikemäärän $p_i = m_i\dot{q}_i$ kanssa, saadaan funktioksi

$$H = \sum_i m_i\dot{q}_i^2 - \frac{1}{2}\sum_i m_i\dot{q}_i^2 + V(\mathbf{q}), \tag{2.11}$$

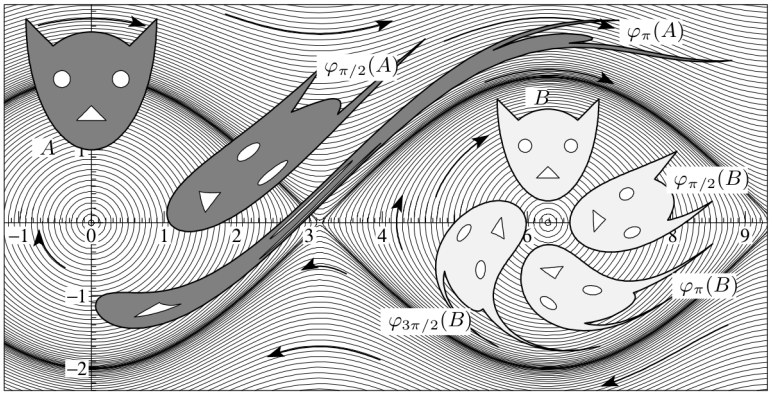
joka voidaan sieventää muotoon

$$H = T + V. \tag{2.12}$$

Eli näillä oletuksilla kineettisestä ja potentiaalienergiasta Hamiltonin funktio kuvaa systeemin kokonaisenergiaa. (Arnold, 1989)

2.3 Symplektisyys

Oletetaan jokin Hamiltonisen systeemin määrittelemä vektorikenttä ja valitaan faasiavaruudesta jokin 2-ulotteinen infinitesimaalinen pinta-ala. Kun annetaan tämän pinta-alan kehittyä ajassa vektorikentän mukaan, sen ala säilyy, mikä ilmenee kuvassa 2.1. Tätä ominaisuutta kutsutaan systeemin virtauksen symplektisyydeksi. (Hairer et al., 2006)



Kuva 2.1: Pinta-aloja A ja B kehitetään ajassa funktiolla φ_t, jota kutsutaan myös systeemin virtaukseksi. Pinta-alojen muoto vääristyy, mutta niiden ala säilyy. (Kuva kirjasta: *Geometric Numerical Integration* s. 185. (Hairer et al., 2006).)

Hamiltoniseen systeemin voidaan tehdä koordinattimuunnos. Jos se säilyttää systeemin symplektisyyden, kutsutaan myös muunnosta symplektiseksi. Jotta muunnos olisi symplektinen, sen pitää täyttää seuraavaksi johdettava ehto.

Ensin helpottaa, jos määritellään Hamiltonin yhtälöt (2.8) ja (2.9) uudelleen vektorin $\mathbf{x} = (q_1, \ldots, q_N, p_1, \ldots, p_N)$ ja $2N \times 2N$ matriisin

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \tag{2.13}$$

avulla. Matriisin komponentti I on $N \times N$ identiteettimatriisi. Näin Hamiltonin yhtälöille saadaan muoto

$$\dot{\mathbf{x}} = J\frac{\partial H}{\partial \mathbf{x}}. \tag{2.14}$$

Sitten voidaan tehdä koordinaattimuunnos $x \mapsto y(x)$, jonka Jacobin matriisi olkoon A. Nyt jos Jacobin matriisi A toteuttaa yhtälön

$$AJA^T = J \tag{2.15}$$

niin sanotaan, että muunnos on symplektinen. (Tuominen, 2017)

Symplektisyys ei ole rajoitettu koordinaattimuunnoksiin. Yleisemmin, jos muunnos $M\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ toteuttaa yhtälön $MJM^T = J$, kutsutaan sitä symplektiseksi. Numeeristen mentelmien kannalta on merkittävää, että muunnos aika-askeleesta seuraavan voi täytää symplektisyyden ehdot, jolloin menetelmä itsessään on symplektinen ja säilyttää systeemin ominaisuudet. (Hairer et al., 2006)

2.4 Harmoninen oskillaattori

Harmonisen oskillaattorin Lagrangen funktio on

$$L(q, \dot{q}) = \frac{1}{2}m\dot{q}^2 - \frac{1}{2}m\omega^2 q^2 \tag{2.16}$$

missä m ja ω ovat vakioita. Liikemäärä voidaan laskea yhtälön (2.4) avulla, jolloin saadaan $p = m\dot{q}$. Sitten liikemäärää käyttämällä voidaan kirjoitaa systeemin Hamiltonin funktio

$$H(q, p) = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 q^2. \tag{2.17}$$

Hamiltonin funktiosta pystytään nyt ratkaisemaan liikeyhtälöt

$$\dot{p} = -m\omega^2 q \tag{2.18}$$

$$\dot{q} = \frac{p}{m}, \tag{2.19}$$

joilla voidaan kuvata systeemin kehitys. Vaihtoehtoisesti Lagrangen funktiosta voidaan johtaa suoraan toisen asteen differentiaaliyhtälö

$$\ddot{q} = -\omega^2 q, \tag{2.20}$$

jonka ratkaisu on yleisesti tunnettu

$$q(t) = A\sin(\omega t + \phi), \tag{2.21}$$

missä A ja ϕ riippuvat systeemin alkuarvoista. Ratkaisu (2.21) on harmonisen oskillaattorin analyyttinen ratkaisu.

