Esa Lappi

Numeeriseen integrointiin perustuvista differentiaaliyhtälön ratkaisumenetelmistä ja niiden opettamisesta

Tampereen yliopisto
Informaatiotieteiden
tiedekunta
Lisensiaatintutkimus
Elokuu 2005

TAMPEREEN YLIOPISTO
INFORMAATIOTIETEIDEN TIEDEKUNTA
ESA LAPPI: Numeeriseen integrointiin perustuvista differentiaaliyhtälön ratkaisumenetelmistä ja niiden opettamisesta
Lisensiaatin tutkimus, 47 s. + liitteet 111s.
Elokuu 2005

TIIVISTELMÄ

Tutkimuksessa tarkastellaan sekä numeeriseen integrointiin perustuvien menetelmien hyvyyttä differentiaaliyhtälön alkuarvo-ongelman numeerisessa ratkaisussa että mitä pitäisi opettaa differentiaaliyhtälöistä ja numeerisista menetelmistä suoraan peruskoulusta osa-aikaisesti it-alan työelämään siirtyville opiskelijoille.

Ensimmäisen kertaluvun differentiaaliyhtälöiden ratkaisun tarkastelussa keskitytään separoituviin differentiaaliyhtälöihin ja lineaariseen differentiaaliyhtälöön, koska ne ratkeavat integroimalla.

Separoituville differentiaaliyhtälöille esitetään Newtonin menetelmän ja numeerisen integroinnin yhdistävä ratkaisualgoritmi, joka on helposti toteutettavissa laskimen näppäilysarjana ja ohjelmointikielillä. Nopeusvertailujen perustella Gaussin integrointiin yhdistetty Newtonin menetelmä oli yleensä nopeampi kuin verrokkina käytetty Rungen-Kuttan menetelmä, kun integrointiin perustuvaa algoritmia on tuettu suppenemista ja integroinnin tarkkuutta ylläpitävillä kontrollirakenteilla. Ensimmäisen kertaluvun lineaariselle yhtälölle tunnetaan yleisesti analyyttinen integrointeihin perustuva ratkaisu, jonka numeerinen ratkaisu oli myös verrokkia nopeampi.

Toisen kertaluvun yhtälöiden osalta tarkastellaan joitakin erikoistapauksia, mutta niissä ei yleisesti havaittu vastaavaa nopeusetua kuin ensimmäisen kertaluvun differentiaaliyhtälöiden numeerisessa ratkaisemisessa.

Tutkimuksessa kuvataan myös Päivölän kansanopiston matematiikkalinjan kesälukukauden osalta erityisesti sen numeerisiin menetelmiin keskittyvän viimeisen opiskeluviikon osalta. Teollisiin esimerkkien perustuvana opetuksen tavoitteena on yhtälön ja yhtälöryhmien ratkaiseminen, ratkaisujen tarkkuuden ja yksikäsitteisyyden osoitusmenetelmät, numeerisen integrointi, separoituvien differentiaaliyhtälöiden ratkaiseminen sekä analyyttisesti että numeerisesti ja antaa kyky ohjelmoida lausekkeita liukuluvuille mielekkäästi ja ohjelmoida numeeriset ratkaisut ensimmäisen ja toisen kertaluvun differentiaaliyhtälöille sekä ymmärtää askelpituuden muutoksen vaikutus Eulerin ja Rungen-Kuttan menetelmää soveltavia simulointimalleja käyttäessä.

Asiasanat: Separoituva differentiaaliyhtälö, Numeeriset menetelmät, Numeeristen menetelmien opetus.

UNIVERSITY of TAMPERE
The Faculty of Information Sciences
ESA LAPPI: Numeeriseen integrointiin perustuvista differentiaaliyhtälön ratkaisumenetelmistä ja niiden opettamisesta
Licentiate thesis, 47 pp. + appendixes 111pp.
August 2005

Abstract

In this work solving initial value problems of differential equations using numerical integration methods is studied. Also a curriculum of numerical methods and differential equations is studied. The curriculum must fit the talented students starting their upper secondary school, and it must give immediate ability to work (part time) in information technology.

The main focus of the mathematical part of this work are separable differential equations and partly first order linear differential equations. A algorithm to solve separable differential equation using numerical integration and the Newton's method for equations is presented. The new algorithm was found faster than the Runge-Kutta method, when the Gaussian quadrature and the Newton's method with some extra features in order to achieve convergence and sufficiently accurate integration were used. Also linear equation was solved faster with numerical integration.

Also some second order differential equations are considered, but the integration based methods were not generally superior to Runge-Kutta method.

The educational part of the study describes the eight week summer term in the Mathematics Program of Päivölän kansanopisto and especially learning of numerical methods during the last weeks. The curriculum is based on industrial examples and main topics are numerical errors and coding formulas without losing significant digits, equation solving and the methods of showing accuracy of the answer, numerical integration, solving separable differential equations (analytical and numerical methods), differential equations and error analysis when using Euler's or Runge-Kutta's method for systems of differential equations

Key words:
Separable Differential Equations. Numerical methods. Teaching mathematics.

Kiitokset

Tätä työtä ovat edesauttaneet monet henkilöt ja heidän edustamansa organisaatiot. Siinä esitetty separoituvan differentiaaliyhtälön numeerinen ratkaisumenetelmä kehitettiin IVO International Oy:n palveluksessa, jolloin silloinen esimieheni Antti Norta toimi innoittajana hyvään tieteelliseen kieleen ja insinöörityöhön. Opintojen ja matematiikkaharrastuksen ylläpitäjänä maininnan ansaitsevat Päivölän kansanopiston matematiikkalinjan ja ammatillisen lisensiaattikoulutuksen käytännön järjestelyistä vastannut Kullervo Nieminen ja lisensiaattikoulutuksen totutuksesta vastannut professori Jorma Merikoski sekä opetuskokeiluihin osallistuneet Päivölän opiskelijat.

Laskimille sopivan differentiaaliyhtälön numeerisen ratkaisumenetelmän ensimmäinen opetuskokeilu tehtiin Helsingin matematiikkalukion matematiikan kesäkoulussa Uumajassa vuonna 2001, jonka opiskelijat osaltaan innoittivat jatkotyöhön. Oy Perkko tarjosi Matemaattisten aineiden opettajien liiton MAOL ry:n syyspäivillä mahdollisuuden esittää metodi käyttäen heidän laskimiaan, mistä myös kiitokset. Kiitoksen ansaitsevat myös jatko-opintoihin kannustavat työnantajat ja heidän edustajansa sekä Puolustusvoimien Teknillisellä Tutkimuslaitoksella että koulumaailmassa.

Erityisen kiitoksen ansaitsevat lapseni Linnea ja Teo sekä vaimoni Merikki, joka on aktiivisesti motivoinut jatkotutkinnon tekoon ja jaksanut kuunnella ja antaa ideoita matemaattisiin ratkaisuihin omista kiireistään huolimatta.

Esipuhe

Useat fysikaaliset ja teolliset sovellukset mallinnetaan ensimmäisen tai toisen kertaluvun differentiaaliyhtälöillä, joissa alkutila on tunnettu ja lopputila halutaan selvittää. Osaan differentiaaliyhtälöistä on tunnettu analyyttiset ratkaisutavat jo 1700-luvulta lähtien. Nämä ovat perustuneet yhtälön muokkaamiseen muotoon, jossa ratkaisu on voitu saavuttaa integroimalla muuttujat kerran tai kahdesti. Usein kuitenkaan tällainen integraalimuodossa esitetty ratkaisu ei ole riittänyt, koska tehtävässä syntyviä integraaleja ei osata tai voida ratkaista suljetussa muodossa.

Yleensä differentiaaliyhtälön alkuarvo-ongelman numeerinen ratkaisu alkaa muodosta $y'(x) = f(x,y)$, vaikka yhtälö voitaisiin kirjoittaa integraalimuotoon. Numeerisessa ratkaisussa ei tällöin ole käytetty hyväksi integroinneilla mahdollisesti saatavaa hyötyä.

Jos differentiaaliyhtälö saadaan muotoon, jossa se voidaan ratkaista numeerisella integroinnilla, ratkaisussa voidaan käyttää numeerisesti tehokkaampia menetelmiä. Tässä tutkimuksessa integrointiin on valittu Gaussin 1814 julkaisema metodi. On huomattava, että integroimismenetelmää voi tilanteen mukaan muuttaa muuksikin, jos se kyseisessä erikoistapauksessa on helpompaa. Esimerkiksi tasavälisissä mittaustuloksissa Newtonin - Cotesin kaavojen käyttö on mielekkäämpää.

Tässä matematiikan opettajien ammatillisen lisensiaattikoulutusohjelman lisensiaatintutkimuksessa kuvataan uutta separoituvan differentiaaliyhtälön numeerista ratkaisumenetelmää ja differentiaalialiyhtälöiden ja numeeristen menetelmien opettamista Päivölän kansanopiston matematiikkalinjan ja Helsingin matematiikkalukion opiskelijoille.

Uuden ratkaisumenetelmän perusidea on kirjoittaa alkuarvoprobleema integraalimuotoon ja tämän jälkeen hakea haluttu suure eli määrätyn integraalin yläraja sopivalla yhtälön numeerisella ratkaisumenetelmällä. Numeerinen integrointi yhdistettynä klassiseen Newtonin yhtälönratkaisumenetelmään ratkaisee separoituvat differentiaaliyhtälöt perinteisiä menetelmiä nopeammin erityisesti silloin, kun Newtonin menetelmän suppeneminen ei ole ongelma. Menetelmä ja sen tuloksia on esitetty tekijän julkaisuissa [9], [20], [21] ja [22]. Se kehitettiin alun perin teollisuuden paloriskianalyysin lämmönsiirto-ongelmiin, mutta sitä voi soveltaa muun muassa kaikkiin niin sanottuihin autonomisiin eli ajasta riippumattomiin fysikaalisiin probleemiin. Vaikka työ yhdistää vuosisatoja tunnetut yksinkertaiset perusmenetelmät, uutena lopputuloksena esitetään helposti opetettava ja ohjelmoitava, monet probleemat vähäisemmällä tietokoneajalla ratkaiseva algoritmi.

Osana tätä tutkimusta on myös kuvaus numeeristen menetelmien ja differentiaaliyhtälöiden opettamisesta jo lukioaikanaan teknologia-alalla osa-aikatyöt aloittaville opiskelijoille ja erityisesti tällaiselle ryhmälle opettavien asioiden tarkastelu välittömien työelämän vaatimusten kannalta. Opetuksen sisältö on valittu teollisuuden kokemusten perusteella, joihin liittyvät tekijän julkaisut [2, (toimistokerrostalon osuus)], [14], [23] ja [24] sekä raportit [25], [26], [27] ja [28].

Opetuksen sisältö käy ilmi liitteenä 1 olevasta differentiaaliyhtälöiden ja numeeristen menetelmien oppimateriaalista, joka samalla sisältää menetelmän laskentaesimerkkejä ja nopeusvertailuja. Liitteenä 2 on osa Oy Perkon julkaisemasta Lukion laskinoppi –opetusmonisteesta [22], jota on myös käytetty oppimateriaalina sekä Päivölässä että Helsingin matematiikkalukiossa. Liite 3 sisältää opitun arviointiin käytetyn MCM/ICM –kilpailun tuloskoosteen.

SISÄLLYS

1. Johdanto

Tämän lisensiaatintutkimuksen tavoitteena on esitellä laskimiin sopiva ja myös perinteisiä numeerisia ratkaisumenetelmiä nopeampi menetelmä separoituvien differentiaaliyhtälöiden ratkaisemiseksi ja kuvata lyhyesti Päivölän matematiikkalinjan differentiaaliyhtälöiden ja numeeristen menetelmien opetuksen tavoitteita.

Koska opiskelijoiden perustyökaluna käyttämät laskimet sisältävät valmistoimintona numeerisen integroinnin, tätä toimintoa hyödyntävät menetelmät piti selvittää ja kehittää niitä edelleen. Tämän tutkimuksen matemaattisen osan sisältönä on laskinten tehokkaan soveltamisen pohjalta kehitetty differentiaaliyhtälön numeerinen ratkaisumenetelmä ja sen vertailu perinteisiin tapoihin ratkaista differentiaaliyhtälöitä. Tutkimusmenetelmänä oli kehitetyn numeeriseen integrointiin perustuvan menetelmän vertailu muihin differentiaaliyhtälöiden numeerisiin ratkaisumenetelmiin sekä teoreettisesti että tietokoneelle ohjelmoitujen algoritmien laskenta-aikoja vertailemalla.

Työn opetuksellisen osan pohjimmaisena taustana ovat paloteknisen insinööritoimistoni Turvallisuusarviointi TA Oy:n aikana havaitsemani osaamistarpeet. Sitä täydentää Päivölän kansanopiston matematiikkalinjan ja Helsingin matematiikkalukion opetussuunnitelman ja kurssien suunnittelu, jonka yhteydessä pohdittiin, mitä välittömästi informaatioteknologia-alan työelämään ja toisaalta yliopistollisiin opintoihin suuntautuvalle opiskelijalle pitää opettaa numeerisista menetelmistä ja differentiaaliyhtälöistä.

Koska kumpikin oppilaitos on matematiikkaan suuntautunut, valtakunnalliset lukion opetussuunnitelman perusteet eivät liian suppeina kelpaa opetussuunnitelman pohjaksi. Toisaalta monien yliopistokurssien sisältö ja erityisesti esitietovaatimukset sekä oppimateriaali eivät kelpaa sellaisenaan juuri peruskoulunsa päättäneiden opintojen pohjaksi.

Siten tutkimusongelmaksi muotoutui, mitä pitää opettaa matematiikkaan suuntautuneille nuorille, kun reunaehtoina ovat käytössä oleva aika, opiskelijoiden rajalliset lähtötiedot, käytössä olevat tietotekniset työkalut ja työelämän ja jatko-opintojen tavoitteet.

Opettavien asioiden valinnan osalta keskitytään erilaisiin teollisiin havaintoihin lähinnä palotekniikasta, jossa on viime vuosikymmeninä kehitetty ja otettu käyttöön lukuisia numeerisia laskentamalleja. Opetuksen tavoitteiden rajoituksia ja

mahdollisuuksia käytössä olevan ajan puitteissa vahvistavat opetuksesta tehdyt havainnot sekä Päivölän Kansanopiston matematiikkalinjalla että Helsingin matematiikkalukiossa.

Opetuksen käytännön toteutusta ja opetukseen valitun aineksen soveltuvuutta muihin kouluihin ja muille oppilaille ei arvioida osana tätä tutkimusta, vaan se jätetään omat oppilaansa tuntevan ammattitaitoisen opettajan harkittavaksi.

2. Integroimalla ratkeavista differentiaaliyhtälöistä

2.1 Johdanto

Ensimmäisen kertaluvun differentiaaliyhtälöiden ratkaisun tarkastelussa keskitytään separoituviin differentiaaliyhtälöihin ja lineaariseen differentiaaliyhtälöön, koska ne ratkeavat integroimalla. Integroimalla ratkeavien differentiaaliyhtälöiden numeeriset tarkastelut tukevat analyyttisten menetelmien oppimista ja ovat teollisesti mielenkiintoisia. Niiden käsittelyn kiinnostavuutta lisää se, että ne ovat numeerisesti ratkaistavissa sekä halvoin laskimin että itse ohjelmoiduin ratkaisualgoritmein.

Separoituville differentiaaliyhtälöille esitetään Newtonin menetelmän ja numeerisen integroinnin yhdistävä ratkaisualgoritmi, joka on sekä helposti ohjelmoitavissa laskimiin että yleensä nopeampi kuin verrokkina käytetty Rungen-Kuttan menetelmä. Nopeusvertailut on esitelty liitteen 1 kolmannessa luvussa ja ohjelmakoodit ovat alaliitteinä A1-A4.

Ensimmäisen kertaluvun lineaariselle yhtälölle tunnetaan yleisesti analyyttinen integrointeihin perustuva ratkaisu, jonka numeerinen ratkaisu on esitetty mm. liitteen 1 luvussa 3.3 ja ohjelmakoodina alaliitteessä A5. Nopeusvertailujen valossa suora integrointi on tässäkin käyttökelpoinen ratkaisu. Hämmästyttävää kyllä sitä ei opeteta edes analyyttisenä ratkaisumenetelmänä suomalaisen lukion differentiaaliyhtälöitä käsittelevissä oppikirjoissa [11], [12], [13], [32] ja [33].

Toisen kertaluvun yhtälöiden osalta tarkastellaan joitakin erikoistapauksia, mutta niissä ei yleisesti saavuteta yhtä selkeää nopeusetua kuin ensimmäisen kertaluvun differentiaaliyhtälöiden numeerisessa ratkaisemisessa.

2.2 Separoituva yhtälö

2.2.1 Analyyttinen ratkaisu

Vaikka separoituvan yhtälön ratkaisu on ennestään hyvin tunnettu, se esitetään tässä johdantona numeeriselle menetelmälle. Tutkitaan separoituvaa differentiaaliyhtälöä muodossa

(1) $\frac{y'}{g(y)} = f(x)$.

Jos $g(y_0) = 0$, yhtälö (1) ei ole määritelty. Jos differentiaaliyhtälö on kirjoitettu muodossa, jossa $g(y_0)$ ei ole nimittäjänä, yhtälöllä on tässä tapauksessa vain vakioratkaisu $y = y_0$. Jos $g(y)$ saavuttaa laskennan aikana arvon nolla, yhtälön ratkaisukäyrä on saavuttanut vakiona pysyvän ”tasapainoarvon”. Tarkastelussa rajoitutaan tämän vuoksi vain niihin tapauksiin, joissa $g(y)\neq 0$.

Integroimalla saadaan

(2) $\int \frac{y'(x)dx}{g[y(x)]} = \int f(x)dx$.

Tässä käytetään integroinnin sijoitusmenetelmää, joka antaa mahdollisuuden muuttaa integrointi x:n suhteen integroinniksi y:n suhteen, eli päästään muotoon

(3) $\int \frac{dy}{g(y)} = \int f(x)dx$.

Jos integrointi onnistuu, ja funktio y saadaan ratkaistuksi, tuloksena on differentiaaliyhtälön (1) toteuttava funktio y. Integroinnin yhteydessä on huomattava integrointivakion C lisääminen ja se, että tarkastelu rajoittuu x-y -tason alueisiin, joissa $g(y)\neq 0$.

Tässä työssä tarkastellaan alkuarvo-ongelmia, joiden integrointiin voidaan käyttää määrättyä integraalia. Olkoon alkuarvoprobleeman ratkaiseva funktio φ. Tällöin tunnetun alkupisteen $(x_0, \varphi(x_0))$ kautta kulkeva ratkaisukäyrä $(x, \varphi(x))$ toteuttaa yhtälön

(4) $\int_{\varphi(x_0)}^{\varphi(x)} \frac{d\eta}{g(\eta)} = \int_{x_0}^{x} f(\xi)d\xi$.

Jos molemmat integraalit saadaan ratkaistuiksi analyyttisesti ja tästä tuloksesta $\varphi(x)$ (tai x) ratkaistuksi joko yleisessä muodossa tai annetuilla numeerisilla arvoilla, alkuarvoprobleemaa voidaan pitää sovellusten kannalta ratkaistuna. Jos integrointi ei onnistu analyyttisesti, se tehdään numeerisesti. Siten yhtälön (4) esittämä muoto separoituvan differentiaaliyhtälön ratkaisuksi on tässä työssä tehtävien numeeristen tarkastelujen lähtökohtana. Numeerisessa integroinnissa käytetään Gaussin integrointia sekä liitteen 1 oppimateriaalissa että liitteen 2 esimerkkilaskimessa.

2.2.2 Numeerinen ratkaisu

2.2.2.1 Perusratkaisu separoituvalle yhtälölle

Tutkitaan tilannetta, jossa on annettu funktiot $f(x)$ ja $g(y)$ sekä alkuarvot $x_0 = a$ ja $y(a) = b$ sekä kiinteä x. Edelleen $\varphi(x)$ differentiaaliyhtälön alkuarvoprobleeman ratkaisufunktion arvo kohdassa x, ja oletetaan, että differentiaaliyhtälöllä on olemassa yksikäsitteinen ratkaisu. Johdetaan numeerinen ratkaisumenetelmä, joka käyttää hyväksi opiskelijoiden käytössä uusien laskimien myötä olevaa numeerista integrointia. Koska ratkaisu perustuu funktion nollakohdan määrittämiseen, muokataan yhtälön (4) perusteella kahden muuttujan funktio

$$(5)\quad R(x,y) = \int_b^y \frac{d\eta}{g(\eta)} - \int_a^x f(\xi)d\xi\,.$$

Kun x on annettu kiinteä luku, $R(x,y)$ voidaan tulkita yhden muuttujan funktioksi $R(y)$, jonka nollakohta y^* on differentiaaliyhtälön toteuttavan funktion arvo $\varphi(x)$. Siten differentiaaliyhtälön ratkaisufunktion φ arvo $\varphi(x)$ saadaan yhden muuttujan funktion R nollakohdasta. Ratkaistaan nollakohta Newtonin menetelmällä

$$(6)\quad y_{i+1} = y_i - \frac{R(y_i)}{R'(y_i)}\,.$$

Sievennetään tulosta käyttämällä tietoa, että annetulla kiinteällä x tarvittava derivaattafunktio

$$(7)\quad R'(y) = \frac{1}{g(y)}\,.$$

Sijoitetaan Newtonin iterointikaavaan (6) osoittajaksi kaavasta 5 saatava $R(y)$ ja nimittäjäksi $R'(y)$. Tuloksena on Newtonin iterointikaava

$$(8)\quad y_{i+1} = y_i - \frac{\int_b^{y_i} \frac{d\eta}{g(\eta)} - \int_a^x f(\xi)d\xi}{1/g(y_i)} = y_i - \left(\int_b^{y_{ii}} \frac{d\eta}{g(\eta)} - \int_a^x f(\xi)d\xi\right) \cdot g(y_i).$$

Tuloksena saadaan lukujono y_i, joka Newtonin iteroinnin supetessa lähestyy haluttua alkuarvo-ongelman ratkaisufunktion φ arvoa $\varphi(x)$.

Perusratkaisuna saatu iterointikaava (8) on sellaisenaan käytettävissä laskimiin ja helposti ohjelmoitavissa taulukkolaskentaohjelmistoon.

Ratkaisun tarkkuuden osoittaminen perustuu funktion $R(y)$ ominaisuuksiin. $R(y)$ on integraalina jatkuva ja voi siten vaihtaa merkkiä vain, jos välissä on nollakohta. Olkoon integroinnin virheen yläraja e. Tällöin ratkaisu y on välillä $[y_1,y_2]$, kun kaikilla virheen e arvoilla tulo $[R(y_1,) \pm e] \cdot [R(y_2) \pm e] < 0$. Esimerkki tehtävän ratkaisusta perusmenetelmällä käyttäen laskinta on liitteen 2 sivulla18 .

Tehtävän ratkaisu on samanlainen, ratkaistiinpa annetulle x –arvolle funktion $\varphi(x)$ –arvoa vai annetulle funktion φ arvolle φ(x) sitä vastaavaa muuttujan x arvoa. Molemmat johtavat samanlaiseen kahdesta määrätystä integraalista muodostuvaan ratkaisuun.

2.2.2.2. Ratkaisun suppeneminen

Tarkastellaan menetelmän suppenemista. Funktion f nollakohdan määrittämiseen Newtonin iteroinnilla tunnetaan yleisesti suppenemisehto [1, s.106]

$$(9)\quad \left|\frac{ff''}{(f')^2}\right| \le L < 1.$$

Sijoitetaan kaavaan (9) funktion f paikalle $R(y)$. Tuloksena saadaan

$$(10)\quad \left|\left(\int_b^y \frac{d\eta}{g(\eta)} - \int_a^x f(\xi)d\xi\right) \cdot g'(y)\right| \le L < 1.$$

Suppenemisehto on tulo, jonka ensimmäinen tekijä eli integraalien erotus kuvaa Newtonin iteroinnin alkuarvauksen hyvyyttä ja toinen on selkeästi funktiokohtainen. Ongelmallisiksi muodostuvat kaavan 10 perusteella tilanteet, joissa g’ on suuri.

Esimerkkinä suppenemistarkasteluun tarkastellaan yksinkertaista differentiaaliyhtälöä $y' = y$, $y(0) = 1$, kun ratkaistavana arvo $y(x)$ annetulle $x > 0$. Nyt $g'(y) = 1$ ja suppeneminen riippuu vain integraalien erotuksesta, joka on tässä tapauksessa

(11) $|\ln(y) - x| < 1$.

Tuloksesta nähdään menetelmän suppeneminen ainakin, kun $e^{x-1} < y(x) < e^{x+1}$. Koska ratkaisu on $y = e^x$, menetelmä suppenee kohti ratkaisua mielekkäillä alkuarvoilla.

Kaikissa tilanteissa Newtonin iterointi ei suppene. Laskimen numeerista integrointia käytettäessä suppenemisongelmaa voidaan pienentää käyttämällä vaimennettua Newtonin menetelmää tai parantamalla alkuarvausta analyyttisin tai heuristisin menetelmin.

Toinen suppenemisongelma liittyy laskentaan liukuluvuilla. Kaavassa (8) esiintyvä integraalien erotus on ongelmalähde, koska kahden likimain yhtä suuren suureen erotus vähentää merkitsevien numeroiden määrää. (vrt. liite 1, 1.luku). Siten integraalien ääretöntä lähestyvillä arvoilla menetelmä on numeerisesti vaikeuksissa tavalla, jonka kiertäminen on vaikeaa. Vaikka tämä ei ole merkittävästi tullut esille sovelluksissa, sitä on pidettävä yhtenä menetelmän selvistä heikkouksista.

2.2.2.3. Ohjelmoitava, parannettu ratkaisu

Edellisen menetelmän nopeuttamiseksi ja mahdollisten suppenemisongelmien vuoksi menetelmää on kehitetty edelleen.

Laskimella toteutettu perusratkaisu sisältää integroinnin joka askeleella alkuarvosta ratkaisuehdotukseen. Siten sama integrointiväli tulee lasketuksi useita kertoja, ja tämän vuoksi integroinnin nopeutta voidaan lisätä käyttämällä hyväksi tunnettua määrätyn integraalin kaavaa

$$(12)\quad \int_a^{x_i} \frac{d\xi}{g(\xi)} = \int_a^{x_{i-1}} \frac{d\xi}{g(\xi)} + \int_{x_{i-1}}^{x_i} \frac{d\xi}{g(\xi)}.$$

Tällöin laskennan kestäessä tallennetaan edellisen integraalin tulos, ja uuden iterointiaskeleen integrointi aloitetaan edellisen askeleen integroinnin loppupisteestä.

Koska integroinnin numeerinen tarkkuus riippuu integrointiaskeleen pituudesta, määritetään aluksi integroinnin tarkkuuden edellyttämä askelpituus.

Kun integroinnin askelpituus on tunnettu, seuraavaksi arvoksi y_i iteroinnissa valitaan joko Newtonin menetelmän edellyttämä arvo tai integrointiaskeleen

määräämä arvo, jos Newtonin menetelmän perusteella integroinnin askelpituus on liian suuri.

Menetelmä ratkaisee ensimmäisen nopeusongelman: samaa integrointiväliä ei lasketa numeerisesti useita kertoja, kun numeerinen integrointi etenee kohti ratkaisua integrointiaskeleittain.

Suppenemisongelma ratkeaa käytännössä myös tällä menetelmällä, koska integroinnin edetessä askeleittain Newtonin menetelmää käytetään vasta ratkaisun loppuvaiheessa, kun arvo y_i on lähellä ratkaisua ja siten suppenemisehdon toisena tekijänä oleva integraalien erotus on pieni. Puutteeksi jää edellä mainittu liukulukujen vähennyslaskusta johtuva ongelma.

Algoritmin tarkka ohjelmakoodi on liitteen 1 alaliitteessä A2. Lähtötiedot: funktiot *f* ja *g*, alkuarvopiste (*a,b*) ja loppupiste *x* sekä optiona käyttäjän antama alkuarvo y_0 Newtonin iterointia varten. Ratkaisualgoritmin vaiheet ovat:

1. Alkuaskeleen määritys
2. Tasapainon tarkastelu
3. Integrointiaskeleen numeerinen määrittäminen
4. Toistona seuraava algoritmi:
    1. Uuden *y*-arvon määritys miniminä seuraavista:
        a. funktion $1/g(y)$ numeerinen integrointi ja Newtonin askel tai sekanttiaskel
        b. integrointiaskel
        c. tasapainon ylitystä seuraava puolitusarvo
        d. optiona vertailu funktion $g(y)$ nollakohtaan
    2. Lukujen tallennus tulostusta varten
5. Tuloksen esittäminen
6. Optiona kuvaajan tulostus

Vaihe 1 perustuu joko käyttäjän antamaan alkuarvaukseen y_0 tai algoritmin laskemaan alkuarvoon

$$(13)\ y_0 = b + \int_a^x f(\xi)d\xi \cdot g(b).$$

Vaiheena 2 tarkastellaan ylittääkö askel ”tasapainopistettä” eli kohtaa $g(y) = 0$. Tämä tapahtuu karkeasti tutkimalla funktion $g(y)$ merkkiä. Merkkitarkastelu ei kuitenkaan toimi aina, vaan ongelmia esiintyy esimerkiksi tilanteissa, joissa kaikilla *y* pätee $g(y) \geq 0$. Siten voidaan helposti konstruoida funktioita, joilla menetelmä ei toimi, jos käyttäjä ei aseta mielekästä alkuarvoa. Nämä funktiot ovat valitettavasti usein vaikeita myös muillekin numeerisille menetelmille kuten Rungen-Kuttan menetelmälle. Esimerkki tällaisesta on liitteen 1 luvun 3.2.3 esimerkkinä 6.

Integrointiaskel määritellään Gaussin integraalille, jota pidetään optimaalisena polynomisovituksista [41]. Tässä olisi mahdollista käyttää myös mitä tahansa muuta integrointimenetelmää ja sen virhetarkastelua.

Kun alkuvaiheet on saatu tehdyksi, alkaa varsinainen iterointi eli *y*-akselin suuntaan integrointi. Valitusta alkuarvosta lasketaan seuraava alkuarvo Newtonin menetelmällä. Jos näin saatu uusi *y*:n arvo poikkeaa edellisestä *y*:n arvosta enemmän kuin integroinnin tarkkuus antaa myöten, askel lyhennetään integrointiaskeleen mittaiseksi. Jos tämä askel ylittää funktion *g(y)* nollakohdan, askelta lyhennetään kunnes nollakohtaa ei ylitetä. Näin saatu uusi *y* sijoitetaan silmukkaan seuraavan *y*-arvon määrittämiseksi.

Ohjelmaa voidaan nopeuttaa hiukan, jos ennalta tiedetään integrointiaskeleita tarvittavan runsaasti. Tällöin laskenta voi edetä integrointiaskelin kunnes funktio *R(y)* vaihtaa merkkiä ja siten Newtonin iterointiaskeleen kuluttama aika jää pois. Tämän jälkeen Newtonin iteroinnin alkuarvo ratkaistaan sekanttimenetelmällä kahden viimeisen arvon välistä, ja tulos saadaan yleensä yhdellä jatkoaskeleella.

2.2.2.4 Vertailu perinteisiin menetelmiin

Menetelmällä voidaan ratkaista numeerisesti differentiaaliyhtälöt käyttämällä laskimissa yleisesti olevaa integrointitoimintoa. Siten se laajentaa monien laskimen käyttöaluetta merkittävästi.

Ohjelmoitava ratkaisu kilpailee muiden ohjelmoitavien ratkaisujen kanssa. Koska esitetty algoritmi on helposti ohjelmoitavissa (Liite 1 alaliite A2), se on näiltä osin kilpailukykyinen muiden yleisten ratkaisujen kanssa.

Sekä numeeristen menetelmien että insinööritieteistäni tuntemani palotekniikan matemaattisten menetelmien oppi- ja käsikirjat sisältävät yleensä Eulerin ja Rungen - Kuttan -menetelmät esim. [10, s.223-225]. Oppikirjoissa esim. [18, s. 947-961], [36, s. 712-751] & [16, s. 486-523] esitetyt muut menetelmät ovat liian vaikeita opiskelijoiden matemaattisen lähtötason vuoksi tai niiden ohjelmointi on vaikeampaa.

Siten ensisijainen kilpaileva menetelmä on Rungen – Kuttan menetelmä, joka on myös mm. Matlabin [31] valmistoimintona. Kun differentiaaliyhtälö ratkaistaan perinteisesti lähtien muodosta $y' = f(x,y)$, saadaan derivaatalle arvo, jonka oletetaan pysyvän vakiona ensimmäisen askeleen ensimmäisen osan ajan. Kun tätä arvoa Rungen-Kuttan menetelmässä myöhempien arvojen avulla täydennetään ja korjaillaan, funktion *y* arvolle saatavan approksimaation virhe on tunnetusti verrannollinen askelpituuden neljänteen potenssiin. Numeerisilla integrointimenetelmillä voidaan helposti päästä suurempiin tarkkuuksiin [1, s.429]. Newtonin menetelmä tarjoaa mahdollisuuden ratkaista integroinnin yläraja, joten

edellä mainitulla menettelyllä saadaan perinteisiin menetelmiin verrattuna kilpailukykyisiä menetelmiä separoituville yhtälöille.

Algoritmin tehokkuudella (efficiency) tarkoitetaan menetelmän vaatimaa tietokoneen muistitilaa tai laskenta-aikaa eli nopeutta [6, s.10]. Muistitilan käytön osalta sekä Rungen - Kuttan -menetelmä että esitetty separoituvan yhtälön ratkaisu eivät ole ongelmallisia, jolloin tehokkuuden ratkaisee laskenta-aika.

Koska esitetty separoituvan differentiaaliyhtälön ratkaisumenetelmä käyttää iteroinnin lopussa Newtonin menetelmää, jonka suppeneminen vaihtelee eri alkuarvauksilla, ei teoreettista varmuutta tehokkaamman integrointimenetelmän tarjoamasta suuremmasta nopeudesta voi yleisesti esittää.

Laskenta-aikaa pitää siis arvioida tilannekohtaisesti. Tietokoneelle ohjelmoitujen algoritmien nopeuden ratkaisee viime kädessä sekuntikello, vaikka analyyttisin ja ennakkolaskentaan perustuvin menetelmin saadaankin arvioita eri algoritmien potentiaalisesta tehokkuudesta. Numeerisia vertailuja on julkaistu tekijän julkaisuissa [20] ja [21] ja lisävertailut on esitetty liitteen 1 luvun 3.2.3 esimerkeissä ja lähdekoodit liitteen 1 alaliitteinä A1-A6.

Tuloksena voidaan todeta, että esitetty numeeriseen integrointiin perustuva menetelmä on tehokkaampi, koska perinteinen menetelmä vaati moninkertaisen ajan.

2.3. Joitakin toisen kertaluvun differentiaaliyhtälöitä

2.3.1 Johdanto

Tämän työn matemaattisen osan painopiste on esitetyissä ensimmäisen kertaluvun differentiaaliyhtälön ratkaisuissa. Koska integroimalla saadaan ratkottua myös toisen kertaluvun yhtälöitä, esitetään ratkaisut joihinkin niistä.

Koska toisen kertaluvun differentiaaliyhtälöillä on suurta fysikaalista merkitystä, niiden opettaminen on mielekästä ja siksi ne käsitellään myös liitteen 1 oppimateriaalissa, vaikka esimerkkien valossa perinteisiin numeerisiin menetelmiin verrattuna ei saatu vastaavaa yksikäsitteistä nopeushyötyä kuin ensimmäisen kertaluvun yhtälöissä. Toisen kertaluvun differentiaaliyhtälöt voidaan kirjoittaa muodossa $y'' = f(y',y,x)$.

Tässä työssä ja sen oppimateriaalina käytetyssä liitteessä rajoitutaan toisen kertaluvun differentiaaliyhtälöissä Rungen – Kuttan menetelmään ja differentiaaliyhtälöihin, joiden ratkaisu voidaan helposti kirjoittaa integraalimuodossa. Ratkaisutapa esitetään lyhyesti kuhunkin alkuarvo-ongelmaan erikseen. Tarkasteltavissa toisen kertaluvun differentiaaliyhtälöissä alkutilasuureita on kolme:

a, y(a)=b ja *y'(a)=c* ja annettuna yhden niistä lopputila, yleensä kiinteä *x*. Ratkaistavaksi jää kaksi muuta, yleensä arvot *y(x)* ja *y'(x)*.

Tässä luvussa tarkasteltavat differentiaaliyhtälöt voidaan ryhmitellä kahteen osaan:

A Tapaukset, joissa *f* riippuu vain yhdestä argumentista:
y'' = f(x), y'' = f(y) ja *y'' = f(y').*

B Separoituvat parit, joissa funktio *f* voidaan esittää tulona:
y'' = g(y') · h(y) ja *y'' = g(y') · h(x).*

Tuloa *y'' = g(y) · h(x)* ei käsitellä, koska sitä ei yleensä saa muotoon, jossa ratkaisu voitaisiin tehdä numeerisilla integroinneilla.

2.3.2 Yhden argumentin tapaukset

2.3.2.1 Yhtälö y'' = f(x)

Tehtävän ratkaisu on suoraviivainen kaksoisintegraali. Tarkastellaan differentiaaliyhtälöä *y''= f(x)* alkuarvoilla *y(a) = b* ja *y'(a) = c*. Olkoon tämän alkuarvo-ongelman ratkaisuna *y = φ(x)* ja annettuna muuttujalle loppuarvo *x*. Kysytään arvoja *φ(x)* ja *φ'(x)*. Annetulle arvolle *x* saadaan sitä vastaava funktion arvo *φ'(x)* suoraan integroimalla funktio *f*, ja tuloksena saadaan

$$(14)\ \varphi'(x) - c = \int_a^x f(\xi)d\xi\ .$$

Lopputulos edellyttää kaksoisintegraalin

$$(15)\ \varphi(x) = \int_a^x \varphi'(\xi)d\xi = \int_a^x (\int_a^\xi f(\eta)d\eta + c)d\xi$$

laskemista.

Esimerkki kaksoisintegraalin numeerisesta ratkaisuohjelmasta on liitteen 1 alaliitteenä A9. Jos funktioiden arvot *φ(x)* tai *φ'(x)* on annettu, vastaava *x* voidaan iteroida esimerkiksi Newtonin menetelmällä kuten ensimmäisen kertaluvun separoituvassa yhtälössä.

2.3.2.2 Ratkaisu yhtälölle y'' = f(y')

Tarkastellaan differentiaaliyhtälöä *y'' = f(y')* alkuarvoilla *y(a) = b* ja *y'(a) = c*. Olkoon tämän alkuarvoprobleeman ratkaisuna *y = φ(x)* ja annettuna muuttujalle loppuarvo *x*. Kysytään arvoja *φ(x)* ja *φ'(x)*. Differentiaaliyhtälö on derivaatan *y'* suhteen separoituva, eli siitä saadaan suoraan integraalimuodossa oleva ratkaisu

$$(16)\ \int_a^x \frac{\varphi''(\xi)d\xi}{f(\varphi'(\xi))} = \int_c^{\varphi'(x)} \frac{d\eta}{f(\eta)} = x\text{-}a\,.$$

Tästä ratkeaa *φ'(x)*, jonka integraali on suoraan *φ(x)*.

Numeerisen ratkaisun perusaskel muodostuu funktion *φ'* arvojen määrityksestä Gaussin integroinnin edellyttämissä pisteissä x_i, jotka saadaan numeerisesti esimerkiksi separoituvan yhtälön ratkaisumenetelmällä; tuloksena arvot $\varphi'(x_i)$. Tämän jälkeen arvo *φ(x)* ratkeaa suoraan integroimalla *φ'*. Tarvittaessa integrointiaskeleita otetaan useita halutun tarkkuuden saavuttamiseksi arvolle *φ(x)*. Tilanne on erikoistapaus tilanteesta *y'' = f(y') f(x)*.

Ratkaisualgoritmi liitteen 1 alaliitteenä A9.

2.3.2.3 Yhtälö y'' = f(y)

Yhtälön yksi analyyttinen ratkaisu lähtee liikkeelle siitä, että yhtälön molemmat puolet kerrotaan funktiolla *y'*. Erikseen pitää tarkastella ratkaisut, joissa missä tahansa vaiheessa ratkaisua *y'(x) = 0*. Yhtälö

(17) *y''(x) y'(x) = f(y(x)) y'(x)*

voidaan integroida puolittain, jolloin alkuarvoprobleeman ratkaisufunktiolle *y = φ(x)* saadaan

$$(18)\ \int_a^x \varphi'(\xi)\varphi''(\xi)d\xi = \int_a^x f(\varphi(\xi))\varphi'(\xi)d\xi\,.$$

Muuttujan vaihdon *dφ' = φ''dx* ja *dφ = φ'dx* tuloksena saadaan

$$(19)\ \tfrac{1}{2}\varphi'(x)^2 - \tfrac{1}{2}c^2 = \int_b^{\varphi(x)} f(\eta)d\eta\,.$$

Yhtälöstä ratkeaa $\varphi'(x)$ ja tulos

$$(20)\ \varphi'(x) = \pm\sqrt{c^2 + 2\int_b^{\varphi(x)} f(\eta)d\eta}$$

on separoituva differentiaaliyhtälö. Siten tulos $y^*=\varphi(x)$ saadaan suoraan numeerisesta ratkaisusta yhtälölle

$$(21)\ \int_b^{y^*} \frac{d\mu}{\pm\sqrt{c^2 + 2\int_b^{\mu} f(\eta)d\eta}} = x - a\ .$$

Tämän yhtälön erikoistapauksia voidaan ratkaista kuhunkin tilanteeseen räätälöidyillä ohjelmilla, mutta yleisen perinteisiä ratkaisutapoja nopeamman ratkaisumenetelmän ohjelmointi on haastavaa, koska se edellyttää esimerkiksi funktion derivaatan merkkitarkastelua ongelmakohdan $\varphi' = 0$ käsittelyssä. Tällöin ratkaisu pitää ”viipaloida” osiin, joissa $\varphi' \neq 0$, ja kussakin palassa valita φ':n alkuarvon mukainen merkki. Jos φ' on nolla, integroinnin merkki voidaan päätellä φ'':n avulla. Jos φ'' on positiivinen, φ' on nollakohdan jälkeen positiivinen ja jos negatiivinen, niin negatiivinen. Jos $\varphi' = 0$ ja $\varphi'' = 0$, kyseessä on tasapainopiste.

