

N-GORPUTZEKO PROBLEMA GRABITAZIONALAREN EBAZPENERAKO ZENBAKIZKO METODOEN AZTERKETA.

Mikel Antonana Otano

Informatika Fakultatea
Euskal-Herriko Unibertsitatea
Donostia

2017



Universidad
del País Vasco Euskal Herriko
Unibertsitatea

Gaien Aurkibidea

I. Introduction	3
1. Sarrera.	5
2. Zenbakizko integratzaile simplektikoak.	13
3. Problemak.	43
4. Koma higikorreko aritmetika.	59
5. Zientzia konputazioa.	69
II. Ekarpenak.	87
6. IRK: Puntu-finkoaren iterazioa.	89
7. IRK: Newtonen Iterazioa.	113
8. IRK: Eguzki-sistema.	141
III. Ondorioak.	159
9. Eztabaida.	161
10. Konklusioak.	163
IV. Eranskinak	165
A. Ekuazio garapenak.	167
B. Kodea	175

C. Notazioa.	181
---------------------	------------

D. Nire-Argibideak	185
---------------------------	------------

Gaien Aurkibidea (detailed)

I. Introduction	3
1. Sarrera.	5
1.1. Ikerketaren testuingurua	5
1.2. Motibazioa	6
1.3. Helburua eta esparrua	7
1.4. Ekarpenak	8
1.5. Tesiaren egitura	11
1.6. Laburpena	11
2. Zenbakizko integratzaile simplektikoak.	13
2.1. Sarrera	13
2.2. Zenbakizko integrazio metodoak	13
2.2.1. Oinarrizko kontzeptuak	13
2.2.2. Sistema-Hamiltondarra	19
2.2.3. Metodo simplektikoak	20
2.3. Runge-Kutta metodo simplektikoak	24
2.3.1. Runge-Kutta metodoak	24
2.3.2. Gauss metodoa	26
2.3.3. IRK implementazioa	29
2.4. Konposizio eta Splitting metodoak	34
2.4.1. Konposizio metodoak	34
2.4.2. Splitting metodoak	36
2.4.3. Eguzki-sistemari egokitutako splitting metodoak	38
2.5. Laburpena	40
3. Problemak.	43
3.1. Sarrera	43
3.2. Pendulu bikoitza	44
3.2.1. Pendulu bikoitz arrunta	44
3.2.2. Pendulu bikoitz zurruna	45
3.3. N-Gorputzen problema	47

3.4.	Eguzki-sistema	49
3.4.1.	Sarrera	49
3.4.2.	Problemak	51
3.5.	Laburpena	58
4.	Koma higikorreko aritmetika.	59
4.1.	Sarrera	59
4.2.	<i>IEEE-754</i> estandarra	59
4.3.	Biribiltze errorea	62
4.4.	Biribiltze errorearen gutxitzeko teknikak	65
4.5.	Laburpena	68
5.	Zientzia konputazioa.	69
5.1.	Sarrera	69
5.2.	Eraginkortasuna	70
5.3.	Hardwarea	72
5.4.	Programazio lengoaiak	77
5.5.	Algebra lineal dentsorako liburutegiak	80
5.6.	Konpiladorea	85
5.7.	Laburpena	86
II.	Ekarpenak.	87
6.	IRK: Puntu-finkoaren iterazioa.	89
6.1.	Sarrera	89
6.2.	Hairer-en implementazioa	90
6.3.	Gure implementazioa	92
6.3.1.	Metodoaren birformulazioa	93
6.3.2.	Geratze irizpidea	95
6.3.3.	Biribiltze errorea gutxitzeko teknikak	97
6.3.4.	Biribiltze errorearen estimazioa	99
6.3.5.	Algoritmoa	101
6.4.	Zenbakizko esperimentuak	103
6.4.1.	Problemak	103
6.4.2.	Energia errorearen jatorria	103
6.4.3.	Errore azterketa estatistikoa	105
6.4.4.	Biribiltze errorearen estimazioa	111
6.5.	Laburpena	111

7. IRK: Newtonen Iterazioa.	113
7.1. Sarrera.	113
7.2. IRK-Newton estandarra.	114
7.3. IRK-Newton eraginkorra.	118
7.3.1. Ekuazio-sistema.	118
7.3.2. IRK metodo simplektikoen garapena.	121
7.3.3. IRK metodo simetriko simplektikoen garapena.	121
7.4. IRK-Newton estandarra (formulazio berria).	125
7.5. IRK-Newton eraginkorra (formulazio berria).	129
7.6. IRK-Newton Mixtoa.	133
7.6.1. Sasi-Newton iterazioa.	133
7.6.2. IRK-Newton Mixtoa	133
7.7. Zenbakizko esperimentuak.	137
7.7.1. Problemak	137
7.7.2. Biribiltze errorearen azterketa.	137
7.7.3. Puntu-finkoa versus Newton iterazioa	139
7.8. Laburpena.	140
8. IRK: Eguzki-sistema.	141
8.1. Sarrera.	141
8.2. Eguzki-sistemaren integraziorako metodoak (review).	141
8.2.1. Efemerideak.	141
8.2.2. Eguzki-sistemaren integrazio luzeak.	144
8.3. Gure implementazioa.	147
8.4. Denbora birparametrizazioa.	154
8.4.1. Adibidea.	155
8.5. Atalen hasieraketa.	156
8.6. Laburpena.	156
III. Ondorioak.	159
9. Eztabaida.	161
9.1. Sarrera.	161
9.2. IRK Puntu finkoa.	161
9.3. IRK Newton.	161
9.4. Laburpena.	162
10. Konklusioak.	163
10.1. Sarrera.	163
10.2. Laburpena.	163

IV. Eranskinak	165
A. Ekuazio garapenak.	167
A.1. Kepler hasierako baliodun problema.	167
A.2. Koordenatu sistemak.	170
A.3. Newton eraginkorraren garapena.	173
B. Kodea	175
B.1. Gauss metodoa	175
B.2. Zientzia konputazioa.	175
B.3. Problemak	178
B.4. Fortran kodeak	179
B.5. IRK-Puntu finkoa	180
B.6. IRK-Newton	180
B.6.1. IRK-Newton koefizienteak.	180
B.6.2. IRK-Newton eraginkorra.	180
B.7. Eguzki-sistema	180
C. Notazioa.	181
C.1. Notazioa.	181
C.2. Hitz-zerrenda.	181
D. Nire-Argibideak	185
D.1. IRK-Newton.	185

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

Vaclav Havel

I. Atala

Introduction

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 funtseko galdera batzuk ezagutu ditugu eta berriki, Karplus-en taldeak 2013. urteko kimika Nobel saria [?] 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 sailkatu ditzakegu:

1. Epe motzeko eta doitasun handiko integrazioak. Eguzki-sistemaren efemide zehatzak [?] edo espazioko satelite artifizialen kokapenen [?] 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 argazkia orokorra (zehaztasun handirik gabe) ezagutzea da. Normalean, problema mota hauetan gorputzen arteko kolisio 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. N-gorputz kopurua miliotakoa [?] izan dateke eta kolisio gertuko egoeren ondorioz, kalkuluetan egindako zenbakizko errore txikiiek soluzioan eragin handia izan ditzakete.

Gure helburua, eguzki-sistemaren epe luzeko eta doitasun handiko integrazioetarako implementazio eraginkorra garatzea da. Aurreko hamarkadetan, eguzki-sistemaren planeten epe luzeko zenbakizko integrazioa erronka garrantzitsua izan da. Adibidez, Sussman-ek eta Wisdom-ek [?, 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 aldaketaren eraginez gertatu zirela azaltzen duen teoria (Milankovitch 1941) [?] 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 konputacionalki oso garestiak dira eta exekuzio denbora luzeak behar dituzte; adibidez, Laskar-ek [?, 2010] bere azken integrazioa burutzeko 18 hilabete behar izan zituen. Azken urteotako konputagailu berrien arkitekturaren bilakaerak, algoritmo azkarren diseinua aldatu du: simulazioak azkartzeako algoritmoak paralelizazioan oinarritu behar dira. Azpimarratzeko da ere, integrazio luze hauen erronka handienetako bat, biribiltze errorearen garapena zaintza da, non errorearen edozein joera ekiditen den [?].

1.2. Motibazioa.

Metodo simplektikoen artean erabilienak, izaera esplizituko algoritmoak dira. Oro har problema zurruna ez bada, metodo esplizituak metodo implizituak baino eraginkorragoak dira. Metodo implizituetan ekuazio sistema ez-lineala askatu behar da (eragiketa garestia) eta honek, metodo esplizituekiko CPU denbora gainkarra suposatzen du. Horregatik problema zurruna ez denean, metodo esplizituak erabili ohi dira eta problema zurruna denean bakarrik jotzen dugu metodo implizituengana. Epe luzeko eguzki-sistemaren integrazioetarako, baieztapen hau eztabaidagarria dela, eta praktikan metodo implizituetan gehiago sakondu behar dela iruditzen zaigu.

Lan honetan, Gauss zenbakizko integrazio metodo simplektikoaren azterketa egingo dugu. Hainbat autorek (Hairer [?][?] eta Sanz Serna[?]) metodo honen

potentziala nabarmendu dute. Azken urtetan, espazioko satelite artifizialen arloan ere, Gauss integrazio metodo implizituarekiko interesa azaldu dute [?][?].

Jarraian, Gauss metodo implizituen ezaugarri interesgarri batzuk nabarmenduko ditugu. Abantaila nagusienetako malgutasuna da. Metodo implizituek implementazio malgua onartzen dute eta ondorioz, integratu nahi dugun problemari egokitzeko eta eraginkortasuna hobetzeko aukera ezberdinak eskaintzen dizkigu. Aipatzeko da ere, metodo esplizituk (simplektikoak) sistema Hamiltondar banagarriean bakarrik aplika daitezkeela eta orduan, Hamiltondarraren egitura hau aprobetxatuz oso eraginkorrik direla. Baino metodo implizituk aldiz, Hamiltondar orokorreki aplika daitezke eta gainera, lehen ordenako ekuazio differentzialetarako metodo simplektikoak, implizituk izan behar dira. Gainera, Gauss metodoak paralelizagarriak dira, hau da, ekuazio differentzial konplexuak kalkulatu behar ditugunean, *s*-ataletako funtziokonputazioak paraleloan exekutatu daitezke. Azkenik ez dugu ahaztu behar, ordena altuko Gauss metodoak existitzen direla eta hauek beharrezkoak ditugula doitasun handiko integrazioetarako.

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

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

1.3. Helburua eta esparrua.

Gure helburua, eguzki-sistemaren epe luzeko integraziorako Gauss metodo implizituaren implementazio eraginkorra proposatzea da. Helburu hau lortzeko, honako aspektu hauek bereziki zainduko ditugu: eguzki-sistemaren problemaren ezaugarriak, biribiltze errorearen garapena eta egungo konputagailuen gaitasunari egokitutako algoritmo azkarren diseinua.

N-gorputzko problema grabitacionalari dagokionez, eguzki-sistemaren eredu simplea integratuko dugu. Eguzki-sistemaren gorputzak masa puntuak kontsideratuko ditugu eta gure ekuazio differentialek, gorputz hauen arteko erakarpen grabitacionalak bakarrik kontutan hartuko dituzte. Beraz, eguzki-sistemaren eredu konplexuagoetako erlatibilitatea ez dago kontutan hartu. Bestalde era honetako integrazioetan, gorputzen hasierako balio eta parametro zehatzak sateliteen bidez jasotako datu errealekin bat datozena egiaztatze prozesua ez dugu landu [?].

Zeintzuk dira eguzki-sistemaren problemaren ezaugarri bereziak? Batetik, bi gorputzen problemaren soluzio zehatza ezaguna eta eguzki-sistemaren gorputzen mugimenduaren konputazioaren oinarria da. Bestetik, badugu gorputz nagusi bat

(eguzkia) eta honen inguruan mugimenduan dauden gorputzak: barne-planetak, masa txikikoak eta eguzkitik gertu daudenak ; kanpo-planetak, masa handikoak eta eguzkitik urrun daudenak. Kanpo-planeta nagusi hauen eboluzioa, eguzki-sistema osoaren zati garrantzitsuena da eta barne-planeten mugimendua kontutan hartzeak ala ez, kanpo-planeten zenbakizko integrazioarengan oso eragin txikia du. Eguzki-sistemaren egitura honi abantaila gehien ateratzen dion planteamendua bilatuko dugu.

Konputagailuen koma-higikorreko aritmetika ondo ulertzea garrantzitsua da. Zenbaki errealen adierazpen finitura erabiltzen denez, bai zenbakiak memorian gordetzerakoan, bai hauen arteko kalkulu aritmetikoak egiterakoan, biribiltze errorea sortzen da. Integratio luzeetan, biribiltze errorea propagatzen da eta une batetik aurrera, soluzioen zuzentasuna ezereztatzen da. Zentzu honetan, doitasuna hobetzeko biribiltze errorea gutxitzen duten teknika bereziak aplikatzea ezinbestekoa izaten da. Integratio 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, hain zuzen ere, doitasun ezberdinak nahasten dituzten implementazioak.

Konputagailu 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. Algoritmo baten kode unitateak paraleloan exekutatzeak badu gainkarga bat eta beraz, algoritmoaren exekuzioa paralelizazioaz azkartzea lortzeko, unitate bakoitzaren tamainak esanguratsua izan behar du. Gure eguzki-sistemaren eredua simplea da eta logikoa da pentsatzea eredu konplexuagoetan, paralelizazioak abantaila handiagoa erakutsiko duela. Bestalde, N -gorputzen kopurua handia den problemetan, hauen arteko interakzio kopuru $\mathcal{O}(N^2)$ handia kalkulatu behar da eta indar hauen hurbilpena modu eraginkorrean kalkulatzeko metodo ezagunak daude: *tree code*[?] eta *fast multipole method*[?] izeneko metodoak. Baina 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 puntu finkoaren iterazioaren implementazioa aztertu dugu eta gure implementazioen oinarriak finkatu ditugu. Bigarren fasean, Gauss metodoaren Newton iterazioaren

implementazio eraginkorra lortzeko ahalegin berezia egin dugu. Problema zurruna denean, puntu finkoaren iterazio ez da eraginkorra eta Newton iterazioa aplikatu behar da. Gainera problema ez-zurruna izanik ere, Newton iterazioak interesgarriak izan daitezke; bereziki doitasun altuko (doitasun laukoitza) konputazioetan iterazio metodoaren konbergentzia ezaugarri onak direla eta. Hirugarren fasean ...

Jarraian, atal bakoitzean egindako ekarprena nagusienak laburtuko ditugu:

1. IRK puntu finkoa.

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

(a) Metodoaren birformulazioa.

Gauss implizitua aplikatzen dugunean, metodoa definitzen duten birlbildutako koefiziente errealkak ($\tilde{a}_{ij}, \tilde{b}_i \in \mathbb{F}$) erabiltzen dira. Formulazio 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, non simplektizidade baldintza zehazki beteko duten koefizienteak modu errazean finkatu daitezkeen.