3. Runge-Kutta-menetelmät

Runge-Kutta-menetelmät ovat joukko numeerisia menetelmiä differentiaaliyhtälöille. Tässä kappaleessa johdetaan yleinen muoto Runge-Kutta-menetelmille ja lasketaan esimerkkinä neljännen kertaluokan menetelmän yksi askel. Esimerkin menetelmää käytetään myöhemmin vertailussa.

3.1 Johto

Menetelmän johdossa seuraillaan lähdettä Hairer et al. (2006). Runge-Kutta-menetelmillä voidaan ratkaista ensimmäisen asteen differentiaaliyhtälöitä, jotka ovat muotoa

$$\frac{dy(t)}{dt} = f(t, y), \tag{3.1}$$

missä $y(t)$ on funktio, joka yritetään ratkaista ja $f(t, y)$ on mielivaltainen tiedetty funktio. Lisäksi tarvitaan alkuarvo $y(t_0) = y_0$, jotta yhtälölle voidaan laskea yksikäsitteinen ratkaisu.

Integroimalla yhtälöä (3.1) puolittain yhden aika-askeleen h verran ja merkitsemällä $y_1 \equiv y(t_0 + h)$ saadaan

$$y_1 = y_0 + \int_{t_0}^{t_0+h} f(t, y(t))dt, \tag{3.2}$$

ja tästä voidaan approksimoida puolisuunnikassäännöllä

$$y_1 = y_0 + \frac{h}{2}\left[f(t_0,\, y_0) + f(t_0 + h,\, y_1)\right]. \tag{3.3}$$

Nyt huomataan ongelma. Arvo y_1, joka halutaan ratkaista, on yhtälössä molemmilla puolilla. Ongelmasta pästään eroon korvaamalla y_1 yhtälön oikealla puolella arviolla $y_1 \approx y_0 + hf(t_0, y_0)$. Sijoitetaan arvio yhtälöön (3.3), jolloin saadaan

$$y_1 = y_0 + \frac{h}{2}\left[f(t_0,\, y_0) + f(t_0 + h,\, y_0 + hf(t_0, y_0))\right]. \tag{3.4}$$

Nyt voidaan huomata, että $k_1 = f(t_0, y_0)$ ja $k_2 = f(t_0 + h, y_0 + hk_1)$ ovat kulmakertoimia ja kun kirjoitetaan (3.4) uudestaan niiden avulla saadaan

$$y_1 = y_0 + \frac{h}{2}k_1 + \frac{h}{2}k_2. \tag{3.5}$$

Eli seuraavan aika-askeleen arvo saadaan ottamalla edellisen askeleen arvo ja sen jälkeen seuraamalla ensimmäistä kulmakerrointa aika-askeleen puoleenväliin, minkä jälkeen seurataan toista kulmakerrointa askeleen loppuun. Tämän menetelmän ero pelkkään aika-askeleen puolittamiseen tulee siitä, että toisen kulmakertoimen laskemisessa on käytetty avuksi ensimmäistä kulmakerrointa.

Kahden kulmakertoimen sijasta menetelmä voidaan yleistää useammalle kulmakertoimelle, joilla kaikilla on oma painotuksensa. Eli seuraavan askeleen arvo n:llä kulmakertoimella saadaan summasta

$$y_1 = y_0 + h\sum_{i=1}^{n} b_i k_i, \tag{3.6}$$

missä b_i ovat tiedettyjä kertoimia. Kahden kulmakertoimen esimerkissä jälkimmäisen kulmakertoimen laskemisessa käytettiin apuna ensimmäistä. Nyt otetaan huomioon kaikki edeltävät kulmakertoimet, jolloin kulmakertoimet saadaan kaavalla

$$k_i = f(t_0 + hc_i\,,\; y_0 + h\sum_{j=1}^{i-1} a_{ij}\,k_j), \tag{3.7}$$

missä c_i ja a_{ij} ovat tiedettyjä kertoimia, jotka yhdessä kaavan (3.6) kertoimien b_i kanssa määrittelevät eri Runge-Kutta menetelmät.

Menetelmän kertoimia a_{ij}, b_i ja c_i ei voi valita mielivaltaisesti. Sen sijaan kertoimet voidaan johtaa menetelmälle, joka on haluttua kertaluokkaa (Butcher, 1963). Menetelmän kertaluokka määräytyy siitä, mitä aika-askeleen kertaluokkaa virhe on. Menetelmä on kertaluokkaa p jos sen virhe on luokkaa $\mathcal{O}(h^{p+1})$ (Hairer et al., 2006).

