Kandidaatintutkielma
Fysikaalisten tieteiden kandiohjelma
Teoreettinen fysiikka

Symplektiset integrointimenetelmät

Arttu Hyvönen

11.5.2020

Ohjaaja(t): Pauli Pihajoki

Tarkastaja(t): arvostelija Testi
arvostelija Arvostelija

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

Symplektiset integrointimenetelmät

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Kandidaatintutkielma	11.5.2020	21

Tiivistelmä — Referat — Abstract

Kirjoita tiivistelmään lyhyt, enintään 250 sanan yhteenveto työstäsi: mitä olet tutkinut, millaisia menetelmiä olet käyttänyt, millaisia tuloksia sait ja millaisia johtopäätöksiä niiden perusteella voi tehdä.

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1. Johdanto

rakenne

* taustaa: numeerisia menetelmiä
* mitä työllä haetaan: Symplektisyys on ihan kätevä ominaisuus menetelmälle
* mitä ihmettä: selitetään hamilton + symplektisyys + menetelmät
* miten näytetään: Verrataan 2 menetelmää, symp. ja ei symp.

2. Liikeyhtälöt

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

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

Lagrangen mekaniikka

Fysikaaliset systeemit useimmiten kehittyvät ajassa seuraten pienimmän vaikutuksen periaatetta[†]. Eli toisin sanoen systeemin, jolla on N vapausastetta, kehitys saadaan funktionaalin

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

ääriarvona. Missä Lagrangen funktio L on kineettisen- ja potentiaalienergian erotus, $\mathbf{q}$ on yleistetty N ulotteinen koordinaattivektori ja $\dot{\mathbf{q}}$ vastaava nopeus. Koordinaattivektoreiden komponentteja tullaan merkitsemään alaindeksillä i.

Funktionaalista 2.1 voidaan edelleen johtaa Eulerin-Lagrangen yhtälöt

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

joista saadaan N toisen asteen differentiaaliyhtälöä systeemin kehitykselle. [Arnold, 1989]

Hamiltonin mekaniikka

Lagrangen mekaniikan avulla saadaan N toisen asteen differentiaaliyhtälöä. Hamiltonin mekaniikalla saman systeemin kehitys kuvataan ensimmäisen asteen differentiaaliyhtälöillä joita on $2N$ kappaletta.

[†]engl. Principle of least action

Legendre muuntamalla Lagrangen funktio $L(\mathbf{q}, \dot{\mathbf{q}}, t)$ 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 $\dot{q}_i$ yleistetyn liikemäärän

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

avulla. Nyt tarkastelemalla Hamiltonin funktion kokonaisderivaattaa päädytään Hamiltonin yhtälöihin

$$\dot{p}_i = -\frac{\partial H}{\partial q_i} \tag{2.5}$$

$$\dot{q}_i = \frac{\partial H}{\partial p_i} \tag{2.6}$$

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 miettiä, että systeemin tila on piste paikka ja liikemäärä koordinaattien 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 $V = V(\mathbf{q})$, voidaan Hamiltonin funktio kirjoittaa muodossa

$$H = T + V \tag{2.7}$$

joka on systeemin kokonaisenergia.[Arnold, 1989]

Symplektisyys

Jos otetaan Hamiltonisen systeemin määrittelemä vektorikenttä ja valitaan siitä pinta-ala. Kun annetaan pinta-alan kehittyä ajassa vektorikentän mukaan, sen koko säilyy (Kuva). Tätä ominaisuutta kutsutaan systeemin virtauksen symplektisyydeksi. [Hairer et al., 2006]

Hamiltoniseen systeemin voidaan tehdä muunnoksia, kuten koordinaattimuunnos, ja jos se säilyttää tämän ominaisuuden, sitä kutsutaan symplektiseksi. Jotta muunnos olisin symplektinen, sen pitää täyttää ehto.