(b) Geratze irizpide berria.

Orokorrean, Hairer-en implementazioaren puntu finkoaren iterazioaren geratze irizpide zuzena dela ikusi dugu baina kasu batzuetan goizegi geratzen dela baiezta dugu. Arrazoia bi direla ikusirik, bere geratze irizpidea bi zentzutan garatu/zorroztu dugu. Lehenik, Hairer-en implementazioan, hobekuntza neurtzeko ataletako differentziaren norma batean oinarritzen da. Normaren independentea den geratze irizpidea aplikatzea zuzenagoa da, eta horregatik, ataletako edozein osagaiaren differentzia txikitzen den bitartean iterazioan jarraitzea finkatu dugu. Bigarrenik, iterazioen 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 ordez, $\tilde{y}_n + e_n$ erabiltea proposatu dugu. Aldaketa honekin, zenbakizko soluzioaren doitasuna zerbait 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 diferentzial 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 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, Gaussien 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. Ekarpen 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 egingo ditugu. Implementazio hauen eraginkortasuna, egungo integratzaile simplektiko esplizituekin konparatuko ditugu.

4. Birparametrizazioa.

Azken urratsean, esperimentalki, eguzki-sistemaren integrazioan denbora 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) kapitulueta, 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 kapituluau, lan honetan zenbakizko esperimentuetan eraibili ditugun hasierako baliodun problemen zehaztasunak eman ditugu. Laugarren kapituluau koma-higikorreko aritmetika murgildu gara eta biribiltze errorearen inguruko gaiak argitu nahi izan ditugu. Bostgarren kapituluau, egungo konputazio zientziaren hardware eta software kontzeptu nagusienak ezagutarazi nahi izan ditugu.

Tesiaren (6 – 8) kapitulueta, gure implementazio berriak garatu ditugu eta zenbakizko esperimentuen bidez, hauen eraginkortasuna erakutsi dugu. Lehenik, 6. kapituluau Gauss metodoaren puntu finkoaren iterazioaren implementazio berria eman dugu. Ondoren, 7. kapituluau Gauss metodoaren Newton iterazioaren implementazio eraginkorrik azaldu ditugu. 8. kapituluau, eguzki-sistemaren integraziorako implementazio deskribapena egin dugu.

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

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

1.6. Laburpena

2. Kapitulua

Zenbakizko integratzaile simplektikoak.

2.1. Sarrera.

Atal honen lehen zatian, zenbakizko integrazio metodoen inguruko oinarrizko definizioak eta kontzeptuak azalduko ditugu. Jarraian, sistema Hamiltondarren eta metodo simplektikoen sarrera emango dugu. Ondoren, zenbakizko integratziale nagusienak aztertuko ditugu: alde batetik, Runge-Kutta metodo simplektikoak (implizituak) ; beste alde batetik, konposizio eta splitting metodo simplektikoak (esplizituak). Azalpenak ulergarriak izateko, zenbakizko metodo sinpletan (Euler metodoak) eta adibideetan oinarritu gara.

2.2. Zenbakizko integrazio metodoak

2.2.1. Oinarrizko kontzeptuak.

Ekuazio diferentzial arruntak, egoera aldaketak dituzten problemen azterketarako eredu matematikoak dira. Zientziaren eta ingeniaritzaren arlo askotan azaltzen zaizkigu: planeten mugimenduen ereduetan (astronomia), erreakzio kimikoen formulazioetan, molekulen dinamiken simulazioetan, zirkuitu elektronikoen diseinuan, populazio-hazkunde eta interakzioetan (biologian), ekonomi azterketetan,

...

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

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

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 diferenzial arrunten soluzioa analitikoa (funtzio ezagunen araberako soluzio zehatza) eta beraz, ia problemak gehienak zenbakizko integrazio metodoen bidez ebatzen dira. Zenbakizko integrazioetan, 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.

Zenbakizko soluzioaren (y_n) kalitatea, bi errore mota nagusien araberakoa izango da: zenbakizko integrazio metodoak sortutakoa (trunkatze errorea) eta konputazioan aritmetika finituak eragindakoa (biribiltze errorea).

Ekuazio diferenziala beti *sistema autonomo* moduan, hau da, denborarekiko independentea, idatz daiteke. Hori horrela izanik, notazioa simplifikatzeko 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.

1. Fluxua.

Fase-espazioko edozein y_0 balioari, $y(t_0) = y_0$ hasierako balio duen $y(t)$ soluzioa esleitzten dion funtziario fluxua deitzen zaio. Izendatzeko φ_t notazioa erabiliko dugu,

$$\varphi_t(y_0) = y(t) \text{ baldin } y(t_0) = y_0.$$

2. Zenbakizko diskretizazioa.

y_0, \dots, y_{n-1}, y_n balioak emanda, $y_{n+1} \approx y(t_{n+1})$ soluzioaren hurbilpena kalkulatzeko formulari zenbakizko fluxua deritzogu. Honako notazioa erabiliko dugu,

$$y_{n+1} = \phi(y_{n+1}, y_n, \dots, y_0; h; f).$$

ϕ zenbakizko metodoa, y_{n+1} balioaren menpe ez dagoenean, y_{n+1} zuzenean kalkula daiteke eta metodoari *esplizitua* dela esaten zaio. Aldiz, ϕ metodoa y_{n+1} menpe dagoenean, y_{n+1} askatzeko zeharkako bidea erabili behar da (adibidez Newton sinplifikatua edo puntu finkoaren metoda) eta metodoari *implizitua* dela esaten zaio.

3. Metodoaren ordena.

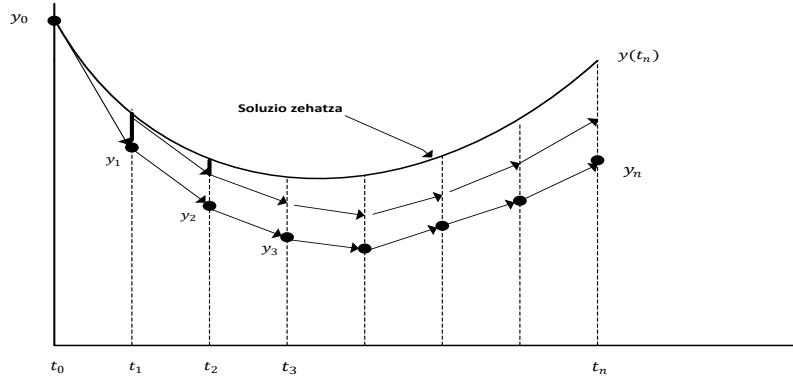
- (a) Errore lokala (*le*), $y(t_k) = y_k$ soluzio zehatzetik abiatuta, integrazio metodoaren urrats bat ($h_k = t_{k+1} - t_k$) eman ondorengo erroreari deritzogu,

$$le(t_{k+1}) = y_{k+1} - y(t_{k+1}), \quad y(t_k) = y_k.$$

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

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

Zenbakizko integrazioaren urrats bakoitzaren abiapuntua, $y(t_k)$ soluzio zehatzaren ordez y_k balioa hartzen denez, garrantzitsua da ulertzea errore lokalaren propagazioa. Horretarako, hurrengo irudian (Irudia 2.1.) oinarrituko gara. Irudiaren lerro bakoitzak, hasierako balio ezberdin baten zenbakizko soluzioa da. Marra bertikal lodi bakoitza errore lokala da. urratsez-urrats propagatzen dena. Errore globala, t_n unean propagatutako errore lokalaren batura da.



2.1. Irudia: Errore lokalaren propagazioa. Irudian soluzio zehatza eta hasierako bi balio ezberdinei dagokien zenbakizko soluzioak irudikatu ditugu.

(c) Metodoaren ordena. h urrats luzera finkoko ϕ metodoak p ordenakoa dela esaten da, $ge(t)$ errore globala $\mathcal{O}(h^p)$ ordenekoa 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 baidu,

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

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,

$$\|y_n - y(t_n)\| \leq Ch,$$

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

Laburki, ekuazio sistema ez-linealen soluzioa askatzeko bi iterazio metodo nagusiak 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,

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

Eta Euler metodo implizituaren ekuazio ez-lineala askatzeko, $k = 1, 2, \dots$ iterazioetarako,

- (a) $r_{n+1}^{[k]} = y_{n+1}^{[k]} - y_n - h_n f(t_{n+1}, y_{n+1}^{[k]}),$
 - (b) Askatu $\Delta y_{n+1}^{[k]}$,
- $$(\Delta y_{n+1}^{[k]} - h_n J_n \Delta y_{n+1}^{[k]}) = -r_{n+1}^{[k]},$$

$$(c) \quad y_{n+1}^{[k+1]} = y_{n+1}^{[k]} + \Delta y_{n+1}^{[k]},$$

non $J_n \approx \frac{\partial f}{\partial y}(t_n, y_n)$ hurbilpen bat den.

Urratsaren konputazioa konplexutasuna, metodo esplizitua baino nabarmen handiagoa da: metodo honetan, iterazio bakoitzean Jacobiarraren ebaluazioa, ekua-zio sistema lineala askatu eta f funtzioaren ebaluazioa kalkulatu behar dira.

Iterazio metodoaren konbergentzia abiadura. $\{x^{[0]}, x^{[1]}, \dots, x^{[k]}\}$, x^* soluzio-rantz 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 ($p = 1$) eta Newtonen iterazioak koadratikoa ($p = 2$) du. Konbergentzia koadratikoa izatea oso interesgarria da, iterazio bakoitzean soluzioaren digitu hamartar zuzenen kopurua bikoizten ditugulako. Konbergentzia linealean, ordea, iterazio bakoitzean digitu hamartar kopuru finkoa hobetzen dugu.

Problema zurruna.

Konputazio konplexutasun handiagoa izan arren, problema batzuetan metodo in-plizituak, esplizituak aplikatzea baino egokiago izan daiteke, eta hori erakusteko Germund Dahlquist-en (1963) adibidea azalduko dugu,

$$\dot{y} = \lambda y, \tag{2.4}$$

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

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

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 differentzialei problema *zurrunkak* (*stiff*) esaten zaie. Problema *zurrunen* artean problema ga-rrantzitsuak ditugu, esaterako eskala anitzeko sistemak.

2.2.2. Sistema-Hamiltondarak.

Hamiltondar sistemak, ekuazio diferentzial mota garrantzitsu bat dira. $H(p, q)$ funtzioko leunari dagokion *Hamiltondar sistema* osatzen duten $2d$ ekuazio diferentzialak, era honetan definitzen dira,

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

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. $H(p, q)$ funtzioko *Hamiltondarra* deritzo, eta sistemaren energia adierazten du. Energia, integrazioan zehar konstante mantentzen da,

$$H(p(t), q(t)) = \text{const.}$$

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,

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

Horien artean, *bigarren ordenako* ekuazio diferentzialak aipatu behar ditugu, zeintzuk Hamiltondar banagarrien kasu partikularra bat diren,

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

eta dagokien ekuazio diferentzialak hauek diren,

$$\dot{p} = -\frac{\partial U(q)}{\partial q}, \quad \dot{q} = p. \tag{2.5}$$

Adibidea. *Bi gorputzen edo Kepler problema.* Planoan elkar erakartzen diren bi gorputzen (adibidez eguzkia eta planeta bat) mugimendua kalkulatzeko, horietako gorputz baten kokapena koordenatu sistemaren jatorria kontsideratuko dugu eta beste gorputzaren kokapenaren koordenatuak $q = (q_1, q_2)$ izendatuko ditugu.

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,

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

Planetaren mugimendua orbita eliptiko bat da. Honako hasierako balioei da-gokien 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, Hamiltondarra modu honetan banatu daiteke $H = H_k + H_I$, non alde nagusia H_K planeta bakoitzaren eguzkiaren inguruko mugimendu Kepleriarrari eta H_I aldiz, planeten arteko interakzioek eragiten duten perturbazio txikiari dagokion.

2.2.3. Metodo simplektikoak.

Hamiltondar sistemen problemetarako, Euler esplizitua eta Euler implizitua ez dira zenbakizko metodo egokiak. Problema hauen propietate geometrikoak mantentzen dituzte integratzaile bereziak beharrezkoak dira [?]. 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 [?] eta ondorengo, azalpen ulergarriak jasotzen dituzten [?] eta [?] lanak aipatu daitezke.

Adibidea. Metodo simplektikoen abantaila azaltzeko penduluaren problemaren adibidea azalduko dugu. Penduluaren problemaren (masa $m = 1$, $l = 1$ luzerako makila eta $g = 1$ grabitazioa) $d = 1$ askatasuneko sistema Hamiltondarra,

$$H(p, q) = \frac{p^2}{2} - \cos(q), \quad (2.6)$$

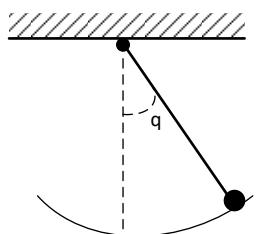
eta ekuazio differentzialak,

$$\dot{p} = -\sin(q), \quad \dot{q} = p. \quad (2.7)$$

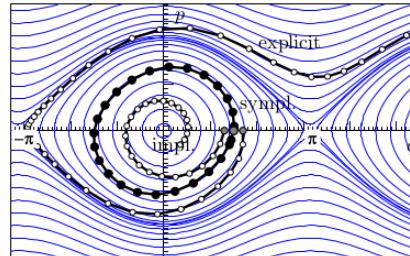
[2.2.](#) irudian ikusi daitekeenez, Euler esplizitu eta implizituaren konportamendu kualitatiboki okerra da. Era honetan definitzen den Euler simplektikoa ordea,

$$\begin{aligned} p_{n+1} &= p_n - h \frac{\partial H}{\partial q}(p_{n+1}, q_n), \\ q_{n+1} &= q_n + h \frac{\partial H}{\partial p}(p_{n+1}, q_n), \end{aligned} \quad (2.8)$$

sistemaren energia kontserbatzen du integrazio luzean zehar.



(a) Pendula.



(b) Integrazioa.

2.2. Irudia: Pendulu problemaren hiru zenbakizko metodoen zenbakizko soluzioak irudikatu ditugu. Hiruretan urrats luzera berdina $h = 0.3$ baina bakoitzas 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 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 differentzial orokorreai aplika daitezke. Bigarren taldean, konposizioan eta splitting-ean oinarritutako metodoak ditugu. Azken talde honekako metodoak, bakarrik Hamiltondar sistemiei aplika daitezke eta gainera, Hamiltondarrak banagarria izan behar du. Konposizio eta splitting metodoak oso eraginkorrak dira.

Metodo simplektikoen propietateak.

Funtsezkoa da, integratzaile simplektiko baten bidez lortutako H Hamiltondar sistemaren zenbakizko soluzioa, perturbatutako beste Hamiltondar \tilde{H} baten soluzio zehatza gisa ulertzea. Kontutan hartu behar da, zenbakizko integrazioak eragiten dituen perturbazio hauek, sistema dinamiko baten H Hamiltondar eredua eraikitzerakoan, hartzen diren hurbilpenek edo datu ez ziurrek eragiten duten baino errore maila txikiagoa suposatzen dutela.