Näin määritellyt menetelmät ovat eksplisiittisiä. Menetelmä voi olla myös implisiittinen, jolloin kulmakertoimet voivat riippua kaikista aika-askeleen muista kulmakertoimista, eivät vain edeltävistä. Molemmassa tapauksessa kertoimet esitetään yleensä Butcher-taulukossa, jollainen on esitetty taulukossa 3.1. Eksplisiittisessä tapauksessa kertoimet a_{ij}, joissa $i \leq j$, ovat nollia, ja monesti jätetään siksi taulukosta pois. (Hairer et al., 2006)

Taulukko 3.1: Butcher-taulukko Runge-Kutta menetelmien kertoimille.

c_1	a_{11}	a_{12}	$\dots$	a_{1n}
c_2	a_{21}	a_{22}	$\dots$	a_{2n}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
c_n	a_{n1}	a_{n2}	$\dots$	a_{nn}
	b_1	b_2	$\dots$	b_n

3.2 Toteutus

Vertailussa käytetään neljännen kertaluokan Runge-Kutta-menetelmää, jonka esitteli Kutta (1901). Se on yleisin käytetty Runge-Kutta-menetelmä ja siitä käytetään lyhennettä RK4. Taulukossa 3.2 on menetelmän kertoimet (Hairer et al., 1993).

Taulukko 3.2: Butcher taulukko RK4-menetelmän kertoimille.

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	2/6	2/6	1/6

Systeemi, joka halutaan ratkaista, koostuu nyt yhtälöistä (2.18) ja (2.19). Eli yhtälöt eivät ole muotoa $\dot{y} = f(t, y)$, jolle menetelmä johdettiin. Sen sijaan systeemissä on kaksi yhtälöä

$$\dot{p} = f(q) \tag{3.8}$$

$$\dot{q} = g(p), \tag{3.9}$$

jotka riippuvat toisistaan. Menetelmää voidaan silti käyttää. Määritellään uusi muuttuja

$$\zeta = \begin{pmatrix} q \\ p \end{pmatrix}, \tag{3.10}$$

ja käytetään sitä yhdessä aiemmin laskettujen tuloksien (2.18) ja (2.19) kanssa. Näin voidaan esittää systeemin kehitys muodossa

$$\dot{\zeta} = \tilde{f}(t, \zeta) = \begin{pmatrix} p/m \\ -m\omega^2 q \end{pmatrix}. \tag{3.11}$$

Nyt yhtälö on oikeaa muotoa ja kulmakertoimet voidaan laskea käyttäen yhtälöä (3.7) ja alkuarvoa

$$\zeta_0 = \begin{pmatrix} q_0 \\ p_0 \end{pmatrix}. \tag{3.12}$$

RK4-menetelmän kertoimilla saadaan kulmakertomiksi

$$k_1 = \tilde{f}(t_0, \zeta_0) = \begin{pmatrix} p_0/m \\ -m\omega^2 q_0 \end{pmatrix} \tag{3.13}$$

$$k_2 = \tilde{f}(t_0 + \frac{h}{2},\ \zeta_0 + \frac{h}{2}k_1) \tag{3.14}$$

$$k_3 = \tilde{f}(t_0 + \frac{h}{2},\ \zeta_0 + \frac{h}{2}k_2) \tag{3.15}$$

$$k_4 = \tilde{f}(t_0 + h,\ \zeta_0 + hk_3). \tag{3.16}$$

Sitten voidaan laskea seuraavan askeleen arvo käyttäen saatuja kulmakertoimia. Tulos on

$$\zeta_1 = \zeta_0 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{3.17}$$

Vertailua varten tehtiin Python-ohjelma, joka käyttää yllä mainittua menetelmää. Ohjelman lähdekoodi on liitteessä B.

4. Loikkakeino

Loikkakeino[†] on Runge-Kutta-menetelmiin verrattuna käyttötarkoituksiltaan rajoitetumpi. Sille on kuitenkin löytynyt käyttöä fysiikan aloilta sen ominaisuuksien vuoksi.

Tässä kappaleessa menetelmä johdetaan ja yksi sen tärkeistä ominaisuuksista, symplektisyys, todistetaan. Lisäksi käydään läpi esimerkki, jonka mukaisesti menetelmä on toteutettu vertailussa.

4.1 Johto

Menetelmän johdossa seuraillaan lähdettä McLachlan and Quispel (2002). Tarkastellaan systeemiä, jonka Hamiltonin funktio on muotoa $H = \frac{1}{2}p^2 + V(q)$. Funktio voidaan jakaa kahteen osaan $H_T = \frac{1}{2}p^2$ ja $H_V = V(q)$, joille voidaan kirjoittaa Hamiltonin yhtälöt erikseen. Saadaan yhtälöt

$$H_T : \ \dot{q} = p \, , \quad \dot{p} = 0 \tag{4.1}$$

$$H_V : \ \dot{q} = 0 \, , \quad \dot{p} = -\frac{\partial V(q)}{\partial q}. \tag{4.2}$$

Hamiltonisen systeemin virtaus φ_t on kuvaus, joka edistää systeemiä ajan t verran lähtien annetuista alkuarvoista. Eli virtaus

$$\varphi_t(q_0, p_0) = (q(t, q_0, p_0), p(t, q_0, p_0)), \tag{4.3}$$

missä $q(t, q_0, p_0)$ ja $p(t, q_0, p_0)$ ovat ratkaisuja systeemin kehitykselle alkuarvoilla q_0 ja p_0. (Hairer et al., 2006)