Ensin helpottaa, jos määritellään Hamiltonin yhtälöt 2.5 ja 2.6 uudelleen vektorin $x = (q_1, \ldots, q_n, p_1, \ldots, p_n)$ ja $2n \times 2n$ matriisin

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \tag{2.8}$$

avulla. Matriisin komponentti I on $n \times n$ identiteetti matriisi. Näin Hamiltonin yhtöille saadaan muoto

$$\dot{x} = J\frac{\partial H}{\partial x} \tag{2.9}$$

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

$$AJA^T = J \tag{2.10}$$

niin sanotaan, että se on symplektinen.[Tuominen, 2017]

Myös muut muunnokset voivat olla symplektisiä. Eli voidaan kirjoittaa yleisemmin, jos muunnos $M : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ toteuttaa yhtälön $MJM^T = J$, kutsutaan sitä symplektiseksi. Numeeristen mentelmien kannalta on merkittävää, että muunnos aikaaskeleesta seuraavan voi täytää symplektisyyden ehdot, jolloin menetelmä itsessään on symplektinen ja säilyttää systeemin ominaisuudet.

Harmoninen oskillaattori

Harmonisen oskillaattorin Lagrangen funktio on

$$L(q, \dot{q}) = \frac{1}{2}m\dot{q}^2 - \frac{1}{2}m\omega^2 q^2 \tag{2.11}$$

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.

$$H(q, p) = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 q^2 \tag{2.12}$$

Hamiltonin funktiosta pystytään nyt ratkaisemaan liikeyhtälöt

$$\dot{p} = -m\omega^2 q \tag{2.13}$$

$$\dot{q} = \frac{p}{m} \tag{2.14}$$

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

$$\ddot{q} = -\omega^2 q \tag{2.15}$$

jonka ratkaisu on yleisesti tunnettu.

$$q(t) = A \sin(\omega t + \phi) \tag{2.16}$$

missä A ja ϕ riippuvat systeemin alkuarvoista.

3. Runge-Kutta

RK johdantoa

Ensimmäisenä menetelmänä käytetään neljännen asteen Runge-Kutta menetelmää.

Menetelmän toiminta/johto

Menetelmän johdossa seuraillaan vastaavaa johto, jonka tekivät Hairer et al. [2006]. Runge-Kutta menetelmällä voidaan ratkaista ensimmäisen asteen differentiaaliyhtälöitä. Eli yhtälöitä, 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 ratkaiseminen voidaan aloittaa.

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

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

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 jokaisella on oma painotuksensa. Eli seuraavan askeleen arvo n:llä kulmakertoimella saadaan summalla

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

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.

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 kuten taulukossa 3.1. Eksplisiittisessä tapauksessa kertoimet a_{ij}, joissa $i \leq j$ ovat nollia ja monesti jätetään siksi taulukossa tyhjiksi.

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

Toteutus

Vertailussa käytetään neljännen asteen Runge-Kutta menetelmää, joka tunnetaan nimellä Runge-Kutta menetelmä[†]. Wilhelm Kutta esitteli menetelmän 1901, minkä jälkeen se on ollut laajassa käytössä. Taulukossa 3.2 on menetelmän kertoimet. [Hairer et al., 1993]

[†]engl. The Runge-Kutta method tai classic Runge-Kutta method

Taulukko 3.2: Butcher taulukko Runge-Kutta 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 2.13 ja 2.14. Eli yhtälöt eivät ole muotoa $\dot{y} = f(t, y)$, jolle menetelmä johdettiin. Sen sijaan systeemissä on kaksi yhtälöä, jotka riippuvat toisistaan.

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

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

Menetelmää voidaan silti käyttää. Ensin lasketaan ensimmäiset kulmakertoimet.

$$k_{p,1} = f(q_0) = -lq_0\,, \quad k_{q,1} = g(p_0) = \frac{1}{m}p_0 \tag{3.10}$$

Toisia kulmakertoimia laskiessa käytetään ensimmäisessä vaiheessa saatuja kulmakermia niin kuin yhtälössä 3.7 määritelläänkin. Nyt vain täytyy käyttää sitä kulmakerrointa, josta funktio riippuu. Eli liikemäärän kulmakertoimia laskiessa käytetään paikan kulmakertoimia ja päinvastoin. Näin saadaan loput kolme paria kulmakertoimia.