Adibidea. Demagun, askatasun maila bakarra duen sistema Hamiltondar banagarria $H = T(p) + V(q)$, Euler metodo simplektikoaren (2.8) bidez integratzen dugula. (p_n, q_n) zenbakizko soluzioak, integratu nahi dugun S Hamiltondar sistemaren $(p(t_n), q(t_n))$ soluzio zehatza hurbiltzen du. Modu honetan, benetako fluxua φ_h^H eta zenbakizko integrazioa ϕ_h^H izendatzen baditugu,

$$\begin{aligned}(p_{n+1}, q_{n+1}) &= \phi_h^H(p_n, q_n), \\ (p(t_{n+1}), q(t_{n+1})) &= \varphi_h^H(p(t_n), q(t_n)),\end{aligned}$$

eta Euler simplektikoa $p = 1$ ordenako metodoa denez,

$$\phi_h^H(p_n, q_n) - \varphi_h^H(p(t_n), q(t_n)) = \mathcal{O}(h^2).$$

Defini daiteke beste S_2 Hamiltondar sistema bat, zeinekin Euler simplektikoa bigarren ordenako den? Hau da, (p_n, q_n) zenbakizko soluzioa, S soluziotik baino gertuago dagoen S_2 sistema?

$$\phi_h^{\tilde{H}}(p_n, q_n) - \varphi_h^{\tilde{H}}(p(t_n), q(t_n)) = \mathcal{O}(h^3).$$

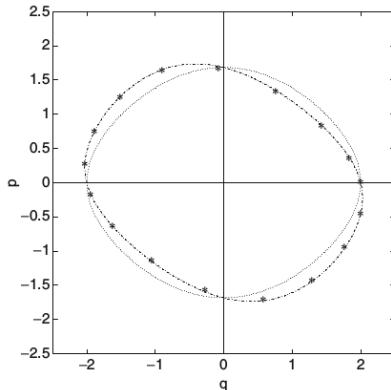
Taylor seriea garatuz, lortuko dugu S_2 sistema ($f = -\partial H/\partial q$, $g = \partial H/\partial p$),

$$\begin{aligned}\frac{d}{dt}p &= f(q) + \frac{h}{2}g(p)f'(q), \\ \frac{d}{dt}q &= g(p) - \frac{h}{2}g'(p)f(q),\end{aligned}$$

non honako Hamiltondarrari dagokion,

$$\tilde{H}_2 = T(p) + V(q) + \frac{h}{2}T'(p)V'(q) = H + \mathcal{O}(h).$$

2.3. irudian, pendulu problemaren (2.7) hasierako balio hauei ($p(0) = 0$, $q(0) = 2$) dagokien integrazioa erakutsi dugu. Izarrak, $h = 0.5$ urratsari dagokion zenbakizko soluzioa da. $H = \text{const}$ puntuko lerroak, penduluaren benetako soluzioa



2.3. Irudia: Penduluaren problema: zenbakizko soluzioa (izarrak), benetako soluzioa (puntu lerroak) and perturbatutako soluzioa (marra lerroak).

irudikatzen du. $\tilde{H}_2 = \text{const}$ marra lerroak, perturbatutako Hamiltondar sistemaren soluzioa irudikatzen du. Zenbakizko soluzioaren eta perturbatutako Hamiltondar sistemaren ibilbideak bat datorzela ikusi daiteke.

Kasu orokorra, posible da S_p Hamiltondar sistema perturbatua eraikitzea, zeinakin Euler simplektikoa p ordenakoa den. Beraz, konklusio gisa esango dugu, Hamiltondar sistema bati aplikatzen diogun metodo simplektikoaren zenbakizko soluzioa, perturbatutako Hamiltondar sistema baten soluzio ia zehatz kontside-ratu behar dugula.

Sarrera hau bukatzeko, metodo simplektikoen propietate nagusienak azpimarratuko ditugu.

1. Metodo simplektikoek, energia oso ondo kontserbatzen dute.

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

$$H(p(t), q(t)) = \text{Const.}$$

Zenbakizko soluzioak ordea, ez du $H(p_n, q_n)$ konstante mantentzen. Integrazio metodoa simplektikoa bada, aritmetika zehatzarekin energia errorea bornatua mantentzen da eta drift-arik gabe. Koma-higikorreko aritmetika-rekin, biribiltze errorearen eraginez energia errorea denboraren erro karra-tuaren 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. Errore propagazio 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 konportamendu txarra azaltzen dute.

3. Metodo simplektiko garrantzitsuenak simetrikoak dira.
4. Urrats luzera finkoa erabili behar da izaera simplektikoa ez galtzeko [?]. Metodo gehienak eraginkorrik izateko, errore estimazio baten arabera integrazioan zehar urrats luzera egokitzen dute. Soluzioaren aldaketa azkarra gertatzen diren uneetarako urratsa txikia, eta aldiz, urratsa handia soluzioa leuna den uneetarako.

2.3. Runge-Kutta metodo simplektikoak.

1988. urtean, Sanz-Sernak *Gauss metodoa* simplektikoa zela ohartu zen, s-ataleko ordena altueneko metodoa ($p = 2s$) izanik. Runge-Kutta metodo implizitua da eta hainbat propietate interesgarri ditu.

2.3.1. Runge-Kutta metodoak.

Runge-Kutta metodoak, 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,

$$\begin{array}{c|ccccc} & c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c & c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \hline & \vdots & \vdots & \vdots & \ddots & \vdots \\ b^T & c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} \quad (2.9)$$

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

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i f(t_n + c_i h, Y_{n,i}) , \quad (2.10)$$

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

$$Y_{n,i} = y_n + h \sum_{j=1}^s a_{ij} f(t_n + c_j h, Y_{n,j}) \quad i = 1, \dots, s. \quad (2.11)$$

$Y_{n,i}$ atalak, $(t_n + c_i h)$, $i = 1, \dots, s$ uneetako soluzioaren hurbilpenak dira,

$$Y_{n,i} \approx y(t_n + c_i h), \quad i = 1, \dots, s.$$

Metodo esplizituak (ERK) eta implizituak (IRK).

Runge-Kutta 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,

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

$Y_{n,i}$ 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,

1/2	1/2	
	1	

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

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

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 baitira. 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 sinplifikazio baldintza* hauen [?] arabera definitzen dira,

$$B(p) : \sum_{i=1}^s b_i c_i^{q-1} = \frac{1}{q}, \quad q = 1, \dots, p. \quad (2.12)$$

$$C(\eta) : \sum_{j=1}^s a_{ij} c_j^{q-1} = \frac{c_i^q}{q}, \quad i = 1, \dots, s, \quad q = 1, \dots, \eta. \quad (2.13)$$

$$D(\zeta) : \sum_{i=1}^s b_i c_i^{q-1} a_{ij} = \frac{b_j}{q} (1 - c_j^q), \quad j = 1, \dots, s, \quad q = 1, \dots, \zeta. \quad (2.14)$$

2.1. taulan, *IRK* metodo nagusienak laburtu ditugu.

2.1. Taula: Runge-Kutta metodo implizituak.

Metodoa	Baldintzak	Ordena
Gauss	$B(2s)$	$C(s)$
Radau IA	$B(2s-1)$	$D(s)$
Radau IIA	$B(2s-1)$	$C(s)$
Lobatto IIIA	$B(2s-2)$	$D(s-1)$
Lobatto IIIB	$B(2s-2)$	$D(s-2)$
Lobatto IIIC	$B(2s-2)$	$D(s-1)$
		$2s$
		$2s-1$
		$2s-2$
		$2s-2$

Simplektikotasuna.

Sanz-Sernak [?] Runge-Kutta metodoa simplektikoa izateko baldintza nahikoa eta beharrezkoa,

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s, \quad (2.15)$$

dela frogatu zuen.

Runge-Kutta metodo esplizituak, simplektikoak ezin daitezkeela izan ondorioztatu daiteke. Bestalde, 1988. urtean, Sanz-Sernak *Gauss metodoa* simplektikoa zela ohartu zen.

2.3.2. Gauss metodoa.

Gauss metodoa beste *IRK* metodoekin alderatuta, bi ezaugarri bereizten ditu: Runge-Kutta metodo simplektiko bakarra eta s -ataletako ordena altueneko ($p = 2s$) *IRK* da.

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 soluzioa, diskretizazio puntuetan ez ezik, interpolazio polinomio batek modu jarraian emandako soluzioa adierazten du.

Guillou-k eta Soule-k (1969), Wright-ek (1970) [?] kolokazio metodoaren definizioa eta modu honetan kalkulatutako s-ataleko Runge-Kutta metodoa balioki-deak direla, frogatu zuten.

$$a_{ij} = \int_0^{c_i} l_j(\tau) d\tau, \quad b_i = \int_0^1 l_i(\tau) d\tau \quad (2.16)$$

non $l_i(\tau)$ Lagrangiar polinomioa dugu $l_i(\tau) = \prod_{l \neq i} \frac{(\tau - c_l)}{(c_i - c_l)}$.

Gauss nodoak. Gauss metodoak c_i ($1 \leq i \leq s$) koefizienteak "sth shifted Legendre" polinomioaren zeroak aukeratuz,

$$\frac{d^s}{dx^s} (x^s (x-1)^s),$$

Nodo hauetan oinarritutako Runge-Kutta metodoa $p = 2s$ ordena du.

Gauss metodoaren koefizienteak.

Hauek dira, $s = 1, s = 2$ eta $s = 3$ ataleako Gauss metodoak [?].

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

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

Gauss metodoaren propietateak.

Jarraian, Gauss metodoaren ezaugarri orokorrak azalduko ditugu:

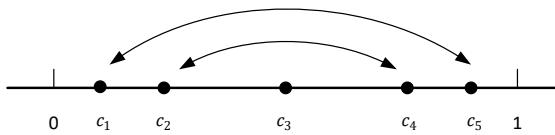
1. Metodo simplektikoa ([2.15](#)).
2. Metodo simetrikoa.

Runge-Kutta metodoa, simetrikoa izateko baldintzak hauak 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 simetrikotan, c_i balioak integrazio urratsaren erdigu-nearekiko simetrikoa dira (ikus [2.4.](#) irudia).



2.4. Irudia: Kolokazio metodoen simetria.

3. Ordena altuko metodoa. Edozein ordenako Gauss metodoa eraiki daiteke. Doitasun handiko konputazioetarako ordena altuko metodoak beharrezkoak dira: doitasun bikoitzeeko aritmetikan ($u \approx 10^{-16}$) $p \geq 8$ ordenako metodoak gomendagarriak dira eta doitasun laukoitzeeko aritmetikan ($u \approx 10^{-35}$) ordena altuagoko metodoak beharrezkoak dira.

4. Metodo orokorra. Gauss metodoa edozein ekuazio diferentzialari aplika daiteke. Sistema Hamiltondarren problemetan, ez du zertain banagarria izan behar.
5. Paralelizagarria. Ekuazio diferentzial garestiak ditugunean, s -ataletako funtzio konputazioak ($f(Y_i)$, $i = 1, \dots, s$) paraleloan kalkula daitezke.
6. Kolokazio metodoa. Zenbakizko soluzioa diskretizazio puntuetan ez ezik, integrazio tarte bakoitzean polinomio interpolatziale batek modu jarraian emandako soluzioa adierazten du.
7. A-stability and B-stability. A-stable methods have a central role in the numerical solution of stiff problems and Gauss methods are likely candidates.

2.3.3. IRK implementazioa.

IRK algoritmoa.

Jarraian, IRK metodoaren implementazioaren 1 algoritmo orokorraren laguntzarekin, implementazioaren ezaugarri orokor batzuk deskribatuko ditugu.

```

 $y_0 = y(t_0);$ 
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
```

Algoritmo 1: IRK Algoritmo orokorra

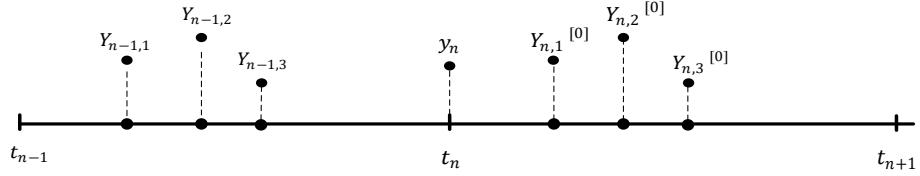
Atalen hasieraketa.

Atalen hasieraketa egokia definitu behar da. Aukera sinpleena $Y_{n,i}^{[0]} = y_n$ hasiera tzea da, baina aurreko urratseko atalen informazioa erabiliz hurbilketa hobea lortu

daiteke. Aurreko urratseko atalen interpolazio polinomio bidezko hasieraketa era honetan adierazi dezakegu,

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

Aurreko urratseko $Y_{n-1,i}$, $i = 1, \dots, s$ eta $(t_{n-1} + h, y_n)$ uneetako balioei dagokien interpolazio polinomio bidez, urrats berriaren atalen hasieraketa $Y_{n,i}^{[0]}$ kalkulatuko dugu (2.5. Irudia).



2.5. 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. Interpolazio polinomioa.

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

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 (B.1. atala). Interpolazio polinomioaren bidezko hasieraketa ona izango da, emandako urratsa ez bada oso handia eta problema zurruna ez denean. 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 [?], hasieraketa aurreratuei buruzko informazioa aurkitu daiteke.

Iterazio metodoa.

IRK metodoen erronka handiena, ekuazio-sistema ez-linealaren zenbakizko soluzioaren implementazio eraginkorra da. Problema ez-zurrenetarako, atalen hasieraketa ($Y_i^{[0]}$) egoki bat duen puntu-finkoaren iterazioa erabil daiteke. Problema zurrenetarako, puntu-finkoaren iterazioak 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 ∑_{j=1}^s a_{ij} F_j^{[k]}, i = 1, ..., s;
end
```

Algoritmoa 2: Puntu-finkoko iterazioa.

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

2. Newtonen iterazioa.

Newton metodoa iterazio bakoitza konputazionalki garestia da. Batetik, $\frac{\partial f}{\partial y}(t_n + c_i h, Y_i^{[k-1]})$, $i = 1, \dots, s$ Jacobiarrik ebaluatu behar dira. Bestetik, s-ataletako IRK metodoa eta d-dimentsioko ekuazio diferentziala baditugu, $sd \times sd$ matrizearen *LU-deskonposaketa* kalkulatu behar da.