Esimerkki ratkaisusta liitteessä 1, luku 4 esimerkit 1 ja 2 ja ratkaisun tekevä ohjelmakoodi alaliitteenä A8.

2.3.3 Tapaukset y'' = f(*) g(*)

2.3.3.1 Yhtälö y'' = g(y') f(x)

Tilanteessa ratkaisufunktio φ on funktion φ' suhteen separoituva differentiaaliyhtälö, jonka ratkaisu on

$$(22)\ \int_a^x \frac{\varphi''(\xi)d\xi}{g(\varphi'(\xi))} = \int_c^{\varphi'(x)} \frac{d\eta}{g(\eta)} = \int_a^x f(\xi)d\xi\ .$$

Separoituvissa yhtälöissä aiemman valossa numeeriseen integrointiin perustuva algoritmi on nopeampi, joten tästä yhtälöstä saadaan joko analyyttisesti tai numeerisesti tuloksena parit x ja $\varphi'(x)$. Näistä edelleen saadaan suoraan numeerisella integroinnilla $\varphi(x)$. Siten numeerinen ratkaisu tapahtuu kaksoisintegraalin avulla.

2.3.3.2 Ratkaisumenetelmä yhtälölle y'' = g(y') · f(y)

Ratkaisu muistuttaa aiempia, koska siinä yhtälö kirjoitetaan muotoon

$$(23)\ \int_a^x \frac{\varphi'\varphi''\,d\xi}{g(\varphi')} = \int_a^x f(\varphi)\varphi'\,d\xi\,.$$

Tämä sievenee kahdeksi integraaliksi, joista saadaan ratkaistuksi funktion *φ'* lauseke *y*:n funktiona esimerkiksi separoituvan yhtälön numeerisella menetelmällä

$$(24)\ \int_c^{\varphi'(x)} \frac{\eta\,d\eta}{g(\eta)} = \int_b^{\varphi(x)} f(\eta)\,d\eta\,.$$

Olkoon ratkaistu $\varphi'(x) = G(\varphi(x))$. Siten alkuperäisen yhtälön ratkaisu saadaan yhtälöstä

$$(25)\ \int_b^{\varphi(x)} \frac{d\eta}{G(\eta)} = x - a\,.$$

Tämä yhtälö voidaan ratkaista numeeriseen integrointiin perustuvalla algoritmilla, jossa perusaskeleena ratkaistaan funktio *φ'* funktion *φ* suhteen esim. Gaussin integroinnin edellyttämissä integrointipisteissä, ja tämän jälkeen ratkaistaan suoraan uusi integraali numeerisesti. Seuraava iteroinnin arvo voidaan ratkaista jälleen Newtonin menetelmällä tai edetä integrointiaskelia, kunnes erotus *x* - *a* on saavutettu. Tämän jälkeen tarkka ratkaisu saadaan esimerkiksi sekanttimenetelmällä.

2.3.4 Yhteenveto toisen kertaluvun menetelmien käsittelystä

Kokonaisuutena toisen kertaluvun differentiaaliyhtälöiden numeeriseen ratkaisuun ei ole esitettävissä mitään selkeästi perinteisiä menetelmiä parempaa uutta algoritmia. Esitettyjen integrointiin perustuvien menetelmien rajoitusten osoittajana ne kuitenkin puoltavat paikkaansa myös osana tätä tutkimusta.

3. Differentiaaliyhtälöiden ratkaisemisen opettamisesta

3.1. Opetuksen tavoitteen asettelu

3.1.1 Teollisuuden vaatimukset

3.1.1.1 Johdanto

Omat kokemukseni työnantajana ja palofysikaalisten laskelmien teettäjänä antavat selkeän käsityksen numeriikan osaamistarpeesta insinööritoimiston näkökulmasta. Siten opetussuunnitelman sisältö perustuu omakohtaisesti tunnistettuun teolliseen tarpeeseen. Tämä esitys noudattaa pääosin liitteen 1 oppimateriaalin asiajärjestystä ja kuvaa kuhunkin opetettavaan asiaan liittyviä insinööritoiminnassa tehtyjä havaintoja.

Paloinsinöörikoulutuksesta on tehty kansanvälinen selvitys [30]. Selvityksen mukaan osattavia tausta-asioita olivat muun muassa virtausmekaniikkaa, aineen- ja lämmönsiirtoa, termodynamiikkaa ja rakenteiden mekaniikkaa. Varsinaiset palotekniikan oppisisällöt kattaisivat palon perusteet, huonepalon dynamiikan, aktiivisen palontorjunnan, passiivisen palontorjunnan ja ihmisten toiminnan tulipaloissa. Näistä kaikki edellyttävät differentiaaliyhtälöiden ratkaisua lukuun ottamatta palontorjunnan teknisten laitteiden yksityiskohtia.

Akateemisen paloinsinöörin opintovaatimukset kertovat tulevista työtehtävistä insinööritoimistoympäristössä. Yrittäjän kannalta katsottuna töihin tulevan opiskelijan ei tarvitse ymmärtää kunkin laskentatilanteen syvällistä fysikaalista taustaa, vaan esimerkiksi osata ohjelmoida annettu laskutapa numeerisesti virheettömästi ja riittävän nopealla algoritmilla, osata tehdä simulointiohjelmalla sarja ajoja eri lähtötietoarvoilla ja käsitellä tulokset, sekä sovittaa ja syöttää kokeellinen data halutussa muodossa laskentamalliin.

Differentiaaliyhtälöiden analyyttisissä ratkaisumenetelmissä joudutaan usein tekemään sijoituksia ja valitsemaan sopivia apumuuttujia, jotta tehtävä yksinkertaistuisi joko analyyttisen ratkaisun löytämiseksi tai laskennallisesti paremman kaavan saamiseksi. Päivölän ja Helsingin matematiikkalukion opiskelijat voivat käyttää apuna omien taitojensa lisäsi myös symbolisesti laskevia ohjelmistoja,

jotka mahdollistavat tehtävien teknisen sieventämisen moniin eri muotoihin. Siten mahdolliset analyyttiset integroinnit ovat ratkaistavissa ja numeerisen ratkaisun ohjelmointitaito on korostuu: nopea ratkaisu pitää voida valita monista saman tehtävän eri esitystavoista. Kokemusteni perusteella edelleen kaiken perustana ovat yhtälöjen analyyttisen käsittelyn harjoittelun yhteydessä syntyneet mielikuvat ja perustaidot. Kun ne yhdistetään matemaattisten ohjelmistojen mahdollisuuksiin, nuoret ohjelmointitaitoiset opiskelijat pystyvät teollisten ongelmien monipuolisiin tarkasteluihin.

Vaikka opettavia asioita perusteleva esimerkkiaineisto on ensisijaisesti palo- ja turvallisuusteknistä, keskustelut mm. Nokian tutkimuslaitoksessa (NRC Toijala) työskentelevien opiskelijoiden esimiesten Matti Karlssonin (1997-2000) ja Kari Lahdensuon (2001-) kanssa eivät ole osoittaneet ristiriitaa valittujen numeerisia menetelmiä koskevien esimerkkien kanssa.

3.1.1.2 Perusasioista

Liitteen 1 oppimateriaalin ensimmäinen luku käsittelee liukulukuesitystä ja laskutoimitusten virhettä. Laskinten ja tietokoneiden laskennan kannalta merkittävää on tuntea käytetty lukujärjestelmä eli liukulukuesitys ja sen rajoitukset. Esimerkiksi ohjelmoinnissa ei lopetusehtona useinkaan voi käyttää yhtäsuuruutta nollan kanssa, koska liukuluvut eivät välttämättä saavuta nollaa, vaikka analyyttinen ratkaisu sellaisen vaikutelman antaisikin. Laskukaavojen järkevään ohjelmointiin riittää liukulukujen perusteiden tuntemus, joten mm. lähellä nollaa olevien epästandardien lukujen käsittely on rajattu pois.

Laskinta käytetään sekä jatkuvasti osana matematiikan ja fysiikan opintoja että myös ajoittain työtehtävissä. Siten sen toiminnan ymmärtämiseen ja siten lukuesitykseen panostaminen on mielekästä ja tärkeää sovellusten kannalta. Havaintojeni mukaan juuri kukaan oppilaista ei tiedosta laskimen laskevan oleellisesti ottaen kaiken likimääräisillä luvuilla, vaan esimerkiksi laskimessa näkyvää lukua 1/5 = 0,2 pidetään tarkkana, vaikka sen liukulukuesitys on likimain yhtä väärä kuin ruudulla kymmennumeroisena desimaalina näkyvän luvun 1/3 = 0,333… liukulukuesitys. Siten laskenta kaksijärjestelmän luvuilla on syytä tiedostaa ja kaksijärjestelmän opettaminen on luonnollista heti numeriikan opetuksen alussa. Koska laskenta tapahtuu suoraan binääriluvuilla, pois on jätetty algoritmit (mm. Hornerin kaava), joilla ihmiset voivat nopeasti muuttaa kymmenjärjestelmän lukuja binääriluvuiksi tai päinvastoin.

Esimerkkinä laskutoimitusten numeerisesta merkityksestä toimivat toisen asteen yhtälöt (liite 1, s.68 ja liite 2 luku ”Toisen asteen yhtälö ja laskimen laskentatarkkuus”). Numeerisesti oikein muotoiltua ratkaisukaavaa edellyttäviä toisen asteen yhtälöitä on esiintynyt mm. sensorien havaitsemisanalyyseissä, joissa antennikorkeus ja maan säde ovat merkittävästi eri suuruusluokkaa. Tämä tuli esille mm. raporttia [23] laadittaessa. Myös kemian tasapainolaskuissa voi esiintyä

samantapaisia tilanteita, kun tasapainoreaktiot tai liukoisuustulo johtavat toisen asteen yhtälöön.

3.1.1.3 Yhtälön ratkaisu

Yhtälön ratkaiseminen lienee itsestään selvä osa mitä moninaisimpien sovellusten käsittelyä. Koska yksittäisen yhtälön ratkaiseminen onnistuu helposti muun muassa laskimen kuvaajanpiirtotoiminnolla tai laskimen valmistoiminnoilla eli niin sanotulla ”solverilla”, laskennan tarkkuudesta ja vastauksen oikeellisuuden osoittamisesta on tullut entistä merkittävämpää. Käytännössä tämä edellyttää yhtälön muuttamista funktioksi ja funktioiden ominaisuuksien perusteella tehtävää tarkastelua.

Laskimien valmistoiminnot käyttävät Newtonin menetelmää [37] [4] [40], minkä vuoksi sen tunteminen helpottaa ratkaisua ja selittää eri alkuarvoilla saatavat erilaiset tulokset. Tärkeimmät ratkaisumenetelmät täytyy myös tuntea, jos opiskelijalla halutaan olevan minkäänlaisia kykyjä kirjoittaa ratkaisuohjelmia eri matemaattisiin ongelmiin. Samoin differentiaaliyhtälöiden numeerisessa ratkaisussa tarvittavia menetelmiä pohjustetaan Newtonin menetelmällä, ja kiintopistemenetelmän suppenemistarkastelu opettaa eri menetelmien hyvyydestä. Koska Newton ja kiintopistemenetelmä ovat suppenemisen kannalta hankalia, myös puolitushakua ja sekanttimenetelmää tarvitaan useissa käytännön tilanteissa.

Erityisesti omaa numeerista ohjelmointitaitoa edellyttäviä tehtäviä esiintyy todennäköisyysjakaumien yhteydessä, kun kokeellinen jakaumatieto on osana yhtälön ratkaisua. Todennäköisyystehtävään liittyvä numeerinen laskutehtävä liittyi mm. ikkunan kautta leviävän palon tarkasteluun [24]. Seinämän lämpenemistarkastelussa sitä käytettiin osana laskentaohjelmaa [25].

3.1.1.4 Numeerinen integrointi

Määrätyt integraalit ovat yleisiä fysikaalisissa sovelluksissa. Niiden numeerinen laskenta on merkittävää ja numeerisen integroinnin toiminto on ollut saatavissa halpoihin laskimiin jo vuodesta 1987 lähtien (Ti-86). Laskentavälineiden käyttämät yleiset menetelmät ovat Simpson ja Gauss, joten niiden opettaminen on luonnollista. Pinta-alan arviointiin liittyvät yksinkertaiset menetelmät esitellään johdantona näihin. Gaussin integroinnin integrointipisteiden johtaminen tehdään kolmannen asteen polynomiin asti ja ohitetaan yleisessä muodossaan.

Adaptiiviset integrointimenetelmät ovat ohjelmoitavuudeltaan selkeästi kiinteää askellusta tehokkaampia, joten erityisesti niiden opettaminen virhetarkastelumenetelmiksi on järkevää. Adaptiiviset algoritmit tarvitsevat pohjakseen askelpituuteen perustuvaa virheanalyysiä, joka pitää myös esitellä. Askelpituuteen liittyvien integroinnin virhekaavojen johtamiseen ei paneuduta

perusteellisesti, koska adaptiivisten menetelmien virhetarkastelussa oleellista on menetelmän asteluku, eivät yksittäiset kertoimet.

Laskimeen ei nytkään voi luottaa sellaisenaan. Esimerkkinä sudenkuoppaongelmasta voidaan esittää mustan kappaleen lämpösäteilyn tarkastelu. Jos sensori havaitsee tietyn aallonpituusalueen, tällä alueella säteilevä energia saadaan määrättynä integraalina [17]. Koska nanometriluokkaa olevien suureiden integrointi voi johtaa virheelliseen tulokseen, esimerkiksi Lukion laskinopin kirjassa (liite 2) käsitellään integrointivälin pidentäminen. Samoin liian laaja tarkastelualue voi johtaa siihen, että tarkasteluun ei osu lainkaan nollasta poikkeavia laskentapisteitä. Esitetyn sensoritekniikkaan liittyvän fysikaalisen esimerkin valossa adaptiivisten menetelmien puutteiden hallinta on tarpeellista myös laskimen käytön ymmärtämiseksi.

Lisäksi integroinnin osalta käytetyt menetelmät pohjustavat differentiaaliyhtälöiden opetusta.

3.1.1.5 Differentiaaliyhtälöistä

Differentiaaliyhtälöt ovat hyvin yleisiä insinööritieteissä. Niitä esiintyy kinemaattisissa tarkasteluissa, lämpöopissa ja yleensä kaikissa järjestelmissä, missä suureiden muutokset riippuvat toisistaan. Koska lukuisat fysikaaliset ja teolliset sovellukset mallinnetaan toisen kertaluvun differentiaaliyhtälöinä, niidenkin käsittely on mielekästä ja järkevää osana opetusta. Esimerkiksi paloanalyyseissä usein toistuva tehtävä on arvioida erilaisten kohteiden lämpenemistä ajan funktiona. Yksinkertaistettu esimerkki on tekijän osuudessa Tammen Pyramidi -sarjan numeeristen menetelmien oppikirjaan [9] sivuilla 98-99.

Differentiaaliyhtälöiden numeerisista ratkaisumenetelmistä Eulerin menetelmää käytetään johdantona differentiaaliyhtälöihin. Sen numeerisesta tehottomuudesta huolimatta sitä käytetään teollisissa sovelluksissa [39] ja taistelumalleissa, joissa aika-askeleittain etenevä mallinnus on itse asiassa probleeman ratkaisua Eulerin menetelmällä. [14] Asiaa on käsitellään myös Lauri Kankaan kirjoittamishetkellä valmistumisvaiheessa olevassa diplomityössä [15, luku 4].

Eulerin menetelmää tehokkaampi Rungen-Kuttan menetelmä on käytössä laajalti, ja muun muassa Matlab –ohjelmisto ja Ti-89 laskin käyttävät adaptiivista Rungen-Kuttan menetelmää differentiaaliyhtälön ratkaisumenetelmänään. Siten ajassa etenevien prosessien laskennassa sekä Eulerin että Rungen –Kuttan menetelmän tunteminen on tärkeää.

Valmisohjelmistojen osalta erityisesti laskennan tarkkuuden arviointi integroinnin tapaan adaptiivisilla laskennan tuloksia hyödyntävillä algoritmeilla on tärkeää, koska käyttäjä voi valmisohjelmia käyttäessään usein itse säätää vain numeerisia syöttötietoja. Tulosten numeerisen virheen perusteiden ymmärtäminen helpottaa

esimerkiksi aika-askeleen valintaa, kun aluksi tehdään kokeiluja kyseisen tilanteen laskennasta eri aika-askeleilla.

Koska monissa laskimissa ei ole kuitenkaan valmisohjelmistona differentiaaliyhtälön ratkaisuohjelmaa, vaan ainoastaan numeerinen integrointi, integrointiin perustuvien menetelmien opetus on myös mielekästä.

Differentiaaliyhtälöiden opetuksessa havainnollistamiseen voi käyttää numeerisesti saatavia ratkaisukäyriä. Tämän vuoksi numeerisen ratkaisun opettaminen heti aluksi tukee analyyttisten ratkaisujen opetusta, ja on perusteluna asioiden järjestykselle oppimateriaalissa. Koska laskentaohjelmien hyvyyttä vertaillaan usein nimenomaan laskennan nopeudella [8], myös tämän lähestymistavan opettaminen on tärkeää heti alusta alkaen.

Ratkaisujen olemassaolo- ja yksikäsitteisyyslause on sekä teoreettisesti että käytännön kannalta merkittävä. Muun muassa palosimuloinneissa on havaittu nimenomaan tasapainotilojen ympärillä tapahtuvan värähtelyn ja ratkaisukäyrältä toiselle hyppäämisen olevan ongelma. Tämä oli havaittavissa Suuren kauppahallin paloselvityksessä [26]. Alustavissa laskentatilanteissa tällainen hyppäys laskentakäyrältä toiselle johti pahimmillaan lämpötilan värähtelyyn ja lopulta putoamiseen yli 500 K lämpötilasta alle 200 K kylmyyteen. Tällaiset numeerisesta ”resonanssista” johtuvat tilanteet tulivat usein vastaan, kun aika-askel ja seinämän lämpenemisessä käytetty seinän jaottelu laskennallisiin viipaleisiin eivät olleet sopusoinnussa. Siten simuloinneissa esiintyi myös aluksi yllättäviltä tuntuneita ongelmia, kun yhtä suuretta muutettiin: saadussa simulointitulosjoukossa keskellä oleva ratkaisukäyrä saattoi olla järjetön, vaikka viereiset toimivat näennäisesti hyvin. Syy oli värähtelyn kaoottinen käyttäytyminen alun perin epäsuhdassa olleilla ajan ja seinäpaksuuden lähtöarvoilla, mikä tuli esille palotehon joillain arvoilla, mutta ei kaikilla. [26]

Kanadalaisen Firecam -ohjelmiston [7] evaluoinnin [2] yhteydessä havaittiin vastaavia ilmiöitä, ja ilman virtaus pienissä huoneissa (eteisissä) johti silloisen ohjelmaversion samanlaiseen tasapainon ylittämiseen ja värähtelyyn, jonka seurauksena oli väärä tulos.

Ongelmaa voidaan korjata aika-askeleen sisään ohjelmoitavalla tasapainotarkastelulla. [27] Vaikka tasapainotarkastelu tehtäisiin jokaisella aika-askeleella, lopputuloksena saatava ohjelmisto voi olla alkuperäistä nopeampi, kun askelpituutta voidaan vastaavasti kasvattaa ilman tasapainojen ylityksistä johtuvia ”värähtelyongelmia”.

Laajojen simulointiohjelmistojen yksityiskohtien käsittely ei ole ajankäytön kannalta mahdollista lyhyellä numeriikan perusteiden kurssilla, mutta niiden kriittisen käytön mahdollistavat perusvalmiudet tulee opettaa. Siten yksinkertaiset ohjelmistoesimerkit opettavat todellisissa numeerisissa ongelmissa tarvittavia taitoja ja idean siitä, että valmiita ohjelmakoodeja voi täydentää tilanteen mukaan muun

muassa tasapainon huomion ottavilla lisäehdoilla. Tämä osaltaan motivoi differentiaaliyhtälöiden olemassaolo- ja yksikäsitteisyyslauseen tapaisten asioiden opiskelua.

3.1.1.6 Laskin, taulukkolaskenta, Matlab ja ohjelmointikielet

Jos tavoitteena on ratkaista numeerisia tehtäviä siten, että tuloksena saatavia arvoja käytetään todellisten teknis-taloudellisten ratkaisujen tekoon ja ilmiöiden mallintamiseen, käytettävät työkalut ja niiden toiminnan tunteminen ovat keskeisiä.

Kaikilla lukiotason opiskelijoilla on funktiolaskin, jonka perustoimintoihin kuuluu yleensä numeerinen integrointi, ja vain muutamalta matematiikkalinjalaiselta puuttui ohjelmoitava graafinen laskin. Siten opetuksen lähtökohdaksi voitiin olettaa mahdollisuus numeeriseen integrointiin ja käytännössä myös ohjelmointiominaisuus.

Taulukkolaskentaan tehdyillä sovelluspohjilla on samoin laaja käyttäjäpiiri, muun muassa kehittämäni joukon taisteluarvon laskentamalli haluttiin nimenomaan taulukkolaskentasovellukseksi. [28] Päivölän opetuksessa taulukkolaskenta oli ATK-luokan myötä käytössä ja valtaosalla opiskelijoista oli taulukkolaskentaohjelmisto myös omilla tietokoneillaan. Taulukkolaskentaohjelmat ovat vakiintuneet yleiskäyttöisiksi osiksi ”toimisto-ohjelmistopaketteja” ja oppilaat voivat saada niitä käytännössä ilmaiseksi (esim. OpenOffice.org). Siten eri menetelmien taulukkolaskentaan soveltuvien ratkaisutapojen kuvaaminen on mielekästä ja muun muassa dosentti Matti Lehtinen on tehnyt aiheesta opiskelijoiden lähtötason kannalta mielekkään oppikirjan [29].

Matlab on matemaattinen laajalti käytetty ohjelmointikieli, jolla tehtävien ratkaisu on nopeaa ja helppoa. Sillä voi hyvin usein tehdä laskennan ”demoversion”, jonka perusteella lopullinen ohjelmointikielellä tehtävä ohjelmisto tehdään. Tästä esimerkkinä on mm. Teknillisen korkeakoulun ja Puolustusvoimien teknillisen tutkimuslaitoksen voimasuhdeanalyysiprojekti [42]. Päivölän kansanopistolla ja Helsingin matematiikkalukiolla on Matlab- lisenssejä, ja opiskelijoilla on vastaavanlaisia Octave ja Scilab –ohjelmistoja käytössään. Siten tämän ohjelmiston ja ohjelmointikielen käyttö esimerkkinä on mahdollista, ja Matlab –tyyppisten ohjelmistojen laajan teollisen ja yliopistokäytön vuoksi ohjelmiston perustoimintojen ja ohjelmoitavuuden opetus on muutenkin hyödyllistä.

Kurssin opetettavasta materiaalista on rajattu varsinaisilla ohjelmointikielillä tehtävät esimerkit. Kuitenkin laskimen ohjelmointia pidetään suotavana ja halukkaat voivat kirjoittaa algoritmit haluamallaan ohjelmointikielellä, vaikka oppimateriaali ei näitä asioita käsittelekään.

3.1.2 Yhteenveto opetettavista asioista

Toteuttamieni teknisluonnontieteellisten ja teollisten projektien perusteella opetuksen tavoitteena on, että kurssin jälkeen opiskelijat:

- Osaavat ohjelmoida tuntemansa ratkaisukaavat numeerisesti mielekkäästi.
- Pystyvät ratkaisemaan sovelluksissa esiintyvät yhtälöt ja yhtälöryhmät numeerisesti ja osoittamaan ratkaisun tarkkuuden ja yksikäsitteisyyden.
- Pystyvät ratkaisemaan separoituvat differentiaaliyhtälöt numeerisesti käytössään olevilla välineillä (sekä laskimet että tietokoneiden valmisohjelmistot).
- Pystyvät ohjelmoimaan numeeriset ratkaisut ensimmäisen ja toisen kertaluvun differentiaaliyhtälöille taulukkolaskennalla, sopivilla matemaattisilla ohjelmistoilla (Matlab, Scilab tai Octave) ja osaamillaan varsinaisilla ohjelmointikielillä.
- Tuntevat laskimien eri tehtäviin käyttämien numeeristen menetelmien perusteet, jotta saatuja ratkaisuja voitaisiin kriittisesti arvioida.
- Ymmärtävät askelpituuden muutoksen vaikutuksen tuloksiin muun muassa Eulerin menetelmää soveltavia simulointimalleja käyttäessään.

Listasta puuttuvat erittäin tarpeelliset vektoreihin ja matriiseihin liittyvät numeeriset menetelmät, koska opiskelijoiden lähtötaso ja käytössä oleva aika rajaavat ne tarkastelun ulkopuolelle.

Työelämän vaatimukset eivät ole mitenkään ristiriidassa jatko-opintojen vaatimusten kanssa. Jatko-opinnoissa edellytetään suomalaisen lukion opetussuunnitelman perusteissa mainittujen asioiden osaamista. Nämäkin asiat yllä oleva opetussuunnitelma kattaa, mutta ohjelmointipainotus ja vaatimus kyvystä osoittaa tai arvioida vastauksen oikeellisuus ja tarkkuus korostuu.

Selkeästi jatko-opintoja ja matemaattista ajattelua kehittäväksi osaksi opintoihin liitetään menetelmien teoreettiset suppenemistarkastelut ja lukujärjestelmien käsittely.

Päivölän kesäopintojen tapainen ratkaisu toteutukseltaan ja oppisisällöiltään löytyy Moskovasta [38], ja siellä on ohjelmointia ja numeriikkaa niin ikään opetettu matemaattisesti suuntautuneille nuorille. Heidän opetusratkaisunsa ja valitut asiat ovat hyvin lähellä Päivölää: ryhmäkoko on ollut likimain sama (n. 25), tietotekniikassa on keskitytty algoritmeihin, analyyttiset ja numeeriset menetelmät kulkevat yhdessä ja virheen arviointikykyä pidetään tärkeänä osana numeriikan opetusta. Eroina voidaan todeta ohjelmoinnin osalta Päivölän Java –kieli vastaan Moskovan Pascal ja Päivölälle ominainen lähiverkon käytön opetus sekä matematiikan osalta Päivölän painotus differentiaaliyhtälöihin. Moskovan mallia ei ole käytetty Päivölän opetuksen esikuvana vaan mallit ovat samantapaisia ilmeisesti samankaltaisen ajattelun tuloksena.

3.1.3 Oppimateriaalista

Koska oppilaat tulevat suoraan peruskoulusta, oppimateriaalin on oltava suomenkielistä. Opetuksen kannalta Suomi on pieni maa, ja suomenkielisiä numeriikan oppikirjoja on vähän. Lukiotason analyysin ja numeeristen menetelmien oppikirjat [11] [12] [13] [32] [33] sisältävät puolisuunnikassäännön ja Simpsonin säännön, separoituvat differentiaaliyhtälöt sekä Eulerin ja Rungen-Kuttan menetelmät. Numeerisen integroinnin käyttäminen separoituvan differentiaaliyhtälön ratkaisussa on mainittu vain tekijän kirjoittamassa oppimateriaalissa [9] & [22], vaikka se on ainoa tapa tavallisten laskimien käyttäjille.

Yhdessäkään ei käsitellä laskimien vuoksi tarvittavaa Gaussin integrointia, eikä niissä käsitellä integroinnin virheanalyysiä, joka perustuu eri jakoväleillä saatuihin likiarvoihin. Tämä virheanalyysi on helposti ohjelmoitava ja siksi ensisijainen. [19] Se on myös tunnettava laskimien sudenkuoppien tunnistamiseksi. Kuriositeettina mainittakoon, että ko. virheanalyysi ei vielä esiintynyt Kreyzigin kirjan kuudennessa painoksessa vuodelta 1988 [19b], mutta se käsitellään kahdeksannessa painoksessa vuodelta 1999 [19].

Lukion opetussuunnitelman perusteet 1994 [34] jättää opettaviin asioihin huomattavan vapauden, kun matematiikan syventävät opinnot kohdassa todetaan:

> ”Pitkässä matematiikassa kursseja voidaan suunnitella seuraavista aihepiireistä:
>
> …
>
> numeeriset menetelmät, joka toteutetaan laskinten ja tietokoneiden avulla ja jossa tutustutaan ratkaisualgoritmeihin ja likimääräismenetelmiin yhtälöiden ja yhtälöryhmien ratkaisussa sekä differentiaali- ja integraalilaskennassa. ”

Vuoden 2005 elokuussa voimaan tulevissa opetussuunnitelman perusteissa differentiaaliyhtälöitä ei ole. [35] Siten opetusta ei voi perustaa suoraan suomalaisiin opetussuunnitelmiin tai mihinkään valmiiseen suomalaiseen oppimateriaaliin, vaan on päädytty liitteen 1 materiaaliin, jota muu materiaali (mm. liite 2) täydentää.

3.2 Opetuksen kuvaus

3.2.1 Päivölän kansanopiston matematiikkalinjan kesälukukausi

Päivölän kansanopiston matematiikkalinjan kesälukukauden opinnot on suunnattu ensisijaisesti juuri peruskoulunsa päättäneille ja ne kestävät kahdeksan kalenteriviikkoa, jotka on rytmitetty kahden viikon jaksoihin: seitsenpäiväinen viikko koulua, viikko vapaata, viikko koulua jne. Syyslukukauden alettua ei enää ole vapaata

väliviikkoa. Tässä kuvattava opetusmalli rajautuu kesälukukauteen. Opistolla oppilaat asuvat opiskeluviikon ajan yleensä samassa talossa, syövät yhdessä opiston ruokalassa ja nukkuvat 2 hengen huoneissa. Opettajista osa asuu samassa talossa oppilaiden kanssa ja erityisesti kesälukukaudella muutkin opettajat osallistuvat tiiviisti myös opiskelijoiden vapaa-aikaan.

Opinnot alkavat klo 9 ja sisältävät neljä opetusrupeamaa: 1) aamupäivän opetus 9-11:30, lounas, 2) iltapäivä ensimmäinen opetusosa 12:30-14:30, kahvi, 3) iltapäivän toinen opetusosa 15-17, päivällinen ja 4) iltaopetus n. 2 kellotuntia (3 oppituntia) välillä 18-21:15 eli yhteensä 11-12 oppituntia päivässä. Välitunteja ei ole.

Viikonloppuna opiskellaan ohjelmointia (15 h), ja sunnuntaiaamuna on koe viikon aikana opiskellusta matematiikasta. Seuraavan opiskeluviikon puolivälissä on edellisen viikon asioita mittaava koe kaikille. Ylimääräinen kertauskoe pidetään sitä seuraavan opiskeluviikon maanantaiaamuna ennen muiden asioiden opetuksen alkua. Siten opettajalla ja opiskelijalla on mahdollisuus saada toistuvasti palautetta osaamistasosta ja vielä opiskelua vaativista asioista.

Opettajan lisäksi saapuvilla on usein vanhempia opiskelijoita neuvomassa laskuharjoitustehtävien ratkomisessa. Ajoittain opetukseen käytetään luokassakin kahta opettajaa. Opetuksessa opettajien vahvuuksia pyritään käyttämään hyväksi roolittamalla tehtäviä. Iltaisin ja tauoilla opettajat ja opiskelijat keskustelevat vapaamuotoisesti sekä opinnollisista että muista aiheista ja opiskelijat voivat pyytää niin toisiaan kuin opettajia selvittämään epäselviksi jääneitä asioita. Toisaalta taukoja ja iltoja voidaan käyttää myös jalkapallon, shakin tai go:n pelaamiseen sekä jutusteluun, jota kutsutaan ”idlaamiseksi”. Näin rakennetaan hyvää ryhmähenkeä, joka noin 30 kesäopiskelijan keskuuteen onkin muodostunut. Ryhmään koetaan kuuluvaksi opiskelijoiden lisäksi ylempien vuosikurssien opiskelijoita ja jopa vuosia sitten opintonsa päättäneitä opiskelijoita sekä opettajat.

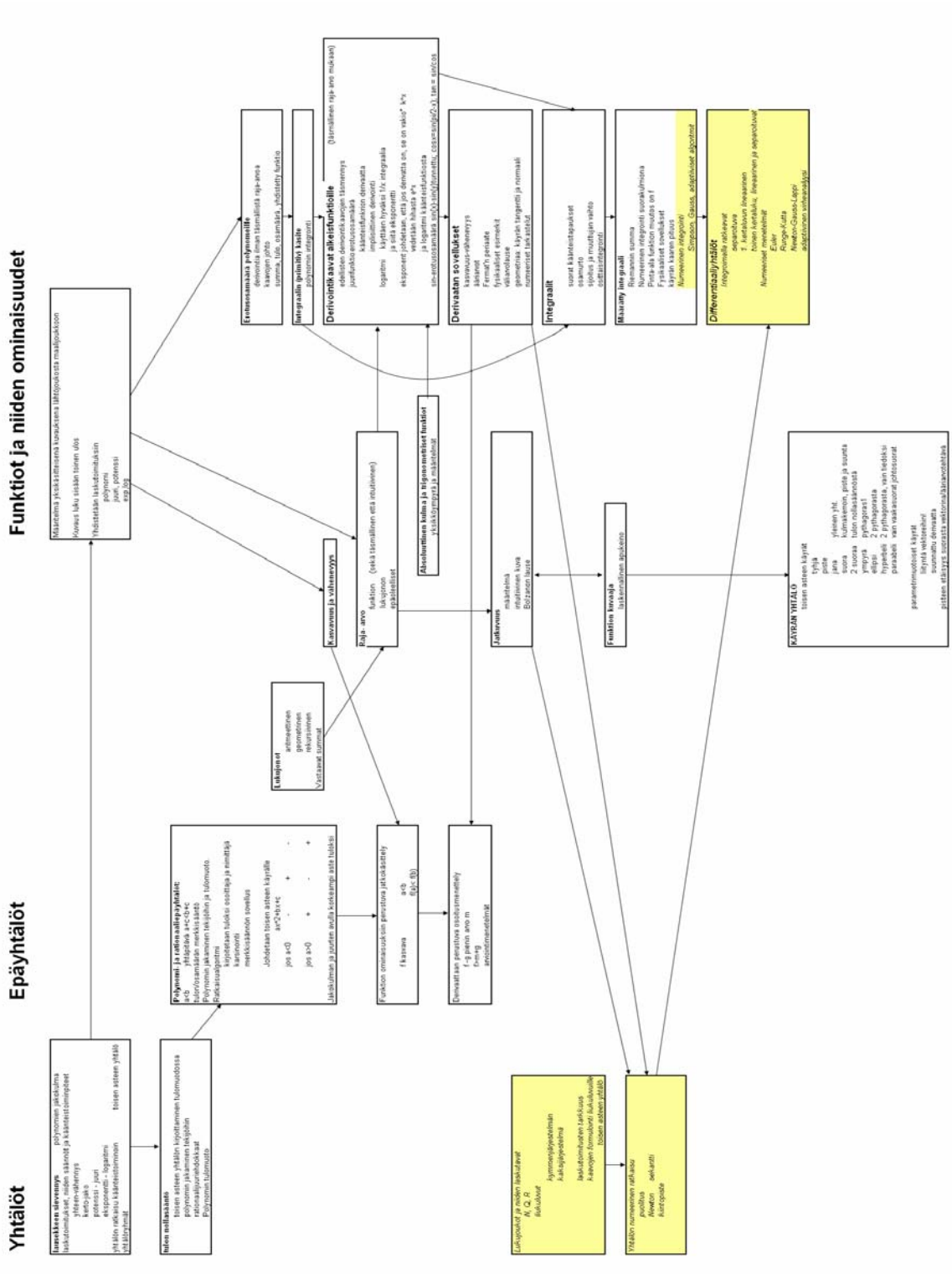
Päivölän kansanopiston matematiikkalinjan kesälukukausiopetus on alkanut vuonna 1997 ja saavuttanut pienten kehitysaskeleiden jälkeen nykymuotonsa oleellisesti ottaen vuonna 2000. Opinnoissa vastaavana opettajana toimi vuoteen 2000 Kullervo Nieminen ja vuodesta 2001 Esa ja Merikki Lappi sekä muina opettajina mm. Anne Kouhia 1997-2003 ja Jukka Ilmonen vuodesta 2003 sekä Janne Puustelli vuodesta 2005. Lapit osallistuivat kesäopetukseen nykymuotoisen matematiikkalinjan alusta lähtien yhdessä Kullervo Niemisen kanssa.

Vuosina 1997-2000 opetusasiat noudattivat likimain lukion normaalia kurssijärjestystä ja opetussuunnitelmaa, mutta vuoden 2001 kokeilun hyvien tulosten jälkeen kesäopetus muodostaa algebran, funktioiden ja analyysin kokonaisuuden, joka opetus ja tarkasteltavat asiat poikkeavat merkittävästi lukion opetussuunnitelman perusteista opetuksen jaksotuksen ja painotuksen osalta, vaikka opetuksen tavoitteet ja keskeiset sisällöt ovatkin kattaneet lukion vuoden 1995 opetussuunnitelman perusteiden mukaiset kurssit 1,2,6,7,8 sekä syventävät kurssit 12 ja 13 eli kurssiniminä: funktiot ja yhtälöt 1&2, differentiaalilaskenta 1&2, integraalilaskenta

sekä syventävät kurssit analyysi ja numeeriset menetelmät, joista kahden viimeisen opetukseen tämä työ keskittyy. Päivölän matematiikan kesäopetusta kuvaa kaavio 1, johon on sijoitettu otsikkotasolla opetettavat asiat ja niiden järjestys.

Numeeristen menetelmien opetus on siten pohjustettu suomalaisittain katsottuna vahvalla aritmetiikan, algebran ja differentiaalilaskennan pohjalla, jonka opetuksessa on jo huomioitu viimeisen kesäviikon viikon aikana opetettavat varsinaiset numeeriset menetelmät.

Liitteen 1 opetusmateriaali on tämän Päivölän viimeisen viikon differentiaaliyhtälöiden ja numeeristen menetelmien opintokokonaisuuden oppimateriaali, jota täydentävät Lukion Laskinopin kirja (ote liitteenä 2), Tammen Pyramidi –sarjan oppikirjat ja yksityiskohtien osalta Esa Lapin laatimat muut harjoitustehtävä- ja opetusmonisteet. Ensimmäistä koko numeeristen menetelmien kurssin oppimäärän sisältänyttä opetusmonistetta käytettiin Helsingin matematiikkalukion kesäkoulussa elokuussa 2001, ja lukuvuonna 2004-2005 materiaali kehittyi liitteen 1 mukaiseksi. Kiinnostuneimmat opiskelijat ovat käyttäneet myös Päivölän Kansanopiston matemaattisen kirjaston laajaa valikoimaa opetuksen syventämiseen. Oleellisesti ottaen samaa oppimateriaalia käytetään myös Helsingin matematiikkalukion ilta- ja viikonloppuopetuksessa.



Kaavio 1. Päivölän kansanopiston kesäopetuskaavio. Numeeristen menetelmien ja differentiaaliyhtälöiden osuus himmeästi väritetty. Osuus käyrän yhtälö ei kuulu kesäopetuksen piiriin, vaan seuraa sitä.

3.2.2 Numeeristen menetelmien ja differentiaaliyhtälöiden opetus

3.2.2.1 Yleiskuva opetuksen järjestelyistä

Tarkasteltavat asiat on opetettu Päivölässä nykymuotoon vakiintuneena yhden viikon aikana tehtävänä opetusrupeamana vuodesta 2002 alkaen ja Helsingin matematiikkalukiossa neljästi, numeriikan osalta kesäkoulussa 2001, vuonna 2002 viikonloppukurssina ja sen jälkeen kahdesti iltaopetuksena. Differentiaaliyhtälöt on pyritty opettamaan samanaikaisesti numeeristen ratkaisujen kanssa vuoden 2002 opetuskokeilusta lähtien. Tässä keskitytään kuvaamaan numeeristen menetelmien ja differentiaaliyhtälöiden opetus Päivölän kansanopistolla, missä opetukseen käytetään yksi kalenteriviikko. Matematiikan opetukseen käytetään kaikki oppitunnit maanantaista klo 12:sta perjantaihin klo 20:een ja lisäksi viikonlopun ohjelmointiopetuksen lomassa on ns. harjoituskoe sunnuntaiaamuna. Kokonaisuutena aikaa on siten 50 oppituntia perusteiden opetukseen, ja tässä ajassa nopeimmat opiskelijat saavuttavat oppimistavoitteet erinomaisesti, suuri osa hyvin ja osa oppii vain alkeet. Opiskelijoilla on myöhemmin mahdollisuus syventää osaamistaan, siksi opintojen päättyessä toisen opiskeluvuoden keväällä lähes koko opiskelijaryhmä onkin saavuttanut hyvän tai erinomaisen osaamistason.

Ensimmäisenä opiskelupäivänä opiskellaan numeeristen menetelmien perusteita: liukuluvut, virhetarkastelu ja funktion nollakohdan hakumenetelmiä. Toisena päivänä jatketaan maanantaina opetetun harjoittelua ja opiskellaan interpolaatiopolynomeja sekä kerrataan edellisen opiskeluviikon integraalilaskentaa. Kolmantena päivänä opiskellaan numeerinen integrointi ja aloitetaan differentiaaliyhtälöt. Neljäs ja viides päivä käytetään differentiaaliyhtälöihin sekä analyyttisesti että numeerisesti, ja opintojakson lopuksi kootaan ja kerrataan viikon anti.

Viikonlopun opetus perjantai–illasta sunnuntai-iltapäivään on ohjelmointia, eikä se suoranaisesti ole osa numeeristen menetelmien opetusta. Kurssia edeltänyt ja sen kestäessä tapahtuva ohjelmoinnin ja algoritmien opetus antaa tärkeitä valmiuksia myös numeeristen menetelmien opetukseen, ja kaikkien opiskelijoiden vähintään auttava ohjelmointitaito helpottaa ratkaisevasti numeeristen menetelmien käsittelyä. Numeerisia ongelmia voidaan käyttää myös ohjelmoinnin harjoitustehtävinä.