$$k_{p,2} = -l(q_0 + \frac{h}{2}k_{q,1})\,, \quad k_{q,2} = \frac{1}{m}(p_0 + \frac{h}{2}k_{p,1}) \tag{3.11}$$

$$k_{p,3} = -l(q_0 + \frac{h}{2}k_{q,2})\,, \quad k_{q,3} = \frac{1}{m}(p_0 + \frac{h}{2}k_{p,2}) \tag{3.12}$$

$$k_{p,4} = -l(q_0 + hk_{q,3})\,, \quad k_{q,4} = \frac{1}{m}(p_0 + hk_{p,3}) \tag{3.13}$$

Sitten voidaan laskea seuraavan askeleen arvo käyttäen saatuja kulmakertoimia.

$$p_1 = p_0 + \frac{h}{6}(k_{p,1} + 2k_{p,2} + 2k_{p,3} + k_{p,4}) \tag{3.14}$$

$$q_1 = q_0 + \frac{h}{6}(k_{q,1} + 2k_{q,2} + 2k_{q,3} + k_{q,4}) \tag{3.15}$$

Vertailua varten tehtiin Python koodi, joka käyttää yllä mainittua menetelmää. Koodi on liitteessä B.

4. Loikkakeino

Loikkakeino johdantoa

Johto

Loikkakeinon johto.

- rajoitukset
- symplektisyys

Toteutus

Menetelmän toteutus.

5. Vertailu

Aiemmin luvuissa 3 ja 4 käytiin läpi loikkakeino -ja Runge-Kutta menetelmien toimintaperiaatteet ja toteutukset. Nyt menetelmät laitetaan käytäntöön ja niillä saatuja tuloksia vertaillaan.

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

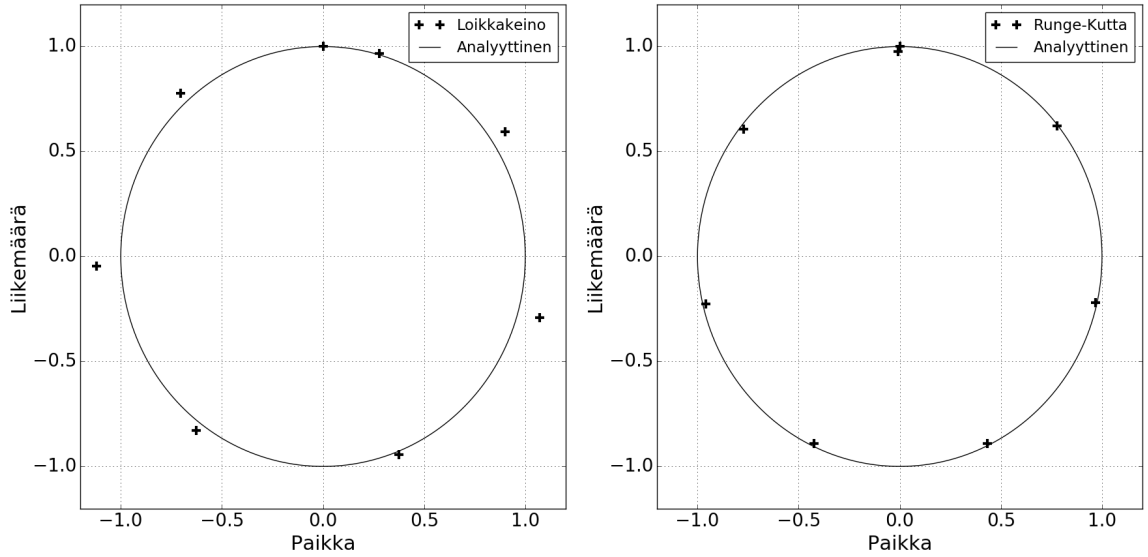
$$p_0 = 1.0\,, \quad q_0 = 0.0 \tag{5.1}$$

Kun valitaan vielä vakioiksi $\omega = 1$ ja $m = 1$ saadaan analyyttiselle ratkaisulle 2.16 arvot