```
for (k=1,2,...) do
    r_i^{[k]} = -Y_i^{[k-1]} + y_n + h ∑_{j=1}^s a_{ij} f(t_n + c_i h, Y_j^{[k-1]});
    Askatu  $\Delta Y_i^{[k]}$ ;
     $\Delta Y_i^{[k]} - h ∑_{j=1}^s a_{ij} \frac{\partial f}{\partial y}(t_n + c_j h, Y_j^{[k-1]}) \Delta Y_j^{[k]} = r_i^{[k]}$ ;
    Y_i^{[k]} = Y_i^{[k-1]} +  $\Delta Y_i^{[k]}$ , i = 1, ..., 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 diferentzialak, 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 diferentzialak $\ddot{q} = f(q)$ era honetako ekuazio diferentzialen kasu partikularrak dira.

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

$$\begin{aligned} p_{n+1} &= p_n + h \sum_{i=1}^s b_i f(t_n + c_i h, Q_{n,i}), \\ q_{n+1} &= q_n + h \sum_{i=1}^s b_i g(t_n + c_i h, P_{n,i}), \end{aligned} \tag{2.17}$$

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

$$\begin{aligned} P_{n,i} &= p_n + h \sum_{j=1}^s a_{ij} f(t_n + c_j h, Q_{n,j}), \\ Q_{n,i} &= q_n + h \sum_{j=1}^s a_{ij} g(t_n + c_j h, P_{n,j}). \end{aligned} \tag{2.18}$$

Problema hauetan, funtsean puntu-finkoaren iterazioa ([2 algoritmoa](#)) aplikatuko dugu, baina Hamiltondarraren egiturari esker, iterazioaren konbergentzia hobetuko dugu,

```
for (k=1,2,...) 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
```

Algoritmo 4: Puntu-finkoaren iterazioa (Metodo partizionatuak).

Bigarren ordeneko EDA

Bigarren ordenako ekuazio diferentzialen $\ddot{q} = f(q)$ azterketa egiteko, lehen ordenako ekuazio diferentzial 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 kalkulatuko dugu,

$$\begin{aligned} p_{n+1} &= p_n + h \sum_{i=1}^s b_i f(t_n + c_i h, Q_{n,i}), \\ q_{n+1} &= q_n + h p_n + h^2 \sum_{i=1}^s \beta_i f(t_n + c_i h, Q_{n,i}), \end{aligned} \quad (2.19)$$

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

$$Q_{n,i} = q_n + h \gamma_i p_n + h^2 \sum_{j=1}^s \tilde{a}_{ij} f(t_n + c_j h, Q_{n,j}). \quad (2.20)$$

Bigarren ordenako ekuazio diferentzialen puntu-finkoaren iterazioa modu era-ginkorrean 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)

IRK algoritmoa-III (bigarren ordenako EDA).

```
for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
    Hasieratu  $Q_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
    while (konbergentzia lortu) do
         $F_{n,i} = f(t_n + c_i h, Q_{n,i})$ ,  $i = 1, \dots, s$ ;
         $Q_{n,i} = q_n + h \gamma_i p_n + h^2 \sum_{j=1}^s \tilde{a}_{ij} f(t_n + c_j h, Q_{n,j})$ ,  $i = 1, \dots, s$ ;
    end
     $\delta p_n = h \sum_{i=1}^s b_i F_{n,i};$ 
     $\delta q_n = h^2 \sum_{i=1}^s \beta_i F_{n,i};$ 
     $p_{n+1} = p_n + \delta p_n;$ 
     $q_{n+1} = q_n + h \gamma_i p_n + \delta q_n;$ 
end
```

Algoritmoa 6: IRK algoritmoa-III (bigarren ordenako EDA)

Batura Konpensatua.

Zenbakizko integrazioaren urratsa 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 erabili ohi da. Teknika hau 4.atalean deskribatu dugu eta zehaztapenak 12 algoritmoan eman ditugu.

2.4. Konposizio eta Splitting metodoak.

Konposizio eta Splitting ideietan oinarritzuz, aplikazio eremu ezberdinatarako hainbat integratzale simplektiko garatu dira. Metodo hauek ez dira orokorrak, problema zehaztan aplikagarriak baizik, eta metodo oso eraginkorrak dira.

2.4.1. Konposizio metodoak.

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

Konposizio orokorrak.

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

$$\Psi_h = \phi_{\gamma_s h} \circ \cdots \circ \phi_{\gamma_1 h}. \quad (2.21)$$

Algoritmoa.

Jarraian, s-ataletako konposizio metodoen implementazioaren 7 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 7: Konposizio metodoak.

Konposizio metodoen implementazioaren ezaugarri nagusienak azpimarratuko ditugu:

1. Esplizitua.

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

2. Sekuentziala.

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

3. Memoria gutxi.

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

4. Bigarren ordenako ekuazio diferentzialak.

Bigarren ordenako ekuazio diferentziala ($\ddot{q} = f(p)$), *Störmer-Verlet* metodoan oinarritutako,

$$\begin{aligned} q_{n+1/2} &= q_n + \frac{h}{2} p_n, \\ p_{n+1} &= p_n + h f(q_{n+1/2}), \\ q_{n+1} &= q_{n+1/2} + \frac{h}{2} p_{n+1}, \end{aligned} \tag{2.22}$$

s -ataletako konposizio metodoarekin integratzeko, urrats bakoitzean, ekuazio diferentzialaren s balioztapena egin behar ditugu.

Konposizio simetrikoak.

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

$$\Psi_h = \phi_{\gamma_s h} \circ \phi_{\gamma_{s-1} h} \circ \cdots \circ \phi_{\gamma_2 h} \circ \phi_{\gamma_1 h} \tag{2.23}$$

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

CO1035: 10 ordenako konposizio metodoa.

Sofroniou eta Spaletta-ren [?, 2004] $s = 35$ eta $p = 10$ ordenako metodoa, orain arteko lortutako ordena altueneko konposizio metodo eraginkorrena konsideratu daiteke (2.2. taula). Konposizio metodo simetrikoa da eta oinarrizko metodoa simetriko eta $p = 2$ ordenakoa dela baliatuz eraikitako metodoa da.

2.2. Taula: 10 ordenako metodoa konposizio metodoa [?, 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 ($\phi_h = \varphi_{h/2}^{H_1} \circ \varphi_h^{H_2} \circ \varphi_{h/2}^{H_1}$) oinarritutako konposizio metodoaren implementazioaren zehaztasunak emango ditugu.

1. Konposizio metodo orokorra.

$$\Psi_h = \phi_{\gamma_{sh}} \circ \phi_{\gamma_{s-1h}} \circ \cdots \circ \phi_{\gamma_{2h}} \circ \phi_{\gamma_{1h}}$$

2. Stömer-Verlet oinarritzko metodoaren konposizio metoda,

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

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,

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

2.4.2. Splitting metodoak.

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

Splitting arruntak.

Splitting arrunta, H Hamiltondarra, $H = H_1 + H_2$ banatu eta Hamiltondar bakoitzari dagokion sistema modu esplizituan integratu daitezkeenean aplika daiteke. Sistemen 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,

$$\phi_h = \varphi_h^{[H_2]} \circ \varphi_h^{[H_1]} \quad \text{edo} \quad \phi_h^* = \varphi_h^{[H_1]} \circ \varphi_h^{[H_2]}. \quad (2.24)$$

Adibidea. Era horretako Hamiltondar banagarriko $H(p, q) = T(p) + V(q)$, $H_1 = T$ eta $H_2 = V$ izanik, zati bakoitzari dagokion fluxua era horretan 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 simetrikoa,

$$\phi_h = \varphi_{h/2}^{[H_1]} \circ \varphi_h^{[H_2]} \circ \varphi_{h/2}^{[H_1]}. \quad (2.25)$$

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

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

Fluxu zehatza eta zenbakizko fluxua konbinatuz.

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

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

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

Splitting orokorrak.

Aurreko splitting metodoen (2.24) orokorpena modu honetan zehaztuko dugu,

$$\phi_h = \varphi_{\beta_s h}^{[H_2]} \circ \varphi_{\alpha_s h}^{[H_1]} \circ \varphi_{\beta_{s-1} h}^{[H_2]} \circ \cdots \circ \varphi_{\beta_1 h}^{[H_2]} \circ \varphi_{\alpha_1 h}^{[H_1]}. \quad (2.26)$$

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

Algoritmoa.