3.2.2.2 Opetuksen yksityiskohtia

Liitteen 1 oppimateriaali kattaa pääosin asiat ja kertoo karkeasti opetettavien asioiden järjestyksen. Opetus alkaa numeeristen menetelmien johdanto-osalla, jossa käydään läpi liukuluvut ja laskennan tarkkuuteen vaikuttavat tekijät. Jokainen testaa oman laskimensa tarkkuuden ja laskutoimitusten virheet käydään läpi. Tämä tapahtuu muun muassa laskemalla laskimella erotus $(2^n+1) - 2^n$. Kun lukua n kasvatetaan, erotus poikkeaa luvusta 1 laskimen liukulukuesitykseen liittyvän tarkkuuden rajalla, joka on

vanhoissa laskimissa n. 30 ja uusissa 43-49. Tämä on todettu havainnolliseksi tavaksi tarkastella laskimen tarkkuuden rajoja ja laskimen mantissan lukuesityksen ”bittien” määrää.

Laskennan tarkkuutta käsittelevää osaa seuraa funktion nollakohdan ja yhtälön numeerisen ratkaisun tarkastelu. Yhtälön numeerisessa ratkaisemisessa painopiste on ratkaisujen lukumäärän ja tarkkuuden arviointiin perustuvissa menetelmissä, koska niitä tarvitaan tulosten analysoinnissa yhtälön ratkaisumenetelmästä riippumatta. Teoriapohjan kannalta tärkeitä ovat kiintopistemenetelmä ja sen suppeneminen sekä Newtonin menetelmä, jota sovelletaan myöhemmin separoituvan differentiaaliyhtälön ratkaisemiseen. Opetettavat iterointimenetelmät on kuvattu liitteen 1 luvussa 1.2.

Kun yhtälöt on opeteltu ratkaisemaan numeerisesti, opiskellaan interpolointia ja numeerinen integrointi. Opetuksessa lähdetään liikkeelle tasavälisiin jakoihin liittyvistä menetelmistä ja niiden virheen arvioinnista. Adaptiivisia menetelmiä ja eri jakoväleillä laskettujen likiarvojen käyttöä virheanalyysiin korostetaan, koska ne voidaan ohjelmoida osiksi algoritmia soveltavia laskentaohjelmia.

Kunkin opiskelijan käyttämän laskimen osalta selvitetään laskimen käyttämät menetelmät numeerisen integroinnin osalta sekä kokeilemalla varmistettu arvio integroinnin tarkkuudesta ja menetelmän toimivuuden rajasta. Kun oppitunneilla on käyty läpi suuri määrä oppilaiden ja opettajien laskimia, havaintona voidaan todeta integroinnin osalta vanhojen ja halpojen laskimien käyttävän Simpsonin sääntöä numeeriseen integrointiin (esim. Ti-68, Sharp EL-506V, Sharp EL9600), mutta uusien toimivan myös adaptiivisella Gaussin integrointikaavalla (TI95, TI96, Sharp EL 9900, Casio Fx 1.0 plus -sarja).

Laskimia testattiin muun muassa integroimalla eksponenttifunktiota e^{-x}. Integrointiväli alkoi nollasta ja ylärajaa kasvatettiin. Tasaväliseen jakoon perustuvan Simpsonin säännön tulos kasvoi nopeasti yli yhden, koska ensimmäisen arvon $e^0 = 1$ painoarvo tuli liian suureksi. Tämä päti kaikilla ”suljetuilla kaavoilla” (closed formula), jotka ottavat integroinnin alkupisteen mukaan laskentaan. Gaussin integrointikaava on avoin (open formula). Sitä soveltavien adaptiivisten algoritmien laskenta-aika piteni ja tulos oli jotakuinkin oikea, kunnes tapahtui romahdus: laskenta tapahtui hetkessä ja tulos oli tasan 0. Siitä voidaan päätellä, että laskenta perustuu avoimeen kaavaan (open furmula), jossa laskentapisteenä ei käytetä integroinnin alarajaa (eikä ylärajaa): kun integrointiväli tulee riittävän pitkäksi, eksponenttifunktion kaikki laskentaan tulevat arvot ovat laskimen tarkkuudella niin lähellä nollaa, että kahdella eri jakovälillä virhettä arvioiva vertailutulosten erotuskin alittaa lopetusrajan.

Tämäkin puoltaa tarvetta opettaa Simpsonin ja Gaussin integrointimenetelmät ja esittää perustelut sekä askelten lukumäärään perustuvista virhekaavoista että adaptiivisten algoritmien lopetusehdoista ja sudenkuopista. Adaptiivisen laskennan periaate on esitelty lyhyesti kirjallisuudessa mm. [1], [19] ja liitteessä 1 sivulla 18.

Integraalilaskennan jälkeen jatketaan differentiaaliyhtälöihin, joiden opetuksessa numeeriset ja analyyttiset menetelmät kulkevat rinnakkain. Numeerisia ratkaisuja ja niiden perusteella piirrettyjä kuvaajia käytetään havainnollistamaan opetusta alusta alkaen. Tämäntapaisesta lähestymistavasta on julkaistu myös englanninkielisiä oppikirjoja, muun muassa [3].

Opiskelijoita rohkaistaan perehtymään oman laskimensa ominaisuuksiin ja käyttämään taulukkolaskentaohjelmistoja ja Octaven ja Matlabin kaltaisia matemaattisia ohjelmistoja. Osa oppitunneista pidetään tämän vuoksi tietokoneiden äärellä. Tämä pätee myös Helsingin matematiikkalukion iltaopetukseen, jossa muun muassa mallinnuskurssin koe tehdään tilassa, jossa opiskelijat voivat käyttää matemaattisia ohjelmistoja sisältäviä tietokoneita.

Ensimmäisen kertaluvun yhtälöiden lisäksi toisen kertaluvun differentiaaliyhtälöt opitaan ratkomaan numeerisesti käyttäen Rungen – Kuttan menetelmää ja valmisohjelmistoja. Lisäksi opetellaan analyyttinen ratkaisutapa sekä helposti integroinnein ratkeaville yhtälöille että toisen kertaluvun lineaarisille vakiokertoimisille differentiaaliyhtälöille.

Differentiaaliyhtälöryhmien ratkaisu ohjelmoimalla Rungen - Kuttan menetelmä laskimiin (Basic) ja varsinaisilla ohjelmointikielillä (Java, C++) jätetään opintojen syventäväksi ainekseksi, jonka vain nopeimmat opiskelevat.

Opetuksessa käytettävät esimerkkitehtävät etenevät menetelmiä kuvaavista perustehtävistä kohti soveltavia ja fysiikan opintoja pohjustavia tehtäviä, vaikka puutteet fysiikan osaamisessa rajoittavat todellisten esimerkkien tehokasta käyttöä. Tämä on kuitenkin pieni haitta verrattuna etuihin, joita matematiikan osaaminen tarjoaa syksyllä alkavassa fysiikan opetuksessa (Päivölä). Suoranaisen opetuksen ulkopuolella opiskelijoilla on käytössä analyyttisiä matemaattisia ohjelmistoja: Helsingin matematiikkalukiossa Mathematica, Päivölässä yksittäisiä Maple – lisenssejä ja oppilailla omia Maxima –ohjelmistoja. Opiskelijat käyttävät niitä tulosten tarkastamiseen ja itsenäisiin analyyseihin.

Opetuksessa eri numeeristen menetelmien ratkaisutavat käydään läpi yleisessä muodossaan ja sen jälkeen harjoitellen sitä, miten niitä voidaan soveltaa laskimilla, miten ne pitää ohjelmoida Matlabin tai varsinaisen ohjelmointikielen avulla ja kuinka niitä voidaan soveltaa taulukkolaskennassa.

3.2.2.3 Saavutettujen valmiuksien arviointia

Päivölän opiskelijoiden ja Helsingin matematiikkalukion erikoiskursseille osallistuvan oppilasaineksen ja käytössä olevien välineiden välillä ei ole havaintojeni mukaan ole oleellista eroa. Kuitenkin Päivölässä numeeristen menetelmien opetuksessa haluttu tulostaso on saavutettu lyhyemmässä opetusajassa. Tässä yksi mahdollinen selittävä

tekijä on ohjelmoinnin ja algoritmien opetus, jota kaikki päivöläläiset ovat saaneet kolmen ohjelmointiviikonlopun aikana ennen numeeristen menetelmien ja differentiaaliyhtälöiden opetusta.

Vaikka opintoympäristö ja kurssin järjestelyt poikkeavat toisistaan jonkin verran, molemmissa kouluissa on ollut sosiaalisesti kiinteä ryhmä, sama opettaja ja sama oppimateriaali, joten nämä eivät selitä eroa. Siten nimenomaan ohjelmointitaito tuntuu selittävän Päivölässä tarvittavan vähäisemmän oppituntimäärän numeeristen menetelmien ja differentiaaliyhtälöiden opiskelussa, koska myös helsinkiläisistä ohjelmoinnin harrastajat näyttävät oppivan muita nopeammin. Käytännön opetushavaintona voidaan todeta, että saman osaamistason saavuttaminen numeerisessa menetelmissä edellyttää Helsingin matematiikkalukiossa ohjelmointipainotteisen lisäkurssin ”matemaattiset tietokoneohjelmistot” suorittamista.

Ohjelmointitaito näkyy myös lähestymistavassa tietoteknisiin välineisiin. Riippumatta lähtötasosta kaikki oppivat nopeasti näppäilysarjat, joilla tehtävät ratkotaan käyttäen laskimen normaaleja ominaisuuksia, mutta laskimen ohjelmointiominaisuutta käyttävät vain aiemmin ohjelmoineet.

Vaikka laskimien ja taulukkolaskennan ohjelmointia ei opeteta erikseen, laskimien ohjelmointitoiminnon käyttö on yleistä. Sen sijaan taulukkolaskennan ohjelmointitoimintojen käytöstä ei ole myöhemmissä opintovaiheissa juurikaan havaittu. Matlabilla ja ohjelmointikielillä tehdyt ohjelmat ovat yleisiä opintojen jatkovaiheissa.

Osaamisen ja opetuksen onnistumisen mittaamiseen on kokeiden lisäksi molemmissa kouluissa käytetty myöhempien kurssien ja harjoitusten tekotapojen seuraamista, tutkielmien ohjaamisen yhteydessä tehtyjä havaintoja sekä mm. kansainvälistä MCM/ICM mallinnuskilpailua, jonka ratkaisuissa on selvästi havaittu saavutettu kyky ratkoa soveltavia ongelmia. Asuminen päivöläläisten oppilaiden kanssa samassa talossa on mahdollistanut oppilaiden kanssa jatkuvan kanssakäymisen ja keskustelut, joihin edelliset havainnot perustuvat. Valitettavasti keskusteluista ei ole päiväkirjaa, mutta mm. MCM/ICM –kilpailutöitä on kouluilla, ja vuosittain tehtävät ja tulokset löytyvät kilpailun järjestäjän nettisivuilta www.comap.com [5]. Tuloksista on myös kooste liitteenä 3.

4. Johtopäätökset

4.1. Numeerisen integroinnin käytöstä differentiaaliyhtälön ratkaisuun

Numeerisen integroinnin käyttö differentiaaliyhtälöiden ratkaisussa voi säästää huomattavasti laskenta-aikaa. Integrointeja käyttävistä ratkaisuista separoituvan yhtälön ratkaisualgoritmi on osoittautunut monessa mielessä käyttökelpoiseksi: se on käytännössä havaittu nopeammaksi kuin yleiset perinteiset menetelmät, se on helppo opettaa opiskelijoille ja sen käyttäminen ratkaisuissa on mahdollista sekä laskimissa että omana pienenä ohjelmanaan. Siten menetelmä soveltuu hyvin käytännön ongelmiin.

Ensimmäisen kertaluvun lineaarisen yhtälön osalta analyyttisestä ratkaisusta tunnetun integraalin ratkaiseminen numeerisesti tuottaa hyviä tuloksia, joten sitä kannattaa käyttää myös numeerisen ratkaisun pohjana.

Esimerkkien perusteella toisen kertaluvun differentiaaliyhtälöissä menetelmien paremmuus riippuu pitkälti siitä, mitä ollaan ratkaisemassa. Jos suoraan integroimalla saatiin haluttu suureen arvo, integrointiin perustuvat menetelmät olivat kilpailukykyisiä, mutta jos ratkaisua jouduttiin iteroimaan, perinteiset numeeriset menetelmät osoittautuivat usein nopeimmiksi.

Tutkimuksessa tarkasteltiin laskennallisesti tilanteita, joissa kaikki integraalit ratkaistiin yleensä numeerisesti. Koska mahdollinen analyyttinen ratkaisu nopeuttaa laskentaa, separoituva differentiaaliyhtälö kannattaa kirjoittaa integraalimuotoon ja ratkaista analyyttisesti integraalit tai edes toinen niistä, jos se on mahdollista. Tutkimuksen tulosten valossa voidaan arvioida, että separoituva differentiaaliyhtälö ratkeaa yleensä perinteisiä numeerisia menetelmiä nopeammin esitellyllä numeerisen integrointiin perustuvalla algoritmilla, ja saavutettu etu kasvaa, jos vähintään toinen integraaleista ratkeaa analyyttisesti.

4.2. Opetukseen liittyviä havaintoja

Ensimmäinen numeerisiin menetelmiin liittyvä havainto on auttavankin ohjelmointitaidon antama etu numeerisia ongelmia ratkottaessa. Kyky tehdä omia ohjelmia ja algoritmeja, jotka sopivat laskimiin, taulukkolaskentaan tai

ohjelmointikielelle tarjoaa huomattavat mahdollisuudet sekä oppia ymmärtämään numeerisen ratkaisun kulkua että lisätä valmiisiin ratkaisuihin tehtävän erityispiirteet huomioon ottavia lisäehtoja.

Siten numeeristen menetelmien yhteydessä oppilaita tulee kannustaa omien ratkaisujen tekoon monipuolisesti eri välineillä, joita ovat esimerkiksi laskimet ja tietokoneiden taulukkolaskentaohjelmat ja matemaattiset ohjelmistot sekä perinteiset ohjelmointikielet. Tähän pitää varata aikaa ja parhaimmillaan osin yhdistää numeeristen menetelmien ja ohjelmoinnin opetus.

Toisaalta kyky arvioida differentiaaliyhtälöiden ominaisuuksia ja käyttäytymistä yksinkertaisten analyyttisten tarkastelujen avulla on tärkeää numeeristen ratkaisujen osalta, koska ohjelmointitaitoinen voi lisätä ratkaisualgoritmeihin yhtälön ominaisuuksien perusteella ehtoja, jotka estävät numeeristen menetelmien harhautumista. Vahva analyyttinen osaaminen ja hyvä ymmärrys differentiaaliyhtälöistä sekä integraaleista on mielestäni edelleen pohja myös numeeristen ratkaisutapojen tehokkaan käytön oppimiseen.

Kokonaisuutena differentiaaliyhtälöiden ja niiden kaikkien erilaisten ratkaisumenetelmien opiskelu samanaikaisesti on esitietojen ja sovellustarpeen valossa järkevää. Kokonaisuus on havaittu toimivaksi Päivölän kansanopistossa vuosina 2002-2004 pidetyillä kursseilla sekä Helsingin matematiikkalukion iltatunneilla.

Viitteet

[1] H-J.Bartsch, *Taschenbuch Matematischer Formeln.* Fachbuchferlag, 1999.

[2] J. Björkman, M. Kokkala, E. Lappi, *Applicability of FiRECAM for risk assessment of Finnish multistorey apartment and office buildings. Third International Conference on Performance-Based Codes and Fire Safety Design Methods*. Lund University, Lund, Sweden, 15-17 June 2000. Proceedings.

[3] P. Blanchard, R. Devaney, G. Hall, *Differential Equations*. Brooks/Cole Publishing Company, 1998.

[4] Casio FX 1.0.

[5] Comap. http://www.comap.com. Luettu 10.3.2005.

[6] T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms. 2nd ed.* The MIT Press, 2001.

[7] C. R. Dutcher et al. *FIRECAM Technical User Guide*. National Reseach Institute 1997, Canada. (versio 1998)

[8] Y. He, M. Luo, J. Min, *Fire growth and smoke spread models –development and experimental validation*. Fire code reform center ltd, Australia, November 1997

[9] K. Hemmo, E. Lappi, R. Lundahl, *Pyramidi: Numeeriset menetelmät*. Kustannusosakeyhtiö Tammi, 2002.

[10] M. Janssens, *An Introduction to Mathematical Fire Modeling. 2nd ed*. Tecnomic Publishing Company, 2000.

[11] P. Jäppinen, A. Kupiainen, M. Räsänen, *Calculus 6: Analyysin jatkokurssi*. Kustannusyhtiö Otava, 1995.

[12] J. Kangasaho, J. Mäkinen, J. Oikkonen, J. Paasonen, M. Salmela, *Analyysi*. WSOY, 1997.

[13] J. Kangasaho, J. Mäkinen, J. Oikkonen, J. Paasonen, M. Salmela, *Numeerinen matematiikka.* WSOY, 1996.

[14] L. Kangas, E. Lappi, An example of Markovian Combat modelling. *Nordic Military Operation Analysis Symposium Proceedings*.
Helsinki 27-28.10.2004.

[15] L. Kangas, *Taistelun stokastinen mallinnus. Diplomityön luonnos 25.5.2005*. Teknillisen korkeakoulun Systeemianalyysin laboratorio.

[16] D. Kincaid, W. Cheney, *Numerical Analysis*. Brooks/Cole Publishing Company, 1991.

[17] J. Kosola, T. Solante, *Digitaalinen taistelukenttä informaatioajan sotakoneen tekniikka 2. painos*. Maanpuolustuskorkeakoulu, Tekniikan laitoksen julkaisusarja 1, tutkimuksia. Helsinki 2003.

[18] R. Kress, *Numerical analysis*. Springer, 1998.

[19] E. Kreyzig, *Advanced Engineering Mathematics. 8th Edition*. John Wiley, 1999.

[19b] E. Kreyzig, *Advanced Engineering Mathematics. 6th Edition*. John Wiley, 1988.

[20] E. Lappi, M. Aunola, *Fast algorithm for Numerical Integration of Separable Differential Equations. Eccomas 2004 Proceedings.* Jyväskylä. 2004.

[21] E. Lappi, *A Simple Numeric Method for Solving Separable Differential Equations. Nordic Matlab Conference Proceedings*, s. 268-272. Kööpenhamina, 2003.

[22] E. Lappi, R. Lappi, *Lukion Laskinoppi*. Oy Perkko, 2003.

[23] E. Lappi, *A method to evaluate operational performance of electric warfare systems. Nordic Military Operation Analysis Symposium Proceedings*.
Helsinki 27-28.10.2004.

[24] E. Lappi, J. Hietaniemi, M. Kokkala. *Ikkunan kautta julkisivulle leviävän palon todennäköisyyspohjainen riskitarkastelu. Palontorjuntatekniikka* 32. vuosikerta (2002) Nro 1. s. 20-24.

[25] E. Lappi, *Seinän lämpenemisen tarkastelumalli palosimulointeihin*. Selvitys RAKE-PALO.008, IVO International Oy,1993.

[26] E. Lappi, *Suuren kauppahallin paloselvitys*. Turvallisuusarviointi TA Oy, Vantaa, 7.7.1994

[27] E. Lappi, *EL-tank –ohjelmisto*. Turvallisuusarviointi TA OY, Vantaa ,1994.

[28] E. Lappi, *Tappioiden ja lääkintähuollon järjestelyjen logistinen vaikutus joukon taistelukykyyn. (ohjelmaversio 0.2.)* Puolustusvoimien Teknillinen Tutkimuslaitos 27.5.2005.

[29] M. Lehtinen, *Pieni Simulointikirja. Simuloinnin alkeita taulukkolaskennan avulla.* Maanpuolustuskorkeakoulu, Tekniikan laitos, Julkaisusarja 5. No 3, 2004.

[30] Magnusson et al. *A PROPOSAL FOR A MODEL CURRICULUM IN FIRE SAFETY ENGINEERING*. In D.Drysdale (ed.), *Fire Safety Journal, Vol 25, No. 1. Special issue,* July 1995.

[31] *Matlab 6.5.1.199709 Release 13*. MathWorks inc. August 4th 2003.

[32] J. Merikoski, K. Väänänen, T. Laurinolli, T. Sankilampi, *Matematiikan taito 13: Analyysi.* WSOY, 1996.

[33] R. Nurmiainen, J. Rauhalinna, *Pyramidi: Analyysi*. Kustannusosakeyhtiö Tammi, 2001.

[34] *Lukion opetussuunnitelmien perusteet 1994*. Opetushallitus, 1994.

[35] *Lukion opetussuunnitelman perusteet 2003* Määräys 33/011/2003, Opetushallitus, 2003.

[36] W. Press, W. Vetterling, S. Teukolsky, B. Flannery, *Numerical recipes in C++. 2nd ed*. Cambridge University Press. 2002.

[37] Sharp EL-9900

[38] Y. Shestopalov, *Teaching applied mathematics and numerical methods on introductory levels.* http://www.icme-organisers.dk/fsg13/shes.pdf. Luettu 10.3.2005.

[39] T. Tanaka, K. Nakumara, *A Model for Predicting Smoke Transport in Building based on Two Layers Zone Concept*. Report of Building Research institute, Ministry of Construction, Japan1989 (Japaninkielinen)

[40] Ti-86, Texas Instruments.

[41] E. Weisstein, *CRC Concise Encylopedia of Mathematics*. CRC Press, 2000.

[42] *Voimasuhdeanalyysi- projektin johtoryhmän pöytäkirja 14.2.2005*. TKK Geoinformatiikan ja kartografian laboratorio. Turvaluokiteltu TLL IV Viranomaiskäyttö.

Liitteet

Liiteluettelo

1. Esa Lappi. Differentiaaliyhtälöitä ja numeerisia menetelmiä.
   Opetuksessa käytetty oppimateriaali, joka on ollut verkossa ja monisteena opiskelijoiden saatavilla.

2. Ote oppikirjana käytetystä Lukion laskinoppi –kirjasta.

3. Comapin järjestämän MCM/ICM -kilpailun tuloksia

Liite 1 Verkossa ja kopiona jaettu oppikirja

Esa Lappi

Differentiaaliyhtälöitä ja numeerisia menetelmiä

SISÄLLYS

1. Numeerisista menetelmistä

1.1 Virheanalyysia

1.1.1 Virhelähteitä

Käytännön sovelluksissa lasketaan aina jonkin verran virheellisillä luvuilla. Tämä pätee kaikkiin numeerisiin menetelmiin ja myös analyyttisiin ratkaisuihin: oikeankin ratkaisun numeerisen likiarvon laskeminen voi johtaa väärään tulokseen, jos analyyttinen ratkaisu on muotoiltu numeerisesti huonosti.

Soveltavan tehtävän virhelähteinä voivat olla seuraavat:

1. Mallin virhe
    a. Kaikkia ilmiöön vaikuttavia tekijöitä ei ole otettu malliin; esimerkiksi ilmanvastus jätetään huomiotta putoamisliikkeessä.
    b. Ilmiöiden tarkkaa matemaattista mallia ei tunneta tai käytetään yksinkertaistettua mallia. Esimerkkinä voitaisiin pitää osakekurssien kehitystä (matemaattinen kaava) ja sään ennustamista (yksinkertaistus).

2. Lähtöarvojen virheellisyys
    a. Sovellusten mittaustulokset ovat aina likiarvoja.
    b. Tarkkoja arvoja ei edes tiedetä. Esimerkiksi uuden tuotteen ominaisuudet ovat vasta suunnittelupöydällä tai asuintalojen palava materiaali riippuu asukkaasta, mutta jo suunnittelun aikana on pakko käyttää simuloinneissa arvioituja arvoja.

3. Numeerisen menetelmän virhe
    a. Numeeriseen menetelmään voi olla sisään rakentunut virhe. Esimerkiksi integroinneissa käytetään polynomia approksimoimaan muita funktioita.
    b. Laskennan lopetusehto voidaan asettaa sellaiseksi, että tulokseen jäävä virhe on hyväksyttävä.

4. Liukulukuesityksen aiheuttama virhe
    a. Tietokoneen esittämät luvut eivät ole reaalilukuja vaan liukulukuja. Siten niillä on rajallinen tarkkuus.
    b. Laskutoimitukset voivat joko kasvattaa tai pienentää virhettä

Numeerinen menetelmä pitäisi valita siten, että lähtöarvoissa olevat virheet eivät kasaudu laskettaessa ja että liukulukulaskutoimitukset eivät kasvata virhettä. Seuraavaksi tarkastellaan lyhyesti laskutoimitusten aiheuttamaa virhettä.

1.1.2 Liukulukuesitys

Laskentavälineet esittävät yleensä luvut liukulukumuodossa, jossa luvulla on mantissa, jossa on *n* kaksijärjestelmän eli binäärijärjestelmän lukua ja eksponentti, jossa on *m* lukua. Luku *n* määrää merkitsevien bittien (tai kymmenjärjestelmässä merkitsevien lukujen) lukumäärän ja m lukualueen, jolla luvut voivat esiintyä.

Luvun *l* esitys *k*-kantaisena liukulukuna voidaan kirjoittaa muodossa

(1) $l = N \cdot k^M$.

Koska luvun mantissa *N* on vakiomittainen, laskennassa eri suuruusluokkaa olevien lukujen merkitsevien numeroiden lukumäärä on sama eli erisuuruisten liukulukujen esitysmuodosta johtuva suhteellinen virhe pysyy samana.

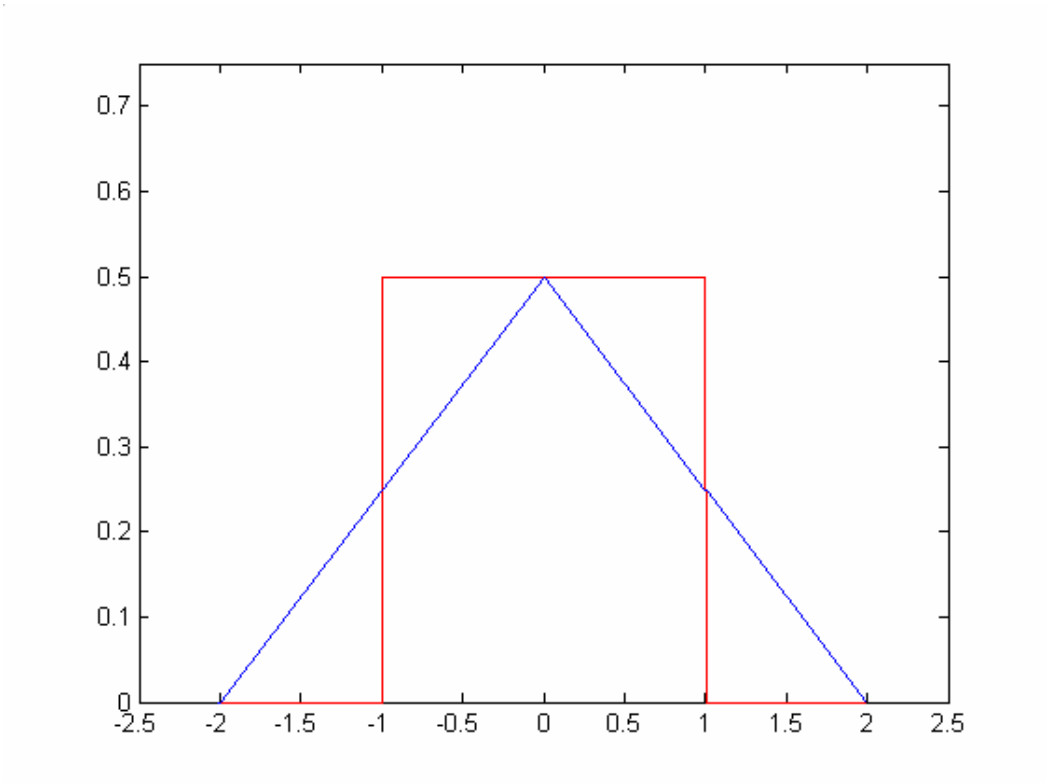
1.1.3 Yhteenlasku

Kun kaksi samanmerkkistä liukulukua lasketaan yhteen, absoluuttinen virhe on samaa suuruusluokkaa kuin alkuperäisten, ja suhteellinen virhe yleensä pienenee.

Tarkastellaan summaa $a + b$, joissa sekä luvun *a* että luvun *b* virhe on suuruusluokkaa *e*. Jos virhe on tasajakautunut, kuten reaaliluvun muunnoksessa
liukuluvuksi, summan $a + b$ virhe on jakautunut välille $[-2e,+2e]$, ja virheen jakauma on ”kolmiojakauman” mukainen (kuva 1). Siten 75% todennäköisyydellä absoluuttinen virhe ei ole suurempi kuin alkuperäisten suureiden virheen yläraja. Edelleen jos *a* ja *b* ovat samaa suuruusluokkaa, suhteellinen maksimivirhe saadaan osamäärästä $2e : 2a = e : a$ eli suhteellinen virhe ei yleensä kasva.

Ainoa ongelmakohta on tilanne, jossa *a* ja *b* ovat eri suuruusluokkaa. Tällöin summan $a + b$ tulos voi olla suurempi luvuista, jos yhteenlaskun jälkeenkin summa pyöristyy alkuperäiseen lukuun. Tällöinkään suhteellinen virhe ei oleellisesti kasva, ellei vastaava yhteenlasku toistu algoritmin sisällä lukuisia kertoja.

Pääsääntönä voidaan siis pitää, että yhteenlaskussa absoluuttinen ja suhteellinen virhe eivät kasva. Usein asia muotoillaan siten, että binäärimuodossa esitetyn liukuluvun viimeinen bitti voi siis olla väärä.



Kahden tasajakautuneen virheen summa on kolmiojakautunut

1.1.4 Vähennyslasku

Numeerisista laskutoimituksista vähennyslasku on selvästi ”vaarallisin”. Tämä näkyy helposti, kun kaksi samaa suuruusluokkaa olevaa lukua vähennetään toisistaan. Tuloksena saatu erotuksen arvo voi olla pelkkää virhettä.

Tarkastellaan tilannetta, jossa $a - b = \varepsilon$, ja lukujen a ja b virhe on e. Tällöin erotuksen absoluuttinen virhe on jälleen kolmiojakautunut, ja suuruusluokkaa e, ja suhteellinen virhe saadaan likimain

(2) $$\frac{e}{a-b} = \frac{e}{\varepsilon} .$$

Siten esimerkiksi jos lukujen erotus on tuhannesosa alkuperäisistä luvuista, suhteellinen virhe on kasvanut tuhatkertaiseksi. Jos lähtöarvojen virhe e on suurempi kuin ε, tuloksessa ei ole todennäköisesti yhtään oikeaa desimaalia, koska suhteellisen virheen arvo voi olla yli 100%.

1.1.5 Kertolasku ja jakolasku

Kertolasku ja jakolasku ovat luonteeltaan laskutoimituksia, joiden tarkastelussa suhteellinen virhe on mielekäs tarkasteltava suure. Kun kaksi lukua kerrotaan keskenään eksponentit lasketaan yhteen ja mantissojen luvut kerrotaan toisillaan. Tulo sisältää edelleen yhtä monta merkitsevää numeroa kuin alkuperäinenkin eli liukulukuesitys ei aiheuta suuria ongelmaa kertolaskussa.

Laskennallisesti virheiden osalta voidaan tutkia oikean tulon $a \cdot b$ ja virheet sisältävän tulon $(a + e)\,(b + e)$ erotusta

(3) $(a + e)\,(b + e) - a\,b = e\,(a + b) + e^2$.

Jos virhe e on pieni verrattuna lukuihin a ja b, niin e^2 on merkityksetön. (Liukuluvuilla graafisissa laskimissa suhde e / a on noin 10^{-14}.) Siten suhteelliselta virheeltään likimain yhtä tarkoille luvuille a ja b tulon suhteellinen virhe on $2e/a$. Kaksijärjestelmän luvussa siis viimeinen bitti on vaarassa.

Jakolaskun osalta tilanne on vastaava. Siten voidaan todeta, että kerto- ja jakolasku eivät kasvata suhteellisen virheen suuruusluokkaa.

1.1.6 Funktion virheestä

Funktioita ohjelmoitaessa edelliset pitää ottaa huomioon. Laskettaessa funktion arvoa periaatteessa virheettömästi laskevalla algoritmilla merkittävää on lähtöarvojen virheen vaikutus tuloksen virheeseen. Mikäli pienetkin muutokset lähtöarvoissa aiheuttavat funktion arvoon suuren muutoksen, vastauksen tarkkuus voi kärsiä. Siten funktiot, joiden derivaatta on suuri, tuottavat ongelmia. Esimerkiksi eksponenttifunktion argumentin absoluuttinen virhe on suoraan tuloksen suhteellista virhettä.

Samoin hyvin lähellä nollaa oleva derivaatta voi olla ongelmallinen. Esimerkiksi ongelmia aiheuttaa tilanne, jossa laskentavälineen tarkkuudella $f(x)$ on nolla pitkällä suljetulla välillä $[a,b]$. Tällöin on mahdotonta saada täsmällistä arvoa x yhtälölle $f(x) = 0$ ellei tehtävää saada muotoilluksi toiseen muotoon, jossa ongelma ratkeaa.

1.2 Yhtälön ratkaiseminen

1.2.1 Tärkeitä lauseita

Yhtälön ratkaisu pyritään yleensä muuttamaan sopivan funktion nollakohdan määrittämiseksi. Tällöin ratkaisun olemassaolon ja tarkkuuden tarkastelussa voidaan

käyttää vastaavan funktion ominaisuuksia. Karkea analyysi tutkittavasta funktiosta on tarpeen ennen nollakohtien numeerista hakemista. Siten voidaan saada tietoa juurten lukumääristä ja karkeasta sijoittumisesta tutkittavalle alueelle. Tämä helpottaa numeerisen ratkaisumenetelmän ja siinä tarvittavien alkuarvojen valintaa.

Alkuarvoina ratkaisualgoritmeihin on usein:

1. jokin lähtöarvo tai alue, jolla ratkaisuja voi esiintyä,
2. ratkaisussa hyväksyttävissä oleva virhe ja
3. tutkittava funktio.

Jos tutkittavat funktiot ovat jatkuvia, tärkeä apuväline on Bolzanon lause:

> Jos jatkuvan funktion *f* arvot *f(a)* ja *f(b)* ovat erimerkkiset, avoimella välillä]*a,b*[on ainakin yksi nollakohta.

Toinen tärkeä tieto sisältyy funktion derivaatan arvoon:

> Jos välillä [*a,b*] derivoituva funktio on aidosti monotoninen eli sen derivaatta ei muuta merkkiään, välillä [*a,b*] voi olla vain yksi nollakohta.

Näitä kysymyksiä on syytä tarkastella ennen seuraavien menetelmien käyttöä. Laskennan jälkeen vastauksen tarkkuus on syytä tutkia niinikään Bolzanon lausetta käyttäen. Olkoon ratkaisuehdokas x. Jos f on jatkuva ja tulo $f(x+\varepsilon)\ f(x-\varepsilon) < 0$, ratkaisun virhe on enintään ε.

1.2.2 Raaka laskenta

Erityisesti jos halutaan löytää pisteestä x_0 lähtien ensimmäinen funktion *f* nollakohta, menetelmäksi voi sopia raaka laskenta: määritetään funktion arvoja valitun askeleen välein, kunnes *f* vaihtaa merkkiä. Tämän ratkaisutavan eräs sovellus on laskea arvoja ja piirtää ne *x - y* –koordinaatistoon, josta katsoja havaitsee nollakohdan.

Algoritmi ”raaka laskenta”:

lähtöarvot: x_0, askel, f(x), (xmax)

tulos: väli, jossa on nollakohta

```
vanha := f(x0)
x:=x0
silmukka alkaa
        x := x+askel
```

```
            uusi := f(x)
            jos (uusi*vanha> 0)
                    vanha := uusi
            jos-ei
                    silmukan lopetusehto:= tosi

            if x>xmax
                    silmukan lopetusehto:=tosi

    silmukka päättyy

            Tulosta x.

Nollakohta on välillä ]x-askel, x].
```

Algoritmin puutteina voidaan pitää sitä, että se vaatii huomattavan paljon laskentaa. Toinen puute on se, että algoritmi ei aina huomaa sellaisia nollakohtia, jotka ovat samalla derivaatan nollakohtia, ja voi hypätä kahden peräkkäisen nollakohdan yli, jos askelväli on suurempi kuin nollakohtien erotus. Jos funktiolla ei ole nollakohtaa, jäädään ikuiseen silmukkaan ilman lopetusarvoa ”*xmax*”. Tämän vuoksi yleensä funktion arvot käydään läpi joltakin tutkittavalta väliltä.

1.2.3 Puolitushaku

Jos funktion arvot tutkittavan välin päätepisteissä ovat erimerkkiset, voidaan käyttää puolitushakua. Puolitushaku sopii esimerkiksi ”raaka laskenta” –menetelmän jatkoksi, koska sen lopputuloksena on juuri vaadittu väli.

Puolitushaussa väli puolitetaan joka askeleella, ja lasketaan siinä funktion arvo. Jos se on nolla, ratkaisu on saavutettu, ja jos se ei ole nolla, saadaan uusi väli, jolla arvot ovat erimerkkiset. Puolitusta jatketaan, kunnes tulos on saatu halutulla tarkkuudella.

Algoritmi ”puolitushaku”

```
Lähtöarvot: alaraja a, yläraja b, funktio f(x), tarkkuus ε.
Tulos: Suljettu väli, jolla ratkaisu on.

Alkuarvot:

Fa := f( a )
Fb := f( b )
jos fa * fb <0

        Silmukka:
```

```
Toista kunnes b – a < ε
        x:= ( a + b ) / 2
        f:= f( x )
        jos f*fa<0
                b := x
                fb := f
        jos f*fb<0
                a:=x
                fa:=f
toisto päättyy.

Tulos: Uudet a ja b. Vastaus on välillä [a,b].
```

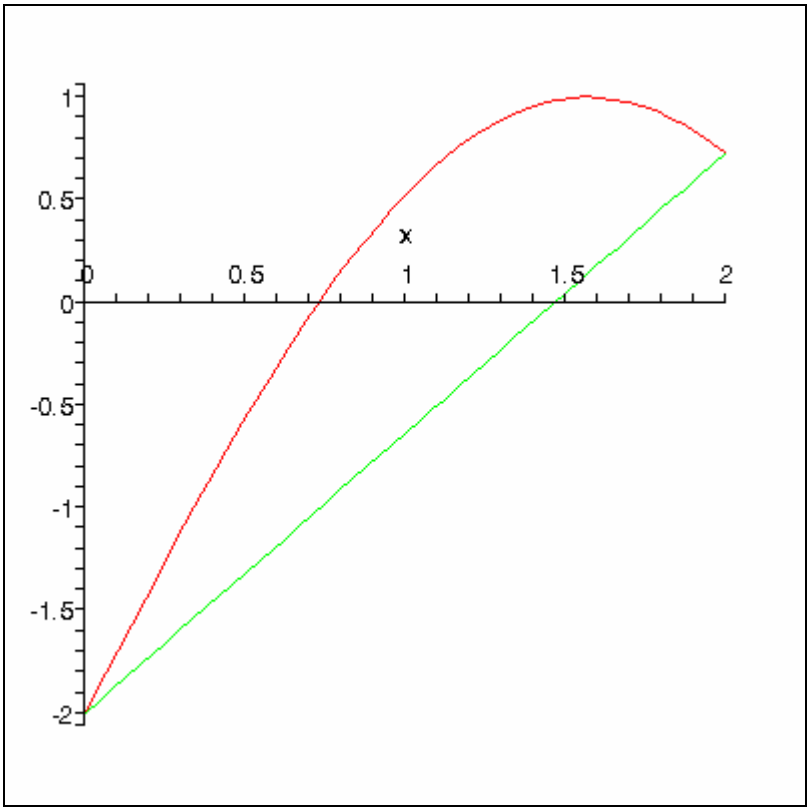
Puolitushaun iterointikierrosten lukumäärä saadaan ratkaistuksi, kun *a*, *b* ja ε on annettu. Koska jokaisella hakukerralla väli puoliintuu, askelten lukumäärä *N* saadaan ratkaisemalla epäyhtälö

(4) $(b-a)\cdot 2^{-N} \le \varepsilon \Leftrightarrow N \ge \frac{\ln(b-a)-\ln(\varepsilon)}{\ln 2}$.

Puolitushaku on kuitenkin harvoin nopein vaihtoehto nollakohdan ratkaisemiseksi, koska sen suppeneminen on vain lineaarista.

1.2.4 Sekanttimenetelmä ja regula falsi

Puolitushakua tehokkaampi menetelmä saadaan, jos puolituksen sijaan seuraava laskentapiste lasketaan käyttäen hyväksi tunnettuja funktion arvoja välin päätepisteissä. Näiden avulla lasketaan seuraava iteroinnin arvo, ja sekanttimenetelmä on rakennettu siten, että se ratkaisee suoraan yhdellä askeleella ensimmäisen asteen yhtälöt. Sillä voidaan hakea arvoja sekä kun ratkaisua haetaan tunnetulta väliltä että kun ei tunneta väliä, jolla nollakohta esiintyy. Sekanttimenetelmässä piirretään pisteiden *(a, f(a))* ja *(b, f(b))* kautta suora, jonka leikkauspiste x-akselin kanssa on uusin iterointiarvo. Menettelyä toistetaan eli kahdesta edellisestä iteroinnin arvosta saadaan aina seuraava. Yksi iterointiaskel on esitetty kuvassa 2.



Sekanttimenetelmässä seuraava x on sekantin ja x-akselin leikkauspiste

Tässä esitetään sekanttimenetelmän muunnos ”regula falsi” tilanteeseen, jossa se korvaa puolitushaun. Regula falsi poikkeaa sekanttimenetelmästä siinä, että laskentapisteissä funktion arvot ovat aina erimerkkiset kuten puolitushaussakin, kun taas sekanttimenetelmässä uusi arvo lasketaan kahden edellisen arvon perusteella riippumatta siitä, jääkö nollakohta näiden väliin.