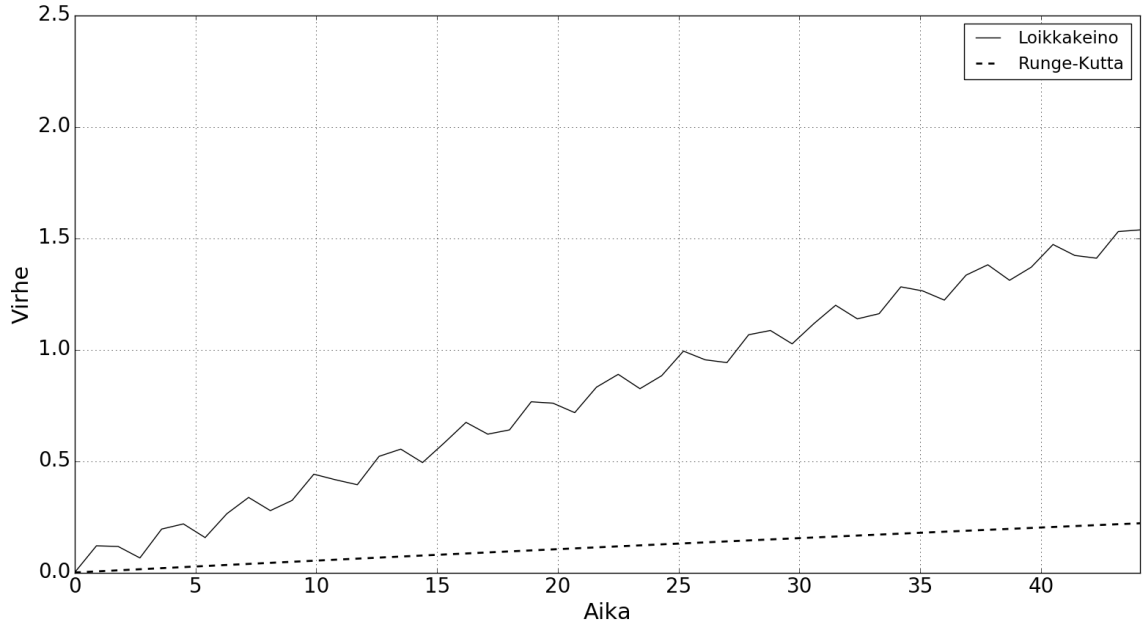
$$A = 1.0\,, \quad \phi = 0.0 \tag{5.2}$$

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

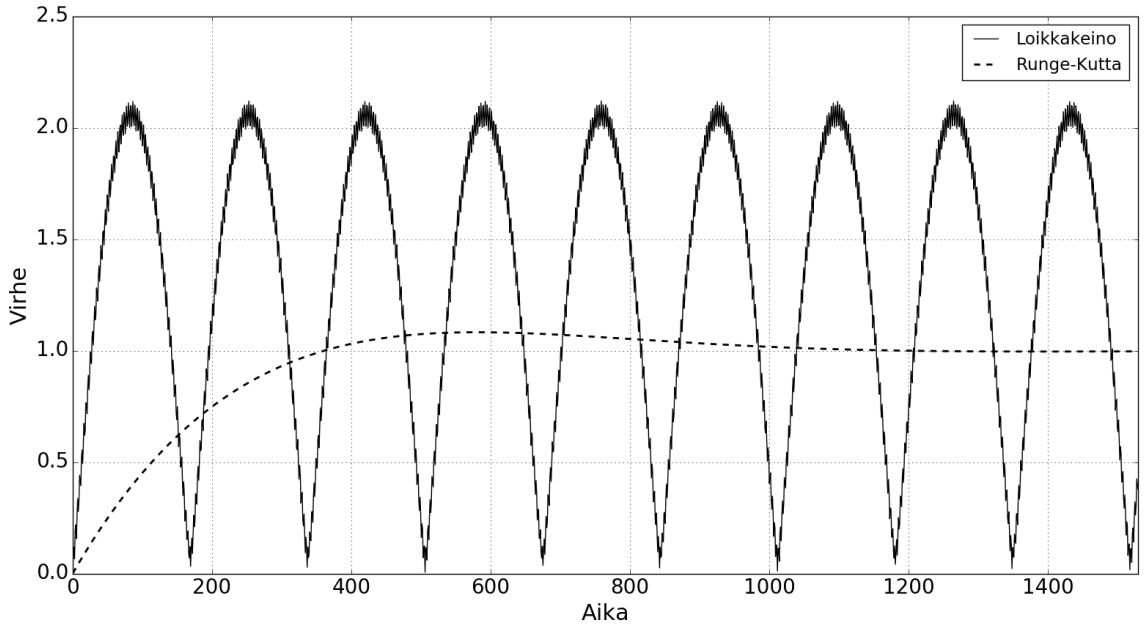
Menetelmien vertailussa tarkastellaan ensin kuinka paljon ne eroavat analyyttisestä ratkaisusta. Tämä tehdään laskemalla jokaisen menetelmällä lasketun pisteen etäisyys analyyttisestä ratkaisusta faasiavaruudessa vastaavalla ajanhetkellä.



