
**N-GORPUTZEKO PROBLEMA GRABITAZIONALAREN
EBAZPENERAKO ZENBAKIZKO METODOEN
AZTERKETA.**

Mikel Antonana Otano

Konputazio zientzien eta Adimen Artifizial saila
Informatika fakultatea



Doktorego tesia
Donostia 2017

Gaien Aurkibidea

1. Sarrera.	3
2. Zenbakizko integratzaile simplektikoak.	11
3. Problemak.	37
4. Koma higikorreko aritmetika.	51
5. IRK: Puntu-finkoaren iterazioa.	61
6. IRK: Newtonen Iterazioa.	85
7. IRK: Eguzki-sistema.	115
8. Eztabaida.	119
9. Ondorioak.	139
I. Eranskinak	141
A. Zientzia konputazioa.	143
B. Ekuazioen garapenak.	161
C. Implementazioak.	173
D. Nire-Argibideak	177
Bibliografia	181

Gaien Aurkibidea (detailed)

1. Sarrera.	3
1.1. Ikerketaren testuingurua.	3
1.2. Motibazioa.	4
1.3. Helburua eta esparrua.	5
1.4. Ekarpenak.	7
1.5. Tesiaren egitura.	9
1.6. Laburpena	10
2. Zenbakizko integratzaile simplektikoak.	11
2.1. Sarrera.	11
2.2. Zenbakizko integrazio metodoak	11
2.3. Runge-Kutta metodo simplektikoak.	20
2.4. Konposizio eta Splitting metodoak.	29
2.5. Laburpena.	36
3. Problemak.	37
3.1. Sarrera.	37
3.2. Pendulu bikoitza.	38
3.3. N-Gorputzen problema.	41
3.4. Eguzki-sistema.	43
3.5. Laburpena.	49
4. Koma higikorreko aritmetika.	51
4.1. Sarrera.	51
4.2. <i>IEEE-754</i> estandarra.	51
4.3. Biribiltze errorea.	54
4.4. Biribiltze errorearen gutxitzeko teknikak.	57
4.5. Laburpena.	60

5. IRK: Puntu-finkoaren iterazioa.	61
5.1. Sarrera.	61
5.2. Hairer-en implementazioa.	62
5.3. Gure implementazioa.	64
5.4. Zenbakizko esperimentuak.	75
5.5. Laburpena.	83
6. IRK: Newtonen Iterazioa.	85
6.1. Sarrera.	85
6.2. IRK-Newton estandarra.	86
6.3. IRK-Newton eraginkorra.	91
6.4. IRK-Newton estandarra (formulazio berria).	98
6.5. IRK-Newton eraginkorra (formulazio berria).	102
6.6. IRK Newtonen iterazio Mixtoa.	106
6.7. Zenbakizko esperimentuak.	110
6.8. Laburpena.	113
7. IRK: Eguzki-sistema.	115
7.1. Sarrera.	115
7.2. Kepler-en fluxua.	115
7.3. Implementazio berria.	115
7.4. Zenbakizko esperimentuak.	116
7.5. Laburpena.	116
8. Eztabaida.	119
8.1. Eguzki-sistemaren integratiorako egungo metodoak.	119
8.2. Ereduen deskonposaketa.	125
8.3. IRK implementazioaren oinarriak.	129
8.4. IRK Newton: konplexutasun analisia.	133
8.5. Doitasun altuko konputazioak.	135
8.6. Paralelizazioa.	136
8.7. Etorkizuneko lanak.	137
9. Ondorioak.	139
I. Eranskinak	141
A. Zientzia konputazioa.	143
A.1. Sarrera.	143
A.2. Eraginkortasuna	144
A.3. Hardwarea.	147

A.4. Programazio lengoaiak	152
A.5. Aljebra lineal dentsorako liburutegiak	155
A.6. Konpiladorea	159
A.7. Laburpena	160
B. Ekuazioen garapenak.	161
B.1. Kepler hasierako baliodun problema.	161
B.2. Koordenatu sistemak.	164
B.3. Newton eraginkorraren garapena.	167
B.4. Kepler fluxuaren aldagai aldaketa.	169
C. Implementazioak.	173
C.1. Egitura orokorra.	173
C.2. IRK puntu-finkoa.	175
C.3. IRK Newton.	175
C.4. IRK Eguzki-sistema.	176
C.5. Konposizio-Splitting metodoak.	176
D. Nire-Argibideak	177
D.1. IRK-Newton.	177
D.2. FMA.	178
D.3. Hitz-zerrenda.	178
Bibliografia	181

Itxaropena ez da dena ondo aterako
den konbentzimendua; nolanahi
ateratzen dela ere, egiten dugunak
zentzua duen ziurtasuna baizik.

Vaclav Havel

1. Kapitulua

Sarrera.

1.1. Ikerketaren testuingurua.

Urte luzez, zientziaren arlo ezberdinek N-gorputzko problema ikertu dute. Astronomoek eguzki-sistemaren planeten mugimendua ulertu nahian egindako lanak edo kimikariek erreakzio kimikoekin esperimentatzeko molekulen dinamikaren azterketak aipatu daitezke. Gainera, N-gorputzen problemaren azterketak garrantzi berezia izan du matematikako eremu ezberdinen garapenean, dinamika ez-lineal eta kaos teorian esaterako.

Garai batean, N-gorputzen problemak teori analitikoen bidez aztertzen ziren baina konputagailuen sorrerarekin, zenbakizko integrazioak tresna nagusia bilakatu ziren. Azken hamarkadetan, bai konputazio teknologien aurrerapenari esker bai algoritmo berrien sorrerari esker, zenbakizko azterketek garapen handia izan dute. Zenbakizko simulazioen laguntzaz, eguzki-sistemaren dinamikaren funtsezko galdera batzuk ezagutu ditugu eta berriki, Karplus-en taldeak 2013. urteko kimika Nobel saria [84] jaso du kimika konputazionalean egindako lanarengatik.

Guk lan honetan, N-gorputzen problema grabitazionala aztertuko dugu. Oro har eta gaia kokatzeko asmoarekin, N-gorputzen ohiko zenbakizko integrazioak hiru taldetan sailka ditzakegu:

1. Epe motzeko eta doitasun handiko integrazioak. Eguzki-sistemaren efemide zehatzak [36] edo espazioko satelite artifizialen kokapenen [13] kalkuluetarako erabili ohi dira.
2. Epe luzeko baina doitasun txikiko integrazioak. Denbora epe luzean, planeta-sistemen mugimendua ezagutzeko egindako ikerketak dira. Azterketa hauetan, helburua gorputzen mugimenduaren argazki orokorra (zehaztasun handirik gabe) ezagutzea da. Normalean, problema mota hauetan gorputzen arteko kolisioak edota kolisiotik gertuko egoerak ez dira izaten.

3. N-gorputz kopurua edozein izanik, hauen arteko kolisioak gerta daitezkeen problemak. Integrazio hauetan, konplexutasun handiari aurre egin behar zaio. Gorputz kopurua miliotakoa [57] izan dateke eta kolisiotik gertuko egoeren ondorioz, kalkulueta egindako zenbakizko errore txikiiek soluzioan eragin handia izan dezakete.

Gure helburua, eguzki-sistemaren epe luzeko eta doitasun handiko integrazioetarako egoki izango den implementazio eraginkorra garatzea da. Aurreko hamarkadetan, eguzki-sistemaren planeten epe luzeko zenbakizko integrazioa erronka garrantzitsua izan da. Adibidez, Sussman-ek eta Wisdom-ek [98, 1993] eguzki-sistemaren 100 milioiko integrazioarekin, planeten mugimendua kaotikoa zela baiezttatu zuten. Aldi berean, paleoklimatologi-zientzialariak orain milioika urte gertatutako klima zikloak (epel, hotz eta glaziazio aroak) azaltzeko, luraren orbitan izandako aldaken eraginez gertatu zirela azaltzen duen teoria (Milankovitch 1941) [12] baiezatzeko, planeten orbiten efemeride zehatzetan oinarritu dira.

Epe luzeko integrazio hauetarako zenbakizko hainbat metodo erabiltzen dira, bereziki beren izaera Hamiltondarra mantentzen duten metodoak (metodo simplektikoak).

Konputazio-teknologi aurrerapenak handiak izan arren, eguzki-sistemaren simulazio hauek konputazionalki oso garestiak dira eta exekuzio denbora luzeak behar dituzte; adibidez, Laskar-ek [73, 2010] bere azken integrazioa burutzeko 18 hilabete behar izan zituen. Azken urteotako konputagailu berrien arkitekturaren bilakaerak, algoritmo azkarren diseinua aldatu du: simulazioak azkartzeko algoritmoak, paralelizazioan oinarritu behar dira. Integrazio luze hauen erronka handienetako bat, biribiltze errorearen garapena zaintza da. Biribiltze errore sistematikoaren hedapenak, errore globalean eragindako joerak ekidin behar dira [72].

1.2. Motibazioa.

Metodo simplektikoen artean erabilienak, izaera esplizituko algoritmoak dira. Oro har problema zurruna ez bada, metodo esplizituak metodo implizituak baino era-ginkorragoak dira. Metodo implizituetan ekuazio sistema ez-lineala askatu behar da (eragiketa garestia) eta honek, metodo esplizituekiko CPU denbora gainkarga suposatzen du. Hala ere, ebatzi beharreko problema zurruna bada, metodo esplizituak urrats oso txikiak eman behar izaten ditu integrazio fidagarriak lortu ahal izateko. Horrek ere, integrazioa garestitzen du. Metodo implizituetan ez da halakorik gertatzen, urrats luzeagoak eman ditzakete nahiz eta problema zurruna izan.

Azken aldian, ordea, ezaian jarri da problemaren zurruntasunaren arabera-ko metodoen aukeraketarako joera hori. Lan honetan, zenbakizko integrazorako

Gaussen metodo implizitu simplektikoaren azterketa egingo dugu. Hainbat auto-rek (Hairer [48, 49] eta Sanz Serna[60]) metodo honen potentziala nabarmendu dute. Azken urtetan, espazioko satelite artifizialen arloan ere, Gaussen integracio metodoarekiko interesa azaldu dute [17, 13].

Gaussen integracio metodo implizituen abantaila nagusienetako malgutasuna da. Ekuazio implizituak ebazteko, teknika ezberdinak konbinatu daitezke eta ondorioz, integratu nahi dugun problemari egokitzeko eta eraginkortasuna hobetzeko aukera asko eskaintzen dizkigu.

Simplektikoak diren metodo esplizituak oso eraginkorrik direla ezin da ukatu, baina metodo hauen erabilera ez da beti posible: sistema Hamiltondar banagarriean bakarrik erabil daitezke. Sistema Hamiltondar orokorrak edota lehen ordenako ekuazio differentialeko sistemak integratzeko metodo simplektikoak, implizituak izan behar dute. Bestalde, Gauss metodoak paralelizagarriak dira, hau da, ekuazio differentzial konplexuak kalkulatu behar ditugunean, s -ataletako funtzio konputazioak paraleloan exekutatu daitezke. Azkenik ez dugu ahaztu behar, ordena altuko Gauss metodoak existitzen direla eta hauek beharrezkoak ditugula doitasun handiko integracioetarako.

Atal hau bukatzeko, Sanz Sernaren [92, 1992] hitzak berreskuratuko ditugu.

On the other hand, little has been undertaken in the construccion of practical high-order methods and the design of serious symplectic software is still waiting consideration.

V.A. Brumberg-ek [20, 2012] lanean, eguzki-sistemaren epe luzeko simulazioak era honetan deskribatzen ditu.

Numerical integration of the equations of motion of celestial bodies over a long interval of time is also not a trivial problem. Analytical and numerical techniques of celestial mechanics have been permanently improved over the history of celestial mechanics. In its turn, it was a stimulatory for many branches of mathematics (the theory of special functions, linear algebra, differential equations, theory of approximation, etc.).

1.3. Helburua eta esparrua.

Gure helburua, eguzki-sistemaren epe luzeko integrazorako Gaussen metodo implizituaren implementazio eraginkorra proposatzea edota, bide horretan aurrera-pausoak ematea da. Helburu hau lortzeko, honako aspektu hauek bereziki zain-

duko ditugu: eguzki-sistemaren problemaren ezaugarriak, biribiltze erroreen garapena eta egungo konputagailuen gaitasunari egokitutako algoritmo azkarren diseinua.

N-gorputzeko problema grabitazionalari dagokionez, eguzki-sistemaren eredu simplea integratuko dugu. Eguzki-sistemaren gorputzak masa puntuak kontsideratuko ditugu eta gure ekuazio diferentzialek, gorutz hauen arteko erakarpen grabitazionalak bakarrik kontutan hartuko dituzte. Beraz, eguzki-sistemaren eredu konplexuagoetako erlatibilitatea ez da erlatabil, gorutzaren formaren eragina, eta beste zenbait indar ez-grabitazionalak ez da kontutan hartu.

Zeintzuk dira eguzki-sistemaren problemaren ezaugarri bereziak? Batetik, planeten mugimendu orbitala, perturbazio txikiak dituen mugimendu Kepleriarrada. Beraz, mugimendu Kepleriarra zehazki kalkula daitekeenez, eguzki-sistemaren planeten orbiten konputazioaren oinarria da. Bestetik, badugu gorutz nagusi bat (eguzkia) eta honen inguruan mugimenduan dauden planetak, bi multzotan bana ditzakegu: barne-planetak, masa txikikoak eta eguzkitik gertu daudenak eta kanpo-planetak, masa handikoak eta eguzkitik urrun daudenak. Kanpo-planeten eboluzioan, barne-planetak eragin oso txikia daukate, eragina, masaren eta distantziaren alderantzizkoaren proportzionala baita. Eguzki-sistema egonkorra kontsideratzen da, hau da, hurrengo bilioi urteetan planeten arteko talkarik ez da espero gertatzea. Orbiten denbora eskalak anitzak dira; ilargiaren luraren inguruko orbitaren periodoa 27.32 egunetakooa, luraren eguzkiaren ingurukoa 1 urtekoa eta Neptunorena 163 urtekoa. Eguzki-sistemaren egitura aberats honi, abantaila gehien ateratzen dion planteamendua bilatuko dugu.

Konputagailuen koma-higikorreko aritmetika ondo ulertzear garrantzitsua da. Zenbaki errealen adierazpen finitua erabiltzen denez, bai zenbakiak memorian gordetzerakoan, bai hauen arteko kalkulu aritmetikoak egiterakoan, biribiltze errorea sortzen da. Integrazio luzeetan, biribiltze errorea hedatu egiten da eta une batetik aurrera, soluzioen zuzentasuna ezereztatzen da. Zentzu honetan, doitasuna hobetzeko biribiltze errorea gutxitzen duten teknika bereziak aplikatzea ezinbestekoa izaten da. Integrazio luzeetan, maiz doitasun handian lan egiteko aukera aipatzen da, baina doitasun altuko aritmetikaren (128-bit) implementazioa software bidezkoa denez, oso motela da eta ez da erabilgarria. Exekuzio denbora onargarriak lortzeko tarteko irtenbideak landu behar dira, esate baterako, doitasun ezberdinak nahasten dituzten implementazioak.

Konputazioko teknologiaren garapenean, algoritmo azkarren diseinua baldintzatzen duten bi ezaugarri azpimarratu behar dira. Batetik, konputagailuak orokorrean paraleloak dira eta algoritmo azkarrak garatzeko, kodearen paralelizazio gaitasunari heldu behar zaio. Bestetik, konputazioaren alde garestiena, memoria eta prozesadorearen arteko datu mugimendua denez, prozesadorearen konputazio handiena komunikazio txikienarekin lortu behar da.

Sarrera honetan paralelizazioari buruzko ohar batzuk ematea komeni da. Al-

goritmo baten kode unitateak paraleloan exekutatzeak badu gainkarga bat eta beraz, algoritmoaren exekuzioa parallelizazioaz azkartzea lortzeko, unitate bakoitzaren tamainak esanguratsua izan behar du. Gure eguzki-sistemaren eredua simplea da eta logikoa da pentsatzea eredu konplexuagoetan, parallelizazioak abantaila handiagoa erakutsiko duela. Bestalde, gorputzen kopurua handia den problemetan, hauen arteko interakzio kopuru handia kalkulatu behar da ($\mathcal{O}(N^2)$) eta indar hauen hurbilpena modu eraginkorrean kalkulatzeko metodo ezagunak daude: *tree code*[11] eta *fast multipole method*[23] izeneko metodoak. Baino gure probleman gorputz kopurua txikia denez, teknika hauek gure eremutik kanpo utzi ditugu.

1.4. Ekarpenak.

Tesiaren lana hiru ataletan banatu dugu. Lehen urratsean, Gauss metodoaren urratsa emateko puntu-finkoaren iterazioaren bidezko implementazioa aztertu dugu eta gure implementazioen oinarriak finkatu ditugu. Bigarren fasean, Gauss metodoaren urratsa emateko, Newton iterazioaren bidezko implementazio eraginkorra lortzeko ahalegin berezia egin dugu. Problema zurruna denean, puntu-finkoaren iterazio ez da eraginkorra eta Newtonen iterazioa aplikatu behar da. Gainera problema ez-zurruna izanik ere, Newton iterazioak interesgarriak izan daitezke; bereziki doitasun altuko (doitasun laukoitza) konputazioetan, metodoaren konbergentzia ezaugarri onak direla-eta. Hirugarren fasean ...

Jarraian, atal bakoitzean egindako ekarpen nagusienak laburtuko ditugu:

1. IRK puntu finkoa.

Gauss metodoaren puntu finkoaren implementazioaren azterketa sakon bat egin dugu eta horretarako, Hairer-en implementazioa [49] hartu dugu gure lanaren abiapuntua. Kalitatezkoa implementazio hau hobetzeko aukerak ikuhi ditugu eta implementazio sendoago bat proposatu dugu. Gure ekarpenak hauek izan dira:

(a) Metodoaren birformulazioa.

Gauss implizitua aplikatzen dugunean, metodoa definitzen duten biribildutako koefiziente errealkak ($\tilde{a}_{ij}, \tilde{b}_i \in \mathbb{F}$) erabiltzen dira. Formula estandarra erabiliz, koefiziente hauek ez dute metodoa simplektikoa izateko baldintza zehazki betetzen eta beraz, izaera simplektikoen propietate onak galtzen dira. Metodoaren birformulazio baliokide bat proposatu dugu, horrela simplektizitatea baldintza zehazki betetzen duten koefizienteak modu errazean finkatu daitezke. Hori dela-eta, Gauss metodoak integral koadratikoak kontserbatuko ditu.

(b) Geratze irizpide berria.

Orokorrean, Hairer-en implementazioaren puntu finkoaren iterazioaren geratze irizpidea zuzena dela ikusi dugu baina kasu batuetan goizegi geratzen dela baieztatu dugu. Arrazoik bi direla ikusirik, bere geratze irizpidea bi zentzutan garatu/zorroztu dugu. Lehenik, Hairer-en implementazioan, hobekuntza neurketa ataletako differentziaren norma batean oinarritzen da. Normarekiko independentea den geratze irizpidea aplikatzea zuzenagoa da, eta horregatik, ataletako edozein osagaiaren differentzia txikitzen den bitartean iterazioak egiten jarraitza finkatu dugu. Bigarrenik, iterazioetan osagai guztien hobekuntza ez du zertan beherakorra izan behar, eta okertzen diren tarteko iterazioak gerta daitezke. Arazo hau gainditzeko, iterazioren batean osagai guztien differentzia handitzea gertatzen denean, seguritateko bi iterazio gehigarri emango ditugu, iteraziotik irten aurretik.

(c) Atalen espresioaren aldaketa.

Integrazioan batura konpensatu estandarra aplikatzen dugunean, zenbakizko soluzioa $\tilde{y}_n, e_n \in \mathbb{F}^d$ lortzen dugu non $\tilde{y}_n + e_n \approx y(t_n)$ den. Hori dela-eta, metodoaren atalen espresioan \tilde{y}_n -ren ordez, $\tilde{y}_n + e_n$ era-biltzea proposatu dugu. Aldaketa honekin, zenbakizko soluzioaren doitasuna zerbaite hobetuko dela espero da.

$$Y_{n,i} = y_n + \left(e_n + \sum_{j=1}^s \mu_{ij} L_{n,j} \right).$$

(d) Biribiltze errorearen estimazioa.

Zenbakizko soluzioaren $\tilde{y}_n + e_n \approx y(t_n), n = 1, 2, \dots$ biribiltze errorearen estimazioa, doitasun txikiagoko bigarren zenbakizko soluzioaren $\hat{y}_n + \hat{e}_n \approx y(t_n), n = 1, 2, \dots$ differentzia gisa kalkulatuko dugu. Erabiltzaileari zenbakizko soluzioaren estimazioa ezagutzeko, exekuzio bakarrean eta *CPU* gainkarga txikiarekin, bi integrazioak sekuentzialki kalkulatzeko aukera eskainiko zaio.

2. IRK Newton.

(a) Ekarpen nagusia.

S-ataletako IRK metodoa, Newton iterazioaren bidez *d*-dimentsioko ekuazio differentzial sistemari aplikatzeko, urrats bakoitzean $sd \times sd$ tamainako hainbat ekuazio sistema (iterazio bakoitzeko bat) askatu behar dira. Atal honetan, jatorrizko sd -dimentsioko ekuazio sistema,

$(s+1)d$ dimentsioko ekuazio-sistema baliokide moduan berridatzi dugu. Ekuazio-sistema baliokidea, $d \times d$ tamainako $[s/2] + 1$ matrize errealen LU-deskonposaketa bidez askatuko dugu. Tamaina txikiko matrizeen LU deskonposaketa azkarra denez, konputazionalki eraginkorra izatea espero dugu.

(b) Doitasun laukoitzeko implementazioa.

Doitasun laukoitzeko exekuzioetan, funtziobalioztapena oso garestia da eta funtziobalioztapenen kopurua gutxitzea bilatuko dugu. Newton osoa aplikatzen dugunean pena mereziko du, Gaussen Newton iterazioko implementazioaren ekuazio lineala askatzeko metodo iteratiboa aplikatzea.

(c) Newton mixtoa.

3. IRK Eguzki-sistema.

Hirugarren urratsean, eguzki-sistemaren epe luzeko integrazioan arituko gara. Ekarpenean handiena, atalen hasieraketa berri bat aplikatzea da alde Keppleriarraren fluxuan oinarrituz. IRK metodoak eskaintzen digun malgutasunari esker eta N gorputzetako problema grabitazionalaren ezaugarriez baliatuz implementazio ezberdinak egin ditugu. Implementazio hauen eraginkortasuna, egungo integratzaile simplektiko esplizituekin konparatu ditugu.

4. Birparametrizazioa.

Azken urratsean, esperimentalki, eguzki-sistemaren integrazioan denboraren birparametrizazio teknikaren aplikazio simple bat erakutsiko dugu. Integratzaile simplektikoak luzera finkoko urratsa eduki behar du eta zentzu honetan, birparametrizazioa eraginkortasuna hobetzeko beste bide bat da.

1.5. Tesiaren egitura.

Tesiaren lehenengo (1 – 5) kapituluetan, zenbakizko integrazio simplektikoak eta zientzia konputazionalaren oinarriak azaldu ditugu. Bigarren kapituluaren lehen zatian, *Zenbakizko integratzaile simplektikoen* inguruko oinarrizko kontzeptuak azaldu ditugu. Kapitulu honen bigarren zatian, Gauss metodo estandarra deskribatu eta bere propietate nagusienak eman ditugu. Kapitulu honen azken zatian, eguzki-sistemaren integraziorako metodo simplektiko eta esplizitu nagusienak laburtu ditugu. Hirugarren kapituluan, lan honetan zenbakizko esperimentuetan eraibili ditugun hasierako baliodun problemen zehaztasunak eman ditugu. Laugarren kapituluan koma-higikorreko aritmetikan murgildu gara eta biribiltze errorearen

inguruko gaiak argitu nahi izan ditugu. Bostgarren kapituluan, egungo konputazio zientziaren hardware eta software kontzeptu nagusienak ezagutarazi nahi izan ditugu.

Tesiaren (6 – 8) kapituluetan, gure implementazio berriak garatu ditugu eta zenbakizko esperimentuen bidez, hauen eraginkortasuna erakutsi dugu. Lehenik, 6. kapituluan Gauss metodoaren puntu finkoaren iterazioaren implementazio berria eman dugu. Ondoren, 7. kapituluan Gauss metodoaren Newtonen iterazioan oinarritutako implementazio eraginkorrik azaldu ditugu. 8. kapituluan, eguzkisistemaren integracioneko implementazio deskribapena egin dugu.

Tesiaren (9 – 10) kapituluetan, gure hipotesiaren eztabaidea eta lanaren konklusioak idatzi ditugu.

Tesiaren bukaeran, hiru eranskinetan lanaren informazio osagarria bildu dugu. A-eranskinean, tesian zehar erabilitako hainbat frogentzako zehaztapenak eman dira. B-eranskinean, garatutako kodeak bildu eta erabiltzaileari erabilgarri izan dakiokoen informazioa laburtu dugu. Azkenik C-eranskinean, erabilitako notazioaren inguruko argibideak eman ditugu.

1.6. Laburpena

2. Kapitulua

Zenbakizko integratzaile simplektikoak.

2.1. Sarrera.

Lan honen xede nagusia, N-gorputzeko sistema grabitazionalaren problemen zenbakizko integratorako algoritmoak garatzea edo daudenak hobetzea da. Era horretako problemen artean, eguzki-sistemaren epe luzeko integratorako implementazioak aztertuko ditugu [64, 20]. Gaur egun, era horretako integracioetarako hainbat zenbakizko metodo erabiltzen dira, bereziki bere izaera Hamiltondarra mantentzen duten metodo simplektikoak [31, 90, 32]. Metodo horien artean, gehien erabiltzen direnak izaera esplizituko algoritmoak diren arren, metodo implizituak, modu egokian implementatz gero, hauek baino eraginkorragoak izan daitezke. Zenbakizko integratorako Gauss metodo implizituaren implementazio eraginkorra aztertuko dugu.

Atal honen lehen zatian, zenbakizko integracio-metodoen inguruko oinarrizko definizioak eta kontzeptuak azalduko ditugu. Jarraian, sistema Hamiltondarrak zer diren eta halakoak integratzeko erabiltzen diren metodo simplektikoak azalduko ditugu. Ondoren, zenbakizko integratzaile simplektiko nagusienak aztertuko ditugu: alde batetik, Runge-Kutta metodo simplektikoak (implizituak); eta bestetik, konposizio eta splitting metodo simplektikoak (esplizituak).

2.2. Zenbakizko integracio metodoak

Oinarrizko kontzeptuak.

Ekuazio diferentzial arruntak, egoera aldaketak dituzten problemen azterketarako eredu matematikoak dira [47]. Zientziaren eta ingeniaritzaren arlo askotan azal-

tzen zaizkigu: planeten mugimenduen ereduetan (astronomia), erreakzio kimikoen formulazioetan, molekulen dinamiken simulazioetan, zirkuitu elektronikoen diseinuan, populazio-hazkunde eta interakzioetan (biologian), ekonomi azterketeetan, ...

Ekuazio diferentzial arrunta, $y(t)$ soluzio ezezagunaren eta bere $\dot{y}(t)$ deribatuaren arteko erlazioa da. t aldagaiak, maiz denbora adierazten du eta $y(t) \in \mathbb{R}^d$ soluzio bektorea da. Deribatua, modu esplizituan denbora eta egoeraren arabera adierazi daitekeenean, era honetako ekuazioak ditugu,

$$\dot{y}(t) = f(t, y(t)). \quad (2.1)$$

\dot{y} notazioa erabiliko dugu dy/dt adierazteko.

Ekuazio diferentzial arruntetarako hasierako baliodun problemetan, $y(t_0) = y_0 \in \mathbb{R}^d$ hasierako balio bat finkatuz, eta $f(t, y)$ funtzioa, $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$, t_0 ingurunean diferentziagarria bada, orduan hasierako baliodun problemaren soluzioa existitzen da eta bakarra da [48],

$$\dot{y}(t) = f(t, y(t)), \quad y(t_0) = y_0. \quad (2.2)$$

Goiko ekuazio-sistema bektoriala (2.2), ekuazio-sistema modu eskalarrean idatziko dugu orokorrean erabiliko dugun notazioa argitzeko:

$$\begin{aligned} \dot{y}^1(t) &= f^1(t, (y^1(t), y^2(t), \dots, y^d(t))), \\ \dot{y}^2(t) &= f^2(t, (y^1(t), y^2(t), \dots, y^d(t))), \\ &\dots, \\ \dot{y}^d(t) &= f^d(t, (y^1(t), y^2(t), \dots, y^d(t))), \\ y(t_0) &= (y_0^1, y_0^2, \dots, y_0^d). \end{aligned}$$

Oso problema gutxitarako aurki daiteke ekuazio diferentzial arrunten soluzio analitikoa (funtzio ezagunen araberako soluzio zehatza) eta beraz, ia problema gehienak zenbakizko integrazio metodoen bidez ebatzen dira. Zenbakizko integracioetan, soluzioaren hurbilpena,

$$y_n \approx y(t_n), \quad t_n = t_{n-1} + h_n, \quad n = 1, 2, \dots$$

une diskretu konkretuetarako, zehaztutako tarte batean ($t_0 \leq t \leq t_f$) lortuko da. Zenbakizko soluzioa, sekuentzialki urratsez-urrats kalkulatzen da eta lortutako balio multzoak $(t_0, y_0), (t_1, y_1), \dots, (t_f, y_f)$ zenbakizko soluzioa definitzen du.

Ekuazio diferentziala beti *sistema autonomo* moduan, hau da, denborarekiko independentea, idatz daiteke. Hori horrela izanik, notazioa sinplifikatzeko era honetako hasierako baliodun problemak konsideratuko ditugu,

$$\dot{y}(t) = f(y(t)), \quad y(t_0) = y_0. \quad (2.3)$$

Definizioak.

Jarraian zenbakizko integrazioen metodoen oinarrizko kontzeptuak eta notazioa finkatuko ditugu [26, 48].

1. Fluxua.

Fase-espazioko edozein y_0 balioari, $y(t_0) = y_0$ hasierako balioa duen $y(t)$ soluzioa esleitzen dion funtziari fluxua deitzen zaio. Izendatzeko φ_h notazioa erabiliko dugu,

$$\varphi_h(y_0) = y(t_0 + h) \text{ baldin } y(t_0) = y_0.$$

2. Zenbakizko diskretizazioa.

y_n balioa emanda, $y_{n+1} \approx y(t_{n+1})$ soluzioaren hurbilpena kalkulatzeko formulari *zenbakizko fluxua* deritzogu. Honako notazioa erabiliko dugu,

$$y_{n+1} = \phi_h(y_n). \quad (2.4)$$

3. Metodoaren ordena.

- (a) Errore lokala, $y(t_0) = y_0$ soluzio zehatzetik abiatuta, integracio metodoaren urrats bat eman ondoren lortutako zenbakizko soluzioak, benteako soluzioarekiko duen aldeari deritzo.

$$\delta(y, h) = \varphi_h(y) - \phi_h(y).$$

- (b) Errore globala, zenbakizko soluzioaren t_0 hasierako unetik t_k une arteko errore globalari esaten zaio,

$$E(t_k) = y_k - y(t_k).$$

- (c) Metodoaren ordena. h urrats luzera finkoko ϕ metodoa p ordenakoa dela esaten da, $E(t)$ errore globala $\mathcal{O}(h^p)$ ordenakoa bada $h \rightarrow 0$,

$$\|y_k - y(t_k)\| = \mathcal{O}(h^p), \quad h \rightarrow 0.$$

Metodoaren ordena $\mathcal{O}(h^p)$ bada, errore lokala $\mathcal{O}(h^{p+1})$ da.

4. Metodo simetrikoak.

Urrats bakarreko ϕ_h metodoa simetrikoa da, honako baldintza betetzen bantu,

$$\phi_h \circ \phi_{-h} = id, \quad \text{edo} \quad \phi_h = \phi_{-h}^{-1}.$$

Euler metodo esplizitua.

Eulerrek 1768. urtean proposatutako zenbakizko metodoa da. Hasierako balio bat emanda (t_n, y_n) eta $h_n > 0$ urrats txikirako, $t_{n+1} = t_n + h_n$ uneko hurbilpena $y_{n+1} \approx y(t_{n+1})$ era honetan kalkulatuko dugu,

$$y_{n+1} = y_n + h_n f(t_n, y_n).$$

Oinarrizko metodo esplizitua da eta urratsa emateko dagoen konputazio konplexutasuna bakarra, f funtzioaren ebaluazio da. Lehen ordenako metodoa da,

$$\|y_n - y(t_n)\| \leq Ch, \quad C \in \mathbb{R}$$

eta beraz, doitasuna bikoizteko, lan konputazionala bikoiztu behar dugu. Ikusiko dugunez, ordena altuagoko metodoekin, doitasuna handiagoa lortuko dugu, lan konputazional gutxiagorekin.

Euler metodo implizitua.

Eulerrek proposatutako beste metodo honek, y_{n+1} hurbilketa, implizituki definitzen den funtsezko ezaugarria du. f funtzioaren argumentua, aurreko hurbilpenaren ordez hurbilpen berria hartuz definitzen da,

$$y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1}).$$

Urratsa emateko, ekuazio-sistema ez-linealaren soluzioa askatu behar da. Horretarako, iterazio metodo bat aplikatu behar da.

Iterazio metodoak

Ekuazio-sistema ez-linealen soluzioa askatzeko bi iterazio metodo azalduko ditugu.

1. Puntu-finkoaren iterazioa.

Demagun $x = f(x)$ ekuazioa, non $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ eta $x^{[0]} \in \mathbb{R}^n$ soluzioaren hasierako hurbilpen bat emanda, puntu-finkoaren iterazioa era honetan definitzen da,

$$x^{[k+1]} = f(x^{[k]}) \quad k = 1, 2, \dots$$

Iterazioak x^* soluzioarengana konbergitu dezake.

Eta Euler metodo implizituaren ekuazio ez-lineala askatzeko, $y_{n+1}^{[0]}$ balioa finkatuta, puntu-finkoaren iterazioa era honetan aplikatuko dugu,

$$y_{n+1}^{[k+1]} = y_n + h_n f(t_{n+1}, y_{n+1}^{[k]}), \quad k = 1, 2, \dots$$

2. Newtonen iterazioa.

Demagun $f(x) = 0$ ekuazioa askatzeko, non $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ eta $x^{[0]} \in \mathbb{R}^n$ soluzioaren hasierako hurbilpen bat emanda, Newtonen iterazioa era honetan definitzen da,

$$\begin{aligned} k &= 1, 2, \dots \\ \Delta x^{[k]} J(x^{[k]}) &= -f(x^{[k]}), \\ x^{[k+1]} &= x^{[k]} + \Delta x^{[k]} \end{aligned}$$

non $J(x) = \frac{\partial f}{\partial x}(x)$ den.

Urratsaren konputazioaren konplexutasuna, metodo esplizituan baino nabarmen handiagoa da: metodo honetan, iterazio bakoitzean Jacobiarren ebaluazioa, ekuazio-sistema lineala askatu eta f funtzioren ebaluazioa kalkulatu behar dira.

Iterazio metodoaren konbergentzia abiadura. $\{x^{[0]}, x^{[1]}, \dots, x^{[k]}\}$, x^* soluziorantz konbergitzen duen bektore seriea bada, errorea $e^{[n]} = x^{[*]} - x^{[n]}$, $n = 1, 2, \dots$ izendatuko dugu. Konbergentzia p ordenakoa dela esaten dugu honakoa betetzen dada,

$$\lim_{n \rightarrow \infty} \frac{\|e^{[n+1]}\|}{\|e^{[n]}\|^p} = C \neq 0.$$

Puntu-finkoaren iterazioak konbergentzia lineala ($p = 1$) du, Newtonen iterazioak, berriz, koadratikoa ($p = 2$). Konbergentzia koadratikoa izatea oso interesgarria da, iterazio bakoitzean soluzioaren digitu hamartar zuzenen kopurua bikoizten dugulako. Konbergentzia linealean, ordea, iterazio bakoitzean digitu hamartar kopuru finkoa hobetzen dugu.

Problema zurruna.

Urrats baten konputazio-kostua handiagoa izan arren, problema batzuetan metodo implizituak, esplizituak aplikatzea baino egokiago izan daiteke, eta hori erakusteko Germund Dahlquist-en (1963) adibidea azalduko dugu,

$$\dot{y} = \lambda y, \tag{2.5}$$

non λ balio absolutuan handia eta negatiboa den. Problema honen soluzio analitikoa ezaguna da, $y(t) = e^{(t-t_0)\lambda}$, eta $t \rightarrow \infty$ doanean, soluzioa zerorantz gertatzen da. Metodo implizituaren bidezko zenbakizko soluzioa zerorantz gerturatzentz da $h > 0$ guztiarako,

$$y_n^{impl} = (1 - h\lambda)^{-n} y_0,$$

eta aldiz, metodo esplizituaren bidezkoa,

$$y_n^{expl} = (1 + h\lambda)^n y_0,$$

zerorantz gerturatuko da soilik h oso balio txikitarako, non $|1 + h\lambda| < 1$ izan behar duen. Beraz, problema honetan λ balio absolutuan handia eta negatiboa definitu dugunez (adibidez $\lambda = -10^{10}$), Euler esplizituan h urrats tamaina oso txikia erabili behar dugu.

Euler implizitua, Euler esplizitua baino eraginkorra den ekuazio differentialei problema *zurrunak* (*stiff*) esaten zaie [48]. Problema *zurrunden* artean problema garrantzitsuak ditugu, esaterako eskala anitzeko sistemak.

Sistema-Hamiltondarak.

Hamiltondar sistemak [93], ekuazio differentzial mota garrantzitsu bat dira. $H(p, q)$ funtzi leunari dagokion *Hamiltondar sistema* osatzen duten $2d$ ekuazio differentzialak, era honetan definitzen dira,

$$\begin{aligned}\frac{dp^i}{dt} &= -\frac{\partial H(p, q)}{\partial q^i}, \\ \frac{dq^i}{dt} &= +\frac{\partial H(p, q)}{\partial p^i}, \quad i = 1, \dots, d,\end{aligned}$$

non $H : \mathbb{R}^{2d} \rightarrow \mathbb{R}$ den eta $p = [p^1, \dots, p^d]^T$, $q = [q^1, \dots, q^d]^T$ domeinuaren aldagaiak diren. Egoera aldagaien bektoreen d dimentsioari, sistemaren *askatasun maila* esaten zaio. $H(p, q)$ funtziari *Hamiltonarra* deritzo, eta sistemaren energia adierazten du. Energia, integrazioan zehar konstante mantentzen da,

$$H(p(t), q(t)) = K, \quad K \in \mathbb{R}.$$

Beste notazio laburtu hau ere erabili ohi da,

$$\dot{y} = J^{-1} \nabla H(y),$$

non $y = (p, q)$, $\nabla H = (\partial H / \partial p^1, \dots, \partial H / \partial p^d; \partial H / \partial q^1, \dots, \partial H / \partial q^d)$ eta

$$J = \begin{pmatrix} 0_{d \times d} & I_{d \times d} \\ -I_{d \times d} & 0_{d \times d} \end{pmatrix}, \quad I_{d \times d} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

Hamiltondar banagarriak. Sistema dinamikoen Hamiltondarak, maiz egitura berezia du,

$$H(p, q) = T(p) + U(q).$$

Horian artean, *bigarren ordenako* ekuazio diferentzialak aipatu behar ditugu. Hauek Hamiltondar banagarrien kasu partikular bat dira. Bere Hamiltondarra

$$H((p, q) = \frac{1}{2} p^T p + U(q),$$

da eta dagokien ekuazio diferentzialak hauek dira,

$$\dot{p} = -\frac{\partial U(q)}{\partial q}, \quad \dot{q} = p. \quad (2.6)$$

Adibidea. *Kepler problema* [48]. Planoan elkar erakartzen diren bi gorputzen (adibidez eguzkia eta planeta bat) mugimendua adierazten duen problema da. Gorputz baten kokapena, koordenatu sistemaren jatorria konsideratuko dugu eta beste gorputzaren kokapenaren koordenatuak $q = (q_1, q_2)$ izendatuko ditugu. $p = (p_1, p_2)$ momentuak dira.

Funtzio Hamiltondarra hauxe da,

$$H(p_1, p_2, q_1, q_2) = \frac{1}{2}(p_1^2 + p_2^2) - \frac{1}{\sqrt{q_1^2 + q_2^2}}.$$

Dagozkion ekuazio diferentzialak,

$$\begin{aligned} \dot{p}_1 &= -\frac{q_1}{(q_1^2 + q_2^2)^{3/2}}, & \dot{p}_2 &= -\frac{q_2}{(q_1^2 + q_2^2)^{3/2}}, \\ \dot{q}_i &= p_i, \quad i = 1, 2. \end{aligned}$$

Planetaren mugimendua orbita eliptiko bat da. Honako hasierako balioei dagokien soluzioa zehatza,

$$q_1(0) = 1 - e, \quad q_2(0) = 0, \quad p_1(0) = 0, \quad p_2(0) = \sqrt{\frac{1+e}{1-e}},$$

e ezentrizidadea ($0 \leq e < 1$) eta $P = 2\pi$ periodoa duen elipsea da.

Hamiltondar perturbatuak. Hamiltondar perturbatuak, honako egitura duten sistemak ditugu,

$$H = H_A + \epsilon H_B \text{ non } |H_B| \ll |H_A|.$$

Adibidea. Eguzki-sistemaren probleman [90, 101], Hamiltondarra modu honeitan bana daiteke $H = H_k + H_I$, non alde nagusia H_K planeta bakoitzaren eguzkiaren inguruko mugimendu Keplerrarrari dagokion eta H_I , aldiz, planeten arteko interakzioek eragiten duten perturbazio txikiari.

Metodo simplektikoak.

Hamiltondar sistemaren problemetarako, Euler esplizitua eta Euler implizitua ez dira zenbakizko metodo egokiak, ez baitute Hamiltondarra mantentzen. Problema hauen propietate geometrikoak mantentzen dituzten integratzaile bereziak beharrezkoak dira [60, 94]. Integratzaile hauek, metodo simplektikoak dira eta abantaila handiena, epe luzeko integrazioetan azaltzen dute.

Metodo simplektikoen lehen aipamenak, 1950 hamarkadan kokatu behar dira eta 1980 hamarkadan, Feng Kang-ek metodo hauen azterketa sakona burutu zuen. Hastapenetako lan monografiko hau [60] eta ondorengo, azalpen ulergarriak jasotzen dituzten [48] eta [76] lanak azpimarratu daitezke.

Adibidea. Metodo simplektikoen abantaila azaltzeko, penduluaren problema aukeratu dugu [44]. Penduluaren problema (masa $m = 1$, $l = 1$ luzerako makila eta $g = 1$ grabitazioa) $d = 1$ askatasuneko sistema Hamiltondarra da:

$$H(p, q) = \frac{p^2}{2} - \cos(q), \quad (2.7)$$

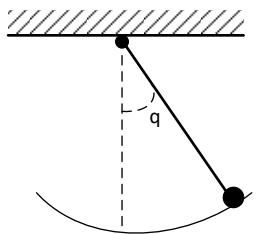
eta ekuazio diferenzialak honakoak dira,

$$\dot{p} = -\sin(q), \quad \dot{q} = p. \quad (2.8)$$

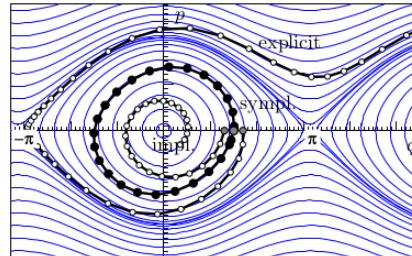
2.1. irudian ikusi daitekeenez, Euler esplizitu eta implizituaren portaera kualitatiboki okerra da. Euler simplektikoak ordea, sistemaren energia kontserbatzen du, bere definizioa [48] ikus daiteke:

$$\begin{aligned} p_{n+1} &= p_n - h \frac{\partial H}{\partial q}(p_{n+1}, q_n), \\ q_{n+1} &= q_n + h \frac{\partial H}{\partial p}(p_{n+1}, q_n). \end{aligned} \quad (2.9)$$

Metodo simplektikoak bi taldeetan banatuko ditugu: metodo simplektiko implizituak eta esplizituak. Lehenengo taldean, Runge-Kutta metodo implizituak (Gauss kolokazio metodoak) ditugu. Metodo hauek zenbakizko metodo estandarrak dira eta ekuazio diferenzial orokorreai aplika dakizkieke. Bigarren taldean, konposizioan eta splitting-ean oinarritutako metodoak ditugu. Azken talde honetako metodoak, bakarrik Hamiltondar sistemei aplika dakizkieke eta gainera, Hamiltondarren banagarria izan behar du. Konposizio eta splitting metodoak oso eraginkorrak dira.



(a) Pendulua.



(b) Integrazioa.

2.1. Irudia: Pendulu problemaren hiru zenbakizko metodoen zenbakizko soluzioak irudikatu ditugu. Hiruretan urrats luzera berdina $h = 0.3$ baina bakoitza hasierako balio ezberdinarekin. Euler esplizitua $(p(0), q(0)) = (0, 1.7)$; Euler simplektikoa $(p(0), q(0)) = (0, 1.5)$; Euler implizitua $(p(0), q(0)) = (0, 1.3)$.

Metodo simplektikoen propietateak.

Integratzaile simplektiko baten bidez sistema Hamiltondar baten soluzioa lortzen dugunean, ez dugu lortzen jatorrizko sistemaren soluzio zehatza, baina lortutako soluzio hori bera jatorrizko sistemaren perturbazio baten soluzio zehatza da [94]. Alegia, lortuta soluzioa, beste sistema Hamiltondar baten soluzio zehatza da.

Metodo simplektikoen propietate nagusienak azpimarratuko ditugu [48, 60].

1. Metodo simplektikoek, energia oso ondo kontserbatzen dute.

Sistema Hamiltondarren $H(p, q)$ funtzi leunari, sistemaren energia deitzen zaio. Soluzio zehatzak, energia konstantea mantentzen du,

$$H(p(t), q(t)) = K, \quad K \in \mathbb{R}.$$

Zenbakizko soluzioak ordea, ez du $H(p_n, q_n)$ konstante mantentzen. Integrazio metoda simplektikoa bada, aritmetika zehatzarekin energia errorea bornatua mantentzen da eta drift gabe. Koma-higikorreko aritmetikarekin, biribiltze errorearen eraginez energia errorea denboraren erro karratuaren arabera hasiko da,

$$\frac{H(p_n, q_n) - H(p_0, q_0)}{H(p_0, q_0)} = K \sqrt{t_n}.$$

non $K \in \mathbb{R}$ den.

2. Errorearen hedapen lineala.

Gehienetan, p eta q egoera aldagaien errorea $\mathcal{O}(h^p)$ da eta hazkunde lineala. Oro har, integratzaile arruntek errore hazkunde koadratikoa dute eta ondorioz, integrazio luzeetan portaera txarra azaltzen dute. Horregatik, metodo simplektikoak egokiak dira integrazio luzetarako.

3. Metodo simplektiko garrantzitsuenak simetrikoak dira.
4. Izaera simplektikoa mantendu beharrik ez dagoenean, erabiltzen diren metodoek eraginkortasuna hobetzeko, urrats luzera egokitu egiten dute, baina metodo simplektikoak, urrats luzera finkoarekin erabiltzen dira [60].

2.3. Runge-Kutta metodo simplektikoak.

1988. urtean, Sanz-Serna [60] *Gauss metodoa* simplektikoa zela ohartu zen. Runge-Kutta metodo inplizitua da eta s-atalekin lor daitekeen ordena altuena du, alegia $p = 2s$ ordenakoa da.

Runge-Kutta metodoak.

Runge-Kutta metodoak [48], urrats bakarreko ekuazio diferentzial arrunten zenbakizko integrazio metodoak dira. b_i , a_{ij} eta $c_i = \sum_{j=1}^s a_{ij}$ ($1 \leq i, j \leq s$) koefiziente errealek s-ataleko Runge-Kutta metodoa definitzen dute eta *Butcher* izeneko taulan moduan laburtu ohi dira,

$$\begin{array}{c|ccccc} c & A \\ \hline b^T & & & & & \end{array}, \quad \begin{array}{c|ccccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} \quad (2.10)$$

Hasierako baliodun problemaren (2.2) $y_n \approx y(t_n)$ soluzioaren hurbilpena era honetan kalkulatzen da,

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i f(t_n + c_i h, Y_{n,i}) , \quad (2.11)$$

non $Y_{n,i}$ atalak era honetan definitzen diren,

$$Y_{n,i} = y_n + h \sum_{j=1}^s a_{ij} f(t_n + c_j h, Y_{n,j}) \quad i = 1, \dots, s. \quad (2.12)$$

Runge-Kutta metodoak, honako integrala hurbiltzen du:

$$y(t_0 + h) = y(t_0) + h \int_0^1 \dot{y}(t_0 + \theta h) d\theta,$$

hurrengo koadratura formularen bidez:

$$y_1 = y_0 + h \sum_{i=1}^s b_i f(t_0 + c_i h, Y_i).$$

non $\dot{y}(t) = f(t, y(t))$ den, b_i pisuak eta c_i nodoak diren.

Era berean, $Y_i \approx y(t_0 + c_i h)$ atal bakoitza ere, 0 eta c_i arteko integralaren hurbilketa da. Eta hauek ere, honako koadratura formularekin hurbiltzen dira:

$$Y_{n,i} = y_0 + h \sum_{j=1}^s a_{ij} f(t_0 + c_j h, Y_j).$$

Metodo esplizituak (ERK) eta implizituak (IRK).

Runge-Kutta metodoak bi talde nagusitan bereiz ditzakegu: esplizituak (*ERK*) non $\forall i \geq j, a_{ij} = 0$ eta implizituak (*IRK*) non $\exists i \geq j, a_{ij} \neq 0$.

1. *ERK* metodoaren adibidea.

$s = 4$ ataletako ERK metodo klasikoa era honetan definitzen da,

$$\begin{array}{c|ccc} & 0 & & \\ \hline 1/2 & & 1/2 & \\ 1/2 & & 0 & 1/2 \\ 1 & & 0 & 0 & 1 \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}$$

$Y_{n,i}$ atalak, esplizituki kalkula daitezke,

$$Y_{n,i} = y_n + h \sum_{j=1}^{i-1} a_{ij} f(t_n + c_j h, Y_{n,j}) \quad i = 1, \dots, s.$$

2. *IRK* metodoaren adibidea.

Honakoa da, $s = 1$ ataleko *Implicit Midpoint method*-aren definizioa,

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$$

$Y_{n,1}$ atala implizituki definituta dago, eta honako ekuazio ez-lineala ebatzi behar da,

$$Y_{n,1} = y_n + \frac{h}{2} f\left(t_n + \frac{h}{2}, Y_{n,1}\right).$$

ERK lau-ataletako metodo klasikoa, $p = 4$ ordenakoa dugu. Ordena altuko *ERK* metodoak aurkitzea konplexua da, koefizienteek bete behar dituzten ordena baldintzen kopurua esponentzialki hazten baita. Esate baterako, ordena altuko *ERK* metodo hauek aurkitu dira: $s = 11$ ataletako $p = 8$ ordenako metodoa, $s = 17$ ataletako $p = 10$ ordenako metodoa eta $s = 25$ ataletako $p = 12$ ordenako metodoa.

Ordena altuko *IRK* metodoak, *ERK* metodoak baino modu errazagoan eraiki daitezke. *Butcher-en simplifikazio baldintzen* [22] arabera definitzen dira.

Simplektikotasuna.

Sanz-Sernak [60] frogatu zuen, Runge-Kutta metodoa simplektikoa izan dadin honako baldintza betetzea nahikoa dela, baina aldi berean beharrezkoa dela:

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (2.13)$$

Baldintza honen arabera, Runge-Kutta metodo esplizituak, simplektikoa ezin daitezkeela izan ondorioztatu daiteke. Bestalde, 1988. urtean, Sanz-Serna [60] ohartu zen *Gauss metodoa* simplektikoa zela.

Gauss metodoa.

Gauss metodoak, bi ezaugarri nagusi dauzka: Runge-Kutta metodo simplektikoa da eta bestetik, s -atalekin eduki daitekeen ordena altuena dauka: $p = 2s$.

Kolokazio metodoak, zenbakizko integrazio metodo garrantzitsuak dira. Gauss metodoa, kolokazio metodoa da eta jarraian, ikuspegi honetatik definituko dugu.

Kolokazio metodoak.

c_1, c_2, \dots, c_s ($0 \leq c_i \leq 1$) nodoetan oinarritutako kolokazio metodoak, s -mailako $u(t)$ polinomioa definitzen du. Polinomioak honakoa betetzen du,

$$\begin{aligned} u(t_0) &= y_0, \\ \dot{u}(t_0 + c_i h) &= f(t_0 + c_i h, u(t_0 + c_i h)), \quad i = 1, \dots, s. \end{aligned}$$

eta soluzioa

$$y_1 = u(t_0 + h) \approx y(t_0 + h).$$

Kolokazio metodoen zenbakizko soluzioak, diskretizazio puntuetan ez ezik, interpolazio polinomio batek modu jarraian emandako soluzioa adierazten du.