Splitting metodoen implementazioaren 8 algoritmo orokorra, laburtu dugu.

```

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 8: Splitting metodoak.

Splitting metodoen implementazioarentzat, konposizio metodoen algoritmoei buruz aipatutako ezaugarri berdinak errepikatu beharko genitzke.

2.4.3. Eguzki-sistemari egokitutako splitting metodoak.

Demagun, N-gorputzeko problema grabitazionalaren Hamiltondarra,

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

Eguzki-sistemaren integratiorako erabiltzen diren koordenatu sistema nagusiak *Jacobi* edo koordenatu heliozentrikoak dira. 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 eguzki inguruko mugimendu Keplériarra den eta H_B aldiz, planeten arteko interakzioek eragiten duten perturbazio txikia.

Berridatzitako Hamiltondar funtzioak (ikus A.2. eranskina), *Jacobi* (H_{Jab}) koordenatuetan H_B bakarrik kokapenen araberakoa eta *Heliozentrikoetan* (H_{Hel}), H_B kokapen naiz abiaduraren araberakoa da.

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

Eguzki-sistemaren N-gorputzeko problema gravitazionalari egokitutako zenbakizko bi integratzaile simplektiko azalduko ditugu. Lehena, Laskar-ek eta Robutel-ek [?, 2001] definitutako *SABAC*₄ integratzailea eta bigarrena, Blanes-ek [?, 2013] [?] definitutako *ABAH*1064 integratzailea.

*SABAC*₄ integratzailea.

Laskarrek (2001) *SABA*_n eta *SBAB*_n integratzaile simplektikoak [?] proposatu zituen. Metodoen ordena $K_{SABA_n} = K_{SBAB_n} = \mathcal{O}(h^{2n}\epsilon + h^2\epsilon^2)$ da; urrats luzera handitarako eta ϵ balio txikitarako, gai nagusia ($h^{2n}\epsilon$) izango da.

Bigarren integratzaile famili bat ere *CSABA*_n eta *CSBAB*_n (*Corrected integrators*) definitzen ditu. Hauetan, urrats bat gehitutako integratzaileak dira eta ordena $K_{CSABA_n} = K_{CSBAB_n} = \mathcal{O}(h^4\epsilon + h^p\epsilon^2)$, non $p = n+2$ edo $p = n+3$ den. Bigarren metodo hauek, urrats tamaina txikia beharrezkoa den doitasun handiko integrazioetarako erabilgarriak dira.

Eguzki sistemaren epe luzeko integrazioan [?] 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 2.3. taulan zehaztu ditugu.

*ABAH*1064 integratzailea.

Blanes-ek (2013), *ABA* eta *ABAH* metodo simplektikoak proposatu zituen [?]. *ABA* metodoak, Jacobi koordenatuak erabiltzen direnean egokitutako integratzaileak eta *ABAH* integratzaileak, koordenatu Heliozentrikoak erabiltzen direnean egokitutako integratzaileak.

Atal honetan, koordenatu Heliozentrikoei egokitutako *ABAH*1064, $p = 10$ ordenako metodoa deskribatuko dugu. Eguzki sistemaren integraziorako koordenatu Heliozentrikoei dagokion Hamiltondarra era honetakoa dugu,

$$H(p, q) = H_K(p, q) + H_I(p, q), \quad H_I(p, q) = T_1(p) + U_1(q).$$

2.3. Taula: *CSABA*₄ splitting metodoa [?].

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		

2.4. Taula: *ABAH1064* splitting metodoa [?].

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

$H_I(p, q)$ 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,

$$ABAH1064 = \prod_{i=1}^s \varphi_{a_i h}^K \circ \tilde{\varphi}_{b_i h}^I$$

non a_i, b_i koefizienteak (2.4. Taula) taulan definitzen diren.

2.5. Laburpena.

Hurrengo taulan (2.5. Taulan) ordena altuko hiru integratzaire simplektikoen ezau-garriak laburtu ditugu.

C kodean konposizio eta splitting metodoak implementatu ditugu. Zenbakizko esperimentuetan, gure implementazio berriak *CO1025* konposizio eta *ABAH1064* metodoekin konparatu ditugu.

Metodo simplektikoei buruzko liburu monografiko hauek gomendatuko ditugu: Sanz-Serna and M.P. Calvo's Numerical Hamiltonian Problems (1994) [?]; E.

2.5. Taula: Integrazio metodoen laburpena

	C1035	ABAH1064	GAUSS-12
	Konposizio met. Sofronio (2004)	Splitting met. Blanes et al. 2013	IRK met.
Hamiltondarra	Orokorra	Perturbatua	Orokorra
Mota	Esplizitua	Esplizitua	Implizitua
Ordena	10	10	12
Atalak	35	9	6
Parall.	Ez	Ez	Bai

Hairer, C. Lubich and G. Wanner's Geometrical Numerical Integration (2001) [?]; B. Leimkuhler and S. Reich's Simulating Hamiltonian Dynamics (2004) [?]; Feng, Kang and Qin, Mengzhao Symplectic geometric algorithms for hamiltonian systems (2010) [?].

3. Kapitulua

Problemak.

3.1. Sarrera.

Tesiaren erdigunea N-gorputzeko problema grabitazionala da. Gure helburua, eguzki-sistemaren problemaren simulaziorako zenbakizko integratzaile eraginkorra implementatzea da. Eguzki-sistemaren bi eredu simple konsideratu ditugu: lehena, *kanpo-planeten* izeneko problema (eguzkia, kanpo-planetak eta Plutonek osatutakoa), eta bigarren eredu konplexuagoa, *9-planeten problema* izeneko problema (eguzkia, 8 planetak eta Plutonek osatutakoa). Bi eredu hauetan, gorputzak masa puntualak dira eta gorputz hauen arteko erakarpen grabitazionalak bakarrik konsideratu ditugu (erlatibilitate efektua eta beste hainbat indar ez-grabitazionalak gure lan-eremutik kanpo utzi ditugu).

Zenbakizko metodo simplektiko nagusienak esplizituak direla eta Hamiltondar banagarriak diren problemei aplika daitezkeela, gogoratu behar dugu. Hortaz gain, problema zurruna bada metodo esplizituak ez direla eraginkorrik eta metodo implizituek abantaila azaltzen dutela esan beharra dago. Gauss metodoa orokorra eta implizitua izanik, gure implementazioa problema zurrunetarako eta Hamiltondar banagarria ez den problemetarako aplikagarria dela ere erakutsi nahi dugu.

Aurretik esandakoari jarraituz, pendulu bikoitzak, problema osagarri gisa aukeratu dugu zenbakizko esperimentuak modu aberatsago eta zabalago batean egiteko. Pendulu bikoitzaren bi bertsio konsideratu ditugu: pendulu bikoitz arrunta eta pendulu bikoitz zurruna.

N-gorputzeko problema grabitazionalaren Hamiltondarra banagarria da baina pendulu bikoitzarena aldiz, ez da banagarria. Bestalde, pendulu bikoitzari malguki bat gehituz problema zurruna bilakatuko dugu eta problema hauen zaitasunei nola aurre egin erakusteko. Gainera, eguzki-sistema kaotiko [?] konsideratzen dela jakinik, pendulu bikoitz arruntak izaera kaotikoa azaltzen duen hasierako balio zehatzak aukeratu ditugu. Problema kaotikoak, hasierako balio edo para-

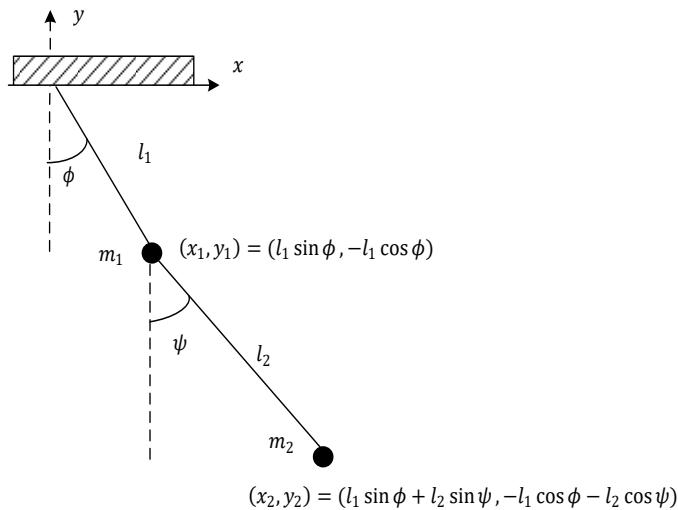
metroen perturbazioekiko, trunkatze edo birbitze erroreakiko esponentzialki oso sentikorrik dira.

3.2. Pendulu bikoitza.

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

3.2.1. Pendulu bikoitz arrunta.

Planoan pendulu bikoitzaren problema (ikus 3.1. irudia) era honetan definituko dugu: m_1 eta m_2 masadun bi penduluz eta elkar lotuta dauden l_1 eta l_2 luzerako makilez (masa gabekoak kontsideratuko ditugunak) osatuta sistema mekanikoa. 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 $\psi = \phi + \theta$.



3.1. Irudia: Pendulu bikoitz arrunta.

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

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

Sistemaren parametroak. Gure esperimentuetarako honako parametroak konsideratuko ditugu,

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

Hasierako balioak. Bi hasierako balio ezberdin konsideratu ditugu [?]: lehenak, izaera ez-kaotiko 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 eta zenbakizko soluzioa, $m = 2^{10}$ urratsero itzuli 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 eta zenbakizko soluzioa, $m = 2^8$ urratsero itzuli dugu.

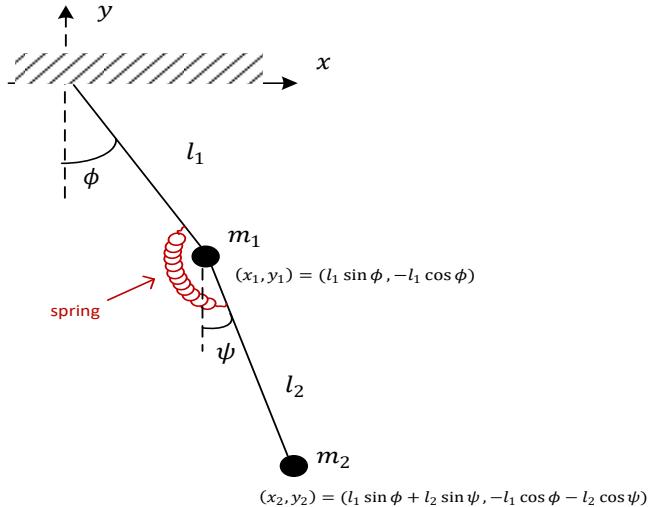
Urrats luzera, $h = 2^{-7}$ aukeratu dugu, trunkatze errorea biribiltze errorea baino txikiagoa izan dadin.

3.2.2. Pendulu bikoitz zurruna.

Pendulu bikoitz arruntari malguki bat gehitutako sistema da (ikus 3.2. irudia) : l_1 eta l_2 luzerako makilez elkar lotuta dauden m_1 eta m_2 masadun bi pendulu eta hauen artean k malgutasun duen malgukia osatzen duten sistema mekanikoa. $k = 0$ balioarentzat, problema ez da zurruna 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 penduluaren ardatz bertikalarekiko angelua da eta bigarren penduluaren angelua, era honetan definituko dugu $\psi = \phi + \theta$.

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



3.2. Irudia: Pendulu bikoitza (zurruna).

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, [?] 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 $k \rightarrow \infty$ bornatua dagoen.

$k = 0$ problema ez-zurrenerako, $h = 2^{-7}$ 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 eta zenbakizko soluzioa, $m = 2^{10}$ urrats oro itzuli dugu.

3.3. N-Gorputzen problema.

Newtonen lege grabitazionalen araberako N-Body problemaren ekuazio differentzialak 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 ordezkatz $p_i = m_i * \dot{q}_i$, N-Body problemaren formulazio Hamiltondarra 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,

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

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

1. Masa zentroaren sei integralak.

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

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

Eta ondorioz,

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

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

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

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

2. Momentu angeluarra.

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

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

3. Energia.

Hamitoniar sistema osoaren energia da eta problemaren beste integrala da,

$$E = H(q, p) = \text{kons.}$$

Problemaren 10 integral hauek, sistemaren ordena gutxitzeo edo zenbakizko integrazioa doitasuna neurtzeko erabili daitezke. Guk koordenatu barizentrikoak (koordenatu sistemaren jatorria masa zentroaren kokapena) erabiliko ditugu eta koordenatu hauetan, $P = 0$ eta $B = 0$ dira. Beraz, gorputzen kokapen eta abiadurak $\hat{Q} = 0$ eta $\hat{V} = 0$ izan daitezzen, integratzeko erabiliko ditugun hasierako balioak (\hat{q}_i, \hat{v}_i) modu horretan finkatuko ditugu,

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

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

3.4. Eguzki-sistema.

3.4.1. Sarrera.

Eguzki-sistemaren planeten orbiten mugimenduaren eredu matematikoa, Hamiltondar sistema bati dagokion ekuazio diferentzial arrunten bidez formulatzen da. Ekuazio diferentzial arrunten sistema, $6N$ ordenakoa da (N planeten kopurua iza-nik).

Eguzki sistemaren eredu simplea integratuko dugu. Eguzki-sistemaren gorputzak masa puntualak konsideratuko ditugu eta gure ekuazio diferentzialak definitzeko, soilik gorputz hauen arteko erakarpen grabitazionalak kontutan hartu ditugu. Eguzki-sistemaren eredu konplexuagoen erlatibilitate efektua, gorputzen formaren eragina, eta beste zenbait indar ez-grabitazionalak ez ditugu konsideratu.

Eguzki-sistemaren gorputz nagusien (eguzkia eta planetak) integrazioetara mugatuko gara. Ereduen gorputz kopurua txikia izango da, $N = 6$ (eguzki-sistemaren eredu simplea) eta $N = 10$ (eguzki sistemaren eredu osoa) gorputz kopuruen arte-koa.

Eguzkiak planetak baino 1.000 aldiz masa handiago du eta hauxe da, eguzki-sistemaren ezaugarri garrantzitsuenetakoak: eguzkiaren grabitazio indar nagusi batetik eta planeten arteko perturbazio txikiak sortzen duten sistema dinamikoa da. Planetak eta bere sateliteen mugimenduan kolisio gertuko egoerarik edo ezentri-zidade handiko orbitalak ez dago. Beraz, ikuspuntu honetatik sistema dinamiko simplea dela, konsideratu daiteke. Eguzki-sistema egonkorra konsideratzen da, hau da, hurrengo bilioi urteetan planeten arteko talkarik edo planeten kanporatz-rik gertatzea, ez da espero [?] [?].

Egonkortasunaren azterketa zehatzago batean, sistema erregularrak eta kao-tikoak bereiz ditzakegu. Sistema erregularretan, hasierako balioen perturbazio txikien eragina, denboran zehar poliki hasiko da eta beraz, epe luzeko integrazio zehatzak posible dira. Sistema kaotikoetan aldiz, eboluzioa hasierako balioekiko oso sentikorra da eta gertuko soluzioen diferentzia esponentzialki hasiko da. Hasierako uneko diferentzia $d(0) = d_0$ bada, orduan distantzia,

$$d(t) \approx d(0)e^{t\lambda}$$

espresioaren arabera hasiko da, non λ Lyapunov esponentziala esaten zaion. Eguzki-sistema kaotikoa konsideratzen da eta Laskar-ek [?] Lyapunov denbora $\lambda^{-1} \approx 5$

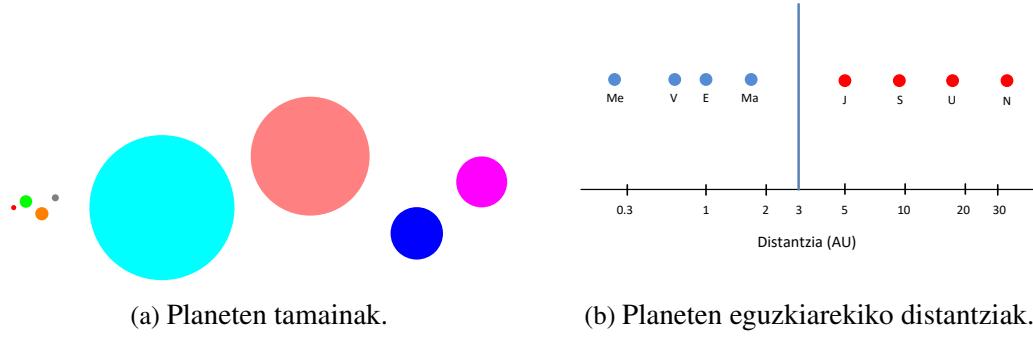
milioi urtetan finkatzen du eta honako espresioa, kalkulatzeko proposatzen du,

$$d(t) \approx d(0)10^{t/10}.$$

Espresio honen arabera, 10^{-10} tamainako hasierako erroreak baditugu, orduan $d(10) \approx 10^{-9}$ eta $d(100) \approx 1$ dira. Ondorioz, eguzki sistemaren $10 - 20$ milioi urteko integrazioetarako, doitasun handiko soluzioak lortuko ditugu baina 100 milioi urteko integrazioen soluzioak ez dira esanguratsuak izango.

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 $h = 0.0025$ urteko 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 da eta urrats kopurua, 5×10^{10} ingurukoa da.

Eguzki-sistemaren problemaren eskalak, anitzak dira. [3.3.](#) irudian, planeta nagusien tamainak eta eguzkiarekiko distantziak irudikatu ditugu. Ondorengo [3.1.](#) taulan, eguzki-sistemaren planeten ezaugarri nagusienak eman ditugu. Denbora eskalari dagokionez, ilargiak luraren inguruko orbita 27.32 egunetakoa da, lurrrak eguzkiarekiko inguruko orbitaren periodoa 1 urtekoa eta Neptunorena 163 urtekoa.



(a) Planeten tamainak.

(b) Planeten eguzkiarekiko distantziak.

3.3. Irudia: Ezkerreko (a) irudian, planeten arteko tamainen proportzioak eta eskubiko irudian, planeten eguzkiarekiko distantziak irudikatu ditugu.

Hiru dira erabiltzen diren koordenatu sistema nagusienak:

1. Koordenatu cartesiarrak.
2. Koordenatu heliozentrikoak.

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

Planeta	Masak kg	Distantzia AU	Periodoa urteak	Ezentrizidadea
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

3. Koordenatu jacobiarak.

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

3.4.2. Problemak.

Eguzki-sistemaren simulaziorako test problemak deskribatuko ditugu. Bi gorputzen problematik abiatuta, gero eta problema konplexuagoak azalduko ditugu. PW, Sharp-ek (2.001) eguzki-sistemaren problemen bilduma interesgarria egin zuen [?], eta bertan problema guzti hauek 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 diferenzialak 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 hasierako baliodun problemaren soluzio zehatza kalkulatu daiteke: une bateko kokapen eta abiadurak emanik, denbora tarte bat (Δt) igarotakoan (positibo ala negatiboa), zehazki kokapen eta abiadura berriak ezagutu daitezke. Az-pimarratu nahi dugu, funtsezkoa dela eguzki-sistemaren integrazio metodoentzat, Kepler problemaren doitasun handiko eta kalkulu eraginkorra konputatzeari. Erreferentziazko kepler problemaren implementazioak, Danby (1992) [?] eta J.Wisdom (2015) [?] ditugu.

Kepler problemaren implementazioa garatu dugu eta ideia nagusia, hitzez azalduko dugu. 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 aldagaiaiak konstante mantentzen dira: beraz E_0 balioa emanda, Δt denbora tartea aurrera egin eta E_1 balioa berria kalkuluko dugu. Azkenik, koordenatu eliptikoetatik koordenatu cartesiarretara itzulpena eginez, kokapen eta abiadura berriak eskuratuko ditugu.

$$(q_0, v_0) \in \mathbb{R}^6 \longrightarrow (a, e, i, \Omega, E_0) \in \mathbb{R}^6$$

$$\downarrow \Delta t$$

$$(q_1, v_1) \in \mathbb{R}^6 \longleftarrow (a, e, i, \Omega, E_1) \in \mathbb{R}^6$$

Implementazioaren garapenaren zehaztasun guztiak eranskinaren A.1. atalean eman ditugu.

Hiru gorputzen problema.

Hiru Gorputzen Problema Murriztuan (*R3BP*), masa ezberdineko hiru gorputz elkarrekiko erakarpen eraginpean, espazioan libreki mugitzen dira. Honako hurbilpenak kontsideratuz, definitzen da Hiru Gorputzen Problema Murriztua:

- Ohikoa den moduan Newton eredu grabitazionalean, gorputzak masa puntualak kontsideratuko ditugu.
- Horietako bi gorputz nagusiak (m_1, m_2), bere masa zentroaren inguruan orbita zirkularrean mugitzen dira. Adibidez, lurra eta ilargia kontsideratu daitezke. Ilargiak ezentrizitate txikiko ($e = 0.05$) mugimendu eliptikoa du eta suposizio hau onargarria da.

- Problema orokorrean, hiru gorputzen masak edozein izan daitezke. Murriztuan aldiz, bi gorputz nagusien masa edozein delarik, hirugarrenarena beste bi gorputzen masa baino askoz ere txikiagoa da. Adibidez satelite artifizial bat konsideratzen badugu, bere masa 1.000 tonatako (10^6 kg) ingurukoa izango da eta ilargiaren masarekin konparatuko bagenu ($I_M = 7.3477 \times 10^{22}$ kg), bere masa askoz ere txikiagoa da.
- Hirugarren gorputzak (m_3) beste bi gorputz nagusien masarekin alderatuta oso masa txikia duenez, ez du m_1 eta m_2 gorputzen mugimenduarekiko eraginik. Hirugarren gorputza, bi gorputz nagusien eraginpean hauen orbitaren plano berdinean mugitzen dela suposatuko dugu.
- Eguzki-sistemaren gainontzeako gorputzen eragina baztertu da. Lurra-IIlargi sistemaren, eguzkiaren grabitazio indarra ez dagoela konsideratu da.

$(x, y) \in \mathbb{R}^2$ masa txikiko gorputzaren kokapen koordenatuak izanik, hauetan errortazio koordenatu sistemaren dagokion mugimenduaren ekuazioak (4 dimentsioko sistema dinamikoa),

$$\begin{aligned}\dot{x} &= p_x + y, \\ \dot{y} &= p_y - x, \\ \dot{p}_x &= p_y - umu \frac{x + \mu}{r_1^3} - \mu \frac{x - (umu)}{r_2^3}, \\ \dot{p}_y &= -p_x - umu \frac{y}{r_1^3} - \mu \frac{y}{r_2^3},\end{aligned}$$

non $r_1 = ((x + \mu)^2 + y^2)^{1/2}$, $r_2 = ((x - umu)^2 + y^2)^{1/2}$, $-\mu = (-m/M + m)$ eta $umu = 1 - \mu$ diren.

Problema ezberdinak deskribatzen dira μ parametroaren arabera: $\mu = 0.01277471$ balioarekin finkatuta, Lurra-IIlargina modeloari dagokio. Gorputz nagusi handiena (Lurra) $(-\mu, 0)$ eta gorputz nagusi txikiena (Ilargin) $(1 - \mu, 0)$ posizioan kokatzen dira. Ilarginak orbita zirkularra du luraren inguruan eta ekuazio differentzialek satelitearen mugimendua deskribatzen dute.

Sistema dinamikoaren energia, soluzioan zehar konstante mantentzen da,

$$E = \frac{1}{2}(p_x^2 + p_y^2) + p_x y - p_y x - \left(\frac{1 - \mu}{r_1} \right) - \frac{\mu}{r_2} - \frac{1}{2}\mu(1 - \mu).$$

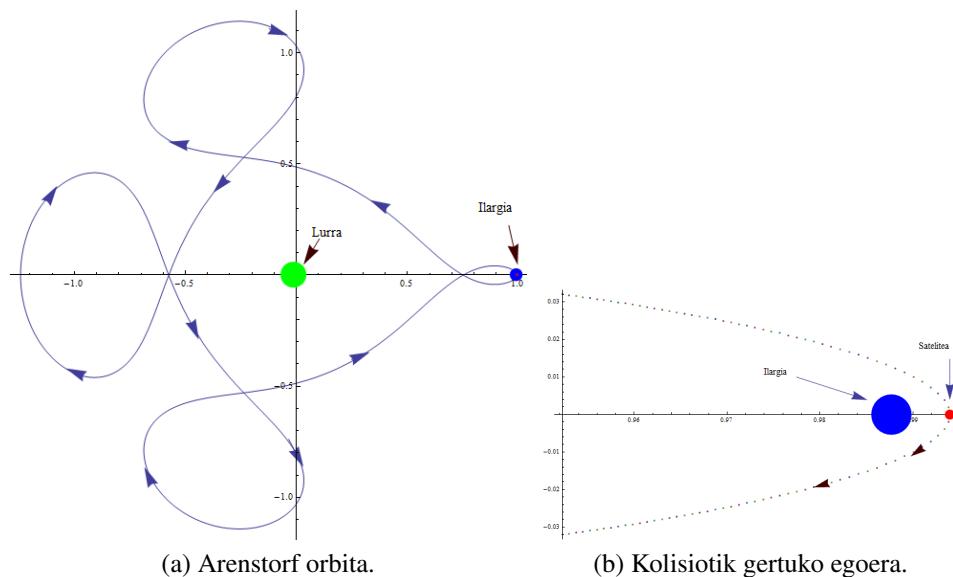
Ezaguna da, hasierako balio hauetarako (3.2. taula) soluzioa

$$t = 17.0652165601579625588917206249$$

periodikoa dela (3.4. irudia),

3.2. Taula: R3BP problemaren hasierako balioak.

Gorputza	Balioak
Satelitea	x, y 0.994 0
	p_x, p_y 0. $-2.00158510637908252240537862224 + 0.994$



3.4. Irudia: R3BP. Arenstorf orbita izeneko soluzio periodikoa. $t = 0$ unean kolisiotik gertuko egoera; Ilargia (0.987723, 0) eta satelitea (0.994, 0) posizioa.

Kampo-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, planetan nagusi hauen eboluzioa eguzki-sistema osoaren zati garrantzitsuena da eta barne-planeten mugimendua kontutan hartzeak ala ez, kanpo-planeten zenbakizko integrazioaren-gan oso eragin txikia du. Bigarrenik, urrats luzera handi erabili daiteke eta beraz, epe luzeko integrazioak errazten dira (konputazio denbora gutxiago behar dela-ko). Hirugarrenik, Pluton orbitaren berezitasunak ikertzea, 1960 – 1980 urteetan interes handikoa izan zen.

Hasierako balioak [?] liburutik hartu ditugu. Planetei dagokien masak 3.3. taulan eta kokapenak/abiadurak 3.4. taulan laburtu ditugu. planeten masak eguzkiaren erlatiboak dira, hau da, eguzkiaren masa 1 da eta grabitazio konstantea $G = 2.95912208286 \times 10^{-4}$. Barne-planeten masak eguzkiaren masari gehitu zaio

eta horregatik, eguzkiaren masak, $m_0 = 1.00000597682$ balioa hartzen du.

3.3. Taula: Kanpo-planeten masak.

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

3.4. 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. Orbita periodo txikiena 50 aldiz txikiagoa (Merkurioren 0.24 urtetako periodo, Jupiterren 11.86 urtetako periodoarekin alderatuta) da. Merkurioren orbitaren ezentrizidadea $e = 0.206$ (Jupiterren orbitaren ezentrizidadea $e = 0.048$) da.

Eedu 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) [?] azken efemeride artikulutik hartu ditugu. Eguzki eta planeten hasierako kokapenak (AU) eta abiadurak (AU/egun), Julian data (TDB) 2440400.5 (1969. ekainaren 28) eta ICRFR2 (International Celestial Reference Frame) koordenatu sisteman [3.6.](#) taulan laburtu dugu. Planeta bakoitzari dagokion, GM balioa [3.5.](#) taulan laburtu dugu.

3.5. 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	$0.109318945074237400e - 10$

Laskar-en eredu.

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

Laskar-ek, 2.011. urteko epe luzeko zenbakizko integratziorako [?] eguzki-sistemaren eredu deskribatuko dugu. Hasierako integrazioetan, eguzkia, 8 planetak, Pluton eta ilargia bakarrik konsideratu zituen. Eguzkiaren erlatibilitate efektua (Saha eta Tremainen [?] teknikaren arabera) eta eredu errealistaren indar ez gravitazional garrantzitsuenak aplikatu zituen. Azken integrazioetan, Zeres, Palas, Vesta, Iris eta Bamberga asteroideak gehitu zituen zituen.

Ilargia gorputz independente gisa konsideratu zuen. Ilargiaren lurraurrekiko distantzia (380.000 km), beste gorputzekiko distantziekin alderatzen badugu (eguzkiarekiko 150.000.000 km eta Artizarrarekiko 45.000.000 km) oso txikia da. Hori dela eta, ilargiaren kokapena, eguzki-sistemaren barizentroarekiko konsideratu

3.6. Taula: Eguzki eta 9 planeten hasierako balioak, , kokapenak (x, y, z) eta abiadurak (v_x, v_y, v_z).

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	-0.40154022645315222236
	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

3.7. 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

ordez, lurrarekiko kontsideratuz doitasun handiago 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 lurrarekiko 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 ebalutzeko (q_e, q_m) aldagaiak erabiliko ditugula eta aldagai berriak. (q_B, q_{em}) integratzeko erabiliko ditugula.