Algoritmi ”sekantti/regula falsi” välille [*a,b*].

Lähtöarvot: alaraja a, yläraja b, funktio f(x), tarkkuus ε.
Tulos: Suljettu väli, jolla ratkaisu on.

Alkuarvot:

Fa := f(a)
Fb := f(b)

Silmukka:

Toista kunnes b-a<ε

*x:= b - fb * (b – a) / (fb – fa)*
f:= f(x)
*jos f*fa<0*
b := x
fb := f

*jos f*fb<0*
a := x
fa := f

toisto päättyy.

Tulos: Uudet a ja b. Vastaus on välillä [a,b].

1.2.5 Newtonin menetelmä

Sekanttimenetelmässä tarkasteltiin koordinaatiston suoraa, jonka määrittävät kaksi pistettä saadaan muuttujan arvoista *a* ja *b* ja vastaavista funktion arvoista *f(a)* ja *f(b)* eli graafisesti suora on funktion kuvaajan sekantti. Kun pisteet *a* ja *b* lähestyvät toisiaan, erotusosamäärän raja-arvona esiintyy funktion derivaatta. Tällöin sekanttimenetelmä muuntuu Newtonin menetelmäksi. Jos siis funktio *f* on derivoituva, saadaan derivaattaan perustuva iterointikaava funktion nollakohdalle

(5) $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$.

Koska Newtonin menetelmä on niin merkittävä, esitetään sen saamiseen toinenkin malli:

Funktion arvo kohdassa *a* on *f(a)*. Halutaan, että *f(x)* = 0. Siten tarvittava funktion arvon muutos on *f(x) - f(a)*, jota vastaava muuttujan arvojen muutos on *x - a*. Koska derivaatta kuvaa funktion arvojen muutosta muuttujan suhteen ja koska etsitään nollakohtaa *f(x)* = 0, saadaan kun $x \approx a$

(6) $$f'(x) \approx \frac{f(x) - f(a)}{x - a} = \frac{-f(a)}{x - a}.$$

Tästä *x* ratkeaa ja tuloksena on Newtonin iterointikaavan askel alkuarvona *a*.

Newtonin iteroinnissa derivaatta *f'(x)* ei saa saavuttaa arvoa nolla. Lisäksi iterointi voi jäädä värähtelemään derivaatan nollakohdan ympäristöön, jossa ei ole lainkaan juurta, tai värähdellä suurenevalla amplitudilla poispäin juuresta. Näistä ongelmista huolimatta Newtonin menetelmä toimiessaan suppenee neliöllisesti ja on helppo ohjelmoida, joten sitä käytetään lukuisissa sovelluksissa [5 s 844]. Ongelmien kiertämiseen voidaan numeeriseen ratkaisijaan ohjelmoida rajoittimia, jotka hidastavat suppenemista normaaleissa tapauksissa, mutta poistavat vaikeiden tapausten aiheuttamia ongelmia.

Algoritmi Newton

```
Alkuarvot: f, f', alkuarvaus a, tarkkuus e,
Epäonnistumisehtona iterointikertojen maksimilukumäärä N.

Toista N kertaa:
        vanha:=a
        f  :=f(a)
                fp :=f'(a)
                jos fp=0
                  tulosta: "Epäonnistui f'=0"
                  LOPETA
        a := a – f / fp
        jos |vanha-a| < e
                  tulosta: a
                  LOPETA.
Toisto päättyy.

Tulosta: "ei ratkaisua N askeleella".
```

1.2.6 Kiintopistemenetelmä

Kiintopistemenetelmässä yhtälö saatetaan muotoon $x = g(x)$, josta saadaan suoraan iterointikaava

(7) $x_{i+1} = g(x_i)$.

Kiintopistemenetelmälle voidaan johtaa suppenemisehto:

> Jos välillä $[a,b]$ on yhtälön $x = g(x)$ juuri ja $|g'(x)| < K < 1$, niin tämä juuri on yksikäsitteinen ja kiintopisteiterointi suppenee.

Todistetaan yksikäsitteisyyttä koskeva osa:

> Yhtälön $g(x) = x$ juuri on sama kuin funktion $f(x) = g(x) - x$ nollakohta. Tutkitaan funktion f derivaattaa. $f'(x) = g'(x) - 1 < 0$, koska $|g'(x)| < 1$. Siten f on monotoninen välillä $[a,b]$, joten sillä voi olla vain yksi nollakohta.

Hahmotellaan ajatus suppenevuuden osoittamiseksi:

> Olkoon juuri x^* eli pätee kaava $x^* = g(x^*)$. Olkoon poikkeama oikeasta tuloksesta
>
> $$e_i = |x_i - x^*|,$$
>
> jolloin iteraation $i+1$ poikkeama on
>
> $$e_{i+1} = |g(x_i) - x^*| = |g(x_i) - g(x^*)|.$$
>
> Osamäärälle pätee väliarvolauseen mukaan
>
> $$\frac{e_{i+1}}{e_i} = \left| \frac{g(x_i) - g(x^*)}{x_i - x^*} \right| = |f'(t)| \leq K < 1,$$
>
> joten $e_{i+1} < K \cdot e_i$.
>
> Olkoon virhe ensimmäisellä askeleella e_0. Tästä saadaan, että n askeleen jälkeen virhe
>
> $$e_n < K^n \cdot e_0 .$$
>
> Kun $n \to \infty$, niin $K^n \to 0$, joten menetelmä suppenee [5, s.840].

Mitä lähempänä K on nollaa, sitä nopeampaa on suppeneminen. Hans-Jochen Bartsch [7, s.104] suosittaa käyttämään muita menetelmiä, jos $K > 0{,}8$, koska tällöin kiintopistemenetelmä on liian hidas. Erityisen kannattavaa käyttö on, jos $0 < K < 0{,}2$.

1.3 Numeerinen integrointi

1.3.1 Perusteita

Määrätyn integraalin numeerinen ratkaisu perustuu pääosin ajatukseen, että laskemalla sopivasti funktion arvoja integrointivälin pisteissä saadaan arvio funktion ja *x*-akselin väliin jäävästä pinta-alasta eli määrätyn integraalin arvosta. Funktion integraalin likiarvo saadaan kaavasta

$$(8)\quad \int_a^b f(x)dx = (b-a)\cdot\sum_{i=1}^{n} w_i f(x_i)\,.$$

Summalausekkeen voidaan ajatella graafisesti vastaavan funktion keskimääräistä korkeutta x –akseliin verrattuna. Se voidaan laskea eri muuttujan arvoilla, ja eri kohdista integrointiväliä laskettavat osat painotetaan eri suuruisilla painokertoimilla. Kun menetelmä kirjoitetaan kaavaksi, integrointiväli kerrotaan joskus osaksi painokertoimia.

Adaptiivisissa menetelmissä integroinnin kestäessä integrointiväliä jaetaan usein osiin siten, että lopputulosta arvioitaessa funktion kulun kannalta hankalissa kohdissa funktion arvo lasketaan useammin kuin muualla.

Tässä tarkastellaan monista eri integrointivaihtoehdoista vain Newtonin-Cotesin kaavoja ja Gaussin integrointia.

1.3.2 Newtonin – Cotesin kaavat

Newtonin - Cotesin kaavoissa integrointivälillä muuttujan arvot jaetaan tasaväleihin. Laskentaan otetaan funktion arvot integroinnin päätepisteissä *a* ja *b* sekä tasavälisen jaon välipisteissä. Funktioiden arvoista laskettava painotettu keskiarvo kerrotaan integroimisvälillä ja tuloksena on integraalin likiarvo. Painokertoimet lasketaan olettaen, että pisteiden kautta kulkisi sopivan asteluvun polynomi.

Puolisuunnikassääntö

Yksinkertaisin kaava saadaan, kun funktio oletetaan vakioksi laskentapisteensä ympäristössä. Tällöin saadaan *n*:lle laskentapisteelle kaava, jossa reunapisteet saavat painokertoimen ½ ja sisäpisteet 1, koska vain reunapisteen toisella puolella on integroitavaa aluetta.

(9) $\int_a^b f(x)dx \approx \Delta x \cdot \left[\sum_{i=1}^{n} f(x_i) - \frac{f(a)+f(b)}{2} \right].$

Menetelmää kutsutaan puolisuunnikassäännöksi, koska sama tulos saadaan olettamalla tasavälisten iterointipisteiden väliin suora. Tämä mielenkiintoinen ominaisuus pätee yleisemminkin: Newtonin-Cotesin kaavoissa astetta alhaisempaan parilliseen polynomiin perustuva iterointikaava integroi täsmälleen oikein myös astetta korkeamman asteen polynomin myös mm. Simpsonin säännöllä. [5, s.874].

Puolisuunnikassäännön virhe saadaan kaavasta

(10) $E = -\frac{(b-a)^3}{12n^2} f''(t) = -\frac{(b-a)}{12}(\Delta x)^2 f''(t).$

Virhekaavassa n on jakovälien lukumäärä, Δx yhden integrointivälin pituus ja t on tietty luku integrointiväliltä $[a,b]$.

Käsitellään tässä vain virhekaavan johtamisen idea. [6. s.192-195] Virhettä arvioitaessa voidaan nojautua interpolaatiopolynomien virheisiin. Tutkitaan erotusta $e(x)=f(x)-p(x)$, joka kuvaa sitä virhettä, joka integroitaessa syntyy polynomin käyttämisestä funktion korvikkeena. Välillä $[a,b]$ kahdesti derivoituvan funktion virheeksi saadaan

(11) $e(x) = (x-a)(x-b)\frac{f''(t)}{2!}.$

Tässä t on tietty luku väliltä $[a,b]$. Integraalin virhe yhden askeleen matkalla on sama kuin virheen integraali. Koska $f''(t)$ on tuntematon vakio, se voidaan siirtää integroinnin ulkopuolelle. Kun merkitään $b = a + h$, ja tehdään sijoitus $z = x - a$ saadaan

(12) $\int_a^{a+h} (x-a)(x-a-h)dx = \int_0^h z(z-h)dz = \frac{1}{6}h^3.$

Kun tämä kerrotaan vakio-osalla, päädytään tulokseen integraalin virheestä E.

Kun tutkitaan virheen suurinta itseisarvoa, voidaan valita toisen derivaatan itseisarvoltaan suurin arvo. Tämän ratkaisu voi kuitenkin olla työlästä, minkä vuoksi on usein käytännöllisempää tehdä likimääräinen virhearvio, joka perustuu eri askelvälillä tehtyihin laskelmiin.

Jos jakovälien lukumäärä kaksinkertaistetaan eli askelpituus Δx puolitetaan, saadaan likimääräiset virhekaavat

(13) $E_n = K_n (\Delta x)^2$, $E_{2n} = K_{2n} (\Delta x/2)^2$.

Kaavassa K edustaa virhetermin loppuosaa, joka ei riipu luvusta Δx. Jos toisen derivaatan arvo $f''(t)$ oletetaan likimain vakioksi kummallakin integrointivälillä, niin $K_n \approx K_{2n}$. Siten $E_n \approx 4\, E_{2n.}$

Olkoot I integraalin oikea arvo ja J_n sekä J_{2n} numeeriset likiarvot. Siten saadaan yhtälöpari

$$(14)\quad \begin{matrix} I - J_n = E_n \approx 4E_{2n} \\ I - J_{2n} = E_{2n} \end{matrix} .$$

Vähentämällä puolittain ja ratkaisemalla E_{2n} saadaan virhearvioksi

$$(15)\quad \left|E_{2n}\right| \approx \left|\frac{J_{2n} - J_2}{3}\right| .$$

Tämäntyyppiset kaavat ovat käyttökelpoisia ohjelmoitaessa numeerisia integrointeja, koska niissä virhearvio perustuu ohjelman suorituksen aikana laskettaviin likiarvoihin. Siten niiden avulla voidaan laatia automaattisesti askelpituutta säätäviä algoritmeja, jotka joko adaptiivisesti säätävät askelväliä koko integrointivälillä laskennan kestäessä tai mahdollistavat askeleen karkean mitoituksen ennen tarkkaa integrointia integrointiväliltä lasketun otoksen avulla. Tällöin tehtävässä on etukäteen valittava haluttu virheen suuruusluokka ε.

Askelpituudelle Δx tai askelten lukumäärälle n saadaan annetulle virhetasolle ε ratkaisuvaihtoehdot. Jos toisen derivaatan arvolle tunnetaan yläraja, askelpituus saadaan yhtälöstä

$$(16)\quad \Delta x = \sqrt{\frac{12\varepsilon}{(b-a)\cdot \max(|f''(t)|)}} .$$

Jos virhe E_n on määritetty laskennallisesti n:lle jakovälille, saadaan tarkkuutta ε edellyttävä jakovälien lukumäärän arvioksi n_ε suhteen perusteella kaavasta

$$(17)\quad n_\varepsilon = \sqrt{\frac{E_n}{\varepsilon_n}} \cdot n .$$

Askelväli tai jakopisteiden lukumäärä saadaan yhtälöstä $n \cdot \Delta x = (b - a)$, jos toinen on ratkaistu. Jos laskenta perustuu numeerisiin likiarvoihin, virhetarkastelussa kannattaa käyttää varmuuskerrointa, joka paikkaa likiarvoisen virhekaavan käyttöä.

Simpsonin sääntö

Olettamalla kolmen pisteen väliin paraabeli saadaan Keplerin integrointikaava, joka tunnetaan parhaiten yleisessä muodossa Simpsonin sääntönä. Sillä saadaan tarkasti integroiduksi enintään kolmannen asteen polynomit.

Nimistä kommenttina todettakoon, että saksalainen lähdemateriaali käyttää Keplerin nimeä mm. ”Keplerische Fassregler” [7, 15. painos s.336]. Anglosaksinen kirjallisuus näyttää käyttävän vain Simpsonin nimeä, ja saksalainen Rainer Kress englanninkielisessä tekstissään käyttää nimeä Simpsonin sääntö ”Simpson’s rule”, joskin toteaa Keplerin tunteneen sen jo vuonna 1612 ja Simpsonin 1743 [6, s.192].

Keplerin kaava on

$$(18)\ \int_a^b f(x)dx = (b-a)\cdot\left[\frac{f(a)+4f(\frac{a+b}{2})+f(b)}{6}\right].$$

Jos halutaan tarkempi tulos, jaetaan integrointiväli puoliksi, ja lasketaan kumpikin väli Keplerin kaavalla. Tällöin painokertoimet ovat kummassakin puoliskossa 1 4 1. Koska keskimmäistä funktion arvoa käytetään kummankin jakovälin laskennassa, saadaan funktion painokertoimiksi osoittajaan 1 4 2 4 1.

Kun Keplerin kaavalla laskettuja integraaleja lasketaan yhteen *n* kappaletta, saadaan yleinen Simpsonin kaava

$$(19)\ \int_a^b f(x)dx = \frac{(b-a)}{\sum_{i=1}^{n} p_i}\cdot\sum_{i=1}^{n} p_i f(x_i),$$

jossa painot p_i saavat arvot 1 4 2 4 … 4 2 4 1.

Tämä muoto on helppo kirjoittaa taulukkolaskentaan. Ohjelmoinnin kannalta on kuitenkin helpointa tehdä rutiini ”Kepler”, jota toistetaan *n* kertaa, jolloin Simpsonin sääntö saa muodon

$$(20)\ \int_a^b f(x)dx = \tfrac{1}{6}\left[\sum_{i=1}^{n}(x_i - x_{i-1})(f(x_{i-1})+4f(\tfrac{x_i+x_{i-1}}{2})+f(x_i))\right].$$

Simpsonin säännön virhekaava saadaan johdetuksi kuten puolisuunnikassäännönkin. Kun integrointi jaetaan n tasaväliseen osaan, joissa kussakin käytetään Keplerin kaavaa [6, s.197], kokonaisvirheen yläraja saadaan yhtälöstä

$$(21)\ E_n = -\frac{(b-a)^5}{180(n)^4} f^{(4)}(t) = -\frac{(b-a)}{180}(\Delta x)^4 f^{(4)}(t) \quad .$$

Tuloksesta nähdään, että virheen yläraja saadaan funktion *f* neljännen derivaatan maksimiarvosta ja funktion arvojen laskentapisteiden etäisyyden Δx neljännestä potenssista. Siten puolisuunnikassääntöä vastaavasti saadaan virhekaavat.

Kun lasketaan jakovälimäärällä *n* ja 2*n,* virhe on

$$(22)\ \left|E_{2n}\right| \approx \left|\frac{J_n - J_{2n}}{15}\right| \quad .$$

Askelpituudelle Δx tai askelten lukumäärälle *n* saadaan virhearvioiden perusteella ratkaisuvaihtoehdot annetulle virhetasolle ε ratkaisut:

Jos neljännen derivaatan arvolle tunnetaan yläraja, askelpituus saadaan kaavasta

$$(23)\ \Delta x = \sqrt[4]{\frac{180\varepsilon}{(b-a)\cdot \max(|f^{(4)}(t)|)}} \quad .$$

Jos virhe E_n on määritetty laskennallisesti *n*:lle jakovälille, saadaan tarkkuutta ε edellyttävä jakovälien lukumäärä n_ε suhteen perusteella kaavasta

$$(24)\ n_\varepsilon = \sqrt[4]{\frac{E_n}{\varepsilon}} \cdot n \, .$$

Simpsonin kaavan virhe on siten merkittävästi pienempi kuin puolisuunnikassäännön.

Kun sovitettavan polynomin astelukua nostetaan, painokertoimet muuttuvat ja tarkkuus paranee. Toisaalta astelukua nostettaessa kunkin likiarvon laskemiseen tarvittavien funktion arvojen lukumäärä kasvaa, ja polynomin arvojen ”värähtely” laskentapisteiden välillä voi tuottaa ongelmia. Siten käytännössä asteluvun nostaminen loputtomiin ei tuota lisähyötyä, vaan suurempi etu saadaan ohjelmoimalla yksinkertaisempaan kaavaan funktion kulun mukaan säätyviä adaptiivisia rakenteita. Painokertoimet ja kaavat voi hakea esimerkiksi Bartsch:n taulukkokirjasta [7, s.427-429].

1.3.3 Gaussin integrointikaava

Gaussin integroinnissa funktion arvoja ei lasketa tasavälisessä jaotuksessa, vaan jakovälin sisällä painokertoimet ja laskentapisteet jaetaan optimaalisesti. Gauss julkaisi menetelmän vuonna 1814 otsikolla ”Methodus nova integralium valores per approximationem inveniendi” [6, s.189].

Menetelmän perusajatus on valita pisteet väliltä [-1,1] optimaalisesti siten, että tuloksena integroidaan oikein mahdollisimman korkean asteen polynomi. Käytännössä laskentapisteet ja niiden painokertoimet on taulukoitu ja saatavissa monista taulukko- tai oppikirjoista, esim. [7, s.429-430], [5 s.877-878]. Pisteet saadaan ratkaisuna Legendren polynomien nollakohdista.

Gauss osoitti, että menetelmällä saadut painokertoimet integroituvat oikein polynomille astetta $2n$ - 1, kun kiinteillä pisteillä päästiin vain tasolle n - 1 [6s.189-192]. Kun yksittäistä ratkaisua haetaan, sopivalla sijoituksella pitää joko integrointiväli [a,b] muuttaa väliksi [-1,1] tai Gaussin algoritmin laskentapisteet integrointiväliltä [-1,1] voidaan muuttaa välille [a,b].

Jos halutaan saada laskentapisteet ja painoarvot välille [a,b], tulee tehdä sijoitus

$$(25)\ z = \frac{b-a}{2}x + \frac{a+b}{2}.$$

Tämä sijoitus muuttaa integrointirajat välistä [-1,1] välille [a,b]. Vastaavat z:n laskentapisteet saadaan sijoittamalla taulukoidut laskentapisteet x_i kaavaan ja kertomalla saadut tulokset vastaavalla painokertoimella. Gaussin integroinnin 6 pisteessä sisältävä Matlab –koodi on liitteenä A1.

Ohessa taulukkolaskennalla toteutettu yhden askeleen ratkaisu integraalille e^x. Käyttäjä syöttää keltaisiin ruutuihin lähtöarvot ja ja funktion. Lihavoidut luvut ovat Gaussin menetelmän laskentapisteet ja niitä vastaavat painokertoimet.

Kuuden pisteen Gaussin algoritmi taulukkolaskentaohjelmaan

Integrointirajat		a= 0		b= 2	
Apusuureet integrointivälin muuttamiseksi		M=(a+b)/2	h=(b-a)/2		
		1	1		
Alkuperäiset laskentapisteet väliltä [-1,1]:					
-0.932469514203152	**-0.661209386466265**	**-0.238619186083197**	**0.238619186083197**	**0.661209386466265**	**0.932469514203152**
Muunnoksen jälkeiset laskentapisteet väliltä [a,b]:					
0.067530486	0.338790614	0.761380814	1.238619186	1.661209386	1.932469514
Alla yläpuolisia x:n arvoja vastaavat f(x):n arvot					
1.069862875	1.403249493	2.141230821	3.450845198	5.265675232	6.90654501
Kutakin laskentapistettä vastaavat painokertoimet (Ai) välillä [0,1]:					
0.17132449237917	**0.36076157304814**	**0.46791393457269**	**0.46791393457269**	**0.36076157304814**	**0.17132449237917**
Painokertoimien, luvun h ja funktion arvojen tulo:					
0.183293714	0.506238494	1.001911739	1.614698554	1.89965328	1.183260318
Integroinnin tulos summana yläpuoliselta riviltä					
6.3890560989E+00					

Tulos on hämmästyttävän hyvä. Se poikkeaa oikeasta vasta kahdennessatoista desimaalissa.

Teoreettinen virhearvio saadaan samaan tapaan kuin Newtonin-Cotesin kaavoilla, mutta n laskentapisteen osalta virhe on verrannollinen askelpituuden $2n+1$ potenssiin. [7, s.429] Siten esimerkiksi yllä olevan esimerkin tilanteen 6 pisteen Gaussin integraalissa paikallinen virhe (local error) on verrannollinen askelpituuden kolmanteentoista potenssiin. Siten askelpituuden puolittamisella saadaan likimääräiseksi virheeksi

$$(26)\ E_{h/2} \approx \frac{|J_{h/2} - J_h|}{2^{12} - 1}.$$

Askelpituudelle tunnetun likimääräisen virheen E ja annetun virheen suuruusluokan ε välille saadaan johdetuksi niin ikään virhekaava

$$(27)\ h_\varepsilon = \sqrt[12]{\frac{\varepsilon}{E_h}} \cdot h.$$

Usein likimääräisen kaavan nimittäjään laitetaan huomattava turvamarginaali. Tässä tutkimuksessa ohjelmoidussa kaavassa käytettiin esimerkiksi arvoa 1023. Samoin

juurikaavassa (27) luku h_ε lasketaan käyttäen kymmenettä juurta turvamarginaalin saamiseksi.

Gaussin menetelmän tarkat kertoimet kuuden pisteen kaavalle saa yllä olevasta taulukosta. Ohessa pisteet ja niiden painokertoimet Bartsch:n taulukkokirjan [7, s.429] mukaan:

Laskentapisteitä	*Integrointipiste x:*	*Painokerroin A*
1	$x_0=0$	$A_0=2h$
2	$x_{0;1}= \pm 1 / \sqrt{3}$	$A_{0;1}=h$
3	$x_{0;2}= \pm \sqrt{0,6}$	$A_{0;2}=5/9\ h$
	$x_1=0$	$A_1=8/9\ h$
4	$x_{0;3}= \pm 0,86113631\ h$	$A_{0;3}=0,34785485 \cdot h$ $x_{1;2}= \pm 0,33998104\ h$ $A_{1;2}=0,65214515 \cdot h$
5	$x_{0;4} = \pm 0,90617985\ h$	$A_{0;4}=0,23692689 \cdot h$
	$x_{1;3} = \pm 0,53846931\ h$	$A_{1;3}=0,47862867 \cdot h$
	$x_2 = 0$	$A2=0,568 \cdot h$

Kuten taulukkolaskennalla ylle laskettu esimerkki osoittaa, menetelmän virhe voi olla on pienempi kuin yllä olevassa taulukossa olevat 8 desimaalin likiarvot. Siksi laskentapohjaan kirjoitettiin painokertoimet 15 desimaalilla, mikä muodostaa menetelmän tarkkuudelle rajan. Vaikka käytettävissä on muitakin integrointimenetelmiä, tässä tutkimuksessa on pääosin käytetty yllä olevaa kuuden pisteen Gaussin integrointikaavaa.

2. Differentiaaliyhtälöistä

2.1 Differentiaaliyhtälön käsite

Differentiaaliyhtälössä esiintyy tutkittavan funktion derivaatta *y'(t)*. Tehtävänä on yleisessä tapauksessa ratkaista kaikki ne funktiot *y(t)*, jotka toteuttavat differentiaaliyhtälön. Ensimmäisen kertaluvun differentiaaliyhtälöissä esiintyy ensimmäinen derivaatta ja ne ovat yleensä saatettavissa muotoon

(28) $y'(t) = f(t, y)$.

Esimerkiksi differentiaaliyhtälön:

(29) $y'(t) = y$

yhtenä ratkaisuna on eksponenttifunktio e^t, koska sen derivaatta on sama kuin funktio itse.

Jos yhtälön oikea puoli voidaan kirjoittaa kahden funktion tulona, jossa toisena muuttujana esiintyy *y* ja toisena *t*, puhutaan separoituvasta differentiaaliyhtälöstä, joka kirjoitetaan yleensä muodossa

(30) $y'(t) = f(t) \cdot g(y)$.

Lineaarinen differentiaaliyhtälö voidaan kirjoittaa muodossa

(31) $y'(t) + a(t) \cdot y(t) = b(t)$.

Jos differentiaaliyhtälössä esiintyy enintään toista derivaattaa *y''*, puhutaan toisen kertaluvun differentiaaliyhtälöistä ja jos enintään kolmatta derivaattaa, kolmannen kertaluvun differentiaaliyhtälöstä ja niin edelleen.

Tässä työssä käsitellään ensisijaisesti alkuarvo-ongelmia, joissa differentiaaliyhtälön lisäksi on annettu tutkittaville muuttujille alkuarvot y_0 ja t_0, ja halutaan saada joko *t* tai *y(t)* ratkaistuksi, kun toinen niistä on tunnettu.

Tällaisia esiintyy fysikaalisissa sovelluksissa, joista esimerkkinä on kaapelin lämpeneminen, jota kuvaava differentiaaliyhtälö on muotoa

$$(32)\ \frac{dT}{dt} = \alpha(T - T_{ymp}) + \sigma \cdot \varepsilon \cdot (T^4 - T_{ymp}^4)\ .$$

Yhtälössä kaapelin lämpötilan T muutos ajan t suhteen riippuu ympäristön lämpötilasta T_{ymp}. Kaapelin lämpötilaan vaikuttaa konvektio, jonka suuruutta kuvaa verrannollisuuskerroin α ja säteily, jota kuvaavat termit σ ja ε. Tehtävänä oli tilanteen mukaan joko ratkaista, missä ajassa saavutetaan annettu lämpötila T, tai mikä on lämpötila ajan t kuluttua, kun alkuaika on t_0 ja alkulämpötila T_0. Koska monissa fysikaalisissa sovelluksissa muuttajana on usein aika, käytetään muuttujana tekstissä x-kirjaimen ohella ajan symbolikirjainta t.

Toisen kertaluvun differentiaaliyhtälöt voidaan ajatella esiintyvän muodossa

$$(33)\ y''(t) = f(y', y, t)\ .$$

Toisen kertaluvun differentiaaliyhtälöiden alkuarvotehtävien ratkaisemiseksi tarvitaan alkuarvot yleensä sekä funktiosta $y(t_0)$ että sen derivaatasta $y'(t_0)$. Tässä työssä käsitellään lähinnä niitä toisen kertaluvun differentiaaliyhtälöitä, jotka on ratkaistavissa peräättäisillä tai sisäkkäisillä integroinneilla yhden muuttujan suhteen.

2.2 Differentiaaliyhtälön ratkaiseminen

2.2.1 Suuntakenttä ja kulkukaavio

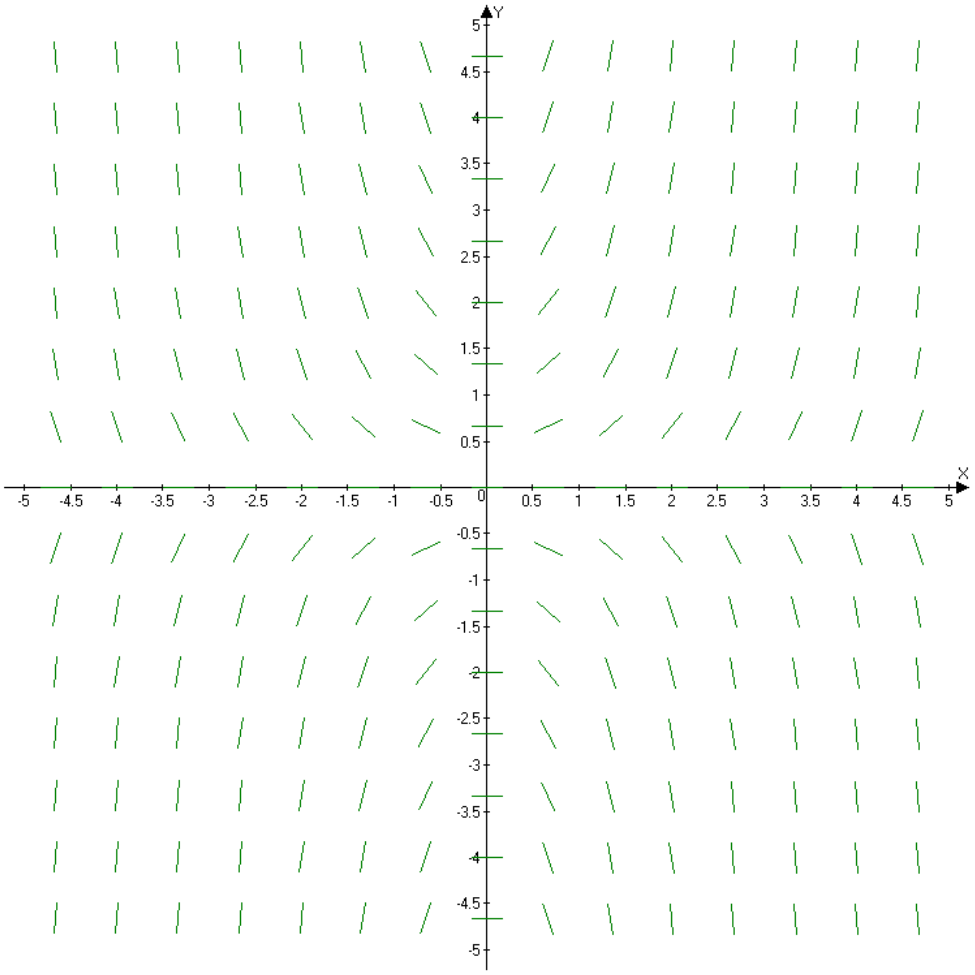
Karkea käsitys differentiaaliyhtälön kulusta saadaan tarkastelemalla yhtälön 28 oikeaa puolta: kun se on positiivinen, funktio on kasvava ja kun se on negatiivinen, vähenevä.

Jos derivaatta $y'(t) = f(x,y)$ on nolla jollain y:n vakioarvolla, kyseessä on ns. tasapainopiste. Siten ratkaisemalla käyrän $y' = f(x,y) = 0$ kuva x - y -tasolla helpottaa merkittävästi funktion kulun tarkastelua, kun kasvavuudet ja vähenevyydet ovat selvillä.

Esimerkki 1:

Tutkitaan differentiaaliyhtälöä $y' = f(x,y) = x \cdot y$. Tulon nollasäännön mukaan koordinaattiakselit ovat käyrän $f(x,y) = 0$ ratkaisut, ja tulon merkkisäännöstä nähdään, että derivaatta on positiivinen I ja III neljänneksessä ja negatiivinen II ja IV x - y -

tason neljänneksessä. Kuvaan voidaan merkitä laskentapisteiden kautta kulkevan funktion y ratkaisukäyrän tangentin suuntaisia janoja eli käytännössä funktion suuntaa kuvaavat viivat, jolloin saadaan ns. suuntakenttä.



Suuntakenttä $y' = y \cdot x$

2.2.2 Olemassaolo- ja yksikäsitteisyyslause

Differentiaaliyhtälön numeerisen ratkaisemisen kannalta merkittävää on mahdollisen ratkaisun olemassaolo ja yksikäsitteisyys. Tämän tarkastaminen ennen numeerista integrointia on välttämätöntä, koska jos samoista alkuarvoista lähtien lähtee useita ratkaisuja, numeeriset menetelmät pitää valita siten, että ratkaisu edustaa fysikaalisesti mielekästä tulosta.

Tutkitaan alkuarvotehtävää $y' = f(x,y)$, $y(x_0) = y_0$ alueella A: $|x - x_0| < a$, $|y - y_0| < b$.

Olemassaololause:

Jos f on jatkuva ja rajoitettu eli on olemassa $K \geq |f(x,y)|$ alueella A, differentiaaliyhtälöllä $y' = f(x,y)$ on ainakin yksi ratkaisu.

Yksikäsitteisyyslause:

Jos lisäksi funktion $f(x,y)$ osittaisderivaatta y:n suhteen on jatkuva ja rajoitettu eli on olemassa M siten, että

$$\left|\frac{\partial f}{\partial y}\right| \leq M$$

kaikissa alueen A pisteissä, ratkaisuja on täsmälleen 1.

Erityisesti fysikaalinen mallinnus ja siinä tehtävät yksinkertaistukset voivat johtaa malliin, jonka ratkaisun arviointi etukäteen voi olla työlästä. Tällöin olemassaolon ja yksikäsitteisyyden tutkiminen ennen aikaa vieviä simulointeja on erittäin hyödyllistä. [5 s.53-54]

2.3 Numeerinen ratkaiseminen

2.3.1 Eulerin menetelmä

Yksinkertaisin numeerinen menetelmä differentiaaliyhtälön $y'=f(t,y)$ ratkaisuun on Eulerin menetelmä. Koska sovelluksissa tehtävät ratkaistaan usein ajan suhteen, muuttujana käytetään symbolia t. Derivaatalle pätee

$$(34)\ y'(t) \approx \frac{\Delta y}{\Delta t} \qquad eli \qquad \Delta y \approx y'(t)\Delta t\ .$$

Siten annetulla muuttujan t muutoksella Δt sitä vastaava funktion arvon muutoksen likiarvo Δy saadaan suoraan lasketuksi.

Kun alkuarvot t_0 ja y_0 tiedetään, y' voidaan laskea yhtälöstä $y'= f(t,y)$. Siten alkuarvoista saadaan seuraava (t,y) -pari lasketuksi. Toistamalla menettelyä valitulle lyhyellä askeleelle Δt saadaan likimääräinen tulos funktion y arvoista pisteissä t_i.

Koska Eulerin menetelmässä funktion y arvona on alkuarvo $y_0 = y(t_0)$ koko askelpituuden Δt matkan, tulos sisältää kumuloituvan virheen. Tämän vuoksi Eulerin menetelmää on pidettävä johdantona parempiin numeeriseen ratkaisumenetelmiin, eikä Eulerin menetelmään perustuvien algoritmien ohjelmointia todellisten ongelmien ratkaisuun voi pitää välttämättä mielekäänä, vaan ohjelman voi suoraan kirjoittaa

esimerkiksi Rungen-Kuttan menetelmälle. Koska kuitenkin Eulerin menetelmä on opetuksellisesti helppo tapa tutustua differentiaaliyhtälöiden perinteisiin numeerisiin ratkaisumenetelmiin, tarkastellaan myös sen virhearvioita. Tuloksia vastaavat virhekaavat voidaan johtaa muillekin numeerisille menetelmille.

Kootaan Eulerin menetelmä *y(t)*:n ratkaisemiseksi alkuarvoista t_0 ja y_0 algoritmiksi:

Lähtöarvot: f(t,y), alkuarvot t_0, y_0 lopetusarvo t, askelmäärä N.
Tulos: y(t)

Alustetaan muuttujat:

$\Delta t := (t-t_0) / N$
$y := y_0$
$t_i := t_0$

Silmukka:

toista N kertaa
$y := \Delta t\, f(t_i,y) + y$
$t_i := t_i + \Delta t$
silmukka loppu

Tulosta y(t).

Tässä käytettiin menetelmää, jossa etukäteen valittiin askelmäärä.

Seuraavaksi esitetään adaptiivisen eli laskennan aikana säätyvän algoritmin perusteet differentiaaliyhtälön likimääräisen tuloksen laskemiseen. Algoritmi käyttää hyväksi Eulerin menetelmän virhearvioita, joka voidaan perustella esimerkiksi Taylorin sarjalla.

Taylorin sarja funktiolle *f(x)* pisteen x_0 ympäristössä saadaan yhtälöstä

$$(35)\ f(x) \approx p(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \ldots + \frac{f^n(x_0)(x - x_0)^n}{n!}$$

Kun polynomi katkaistaan potenssin *n* kohdalta, sen virhe $\varepsilon = p(x) - f(x)$ on tunnetusti laskettavissa kaavasta

$$(36)\ \varepsilon(x) = \frac{f^{n+1}(a)(x - x_0)^{n+1}}{(n+1)!},$$

missä *a* on jokin luku väliltä lukujen x_0 ja *x* välistä.

Koska Eulerin menetelmässä funktion *f(t,y)* Taylorin polynomi on katkaistu ensimmäisen potenssin kohdalta, menetelmän virhettä arvioidaan toista astetta olevan virhetermin perusteella. Siten funktiolle *y* saadaan likimääräinen arvio

(37) $y = y_0 + y'(t_0)\,\Delta t + \tfrac{1}{2}\,y''(a)\,(\Delta t)^2$.

Siitä voidaan arvioida laskentamenetelmän virhettä sekä toisen derivaatan että askelvälin perusteella. Jos itseisarvon $|y''|$ maksimiarvo M tunnetaan, saadaan virherajaksi

(38) $\varepsilon = \tfrac{1}{2}\,M\,(\Delta t)^2$.

Jos hyväksyttävä virhe ε on annettu, saadaan ratkaistuksi askelpituus

(39) $\Delta t = \sqrt{\dfrac{2\varepsilon}{M}}$.

Usein helpommin ohjelmoitava ratkaisu saadaan käyttämällä laskettuja likiarvoja. Koska virheessä esiintyy termi Δt korotettuna toiseen, laskemalla tulos jakovälillä Δt ja puolikkaalla jakovälillä $\tfrac{1}{2} \cdot \Delta t$, jälkimmäisessä tuloksen virhe putoaa likimain neljäsosaan. Olkoon oikea tulos *y*, laskennan tulos kokoaskeleelle y_1, puoliaskeleelle $y_{1/2}$, kokoaskeleen virhe ε_1 ja puolitetun askeleen avulla saadun tuloksen virhe $\varepsilon_{1/2}$. Tällöin saadaan yhtälöpari

(40) $\begin{aligned} y - y_1 &= \varepsilon_1 \\ y - y_{1/2} &= \varepsilon_{1/2} \end{aligned}$.

Kun yhtälöt vähennetään puolittain toisistaan, tuntematon *y* häviää pois, ja saadaan numeeristen kaavojen erotukseksi virheiden erotus

(41) $-y_1 + y_{1/2} = \varepsilon_1 - \varepsilon_{1/2} \approx 4\,\varepsilon_{1/2} - \varepsilon_{1/2} = 3\,\varepsilon_{1/2}$.

Siten nähdään, että tarkemman numeerisen laskennan likiarvon virhe on noin kolmasosa eri menetelmien virheiden erotuksesta. Siten hyväksyttävä virhetaso saavutetaan joko puolittamalla askelta kunnes virheraja on saavutettu tai laskennallisena likiarvona.

Olkoot ensimmäisen puolituksen jälkeen saadut virhearvot e_1 ja askel h_1. Haluttu tarkkuus olkoon e_2 ja sitä vastaava askelpituus h_2, joka saadaan tällöin kaavasta

(42) $h_2 = h_1\sqrt{\frac{e_2}{e_1}}$.

Eulerin menetelmällä laskevan tietokoneohjelman sisään voidaan ohjelmoida edelliset virhettä ja askelväliä arvioivat laskutavat. Tällöin paikalliselle virheelle sopivassa yllä olevassa askelpituuden kaavassa juuren astelukua kannattaa pudottaa yhdellä [5, s.951], koska se kuvaa paikallista virhettä. Kumuloituva virhe pudottaa globaalin virheen astetta alemmaksi. Siten tässä neliöjuuri poistetaan, muilla menetelmillä esim. neljäs juuri muutetaan kolmanneksi juureksi.

Eulerin menetelmän tarkkuutta voidaan parantaa monin tavoin, joille on tyypillistä laskea ensin ennuste funktion *y* arvosta ja laskea tämän ennusteen perusteella sama väli uudelleen. Englannin kielellä näitä kutsutaan termillä ”predictor-corrector methods”.

2.3.2 Rungen-Kuttan menetelmä

Numeerisissa sovelluksissa ja esimerkiksi laskimissa usein käytettävä menetelmä on Rungen-Kuttan menetelmä, joka on nopea, helposti ohjelmoitava eikä tarvitse alkuarvauksia lähteäkseen liikkeelle. Yleisin muoto on neljännen asteen metodi, jossa kunkin askelvälin sisällä lasketaan neljä kertaa funktion *f(x,y)* arvo, joista otetaan painotettu keskiarvo [5, s.945-950].