Kolokazio metodoaren definizioa eta modu honetan kalkulatutako s-ataleko Runge-Kutta metodoa baliokideak dira [48],

$$a_{ij} = \int_0^{c_i} l_j(\tau) d\tau, \quad b_i = \int_0^1 l_i(\tau) d\tau \quad (2.14)$$

non $l_i(\tau)$, Lagrangiar polinomioa dugun,

$$l_i(\tau) = \prod_{l \neq i} \frac{(\tau - c_l)}{(c_i - c_l)}. \quad (2.15)$$

Aukeratutako c_i ($1 \leq i \leq s$) koefizienteen arabera, kolokazio metodo ezberdinak eraikitzen dira. Koefizienteak era honetan finkatzen badira,

$$c_i = \frac{1}{2}(1 + \bar{c}_i)$$

non \bar{c}_i , s -mailako Legendre polinomioaren zeroak diren, orduan Gauss kolokazio metodoa lortzen dugu. Nodo hauetan oinarritutako Runge-Kutta metodoak, $p = 2s$ ordena du.

Gauss metodoaren koefizienteak.

$s = 1$, $s = 2$ eta $s = 3$ atalako, Gauss kolokazio metodoaren [48] *Butcher* koefiziente taulak [48] zehaztuko ditugu,

$$\begin{array}{c|cc} \frac{1}{2} & \frac{1}{2} \\ \hline 1 & \end{array}, \quad \begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \hline \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array},$$

$$\begin{array}{c|cccc} \frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{15} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\ \hline \frac{1}{2} & \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\ \hline \frac{1}{2} + \frac{\sqrt{15}}{10} & \frac{5}{36} + \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}.$$

Gauss metodoaren propietateak.

Gauss metodoen ezaugarri nagusien artean honakoak daude:

1. Metodo simplektikoa ([2.13](#)) da.

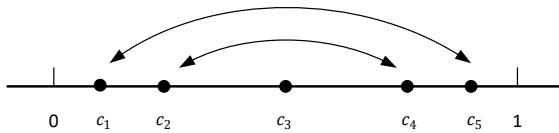
2. Metodo simetrikoa da.

Runge-Kutta metodoa, simetrikoa izateko baldintzak hauek dira,

$$\begin{aligned} b_i &= b_{\sigma(i)}, \quad c_{\sigma(i)} = 1 - c_i, \\ b_j &= a_{\sigma(i), \sigma(j)} + a_{i,j}, \quad i = 1, 2, \dots, \lfloor s/2 \rfloor \end{aligned}$$

non $\sigma(i) = s + 1 - i$.

Kolokazio metodo simetrikoetan, c_i balioak integrazio urratsaren erdigu-nearekiko simetrikoak dira (ikus [2.2.](#) irudia).



2.2. Irudia: Kolokazio metodoen simetria.

3. Ordena altuko metodoa. Edozein ordenako Gauss metodoa eraiki daiteke. Doitasun handiko konputazioetarako ordena altuko metodoak beharrezkoak dira: doitasun bikoitzeko aritmetikan (biribiltze unitatea $u \approx 10^{-16}$) $p \geq 8$ ordenako metodoak gomendagarriak dira eta doitasun laukoitzeko aritmetikan (biribiltze unitatea $u \approx 10^{-35}$) ordena altuagoko metodoak beharrezkoak dira.
4. Metodo orokorra da. Gauss metodoa edozein ekuazio diferentziala ebazteko erabil daiteke. Sistema Hamiltondarren kasuan, ez du zertan banagarria izan behar Hamiltondarra.
5. Paralelizagarria da. Ekuazio diferenzial garestiak ditugunean, s -ataletako funtzioen konputazioak ($f(Y_i)$, $i = 1, \dots, s$) paraleloan kalkula daitezke.
6. Kolokazio metodoa da. Zenbakizko soluzioak diskretizazio puntuetaez ezik, urrats bakoitzaren integrazio tarte osoaren polinomio interpolatzaile batek modu jarraian emandako soluzioa adierazten du.

7. Birparametrizazioa. Metodo simplektiko implizuetan, birparametrizazio teknika zuzenean aplika daiteke. Eguzki-sistemaren integrazioetarako tresna baliagarria dela frogatu da [40].

IRK implementazioa.

IRK algoritmoa.

IRK metodoaren implementazio orokorra, 1 algoritmoan ikus daiteke. Implementazio eraginkorra egin ahal izateko, algoritmo horretako bete behar garrantzitsuenak ondo kudeatzea komeni da.

```

 $y_0 = y(t_0);$ 
for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $k = 0;$ 
    Hasieratu  $Y_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
    while (konbergentzia lortu) do
         $k = k + 1;$ 
         $F_{n,i}^{[k]} = f(t_n + c_i h, Y_{n,i}^{[k-1]})$ ;
        Askatu ( $Y_{n,i}^{[k]} = y_n + h \sum_{j=1}^s a_{ij} F_{n,j}^{[k]}$ );
        konbergentzia  $\leftarrow$  GeratzeErizpidea( $Y^{[k]}$ ,  $Y^{[k-1]}$ );
    end
     $\delta_n = h \sum_{i=1}^s b_i F_{n,i}$ ;
     $y_{n+1} = y_n + \delta_n$ ;
end
```

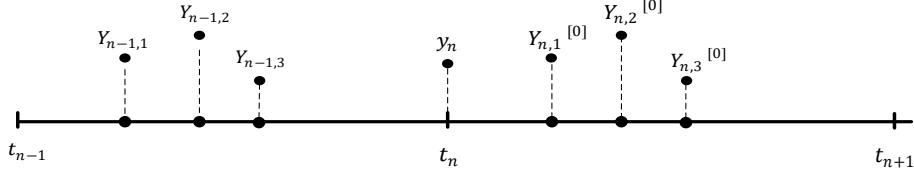
Algoritmoa 1: IRK Algoritmo orokorra

Atalen hasieraketa.

Atalen hasieraketa egokia definitu behar da. Aukera simpleena $Y_{n,i}^{[0]} = y_n$ hasiera-tzea da, baina aurreko urratseko atalen informazioa erabiliz hurbilketa hobea lor daiteke [48]. Aurreko urratseko atalak interpolatzen dituen polinomioaren bidezko hasieraketa era honetan adieraz dezakegu,

$$Y_{n,i}^{[0]} = g(Y_{n-1,i}), \quad i = 1, \dots, s.$$

Aurreko urratseko $Y_{n-1,i}$ atalak eta $(t_{n-1} + h, y_n)$ zenbakizko soluzioa ezagututa, polinomio interpolatzialearen bidez, urrats berriaren atalen hasieraketa $Y_{n,i}^{[0]}$ kalkula daiteke (2.3. Irudia).



2.3. Irudia: Interpolazioa.

1. ($n - 1$). urratsaren informazioa.

$$\begin{cases} Y_{n-1,i} = y_{n-1} + h \sum_{j=1}^s a_{ij} f(Y_{n-1,j}), \\ y_n = y_{n-1} + h \sum_{j=1}^s b_j f(Y_{n-1,j}), \end{cases} \Rightarrow Y_{n-1,i} = y_n + h \sum_{j=1}^s (a_{ij} - b_j) f(Y_{n-1,j}).$$

2. Polinomio interpolatzaila.

$$P(t) = l_1(t)Y_{n-1,1} + \dots + l_s(t)Y_{n-1,s} + l_{s+1}(t)y_n$$

non $l_i(t)$ Lagrangiar polinomioa dugun,

$$l_i(t) = \prod_{l \neq i, l=1}^{s+1} \frac{(t - (t_{n-1} + hc_l))}{(c_i - c_l)}, \quad c_{s+1} = 1.$$

3. Atalen hasieraketa.

$$Y_{n,i} \approx Y_{n,i}^{[0]} = P(t_n + hc_i) = y_n + h \sum_{j=1}^s \lambda_{ij} f(Y_{n-1,j}).$$

Modu honetan s-ataletako IRK metodo bakoitzari dagokion, λ_{ij} interpolazio-rako koefizienteak lortu daitezke. Polinomio interpolatzailaren bidezko hasieraketa, emandako urratsa oso handia eta problema zurruna ez bada, ona izango da. Era berean aipatu nahi genuke, atal askotako metodoetan (adibidez $s = 16$) interpolaziozko koefizienteen kalkuluan ezabapen arazoak, doitasun handian lan egitera behartzen gaituela interpolaziozko hasieraketa ona izateko.

Laburta-ren lanean [70], hasieraketa aurreratuei buruzko informazioa aurki daiteke.

Iterazio metodoa.

IRK metodoen erronka handiena, ekuazio-sistema ez-linealaren zenbakizko soluzioaren implementazio eraginkorra da [48, 94]. Problema ez-zurrenetarako, atalen hasieraketa ($Y_i^{[0]}$) egoki bat duen puntu-finkoaren iterazioa erabil daiteke. Problema zurrenetarako, puntu-finkoaren iterazioan konbergentzia egon dadin, urrats tamaina txikiegia erabiltzea behartuko luke eta ondorioz, Newtonen iterazio sinplifikatua erabili ohi da.

1. Puntu-finkoaren iterazioa.

```

for (k=1,2,...) do
     $F_i^{[k]} = f(t_n + c_i h, Y_i^{[k-1]});$ 
     $Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} F_j^{[k]}, \quad i = 1, \dots, s;$ 
end

```

Algoritmoa 2: Puntu-finkoaren iterazioa.

Konbergentzia $\|Y^k - Y\| = \mathcal{O}(h)\|Y^{k-1} - Y\|$.

2. Newtonen iterazioa.

Newtonen iterazio bakoitza konputazionalki garestia da. Batetik, $\frac{\partial f}{\partial y}(t_n + c_i h, Y_i^{[k-1]})$, $i = 1, \dots, s$ Jacobiarak ebaluatu behar dira. Bestetik, s-ataletako IRK metodoa eta d-dimentsioko ekuazio diferenziala baditugu, $sd \times sd$ dimentsioko sistema lineala ebatzi behar da.

```

for (k=1,2,...) do
     $r_i^{[k]} = -Y_i^{[k-1]} + y_n + h \sum_{j=1}^s a_{ij} f(t_n + c_i h, Y_j^{[k-1]});$ 
    Askatu  $\Delta Y_i^{[k]}$ ;
     $\Delta Y_i^{[k]} - h \sum_{j=1}^s a_{ij} \frac{\partial f}{\partial y}(t_n + c_j h, Y_j^{[k-1]}) \Delta Y_j^{[k]} = r_i^{[k]};$ 
     $Y_i^{[k]} = Y_i^{[k-1]} + \Delta Y_i^{[k]}, \quad i = 1, \dots, s;$ 
end

```

Algoritmoa 3: Newton metodoaren iterazioa

Konbergentzia $\|Y^k - Y\| = \mathcal{O}(h^2)\|Y^{k-1} - Y\|$.

Hamiltondar banagarriak (Metodo partizionatuak).

Era honetako ekuazio diferenzialak, garrantzitsuak dira,

$$\dot{p} = f(q), \quad \dot{q} = g(p).$$

Esaterako, Hamiltondar banagarriak $H(q, p) = T(p) + U(q)$ eta bigarren ordenako ekuazio diferenzialak $\ddot{q} = f(q)$ era honetako ekuazio diferenzialen kasu partikularrak dira.

Zenbakizko soluzioa $(p_{n+1}, q_{n+1}) \approx (p(t_{n+1}), q(t_{n+1}))$ era honetan kalkulatuko dugu [60],

$$\begin{aligned} p_{n+1} &= p_n + h \sum_{i=1}^s b_i f(t_n + c_i h, Q_{n,i}), \\ q_{n+1} &= q_n + h \sum_{i=1}^s b_i g(t_n + c_i h, P_{n,i}), \end{aligned} \quad (2.16)$$

non $P_{n,i}, Q_{n,i}$ $i = 1, \dots, s$ atalak, honako ekuazio-sistemaren bidez definitutakoak diren,

$$\begin{aligned} P_{n,i} &= p_n + h \sum_{j=1}^s a_{ij} f(t_n + c_j h, Q_{n,j}), \\ Q_{n,i} &= q_n + h \sum_{j=1}^s a_{ij} g(t_n + c_j h, P_{n,j}). \end{aligned} \quad (2.17)$$

Problema hauetan, funtsean puntu-finkoaren iteraziona (2 algoritmoa) aplikatzean, Hamiltondarraren egiturari esker, iterazioaren konbergentzia hobetu egiten da,

```
for ( $k=1,2,\dots$ ) do
   $P_{n,i}^{[k]} = p_n + h \sum_{j=1}^s a_{ij} f(t_n + c_j h, Q_{n,j}^{[k-1]});$ 
   $Q_{n,i}^{[k]} = q_n + h \sum_{j=1}^s a_{ij} g(t_n + c_j h, P_{n,j}^{[k]}), \quad i = 1, \dots, s;$ 
end
```

Algoritmoa 4: Puntu-finkoaren iteraziona (Metodo partizionatuak).

Bigarren ordeneko EDA

Bigarren ordenako ekuazio diferenzialen $\dot{q} = f(q)$ azterketa egiteko, lehen ordenako ekuazio diferenzial moduan idatziko dugu,

$$\dot{p} = f(q), \quad \dot{q} = p.$$

Zenbakizko soluzioa $(p_{n+1}, q_{n+1}) \approx (p(t_{n+1}), q(t_{n+1}))$ era honetan kalkula daiteke [60],

$$\begin{aligned} p_{n+1} &= p_n + h \sum_{i=1}^s b_i f(t_n + c_i h, Q_{n,i}), \\ q_{n+1} &= q_n + h p_n + h^2 \sum_{i=1}^s \beta_i f(t_n + c_i h, Q_{n,i}), \end{aligned} \quad (2.18)$$

non $Q_{n,i}$, $i = 1, \dots, s$ atalak, honako ekuazio-sistemen bidez definitutakoak diren,

$$Q_{n,i} = q_n + h\gamma_i p_n + h^2 \sum_{j=1}^s \tilde{a}_{ij} f(t_n + c_j h, Q_{n,j}). \quad (2.19)$$

Bigarren ordenako ekuazio diferenzialen kasuan, puntu-finkoaren iterazioa modu eraginkorrean aplika daiteke,

```
for ( $k=1,2,\dots$ ) do
     $Q_{n,i}^{[k]} = q_n + h\gamma_i p_n + h^2 \sum_{j=1}^s \tilde{a}_{ij} f(t_n + c_j h, Q_{n,j}^{[k-1]});$ 
end
```

Algoritmoa 5: Puntu-finkoaren iterazioa (bigarren ordenako EDA)

Batura Konpensatua.

Zenbakizko integrazioaren urrats bakoitzean,

$$y_{n+1} = y_n + \delta_n,$$

batura kalkulatu behar dugu. Normalean $|\delta_n| \ll |y_n|$ izango da eta integrazio luzeetan, batura honek soluzioaren doitasun galera eragingo du. Hau ekiditeko *batura konpensatu* teknika [83, 53, 48] erabili ohi da. Teknika hau 4.atalean deskribatu dugu eta zehaztapenak 11 algoritmoan eman ditugu.

2.4. Konposizio eta Splitting metodoak.

Konposizio eta Splitting ideietan oinarrituz, aplikazio eremu ezberdinatarako hainbat integratziale simplektiko [94] garatu dira. Metodo hauek ez dira orokorrak, problema zehatzetan aplikagarriak baizik, eta metodo oso eraginkorrapak dira.

Konposizio metodoak.

Konposizio metodoak, oinarrizko metodo bat edo gehiago konposatuz eraikitako zenbakizko integrazio metodoak dira [48]. Oinarrizko metodoekin segidan exekutatutako azpi-urrats kopuru batek, konposizio metodoaren integrazioaren urrats bat osatzen du. Helburua, orden baxuko metodo batetik abiatuta, ordena altuko metodoa eraikitzea da; konposizio metodoak, konposatutako oinarrizko metodoaren propietateak (simetrikoa, simplektikoa,...) jasotzen ditu.

Konposizio orokorrak.

ϕ_h oinarritzko metodoa eta $\gamma_1, \dots, \gamma_s$ zenbaki errealki emanik, urrats luzera hauen $\gamma_1 h, \gamma_2 h, \dots, \gamma_s h$ konposaketari dagokion konposizio metodoa era honetan definituko dugu,

$$\Psi_h = \phi_{\gamma_s h} \circ \dots \circ \phi_{\gamma_1 h}. \quad (2.20)$$

Algoritmoa.

Jarraian, s-ataletako konposizio metodoen implementazioaren 6 algoritmo orokorra laburtu dugu.

```

for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $Y_{0,n} = y_n;$ 
    for  $i = 1, 2, \dots, s$  do
         $| Y_{i,n} = \phi_{\gamma_i h}(Y_{i-1,n});$ 
    end
     $y_{n+1} = Y_{s,n};$ 
end
```

Algoritmoa 6: Konposizio metodoen implementazioa.

Konposizio metodoen implementazioaren ezaugarri nagusienak hauek dira:

1. Esplizituak dira.

Konposizio metodo hauek esplizituak dira. Metodo hauetan ez da ekuaziosistemarik askatu behar, eta beraz implementazioa simplea da.

2. Sekuentzialak dira.

Azpi-urrats bakoitzaren kalkulua modu sekuentzialean ($i = 1, \dots, s$) egin behar dugu.

3. Memoria gutxi behar dute.

Ez da tarteko baliorik eta datu-egitura berezirik memorian gorde behar.

4. Bigarren ordenako ekuazio differentzialentzat egokiak dira.

Bigarren ordenako ekuazio differentziala ($\ddot{q} = f(p)$), *Störmer-Verlet*(leapfrog izenarekin ere ezaguna) metodoan oinarritutako,

$$\begin{aligned}
 q_{n+1/2} &= q_n + \frac{h}{2} p_n, \\
 p_{n+1} &= p_n + h f(q_{n+1/2}), \\
 q_{n+1} &= q_{n+1/2} + \frac{h}{2} p_{n+1},
 \end{aligned} \quad (2.21)$$

s -ataletako konposizio metodoarekin integratzeko, urrats bakoitzean, ekua-zio diferenzialaren s balioztapena egin behar ditugu.

Konposizio simetrikoak.

ϕ_h oinarrizko metodoa $p = 2$ ordenakoa eta simetrikoa izanik, era honetako konposizioak aurkitu dira,

$$\Psi_h = \phi_{\gamma_s h} \circ \phi_{\gamma_{s-1} h} \circ \cdots \circ \phi_{\gamma_2 h} \circ \phi_{\gamma_1 h} \quad (2.22)$$

non $\gamma_s = \gamma_1, \gamma_{s-1} = \gamma_2, \dots$

CO1035: 10 ordenako konposizio metodoa.

Sofroniou eta Spaletta-ren [96, 2004] $s = 35$ eta $p = 10$ ordenako metodoa, orain arteko ordena altueneko konposizio metodo eraginkorrena bezala har daiteke (2.1. taula). Konposizio metodo simetriko : oinarrizko metodoa simetrikoa eta $p = 2$ ordenakoa da.

2.1. Taula: 10 ordenako metodoa konposizio metodoa [48, 158.or] (CO1035).

Koefizientea	Balioa	Koefizientea	Balioa
$\gamma_1 = \gamma_{35}$	0.07879572252168641926390768	$\gamma_{10} = \gamma_{26}$	-0.39910563013603589787862981
$\gamma_2 = \gamma_{34}$	0.31309610341510852776481247	$\gamma_{11} = \gamma_{25}$	0.10308739852747107731580277
$\gamma_3 = \gamma_{33}$	0.02791838323507806610952027	$\gamma_{12} = \gamma_{24}$	0.41143087395589023782070412
$\gamma_4 = \gamma_{32}$	-0.22959284159390709415121340	$\gamma_{13} = \gamma_{23}$	-0.00486636058313526176219566
$\gamma_5 = \gamma_{31}$	0.13096206107716486317465686	$\gamma_{14} = \gamma_{22}$	-0.39203335370863990644808194
$\gamma_6 = \gamma_{30}$	-0.26973340565451071434460973	$\gamma_{15} = \gamma_{21}$	0.05194250296244964703718290
$\gamma_7 = \gamma_{29}$	0.07497334315589143566613711	$\gamma_{16} = \gamma_{20}$	0.05066509075992449633587434
$\gamma_8 = \gamma_{28}$	0.11199342399981020488957508	$\gamma_{17} = \gamma_{19}$	0.04967437063972987905456880
$\gamma_9 = \gamma_{27}$	0.36613344954622675119314812	γ_{18}	0.04931773575959453791768001

Hamiltondarra $H = H_1 + H_2$ izanik, eta *Stömer-Verlet* metodoan (2.21) ($\phi_h = \varphi_{h/2}^{H_1} \circ \varphi_h^{H_2} \circ \varphi_{h/2}^{H_1}$) oinarritutako konposizio metodoaren implementazioaren zehaztasunak honakoak dira.

1. Konposizio metodo orokorra.

$$\Psi_h = \phi_{\gamma_s h} \circ \phi_{\gamma_{s-1} h} \circ \cdots \circ \phi_{\gamma_2 h} \circ \phi_{\gamma_1 h}$$

2. *Stömer-Verlet* oinarrizko metodoaren konposizio metodoa,

$$\Psi_h = (\varphi_{h\gamma_s/2}^{H_1} \circ \varphi_{h\gamma_s}^{H_2} \circ \varphi_{h\gamma_s/2}^{H_1}) \circ \cdots \circ (\varphi_{h\gamma_1/2}^{H_1} \circ \varphi_{h\gamma_1}^{H_2} \circ \varphi_{h\gamma_1/2}^{H_1}).$$

3. Jarraian dauden φ^{H_1} fluxuak era honetan elkartu daitezke,

$$\Psi_h = \varphi_{ha_{s+1}}^{H_1} \circ \varphi_{hb_s}^{H_2} \circ \varphi_{ha_s}^{H_1} \circ \cdots \circ \varphi_{hb_2}^{H_2} \circ \varphi_{ha_2}^{H_1} \circ \varphi_{hb_1}^{H_2} \circ \varphi_{ha_1}^{H_1}$$

non $a_1 = a_{s+1} = \gamma_1/2$, $b_i = \gamma_i$, $a_k = (\gamma_k + \gamma_{k-1})/2$, $i = 1, \dots, s$ eta $k = 2, \dots, s$.

4. Azkenik, integrazioaren tarteko urratsetan, lehen atala $\varphi_{ha_{s+1}}^{H_1}$ eta azkena $\varphi_{ha_1}^{H_1}$ bakar batean elkartu daitezke,

$$\Psi_h = \varphi_{h2a_{s+1}}^{H_1} \circ \varphi_{hb_s}^{H_2} \circ \varphi_{ha_s}^{H_1} \circ \cdots \circ \varphi_{hb_2}^{H_2} \circ \varphi_{ha_2}^{H_1} \circ \varphi_{hb_1}^{H_2}.$$

Splitting metodoak.

Splitting metodoak, $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ sistema osoa integratzeko, $f^{[i]}$ ($f = \sum_{i=1}^m f^{[i]}$) azpiproblemek deskonposatu daitezkeen ekuazio diferentzialetarako zenbakizko integrazio metodoak dira [94, 48].

Splitting metodoa modu errazean aplika daiteke, H Hamiltondarra, $H = H_1 + H_2$ eran bana daitekeenean eta Hamiltondar bakoitzari dagokion sistema modu esplizituan integratu daitekeenean.

Splitting arruntak.

H_1 eta H_2 sistemekin fluxu zehatzak $\varphi_t^{[H_2]}$ eta $\varphi_t^{[H_1]}$ izendatzen baditugu, honako splitting metodoak definituko ditugu:

1. Lie-Trotter splitting $p = 1$ ordenako metodoak,

$$\phi_h = \varphi_h^{[H_2]} \circ \varphi_h^{[H_1]} \quad \text{edo} \quad \phi_h^* = \varphi_h^{[H_1]} \circ \varphi_h^{[H_2]}. \quad (2.23)$$

Adibidea. Era honetako Hamiltondar banagarriko $H(p, q) = T(p) + V(q)$, $H_1 = T$ eta $H_2 = V$ izanik, zati bakoitzari dagokion fluxua era honetan definitzen da,

$$\varphi_t^{[H_1]} = (p, q + t\nabla T(p)), \quad \varphi_t^{[H_2]} = (p - t\nabla V(q), q).$$

Adibide honen Splitting metodoa partikularra, *Euler metodo simplektiko* ize-narekin ezaguna da,

$$\begin{aligned} p_{n+1} &= p_n - h\nabla V(q_{n+1}), \\ q_{n+1} &= q_n + h\nabla T(p_n). \end{aligned}$$

2. Strang-Marchuk splitting $p = 2$ ordenako metodo simetrikoak,

$$\phi_h = \varphi_{h/2}^{[H_1]} \circ \varphi_h^{[H_2]} \circ \varphi_{h/2}^{[H_1]}. \quad (2.24)$$

Adibidea. Hamiltondar banagarrietarako $H(p, q) = T(p) + V(q)$, Störmer-Verlet metodo ezaguna lortzen da,

$$\begin{aligned} q_{n+1/2} &= q_n + \frac{h}{2} \nabla T(p_n), \\ p_{n+1} &= p_n - h \nabla V(q_{n+1/2}), \\ q_{n+1} &= q_{n+1/2} + \frac{h}{2} \nabla T(p_{n+1}). \end{aligned}$$

Fluxu zehatza eta zenbakizko fluxua konbinatzu.

Demagun, sistemaren bi fluxu zehatzetariko bat $\varphi_t^{[H_1]}$ edo $\varphi_t^{[H_2]}$ ezin daitekeela kalkulatu. Kasu honetan ere, splitting teknika metodo simplektikoak eraikitzeko aplika daiteke. Adibidez, $\varphi_t^{[H_2]}$ ezezaguna bada eta Lie-Trotter teknika aplikatuz,

$$\phi_h = \varphi_h^{[H_1]} \circ \varphi_h^{[H_2]}, \quad \phi_h^* = \varphi_h^{[H_2]} \circ \varphi_h^{[H_1]},$$

eta lortzen den zenbakizko metodoak, $\phi_t^{[H_2]}$ zenbakizko metodoaren propietateek mantentzen ditu.

Splitting orokorrak.

Aurreko splitting metodoen (2.23) orokorpena modu honetan zehaztuko dugu,

$$\phi_h = \varphi_{\beta_s h}^{[H_2]} \circ \varphi_{\alpha_s h}^{[H_1]} \circ \varphi_{\beta_{s-1} h}^{[H_2]} \circ \cdots \circ \varphi_{\beta_1 h}^{[H_2]} \circ \varphi_{\alpha_1 h}^{[H_1]}. \quad (2.25)$$

non β_i, α_i koefizienteak ($\sum \beta_i = 1, \sum \alpha_i = 1$) metodoaren ordena definitzen duten.

Algoritmoa.

Splitting metodoen implementazio orokorra, 7 algoritmoan ikus daiteke:

```

for  $n \leftarrow 0$  to (endstep-1) do
     $Y_{0,n} = y_{n-1};$ 
    for  $i=1,2,\dots,s$  do
         $| Y_{i,n} = (\varphi_{\beta_i h}^{[H_2]} \circ \varphi_{\alpha_i h}^{[H_1]})(Y_{i-1,n});$ 
    end
     $y_{n+1} = Y_{s,n};$ 
end
```

Algoritmoa 7: Splitting metodoak.

Splitting metodoen implementazioarentzat, konposizio metodoen algoritmoei buruz aipatutako ezaugarri berdinak errepikatu beharko genituzke (esplizituak dira, sekuentzialki exekutatzen dira, memoria gutxi, ...).

Eguzki-sistemari egokitutako splitting metodoak.

Demagun, N-gorputzeko problema grabitazionalaren Hamiltondarra,

$$H(p, q) = T(p) + U(q).$$

Eguzki-sistemaren integratorako erabiltzen diren koordenatu sistema nagusiak, *Jacobi* eta koordenatu heliozentrikoak dira [101, 32]. Bi koordenatu sistema hauekin, Hamiltondarra beste modu honetan berridatzi daiteke,

$$H = H_A + \epsilon H_B, \quad |H_B| \ll |H_A|,$$

non alde nagusia H_A planeta bakoitzaren eguzkiaren inguruko mugimendu Keplériarra den eta H_B aldiz, planeten arteko interakzioek eragiten duten perturbazio txikia. Jacobi koordenatuetan H_B -k ez dauka p -ren menpekotasunik, heliozentrikoetan, ordea, bi aldagaien menpekotasuna dauka (ikus ?? eranskinean):

$$\begin{aligned} H_{Jab} &= H_A(p, q) + H_B(q), \\ H_{Hel} &= H_A(p, q) + H_B(p, q), \end{aligned}$$

$H_A(p, q)$ (mugimendu Keplériarrari dagokion zatia), Kepler-en fluxua kalkulatzen duen implementazio bat aplikatuz (3.4. atala) zehazki kalkulatuko da.

Eguzki-sistemaren problema grabitazionalari egokitutako zenbakizko bi integratzaile simplektiko azalduko ditugu. Lehena, Laskar-ek eta Robutel-ek [75] definitutako *SABAC*₄ integratzailea eta bigarrena, Blanes-ek [16, 32] definitutako *ABAH1064* integratzailea.

*SABAC*₄ integratzailea.

Laskarrek [75, 2001], *CSABA*_n eta *CSBAB*_n integratzaile simplektikoak proposatu zituen. Metodo hauek koeficiente positiboekin eraikitako metodo simplektikoak dira eta $\mathcal{O}(h^k\epsilon + h^4\epsilon^2)$ ordenakoak dira, n bikoitia denean $k = n + 2$ izanik eta n bakoitia denean $k = n + 3$ izanik.

Eguzki-sistemaren epe luzeko integrazioan [73] erabilitako *CSABA*₄ integratzailea deskribatuko dugu. Hamiltondarra $H = H_A + \epsilon H_B$ bada, era honetan definituko dugu metodoa,

$$\begin{aligned} SABA_4 &= \varphi_{c_1h}^{[A]} \circ \varphi_{d_1h}^{[B]} \circ \varphi_{c_2h}^{[A]} \circ \varphi_{d_2h}^{[B]} \circ \varphi_{c_3h}^{[A]} \circ \varphi_{d_2h}^{[B]} \circ \varphi_{c_2h}^{[A]} \circ \varphi_{d_1h}^{[B]} \circ \varphi_{c_1h}^{[A]}, \\ CSABA_4 &= \varphi_{-c/2}^{[B]} \circ SABA_4 \circ \varphi_{-c/2}^{[B]}, \end{aligned}$$

eta koefizienteak 2.2. taulan zehaztu ditugu.

2.2. Taula: $CSABA_4$ splitting metodoa [75].

Koefiziente	Balioa	Koefiziente	Balioa
c_1	$\frac{1}{2} - \frac{\sqrt{525+70\sqrt{30}}}{70}$	d_1	$\frac{1}{4} - \frac{\sqrt{30}}{72}$
c_2	$\frac{(\sqrt{525+70\sqrt{30}} - \sqrt{525-70\sqrt{30}})}{70}$	d_2	$\frac{1}{4} + \frac{\sqrt{30}}{72}$
c_3	$\frac{\sqrt{525-70\sqrt{30}}}{35}$		
c	0.00339677504820860133153215778349		

2.3. Taula: $ABAH1064$ splitting metodoa [16].

Koefiziente	Balioa	Koefiziente	Balioa
$a_1 = a_9$	0.04731908697653382270404371796320	$b_1 = b_9$	0.11968846245853220353128642974898
$a_2 = a_8$	0.26511052357487851595394800361856	$b_2 = b_8$	0.37529558553793742504201285376875
$a_3 = a_7$	-0.0099765228838112408432674681648	$b_3 = b_7$	-0.4684593418325993783650820409805
$a_4 = a_6$	-0.0599291997349415512639524798772	$b_4 = b_6$	0.33513973427558970103930989429495
a_5	0.25747611206734045344922822646033	b_5	0.27667111912108009750494572633568

ABAH1064 integratzaila.

Blanes-ek (2013), ABA eta $ABAH$ metodo simplektikoak proposatu zituen [16, 32]. ABA metodoak, Jacobi koordenatuetara egokitutako integratzailak eta $ABAH$ integratzailak, koordenatu heliozentrikoetara egokitutako integratzailak.

Atal honetan, koordenatu heliozentrikoei egokitutako $ABAH1064$, $p = 10$ ordenako metodoa deskribatuko dugu. Eguzki-sistemaren integraziorako koordenatu heliozentrikoei dagokion Hamiltondarra era honetakoa dugu,

$$H_{Hel}(p, q) = H_K(p, q) + H_I(p, q), \quad H_I(p, q) = T_1(p) + U_1(q).$$

$H_I(p, q)$ fluxua zehazki kalkulatu ordez honen hurbilpen bat erabiliko dugu,

$$\varphi_t^I \approx \tilde{\varphi}_t^I = \varphi_{tb_i/2}^{[U_1]} \circ \varphi_{tb_i}^{[T_1]} \circ \varphi_{tb_i/2}^{[U_1]}.$$

$ABAH1064$, $p = 10$ eta $s = 9$ splitting metodoa definituko dugu,

$$ABAH1064 = \prod_{i=1}^s \varphi_{a_i h}^K \circ (\varphi_{hb_i/2}^{[U_1]} \circ \varphi_{hb_i}^{[T_1]} \circ \varphi_{hb_i/2}^{[U_1]})$$

non a_i , b_i koefizienteak, 2.3. taulan definitzen diren.

2.5. Laburpena.

Atal honetan, metodo simplektikoen ikuspegi orokorra eman dugu. Hurrengo 2.4. taulan, ordena altuko lau integratzaile simplektikoen ezaugarriak laburtu ditugu. Izaera implizituen implementazioen (1 algoritmoa) eta izaera esplizituen (6, 7 algoritmoak) implementazioen konplexutasuna erakutsi dugu.

2.4. Taula: Integrazio metodo simplektikoen laburpena

	<i>C1035</i>	<i>CSABA</i> ₄	<i>ABAH1064</i>	<i>GAUSS – 12</i>
Hamiltondarra	Banagarria	Banagarria	Perturbatua	Orokorra
Mota	Esplizitua	Esplizitua	Esplizitua	Implizitua
Koordenatuak	Orokorra	Helio/Jacobi	Heliozentrikoa	Orokorra
Ordena	10		10	12
Atalak	35	9	9	6
Paralelizagarri	Ez	Ez	Ez	Bai

Metodo simplektikoei buruzko liburu monografiko hauek gomendatuko ditugu: [60],[48],[76] eta [33]. Eguzki-sistemaren epe luzeko simulazioei buruzko lan hauek ere azpimarratuko ditugu: [20],[64],[82],[58] eta [63].

3. Kapitulua

Problemak.

3.1. Sarrera.

Gure helburua eguzki-sistemaren simulaziorako zenbakizko integratzailak hobetzea eta hobekuntza horiek barneratzen dituen implementazioa eskaintza da. Eguzki-sistematztat nahi bezain komplexua den sistema har daiteke: planeta guztiek, planeta nanoak, planeten ilargiak, kometak, erlatibitate efektua eta abar har daitezke kontutan. Baino sistema simplifikatuan hobekuntzak egiten badira, sistema konplexuak ere hobeto integratzeko bidea irekiko da. Hori dela-eta, guk eredu simplekin jardun dugu. Esate baterako, eguzki-sistemaren bi eredu simple konsideratu: kanpo-planeten problema (eguzkia, kanpo-planetak eta Plutonek osatutakoa), eta bigarren eredua, *9-planeten problema* problema (eguzkia, 8 planetak eta Plutonek osatutakoa). Bi eredu hauetan, gorputzak masa puntuak dira eta gorputz hauen arteko erakarpen gravitazionalak bakarrik hartu ditugu kontutan.

Aipatu beharra dago, zenbakizko metodo simplektiko nagusienak esplizituak direla eta metodo horiek, Hamiltondar banagarria duten problemetan bakarrik erabil daitezkeela. Gainera, problema zurruna bada metodo esplizituak ez dira eraginkorrak eta metodo implizituek abantaila azaltzen dute. Gauss metodoa, orokorra eta implizitua izanik, problema zurrunetarako eta Hamiltondar banagarria ez den problemetarako aplikagarria dela ere erakutsi nahi dugu.

Hori dela-eta, problema osagarri gisa aukeratu dugu pendulu bikoitzaren problema, zenbakizko esperimentuak modu aberatsago eta zabalago batean egiteko. Pendulu bikoitzaren bi bertsio konsideratu ditugu: pendulu bikoitz arrunta eta pendulu bikoitz zurruna.

N-gorputzeko problema gravitazionalaren Hamiltondarra banagarria da baina pendulu bikoitzarena aldiz, ez da banagarria. Bestalde, pendulu bikoitzari malguki bat gehituz problema zurruna bilakatuko dugu, problema hauen zaitasunei nola aurre egin erakusteko. Gainera, eguzki-sistema kaotiko [71] konsideratzen dela

jakinik, pendulu bikoitz arruntak izaera kaotikoa azaltzen duen hasierako balio zehatzak aukeratu ditugu. Problema kaotikoak, hasierako balio edo parametroen perturbazioekiko, trunkatze edo birbitze erroreekiko esponentzialki sentikorrik dira.

Atal honetan, tesiaren zenbakizko esperimentuetan erabili ditugun problemak deskribatu ditugu, problema bakoitzari dagokion Hamiltondarra eta hasierako balioak zehaztuz. Lehenik, pendulu bikoitzaren problema eta N-gorputzen problema orokorra azaldu ditugu. Bigarrenik, eguzki-sistemaren problema grabitazionalean murgildu gara eta eredu ezberdinena zehaztapenak eman ditugu: Kepler problema, kanpo-planeten problema eta 9-planeten problema.

3.2. Pendulu bikoitza.

Pendulu bikoitzaren bi bertsio deskribatuko dugu: lehena, pendulu bikoitz arrunta eta bigarren problema konplexuagoa, pendulu bikoitz zurruna.

Pendulu bikoitz arrunta.

Pendulu bikoitzaren problema (ikus 3.1. irudia), planoan mugitzen diren eta elkarri lotuta dauden bi penduluk osatzen dute: penduluen masak m_1 eta m_2 dira, penduluetako bat puntu finko batetik zintzilik dago l_1 luzera duen eta okertzen ez den masarik gabeko lotura baten bidez. Beste pendulua, m_1 masa duen penduluaren pisuari lotuta dago, lotura honen luzera l_2 da eta hau ere, masarik gabekoa okertzen ez dena.

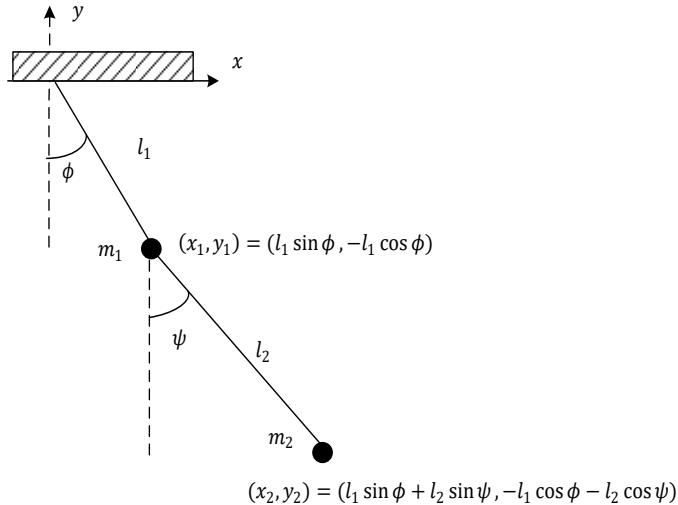
Sistemaren egoera aldagaiak, bi angelu $q = (\phi, \theta)$ eta dagozkion momentuak $p = (p_\phi, p_\theta)$ dira. ϕ lehen penduluaren ardatz bertikalarekiko angelua da eta bigarren penduluaren angelua, era honetan definituko dugu $\psi = \phi + \theta$.

Hamiltondar funtzioa $H(q, p)$ honakoa da,

$$-\frac{l_1^2 (m_1 + m_2) p_\theta^2 + l_2^2 m_2 (p_\theta - p_\phi)^2 + 2 l_1 l_2 m_2 p_\theta (p_\theta - p_\phi) \cos(\theta)}{l_1^2 l_2^2 m_2 (-2 m_1 - m_2 + m_2 \cos(2\theta))} - g \cos(\phi) (l_1 (m_1 + m_2) + l_2 m_2 \cos(\theta)) + g l_2 m_2 \sin(\theta) \sin(\phi), \quad (3.1)$$

Sistemaren parametroak. Gure esperimentuetarako honako parametroak kontsideratuko ditugu,

$$g = 9.8 \text{ m/s}^2, \quad l_1 = 1.0 \text{ m}, \quad l_2 = 1.0 \text{ m}, \quad m_1 = 1.0 \text{ kg}, \quad m_2 = 1.0 \text{ kg}.$$



3.1. Irudia: Pendulu bikoitz arrunta.

Hasierako balioak. Bi hasierako balio ezberdin konsideratu ditugu [29]: lehenak, izaera ez-kaotikoa du eta bigarrenak, izaera kaotikoa duen mugimendua agertzen du.

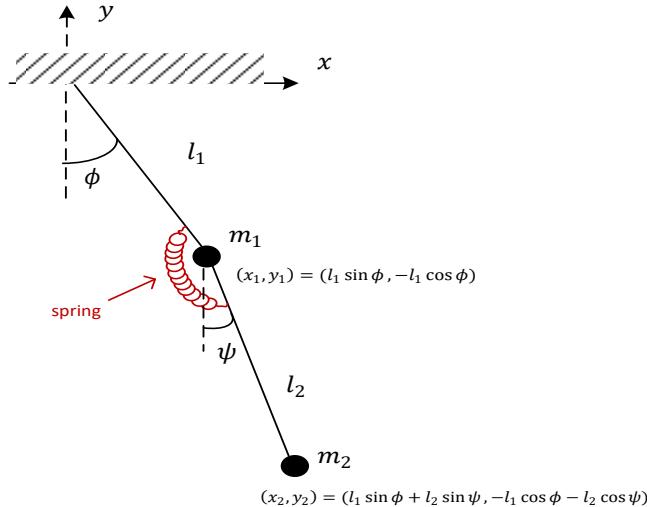
1. Hasierako balio ez-kaotikoak (NCDP): $q(0) = (1.1, -1.1)$ eta $p(0) = (2.7746, 2.7746)$. $T_{end} = 2^{12}$ segundoko integrazioa egin dugu.
2. Hasierako balio kaotikoak (CDP): $q(0) = (0, 0)$ eta $p(0) = (3.873, 3.873)$. $T_{end} = 2^8$ segundoko integrazioa egin dugu.

Urrats luzera, $h = 2^{-7}$ aukeratu dugu, trunkatze errorea biribiltze errorea baino txikiagoa izan dadin.

Pendulu bikoitz zurruna.

Pendulu bikoitz arruntari, malguki bat gehitutako sistema da (ikus 3.3. irudia) : m_1 eta m_2 masadun bi penduluk eta hauen artean k parametroaren araberako malgutasuna duen malgukiak osatzen duten sistema mekanikoa. $k = 0$ balioarentzat, problema ez da zurruna (hau da, pendulu bikoitz arrunta) eta problemaren zurruntasuna, k balioarekin batera handitzen da.

Sistemaren egoera aldagaiak, bi angelu $q = (\phi, \theta)$ eta dagozkion momentuak $p = (p_\phi, p_\theta)$ dira. ϕ lehen penduluak ardatz bertikalarekiko duen angelua da eta bigarren penduluaren angelua, era honetan definituko dugu $\psi = \phi + \theta$.



3.2. Irudia: Pendulu bikoitza (zurruna).

Hamiltondar funtzioa. Formulazio Lagrangiarrean ($L = T - V$), energia potentzialari $1/2 k \theta^2$ gaia gehituz, dagokion $H(q, p)$ funtzioko Hamiltondarra lortuko dugu,

$$\begin{aligned}
 & - \frac{l_1^2 (m_1 + m_2) p_\theta^2 + l_2^2 m_2 (p_\theta - p_\phi)^2 + 2 l_1 l_2 m_2 p_\theta (p_\theta - p_\phi) \cos(\theta)}{l_1^2 l_2^2 m_2 (-2 m_1 - m_2 + m_2 \cos(2\theta))} \\
 & - g \cos(\phi) (l_1 (m_1 + m_2) + l_2 m_2 \cos(\theta)) + g l_2 m_2 \sin(\theta) \sin(\phi) + \frac{k}{2} \theta^2,
 \end{aligned} \tag{3.2}$$

Sistemaren parametroak. Honako parametroak konsideratu ditugu,

$$g = 9.8 \text{ m/s}^2, \quad l_1 = 1.0 \text{ m}, \quad l_2 = 1.0 \text{ m}, \quad m_1 = 1.0 \text{ kg}, \quad m_2 = 1.0 \text{ kg},$$

eta k malgutasun parametroaren balio batzuk finkatu ditugu, zurruntasun maila ezberdineko pendulu bikoitzaren dinamikak aztertzeko

$$k = 2^{2i}, \quad i = 0, \dots, 11.$$

Hasierako balioak. Hasierako balioak, era honetan aukeratu ditugu:

1. $k = 0$ problemarako, [29] artikulutik izaera ez-kaotikoa duen hasierako balioak hartu ditugu: $q(0) = (1.1, -1.1)$ and $p(0) = (2.7746, 2.7746)$.

2. $k \neq 0$ problemetarako hasierako balioak,

$$q(0) = \left(1.1, \frac{-1.1}{\sqrt{1+100k}} \right), \quad p(0) = (2.7746, 2.7746),$$

aukeratu ditugu, non sistemaren energia $k \rightarrow \infty$ doanean bornatua dagoen.

$k = 0$ problema ez-zurrunerako, $h = 2^{-7}$ urrats luzera, trunkatze errorea birlitzte errorea baino txikiagoa izan dadin finkatu dugu eta gainontzeko integrazio guztiarako urrats luzera berdina erabili dugu. $k > 0$ zurruntasun balio batetik aurrera, trunkatze errorea birlitzte errorea baino garrantzitsuago da. $T_{end} = 2^{12}$ segundoko integrazioa egin dugu.

3.3. N-Gorputzen problema.

Problemaren formulazioa eman aurretik, K.Tanikawa-k eta T.Ito-k [58] 3-gorputzen problemari buruzko deskribapena aipatuko nahi genuke,

Never be attracted to the three-body problem. It is too dangerous.
The three-body problem has long been an attractive but dangerous subject for students. This is because it has quite a simple setting and it appears relatively easy. However, it has been investigated for so many years that it is very difficult to obtain anything new.

Newtonen lege grabitazionalen araberako N-gorputzen problemaren ekuazio diferentzialak era honetan definitzen dira,

$$m_i \ddot{q}_i = G \sum_{j=0, j \neq i}^N \frac{m_i m_j}{\|q_j - q_i\|^3} (q_j - q_i), \quad i = 0, 1, \dots, N, \quad (3.3)$$

non $(N + 1)$ gorputz kopurua den, eta $q_i \in \mathbb{R}^3$, $m_i \in \mathbb{R}$, $i = 0, \dots, N$ gorputz bakoitzaren kokapena eta masa den.

Hamiltondar sistema. Momentuen definizio hau ordezkatzuz $p_i = m_i * \dot{q}_i$, N-gorputzeko problemaren formulazio Hamiltonarra lortzen da,

$$H(q, p) = \frac{1}{2} \sum_{i=0}^N \frac{\|p_i\|^2}{m_i} - G \sum_{0 \leq i < j \leq N} \frac{m_i m_j}{\|q_i - q_j\|}. \quad (3.4)$$

Ekuazio differentzialak. Abiaduraren eta kokapenaren araberako ekuazioak hauek dira,

$$\begin{aligned}\dot{q}_i &= v_i, \quad i = 0, 1, \dots, N, \\ \dot{v}_i &= \sum_{j=0, j \neq i}^N \frac{Gm_j}{\|q_j - q_i\|^3} (q_j - q_i), \quad i = 0, 1, \dots, N\end{aligned}\tag{3.5}$$

Problemaren integralak. Integrazioan zehar konstante mantentzen diren kantitateei problemaren integralak edo inbarianteak deitzen zaie. N-gorputzen problemak 10 integral ditu [65]:

1. Masa zentroaren sei integralak.

Era horretan definitzen dugun P , konstantea dela modu errazean frogatzen daiteke,

$$P = \sum_{i=0}^N m_i \dot{q}_i = \sum_{i=0}^N p_i.$$

Eta ondorioz,

$$O = \sum_{i=0}^N m_i q_i = Pt + B.$$

$P, B \in \mathbb{R}^3$ bektoreen osagaiei, masa zentroaren sei integralak esaten zaie. Masa zentroaren kokapena (Q) eta abiadura (V) era horretan definitzen dira,

$$Q = \frac{\left(\sum_{i=0}^N m_i q_i \right)}{M}, \quad V = \frac{\left(\sum_{i=0}^N m_i \dot{q}_i \right)}{M}$$

non $M = \sum_{i=0}^N Gm_i$ den.

2. Momentu angeluarra.

Momentu angeluarra $L \in \mathbb{R}^3$, problemaren beste hiru integralak dira,

$$L = \sum_{i=0}^N p_i \times q_i = \sum_{i=0}^N q_i \times m_i \dot{q}_i.$$

3. Energia.

Hamitoniar sistema osoaren energia da eta problemaren beste integrala da,

$$E = H(q, p).$$

Problemaren 10 integral hauek, sistemaren ordena gutxitzeko edo zenbakizko integrazioaren doitasuna neurtzeko erabil daitezke. Guk koordenatu barizentrikoak (koordenatu sistemaren jatorria masa zentroaren kokapena) erabiliko ditugu eta koordenatu hauetan, $P = 0$ eta $B = 0$ dira. Beraz, integratzeko erabiliko ditugun hasierako balioak (\hat{q}_i, \hat{v}_i) modu honetan finkatuko ditugu,

$$\begin{aligned}\hat{q}_i &= q_i - Q, \\ \hat{v}_i &= v_i - V, \quad i = 0, \dots, N,\end{aligned}$$

eta era honetan, masa zentroaren kokapen eta abiadurak,

$$\hat{Q} = \frac{\left(\sum_{i=0}^N m_i \hat{q}_i\right)}{M} = 0, \quad \hat{V} = \frac{\left(\sum_{i=0}^N m_i \hat{v}_i\right)}{M} = 0$$

izan daitezen,

Momentu angeluarra eta energia, zenbakizko integrazioaren doitasuna neurtzeko erabili ohi dira. Energia zenbakizko integrazioen biribiltze errorea neurtzeko integral egokiena da.

3.4. Eguzki-sistema.

Sarrera.

Eguzki-sistemaren planeten orbiten mugimenduaren eredu matematikoa, sistema Hamiltondar bati dagokion ekuazio diferentzial arrunten bidez formulatzen da. N planeta badaude, $6N$ ekuazio diferenzialeko sistema izango da.

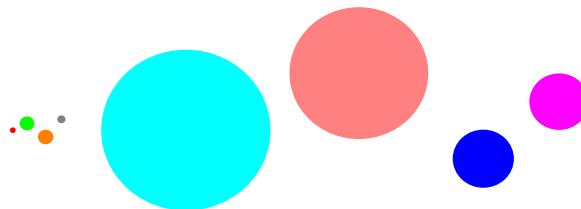
Eguzki sistemaren eredu simplea integratuko dugu. Eguzki-sistemaren gorputzak masa puntualak konsideratuko ditugu eta gure ekuazio diferenzialak definitzeko, soilik gorputz hauen arteko erakarpen grabitazionalak hartu ditugu kontutan.

Eguzki-sistemaren gorputz nagusien (eguzkia eta planetak) integrazioetara mugatuko gara. Ereduen gorputz kopurua txikia izango da, kanpo-planeten probleman $N = 6$ eta 9-planeten probleman $N = 10$ izango da.

Eguzkiak, planetak baino 1.000 aldiz masa handiago du eta hauxe da, eguzki-sistemaren ezaugarri garrantzitsuenetakoak: eguzkiaren grabitazio indar nagusiak eta planeten arteko perturbazio txikiak sortzen duten sistema dinamikoa da. Planeten eta beren sateliteen mugimenduan kolisiotik gertuko egoerarik edo eszentrikotasun handiko orbitalik ez dago. Beraz, ikuspuntu honetatik sistema dinamiko simplea dela, esan daiteke. Eguzki-sistema egonkorra konsideratzen da, hau da, epe luze batean planeten arteko talkarik edo planeten kanporatzerik gertatzea, ez da espero [71, 50].

Eguzki-sistemaren jatorria, orain 5×10^9 urtetan finkatzen da eta beraz, eguzki-sistemaren eboluzioa integratzeko, urrats kopuru oso handia beharrezkoa da. Eguzki-sistemaren eredu osoaren (9-planeten problema) integrazioetako ohiko urratsa $h = 0.0025$ urtekoa bada (orbita txikierekiko periodoaren %1), eman beharreko urrats kopurua 2×10^{12} izango da. Era berean, eguzki-sistemaren eredu simpleagoaren (kanpo-planeten problema) integrazioetako urrats tamaina handiago bada ere, urrats kopurua 5×10^{10} ingurukoa da.

Eguzki-sistemaren problemaren eskalak, anitzak dira. (??) irudian, planeta nagusien tamainak irudikatu ditugu eta (3.1.) taulan, eguzki-sistemaren planeten ezaugarri nagusienak eman ditugu. Aniztasunaren adierazgarri, ilargiaren, lurrauen eta Neptunoren orbitak konpara ditzakegu: ilargiak 27.32 eguneko periodoa duen orbita dauka, lurrauk urtebetekoa eta Neptunok 163 urtekoa.