Yhtälöistä (4.1) ja (4.2) voidaan ratkaista virtaukset H_T:n ja H_V:n kuvaamille systeemeille. Integroimalla yhtälöä $\dot{q} = p$ puolittain välin $t \in [0, t']$ yli ja käyttämällä hyväksi tietoa $p(t') = p(0)$ saadaan H_T:n virtaukseksi

$$q(t') = q(0) + t' \, p(0) \, , \quad p(t') = p(0). \tag{4.4}$$

Samalla tavalla saadaan laskettua H_V:n virtaus

$$q(t') = q(0) \, , \quad p(t') = p(0) - t' \, \frac{\partial V(q(0))}{\partial q}. \tag{4.5}$$

[†]Engl. The Leapfrog method

Sitten valitsemalla alkuarvot $(q(0), p(0)) = (q_n, p_n)$ ja aika-askeleen pituus $t' = h$, saadaan menetelmä

$$p_{n+1} = p_n - h\frac{\partial V(q_n)}{\partial q} \tag{4.6}$$

$$q_{n+1} = q_n + hp_{n+1}, \tag{4.7}$$

missä numero alaindeksissä merkitsee aika-askelta. Jos otetaan vielä mukaan edeltävä askel

$$q_n = q_{n-1} + hp_n, \tag{4.8}$$

ratkaistaan siitä p_n ja yhtälöstä (4.7) p_{n+1}, voidaan ne sijoittaa yhtälöön (4.6), jolloin saadaan loikkakeino

$$q_{n+1} - 2q_n + q_{n-1} = -h^2 \frac{\partial V(q_n)}{\partial q}. \tag{4.9}$$

Tässä muodossa menetelmä vaatii kahden edellisen askeleen arvon seuraavan laskemiseksi. Voimme kuitenkin approksimoida liikemäärää yhtälöillä

$$p_{n+1/2} = \frac{q_{n+1} - q_n}{h} \tag{4.10}$$

$$p_n = \frac{q_{n+1} - q_{n-1}}{2h}, \tag{4.11}$$

ja näiden avulla kirjoittaa yhtälö (4.9) uudelleen muodossa

$$p_{n+1/2} - p_{n-1/2} = -h \frac{\partial V(q_n)}{\partial q}. \tag{4.12}$$

Yhtälöistä (4.10) ja (4.11) seuraa myös relaatio

$$p_{n+1/2} - 2p_n + p_{n-1/2} = 0, \tag{4.13}$$

jonka avulla voidaan eliminoida $p_{n-1/2}$ yhtälöstä (4.12). Näin voidaan menetelmä kirjoittaa muodossa

$$p_{n+1/2} = p_n + \frac{h}{2} f(q_n) \tag{4.14}$$

$$q_{n+1} = q_n + hp_{n+1/2} \tag{4.15}$$

$$p_{n+1} = p_{n+1/2} + \frac{h}{2} f(q_{n+1}), \tag{4.16}$$

missä on merkitty $-\frac{\partial V(q)}{\partial q} = f(q)$. Näillä merkinnöillä alussa valitun systeemin liikeyhtälöt voidaan kirjoittaa muodossa $\ddot{q} = f(q)$. (Hairer et al., 2006)

Yllä tehty johto vaati, että liikeyhtälöt ovat muotoa $\ddot{q} = f(q)$. Yleisemmin loikkakeino toimii, kun Hamiltonin funktio on muotoa $H(\mathbf{q}, \mathbf{p}) = T(\mathbf{p}) + V(\mathbf{q})$ (Pihajoki, 2015).

4.2 Loikkakeinon symplektisyys

Näytetään ensin, että yhtälöiden (4.6) ja (4.7) määrittelemä menetelmä on symplektinen. Lasketaan menetelmän määrittelemän muunnoksen Jacobin matriisi

$$A = \begin{pmatrix} \frac{\partial q_{n+1}}{\partial q_n} & \frac{\partial q_{n+1}}{\partial p_n} \\ \frac{\partial p_{n+1}}{\partial q_n} & \frac{\partial p_{n+1}}{\partial p_n} \end{pmatrix} = \begin{pmatrix} 1 + h^2 f'(q_n) & h \\ h f'(q_n) & 1 \end{pmatrix}. \tag{4.17}$$

Sitten voimme tarkistaa menetelmän symplektisyyden sijoittamalla saatu matriisi ehtoon (2.15).

$$\begin{pmatrix} 1 + h^2 f'(q_n) & h \\ h f'(q_n) & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 1 + h^2 f'(q_n) & h f'(q_n) \\ h & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{4.18}$$

Siis ehto $AJA^T = J$ toteutuu, joten tämä menetelmä on symplektinen. Tästä seuraa myös loikkakeinon symplektisyys, koska loikkakeino saatiin käyttäen vain useampaa edellisen menetelmän askelta (Hairer et al., 2006).

4.3 Toteutus

Loikkakeinon rajoituksien vuoksi käytetään liikeyhtälöiden muotoa (2.20). Eli nyt $f(q) = -\omega^2 q$. Valitaan alkuarvot q_0 ja p_0, minkä jälkeen menetelmää voidaan iteroida askel kerrallaan.