Tuloksena saadaan algoritmi:

Lähtöarvot: f(t,y), alkuarvot t_0, y_0 lopetusarvo t, askelmäärä N.
Tulos: y(t)

Alustetaan muuttujat:

$\Delta t = (t-t_0)/N$
$y = y_0$
$t_i = t_0$

Silmukka:

toista N kertaa
$k1 := \Delta t \cdot f(t_i , y)$
$k2 := \Delta t \cdot f(t_i + \Delta t / 2 , y + k_1 / 2)$
$k3 := \Delta t \cdot f(t_i + \Delta t / 2 , y + k_2 / 2)$
$k4 := \Delta t \cdot f(t_i + \Delta t , y + k_3)$

$y := y + (k_1 + 2k_2 + 2k_3 + k_4) / 6$

$t_i := t_i + \Delta t$
silmukka loppu

tulosta y(t)

Rungen-Kuttan menetelmän virhe on verrannollinen askeleen neljänteen potenssiin. Sen virhekaava jakovälille ja sen puolikkaalle saadaan virhearvio samalla tavalla kuin Eulerin menetelmän yhteydessä, mutta nyt tarkkuus paranee kertoimella 15:

(43)-$y_1 + y_{½} = \varepsilon_1 - \varepsilon_{½}, \approx 16\ \varepsilon_{½},- \varepsilon_{½},= 15\ \varepsilon_{½},.$

Matemaattisten ohjelmistojen adaptiiviset menetelmät käyttävät kuitenkin usein virhearvioon kahden eri tarkkuuden Rungen-Kuttan menetelmää [5 s.950], jolloin virheen tarkastus ei kuluta lisää funktion laskutoimituksia, vaan virhe saadaan

arvioiduksi kertoimien k_1, k_2, k_3, k_4 avulla. Siten tarvittavien funktioiden laskumäärää voidaan vähentää.

2.4 Nopeuden arviointimenetelmiä

Ennen numeeristen esimerkkien tarkastelemista esitetään muutama menetelmä eri laskentatapojen hyvyyden arvioimiseen. Vertailtavia kriteerejä yleensä ovat ratkaisun nopeus ja tarkkuus sekä ratkaisun kuluttama muistitila. Näistä viimeistä ei tarkastella tässä tutkimuksessa.

Ensimmäinen ja selkein vertailu saadaan valmiille ohjelmalle: asetetaan kilpaileville menetelmille sama tarkkuusvaatimus ja otetaan simulointiajat. Käytännössä tällöin ajetaan samaa ratkaisua useita kertoja satunnaisten virheiden poistamiseksi.

Toinen tapa perustuu ennalta tunnetun ratkaisun käyttöön. Kun ratkaisun likiarvo tunnetaan, voidaan eri menetelmien virhettä tutkia käyttämällä vertailtavissa menetelmissä likimain yhtä monta funktion arvon laskentakertaa. Tämä menetelmä on kätevin taulukkolaskentaa käyttävillä. Yksinkertaisimmillaan vertailu voidaan tehdä laskemalla enemmän laskentaa vaativalla menetelmällä vain yksi askel ja tarkastaa tarkkuus. Tämän jälkeen sama tulos lasketaan kilpailevalla menetelmällä, jossa joko asetetaan laskutoimitusten määrä vastaavaksi ja verrataan virhettä tai asetetaan virhe samaksi säätämällä laskutoimituksien lukumäärää.

Esimerkki 2: Laskentanopeuden vertailu Newton-Cotes vs Gauss

Esimerkiksi integroidaan $f(x) = \sin x$ välillä $[0,\pi]$ siten, että funktion sin x arvot lasketaan kuudessa pisteessä. Simpsonin säännöllä kuuteen jakoväliin tarvitaan seitsemän pistettä.

Tulokset taulukkolaskentaohjelmistosta ovat seuraavat:

Laskentapisteet Gaussin integraalille:						
0.106077	***0.532171***	***1.195974***	***1.945618***	***2.609422***	***3.035516***	
Funktion arvot						
0.105878	*0.507405*	*0.930573*	*0.930573*	*0.507405*	*0.105878*	
Painokertoimet						
0.028493	*0.287538*	*0.683969*	*0.683969*	*0.287538*	*0.028493*	
Tulos ja virhe						
1.99999999947727			*-5.22731E-10*			
Sama Simpsonin säännöllä:						
Laskentapisteet tasavälisen jaon mukaisesti						
0	*0.523599*	*1.047198*	*1.570796*	*2.094395*	*2.617994*	*3.141593*
Painokertoimet						
1	*4*	*2*	*4*	*2*	*4*	*1*
Funktioiden painotetut arvot						
0	*2*	*1.732051*	*4*	*1.732051*	*2*	*5.67E-16*
Tulos ja virhe:						
2.000863189674	*0.000863189674*					

Simpsonin säännön virhe on $1{,}65 \cdot 10^{6}$ –kertainen. Kun Simpsonin säännöllä funktion *f(x)* arvo lasketaan 201 kertaa, virhe on $6{,}76 \cdot 10^{-10}$ eli samaa suuruusluokkaa 6 pisteen Gaussin kaavaan verrattuna. Siten Gaussin integroinnin käyttö suoraan pudottaa käytettävän laskenta-ajan noin kolmeen prosenttiin Simpsonin säännön laskenta-aikaan verrattuna. Kokonaissäästö ohjelmoidulle ratkaisulle ei ole aivan yhtä hyvä, koska ohjelmien käynnistys ja tulostustoiminnot vievät saman ajan molemmissa tapauksissa.

3. Ensimmäisen kertaluvun differentiaaliyhtälöistä

3.1 Johdanto

Ensimmäisen kertaluvun differentiaaliyhtälöiden ratkaisun tarkastelussa keskitytään separoituviin differentiaaliyhtälöihin ja lineaariseen differentiaaliyhtälöön, koska ne ratkeavat integroimalla. Esimerkkitehtävät ratkotaan integrointeihin perustuvilla menetelmillä ja Rungen-Kuttan menetelmällä, jota pidetään esitettävän numeeriseen integrointiin perustuvan menetelmän verrokkina.

Menetelmää voi luonnollisesti soveltaa myös muihin integraalein ratkeaviin differentiaaliyhtälöihin. Tehtävien ratkaisuissa luonnollisesti analyyttisiä ratkaisuja kannattaa käyttää hyväksi aina, kun sellaiset vähentävät numeerisen integroinnin tarvetta. Näissä esimerkeissä kuitenkin valotetaan nimenomaan numeerista ratkaisua, joten ratkeavien välitulosten osalta analyyttisestä ratkaisusta saatavaa hyötyä ei aina käytetä hyväksi.

Differentiaaliyhtälöiden analyyttisissä ratkaisumenetelmissä joudutaan usein tekemään sijoituksia ja valitsemaan sopivia apumuuttujia, jotta tehtävä yksinkertaistuisi joko analyyttisen ratkaisun löytämiseksi tai laskennallisesti paremman kaavan saamiseksi. Tässä voi käyttää apuna omien taitojen lisäsi myös symbolisesti laskevia ohjelmistoja.

Koska uudet ohjelmistot mahdollistavat tehtävien helpon sieventämisen mahdollisimman pitkälle ja moniin eri muotoihin, numeerisen ratkaisun ohjelmointitaito on merkittävää: nopea ratkaisu pitää voida valita monista saman tehtävän eri esitystavoista. Ohjelmointitaidon lisäksi yhtälöjen analyyttisen käsittelyn taito ja kokemus yhtälöiden muokkaamisesta eri muotoihin antaa entistäkin suuremmat mahdollisuudet ongelmien ratkaisuun.

Kokemukseni perusteella todellisten probleemien ratkaisussa tarvitaan intuitiota ja harjaantumista, jotka saavutetaan edelleen itse tehtävällä laskentatyöllä. Siten mekaanisten sievennysten opettaminen ja opettelu on edelleen mielekästä, koska se luo pohjan myös symbolisten ohjelmistojen tehokkaalle käytölle.

3.2 Separoituva yhtälö

3.2.1 Analyyttinen ratkaisu

Monet tekniset ja luonnontieteelliset mallinnustehtävät johtavat joko suoraan separoituviin differentiaaliyhtälöihin tai sellaisiin differentiaaliyhtälöihin, jotka sopivalla sijoituksella saadaan separoituviksi. Tutkitaan separoituvaa differentiaaliyhtälöä muodossa

$$(44) y' = f(t)\, g(y) \Leftrightarrow \frac{y'}{g(y)} = f(t),\ g(y) \neq 0.$$

Jos $g(y_0) = 0$, ollaan tasapainopisteessä eli yhtälöllä on vain vakioratkaisu $y = y_0$. Jos $g(y)$ saavuttaa laskennan aikana arvon nolla, yhtälön ratkaisukäyrä on saavuttanut tasapainoarvon ja pysyy tässä vakioarvossa tästä eteenpäin. Tarkastelussa rajoitutaan jatkossa niihin tapauksiin, joissa $g(y) \neq 0$.

Integroidaan molemmat puolet muuttujan t suhteen, jolloin saadaan

$$(45) \int \frac{y'(t)dt}{g(y)} = \int f(x)dx.$$

Tässä käytetään integroinnin sijoitusmenetelmää, joka antaa mahdollisuuden muuttaa integrointi t:n suhteen integroinniksi y:n suhteen eli päästään muotoon

$$(46) \int \frac{dy}{g(y)} = \int f(t)dt.$$

Jos integrointi onnistuu, ratkaistaan funktio y näin saatavasta tehtävästä. Integroinnin yhteydessä on huomattava integrointivakion C lisääminen ja se, että tarkastelu rajoittuu yksi kerrallaan t-y -tason alueisiin, joissa $g(y) \neq 0$.

Tässä työssä tarkastellaan alkuarvoprobleemia, joiden integrointiin voidaan käyttää määrättyä integraalia. Tällöin kaava (46) saadaan muotoon

$$(47) \int_{y_0}^{y(t)} \frac{dy}{g(y)} = \int_{t_0}^{t} f(t)dt.$$

Jos molemmat integraalit saadaan ratkaistuiksi ja y (tai t) ratkaistuksi annetulla arvolla, tehtävä on valmis. Jos integrointi ei onnistu analyyttisesti, se tehdään

numeerisesti. Siten yhtälön (47) esittämä muoto separoituvan differentiaaliyhtälön ratkaisuksi on tässä työssä tehtävien numeeristen tarkastelujen lähtökohtana.

3.2.2 Numeerinen ratkaisu

Erityisesti numeerisen ratkaisun osalta ratkaisun olemassaoloa ja yksikäsitteisyyttä on syytä tarkastella ennen laskennallisten menetelmien suoraviivaisen käytön aloittamista. Separoituvan differentiaaliyhtälön osalta erityistä mielenkiintoa kannattaa suunnata funktion *g(y)* nollakohtiin, jotka toimivat tehtävässä tasapainokohtina, joissa ratkaisu on vakiofunktio. Jos separoituva differentiaaliyhtälö toteuttaa yksikäsitteisyyslauseen vaatimuksen, tuloksena saadaan mahdollisten numeeristen ratkaisujen rajat.

Tutkitaan tilannetta, jossa on annettu funktiot *f(t)* ja *g(y)* sekä alkuarvot y_0 ja t_0 sekä haluttu ratkaisupiste *t*. Tavoitteena on selvittää arvo *y(t)*. Ratkaisu lähtee liikkeelle edellisestä integraalimuotoisesta ratkaisusta, josta muokataan funktio *R(y)*.

$$(48) R(y) = \int_{y(t_0)}^{y(t)} \frac{dx}{g(x)} - \int_{t_0}^{t} f(t)dt$$

Ratkaistaan funktion *R(y)* nollakohta Newtonin menetelmällä

$$(49) y_{i+1} = y_i - \left(\int_{y(t_0)}^{y_i} \frac{dx}{g(x)} - \int_{t_0}^{t} f(t)dt \right) \cdot g(y_i).$$

Perusratkaisuna saatu iterointikaava on sellaisenaan käytettävissä laskimiin ja helposti ohjelmoitavissa taulukkolaskentaohjelmistoon.

Newtonin menetelmän mahdollisen suppenemisongelman, käytettävän integrointimenetelmän askelpituuden ja mahdollisen tasapainon (*g(y)*=0) ylihyppäämisprobleeman ratkaisemiseksi perusmenetelmästä koodattiin tarkastuslaskelmat sisältävä versio, jonka Matlab -koodi on esitetty liitteenä A2.

Siinä myös integroinnin nopeutta lisättiin käyttämällä hyväksi tunnettua määrätyn integraalin kaavaa

$$(50) \int_{x(t_0)}^{x_i} \frac{dx}{g(x)} = \int_{x(t_0)}^{x_{i-1}} \frac{dx}{g(x)} + \int_{x_{i-1}}^{x_i} \frac{dx}{g(x)}.$$

Ratkaisualgoritmin vaiheet ovat suoraan esitettyinä:

Alkuaskeleen määritys
Tasapainon tarkastelu
Integrointiaskeleen numeerinen määrittäminen
Toistona seuraava algoritmi:
Uuden *y*-arvon määritys miniminä seuraavista:
funktion 1/*g(y)* numeerinen integrointi ja Newtonin askel tai sekanttiaskel
integrointiaskel
tasapainon ylitystä seuraava puolitusarvo
optiona vertailu funktion *g(y)* nollakohtaan

Lukujen tallennus tulostusta varten
Tuloksen esittäminen
Optiona kuvaajan tulostus

Vaihe 1 perustuu joko käyttäjän antamaan Newtonin iteroinnin lähtöarvoon y_1 tai algoritmin itse laskemaan tulokseen alkuarvolla y_0

$$(51)\ y_1 = y_0 + \int_{t_0}^{t} f(t)dt \cdot g(y(t_0)) .$$

Seuraavana vaiheena tarkastellaan ylittääkö askel tasapainopistettä. Tämä tapahtuu karkeasti tutkimalla funktion *g(y)* merkkiä. Siten voidaan helposti konstruoida funktioita, joilla menetelmä ei toimi, jos käyttäjä ei aseta mielekästä alkuarvoa. Nämä funktiot ovat valitettavasti usein vaikeita myös muillekin numeerisille menetelmille kuten Rungen-Kuttan menetelmälle.

Integrointiaskel määritellään Gaussin integraalille. Tässä olisi mahdollista käyttää myös mitä tahansa muuta integrointimenetelmää ja sen virhetarkastelua.

Kun alkuvaiheet on saatu tehdyksi, alkaa varsinainen iterointi eli *y*-akselin suuntaan integrointi. Valitusta alkuarvosta lasketaan seuraava alkuarvo Newtonin menetelmällä. Jos näin saatu uusi *y*:n arvo on pidempi kuin integroinnin tarkkuus antaa myöten, askel lyhennetään integrointiaskeleen mittaiseksi. Jos tämä askel ylittää funktion *g(y)* nollakohdan, askelta lyhennetään kunnes nollaa ei ylitetä. Näin saatu uusi *y* sijoitetaan silmukkaan seuraavan *y*-arvon määrittämiseksi.

Ohjelmaa voidaan nopeuttaa hiukan, jos ennalta tiedetään integrointiaskeleita tarvittavan runsaasti. Tällöin laskenta voi edetä integrointiaskelin kunnes funktio *R(y)* vaihtaa merkkiä. Tämän jälkeen Newtonin iteroinnin alkuaskel ratkaistaan sekanttimenetelmällä kahden viimeisen arvon välistä, ja tulos saadaan yleensä yhdellä jatkoaskeleella.

Ratkaisun tarkkuuden osoittaminen perustuu niinikään funktion *R(y)* ominaisuuksiin. *R(y)* on integraalina jatkuva ja voi vaihtaa merkkiä vain, jos välissä on nollakohta.

Olkoon integroinnin virhe e. Tällöin oikea arvo y on välillä $[y_1,y_2]$, kun tulo $[R(y_1,) \pm e] \cdot [R(y_2) \pm e] < 0$.

Tehtävän ratkaisu on samanlainen, ratkaistiinpa annetulle t –arvolle y –arvoa vai päinvastoin. Molemmat johtavat samanlaiseen kahdesta määrätystä integraalista muodostuvaan ratkaisuun.

Kun differentiaaliyhtälö ratkaistaan lähtien muodosta $y' = f(t,y)$, saadaan derivaatalle arvo, jonka oletetaan pysyvän vakiona ensimmäisen askeleen ensimmäisen osan ajan. Kun tätä arvoa Rungen-Kuttan menetelmässä myöhempien arvojen avulla täydennetään ja korjaillaan, funktion y arvolle saatavan approksimaation virhe on verrannollinen askelpituuden neljänteen potenssiin. Numeerisilla integrointimenetelmillä voidaan helposti päästä suurempiin tarkkuuksiin, ja kun Newtonin menetelmä tarjoaa mahdollisuuden ratkaista integroinnin yläraja, saadaan kilpailukykyiseltä tuntuva menetelmä.

3.2.3 Esimerkkejä

Menetelmän käytännön hyvyyden arvioimiseksi esitetään muutamia esimerkkejä. Osa niistä on toteutettu taulukkolaskennalla, ja osassa asetetaan esitettyä integrointiin perustuvaa menetelmää vastaan Matlab –ohjelmiston valmistoiminto differentiaaliyhtälön ratkaisemiseksi.

Esimerkki 1
Menetelmän kilpailukykyä havainnollistava $y' = f(x)$, jonka ratkaisu on $y = \int f(x)\,dx$.

Vertailukohtana numeeriselle differentiaaliyhtälön ratkaisulle käytetään Rungen-Kuttan menetelmää, joka pelkistyy tässä tapauksessa Simpsonin säännöksi. Edellisen luvun laskuesimerkissä 2 sama funktio integroitiin Simpsonin säännöllä ja Gaussin integroinnilla. Se kuvaa Rungen –Kuttan menetelmän ja tehokkaaseen integrointimenetelmään perustuvan ratkaisun eroa. Koska Gaussin integrointi on tehokkaampi kuin Simpsonin sääntö (laskentakertojen suhde oli edellisen luvun esimerkissä noin 30:1), nähdään integrointiin perustuvien menetelmien olevan potentiaalisesti parempia kuin Rungen – Kuttan menetelmä.

Esimerkki 2.
Taulukkolaskennalla klassikko *y'=y*, *y*(0)=1, *y*(2)=?

Tässä tapauksessa joudutaan käyttämään Gaussin integroinnin lisäksi Newtonin menetelmää arvon *y*(2) määrityksen. Kun Newtonin iteroinnin lähtöarvo *y1* määräytyy automaattisesti ilman käyttäjän kontrollia, saadaan tulos seuraavasti:

y0	1
y1	3
y2	5.704164
y3	7.180423
y4	7.386084
Y5	7.389057
y6	7.389058
y7	7.389058

Tuloksen saamiseksi funktion *f(y)* arvo laskettiin integroinnin yhteydessä 7·6=42 kertaa ja virhe on $1{,}6 \cdot 10^{-6}$. Tulos poikkeaa oikeasta arvosta $e^2 \approx 7{,}38956$ vasta seitsemännessä merkitsevässä numerossa. Vertailun vuoksi Rungen-Kuttan menetelmässä askelvälillä 0,1 funktio *f(y)* tulee lasketuksi 84 kertaa. Tulos on tällöin 7,38898 ja virhe kertaluokkaa suurempi $1{,}13 \cdot 10^{-5}$.

Esimerkki 3
Oman lajinsa yksinkertaisin *y' = xy*, *y*(0), *y*(2)=?

Tilanne on sikäli mielenkiintoinen, että integraalien numeerisessa ratkaisumenetelmässä päädytään täsmälleen samaan tehtävään kuin edellisessä esimerkissä, koska separoituvan yhtälön ratkaisussa x:n suhteen integrointi johti sattumalta samaan numeroarvoon. Erotuksena on vain se, että funktio *f(x) = x* piti laskea kuudesti ennen Newtonin iteroinnin alkua. Tulokset voidaan koota taulukkoon:

	Laskutoimituksia funktiolla			
	f(x)	*g(y)*	*tulos*	*virhe*
Integrointi ja Newton	*6*	*43*	*7.389058*	*-1.63E-06*
Runge-Kutta	*84*	*84*	*7.388998*	*5.86E-05*
Analyyttinen ratkaisu			*7.389056*	

Jos ratkaisussa haetaan suurempaa tarkkuutta, iteroinnissa joudutaan lyhentämään integrointiaskelta. Integroinnin askeleen säätäminen kuluttaa 12 funktion laskentaa, jos tarkkuuden laskennassa saatavaa tulosta ei käytetä hyväksi.

Seuraavaksi esitetään iteroinnin kulku, kun ennen jokaista iterointiaskelta integroinnin askelpituudeksi valitaan joko Newtonin iteroinnin tai integroinnin tarkkuuden edellyttämä askelpituus sen mukaan, kumpi askelpituuksista on lyhyempi. Esimerkissä integrointiaskeleen maksimipituudeksi on valittu 1, koska sitä vastaava numeerisen integroinnin virhe käyttäen kahteen eri likiarvoon perustuvaa virhekaavaa

on todettu pienemmäksi kuin haluttu tarkkuus 10^{-9}. Tällöin lopullinen virhe on hieman suurempi, mutta silti samaa suuruusluokkaa. Lopputuloksena saatava iteroinnin kulku:

Y0	1
Y1	2
Y2	3
Y3	4
Y4	5
Y5	6
Y6	7
Y7	7.378629
Y8	7.389049
y9	7.389056
y10	7.389056

Iterointi suppeni kymmenessä askeleessa integraalin ylärajaan, ja virhe johtuu integroinnin epätarkkuudesta.

Tarvittavien laskutoimitusten lukumäärän vertailu antaa tuloksen:

	Runge-Kutta	*Integrointi ilman tarkuuden säätöä*	*Integrointi ja Newton*	
Virhe	*5.85651E-05*	*-1.6E-06*	*4.98E-09*	
Laskenta g(y)	*84*	*43*	*61*	*12*
Laskenta f(x)	*84*	*6*	*6*	*12*

Integrointi ja Newton -sarakkeen vasen puoli kuvaa integroinnissa tarvittujen laskutoimitusten lukumäärää ja oikea sarake tarkkuuden määrityksen edellyttämien laskutoimitusten lukumäärää. Kun numeeriseen integrointiin perustuva algoritmi ja Rungen-Kuttan menetelmä käyttävät saman määrän laskutoimituksia funktioon *g(y)*, virheen ero on viisi dekadia. Jos halutaan saada numeerisen integroinnin perusalgoritmin tarkkuus $1{,}6 \cdot 10^{-6}$, tarvittavien askelten lukumääräksi saadaan kokeilemalla 50 eli laskutoimituksia tarvitaan noin 200 sekä funktiolle *f(x)* että *g(y)*. Saavutettu hyöty on siis suuruusluokkaa 5:1.

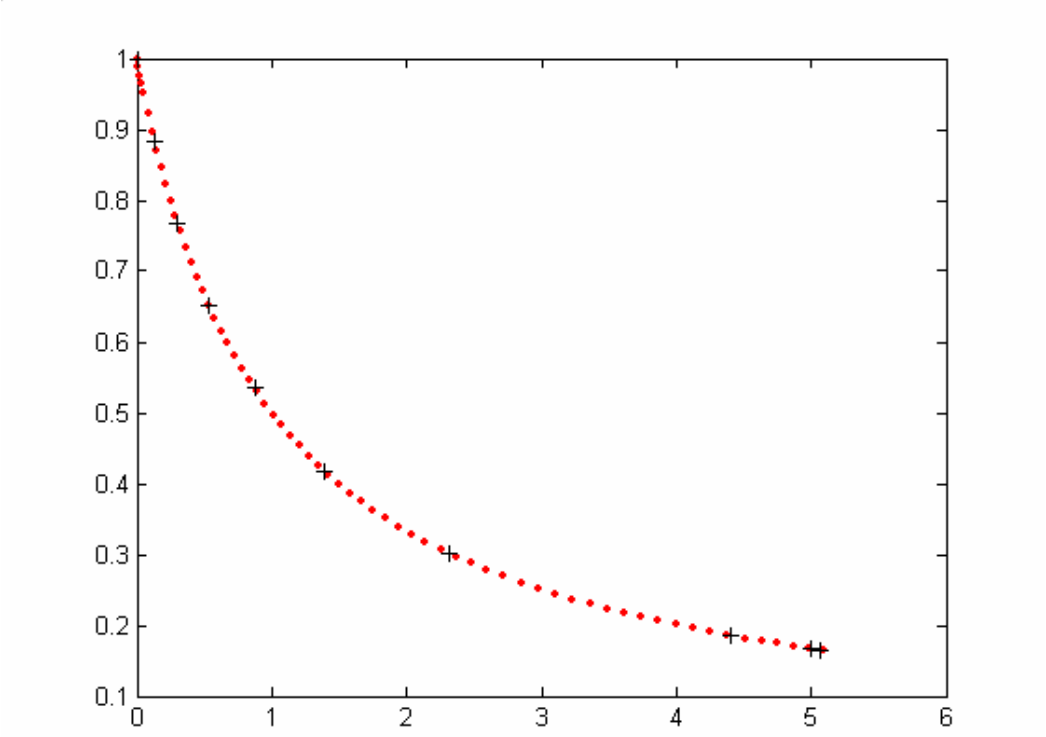
Esimerkki 4
Tasa-arvopistettä $y = 0$ lähestyvä $y' = -y^2$, $y'(0)=1$ $y(5)=?$

Tehtävässä asetetaan oma ratkaisukoodi haastamaan tunnetun Matlab -ohjelmiston (versio 6.5.1) valmista algoritmia. Tehtävän ratkaiseva Matlab -koodi on liitteenä A2 ja A3.

Tehtävässä verrattiin tarvittavaa laskenta-aikaa eri vaihtoehtojen välillä. Vertailun suorittanut koodi liitteenä A4.

Matlabin ode45() ratkaisee differentiaaliyhtälön adaptiivisella algoritmilla, jossa askelpituutta säädetään käyttäjän antaman virheen mukaan. Sitä verrataan omaan ohjelmakoodiin, jossa käyttäjä niin ikään antaa virherajan.

Kummassakin ohjelmassa oli funktion piirto-optio. Kun ratkaisut piirretään samaan kuvaajaan, nähdään saman tarkkuuden edellyttävän tiuhempaa laskentapistemäärää Rungen-Kuttan menetelmässä (pisteet) verrattuna risteillä merkittyihin integrointiin perustuviin pisteisiin. Jokaisen punaisen pisteen yhteydessä tulon *f(x) g(y)* arvo tulee lasketuksi 4 kertaa Rungen-Kuttan menetelmän yhteydessä. Mustat ristit kuvaavat integrointiin perustavalla algoritmilla tehtyä laskentaa, jossa kunkin ristillä merkityn arvon laskenta on edellyttänyt funktion *g(y)* arvon laskemista kuudesti. Tehokkaampi integrointimenetelmä mahdollistaa pidemmän askeleen.



Differentiaaliyhtälön $y' = -y^2$ ratkaisu

Kuvasta nähdään myös, miten integroinnin tarkkuus määrää askelen kunnes ollaan hyvin lähellä tulosta, jossa Newtonin menetelmä suppeni nopeasti (2 askelta) haluttuun tulokseen.

Vertailussa olivat valmistoiminto Runge-Kutta (RK) tarkkuusvaatimuksella 1e-6, suppenemissäädöt sisältänyt integrointiin ja Newtonin iteraatioon perustuva algoritmi, ja ilman tarkastuksia Gaussin integroinnin ja Newtonin menetelmän yhdistävä algoritmi, joka ei supennut tässä esimerkissä huonolla tai kaavalla 51 lasketulla Newtonin menetelmän lähtöarvolla *y1*.

Jos käyttäjä on tehnyt tutkimukset funktioiden kulusta etukäteen ja selvittänyt funktion *g(y)* nollakohdat, lisäämällä koodiin hieman hidastavan ”onko ratkaisu ylittänyt tasapainon” –vertailun (algoritmin kohta 4.1.d) suppenemisongelmat saadaan poistetuksi.

Taulukossa ”Aika 1” on Rungen-Kuttan menetelmän käyttämä aika jaettuna suppenemiskontrollin sisältävän algoritmin ajalla. ”Aika 2” on Rungen-Kuttan menetelmän aika jaettuna ajalla, joka kuluu integroinnin ja Newtonin menetelmän yhdistelmään ilman suppenemistarkasteluja. ”Virhe 1” on suppenemiskontrollit sisältävän algoritmin tuloksen virhe ja ”Virhe 2” ilman suppenemiskontrollia saatavan tuloksen virhe.

Oikea	*virhe RK*	*virhe 1*	*virhe 2*	*Aika1*	*Aika 2*
0.14493	*-1.2974e-007*	*8.8104e-012*	*NaN*	*3*	*0.2069*

Tehtävässä suppenemiskontrolloidun algoritmin (liite A2) virhe on pienin, ja Rungen-Kuttan menetelmää käyttävä valmistoiminto ode45 käytti kolminkertaisen ajan epätarkemman tuloksen saamiseen. Tilanne mallintaa myös ilman kontrollia olevan Newtonin menetelmän murheen: huonon alkuarvauksen vuoksi se ei supennut lainkaan, ja aikaakin kului viisinkertaisesti valmistoimintoon verrattuna.

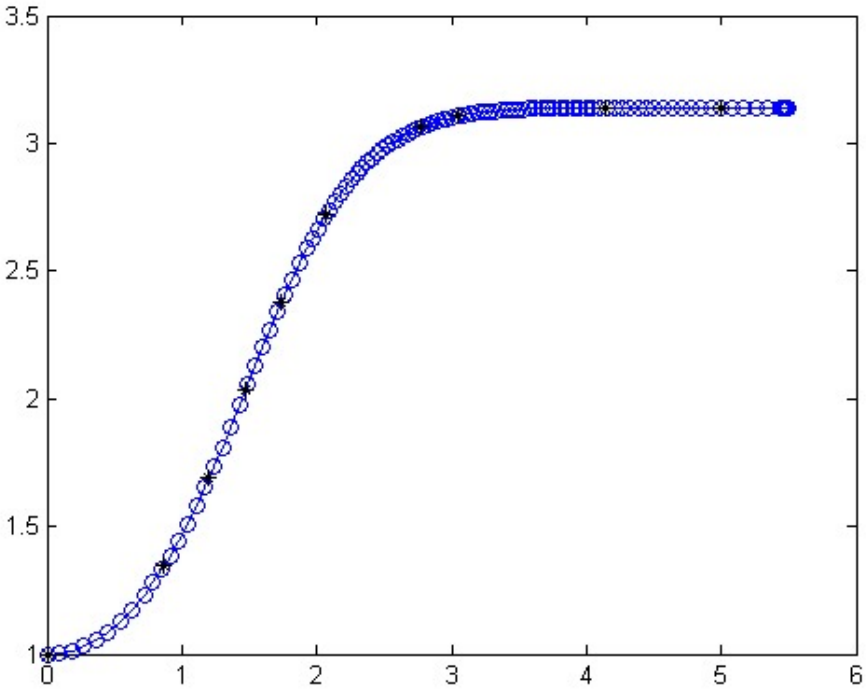
Kun sama differentiaaliyhtälö ratkaistiin, mutta tällä kertaa arvon $y(1)$ ratkaisemiseksi, saadaan tulokseksi:

Oikea	*virhe RK*	*virhe1*	*virhe 2*	*Aika1*	*Aika*
0.47619	*-1.6339e-008*	*2.7756e-016*	*2.8683e-009*	*3*	*12*

Nyt Newtonin menetelmä suppenee, ja tulos saavutetaan aikasuhteessa 12. Virhe on pienempi kuin valmistoiminnon. Virhesuojattu algoritmi antaa edelleen kolminkertaisen aikaedun valmistoimintoon verrattuna.

Esimerkki 5
Tasapainoa lähestyvä *y’* = sin*(y)* · ; *y(* 0 *)* = 1, *y(* 5 *)* = *?*

Vertailu tehtiin samalla ohjelmistorakenteella kuin edellisessäkin simuloinnissa. Nyt funktion arvot ovat yksikäsitteisyyslauseen mukaisesti puristuneet funktion g(y) nollakohtien $y = 0$ ja $y = \pi$ väliin, ja y lähestyy arvoa π.



Yhtälön y' = sin(y)·t ratkaisu. Pallot Runge-Kutta, ristit numeeriseen integrointiin perustuva menetelmä.

Tehtävä on valittu siten, että normaali Newtonin iteraatio on vaikeuksissa funktion muodon vuoksi.

Esitetty algoritmi suppeni vastaukseen kymmenellä askeleella, ja iteroinnin vaiheet on piirretty risteinä kuvaan. Jokainen risti edellyttää 6 funktion *g(y)* arvon laskemista integroinnissa ja tarkastusten vuoksi vielä lisäaikaa. Pallo sisältää 4 funktion arvon laskemista. Nopeusetu syntyy samaan tarkkuuteen pääsemisestä pidemmin integrointiaskelein.

Lisätarkasteluna esimerkkiyhtälössä alettiin kasvattaa muuttujan *t* loppuarvoa ja katsottiin muutoksen vaikutusta simuloinnin suppenemiseen.

Taulukossa on esitetty saadut tulokset:

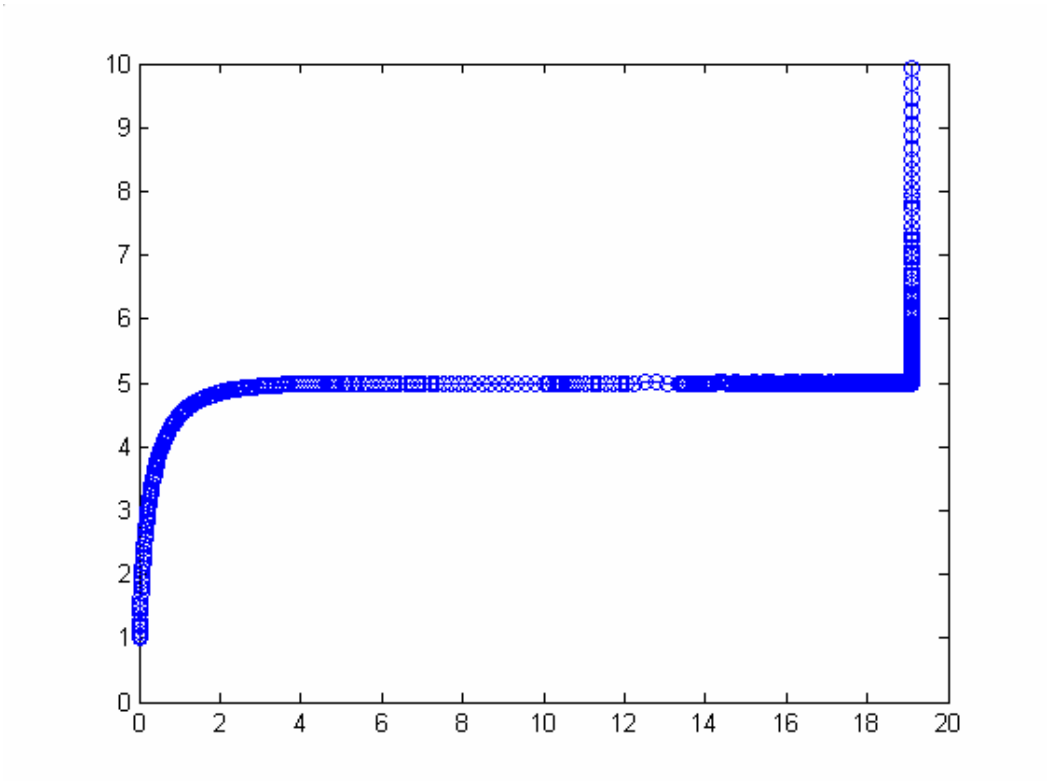
t	y	aikasuhde 1	aikasuhde 2
1	1.109	3.3333	10
1.5	1.466	1.8333	11
2	2.0692	2.25	9
2.5	2.6559	3.0333	9.1
3	2.9811	2.75	11
3.5	3.1009	3	12
4	3.1336	3.5	14
4.5	3.1404	2.8	X
5	3.1414	3	5, väärä juuri
5.5	3.1416	3.9	X

Tuloksista näkyy, miten Newtonin iterointi ilman kontrollirakenteita (aikasuhde 2) on nopea toimiessaan, mutta voi hypätä ulos sallitusta ratkaisualueesta ja supeta sen jälkeen kokonaan väärään juureen tai olla kokonaan suppenematta (”X” taulukossa). Kontrollirakenteilla tuettu algoritmi (aikasuhde 1) ratkaisee ongelman kaikkine tarkastuksineen kolmasosassa ajasta, jonka valmisohjelmisto käyttää.

Esimerkki 6
Patologinen $y' = e^x (y - 5)^2$; $y(0) = 1$; $y(x) = ?$

Tässä y lähestyy tasapainopistettä $y = 5$. Koska funktio on kasvava tasapainon molemmin puolin, iterointi voi hypätä tasapainopisteen $y = 5$ yli äärettömyyksiin. Matlabin valmisohjelmalla näin käy, kun $x \approx 19{,}5$.

Yhtälö on vaikea myös suppenemiskontrollin sisältävälle numeeriseen integrointiin perustuvalle algoritmille, koska siinä tarkastus perustui derivaatan merkin vaihtumiseen. Ongelmalta suojaa vain algoritmin kohdan 4.1.d toteuttaminen eli ensin tehty tasapainopisteiden tarkastelu. Sama ehto voidaan ohjelmoida Rungen-Kuttan menetelmään, jolloin silläkään laskettaessa arvot eivät karkaa tasapainojen yli.



Ratkaisu karkaa yli tasapainopisteen kohti ääretöntä

Esimerkki osoittaa karkeiden kvantitatiivisten ja kvalitatiivisten tarkastelujen välttämättömyyden ennen numeerisiin ratkaisuihin hyppäämistä. Samoin se osoittaa ohjelmointitaidon ja sen opettamisen merkityksen numeriikassa: kyky lisätä ominaisuuksia valmiisiin ohjelmakoodeihin parantaa laskennan valmiuksia merkittävästi.

Jos yhtälö halutaan ratkaista, korjattuna vaihtoehtona toimii sekanttimenetelmä tai regula falsi, joka pitää juuren halutussa rajassa tai jopa kylmästi toteutettu puolitushaku, jossa tasapainopiste on toisena rajana. Jos tulos tulee riittävän lähelle tasapainoa, lopetusehto kertoo funktion *y* saavuttaneen tasapainon. Muuten juuri saadaan normaalisti funktion *R(y)* merkkiä tutkimalla. Tämä perusversiota hieman hitaampi algoritmi ratkaisee tasapaino-ongelman, mutta edellyttää *g(y)*:n nollakohtien ratkaisemista ennen laskennan aloittamista.

3.2.4 Johtopäätökset integrointiin perustuvasta algoritmista

Johtopäätöksenä voidaan todeta, että separoituvan differentiaaliyhtälön ratkaisu Newtonin yhtälönratkaisu- ja Gaussin integrointimenetelmää soveltavalla algoritmilla tarjoaa selkeän mahdollisuuden säästää laskenta-aikaa. Siten sillä on käyttöä esimerkiksi Monte Carlo -simulointien sisällä ja vähän laskenta-aikaa vaativien

prosessien ohjauksessa. Vastaukseen saadaan suuri tarkkuus, joka on myös helposti arvioitavissa.

Huomiona voidaan todeta, että lopullisen ratkaisun määritys riippuu integroinnin tarkkuudesta. Jos integroinnin tarkkuus oli riittävä eikä integrointiaskeleen pituus ollut iteroinnin rajoittava tekijä, Newtonin menetelmä oli kokeilujen valossa kätevin. Jos taas integroinnin askelpituus oli ensisijainen tulosta rajoittava tekijä, tehtävä oli viimeistä askelta lukuun ottamatta ratkaistu ”raaka voima” menetelmällä. Siten melko hyvin toimiva menetelmä saadaan myös yhdistämällä ”raaka voima” ja sekanttimenetelmä tai suppenemiseltaan varma regula falsi. Nopeuskilpailuissa menetelmien paremmuus vaihtelee alkuarvauksen ja Newtonin iteroinnin aiheuttaman satunnaisen tekijän vuoksi.

Erityisen tehokas integrointiin perustuva lähestymistapa on autonomisten eli ajasta riippumattomien differentiaaliyhtälöiden ratkaisemisessa, koska tällaiset ratkeavat suoraan yhdellä integraalilla. Huomattavaa myös on, että mikäli jompikumpi integraali ratkeaa analyyttisesti, voidaan analyyttistä ratkaisua käyttää hyväksi integrointeihin perustuvissa menetelmissä, mikä kasvattaa saavutettavaa hyötyä entisestään. Samoin esitetty algoritmi on tehokas käänteisen probleeman ”millä *t* annettu *y*” ratkaisussa.

Esitetyt ratkaisutavat on opetuskokeiluissa havaittu helpoiksi ymmärtää ja ne soveltuvat hyvin lukiolaisilla käytössä oleviin laskimiin. Oppilaat ovat käyttäneet integroinnin ylärajan ratkaisemista Newtonin yhtälönratkaisumenetelmällä pienten probleemien ratkaisussa. Menetelmä on mahdollistanut fysiikan numeeristen differentiaaliyhtälöiden käyttämisen opetuksen syventämiseen ja mielenkiintoisten laskennallisten esimerkkien tarjoamiseen.

3.3 Lineaarinen yhtälö

3.3.1 Analyyttinen ratkaisu

Ensimmäisen kertaluvun lineaarinen differentiaaliyhtälö *y’ + a(t) y = b(t)* on ratkaistavissa käyttämällä tunnettua tekniikkaa, jota kutsutaan integroivaksi tekijäksi. Yhtälö kerrotaan funktiolla

(52) $\mu(t) = e^{\int a(t)\,dt}$,

missä integraali on funktion *a(t)* jokin integraalifunktio. Lopputuloksena saadaan yhtälö

(53) $y'(t) \cdot \mu(t) + a(t) \cdot \mu(t) \cdot y(t) = b(t) \cdot \mu(t)$.

Kaavan 53 vasen puoli on tulon $\mu(t)\ y(t)$ derivaatta, joten integroimalla t:n suhteen ja lisäämällä integroimisvakio C saadaan

(54) $y(t) \cdot \mu(t) = \int b(t) \cdot \mu(t)dt \ + C,$

ja edelleen

(55) $y(t) = \frac{1}{\mu(t)} \int b(t) \cdot \mu(t)dt + \frac{C}{\mu(t)}$.

Oikean puolen toinen termi eli integroimisvakion C ja funktion $\mu(t)$ osamäärä on separoituvan homogeeniyhtälön $y' = -\ a(t)\ y$ ratkaisu. Ensimmäistä termiä yritetään usein ratkaista niin sanotun yritteen avulla. Tällaisia ratkaisutapoja ei käsitellä tässä, koska yhtälö (54) antaa tehtävän numeerisen ratkaisun perustan. Muita lineaarisen differentiaaliyhtälön ratkaisutapoja on esitetty useimmissa differentiaaliyhtälöitä käsittelevissä kirjoissa, kuten esimerkiksi [7, s.476-477].