3.3. Irudia: Irudian, planeten arteko tamainen proportzioak irudikatu ditugu. Ezkerretik eskuinera: Merkurio, Artizarra, Lurra, Marte, Jupiter, Saturno, Urano eta Neptuno

3.1. Taula: Eguzki-sistemaren planeta nagusien masak, eguzkiarekiko distantziak, orbitaren periodoa eta eszentrikotasuna.

Planeta	Masak kg	Distantzia AU	Periodoa urteak	Eszentrikotasuna
Eguzkia	1.99×10^{30}			
Merkurio	3.30×10^{23}	0.39	0.24	0.205
Artizarra	4.87×10^{24}	0.72	0.007	0.007
Lurra	5.97×10^{24}	1.00	1.007	0.017
Marte	6.42×10^{23}	1.52	1.88	0.094
Jupiter	1.90×10^{27}	5.20	11.86	0.049
Saturno	5.68×10^{26}	9.54	29.42	0.057
Urano	8.68×10^{25}	19.19	83.75	0.046
Neptuno	1.02×10^{26}	30.06	163.72	0.011
Pluton	1.31×10^{22}	39.53	248.02	0.244

Hiru dira erabiltzen diren koordenatu sistema nagusienak:

1. Koordenatu cartesiarrak.
2. Koordenatu heliozentrikoak.
3. Koordenatu jacobiarra.

Ohikoa da ekuazio diferentzialak koordenatu heliozentrikoen (eguzkiaren zentroarekiko) arabera definitzea. Ekuazioen garapen osoa eranskinean eman dugu.

Problemak.

Eguzki-sistemaren simulaziorako test problemak deskribatuko ditugu. Bi gorputzen problematik abiatuta, gero eta problema konplexuagoak azalduko ditugu. PW, Sharp-ek [95] eguzki-sistemaren problemen bilduma interesgarria egin zuen, eta bertan problema hauek guztiak problema ez-zurrunak konsideratzen dituela nabarmentzekoa da.

Kepler problema.

Kepler problema, bi gorputzen problemaren kasu partikularra da eta honako Hamiltondarra dagokio,

$$H(p, q) = \frac{p^2}{2m} - \frac{\mu}{\|q\|}, \quad (3.6)$$

non m eta μ konstanteen balioak, formulazioaren araberakoak diren.

Koordenatu sistema $q = q_2 - q_1$ duen formulazioa aukeratzen badugu, konstanteen balioak hauetan dira,

$$m = (1/m_1 + 1/m_2)^{-1}, \quad \mu = Gm_1m_2,$$

eta ekuazio diferentzialak era honetan definitzen dira,

$$\dot{q} = p, \quad \dot{p} = -\frac{k q}{\|q\|^3}, \quad (3.7)$$

non $k = \mu/m$ eta $q, p \in \mathbb{R}^3$.

Kepler problemaren soluzio zehatza kalkula daiteke: une bateko kokapen eta abiadurak emanik, denbora tarte bat (Δt) igarotakoan (positiboa ala negatiboa), zehazki kokapen eta abiadura berriak ezagutu daitezke. Eguzki-sistemaren integrazio metodoentzat, Kepler problema doitasun handian eta era eraginkorrean konputatzea, funtsezkoa da. Kepler problemaren erreferentziazko implementazioak, Danby [27] eta J.Wisdom [102] ditugu.

Kepler problemaren implementazioaren lan egiteko modua, honakoa da. Lehenik, koordenatu cartesiarretatik ($q, p \in \mathbb{R}^3$), koordenatu eliptikoetara (a, e, i, Ω, E),

itzulpena egingo dugu. Koordenatu eliptikoetan, E (*eccentric anomaly*) aldagaia izan ezik, beste aldagaiaik konstante mantentzen dira: beraz E_0 balioa emanda, Δt denbora tartea aurrera egin eta E_1 balioa berria kalkulatuko dugu. Azkenik, koordenatu eliptikoetatik koordenatu cartesiarretara itzulpena eginez, kokapen eta abiadura berriak eskuratuko ditugu.

$$\begin{aligned} (q_0, v_0) \in \mathbb{R}^6 &\longrightarrow (a, e, i, \Omega, E_0) \in \mathbb{R}^6 \\ &\downarrow \Delta t \\ (q_1, v_1) \in \mathbb{R}^6 &\longleftarrow (a, e, i, \Omega, E_1) \in \mathbb{R}^6 \end{aligned}$$

Implementazioaren garapenaren zehaztasun guztiak eranskinaren [B.1](#). atalean eman ditugu.

Kanpo-planeten problema.

Kanpo-planeten problemaren ereduan, eguzkia, lau planeta nagusiak (Jupiter, Saturno, Urano, Neptuno) eta Pluton konsideratuko ditugu. Eguzki-sistemaren kanpo-planeten mugimenduaren azterketa interesgarria da. Lehenik, planeta nagusi hauen eboluzioa eguzki-sistema osoaren zati garrantzitsuena da eta barne-planeten mugimendua kontutan hartzeak ala ez, kanpo-planeten zenbakizko integracioarengan oso eragin txikia du. Bigarrenik, urrats luzera handia erabili daiteke eta beraz, epe luzeko integracioak errazten dira (konputazio denbora gutxiago behar delako). Hirugarrenik, Pluton orbitaren berezitasunak ikertzea, 1960 – 1980 urteetan interes handikoa izan zen.

Hasierako balioak [48] liburutik hartu ditugu. Planetei dagokien masak [3.2.](#) taulan eta kokapenak/abiadurak [3.3.](#) taulan laburtu ditugu. planeten masak eguzkiarekiko erlatiboak dira, hau da, eguzkiaren masa 1 da eta gravitazio konstantea $G = 2.95912208286 \cdot 10^{-4}$. Barne-planeten masak eguzkiaren masari gehitu zaio eta horregatik, eguzkiaren masak, $m_0 = 1.00000597682$ balioa hartzen du.

3.2. Taula: Kanpo-planeten masak.

Gorputza	Masa
Eguzkia	1.000005976823
Jupiter	0.000954786104043
Saturno	0.000285583733151
Urano	0.0000437273164546
Neptuno	0.0000517759138449
Pluton	$1/(1.3 \cdot 10^8)$

3.3. Taula: Kanpo-planeten problemaren hasierako balioak, kokapenak (x, y, z) eta abiadurak (v_x, v_y, v_z).

Gorputza	Balioa			
Eguzkia	x, y, z	0.	0	0.
	v_x, v_y, v_z	0.	0.	0.
Jupiter	x, y, z	-3.5023653	-3.8169847	-1.5507963
	v_x, v_y, v_z	0.00565429	-0.00412490	-0.00190589
Saturno	x, y, z	9.0755314	-3.0458353	-1.6483708
	v_x, v_y, v_z	0.00168318	0.00483525	0.00192462
Urano	x, y, z	8.3101420	-16.2901086	-7.2521278
	v_x, v_y, v_z	0.00354178	0.00137102	0.00055029
Neptuno	x, y, z	11.4707666	-25.7294829	-10.8169456
	v_x, v_y, v_z	0.00288930	0.00114527	0.00039677
Pluton	x, y, z	-15.5387357	-25.2225594	-3.1902382
	v_x, v_y, v_z	0.00276725	-0.00170702	-0.00136504

9-planeten problema.

Eguzki-sistemaren 9-planeten zenbakizko integrazioak, kanpo-planeten problemak baino konplexutasun handiago du. Planeten eta eguzkiaren arteko interakzio kopurua 45 (kanpo-planeten probleman 15) da. Planeten orbiten periodoaren arteko aldea askoz handiago da. Merkurioren orbitaren eszentrikotasuna $e = 0.206$ (Jupiterren orbitaren eszentrikotasuna $e = 0.048$) da.

Eredu honetan, lur-ilargi sistema (*EMB*) masa puntual bakarra konsideratzen da. Lur-ilargi sistemaren masa, bi gorputzen masen arteko batura da eta kokapena, lur-ilargi sistemaren barizentroan finkatzen da.

Hasierako balioak *DE-430* (2.014) [36] azken efemeride artikulutik hartu ditugu. Eguzki eta planeten hasierako kokapenak (AU) eta abiadurak (AU/egun), (3.5.) taulan laburtu ditugu. Era berean, planeta bakoitzari dagokion Gm balioa (3.4.) taulan laburtu dugu.

Laskar-en eredua.

Eguzki-sistemaren mugimenduaren azterketa zehatza egiteko, planeten eta ilargiaren orbiten mugimenduaren ekuazioak nahiz lur eta ilargiaren errotazio mugimenduaren ekuazioak integratu behar dira.

Laskar-ek, 2.011. urteko epe luzeko zenbakizko integratorako [73] eguzki-sistemaren eredua deskribatuko dugu. Hasierako integratioetan, eguzkia, 8 plane-

3.4. Taula: Planeten GM balioak.

Gorputza	$GM (au^3/day^3)$
Eguzkia	$0.295912208285591100e - 03$
Merkurio	$0.491248045036476000e - 10$
Artizarra	$0.724345233264412000e - 09$
Lurra	$0.888769244512563400e - 09$
Marte	$0.954954869555077000e - 10$
Jupiter	$0.282534584083387000e - 06$
Saturno	$0.845970607324503000e - 07$
Urano	$0.129202482578296000e - 07$
Neptuno	$0.0152435734788511000e - 07$
Pluton	$0.217844105197418000e - 11$
Ilargia	$0.109318945074237400e - 10$

tak, Pluton eta ilargia bakarrik konsideratu zituen. Eguzkiaren erlatibilitate efektua (Saha eta Tremain-ek [91] finkatutako teknikaren arabera) eta eredu errealistaren indar ez grabitazional garrantzitsuenak aplikatu zituen. Azken integrazioetan, Zeres, Palas, Vesta, Iris eta Bamberga asteroideak gehitu zituen.

Ilargia gorputz independente gisa konsideratu zuen. Ilargiaren lurrireko distantzia (380.000 km), beste gorputzekiko distantziekin alderatzen badugu (eguzkiarekiko 150.000.000 km eta Artizarrarekiko 45.000.000 km) oso txikia da. Horri dela eta, ilargiaren kokapena, eguzki-sistemaren barizentroarekiko konsideratu ordez, lurrireko konsideratuz doitasun handiagoa lortuko da. Lurraren eguzkiarekiko kokapena (q_e) eta ilargiaren eguzkiarekiko kokapen (q_m), hurrenez-hurren, lur-ilargi sistemaren barizentroaren eguzkiarekiko kokapena (q_B) eta ilargiaren lurrireko kokapena (q_{em}) aldagaiekin ordezkatzen dira,

$$q_B = \frac{Gm_e q_e + Gm_m q_m}{Gm_e + Gm_m},$$

$$q_{em} = q_m - q_e.$$

Argitzea komeni da, ekuazio diferenzialaren eskubiko aldeko expresioa ebaluatzeako (q_e, q_m) aldagaiak erabiliko ditugula eta (q_B, q_{em}) aldagai berriak integratzeko.

3.5. Taula: Eguzki eta 9 planeten hasierako balioak, kokapenak (x, y, z) (AU) eta abiadurak (v_x, v_y, v_z) (AU/egun). Julian data (TDB) 2440400.5 (1969. ekainaren 28) eta ICRFR2 (International Celestial Reference Frame) koordenatu sistemaren emanak dira.

Gorputza	Balioa			
Eguzkia	x, y, z	0.00450250878464055477	0.00076707642709100705	0.00026605791776697764
	v_x, v_y, v_z	-0.00000035174953607552	0.00000517762640983341	0.00000222910217891203
Merkurio	x, y, z	0.36176271656028195477	-0.09078197215676599295	-0.08571497256275117236
	v_x, v_y, v_z	0.00336749397200575848	0.02489452055768343341	0.01294630040970409203
Artizarra	x, y, z	0.61275194083507215477	-0.34836536903362219295	-0.19527828667594382236
	v_x, v_y, v_z	0.01095206842352823448	0.01561768426786768341	0.00633110570297786403
EMB	x, y, z	0.12051741410138465477	-0.92583847476914859295	-0.4015402264531522236
	v_x, v_y, v_z	0.01681126830978379448	0.00174830923073434441	0.00075820289738312913
Marte	x, y, z	-0.11018607714879824523	-1.32759945030298299295	-0.60588914048429142236
	v_x, v_y, v_z	0.01448165305704756448	0.00024246307683646861	-0.00028152072792433877
Jupiter	x, y, z	-5.37970676855393644523	-0.83048132656339789295	-0.22482887442656542236
	v_x, v_y, v_z	0.00109201259423733748	-0.00651811661280738459	-0.00282078276229867897
Saturno	x, y, z	7.89439068290953155477	4.59647805517127300705	1.55869584283189997764
	v_x, v_y, v_z	-0.00321755651650091552	0.00433581034174662541	0.00192864631686015503
Urano	x, y, z	-18.26540225387235944523	-1.16195541867586999295	-0.25010605772133802236
	v_x, v_y, v_z	0.00022119039101561468	-0.00376247500810884459	-0.00165101502742994997
Neptuno	x, y, z	-16.05503578023336944523	-23.94219155985470899295	-9.40015796880239402236
	v_x, v_y, v_z	0.00264276984798005548	-0.00149831255054097759	-0.00067904196080291327
Pluton	x, y, z	-30.48331376718383944523	-0.87240555684104999295	8.91157617249954997764
	v_x, v_y, v_z	0.00032220737349778078	-0.00314357639364532859	-0.00107794975959731297

ko erabiliko ditugula.

```

Lurra, Ilargia = { $q_B, q_{em}$ };
for  $i \leftarrow 1$  to endstep do
    { $q_e, q_m$ }  $\leftarrow$  { $q_B, q_{em}$ };
    Ebaluatu  $\dot{y} = f(y)$ ;
    { $q_B, q_{em}$ }  $\leftarrow$  { $q_e, q_m$ };
    Integrazioa ( $q_B, q_{em}$ );
end

```

Algoritmoa 8: Ilargiaren kalkuluak.

3.5. Laburpena.

Atal honetan, pendulu bikoitzaren problema eta eguzki-sistema gravitazionalaren eredu ezberdinen zehaztasunak eman ditugu. Batetik, pendulu bikoitzaren problemaren hasierako balioak [29] artikulutik hartu ditugu. Bestetik, eguzki-sistema gravitazionalaren problemaren integratorako hasierako balioak lan hauetatik hartu ditugu: kanpo-planeten problemarentzat [48] liburutik eta 9-planeten problemarentzat 2014. urteko efemerideen [36] artikulutik hartu ditugu. Eguzki-sistemaren problemaren integratorako hasierako balioak jasotzen dituzten beste lan hauek

3.6. Taula

Planeta	Distantzia AU	Periodoa urte	GM ($au^3/egun^3$)	Ezentrizitatea
Eguzkia			$0.2959e - 03$	
Merkurio	0.39	0.24	$0.4912e - 10$	0.205
Artizarra	0.72	0.007	$0.7243e - 09$	0.007
Lurra	1.00	1.007	$0.8887e - 09$	0.017
Ilargia			$0.1093e - 10$	0.055
Marte	1.52	1.88	$0.9549e - 10$	0.094
Jupiter	5.20	11.86	$0.2825e - 06$	0.049
Saturno	9.54	29.42	$0.8459e - 07$	0.057
Urano	19.19	83.75	$0.1292e - 07$	0.046
Neptuno	30.06	163.72	$0.1524e - 07$	0.011
Zeres	2.77	4.6	$0.1400e - 12$	0.07
Palas	2.77	4.61	$0.3104e - 13$	0.23
Vesta	2.36	3.63	$0.3854e - 13$	0.08
Iris	2.38	3.68	$0.2136e - 14$	0.21
Bamberga	2.68	4.39	$0.1388e - 14$	0.34
Pluton	39.53	248.02	$0.2178e - 11$	0.244

3.7. Taula: Ilargiaren Lurrarekiko hasierako balioak.

Gorputza	Balioa			
Ilargia	x, y, z	-0.00080817735147818490	-0.00199462998549701300	-0.00108726268307068900
	v_x, v_y, v_z	0.00060108481561422370	-0.00016744546915764980	-0.00008556214140094871

ere aipatu nahi genitzke: P.W. Sharp-ek eguzki-sistemaren problemen bilduma [95] eta Laskar-en [74] artikuluaren informazio osagarria.

4. Kapitulua

Koma higikorreko aritmetika.

4.1. Sarrera.

Konputagailuetan, zenbaki errealak (\mathbb{R}) bit kopuru finituaren bidez adierazi behar dira eta honetarako, koma-higikorreko adierazpen sistema (\mathbb{F}) erabiltzen da. Zenbaki erreal batzuk, \mathbb{F} sistemaren adierazpen zehatza dute, baina beste batzuk hurbildu egin behar dira. Era berean, eragiketa aritmetikoak ($+, -, *, /$) kalkulu gehienetan ere, emaitzaren hurbilpena egin beha da. \mathbb{R} sistematik \mathbb{F} sistemara bihurtzeko funtzioari biribiltzea esaten zaio. Oro har, konputazio zientzian, biribiltze errore honen eragina garrantzitsua da eta errorea gutxitzeko ahalegin berezia beharrezkoa da.

Egungo konputagailuen koma-higikorreko aritmetikaren implementazioak, *IEEE-754* estandarrean [56] oinarritzen dira. *IEEE-754* estandarrak, koma-higikorreko aritmetikaren konputaziorako formatu eta metodoak definitzen ditu. Konputazioen fidagarritasuna eta aplikazioen portabilitatea bermatzen ditu.

Atal honetan, koma-higikorreko aritmetika eta biribiltze errorearen oinarria azalduko ditugu. Ondoren, konputazioetan biribiltze erroreak gutxitzeko teknika ezagun batzuk azalduko ditugu.

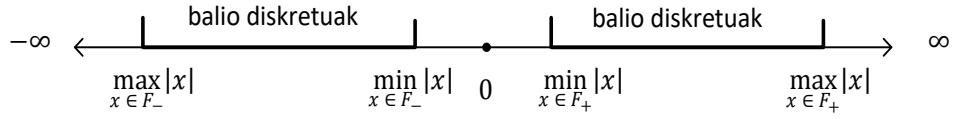
4.2. IEEE-754 estandarra.

Koma-higikorreko zenbaki multzoa finitua da eta \mathbb{F} izendatuko dugu. Koma-higikorreko adierazpen zehatza duten zenbaki errealei koma-higikorreko zenbakiak deritzogu,

$$\mathbb{F} \subset \mathbb{R}.$$

\mathbb{F} zenbaki multzoa, 4.1.irudian laburtu dugu. Bai zenbaki positiboentzat, bai negatiboentzat, adierazi daitekeen zenbaki handienaren eta txikienaren arteko ba-

lio bakanez osatuta dago. Multzoaren kanpoaldean zenbaki hauek guztiak ditugu: batetik overflow tartean $(-\infty, \max_{x \in \mathbb{F}_-} |x|)$ eta $(\max_{x \in \mathbb{F}_+} |x|, \infty)$ daudenak; bestetik underflow tartean $(\min_{x \in \mathbb{F}_-} |x|, 0)$ eta $(0, \min_{x \in \mathbb{F}_+} |x|)$ daudenak.



4.1. Irudia: Koma-higikorreko zenbakien multzoa.

IEEE-754 estandarraren arabera, n -biteko koma-higikorreko adierazpenak bi zati ditu (ikus 4.2. irudiko adibidea),

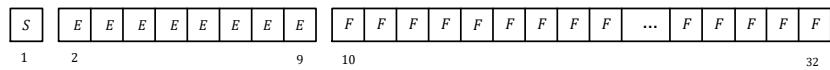
1. m bitez osatutako zatia, mantisa (M) izenekoa. Horietako bit batek (S) zeinuadierazten du. Bestalde M mantisa modu normalizatu honetan emana da, $\pm 1.F$ eta zati erreala (F) bakarrik gorde behar da.
2. Esponentea (E), $(n - m)$ bitez adierazitako zenbaki osoa. Zeinuarentzat ez da bit zehatzik, baizik *bias* izeneko balio bat gehituz adierazten dira zenbaki positiboak eta negatiboak.

Beraz, oinarri bitarrean koma-higikorreko zenbaki hauek adierazten dira,

$$M \times b^E, \quad b = 2,$$

eta biribiltze unitatea (*unit roundoff*) era honetan definituko dugu,

$$u = 2^{-m}.$$



4.2. Irudia: 32-biteko koma-higikorreko zenbakiaren adierazpena: esponentearentzat 8-bit eta mantisarentzat 24-bit (bit bat zeinuarentzat eta beste 23 bit, $1.F$ eran normalizatutako mantisarentzat) banatuta.

IEEE-754 estandarrean, oinarri bitarreko koma-higikorreko hiru formatu definitzen dira: bata doitasun arrunta (*single precision*), bestea doitasun bikoitza

4.1. Taula: IEEE-754 koma-higikorreko formatuak

Formatoa	Tamaina	Mantisa	Esponentea	Tartea	$u = 2^{-m}$
	n	m	n-m		
Arrunta	32 bit	24	8	$10^{\pm 38}$	6×10^{-8}
Bikoitza	64 bit	53	11	$10^{\pm 308}$	1×10^{-16}
Laukoitza	128 bit	113	15	$10^{\pm 11356}$	1×10^{-35}

(*double precision*) eta hirugarrena doitasun laukoitza (*quadruple precision*) ize-nekoak (4.1. Taula).

Doitasun bikoitzeko oinarrizko eragiketak (batuketa, kenketa, biderketa, zatiketa, eta erro karratua) hardware bidez exekutatzen dira [83] eta azkarrik dira. Makina ziklo bakoitzeko, 2 eta 4 batuketa, kenketa edo biderketa egin ohi dira; zatiketa eta erro karratua aldiz, eragiketa motelagoak dira. Bestalde, doitasun arruntaren aritmetika, doitasun bikoitza baino azkarragoa da: garaiatu behar den bit kopuru erdia delako eta gainera, hardware bereziei (adibidez *Intel* makinetan *SSE* moduluak) esker, eragiketa aritmetikoak azkarragoak direlako. 2008. urtean, IEEE-754 estandarrak, 128-biteko koma-higikorreko aritmetika onartu zuen, baina inplementazioa softwarez bidezkoa da eta exekuzioa gutxi gorabehera doitasun bikoitzeko aritmetika baino 10-15 aldiz motelagoa da.

Problema batzuk, doitasun bikoitza baino doitasun handiagoa behar dute [61]. Doitasun laukoitza edo altuagoa, software liburutegien bidez emulatzen ohi dira. Doitasun altuko zenbakiak adierazteko nagusiki bi modu bereizten dira:

1. *Digitu-anitzeko adierazpena*. Zenbakiak esponente bakarra eta mantisa bat baino gehiagorekin adierazten dira (adb. *GNU MPFR liburutegia* [37]).
2. *Termino-anitzeko adierazpena*. Zenbakiak ebaluatu gabeko hainbat koma-higikorreko makina zenbaki estandarren batura gisa adierazten dira (adb. *Bailey QD liburutegia*) [52] eta exekuzioaren ikuspegitik, hardware bidezko inplementazioaren abantaila dute.

Doitasun laukoitzeko gure esperimentuetarako, *GCC libquadmath liburutegia* [41] erabili dugu. Doitasun laukoitzean exekutatutako integrazioen zenbakizko soluzioak, soluzio zehatzak konsideratu ditugu eta doitasun bikoitzeko inplementazioaren errorea, soluzio zehatzaren diferentzia gisa kalkulatu dugu.

Laskar-ek epe luzeko eguzki-sistemaren simulazioaren (-250 eta $+250$ milioitako integrazio tartea) konputaziorako kalkuluak [73], kontu handiz eta doitasun handian egin behar ditu. Dena den, era honetako problemak salbuespenak dira eta ez da ohikoa izaten doitasun handian lan egin beharra. Egia da ere, neurri

fisiko oso gutxi ezagutzen direla hain doitasun handian (adibidez 50-bitekin, lurra eta ilargiaren arteko distantzia, milimetroko errorearekin adieraz daiteke).

4.3. Biribiltze errorea.

Zenbakizko integrazioen errorea, trunkatze eta biribiltze errorez osatuta dago. Urrats luzera adina txikia aukeratuz, trunkatze errorea biribiltze errorea baino txikiago izango da eta beraz, zenbakizko integracio hauetan errorean biribiltze errorea nagusitzen da. Epe luzeko eta doitasun handiko integrazioetan, urrats luzera txikia erabiltzen denez, biribiltze errorea gutxitzea funtsezkoa izango da.

Bi biribiltze mota bereiziko ditugu, bata adierazpenaren errorea eta bestea, aritmetikaren errorea.

Adierazpenaren errorea.

Zenbaki erreals batzuk, \mathbb{F} koma-higikorreko multzoan zehazki adieraz daitezke eta beste batzuk ordea, hurbilpen batez adierazi behar dira. $x \in \mathbb{R}$ izanik, $fl : \mathbb{R} \rightarrow \mathbb{F}$ koma-higikorreko zenbakia esleitzen dion funtzioari deituko diogu: $x \in \mathbb{R}$ balioaren gertuen dagoen $fl(x) \in \mathbb{F}$, itzultzen duen funtzioa bezala definitzen da. Hau da, $f_1, f_2 \in \mathbb{F}$ jarraian dauden koma-higikorreko zenbakiak badira eta $x \in \mathbb{R}$, $f_1 \leqslant x \leqslant f_2$ bada,

$$fl(x) = \begin{cases} f_1 & \text{if } |x - f_1| < |x - f_2| \\ f_2 & \text{if } |x - f_1| \geqslant |x - f_2| \end{cases}.$$

Jarraian, koma-higikorreko adierazpenaren errore absolutua eta errore erlatiboa finkatuko ditugu.

- Errore absolutua,

$$\Delta x = fl(x) - x = \tilde{x} - x.$$

- Errore erlatiboa,

$$\delta x = \frac{\Delta x}{x} = \frac{\tilde{x} - x}{x}.$$

- Aurreko bi definizioen ondorioz honako formula erabilgarria dugu,

$$\tilde{x} = x + \Delta x = x(1 + \delta x).$$

Koma-higikorreko zenbaki sistema bitarrean (m = mantisa adierazteko bit kopurua izanik) $|x|$ balioa, \mathbb{F} multzoaren zenbaki txikienaren eta handienaren artean badago,

$$|\delta x| < u \text{ non } u = 2^{-m},$$

bermatuta dagoela froga daiteke [26].

Aritmetikaren errorea.

Koma-higikorreko zenbakien arteko eragiketa baten emaitzak, ez du zertan \mathbb{F} multzoan adierazpen zehatza izan behar eta orduan, emaitza biribildu egingo da. Adibidez, m digituzko bi zenbakien biderketaren emaitza zehatza adierazteko, $2m$ digituzko mantisa behar dugu (m digituzko galera) [39]. Salbuespena, biderkagaietako bat 2-ren berretura denean gertatzen da, orduan biderketa zehatza baita.

Adibidea. Demagun hiru digitu hamartar errealeko aritmetikarekin ari garela lanean.

Emaitza zehatza, $1,343 \times 2,103 = 2,824229$.

Hiru digitu hamartar errealeko aritmetika, $1,343 \times 2,103 \approx 2.824$.

Hauek zenbaki errealen arteko funtsezko eragiketak badira, $* : \mathbb{R}^2 \rightarrow \mathbb{R}$,

$$* \in \{+, -, \times, /\},$$

koma-higikorreko zenbakien arteko funtsezko eragiketak era honetan izendatuko ditugu $\circledast : \mathbb{F}^2 \rightarrow \mathbb{F}$,

$$\circledast \in \{\oplus, \ominus, \otimes, \oslash\}.$$

$\tilde{x}, \tilde{y} \in \mathbb{F}$ emanik eta $z = \tilde{x} * \tilde{y}$ emaitza zehatza bada, $\tilde{z} = \tilde{x} \circledast \tilde{y}$ (edo $\tilde{z} = fl(\tilde{x} * \tilde{y})$) eragiketaren emaitzaren errore absolutua eta errore erlatiboa definituko ditugu,

- Errore absolutua,

$$\Delta z = \tilde{z} - z = (\tilde{x} \circledast \tilde{y}) - (\tilde{x} * \tilde{y}).$$

- Errore erlatiboa,

$$\delta z = \frac{\Delta z}{z} == \frac{(\tilde{x} \circledast \tilde{y}) - (\tilde{x} * \tilde{y})}{(\tilde{x} * \tilde{y})}.$$

- Honako erlazio hau ondorioztatu daiteke,

$$\tilde{z} = (\tilde{x} \circledast \tilde{y}) = z + \Delta z = z(1 + \delta z).$$

Koma-higikorreko aritmetikan, $|\delta z| < u$, non $u = 2^{-m}$, beteko dela frog daiteke [26].

Zenbakizko algoritmoen biribiltze errorearen eraginaren azterketa formalak, propietate hauetan oinarritzen dira. Bestalde, errore erlatiboak emaitzaren digitu zuzenak neurtzen du:

$$\delta z \approx 10^{-k} \Rightarrow \approx k \text{ digitu hamartar zuzen.}$$

Biribiltze errorearen hedapena.

Ohiko konputazioetan, eragiketa aritmetiko kopuru handia egin behar dugu emaitza lortzeko. Batzuetan, eragiketen biribiltze erroreak elkar ezerezatzen dira baina kasu txarrenean, biribiltze errorea metatu eta magnitude handikoa izan daiteke.

Adibidea. Modu honetako batura batean , non $n > 2$ eta $\tilde{x}_1, \dots, \tilde{x}_n \in \mathbb{F}$,

$$\bigoplus_{i=1}^n (\tilde{x}_i) = \left(\sum_{i=1}^n \tilde{x}_i \right) (1 + \delta),$$

$|\delta| < u$ beteko denik, ezin daiteke bermatu.

Analisi zehatza egiten badugu $n = 3$ adibiderako, honako espresioa lortzen dugu,

$$((\tilde{x}_1 \oplus \tilde{x}_2) \oplus \tilde{x}_3) = ((\tilde{x}_1 + \tilde{x}_2)(1 + \delta_1) + \tilde{x}_3)(1 + \delta_2), \quad \delta_1, \delta_2 < u.$$

Ezabapen arazoa.

Algoritmoen kalkuluetan, doitasuna galera azkarra gerta daiteke. Horren adibidea ezabapen arazoa dugu: oso antzekoak diren bi zenbaki arteko kendura egiten dugunean gerta daitekeena.

Adibidea. Mathematican kalkulatutako adibide honetan, ezabapen errorea nola gertatzen den erakutsi dugu.

```
>> InputForm[N[Pi]]
>> 3.141592653589793

>> y=N[Pi]*10^(-10);
>> InputForm[y]
>> 3.1415926535897934*10^(-10)

>> z=1+y;
>> InputForm[z]
>> 1.0000000003141594 # 16-digitu hamartar zuzenak.
```

```
>> InputForm[z - 1.]
>> 3.141593651889707*10^(-10)    # 6-digitu hamartar zuzenak.
```

4.4. Biribiltze errorearen gutxitzeko teknikak.

Batuketa eta biderketa eragiketen biribiltze errorea kalkulatzeko algoritmoak ezagunak dira [28][53]. Algoritmo hauek, *termino-anitzeko adierazpen* implementazioetan erabiltzen dira eta baturaren kasuan, batura konpensatu izeneko algoritmoaren oinarria da. Ikusiko dugun bezala, algoritmo simpleak dira eta konputazio kostu txikia dute.

Teknika hauek, zenbakizko integrazioaren implementazioaren kalkulu "kritikoetan" erabiliko ditugu, soluzioaren doitasuna handitzeko asmoarekin.

Batura: Fast2Sum.

Fast2Sum algorithmoa, 1971.an Dekker-ek [28] asmatu zuen. Koma-higikorreko $\tilde{x}, \tilde{y} \in \mathbb{F}$ bi zenbakien non $|\tilde{x}| \geq |\tilde{y}|$, arteko batuketari $\tilde{z} = \tilde{x} \oplus \tilde{y}$ dagokion biribiltze errorea e , non $\tilde{z} + e = \tilde{x} + \tilde{y}$ den, era honetan kalkulatu daiteke,

$$\begin{aligned}\tilde{z} &= \tilde{x} \oplus \tilde{y}; \\ e &= \tilde{y} \ominus (\tilde{z} \ominus \tilde{x});\end{aligned}$$

Algoritmoa 9: Fast2Sum.

4.3.irudiaren laguntzarekin hobeto uler daiteke batuketaren biribiltze errorearen kalkulua [53].

Batura konpensatua.

Era honetako batura errekurtsiboetan,

$$z_{n+1} = z_0 + \sum_{i=0}^n x_i,$$

biribiltze errorea gutxitzeko teknika ezaguna da [53, 83, 48]. Ideia da, bi zenbakien baturan egindako biribiltze errorea lortu, eta errore hau hurrengo baturan erabiltzea. Jarraian azaltzen den moduan, urrats bakoitzaren amaieran errore esti-

\tilde{x}

\tilde{x}_1	\tilde{x}_2
---------------	---------------

 \tilde{y}

\tilde{y}_1	\tilde{y}_2
---------------	---------------

$$\tilde{z} = \tilde{x} \oplus \tilde{y}$$

\tilde{x}_1	$\tilde{x}_2 \oplus \tilde{y}_1$
---------------	----------------------------------

$$\tilde{z} \ominus \tilde{x}$$

\tilde{y}_1	0
---------------	---

$$\tilde{y} \ominus (\tilde{z} \ominus \tilde{x})$$

\tilde{y}_2	0
---------------	---

4.3. Irudia: Batuketaren biribiltze errorea.

mazioa (e_i) kalkulatuko dugu eta hurrengo urratsean, batugaiari gehituko diogu.

```

 $\tilde{z}_0 = z_0; e_0 = 0;$ 
for  $i \leftarrow 0$  to  $n$  do
     $x = \tilde{z}_i;$ 
     $y = x_i + e_i;$ 
     $\tilde{z}_{i+1} = x + y;$ 
     $e_{i+1} = (x - z) + y;$ 
end

```

Algoritmoa 10: Kahan-en batura konpensatua.

Knuth-ek eta Kahan-ek [83], batura konpensatuko algoritmoaren bidez kalkulatutako z_{n+1} batura honakoa betetzen duela,

$$\left| z_{n+1} - (z_0 + \sum_{i=0}^n x_i) \right| \leq (2u + \mathcal{O}(nu^2)) \left(|z_0| + \sum_{i=1}^n |x_0| \right),$$

frogatu zuten.

Jakina da, algoritmoaren gaiak bektoreak, $\tilde{z}_0, e_0, x_0, x_1, \dots, x_n \in \mathbb{F}^d$ diren kasurako orokortu daitekeela. Beraz, 10 algoritmoa n eta d parametroak dituen funtzio familia gisa interpretatu daiteke,

$$S_{n,d} : \mathbb{F}^{(n+3)d} \rightarrow \mathbb{F}^{2d}, \quad (4.1)$$

zein $\tilde{z}_0, e_0, x_0, x_1, \dots, x_n \in \mathbb{F}^d$ argumentuak emanik, $\tilde{z}_{n+1}, e_{n+1} \in \mathbb{F}^d$ balioak itzultzen dituen, eta $(\tilde{z}_{n+1} + e_{n+1}) \approx (\tilde{z}_0 + e_0 + x_0 + x_1 + \dots + x_n)$ adierazi nahi duen.

Zenbakizko integrazioak.

Zenbakizko integrazioetan, era honetako batura errekurtsiboa kalkulatu behar ditugu [48],

$$y_{n+1} = y_n + \delta_n,$$

non $|\delta_n| < |y_n|$ izan ohi da. Beraz, integrazioaren batura honen birbiltze errorea gutxitzeko, batura konpensatua erabiliko dugu.

$y_{n+1} \in \mathbb{R}^d$, $y_{n+1} = \tilde{y}_n + \tilde{\delta}_n$ batura zehatza izanik eta $\tilde{y}_{n+1} \in \mathbb{F}^d$, $\tilde{y}_{n+1} = \tilde{y}_n \oplus \tilde{\delta}_n$ koma-higikorreko hurbilpena izanik, batura konpensatuaren bidez lortutako errorearen estimazioa e_{n+1} ,

$$\tilde{y}_0 = fl(y_0); e_0 = fl(y_0 - \tilde{y}_0);$$

for $n = 0, 1, 2, \dots$ **do**

$$\left| \begin{array}{l} inc = \tilde{\delta}_n \oplus e_n; \\ \tilde{y}_{n+1} = \tilde{y}_n \oplus inc; \\ e_{n+1} = (\tilde{y}_n \ominus \tilde{y}_{n+1}) \oplus inc; \end{array} \right.$$

end

Algoritmoa 11: Batura konpensatua (zenbakizko integrazioa).

baturan egindako biribiltze errore zehatza da,

$$y_{n+1} = \tilde{y}_{n+1} + e_{n+1}. \quad (4.2)$$

Goian aipatutako ideia, beste ikuspegi batetik ere uler daiteke. Zenbakizko soluzioa, doitasun bikoitzeko bi balioen batura gisa $y_n = \tilde{y}_n + e_n$ (ia doitasun laukoitza), adierazten ari gara eta beraz, interpretazio honen arabera, konputazio eragiketa batzuk ia doitasun laukoitzean egiten ariko ginateke. Zentzu honetan gure implementazioan, hasierako balio zehatza $y_0 = y(t_0)$, bi balioen batura gisa $y_0 = \tilde{y}_0 + e_0$ ulertu behar da eta era honetan hasieratuko dugu,

$$\begin{aligned} \tilde{y}_0 &= fl(y_0), \\ e_0 &= fl(y_0 - \tilde{y}_0). \end{aligned}$$

Bidekerta: 2MultFMA.

IEEE 754-2008 estandarrean, *FMA* [83] (*fused multiply-add*) instrukzioa gehitu zen eta hurrengo urteetan, ordenagailu arruntetan zabaltzea espero da. Instrukzio

honen garrantzia handia da: orokorrean konputazioak azkartzen ditu eta biderketa eskalarren, matrize biderkaduren eta polinomio ebaluazioen biribiltze errore txikitzen du. *FMA* instrukzioa, zatiketa eta erro karratuaren algoritmo azkarren diseinuan ere erabiltzen da.

FMA instrukzioak, era honetako konputazioetan biribiltze errore bakarra beramatzen du,

$$fl(\tilde{x} \times \tilde{y} \pm \tilde{z}) = (\tilde{x} \times \tilde{y} \pm \tilde{z})(1 + \delta), \quad \delta < u \text{ non } u = 2^{-m}.$$

FMA instrukzioa erabilgarri dagoenean, biderketaren biribiltze errorea kalkulatzea erraza da; $\tilde{x}, \tilde{y} \in \mathbb{F}$ bi zenbakien arteko biderketari $\tilde{z} = fl(\tilde{x} \times \tilde{y})$ dagokion biribiltze errorea e , non $\tilde{z} + e = \tilde{x} \times \tilde{y}$ den, era honetan kalkulatu daiteke,

$$\begin{aligned} \tilde{z} &= fl(\tilde{x} \times \tilde{y}); \\ e &= fl(\tilde{x} \times \tilde{y} - \tilde{z}); \end{aligned}$$

Algoritmoa 12: 2MultFMA.

Sterbenz Teorema.

Sterbenz teoremaren arabera [97], bi zenbaki elkarrekiko gertu daudenean, honako baldintza betetzen bada, horien arteko kendura zehatza da.

$$x, y \in \mathbb{F}, \quad \frac{y}{2} \leqslant x \leqslant 2y \quad \Rightarrow \quad x - y \in \mathbb{F}. \quad (4.3)$$

4.5. Laburpena.

Atal honetan, koma-higikorreko aritmetikaren deskribapena egin ondoren, konputazioen doitasuna handitzeko tresnak azaldu ditugu. Tresna hauek, konputazio kostu txikia dute eta zenbakizko integrazioetan, biribiltze errorea txikitzeko aplikatuko ditugu.

Koma-higikorreko aritmetikan sakontzeko honako bibliografia azpimarratuko dugu: "Numerical computing with IEEE floating point arithmetic", Michael L.Overton [86], "Handbook of floating-point arithmetic", Jean-Michael Muller [83], "Accuracy and stability of numerical algorithms", Nicholas J Higham [53] eta "A graduate introduction to numerical methods", Rober M Corless [26].

5. Kapitulua

IRK: Puntu-finkoaren iterazioa.

5.1. Sarrera.

Sistema Hamiltondar ez-zurrunen doitasun altuko zenbakizko integraciota rako, puntu-finkoaren iterazioan oinarritutako IRK metodoaren implementazioa garatu dugu. Konputazioetako koma-higikorreko aritmetika erabiltzen denez, biribiltze erroreak integrazioen doitasuna mugatzen du. Hortaz, epe luzeko doitasun altuko zenbakizko integrazioen implementazioetan, biribiltze errorearen eragina txikia izatea eta honen estimazioa ezagutzea interesgarria izan daiteke.

Runge-Kutta inplizitu simplektikoaren (Gauss nodoetan oinarritutako Runge-Kutta kolokazio metodoa) implementazioa proposatu dugu, eta biribiltze errorearen garapena txikia izateko ahalegin berezia egingo dugu. Implementazioa, problema ez-zurrunetan aplikatzeko garatu dugunez, ekuazio-sistema inplizitua, puntu-finkoaren iterazioaren bidez ebatziko dugu (puntu-finkoaren iterazioan eta Newtonen iterazio simplifikatuan oinarritutako implementazioen eraginkortasun azterketak, [48, 60] lanetan kontsultatu daiteke).

Integrazioaren exekuzio denborak onargarriak izan daitezten, honako suposizioa egingo dugu: ekuazio diferentzialaren eskuin aldeko funtziaren sarrera eta irteera argumentuak, makina zenbakiak (koma-higikorreko aritmetika hardware bidezko exekuzioa azkarra duen datu-mota) direla. Gaur-egungo zientzia-konputazioa, 64-biteko koma-higikorreko aritmetikan (*double* datu-mota) oinarritzen da eta beraz, erabiltzaileak ekuazio diferentziala, datu-mota honetan zehaztuko duela suposatu dugu.

Gure implementazioa, biribiltze errorearen garapenaren ikuspegitik, ia optimoa izatea nahi dugu, hau da, puntu-finkoaren iterazioan oinarritutako implementazio onenaren birbiltze errorearen garapen antzekoa duen implementazioa lortu nahi dugu. Era berean, biribiltze errorearen estimazioa kalkulatzeko teknika garatu dugu, doitasun txikiagoko bigarren integrazio baten soluzioaren differentzia gisa

kalkulatzen dena.

Lehenengo, Hairer-en IRK metodo simplektikoaren implementazioa [49] aztertuko dugu. Ondoren, IRK implementazio hau hobetzeko gure proposamenak azalduko ditugu eta azkenik, zenbakizko integrazioen bidez, gure implementazio berriaren abantailak erakutsiko ditugu.

5.2. Hairer-en implementazioa.

IRK implementazio estandarra.

Gure abiapuntua, Hairer-ek [49] proposatutako IRK metodoaren implementazioa (batura konpensatuaren teknikarekin [53] garatutakoa) da. Lan honetan, puntu-finkoaren iterazioan oinarritutako IRK metodo simplektikoaren implementazio estandarrean, biribiltze erroreak energian errore sistematiko bat eragiten zuela ohartu zen, beste metodo simplektiko esplizituetan gertatzen ez zena. Bere azterketaren ondorioen arabera, errore sistematiko honen jatorriak bi dira:

1. Aplikatutako IRK metodoa ez da simpletikoa (5.8), integrazioan $a_{ij}, b_i \in \mathbb{R}$ koefiziente zehatzak erabili ordez, biribildutako $\tilde{a}_{ij}, \tilde{b}_i \in \mathbb{F}$ erabiltzen direlako.
2. Geratze irizpide estandarrarekin,

$$\Delta^{[k]} = \max_{i=1, \dots, s} \|Y_i^{[k]} - Y_i^{[k-1]}\|_\infty \leq \text{tol} \quad (5.1)$$

(non tol finkatutako tolerantzia den), urrats bakoitzeko aplikatutako puntu-finkoaren iterazioek, errore sistematikoa eragiten dute.

Konponbideak.

Energiaren errore sistematikoa desagertzeko, implementazio estandarrean honako aldaketak proposatu zituen:

1. Metodoaren koefizienteen doitasuna handitzea, koefiziente bakoitza komahigikorreko bi koefizienteen batura konsideratz,

$$a_{ij} = \hat{a}_{ij} + \tilde{a}_{ij}, \quad b_i = \hat{b}_i + \tilde{b}_i \quad (5.2)$$

non $\hat{a}_{ij} > \tilde{a}_{ij}$ eta $\hat{b}_i > \tilde{b}_i$ diren.

Adibidez, koefizienteak era honetan zehaztu daitezke,

$$\hat{a}_{ij} = (a_{ij} \otimes 2^{10}) \oslash 2^{10}, \quad \tilde{a}_{ij} = a_{ij} \ominus \hat{a}_{ij}.$$

2. Puntu-finkoaren iterazioen geratze irizpide berria; iterazioak geratu, definitutako norma txikitzeari uzten dionean edo konbergentzia lortu duenean,

$$\Delta^{[k]} = 0 \text{edo } \Delta^{[k]} \geq \Delta^{[k-1]}. \quad (5.3)$$

Hairer-en implementazioaren algoritmoa.

Hairer-ek bere implementazioaren **Fortran kodea** eskuragarri du. Jarraian, bere implementazioaren **13** algoritmoa eta erabilitako batura konpensatuaren teknikaren **14** algoritmoa, zehaztu ditugu.

```

 $y_0 = y(t_0); e_0 = 0;$ 
for  $n \leftarrow 0$  to (endstep − 1) do
     $k = 0;$ 
     $Y_{n,i}^{[0]} = y_n + h c_i f(y_n);$ 
    while ( $\Delta^{[k]} \neq 0$  and  $\Delta^{[k]} < \Delta^{[k-1]}$ ) do
         $k = k + 1;$ 
         $F_{n,i}^{[k]} = f(Y_{n,i}^{[k-1]});$ 
         $Y_{n,i}^{[k]} = y_n + h \left( \sum_{j=1}^s \hat{a}_{ij} F_{n,j}^{[k]} \right) + h \left( \sum_{j=1}^s \tilde{a}_{ij} F_{n,j}^{[k]} \right);$ 
         $\Delta^{[k]} = \max_{i=1,\dots,s} \|Y_{n,i}^{[k]} - Y_{n,i}^{[k-1]}\|_\infty;$ 
    end
     $(y_{n+1}, e_{n+1}) \leftarrow \text{BaturaKonpensatua}(y_n, e_n, F_n^{[k]});$ 
end
```

Algoritmoa 13: Hairer-en IRK implementazioa

Function BaturaKonpensatua ($y_n, e_n, F_n^{[k]}$)

```

 $\hat{\delta}_n = h \left( \sum_{i=1}^s \hat{b}_i F_i^{[k]} \right);$ 
 $\tilde{\delta}_n = h \left( \sum_{i=1}^s \tilde{b}_i F_i^{[k]} \right);$ 
 $ee = \hat{\delta}_n + e_n;$ 
 $yy = y_n + ee;$ 
 $ee = (y_n - yy) + ee;$ 
 $ee = \tilde{\delta}_n + ee;$ 
 $y_{n+1} = y_n + ee;$ 
 $e_{n+1} = (yy - y_{n+1}) + ee;$ 
return ( $y_{n+1}, e_{n+1}$ );
```

Algoritmoa 14: Hairer-en IRK implementazioaren, batura konpensatua

Hairer-en implementazioaren arazoak.

Hairer-ek bere implementazioa, *Hénon-Helies* eta eguzki-sistemaren kanpo-planeten problemetarako energiaren errore sistematikorik ez zegoela baiezktatu zuen. Energiaren errorea, $k\sqrt{t_n}$ espresioaren arabera handitzen dela erakutsi zuen eta implementazioak, *Brouwer legea* [43] betetzen duela ondorioztatu zuen. Integracioen azterketa estatistikoa egin zuen biribiltze errorearen ausazkotasuna baiezttatzeko. Problemaren hasierako balio zehatz bati dagokion perturbatutako $P = 1.000$ integrazio exekutatu zituen eta integrazio horien energia errorearen batezbestekoa (μ), zero eta desbideratze estandarra (σ), $k\sqrt{t_n}$ espresioaren araberakoa zela erakutsi zuen. Era berean, integrazio amaiera uneko energi erroreen histogramak, $N(\mu, \sigma)$ distribuzio normala betetzen duela erakutsi zuen.

Hairer-en *Fortran* implementazioarekin, zenbakizko integrazio berriak egin ditugu eta zenbait kasutan, geratze irizpidea goizegi geratzen dela konprobatu dugu. Eguzki-sistemaren kanpo-planeten hasierako baliodun problemaren integrazioa, $h = 500/3$ eguneko urrats luzerarekin zuzena da baina aldiz okerra $h = 1000/3$ urratsarekin. Gainera, bere implementazioaren biribiltze errorearen propagazioa optimoa ez dela uste dugu.

5.3. Gure implementazioa.

IRK metodoaren puntu-finkoaren iterazioan oinarritutako implementazioa hobetzeko lau proposamen egingo ditugu. Lehenik, Runge-Kutta metodoaren birformulazio bat proposatuko dugu, metodoa definitzen duten koma-higikorreko koefizienteek, izaera simplektikoa zehazki bete dezaten. Bigarrenik, Hairer-ek proposatutako geratze irizpidearen [49] arazoak gainditzen, geratze irizpide sendoagoa eta normaren independentea garatu dugu.

Hirugarrenik, biribiltze errorea gutxitzeko helburuarekin, koma-higikorreko konputazioak bereziki zaindu ditugu. Kahan-en batura konpensatuaren [62] [53] [83] algoritmoaren aldaera bat aplikatu dugu.

Azkenik, biribiltze errorearen estimazioa monitorizatzeko teknika proposatu dugu. Biribiltze errorearen estimazioa, integrazio nagusi eta doitasun txikiago-ko bigarren integrazio baten soluzioen arteko diferentzia gisa kalkulatuko dugu. Zenbakizko soluzio hauek, bi modutara kalkulatu daitezke: paraleloan, exekuzio independenteak konsideratuta; edo bi soluzioen integrazio sekuentziala kalkulatuz, konputazio kostu gehigarri txikiarekin.

Kontsideratu dezagun, honako hasierako baliodun problema,

$$\dot{y} = f(t, y), \quad y(t_0) = y_0, \tag{5.4}$$

non $y_0 \in \mathbb{R}^d$ eta $f : \mathbb{R}^{d+1} \longrightarrow \mathbb{R}^d$ diren.

Denbora diskretizazioa $t_0 < t_1 < t_2 < \dots$ emanik, (5.4) problemaren $y(t)$ soluzioaren $y_n \approx y(t_n)$ ($n = 1, 2, \dots$) zenbakizko soluzioa, integrazio metodo bat aplikatuz lortuko dugu,

$$y_{n+1} = \Phi(y_n, t_n, t_{n+1} - t_n), \quad (5.5)$$

non $\Phi : \mathbb{R}^{d+2} \rightarrow \mathbb{R}^d$ den.

S-ataletako IRK metodoaren kasuan, a_{ij}, b_i , eta $c_i \in \mathbb{R}$ ($1 \leq i, j \leq s$) koefizienteek definitzen dute Φ integrazio metodoa,

$$\Phi(y, t, h) = y + h \sum_{i=1}^s b_i f(t + c_i h, Y_i), \quad (5.6)$$

non $c_i = \sum_{j=1}^s a_{ij}$ izan ohi da eta Y_i atalak, era horretan implizituki definitzen diren,

$$Y_i = y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j) \quad i = 1, \dots, s. \quad (5.7)$$

IRK metodoa simplektikoa da, baldin soilik honako baldintza betetzen bada [60],

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (5.8)$$

IRK metodoa simplektikoa bada, (5.4) sistemaren integral koadratikoak zehazki mantenduko ditu eta sistema Hamiltondarra balitz, $H(y)$ funtzioko Hamiltondarraren balioa integrazio tarte luzeetarako ondo kontserbatuko du.

Metodoaren birformulazioa.

Metodoa definitzen duten $a_{ij}, b_i \in \mathbb{R}$ koefizienteak, $\tilde{a}_{ij} := fl(a_{ij}), \tilde{b}_i := fl(b_i) \in \mathbb{F}$ biribildutako koefizienteen hurbilpenekin ordezkatzen ditugunean, hauek ez du te simplektikoa izateko baldintza (5.8) beteko. Ondorioz, metodoak ez ditu integral koadratikoak kontserbatuko eta Hamiltondar funtzioaren eboluzioan, drift lineala ageriko da [60].

Arazo hau gainditzeko, IRK metodoa era horretan birformulatuko dugu,

$$Y_{n,i} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}, \quad L_{n,i} = h b_i f(Y_{n,i}), \quad i = 1, \dots, s, \quad (5.9)$$

$$y_{n+1} = y_n + \sum_{i=1}^s L_{n,i}, \quad (5.10)$$

non

$$\mu_{ij} = a_{ij}/b_j, \quad 1 \leq i, j \leq s.$$

Runge-Kutta metodoa simplektikoa izateko baldintza (5.8), modu honetan berridatziko dugu,

$$\mu_{ij} + \mu_{ji} - 1 = 0, \quad 1 \leq i, j \leq s. \quad (5.11)$$

Birformulazio honek formulazio estandarrarekiko duen abantaila, simplektizidade baldintzaren espresioan biderketarik ez agertzeak, baldintza zehazki beteko duten $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak aurkitzeko bidea errazten duela da. Jarraian, Gauss kolokazio metodoaren $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak finkatzeko teknika deskribatuko dugu.