```
Lurra, Ilargia = {q_B, q_{em}};
for i ← 1 to endstep do
    {q_e, q_m} ← {q_B, q_{em}};
    Ebaluatu ȳ = f(y);
    {q_B, q_{em}} ← {q_e, q_m};
    Integrazioa (q_B, q_{em});
end
```

Algoritmoa 9: Ilargiaren kalkuluak.

3.8. 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

3.5. Laburpena.

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 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 aritmetikaren oinarria eta biribiltze errorea azalduko ditugu. Ondoren, konputazioetan biribiltze erroreak gutxitzeko teknika ezagun batzuk azalduko ditugu.

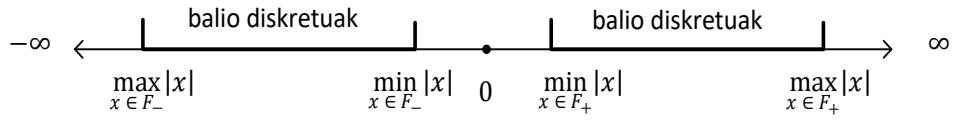
4.2. IEEE-754 estandarra.

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),

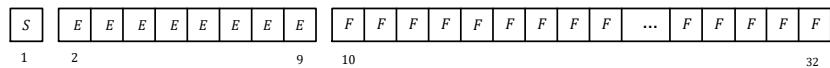
1. m bitez osatutako zatia, mantisa (M) izenekoa. Horietako bit batek (S) zeinuadierazten du. Bestalde M mantisa modu normalizatu honetan emana da, $\pm 1.F$ eta zati erreala (F) bakarrik gorde behar da.
2. Esponentea (E), $(n - m)$ bitez adierazitako zenbaki osoa. Zeinuarentzat ez da bit zehatzik, baizik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

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



32

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 [?] 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 [?]. 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*).
2. *Termino-anitzeko adierazpena*. Zenbakiak ebaluatu gabeko hainbat koma-higikorreko makina zenbaki estandarren batura gisa adierazten dira (adb. *Bailey QD liburutegia*) eta exekuzioaren ikuspegitik, hardware bidezko implementazioaren abantaila dute.

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

Laskar-ek epe luzeko eguzki-sistemaren simulazioaren (-250 eta $+250$ milioitako integrazio tarteak) konputaziorako kalkuluak [?], 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 adierazi 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 $x \in \mathbb{R}$ bada, non $f_1 \leqslant x \leqslant f_2$,

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

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 frogatu daiteke.

Aritmetikaren errorea.

Koma-higikorreko zenbakien arteko eragiketa baten emaitzak, ez du \mathbb{F} multzoan adierazpen zehatza izan behar eta emaitzaren hurbilpena kontsideratuko da. Adibidez, orokorrean m digituzko bi zenbakien biderketaren emaitza zehatza adierazteko, $2m$ digituzko mantisa behar dugu (m digituzko galera) [?]. Salbuespina, 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 frogatu daiteke.

Zenbakizko algoritmoen biribiltze errorearen eraginaren azterketak, propietate honetan oinarritzen dira. Bestalde, errore erlatiboak emaitzaren digitu zuzenak neurtzen du:

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

Biribiltze errorearen propagazioa.

Ohiko konputazioetan, eragiketa aritmetiko kopuru handia egin behar dugu emaitza lortzeko. Batzuetan, eragiketa bakoitzaren 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 [?][?]. 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 [?] asmatu zuen. Koma-higikorreko $\tilde{x}, \tilde{y} \in \mathbb{F}$ bi zenbakien non $|\tilde{x}| \geq |\tilde{y}|$, arteko baturari $\tilde{z} = \tilde{x} \oplus \tilde{y}$ 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 10: Fast2Sum.

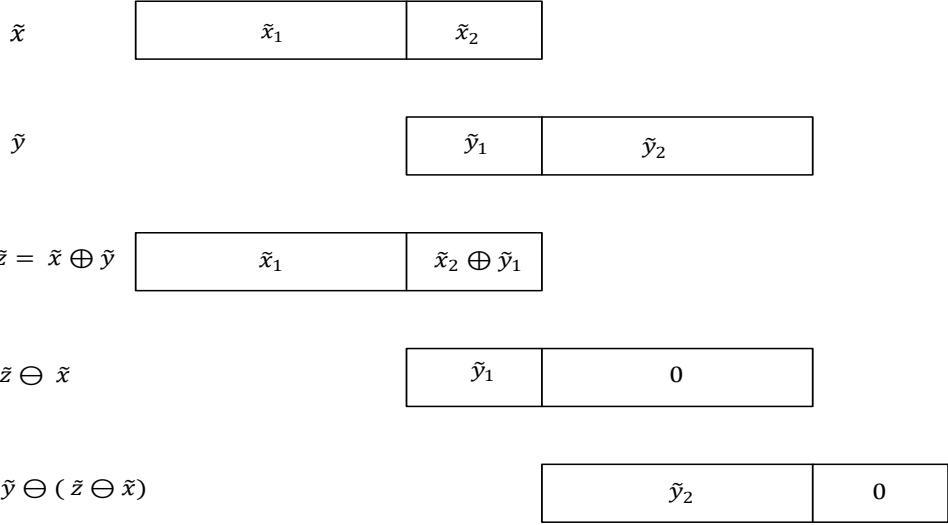
4.3.irudiaren laguntzarekin hobeto uler daiteke baturaren biribiltze errorearen kalkulua.

Batura konpensatua.

Era honetako batura errekurtsiboetan,

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

biribiltze errorea gutxitzeko teknika ezaguna da [?]. Ideia da, bi zenbakien baturan egindako biribiltze errorea lortu, eta errore hau hurrengo baturan erabiltzea. Jarraian azaltzen den moduan, urrats bakoitzaren amaieran errore estimazioa (e_i)



4.3. Irudia: Batuketaren biribiltze errorea.

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 11: Kahan-en batura konpensatua.

Knuth-ek eta Kahan-ek [?], 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, 11 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 ditu, eta $(\tilde{z}_{n+1} + e_{n+1}) \approx (z_0 + e_0 + x_0 + x_1 + \dots + x_n)$ adierazi nahi duen.

Zenbakizko integrazioak.

Zenbakizko integrazioetan, era honetako batura errekurtsiboa kalkulatu behar ditugu,

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

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 12: 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 azaldu daiteke. Zenbakizko soluzioa, doitasun bikoitzeko bi balioren bidez (\tilde{y}_n, e_n) adierazten ari gara (ia doitasun laukoitza) eta beraz, interpretazio honen arabera, konputazio eragiketa batzuk ia doitasun laukoitzean egiten ariko ginateke. Zentzu honetan gure implementazioan, hasierako balio zehatza $y_0 = y(t_0)$, doitasun bikoitzeko bi zenbakiren bidez (\tilde{y}_0, e_0) 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* (*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 eskalaren, 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 13: 2MultFMA.

Sterbenz Teorema.

Sterbenz teoremaren arabera [?], 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.

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

5. Kapitulua

Zientzia konputazioa.

5.1. Sarrera.

Azken hamarkadetan, zientzia konputazionalaren hazkundea oso handia izan da eta bere erabilera ia zientzia arlo guztietara zabaldu da. Zientzialariek ahalmen handiko tresna berria (zenbakizko simulazioa) eskuragarri dute, neurri handi batean 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 txikienarekin, diseinatu behar dira.

Azkenik aipatu nahi dugu ere, aplikazio baten eraginkortasuna ez dela exekuzio denboraren arabera bakarrik neurtu behar. Hori bezain garrantzitsua da kode ona idaztea [?] eta zentzu honetan hiru ezaugarri hauek bereziki zaindu behar dira:

- Errorerik gabeko kodea.
- Kode argia idaztea.
- Etorkizunean erraz aldatu daitekeen kodea.

5.2. 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 mega-flops).

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, diferentzia hau handituz joango dela. Beraz, kodearen exekuzioa azkartzeko derrigorrezkoa da konputagailuan memorien arteko datuen mugimendua minimizatzea.

Exekuzio denboren neurketa.

Unixeko *time* agindua, konputazioen denborak ezagutzeko erabili daiteke:

```
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_cpy_time;

clock0= clock();
<neurtu nahi den kodea>
clock1=clock();

elapsed_cpy_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.