Ensin lasketaan liikemäärä askeleen puolessa välissä

$$p_{1/2} = p_0 \frac{h}{2} \omega^2 q_0, \tag{4.19}$$

minkä jälkeen voidaan laskea paikan ja liikemäärän arvot askeleen lopussa

$$q_1 = q_0 + h\, p_{1/2} \tag{4.20}$$

$$p_1 = p_{1/2} - \frac{h}{2} \omega^2 q_1. \tag{4.21}$$

Vertailua verten tehtiin Python ohjelma, joka käyttää yllä mainittua menetelmää. Ohjelman lähdekoodi on liitteessä A.

5. Vertailu

Aiemmin luvuissa 3 ja 4 käytiin läpi loikkakeino- ja RK4-menetelmien toimintaperiaatteet ja toteutukset. Tässä luvussa menetelmiä testataan käytännössä ja niillä saatuja tuloksia vertaillaan.

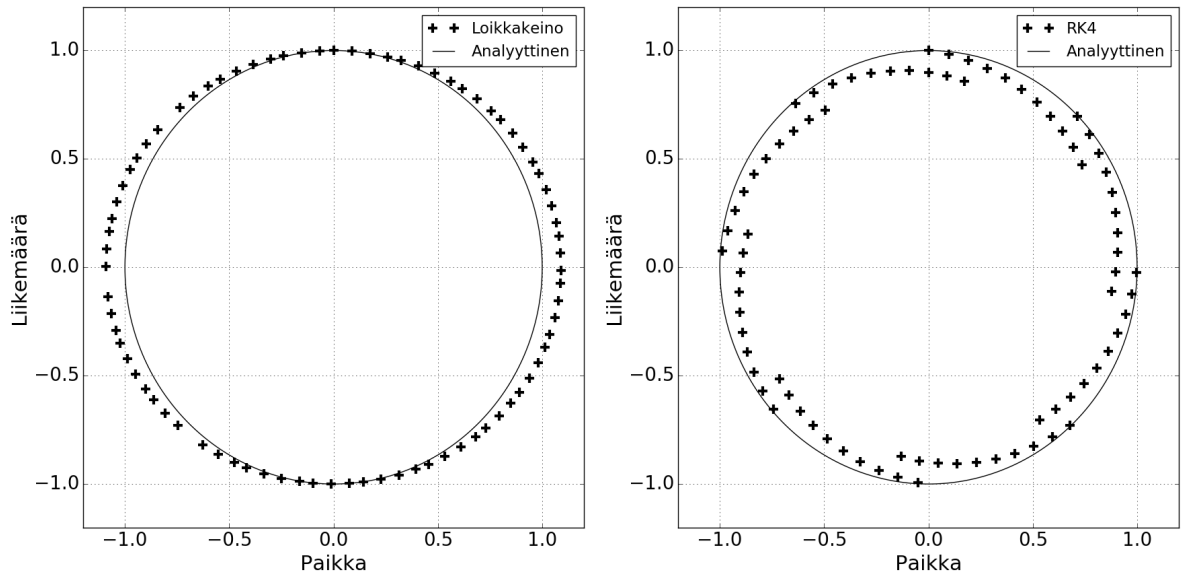
Jotta menetelmiä voitaisiin käyttää, täytyy tietää systeemin alkuarvot ja vakioiden suuruudet. Koska tarkoituksena on vain vertailla menetelmiä, valitaan alkuarvoiksi

$$p_0 = 1.0\,, \quad q_0 = 0.0\,. \tag{5.1}$$

Kun valitaan vielä vakioiksi $\omega = 1$ ja $m = 1$ saadaan analyyttiselle ratkaisulle (2.21) arvot

$$A = 1.0\,, \quad \phi = 0.0\,. \tag{5.2}$$

Näiden valintojen vuoksi saaduilla tuloksilla ei ole yksiköitä ja analyyttinen ratkaisu on faasiavaruuden käyrä, joka seuraa yksikköympyrää kuten kuvasta 5.1 nähdään.



Kuva 5.1: Loikkakeinolle (vas.) ja RK4-menetelmälle (oik.) laskettiin 80 ensimmäistä arvoa aika-askeleella $h = 0.8$.