1. Koefizienteen matrize diagonaleko balioek koma-higikorreko adierazpen zehatza dute,

$$\tilde{\mu}_{ii} := 1/2, \quad i = 1, \dots, s.$$

2. Koefiziente matrizearen behe-diagonaleko balioak finkatuko ditugu,

$$\tilde{\mu}_{ij} := fl(\mu_{ij}), \quad 1 \leq j < i \leq s.$$

3. Koefiziente matrizearen goi-diagonaleko balioak esleituko ditugu,

$$\tilde{\mu}_{ji} := 1 - \tilde{\mu}_{ij}, \quad 1 \leq j < i \leq s.$$

Sterbenz-en teoremak (4.3), $1/2 < |\mu_{ij}| < 2$ denez, $1 - \tilde{\mu}_{ij}$ balioak koma-higikorreko adierazpen zehatza izango duela ziurtatzen du. Laburtuz, hauek ditugu birformulatutako simplektizitate baldintza (5.11) zehazki betetzen duten koma-higikorreko $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak,

$$\tilde{\mu} = \begin{pmatrix} 1/2 & 1 - fl(\mu_{21}) & \dots & 1 - fl(\mu_{s1}) \\ fl(\mu_{21}) & 1/2 & \dots & 1 - fl(\mu_{s2}) \\ \vdots & \vdots & \ddots & \vdots \\ fl(\mu_{s1}) & fl(\mu_{s2}) & \dots & 1/2 \end{pmatrix} \in \mathbb{R}^{s \times s}. \quad (5.12)$$

Bestalde, b_i koefizienteak eta ν_{ij} atalen hasieraketarako interpolazio koefizienteak finkatzeko zehaztapenak ere emango ditugu.

1. hb_i koefizienteak.

Gure implementazioan, $hb_i = h \times b_i$ koefizienteak aurre-kalkulatuko ditugu.

Koefiziente hauek simetrikoak direla eta $\sum_{i=1}^s hb_i = h$ berdintza bete behar dela jakinda, modu honetan kalkulatuko ditugu,

$$hb_i = fl(h \times b_i), \quad i = 2, \dots, s-1,$$

$$hb_1 := hb_s := \left(h - \sum_{i=2}^{s-1} hb_i \right) / 2.$$

2. ν_{ij} interpolazio koefizienteak.

Formulazio estandarraren λ_{ij} koefizienteetatik abiatuta (2.3.atala), formulazio berriari dagozkion interpolazio ν_{ij} koefizienteak era honetan definituko ditugu,

$$Y_{n,i}^{[0]} = y_n + h \sum_{j=1}^s \nu_{ij} L_{n-1,j}, \quad \nu_{ij} = \lambda_{ij}/b_j \quad 1 \leq i, j \leq s. \quad (5.13)$$

Geratze irizpidea.

Ekuazio implizituaren (5.9) soluzioaren hurbilpena lortzeko, puntu-finkoaren iterazioa gogoratuko dugu: iterazioaren abiapuntua $Y_{n,i}^{[0]}$ finkatu eta $k = 1, 2, \dots$ iterazioetarako, $Y_{n,i}^{[k]}$ hurbilpenak lortu geratze irizpidea bete arte.

```
for ( $k=1,2,\dots$  konbergentzia lortu arte) do
     $L_{n,i}^{[k]} = h b_i f(Y_{n,i}^{[k-1]}), \quad i = 1, \dots, s;$ 
     $Y_{n,i}^{[k]} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]}, \quad i = 1, \dots, s;$ 
end
```

Algoritmoa 15: IRK puntu-finkoaren iterazioa

Iterazioa, $Y_{n,i}^{[0]} = y_n$ balioarekin edo aurreko urratsetako atalen balioen interpolazioz lortutako balioekin [48] hasieratu daiteke. Urrats luzera h adina txiki aukeratuz gero, iterazioek, (5.9) ekuazio aljebraikoen soluzioa den puntu-finkora konbergituko dute. Gure zenbakizko esperimentuetan nahiz Hairer-ek eginda esperimentuetan ere [49], h urrats luzera txikiarekin integrazioaren urrats gehienetan, puntu-finkoa lortzen dela baiezttatu dugu.

IRK metodoaren implementazio estandarraren geratze irizpidea honakoa da,

$$\begin{aligned} \Delta^{[k]} &= (Y_{n,1}^{[k]} - Y_{n,1}^{[k-1]}, \dots, Y_{n,s}^{[k]} - Y_{n,s}^{[k-1]}) \in \mathbb{F}^{sd}, \\ \|\Delta^{[k]}\| &\leq tol \end{aligned} \quad (5.14)$$

non $\|\cdot\|$ aurre-finkatutako bektore norma eta tol , tolerantzia errorea den. Tolerantzia txikiegia aukeratzen bada, tolerantzia hori ez lortzea eta infinituki iterazioak exekutatzea gerta daiteke. Baino tolerantzia ez bada behar bezain txikia aukeratzen, iterazioa puntu-finkora iritsi aurretik geratuko da eta lortutako $Y_i^{[k]}$ hurbilpenaren errorea, biribiltze errorea baino handiagoa izango da. Gainera, Hairer-ek [49] iterazio errorea modu sistematikoan metatzeko konprobatu zuela.

Hairer-ek proposatutako geratze irizpidea gogoratuko dugu; $\Delta^{[k]} = 0$, puntu-finkora iritsi delako ; edo $\|\Delta^{[k]}\| \geq \|\Delta^{[k-1]}\|$, biribiltze errorea nagusi delako, non

$$\|\Delta^{[k]}\| := \max_{i=1,\dots,s} \|Y_i^{[k]} - Y_i^{[k-1]}\|_\infty.$$

Orokorean, geratze irizpide honek ondo funtzionatzen du baina zenbait esperimentuetan, iterazioak goizegi geratzen direla konprobatu dugu. Hairer-ek, eguzki-sistemaren kanpo-planeten problemaren $h = 500/3$ eguneko urrats luzerarekin egindako integrazioan ondo funtzionatzen du, baina $h = 1000/3$ urrats luzerarekin integratzerakoan, tamaina handiko energia errore agertzen da. Energiaren errore erlatiboaren eboluzioa 5.1.(a) irudian erakutsi dugu. Integrazioaren lehen urratsaren iterazioak aztertzen badugu,

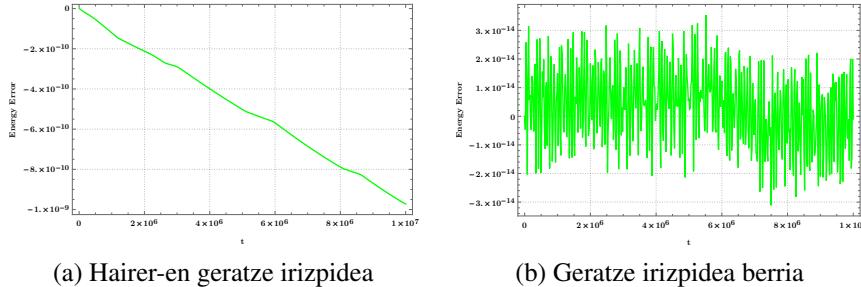
$$\|\Delta^{[1]}\| > \|\Delta^{[2]}\| \cdots > \|\Delta^{[12]}\| = 3.91 \times 10^{-14} \leq \|\Delta^{[13]}\| = 4.35 \times 10^{-14}$$

13.iterazioan geratuko dela konprobatuko dugu. Puntu-finkoaren iterazioa goizegi geratu da, hurrengo iterazioetan ikusi daitekeenez, $\|\Delta^{[13]}\| > \|\Delta^{[14]}\| > \|\Delta^{[15]}\| > \|\Delta^{[16]}\| = 0$ konbergentziaren hobekuntza gertatzen baita.

Hairer-en geratze irizpide aztertu ondoren bi ondorio atera daitezke. Hori azaltzeko, lehenik honako notazioa finkatuko dugu,

$$\Delta_j^{[k]}, \text{ non } \Delta^{[k]} \in \mathbb{F}^{sd} (1 \leq j \leq sd).$$

Lehenik, ezin daiteke suposatu $\{\Delta_j^{[0]}, \Delta_j^{[1]}, \dots, \Delta_j^{[k]}\}$ segida beherakorra denik. Bigarrenik, $\|\Delta^{[k]}\| \geq \|\Delta^{[k-1]}\|$ baldintzak, biribiltze errorea nagusia dela adierazten duen arren, $\exists j \in \{1, \dots, sd\}$ daiteke osagairik, non $|\Delta_j^{[k]}| < |\Delta_j^{[k-1]}|$ hobetzeko tartea duen eta horregatik, iterazio gehiago eman beharko genitzuke.



5.1. Irudia: Energia errore erlatiboaren eboluzioa, $h = 1000/3$ urrats luzerarekin eguzki-sistemaren kanpo-planeten problemaren integraziorako [49]. (a) Hairer-en geratze irizpidea, (b) Geratze irizpidea berria

Arazo hauei aurre egiteko, geratze irizpidea berri bat proposatuko dugu. Iterazioak $k = 1, 2, \dots$ jarraitzea, $\Delta^{[k]} = 0$ bete arte edo honako baldintza bi iterazio jarraietan betetzen den artean,

$$\forall j \in \{1, \dots, sd\}, \quad \min \left(\{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} / \{0\} \right) \leq |\Delta_j^{[k]}|. \quad (5.15)$$

$K, k = K - 1$ eta $k = K$ balioetarako (5.15) baldintza betetzen ez duen lehen zenbaki oso positiboa bada, $y_{n+1} \approx y(t_{n+1})$ era honetan kalkulatuko dugu,

$$y_{n+1} = y_n + \sum_{i=1}^s L_{n,i}^{[K]}.$$

Hainer-ek, eguzki-sistemaren kanpo-planeten problemaren $h = 1000/3$ eguneko urrats luzerarekin egindako integrazioa errepikatu dugu eta geratze irizpide berriarekin, energia errorearen eboluzio zuzena dela ikus daiteke (5.1.(b) Irudia).

Tolerantzi testa.

Urrats gehienetan, puntu-finkoaren iterazioak $\forall j, \Delta_j^{[K]} = 0$ bete delako geratuko dira. Gainontzeko urrats gutxi horietan, zeinetan iterazioa $\exists j, \Delta_j^{[K]} \neq 0$ izanik geratu den, orduan urratsa onargarria den ala ez erabaki behar dugu. Iterazioa, biribiltze errorearen eraginez edo h urrats luzera behar adina txikia aukeratu ez delako, geratu daiteke.

Puntu-finkoaren iterazioak amaitzerakoan, urratsa eman aurretik erabiltzaileak definitutako tolerantzia lortu den ala ez aztertuko dugu. Horretarako, iterazioaren azken bi hurbilpenak konparatuko ditugu. Honako notazioaren laguntzarekin,

$$Y_i = Y_i^{[k]}, \quad \tilde{Y}_i = Y_i^{[k-1]}, \quad i = 1, \dots, s,$$

erabiltzaileak finkatutako tolerantzia erlatiboa eta tolerantzia absolutuaren parametroen arabera ($rto_i, atol_i, i = 1, \dots, d$), *distantzia normalizatua* definituko dugu,

$$\max_{i=1, \dots, d} \frac{\max_{j=1, \dots, s} |Y_j^i - \tilde{Y}_j^i|}{\left(((\max_{j=1, \dots, s} |Y_j^i| + \max_{j=1, \dots, s} |\tilde{Y}_j^i|)/2) rtol_i + atol_i \right)}.$$

Distantzi normalizatua > 1 bada, orduan ez da lortu tolerantzia eta integrazioa amaituko dugu. Azpimarratu behar da, tolerantzia ez dugula erabiliko puntu-finkoaren iterazio geratzeko, behin iterazioa geratu denean, urratsa onargarria den ala ez erabakitzeko baizik.

Biribiltze errorea gutxitzeko teknikak.

4.4. ataleko koma higikorreko aritmetikaren azalpenetan, batuketa nahiz biderketa eragiketen biribiltze errore zehatza, modu errazean kalkulatu daitekeela ikusi genuen. Eragiketa hauen biribiltze erroreak, ondorengo konputazioetan erabiliko ditugu, soluzioaren doitasuna hobetzeko.

Batura errekurtsiboen konputazioen doitasuna hobetzeko teknikari *batura konpensatua* esaten zaio (10) eta zenbakizko integrazioetan erabili ohi da. Atal honetan, batetik IRK metodoetan batura konpensatuaren aplikazio estandarra hobetze-ko proposamena azalduko dugu. Beste aldetik, IRK metodoaren gure implementazioan biribiltze errorearen beste jatorri nagusiak (biderketa eta batuketa bat) modu finagoan kalkulatzeko proposamena egingo dugu.

Batura konpensatua.

Integrazioaren zenbakizko soluzioa, $y_n \approx y(t_n) \in \mathbb{R}^d$, $n = 1, 2, \dots$, bi bektoreen batura gisa, $\tilde{y}_n + e_n \in \mathbb{F}^d$ lortuko dugu. Hasierako balioa $y_0 \in \mathbb{R}^d$, $\tilde{y}_0 + e_0$ batura moduan adieraziko dugu, non $\tilde{y}_0 = f_l(y_0)$ eta $e_0 = f_l(y_n - \tilde{y}_0)$ diren.

IRK metodoaren implizituki $Y_{n,i}$ atalak askatzeko ekuazioetan, \tilde{y}_n balioa era-bili ordez, $(\tilde{y}_n \oplus e_n)$ expresioa erabiltzea proposatuko dugu,

$$L_{n,i}^{[k]} = h b_i f(Y_{n,i}^{[k-1]}), \quad Y_{n,i}^{[k]} = \tilde{y}_n \oplus \left(e_n \oplus \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]} \right). \quad (5.16)$$

Aldaketa honekin, lortutako zenbakizko soluzioaren doitasuna, batura kon-pensatu estandarrarekin baino zerbait hobea izatea espero dugu.

Urratsaren konputazioa.

$\tilde{y}_{n+1}, e_{n+1} \in \mathbb{F}^d$, non $\tilde{y}_{n+1} + e_{n+1} \approx y(t_{n+1})$ den, era honetan kalkulatuko dugu:

1. Biderketaren biribiltze errorea.

$hb_i f(Y_{n,i})$ biderketaren biribiltze errorea kalkulatu eta e_n gaiari gehituko diogu. Biderketaren biribiltze errorea jasotzeko, *FMA* eragiketan oinarritu-tako teknika (12 algoritmoa) aplikatuko dugu.

$$\begin{aligned} E_{n,i} &= hb_i f(Y_{n,i}^{[K-1]}) - L_{n,i}^{[K]}, \quad i = 1, \dots, s, \\ \delta_n &= e_n + \sum_{j=1}^s E_{n,j}. \end{aligned}$$

2. Batura konpensatua.

Azkenik, batura konpensatua (4.1) aplikatuko dugu,

$$(\tilde{y}_{n+1}, e_{n+1}) = S_{s,d}(\tilde{y}_n, \delta_n, L_{n,1}^{[K]}, \dots, L_{n,s}^{[K]}). \quad (5.17)$$

Function BaturaKonpensatua ($\tilde{y}_n, \delta_n, L_{n,1}^{[K]}, \dots, L_{n,s}^{[K]}$)

```

 $s_0 = \tilde{y}_n$ 
 $ee = \delta_n$ 
for  $i \leftarrow 1$  to ( $s$ ) do
     $s_1 = s_0$ 
     $inc = L_{n,i}^{[K]} + ee$ 
     $s_0 = s_1 + inc$ 
     $ee = (s_1 - s_0) + inc$ 
end
 $\tilde{y}_{n+1} = s_0$ 
 $e_{n+1} = ee$ 
return ( $\tilde{y}_{n+1}, e_{n+1}$ )

```

Algoritmoa 16: BaturaKonpensatua, $S_{s,d}(\tilde{y}_n, \delta_n, L_{n,1}^{[K]}, \dots, L_{n,s}^{[K]})$ funtziaren implementazioa da

Biribiltze errorearen estimazioa.

Zenbakizko soluzioaren $\tilde{y}_n + e_n \approx y(t_n)$ $n = 1, 2, \dots$, biribiltze errorearen estimazioa, doitasun txikiagoko bigarren zenbakizko integrazio baten soluzioaren $\hat{y}_n + \hat{e}_n \approx y(t_n)$ differentzia gisa kalkulatuko dugu. Jarraian, zehaztapenak emanago ditugu.

$r \geq 0$ zenbaki osoa, eta $x \in \mathbb{F}$ (m -biteko doitasuneko koma-higikorreko zenbakia) izanik, honako funtzia definituko dugu,

Function flr (x, r)

```

 $res = (2^r x + x) - 2^r x$ 
return res

```

Algoritmoa 17: flr

Funtzia honek, $(m - r)$ -biteko doitasuneko koma-higikorreko zenbakia itzultzen du, edo beste modu batera esanda, $x \in \mathbb{F}$, m -biteko koma-higikorreko zenbakiaren azken r bitak zeroan jartzen dituen funtzia da.

Bigarren zenbakizko soluzioa, $(\hat{y}_n + \hat{e}_n)$, r ($r < m$) balio bat finkatuta eta aurreko implementazioaren kalkulua (5.17), beste era honetan kalkulatuko dugu,

$$(\hat{y}_{n+1}, e_{n+1}) = S_{s,d}(\hat{y}_n, \hat{\delta}_n, flr(L_{n,1}^{[K]}, r), \dots, flr(L_{n,s}^{[K]}, r)).$$

Zenbakizko integrazioaren biribiltze errorearen estimazioa, soluzio nagusiarren ($y_n + e_n$) eta r balio txiki baterako (esaterako $r = 3$) kalkulatutako bigarren zenbakizko soluzioaren ($\hat{y}_n + \hat{e}_n$) arteko differentziaren norma bezala kalkulatuko dugu.

$$estimazio_n^i = \|(y_n^i + e_n^i) - (\hat{y}_n^i + \hat{e}_n^i)\|_2, \quad i = 1, \dots, d. \quad (5.18)$$

Biribiltze errorearen estimazioa lortzeko, bi integrazioak sekuentzialki eta *CPU* konputazio kostu txikiarekin kalkulatzeko teknika deskribatuko dugu. Urrats ba-

koitzean, bi integrazioen $Y_{n,i}, \hat{Y}_{n,i}$ ($i = 1, \dots, s$) atalako balioak, biribiltze errorearen estimazioa handiegia ez den artean, antzekoak mantentzen dira. Ondorioz, lehen integrazioaren bukaerako $Y_{n,i}^{[k]}$ ($i = 1, \dots, s$) atalen balioak, bigarren integrazioaren $\hat{Y}_{n,i}^{[0]}$ ($i = 1, \dots, s$) atalen hasieraketarako erabiltzen baditugu, bigarren integrazioak, iterazio kopuru txikia beharko ditu (ikus 18 algoritmoa).

```

for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $Y_n^{[0]} = G(Y_{n-1}, h);$ 
    ... lehen integrazioa ...;
     $(y_{n+1}, e_{n+1}) \leftarrow BaturaKonpensatua(y_n, \delta_n, L_n^{[K]});$ 
    if ( $initwithfirst$ ) then
         $\hat{Y}_n^{[0]} = Y_n^{[k]} + (\hat{y}_n - y_n);$ 
    else
         $\hat{Y}_n^{[0]} = G(\hat{Y}_{n-1}, h);$ 
    end
    ... bigarren integrazioa ...;
     $(\hat{y}_{n+1}, \hat{e}_{n+1}) \leftarrow BaturaKonpensatua(\hat{y}_n, \hat{\delta}_n, flr(\hat{L}_n^{[K]}, r));$ 
     $estimation_{n+1} = \|(y_{n+1} + e_{n+1}) - (\hat{y}_{n+1} - \hat{e}_{n+1})\|_2;$ 
end
```

Algoritmoa 18: RKG2: errore estimazioa

non $G()$ interpolazio funtzioa den eta $initwithfirst$ aldagaiak egiazko balioa izango duen, integrazioen arteko diferentzia txikian den artean.

Algoritmoa.

Azkenik, puntu-finkoaren iterazioan oinarritutako IRK implementazio berriari dagoen algoritmoa laburtuko dugu ([19](#) algoritmoa).

```

 $\tilde{y}_0 = fl(y_0);$ 
 $e_0 = fl(y_0 - \tilde{y}_0);$ 
for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $k = 0;$ 
    Hasieratu  $Y_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
    while (not konbergentzia) do
         $k = k + 1;$ 
         $F_{n,i}^{[k]} = f(Y_{n,i}^{[k-1]})$ ;
         $L_{n,i}^{[k]} = h b_i F_{n,i}^{[k]}$ ;
         $Y_{n,i}^{[k]} = \tilde{y}_n + (e_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]})$ ;
        konbergentzia  $\leftarrow$  GeratzeIrizpidea( $Y^{[k]}$ ,  $Y^{[k-1]}$ ,  $\Delta_{min}$ );
    end
    if ( $\exists j$  non  $\Delta_j^{[K]} \neq 0$ ) then
        if ( $NormalizeDistance(Y^{[k]}, Y^{[k-1]}) > 1$ ) then
            fail convergence;
        end
    end
     $E_{n,i} = h b_i f_{n,i}^{[k]} - L_{n,i}^{[k]}$ ;
     $\delta_n = e_n + \sum_{i=1}^s E_{n,i}$ ;
     $(\tilde{y}_{n+1}, e_{n+1}) \leftarrow$  baturakonpensatua( $\tilde{y}_n, \delta_n, L_n^{[k]}$ );
end
```

Algoritmoa 19: IRK (puntu-finkoaren iterazio).

Zenbakizko soluzioa.

Integrazio tartea $[t_0, t_{end}]$ eta urrats tamaina h bada, emandako urrats kopurua $N = (t_{end} - t_0)/h$ izango da. Bestalde, m erabiltzaileak finkatutako soluzioen frekuentzia bada, $t_i = t_0 + i * (m \cdot h)$, $i = 1, \dots, N/m$ uneetarako, ohiko da zenbakizko soluzioa fitxategi bitar batean itzultzea.

Erabiltzaileak, bi integrazio mota exekutatu ditzake:

1. Integrazio arrunta.

Zenbakizko integrazio bakarra konputatzen da eta zenbakizko soluzioa (y_i, e_i) fitxategi batean itzultzen dugu. Fitxategiaren lerro bakoitzaren egitura ho-

nakoa da:

$$(t_i, y_i, e_i) \text{ non } t_i \in \mathbb{R} \text{ eta } y_i, e_i \in \mathbb{R}^d.$$

$$y_i = (q_i, p_i) \text{ eta } e_i = (eq_i, ep_i).$$

non

$$(q_i + eq_i, p_i + ep_i) \approx (q(t_i), p(t_i)), \quad i = 1, \dots, N/m.$$

2. Integrazioa errore estimazioarekin.

Integrazioaren zenbakizko soluzioa (y_i, e_i) eta errorearen estimaziona (est_i) fitxategi batean itzultzen ditugu. Lerro bakoitzaren egitura honakoa da,

$$(t_i, y_i, e_i, est_i) \text{ non } t_i \in \mathbb{R} \text{ eta } y_i, e_i, est_i \in \mathbb{R}^d.$$

$$y_i = (q_i, p_i), \quad e_i = (eq_i, ep_i) \text{ eta } est_i = (estq_i, estp_i).$$

5.4. Zenbakizko esperimentuak.

Atal honetan, puntu-finkoaren iterazioan oinarritutako 6-ataletako Gauss metodoaren implementazioarekin egindako zenbakizko esperimentuak azalduko ditugu. Esperimentu hauen konputaziorako, 64-biteko doitasuneko IEEE koma-higikorreko aritmetika erabili dugu.

Problemak.

Bi Hamiltondar sistema konsideratuko ditugu: pendulu bikoitz arruntaren eta kanpo-planeten problemak [48] [29]. Integrazio guztietaan, h urrats luzera, truncatze errorea biribiltze errorea baino txikiago izan dadin aukeratu dugu.

Pendulu bikoitz arrunta

Pendulu bikoitz arruntaren Hamiltondarra eta parametroak, 3.2. atalean definitu ditugu. Sistema Hamiltondar honetarako, bi hasierako balio konsideratu ditugu: hurrenez hurren, izaera ez-kaotikoa (NCDP) eta kaotikoa (CDP) duten mugimenduak eragiten dituztenak. Bi hasierako baliodun problema hauek (NCDP eta CDP), energia errorearen eboluzioaren eta biribiltze errorearen estimazioaren azterketa egiteko konsideratuko ditugu. Energia errorearen jatorria aztertzeko integrazio luzerako ordea, problema ez-kaotiko (NCDP) bakarrik konsideratu dugu.

Kanpo-planeten problema

Eguzki-sistemaren kanpo-planeten eredu Newtoniarra, 3.4. atalean azaldu dugu. Hamiltondar sistema banagarria da,

$$H(q, p) = T(p) + Uq,$$

eta ezaguna da, puntu-finkoaren bertsio partizionatua (4), puntu-finkoaren iterazio estandarra baino eraginkorragoa dela [92]. Dena den, zenbakizko esperimentuetarako, Hairer-en [49] lanean bezala, puntu-finkoaren bertsio estandarraren emaitzak erakutsi ditugu. Puntu-finkoaren bertsio partizionatua aplikatu dugunean, antzeko emaitzak lortu ditugu baina urrats bakoitzean iterazio kopuru gutxiago behar izan ditu.

Energia errorearen jatorria.

Doitasun bikoitzeko IRK metodo simplektikoaren implementazioaren zenbakizko soluzioaren $\tilde{y}_n + e_n \approx y(t_n)$ ($n = 1, 2, \dots$) errorea, jatorri ezberdineko erroreen konbinazioa da:

1. Trunkatze errorea: hasierako baliodun problemaren soluzio zehatza $y(t_n)$ ($n = 1, 2, 3, \dots$), (5.9)-(5.10) metodoa aplikatuz (b_i, μ_{ij} koefiziente zehatzekin) lortutako zenbakizko soluzioa y_n ordezkatzerakoan eragindako errorea.
2. Iterazio errorea: praktikan, puntu-finkoaren (15) K iterazio finitu aplikatzen da, eta (5.9) sistemaren $L_{n,i}, Y_{n,i}$ ($i = 1, \dots, s$) soluzioa, $L_{n,i}^{[K]}, Y_{n,i}^{[K]}$ hurbilpenarekin ordezkatzen da. Hurbilpen honen araberako \bar{y}_{n+1} zenbakizko soluzioa kalkulatzen da,

$$\bar{y}_{n+1} = y_n + \sum_{i=1}^s L_{n,i}^{[K]}.$$

3. Funtzioa zehatza $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$, bere doitasun bikoitzeko bertsioaz $\tilde{f} : \mathbb{F}^d \rightarrow \mathbb{F}^d$ ordezkatzerakoan, sortutako errorea. Ordezkapen honek, bi eragin ditu: batetik, urrats gehienetan K iterazio finituetan puntu-finkora iritsiko gara, zeinek ekidin ezineko iterazio errorea sortuko duen; bestetik, \tilde{f} funtzioaren konputazioan sortatutako biribiltze erroreak.
4. IRK metodoaren koefiziente zehatzak $b_i, \mu_{ij} \in \mathbb{R}$, dagozkien doitasun bikoitzeko koefiziente $\tilde{b}_i, \tilde{\mu}_{ij} \in \mathbb{F}$ erabiltzeagatik eragindako errorea.
5. Algoritmoaren implementazioaren eragiketa aritmetikoak (\tilde{f} funtzioaren ebaluazioa egindakoaz gain), doitasun bikoitzean kalkulatzeagatik eragindako errorea.

Errore jatorri hauek energian duten eragina estimatzeko, honako algoritmoak implementatu ditugu:

A. Implementazio zehatza.

Trunkatze errorea estimatzeko, konputazio guztiak (ekuazio diferentzialaren funtzioaren ebaluazioa barne) doitasun laukoitzeko (128-bit) koma higikorreko aritmetikan kalkulatzen duen implementazioa aplikatuko dugu.

B. Implementazio superideala.

Zenbakizko integracio hau, iterazio errorea estimatzeko erabiliko dugu. Konputazio guzia doitasun laukoitzean egindako implementazioa baina geratze irizpidea, doitasun bikoitzean neurrtuko dugu,

$$\Delta^{[k]} = |\text{double}(Y^{[k]}) - \text{double}(Y^{[k-1]})|.$$

C. Implementazio ideala.

Ekuazio diferenzialaren eskuin aldeko funtzioren ebaluazioa izan ezik, beste eragiketa guztiak doitasun laukoitzean kalkulatzen dituen implementazioa da. Ekuazio diferenziala doitasun bikoitzean kalkulatzeak, eragiten duen errorea neurtzeko erabiliko dugu eta integracio hau hobetu ezin daitekeen integracioa konsideratuko dugu.

D. Implementazio sasi-idealak.

Doitasun bikoitzeko koefizienteak ($\tilde{\mu}_{ij}, \tilde{b}_i \in \mathbb{F}$) erabiltzeak, eragiten duen errorea neurtzeko integracioa da. Konputazio guzta doitasun laukoitzean kalkulatzen da baina doitasun bikoitzeko koefizienteen balioak erabiliz (hauei doitasun bikoitzeko koeficiente koadrifikatuak esaten diogu).

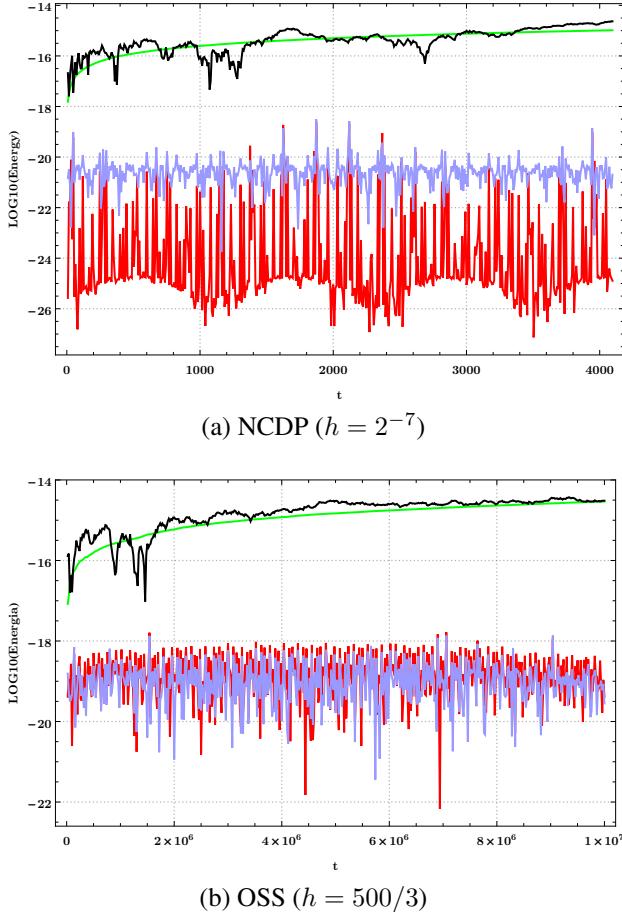
Hasierako baliodun problema bakoitzaren, goian azaldutako A–D implementazio bakoitzari dagokion energia errorearen eboluzioa [5.2.](#) irudian erakutsi dugu. Aukeratutako h urrats luzerarentzat, trunkatze errorea biribiltze errorearen azpitik dagoela baieztago dugu. Metodoaren doitasun bikoitzeko koefizienteak ($\tilde{b}_i, \tilde{\mu}_{ij} \in \mathbb{F}$) erabiltzeak, ez du eraginik biribiltze errorearen garapenean. Iterazio errorea, biribiltze errorearen oso antzeko da, eta energia errorearen drift lineala eragitea espero daiteke.

Errore azterketa estatistikoa.

Biribiltze erroreak eragiten duen zenbakizko erroreen azterketa fidagarriagoa egiteko, analisi estatistikoa aplikatu dugu (Hairer-en [[49](#)] lanean bezala). Problema bakoitzarentzat, hasierako balioaren osagai bakoitza ausaz perturbatutako ($\mathcal{O}(10^{-6})$ tamainako errore erlatiboarekin) $P = 1000$ integracio exekutatu ditugu eta emaitza hauen guztien batezbestekoan oinarritu gara, biribiltze errorearen azterketa zehatzagoa egiteko.

Puntu-finkoaren iterazioan oinarritutako 6-ataletako Gauss kolokazio metodoaren hiru implementazio konparatu ditugu:

1. Implementazio idealak: ekuazio diferenzialaren eskuin aldeko funtzioren ebaluazioa izan ezik, beste eragiketa guztiak doitasun laukoitzean (128-bit) kalkulatzen dituen implementazioa da. Implementazio hau FPIEA (fixed point iteration with exact arithmetic) izendatuko dugu.
2. Doitasun bikoitzeko gure implementazioa berria. Implementazioa hau, DP izendatu dugu.
3. Hairer-ek proposatutako implementazioa [[49](#)]. Zenbakizko esperimentuera rako, Hairer-en IRK metodoaren [Fortran kodea](#) exekutatu dugu.



5.2. Irudia: Algoritmo implementazio ezberdinatarako, energiaren errore erlatiboa esaka-la logaritmikoan irudikatu dugu: A-algoritmoa trunkatze errorearen estimazioa (gorriz), B-algoritmoa iterazio errorearen estimazioa (berdez), C-algoritmoa doitasun bikoitzean \tilde{f} funtziokoaren ebaluazioaren eraginaren estimazioa (beltzez), doitasun bikoitzeko $\tilde{b}_i, \tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak aplikatzearen estimazioa (urdinez). Hasierako baliodun problema bakoitzarentzat, irudi bana egin dugu: pendulu bikoitzaren problema ez-kaotikoa (a) eta kanpo-planeten problema (b).

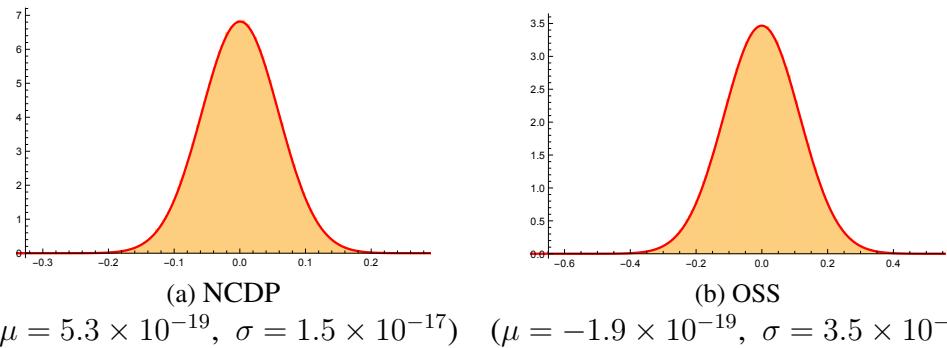
Batetik, DP implementazioaren biribiltze errorearen garapena, FPIEA implementazioaren (aritmetika zehatza) errorearekiko kualitatiboki antzekoa den eta magnitudean gertu dagoen ziurtatu nahi dugu. Bestalde, DP implementazioa, Haireren implementazioarekin konparatu nahi dugu.

Aipatutako hiru implementazioetarako, 5.1. taulan puntu-finkoa lortu den urratzen portzentaia eta iterazio batezbestekoa laburtu ditugu .

5.1. Taula: Puntu-finkoa lortu den urratsen portzentaia eta iterazio batezbestekoak, pendulu bikoitzaren problema ez-kaotikorako (NCDP), pendulu bikoitzaren problema kaotikorako (CDP), eta kanpo-planeten problemarako (OSS). Zutabetan, hiru implementazioak konparatu ditugu: FPIEA (ideala), DP (doitasun bikoitza) eta Hairer-en kodea.

	FPIEA		DP		Haire	
	%	#	%	#	%	#
NCDP	98.7	9.5	98.8	8.6	98.5	8.6
CDP	98.9	9.5	98.9	8.6	98.4	8.6
OSS	97.4	15.2	97.4	14.2	87.5	14.1

Energia differentzien banaketa.



5.3. Irudia: DP implementazioarekin lortutako KP energia differentzien histogramak, eta $N(\mu, \delta)$ banaketa normala, pendulu bikoitzaren problema ez-kaotikoarentzat (NCDP) eta kanpo-planeten problemarentzat (OSS). Ardatz horizontala 10^{15} balioarekin biderkatu dugu eta ardatz bertikalak, maiztasuna adierazten du

Integratzailearen implemtazioa ona bada, biribiltze erroreak eragindako energiaren errore lokala $H(y_n) - H(y_{n-1})$, ausazkoa izatea espero da. Hortaz, zenbakizko soluzioa m urratsero jasotzen dugula jakinik, energi differentzia $H(y_{km}) - H(y_{km-m})$ ausazkoa izatea espero da, μ batezbestekoa ($\mu = 0$ idealki) eta σ desbideratzea duen banaketa Gausiararekin. Ondorioz, metatutako energia differentzia,

$$H(y_{km}) - H(y_0),$$

$t_{mk} = t_0 + kmh$ uneetarako, $k^{1/2}\sigma = (t_{mk}/(mh))^{1/2}\sigma$ desbideratze estandarra duen ausazko ibilbide Gaussiar bat (*random walk*) jarraituko du. Honi, konputazio zientzian [43] Brouwer legea deritzote, Brouwer-ek [18] Kepler problemarentzat egin zuen zenbakizko integracioaren birbiltze errorearen azterketa gogoratu.

Ideia hau jarraituz, doitasun bikoitzeko (DP) implementazioan, m urrats arteko energiarenei differentziak,

$$\frac{H(y_{km}) - H(y_{km-m})}{H(y_0)},$$

banaketa Gaussiarra dagokion aztertuko dugu.

Integrazio tarteak $[t_0, t_{end}]$ eta P perturbatutako hasierako balioen kopurua bada, KP energia differentzienei balio ditugu, non $K = (t_{end} - t_0)/(mh)$ den. DP implementazioarekin lortutako KP energia differentzienei histograma eta $N(\mu, \sigma)$ banaketa normala irudikatu ditugu (non μ eta σ , balioei dagokien batezbestekoa eta desbideratze tipikoak diren). [5.3.](#) irudian, pendulu bikoitzaren problema ez-kaotikoari (NCDP), eta kango-planeten problemari (OSS) dagozkien histogramak, $N(\mu, \sigma)$ banaketa normalari oso ondo egokitzen zaiela ikus daiteke.

Energia errorearen batezbestekoaren eta desbideratze estandarraren eboluzioa

[5.4.](#) irudian, FPIEA, DP eta Hairer-en implementazioetarako, NCDP eta OSS problemen integrazioen energia errorearen batezbestekoa eta desbideratze estandarra irudikatu ditugu.

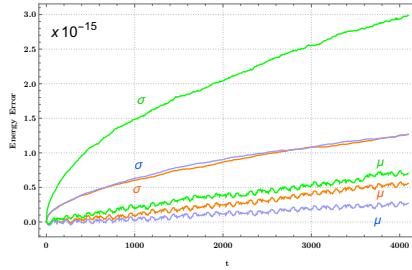
FPIEA, puntu-finkoaren iterazioan oinarritutako IRK implementazio optimoeña kontsideratu daiteke, f ebaluatzea, dagokion doitasun bikoitzeko \tilde{f} funtzioa aplikatuko dugula suposatzen badugu. Esperimentu hauetan, FPIEA implementazioaren geratze irizpidea DP implementazioarena baino gogorragoa erabili dugu: iterazioa geratu dugu $\Delta^{[k]} = 0$ delako edo ([5.15](#)) baldintza hamar iterazio jarraitetan bete delako. Era honetan, puntu-finkoa lortu ez den urratsetarako, iterazio errorea ekiditen saiatu gara.

[5.4.](#) irudiko zenbakizko esperimentuetan, DP implementazioaren energi errorearen batezbesteko eta desbideratze tipikoaren eboluzioa ia optimoa da (FPIEA implementazioarekiko gertu).

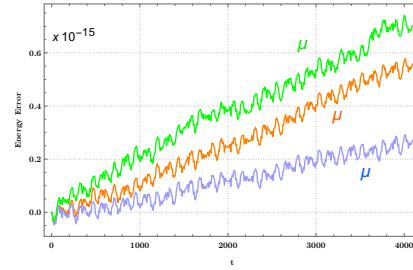
Puntu-finkoaren iterazioan oinarritutako IRK metodoen implementazioan, energi errorearen batezbestekoan drift lineal txiki bat (NCDP problemarentzat) ekidin ezinezkoa dela sinistuta gaude. Esperimentu hauen energi errorearen eboluzioak, [5.2.](#) irudian erakutsitako iterazio errorea, birbiltze errorearen gertu egoteagatik egindako iruzkinarekin kontsistenteari dira.

Energia drift-a, ez da IRK metodo simplektikoen berezko arazoa. [5.5.](#) irudian, Newton simplifikatuaren iterazioan oinarritutako IRK implementazioarekin NCDP problemaren aurreko esperimentua errepikatu dugu eta energi errorearen batezbestekoaren eboluzioan ez da drift linealik agertzen.

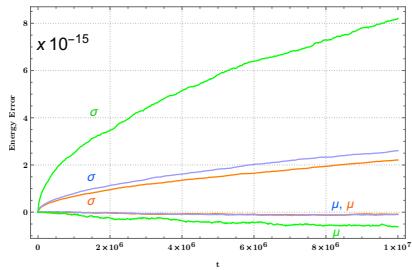
Zenbakizko esperimentuen atala amaitzeko, [5.6.](#) irudian, FPIEA, DP eta Hairer-en implementazioen integrazioen kokapen errorearen eboluzioa (batezbestekoa eta desbideratze estandarra) erakutsi ditugu. Emaitza hauek, DP implementazioa,



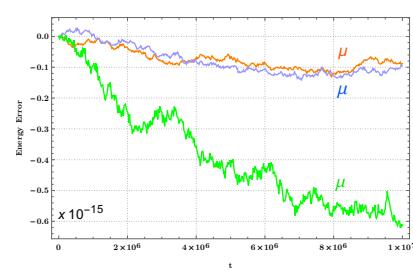
(a) NCDP: energi errorea



(b) NCDP: energi errorearen batezbestekoa

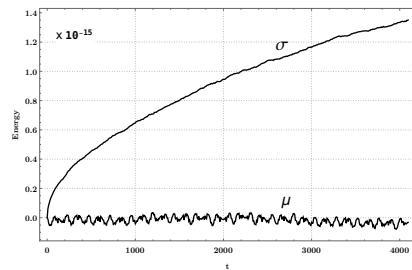


(c) OSS: energi errorea

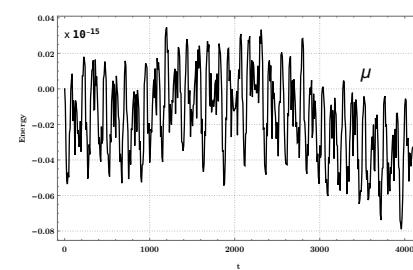


(d) OSS: energi errorearen batezbestekoa

5.4. Irudia: Energi errorearen batezbestekoa (μ) and desbideratze estandarra (σ) (ezkerrean) eta energi errorearen batezbestekoaren zehaztapena (eskubian), DP implementazioarentzat (urdinez), FPIEA implementazioarentzat (laranjaz), eta Hairer-en implementazioarentzat (berdez). Pendulu bikoitzaren problema ez-kaotikoa (a,b) eta kanpo-planeten problema (c,d)



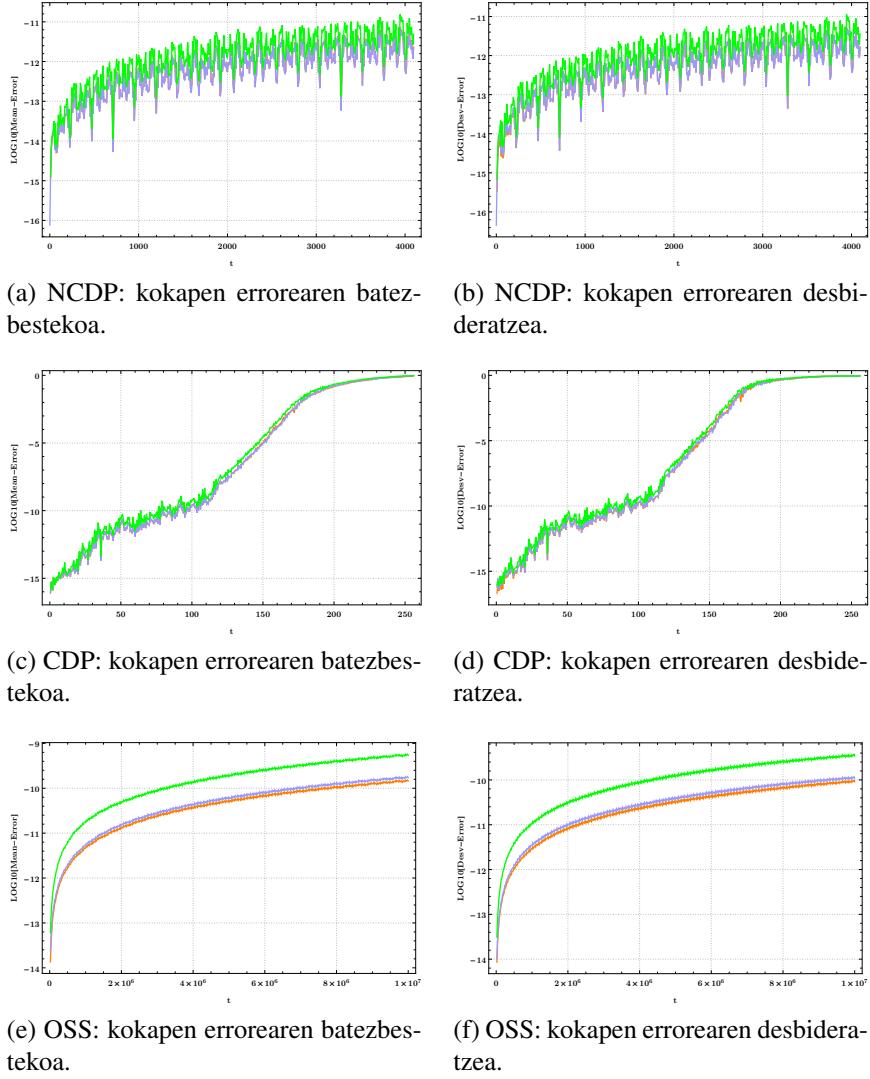
(a) NCDP: energi errorea.



(b) NCDP: energi errorearen batazbestekoa.

5.5. Irudia: Energi errorearen batezbestekoa (μ) eta desbideratze estandarra (σ), Newton sinplifikatuaren iterazioan oinarritutako IRK implementazioa aplikatuta

puntu-finkoaren iterazioan oinarritutako implementazio optimoaren oso gertu da-goenaren ideia indartu egiten dute.



5.6. Irudia: Kokapen errorearen batezbestekoa (ezkerrean) eta desbideratze estandarra (eskubian), DP implementazioarentzat (urdinez), FPIEA implementazioarentzat (laranjaz) eta Hairer-en implementazioarentzat (berdez): NCDP (a,b), CDP (c,d) eta OSS (e,f)

Biribiltze errorearen estimazioa

[5.3.](#) atalean deskribatutako biribiltze errorearen estimazioa kalkulatzeko teknika aplikatu dugu. [5.7.](#) irudian, hiru problemetarako (perturbatu gabeko hasierako balioa erabiliz) DP implementazioaren integrazioaren kokapen errorea eta kokapen errorearen estimazioa ($r = 3$ balioarekin) irudikatu ditugu. Era berean, hiru problemen hasierako balioen perturbatutako $P = 1000$, DP implementazioaren integrazioen kokapen errorearen batezbestekoa eta kokapen errorearen estimazioaren batezbestekoak konparatu ditugu. Emaitza hauek, proposatutako biribiltze errorearen estimazioa kalkulatzeko teknikaren erabilgarritasuna erakusten dutela uste dugu.

5.5. Laburpena.

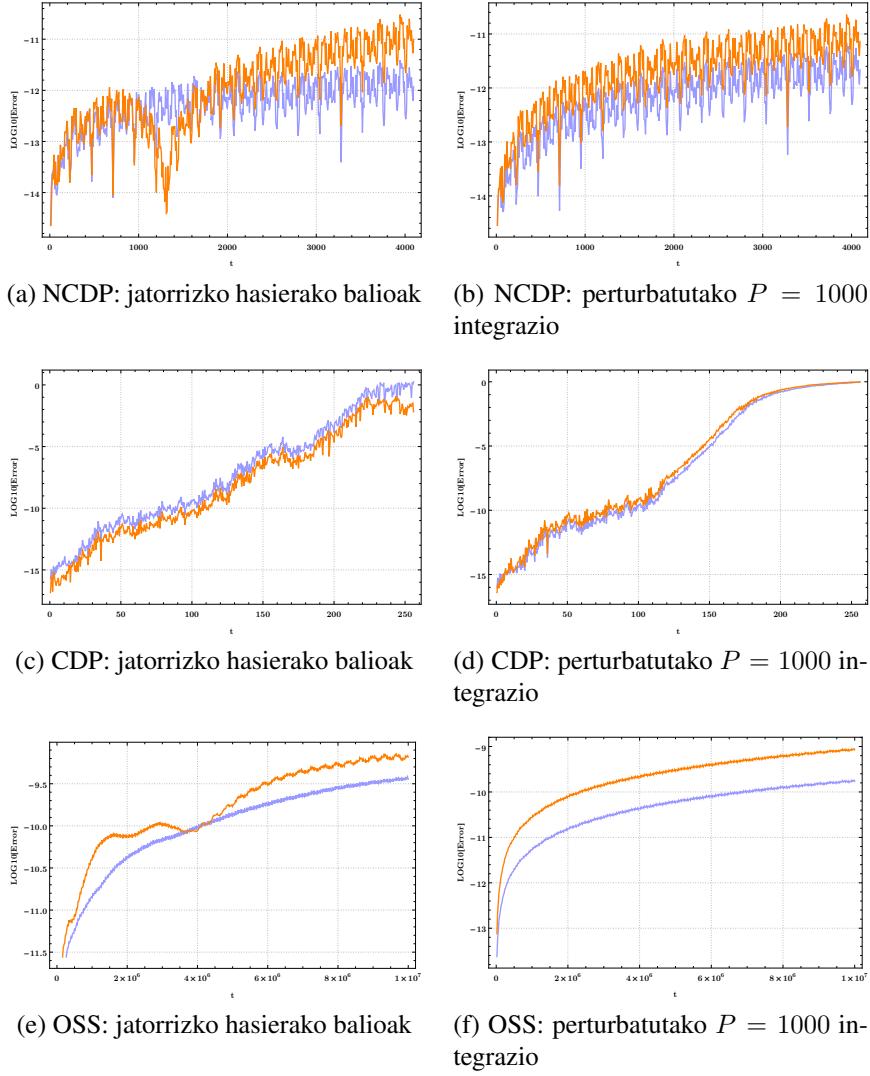
Runge-Kutta metodo implizituak (adibidez, Gauss nodoetan oinarritutako Runge-Kutta kolokazio metodoak) Hamiltondar sistemaren doitasun altuko integrazioetarako aproposak dira. Problema ez-zurrenetarako, puntu-finkoaren iterazioan oinarritutako implementazioa, Newton metodoaren iterazioan oinarritutako implementazioak baino eraginkorragoa da.

Implementazio berri honetan, biribiltze errorearen eragina txikitzeko ahalegin berezia egin dugu eta gainera, biribiltze errorearen estimazio kalkulatzeko aukera eman dugu. Gure implementazioak, puntu-finkoaren iterazioan oinarritutako implementazio onenaren birbiltze errorearen eboluzio antzekoa du eta zentzu honetan, gure implementazioa ia optimoa da. Zenbakizko esperimentuek idei hau baiezztatu dute.

Implementazioaren gakoetako bat, puntu-finkoaren iterazioaren geratze irizpide berria da. Geratze irizpidea, beste eremu batzuetan aplikagarria izan daitekeela pentsatzen dugu.

Bestalde, puntu-finkoaren iterazioaren implementazioaren zenbakizko esperimentu batzuetan, energi errorearen drift lineal txiki bat ekidin ezinezkoa agertu zaigu. Energi errorearen drift-a gainditzea garrantzitsua denenerako, Newtonen iterazioan oinarritutako implementazioa beharrezkoa izan daiteke.

Azkenik, aipatu nahi dugu, atal honen edukiak [Numerical Algorithms](#) aldizkarian publikatutako [7] artikuluan aurki daitezkeela eta implementazioaren [kodea](#) eskuragarri jarri dugula.



5.7. Irudia: Kokapen errorea (urdinez) eta kokapen errorearen estimazioa (laranjaz). Ezkerrean, perturbatu gabeko hasierako balioen integrazioak eta eskubian, hasierako balioen perturbatutako $P = 1000$ integrazioen batezbestekoak

6. Kapitulua

IRK: Newtonen Iterazioa.

6.1. Sarrera.

Atal honetan, Newtonen iterazioan oinarritutako IRK metodoen implementazio eraginkorra ikertuko dugu. Problema zurruna denean, puntu-finkoaren iterazio ez da eraginkorra eta Newtonen iterazioa aplikatu behar da. Gainera problema ez-zurruna izanik ere, Newton iterazioak interesgarriak izan daitezke; bereziki doitasun altuko (doitasun laukoitz) konputazioetan iterazio metodoaren konbergentzia ezaugarri onak direla-eta.