5.3. Hardwarea.

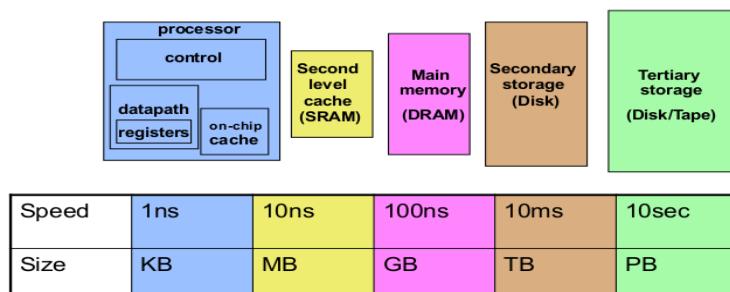
Orokorean, gaur-egungo konputagailuak (super-konputagailu, eramangarri,...) paraleloak dira. 1986 – 2002 urteen artean, txip barruko transistore dentsitatea han-ditzen zen heinean, prozesadore bakarreko konputagailuen eraginkortasuna hobetuz joan zen. Baina teknologi honen garapena muga fisikoetara iritsi zenean, bide honetatik konputagailuen abiadura hobetzea ezinezkoa bilakatu zen. Horrela,

2005.urtetik aurrera fabrikatzailen 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 ezberdinen hierarkia ([5.1.irudia](#)) eta funtzionamendua deskribatuko dugu.



5.1. 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) txikiena 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 datu (argazki, bideo,...) eta erabilgarri ditugun programa guztiak gordetzen dira.

CPU-k datu bat behar duenean, memoria hierarkian zehar bilatuko du: lehenik L_1 cachean, ondoren L_2 cachean,...eta hauetan ez badago, memoria nagusira joko du. Memoria nagusi eta cache memoria arteko irakurketa eta idazketa guzti hauetan, informazio kontsistentzia mantentzeko hainbat arau aurrera ematen dira.

Cache memoria lerroka egituratuta dago eta lerro bakoitza 64 edo 128 bytez (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. Programatzaileak,

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, cachera 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 lengoainan 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 14): matrizearen lehen osagaia $a(1, 1)$ behar dugunean, memoria nagusitik Cachera 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 zutabekoa (Algoritmoa 15): bigarren osagaia ($a(2, 1)$) gehitzeko memoria irakurketa berri bat egin behar dugu.

```

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 14: Memoria atzipena eraginkorra.

Arkitektura motak.

Prozesadore anitzeko konputagailu hardware berriak, konplexuak eta heterogeneoak dira. Egoera honetan, programzaileak zaitasun handiak ditu arkitektura berriek eskaintzen dituzten gaitasunak ondo kudeatzeko [?].

```

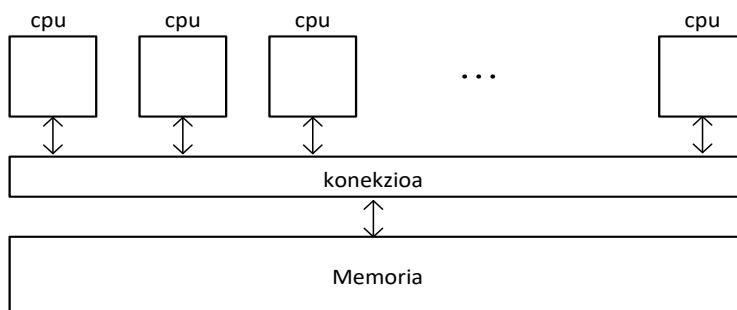
int n;
double a[n][m];
sum = 0;
for j ← 1 to m do
    for i ← 1 to n do
        sum += a(i, j);
    end
end

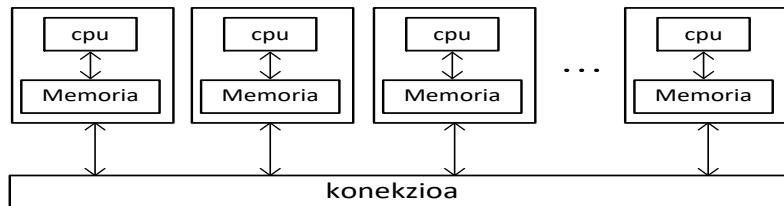
```

Algoritmoa 15: Memoria atzipena ez-eraginkorra.

Lehen hurbilpen modura bi sistema nagusi bereiziko ditugu: memoria konpartitutako eta memoria banatutako sistemak. Memoria konpartitutako sistemetan, prozesadore guztiak memoria osoa konpartitzen dute eta implizituki konpartitutako datuen atzipenaren bidez komunikatzen dira ([5.2. irudia](#)). Memoria banatutako sistemetan aldiz, prozesadore bakoitzak bere memoria pribatua du eta esplizituki bidalitako mezuen bidez komunikatzen dira ([5.3. irudia](#)). Aipatzekoada, 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. Jokoentzako animazio industriean, grafiko oso azkarra beharrak bultzatuta sortutako teknologia da. Oinarrian, imajinak pantailaratzeko prozesagailu asko paraleloan lan egiten dute eta azken hamarkadan, *GPU* unitate hauek zientzia konputaziora zabaldu dira.

**5.2. Irudia:** Memoria konpartitutako sistemak.



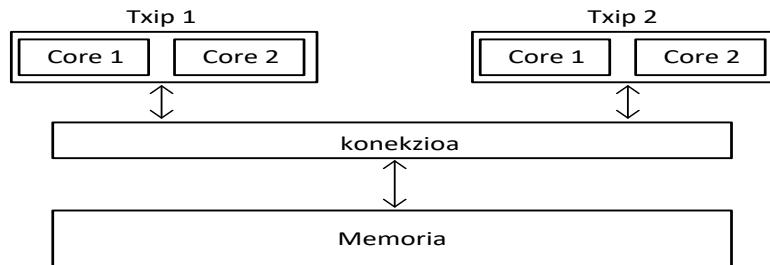
5.3. Irudia: Memoria banatutako sistemak.

Oinarrizko konputagailu paraleloa.

Gure lanerako, oinarrizko konputagailu paraleloak konsideratuko ditugu: memoria konpartitutako eta prozesadore anitzeko unitate bat edo gehiagoz osatutako sistemak. Prozesadore anitzeko unitate bakoitzak txipean CPU bat baino gehiago ditu. Normalean CPU bakoitzak $L1$ bere cache memoria du. Aipatzeko da, era honetako sistemetan prozesadore kopurua mugatua dela (normalean ≤ 32).

Memoria konpartitutako sistemen artean bi mota bereiziko ditugu:

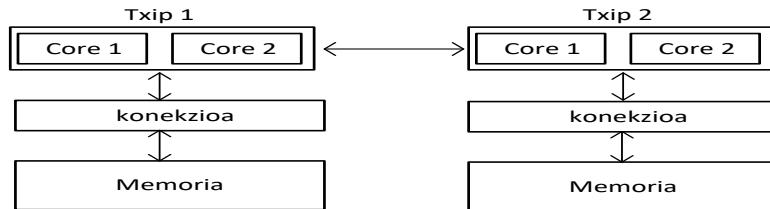
1. UMA sistemak (uniform memory access). Txip prozesadore guztiak zuzean memoria konektatuta daude eta guztiak atzipen denbora berdina dute ([5.4..irudia](#)).



5.4. Irudia: Memoria konpartitutako sistemak (UMA).

2. NUMA sistemak (nonuniform memory access). Txip prozesadore bakoi-tza, 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 ([5.5..irudia](#)).



5.5. Irudia: Memoria konpartitutako sistemak (NUMA).

SIMD eragiketak.

Konputagailuek, *Single Instruction Multiple Data* (SIMD) instrukzioetan oinarritutako paralelizazio eskaintzen dute. 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 16: SIMD (bektorizazioa).

5.4. Programazio lengoaiak.

Fortran eta C, aplikazio zientifikoetan gehien erabiltzen diren programazio lengoaiak dira [?]. Fortran (formula translation) 1950 hamarkadan garatutako goimailako lehen lengoaia izan zen eta oraindik ere, oso zabaldua da. Fortran estandarraren hainbat bertsio sortu dira: Fortran 66, 77, 90, 95, 2003 eta 2008. Hauetako bertsio bakoitzean funtzionalitate berriak eta C lengoaiarekin lan egiteko bateragarritasuna gehitu zaizkio. C lengoaia, 1970 hamarkadan jaio zen eta hardwarearekiko hurbiltasun "ezaugarri nagusiak, konpiladoreari kode eraginkorra sortzeko aukera ematen dio. C lengoaia *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 lengoaiiek, ez dituzte kodea paraleloan exekutatzeko tresnarik, hau da, ez dago konputazio banatu, eta prozesadore ezberdinen artean aldi berean exekuzioak zehazteko modurik. Konputazioa paraleloa implementatzeko, bi dira interfaze aplikazio programa (*API*) moduan implementatuta dauden sistema nagusiak [?]:

1. *MPI* (Message Passing Interface). Erabiliena da, memoria banatutako sistemetarako pentsatua baina memoria konpartitutako sistemetan ere aplika daitekeena.
2. *Open MP* (Open Specifications for MultiProcessing). Erabiltzeko errazagoa eta memoria konpartitutako sistemetan bakarrik aplika daitekeena.

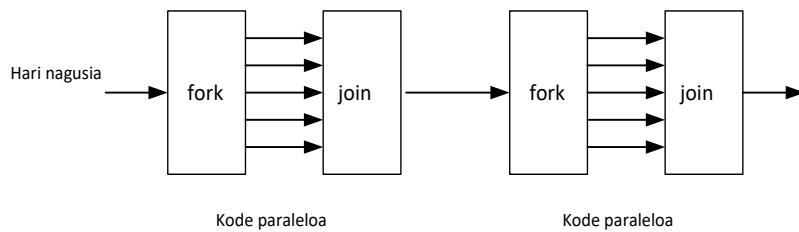
Eraginkortasun altuko konputazioaren (*high performance computing*) programazioa konplexua da: espezializazio handikoa eta konputagailuen hardware jakin baterako egokitutakoa. Zaitasun hauek, proiektu zientifikoak aurrera ateratzeko eta mantentzeko arazo asko eragiten ditu. Azken urteotan, eragozpen hau gainditzeko, programazio lengoia interesarriak sortu dira (adibidez, Julia [?] edo Chapel [?]) baina oraindik, hauen arrakasta ikustekoa da.

Azkenik, zientzia konputazioan *problemak ebazteko inguruneak* (Problem Solving Enviroments) 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 [?] 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, 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 (5.6. irudian) adierazi dugu.



5.6. Irudia: OpenMp programazio modeloa.

- 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 lengoian, *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 aldagaia guziak harien artean konpartituta daude eta aldagaia pribatuak direla zehazteko, esplizituki adierazi behar da. Goiko adibidea salbuespena da; *for* iterazioaren kontagailua (adibideko *i* aldagaia) pribatua da.

Algoritmo batean, kodearen zati bat da paralelizagarria. Suposa dezagun, 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, $(1 - 1/S)$ paralelizagarria konsideratzuz, 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.

5.5. 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 [?]. 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 funtzioko *Fortran* lengoaien 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=HERmitian ; HB=Hermitian Band; HP=Hermitian Packed.
 - (d) Matrize triangularrak.
TR=TRiangular ; TB=TRiangular Band; TP=Triangular Packed.

BLAS.

BLAS (www.netlib.org/blas) liburutegiak, 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. Fabrikatzale 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 implementatuta dago eta C lengoaiatik BLAS funtzioen erabilpena errazteko, *cblas* interfazea erabiltzea gomendagarria da.

Adibidea. BLAS liburutegiaren eraginkortasuna, *cblas_dgemm()* matrizeen biderkadura funtzioaren bidez aztertu dugu eta gure implementazio arrunta baino $10 \times$ azkarragoa dela baiezta dugu (Taula 5.1.).

LAPACK.

LAPACK 1992. urtean garatu zen [?] eta aljebra linealaren problemak ebazteko funtzioen liburutegia da. Jatorrizko bertsioa Fortran 77 lengoian implementatuta dago eta liburutegiaren dokumentazioa nahiz kodea Netlib software bilgunean (<http://www.netlib.org/lapack/>) eskuratu daiteke. Matrize dentsoetarako garatuta dago eta problema hauetarako errutinak biltzen ditu:

1. Ekuazio-sistema linealen ebazpena: $AX = b$.
2. Linear least square problems: $\|Ax - b\|$ minimizatzen duen x balioa bilatu.
3. Eigenvalues problems.
4. Balio singularren deskonposaketa (SVD).

5.1. 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 bakoitzak 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 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:

1. Problema osoa ebazten dituzten errutinak (drivers). Talde honetan funtzio arruntak eta funtzio 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 matrizaren LU faktorizazioa kalkulatzen du, $A = P * L * U$.

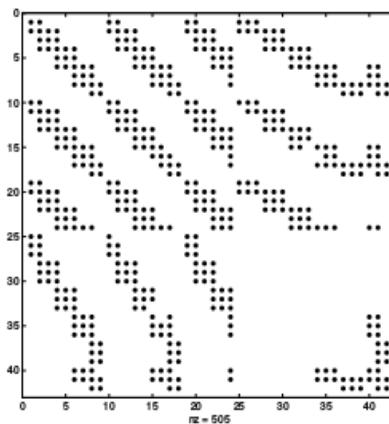
LAPACKE-dgetrs (LAPACK-ROW-MAJOR,trans,n,nrhs,A,lda,ipiv,B,ldb); Errutina honek $A * X = B$ ekuazio sistemaren X soluzioa kalkulatzen du.

3. Errutina laguntzaileak.

Matrize bakanak.

$A \in \mathbb{R}^{m \times n}$ matrizeari bakanera esaten zaio, baldin abantaila atera daitekeen zero osagai kopuru adina baditu. Honek esan nahi du, matrizearen zero ez diren osagaien kopurua n_{nz} ,

$$n_{nz} \ll mn.$$



5.7. Irudia: Matrize bakanak.

Matrizearen bakantasuna, konputazioaren memoria eta exekuzio denbora guitzitzeo erabil daiteke.

1. Doitasun bikoitzeko $A \in \mathbb{R}^{m \times n}$ matrizea,
 - (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.

5.6. Konpiladorea.

Konpiladorearen zeregina konplexua da, goi mailan idatzitako programari dago-kion makina kodea sortzea (konputagailuaren errekurtoak modu eraginkorrean erabiltzen dituenak). Konpiladoreak heuristikotan oinarritutako kode aldaketak eragiten ditu, eraginkortasuna hobetzeko asmoarekin. Horregatik, programatzai-leak 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: adibidez intel-en *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.

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 experimentuetarako 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*-a, lengoia berezi bat erabiliz, konpilazio prozesua automatizatzeko programa moduko bat dugu. [B.2.](#) eranskinean, *Makefile* lengoaiaren oinarrizko adibideak eman ditugu.

5.7. 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.
4. Kodearen exekuzio denboraren analisia egiteko tresnak (adibidez gprof) daude. Algoritmoaren funtzio bakoitzaren exekuzio denborari buruzko informazio erabilgarria lortuko dugu.

Atal honi dagokion gomendatutako bibliografia: Victor Eijkhout [?], Greg Wilson [?], Leslie Hogben [?]

II. Atala

Ekarpenak.

6. Kapitulua

IRK: Puntu-finkoaren iterazioa.

6.1. Sarrera.

Sistema Hamiltondar ez-zurrunen doitasun altuko zenbakizko integraciolarako, puntu-finkoaren iterazioan oinarritutako IRK metodoaren implementazioa garatu dugu. Konputazioetarako 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 implizitu eta simplektikoaren (Gauss nodoetan oinarritutako RK kolokazio metoda) implementazioa proposatu dugu, zeinetan biribiltze errorearen garapena txikia izateko ahalegin berezia egingo dugun. Implementazioa, problema ez-zurrunetan aplikatzeko garatu dugunez, ekuazio-sistema implizitua, puntu-finkoaren iterazioaren bidez ebatziko dugu (puntu-finkoaren iterazioan eta Newtonen iterazio simplifikatu oinarritutako implementazioen eraginkortasun azterketak lan honetan [?] kontsultatu daiteke).

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.

Integracioaren exekuzio denborak onargarriak izan daitezen, 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-egun zientzia-konputazioa, 64-biteko koma-higikorreko aritmetikan (*double* datu-mota) oinarritzen da eta beraz, erabiltzaileak ekuazio diferentziala, datu-mota honetan zehaztuko duela suposatuko dugu.

Lehenengo, Hairer-en IRK metodoaren implementazioa [?] aztertuko dugu eta

implementazio honetan aurkitu ditugun hainbat arazo azalduko ditugu. Ondoren, IRK implementazio hau hobetzeko gure proposamenak emango ditugu eta azkenik, zenbakizko integrazioen bidez, bere abantailak erakutsiko ditugu.

6.2. Hairer-en implementazioa.

IRK implementazio estandarra.

Gure abiapuntua, Hairer-ek [?] proposatutako IRK metodoaren implementazio da. Lan honetan, puntu-finkoaren iterazioan oinarritutako IRK metodo simpletikoen 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 bidera:

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

$$\Delta^{[k]} = \max_{i=1,\dots,s} \|Y_i^{[k]} - Y_i^{[k-1]}\|_\infty \leq \delta \quad (6.1)$$

(non δ finkatutako tolerantzia den), urrats bakoitzean aplikatutako puntu-finkoaren iterazioak, errore sistematikoa eragiten du.

Konponbideak.

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

1. Metodoaren koefizienteen doitasuna handitzea, koeficiente bakoitza komahigikorreko bi koefizienteen batura konsideratuz,

$$a_{ij} = \hat{a}_{ij} + \tilde{a}_{ij}, \quad b_i = \hat{b}_i + \tilde{b}_i \quad (6.2)$$

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,

$$\Delta^{[k]} = 0 \text{ edo } \Delta^{[k]} \geq \Delta^{[k-1]}. \quad (6.3)$$

Hairer-en implementazioaren algoritmoa.

Hairer-ek bere *Fortran* implementazioa eskuragarri du ([Fortran kodea](#)). Jarraian, bere implementazioaren [17](#) algoritmoa eta erabilitako batura konpensatuaren teknika ([18](#) 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
     $(y_{n+1}, e_{n+1}) \leftarrow BaturaKonpensatua(y_n, e_n, F_n^{[k]});$ 
end