3.3.2 Numeerinen ratkaisu

Numeerinen ratkaisu perustuu jälleen määrättyyn integraaliin. Nyt valitaan integroivaksi tekijäksi määrätyn integraalin sisältävä

(56) $\mu(t) = e^{\int_{t_0}^{t} a(t)dt}$.

Tällöin nähdään suoraan, että

(57) $\mu(t_0) = 1.$

Kun integrointi suoritetaan arvosta t_0 arvoon t, saadaan

(58) $y(t) \cdot \mu(t) - y(t_0) = \int_{t_0}^{t} \mu(t)b(t)dt$.

Kun t_0, t ja $y(t_0)$ on annettu, saadaan ratkaisu. Ratkaisualgoritmi sisältää seuraavat vaiheet:

1. Jaa integroimisväli $[t_0,t]$ valitun integrointimenetelmän edellyttämiin osaväleihin.

2. Laske edelleen integrointimenetelmällä funktion $\mu(t)$ arvot kussakin osavälipisteessä. Tässä numeerisen integroinnin suuri tarkkuus on tärkeää, koska integraalin virheet sijoitetaan eksponenttifunktioon.

3. Laske integraali $\int b(t) \cdot \mu(t) dt$

4. Ratkaise y.

5. Tee virhetarkastelu.

Virhetarkastelu voidaan tehdä perinteisellä adaptiivisella integroinnilla, jossa joko jakoväliä tai integroinnin astelukua muutetaan virheen selvittämiseksi. Adaptiivinen virhetarkastelu on helposti ohjelmoitavissa tai jopa kokeiltavissa itse. Lähtökohtaisesti näyttää ilmeiseltä, että kahden numeerisen integroinnin teko on nopeampaa kuin askelittain etenevät differentiaaliyhtälöiden ratkaisumenetelmät.

Perusratkaisun toteuttava yksinkertainen Matlab-koodi on liitteenä A5.

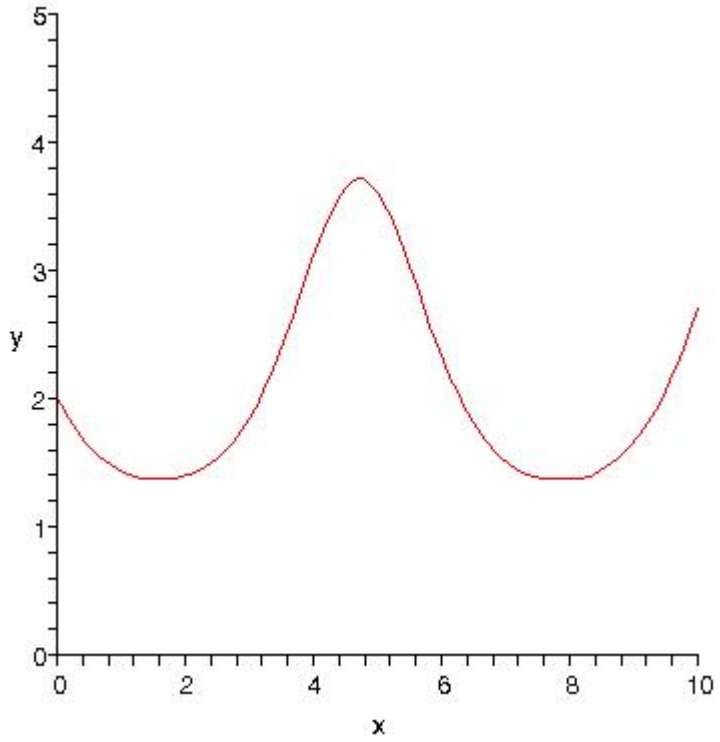
3.3.3 Esimerkkejä

Esimerkki 1. $y' + y\cos(t) = \cos(t), y(0) = 2, y(\pi) = ?$.

Tarkastellaan tehtävän ratkaisun edellyttämiä funktioiden *a* ja *b* laskentakertojen lukumäärää. Integroivan tekijän μ laskennassa tarvitaan 6 kertaa *a* –funktion laskemista ja eksponenttifunktion laskeminen. Integraalin yhteydessä tarvitaan vielä 6 funktion *b* laskentakertaa. Kiinteän askeleen Rungen - Kuttan laskenta tehtiin taulukkolaskennalla. Alla olevassa taulukossa on vertailtu arvon $y(\pi/2)$ ratkaisuun tarvittavia laskutoimituksia:

	Runge-Kutta	*Integrointi*
virhe	*7.55E-03*	*-2.30E-08*
a(x)	*84*	*36*
b(x)	*84*	*6*
exp(x)	*0*	*36*

Tuloksesta nähdään, että integroinnin käyttö antaa paremman tarkkuuden vähemmillä laskutoimituksilla.



Ratkaisu y' + y cost = cos(t)

Tämä yhtälö on myös separoituva, eli sen yhteydessä voidaan vertailla separoituvan yhtälön numeeriseen integrointiin ja Newtonin menetelmään perustuvaa algoritmia (taulukossa virhe A2) ja ensimmäisen kertaluvun lineaarisen yhtälön suoraa ratkaisijaa (taulukossa virhe A5). Algoritmit ovat liitteinä A2 ja A5, joita verrataan Matlabin valmistoimintoon. Numeerisia tuloksia verrataan analyyttisenä ratkaisuna saatavaan funktioon $y = 1+ e^{-\sin(t)}$. Lähtöarvoissa asetettu virhetaso on 10^{-6}. Esimerkissä Gaussin integraalissa painokertoimissa oli vain kahdeksan merkitsevää numeroa, minkä vuoksi tarkkuus on lyhyillä integrointiaskeleilla suuruusluokkaa 10^{-9} askelpituudesta riippumatta.

Tuloksena saadaan $y(1)$ ratkaisuksi:

y(1)	*Virhe R-K*	*Virhe A2*	*virhe A5.*	*suhde 1*	*suhde 2*
1.4569	*-6.5205e-011*	*5.5138e-006*	*-3.5097e-009*	*3.8333*	*4.1441*

Rungen-Kuttan menetelmä oli tässä simuloinnissa tarkin ja hitain, suora integrointi nopein ja hieman epätarkempi. Numeeriseen integrointiin ja Newtonin menetelmään perustuvassa algoritmissa kosinin integraalin laskennassa oleva virhe näkyy tuloksen tarkkuudessa. Laskenta-aikaa kului enemmän kuin suorassa integroinnissa. Siten suora integrointi oli aidosti parempi kuin separoituvan yhtälön numeerinen algoritmi.

Tarkempi analyysi saadaan, kun tulokset tulostetaan väliltä [0,π]. Suhde 1 kuvaa separoituvan yhtälön ratkaisualgoritmilla saadun ajan säästökerrointa ja suhde 2 suoran integroinnin säästökerrointa suhteessa Rungen-Kuttan menetelmällä laskevaan valmistoimintoon.

t	*Virhe R-K*	*Virhe A2*	*virhe A5*	*suhde 1*	*suhde 2*
0.31416	*-1.2015e-012*	*-2.2712e-009*	*-2.2675e-009*	*3.925*	*5.2333*
0.62832	*-2.9826e-011*	*5.1217e-009*	*-3.2471e-009*	*3.75*	*5.625*
0.94248	*-6.2581e-011*	*2.9417e-008*	*-3.5257e-009*	*4.0541*	*5*
1.2566	*2.6724e-010*	*2.2475e-007*	*-4.5906e-009*	*4.2342*	*5.2222*
1.5708	*1.3386e-009*	*0.36788*	*-2.3049e-008*	*3.3333*	*5.2222*
1.885	*4.3132e-009*	*4.512e-007*	*-4.0711e-008*	*4.2342*	*5.2222*
2.1991	*1.3591e-008*	*1.4999e-007*	*1.4682e-006*	*4.2727*	*5.1648*
2.5133	*3.6835e-008*	*-3.5334e-009*	*1.4276e-005*	*4.7*	*5.1648*
2.8274	*6.2617e-008*	*-2.6268e-009*	*5.1995e-005*	*4.3727*	*6.0125*
3.1416	*-1.1477e-007*	*2.2204e-016*	*0*	*8.0167*	*6.0125*

Tuloksista havaitaan suoran (yhden askeleen) integroinnin ylivoimainen nopeus, mutta myös tarkkuuden aleneminen askelpituuden kasvaessa. Kun integrointiväli on puoli jaksoa, Gaussin menetelmän virheeksi tulee nolla kaavan symmetriaominaisuuden vuoksi. Tämä esimerkki osoittaa myös askelväliin perustuvan integroinnin tarkkuuden arvioinnin riskin: tarkkuus voi parantua, jos esimerkiksi symmetriasyistä integrointi antaa täsmälleen oikean vastauksen pidemmällä askeleella. Myös separoituvan yhtälön ratkaisualgoritmin lähtöarvona olevat integraalit ratkeavat oikein, ja kun alkuarvaus on kohdallaan, tuloksen tarkkuus on erittäin hyvä. Rungen-Kuttan menetelmän tarkkuus hiipuu hiljalleen ratkaisun edetessä.

Tuloksen valossa nähdään jaksollisten funktioiden integrointiin yleisesti pätevä ohje: laske tulos tarkasti yhdestä jaksosta, kerro täysien jaksojen määrällä ja laske tarkka arvio viimeisestä osasta, joka tulee viimeisen täyden jakson perään.

Mielenkiintoinen on myös tasan neljännesjakson arvon huono tarkkuus separoituvan differentiaaliyhtälön integrointiin perustuvassa ratkaisualgoritmissa. Tällöin alkuarvona olevan integraalin ja alkuarvauksen virheet ovat maksimissaan ja algoritmi harhautuu suoraan tasapainopisteeseen $y = 1$. Jälleen tasapainon väärälle puolelle hyppäämisen estävä toiminto poistaisi tämänkin virheen.

Edellisen tehtävän osalta tilanne, jossa y:n arvon sijasta ratkaistaan t, esimerkkinä *y(t)* = 3.

Annetulle y sitä vastaavan arvon t ratkaiseminen on pääsääntöisesti työläämpää. Siinä ratkaistaan ensimmäinen sellainen t, jolla pätee $y(t) = y_1$. Rungen-Kuttan menetelmässä laskentaa jatketaan kunnes haluttu y on saavutettu, eli pitää ohjelmoida lopetusehto. Suoralla integroinnilla voidaan menetellä kuten Rungen-Kuttan menetelmässä, mutta tällöin ei mahdollisuudesta ottaa pitkiä integrointiaskeleita saada kaikkea hyötyä käyttöön. Edellisistä poiketen separoituvan yhtälön algoritmilla käänteinen ongelma tarkoittaa vain ratkaisua t:n integraalin ylärajan suhteen, eli oleellisesti mikään ei muutu.

Yleisestikin ongelman laadullinen tarkastelu on tarpeen ennen numeerisen ratkaisemisen aloittamista. Ratkaistaan yhtälöstä $y' = \cos(t) \cdot (1 - y)$, milloin funktio on kasvava ja milloin vähenevä. Koska alkuarvo $y(0) = 2$ on tasapainopisteen $y = 1$ ”yläpuolella”, tulon ensimmäinen tekijä on negatiivinen, eikä yksikäsitteisyyslauseen perusteella ratkaisu voi hypätä derivaatan nollakohdan $y = 1$ ”alapuolelle”. Tulon merkkisäännön perusteella funktio on siten vähenevä, kun $\cos(t)>0$ eli $0 < t < \pi/2$ ja vastaavasti kasvava, kun $\cos(t)<0$ eli $\pi/2 < t < 3\pi/2$.

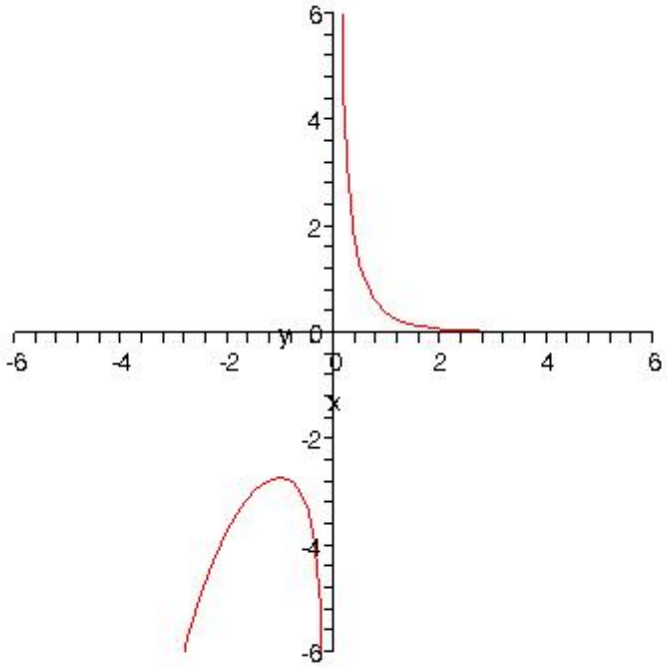
Koska funktion arvo $y = 3$ saavutetaan ensimmäisen kerran kasvavalla osuudella, mahdollisen ratkaisun pitää olla välillä $\pi/2 < t < 3\pi/2$ tai jollain myöhemmällä välillä, jolla funktio on kasvava. Näin on saatu käsitys ratkaisun alkuarvosta ja välistä, jolla ratkaisu on.

Tuloksen ratkaisemiseksi ohjelmoidaan ”regula falsi” -menetelmä (Matlab -koodi liitteenä A6) ja käytetään sitä suoran integroinnin tukena alkuarvona π ja $3\pi/2$ ja käyttäen tarkkuutena arvoa 10^{-6}. Iteraatio suppeni vastaukseen kymmenessä askeleessa. Algoritmi on käytännössä yhtä nopea kuin valmisohjelmiston käyttö, vaikka valmisohjelma saavutti noin 30% nopeammassa ajassa ensimmäisen arvon, jolla y>3. Tämä ei kuitenkaan tarkoita lopullisen ratkaisun saamista, koska tuloksen tarkkuus (askelväli) oli huono ja tarkempi vastaus pitäisi tämän jälkeen hakea esimerkiksi interpoloimalla tulosvektorin pisteiden välistä haluttu arvo.

Esimerkki 2. $y' + x\,y = e^{-x}$, $y(0) = 1$

Tämä esimerkki ei ole separoituva, eli suora integrointi ”kilpailee” taulukkolaskennalla saatavan kiinteän askeleen Rungen-Kuttan menetelmän kanssa ja valmistoiminnon kanssa.

Tutkitaan ensin karkeasti differentiaaliyhtälöä tarkastelemalla milloin se on kasvava ja milloin vähenevä. Derivaatan nollakohtakäyrä on esitetty kuvassa 8, josta nähdään myös alueet, joissa derivaatta on positiivinen eli funktio kasvava. Ensimmäisessä neljänneksessä funktio on vähenevä nollaviivan yläpuolella ja kolmannessa sen alapuolella. Funktio on kasvava muualla.



Derivaatan nollakohtakäyrä $e^{-x} - xy = 0$.

Vaikka alkuarvolla $y(0) = 1$ funktion y derivaatta on positiivinen, siirrytään lopulta vähenevään funktioon. Siten tuloskäyrät näyttävät taipuvan kohti nollaa, kun x kasvaa. Koska derivaatta on nollakohtakäyrän alapuolella kasvava ja sen yläpuolella vähenevä, ei ratkaisu sisällä hankalia tasapainopisteitä, joiden yli ratkaisukäyrä voisi hyppiä. Siten ratkaisua voidaan lähteä hakemaan ilman suurta riskiä suoraan numeerisesti.

Menetelmiä vertailtaessa käytettiin yhden askeleen ensimmäisen kertaluvun lineaarisen yhtälön ratkaisijaa ja taulukkolaskentaan ohjelmoitua Rungen-Kuttan menetelmää. Oikean tuloksen likiarvo haettiin Matlabin avulla. Taulukossa vertaillaan tarvittavien laskutoimitusten lukumäärää:

	Runge-Kutta	*Integrointi*
virhearvio	*2E-08*	*3E-10*
a(x)	*84*	*36*
b(x)	*84*	*6*
exp(x)	*0*	*36*

Tuloksena saatiin parempi tarkkuus vähemmällä vaivalla. Tässä esimerkissä funktiona a oli pelkkä x, eli sen laskemiseen ei mennyt aikaa Rungen-Kuttan menetelmässä, ja Gaussin integrointikaava laskee polynomin arvon tarkasti. Algoritmin analyysi osoittaa, että mitä mutkikkaampi b-funktio, sitä edullisemmaksi integrointi käy. Funktion a osalta sen hankaluus hidastaa molempien laskentaa, koska se lasketaan

kuudesti jokaista funktion *b* laskentakertaa kohti. Lukujen valossa myös funktion *a*:n laskentaan kuluvan ajan piteneminen antaa suoralle integroinnille etua.

Nopeusvertailu Matlabin valmistoiminnon kanssa antaa tulokseksi:

x	*y*	*ero RK - integrointi*	*aikasuhde*
0.5	*1.2429*	*2.1637e-009*	*7.5125*
1	*1.0461*	*-1.9024e-009*	*6.0125*
1.5	*0.66267*	*-2.3349e-008*	*5.2333*
2	*0.33151*	*2.0169e-007*	*6.0125*
2.5	*0.13752*	*1.668e-007*	*6.1222*
3	*0.051024*	*2.6556e-007*	*7.2333*
3.5	*0.018928*	*7.6318e-007*	*9.0125*
4	*0.0077798*	*2.2093e-006*	*9.0111*
4.5	*0.0036202*	*5.2446e-006*	*9.4556*
5	*0.0018284*	*9.6292e-006*	*11.512*

Jos halutaan lisätä integroimalla saadun ratkaisun tarkkuutta tai tallentaa välituloksia kuvaajan piirtoa varten, funktion arvot pitää ratkoa lyhyemmällä askeleella. Kun laskentaa tarkennettiin, havaittiin että ykkösen askeleella suora integrointi antaa tarkemman tuloksen kuin valmistoiminnon tulos tarkkuusasetuksella 10^{-6}, jota käytettiin aikavertailussa.

4. Toisen kertaluvun differentiaaliyhtälöistä

4.1 Johdanto

Toisen kertaluvun differentiaaliyhtälöille on lukuisia sovelluksia, koska esimerkiksi Newtonin mekaniikassa dynamiikan peruslain mukaisesti voimien vektorisumma on suoraan verrannollinen kiihtyvyyteen eli paikan toiseen derivaattaan. Vaikka moniin fysiikan ongelmiin liittyvät toisen kertaluvun differentiaaliyhtälöt ovat myös analyyttisesti ratkaistavissa, käytännön sovelluksissa tarvitaan numeerisia menetelmiä.

Tässä työssä rajoitutaan seuraaviin tapauksiin, jotka voidaan helposti kirjoittaa integraalimuodossa:

A Yhden funktion tapaukset:
$y'' = f(t)$, $y'' = f(y)$ ja $y'' = f(y')$

B Separoituvat parit:
$y'' = g(y') \cdot f(y)$ ja $y'' = g(y') \cdot f(t)$

Ratkaisu esitetään kuhunkin alkuarvoprobleemaan erikseen. Usein sovellusten toisen kertaluvun differentiaaliyhtälössä alkutilasuureita on kolme, *t, y* ja *y'* ja annettuna yhden niistä lopputila. Ratkaistavaksi jää kaksi muuta. Niiden ratkaisemisessa yleensä ensimmäinen ratkeaa osana toisen ratkaisua.

Tässä luvussa kuvataan myös toisen kertaluvun differentiaaliyhtälön kirjoittaminen ensimmäisen kertaluvun differentiaaliyhtälöryhmäksi, koska sitä tarvitaan menetelmien tehokkuusvertailuun.

Differentiaaliyhtälöryhmäle voidaan kirjoittaa sekä Eulerin että Rungen-Kuttan menetelmää vastaavat ratkaisutavat. Niissä funktiot *y* ja *y'* ratkaistaan askel kerrallaan muuttujan *t* suhteen kuten edellisissäkin differentiaaliyhtälöissä. Sekä arvo *y* että *y'* lasketaan tällöin samanaikaisesti, joten tällöin ryhmän ratkaisu ei edellytä erillisiä tarkasteluja toisin kuin useissa integrointimenetelmissä.

4.2 Yhden muuttujan tapaukset

4.2.1 Yhtälö y'' = f(t)

Tehtävän ratkaisu on suoraviivainen kaksoisintegraali. Alkuarvot olkoot t_0, y_0, ja y'_0 ja annettua aika t. Kysytään arvoja $y(t)$ ja $y'(t)$. Annetulle arvolle t saadaan $y'(t)$ suoraan integroimalla $f(t)$.

$$(59)\ y'(t) - y_0 = \int_{t_0}^{t} f(t)dt$$

Lopputulos edellyttää kaksoisintegraalin laskemista:

$$(60)\ y(t) = \int_{t_0}^{t_1}\int_{t_0}^{t} f(x)dxdt$$

Kaksoisintegraaleille on valmiitakin toimintoja, mutta yksinkertaisen integrointikaavan ohjelmointi ei ole mikään ongelma. Yksinkertaisimmin integraalin perusaskeleeksi saadaan seuraava ensimmäisen kertaluvun lineaarisesta yhtälöstä tuttu metodi: ensimmäisen integraali lasketaan integrointivälin pisteisiin, joita tarvitaan toisen integroinnin suorittamiseen.

Jos y' tai y'' on annettu, t voidaan iteroida kuten ensimmäisen kertaluvun separoituvassa yhtälössä. Ideana on siis jälleen käyttää Newtonin menetelmää integraalin ylärajan määritykseen. Näissä tapauksissa numeerinen integrointi on edellisten kohtien perusteella selvästi nopeampi vaihtoehto.

4.2.2 Yhtälö y'' = f(y')

Differentiaaliyhtälö on derivaatan y' suhteen separoituva, eli siitä saadaan suoraan integraalimuodossa oleva ratkaisu

$$(61)\ \int_{t_0}^{t} \frac{y''dt}{f(y')} = \int_{y'_0}^{y(t)} \frac{dy'}{f(y')} = t\text{-}t_0.$$

Tästä saadaan $y'(t)$, jonka integraali on suoraan $y(t)$.

Numeerisen ratkaisun perusaskel muodostuu y':n määrityksestä Gaussin integroinnin edellyttämissä pisteissä t_i, jotka saadaan numeerisesti esimerkiksi separoituvan

yhtälön ratkaisumenetelmällä; tuloksena arvot $y'(t_i)$. Tämän jälkeen y ratkeaa suoraan integroimalla $y'(t)$. Näitä askeleita otetaan, kunnes haluttu y on saavutettu.

Tilanne on erikoistapaus tilanteesta $y'' = f(y')\, f(t)$.

4.2.3 Yhtälö y'' = f(y)

Yhtälön yksi analyyttinen ratkaisu lähtee liikkeelle siitä, että yhtälön molemmat puolet kerrotaan funktiolla y'. Kohdat, joissa $y' = 0$, pitää tarkastella erikseen.

(62) $y''\ y' = f(y)\, y'$

Tämä yhtälö voidaan integroida t:n suhteen, jolloin saadaan:

(63) $$\int_{t_0}^{t} y'\, y''\, dt = \int_{t_0}^{t} f(y) y'\, dt$$

Muuttujan vaihto toimii: $dy' = y''dt$ ja $dy = y'dt$, jolloin tuloksena saadaan:

(64) $$\tfrac{1}{2} y'(t)^2 - \tfrac{1}{2} {y'_0}^2 = \int_{y_0}^{y(t)} f(y)dy$$

Yhtälöstä ratkeaa y' ja tuloksena saadaan separoituva differentiaaliyhtälö

(65) $$y'(t) = \pm\sqrt{{y'_0}^2 + 2\int_{y_0}^{y(t)} f(y)dy}\ .$$

Sen ratkaisu sisältää kaksoisintegraalin, jonka ratkaisumenetelmä on jo edellä kuvattu.

Siten tulos saadaan suoraan numeerisesta ratkaisusta yhtälölle

(66) $$\int_{y_0}^{y(t)} \frac{dy}{\pm\sqrt{{y'_0}^2 + 2\int_{y_0}^{y(t)} f(y)dy}} = t - t_0\ .$$

Tämän yhtälön yleinen ratkaisumenetelmä on numeerisesti vaikea, koska se edellyttää funktion derivaatan merkkitarkastelua. Tämäkin voidaan ohjelmoida.

Ratkaisu pitää ”viipaloida” osiin, joissa $y' \neq 0$, ja kussakin palassa valita y':n alkuarvon mukainen merkki. Jos y' on nolla, integroinnin merkki voidaan päätellä y'':n avulla. Jos y'' on positiivinen, y' on nollakohdan jälkeen positiivinen ja jos negatiivinen, niin negatiivinen. Jos $y' = 0$ ja $y'' = 0$, kyseessä on tasapainopiste.

Algoritmi toisen kertaluvun yhtälölle $y'' = f(y)$

Annettu: f(y), alku- ja loppuarvot, t_a, t_l, $y(t_a)$, ja $y'(t_a)$

Valitse iterointia varten arvio funktion loppuarvosta $y(t_b)$.

Tutki y' funktion merkki. (Nelijuuren edessä joko + tai -).

Ratkaise ulkointegraalin numeerisen integroinnin edellyttämissä pisteissä sisäintegraalin arvot numeerisella integroinnilla. Integrointiaskeleen maksimipituus voidaan ratkaista numeerisen integraalin virhekaavojen avulla, jolloin seuraava y ei ole loppuarvo $y(t_b)$, vaan integraalin virheen perusteella laskettu uusi y:n arvo.

Laske juurilauseke; vertaa nollaan.
ongelma: lyhennä askelta valitsemalla uusi alkuarvo $y(t_b)$:ksi.
ei ongelmaa: ratkaise ulkointegraali

Toista perusaskelta, kunnes integraali suurempi/pienempi kuin $t\text{-}t_0$.

Hae tarkka arvo sekanttimenetelmällä tai Newtonin menetelmällä.

Integroinnin tekevä Matlabin m. tiedosto on liitteenä A8.

Tällä differentiaaliyhtälöllä on paljon yksinkertaisia fysikaalisia vastineita, joten tutkitaan sitä numeerisesti kolmessa pelkistetyssä tilanteessa. Näistä kaksi ensimmäistä on ratkaistavissa analyyttisesti suoraan toisen kertaluvun vakiokertoimisina differentiaaliyhtälöinä. Viimeinen edustaa yleistä fysiikan kenttiin liittyvää epälineaarista toisen kertaluvun differentiaaliyhtälöä.

Esimerkki 1
y'' = y ; y(0)=1, y'(0)=1. y(1)=?, y'(1)=?

Funktion analyyttinen ratkaisu on $y = e^x$. Kun merkitään $r = y'$, saadaan toisen kertaluvun differentiaaliyhtälöstä ensimmäisen kertaluvun differentiaaliyhtälöpari:

$$(67)\quad \begin{aligned} y' &= r \\ r' &= y \end{aligned}$$

Perinteinen numeerinen ratkaisu tehdään vektorille $\mathbf{v} = [y\ r]^T$, jonka arvot voidaan laskea Rungen-Kuttan menetelmällä, jossa lasketaan vektorin **v** arvot kuten yhden muuttujan tapauksessa. Ratkaisun toteuttava Excel-taulukko on liitteenä A7. Askeleella 0,1 virhe on $2{,}1 \cdot 10^{-6}$.

Vaihtoehtoinen numeerinen ratkaisu perustuu yhtälön 65 oikean puolen numeeriseen integrointiin. Siinä otetaan arvon *y(1)* ratkaisemiseksi ykkösen askeleita, jonka jälkeen sovelletaan suppenemisen kannalta varmaa regula falsi -menetelmää. Iteroinnin kulku on seuraava:

y	*t*	*ero*
1	*0*	*1*
2	*0.6931*	*-0.3069*
3	*1.0986*	*0.0986*
2.7568	*1.0141*	*0.0141*
2.7236	*1.002*	*0.002*
2.719	*1.0002725*	*2.72E-04*
2.7184	*1.0000378*	*3.78E-05*
2.7183	*1.0000053*	*5.25E-06*
2.7183	*1.0000007*	*7.30E-07*
2.7183	*1.0000001*	*1.01E-07*

Tulos on oikein vaaditulla kuudella merkitsevällä numerolla. Laskutoimituksia funktiolle *f* tuli 3 integroinnin vuoksi ja 6 arvon *y*(1) saavuttamiseen. Funktion f(y) laskentakertoja tuli yhdeksästä iterointikierroksesta johtuen huimaavat $9 \cdot 36 = 324$ kpl, kun Rungen-Kuttan menetelmällä niitä oli vain 44. Suureen *y'* arvot ratkeavat suoraan osana laskentaa yhtälöstä 64.

Siten suora ongelma ratkeaa nopeammin Rungen-Kuttan menetelmällä, jos lopullisen ratkaisun iterointiin käytettäviin laskutoimituksiin kuluva aika on suuri verrattuna integrointiin kokonaisuutena käytettyyn aikaan.

Jos tavoitteena on piirtää funktion kuvaajaa, ei iterointia tarvita, ja integrointi on jälleen kilpailukykyinen.

Kun arvo *y* on tunnettu, integrointiin perustuva menetelmä sopii hyvin arvon *t* ratkaisemiseen, koska tulos saadaan suoraan numeerisena integraalina. Esimerkissä käytetty integrointimenetelmä antaa ajan *t* annetulle *y* suoraan yhdellä askeleella, jos $y < 2$. Tällöin suora integrointi on huomattavasti nopeampi. Jos $y > 2$, askelväliä voidaan lyhentää, jolloin integraali saadaan silloinkin lasketuksi huomattavasti nopeammin kuin saman ongelman ratkaisu Rungen –Kuttan menetelmällä edellyttää..

Esimerkki 2
***y'' = -y; y*(0) = 0, *y'*(0) = 1. *y(1)* = ?, *y'*(1) = ?**

Yhtälön analyyttinen ratkaisu on $y = \sin(t)$. Rungen-Kuttan menetelmällä saadaan seuraavat tulokset. Taulukossa sarakkeessa "oikea" on analyyttisen ratkaisun perusteella taulukkolaskennan laskema numeerinen likiarvo

t	*"Oikea"*	*R-K*	*virhe*
0	*0*	*0*	*0*
0	*1*	*1*	*0*
0.15708	*0.156434*	*0.156434*	*7.96E-07*
0.15708	*0.987688*	*0.987688*	*-2.1E-08*
0.314159	*0.309017*	*0.309015*	*1.57E-06*
0.314159	*0.951057*	*0.951057*	*-2.9E-07*
0.471239	*0.45399*	*0.453988*	*2.25E-06*
0.471239	*0.891007*	*0.891007*	*-8E-07*
0.628319	*0.587785*	*0.587782*	*2.8E-06*
0.628319	*0.809017*	*0.809019*	*-1.5E-06*
0.785398	*0.707107*	*0.707104*	*3.16E-06*
0.785398	*0.707107*	*0.707109*	*-2.4E-06*
0.942478	*0.809017*	*0.809014*	*3.29E-06*
0.942478	*0.587785*	*0.587789*	*-3.5E-06*
1.099557	*0.891007*	*0.891003*	*3.16E-06*
1.099557	*0.45399*	*0.453995*	*-4.6E-06*
1.256637	*0.951057*	*0.951054*	*2.74E-06*
1.256637	*0.309017*	*0.309023*	*-5.8E-06*
1.413717	*0.987688*	*0.987686*	*2.04E-06*
1.413717	*0.156434*	*0.156441*	*-6.9E-06*
1.570796	*1*	*0.999999*	*1.04E-06*
1.570796	*6.13E-17*	*7.9E-06*	*-7.9E-06*

Arvolle $y(\pi/2) = 1$ saadaan siten virheeksi noin miljoonasosa. Tulos hieman huononee laskennan kestäessä, kun virhe kertaantuu kierros kierrokselta.

Kun tehtävä ratkaistaan suoraan integroimalla yhdellä askeleella, lähestytään funktion $y'(t)$ nollakohtaa pisteessä $t = \pi/2$. Tässä on algoritmin akilleenkantapää. Kohdassa $y'(t) = 0$ vaihtuu integraali positiivisesta negatiiviseksi, eli sen yli ei voida tehdä yhden askeleen integrointia, vaan nollakohta pitää iteratiivisesti hakea, ja vasta tämän jälkeen edetä sen yli.

Seuraavassa taulukossa on ratkaisut yhden askeleen integroinneille väliltä [0,*y*]. Niihin on siis kulunut 36 funktion *f(y)* ratkaisukertaa ja 6 neliöjuurta eli tulos vastaa suuruusluokaltaan kymmenen askelvälin Rungen-Kuttan menetelmää.

Valittu y	*Laskettu t*	*virhe*
0	*0*	*0*
0.1	*0.10017*	*alle 1E-18*
0.2	*0.20136*	*-2.78E-17*
0.3	*0.30469*	*-1.33E-14*
0.4	*0.41152*	*-1.24E-12*
0.5	*0.5236*	*-5.57E-11*
0.6	*0.6435*	*-1.66E-09*
0.7	*0.7754*	*-4.10E-08*
0.8	*0.92729*	*-1.01E-06*
0.9	*1.1197*	*-3.35E-05*
1	*1.4761*	*-0.00448*
1.1	*X*	*X*

Tuloksista nähdään suoran integroinnin ylivoimainen tarkkuus kunnes aletaan lähestyä funktion *y'(t)* nollakohtaa. Esimerkiksi kymmenen askeleen Rungen-Kuttan menetelmällä kohdassa $y = 0{,}4$ virhe on suuruusluokkaa 10^{-8} kun se suorassa integroinnissa on 10^{-12}. Siten suora integrointi kannattaa käänteisessä probleemassa ja kun $y' \neq 0$.

Esimerkki 3
***y'' = 1/y*^{2} *; y*(0) = 1, *y'*(0)=1. *y(t)* =?, *y'(t)* =?**

Kyseessä on tyypillinen voimakenttä, jonka voimakkuus on kääntäen verrannollinen etäisyyden neliöön. Tässä kyseessä voisi olla esimerkiksi elektronin poistuminen negatiivisen varauksen luota.

Tehtävä ratkaistaan jälleen Rungen-Kuttan menetelmällä ja integrointimenetelmällä. Tässä tapauksessa *y'(t)* ei lähesty nollaa ja tehtävän voi helposti ratkaista integroimalla.

t	*integrointi*	*R-K*	*ero*
0	*1*	*1*	*0*
0		*1*	
0.1	*1.104683*	*1.104683*	*1.66E-07*
0.1		*1.090653*	
0.2	*1.21759*	*1.21759*	*3.01E-07*
0.2		*1.165079*	
0.3	*1.337268*	*1.337267*	*4.27E-07*
0.3		*1.226544*	
0.4	*1.462557*	*1.462556*	*5.53E-07*
0.4		*1.277704*	
0.5	*1.592536*	*1.592535*	*6.88E-07*
0.5		*1.320658*	
0.6	*1.72647*	*1.726469*	*8.47E-07*
0.6		*1.357042*	
0.7	*1.863768*	*1.863767*	*1.07E-06*
0.7		*1.388129*	
0.8	*2.003953*	*2.003952*	*1.42E-06*
0.8		*1.414909*	
0.9	*2.146634*	*2.146632*	*2.03E-06*
0.9		*1.43816*	
1	*2.29149*	*2.291486*	*3.07E-06*
1		*1.458492*	

Tulokset osoittavat molempien menetelmien saavuttavan vaaditun tarkkuustason. Kun laskentaa jatkettiin, havaittiin, että integrointi antaa yhdelläkin askeleella vielä tarkkoja arvoja kohdassa $t = 2$. Siten integrointi on tarkempi, jos halutaan ratkoa yksittäisiä pisteitä esimerkiksi kuvanpiirtoa varten. Haluttaessa ratkaista annettua t vastaavaa etäisyyttä y joudutaan lopussa käyttämään numeerista hakua, minkä vuoksi pienillä muuttujan t arvoilla Rungen-Kuttan menetelmä on tehokkaampi. Toisaalta haluttaessa saada annetulle etäisyydelle y sitä vastaava aika t kannattaa käyttää integrointia.

4.3 Tapaukset *y’’ = f(*) · g(*)*

4.3.1 Yhtälö y’ ’ = g(y’) f(t)

Tässä funktio *g(y’) f(t)* on funktion (muuttujan) *y’* suhteen separoituva differentiaaliyhtälö, jonka ratkaisu on

$$(68)\ \int_{t_0}^{t} \frac{y''dt}{f(y')} = \int_{y'_0}^{y(t)} \frac{dy'}{f(y')} = \int_{t_0}^{t} f(t)dt\,.$$

Separoituvissa yhtälöissä numeeriseen integrointiin perustuva algoritmi on edellisen luvun tulosten perusteella nopeampi, joten tästä yhtälöstä saadaan joko analyyttisesti tai numeerisesti tuloksena parit *t* ja *y’(t)*. Näistä saadaan suoraan numeerisella integroinnilla *y(t)*. Siten ratkaisu noudattaa esimerkin 1 kaksoisintegraalilla ratkaistavan tehtävän ratkaisua.

4.3.2 Yhtälö y’’ = g(y’) f(y)

Ratkaisu muistuttaa aiempia, koska siinä yhtälö kirjoitetaan muotoon

$$(69)\ \int_{t_0}^{t} \frac{y'y''dt}{g(y')} = \int_{t_0}^{t} f(y)y'dt\,.$$

Tämä sievenee kahdeksi integraaliksi, joista saadaan ratkaistuksi *y’ y*:n funktiona esimerkiksi separoituvan yhtälön numeerisella menetelmällä.

$$(70)\ \int_{y'_0}^{y'} \frac{y'dy'}{g(y')} = \int_{y_0}^{y} f(y)dy$$

Merkitään *G(y) = y’*. Saadaan on separoituva yhtälö, jonka ratkaisu on

$$(71)\ \int_{y(t_0)}^{y(t)} \frac{dy}{G(y)} = t - t_0\,.$$

Saadaan siis numeerinen algoritmi, jossa perusaskeleena ratkaistaan *y' y*:n suhteen integrointipisteissä, ja tämän jälkeen ratkaistaan suoraan uusi integraali numeerisesti. Seuraava arvaus voidaan hakea jälleen Newtonin menetelmällä tai edetä integrointiaskelia, kunnes erotus $t - t_0$ on saavutettu. Tämän jälkeen tarkka ratkaisu saadaan esimerkiksi sekanttimenetelmällä.

Esimerkki 4
***y'' = y' · y ; y*(0) = *y'*(0) =1**

Tehtävän analyyttinen ratkaisu on

$$(72)\ y(t) = \tan(\frac{x}{2} + \frac{\pi}{4}).$$

Tehtävässä funktio kasvaa äärettömyyksiin, kun tangentti lähestyy arvoa π/2. Tällaiset ”räjähtävät” differentiaaliyhtälöt ovat hankalia numeerisille menetelmille. Alla tulokset Rungen-Kuttan menetelmällä laskettuna ja sarake ”oikea” taulukkolaskennan likiarvo analyyttisestä ratkaisusta.

t	*R-K*	*”oikea”*	*virhe*
0	*1*	*1*	*0*
0	*1*	*1*	*0*
0.15708	*1.170844*	*1.17085*	*5.71E-06*
0.15708	*1.185442*	*1.185444*	*2.05E-06*
0.314159	*1.376365*	*1.376382*	*1.73E-05*
0.314159	*1.447204*	*1.447214*	*9.26E-06*
0.471239	*1.631809*	*1.631852*	*4.29E-05*
0.471239	*1.831438*	*1.83147*	*3.22E-05*
0.628319	*1.962506*	*1.962611*	*0.000105*
0.628319	*2.425813*	*2.42592*	*0.000107*
0.785398	*2.413937*	*2.414214*	*0.000276*
0.785398	*3.413831*	*3.414214*	*0.000382*
0.942478	*3.076842*	*3.077684*	*0.000842*
0.942478	*5.234465*	*5.236068*	*0.001603*
1.099557	*4.162012*	*4.1653*	*0.003288*
1.099557	*9.165793*	*9.174861*	*0.009068*
1.256637	*6.293824*	*6.313752*	*0.019927*
1.256637	*20.34188*	*20.43173*	*0.089845*
1.413717	*12.40087*	*12.7062*	*0.305331*
1.413717	*78.10608*	*81.22382*	*3.117744*
1.570796	*70.90023*	*1.63E+16*	*1.63E+16*
1.570796	*2591.367*	*1.33E+32*	*1.33E+32*

Tuloksista nähdään, että menetelmä ei luonnollisesti selviä äärettömyydestä, vaan ”jää” jälkeen. Viimeinen arvo on taulukkolaskennan tarkkuudella laskettu y(π/2). Oikeaa arvoa ei ole, koska funktio lähestyy ääretöntä.

Integrointiin perustuva ratkaisu toteutettiin Matlabilla. Yhden askeleen integroiva .m -tiedosto on liitteenä A9 . Liitteenä A10 on sekä yhdellä askeleella että lyhyin perättäisin askelein ratkaisun tekevä .m -tiedosto.

Jotta virhe olisi vertailukelpoinen edellisen taulukon Rungen-Kuttan menetelmän virheen kanssa, virhesarakkeeseen kirjoitettiin laskettua t-arvoa vastaavan y-arvon ja alkuarvon ”valittu y” erotus.