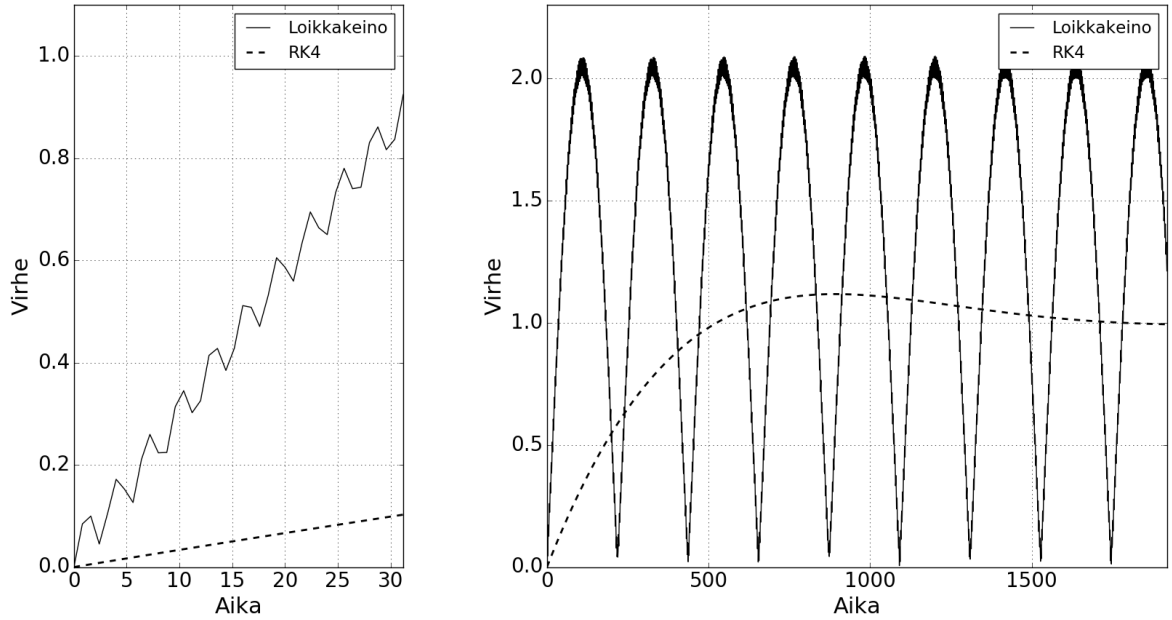
Menetelmien vertailussa tarkastellaan ensin kuinka paljon ne eroavat analyyttisestä ratkaisusta. Tämä tehdään laskemalla jokaisen menetelmällä lasketun pisteen etäisyys

analyyttisestä ratkaisusta vastaavalla ajanhetkellä. Etäisyys lasketaan faasiavaruudessa, eli huomioon otetaan sekä paikka että liikemäärä. Numeerisesti lasketun pisteen (q_l, p_l) etäisyydeksi analyyttisestä ratkaisusta (q_a, p_a) saadaan siis

$$d = \sqrt{(q_a - q_l)^2 + (p_a - p_l)^2}. \tag{5.3}$$

Koska etäisyys kertoo kuinka kaukana menetelmän antama arvo on tarkasta arvosta, kutsutaan tätä etäisyyttä tästä eteenpäin virheeksi.

Kuvassa 5.2 nähdään numeeristen menetelmien laskemien arvojen virheen kehitys ajan kuluessa.

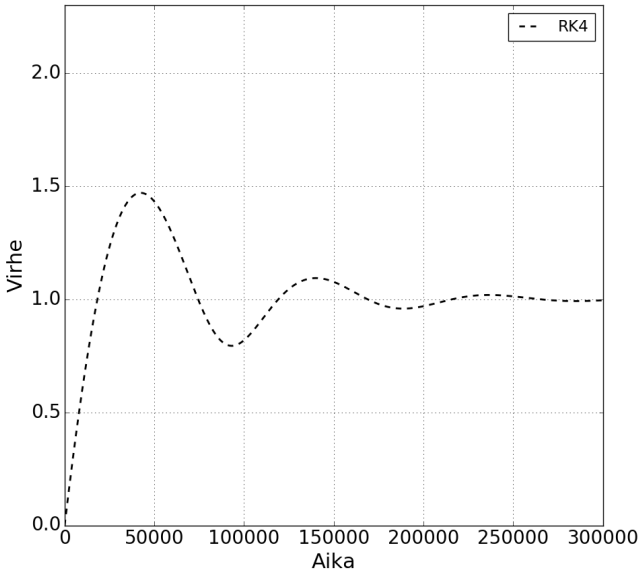


Kuva 5.2: Numeeristen menetelmien ($h = 0.8$) virheen suuruus ensimmäisten 40 askeleen (vas.) ja 2400 askeleen (oik.) aikana. Virheellä tarkoitetaan nyt kaavalla (5.3) määriteltyä etäisyyttä analyyttisestä ratkaisusta.

Virheen kehityksestä voi huomata, että loikkakeinon virhe kasvaa aluksi nopeammin kuin RK4-menetelmän. Pidemmällä aikavälillä ilmenevän oskillaation ja kuvassa 5.1 nähtävän radan muodon perusteella voi päätellä, että suurin osa loikkakeinon virheestä on vaihevirhettä. Eli loikkakeino ei onnistu kuvaamaan harmonisen oskillaattorin, jota systeemi kuvaa, värähdyksen jaksonaikaa yhtä tarkasti kuin RK4-menetelmä.