Ikusiko dugunez, d -dimentsioko ekuazio diferenzialen sistema, Newtonen iterazioan oinarritutako s -ataletako IRK metodoaren bidez integratzeko, era honetako ekuazio-sistema lineala iteratiboki askatu behar da

$$(I_d \otimes I_s - h A \otimes J) \in \mathbb{R}^{sd \times sd}, \quad (6.1)$$

non $A \in \mathbb{R}^{s \times s}$ Runge-Kutta metodoaren koefizienteak diren eta J matrizea, ataldean ebaluatutako Jacobiar matrizearen hurbilpen komuna den. Integrazioaren urrats bakoitzean, $sd \times sd$ tamainako ekuazio-sistema lineala askatu behar da.

Lan hauetan [21, 78, 15], (6.1) matrizearen egitura berezia aprobetxatz, ekuazio-sistema linealak modu eraginkorrean ebazteko proposamena egin zuten. Zehazki, ia blokeka diagonala den (6.1) matrizearen antzekoa da; era honetako $I_d - h\lambda_j J \in \mathbb{R}^{d \times d}$ ($j = 1, \dots, s$) s -bloke duen matrizea, bloke bat A matrizearen λ_j balio propio bakoitzeko. Normalean, ordena altuko IRK metodoaren A matrize koefizienteak, $[s/2]$ balio propio konplexu pareak ditu (s bakoitia denean, balio propio erreal bat gehituta).

Gure ekarpenean, sd -dimentsioko (6.1) ekuazio-sistema, $(s+1)d$ dimentsioko sistema gisa berridatziko dugu, eta $d \times d$ tamainako $[s/2] + 1$ matrizeen LU deskonposaketa (eta tamaina bereko matrize batzuen biderketa) kalkulatz, askatuko dugu. Tamaina txikiko matrizeen LU deskonposaketa azkarra denez, konputazionalki eraginkorra izatea espero dugu. Algoritmoa, IRK metodoa simetriko eta

simplektikoetarako (bi propietateak betetzen dituzten metodoetarako) garatu dugu. Dena den, bai IRK metodo ez-simetriko simplektikoetan, bai IRK metodo simetriko ez-simplektikoetan aplika daiteke.

Newtonen iterazio metodoaren bidez, $u \in \mathbb{R}^n$ eta $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ emanik, $F(u) = 0$ betetzen duen $u^{[*]}$ soluzioa aurkitu nahi dugu. Hasierako soluzioaren $u^{[0]}$ estimazioa emanik, Newtonen iterazio sinplifikatua era honetan definituko dugu (algoritmoa 20).

```

Hasieratu  $u^{[0]}$                                      (64 - bit);
 $M = LU(J)$                                      (32 - bit);
for ( $k=1,2,\dots$  konbergentzia lortu arte) do
     $F^{[k]} = F(u^{[k-1]})$                          (64 - bit);
    Askatu  $M \Delta u^{[k]} = -F^{[k]}$              (32 - bit);
     $u^{[k]} = u^{[k-1]} + \Delta u^{[k]}$            (64 - bit);
end
```

Algoritmoa 20: Newton sinplifikatua.

non $J \in \mathbb{R}^{n \times n}$, $J^{[k]} = J(u^{[k]})$ matrize Jacobiarraren hurbilpena den,

$$J(u^{[k]}) = (J_{ij}(u^{[k]}))_{i,j}^n \text{ non } J_{ij}(u^{[k]}) = \partial f_i / \partial u_j(u^{[k]}), \quad 1 \leq i, j \leq n.$$

Newton metodoaren eragiketa konplexuenak doitasun txikiagoan kalkula daitezke [9] eta honek, konputazionalki abantaila interesgarria suposatzen du. 20 algoritmoaren eskubi aldean, doitasun bikoitzeko (64-bit) implementazioa balitz, eragiketa bakoitzaren doitasuna zehaztu dugu: Jacobiarraren balioztapena eta algebra linealeko eragiketak, doitasun arruntean (32-bit) kalkulatu daitezke.

IRK Newtonen iterazioaren azterketa, era honetan egituratu dugu. Lehenengo, (6.2.) atalean, Newtonen iterazio estandarraren azalpenak eman ditugu eta notazioa finkatu dugu. (6.3.) atalean, Newtonen iterazioen ekuazio-sistema modu eraginkorrean askatzeko teknika deskribatu dugu. Hurrengo, (6.4.)-(6.5.) ataletan, Runge-Kutta formulazio berriarekin aplikatzeko zehaztasunak eman ditugu. (6.6.) atalean, Newtonen iterazioan oinarritutako IRK metodoaren implementazio berria aurkeztu dugu. Azkenik, (6.7.) atalean, implementazio berriarekin egindako zenbakizko esperimentuen emaitzak eman ditugu.

6.2. IRK-Newton estandarra.

Demagun honako hasierako baliodun problema,

$$\dot{y} = f(t, y), \quad y(t_0) = y_0, \tag{6.2}$$

non $y_0 \in \mathbb{R}^d$ eta $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$ diren.

Denbora diskretizazioa $t_0 < t_1 < t_2 < \dots$ emanik, (6.2) hasierako baliodun problemaren $y(t)$ soluzioaren $y_n \approx y(t_n)$, ($n = 1, 2, \dots$) zenbakizko soluzioa, integrazio metodo bat aplikatuz

$$y_{n+1} = \Phi(y_n, t_n, t_{n+1} - t_n), \quad (6.3)$$

lortuko dugu, non $\Phi : \mathbb{R}^{d+2} \rightarrow \mathbb{R}$ den.

S-ataletako IRK metodoaren kasuan, a_{ij} , b_i , eta c_i ($1 \leq i, j \leq s$) koefizienteek definitzen dute Φ integrazio metodoa,

$$\Phi(y, t, h) = y + h \sum_{i=1}^s b_i f(t + c_i h, Y_i) , \quad (6.4)$$

non $c_i = \sum_{j=1}^s a_{ij}$ izan ohi da eta Y_i atalak era honetan implizituki definitzen diren,

$$Y_i = y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j) \quad i = 1, \dots, s. \quad (6.5)$$

$Y_i \in \mathbb{R}^d$, $i = 1, \dots, s$ ezezagunen eta sd tamainako ekuazio-sistema ez linea-
la (6.5) askatzeko, iterazio metodo bat aplikatu behar dugu. Iterazio metodo sin-
pleena, puntu-finkoaren iterazioa da. Problema zurruna denean, puntu-finkoaren
iterazio ez da egokia eta orduan, Newtonen iterazioa aplikatu beharra dago. Pro-
blema ez-zurruna izanik ere, Newton iterazioak interesgarriak izan daitezke; be-
reziki doitasun altuko (doitasun laukoitzta) konputazioetan, doitasun ezberdinak
nahasten [9] dituen teknikari esker (algebra lienaleko eragiketak eta Jacobiarraren
balioztapena doitasun txikiagoan kalkulatza baitago).

Edozein kasutan, Newtonen iterazio bakoitzean, Jacobiarraren s balioztapen
eta $sd \times sd$ sistemaren LU deskonposaketa kalkulatu behar direnez, aldaera kon-
putazionalki merkeagoak aplikatzen dira.

Newton iterazioa.

Newtonen iterazioa, (6.5) ekuazio implizituko Y_i ($i = 1, \dots, s$) atalentzako $Y_i^{[k]}$
 $k = 1, 2, \dots$ hurbilpenak kalkulatzeko algoritmoa, modu honetan definituko du-
gu,

$$1) \quad r_i^{[k]} := -Y_i^{[k-1]} + y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j^{[k-1]}), \quad i = 1, \dots, s, \quad (6.6)$$

2) Askatu $\Delta Y_i^{[k]}$,

$$\Delta Y_i^{[k]} - h \sum_{j=1}^s a_{ij} J_j^{[k]} \Delta Y_j^{[k]} = r_i^{[k]} \quad i = 1, \dots, s, \quad (6.7)$$

$$\text{non } J_i^{[k]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[k]}) \quad i = 1, \dots, s,$$

$$3) \quad Y_i^{[k]} := Y_i^{[k-1]} + \Delta Y_i^{[k]}, \quad i = 1, \dots, s, \quad (6.8)$$

Iterazio bakoitzeko, $J_i^{[k]}$ Jacobiarren s ebaluazio eta $sd \times sd$ tamainako ekuazio-sistemaren LU deskonposaketa kalkulatu behar dugu. Eragiketa hauek konplexuak dira eta horregatik, Newton osoaren inplementazioa, konputazionalki garestia da. Aukera eraginkorragoen artean, bi aipatuko ditugu:

- Newton simplifikatuaren iterazioak aplikatzea. Aukera honetan, (6.7) ekua-zioaren $J_i^{[k]}$ Jacobiar matrizeak, $J_i^{[0]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[0]})$ matrizeekin ordezkatuko ditugu. Urrats bakoitzean, LU deskonposaketa behin bakarrik kalkulatu behar dugu.

$$\Delta Y_i^{[k]} - h \sum_{j=1}^s a_{ij} J_j^{[0]} \Delta Y_j^{[k]} = r_i^{[k]} \quad i = 1, \dots, s.$$

Problema zurruna denean, atalen hasieraketa $Y_i = y_n$, $i = 1, \dots, s$ erabil ohi da, eta orduan, $J_i^{[0]} = J := \frac{\partial f}{\partial y}(y)$, $i = 1, \dots, s$ ordezkatuko dugu eta ekuazio-sistema lineala era honetan sinplifikatzen zaigu,

$$(I_s \otimes I_d - h A \otimes J) \Delta Y^{[k]} = r^{[k]}.$$

- Jatorrizko Newtonen iterazioaren (6.7) ekuazio-sistema, matrize honen,

$$(I_s \otimes I_d - h A \otimes J) \quad (6.9)$$

alderantzizkoarekin aurre-baldintzatuta, iterazio metodo [89] baten bidez ebaaztea. Praktikan, (6.7) ekuazio-sistemaren soluzioaren hurbilpen bat lortuko dugu, eta metodo hauek, Sasi-Newton (inexact Newton) izenarekin ezagutzen dira.

Goiko bi aukeretan, era honetako ekuazio-sistemak askatu behar ditugu,

$$(I_d \otimes I_d - h A \otimes J) \Delta Y = r \quad (6.10)$$

emandako $r \in R^{sd}$ izanik. Ekuazio-sistema $sd \times sd$ tamainako matrize osoaren LU deskonposaketa eginez ebatzi daiteke baina modu eraginkorragoan egiteko bideak aztertuko ditugu.

Modu estandarrean [21, 78, 15], A matrizearen diagonalizazioa egiten da $\Lambda = S^{-1}AS = \text{diag}(\lambda_1, \dots, \lambda_s)$ eta era honetako matrizearen,

$$I_s \otimes I_d - h \Lambda \otimes J = (S^{-1} \otimes I_d) (I_s \otimes I_d - h A \otimes J) (S \otimes I_d)$$

LU deskonposaketa kalkulatuko da. Teknika honetan, A matrizearen balio propio erreals (edo balio propio konplexu) bakoitzari dagokion $d \times d$ matrize errealen (edo konplexuen) LU deskonposaketak kalkulatu behar dira.

Beste autore batzuk [19, 59], (6.9) ekuazio-sistema askatzeko, iterazio metodo baten bidez matrize alderantzizko honekin,

$$I_d \otimes I_s - h \bar{A} \otimes J, \quad (6.11)$$

non $\bar{A} \in \mathbb{R}^{s \times s}$ (LU deskonposaketa modu eraginkorragoan askatzeko aukeratua) aurre-baldintzatuta ebaaztea proposatzen dute.

Newton simplifikatuaren iterazioa.

Newton simplifikatuaren iterazioan, $J_i^{[k]}$ Jacobiarak, $J_i^{[0]} = \partial f / \partial y(t + c_i h, Y_i^{[0]})$ $i = 1, \dots, s$ Jacobiarrez ordezkatzen dira eta orduan, askatu beharreko ekuazio-sistema honakoa da,

$$\Delta Y_i^{[k]} - h \sum_{j=1}^s a_{ij} J_j^{[0]} \Delta Y_j^{[k]} = r_i^{[k]}, \quad i = 1, \dots, s, \quad (6.12)$$

non $J_i^{[0]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[0]}), \quad i = 1, \dots, s$ den.

Lehen sinplifikazio honetan, integrazioaren urrats bakoitzeko, $J_i^{[0]}$ Jacobia-rraren s-ebaluazio eta $sd \times sd$ tamainako matrizearen LU deskonposaketa behin bakarrik kalkulatu behar dugu. Modu baliokidean, ekuazio lineala notazio matriziala erabiliz laburtu daiteke,

$$\left(I_s \otimes I_d - h \begin{bmatrix} a_{11} J_1^{[0]} & \dots & a_{1s} J_s^{[0]} \\ a_{21} J_1^{[0]} & \dots & a_{2s} J_s^{[0]} \\ \vdots & \ddots & \vdots \\ a_{s1} J_1^{[0]} & \dots & a_{ss} J_s^{[0]} \end{bmatrix} \right) \Delta Y^{[k]} = r^{[k]}.$$

non,

$$Y^{[k]} = \begin{bmatrix} Y_1^{[k]} \\ \vdots \\ Y_s^{[k]} \end{bmatrix} \in \mathbb{R}^{sd}, \quad r^{[k]} = \begin{bmatrix} r_1^{[k]} \\ \vdots \\ r_s^{[k]} \end{bmatrix} \in \mathbb{R}^{sd},$$

$$J_{is}(y) = \left(\frac{\partial f^i}{\partial y^j}(y) \right)_{i,j}^d = \begin{bmatrix} \frac{\partial f^1}{\partial y^1} & \cdots & \frac{\partial f^1}{\partial y^d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f^d}{\partial y^1} & \cdots & \frac{\partial f^d}{\partial y^d} \end{bmatrix} \in \mathbb{R}^{d \times d}, \quad is = 1, \dots, s.$$

Newton super-simplifikatuaren iterazioa.

Bigarren simplifikazioa bat aplika daiteke, $J_i^{[0]} = \partial f / \partial y (t + c_i h, Y_i^{[0]})$, $i = 1, \dots, s$ matrizeak, $J_i^{[0]} \approx J$ hurbilpen bakarrarekin ordezkatuz. Era honetako ekuazio-sistema lortuko dugu,

$$(I_s \otimes I_d - h A \otimes J) \Delta Y^{[k]} = r^{[k]}. \quad (6.13)$$

non I_s, I_d identitateak eta $A = (a_{ij})_{i,j}^s$ koefizienteen matrizeak diren.

Nahiz eta atalen hasieraketarako, $Y_i^{[0]} = y_n$, ($i = 1, \dots, s$) ez den teknika aplikatu, askotan gertatzen da (6.12) sistema lineala, (6.13) sistemarekin ordezkatzea. Aukera egokia da [105], $J = \frac{\partial f}{\partial y}(t + \bar{c} h, \bar{y})$ aplikatzea, non $\bar{c} = \frac{1}{s} \sum_{i=1}^s c_i$ (metodo simetrikoan $\bar{c} = \frac{1}{2}$ da) eta $\bar{y} = \frac{1}{s} \sum_{i=1}^s Y_i^{[0]}$. Maiz, $\partial f / \partial y$ konputazio-nalki merkeagoa den hurbilketa batez ordezkatzea, nahikoa izango da.

Newton iterazio bertsio honi super-simplifikatua deitu diogu. Iterazio bakoitzean f funtziaren s ebaluazio eta sd dimentsioko ekuazio-sistema lineala askatu behar da. $(I_s \otimes I_d - h A \otimes J)$ matriza iterazio guztietarako berdina da, bere LU deskonposaketa behin bakarrik egin behar da baina konputazionalki garestia da [21, 46]. Hau da aljebra linealari dagokion eragiketen konplexutasuna,

$$\begin{aligned} \text{LU deskonposaketa, } & 2s^3d^3/3 + \mathcal{O}(d^2), \\ \text{Back substitution, } & 2s^2d^2 + \mathcal{O}(d). \end{aligned}$$

Jarraian, Newton super-simplifikatuaren implementazioaren algoritmo orokorra laburtu dugu (Algoritmoa 21).

Algoritmoa.

```

 $\tilde{y}_0 = fl(y_0);$ 
 $e_0 = fl(y_0 - \tilde{y}_0);$ 
for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $k = 0;$ 
    Hasieratu  $Y_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
     $J = \frac{\partial f}{\partial y}(t + h/2, y_n);$ 
     $M = LU(I_s \otimes I_d - h A \otimes J);$ 
    while (not konbergentzia) do
         $k = k + 1;$ 
         $r_i^{[k]} = -Y_{n,i}^{[k-1]} + y_n + (e_n + h \sum_{j=1}^s a_{ij}f(t + c_j h, Y_{n,j}^{[k-1]}));$ 
        Askatu ( $M \Delta Y_n^{[k]} = r^{[k]}$ );
         $Y_n^{[k]} = Y_n^{[k-1]} + \Delta Y_n^{[k]};$ 
        konbergentzia  $\leftarrow$  GeratzeErizpidea( $Y_n^{[k]}$ ,  $Y_n^{[k-1]}$ ,  $\Delta_{min}$ );
    end
    if ( $\exists j$  non  $\Delta_j^{[K]} \neq 0$ ) then
        if ( $NormalizeDistance(Y_n^{[k]}, Y_n^{[k-1]}) > 1$ ) then
            fail convergence;
        end
    end
     $(\tilde{y}_{n+1}, e_{n+1}) \leftarrow$  baturakonpensatua( $y_n, e_n, Y_n^{[k]}$ );
end

```

Algoritmoa 21: IRK (Newton super-simplifikatua).

6.3. IRK-Newton eraginkorra.

Ekuazio-sistema.

Atal honetan, honako ekuazio lineala modu eraginkorrean askatzeko implementazioa proposatuko,

$$(I_s \otimes I_d - h A \otimes J) \Delta Y = r, \quad (6.14)$$

non $J \in \mathbb{R}^{d \times d}$ eta $r \in \mathbb{R}^{s \times d}$ emandako matrizeak izanik.

S -ataletako IRK metodoa, Newton iterazioaren bidez d -dimentsioko ekuazio differentzial sistemari aplikatzeko, urrats bakoitzean $sd \times sd$ tamainako hainbat ekuazio-sistema (iterazio bakoitzeko bat) askatu behar dira. Atal honetan, jatorrizko sd -dimentsioko ekuazio-sistema, $(s + 1)d$ dimentsioko ekuazio-sistema baliokide moduan berridatziko dugu. Ekuazio-sistema baliokide hau, $d \times d$ tamainako $[s/2] + 1$ matrize errealen LU deskonposaketa bidez askatuko dugu. Tamaina

txikiko matrizeen LU deskonposaketa azkarra denez, konputazionalki eraginkorragoa izatea espero dugu.

Gauss nodoetan oinarritutako Runge-Kutta kolokazio metodoak, simplektikoa eta simetrikoak [92] dira.

1. Simplektikoa.

Runge-Kutta metodoa simplektikoa izateko baldintza,

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (6.15)$$

2. Simetrikoa.

Runge-Kutta metodoa simetrikoa izateko baldintza,

$$\begin{aligned} b_{s+1-i} &= b_i, \quad c_{s+1-i} = 1 - c_i, \quad 1 \leq i, j \leq s, \\ b_j &= a_{s+1-i, s+1-j} + a_{i,j}, \quad 1 \leq i, j \leq s. \end{aligned} \quad (6.16)$$

Implementazio eraginkorra garatzeko, goiko bi propietate hauetan oinarrituko gara. S-ataletako IRK metodoaren formulazioa (6.4), modu baliokide honetan berridatzi daiteke,

$$\Phi(y, t, h) := y + z, \quad (6.17)$$

non $Y_i \in \mathbb{R}^d$ atalak eta $z \in \mathbb{R}^d$ gehikuntza, implizituki era honetan definitzen diren,

$$Y_i = y + \frac{z}{2} + h \sum_{j=1}^s \bar{a}_{ij} f(t + c_j h, Y_j) \quad i = 1, \dots, s, \quad (6.18)$$

$$z = h \sum_{i=1}^s b_i f(t + c_i h, Y_i), \quad (6.19)$$

non

$$\bar{a}_{ij} = a_{ij} - \frac{b_j}{2}, \quad 1 \leq i, j \leq s \text{ den.} \quad (6.20)$$

Ekuazio implizitua ebatzeko Newton iterazio sinplifikatua aplikatzen badugu, $(s+1) \times d$ dimentsioko ekuazio-sistema askatu behar dugu,

$$\begin{aligned} (I_s \otimes I_d - h \bar{A} \otimes J) \Delta Y - \frac{1}{2} (e_s \otimes I_d) \Delta z &= r, \\ (-h e_s^T B \otimes J) \Delta Y + \Delta z &= 0, \end{aligned} \quad (6.21)$$

non $e_s = (1, \dots, 1)^T \in \mathbb{R}^s$, eta $\bar{A} = (\bar{a}_{ij})_{i,j=1}^s$ den. Argi dago, $(\Delta Y, \Delta z)$ (6.21) ekuazio-sistemaren soluzioa bada, orduan ΔY gure jatorrizko (6.14) ekuazio-sistemaren soluzioa dela.

Ekuazio-sistemaren adierazpen matriziala lagungarria izan daiteke,

$$\begin{bmatrix} -I_d/2 & & & \\ -I_d/2 & \ddots & & \\ & \vdots & \ddots & \\ I_s \otimes I_d - h \bar{A} \otimes J & & & -I_d/2 \\ -hb_1 J & -hb_2 J & \cdots & -hb_s J & I_d \end{bmatrix} \begin{bmatrix} \Delta Y_1 \\ \Delta Y_2 \\ \vdots \\ \Delta Y_s \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_s \\ 0 \end{bmatrix}$$

Aldi berean, koefiziente notazioa berri hau finkatuta,

$$\bar{c}_i = c_i - \frac{1}{2}, \quad \bar{a}_{ij} = a_{ij} - \frac{b_j}{2}, \quad 1 \leq i, j \leq s,$$

dagokion propietate simplektikoa (6.15) eta simetrikoa (6.16) berridatziko ditugu,

1. Simplektikoa.

Runge-Kutta metodoa simplektikoa da,

$$(B\bar{A}) \text{ antisimetrikoa bada,} \quad (6.22)$$

non $\bar{A} = (\bar{a}_{ij})_{i,j=1}^s$ eta $B, (b_1, b_2, \dots, b_s)$ balioen matrize diagonala diren.

2. Simetrikoa.

Runge-Kutta metodoa simetrikoa izango da, koefizienteek baldintza hauek betetzen dituztenean,

$$\begin{aligned} b_{s+1-i} &= b_i, & \bar{c}_{s+1-i} &= -\bar{c}_i, & 1 \leq i \leq s, \\ \bar{a}_{s+1-i, s+1-j} &= -\bar{a}_{ij}, & 1 \leq i, j \leq s. \end{aligned} \quad (6.23)$$

Implementazio berrian, matrizeen dimentsioak zehazteko, parametro berri hauetan oinarrituko gara, $m = [(s+1)/2]$, eta $s-m = [s/2]$. Metodoaren s -atalen kopurua bikoiti ala bakoiti izan, bi kasu bereiziko ditugu:

- s bikoitia (Adibidea $s = 6 \rightarrow m = 3, s-m = 3$).
- s bakoitia (Adibidea $s = 7 \rightarrow m = 4, s-m = 3$).

IRK metodo simplektikoen garapena.

Lehenengo, IRK metodo simplektikoak konsideratuko ditugu. $(B\bar{A})$ antisimetrikoa bada, orduan $B^{1/2}\bar{A}B^{-1/2}$ antisimetrikoa da. Era berean, honek \bar{A} diagonalizagarria dela eta balio propio irudikari puruak dituela, suposatzen du. Beraz, Q , $s \times s$ tamainako matrize ortogonala existitzen da,

$$Q^{-1}\bar{A}Q = \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \quad (6.24)$$

non D , balio erreal positiboen matrize diagonala eta $m \times (s-m)$ tamainako den. (6.21) ekuazio-sistemari, aldagai aldaketa hau aplikatzuz,

$$\Delta Y = (Q \otimes I_d) W,$$

honako ekuazio-sistema baliokidea lortuko dugu (garapenean (6.24) erabili dugu),

$$\begin{pmatrix} I_m \otimes I_d & -h D \otimes J \\ h D^T \otimes J & I_{s-m} \otimes I_d \end{pmatrix} W - \frac{1}{2} (Q^{-1} e_s \otimes I_d) \Delta z = (Q^{-1} \otimes I_d) r, \quad (6.25)$$

$$-h (e_s^T B Q \otimes J) W + \Delta z = 0,$$

Eranskinean (??), ekuazio baliokideak lortzeko eman diren urratsen zehaztanenak eman ditugu.

Sistemaren (6.25) bloke bakanen egiturari esker, LU deskonposaketaren konputazioa, $d \times d$ tamainako matrizeen biderkaduren eta $[s/2] + 1$ matrize errealeen ($d \times d$) LU deskonposaketeten bidez kalkulatuko dugu,

1. $I_d + h^2 \sigma_i^2 J^2 \in \mathbb{R}^{d \times d}$, $i = 1, \dots, [s/2]$ matrizeak,
non $\sigma_1, \dots, \sigma_{[s/2]} \geq 0$, D matrizearen diagonaleko balioak diren.
2. Aurreko matrizeetatik lortutako $d \times d$ dimentsioko matrizea.

IRK metodo simetriko simplektikoen garapena.

Atal honetan, (6.15) propietate simplektikoaz gain, (6.16) simetria propietate ere betetzen dituzten IRK metodoak konsideratuko ditugu. Lehenengo, garapenean erabiliko ditugun matrize laguntzaileak definituko ditugu.

1. P matrizea.

Kontsideratu $P = (P_1 \ P_2) \in \mathbb{R}^{s \times s}$ matrize ortogonala, non $P_1 \in \mathbb{R}^{s \times m}$ eta $P_2 \in \mathbb{R}^{s \times (s-m)}$ diren. Era honetan definituko dugu, $x = (x_1, \dots, x_s)^T \in$

\mathbb{R}^s , $P_1^T x = (y_1, \dots, y_m)^T$, eta $P_2^T x = (y_{m+1}, \dots, y_s)^T$ non,

$$\begin{aligned} y_i &= \frac{\sqrt{2}}{2}(x_{s+1-i} + x_i), \quad i = 1, \dots, [s/2], \\ y_i &= \frac{\sqrt{2}}{2}(x_{s+1-i} - x_i), \quad i = m+1, \dots, s, \\ y_m &= x_m, \quad s \text{ bakoitia bada.} \end{aligned}$$

2. K matrizea.

Batetik, (6.23) simetria propietateak, $P_i^T B^{\frac{1}{2}} \bar{A} B^{-\frac{1}{2}} P_i = 0$, $i = 1, 2$ dela eta bestetik, propietate simplektikoak $B^{1/2} \bar{A} B^{-1/2}$ antisimetriko dela ziurtatzen dutenez, \bar{A} matriza honako matrizearen antzekoa dela ondorioztatu daiteke,

$$P^T B^{\frac{1}{2}} \bar{A} B^{-\frac{1}{2}} P = \begin{pmatrix} 0 & K \\ -K^T & 0 \end{pmatrix} \quad (6.26)$$

non $K = P_1^T B^{\frac{1}{2}} \bar{A} B^{-\frac{1}{2}} P_2 \in \mathbb{R}^{m \times (s-m)}$ den.

3. D matrizea.

$K = UDV^T$ balio singulararen deskonposaketa izanik, non $U \in \mathbb{R}^{m \times m}$, eta $V \in \mathbb{R}^{(s-m) \times (s-m)}$ matrize ortonormalak diren eta $D \in \mathbb{R}^{m \times (s-m)}$, K matrizearen balio singularren $(\sigma_1, \dots, \sigma_{s-m})$ matrize diagonala den,

$$D = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \sigma_{s-m} \end{pmatrix}, \quad D = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \sigma_{s-m} \\ 0 & 0 & \dots & 0 \end{pmatrix}. \quad (6.27)$$

s bikoitia bada, D matriza ezkerrean eta s bakoitia bada, D matriza esku-bian ($\sigma_m = 0$) irudikatu dugu.

4. Q matrizea.

Simetrikoak (6.24) izategatik, berdintza hauek baiezta daitezke,

$$Q = (Q_1 \ Q_2) = B^{-1/2} (P_1 \ P_2) \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} = B^{-1/2} (P_1 U \ P_2 V), \quad (6.28)$$

$$Q^{-1} = Q^T B. \quad (6.29)$$

Matrizearen dimentsioak laburtuz, $Q = (Q_1 \ Q_2) \in \mathbb{R}^{s \times s}$, $Q_1 \in \mathbb{R}^{s \times m}$ eta $Q_2 \in \mathbb{R}^{s \times (s-m)}$ ditugu.

(6.21) ekuazio-sistemari, aldagai aldaketa hau aplikatuz,

$$\Delta Y = (Q \otimes I_d)W = (Q_1 \otimes I_d)W' + (Q_2 \otimes I_d)W'', \quad (6.30)$$

non $W = \begin{pmatrix} W' \\ W'' \end{pmatrix}$, $W' \in \mathbb{R}^{m \times d}$, $W'' \in \mathbb{R}^{(s-m) \times d}$ diren,

eta metodoa simetrikoa (6.23) izatearen lehen baldintzagatik, $e_s^T B P_2 = 0$ eta $e_s^T B Q_2 = e_s^T B P_2 V = 0$ berdintasunak aplikatuz, honako ekuazio-sistema balio-kidea lortuko dugu,

$$\begin{aligned} W' - h(D \otimes J)W'' - \frac{1}{2}(Q_1^T B e_s \otimes I_d)\Delta z &= (Q_1^T B \otimes I_d)r, \\ h(D^T \otimes J)W' + W'' &= (Q_2^T B \otimes I_d)r, \\ -h(e_s^T B Q_1 \otimes J)W' + \Delta z &= 0. \end{aligned} \quad (6.31)$$

Eranskinean (??), ekuazioak lortzeko urratsen zehaztapenak eman ditugu.

Matrizearen egitura. $S = 6$ ataletako IRK metodoari dagokion ekuazio-sistemaren matrizearen egitura berezia ikus daiteke (6.32). Aldagai aldaketarekin lortutako ekuazio-sistema, blokeka diagonala da eta hau aprobetxatz, Newton iterazioaren implementazio eraginkorra lortuko dugu.

$$\left[\begin{array}{ccc|ccc} I_d & & & -h\sigma_1 J & & -\frac{\alpha_1}{2} I_d \\ & I_d & & -h\sigma_2 J & & -\frac{\alpha_2}{2} I_d \\ & & I_d & -h\sigma_3 J & & -\frac{\alpha_3}{2} I_d \\ \hline h\sigma_1 J & & & I_d & & 0 \\ h\sigma_2 J & & & I_d & & 0 \\ h\sigma_3 J & & & I_d & & 0 \\ \hline -h\alpha_1 J & -h\alpha_2 J & -h\alpha_3 J & 0 & 0 & 0 \end{array} \right] = \begin{bmatrix} W' \\ \hline W'' \\ \hline \Delta z \end{bmatrix} \begin{bmatrix} R' \\ \hline R'' \\ \hline 0 \end{bmatrix} \quad (6.32)$$

non

$$\begin{bmatrix} R' = (Q_1^T B^{1/2} \otimes I_d)r \\ R'' = (Q_2^T B^{1/2} \otimes I_d)r \end{bmatrix}, \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{pmatrix} = Q_1^T B e_s.$$

Jarraian, ekuazio-sistemaren ezezagunak ($\Delta z, W', W''$) askatzeko aplikatuko ditugun espresioak laburtuko ditugu.

W'' kalkulatzeko ekuazioak. Sistemaren (6.31) bigarren ekuaziotik W'' askatu,

$$W'' = -h(D^T \otimes J)W' + (Q_2^T B \otimes I_d)r. \quad (6.33)$$

W' kalkulatzeko ekuazioak. W'' lehen ekuazioan (6.31) ordezkatz, honako ekuazio-sistema lortuko dugu,

$$\begin{aligned} (I_m \otimes I_d + h^2 DD^T \otimes J^2) W' - \frac{1}{2}(Q_1^T B e_s \otimes I_d) \Delta z &= R, \\ -h (e_s^T B Q_1 \otimes J) W' + \Delta z &= 0, \\ \text{non } R &= (Q_1^T B \otimes I_d) r + h (DQ_2^T B \otimes J) r \in \mathbb{R}^{md}. \end{aligned} \quad (6.34)$$

Goiko ekuazio-sistema honako notazioaren arabera,

$$R = \begin{bmatrix} R_1 \\ \vdots \\ R_m \end{bmatrix}, \quad W' = \begin{bmatrix} W_1 \\ \vdots \\ W_m \end{bmatrix}, \quad R_i, W_i \in \mathbb{R}^d, \quad i = 1, \dots, m$$

era honetan berridatziko dugu,

$$(I_d + h^2 \sigma_i^2 J^2) W_i - \frac{\alpha_i}{2} \Delta z = R_i, \quad i = 1, \dots, m, \quad (6.35)$$

$$-h J \sum_{i=1}^m \alpha_i W_i + \Delta z = 0, \quad (6.36)$$

non,

$$\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{pmatrix} = Q_1^T B e_s,$$

eta $\sigma_1 \geq \dots \geq \sigma_{s/2}$, K matrizearen balio singularrak diren; s bakoitia denean $\sigma_m = 0$ dela gogoratu (6.27).

Δz kalkulatzeko ekuazioak. Aurreko (6.35) ekuaziotik, W_i askatuz,

$$W_i = (I_d + h^2 \sigma_i^2 J^2)^{-1} (R_i + \frac{\alpha_i}{2} \Delta z),$$

eta (6.36) ekuazioan ordezkatz, $\Delta z \in \mathbb{R}^d$ askatzeko ekuazioak lortuko ditugu,

$$M \Delta z = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i, \quad (6.37)$$

non

$$M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1} \in \mathbb{R}^{d \times d}. \quad (6.38)$$

Ezezagunak askatzeko laburpena. Sistemaren ezezagunak askatzeko ekuazioak eta ordena laburtuko dugu: lehen, $\Delta z \in \mathbb{R}^d$ (6.37) ekuaziotik askatuko dugu; bigarren, $W' \in \mathbb{R}^{md}$ (6.35) ekuaziotik askatuko dugu; hirugarren, $W'' \in \mathbb{R}^{(s-m)d}$ (6.33) ekuaziotik askatuko dugu; eta azkenik, ΔY (6.30) ekuaziotik askatuko dugu.

6.4. IRK-Newton estandarra (formulazio berria).

IRK puntu-finkoaren implementazioan erabilitako birformulazio (5.atala), IRK-Newton implementazioan ere aplikatuko dugu. Horrela, IRK metodoa simplektikoa izatea ziurtatzen dugu. IRK Newtonen iterazio implementazioan ordea, L_i ($i = 1, \dots, s$) aldagai ezezagunak eta Y_i ($i = 1, \dots, s$) aldagai laguntzaileak kontsideratuko ditugu, biribiltze errorea gutxitzeko helburuarekin [85].

IRK metodoaren formulazio estandarra (6.4), era honetan berridatziko dugu,

$$\Phi(y, t, h) := y + \sum_{i=1}^s L_i, \quad (6.39)$$

non $L_i \in \mathbb{R}^d$, $i = 1, \dots, s$ implizituki era honetan definitzen diren,

$$L_i = h b_i f(t + c_i h, y + \sum_{j=1}^s \mu_{ij} L_j), \quad i = 1, \dots, s, \quad (6.40)$$

eta

$$\mu_{ij} = a_{ij}/b_j, \quad 1 \leq i, j \leq s.$$

Newton simplifikatuaren iterazioa.

Formulazio berriari dagokion, Newton iterazioa definituko dugu: $k = 1, 2, \dots$ iterazioetarako, $L_i^{[k]}$ hurbilpenak era honetan kalkulatuko ditugu,

- 1) $Y_i^{[k]} := y + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, \quad i = 1, \dots, s.$
 - 2) Askatu $\Delta L_i^{[k]}$ from
- $$\Delta L_i^{[k]} - h b_i J_i^{[k]} \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s, \quad (6.41)$$
- non $J_i^{[k]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[k]})$ for $i = 1, \dots, s$,
- 3) $L^{[k]} := L^{[k-1]} + \Delta L^{[k]}.$

Newton sinplifikatuaren iterazioan, $J_i^{[k]}$ Jacobiarra $J_i^{[0]} = \partial f / \partial y(t + c_i h, Y_i^{[0]})$ $i = 1, \dots, s$ Jacobiaraz ordezkatzen da eta askatu beharreko ekuazio-sistema honako da,

$$\Delta L_i^{[k]} - hb_i J_i^{[0]} \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s.$$

Modu baliokidean, ekuazio lineala notazio matriziala erabiliz laburtu daiteke,

$$\left(I_s \otimes I_d - h \begin{bmatrix} b_1 \mu_{11} J_1^{[0]} & \dots & b_1 \mu_{1s} J_1^{[0]} \\ b_2 \mu_{21} J_2^{[0]} & \dots & b_2 \mu_{2s} J_2^{[0]} \\ \dots & \ddots & \dots \\ b_s \mu_{s1} J_s^{[0]} & \dots & b_s \mu_{ss} J_s^{[0]} \end{bmatrix} \right) \Delta L^{[k]} = g^{[k]},$$

non,

$$L^{[k]} = \begin{bmatrix} L_1^{[k]} \\ \vdots \\ L_s^{[k]} \end{bmatrix} \in \mathbb{R}^{sd}, \quad g^{[k]} = \begin{bmatrix} g_1^{[k]} \\ \vdots \\ g_s^{[k]} \end{bmatrix} \in \mathbb{R}^{sd},$$

$$J_{is}(y) = (\partial f^i / \partial y^j(y))_{i,j}^d = \begin{bmatrix} \frac{\partial f^1}{\partial y^1} & \dots & \frac{\partial f^1}{\partial y^d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f^d}{\partial y^1} & \dots & \frac{\partial f^d}{\partial y^d} \end{bmatrix} \in \mathbb{R}^{d \times d}, \quad is = 1, \dots, s.$$

Newton super-sinplifikatuaren iterazioa.

Honako bigarren sinplifikazioarekin, $J_i^{[0]} = \partial f / \partial y(t + c_i h, Y_i^{[0]})$, $i = 1, \dots, s$ matrizeak, $J_i^{[0]} \approx J$, $i = 0, \dots, s$ hurbilpena ordezkatuz, ekuazio-sistema lineal hau lortuko dugu,

$$(I_s \otimes I_d - h BAB^{-1} \otimes J) \Delta L = g. \quad (6.42)$$

non I_d, I_s identitate matrizeak eta B , (b_1, b_2, \dots, b_s) koefizienteen matrize diagonala diren.

Algoritmoa.

Formulazio berriari dagokion Newton super-simplifikatuaren implementazioa, (22) algoritmoan laburtu dugu.

```

 $\tilde{y}_0 = fl(y_0);$ 
 $e_0 = fl(y_0 - \tilde{y}_0);$ 
for  $n \leftarrow 0$  to (endstep − 1) do
     $k = 0;$ 
    Hasieratu  $L_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
     $J = \frac{\partial f}{\partial y}(y_n);$ 
     $M = LU(I_s \otimes I_d - h BAB^{-1} \otimes J);$ 
    while (not konbergentzia) do
         $k = k + 1;$ 
         $Y_{n,i}^{[k]} = y_n + (e_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k-1]});$ 
         $g_i^{[k]} = -L_{n,i}^{[k-1]} + h b_i f(t + c_i h, Y_{n,i}^{[k]});$ 
        Askatu ( $M \Delta L_n^{[k]} = g^{[k]}$ );
         $L_n^{[k]} = L_n^{[k-1]} + \Delta L_n^{[k]};$ 
        konbergentzia  $\leftarrow$  GeratzeErizpidea( $L_n^{[k]}$ ,  $L_n^{[k-1]}$ ,  $\Delta_{min}$ );
    end
    if ( $\exists j$  non  $\Delta_j^{[K]} \neq 0$ ) then
        if (NormalizeDistance( $Y_n^{[k]}$ ,  $Y_n^{[k-1]}$ ) > 1) then
            | fail convergence;
        end
    end
     $\beta_n = e_n + \sum_{j=1}^s \Delta L_{n,j}^{[k]};$ 
     $(\tilde{y}_{n+1}, e_{n+1}) \leftarrow$  baturakonpensatua( $\tilde{y}_n, \beta_n, L_n^{[k-1]}$ );
end

```

Algoritmoa 22: IRK (Newton super-simplifikatua).

Interpolazio koefizienteak. $L_{n,i}^{[0]}$ atalen hasieraketarentzat dagokien koefizienteak era honetan definituko ditugu. IRK puntu-finkoaren implementazioan finkatu genituen (5.13) interpolazio koefizienteetatik abiatuta modu errazean definituko

ditugu formulazio honi dagozkion interpolazio koefizienteak.

$$\begin{cases} Y_{n,i}^{[0]} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[0]} \\ Y_{n,i}^{[0]} = y_n + \sum_{j=1}^s \nu_{ij} L_{n-1,j} \end{cases} \Rightarrow L_n^{[0]} = (Mu^{-1}Nu)L_{n-1},$$

$$\Rightarrow (Mu^{-1}Nu)_{i,j}^s = \lambda_{ij}/a_{ij}.$$
(6.43)

Geratze irizpidea. Puntu-finkoaren iterazioan oinarritutako implementazioarentzat definitutako geratze irizpide berdina (5.15) erabiliko dugu baina $L_{n,i}$, $i = 1, \dots, s$ aldagaiei aplikatuta.

$$\Delta^{[k]} = (L_{n,1}^{[k]} - L_{n,1}^{[k-1]}, \dots, L_{n,s}^{[k]} - L_{n,s}^{[k-1]}) \in \mathbb{F}^{sd},$$

Honako notazioa finkatuko dugu,

$$\Delta_j^{[k]}, \text{ non } \Delta^{[k]} \in \mathbb{F}^{sd} (1 \leq j \leq sd).$$

Iterazioak $k = 1, 2, \dots$ jarraitza, $\Delta^{[k]} = 0$ bete arte edo honako baldintza bi iterazio jarraietan betetzen den artean,

$$\forall j \in \{1, \dots, sd\}, \quad \min \left(\{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} / \{0\} \right) \leq |\Delta_j^{[k]}|. \quad (6.44)$$

Batura konpensatua. $\tilde{y}_{n+1}, e_{n+1} \in \mathbb{F}^d$, non $\tilde{y}_{n+1} + e_{n+1} \approx y(t_{n+1})$ era honetan kalkulatuko dugu:

1. $\Delta L^{[k]}$ gaiak gehitu.

$$\delta_n = e_n + \sum_{j=1}^s \Delta L_{n,j}^{[k]}$$

2. Batura konpensatua.

Azkenik, batura konpensatua aplikatuko dugu,

$$(\tilde{y}_{n+1}, e_{n+1}) = S_{s,d}(\tilde{y}_n, \delta_n, L_{n,1}^{[k-1]}, \dots, L_{n,s}^{[k-1]}) \quad (6.45)$$

Function BaturaKonpensatua ($y_n, \delta_n, L_n^{[k-1]}$)

```

 $s_0 = y_n$ 
 $ee = \delta_n$ 
for  $i \leftarrow 1$  to ( $s$ ) do
     $s_1 = s_0$ 
     $inc = L_{n,i}^{[k-1]} + ee$ 
     $s_0 = s_1 + inc$ 
     $ee = (s_1 - s_0) + inc$ 
end
 $y_{n+1} = s_0$ 
 $e_{n+1} = ee$ 
return ( $y_{n+1}, e_{n+1}$ )

```

Algoritmoa 23: BaturaKonpensatua $S_{s,d}(\tilde{y}_n, \delta_n, L_{n,1}^{[k-1]}, \dots, L_{n,s}^{[k-1]})$ funtziaren implementazioa da

6.5. IRK-Newton eraginkorra (formulazio berria).

Formulazio berrian, modu eraginkorrean askatu behar dugun ekuazio-lineala honakoa da,

$$(I_s \otimes I_d - h BAB^{-1} \otimes J) \Delta L = g, \quad (6.46)$$

$J \in \mathbb{R}^{d \times d}$ eta $g \in \mathbb{R}^{s \times d}$ emandako matrizeak izanik.

Ekuazio-lineala (6.46) ebazteko, aurreko (6.3. atala eta 6.3. atala) ataletan (6.14) moduko sistemak askatzeko deskribatutako teknika egokituko dugu. Jarrain, IRK metodo simetriko simplektikoetarako (6.3. atala) garatutako teknika, formulazio berriko (6.46) sistema ebazteko nola aplikatu daitekeen deskribatuko dugu.

Formulazio estandarretik formulazio berrirako urratsa.

Formulazio berriaren implementazio eraginkorra, formulazio estandarrean eman-dako ekuazioak (6.3. atala) moldatzuz zehaztuko dugu. Aurreko ataleko ekuazioetan, bi formulazioen aldagaien arteko erlazioak ordezkatuz,

$$\Delta L = (B \otimes I_d) \Delta Y, \quad (6.47)$$

$$r = (B^{-1} \otimes I_d) g, \quad (6.48)$$

formulazio berrirako ekuazio baliokideak lortuko ditugu.

1. Aldagai aldaketa. Formulazio estandarraren aldagai aldaketari (6.30) goiko lehen erlazioa aplikatuz,

$$\Delta L = (BQ_1 \otimes I_d) W' + (BQ_2 \otimes I_d) W''. \quad (6.49)$$

2. $R \in \mathbb{R}^{md}$ matrizea. Formulazio estandarraren R matrizearen ekuazioan (6.34) goiko bigarren erlaziona aplikatuz,

$$\begin{aligned} R &= (Q_1^T \otimes I_d) g + h (DQ_2^T \otimes J) g, \\ R &= Q_1^T g + h DQ_2^T g J^T. \end{aligned} \quad (6.50)$$

3. $W'' \in \mathbb{R}^{(s-m)d}$ matrizea. Formulazio estandarraren W'' matrizearen ekuazioan (6.33) goiko bigarren erlaziona aplikatuz,

$$W'' = -h (D^T \otimes J) W' + (Q_2^T \otimes I_d) g. \quad (6.51)$$

Formulazio berrian, IRK Newton simplifikatuaren implementazioaren urratsak hauek dira,

1. LU deskonposaketak.

- (a) $\mathbb{R}^{d \times d}$ matrizeen LU deskonposaketa,

$$I_d + h^2 \sigma_i^2 J^2, \quad i = 1, \dots, [s/2].$$

- (b) $M \in \mathbb{R}^{d \times d}$ matriza kalkulatu,

$$M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1} \in \mathbb{R}^{d \times d}.$$

- (c) M matrizearen LU deskonposaketa

$$M \Delta z = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i.$$

2. Ekuazio-sistemaren (6.46) soluzioa ebatzi.

- $R \in \mathbb{R}^{md}$ kalkulatu,

$$R = (Q_1^T \otimes I_d) g + h (DQ_2^T \otimes J) g.$$

- d kalkulatu,

$$d = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i,$$

- $\Delta z \in \mathbb{R}^d$, ekuazio-sistematik askatu,

$$M \Delta z = d.$$

- $W_1, \dots, W_m \in \mathbb{R}^d$ kalkulatu,

$$(I_d + h^2 \sigma_i^2 J^2) W_i - \frac{\alpha_i}{2} J \Delta z = R_i, \quad i = 1, \dots, m.$$

- $W'' \in \mathbb{R}^{sd}$ kalkulatu,

$$W'' = (-hD^T) W' J^T + Q_2^T g.$$

- $\Delta L \in \mathbb{R}^{sd}$ kalkulatu,

$$\Delta L = (BQ_1 \otimes I_d) W' + (BQ_2 \otimes I_d) W'',$$

IRK Newton sinifikatuaren implementazioa, [24](#) algoritmoan laburtu dugu.

```

 $\tilde{y}_0 = fl(y_0);$ 
 $e_0 = fl(y_0 - \tilde{y}_0);$ 
for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $k = 0;$ 
    Hasieratu  $L_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
     $J = \frac{\partial f}{\partial y}(y_n);$ 
     $M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1};$ 
    lum =  $LU(M);$ 
    while (not konbergentzia) do
         $k = k + 1;$ 
         $Y_{n,i}^{[k]} = y_n + (e_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k-1]});$ 
         $g_i^{[k]} = -L_{n,i}^{[k-1]} + h b_i f(t + c_i h, Y_{n,i}^{[k]});$ 
         $R = Q_1^T g + (h D Q_2^T) g J^T;$ 
         $d = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i;$ 
        Solve(lum  $\Delta z = d$ );
         $W_i = (I_d + h^2 \sigma_i^2 J^2)^{-1} (R_i + \frac{\alpha_i}{2} \Delta z)$ ,  $i = 1, \dots, m$ ;
         $W^{\circ} = (-h D^T) W^{\circ} J^T + Q_2^T g;$ 
         $\Delta L = B Q_1 W^{\circ} + B Q_2 W^{\circ};$ 
         $L_n^{[k]} = L_n^{[k-1]} + \Delta L_n^{[k]};$ 
        konbergentzia  $\leftarrow$  GeratzeErizpidea( $L_n^{[k]}$ ,  $L_n^{[k-1]}$ ,  $\Delta_{min}$ );
    end
    if ( $\exists j$  non  $\Delta_j^{[K]} \neq 0$ ) then
        if ( $NormalizeDistance(Y_n^{[k]}, Y_n^{[k-1]}) > 1$ ) then
            | fail convergence;
        end
    end
     $\delta_n = e_n + \sum_{j=1}^s \Delta L_{n,j}^{[k]};$ 
     $(\tilde{y}_{n+1}, e_{n+1}) \leftarrow$  baturakonpensatua( $\tilde{y}_n, \delta_n, L_n^{[k-1]}$ );
end

```

Algoritmoa 24: IRK (NSS-Eraginkorra).

6.6. IRK Newtonen iterazio Mixtoa.

Sasi-Newton iterazioa.

Hurrengo 6.6. ataleko IRK metodoaren implementazio berrirako, Newton iterazio metodoaren (6.41) aldaera bat konsideratuko dugu. Newtonen iterazio bakoitzaren ekuazio-sistema hau,

$$\Delta L_i^{[k]} - h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s, \quad (6.52)$$

non

$$g_i^{[k]} = -L_i^{[k-1]} + h b_i f\left(t + c_i h, y + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}\right), \quad i = 1, \dots, s, \quad (6.53)$$

eta

$$\Delta L^{[k]} = \begin{pmatrix} \Delta L_1^{[k]} \\ \vdots \\ \Delta L_s^{[k]} \end{pmatrix} \in \mathbb{R}^{sd}, \quad g^{[k]} = \begin{pmatrix} g_1^{[k]} \\ \vdots \\ g_s^{[k]} \end{pmatrix} \in \mathbb{R}^{sd},$$

zehazki askatu ordez, modu iteratiboan askatuko dugu. Barne iterazioa (25 algoritmoa) aplikatuz, $\Delta L^{[k]} \in \mathbb{R}^{sd}$ soluzioaren $\Delta L_i^{[k,0]}, \Delta L_i^{[k,1]}, \Delta L_i^{[k,2]}, \dots$ hurbilpe-nak kalkulatuko ditugu.

```

 $\Delta L^{[k,0]} = (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} g^{[k]};$ 
while GeratzeErizpidea ( $fl_{32}(\Delta L^{[k,0]}), \dots, fl_{32}(\Delta L^{[k,l]})$ ) do
     $l = l + 1;$ 
     $G_i^{[k,l]} = g_i^{[k]} - \Delta L_i^{[k,l-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,l-1]}, \quad i = 1, \dots, s;$ 
     $\Delta L^{[k,l]} = \Delta L^{[k,l-1]} + (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} G^{[k,l]};$ 
end
```

Algoritmoa 25: Barne iterazioa.

non $fl_{32}(x)$, $x \in \mathbb{R}$ zenbakiaren gertuen dagoen IEEE doitasun arrunteko (32-bit) balioa adierazten duen.

IRK implementazio berri honen (26 algoritmoa) (6.52) ekuazio-sistema linea-laren J_i Jacobiar matrizeen ebaluazioak, doitasun arrunta duten Y_i atalekin kalkulatuko ditugu. Beraz, (25 algoritmoa) barne iterazioen geratze irizpidea, $fl_{32}(\Delta L^{[k,l]}) = fl_{32}(\Delta L^{[k,l-1]})$ doitasun arruntean betetzen dela aztertzea nahikoa izango dugu.

IRK-Newton Mixtoa

Zenbakizko soluzioa $y_n \approx y(t_n) \in \mathbb{R}^d$, $n = 1, 2, \dots$, bi bektoreen batura gisa, $\tilde{y}_n + e_n \in \mathbb{F}^d$ lortuko dugu. Hasierako balioa $y_0 \in \mathbb{R}^d$, $\tilde{y}_0 + e_0$ batura moduan adieraziko dugu, non $\tilde{y}_0 = fl(y_0)$ eta $e_0 = fl(y_n - \tilde{y}_0)$ diren.