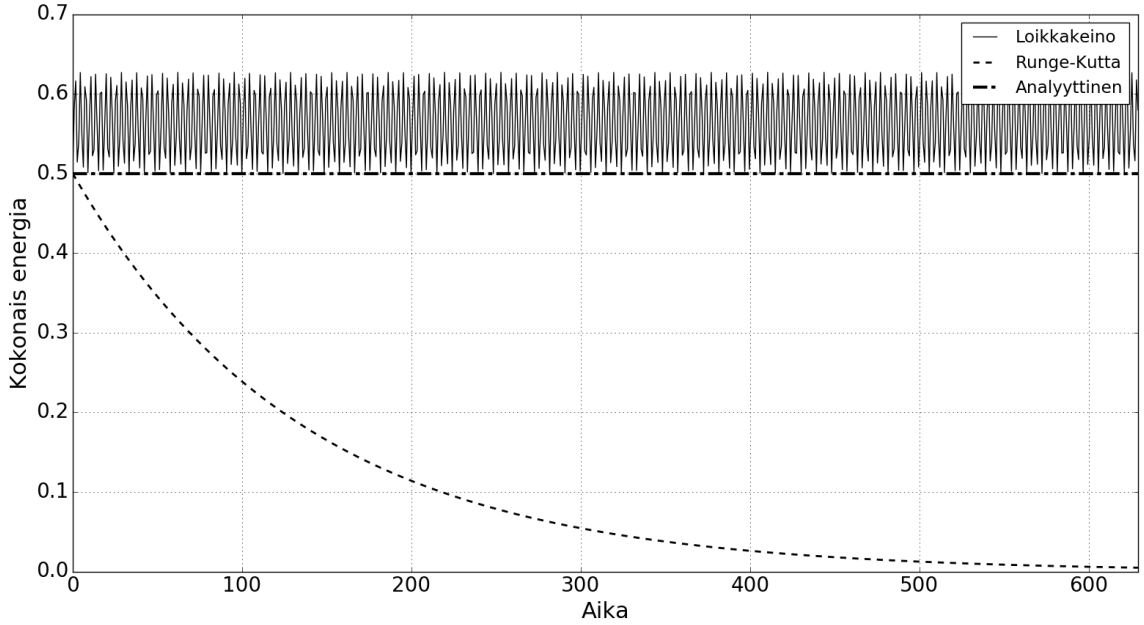
Kuva 5.1: Harmonisen oskillaattorin ensimmäinen värähdys kuvattuna faasiavaruudessa. Numeerisilla menetelmillä laskettiin seitsemän ensimmäistä askelta aika-askeleella 0.9.



Kuva 5.2: Numeeristen menetelmien virhe ajankuluessa.



Kuva 5.3: Numeeristen menetelmien virhe pidemmällä ajanjaksolla.



Kuva 5.4: Kokonaisenergian kehitys ajan kuluessa numeerisilla menetelmillä.

6. Päätelmät

Lyhyellä ajanjaksolla Runge-Kutta antaa tarkempia tuloksia, mutta kun aikaa kuluu enemmän loikkakeino säilyttää realistisemman kuvan systeemistä.

Liitteet

Loikkakeino menetelmän koodi

```
# 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)]
```

Runge-Kutta menetelmän koodi

```
# 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(method, 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 = method(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.

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.

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

Tuominen, K. (2017). Analytical mechanics.