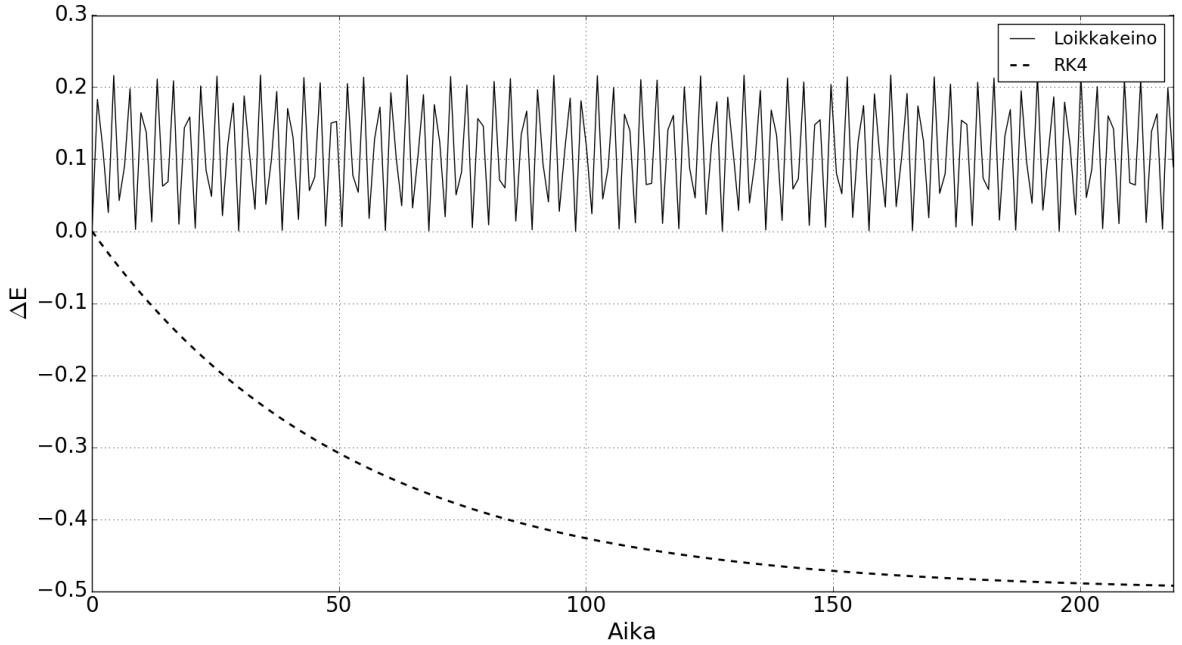
Lisäksi loikkakeinon virheessä esiintyy pienempi oskillaatio, jolla on huomattavasti lyhyempi jaksonaika. Tämä johtuu faasiavaruuden radan muodosta, jota loikkakeino seuraa. Kuvassa 5.1 nähdään selvästi kuinka loikkakeinon laskemat pisteet ovat paikka-akselilla liian kaukana origosta.

RK4-mentelmän vaihevirhe on pienempi. Vaihevirheestä tulee merkittävämpi pienemmillä aika-askeleilla, kuten kuvasta 5.3 huomataan. Virhe kuitenkin lähestyy aina arvoa yksi ja jää siihen.



Kuva 5.3: RK4-menetelmän virhe pienemmällä aika-askeleella ($h = 0.3$).

Kuvan 5.1 perusteella voidaan päätellä, että RK4-menetelmän laskemat pisteet lähestyvät origoa, mikä selittää myös sen miksi virheen suurus lähestyy arvoa yksi. Koska systeemin kokonaisenergia riippuu liikemäärän ja paikan neliöistä, pisteen etäisyys origosta faasiavaruudessa kuvaa pisteen energiaa. Voidaan siis päätellä, että RK4-menetelmä menettää energiaa ajan-kuluessa. Kuvassa 5.4 on kokonaisenergian virheen kehitys molemmalle menetelmälle.



Kuva 5.4: Kokonaisenergian virheen (ΔE) kehitys ajan kuluessa numeerisilla menetelmillä ($h = 1.1$). Virhe on laskettu numeerisen ja analyyttiseen ratkaisun erotuksena. Analyyttisen ratkaisun antama energia on 0.5 .

Kuten muista tuloksista pystyi päättelemään, RK4-menetelmän laskema kokonaisenergia vähenee ajan kuluessa, ja lähestyy lopulta nollaa. Loikkakeinon laskema energia sen sijaan heilahtelee edestakaisin, mutta pysyy rajoitetulla välillä.

6. Päätelmät

Loikkakeinon ja RK4-menetelmän numeeriset ratkaisut harmoniselle oskillaattorille käyttäytyivät eri tavoilla. Loikkakeino säilytti systeemin kulkeman radan muodon samana, mutta ajautui nopeasti eri vaiheeseen analyyttisen ratkaisun kanssa. RK4-menetelmä kuvasi värähtelytaajuuden paremmin, mutta ei onnistunut säilyttämään systeemin kokonaisenergiaa.

Menetelmien välillä oli suuri tarkkuusero heti ensimmäisten aika-askelten jälkeen, mikä ilmeni kuvassa 5.2. Tämä selittyy osittain sillä, että loikkakeino on toisen kertaluokan menetelmä ja RK4 puolestaan neljännen. Toisaalta tämä tarkoittaa myös sitä, että RK4 on laskennallisesti raskaampi. Loikkakeinosta on myös korkeamman kertaluokan variantteja (Hut et al., 1995), joilla voi mahdollisesti saada tarkempia tuloksia myös lyhyellä ajanjaksolla.