Zehazki, $(\tilde{y}_{n+1}, e_{n+1}) = \tilde{\Phi}(\tilde{y}_n, e_n, t_n, h)$ IRK metodoaren urrats berriaren zenbakizko soluzioa, bost fazeetan kalkulatuko dugu:

1. $L^{[0]} = 0 \in \mathbb{R}^{sd}$ atalak hasieratu, eta Newton super-simplifikatuaren iterazioak aplikatu ((6.41) iterazioaren ekuazio-sistema, (6.42) sistemarekin ordezkatuz),

$$L^{[1]} = L^{[0]} + \Delta L^{[1]}, \quad L^{[2]} = L^{[1]} + \Delta L^{[2]}, \dots$$

geratze irizpidean, $\text{fl}_{32}(L^{[k]}) = \text{fl}_{32}(L^{[k-1]})$ bete arte.

2. $L^{[k]}$ berriari dagokion Jacobiarak ebaluatu

$$J_i = \frac{\partial f}{\partial y} \left(t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k]} \right), \quad i = 1, \dots, s.$$

3. Lehen fasean lortutako $\Delta L^{[k]} \in \mathbb{R}^{sd}$ balioa, (6.52) ekuazio-sistema linealaren $\Delta L^{[k]}$ soluzio zehatzaren $\Delta L^{[k,0]}$ hurbilpena konsideratu eta barne iterazioak (25 algoritmoa) aplikatu, $\Delta L^{[k]}$ soluzioaren $\Delta L^{[k,\ell]}$ hurbilpena (gutxienez doitasun arruntarekin) lortzeko.
4. $L^{[k]} = L^{[k-1]} + \Delta L^{[k,\ell]}$, eta $k = k + 1$ eguneratu ondoren, Sasi-Newton iterazio bat aplikatuko dugu bigarren urratsean kalkulatutako J_i Jacobiarren balioak erabiliz. Ekuazio-sistema linealaren (6.52)-(6.53), $\Delta L^{[k]}$ soluzioaren $\Delta L^{[k,\ell]}$ hurbilpenak (berriz ere doitasun arruntean) 25 algoritmoa aplikatuz kalkulatuko ditugu.
5. Azkenik, $(\tilde{y}_{n+1}, e_{n+1}) = \tilde{\Phi}(\tilde{y}_n, e_n, t_n, h)$ urrats berriaren zenbakizko soluzioa kalkulatuko dugu,

$$\tilde{\Phi}(\tilde{y}_n, e_n, t_n, h) = (\tilde{y}_n + e_n) + \sum_{i=1}^s (L_{n,i}^{[k-1]} + \Delta L_{n,i}^{[k,\ell]}).$$

Horretarako, Kahan-en batura konpensatua (10 algoritmoa) modu honetan aplikatuko dugu:

- (a) $\Delta L^{[k]}$ gaien batura (magnitude txikiko bektoreen batura).

$$\delta_n := e_n + \sum_{i=1}^s \Delta L_{n,i}^{[k,\ell]}$$

- (b) Batura konpensatua.

$$(\tilde{y}_{n+1}, e_{n+1}) = S_{s,d}(\tilde{y}, \delta_n, L_{n,1}^{[k-1]}, \dots, L_{n,s}^{[k-1]}).$$

Implementazio honen hainbat zehaztasun azpimarratuko ditugu:

- Algoritmoaren era honetako sistema linealak ($I_s \otimes I_d - h B A B^{-1} \otimes J$), 7.5.ataleko Newton implementazio eraginkorrarekin ([24 algoritmoa](#)) askatuko ditugu.
- $\mu_{ij} \in \mathbb{F}$ koefizienteek, zehazki [\(5.8\)](#) propietate simplektikoa eta simetria propietatea $\mu_{j,i} = \mu_{s+1-i,s+1-j}$ betetzen dituzte.
- $g_i^{[k]}$ ($i = 1, \dots, s$) hondarren [\(6.53\)](#) konputaziorako, $y \in \mathbb{R}^d$ balioaren ordez, $\tilde{y} + e$ ($\tilde{y}, e \in \mathbb{F}^d$) espresioa erabili beharko litzateke. Hori horrela egingo balitz, eragina oso txikia izango litzateke, eta azken Sasi-Newton iterazioan (4.fasea) bakarrik kontutan hartzea erabaki dugu. Gainera, azken Sasi-Newton iterazioan $\tilde{y} + e$ zuzenean balioa erabili ordez, J_i Jacobiarra erabili dugu honako hurbilpenarekin,

$$h b_i f \left(t + c_i h, \tilde{y} + e + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]} \right) - L_i^{[k-1]} \approx \\ \left(h b_i f_i^{[k]} - L_i^{[k-1]} \right) + h b_i J_i e,$$

non $f_i^{[k]} = f \left(t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]} \right)$.

Jarraian, implementazioaren implementazioaren [26](#) algoritmoa laburtu dugu.

```

 $L^{[0]} = 0;$ 
 $J = \frac{\partial f}{\partial y}(t + h/2, \tilde{y});$ 
 $M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1};$ 
Compute the LU decomposition of M;
***** 1-Fasea *****/;

 $k = 0;$ 
while ContFcn( $fl_{32}(L^{[0]}), \dots, fl_{32}(L^{[k]})$ ) do
     $k = k + 1;$ 
     $Y_i^{[k]} = \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, i = 1, \dots, s;$ 
     $f_i^{[k]} = f(t + c_i h, Y_i^{[k]}), i = 1, \dots, s;$ 
     $g_i^{[k]} = h b_i f_i^{[k]} - L_i^{[k-1]}, i = 1, \dots, s;$ 
     $\Delta L^{[k]} = (I_s \otimes I_d - h B A B^{-1} \otimes J)^{-1} g^{[k]};$ 
     $L^{[k]} = L^{[k-1]} + \Delta L^{[k]};$ 
end
***** 2-Fasea *****/;

 $J_i = \frac{\partial f}{\partial y} \left( t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k]} \right), i = 1, \dots, s;$ 
***** 3-Fasea *****/;

 $\ell = 0;$ 
 $\Delta L^{[k,0]} = \Delta L^{[k]};$ 
while ContFcn( $fl_{32}(\Delta L^{[k,0]}), \dots, fl_{32}(\Delta L^{[k,\ell]})$ ) do
     $\ell = \ell + 1;$ 
     $G_i^{[k,\ell]} = g_i^{[k]} - \Delta L_i^{[k,\ell-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,\ell-1]}, i = 1, \dots, s;$ 
     $\Delta L^{[k,\ell]} = \Delta L^{[k,\ell-1]} + (I_s \otimes I_d - h B A B^{-1} \otimes J)^{-1} G^{[k,\ell]};$ 
end
 $L^{[k]} = L^{[k-1]} + \Delta L^{[k,\ell]};$ 
***** 4-Fasea *****/;

 $k = k + 1;$ 
 $Y_i^{[k]} = \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, i = 1, \dots, s;$ 
 $f_i^{[k]} = f(t + c_i h, Y_i^{[k]}), i = 1, \dots, s;$ 
 $g_i^{[k]} = (h b_i f_i^{[k]} - L_i^{[k-1]}) + h b_i J_i e, i = 1, \dots, s;$ 
 $\ell = 0;$ 
 $\Delta L^{[k,0]} = (I_s \otimes I_d - h B A B^{-1} \otimes J)^{-1} g^{[k]};$ 
while ContFcn( $fl_{32}(\Delta L^{[k,0]}), \dots, fl_{32}(\Delta L^{[k,\ell]})$ ) do
     $\ell = \ell + 1;$ 
     $G_i^{[k,\ell]} = g_i^{[k]} - \Delta L_i^{[k,\ell-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,\ell-1]}, i = 1, \dots, s;$ 
     $\Delta L^{[k,\ell]} = \Delta L^{[k,\ell-1]} + (I_s \otimes I_d - h B A B^{-1} \otimes J)^{-1} G^{[k,\ell]};$ 
end
***** 5-Fasea *****/;

 $\delta = e + \sum_{i=1}^s \Delta L_i^{[k,\ell]};$ 
 $(\tilde{y}^*, e^*) = S_{s,d}(\tilde{y}, \delta, L_1^{[k-1]}, \dots, L_s^{[k-1]});$ 

```

Algoritmoa 26: IRK implementazio Mixtoa

6.7. Zenbakizko esperimentuak.

Newton iterazioan oinarritutako 6-ataletako Gauss kolokazio metodoaren implementazioarekin egindako zenbakizko esperimentuak azalduko ditugu. Esperimentu hauen konputaziorako, 64-biteko doitasuneko IEEE koma-higikorreko aritmetika erabili dugu.

Problemak

Pendulu bikiotz zurruna

Pendulu bikoitz zurrunaren problemaren Hamiltondarra eta parametroak, [3.2.](#), atalean definitu dugu. k parametroak malgukiaren zurruntasun maila finkatzen du: $k = 0$ balioarentzat, problema ez da zurruna eta problemaren zurruntasuna, k balioarekin batera handitzen da.

Hasierako balioak, era honetan aukeratu ditugu: $k = 0$ problema ez zurrunarentzat, [\[29\]](#) artikulutik izaera ez-kaotikoa duen hasierako balioa hartu dugu: $q(0) = (1.1, -1.1)$ and $p(0) = (2.7746, 2.7746)$. $k \neq 0$ problemen hasierako balioak,

$$q(0) = \left(1.1, \frac{-1.1}{\sqrt{1 + 100k}} \right), \quad p(0) = (2.7746, 2.7746),$$

espresioen bidez finkatu ditugu, non sistemaren energia $k \rightarrow \infty$ handitzen den heinean, bornatua dagoen.

Integrazio guztiak, $h = 2^{-7}$ urrats luzera erabiliz egin ditugu; problema ez zurrunarentzat ($k = 0$) trunkatze errorea biribiltze errore baino txikiago izan dadin aukeratu dugu. Trunkatze errorea, biribiltze errorea baino handiagoa izango da $k > 0$ zurruntasun balio handietarako. $T_{end} = 2^{12}$ segundoko integrazioak egin ditugu eta zenbakizko soluzioa, $m = 2^{10}$ urratsero itzuli dugu.

Biribiltze errorearen azterketa.

Lehenengo, Newtonen iterazioan oinarritutako IRK implementazio berriaren biribiltze errorearen hedapena egokia dela aztertuko dugu. Aurreko artikuluan [\[7\]](#), biribiltze errorearen hedapena gutxitzeko ahalegin berezia eginez, puntu-finkoaren iterazioan oinarritutako IRK implementazioa proposatu genuen. Bi implementazioen, 6-ataleko Gauss kolokazio metodoaren biribiltze erroreak konparatuko ditugu.

Pendulu bikoitzaren hiru k balioetarako, energi errorearen azterketa zehatza egin dugu: $k = 0$, non biribiltze errorea trunkatze erroreari nagusitzen zaion; $k = 2^{10}$, non bi erroreak tamaina berekoak diren; eta $k = 2^{12}$, non trunkatze

errorea birbiltze erroreari nagusitzen zaion. Biribiltze errorearen konparaketa sendoago izan dadin ([49] lanean bezala), azterketa estatistikoa egin dugu. Problema bakoitzarentzat, hasierako balioak $\mathcal{O}(10^{-6})$ errore tamainako ausaz perturbatutako $P = 1000$ integracio konputatu ditugu.

6.1. irudien zenbakizko integrazioek, gure implementazio berriaren biribiltze errorearen propagazio ona erakusten dute. Alde batetik, $k = 0$ eta $k = 2^{10}$ balioetarako, puntu-finkoaren iterazioan oinarritutako implementazioak energia batez-bestekoan, drift lineala agertzen du eta Newton iterazioan oinarritutako implementazioak, ordea ez du energia driftarik agertzen. Beste alde batetik, bi implementazioetan, energiaren desbideratze estandarrak antzekoak dira eta $t^{1/2}$ espresioaren proporcionalak dira.

Puntu-finkoa versus Newton iterazioa

6.1. taulan, k parametroaren lau balioetarako, bi implementazioen eraginkortasunaren adierazle nagusienak laburtu ditugu.

6.1. Taula

C	0	2^3	2^6	2^8
E_0	-14.39	-5.75	-5.64	-5.64

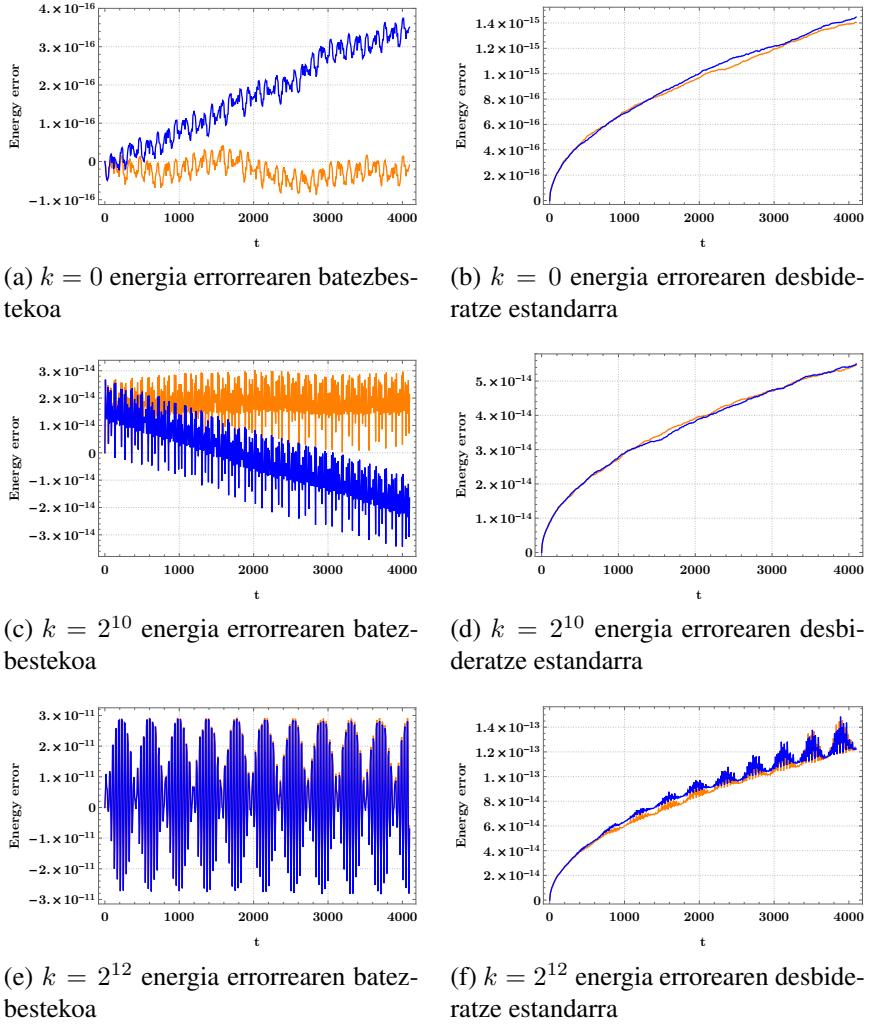
Fixed-points it.

Elapsed-time (sec.)	10	12	19	51
It. per step	8.58	11.1	22.	64.2
Energy	2.96×10^{-15}	1.81×10^{-14}	2.94×10^{-11}	6.33×10^{-5}

Newton it.

Elapsed-time (sec.)	18	20	19	18
It. per step	5.09	5.53	5.58	5.01
L. solves per step	11.37	12.92	12.72	11.04
Energy	1.6×10^{-15}	1.74×10^{-14}	2.94×10^{-11}	6.33×10^{-5}

Eraginkortasuna neurtzeko, bi implementazioen exekuzio sekuentzialen cputenborak konparatu ditugu. Hortaz gain, bi implementazioen urratseko iterazio



6.1. Irudia: Energia errore batezbestekoaren (ezkerrean) eta desbideratze estandarraren eboluzioa (eskubian), puntu-finkoaren implementazioa (urdinez), eta Newton implementazioa (laranjaz). $k = 0$ problema ez-zurruna (a,b), $k = 2^{10}$ lehen problema zurruna (c,d) eta $k = 2^{12}$ bigarren problema zurruna (e,f)

batezbestekoak (It. per step) alderatu ditugu eta Newton implementazioan, urratseko sistema linealen ebazpen batezbestekoa (L.solves per step) eman dugu. Zenbakizko soluzioaren doitasuna neurtzeko, energia errore erlatibo maximoa eman dugu,

$$\max \left| \frac{E(t_n) - E(t_0)}{E(t_0)} \right|, \quad t_n = t_0 + nh, \quad n = 1, 2, \dots$$

k balio txikienetarako, puntu-finkoaren implementazioa Newton implementa-

zioa baino eraginkorragoa da. Baino, pendulu bikoitzaren zurruntasun maila handitzen dugunean, puntu-finkoaren iterazio kopurua gero eta handiagoa den bitartean, Newton implementazioaren iterazio kopurua zerbait txikiagoa da k balio handienetarako. Beraz, zurruntasuna handitzen dugunean, Newton implementazioa gero eta eraginkorragoa bilakatzen da. $k = 2^{18}$ baliotik aurrera, puntu-finkoak ez du konbergitzen eta Newton implementazioak, iterazio kopuru antzekoarekin konbergitzen du.

6.8. Laburpena.

IRK metodoen Newton simplifikatuaren iterazioen ekuazio-sistema, modu eraginkorran askatzeko implementazioa proposatu dugu. Newtonen iterazioan oinarrtitako IRK metodoaren implementazio berria aurkeztu dugu eta implementazio honen biribiltze errorearen hedapena egokia dela baiezttatu dugu. Problema zurrutarako, Newton simplifikatuaren iterazioa, puntu-finkoaren iterazioa baino eraginkorragoa dela ikusi dugu.

Newtonen iterazioan oinarrtitako IRK metodoen implementazioen inguruko lan hauek aipatuko ditugu: "On the implementation of implicit Runge-Kutta methods", J.C. Butcher [21], "Geometric numerical integration: structure-preserving algorithms for ordinary differential equations", E.Hairer et al [48].

Azkenik, aipatu nahi dugu, atal honen edukiak [Numerical Algorithms](#) aldizkarian publikatutako [6] artikuluan aurki daitezkeela eta implementazioaren [kodea](#) eskuragarri jarri dugula.

7. Kapitulua

IRK: Eguzki-sistema.

7.1. Sarrera.

Koordenatu kartesiarrak erabiltzearen abantaila.

7.2. Kepler-en fluxua.

7.3. Implementazio berria.

Eguzki sistemarako honako idei berri bat azalduko dugu. Honako ekuazio diferentziala dugularik,

$$\dot{y} = k(y) + \epsilon g(y)$$

Alde kepleriarraren fluxua ezaguna dugu,

$$\begin{aligned}\varphi_{\Delta t}^k : \mathbb{R}^d &\longrightarrow \mathbb{R}^d \\ y_0 &\longrightarrow y_1.\end{aligned}$$

Aldagai aldaketa bat egin daiteke,

$$\begin{aligned}y(t_0 + \Delta t) &= \varphi_{\Delta t}^k(z(t_0 + \Delta t)), \quad y(t_0) = z(t_0), \\ z(t_0 + \Delta t) &= \varphi_{-\Delta t}^k(y(t_0 + \Delta t)).\end{aligned}$$

Aldagai berriarekiko ekuazio diferentziala mantso aldatzen den funtzioa da,

$$\dot{z} = \epsilon r(z, t).$$

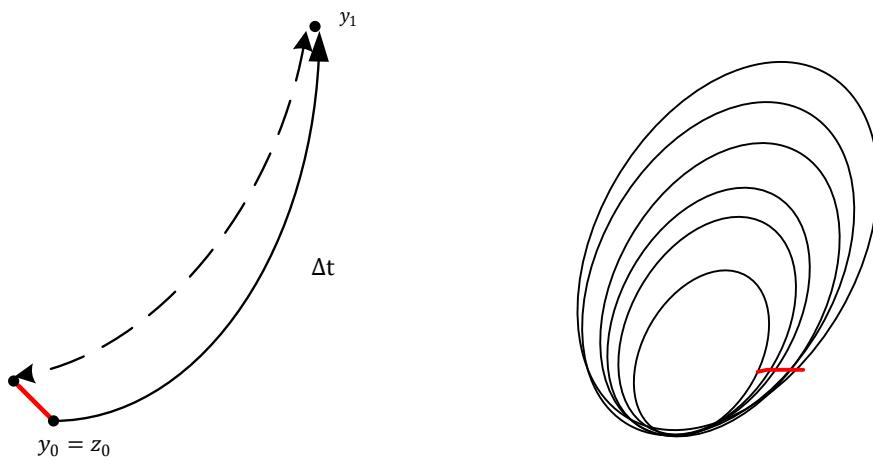
Idea hau bi modutara aplika daiteke,

1. Gauss inplizituaren integracio metodoan.

2. Atalen hasieraketa ona lortzeko. Orokorrean, interpolazio bidezko hasieraketa ona izateko, urratsa txikia izan behar du (periodo bat baino txikiagoa izan behar du). Teknika hau erabiliz, interpolazioaren errorea $\mathcal{O}(\epsilon)$ mailakoa izango da.

Proposamen honetan hasieraketa z aldagai berria erabiliz era honetan egingo dugu:

- Y_{n-1} atalei kepler **denboran atzeratuz**, Z_{n-1} aldagai berriarekiko atalak lortuko ditugu.
- Z_{n-1} alatak interpolauz, $Z_n^{[0]}$ hasieraketak lortuko ditugu.
- $Z_n^{[0]}$ atalei kepler **denboran aurreratuz**, $Y_n^{[0]}$ hasieraketak lortuko ditugu.



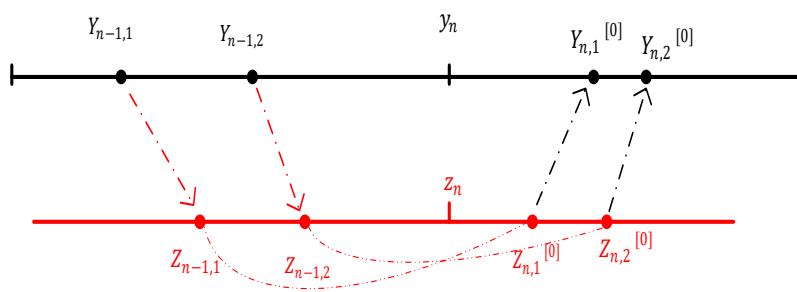
(a) Atalen hasieraketa1.

(b) Atalen hasieraketa2.

7.1. Irudia: (a) irudian, (b)

7.4. Zenbakizko esperimentuak.

7.5. Laburpena.

**7.2. Irudia:** Atalen hasieraketa3.

8. Kapitulua

Eztabaida.

8.1. Eguzki-sistemaren integrazio rako egungo metodoak.

Tesiaren sarreran azaldu genuenez, gure helburua eguzki-sistemaren epe luzeko eta doitasun handiko implementazio eraginkorra proposatzea da. Atal honetan, egungo eguzki-sistemaren simulazioetan erabiltzen diren metodo eta implementazioen laburpena egingo dugu. Bi eratako simulazioak aztertuko ditugu. Batetik, eguzki-sistemaren efemerideak ditugu; eguzki-sistemaren eredu konplexuak eta epe tarte txikietarako (ehunka urtekoak) integrazioak konputatzen dituzte. Bestetik, astronomi arloaren inguruko ikerketetarako eguzki-sistemaren integrazio luzeak ditugu: eguzki-sistemaren eredu simpleagoak eta milioika urteko integrazioak konputatzen dituzte.

Efemerideak.

Konputagailuen aurreko garaian, efemerideak teoria analitikoetan oinarritutako serie funtzioen bidez kalkulatzen ziren. Soluzio hauetan, Fourier-en serie trigonométriko luzeen ebaluazioa egin behar zen. 1960 hamarkadan, eguzki-sistemaren ezagutza hobetu zenean (espazio bidaia eta behatoki astronomikoen aurrerapenak medio), serie oso luzeak kalkulatu behar zituzten, eta orduan zenbakizko integrazioen bidezko soluzioak eraginkorragoak bilakatu ziren [63].

Eguzki-sistemaren gorputzen efemeride modernoak, mugimenduaren ekuazio diferentzialen (3.5) zenbakizko integrazioaren bidez kalkulatzen dira. Integrazioaren hasierako balioak eta ereduaren parametroak, sateliteen bidez jasotako datuekin kontrastatu egiten dira.

Efemerideak, *Chebyshev* polinomio moduan adierazten dira. Integrazio tarreak, 2.000. urte inguruko ehunka urtekoak izaten dira. Zenbakizko integrazio

hauetan, biribiltze errorea gai garrantzitsua da. 128-biteko aritmetikaren aukera bazterzen da, konputazioa oso garestia delako eta 64-biteko doitasuneko aritmetika hobetzen dituzten teknika konputazionalki merkeagoak, aplikatzen dira.

Efemerideetarako, eguzki-sistemaren eredu konplexua aplikatzen da. Gorputz nagusien arteko indar grabitacionalez gain, erlatibilitate efektua, asteroideek era-gindako grabitazio indarrak, gorputzen formen eragina eta beste hainbat indar ez grabitacionalak kontutan hartzen dituzte. Mugimenduaren ekuazio diferentzialak, era honetakoak izaten dira [34],

$$\ddot{x}_{\text{Planet}} = \sum_{A \neq B} \mu_B \frac{r_{AB}}{\|r_{AB}\|^3} + \ddot{x}_{GR}(\beta, \gamma, c^{-4}) + \ddot{x}_{AST,300} + \ddot{x}_{J_2}.$$

Hauek dira, ekuazio hauen konplexutasuna erakusten duten ezaugarri batzuk:

- Gorputz kopurua: 8 planetak, ilargia, Pluton eta 300 asteroide.
- Erlatibilitate efektua (GR): Einstein-Imfeld-Hoffmann, c^{-4} PPN hurbilketa.
- J_2 : eguzkia esferikoa ez izatearen eragina.
- Urrats luzera, $h = 0.055$ egunekoa da.

Erlatibilitate efektua.

Eguzkiaren erlatibilitate efektua kontutan hartzen duten ekuazio diferentzialak azalduko ditugu [67].

$$\dot{q}_i = v_i, \quad i = 0, 1, \dots, N \quad (8.1)$$

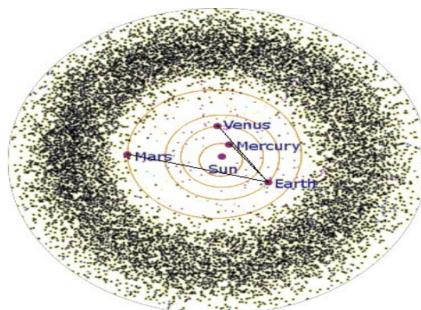
$$\begin{aligned} \dot{v}_i &= \sum_{j=0, j \neq i}^N \frac{Gm_j}{\|q_j - q_i\|^3} (q_j - q_i) \left(1 - \frac{2(\beta + \gamma)}{c^2} \sum_{k=0, k \neq i}^N \frac{Gm_k}{\|q_k - q_i\|} - \frac{2\beta - 1}{c^2} \sum_{k=0, k \neq j}^N \frac{Gm_k}{\|q_k - q_j\|} \right. \\ &\quad + \gamma \left(\frac{v_i}{c} \right)^2 + (1 + \gamma) \left(\frac{v_j}{c} \right)^2 - \frac{2(1 + \gamma)}{c^2} v_i v_j \\ &\quad \left. - \frac{3}{2c^2} \left(\frac{(q_i - q_j)v_j}{\|q_j - q_i\|} \right)^2 + \frac{1}{2c^2} (q_j - q_i) \dot{v}_i \right) \\ &+ \frac{1}{c^2} \sum_{j=0, j \neq i}^N \frac{Gm_j}{\|q_j - q_i\|^3} ((q_i - q_l)((2 + 2\gamma)v_i - (1 + 2\gamma)v_j))(v_i - v_j) \\ &\quad + \frac{3 + 4\gamma}{2c^2} \sum_{j=0, j \neq i}^N \frac{Gm_j \dot{v}_j}{\|q_j - q_i\|} \quad (8.2) \end{aligned}$$

8.1. Taula: Konstanteak

c	299792.458 km/s	Argiaren abiadura
au	149597870.700 km	Unitate Astronomikoa
β	1.0	PPN parametroa
γ	1.0	PPN parametroa

Asteriodeak.

Astroideek, bereziki Marte planetaren mugimenduarengan eragina dute (8.1. irudia) eta kontutan hartzekoak, barne-planeten mugimenduaren doitasun handiko emaitzak behar ditugunean. Bost asteroide nagusiren masak (Ceres, Pallas, Vesta, Iris eta Bamberga) Merkurio eta Pluton planeten mailakoak direnez, integracioetan gehitzen dira. Asteroide txikien talde handia, estimazioen bidez simulatzen dira.



8.1. Irudia: Asteroideak.

Hiru efemerideak.

Gaur-egun, eguzki-sistemaren planeten hiru efemeride kalkulatzen dira.

1. Jet Propulsion Laboratory (*EEBB*) NASA-ko erakundeak DE (Development Ephemerides) izeneko efemerideak konputatzen dituzte.

1984. urtean kalkulatu zen lehen efemeridea (DE-200) eta 2.014. urteko *DE-430* [36] efemeridea da, publikatutako azken efemeridea. Azken efemeride honen integrazio tartea, 1550 – 2650 urtetakoa izan da.

Zenbakizko integrazio metodoa, urrats luzera eta ordena aldakorreko *Multistep Adams* metodoa [68] (*DIVA/QIVA*) da. *QIVA* doitasun laukoitzeko (128-bit) bertsioari deitzen zaio: mugimenduaren ekuazioen Newton zatia,

doitasun laukoitzean kalkulatzen da eta ekuazioaren gainontzeko zatia, doitasun bikoitzean.

2. Institut de Méchanique Céleste et de Calcul des Ephémérides (IMCCE, Paris Observatory) INPOP (Intégrateur Numerique Planétaire de l'Observatoire de Paris) izeneko efemerideak.

2.000. urte arte, efemerideak kalkulatzeko teori analitikoetan oinarritu ziren. 2.003. urtean, zenbakizko integrazioaren bidezko lehen efemeridea kalkulatu zuten eta *INPOP13c* [35, 2.014] publikatutako azkena da.

Zenbakizko integrazio metodoa: 12 ordeneko *Adams-Cowell* metodoa da eta urrats finkoa aplikatzen dute.

Doitasuna: C lengoian implementatuta dago eta *Intel* makinetako 80-biteko doitasuna erabiltzen du. Era berean, modu merkean doitasun laukoitza simulatzu, urrats zuzentzaile (corrector step) bat aplikatzen dute [35].

3. Institute of Applied Astronomy (IAA, St. Petersburg), EPM (Ephemerides Planets-Moon) izeneko efemerideak.

1.980. urtetik aurrera, zenbakizko integrazioen bidezko efemerideak kalkulatu dituzte eta *EPM2.013* [88, 2.014] publikatutako azken efemeridea da.

Zenbakizko integrazio metodoa, *Everhart* izeneko *IRK* metodoa (Gauss-Radau) da. 23 ordeneko metodoa eta urrats luzera finkoa aplikatzen dute.

Doitasuna. Implementazioak (software package ERA), *Intel* makinetako 80-biteko doitasuna erabiltzen dute.

Eguzki-sistemaren integrazio luzeak.

A.Morbidellik [82] eguzki-sistemaren zenbakizko integrazioen algoritmoen garapenaren azterketan, garai hauek bereizten ditu:

1. Garai klasikoa.

90. hamarkada hasiera arte, urrats luzera aldakorreko integratzaileak erabilten dira: Runge-Kutta (Dormand et al. 1987), Bulirsch and Stoer (1966), Radau (Everhart, 1985), eta Störmer (1990). Garai honetan integrazio tarreak, $10^4 - 10^6$ urte artekoak dira.

2. Garai simplektikoa.

Wisdom eta Holman-en [98, 1991] lanarekin, eguzki-sistemaren azterketa rako integratzaile simplektikoen erabilera zabaldu zen. Garai honetan, ($10^8 - 10^9$) urte arteko eguzki-sistemaren integrazioak egin ziren.

3. Garai estatistikoak.

Planeten eta gorputz txikien (asteroide, meteoritoak) arteko kolisio gertuko egoerak kalkulatzen dituzten algoritmoak garatu ziren. Implementazio berri hauetan, milaka gorputzen integrazio azkarra egin daiteke. Horrela, asteroide eta meteoritoen orbiten distribuzio azterketa estatistikoak egin ziren.

4. Planeten sorrera garaiko azterketak.

Eguzki-sistemaren sorrerari buruzko simulazioak nagusituko dira; masa handiko gorputzen arteko kolisio gertuko egoerak gertatzen diren integrazioak konputatzen dira.

Eguzki-sistemaren integrazioetarako nagusiki, bi integratzaile famili aplikatzenten dira:

1. Integratzaile simetrikoak.

Metodo simetrikoen artean nagusiena, 4 ordenako *Hermite* [3] integratzailea da. Urrats luzera tamaina aldakorreko integratzailea da, modu errazean implementatu daiteke. Hermite integratzailea konputazionalki garestia da eta bereziki, gorputz kopuru handia dituzten eta kolisio gertuko egoerak maiz gertatzen diren problemetan (eguzki-sistemaren sorrera, ...) aplikatzenten da.

2. Simplektikoak.

Gaur-egun, eguzki-sistemaren epe luzeko integrazioetarako, integratzaile simplektikoak nagusitu dira.

Eguzki-sistemari egokitutako integratzaile simplektikoak.

Wisdom-ek eta Holman-ek [98, 1991], eguzki-sistemaren epe luzeko simulazioetarako integratzaile simplektikoak (*WH*) arrakasta izan zuen. Eguzki-sistema, mugimendu perturbatua duen sistema dinamikoa da eta ezaugarri honi egokitutako integratzaile eraginkorra garatu zuten. Jacobi koordenatuak aplikatuz, N-gorputzen problemaren Hamiltondarra, bi zatitan banatu zuten,

$$H(q, p) = H_K(p) + H_I(q) \quad , \quad H_K \gg H_I,$$

non H_K , Hamiltondar Kepleriarra (planeten eguzkiarekiko mugimendu kleperiarra) eta H_I , interakzioen Hamiltondarra (planeten arteko grabitazio interakzioak) diren. Integrazioaren urrats bakoitzean, Hamiltondar bakoitzaren soluzioa tartekatuz, problema osoaren ebazpena kalkulatzen da.

WH integratzailea, ondorengo metodo askoren aurrekaria kontsideratu bada ere, bere aplikagarritasuna mugatua da. Batetik, izar anitzeko planeten sistemak

edo planeta-ilargiak sistemak integratzeko ez da egokia. Bestetik, *WH* metodo sinplektikoa denez, urrats luzera finkoa aplikatu behar da eta hau, gorputzen arteko kolisiotik gertuko egoerak dituzten problemak modu eraginkorrean integratzeko eragozpen bat da. Arazo hauek gainditzeko, urteetan zehar algoritmo honen aldaerak proposatu dira eta jarraian, nagusienak aipatuko ditugu.

Levinson eta Duncan-ek [77, 1994], *WH* implementazioa, integratzaile ez sinplektiko batekin konbinatu zuten, kolisiotik gertuko egoeren kalkulua hobetzeko. *SWIFT* paketean, *RMVS3* izeneko integratzailea implementatu zuten. Duncan, Levinson eta Lee-k [30, 1998], koordenatu heliozentrikoak erabiliz, Hamiltondarra beste modu honetan banatu zuten,

$$H(q, p) = H_K(p) + (H_C(p) + H_I(q))$$

eta kolisiotik gertuko egoerei, urratsa luzera txikituz aurre egin zioten. Implementazio honek, *SYMBA* izena du. Chambers-ek [24] koordenatu heliozentrikoetan oinarritu zen eta kolisiotik gertuko egoerak gertatzen diren uneetan, beste integratzaile (Bulirsch-Stoer metodoa) batekin konbinatuz integratzen du. Implementazio honek, *MERCURY* izena du. Levinson eta Duncan-ek (2000), aurreko implementazioaren arazo batzuk konponduz, *Modified SYMBA* izeneko garapen berria burutu zuten. Kvaerno eta Leimkuhler [69] eta beste autore batzuk ere, antzeko ideiak landu dituzte.

Wisdom eta Holmanek proposatutko Hamiltondarraren banaketa, (2.21) *Leapfrog* metodoaren bidez integratzen da eta beraz, 2 ordeneko da. Orden altuagoko ($p > 2$) metodoak definitzeko, koefiziente negatiboak erabili behar zela uste zen [106, 75] eta orduan ez dira interesgarriak, *Leapfrog* metodoak eraginkorragoak baitira.

Beranduago, McLachlan-ek [80, 1995] eta Laskar-ek [75, 2001] koefiziente negatiboen arazoa gainditu zuten eta koefiziente positiboekin definitutako ordena altuko Splitting eskemak aurkitu zituzten. Berriki, Blanes-ek [16, 2012] ordena altuko Splitting eskema berriak eta eraginkorrak aurkitu ditu.

Hernandez eta Bertschinger-ek [51, 2015] N-gorputzen problema grabitazional eta kolisioidunetarako 2 ordenako integratzaile sinplektiko berri bat proposatu dute. Hernandez eta Bertschinger-ek [51] koordenatu kartesiarretan oinarriztuz, N-gorputzen problema, 2-gorputzen azpiproblemetan banatzen dute.

Splitting metodoak.

Splitting metodoak, era honetako perturbatutako Hamiltondar sistemak,

$$H = H_A + \epsilon H_B, \quad (8.3)$$

integratzeko aplika daitezke, non H_A eta H_B independenteki integragarriak diren. Pentsatzeko da, eguzki-sistemaren eredu errealistetan hainbat indar ez grabitazionalak modu honetan gehitzeko, zaitasunak izan ditzakegula.

Laskar-ek [73, 2011], epe luzeko zenbakizko integratiorako $1/c^2$ ordenako eguzkiaren erlatibilitatea kontsideratu zuen. Horretarako, Saha eta Tremaine-ek [91] finkatutako teknikan oinarritzen da. Teknika honetan, (8.3) Hamiltondar banagarriei egokitzen zaien erlatibilitatea kontsideratzen da. Gainera modu eraginkorrean kalkulatu daiteke. Kontutan hartuko bagenu, bi planeta handienetan (Jupiter eta Saturno) erlatibilitatea kontsideratzen da, ekuazioak ez dira gehiegizko konplikatzen eta errorea txikiagoa lortuko genuke. Splitting metodoetan ez da posible erlatibilitatea kontsideratzen da, ez direla kontutan hartzen.

Eguzki-sistemaren integratiorako, (8.3) Hamiltondarraren banaketa, Jacobi koordenatuak edo heliozentrikoak aplikatuz lortzen da. Koordenatu cartesiarrak, ezin daitezke erabili.

Muga hauek ez ditugu, IRK metodoak aplikatzeko. Ekuazio diferentzial orokorrean aplika daitekeenez, askatasun osoa dugu behar diren ekuazioak gehitzeko eta interesatzen zaigun koordenatu sistema aplikatzeko.

8.2. Ereduen deskonposaketa.

IRK metodoen implementazio eraginkorren azterketaren hasierako fasean, problemaren deskonposaketan oinarritutako ideia ikertu dugu. Eguzki-sistemaren eredu, problema simple (konputazionalki merkean) eta problema konplexu batean (konputazionalki garestia) bana daiteke. Problemaren alde konplexuaren balioztapen kopuru txikituz, implementazio eraginkorra lortzeko asmoarekin, Newton simplifikatuaren iterazio berezi bat aplikatu dugu. Newton simplifikatuaren iterazio berezi honetan, ekuazio-sistema lineala LU deskonposaketaren bidez askatu beharrean, puntu-finkoaren bidez ebatzi dugu.

Atal honen bukaeran emango ditugun arrazoiengatik, beste bide batzuk ikerketa zuzenago dela ikusi dugu eta planteamendu hau alde batera utzi dugu. Dena den, planteamendu honen laburpen bat emango dugu eta implementazio bide honetan sortutako hainbat ideia interesgarri jaso ditugu.

Implementazioa.

Eguzki-sistema, perturbatutako mugimendu Kepleriar gisa har daiteke,

$$f(y) = k(y) + \epsilon g(y), \quad g \ll k, \quad (8.4)$$

$k(y)$ sistema dinamikoaren alde simplifikatua da eta ebaluatzeko kostua txikia du. $g(y)$, ordea, sistemaren alde konplexua da eta ebaluatzeko kostua handia du.

Ekuazio diferentzialaren (8.4) deskonposaketa badugu, IRK metodoaren atalen ekuazio sistema ebazteko,

$$Y_i = y_n + h \sum_{j=1}^s a_{ij} f(Y_j),$$

era honetako iterazio proposatzen dugu,

$$\begin{aligned} k &= 1, 2, \dots \\ Y_i^{[k]} &= y_n + h \sum_{j=1}^s a_{ij} (k(Y_j^{[k]}) + g(Y_j^{[k-1]})) \end{aligned} \quad (8.5)$$

Iterazio hau aplikatzea, Jacobiarren hurbilpentzat, $J_i = \partial k / \partial y(Y_i)$, $i = 1, \dots, s$ hartzen duen Newton simplifikatua aplikatzearen baliokidea da. Hurrengo 27 meta-algoritmoan, iterazio aplikatzeko zehaztasun gehiago ematen ditugu,

```

for  $n \leftarrow 1$  to endstep do
    Hasieraketa  $Y_i^{[0]}, W_i^{[0]}$ ;
    k=1;
    Askatu  $Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} k(Y_j^{[k]}) + W_i^{[k-1]}$  ;
    while (konbergentzia ez lortu) do
        k++;
         $W_i^{[k-1]} = h \sum_{j=1}^s a_{ij} g(Y_j^{[k-1]})$ ;
        Askatu  $Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} k(Y_j^{[k]}) + W_i^{[k-1]}$  ;
    end
     $y_{n+1} = y_n + h \sum_{j=1}^s b_i f(Y_j^{[k]})$  ;
end

```

Algoritmoa 27: Meta-algoritmoa

Meta-algoritmoaren barne ekuazio-sistema askatzeko, metodo egokiena aplikatzeko askatasuna dugu. Pentsa liteke, problema zurruna ez bada puntu-finkoaren bidez askatza eta problema zurruna bada, berriz, Newton simplifikatuen iterazioaren bidez.

Barne ekuazio-sistema,

$$Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} k(Y_j^{[k]}) + W_i^{[k-1]},$$

puntu-finkoaren iterazioaren bidez askatzeko, 28 barne-iterazioa aplikatuko dugu.

```

 $l = 0;$ 
 $Y_i^{[k,0]} = Y_i^{[k-1]};$ 
while (konbergentzia ez lortu) do
     $l = l + 1;$ 
     $K_i^{[k,l]} = k(Y_i^{[k,l-1]});$ 
     $Y_i^{[k,l]} = y_n + h \sum_{j=1}^s a_{ij} K_j^{[k,l]} + W_i^{[k-1]};$ 
end

```

Algoritmoa 28: Barne-iterazioa: puntu-finkoaren iterazioa

Orokorpena.

Eredu konplexuak.

Aurreko atalean, maila bakarreko ereduen deskonposaketa aztertu dugu. Ideia orokortuz, eredu deskonposaketa maila ezberdinetan aplika daiteke. $\dot{y} = f(y)$ problema emanik,

$$1. \text{ maila } \begin{cases} \text{Eredu osoa. } f(y) \\ \text{Eredu simplea. } \tilde{f}(y) \end{cases} \Rightarrow f = \tilde{f} + (f - \tilde{f})$$

$$2. \text{ maila } \begin{cases} \text{Eredu osoa. } \tilde{f}(y) \\ \text{Eredu simplea. } \tilde{\tilde{f}}(y) \end{cases} \Rightarrow \tilde{f} = \tilde{\tilde{f}} + (\tilde{f} - \tilde{\tilde{f}})$$

...

Adibidea. Eguzki-sistemaren eredu konplexuaren ekuazio-sisteman, alde Keplériarra eta perturbazio maila ezberdinak bereiz daitezke. Meta-algoritmoa, deskonposaketaren maila bakoitzari modu errekurtsiboan aplika daiteke. Demagun, eguzki-sistemaren problemaren ekuazio diferentzial hauek ditugula,

$$\dot{y} = f(y), \quad f(y) = k(y) + g(y) + r(y),$$

non

$k(y)$: keplériarra.

$g(y)$: planeten arteko grabitazio interakzioak.

$r(y)$: planeten eguzkiarekiko erlatibilitate efektua.

Meta-algoritmoa modu honetan aplika daiteke,

1. maila

$$Y_i = y_n + h \sum_{j=1}^s a_{ij} f(Y_j).$$

2. maila, $k = 1, 2, \dots$

$$Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} k(Y_j^{[k]}) + \delta_1^{[k-1]},$$

$$\text{non } \delta_1^{[k-1]} = h \sum_{j=1}^s a_{ij} (g(Y_j^{[k-1]}) + r(Y_j^{[k-1]})).$$

3. maila, , $l = 1, 2, \dots$

$$Y_i^{[k,l]} = y_n + h \sum_{j=1}^s a_{ij} \left(k(Y_j^{[k,l]}) + g(Y_j^{[k-1,l]}) \right) + \delta_2^{[l-1]},$$

$$\text{non } \delta_2^{[l-1]} = h \sum_{j=1}^s a_{ij} r(Y_j^{[k-1,l-1]}).$$

Problema independenteak.

Batzuetan, problemaren eredu sinplifikatua, azpiproblema independentetan bana daiteke,

$$f \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} k_1(y_1) \\ k_2(y_2) \end{pmatrix} + \begin{pmatrix} g_1(y_1, y_2) \\ g_2(y_1, y_2) \end{pmatrix}.$$

Hortaz, azpiproblema bakoitzari dagokion barne-iterazioak independenteak dira eta paraleloan kalkula daitezke. Eguzki-sistema eredu grabitacionala, azaldutakoaren adibide argia da; planeta bakoitzarentzat, $k(y)$ eguzkiarekiko interakzioa azpiproblema independentea da. Dena den, zenbakizko esperimentuetarako aukea eraginkorrena, eredu sinplifikatua bi azpiproblemetan banatzea dela konprobatu dugu: alde batetik, barne-planeten eredu sinplifikatuak osatutako azpiproblema eta beste aldetik, kanpo-planeten eredu sinplifikatuak osatutakoa.

Tolerantzia aldakorra.

Kanpo eta barne-iterazioetarako geratze irizpide berdinak definitu ditugu. Kanpo-iterazioetarako, tolerantzia finkoa erabili dugu eta barne-iterazioetarako, ordea, tolerantzia aldakorra. Tolerantzia aldakorra, kanpo-iterazio (konputazionalki garrestia) bakoitzarentzat, barne-iterazio (konputazionalki merkea) kopuru nahikoak eta beharrezkoak eman dadin, definitu dugu.

Eragozpenak.

Esan dugunez, planteamendu honetan Newton sinplifikatua aplikatu dugu eta ekuazio-sistema lineala LU deskonposaketaren bidez askatu beharrean, puntu-finkoaren iterazioaren bidez askatu dugu. Planteamendu honi IRK-Newton sinplifikatuaren implementazio estandarrarekiko, hiru desabantaila aurkitu dizkiogu:

1. Barne kalkuluen doitasuna.

Planteamendu honen barne-iterazioak konputazioak, implementazioaren oinarritzko doitasunean kalkulatu behar dira. IRK-Newton sinplifikatuaren implementazioaren eragiketa konplexuenak, doitasun txikiagoan kalkula daitezke [9].

2. Jacobiarraren balioztapena.

Aztertutako planteamendu honetan, Jacobiarra iterazio bakoitzean aldatzen denez, iterazio guztietan balioztatu behar da. IRK-Newton sinplifikatu es-tandarrean, Jacobiarra urrats bakoitzean behin bakarrik kalkulatu behar da.

3. Ereduen deskonposaketa.

Ereduen deskonposaketak, nolabaiteko konplexutasuna gehitzen du. Hiru eredu ezberdin bereizi behar ditugu: eredu osoa $f(y)$, eredu sinplifikatua $\tilde{f}(y)$ eta perturbazioa $g = f(y) - \tilde{f}(y)$.

8.3. IRK implementazioaren oinarriak.

Atalen hasieraketa.

Iterazio metodoetan aplika daitezkeen atalen hasieraketa teknika ezberdinak ikertu ditugu.

1. Metodo esplizituak.

Atalen hasieraketa metodo esplizitu bat aplikatz lor daiteke. Hurbilpen merkea lortze aldera, ordena txikiko metodoak aplikatu ditugu: Euler $\mathcal{O}(h)$ eta Euler hobetuaren $\mathcal{O}(h^2)$ metodoa. Atalen hasieraketa metodo hauek, ekuazio diferenzialaren balioztapena beharrezkoa da eta era honetan apli-

katu ditugu:

Euler :

$$Y_i = Y_{i-1} + h(c_i - c_{i-1})f(Y_{i-1}).$$

Euler hobetua :

$$Y_i = Y_{i-1} + h(c_i - c_{i-1})f(k_i),$$

$$\text{non } k_i = Y_{i-1} + \frac{h}{2}(c_i - c_{i-1})f(Y_{i-1}).$$

2. Kepler fluxua.

Eguzki-sistemaren, perturbatutako sistema Kepleriarra denez, atalak Kepler fluxuaren bidez hasieratu daitezke. Kepler fluxuaren funtziokoak, $y(t_n)$ une bateko kokapen eta abiadurak emanik, Δt_n denbora igarotakoan kokapen eta abiadura zehatzak itzultzen du,

$$\text{Keplerflow}(\Delta t_n, y(t_n)) \rightarrow y(t_n + \Delta t_n).$$

Beraz, y_n egoeratik abiatuta, atal bakoitzari dagokion hasieraketa,

$$Y_{n,i}^{[0]} = y(t_n + hc_i), \quad i = 1, \dots, s,$$

Kepler fluxua $\Delta t_n = hc_i$ denbora aurrera eginez lortuko dugu.

3. Interpolazio polinomioak.

Aurreko urratseko ataletako informazioa erabiliz, urrats berriaren atalen hasieraketa lortzen dugu. Problema zurruna ez denean interesgarria da. Era honetako hasieraketa merkea da, ez baita funtziobalioztapen berririk egiten. Era honetan aplikatzen da,

$$Y_{n,i}^{[0]} = y_n + h \sum_{j=1}^s \lambda_{ij} f(Y_{n-1,j}),$$

non λ_{ij} aurre-kalkulatutako interpolazio koefizienteak diren.

Bide honetatik, teknika aurreratuagoak ere aztertu ditugu (maila txikiko polinomioen bidezko hasieraketak). Era berean, B-Serieak [25] teknikan oinarrituz, interpolazio estandarra [70] hobetzeko saiakera egin dugu baina ez dugu hau hobetzea lortu. Metodo estandarrarekin, urrutien dagoen ataletarako hasieraketa txarra lortzen da eta atal askotako metodoentzat, arazo hau, eragozpen handia izan daiteke.

Problema ez-zurrunetarako, interpolazio polinomioen bidezko hasieraketa ona eta merkea lortzen da. Problema zurrunetarako, ordea, atalak aurreko urratseko soluzioarekin hasieratuko ditugu,

$$Y_{n,i}^{[0]} = y_n, \quad i = 1, \dots, s.$$

Geratze irizpidea.

Iterazio metodo bat aplikatzeko, geratze irizpide sendoa finkatza funtsezkoa da. Iterazioak, biribiltze errorearen eragina azaltzen denean geratu behar dira. Batez, iterazioak beranduegi geratzen baditugu, alferrikako iterazioak emango ditugu, eraginkortasunaren kalterako. Bestetik, iterazioak goizegi geratzen baditugu, biribiltze errorea handitu eta trunkatze errorea eragingo ditu.

Geratze irizpide estandarrak aplikatzerakoan, hainbat arazo aurkitu ditugu eta ondorioz, geratze irizpide berri bat definitzeko beharra ikusi dugu. Geratze irizpide berria definitzeko, aukera ezberdinak aztertu ditugu. Garapen honetako, geratze irizpide bertsio nagusienak azalduko ditugu.

Norma.

Iterazio bakoitzaren hobekuntza ($\Delta^{[k]}$) neurtzeko, norma ezberdinak aplikatzen dira. Honakoa da, Hairer-ek puntu-finkoaren iterazioan oinarritutako IRK implemenazioaren geratze irizpidean aplikatzen duen norma,

$$\Delta^{[k]} = \max_{i=1,\dots,s} \|Y_i^{[k]} - Y_i^{[k-1]}\|_\infty.$$

Norma hau zalantzan jarri dugu eta iterazioaren hobekuntza hobeto neurtzen duen norma finkatzen saiatu gara. Honako aukerak aztertu ditugu,

1. Lehen bertsioa.

Diferentziak modu erlatiboan neurtzeko beharrak bultzatuta era honetan definitu dugu,

$$\Delta = \max_{1 \leq j \leq d} \frac{\max_{1 \leq i \leq s} |\Delta Y_i^j|}{(\max_{1 \leq i \leq s} |Y_i^j|) + \delta},$$

non $\delta \approx 10^{-20}$, zatitzalea zero ez izateko finkatzen dugun balio txikia eta $y = (y^1, \dots, y^d)$ den.