Kootaan tulokset:

Valittu y	*Laskettu t*	*Laskettu t*	*"oikea t"*	*virhe: y-y("oikea t")*	
	yksi askel	*0,5 askel*		*yksi askel*	*0,5 askel*
1	*0*	*0*	*0*	*0*	*0*
1.5	*0.39479*	*0.39479*	*0.39479*	*-4.37E-14*	*-4.37E-14*
2	*0.6435*	*0.6435*	*0.64350*	*-2.32E-10*	*-7.04E-14*
2.5	*0.80978*	*0.80978*	*0.80978*	*-1.33E-08*	*-1.02E-13*
3	*0.9273*	*0.9273*	*0.92730*	*-9.28E-08*	*-1.40E-13*
3.5	*1.0142*	*1.0142*	*1.01420*	*3.55E-07*	*-1.84E-13*
4	*1.0808*	*1.0808*	*1.08084*	*6.28E-06*	*-2.36E-13*
4.5	*1.1335*	*1.1335*	*1.13346*	*3.52E-05*	*-2.98E-13*
5	*1.176*	*1.176*	*1.17601*	*1.28E-04*	*-3.63E-13*
5.5	*1.2111*	*1.2111*	*1.21109*	*3.55E-04*	*-4.41E-13*
6	*1.2405*	*1.2405*	*1.24050*	*8.22E-04*	*-5.14E-13*
6.5	*1.2656*	*1.2655*	*1.26550*	*1.66E-03*	*-6.11E-13*
7	*1.2871*	*1.287*	*1.28700*	*3.01E-03*	*-6.96E-13*
7.5	*1.3059*	*1.3057*	*1.30569*	*5.02E-03*	*-7.99E-13*
8	*1.3223*	*1.3221*	*1.32209*	*7.78E-03*	*-9.15E-13*
8.5	*1.3369*	*1.3366*	*1.33658*	*1.13E-02*	*-1.04E-12*
9	*1.3499*	*1.3495*	*1.34948*	*1.57E-02*	*-1.15E-12*
9.5	*1.3615*	*1.361*	*1.36104*	*2.06E-02*	*-1.27E-12*
10	*1.372*	*1.3715*	*1.37146*	*2.59E-02*	*-1.42E-12*
100	*1.0139*	*1.5508*	*1.55080*	*-9.65E+01*	*-1.37E-10*
1000	*0.15934*	*1.5688*	*1.56880*	*X*	*-1.42E-08*

Tuloksista havaitaan, että yhdellä askeleella saadaan oikea tulos miljoonasosan tarkkuudella y:n arvoon 4 saakka. Jos edetään integroiden askel kerrallaan askelvälillä 0,5 virhe ei edes vielä kohdassa $y = 1000$ ole merkittävä, joskin tarvittava laskenta-aikakin on pitkä.

Yhden askeleen integroinnissa äärettömän rajamailla tulokset vaihtelivat Newtonin supetessa ajoittain kohti vääriä arvoja. Tulos ajaksi t ylittää kriittisen rajan $\pi/2$ kohdassa $y = 1000$. Vaikka tuloksen virhe ajan t suhteen esiintyy vasta kolmannessa desimaalissa, joudutaan määrittelyjoukon ulkopuolelle eikä virhettä voida arvioida edellisessä kohdassa esitetyllä erotuksella.

Tarkastelua jatkettiin tekemällä uusi laskenta edelliseen taulukkolaskentaohjelmiston taulukkoon. Kun arvioidaan menetelmien potentiaalista nopeutta, havaitaan että

arvoon $y(\pi/4) \approx 2{,}4$ pääsemiseksi Rungen-Kuttan menetelmällä tarvitaan 44 funktioiden *f* ja *g* laskukertaa ja tällöin saavutettu virhe on $2 \cdot 10^{-5}$.

Integroinnissa tulos arvoon $y = 2{,}4$ saatiin vastaavasti edellisen taulukon tapaan ratkaistulla *y*:n virhetarkkuudella $7 \cdot 10^{-9}$. Tällöin funktio *g(y)* tulee lasketuksi 42 kertaa *y'* integroinnin yhteydessä ja kokeilussa koodi ”Lappismeth” sisälsi 14 iterointia, joista kussakin oli 6 funktion *f(y')* ratkaisua ja askelpituudessa vielä 18 lisää. Siten funktio *f* tuli lasketuksi yli 100 kertaa ja funktio *g* 42 kertaa.

Rungen-Kuttan menetelmässä tarkkuussuuruusluokka $9 \cdot 10^{-8}$ saavutetaan 80 jakovälillä ja $6 \cdot 10^{-9}$, kun jakovälejä oli 160. Tällöin funktioiden laskutoimituksia tulee nelinkertaisesti.

Lopputuloksena voidaan siis todeta, että hyvin ohjelmoituna integrointi kilpailee hyvin aikavertailussa Rungen-Kuttan menetelmän kanssa, jos tavoitteena on laskea pistepareja (*y , t*) kuvaajan piirtoa varten tai jos halutaan ratkaista annetulle suureen arvolle *y* sitä vastaava suureen arvo *t*.

Annetulle luvulle *t* funktion arvoa *y* laskettaessa Rungen-Kuttan menetelmä on vahvoilla. Integrointimenetelmällä jokainen iterointiaskel tuo yli 100 funktion *f* laskentakertaa. Kun tarkan arvon *t* iterointi annetulle arvolle *y(t)* vie 10 askelta integroinnin alkuaskelten lisäksi, funktio *f* lasketaan iterointivaiheessa 1000 kertaa, mikä vaikuttaa merkittävästi nopeuskilpailun tulokseen.

Toisen kertaluvun yhtälöiden osalta siis nähdään selvästi, että sopiva menetelmä pitää valita tilanteen mukaan. Integrointimenetelmien soveltamisen osalta lisätutkimukset ja ohjelmakoodien optimointi ovat kuitenkin vielä tarpeen.

Liitteet

Liite1 Harjoitustehtävät

Harjoitustehtäviä lukuun Numeerisista menetelmistä

Virhetarkastelut

1. Tutki jakolaskun virhettä, kun lähtöarvojen suhteellinen virhe on annettu. (vrt. yhtälö 9 ja kertolaskun virhe).

2. Kannattaako summa $\Sigma(i^{-1})$ laskea numeerisesti arvosta $i = 1$ arvoon $i = 10^{10}$ vai arvosta $i = 10^{10}$ arvoon $i = 1$?

3. Toisen asteen yhtälön $ax^2 + bx + c = 0$ ratkaisukaavassa:

$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ esiintyy vähennyslasku jommankumman juuren laskennassa riippumatta luvun $-b$ merkistä. Tämä voi aiheuttaa ongelmia.

   1) Kirjoita ratkaisualgoritmi, jossa vähennyslasku vältetään. Vihje: Kun toinen juuri on ratkaistu ilman vähennyslaskua, toisen juuren saa juurten tulon kaavasta $x_1 x_2 = c/a$.

   2) Tarkastele, mikä suhteen $b^2 / 4ac$ pitää olla, että omassa laskimessasi perinteinen ratkaisukaava antaa toiseksi juureksi arvon 0.

4. Funktion arvossa esiintyvää virhettä annetun x:n virheen suhteen voi tutkia derivaatan avulla: $f(x+h) - f(x) \approx f'(x) \cdot h$. Tutki funktioiden $\sin(x)$ $\cos(x)$, e^x, x^2, ja $1/x$ suhteellista ja absoluuttista virhettä kun x:n suhteellinen virhe on 1% kohdassa $x = 5$.

5. Tilastollinen tehtävä: Mikä on n liukuluvun summan virheen jakauma, kun liukuluvut ovat samaa suuruusluokkaa ja kunkin niistä virhe on tasajakautunut?

Yhtälön ratkaisu

1. Tutki, montako juurta ja millä alueille ne ovat:
   a. $xe^{-x} = 0.001$
   b. $x^3 - 5x^2 = 6$
   c. $x = 2\cos(x)$
   d. $e^x = \sin(x)$ (vain idea).

2. Tarkkuuden osoitus:
   a. Osoita, että yhtälöllä $\tan(x) - 1 = x$ on juuri välillä [0 ; 1,5].
   b. Olkoon $f(z) = e^z$ ja funktion $z(x) = x^2$, Laske arvot $f(x=100\ 000)$ ja $f(x=1\ 000\ 000)$. Onko välillä nollakohtaa?

3. Numeerisesti ratkaistavan yhtälön ratkaisualgoritmi voidaan muotoilla: tutki, muuta sopivaksi funktioksi, selvitä nollakohtien lukumäärä ja sijaintialueet, valitse ratkaisumetodi, hae tulokset ja osoita tarkkuus seuraaville yhtälöille. Ratkaise tehtävät 1a-1d; kohdassa 1d vain suurin juuri.

4. Ratkaise $\ln(x+5) = x$.

5. Menetelmäkohtaisia ”jippoja”:
   a. Kiintopistemenetelmässä ei ole samantekevää, miten yhtälö kirjoitetaan muotoon $x = g(x)$. Vertaa suppenemista, kun neljännen asteen yhtälö $x^5 - 2x^3 + 2x = 1$ kirjoitetaan muodossa $x=x^5-2x^3-1$ tai muodossa $x = 1/(x^4 -2x^2 + 2)$
   b. Ratkaise yhtälö 4 käyttäen ”vaimennettua Newtonin menetelmää”. Vaimennettu Newton parantaa jonkin verran juureen suppenevan alkuarvauksen valintaprobleemaa, mutta sen hintana on hitaampi suppeneminen kohti juurta. Alla olevassa kaavassa λ on vaimennuskerroin, joka saa arvoja nollan ja yhden väliltä. Vaimennetun Newtonin kaava on:

      $$x_{i+1} = x_i - \lambda f(x_i) / f'(x_i)$$

6. Teoriatehtävä: Johda Newtonin suppenemiskaava kiintopisteen suppenemisen kaavasta. Miten vaimennus vaikuttaa suppenemiseen?

Numeerinen integrointi

1. Ohjelmointiharjoitus: Kirjoita laskimeesi tai taulukkolaskentaan ratkaisukaavat Simpsonin ja Gaussin integrointikaavoille.

2. Johda virhekaavat (yhtälöt 21-23) Simpsonin säännölle käyttäen esimerkkinä vastaavien puolisuunnikassäännön virhekaavojen johtamista. Oleta kaava 20 tunnetuksi.

3. Ratkaise seuraavien funktioiden integraalit välillä [0,2] käyttäen puolisuunnikassääntöä 3 ja 6 laskentapisteessä, Keplerin kaavaa, Simpsonin sääntöä käyttäen 4 jakoväliä ja Gaussin integrointia 2,3 ja 6 laskentapisteellä:
   a. x^5
   b. $\sin(x)$
   c. e^x

   Tee virhetarkastelu käyttäen sekä derivaattaan perustuvaa virhekaavaa että laskenta-askelten lukumäärän puolittamiseen perustuvaa virhekaavaa. Tarkastele Simpsonin sääntöä ja puolisuunnikassääntöä. Vertaa näin saatuja virheitä analyyttisen ratkaisun perusteella laskettuun virheeseen.

4. Osoita, että Kepler integroi oikein funktion x^3 välillä $[a,b]$.

5. Halutaan, että funktio x^2 integroidaan oikein välillä [0,1] käyttämällä vain yksi funktion $f(x)$ laskentakerta. Valitse laskentapiste siten, että tulos on oikein. Laajenna menetelmä välille [-1,1] kaksipisteiseksi siten, että otetaan saadun pisteen vastaluku ja lasketaan funktion arvo siinäkin pisteessä. Sijoituskeinolla saadaan mikä tahansa integrointiväli muutetuksi väliksi [-1,1]. Kokeile, integroiko saatu menetelmä oikein funktion x^3 välillä $[a,b]$.

6. Ratkaise $f(3)$, kun $f(0)=1$ ja $f'(x) = e^{\sin(x)}$.

Harjoitustehtäviä differentiaaliyhtälöistä

Perustehtäviä differentiaaliyhtälöistä

1. Osoita, että $y(x) = Ce^x$ on differentiaaliyhtälön $y' = y$ yksi ratkaisu.

2. Ratkaise edellinen differentiaaliyhtälö separoimalla.

3. Ratkaise differentiaaliyhtälö $y'(t) = (1+\cos(t)) \cdot y$ separoimalla.

4. Ratkaise differentiaaliyhtälö $y' = t / y$, $y(0) = 1$.

5. Ratkaise differentiaaliyhtälö $y' = y / t$, $y(1)=1$.

6. Ratkaise differentiaaliyhtälö $y' = y - t$, $y(1)=1$. (Sijoitus $z = y - t$ johtaa z:n suhteen separoituvaan differentiaaliyhtälöön.)

7. Ratkaise differentiaaliyhtälö $y' = (1-y)^{1/2}$.

8. Funktion $f(x)$ pisteeseen x piirretyn tangentin kulmakerroin on kaksinkertainen verrattuna funktion arvoon pisteessä $(x,f(x))$. Funktio kulkee pisteen (1,1) kautta. Ratkaise $f(x)$.

9. Millä x-y –tason alueella funktio y on kasvava, kun $y'(x)=1-x^2-y^2$.

10. Hahmottele suuntakenttä edellisen tehtävän funktiosta kokonaislukupisteisiin, kun x ja y ovat välillä [-3,3].

11. Ratkaise funktion y käänteisfunktio $x(y)$, kun $y'(x) = y \cdot (1-y^2)$. (Osamurtokehitelmä!)

12. Tutki, onko yllä olevissa tehtävissä tasapainopisteitä, joihin y jää. Onko tasapaino sellainen, että jos y poikkeaa tasapainosta vähän, se palaa takaisin kohti tasapainoa (stabiili eli vakaa tasapaino) vai johtaako poikkeama y-arvosta siihen, että arvot etääntyvät tasapainosta?

13. Oletetaan, että ilmanvastusvoima ja siten ilmanvastuksen aiheuttama nopeuden muutos ajan suhteen eli kiihtyvyys a on a) suoraan verrannollinen nopeuteen $a = -kv$

b) verrannollinen nopeuden neliöön $a = -kv^2$. Jotta nopeus pysyisi vakiona vapaassa pudotuksessa, ilmanvastuksen aiheuttaman kiihtyvyyden on oltava yhtä suuri kuin putoamiskiihtyvyys g=9,8m/s^2.

1) Kirjoita tilannetta kuvaava differentiaaliyhtälö sekä kohtaan a) että b).
2) Ratkaise vakion k arvo, kun vapaa putoamisnopeus on 40 m/s.
3) Ratkaise kumpikin syntyvä differentiaaliyhtälö.
4) Ratkaise, missä ajassa saavutetaan nopeus 30m/s, kun kappale pudotetaan korkeasta tornista.

Olemassaolo- ja yksikäsitteisyyslause

1. Tutki kaikkien edellisten tehtävien osalta ratkaisun olemassaolo ja yksikäsitteisyys.
2. Onko yhtälön $y' = y(1-y^2)^{½}$ ratkaisu yksikäsitteinen?

Numeerisia tehtäviä.

1. Tee ohjelmakoodi taulukkolaskentaan tai laskimeen, joka ratkaisee differentiaaliyhtälön Eulerin menetelmällä.

2. Ratkaise perustehtävien differentiaaliyhtälöistä numeerisesti $y(2)$. Jos alkuarvoja ei ole annettu, käytä pistettä (1,1).

3. Opettele käyttämään kuvanpiirtotoimintoa ja piirrä samaan kuvaan sekä analyyttinen että numeerinen ratkaisu $y' = y$ ja $y(0) = 1$.

4. Tee ratkaisukoodi, joka käyttää Rungen-Kuttan menetelmää. Vertaa tarkkuutta edellisiin.

5. Origokeskeisen ympyrän yhtälö on $y^2 + x^2 = r^2$, jossa r on ympyrän säde. Derivoi kaava muuttujan x suhteen ja ratkaise y', jolloin saat ympyrää kuvaavan differentiaaliyhtälön. Ratkaise sitten alkuarvotehtävä $y(0)=r$ numeerisesti ja katso miten eri algoritmit selviävät kohdasta $x \to r$.

6. Olkoon $y(0)=1$. Ratkaise $y(2)$, kun $y' + xy = \sin(x)$.

7. $y(1)=1$ ja $y''=1/y$. Ratkaise

a. $y(2)$ ja $y'(2)$,
b. muuttujan t arvot, kun
    i. $y'(t) = 2$
    ii. $y(t) = 2$,

8. Kirjoita edellinen toisen kertaluvun DY ensimmäisen kertaluvun systeemeiksi.

9. Ohjelmointitehtävä: Kirjoita Eulerin menetelmä 1. kertaluvun systeemeille.

10. Lancesterin yhtälöt kuvaavat taistelua. Olkoon y_p punaisen joukon vahvuus ja y_s sinisen joukon vahvuus. Tappiot riippuvat vastustajan määrästä, omien joukkojen määrästä ja taisteluteknisestä kertoimista α ja β. Ratkaise, milloin ja kumman joukon (sininen vai punainen) vahvuus on pudonnut ensin 60% alkuperäisestä, kun $y_p = 200$ ja $y_s = 150$ sekä $\alpha=0{,}01\ d^{-1}$ ja $\beta=0{,}03\ d^{-1}$ ja tilannetta kuvaava differentiaaliyhtälösysteemi on:

$$y_p'(t) = -\alpha \cdot y_p \cdot y_s$$
$$y_s'(t) = -\beta \cdot y_s \cdot y_p$$

11. Ratkaise numeerisesti pakonopeus maan gravitaatiokentästä. Lähtöarvot löytyvät fysikaalisista taulukoista.

12. Teoriatehtäviä: Johda Newtonin iteraation suppenemiskaavasta differentiaaliyhtälön integroinnin ylärajan ratkaisevalle menetelmälle suppenemisalueen arvio. Miten tulokseen vaikuttaa integraalin $\int f(t)\ dt$ arvo? Pohdi hyvän alkuarvauksen merkitystä ja miten vaimennettu Newtonin menetelmä toimisi tässä tapauksessa.

Liitesarja A Ohjelmakoodit

Liite A1Gaussin integrointi

```
function vast=Gaussint(integrated,a,b);


% gaussint integroi funktion staattisella gaussin integrointikaavalla.
% käytetään nopeuttamaan integrointia.
% kaavaa sisältää sijoituksen ja kerroin vektorin arvot.


xvekt=[-0.93246951,-0.66120939,-0.23861919,0.23861919,0.66120939,0.93246951];
Avekt=[0.17132449,0.36076157,0.46791393,0.46791393,0.36076157,0.17132449];
m=(b+a)/2;
h=(b-a)/2;
xvalues=h.*xvekt + m;

for i=1:6
   gxvalues(i) = feval(integrated,xvalues(i));
end

vast=(Avekt*gxvalues')*h;
```

Liite A2 Integrointiin ja suppenemiskontrolloituun Newtonin iterointiin perustuva algoritmi

```
function r=Lappismeth(Ix,y0,y1,gy)
% Lappismeth.m
% ratkaisee separoituvan DY:n y'(x) = f(x) g(y)
% siihen sijoitetaan alkuarvot Ix  y0 y1 ja funktio gy
% x0, x1 ja y0 sekä alkuarvaus.
% koodiriville asetetaan ratkaistavan differentiaaliyhtälön f(x) ja g(y).
% Esa Lappi 1.4.2004
% algoritmi
% simulointi 1


% nollataan laskurit

   Iy=0;              % alkuarvo Iy:lle
                      %maxy=1e305;
   n=0;               % kierroslaskuri
   lopetusarvo=1E-12     ;   % simuloinnin lopetusehto
   virheraja=lopetusarvo/100;
   virhe= 5*lopetusarvo;              % alkuehto lopetukselle

   % tarkastetaan, ollaanko jo tasapainotilassa:
   gnolla=feval(gy,y0);
   if (gnolla==0)
      r=y0
      break
   end


  % y1= y0+Ix*gnolla;          % jos ei muuta alkuehtoa

                      aika(1)=0; %alustetaan alkuarvot
                      lampo(1)=y0; % alustetaan alkuarvot

   pieni=min(lopetusarvo,1e-12); % jos tehtävässä tarvitaan "epsilonia"


   %Tarkastellaan suppenemista:

   gyy=feval(gy,y1) ;          %   funktion g(y) arvo kohdassa y1
   vertailu=gnolla*gyy;        %   jos vaihtaa merkkiä, ohitettu ratkaisukohta
   while (vertailu<=0)
      y1=y0+0.1*(y1-y0);
```

```
      gyy=feval(gy,y1);
      vertailu=gnolla*gyy;
   end;


% askelpituuuden arviointi


   puoli=(y0+y1)/2;
   Ikoko=gaussint(gy,y0,y1);                % kokoaskel
   Ipuoliaskel1=gaussint(gy,y0, puoli);     % puoliaskel 1
   Ipuoliaskel2=gaussint(gy,puoli,y1);      % puoliaskel 2
   Ivirhe=abs(Ikoko-(Ipuoliaskel1+Ipuoliaskel2));% virhearvio
   if (abs(Ivirhe)>0)
      askelpituus=0.5*(y1-y0)*(virheraja/(Ivirhe))^(1/10);     %askelpituus
      if (abs(y1-y0)>abs(askelpituus))
         y1=y0+askelpituus;
         gyy=feval(gy,y1);
      end
      if (abs(askelpituus)<lopetusarvo)
         break;
      end;

   else
      askelpituus=Ikoko/1E-15;
   end


%looppi

erotus=sign(y1-y0);
while (virhe>lopetusarvo)

   n=n+1 ;            %kierroslaskuri

% algoritmin sydän:

      I=apugaussint(gy,y0,y1);
      Iy=I+Iy;
      y2 = y1 - (Iy-Ix)*gyy;    % y2=y1-(Iy-Ix)*g(y)


         % talletetaan tulokset piirtoa varten:
            aika(n+1)=Iy;  % autonomisella
            lampo(n+1)=y1;


   virhe=abs(y2-y1);

      if (virhe>abs(askelpituus))
         y2=y1-sign(erotus)*askelpituus;
         virhe=abs(y2-y1);
```

```
    end;

  % tutkitaan derivaattaa:
        gyy=feval(gy,y2);          % gyy tutkii g(y):n arvon
        vertailu=gnolla*gyy;       % g(y) ei saa vaihtaa merkkiä
        while (vertailu<0)         % jos g(y) vaihtaa merkkiä,
          y2=y1+0.5*(y2-y1);       % lyhennetään askelta
          gyy=feval(gy,y2);
          vertailu=gnolla*gyy;

        end;


  if (abs(gyy)<pieni)
      y2=y1;
      break;
  end;
  virhe=abs(y2-y1);
  erotus=Iy-Ix;
  y0=y1;
  y1=y2;

end; %ydinlooppi


r=y2;
%aika=sqrt(2*aika);
%plot(aika,lampo,'k+')
```

Liite A3 Perusmenetelmä integraalin ylärajan ratkaisemiseen Newtonin iteroinnilla

```
function r=NewtonLappi(Ix,y0,y1,gy)
% NewtonLappi.m
% ratkaisee separoituvan DY:n y'(x) = f(x) g(y)
% siihen sijoitetaan alkuarvot Ix  y0 y1 ja funktio gy
% x0, x1 ja y0 sekä alkuarvaus.
% koodiriville asetetaan ratkaistavan differentiaaliyhtälön f(x) ja g(y).


% Esa Lappi 1.10.2003


% nollataan laskurit

Iy=0;               % alkuarvo Iy:lle
%x0=0;               % x:n alkuarvo
%x1=2;               % x:n loppuarvo
%y0=1;               % y:n alkuarvo
n=0;                % kierroslaskuri
lopetusarvo=1E-9     ;   % simuloinnin lopetusehto
virhe= 5;               % alkuehto lopetukselle
%y1= y0+Ix*y0;          % jos ei muuta alkuehtoa


% Ohjelmassa voidaan käyttää vaimennettua Newtonin menetelmää
% jos suppeneminen on ongelma


damping=1;     %vaimennuskerroin, jos 1, vaimennus ei käytössä.
```

```
% looppi eli silmukka alkaa
while (virhe>lopetusarvo)
   n=n+1;              %kierroslaskuri
   %I=quadl(gy,y0,y1);
   I=gaussint(gy,y0,y1);
   Iy=I+Iy;
   %damping = n/(n+10);
   y2 = y1 - damping*(Iy-Ix)/feval(gy,y1);    % y2=y1-(Iy-Ix)*g(y)
   virhe=abs(y2-y1);
   y0=y1;
   y1=y2;
end;
r=y2;
```

Liite A4 Algoritmien laskenta-aikoja vertaileva ohjelma

```
%simuLvsRK
%laskee saman arvon n kertaa.
%eli ensin lappi
% sitten RK

%c Esa Lappi 1.10.2003

loppiluku=10
loppuaika=1.0;
for i=1:5
   tic;
   for looppi=1:loppiluku;
      Ix=loppuaika; % tähän Ix
      yalku=1;
      yarvaus=0.5;
      tulos=Lappismeth(Ix,yalku,yarvaus,@funktiogy);
   end
   tulos;
   aika1=toc;

   tic;
   for looppi=1:loppiluku;
      Ix=loppuaika;            %0.5*loppuaika^2;
      yalku=1;
      yarvaus=0.55;
      tulosNL=Newtonlappi(Ix,yalku,yarvaus,@integroitava);
   end
   tulosNL;
   aika3=toc;

   tic;

   for looppi=1:loppiluku;
      yalku=1;
      options=odeset('RelTol',1e-6,'AbsTol',1e-6);
      [aikat,arvoty]=ode45(@ypilkku,[0,loppuaika],yalku,options);
   end
   aika2=toc; %valmisohjelmiston aika
   apu=size(arvoty);

   tulos2=arvoty(apu(1));
   oikea=1./(1+loppuaika);
   virheRK=oikea-tulos2;
   virheLM=oikea-tulos;
   virheNL=oikea-tulosNL;
   aikasuhde=aika2/aika1;
```

```
    aikasuhde2=aika2/aika3;
    tulosvirheaikasuhde=[oikea,virheRK,virheLM,virheNL,aikasuhde,aikasuhde2]

   %hold on
    %plot(aikat,arvoty,'r.')
    %hold off

  kuvatus(i,:)=[loppuaika,aikasuhde];
  loppuaika=loppuaika+0.5;
end
```

Liite A5 Lineaarisen differentiaaliyhtälön ratkaisuohjelma

```
function vast=Enkerlin(afunk,bfunk,a,b,ynolla);

% enkerlin
% ratkaisee yhtälön y'+ay=b
% tarvitsee lähtöarvona:
% funktion a(x),funktion b(x), x:n alarajan a,
% x:n ylärajan b ja y:n alkuarvon ynolla
% integroi funktiot staattisella gaussin integrointikaavalla.
% käytetään nopeuttamaan integrointia.
% kaavaa sisältää sijoituksen ja kerroin vektorin arvot.
% c Esa Lappi 1.8.2004

xvekt=[-0.93246951,-0.66120939,-0.23861919,0.23861919,0.66120939,0.93246951];
Avekt=[0.17132449,0.36076157,0.46791393,0.46791393,0.36076157,0.17132449];
m=(b+a)/2;
h=(b-a)/2;
xvalues=h.*xvekt + m;

%integroivan tekijän kaskeminen
vanha=0;
ala=a;
for i=1:6
   vanha=vanha+gaussint(afunk,ala,xvalues(i));
   ala=xvalues(i);
   myy(i)=exp(vanha);
   barvot(i)=feval(bfunk,xvalues(i));
   isoint(i)=myy(i)*barvot(i);
end


myyloppu=exp(vanha+gaussint(afunk,xvalues(i),b));
isointvast=(Avekt*isoint')*h;
vast=ynolla/myyloppu+isointvast/myyloppu
```

Liite A6 Käänteisen probleeman ratkaisussa käytetty ”regula falsi” -menetelmä

```
function tulos=sekantti(alax,ylax,funktio,epsilon);
%sekantti.m
%ratkaisee yhtälön sekanttimenetelmällä, kun juuri on alkuarvojen välissä.

alaarvo=feval(funktio,alax);
ylaarvo=feval(funktio,ylax);

if (alaarvo*ylaarvo<0)
   while (abs(ylax-alax)>epsilon)
      uusix=ylax-ylaarvo*(ylax-alax)/(ylaarvo-alaarvo);
      uusiy=feval(funktio,uusix);
      if (uusiy==0)
         alax=uusix;
         ylax=uusix;
      end
      if (uusiy*alaarvo<0)
         ylax=uusix;
         ylaarvo=uusiy;
      elseif (uusiy*ylaarvo<0)
      alax=uusix;
      alaarvo=uusiy;
      end
   end
   tulos=uusix;
end
```

Liite A7 Rungen-Kuttan menetelmä y''=y ratkaisuun

Tuloksissa kullekin x ratkaistu sekä y että y', kun y'(0) = 1 ja y(0) = 1 ja kysytty suure on y(1) = e.

Kertoimien k laskentaa helpottamaan lasketaan funktioiden derivaattojen arvot aputuloksina.

Alkuarvot ja askel otsikkoa seuraavalla rivillä.

Runge-Kutta 2d		y''=y						
x 0			y 0		y' 1		askel 0.1	
x	**y ja y'**	*k1*		*k2*		*k3*	k4	
0	**1**	*1*	1.05	*1.05*	1.053	*1.0525*	1.10525	*1.10525*
0	**1**	*1*	1.05	*1.05*	1.053	*1.0525*	1.10525	*1.10525*
0.1	**1.1051708**	*1.10517*	1.16043	*1.16043*	1.163	*1.163192*	1.22149	*1.22149*
0.1	**1.1051708**	*1.10517*	1.16043	*1.16043*	1.163	*1.163192*	1.22149	*1.22149*
0.2	**1.2214026**	*1.2214*	1.28247	*1.28247*	1.286	*1.285526*	1.34996	*1.349955*
0.2	**1.2214026**	*1.2214*	1.28247	*1.28247*	1.286	*1.285526*	1.34996	*1.349955*
0.3	**1.3498585**	*1.34986*	1.41735	*1.41735*	1.421	*1.420726*	1.49193	*1.491931*
0.3	**1.3498585**	*1.34986*	1.41735	*1.41735*	1.421	*1.420726*	1.49193	*1.491931*
0.4	**1.4918242**	*1.49182*	1.56642	*1.56642*	1.57	*1.570145*	1.64884	*1.648839*
0.4	**1.4918242**	*1.49182*	1.56642	*1.56642*	1.57	*1.570145*	1.64884	*1.648839*
0.5	**1.6487206**	*1.64872*	1.73116	*1.73116*	1.735	*1.735278*	1.82225	*1.822248*
0.5	**1.6487206**	*1.64872*	1.73116	*1.73116*	1.735	*1.735278*	1.82225	*1.822248*
0.6	**1.822118**	*1.82212*	1.91322	*1.91322*	1.918	*1.917779*	2.0139	*2.013896*
0.6	**1.822118**	*1.82212*	1.91322	*1.91322*	1.918	*1.917779*	2.0139	*2.013896*
0.7	**2.0137516**	*2.01375*	2.11444	*2.11444*	2.119	*2.119474*	2.2257	*2.225699*
0.7	**2.0137516**	*2.01375*	2.11444	*2.11444*	2.119	*2.119474*	2.2257	*2.225699*
0.8	**2.2255396**	*2.22554*	2.33682	*2.33682*	2.342	*2.34238*	2.45978	*2.459778*
0.8	**2.2255396**	*2.22554*	2.33682	*2.33682*	2.342	*2.34238*	2.45978	*2.459778*
0.9	**2.4596014**	*2.4596*	2.58258	*2.58258*	2.589	*2.58873*	2.71847	*2.718474*
0.9	**2.4596014**	*2.4596*	2.58258	*2.58258*	2.589	*2.58873*	2.71847	*2.718474*
1	**2.7182797**	*2.71828*	2.85419	*2.85419*	2.861	*2.860989*	3.00438	*3.004379*
1	**2.7182797**	*2.71828*	2.85419	*2.85419*	2.861	*2.860989*	3.00438	*3.004379*

Liite A8. Differentiaaliyhtälön y''=f(y) ratkaisun perusaskel

```
function vast=askelyppony(yfunk,yalku,yloppu,ypnolla);

% askelyppony
% ratkaisee yhtälön y"= a(y) yhden integrointiaskeleen
% tarvitsee lähtöarvona:
% funktion a(x), ja y:n alkuarvon ynolla
% ja y' alkuarvon ypnolla

% integroi funktiot staattisella gaussin integrointikaavalla.
% käytetään nopeuttamaan integrointia.
% kaavaa sisältää sijoituksen ja kerroin vektorin ar


% tulos yploppu.

m=(yalku+yloppu)/2;
h=(yloppu-yalku)/2;

xvekt=[-0.932469514203152,-0.661209386466265,-
0.238619186083197,0.238619186083197,0.661209386466265,0.932469514203152];
Avekt=[0.17132449237917,0.36076157304814,0.46791393457269,0.4679139345726
9,0.36076157304814,0.17132449237917];

xvalues=h.*xvekt + m;
%sisäintegraalin laskeminen
merkki=1;
if (ypnolla<0)
   merkki=-1;
end % if ypnolla<0

vanha=0;
ala=yalku;
```

```
for i=1:6
   vanha=vanha+gaussint(yfunk,ala,xvalues(i));
   ala=xvalues(i);
   yp(i)=vanha;

   isoint(i)= merkki/sqrt(ypnolla*ypnolla+2*yp(i));
end

isointvast=(Avekt*isoint')*h;
vast=isointvast;
```

Liite A9 Yhtälön y''=f(y')g(y) ratkaisun perusaskel

```
function vast=askeltoikertsep(afunk,bfunk,ynolla,yloppu,ypnolla);

% askeltoikertsep
% ratkaisee yhtälön y"= a(y') b(y) yhden askeleen
% tarvitsee lähtöarvona:
% funktion a(x),funktion b(x), y-arvojen askelpituuden
% ja y' alkuarvon y nolla

% tuloksena y-arvoa vastaava y' ja t
% integroi funktiot staattisella gaussin integrointikaavalla.
% käytetään nopeuttamaan integrointia.
% kaavaa sisältää sijoituksen ja kerroin vektorin arvot.

yaskel=yloppu;

% Valitaan pisteet y joille lasketaan vastaava y'


xvekt=[-0.932469514203152,-0.661209386466265,-
0.238619186083197,0.238619186083197,0.661209386466265,0.932469514203152];
Avekt=[0.17132449237917,0.36076157304814,0.46791393457269,0.4679139345726
9,0.36076157304814,0.17132449237917];

m=(ynolla+yaskel)/2;
h=(yaskel-ynolla)/2;
xvalues=h.*xvekt + m;

%funktion G(Y)

vanha=0;
ala=ynolla;
ala2=ypnolla;
for i=1:6
   IY(i)=gaussint(bfunk,ala,xvalues(i));
   raja=ala2+0.25*IY(i);
   yp(i)=lappismeth(IY(i),ala2,raja,@apuri);
   ala=xvalues(i);
   ala2=yp(i);
end
xvalues(7)=yaskel;
IY(7)=gaussint(bfunk,ala,xvalues(7));
```

```
yp(7)=lappismeth(IY(7),ala2,raja,@apuri);


for i=1:6
   gxvalues(i) = 1/yp(i);
end
aika=(Avekt*gxvalues')*h;

vast=[aika,yp(7)];
```

Liite A10 Yhtälön y''=f(y')g(y) tulosvertailuun askeleittain laskeva Matlab tiedosto

```
%kierros.m

alkuarvoy=1;
alkuarvoyp=1;
askeltulos=0;
for i=1:20
   q(i)=1+i*0.5;
   tulos=askeltoikertsep(@afunk,@bfunk,1,q(i),1);
   tulost(i)=tulos(1);
   virhe(i)=tan(tulos(1)/2+pi/4)-q(i);
   askeltulosuusi=askeltoikertsep(@afunk,@bfunk,alkuarvoy,q(i),alkuarvoyp);
   askeltulos(1)=askeltulos(1)+askeltulosuusi(1)
   alkuarvoy=q(i);
   alkuarvoyp=askeltulosuusi(2);
   askeltulost(i)=askeltulos(1);
   askeltulosvirhe(i)=tan(askeltulos(1)/2+pi/4)-q(i);
end
```

Viitteet

[1] Lappi, Esa. A Simple Numeric Method for Solving Separable Differential Equations. in Nordic Matlab Conference Proceedings. S.268-272. Kööpenhamina 2003.

[2] Hemmo, Katariina, Lappi, Esa & Lundahl, Reijo. Pyramidi, Numeeriset menetelmät. Kustannusosakeyhtiö Tammi. Helsinki 2002.

[3] Lappi, Esa & Aunola, Matias. Fast algoritm for Numerical Integration of Separable Differential Equations. Eccomas 2004 Proceedins. Jyväskylä 2004.

[4] Lappi, Esa & Lappi, Riku. Lukion Laskinoppi. Oy Perkko. Helsinki. 2003.

[5] Kreyzig, Erwin. Advanced Engineering Mathematics. 8th Edition. John Wiley & sons, Inc. Singapore 1999.

[6] Kress, Rainer. Numerical analysis. Spinger-Verlag New York 1998

[7] Bartsch, Hans-Jochen. Taschenbuch Matematischer Formeln. Fachbuchferlag Leipzig, 1999.

Liite 2

Lainaus Oy Perkon vuonna 2003 julkaisemasta
Lappi, Esa & Lappi, Riku. Lukion laskinoppi- kirjasta; alkuperäiset sivut: 33-37,50-60 ja 64-67.

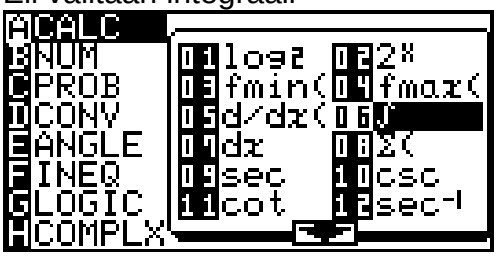
Numeerinen integrointi

Laskin laskee myös suoraan määrätyn integraalin numeerisen arvon. Sitä voi käyttää mm. analyyttisten integrointien tarkastuksiin ja lukuisiin käytännön sovelluksiin. Usein käytännön sovelluksissa esiintyvät funktiot eivät helposti integroidu ns. suljetussa muodossa, jolloin likiarvo on ainut ratkaisu. Numeerista integraalia voi käyttää myös algoritmien sisällä, jolloin laskin voidaan ohjelmoida ratkaisemaan mm. separoituvia differentiaaliyhtälöitä.

Näppäily:

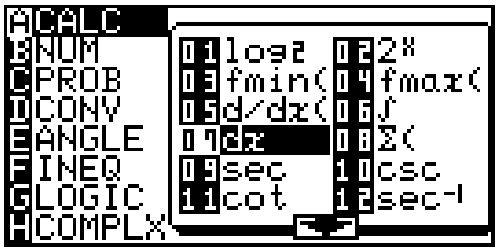
Math calc ∫ enter alaraja **nuoli** yläraja **nuoli** 1/x **math** calc dx **enter** **enter**

Eli valitaan integraali



Syötetään rajat ja integroitava funktio

Lisätään loppuun dx



Ja ensimmäinen **ENTER** viimeistelee integraalin ja toinen **ENTER** laskee sen.

Huomaa käskysulut eli integrointi päättyy dx-merkkiin.

Numeerinen integraali lasketaan adaptiivisella Gaussin integrointiin perustuvalla menetelmällä, ja sen tarkkuus on yleensä melko hyvä. Seuraavaksi käydään läpi joitain virhelähteitä.

Kommentti numeerisista virheistä

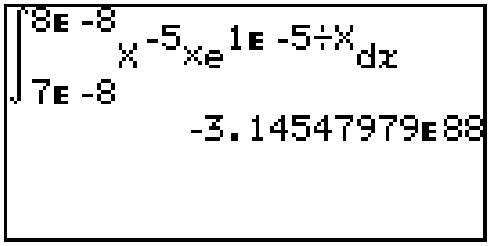
Integrointimenetelmä ja laskimen muut rajoitukset voivat tuottaa ongelmia, jos integrointiväli on hyvin lyhyt tai hyvin pitkä. Sopivalla sijoituksella voidaan ongelma kiertää. Tarkastellaan esimerkkiä, jossa integrointiväli on hyvin lyhyt (suuruusluokkaa miljoonasosa tai alle), ja funktion arvot suuria ja funktion derivaattojen arvot erittäin suuria. Tällöin laskimen toimintaan sisältyy numeerisen virheen mahdollisuus, jossa ns. ylivuodon sijaan laskin antaa tulokseksi halutun luvun vastaluvun. Tämä voidaan välttää sijoituksella z=kx, jossa k on sopiva suuri luku, esim. miljoona. Tällöin funktio muuttuu siten, että

x:n paikalle kirjataan z/k, dx:n arvo muuttuu dz/k:ksi ja integroinnin rajat alaraja a muuttuu arvoksi ka ja yläraja arvoksi kb.

Numeerisen integroinnin virheet tulevat useimmiten esille kolmessa tilanteessa: 1) funktion arvot vaihtelevat rajusti, jolloin yksittäisten pisteiden osalta numeeriseen menetelmään valittavat arvot eivät kuvaa koko funktiota. 2) Integroimisväli on hyvin lyhyt, ja integraalin arvot ovat lukujen kuvaustavan rajalla. 3) Integroimisväli on pitkä, jolloin joko integroimisaika venyy tai integraali laskettaessa numeerinen esitystapa saa arvoksi pelkkiä nollia pitkältä matkalta. Tarkastellaan tarkemmin probleemia 2 ja 3. Tilanteen 1 osalta ei oikein vaihtoehtoja ole, koska riittävän ilkeän funktion numeerinen integrointi voi johtaa laskimen harhaan. Onneksi tällaisia ei useinkaan esiinny käytännön tehtävien osalta.

Lyhyen integroimisvälin probleemat

Tutkitaan integraalia:



Tulos näytöllä on yllättävästi negatiivinen, vaikka funktio on aina f>0.

Integroitavan funktion arvot ovat koko integroimisvälillä positiiviset ja yläraja suurempi kuin alaraja. Siten määrätty integraali on varmasti suurempaa kuin nolla eikä negatiivista! Yllättävästi saatu väärä tulos on oikean tuloksen vastaluku.

Laskin laski tuloksen väärin, koska integraalin integroimisväli on erittäin lyhyt, ja funktiot arvot välillä erittäin suuria. Tilanteessa ollaan ns. ylivuodon rajalla, ja integrointimenetelmä ja sen ohjelmointi ajautuvat harhaan ja juuri ennen ylivuodon virheilmoitusta laskimen lukumerkintä johtaa oikean tuloksen vastalukuun..