Pidemmällä aikavälillä kokonaisenergian kehitys antaa paremman kuvan menetelmien eroista. RK4-menetelmän laskemien arvojen virhe kasvoi ajan kuluessa. Loikkakeinon laskema energia sen sijaan pysyi rajoitetulla välillä, mikä on seurausta siitä, että loikkakeino säilyttää systeemin kulkeman radan muodon. Radan säilyminen on puolestaan seurausta loikkakeinon symplektisyydestä.

Eli lyhyellä ajanjaksolla Runge-Kutta antoi tarkempia tuloksia, mutta kun aikaa kului enemmän symplektinen loikkakeino säilytti realistisemman kuvan systeemistä.

Liitteet

A Loikkakeino-ohjelman lähdekoodi

```
# Loikkakeino
def leapfrog(h, t, q0, p0, ddq):
    # liikemaara askeleen puolessa valissa
    p12 = p0 + ddq(t, q0)*h*0.5
    # paikka askeleen lopussa
    q1 = q0 + p12*h
    # liikemaara askeleen lopussa
    p1 = p12 + ddq(t, q1)*h*0.5
    return [q1, p1]

def calculate(steps, h, t0, q0, p0, ddq):
    q = [q0]
    p = [p0]
    t = t0
    # lasketaan arvot jokaiselle askeleelle ja lisataan ne listaan
    for _ in range(steps):
        tmp = leapfrog(h, t, q[-1], p[-1], ddq)
        q.append(tmp[0])
        p.append(tmp[1])
        t += h
    return q, p

def main():
    steps = 50
    h = 0.9
    # maaritellaan paikan toinen aikaderivaatta
    def ddq(t, q): return -q
    # lasketaan systeemin ja sen kokonaisenergian kehitys
    lf_plot = calculate(steps, h, 0, 0.0, 1.0, ddq)
    tot_e_lf = [(lf_plot[1][i]**2)/2 + (lf_plot[0][i]**2)/2 for i in range(steps)]
```

B Runge-Kutta-ohjelman lähdekoodi

```
# Runge–Kutta
def rk4(h, t, q0, p0, dq, dp):
    # lasketaan kulmakertoimet
    k1 = h*dq(t, q0, p0)
    l1 = h*dp(t, q0, p0)

    k2 = h*dq(t+0.5*h, q0+0.5*k1, p0+0.5*l1)
    l2 = h*dp(t+0.5*h, q0+0.5*k1, p0+0.5*l1)

    k3 = h*dq(t+0.5*h, q0+0.5*k2, p0+0.5*l2)
    l3 = h*dp(t+0.5*h, q0+0.5*k2, p0+0.5*l2)

    k4 = h*dq(t+h, q0+k3, p0+l3)
    l4 = h*dp(t+h, q0+k3, p0+l3)
    # paikka ja liikemaara askeleen lopussa
    q1 = q0 + (k1 + 2*k2 + 2*k3 + k4)/6.0
    p1 = p0 + (l1 + 2*l2 + 2*l3 + l4)/6.0
    return [q1, p1]

def calculate(steps, h, t0, q0, p0, dq, dp):
    q = [q0]
    p = [p0]
    t = t0
    # lasketaan arvot jokaiselle askeleelle ja lisataan ne listaan
    for _ in range(steps):
        tmp = rk4(h, t, q[-1], p[-1], dq, dp)
        q.append(tmp[0])
        p.append(tmp[1])
        t += h
    return q, p

def main():
    steps = 50
    h = 0.9
    # maaritellaan paikan ja liikemaaran aikaderivaatat
    def dq(t, q, p): return p
    def dp(t, q, p): return -q
    # lasketaan systeemin ja sen kokonaisenergian kehitys
    rk_plot = calculate(steps, h, 0, 0.0, 1.0, dq, dp)
    tot_e_rk = [(rk_plot[1][i]**2)/2 + (rk_plot[0][i]**2)/2 for i in range(steps)]
```

Kirjallisuutta

Arnold, V. I. (1989). *Mathematical Methods of Classical Mechanics*. Springer.

Butcher, J. C. (1963). Coefficients for the study of Runge-Kutta integration processes. *Journal of the Australian Mathematical Society*, 3(2):185–201.

Hairer, E., Lubich, C., and Wanner, G. (2006). *Geometric Numerical Integration*. Springer.

Hairer, E., Norsett, S. P., and Wanner, G. (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer.

Hut, P., Makino, J., and McMillan, S. (1995). Building a better leapfrog. *The Astrophysical Journal*, 443:L93–L96.

Kutta, W. (1901). Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys.*, 46:435–453.

McLachlan, R. I. and Quispel, G. R. W. (2002). Splitting methods. *Acta Numerica*, 11:341–434.

Nolte, D. D. (2015). *Introduction to Modern Dynamics: Chaos, Networks, Space and Time*. Oxford University Press.

Pihajoki, P. (2015). Explicit methods in extended phase space for inseparable hamiltonian problems. *Celestial Mechanics and Dynamical Astronomy*, 121:211–231.

Ruth, R. D. (1983). A canonical integration technique. *IEEE Trans. Nucl. Sci.*, 30(CERN-LEP-TH-83-14):2669–2671.

Tuominen, K. (2017). Analytical mechanics. https://geom.mathstat.helsinki.fi/moodle/pluginfile.php/44044/mod_resource/content/4/amek_notes.pdf.