2. Bigarren bertsioa.

Kokapenen ($Q_i \in \mathbb{R}^d$) eta abiaduren ($V_i \in \mathbb{R}^d$) izaera ezberdina dela jabetuta, norman bi kontzeptu hauek banatu ditugu,

$$\Delta =$$

$$\max \left(\max_{1 \leq j \leq d} \frac{\max_{1 \leq i \leq s} \|Q_i^j - \tilde{Q}_i^j\|^2}{\max_{1 \leq i \leq s} \|Q_i^j\|^2}, \max_{1 \leq j \leq d} \frac{\max_{1 \leq i \leq s} \|V_i^j - \tilde{V}_i^j\|^2}{\max_{1 \leq i \leq s} \|V_i^j\|^2} \right),$$

non,

$$Y = \begin{pmatrix} Q_1^1 & \dots & Q_1^d & V_1^1 & \dots & V_1^d \\ Q_2^1 & \dots & Q_2^d & V_2^1 & \dots & V_2^d \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Q_s^1 & \dots & Q_s^d & V_s^1 & \dots & V_s^d \end{pmatrix} \quad \tilde{Y} = \begin{pmatrix} \tilde{Q}_1^1 & \dots & \tilde{Q}_1^d & \tilde{V}_1^1 & \dots & \tilde{V}_1^d \\ \tilde{Q}_2^1 & \dots & \tilde{Q}_2^d & \tilde{V}_2^1 & \dots & \tilde{V}_2^d \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \tilde{Q}_s^1 & \dots & \tilde{Q}_s^d & \tilde{V}_s^1 & \dots & \tilde{V}_s^d \end{pmatrix}.$$

3. Hirugarren bertsioa.

Iterazioaren hobekuntza, norma jakin bat aplikatuz zenbaki eskalar bakarra-rekin neurtu beharrean,

$$\Delta^{[k]} = |Y^{[k]} - Y^{[k-1]}| \in \mathbb{R}^{sd} \quad (8.6)$$

matrizea erabiliko dugu. Modu honetan, geratze irizpidea normaren independentea izango da eta gainera, geratze irizpide segurua eraikitzeko aukera emango digu.

Geratze irizpidea.

Oraingoan ere, geratze irizpidearen abiapuntua Hairer-en geratze irizpidea izan da,

$$\Delta^{[k]} = 0 \text{ or } \Delta^{[k]} \geq \Delta^{[k-1]}.$$

Geratze irizpide hau hobetzeko proposamen batzuk garatu ditugu,

1. Lehen bertsioa.

Abiapuntuko geratze irizpidea, batzuetan goizegi geratzen da eta ziurtasuna handitu beharra dago. Horregatik, geratze irizpidean, $k - 2$. iterazioaren informazioa erabiltzea aztertu dugu,

$$(\Delta^{[k]} = 0) \text{ or } (\Delta^{[k]} \geq \Delta^{[k-1]} \text{ and } \Delta^{[k]} \geq 0.81 * \Delta^{[k-2]}).$$

2. Bigarren bertsioa.

Bertsio honetan, $k - 3$. iterazioaren informazioa erabiliz saiakera berri proposatu dugu,

$$(\Delta^{[k]} \leq tol) \text{ or } (zat^{[k]} \geq koeff * \max(zat^{[k-1]}, zat^{[k-2]})),$$

non $zat^{[k]} = \Delta^{[k]} / \Delta^{[k-1]}$, $tol \approx 10^{-16}$ eta $koeff = 10$ den.

3. Hirugarren bertsioa.

$$(\Delta^{[k]} \leq tol) \text{ or } \left(\frac{zat^2}{(1-zat)} \Delta^{[k-1]} \leq tol \right) \text{ or } \left(\frac{zat}{(1-zat)} \Delta^{[k-2]} \leq tol \right),$$

non $zat = \max(zat^{[k-1]}, zat^{[k-2]})$, $tol \approx 10^{-16}$ den.

Geratze irizpide honetan, iteraziotik irteten denean, $z \geq 1$ betetzen bada,

- (1). $\Delta^{[k]} > c tol \rightarrow$ konbergentzia arazoak (exekuzio geratu),
- (2). $\Delta^{[k]} \leq c tol \rightarrow$ birbiltze errorea (exekuzio jarraitu),

non $c \approx 10^4$ konbergentzi koefizientea den.

4. Laugarren bertsioa.

Azkenik, honakoa proposatu dugu: (8.6) $\Delta^{[k]} \in \mathbb{R}^{sd}$ izanik, iterazioak $k = 1, 2, \dots$ jarraitzea, $\Delta^{[k]} = 0$ bete arte edo honako baldintza bi iterazio jarraietan betetzen den artean,

$$\forall j \in \{1, \dots, sd\}, \quad \min \left(\{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} / \{0\} \right) \leq |\Delta_j^{[k]}|.$$

Geratze irizpide honetan, iterazioa gertatu bada $\exists j, \Delta_j^{[K]} \neq 0$ delarik, orduan urratsa onargarria den ala ez erabaki behar dugu.

8.4. IRK Newton: konplexutasun analisia.

$(I_s \otimes I_d - h A \otimes J)\Delta Y = r$ ekuazio sistema modu eraginkorrean askatzeko, implementazio estandarraren eta gure implementazioen konplexutasunak konparatuko ditugu.

Implementazio estandarra.

Butcher edota Hairer-en implementazio estandarraren konputazio bi modutan egin daiteke:

1. Zenbaki konplexuak.

A matrizearen diagonalizazioak balio propio konplexuak ditu,

$$P^{-1}AP = \begin{bmatrix} \gamma_1 & & & \\ & \bar{\gamma}_1 & & \\ & & \gamma_2 & \\ & & & \bar{\gamma}_2 \\ & & & & \gamma_3 \\ & & & & & \bar{\gamma}_3 \end{bmatrix},$$

eta ekuazio-sistema, zenbaki konplexuen aritmetika erabiliz ebatzi daiteke.

$$\begin{aligned}(I - h\gamma_j J) X &= b, \quad j = 1, \dots, 3, \\ (I - h\bar{\gamma}_j J) X &= \bar{b}.\end{aligned}$$

2. Zenbaki errealkak.

Zenbaki konplexuekin ez bada lana egin nahi, zenbaki errealeko deskonposaketa baliokidea,

$$\gamma_j = \alpha_j + i \beta_j,$$

$$P^{-1}AP = \begin{bmatrix} \alpha_1 & -\beta_1 & 0 & 0 & 0 & 0 \\ \beta_1 & \alpha_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_2 & -\beta_2 & 0 & 0 \\ 0 & 0 & \beta_2 & \alpha_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_3 & -\beta_3 \\ 0 & 0 & 0 & 0 & \beta_3 & \alpha_3 \end{bmatrix}$$

Hairer-en implementazioan,

- s bikoitia bada $\rightarrow (2d \times 2d)$ tamainako $[s/2]$ LU deskonposaketa.
- s bakoitia bada $\rightarrow (2d \times 2d)$ tamainako $(s+1)/2$ LU deskonposaketa.

Gure implementazio berria.

\bar{A} matrizearen, $\bar{A} = P^{-1}DP$ diagonalizatzen dugu eta D matrizeak, irudikari puruak ditu,

$$\bar{A} = Q^{-1}RQ, \quad R = \begin{bmatrix} 0 & -\gamma_1 & & & & \\ \gamma_1 & 0 & & & & \\ & & 0 & -\gamma_2 & & \\ & & \gamma_2 & 0 & & \\ & & & & 0 & -\gamma_3 \\ & & & & \gamma_3 & 0 \end{bmatrix}$$

Gure implementazioan,

- s bikoitia bada $\rightarrow (d \times d)$ tamainako $[s/2] + 1$ LU deskonposaketa.
- s bakoitia bada $\rightarrow (d \times d)$ tamainako $(s+1)/2$ LU deskonposaketa.

Konplexutasun konparaketa

Lehenengo eragiketa aljebraikoen konplexutasunak gogoratuko ditugu,

$$\text{LU deskonposaketa : } 2s^3d^3/3 + \mathcal{O}(d^2),$$

$$\text{Back substitution : } 2s^2d^2 + \mathcal{O}(d),$$

$$\text{inv : } 2s^3d^3$$

Bi implementazioen konplexutasunen laburpena,

8.2. Taula:

s	LU		Back Substitution	
	Estandarra	Berria	Estandarra	Berria
$2m$	$\frac{8m}{3}d^3$	$\frac{2}{3}(2m+1)d^3$	$4m(2d^2)$	$(6m+4)d^2$
$2m+1$	$\left(\frac{8m}{3} + \frac{2}{3}\right)d^3$	$\left(\frac{4m}{3} + \frac{2}{3}\right)d^3$	$(8m+2)d^2$	$(6m+4)d^2$

8.5. Doitasun altuko konputazioak.

Extended precision.

1. JM Muller,

The legacy x87 instructions of the IA-32 instruction set can operate on a double-extended precision format with 64 bits of significand and 15 bits of exponent. The corresponding floating-point operators can be instructed to round to single, to double, or to double-extended.

The IA-64 instruction set also defines several double-extended formats, including one 80-bit format compatible with IA-32 and one 82-bit format with a 64-bit significand and a 17-bit exponent. The two additional exponent bits are designed to avoid intermediate overflows in certain computations on 80-bit operands.

Rump's example(2.orrialdea).

1. INFLUENCE OF THE METHODS...

In addition, in all ephemerides published by the present time the coefficients of the Chebyshev polynomials are given only with 16 decimal places, that is, in the form of 64-bit computer numbers with floating point (the so-called double-precision numbers). However, calculations with increased precision are more often used nowadays, for example, using 80-bit computer numbers with floating point which have 19 decimal places [10, 11] or 128-bit numbers with 34 decimal places (the so-called quadruple precision numbers).

2. Laskar-ek [73]:

The numerical integrator is the same symplectic integrator of that was used in the La2004 solution but it was entirely rewritten in C in order to access the extended precision of the Intel architecture.

Here, we integrated the solutions in extended precision on Xeon Intel processors, which allows arithmetics in 80 bits instead of 64 bits in double precision.

3. DE ephemerideak:

QIVA doitasun laukoitzeko (128-bit) bertsioari deitzen zaio: mugimenduaren ekuazioen Newton zatia,

4. INPOP ephemerideak.

Hiru motako integrazioak aipatzen ditu: (a) double, (b) extended precision, (c) extended precision with corrector step (Nevertheless, it was possible to obtain an additional order of magnitude improvement by using a single addition in simulated quadruple precision in the corrector step with a very small overhead).

5. EPM ephemerideak (2013).

80-bites instead of 64-bites floating point calculation was realized in the numerical integration rewriting all ERA subroutines with the extended precision

8.6. Paralelizazioa.

Parareal Algorithms.

Using more number of nodes requires more computational cost to advance a step-size (h), and so, we have to known which scheme is most efficient. We

consider one scheme is more efficient than another, if this scheme achieves the same precision with the smaller value of h/s . It seems the integrator $s=6$ is the most efficient. But since IRK is parallelizable at the node level, using more nodes (s) can improve the speed of the implementation if multiple processors are used. In this sense, Parareal algorithm has been proposed by several authors [?], [?]. This is a subject to future work.

Baztertu ditugun gaiak.

1. Molekulen simulazioa.

Molekula simulazioko problema bat aztertzea esfortzu handia izango dela iruditu zaigu. Understanding Molecule simulation liburuan agertzen den MTS metodoak aipatu ditugu: problema hauetan egiten den indarraren banaketa: $F = F_{\text{slow}} + F_{\text{fast}}$, gure planteamenduaren baliokidea dela ikusi daiteke (honako planteamenduari denboraren birparametrizazioa aplikatuta, $F = K + g$ baliokidea dela pentsa daiteke).

8.7. Etorkizuneko lanak.

Kepler fluxuaren aldagai aldaketa.

Orain Fluxuaren aldagai aldaketa aplikatu dugunean, IRK puntu-finkoan oinarritu gara. Esperimentuetan, oso azkar konbergitzen duela ikusi dugu. Bestalde, guzki-sistemaren eredutan kepleriarrok ez diren hainbat perturbazio izan daitezkeela; aldagai hauen (W), konbergentzia azkartzeko, Newton simplifikatua era-biltzea komeniko zaigula uste dugu.

Zenbakizko integrazio metodo inplizituak.

Gauss, Chebichev, ...

Denbora birparametrizazioa.

Exzentrititate handiko orbitadun sistementzako hainbat “erregularizazio simpletiko integratzaileak” aztertu dira: Levison and Duncan (1994). “RMVS”: Regularized mixed variable symplectic integrator”, Mikkola (1997), Fukushima (2001), Beus(2003).

Erregularizazioa aplikatzen dituzte metodo simpletiko ezedinen “review” honakoa dugu: “Rauch and Holman (1999). Dynamical chaos in the Wisdom-Holman integrator”.

9. Kapitulua

Ondorioak.

Ideiak

Lehenengo puntu-finkoaren iterazioan oinarritutako eta Newton simplifikatuaren iterazioan oinarritutako IRK implementazioak jorratu dugu. Bi implementazio hauean, biribiltze errorearen ikuspegitik optimoa izatea lehenetsi dugu. Newton simplifikatuaren IRK implementazio estandarra konplexua eta konputazionalki garestia denez, modu eraginkorrean kalkulatzeko proposamen egin dugu.

Eguzki-sistema integratzeko, IRK metodo partizionatuak Newton simplifikatuaren IRK metodoak baino egokiagoak direla baiezttatu dugu. Dena den, puntu-finkoaren implementazioak energia drift-a azaltzen du eta hau garrantzitsua balitz, Newton-en implementazioa aplikatu beharko litzateke.

Laburpena

Discovery Through the Power of Mathematics, Physics, and the Imagination 'New Worlds, New Horizons in Astronomy and Astrophysics' liburuko, ondorio hau lapurtu dugu, ISBN 978-0-309-15802-2 | DOI 10.17226/12951

Discovery Through the Power of Mathematics, Physics, and the Imagination.

Finally, it is important to remember that many of the most far-reaching and revolutionary discoveries in astronomy were not solely the direct result of observations with telescopes or numerical simulations with computers. Rather, they also sprang from the imagination of inspired theorists thinking in deep and original ways about how to understand the data, and making testable predictions about new ideas.

In the coming decade, major challenges loom that require the development of fundamental new theories. Observations and computer simulations are necessary components, but to complete the path from discovery to understanding, theorists will need to freely exercise their imaginations.

I. Atala
Eranskinak

A. Eranskina

Zientzia konputazioa.

A.1. Sarrera.

Azken hamarkadetan, zientzia konputazioaren hazkundeak oso handia izan da eta bere erabilera ia zientzia arlo guzietara zabalduta da [42]. Zientzialariek ahalmen handiko tresna berria (zenbakizko simulazioa) eskuragarri dute, neurri handi batetan konputagailuen teknologiaren garapen handiari esker. Egungo oinarrizko konputagailuek, orain urte gutxitako superordenagailuen ahalmen berdina dute eta superordenagailuen konputazio gaitasuna ere, maila berdinean hazi da. Zenbakizko algoritmoek ere, garapen handia izan dute; algoritmo eraginkorragoak, idei berriak sortuz eta konputagailuen gaitasunei egokituz, garatu dira.

Konputazioaren alde garestiena, memoria eta prozesadorearen arteko datu mugimendua da. Prozesadoreak gero eta azkarragoak dira, baina memori atzipenaren abiaduraren hobekuntza mugatuagoa dago. Horregatik algoritmo eraginkorrapak, prozesadorearen konputazio arik eta handiena, memoria komunikazio arik eta txikiarenarekin, diseinatu behar dira.

Implementazio baten eraginkortasuna ez da exekuzio denboraren arabera bakanrik neurtu behar. Hori bezain garrantzitsua da kode ona idaztea [100] eta zentzu honetan hiru ezaugarri hauek bereziki zaindu behar dira:

1. Errorerik gabeko kodea.
2. Kode argia idaztea.
3. Etorkizunean erraz aldatu daitekeen kodea.

Atal honetan, zientzia konputazioaren funtsezko osagaiak azalduko ditugu. Lehenengo, konputazioaren eraginkortasuna aztertzeko eta exekuzioak neurtzeko tresnen zehaztapenak emango ditugu. Ondoren, konputagailu hardware berriak

eta paralelizaio gaitasunak laburtu dugu. Jarraian software ikuspegitik, programazio lengoaiak eta aljebra linealerako (LAPACK) liburutegia landu dugu. Azkenik, konpiladoreari buruzko argibide batzuk eman ditugu.

A.2. Eraginkortasuna

Zientzia konputazioaren implementazio berri baten eraginkortasuna neurtzeko, koma-higikorreko eragiketa kopurua (*flops*) erabili ohi da. Problema handia denean, datuen mugimendua koma-higikorreko eragiketak baino garestiagoa da eta eraginkortasuna, eragiketa kopuruaren arabera neurtzea okerra izan daiteke.

Prozesadoreen maiztasun-abiadura hertzetan neurtzen da, hau da, *makina ziklo segundoko* kopuruaren arabera. Une honetako prozesadoreak gigahertz (Giga = 10^9) mailakoak dira. Koma-higikorreko oinarrizko eragiketa bat ($\oplus, \ominus, \otimes, \oslash$) exekutatzeko ziklo gutxi batzuk behar dira eta beraz, 1 GHz-ko prozesadore batek $> 10^8$ koma-higikorreko eragiketa segundoko exekutatzen ditu (> 100 megaflops) [87].

Adibidea. Demagun A , B eta C ($n \times n$) dimentsioko matrizeak ditugula eta $C = AB$ matrize arteko biderketa egiteko behar dugun denbora jakin nahi dugula.

$$c_{ij} = \sum_{i,j=1}^n a_{ij} * b_{ji}.$$

- c_{ij} gai bakoitzak kalkulatzeko n biderketa eta $(n-1)$ batura egin behar ditugu.
- C matrizeak n^2 osagaia ditu $\Rightarrow \mathcal{O}(n^3)$ koma-higikorreko ariketak exekutatu behar dira.

Matrizearen tamaina $n = 100$ bada, orduan $\mathcal{O}(n^3) = 10^6$ eragiketa egin behar ditugu. 1-GHz prozesadore batean exekutatzeko, 10^{-2} segundo baino gehiago beharko genituzke. Baino matrize honek, 3.9 MB memoria beharrezkoa du eta konputagailuaren Cache memoria baino handiagoa dela suposatuz, exekuzio denboran datuen mugimenduaren eragina nabarmena izango da.

Konputazio gaitasuna (*peak*), hardwareak fisikoki exekutatu dezakeen eragiketa kopuru maximoa bada, aplikazio gehienak, konputagailuaren konputazio gaitasunaren %10 baino gutxiagorekin exekutatzen dira. Eraginkortasun horren txikia, memoria irakurketa/idazketetan galtzen da. Azpimarratu nahi dugu, t_f eragiketa aritmetiko bat egiteko denbora bada eta t_m , datu bat memoria nagusitik cache memoriara mugitzeko denbora bada,

$$t_f \ll t_m,$$

eta etorkizunean, differentzia hau handituz joango dela. Beraz, kodearen execuzioa azkartzeko derrigorrezkoa da konputagailuan memorien arteko datuen mugimendua minimizatzea.

Execuzio denboren neurketa.

Unixeko *time* agindua, konputazioen denborak ezagutzeko erabili daiteke [87]:

```
S time ./a.out
<kodearen irteera >

real 0m38.856 s
user 0m38.789 s
sys 0m0.004 s
```

Agindu honekin, *./a.out* C programa exekutatuko da eta ondoren, programa exekutatzeko behar izan duen denboraren informazioa pantailaratuko du:

- *real*: hasi eta bukatu arteko denbora (*wall-time* edo *elapsed-time*).
- *user*: prozesadoreak gure programa exekutatzen erabili duen denbora (*CPU-time*).
- *sys*: programa exekutatu ahal izateko, sistema eragile lanetan emandako denbora.

Programa osoaren konputazio denborak ezagutu beharrean, kodearen zati bat neurtu nahi dugunean, C lengoaiaren bi funtzio hauek erabilgarriak ditugu:

1. *clock()*.

Funtzioaren bi deien arteko CPU denbora neurtzeko erabiliko dugu (**CPU time**).

```
#include <time.h>

clock_t clock0, clock1;
double elapsed_cpu_time;

clock0= clock();

<neurtu nahi den kodea>

clock1=clock();

elapsed_cpu_time=(clock1 - clock0)/CLOCKS_PER_SEC;
```

2. `time()`.

Funtzioaren bi deien artean igarotako denbora neurtzeko erabiliko dugu (**elapsed-time**).

```
#include <time.h>

time_t time0, time1;
double elapsed_time;

time(&time0);

<neurtu nahi den kodea>

time(&time0);

elapsed_time=difftime(time1, time0);
```

CPU denborari buruzko argibide bat ematea komeni da. Neurtzen ari garen kodea sekuentzialki exekutatzen bada, hau da, hari (*thread*) bakarrekoa, orduan kode zati hori exekutatzeko erabili duen CPU denbora itzuliko du. Aldiz, kodea paraleloan exekutatzen bada, orduan , hari guztien CPU denboren batura itzuliko ditu.

Adibidea. Argiago azaltzeko, $C = AB$ matrize biderketaren kodearen bi exekuzioen denboren neurketak zehaztuko ditugu:

1. 200×200 tamainako, bi matrizeen biderketa sekuentzialaren denborak hauek izan dira:

```
elapsed-time=2.1 s
elapsed-cpu-time=2.07 s
```

2. 200×200 tamainako, bi matrizeen biderketa paraleloaren (hariak=2) denborak hauek izan dira:

```
elapsed-time=1.0 s
elapsed-cpu-time=2.35 s
```

Elapsed-time deiturikoa, kode paraleloen denborak neurtzeko irizpidea da. Programazio paraleloan, algoritmoen exekuzio denborak egokien neurtzen duen aldagaia da baina, une berean programa bakarra exekutatzea behartuta gaude.

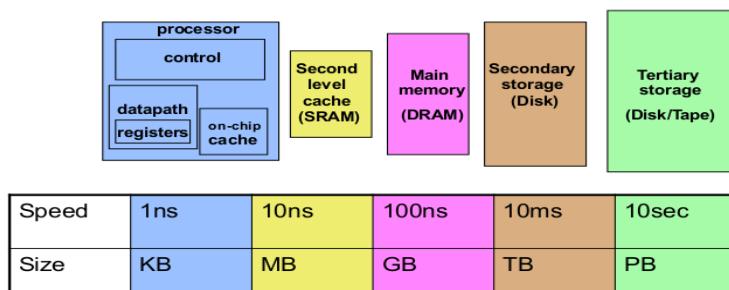
A.3. Hardwarea.

Orokorrean, gaur-egungo konputagailuak (super-konputagailu, eramangarri,...) paleloak dira. 1986 – 2002 urteen artean, txip barruko transistore dentsitatea handitzen zen heinean, prozesadore bakarreko konputagailuen eraginkortasuna hobetuz joan zen. Baino teknologi honen garapena muga fisikoetara iritsi zenean, bide honetatik konputagailuen abiadura hobetzea ezinezkoa bilakatu zen. Horrela, 2005.urtetik aurrera fabrikatzaileek konputagailuen gaitasuna hobetzeko, txipan prozesadore bat baino gehiago erabiltzea erabaki zuten.

Konputagailuen eredu aldaketa honen ondorioz, algoritmo azkarrak garatzeko kodearen paralelizazio gaitasunari heldu behar zaio. Programazio paralelo teknikak implementatzeko, beharrezko da prozesadore berrien hardware arkitektura berriak ulertzea. Gaia nahiko konplexua izanik, ikuspegi orokor bat ematera mugatuko gara.

Memori hierarkia.

Memorien arteko datuen komunikazioak, algoritmoaren eraginkortasuna baldintzatuko du eta zentzu honetan, konputagailuaren memoria hierarkiaren kudeaketa egokia egitea funtsezkoa da. Konputagailuaren memoria mota ezberdinaren hierarkia ([A.1.irudia](#)) eta funtzionamendua deskribatuko dugu.



A.1. Irudia: Memoria hierarkia.

CPU-k, koma-higikorreko eragiketak exekutatzen ditu: erregistroetatik datuak irakurri, eragiketak kalkulatu eta emaitza erregistroetan idazten ditu. Memoria nagusia eta erregistroen artean, 2 edo 3 mailako cache memoria dugu: lehen cache memoria (L_1) txikieta eta azkarrena da, eta beste mailak (L_2, L_3, \dots), handiagoak eta motelagoak dira. Memoria nagusian, exekutatzen diren programak eta datuak gordetzen dira (1 – 4 GB artekoa). Azkenik, disko gogorrean konputagailuko datuak (argazki, bideo,...) eta erabilgarri ditugun programa guztiak gordetzen dira.

CPU-k datu bat behar duenean, memoria hierarkian zehar bilatuko du: lehenik $L1$ cachean, ondoren $L2$ cachean,...eta hauetan ez badago, memoria nagusira joko du. Memoria nagusi eta cache memoria arteko irakurketa eta idazketa guzti hauetan, informazio konsistentzia mantentzeko hainbat arau aurrera ematen dira.

Cache memoria lerroka egituratuta dago eta lerro bakoitza 64 edo 128 bytetz (8 edo 16 doitasun bikoitzeko zenbaki) osatuta dago. Programa batek datu bat behar duenean, memoria nagusitik lerro tamainako datu taldea (memorian jarraian gordetako datuak) irakurriko du eta cachean idatziko du. Programatzaleak, algoritmo eraginkorrik implementatzeko memorien arteko komunikazio hau minimizatzen saiatu behar du eta horretarako, implementazioaren diseinua datuen memoria atzipen jarraian oinarritu behar du. Ezaugarri hau, *spatial/data locality* izenaz ezaguna da eta helburua, cacheria ekartzen diren datuak, memoria nagusian idatzi aurretik gutxienez behin erabiltzea da.

Adibidea. Adibide honetan, $A = (a_{ij})_{i,j}^{n,m}$ matrize baten osagaien batura ($\text{sum} = \sum_{i,j=0}^{n,m} a_{ij}$) kalkulatzeko bi implementazio aztertuko ditugu. C lengoian matrizeak lerroka gordetzen dira ($n = m = 100$),

$$A = \begin{pmatrix} 1 & 2 & 3 & \dots & 100 \\ 101 & 102 & 103 & \dots & 200 \\ 201 & 202 & 203 & \dots & 300 \\ \dots & \dots & \dots & \dots & \dots \\ 9.901 & 9.902 & 9.903 & \dots & 10.000 \end{pmatrix}.$$

eta horregatik, lehen aukera bigarrena baino eraginkorragoa izango da. Lehen implementazioan, kanpo iterazioa lerroka (Algoritmoa 29): matrizearen lehen osagaia $a(1, 1)$ behar dugunean, memoria nagusitik Cacheria osagai honetaz gain, jarraiko 16 osagaia ekarriko dira ($a(1, 1), a(1, 2), \dots, a(1, 16)$). Honela, hurrengo 15 batura egiteko behar ditugun datuak Cachean eskura izango ditugu memoria irakurketa berririk egin gabe. Bigarren implementazioan, kanpo iterazioa zutabeka (Algoritmoa 30): bigarren osagaia ($a(2, 1)$) gehitzeko memoria irakurketa berri bat egin behar dugu.

Arkitektura motak.

Prozesadore anitzeko konputagailu hardware berriak, konplexuak eta heterogeneoak dira. Egoera honetan, programatzaleak zaitasun handiak ditu arkitektura berrieik eskaintzen dituzten gaitasunak ondo kudeatzeko [79].

Lehen hurbilpen modura bi sistema nagusi bereiziko ditugu: memoria konpartitutako eta memoria banatutako sistemak. Memoria konpartitutako sistemetan,

```

int n;
double a[n][m];
sum = 0;
for i ← 1 to n do
    for j ← 1 to m do
        sum += a(i, j);
    end
end

```

Algoritmoa 29: Memoria atzipena eraginkorra.

```

int n;
double a[n][m];
sum = 0;
for j ← 1 to m do
    for i ← 1 to n do
        sum += a(i, j);
    end
end

```

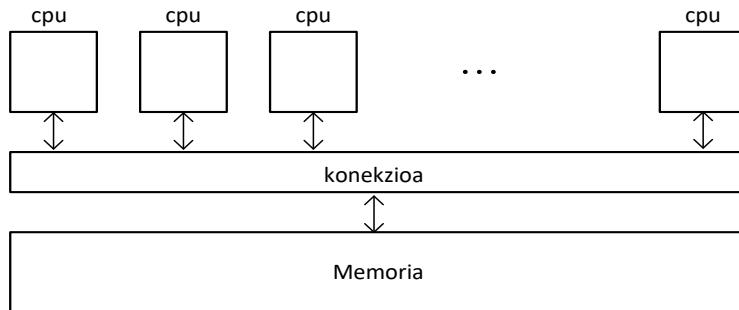
Algoritmoa 30: Memoria atzipena ez-eraginkorra.

prozesadore guztiekin memoria osoa konpartitzen dute eta implizituki konpartitutako datuen atzipenaren bidez komunikatzen dira ([A.2.](#) irudia). Memoria banatu-takotako sistemetan aldiz, prozesadore bakoitzak bere memoria pribatua du eta esplizituki bidalitako mezuen bidez komunikatzen dira ([A.3.](#) irudia). Aipatzeko da, sistema handietan bi memoria motak nahasten direla, hau da, batetik goiko mailan memoria banatuta alde bat eta bestetik, konputazio unitate bakoitzak memoria konpartitutako aldea.

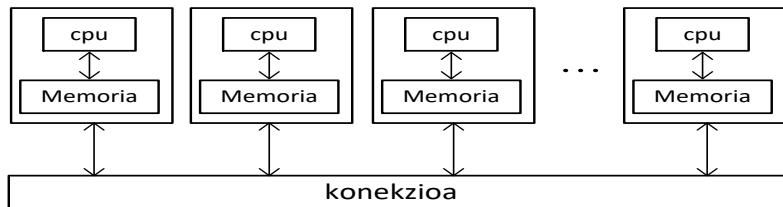
Hirugarren sistema osagarria ere aipatuko dugu, GPU (Graphical Processor Unit) unitateetan oinarritutako konputazioa. Joko-en eta animazio industrian, gra-fiko oso azkarrak beharrak bultzatuta sortutako teknologia da. Oinarrian, imajinak pantailaratzeko prozesagailu asko paraleloan lan egiten dute eta azken hamarkadan, *GPU* unitate hauek zientzia konputaziora zabaldu dira.

Oinarrizko konputagailu paraleloa.

Gure lanerako, oinarrizko konputagailu paraleloak konsideratuko ditugu: memo-ria konpartitutako eta prozesadore anitzeko unitate bat edo gehiagoz osatutako sistemak. Prozesadore anitzeko unitate bakoitzak txipean CPU bat baino gehia-go ditu. Normalean CPU bakoitzak *L1* bere cache memoria du. Aipatzeko da, era honetako sistemetan prozesadore kopurua mugatua dela (normalean ≤ 32)



A.2. Irudia: Memoria konpartitutako sistemak.

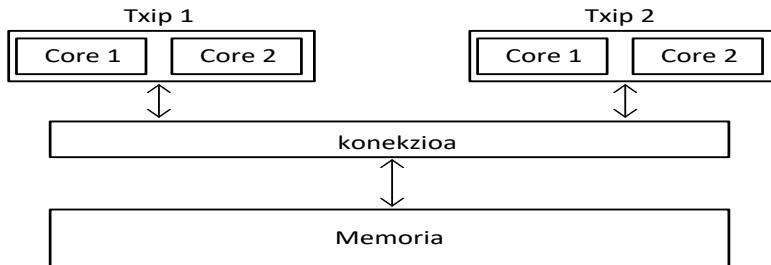


A.3. Irudia: Memoria banatutako sistemak.

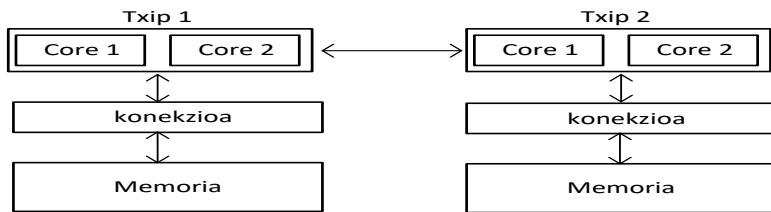
[87, 99].

Memoria konpartitutako sistemen artean bi mota bereiziko ditugu:

1. UMA sistemak (uniform memory access). Txip prozesadore guztiak zuzean memoria konektatuta daude eta guztiak atzipen denbora berdina dute ([A.4..irudia](#)).
2. NUMA sistemak (nonuniform memory access). Txip prozesadore bakotza, hardware berezi baten bidez zuzenean memoria bloke batikonektatuta dago. Zuzenean konektatuta dagoen memori blokearen atzipen denbora, beste txipan zehar konektatutako memoriaren atzipena baina azkarragoa da ([A.5..irudia](#)).



A.4. Irudia: Memoria konpartitutako sistemak (UMA).



A.5. Irudia: Memoria konpartitutako sistemak (NUMA).

Bektorizazioa (SIMD).

Konputagailuek, *Single Instruction Multiple Data* (SIMD) instrukzioetan oinarritutako paralelizazio eskaintzen dute [83, 5]. CPU-ak eragiketa berdina aplikatzen du bektore erregistroan gorde diren zenbaki multzo bati. Bektore erregistro hauen tamaina 256-biteko ingurukoa ($256/64 = 4$ doitasun bikoitzeko zenbakia) izan ohi da eta oro har, oinarrizko eragiketa aritmetikoak (batuketa, kenketa, biderketa eta zatiketa) aplika daitezke.

Adibide moduan, hurrengo pseudokodearen bidez, iterazio bakoitzean 4 osagaien bektore baten batura erakutsiko nahi dugu,

```

for  $i = 0$ ;  $i < n$ ;  $i += 4$  do
     $A[i : (i + 4)] = A[i : (i + 4)] + B[i : (i + 4)];$ 
end

```

Algoritmoa 31: SIMD (bektorizazioa).

A.4. Programazio lengoaiak.

Fortran eta C, aplikazio zientifikoetan gehien erabiltzen diren programazio lengoaiak dira [54]. Fortran (formula translation) 1950 hamarkadan garatutako goi-mailako lehen lengoia izan zen eta oraindik ere, oso zabaldua da. Fortran estandarraren hainbat bertsio sortu dira: Fortran 66, 77, 90, 95, 2003 eta 2008. Hauetako bertsio bakotzean funtzionalitate berriak eta C lengoaiarekin lan egiteko bateragarritasuna gehitu zaizkio. C lengoia, 1970 hamarkadan jaio zen eta hardwarearekiko hurbiltasun "ezaugarri nagusiak, konpiladoreari kode eraginkorra sortzeko aukera ematen dio. C lengoia *UNIX* sistema eragileari lotuta jaio zen eta hurrengo garapenetan bere izaera askea mantendu du. C lengoaiaren estandarrak 1989, 1999 eta 2011 dira.

Fortran eta C lengoaiak, ez dituzte kodea paraleloan exekutatzeko tresnarik, hau da, ez dago konputazio banatu, eta prozesadore ezberdinenean aldi berean exekuzioak zehazteko modurik. Konputazioa paraleloa implementatzeko, bi dira interfaze aplikazio programa (*API*) moduan implementatuta dauden sistema nagusiak [87]:

1. *MPI* (Message Passing Interface). Erabiliena da, memoria banatutako sistemetarako pentsatua baina memoria konpartitutako sistemeten ere aplika daitekeena.
2. *OpenMP* (Open Specifications for MultiProcessing). Erabiltzeko errazagoa eta memoria konpartitutako sistemeten bakarrik aplika daitekeena.

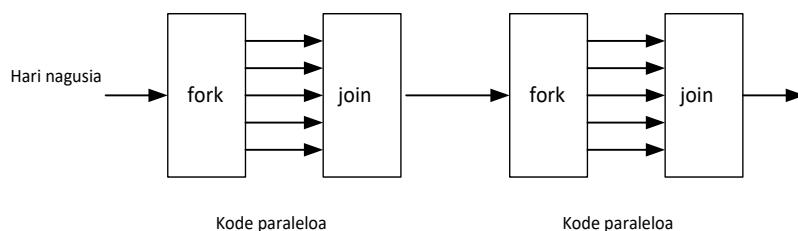
Eraginkortasun altuko konputazioaren (*high performance computing*) programazioa konplexua da: espezializazio handikoa eta konputagailuen hardware jakin baterako egokitutakoa. Zaitasun hauek, proiektu zientifikoak aurrera ateratzeko eta mantentzeko arazo asko eragiten ditu. Azken urteotan, eragozpen hau gainditzeko, programazio lengoia interesgarriak sortu dira (adibidez, Julia [14] edo Chapel [10]) baina oraindik, hauen arrakasta ikustekoa da.

Azkenik, zientzia konputazioan *problemak ebazteko inguruneak* (Problem Solving Environments) deituriko softwareak aipatuko ditugu. Ingurune hauek, programazio leihoko interaktibo batean, goi-mailako lengoai batean implementazioen garapena eta emaitzen azterketa egiteko aukera eskaintzen dute. Matlab eta Mathematica [104] programazio ingurune nagusienak dira. Guk Mathematica bi modutara erabili dugu. Lehenik, prototipoak garatzeko tresna gisa: idei berriak garatu eta probatu, implementazioa C lengoian egin aurretik. Bigarrenik, gure C implementazioen esperimentuak Mathematica ingurunetik exekutatu eta emaitzak grafikoki aztertu ditugu. Era honetan, irakurleari Mathematicako dokumentuetan esperimentuen zehaztasun guztiak eta esperimentu berdina errepikatzeko aukera ematen diogu.

OpenMP

OpenMP [2] memoria konpartitutako sistemetan programazio paraleloa exekutatzeko interfaze aplikazio programa (*API*) da. *OpenMP* programazioan, memoria osoaren atzipena duten prozesadore multzo batek osatzen du konputazio sistema.

Hasieran programaren hari bakarra prozesadore batean exekutatuko da, kode-paraleloko unera iritsi arte. Orduan, hari multzo independenteak exekutatuko dira une paraleloa bukaera iritsi arte. Exekuzio kontrolari, *fork-join* eredu deitzen zaio eta grafikoki (A.6. irudian) adierazi dugu.



A.6. Irudia: OpenMp programazio eredu.

- OpenMP programen hasieran prozesu bakarra dago, hari (thread) nagusia.
- FORK: hari nagusiak, hari talde paraleloa sortzen du.
- JOIN: kode-paraleloko hari guztiak bukatzen dutenean (sinkronizazioa), hari nagusiak soilik jarraitzen du.

Paralelizazioan hari kopurua zehaztu behar da, eta ohikoa izaten da hari bat prozesadore bakoitzeko sortzea. Konpilazio direktiben bidez (C kodean *pragma* izeneko preprozesadore aginduak), paralelizazioa nola exekutatu behar den zehazten da.

- Kode paralelizagarria adierazi.
- Hariaren datu pribatuak zehaztu.
- Harien arteko sinkronizazioa.

Adibidea1. C lengoaian, *OpenMP* konpilazio direktibak adierazteko *pragma* hitza lerroaren hasieran idatziko dugu. Adibide honetan, programaren *for-iterazioa* paraleloan exekutatu daitekeela eta hari kopurua bi dela zehaztu dugu.

```
#    include <omp.h>

    int thread_count=2;

# pragma omp parallel for num_threads(thread_count)
for (i = 0; i<n; i++)
{
    ! Aginduak
}
```

Konpilazioan, *-fopenmp* aukera zehaztu behar dugu,

```
$ gcc -g -Wall -fopenmp adibidea.c -o adibidea.o.
```

Orokorrean, defektuz aldagai guztiak harien artean konpartituta daude eta aldagaiak pribatuak direla zehazteko, esplizituki adierazi behar da. Goiko adibidea salbuespena da; *for* iterazioaren kontagailua (adibideko *i* aldagai) pribatua da.

Algoritmo batean, kodearen zati bat da paralelizagarria [87]. Suposa dezan gun, konputazioaren %50 sekuentziala dela eta beste %50 paraleloan exekutatu daitekeela. Zati sekuentzialak, lortu daitekeen konputazio optimoena (zati paralelizagarriaren exekuzio denbora zero dela konsideratzea) mugatuko du eta beraz, gehienez exekuzio sekuentziala baino bi aldiz azkarrago izango da. Kontzeptu hau orokortzen badugu, konputazioaren $(1/S)$ sekuentziala eta gainontzekoa, $(1 - 1/S)$ paralizagarria konsideratuz, orduan kode optimoena prozesadore kopurua edozein delarik, S faktorea hobea izango da. T_s makina sekuentzial batean exekuzio denbora izendatzen badugu, P prozesadore kopurua erabiliz lortuko den konputazio denbora T_p ,

$$T_p = (1/S)T_s + (1 - 1/S)T_s/P,$$

eta prozesadore kopuru oso handia konsideratuko bagenu,

$$T_p \rightarrow (1/S)T_s, \quad P \rightarrow \infty.$$

Konputazio paraleloan, T_p denborari paralelizazioak duen gainkarga gehitu behar zaio. Gainkarga hau, kontzeptu ezberdinez osatuta dago eta milsegunduko mailako eragiketak izan daitezke.

- Prozesuak edo hariak sortzeko denbora.
- Sinkronizazio denbora.
- Datu konpartitutako komunikazioa.

A.5. Aljebra lineal dentsorako liburutegiak.

Zenbakizko integrazioen aljebra lineala, konputazioaren alde konplexua da. Aljebra linealeko eragiketak implementatzen dituzten kalitate handiko liburutegiak daude eta implementazio berriak, liburutegi hauetan oinarritzea gomendagarria da [55]. Liburutegi hauek, ondo probatutako softwareak dira, konplexutasun handikoak, modu seguruan eta azkarrean exekutatzeko diseinatu dira.

Hauek dira, aljebra linealerako liburutegi aipagarrienak:

1. BLAS (Basic Linear Algebra Subroutines): matrize eta bektoreen arteko eragiketa aritmetikoak biltzen dituen liburutegia.
2. LAPACK (Linear Algebra Package): aljebra linealaren problemak ebazteko liburutegia.

Implementazio hauen funtzioak *Fortran* lengoain garatuta daude eta ezaugarri hauek dituzte:

1. Datu-mota hauetarako aplika daitezke:
 - (a) S: doitasun arrunta (*float*, 32-bit).
 - (b) D: doitasun bikoitza (*double*, 64-bit).
 - (c) C: zenbaki konplexua doitasun arruntean (*complex*).
 - (d) Z: zenbaki konplexua doitasun bikoitzean (*complex double*).
2. Matrize dentsoetarako liburutegiak dira. Matrize egitura hauek zehaztu daitezke.
 - (a) Matrize orokorrak.
GE=General; GB=General Band.
 - (b) Matrize simetrikoak.
SY=SYmmetric ; SB=Symmetric Band; SP=Symmetric Packed.
 - (c) Hermitiar matrizeak.
HE=Hermitean ; HB=Hermitian Band; HP=Hermitian Packed.
 - (d) Matrize triangularrak.
TR=TRiangular ; TB=TRiangular Band; TP=Triangular Packed.

BLAS.

BLAS liburutegiak [1], bektore eta matrizeen arteko funtziotako estandarrak biltzen ditu. Liburutegia, 142 errutinaz osatuta dago eta hauek, hiru taldeetan sailkatzen dira:

1. BLAS-1: $\mathcal{O}(n)$ bektore-bektore eragiketak.

Adibidea. $y = \alpha * x + y$, non $\alpha \in \mathbb{R}$, eta $x, y \in \mathbb{R}^n$.

$2n$ eragiketa aritmetiko eta $3n$ irakurketa/idazketa.

Konputazio intentsitatea: $2n/3n = 2/3$.

2. BLAS-2: $\mathcal{O}(n^2)$ matrize-bektore eragiketak.

Adibidea. $y = \alpha * A * x + \beta * y$, non $\alpha, \beta \in \mathbb{R}$, $x, y \in \mathbb{R}^n$ eta $A \in \mathbb{R}^{n \times n}$.

$\mathcal{O}(n^2)$ eragiketa aritmetiko eta $\mathcal{O}(n^2)$ irakurketa/idazketa.

Konputazio intentsitatea: $\approx 2n^2/n^2 = 2$.

3. BLAS-3: $\mathcal{O}(n^3)$ matrize-matrize eragiketak.

Adibidea. $C = \alpha * A * B + \beta * C$, non $\alpha, \beta \in \mathbb{R}$ eta $A, B \in \mathbb{R}^{n \times n}$.

$\mathcal{O}(n^3)$ eragiketa aritmetiko eta $\mathcal{O}(n^2)$ irakurketa/idazketa.

Konputazio intentsitatea: $\approx 2n^3/4n^2 = n/2$.

BLAS-1 eta BLAS-2 funtzioen konputazio intentsitatea txikia da eta beraz, talde hauetako funtzioetan, datuen komunikazioa nagusia da. BLAS-3 funtzioetan aldiz, konputazio intentsitatea handiagoa da eta ezaugarri honi esker, tamaina handiko matrizeen kalkuluetan, konputagailuaren konputazio gaitasuna ondo aprobetxatu ahal izango da

Aljebra linealeko aplikazioen exekuzio denboraren zati garrantzitsuena, behe-mailako eragiketa hauen konputazioak ematen du. Behe-mailako eragiketen hauen optimizazioak, konputagailu bakoitzaren araberakoak dira eta espezializazio handia eskatzen du. Fabrikataile bakoitzak optimizatutako bere BLAS liburutegia du (AMD-ACML,Intel-MKL). Bestalde, optimizatutako BLAS instalazioa, ATLAS (Automatically Tuned Linear Algebra Software) izeneko aplikazioaren bidez ere egin daiteke.

Implementazio guztiak, interfaze berdina erabiltzen dute eta beraz, BLAS-en oinarritutako garapena edozein konputagailuan erabili daiteke (portabilitatea). BLAS liburutegia, Fortran lengoian implemtatuta dago eta C lengoaiatik BLAS funtzioen erabilpena errazteko, *cblas* interfazea erabiltzea gomendagarria da.

Adibidea. BLAS liburutegiaren eraginkortasuna, *cblas_dgemm()* matrizeen biderkadura funtzioren bidez aztertu dugu eta gure implementazio arrunta baino $10 \times$ azkarragoa dela baiezta dugu (Taula A.1.).

A.1. Taula: BLAS liburutegiaren eraginkortasuna. C lengoaiako gure garapena (C-arrunta) eta BLAS liburutegiaren cblas_dgemm() implementazioak konparatu ditugu. n tamainako ezberdineko matrizreak biderkatzu ditugu, eta biderketa bakoitzaz n_{test} alditan errepikatu dugu, kasu guztiak eragiketa aritmetiko kopuru berdina izan ditzaten

n	nests	C-Arrunta		cblas_dgemm	
		Wall T.	CPU T.	Wall T.	CPU T.
10	5.00×10^8	478.	478.	205.	206.
20	6.25×10^7	491.	491.	92.	91.
30	1.85×10^7	474.	474.	78.	78.
40	7.81×10^6	523.	523.	66.	66.
50	4.00×10^6	493.	493.	64.	64.
60	2.31×10^6	479.	479.	58.	58.
70	1.45×10^6	475.	475.	43.	170.
80	9.76×10^5	469.	469.	45.	177.
90	6.85×10^5	491.	491.	47.	186.
100	5.00×10^5	466.	466.	39.	156.
200	6.25×10^4	504.	504.	34.	138.
400	7.81×10^3	657.	657.	35.	140.

LAPACK.

LAPACK, 1992. urtean garatu zen [4, 53] eta aljebra linealaren problemak ebazteko funtzioen liburutegia da. Jatorrizko bertsioa *Fortran* 77 lengoian implementatuta dago eta liburutegiaren dokumentazioa nahiz kodea *Netlib* software bilgunean eskuratu daiteke. Matrize dentsoetarako garatuta dago eta problema hauetarako errutinak biltzen ditu:

1. Ekuazio-sistema linealen ebaezpena: $AX = b$.
2. Linear least square problems: $\|Ax - b\|$ minimizatzen duen x balioa bilatu.
3. Eigenvalues problems.
4. Balio singularren deskonposaketa (SVD).

LAPACK liburutegia, konputagailu sekuentzial eta memoria konpartitutako konputagailuetan erabilgarria izateko diseinatuta dago. Eraginkortasuna *BLAS* funtzio optimizatuen menpe dago eta funtzioen implementazioa, gehien bat *BLAS-3* taldeko funtzioetan oinarritzen da.

LAPACKE liburutegia C-lengoaiatik LAPACK funtzioei deitzeko interfazea da. LAPACK liburutegiaren funtzioak, hiru taldeetan banatzen dira [4, 1]:

1. Problema osoa ebazten dituzten errutinak (drivers). Talde honetan funtziok arruntak eta funtziok espezializatuak daude.

Adibidea.

LAPACKE-dgelsv (LAPACK-ROW-MAJOR, n, nrhs, A, lda, ipiv, B, ldb);
Matrize orokoren (GE), $A * X = B$ sistema lineala ebazten du,

2. Konputazio errutinak. Lan zehatz bat exekutatzen duen errutinak.

Adibidea.

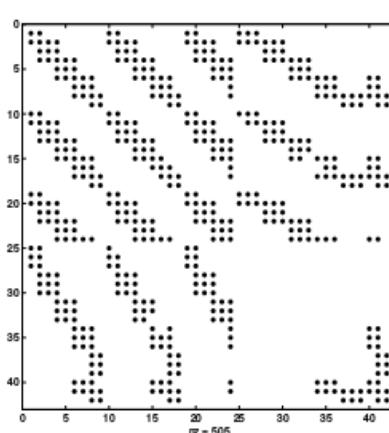
LAPACKE-dgetrf (LAPACK-ROW-MAJOR, n, m, A, lda, ipiv);
Errutina honek A ($n \times m$) tamainako matrizearen LU faktorizazioa kalkulatzen du, $A = P * L * U$.
LAPACKE-dgetrs (LAPACK-ROW-MAJOR,trans,n,nrhs,A,lda,ipiv,B,ldb);
Errutina honek $A * X = B$ ekuazio sistemaren X soluzioa kalkulatzen du.

3. Errutina laguntzaileak.

Matrize bakanak.

$A \in \mathbb{R}^{m \times n}$ matrizeari bakanak esaten zaio, baldin abantaila atera daitekeen zero osagai kopuru adina baditu. Honek esan nahi du, matrizearen zero ez diren osagaien kopurua n_{nz} ,

$$n_{nz} \ll mn.$$



A.7. Irudia: Matrize bakanak.

Matrizearen bakantasuna, konputazioaren memoria eta exekuzio denbora gutxitzeko erabil daiteke.

1. Doitasun bikoitzeko $A \in \mathbb{R}^{m \times n}$ matriza,
 - (a) Dentsoa: $8mn$ byte.
 - (b) Bakana: $\approx 16n_{nz}$ (gordetzeko teknikaren arabera).
2. $y = y + Ax$, $y, x \in \mathbb{R}^n$ eta $A \in \mathbb{R}^{m \times n}$,
 - (a) Dentsoa: $\mathcal{O}(mn)$, eragiketa aritmetikoak.
 - (b) Bakana: $\mathcal{O}(n_{nz})$, eragiketa aritmetikoak.

A.6. Konpiladorea.

Konpiladorearen zereginan konplexua da, goi mailan idatzitako programari dagoen makina kodea sortzea (konputagailuaren errekurtoak modu eraginkorrean erabiltzen dituenak) [99]. Konpiladoreak heuristikotan oinarritutako kode aldaketak eragiten ditu, eraginkortasuna hobetzeko asmoarekin. Horregatik, programatzaileak konpiladorearen optimizazio automatiko hauek kontutan hartu behar ditu eta ahal duen neurrian, bere kodean konpiladorearen optimizazioak erraztu.

Konpiladoreak. Konpiladore ezberdinak daude:

1. *gcc* (GNU open source compiler).

```
$ gcc -v
$ gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.2)
```

2. Konpiladore komertzialak: Intel (*icc*),...

Optimizazioak. Konpiladoreek, optimizazio maila estandarrak eskaintzen dituzte. Orokorean, hurrengo kode optimizazioak izango ditugu:

1. *-O0*. Kode optimizazio nagusienak aplikatzeko aukera da. Kodea *debugger* moduan aztertzen ari garenean gomendatzen da.
2. *-O2*. Kode eraginkorra sortzeko aukera gomendarriena.

Maila altuagoko optimizazioak aplika daitezke, baina optimizazio hauek arriskutsuak izan daitezke. Kodearen exekuzio denboraren analisia egiteko tresnak (*gprof*) daude. Algoritmoaren funtzio bakoitzaren exekuzio denborari buruzko informazio erabilgarria lortuko dugu.

Konpilazio aginduak .

1. Konpilazioa, esteka egin eta adibidea.exe exekutagarria sortzeko

```
$ gcc adibidea.c -o adibidea.exe
```

2. Konpilazio eta esteka urratsak banatuta.

```
$ gcc adibidea.c # creates adibidea.o
$ gcc adibidea.o -o adibidea.exe
```

Gure esperimentuetarako era honetan burutu dugu konpilazioa,

```
gcc -O2 -Wall -std=c99 -fno-common adibidea.c
```

Makefile. Normalean, aplikazioaren kodea fitxategi ezberdinetan egituratzen da eta konpilazio prozesua konplexua izan daiteke. *Makefile* fitxategia, lengoia berezi bat erabiliz, konpilazio prozesua automatizatzeko programa moduko bat dugu [99]. ?? eranskinean, *Makefile* lengoaiaren oinarrizko adibideak eman ditugu.

A.7. Laburpena.

Algoritmo bat implementatzen dugunean kontutan hartu beharrekoa:

1. Lerro edo zutabe araberako iterazioak exekuzio denboran eragin handia du.
2. Kodea garbia eta ulergarria mantendu behar da.
3. LAPACK eta BLAS liburutegiak oso eraginkorrak dira, eta implementazio berriean erabiltzea komenigarria da.