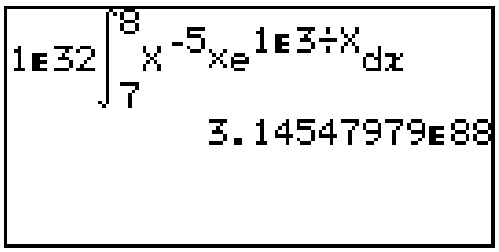
Ongelman poistaminen:

Tekemällä sijoitus $z = 10^8 x$ = 1E8 x (eli $x = 10^{-8} z$ =1E-8 z) saadaan integroimisrajoiksi alaraja $a_z = 7$ ja yläraja $b_z = 8$, $dx = 10^{-8} dz$ = 1E-8 dz sekä $x^{-5} = (1E\text{-}8\ x)^{-5} = 1E40\ z^{-5}$.

Siten uusi integraali on:

$$\int_{7}^{8} 1E40 \cdot z^{-5} e^{1E-5/(1E-8 \cdot z)} \cdot 1E-8 dz = 1E32 \cdot \int_{7}^{8} z^{-5} e^{1E3/z} dz$$

Ja se näppäillään laskimeen käyttäen muuttuja X, jolloin saadaan:

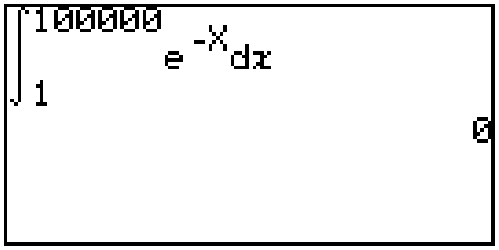


Tulos on edellisen vastaluku ja likimain oikein. Integroimisväli ei enää ole liian lyhyt, ja funktion arvot tutkittavalla välillä pysyvät kohtuullisina. Sijoitusmenetelmän opiskelu helpottaa siten integraalien numeerista käsittelyä!

Pitkän integroimisvälin probleemat

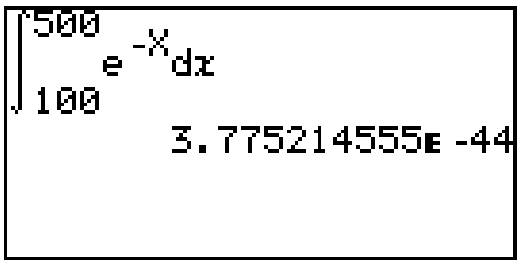
Jos integroimisväli on hyvin pitkä (joko suoraan tai kun halutaan laskea integraalin raja-arvo, kun ylä- tai alaraja lähestyy ääretöntä), voi myös esiintyä toisenlaista numeerista virhettä tai laskin voi jopa kaatua. Tällöin usein puree integroimisvälin jakaminen viipaleisiin, joiden sisällä tehdään tarvittaessa yllä oleva sijoitus niin päin, että integroimisväli lyhenee.

Esimerkki:



Laskin antaa tulokseksi pyöreän nollan, mikä on ilmeisen väärin. Pitkä integroimisväli johti siihen, että kaikki numeeriseen integrointiin tulleet funktion arvot ovat niin lähellä nollaa, että laskin harhautuu.

Oikeatulos saadaan, kun integraali jaetaan osiin:



Nyt käytetään hyväksi muita matematiikan tietoja integraalin arvon määritykseen. Ensinnäkin integroitava funktio e^{-x} on vähenevä. Jo edellisen integroitavan "viipaleen" ylärajalla sen arvo $e^{-500} = 7{,}1245764 * 10^{-218}$, mikä tarkoittaa laskimen tarkkuudella jo nollaa. Koska funktio on laskimelle nolla, kun x>500, laskin antaa integroimisvälillä 500:sta 100000:een integraaliksikin arvon nolla. Tarkastellaan, onko tämä virhe merkittävä kokonaisuuden kannalta. Yläarvo integraalille väliltä 500:sta 100000 on enintään integroitavan funktion suurin arvo kerrottuna integroimisvälin pituudella.

Tästä saadaan puuttuvan integraalin viimeisen välin karkeaksi ylälikiarvoksi $100\,000 * 7{,}12 * 10^{-218} = 7{,}12 * 10^{-213} < 10^{-212}$. Siten integraalin arvo väliltä yhdestä sataan on likimain oikea arvo integraalille koko integroimisväliltä, koska integraalin arvo väliltä sadasta viiteen sataan on suuruusluokkaa 10^{-44}, ja vaikuttaa vasta 44. desimaaliin, ja loppuintegraali vasta pitkälti yli 200. desimaaliin.

Siis vastauksen likiarvo on 0,367879441.

Tämäntapainen menettely on erittäin näppärää, kun pitää tarkastaa analyyttinen lukiotehtäviä, joissa integroidaan kohti ääretöntä.

Syventävä kurssi ”Numeeriset menetelmät”

Liukuluvut (teoriaa)

Laskimen laskennan selvittämiseksi käydään lyhyesti läpi liukulukujen ominaisuudet. Tietyissä tilanteissa laskimen lukuesitys ja sen rajoitukset ovat merkittävässä roolissa.

Laskin laskee

1) *kaksi- eli binäärijärjestelmän luvuilla*

2) *luvut on esitetty liukulukumuodossa, eli luku sisältää etumerkkitiedot (1 bitti), n-bittisen mantissan (=n merkitsevää numeroa) ja m-bittisen eksponentin (vastauksen suuruusluokka).*

Laskimen laskentatarkkuus on 46 bittiä eli noin 12 merkitsevää numeroa kymmenjärjestelmässä. Näytöllä näkyy 10 desimaalia, eli pienet pyöristysvirheet eivät vielä näy ruudulla.

Jotta liukulukujen laskentasäännöt kävisivät selviksi, tarkastellaan ensin liukulukumuotoisten lukujen ominaisuuksia käyttäen perinteistä kymmenjärjestelmää, jossa mantissassa on 3 desimaalia ja eksponentissa 2 desimaalia.

*Luku 5346 on lopusta katkaisevassa liukulukuesityksessä 5,34*10^(+03) merkitään usein 5,34E+03 ja pyöristävässä 5,35*10^(+03) eli toisin merkittynä 5,35E+03. Luvusta laskimen muistissa olisivat +534 ja +03.*

*Suurin luku, joka voidaan esittää esimerkkiliukuluvuilla on 9,99*10^(+99) ja itseisarvoltaan pienin ohjelmoinnista riippuen joko 1,00*10^(-99) tai 0,01*10^(-99). Jos luvut ylittävät 9,99E+99, laskin kaatuu. Jos luku on pienempi kuin 1E-99, se tulkitaan nollaksi. Nämä samat rajat ovat likimain SharpEL9900 laskimessa lukujen suuruudesta. Eli näennäisesti kaikki normaalit luvut saataisiin tällaisella kuvitellulla laskimella lasketuksi.*

Merkittävämpää onkin merkitsevien lukujen määrä eli mantissan pituus. Tämän havainnollistamiseksi käsitellään esimerkkinä yhteenlasku liukuluvuilla, joissa on 3 numeroa mantissassa. Yleensä yhteenlaskussa ongelmia ei juurikaan esiinny, mutta kolmedesimaalisilla luvuilla , joista toinen on yli 1000 kertainen, summa on toinen luvuista: esim. 1,343 + 1521 = 1522,343, mutta laskimen mielestä ensinnäkin viimeiset numerot putoavat pois eli laskin laskee vain kolmella merkitsevällä numerolla, koska mantissassa on vain kolme numeroa. Laskimen mukaan on laskettu luvut 1,52E(+03) + 1,34E(+00) ja tulokseen tulee vain kolme merkitsevää numeroa eli vastaus on 1,52E(+3). Vastaus on yhden laskun jälkeen edelleen kolmella merkitsevällä numerolla oikein, mutta viimeinen desimaali on epävarma. Jos siis laskettaisiin useita yhteenlaskuja sisältävää summaa, esimerkiksi 1521 +1,343

+1,343+……+1,343, tulokseksi saataisiin 1520, koska kunkin laskutoimituksen jälkeen muistissa olisi aina vain 153E+03.

Yhteenlaskun virhe tulee siis laskimen 12 desimaalisessa järjestelmässä merkittäväksi vain, kun lasketaan tuhansien suureiden summia. Laskimen oikealla laskentatarkkuudella virhe näkyy näyttöruudulla viimeisissä numeroissa vain, jos yhteenlaskuja on vähintään kymmeniä. Siten yhteenlasku on yleisesti ottaen näytössä näkyvällä tarkkuudella oikein (virhe viimeisessä desimaalissa, joka ei näy näytöllä).

Esimerkki: ohjelmoidaan laskin laskemaan summaa Sigma $1/x_i$. Summa lakkaa kokonaan kasvamasta, kun summan arvo + uusin termi pyöristyy alkuperäiseen summan arvoon. Tämä tapahtuu, kun suhde:
*(summan arvo) / ($1/x_i$) = (summan arvo) * x_i > 2^46 ≈ 7E13,*
koska iso + pieni pyöristyy liukulukujen vuoksi tämän jälkeen samaksi kuin iso.

Vähennyslasku

Vähennyslaskulla on kaikki yhteenlaskun haitat, kun luvut ovat eri suuruusluokkaa, mutta lisäksi vähennyslasku voi johtaa katastrofaalisesti virheelliseen tulokseen, kun vähennetään kaksi likimain yhtä suurta numeroa:

*Esimerkki: Vähennyslasku 1534,2-1534,1 = 0,1, mutta kuvitellun esimerkkitapauksen laskimen mielestä 1,53*10^(03) – 1,53*10^(03) = 0. Tai esimerkiksi näinkin 1540,1 - 1534,9 = 5,2, mutta kuvitellun laskimen mielestä 1540-1530=10. Tuloksessa ei ole yhtään merkitsevää numeroa, ja virhe on n. 100% eli käytännön tehtävissä yleensä liian suuri.*

Esimerkissä oli 4 merkitsevää numeroa ja laskimessa 12. Tästä huolimatta vähennyslasku voi johtaa jopa laskimen laskentatarkkuudella vääriin tuloksiin, mistä jäljempänä esimerkkinä käsitellään toisen asteen yhtälön ratkaisua.

Kerto- ja jakolasku

*Kertolaskussa merkitsevien numeroiden lukumäärä pysyy samana, mutta viimeisen bitti 2-järjestelmällä laskettaessa voi mennä väärin. Sama pätee jakolaskussa. Esimerkki 1,545*3217 = 4970,265. Laskimen tarkkuudella (1,55*10^00)*3220 = 4991 pyöristyy 4990. Viimeiseen desimaaliin tulee pyöristyksistä johtuvaa virhettä. Koska laskin laskee binääriluvuilla, ns. viimeinen bitti voi olla väärin. Koska laskimen näytöllä näkyy 10 merkitsevää numeroa, ja laskimen muistissa on 12, näytöllä olevat numerot ovat kerto- ja jakolaskussa yleensä oikein.*

Toisen asteen yhtälö ja laskimen laskentatarkkuus

Toisen asteen yhtälön numeerisesti oikea ratkaisukaava ja esimerkki toisen asteen yhtälöstä, jonka ratkaisu edellyttää numeeristen tekijöiden huomioon ottamista.

Yhtälön $ax^2 + bx + c = 0$ "numeerisesti oikea ratkaisukaava" on seuraava:

Ensimmäisen juuren X1 ratkaisu riippuu tekijän b merkistä:

Jos b > 0, eli b on positiivinen, valitaan neliöjuuren eteen -

$$X1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Jos b < 0, valitaan neliöjuuren eteen +

$$X1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Toinen juuri saadaan kummassakin yllä olevassa tapauksessa ensimmäisestä juuresta

*X2 = c / (x1 * a)*

Esimerkki :

$X^2 + 10^{15} x - 10^{15} - 1 = 0$

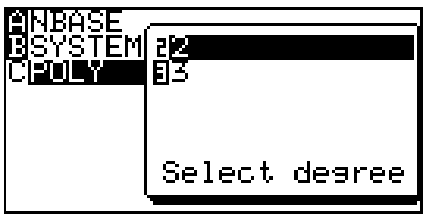
Koska positiivinen eli b>0, ensimmäinen juuri kaavasta:

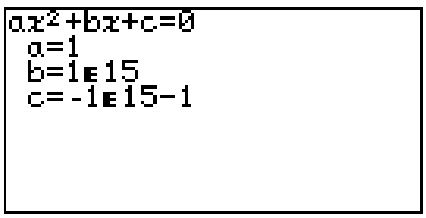
$$x1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{-10^{15} - \sqrt{10^{30} - 4 \cdot 1 \cdot (-10^{15} - 1)}}{2 \cdot 1}$$ *= - 1 000 000 000 000 000 001 (tarkka arvo), likiarvo kymmenellä merkitsevällä numerolla on 1,000 000 000 E15*

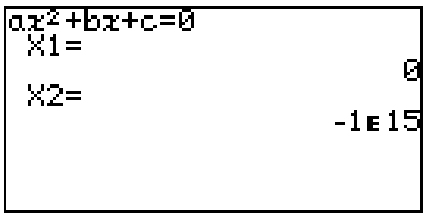
Ja toinen juuri kaavalla:

*x2 = c/(a * x1) = (- 10^{15} –1)/(-1*E-15) = 1*

Valmistoiminto ja perinteinen kaava antavat toisen juuren oikein 12 merkitsevällä numerolla, mutta toinen juuri on väärin. Ks. alla oleva tulos:







Numeerisesti oikeaa kaavaa voi käyttää aina. Mutta perinteisen kaavan ja laskimen valmistoiminnon numeerista hyvyyttä arvioitaessa pitää tarkastella b ja 4ac :n suhdetta. Jos suhde $|b^2 / 4ac|$ on suurempi kuin 10^{12} pääsääntöisesti vain numeerisesti oikea kaava antaa oikean vastauksen (12 merk. nro), ja peruskaava varmasti väärän.

Nyrkkisääntö:

Voit käyttää laskimen valmistoimintoa tai perinteistä ratkaisukaavaa, jos $|b^2 / 4ac| < 10^6$. Tulos on silloin oikein vähintään 6 merkitsevällä numerolla.

Yleensä $|b^2 / 4ac|$ on riittävän pieni käytännön tapauksissa, mutta esimerkiksi kemian tasapainolaskuissa pitää joskus käyttää numeerisesti oikeaa ratkaisukaavaa.
Esimerkissä lukuarvot olivat: $b = 10^{15}$ eli b^2 on 10^{30} ja $4ac = 4*1*(-10^{15} - 1) = -4*10^{15}$.
Suhde on $2,5 * 10^{14} > 10^{12}$. Laskimen laskentatarkkuus ei riitä.

Yhtälöjen ratkaisu

Johdanto

Laskimella voidaan ratkaista numeerisesti yhtälöitä joko 1) kuvanpiirtotoiminnon avulla 2) valmistoiminnolla "SOLVER" ja 3) itse kirjoitetuilla numeerisilla kaavoilla.

Riippumatta menetelmästä vastausten tarkkuus ja lukumäärä pitää osoittaa tilanteen edellyttämällä tarkkuudella. Hyvä tapa tutkia vastausten lukumäärää on esimerkiksi muuttaa yhtälö funktion nollakohtien määritysprobleemaksi ja tutkia funktion kulkukaaviota. Funktiolla voi myös perustella vastausten tarkkuuden: (jos funktio on jatkuva, voidaan käyttää ns. Bolzanon lausetta jonka mukaan jatkuva funktio, jonka arvo pisteessä a negatiivinen ja pisteessä b positiivinen saa arvon nolla ainakin yhdessä pisteessä välillä]a,b[.

Esimerkki: Jos jatkuvalla funktiolla $f(4,5) < 0$ ja $f(5,5)>0$, välillä $4,5<x<5,5$ on ratkaisu. Koska kaikki luvut ko. väliltä pyöristyvät luvuksi 5, ratkaisu on varmasti ykkösten tarkkuudella 5.

*Halutaan osoittaa että jatkuvan funktion y=f(x) nollakohtaa etsittäessä ratkaisu on x=5,4 yhden desimaalin tarkkudella: Ratkaistaan funktion arvot kohdissa x=5,45 ja x=5,35 eli lasketaan f(5,45) ja f(5,35). Jos ne ovat erimerkkiset, eli f(5,45)*f(5,35)<0, välissä on ratkaisu. Tulomuotoinen ratkaisun tarkkuuden osoittamismenetelmä on helppo ohjelmoida, jos haluaa itse kirjoittaa numeerisen laskukaavan yhtälölle.*

Kuvaajan piirtotoiminnon avulla tapahtuva ratkaiseminen

Yhtälön juuren ratkaisu kuvanpiirron avulla perustuu perättäisiin operaatioihin, joissa 1) piirretään haluttu kuvaaja, 2) trace toiminnon ja nuolinäppäiden avulla haetaan likimääräinen vastaus 3) zoom in toiminnolla parannetaan takkuutta ja palataan kohtaan 2 kunnes vastaus on saatu sopivalla tarkkuudella.

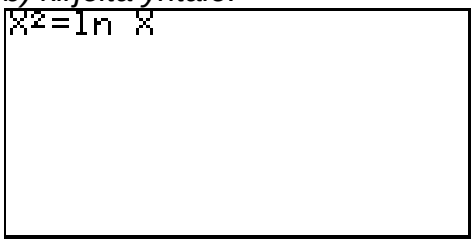
Tämän jälkeen tuloksen tarkkuus voidaan osoittaa ylläkuvatulla mekanismilla.

Solver-toiminnon käyttäminen

Laskimessa on yhtälön ratkaisua varten valmistoiminto eli ”Solver”. Siihen voi suoraan syöttää ratkaistavan ongelman, jota laskin yrittää ratkaista ensin yksinkertaisesti, sitten Newtonin menetelmällä (ks. seuraava kohta) ja lopulta se opastaa graafiseen ratkaisuun. Solverin käyttö tapahtuu seuraavasti:

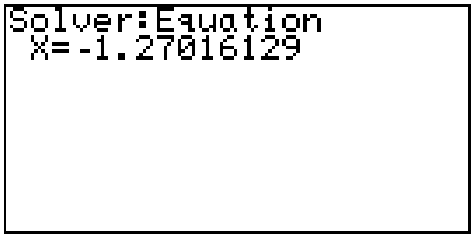
a) aktivoi solveri painamalla ***2ndf*** ***SOLVER***

b) kirjoita yhtälö:

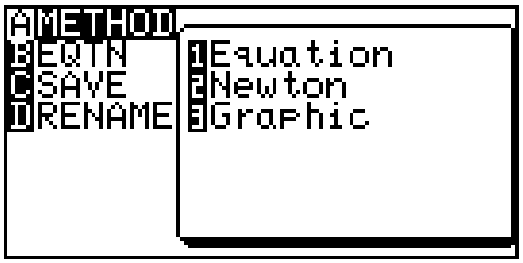


Paina ***ENTER***

ja saat tuloksen



c) Jos haluat muuttaa käytettyä menetelmää, paina uudelleen ***2ndf*** ***SOLVER***



SOLVER- toimintoa koskevat muut ohjeet ovat käyttöoppaan sivuilla 194-201.

Jos käytät solveria, tutki ensin funktion kulkua muilla menetelmillä, että löydät varmasti kaikki juuret, etkä putoa mahdolliseen numeeriseen ansaan. Toiminnon helppous voi houkutella aivottomaan käyttöön, varsinkin fysiikan tehtävissä. Valitettavasti Solverilla tempaistu ratkaisu ei kelpaa ylioppilas- eikä vissiin muissakaan kokeissa, mutta sillä on kiva tarkistaa analyyttisiä ratkaisuja.

Tunnetuimmat yhtälön numeerisen ratkaisun menetelmät itse näppäiltyinä

1) *Kiintopisteiterointi $x_{n+1} = g(x_n)$*
   a. *Kirjoita yhtälö muotoon x = g(x)*
   b. *Tutki, onko |g'(x)| < 1. Jos jollain välillä a<x<b on juuri ja tällä välillä |g'(x)|<1, kiintopisteiterointi suppenee kohti juurta. Jos g(x) ei toimi, koska g'(x)>1, g:n käänteisfunktio toimii. Käytännössä suppeneminen on mukavan nopeaa, jos |g'(x)| < 0,8.*
   c. *Valitse alkuarvo x_0 tutkittavalta alueelta eli a<x_0<b. Syotä luku laskimeen ja paina enter.*
   d. *Kirjoita kaava g(x) laskimeen niin, että joka paikassa x:n paikalla on "ans" eli edellisen laskun tulos.*
   e. *paina **ENTER**iä, kunnes laskimen arvo on vakiintunut*

Esimerkki: x – 5 = ln x;

Ratkaisu: Yhtälö on määritelty, kun x>0. Etsitään positiivisia juuria kiintopisteiteraatiolla:

Ratkaistaan yhtälö muotoon x = lnx + 5 = g(x).

Iteraation suppenemistarkastelu: g'(x) = 1/x eli iteraatio suppenee, kun 1/x < 1 eli kun x>1. (määrittelyjoukko x>0)

Valitaan alkuarvoksi esim. x=100.

Näppäily:

*100 **enter**.*

Kirjoita ratkaisukaava:

ln(ans) + 5

ja paina

***enter**.*

Toista enterin painamista kunnes iteraatio on supennut

Nyt saatiin yhtälön juuri alueelta x>1.

Mahdollinen juuri alueelta 0<x<1 saadaan käyttämällä g:n käänteisfunktiota, eli ratkaisemalla yhtälö muotoon x = e^(x-5) = g_2(x).

Tällöin näppäily menee:

0 **ENTER** **2ndF** **e^x** *(* **2ndF** **ANS** *– 5)* **ENTER**

Näytöllä näkyy tulos 0.006737946.

Näppäily jatkuu:

ENTER	*0.0067835*
ENTER	*0.006783809*
ENTER	*0.006783811*
ENTER	*0.006783811,*

valmista tuli, koska muita juuria ei voi olla kiintopisteiteraation ominaisuuksien takia.

2) Newtonin iterointi

Newtonin iterointi on menetelmä funktion nollakohdan ratkaisemiseksi: millä x funktio f(x) = 0.

Se on laskimen solverissa valmistoimintona, mutta kaavaa voidaan käyttää myös käsin. Tällöin voidaan myös käyttää hitaammin suppenevaa, mutta juuren varmemmin löytävää vaimennuttua Newtonin iterointia.

Newtonin iterointi perustuu funktion derivaattaan, jonka avulla alkuarvausta parannetaan.

Derivaatta on erotusosamäärän raja-arvo, ja likimäärin pätee:

$f'(x0) \approx [f(x) - f(x0)] / [x - x0]$.

Nyt x0 on tunnettu alkuarvaus, jonka perusteella halutaan se x, jolla f(x) = 0. Kun ratkaistaan yllä olevasta f'(x0):n kaavasta x, saadaan:

$x = x0 - f(x0) / f'(x0)$,

joka iterointikaavana kirjoitetaan muotoon

$x_{i+1} = x_i - f(x_i) / f'(x_i)$

Tämä näppäillään laskimeen joko käyttäen ans – toimintoa tai sijoittamalla alkuarvo muuttujaan X käyttämällä **STO** *- näppäintä ja kirjoittamalla kaava käyttäen X:ää.*

Vaihtoehto 1:

alkuarvaus ***ENTER***

iterointikaava, jossa x:n paikalla ans

painetaan ***ENTER****iä, kunnes vastaus on löytynyt.*

Jos ei löydy, vaihdetaan alkuarvausta ja lopulta vaihdetaan numeerista ratkaisumenetelmää.

Vaihtoehto 2:
alkuarvaus ***STO*** *X*

iterointikaava x:n avulla kirjoitettuna ***STO*** *X*
painetaan ***ENTER****iä, kunnes vastaus on löytynyt.*

Jos ei löydy, vaihdetaan alkuarvausta ja lopulta vaihdetaan numeerista ratkaisumenetelmää.

Esimerkki:
Ratkaise numeerisesti jokin yhtälön ln(x) +5 = x juurista.

Ratkaisu:
Tutkitaan funktiota f(x) = ln(x) +5 – x , jonka nollakohdat ovat tehtävän ratkaisuja.
Funktion derivaatta on f'(x) = 1/x –1. Siis Newtonin iterointikaava on

$$x_{i+1} = x_i - [\ln(x_i) + 5 - x_i] / [1 / x_i - 1]$$

Tapa 1: Näppäillään laskimeen alkuarvaus 5 ja ***ENTER****. Kirjoitetaan iterointikaava:*

ans – (ln (ans) + 5 – ans) / (1 / ans – 1) ***ENTER*** *,* ***ENTER*** *,* ***ENTER***
….

Ja tuloksena lukujono:

```
Ans)+5-Ans)÷(1÷Ans-1)
             7.011797391
             6.936914517
             6.936847407
             6.936847407
             6.936847407
```

Tapa 2: Näppäillään 5 ***STO*** *X*

X – (ln (X) + 5 – X) / (1 / X – 1) ***STO*** *X* ***ENTER*** *,* ***ENTER*** *,* ***ENTER***
…

Ja tuloksena sama lukujono kuin äsken.

```
5⇒X
                          5
ln (X)+5-X)÷(1÷X-1)⇒X
            7.011797391
            6.936914517
            6.936847407
```

Huom! yhtälöllä on toinen juuri lähellä nollaa, mutta sen saaminen esim. alkuarvauksella 1 ei onnistu, koska luku 1 on derivaatan nollakohta. Siten Newtonin iteroinnin alkuarvaus on tärkeä tarkastaa etukäteen jotakuinkin kelvolliseksi. Alkuarvaus 0,5 on sinänsä kelvollinen, mutta sekin johtaa sekin ongelmiin, koska seuraava luku menee reilusti pakkasen puolelle, missä reaalinen logaritmi ei ole määritelty. Jos laskin on kompleksiluvuilla, Newton voi lopulta toimia koukattuaan kompleksialueen kautta, mutta se on epävarmaa ja vastauksen tulkinta voi edellyttää kompleksilukujen osaamista.

Vaimennettu Newton parantaa jonkin verran juureen suppenevan alkuarvauksen valintaprobleemaa, mutta sen hintana on hitaampi suppeneminen kohti juurta.

Vaimennetun Newtonin kaava on:

$$x_{i+1} = x_i - \lambda\, f(x_i) / f'(x_i)$$

Tässä λ on vaimennuskerroin, joka saa arvoja nollan ja yhden väliltä. Aluksi sen arvo voi olla esim. 0,1, mutta kun ollaan lähellä ratkaisua, arvon voi muuttaa pikkuhiljaa esim. luvun 0,5 kautta tasan ykköseksi. Muuten iterointi kestää noin 10-kertaisen ajan.

Lasketaan edellinen esimerkki alkuarvosta 0,5 lähtien käyttäen vaimennettua Newtonin iterointia:
Näppäillään laskimeen alkuarvaus 0,5 ja enter. Kirjoitetaan iterointikaana:

ans – 0,1(ln (ans) + 5 – ans) / (1 / ans – 1)* ***ENTER*** ,
ENTER , ***ENTER*** ….

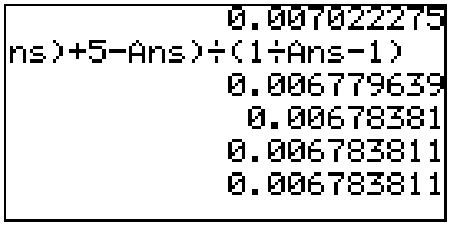
Ja tuloksena lukujono:

alkutila ja reippaan näppäilyn jälkeen ollaan vielä kesken

```
0.5
                        0.5
s)+5-Ans)÷(1÷Ans-1)
            0.119314718
            0.081994278
            0.060407052
```

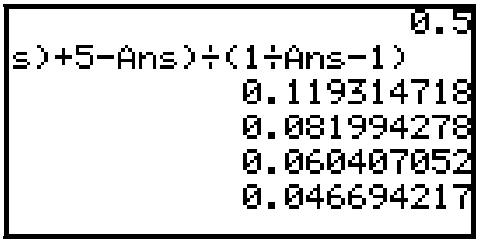
```
            0.00719316
           0.007151006
           0.007113303
            0.00707956
           0.007049345
           0.007022275
```

Muutetaan vaimennuskerroin ykköseksi:
2ndF ***ENTRY*** *ja muutetaan nuolinäppäimellä vaimennus ykköseksi.*



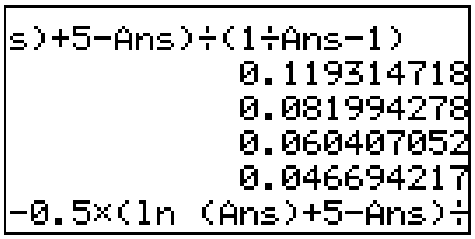
Ja iterointi suppeni arvoon x=0,006783811 parilla uudella askeleella.

Jos neljän iterointiaskeleen jälkeen vaimennus olisi muutettu arvoon 0,5 ja taas kolmen askeleen jälkeen ykköseksi olisi iterointi kulkenut näin:
Näppäilyt
0,5 ***ENTER***
ans – ***0,1 ****** *(ln (ans) + 5 – ans) / (1 / ans – 1)* ***ENTER***
ENTER
ENTER



2ndF ***ENTRY***

Käydään muuttamassa 0,1 arvoon 0,5 nuolinäppäimien avulla:



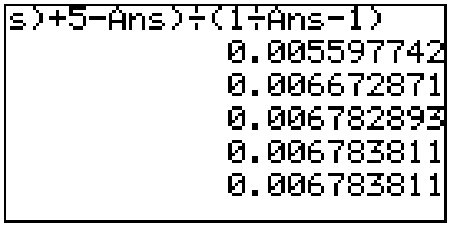
ans – ***0,5 ****** *(ln (ans) + 5 – ans) / (1 / ans – 1)* ***ENTER***

ENTER
ENTER
ENTER
ENTER

2ndF ***ENTRY*** *ja käyty muuttamassa 0,5 luvuksi 1:*

ENTER
ENTER
ENTER

ENTER

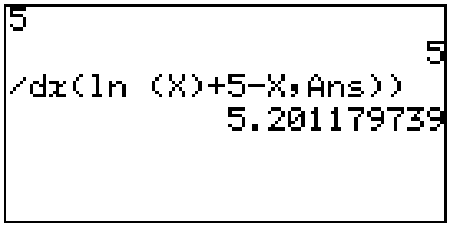


Newtonin iteroinnissa voi käyttää myös hyväksi laskimen valmistoimintoa derivoinnille, jolloin kaava jakoviivan alla oleva derivaatta lasketaan numeerisesti.

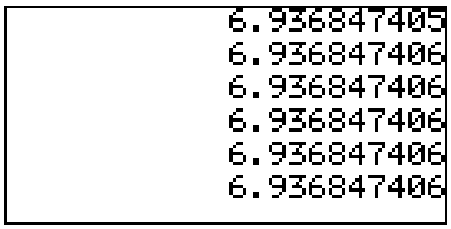
Tällöin näppäily olisi:

5 ***ENTER***

*ans – **0,1 *** *(ln (ans) + 5 – ans) / (* ***MATH*** *calc d/dx* ***ENTER*** *ln (X) + 5 – X, ans))*
ENTER



ENTER
ENTER
Jne..



Sormia rasittaneen hakkaamisen jälkeen vaimennettu Newton päätyi maaliinsa.

Separoituvan differentiaaliyhtälön numeerinen ratkaiseminen *(syventävä kurssi ”analyysi”)*

Tässä tarkastellaan differentiaaliyhtälön numeerista ratkaisua. Valmistoimintoa siihen ei ole, mutta Sharp EL-9000 sarjan laskimissa on integraalin numeerisen ratkaisun toiminto. Sen avulla voidaan tarkastaa differentiaaliyhtälöihin liittyviä tuloksia ja ratkaista separoituvan differentiaaliyhtälön y’ = f(t) * g(y) alkuarvoprobleema eli funktion y(t) arvo tietyssä kohdassa t, kun alkuarvo t_0 and $y(t_0)$ on annettu.

Menetelmä yhdistää Newtonin menetelmän yhtälön ratkaisemiseksi ja integraalitoiminnon, jolloin saadaan halutulle y = y(t) iteraatiokaava:

$$y_i = y_{i-1} - \left(\int_{y(t_0)}^{y_i} \frac{dy}{g(y)} - \int_{t_0}^{t} h(t)dt \right) \cdot g(y_i)$$

Menetelmä käyttää laskimen valmistoimintoa ja on siten kätevämpi kuin ohjelmointia vaativat perinteiset differentiaaliyhtälön numeeriset ratkaisumenetelmät. Kuitenkin se vie käytännössä enemmän liukulukuoperaatioita kuin esimerkiksi hyvin ohjelmoitu Rungen-Kuttan menetelmä. Tämä opas ei käsittele laskimen ohjelmointitoimintoa, joten ohjelmointia vaativia menetelmiäkään differentiaaliyhtälön ratkaisemiseksi ei käsitellä, mutta ratkaisukaavat löytyvät mm. Tammen Pyramidi –sarjan Numeeristen menetelmien kirjasta.

Iterointikaavan johto:

Lähtötilanne:

Ratkaise tuntematon y(t) annetulla t, kun
a) y’(t) = f(y,t) = g(y) h(t),
b) Alkuarvot $y(t_0)$ ja t_0 tunnetaan

Lisäksi oletetaan yhtälöllä olevan ratkaisu eli

c) Funktiot f(y,t) ja g’(y) ovat jatkuvia

Differentiaaliyhtälön ratlkaisu alkaa separoinnilla kuten analyyttinenkin ratkaisu:

Kaava 1a jaetaan funktiolla $g \neq 0$ ja integroidaan molemmat puolet.

(2) $$\int_{y(t_0)}^{y(t)} \frac{dy}{g(y)} = \int_{t_0}^{t} h(t)dt$$

Jos yhtälön molemmat puolet osataan integroida, tehtävä ratkeaa siitä eteenpäin normaalisti, mutta jos integrointia ei haluta tehdä tai se ei onnistu, käytetään numeerista integrointitoimintoa. Merkitään yhtälön oikean puolen integraalia:

(3) $I_t = \int_{t_0}^{t} h(t)dt$

Tässä tiedetään yleensä sekä alkuarvo t0 että integraalin ylärajana oleva t. Se voidaan siis laskea joko analyyttisesti tai numeerisesti suoraan laskimella.

Yhtälön vasen puoli on

(4) $I(y(t)) = \int_{t_0}^{t} h(t)\,dt$

Merkitään funktion y arvoa kohdassa t pelkällä y:llä eli y(t) = y, jolloin yhtälö (2) voidaan kirjoittaa muotoon:

(5) $I(y) - I_t = 0$

Tämän yhtälön vasen puoli on tuntemattoman y suhteen jatkuva funktio. Merkitään sitä R:llä eli:

(6) $R(y) = I(y) - I_t.$

Funktion R(y) ominaisuudet ja derivaatta R'(y) = 1/g(y) antavat meille ratkaisun tarkkuuden ja ratkaisujen lukumäärän kannalta tärkeää tietoa.

Tehtävänä on ratkaista funktion R(y) nollakohta, ja se voidaan tehdä edellä kuvatulla Newtonin menetelmällä:

(7) $y_{i+1} = y_i - R(y_i) / R'(y_i) = y_i - R(y_i)\, g(y_i) =$

$$y_i - \left(\int_{y(t_0)}^{y_i} \frac{dy}{g(y)} - \int_{t_0}^{t} h(t)dt \right) \cdot g(x_i)$$

Newtonin menetelmässä alkuarvauksella on ratkaiseva merkitys, eli alkuarvaus y_0 pitää välita "sopivasti" tehtävän antamien vihjeiden perusteella. Tällöin voidaan käyttää myös vaimennusta kuten normaalissakin Newtonin menetelmässä.
Jos vaimennuskerrointa merkitään λ :lla, saadaan kaava:

(8) $y_{i+1} = y_i - \lambda\, R(y_i) / R'(y_i); \qquad (0<\lambda<1)$

Esimerkkitehtävä:

y'(t) = e^{y^2} ,
y(0) =1 ,
y(2) =?.

Eksponenttifunktion e^{y^2} integrointi ei onnistu analyyttisesti, mutta numeerisesti saadaan laskimesta $\int_1^x e^{x^2} dx$.

Siten separoituva yhtälö päätyy muotoon:

(9) $$\int_1^y \frac{dy}{e^{y^{-2}}} = \int_0^2 dt$$

Josta saadaan:

(10) $$\int_1^y \frac{dy}{e^{y^{-2}}} - \int_0^2 dt = 0$$

Siten saadaan Newtonin iterointia varten funktio, jossa oikea puoli voidaan ratkaista valmiiksi:

(11) **R(y) =** $$\int_1^x \frac{dx}{e^{x^{-2}}} - \int_0^2 dt$$

Jos halutaan arvioida ratkaisujen lukumäärää, todetaan R(y) aina kasvavaksi jatkuvaksi funktioksi, koska R'(y) = $e^{y^2} > 0$ *kaikilla y. Kasvavalla funktiolla on vain yksi nollakohta eli tehtävällä R(y) = 0 on enintään yksi ratkaisu (tai ei yhtään).*

Newtonin iterointikaava yhtälölle R(y)=0 antaa:

(12) $\mathbf{y_{i+1}} = y_i - \left(\int_{y(t_0)}^{y_i} \frac{dy}{e^{-y^2}} - 2 \right) \cdot e^{-y_i^2}$

Alkuarvoa varten tehdään pieni kokeilu: R(1) = -2 eli negatiivinen. Koska R on kasvava todetaan, että y > 1.

Tehdään alkuarvaus, joka on tällä rajalla eli y(t=2) arvioidaan lähtöarvolla $y_0 = 1.00$.

Laskimeen syötettäessä integroitavan funktion muuttujana pidetään X –kirjainta eli saadaan näppäilyt:

$1.0 \rightarrow X$ *Syötetään muistiin muuttujaan x arvo 1.*

$\left(\int_1^x \frac{dx}{e^{-x^2}} - 2 \right) \cdot e^{-x^2} \rightarrow X$ *Kirjoitetaan iterointikaava käyttäen integraalitoimintoa, muuttujana x, integraalin päättymismerkiksi dx.*

Sama kaava voidaan tehdä ANS –näppäimen avulla:

1.0 ENTER

$$ans - (\int_{1}^{ans} \frac{dx}{e^{-x^2}} - 2) \cdot e^{-ans^2}$$

Enter-näppäimen peräkkäiset painelut antavat kummassakin tapauksessa lukusarjan:

1.0	*alkuarvaus* y_0
1.735758882	y_1
1.54282203	y_2
1.447128316	y_3
1.4303555	y_4
1.429936675	y_5
1.429936424	$y_6 \cong y_7$, *eli iterointi suppene onnellisesti.*

Vastauksen tarkkuuden voi osoittaa myös laskimen integrointitoiminnon avulla: Jos E (>0) on laskimen integroinnin virhe (asetusarvo 1E-5). Kaavana tarkkuuden osoitus tarkoittaa: sitä, että jos $R(y-e_1) < -E$ *ja* $R(y+e_2) > +E$ *positiivisille luvuille* e_1 *ja* e_2, $[y-e_1 , y+e_2]$ *on juuri.*

Siten esimerkkitehtävässä laskemalla lausekkeet saadaan likiarvot:

$$R(1.429935) = \int_{1}^{1.429935} \frac{dx}{e^{x^{-2}}} - 2 = -0.000011003 < -1E\text{-}5$$

$$R(1.429938) = \int_{1}^{1.429938} \frac{dx}{e^{x^{-2}}} - 2 = 0.000012177 > 1E\text{-}5$$

Siten ratkaisu pyöristyy 6 merkitsevällä numerolla arvoon y(2) = 1.42994.

Menetelmää voi hyvin käyttää analyyttisesti ratkaistavien differentiaaliyhtälöiden alkuarvoprobleemien ratkaisuun.

Tällöin voi numeerisen ratkaisun lisäksi tyytyä vain tekemään sijoitukset kaavaan 2, jolloin jos vasemman puolen integraalin arvo on sama kuin oikean puolen integraalin arvo, analyyttinen ratkaisu on todennäköisesti oikein, mutta tämä ei takaa sitä, että ratkaisu olisi ainoa oikea.

Jos sijoitus alkuperäiseen yhtälöön johtaa väärään arvoon, voi tehtävän numeerinen ratkaisu helpottaa analyyttisessä ratkaisussa tehtyjen virheiden löytämistä.

Liite 3

Comap:n järjestämän MCM/ICM -kilpailun tuloksia

Comap ”the Consortium for Mathematics and its applications” on yhdysvaltalainen yleishyödyllinen yhdistys, jonka tavoitteena on edistää kaikenikäisten matematiikan oppimista. Se järjestää vuosittain yliopistoille suunnatun MCM/ICM mallinnuskilpailun, johon myös Päivölästä ja Helsingin matematiikkalukiosta on osallistuttu. Lisätiedot kilpailusta voi lukea yhdistyksen sivulta www.comap.com/udergraduate/contests/mcm.

Tulokset jakautuvat neljään arvolauselmaan. Alla olevassa taulukossa on vuoden 2005 tulokset Päivölän osalta ja kaikille 828 osallistujalle sekä Päivölän kokonaismenestys.

	Kaikki osallistujat	Päivölä 2005		Päivölä 1998-
Outstanding Winner	13 (1,6%)	0		0
Meritorious	111 (13%)	1	(20%)	3 (15%)
Honorable Mention	284 (34%)	1	(20%)	6 (30%)
Successful Participant	420 (51%)	3	(60%)	n.11* (55%)
Yhteensä:	828	5		20

Helsingin matematiikkalukiosta on osallistuttu vuodesta 2003 alkaen. Osa opiskelijoista on opiskellut yliopistoyhteistyön yhteydessä ohjelmointia. Tulokset on esitetty yhteensä ja suluissa lisäksi sellaisten ryhmien lukumäärä, joissa on ollut ohjelmointia opiskelleita.

	2003	2004	2005	yhteensä
Outstanding Winner	0	0	0	0
Meritorious	0	0	0	0
Honorable Mention	1 (1)	1 (1)	0	2 (2)
Successful Participant	1 (1)	3 (1)	2	6 (2)

Tulokset osaltaan vahvistavat käsitystä ohjelmointitaidon tärkeydestä, vaikka otoksen pienuuden vuoksi mitään tilastollisia päätelmiä ei voi tehdä.

*Päivölän lukumäärät perustuvat internetistä haettuihin tuloksiin vuosilta 2000,2002-2005 ja vuosilta 1998-1999 ja 2001 osalta muistiinpanoihin menestyneistä. Lukumäärä on siten arvio.