Atal honi dagokion gomendatutako bibliografia: "An introduction to parallel programming", P. Pacheco [87], "Introduction to High Performance Scientific Computing", V. Eijkhout [99], "Performance optimization of numerically intensive codes", Goedecker [42], "Handbook of linear algebra", Leslie Hogben [55].

LAPACK eta BLAS liburutegiak erabiltzeko informazio interesgarria "Intel Math Kernel Library. Reference Manual" [1] dokumentuan aurki daiteke.

B. Eranskina

Ekuazioen garapenak.

B.1. Kepler hasierako baliodun problema.

Keplerren ekuazioa, kokapen eta abiadura berriak kalkulatzeko oinarrizkoa da eta era honetan definitzen da,

$$E - e \sin E = M,$$

non $M = n(t - T)$ (*mean anomaly*), $n = k a^{-3/2}$ (*mean motion*) eta $T, M = 0$ deneko integracio konstantea da. E (*eccentric anomaly*) eta t -ren arteko erlazio hau erabiliz kalkulatzen da mugimendua. Mugimendu eliptikoaren kasura mugatuko gara ($0 \leq e < 1$) eta Keplerren ekuazioa transentalala denez, zenbakizko metodo baten bidez ebatziko dugu.

Garapena.

Gure abiapuntua, honakoa da,

$$\begin{aligned} E_0 - e \sin E_0 &= n(t_0 - t_p), \\ E_1 - e \sin E_1 &= n(t_1 - t_p) \end{aligned}$$

non $n = 2\pi/P$ eta P periodoa diren.

Bi ekuazioen arteko kendura eginez,

$$E_1 - E_0 - e(\sin(E_1) - \sin(E_0)) = n\Delta t \longrightarrow \Delta E - e(\sin(E_0 + \Delta E) - \sin(E_0)) = n\Delta t$$

non $E_1 = E_0 + \Delta E$ den.

Honako notazioa erabiliz adieraziko dugu,

$$\Delta E - ce \sin(\Delta E) - se(\cos(\Delta E) - 1) = n\Delta t,$$

non $ce = e \cos(E_0)$ eta $se = e \sin(E_0)$ den.

Newton metodoa. Ekuazio ebatzeko, Newton metodoa aplikatuko dugu,

1. $f(\Delta E) = \Delta E - ce \sin(\Delta E) - se(\cos(\Delta E) - 1) - n\Delta t = 0.$
2. $f'(\Delta E) = 1 - ce \cos(\Delta E) + se \sin(\Delta E).$
3. $\Delta E^{[k+1]} = \Delta E^{[k]} - \frac{f(\Delta E^{[k]})}{f'(\Delta E^{[k]})}.$

Hasierako balioa. $\Delta E^{[0]}$ hasierako balioa, finkatzea da dugun zaitasun handiena. Horretarako honako garapena egingo dugu,

$$\begin{aligned}\Delta E - ce \sin(\Delta E - se (\cos(\Delta E) - 1)) &= n\Delta t, \\ x = \Delta E - n\Delta t,\end{aligned}$$

eta beraz,

$$x - ce \sin(n\Delta t + x) - se(\cos(n\Delta t + x) - 1) = 0.$$

Honako baliokidetasun trigonometrikoak ordezkatzuz,

$$\begin{aligned}\cos(A + B) &= \cos(A)\cos(B) - \sin(A)\sin(B), \\ \sin(A + B) &= \sin(A)\cos(B) + \cos(A)\sin(B),\end{aligned}$$

berdintza hau lortzen dugu,

$$\begin{aligned}x - (se \cos(n\Delta t) + ce \sin(n\Delta t)) \cos(x) \\ + (se \sin(n\Delta t) - ce \cos(n\Delta t)) \sin(x) + se = 0.\end{aligned}$$

x txikia dela suposatuz, honako hurbilpenak ordezkatuko dugu,

$$x \approx \sin(x), \cos(x) \approx 1 - \frac{x^2}{2}$$

eta honako berdintza lortuko dugu,

$$\begin{aligned}(se \cos(n\Delta t) + ce \sin(n\Delta t)) \frac{x^2}{2} \\ + (1 + se \sin(n\Delta t) - ce \cos(n\Delta t))x - (se) = 0.\end{aligned}$$

Azkenik, goiko ekuazio hau askatuz ($Ax^2 + Bx + C = 0 \rightarrow x = -B \pm \sqrt{B^2 - 4AC}/2A$) lortuko dugu,

$$\Delta E^{[0]} = x + n\Delta t.$$

Koordenatu kartesiarren kalkulua ekuazio hauen bidez egindo dugu,

$$(q_1, v_1) = (q_0, v_0) + (q_0, v_0) \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$\begin{aligned} b_{11} &= (C - 1) \frac{a}{\|q\|}, \\ b_{21} &= \Delta t + (S - \Delta E) \frac{a^{\frac{3}{2}}}{\mu^{\frac{1}{2}}} \\ b_{12} &= \frac{s}{\|q\| \sqrt{a(1 - ce C + se S)}}, \\ b_{22} &= \frac{C - 1}{1 - ce C + se S}. \end{aligned}$$

Non osagai bakoitzaren definizioa,

$$\begin{aligned} C &= \cos(\Delta E), \quad S = \sin(\Delta E), \\ ce &= e \cos(E_0) = \|q\| \|v\|^2 - 1, \\ se &= e \sin(E_0) = \frac{(q \cdot v)}{\sqrt{\mu a}}, \\ a &= \frac{\mu \|q\|}{2\mu - \|q\| \|v\|^2}, \\ n &= \frac{\mu^{\frac{1}{2}}}{a^{\frac{3}{2}}}. \end{aligned}$$

Ezabapen arazoa. ΔE txikia denean, $\cos(\Delta E) - 1$ espresioaren kalkuluaren ezabapen arazoak biribiltze errore handia eragin dezake. Hori konpontzeko balio-kidetasun trigonometriko hau erabiliko dugu,

$$\cos(\Delta E) - 1 = -\frac{\sin^2(\Delta E)}{1 + \cos(\Delta E)}.$$

Eta beraz, Keplerren ekuazioak hauek izango dira,

$$f(\Delta E) = \Delta E - ce \sin(\Delta E) + se \left(\frac{\sin^2(\Delta E)}{1 + \cos(\Delta E)} \right) - n \Delta t = 0$$

Eta (q_1, v_1) balioak kalkulatzeko,

$$b_{11} = (C - 1) \frac{a}{\|q\|}, \rightarrow b_{11} = -\frac{\sin^2(\Delta E)}{1 + \cos(\Delta E)} \frac{a}{\|q\|}$$

B.2. Koordenatu sistemak.

Lehenik koordenatu barizentrikoei $q_i, p_i \in \mathbb{R}^3$, $i = 0, \dots, N$ dagokien Hamilton-darra gogoratuko dugu,

$$H(q, p) = \frac{1}{2} \sum_{i=0}^N \frac{\|p_i\|^2}{m_i} - G \sum_{0 \leq i < j \leq N} \frac{m_i m_j}{\|q_i - q_j\|}. \quad (\text{B.1})$$

Koordenatu Heliozentrikoak.

Koordenatu barizentrikoetatik abiatuta eta aldagai aldaketa bat aplikatuz ekuazio koordenatu heliozentrikoen $Q_i, P_i \in \mathbb{R}^3$, $i = 0, \dots, N$ arabera berridatziko ditugu.

Aldagai aldaketa.

Lehenik honako aldagai aldaketa aplikatuko dugu,

$$\begin{aligned} Q_0 &= q_0, \quad Q_i = q_i - q_0, \\ P_0 &= \sum_{i=0}^N p_i, \quad P_i = p_i, \quad i = 1, \dots, N. \end{aligned}$$

Hamiltondarra era honetan deskonposatu daiteke,

$$H = H_K + (T_1 + U_1)$$

non

$$\begin{aligned} H_K &= \sum_{i=1}^N \left(\frac{\|P_i\|^2}{2\mu_i} - \frac{Gm_0m_i}{\|Q_i\|} \right), \quad \mu_i = \frac{m_0m_i}{(m_0 + m_i)}, \\ T_1 &= \frac{1}{m_0} \left(\sum_{0 < i < j \leq N} P_i P_j \right), \\ U_1 &= - \sum_{0 < i < j \leq N} \frac{Gm_i m_j}{\|Q_i - Q_j\|}. \end{aligned} \quad (\text{B.2})$$

Hamiltondarra abiaduraren arabera idazteko $V_i = P_i/\mu_i$ berdintza ordezkatu-

ko dugu,

$$\begin{aligned} H_K &= \sum_{i=1}^N \left(\frac{\|V_i\|^2 \mu_i}{2} - \frac{Gm_0 m_i}{\|Q_i\|} \right), \quad \mu_i = \frac{m_0 m_i}{(m_0 + m_i)}, \\ T_1 &= m_0 \left(\sum_{0 < i < j \leq N}^N \frac{m_i m_j V_i V_j}{(m_0 + m_i)(m_0 + m_j)} \right), \\ U_1 &= - \sum_{0 < i < j \leq N}^N \frac{Gm_i m_j}{\|Q_i - Q_j\|}. \end{aligned} \quad (\text{B.3})$$

Ekuazio differentzialak.

Hamiltondar bakoitza independenteki kontsideratuta dagokio ekuazio differentzialak lortzen ditugu:

1. H_K .

$$\begin{aligned} \dot{Q}_i &= \nabla_p H_k \Rightarrow \dot{Q}_i = P_i \left(\frac{m_0 + m_i}{m_0 m_i} \right), \\ \dot{P}_i &= -\nabla_q H_k \Rightarrow \dot{P}_i = -\frac{Gm_0 m_i}{\|Q_i\|^3} Q_i, \quad i = 1, \dots, N. \end{aligned}$$

2. T_1 .

$$\begin{aligned} \dot{Q}_i &= \nabla_p T_1 \Rightarrow \dot{Q}_i = \sum_{j \neq i, j=1}^N \frac{P_j}{m_0}, \\ \dot{P}_i &= -\nabla_q T_1 \Rightarrow \dot{P}_i = 0, \quad i = 1, \dots, N. \end{aligned}$$

3. U_1 .

$$\begin{aligned} \dot{Q}_i &= \nabla_p U_1 \Rightarrow \dot{Q}_i = 0, \\ \dot{P}_i &= -\nabla_q U_1 \Rightarrow \dot{P}_i = \sum_{j \neq i, j=1}^N \left(\frac{-Gm_i m_j}{\|Q_i - Q_j\|^3} (Q_i - Q_j) \right), \quad i = 1, \dots, N. \end{aligned}$$

Azkenik, $V_i = P_i / \mu_i$ aplikatuta integrazioan erabiliko ditugun ekuazioak la-burtuko ditugu.

1. H_k .

$$\begin{aligned} \dot{Q}_i &= V_i \\ \dot{V}_i &= -\frac{G(m_0 + m_i)}{\|Q_i\|^3} Q_i, \quad i = 1, \dots, N. \end{aligned}$$

2. T_1 .

$$\begin{aligned}\dot{Q}_i &= \sum_{j \neq i, j=1}^N \frac{V_j m_j}{(m_0 + m_j)}, \\ \dot{V}_i &= 0, \quad i = 1, \dots, N.\end{aligned}$$

3. U_1 .

$$\begin{aligned}\dot{Q}_i &= 0, \\ \dot{V}_i &= -\frac{G(m_0 + m_i)}{m_0} \sum_{j \neq i, j=1}^N \left(\frac{m_j}{\|Q_i - Q_j\|^3} (Q_i - Q_j) \right), \quad i = 1, \dots, N.\end{aligned}$$

Energia.

Koordenatu heliozentrikoetan integrazioak egiten ditugunean, sistemaren energia kalkulatzeko, soluzioa koordenatu sistema barizentrikoetara bihurtuko dugu. Hauek dira koordenatu heliozentrikoetatik abiatuta (Q_i, V_i), koordenatu barizentrikoak (q_i, v_i) kalkulatzeko ekuazioak,

1. $q_i, \quad i = 0, \dots, N$.

$$\begin{aligned}q_0 &= -\sum_{i=1}^M \frac{m_i Q_i}{M}, \quad M = \sum_{i=0}^N m_i, \\ q_i &= q_0 + Q_i, \quad i = 1, \dots, N.\end{aligned}$$

2. $v_i, \quad i = 0, \dots, N$.

$$\begin{aligned}v_i &= \frac{m_0}{m_0 + m_i} V_i, \quad i = 1, \dots, N \\ P_0 &= \sum_{i=0}^N p_i = \sum_{i=0}^N m_i v_i = 0 \Rightarrow m_0 v_0 + \sum_{i=1}^N m_i v_i = 0 \Rightarrow v_0 = -\frac{1}{m_0} \sum_{i=1}^N m_i v_i.\end{aligned}$$

Koordenatu Jacobiarak.

Koordenatu barizentrikoetatik abiatuta eta aldagai aldaketa bat aplikatuz, ekuazioak koordenatu Jacobiarren $Q_i, P_i \in \mathbb{R}^3$, $i = 0, \dots, N$ arabera berridatziko ditugu.

Aldagai aldaketa.

Lehenik honako aldagai aldaketa aplikatuko dugu,

$$Q_0 = (m_0 q_0 + \cdots + m_n q_n) / \eta_N, \quad Q_i = q_i - \left(\sum_{j=0}^{i-1} m_j q_j \right) / \eta_{i-1}$$

$$P_0 = \sum_{i=0}^N p_i, \quad P_i = \left(\eta_{i-1} p_i - m_i \sum_{j=0}^{i-1} p_j \right) / \eta_i, \quad i = 1, \dots, N.$$

non $\eta_i = \sum_{j=0}^i m_j$ den.

Era berean, Jacobi masak $m'_i = (\eta_{i-1} m_i) / \eta_i$ eta $\mu'_i = m_i \eta_{i-1}$ ekuazioetan ordezkatuko ditugu. Hamiltondarra era honetan deskonposatu daiteke,

$$H = H_K + H_I,$$

non

$$H_K = \sum_{i=1}^N \left(\frac{\|P_i\|^2}{2m'_i} - \frac{\mu'_i}{\|Q_i\|} \right),$$

$$H_I = \left(\frac{\mu'_i}{Q_i} - \frac{Gm_0 m_i}{q_i} \right) - \sum_{0 < i < j \leq N} \frac{Gm_i m_j}{\|Q_i - Q_j\|}.$$

B.3. Newton eraginkorraren garapena.

Formulazio estandarrean honako ekuazio sistema askatzeko metodoa proposatzen da,

$$(I_s \otimes I_d - h A \otimes J) \Delta Y = r. \quad (\text{B.4})$$

Lehenengo modu orokorrean eta ondoren, metodoa simetrikoa dela kontutan harturik garapenaren zehaztasunak emango ditugu.

Kasu orokorra

Honako ekuazio sistemari,

$$(I_s \otimes I_d - h \bar{A} \otimes J) \Delta Y - \frac{1}{2} (e_s \otimes I_d) \Delta z = r, \quad (\text{B.5})$$

$$(-h e_s^T B \otimes J) \Delta Y + \Delta z = 0, \quad (\text{B.6})$$

aldagai aldaketa hau, aplikatuko doigu.

$$\Delta Y = (Q \otimes I_d) W. \quad (\text{B.7})$$

1. Lehen urratsa.

Ekuazioa sistemaren lehen ekuazioari (B.5), aldagai aldaketa (B.7) aplikatu eta $(Q^{-1} \otimes I_d)$ gaia ezkerretik biderkatuz,

$$(Q^{-1} \otimes I_d) (I_s \otimes I_d - h \bar{A} \otimes J) (Q \otimes I_d) W - (Q^{-1} \otimes I_d) \left(\frac{1}{2} e_s \otimes I_d \right) \Delta z = (Q^{-1} \otimes I_d) r.$$

Eta garatuz,

$$(I_s \otimes I_d - h Q^{-1} \bar{A} Q \otimes J) W - \frac{1}{2} (Q^{-1} e_s \otimes I_d) \Delta z = (Q^{-1} \otimes I_d) r.$$

2. Bigarren urratsa.

Metodoa simetrikoa bada,

$$Q^{-1} \bar{A} Q = \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \quad (\text{B.8})$$

eta beraz,

$$(I_s \otimes I_d - h \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \otimes J) W - \frac{1}{2} (Q^{-1} e_s \otimes I_d) \Delta z = (Q^{-1} \otimes I_d) r.$$

Eta $I_s \otimes I_d$ bloke moduan idatzia,

$$I_s \otimes I_d = \begin{pmatrix} I_m \otimes I_d & 0 \\ 0 & I_{s-m} \otimes I_d \end{pmatrix} \quad (\text{B.9})$$

$$\begin{pmatrix} I_m \otimes I_d & -hD \otimes J \\ hD^T \otimes J & I_{s-m} \otimes I_d \end{pmatrix} W - \frac{1}{2} (Q^{-1} e_s \otimes I_d) \Delta z = (Q^{-1} \otimes I_d) r.$$

3. Hirugarren urratsa.

Bigarren ekuazioari (B.5) aldagai aldaketa (B.7) aplikatuz,

$$(-h e_s^T B \otimes J)(Q \otimes I_d) W + \Delta z = 0,$$

eta garatuz,

$$-h(e_s^T B Q \otimes J)W + \Delta z = 0.$$

Kasu simetrikoa

Kasu orokorraren emaitzan, kasu simetrikoa ordezkatuz

$$W = \begin{pmatrix} W' \\ W'' \end{pmatrix}, \quad Q = (Q_1 \ Q_2)$$

dagokion ekuazioak lortuko ditugu ekuazioak.

Honako berdintza hauek kontutan hartuz, $Q^{-1} = Q^T B$, $e s^T B Q_2 = 0$, $Q_2^T B e s = 0$ honako ekuazioak lortuko ditugu,

$$\begin{aligned} W' - h(D \otimes J) W'' - \frac{1}{2} (Q_1^T B e_s \otimes I_d) \Delta z &= (Q_1^T B \otimes I_d) r, \\ h(D^T \otimes J) W' + W'' &= (Q_2^T B \otimes I_d) r, \\ -h(e_s^T B Q_1 \otimes J) W' + \Delta z &= 0. \end{aligned}$$

B.4. Kepler fluxuaren aldagai aldaketa.

Alde Kepleriar bakarra.

Satelite baten orbitaren ekuazio diferentzialak, era honetan idatz daiteke,

$$\frac{d}{dt} \begin{pmatrix} q \\ v \end{pmatrix} = k(t, q, v) + g(t, q, v),$$

non,

$$k(t, q, v) = \begin{pmatrix} v \\ -\frac{\mu}{\|q\|^3} q \end{pmatrix}.$$

Aldagai aldaketa. Honako aldagai aldaketa aplikatuko dugu,

$$\begin{pmatrix} q \\ v \end{pmatrix} = \varphi_t(Q, V) \tag{B.10}$$

Aldagai berriekiko ekuazio diferentzialak. Aldagai berriarekiko ekuazio diferentzialak definituko ditugu. (B.10) ekuazioak deribatuko ditugu (katearen erre-gela),

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} q(t) \\ v(t) \end{pmatrix} &= \frac{d}{dt} (\varphi_t(Q(t), V(t))), \\ \frac{d}{dt} \begin{pmatrix} q(t) \\ v(t) \end{pmatrix} &= \dot{\varphi}_t(Q(t), V(t)) + \varphi'_t(Q(t), V(t)) \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix}. \end{aligned}$$

Fluxuaren definizioaz,

$$\dot{\varphi}_t(Q(t), V(t)) = k(q(t), v(t)),$$

eta beraz,

$$g(t, q, v) = g(t, \varphi_t(Q, V)) = \varphi'_t(Q(t), V(t)) \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix}.$$

Azkenik, $Q(t), V(t)$ aldagairekiko deribatu partzialen espresioak lortuko ditugu,

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix} &= \varphi'_t(Q(t), V(t))^{-1} g(t, \varphi_t(Q, V)), \\ G(t, Q, V) &= \varphi'_t(Q(t), V(t))^{-1} g(t, \varphi_t(Q, V)). \end{aligned} \quad (\text{B.11})$$

Perturbazio Hamilondarra. Lehenik, perturbazioa Hamilondarra den kasua kontsideratuko dugu,

$$g(t, q, v) = \begin{pmatrix} \nabla_p r(t, q, v) \\ -\nabla_q r(t, q, v) \end{pmatrix} = J^{-1} \nabla r(t, q, v),$$

non,

$$J = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}, \quad J^{-1} = -J.$$

Bestalde, fluxu Kepleriarra Hamilondarra da eta simplektikoa izateagatik honako propietatea betetzen du,

$$(\varphi'_t(Q, V))^T J \varphi'_t(Q, V) = J. \quad (\text{B.12})$$

Eta goiko (B.12) garatuz, honakoa ere betetzen dela badakigu,

$$J^{-1}(\varphi'_t(Q, V))^T = (\varphi'_t(Q, V))^{-1} J^{-1}.$$

Definizio hau emanik,

$$R(t, Q, V) = r(t, \varphi_t(Q, V)),$$

berdintza hau betetzen dela erakutsi nahi dugu,

$$J^{-1}(\varphi'_t(Q, V))^T \nabla r = J^{-1} \nabla R(t, Q, V).$$

$\nabla R(t, Q, V)$ katearen erregela aplikatuz zuzenean baieztagatzen da.

Perturbazio Ez-Hamiltondarra. Kepler-en fluxua simplektikoa da eta propietate horretan oinarritzen gara fluxuaren alderantzikoaren kalkulua saihesteko.

$$\begin{cases} \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix} = (\varphi'_t(Q, V))^{-1} g(t, \varphi_t(Q, V)) \\ J^{-1}(\varphi'_t(Q, V))^T J = (\varphi'_t(Q, v))^{-1} \end{cases} \rightarrow \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix} = J^{-1}(\varphi'_t(Q, V))^T J g. \quad (\text{B.13})$$

Deribazio automatiko prozedura aplikatuz (B.13) kalkulatuko dugu.

Algoritmoa. (Q, V) aldagai berrien kalkulua hiru urratsetan egingo dugu,

1. KeplerFlowGen.

$$(q, v) = \varphi_t(Q, V).$$

(aux, gero erabiliko ditugun tarteko aldagaiak ere itzuliko ditu).

2. GFcn.

$$g = GFcn(t, q, v)$$

3. KeplerFlowGFcnaux.

$$J^{-1}(\varphi'_t(Q, V))^T J g$$

Alde Kepleriar bat baino gehiago.

Alde Kepleriar bat baino gehiago den kasurako azterketa egingo dugu. k , alde Kepleriar kopurua esago diogu.

$$\frac{d}{dt} \begin{pmatrix} q \\ v \end{pmatrix} = \begin{pmatrix} q_1 \\ v_1 \\ q_2 \\ v_2 \\ \vdots \\ q_k \\ v_k \\ w \end{pmatrix} = \begin{pmatrix} v_1 \\ -\mu_1 q_1 / \|q_1\|^3 \\ v_2 \\ -\mu_2 q_2 / \|q_2\|^3 \\ \vdots \\ v_k \\ -\mu_k q_k / \|q_k\|^3 \\ 0 \end{pmatrix} + g(t, q_1, v_1, \dots, q_k, v_k, w).$$

non

$$g(t, q_1, v_1, \dots, q_k, v_k, w) = \begin{pmatrix} g_1(t, q_1, v_1, \dots, q_k, v_k, w) \\ g_2(t, q_1, v_1, \dots, q_k, v_k, w) \\ \vdots \\ g_k(t, q_1, v_1, \dots, q_k, v_k, w) \\ g_{k+1}(t, q_1, v_1, \dots, q_k, v_k, w) \end{pmatrix}$$

Gorputz bakoitzari dagokion aldagai aldaketa lokala da,

$$\begin{pmatrix} q_j \\ v_j \end{pmatrix} = \varphi_t^{\mu_j}(Q_j, V_j), \quad j = 1, \dots, k, \quad (B.14)$$

$w = W.$

Fluxuaren Jacobiarra diagonala da,

$$\begin{pmatrix} \varphi_t^{\mu'_1}(Q_1, V_1) & & & \\ & \ddots & & \\ & & \varphi_t^{\mu'_k}(Q_k, V_k) & \\ & & & I \end{pmatrix}.$$

(B.13) aplikatzen badugu,

$$\frac{d}{dt} \begin{pmatrix} q \\ v \end{pmatrix} = \begin{pmatrix} q_1 \\ v_1 \\ \vdots \\ q_k \\ v_k \\ w \end{pmatrix} = \begin{pmatrix} \varphi_t^{\mu'_1}(Q_1, V_1) g_1 \\ \varphi_t^{\mu'_2}(Q_2, V_2) g_2 \\ \vdots \\ \varphi_t^{\mu'_k}(Q_k, V_k) g_k \\ g_{k+1} \end{pmatrix}$$

C. Eranskina

Inplementazioak.

Lan honetan, ekarpen bakoitzari dagokion C lengoaiako implementazio bat garatu dugu eta [kode bilgunean](#) eskuragarri jarri dugu. Kodea eskuragarri jartzeari, lehentasuna eman diogu [8, 100]. Batetik, garapenak modu argian dokumentatzea behartu gaitu; bertan zehaztasun guztiak aztertu daitezke eta gure lanaren baliagarritasuna baiezztatu daiteke. Bestetik, ikerlariei gure lana erabiltzeko eta hobetzeko aukera eskaintzen diegu.

Konputazio zientzian, idatzitako kodearen 200 lerro oro, errore bat izaten dela [38] estimatzen da. Gure implementazioak ondo frogatu baditugu ere, errorerik ez denik izango ezin dugu ziurtatu. Akatsen bat izatekotan, larria ez izatea espero dugu eta edozein kasutan erabiltzaileak jakinaraztea eskertu genuke.

Implementazioen kdea antolatzeko, irizpide berdinak erabili ditugu. Lehenengo, implementazioen egitura orokorra azaldu dugu eta ondoren, garapen bakotzaren argibideak emango ditugu.

C.1. Egitura orokorra.

Implementazioen edukia, direktorio hauetan banatu dugu:

1. CoefficientsData.

Mathematican, IRK Gauss kolokazio metodoaren koefizienteak sortzeko aplikazio bat garatu dugu. Gure esperimentuetarako $s = 6, s = 8$ eta $s = 16$ ataletako Gaussen metodoak aplikatu ditugu eta metodo bakoitzari dagokien doitasun bikoitze (64-bit) koefizienteak, azpidirektorio batean bildu ditugu.

2. PerturbationsData.

Errorearen azterketa estatistikoak egiteko, hasierako balio ezagun baten $P = 1000$ hasierako balio perturbatuak sortu ditugu; hasierako balio originalaren

osagai bakoitza ausaz perturbatu dugu ($\mathcal{O}(10^{-6})$ tamainako errore erlatiboarekin). Problema bakoitzari dagokion, $P = 1000$ hasierako balio perturbatuak, fitxategi bitar batean idatzi dugu eta horrela, fitxategi hauetatik hasierako balioak irakurrita, esperimentu bera errepikatu daiteke.

3. Packages.

Esperimentuak, Mathematica ingurunetik exekutatu ditugu eta direktorio honetan, soluzioen azterketak egiteko hainbat funtzi garatu ditugu. Batez, problema bakoitzari dagokion ekuazio diferentzialak eta Hamiltondarra implementatu ditugu. Bestetik, esperimentuen grafikoak irudikatzeko funtzioak garatu ditugu.

4. Examples.

Mathematica erabiliz egindako zenbakizko integrazioen adibideak eman ditugu. Adibide bakoitza, azpidirektorio batean bildu dugu: zenbakizko integrazioa eta honen analisia, modu independentean exekutatu daitezke. Horretarako, integrazioaren soluzioak fitxategi bitarretan idazten dira eta analisiak, fitxategi hauek irakurriko ditu.

5. Code.

C lengoian garatutako gure implementazioa.

Exekuzioa.

Code/Readme.txt fitxategian, implementazioaren argibideak eman ditugu. Implementazioa bi modutan exekutatu daiteke.

1. Linux terminala.

Exekuzioa Linux terminaletik exekutatu daiteke eta exekutatzeko adibidea bat "terminal.c" fitxategian eman dugu.

2. Mathematica ingurunetik.

Bestalde, implementazioa Mathematicatik exekutatzeko prest dago eta "Examples" direktorioan hainbat adibide eman ditugu. Mathematicatik gure C implementazioaren funtzi nagusia deitu ahal izateko, 'math-IRK.tm' eta 'math-IRK.c' fitxategiak definitu ditugu. Exekutarria, Mathematicatik bateragarria izan dadin, konpilazio eta esteka egiteko moduak *makefile* fitxategian kontsultatu daitezke.

Erabiltzaileak, bere problemaren ekuazio diferentzialak eta integrazioen emaitzak definituko dituela espero da.

Paralelizazioa.

IRK metodoen s-ataletako funtzioen konputazioak ($f(Y_i)$, $i = 1, \dots, s$), paraleloan exekutatu daitezke. Gure implementazioan, OpenMP erabili dugu: *PARALLEL* konpilazio parametroarekin aktibatu daiteke eta orduan, prozesadore kopuruoa 'threadcount' aldagaien, zehaztu behar da.

C.2. IRK puntu-finkoa.

Puntu-finkoaren iterazioan oinarritutako IRK metodoaren implementazioa, [kodea](#) helbidean eskuragarri dago. Bi modutara exekutatu daiteke: puntu-finkoaren iterazio estandarra edo puntu-finkoaren iterazio partizionatua.

Code/Readme.txt fitxategian, IRK puntu-finkoaren implementazioaren argibide guztiak eman ditugu. Integrazioaren funtzi nagusiaren deia honakoa da:

```
IRKFPI ( t0 , t1 , h,& method ,& u,& system ,& options ,& thestat );
```

Exekuzioa Linux terminaletik exekutatu daiteke eta exekutatzeko adibidea bat "terminal.c" fitxategian eman dugu. Bestalde, implementazioa Mathematicatik exekutatzeko prest dago eta "Examples" direktorioan hainbat adibide eman ditugu.

Implementazio honetan, ez dugu *BLAS* liburutegia erabili.

C.3. IRK Newton.

Newton sinplifikatuaren iterazioan oinarritutako IRK metodoaren implementazioa, [kodea](#) helbidean eskuragarri dago. Bi implementazio exekutatu daitezke: Newton sinplifikatuaren implementazio eraginkorra eta artikuluan, proposatutako Newton iterazioan oinarritutako IRK implementazio berria.

Code/Readme.txt fitxategian, IRK puntu-finkoaren implementazioaren argibide guztiak eman ditugu. Integrazioaren funtzi nagusiaren deia honakoa da:

```
IRKNEWTON ( t0 , t1 , h,& method ,& u,& system ,& options ,& thestat );
```

Exekuzioa Linux terminaletik exekutatu daiteke eta exekutatzeko adibidea bat "terminal.c" fitxategian eman dugu. Bestalde, implementazioa Mathematicatik exekutatzeko prest dago eta "Examples" direktorioan hainbat adibide eman ditugu.

Implementazio honetan, *BLAS* eta *LAPACK* liburutegiak erabili ditugu.

C.4. IRK Eguzki-sistema.

C.5. Konposizio-Splitting metodoak.

Newton simplifikatuaren iterazioan oinarritutako IRK metodoaren implementazioa, [kodea](#) helbidean eskuragarri dago. Bi implementazio exekutatu daitezke: konposizio metodoak (CO1035) eta splitting metodoak (ABAH1064).

Code/Readme.txt fitxategian, konposizio/Splitting implementazioaren argibide guztiak eman ditugu. Integrazioaren funtziogun nagusiaren deia honakoa da:

```
Solve_Comp ( t0 , t1 , h,& method , basic ,& system ,& options ,& u,& thestat );
```

Exekuzioa Linux terminalatik exekutatu daiteke eta exekutatzeko adibidea bat "terminal.c" fitxategian eman dugu. Bestalde, implementazioa Mathematicatik exekutatzeko prest dago eta "Examples" direktorioan hainbat adibide eman ditugu.

Eguzki-sistemari egokitutako Splitting metodoak aplikatzeko, Kepler fluxuan implementazio berri bat garatu dugu. Implementazio honen C kodean, 'Kepler.c' fitxategian aurkitzen da.

D. Eranskina

Nire-Argibideak

Niretzako dokumentatutako argibide batzuk. Ez ditut tesian sartu behar.

D.1. IRK-Newton.

Newton simplifikatuaren iterazioa.

Newton simplifikatuaren iterazioan formulazio berrian ekuazio honen jatorria:

$$\begin{aligned} \text{Askatu } \triangle L_i^{[k]} \\ \triangle L_i^{[k]} - hb_i J_i \sum_{j=1}^s \mu_{ij} \triangle L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s. \end{aligned}$$

Frogapena.

Abiapuntua.

$$\begin{aligned} \triangle L_i &= L_i - L_i^{[k]}, \\ L_i - hb_i f(y_n + \sum_{j=1}^s \mu_{ij} L_j) &= 0 \end{aligned}$$

Honako garapena egingo dugu.

$$L_i - hb_i f(y_n + \sum_{j=1}^s \mu_{ij} L_j) = L_i^{[k]} + \triangle L_i - hb_i f(y_n + \sum_{j=1}^s \mu_{ij} (L_j^{[k]} + \triangle L_j)).$$

Eta $f(y_n + \sum_{j=1}^s \mu_{ij}(L_j^{[k]} + \Delta L_j))$ linealizatuz,

$$\approx L_i^{[k]} + \Delta L_i - hb_i f(y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k]}) - hb_i f'(y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k]})(\sum_{j=1}^s \mu_{ij} \Delta L_j) =$$

$$\Delta L_i - hb_i J_i (\sum_{j=1}^s \mu_{ij} \Delta L_j) = -L_i^{[k]} + hb_i f(y_n + \sum_{j=1}^s \mu_{ij} L_j)$$

non $J_i = f'(y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k]}) = f'(Y_i)$.

D.2. FMA.

Nola jakin gure *linux* konputagailu batek *FMA* instrukzioak dituen ala ez ? Hona-ko agindua exekutatu eta *fma* flaga azaltzen den ala ez begiratu behar dugu.

```
$ grep fma < /proc/cpuinfo
```

eta konpilatzeko *-mfma* flag-a zehaztu behar da,

```
$ gcc -O2 -Wall -std=c99 -fno-common -mfma adibidea.c
```

D.3. Hitz-zerrenda.

D.1. Taula: Hitz-zerrenda.

Euskaraz	Ingelesez	Laburdura	Adibidea
Ekuazio diferentzial arrunta	Ordinary Differential equation	<i>ODE</i>	$\dot{y} = f(t, y)$
Hasierako baliodun problema	Initial value problem	<i>IVP</i>	
Zurruna	Stiff		
Simplektikoa	Symplectic		
Desplazamendu	Drift		
	Splitting methods		
	Composition methods		
Runge-Kutta Esplizitua	Explicit Runge-Kutta	<i>ERK</i>	
Runge-Kutta Implizitua	Implicit Runge-Kutta	<i>IRK</i>	
	A-stability, B-stability		
	Implicit Midpoint method		
	Adjoint		
	bias		
	compensated summation		
Biribiltzea	roundoff		
Portabilitate	portable		
Doitasun arrunta	Single precision		
Doitasun bikoitza	Double precision		
Doitasun laukoitza	Quadruple precision		
Multiple-digit representation	Digito-anitzeko adierazpena		
Multiple-term representation	Termino-anitzeko adierazpena		
Haria	Thread		
	Graphical Processor Unit	<i>GPU</i>	
	Least Square		
	Least Eigenvalues problems		
Balio singulararen deskonposaketa	Singular values descomposition	<i>SVD</i>	
Ezentrizidadea	Eccentricity		
	Eccentric anomaly		
Astronomical unit (AU)	Unitate astronomikoa		
Julian date	Data juliotar		
LU Decomposition (low, up)	LU-deskonposaketa		
Flops	Koma-higikorreko eragiketa segunduko		
Peak	Exekuzio gaitasuna		
Wall-time, elapsed-time			
CPU-time			
Cache memoria			
spatial/data locality			
Single Instruction Multiple Data		<i>SIMD</i>	
Fork-join			
Application programming interface	Interfaze aplikazio programa	<i>API</i>	
Eraginkortasun altuko konputazioa	High performance computing	<i>HPC</i>	
Problem solving enviroments	Problemak ebazteko inguruneak	<i>PSE</i>	
Portable			
Linear least square problems			
Eigenvalues problems			
Sparse matrices	Matrize bakanak		

Bibliografia

- [1] *Intel Math Kernel Library. Refrence Manual*. Intel, 2015.
- [2] *he openmp api specification for parallel programming*, 2017.
- [3] Sverre Aarseth, Christopher Tout, and Rosemary Mardling. *The Cambridge n-body lectures*, volume 760. Springer, 2008.
- [4] Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, et al. *LAPACK Users' guide*. SIAM, 1999.
- [5] KV Vladimirov Andrey. *Test-driving intel xeon phi coprocessors with a basic n-body simulation*. *Coflax International*, 3:2, 2013.
- [6] Mikel Antonana, Joseba Makazaga, and Ander Murua. *Efficient implementation of symplectic implicit runge-kutta schemes with simplified newton iterations*. *arXiv preprint arXiv:1703.07697*, 2017.
- [7] Mikel Antonana, Joseba Makazaga, and Ander Murua. *Reducing and monitoring round-off error propagation for symplectic implicit runge-kutta schemes*. *Numerical Algorithms*, pages 1–20, 2017.
- [8] Harald Atmanspacher and Sabine Maasen. *Reproducibility: Principles, Problems, Practices, and Prospects*. John Wiley & Sons, 2016.
- [9] Marc Baboulin, Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Julie Langou, Julien Langou, Piotr Luszczek, and Stanimire Tomov. *Accelerating scientific computations with mixed precision algorithms*. *Computer Physics Communications*, 180(12):2526 – 2533, 2009. 40 {YEARS} {OF} CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures.
- [10] Pavan Balaji. *Programming models for parallel computing*. MIT Press, 2015.

- [11] Josh Barnes and Piet Hut. [A hierarchical \$O\(n \log n\)\$ force-calculation algorithm](#). *nature*, 324(6096):446–449, 1986.
- [12] André Berger. [A brief history of the astronomical theories of paleoclimates](#). Springer, 2012.
- [13] Gregory Beylkin and Kristian Sandberg. [Ode solvers using band-limited approximations](#). *Journal of Computational Physics*, 265:156–171, 2014.
- [14] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. [Julia: A fresh approach to numerical computing](#). *arXiv preprint arXiv:1411.1607*, 2014.
- [15] Theodore A Bickart. [An efficient solution process for implicit runge–kutta methods](#). *SIAM Journal on Numerical Analysis*, 14(6):1022–1027, 1977.
- [16] Sergio Blanes, Fernando Casas, Ariadna Farres, Jacques Laskar, Joseba Makazaga, and Ander Murua. [New families of symplectic splitting methods for numerical integration in dynamical astronomy](#). *Applied Numerical Mathematics*, 68:58–72, 2013.
- [17] Ben K Bradley, Brandon A Jones, Gregory Beylkin, Kristian Sandberg, and Penina Axelrad. [Bandlimited implicit runge–kutta integration for astrodynamics](#). *Celestial Mechanics and Dynamical Astronomy*, 119(2):143–168, 2014.
- [18] Dirk Brouwer. [On the accumulation of errors in numerical integration](#). *The Astronomical Journal*, 46:149–153, 1937.
- [19] Luigi Brugnano, Gianluca Frasca Caccia, and Felice Iavernaro. [Efficient implementation of gauss collocation and hamiltonian boundary value methods](#). *Numerical Algorithms*, 65(3):633–650, 2014.
- [20] VA Brumberg. [Celestial mechanics: Past, present, future](#). *Solar System Research*, 47(5):347–358, 2013.
- [21] John C Butcher. [On the implementation of implicit runge-kutta methods](#). *BIT Numerical Mathematics*, 16(3):237–240, 1976.
- [22] John Charles Butcher. [Numerical Methods for Ordinary Differential Equations](#). Second edition, Wiley, 2008.
- [23] J Carrier, Leslie Greengard, and Vladimir Rokhlin. [A fast adaptive multipole algorithm for particle simulations](#). *SIAM journal on scientific and statistical computing*, 9(4):669–686, 1988.

- [24] John E Chambers. [A hybrid symplectic integrator that permits close encounters between massive bodies](#). *Monthly Notices of the Royal Astronomical Society*, 304(4):793–799, 1999.
- [25] Philippe Chartier, Ander Murua, and Jesus Maria Sanz-Serna. [Higher-order averaging, formal series and numerical integration i: B-series](#). *Foundations of Computational Mathematics*, 10(6):695–727, 2010.
- [26] Robert M Corless and Nicolas Fillion. [A graduate introduction to numerical methods](#). *AMC*, 10:12, 2013.
- [27] John Danby. Fundamentals of celestial mechanics. *Richmond: Willman-Bell, 1 c1992, 2nd ed.*, 1, 1992.
- [28] Theodorus Jozef Dekker. [A floating-point technique for extending the available precision](#). *Numerische Mathematik*, 18(3):224–242, 1971.
- [29] DumitruN. Deleanu. [Fast detection of chaotic or regular behavior of double pendulum system: application of the fast norm vector indicator method](#).
- [30] Martin J Duncan, Harold F Levison, and Man Hoi Lee. [A multiple time step symplectic algorithm for integrating close encounters](#). *The Astronomical Journal*, 116(4):2067, 1998.
- [31] Siegfried Eggl and Rudolph Dvorak. [An introduction to common numerical integration codes used in dynamical astronomy](#). In *Dynamics of small solar system bodies and exoplanets*, Springer, 2010, pages 431–480.
- [32] Ariadna Farrés, Jacques Laskar, Sergio Blanes, Fernando Casas, Joseba Makazaga, and Ander Murua. [High precision symplectic integrators for the solar system](#). *Celestial Mechanics and Dynamical Astronomy*, 116(2):141–174, 2013.
- [33] Kang Feng and Mengzhao Qin. [Symplectic geometric algorithms for hamiltonian systems](#). Springer, 2010.
- [34] A. Fienga. [International workshop in astronomy and dynamics](#). In *INPOP planetary ephemerides: from 2013 to 2015.*, Observatoire de Paris, 2015.
- [35] A Fienga, H Manche, J Laskar, and Mickael Gastineau. [Inpop06: a new numerical planetary ephemeris](#). *Astronomy & Astrophysics*, 477(1):315–327, 2008.

- [36] William M Folkner, James G Williams, Dale H Boggs, Ryan S Park, and Petr Kuchynka. [The planetary and lunar ephemerides de430 and de431](#). *Interplanet. Netw. Prog. Rep.*, 196:1–81, 2014.
- [37] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. [Mpfr: A multiple-precision binary floating-point library with correct rounding](#). *ACM Transactions on Mathematical Software (TOMS)*, 33(2):13, 2007.
- [38] Daan Frenkel and Berend Smit. [Understanding molecular simulation: from algorithms to applications](#), volume 1. Academic press, 2001.
- [39] Toshio Fukushima. [Reduction of round-off error in symplectic integrators](#). *The Astronomical Journal*, 121(3):1768, 2001.
- [40] Toshio Fukushima. [Numerical comparison of two-body regularizations](#). *The Astronomical Journal*, 133(6):2815, 2007.
- [41] GNU. [libquadmath v 1.3](#), 2008.
- [42] Stefan Goedecker and Adolfy Hoisie. [Performance optimization of numerically intensive codes](#). SIAM, 2001.
- [43] KR Grazier, WINewman, James M Hyman, Philip W Sharp, and David J Goldstein. [Achieving brouwer’s law with high-order stormer multistep methods](#). *ANZIAM Journal*, 46:786–804, 2005.
- [44] E. Hairer and C. Lubich. [Numerical solution of ordinary differential equations](#). In *The Princeton Companion to Applied Mathematics*, Nicholas J. Higham, Mark R. Dennis, Paul Glendinning, Paul A. Martin, Fadil Santosa, and Jared Tanner, editors, Princeton University Press, Princeton, NJ, USA, 2015, pages 293–305.
- [45] E Hairer, SPNørsett, and G Wanner. [Solving ordinary differential equations I: nonstiff problems, vol. 8](#). 1993.
- [46] E Hairer and G Wanner. [Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems](#). Springer, Berlin, 1996.
- [47] E. Hairer and G. Wanner. [Initial value problems](#). In *Encyclopedia of Applied and Computational Mathematics*, B Engquist, editor, Springer, 2015, pages 691–694.

- [48] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.
- [49] Ernst Hairer, Robert I McLachlan, and Alain Razakarivony. Achieving brouwer's law with implicit runge–kutta methods. *BIT Numerical Mathematics*, 48(2):231–243, 2008.
- [50] Wayne B Hayes. Is the outer solar system chaotic? *Nature Physics*, 3(10):689–691, 2007.
- [51] David M Hernandez and Edmund Bertschinger. Symplectic integration for the collisional gravitational n-body problem. *Monthly Notices of the Royal Astronomical Society*, 452(2):1934–1944, 2015.
- [52] Yozo Hida, Xiaoye S Li, and David H Bailey. Algorithms for quad-double precision floating point arithmetic. In *Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium on*, IEEE, 2001, pages 155–162.
- [53] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. Siam, 2002.
- [54] Nicholas J Higham. Programming languages: An applied mathematics view. 2015.
- [55] Leslie Hogben. *Handbook of linear algebra*. Chapman and Hall/CRC, 2013.
- [56] IEEE. *Ieee standard for floating-point arithmetic*. 2008.
- [57] Tomoaki Ishiyama, Keigo Nitadori, and Junichiro Makino. 4.45 pflops astrophysical n-body simulation on k computer: the gravitational trillion-body problem. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, IEEE Computer Society Press, 2012, page 5.
- [58] Takashi Ito and Kiyotaka Tanikawa. Trends in 20th century celestial mechanics. *Publ. Natl. Astron. Obs. Jpn*, 9:55–112, 2007.
- [59] Laurent O. Jay. Preconditioning of implicit runge-kutta methods. *Scalable Computing: Practice and Experience*, 10, 2009.
- [60] MP Calvo JM Sanz-Serna. *Numerical Hamiltonian problems*. Chapman and Hall, 1994.

- [61] Mioara Joldeş, Olivier Marty, Jean-Michel Muller, and Valentina Popescu. [Arithmetic algorithms for extended precision using floating-point expansions](#). *IEEE Transactions on Computers*, 65(4):1197–1210, 2016.
- [62] W Kahan. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.
- [63] George H Kaplan, John A Bangert, Agnes Fienga, William Folkner, Catherine Hohenkerk, Marina Lukashova, Elena V Pitjeva, P Kenneth Seidelmann, Michael Sveshnikov, Sean Urban, et al. [Historical reflections on the work of iau commission 4 \(ephemerides\)](#). *arXiv preprint arXiv:1511.01546*, 2015.
- [64] KV Kholshevnikov and ED Kuznetsov. [Review of the works on the orbital evolution of solar system major planets](#). *Solar System Research*, 41(4):265–300, 2007.
- [65] Sergei A Klioner. [Basic celestial mechanics](#). *arXiv preprint arXiv:1609.00915*, 2016.
- [66] D Knuth. [The art of computer programming: Vol 2/seminumerical algorithms](#), 1969.
- [67] Sergei Kopeikin, Michael Efroimsky, and George Kaplan. [Relativistic celestial mechanics of the solar system](#). John Wiley & Sons, 2011.
- [68] Fred T Krogh. [An adams guy does the runge-kutta](#). 1997.
- [69] Anne Kvaerno and Ben Leimkuhler. [A time-reversible, regularized, switching integrator for the n-body problem](#). *SIAM Journal on Scientific Computing*, 22(3):1016–1035, 2000.
- [70] MP Laburta. [Construction of starting algorithms for the rk-gauss methods](#). *Journal of computational and applied mathematics*, 90(2):239–261, 1998.
- [71] Jacques Laskar. [The limits of earth orbital calculations for geological timescale use](#). *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 357(1757):1735–1759, 1999.
- [72] Jacques Laskar. [Numerical challenges in long term integrations of the solar system](#). In *Computer Arithmetic (ARITH), 2015 IEEE 22nd Symposium on*, IEEE, 2015, pages 104–104.

- [73] Jacques Laskar, Agnes Fienga, Mickael Gastineau, and Herve Manche. [La2010: a new orbital solution for the long-term motion of the earth.](#) *Astronomy & Astrophysics*, 532:A89, 2011.
- [74] Jacques Laskar and Mickael Gastineau. [Existence of collisional trajectories of mercury, mars and venus with the earth.](#) *Nature*, 459(7248):817–819, 2009.
- [75] Jacques Laskar and Philippe Robutel. [High order symplectic integrators for perturbed hamiltonian systems.](#) *Celestial Mechanics and Dynamical Astronomy*, 80(1):39–62, 2001.
- [76] Benedict Leimkuhler and Sebastian Reich. [Simulating hamiltonian dynamics](#), volume 14. Cambridge University Press, 2004.
- [77] Harold F Levison and Martin J Duncan. [The long-term dynamical behavior of short-period comets.](#) *Icarus*, 108(1):18–36, 1994.
- [78] Werner Liniger and Ralph A Willoughby. [Efficient integration methods for stiff systems of ordinary differential equations.](#) *SIAM Journal on Numerical Analysis*, 7(1):47–66, 1970.
- [79] Piotr Luszczek, Jakub Kurzak, and Jack Dongarra. [Looking back at dense linear algebra software.](#) *Journal of Parallel and Distributed Computing*, 74(7):2548–2560, 2014.
- [80] Robert I McLachlan. [Composition methods in the presence of small parameters.](#) *BIT Numerical Mathematics*, 35(2):258–268, 1995.
- [81] Robert I McLachlan and Pau Atela. [The accuracy of symplectic integrators.](#) *Nonlinearity*, 5(2):541, 1992.
- [82] A Morbidelli. [Modern integrations of solar system dynamics.](#) *Annual Review of Earth and Planetary Sciences*, 30(1):89–112, 2002.
- [83] Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, and Serge Torres. [Handbook of floating-point arithmetic.](#) Springer Science & Business Media, 2009.
- [84] Nobelprize.org. [The nobel prize in chemistry 2013](#), May 2014.
- [85] Hans Olsson and Gustaf Sderlind. [The approximate runge-kutta computational process.](#) *BIT Numerical Mathematics*, 40(2):351–373, 2000.

- [86] Michael L Overton. *Numerical computing with IEEE floating point arithmetic*. Siam, 2001.
- [87] Peter Pacheco. *An introduction to parallel programming*. Elsevier, 2011.
- [88] EV Pitjeva and NP Pitjev. Development of planetary ephemerides epm and their applications. *Celestial Mechanics and Dynamical Astronomy*, 119(3-4):237–256, 2014.
- [89] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
- [90] Prasenjit Saha and Scott Tremaine. Symplectic integrators for solar system dynamics. *The Astronomical Journal*, 104:1633–1640, 1992.
- [91] Prasenjit Saha and Scott Tremaine. Long-term planetary integration with individual time steps. *arXiv preprint astro-ph/9403057*, 1994.
- [92] Jesús María Sanz-Serna. Symplectic integrators for hamiltonian problems: an overview. *Acta numerica*, 1:243–286, 1992.
- [93] JM. Sanz-Serna. Hamiltonian systems. In *Encyclopedia of Applied and Computational Mathematics*, B Engquist, editor, Springer, 2015, pages 617–624.
- [94] JM. Sanz-Serna. Symplectic methods. In *Encyclopedia of Applied and Computational Mathematics*, B Engquist, editor, Springer, 2015, pages 1451–1458.
- [95] PW Sharp. Long initial value test problems from simulations of the solar system. Technical report, Department of Mathematics, The University of Auckland, New Zealand, 2001.
- [96] Mark Sofroniou and Giulia Spaletta. Derivation of symmetric composition constants for symmetric integrators. *Optimization Methods and Software*, 20(4-5):597–613, 2005.
- [97] Pat H Sterbenz. *Floating-point computation*. Prentice Hall, 1973.
- [98] Gerald J Sussman and Jack Wisdom. Chaotic evolution of the solar system. Technical report, DTIC Document, 1992.
- [99] Robert van de Geijn Victor Eijkhout and Edmond Chow. *Introduction to High Performance Scientific Computing*. lulu.com, 2011.

- [100] Greg Wilson, DA Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, et al. [Best practices for scientific computing](#). *PLoS Biol*, 12(1):e1001745, 2014.
- [101] Jack Wisdom. [Symplectic correctors for canonical heliocentric n-body maps](#). *The Astronomical Journal*, 131(4):2294, 2006.
- [102] Jack Wisdom and David M Hernandez. [A fast and accurate universal kepler solver without stumpff series](#). *Monthly Notices of the Royal Astronomical Society*, 453(3):3015–3023, 2015.
- [103] Jack Wisdom and Matthew Holman. [Symplectic maps for the n-body problem](#). *The Astronomical Journal*, 102:1528–1538, 1991.
- [104] Wolfram Research, Inc. [Mathematica](#).
- [105] Dexuan Xie. A new numerical algorithm for efficiently implementing implicit runge-kutta methods. *Department of Mathematical Sciences. University of Wisconsin, Milwaukee, Wisconsin, USA*, 2009.
- [106] Haruo Yoshida. [Recent progress in the theory and application of symplectic integrators](#). In *Qualitative and Quantitative Behaviour of Planetary Systems*, Springer, 1993, pages 27–43.