```

Algoritmoa 17: Hairer (IRK)

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 18: Hairer (Batura konpensatua)**Hairer-en implementazioaren arazoak.**

Hairer-ek bere implementazio berria, *Hénon-Helies* eta eguzki-sistemaren kanpo-planeten problemetarako energiaren errore sistematikorik ez zegoela baieztatu

zuen. Energia errorea, $k\sqrt{t_n}$ espresioaren arabera handitzen dela erakutsi zuen eta implementazio berriak *Brouwer legea* [?] betetzen duela ondorioztatu zuen. Integrazioen azterketa estatistikoa egin zuen biribiltze errorearen ausazkotasuna baiezatzeko. Problemaren hasierako balio zehatz bati dagokion perturbatutako $P = 1.000$ integracio exekutatu zituen eta integracio horien energia errorearen batezbestekoa (μ), zero eta desbideratze estandarra (σ), $k\sqrt{t_n}$ espresioaren araberakoa zela erakutsi zuen. Era berean, integracio amaiera uneko energi erroreen histogramak, $N(\mu, \sigma)$ distribuzio normala betetzen duela erakutsi zuen.

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

6.3. 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 [?] arazoak gainditzeko, geratze irizpide sendoagoa eta normaren independentea garatu dugu.

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

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

Kontsideratu dezagun, honako hasierako baliodun problema,

$$\dot{y} = f(t, y), \quad y(t_0) = y_0, \quad (6.4)$$

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, (6.4) problemaren $y(t)$ soluzioaren $y_n \approx y(t_n)$ ($n = 1, 2, \dots$) zenbakizko soluzioa, integracio metodo bat aplikatuz

$$y_{n+1} = \Phi(y_n, t_n, t_{n+1} - t_n), \quad (6.5)$$

lortuko dugu, 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,

$$\Phi(y, t, h) = y + h \sum_{i=1}^s b_i f(t + c_i h, Y_i) , \quad (6.6)$$

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

$$Y_i = y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j) \quad i = 1, \dots, s. \quad (6.7)$$

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

,

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (6.8)$$

6.3.1. Metodoaren birformulazioa.

Metodoa definitzen duten $a_{ij}, b_i \in \mathbb{R}$ koefizienteak, biribildutako koefizienteen hurbilpenez $\tilde{a}_{ij} := fl(a_{ij}), b_i := fl(b_i) \in \mathbb{F}$ ordezkatzen ditugu eta hauek, ez dute simplektikoa izateko baldintza (6.8) betetzen. Hori dela eta, metodoak ez ditu integral koadratikoak kontserbatuko eta Hamiltondar funtzioaren eboluzioan, drift lineala ageriko da [?].

Arazo hau gainditzeko, IRK metodoa era honetan birformulatuko dugu,

$$Y_{n,i} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}, \quad L_{n,i} = h b_i f(Y_{n,i}), \quad i = 1, \dots, s, \quad (6.9)$$

$$y_{n+1} = y_n + \sum_{i=1}^s L_{n,i}, \quad (6.10)$$

non

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

Eta Runge-Kutta metodoa simplektikoa izateko baldintza (6.8), modu honetan berridatziko dugu,

$$\mu_{ij} + \mu_{ji} - 1 = 0, \quad 1 \leq i, j \leq s. \quad (6.11)$$

Birformulazio honek formulazio estandarrarekiko duen abantaila, simplektizitatea baldintzaren espresioan biderketarik ez agertzeak, baldintza zehazki beteko duten $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak aurkitzeko bidea errazten duela da. Jarraian, koefizienteak finkatzeko urratsak azalduko ditugu:

1. μ_{ij} koefizienteak.

S-ataleko Gauss kolokazio metodoen koefizienteen matrize diagonaleko balioak finkoak dira ($\tilde{\mu}_{ii} := 1/2$, $i = 1, \dots, s$) eta balio honek, koma-higikorreko adierazpen zehatza du.

Gainontzeko koefizienteak, bi urratsetan kalkulatuko ditugu:

- Lehenengo, koefiziente matrizearen behe-diagonaleko balioak finkatuko ditugu,

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

- Bigarrenik, 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 (6.11) zehazki betetzen duten koma-higikorreko $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak,

$$\tilde{\mu} = \begin{pmatrix} 1/2 & 1 - fl(\mu_{21}) & \dots & 1 - fl(\mu_{s1}) \\ fl(\mu_{21}) & 1/2 & \dots & 1 - fl(\mu_{s2}) \\ \vdots & \vdots & \ddots & \vdots \\ fl(\mu_{s1}) & fl(\mu_{s2}) & \dots & 1/2 \end{pmatrix} \in \mathbb{R}^{s \times s}. \quad (6.12)$$

2. 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,

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

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

3. ν_{ij} interpolazio koefizienteak.

Formulazio estandarraren λ_{ij} koefizienteetatik abiatuta (2.3.3. atala), formulazio berriari dagozkion interpolazio ν_{ij} koefizienteak era honetan definituko ditugu,

$$Y_{n,i}^{[0]} = y_n + h \sum_{j=1}^s \nu_{ij} L_{n-1,j}, \quad \nu_{ij} = \lambda_{ij} / b_j \quad 1 \leq i, j \leq s. \quad (6.13)$$

6.3.2. Geratze irizpidea.

Ekuazio implizituaren (6.9) soluzioaren hurbilpena lortzeko puntu-finkoaren iterazioa era honetan definituko 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
     $L_{n,i}^{[k]} = hb_if(Y_{n,i}^{[k-1]}), i = 1, \dots, s;$ 
     $Y_{n,i}^{[k]} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]}, i = 1, \dots, s;$ 
end
```

Algoritmoa 19: IRK puntu-finkoaren iterazioa.

Iterazioa, $Y_{n,i}^{[0]} = y_n$ balioarekin edo aurreko urratsetako atalen balioen interpolazioz lortutako balioekin [?] hasieratu daiteke. Urrats luzera h adina txiki aukeratuz gero, iterazioek ekuazio aljebraikoen (6.9) soluzioa den puntu-finkora konbergituko dute. Gure zenbakizko esperimentuetan nahiz Hairer-ek eginda esperimentuetan ere [?], h urrats luzera txikiarekin integrazioaren urrats gehienetan, puntu-finkoa lortzen dela baiezta dugu.

IRK metodoaren implementazio estandarraren geratze irizpidea honakoa da,

$$\begin{aligned} \Delta^{[k]} &= (Y_{n,1}^{[k]} - Y_{n,1}^{[k-1]}, \dots, Y_{n,s}^{[k]} - Y_{n,s}^{[k-1]}) \in \mathbb{F}^{sd}, \\ \|\Delta^{[k]}\| &\leq tol \end{aligned} \quad (6.14)$$

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. Gogorari behar da ere, Hairer-ek [?] iterazio errorea modu sistematikoan metatzeko dela 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 luzearekin egindako integrazioan ondo funtzionatzen du, baina $h = 1000/3$ urrats luzerarekin integratzerakoan, tamaina handiko energia errore agertzen da. Energia errore erlatiboaren eboluzioa irudikatu dugu (6.1.(a) Irudia). Integrazioaren

lehen urratsaren iterazioak aztertzen badugu,

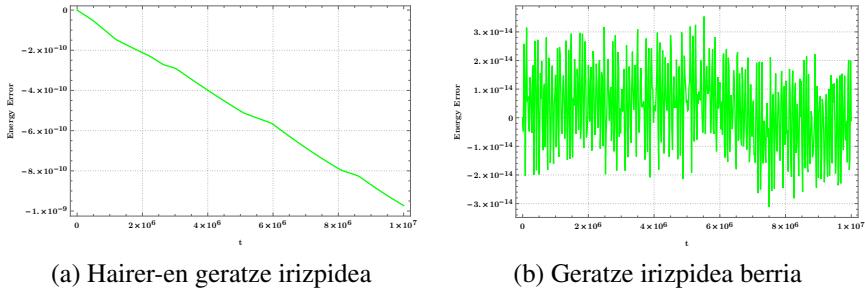
$$\|\Delta^{[1]}\| > \|\Delta^{[2]}\| \dots > \|\Delta^{[12]}\| = 3.91 \times 10^{-14} \leq \|\Delta^{[13]}\| = 4.35 \times 10^{-14}$$

13.iterazioan geratuko dela konprobatuko dugu. Puntu-finkoaren iterazioa goizagi 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,

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

Batetik, $\{\Delta_j^{[0]}, \Delta_j^{[1]}, \dots, \Delta_j^{[k]}\}$ segida beherakorra denik, ezin daiteke suposatu. 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 litzateke.



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

$$\forall j \in \{1, \dots, sd\}, \quad \min \left(\{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} / \{0\} \right) \leq |\Delta_j^{[k]}|. \quad (6.15)$$

Hairer-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 (6.1.(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,

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

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. Puntu-finkoaren iterazioa, konbergentzia lortu duelako edo konbergentzia arazoak izan dituelako gera daiteke.

6.3.3. 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 (11) eta zenbakizko integrazioetan erabili ohi da. Atal honeitan, batetik IRK metodoetan batura konpensatuaren aplikazio estandarra hobetzeko proposamena azalduko dugu. Beste aldetik, IRK metodoaren gure implementazioan biribiltze errorearen beste jatorri nagusiak (biderketa eta batuketa bat) modu finagoan kalkulatzea proposatuko dugu.

Batura konpensatua.

Zenbakizko soluzioa, $y_n \approx y(t + hn) \in \mathbb{R}^d$, $n = 1, 2, \dots$, 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 $e_0 = fl(y_0 - \tilde{y}_0)$ diren.

Hori dela eta, IRK metodoaren inplizituki $Y_{n,i}$ atalak askatzeko ekuazioetan, \tilde{y}_n balioa erabili ordez, $(\tilde{y}_n \oplus e_n)$ espresioa erabiltzea proposatuko dugu,

$$L_{n,i}^{[k]} = hb_i f(Y_{n,i}^{[k-1]}), \quad Y_{n,i}^{[k]} = \tilde{y}_n \oplus (e_n \oplus \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]}). \quad (6.16)$$

Aldaketa honekin, lortutako zenbakizko soluzioaren doitasuna, batura konpensatu 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 oinarritutako teknika (13 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,

$$(\tilde{y}_{n+1}, e_{n+1}) = S_{s,d}(\tilde{y}_n, \delta_n, L_{n,1}^{[k]}, \dots, L_{n,s}^{[k]}). \quad (6.17)$$

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 20: BaturaKonpensatua

6.3.4. Biribiltze errorearen estimazioa.

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

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

```
Function flr(x,r)
  res = (2^r * x + x) - 2^r * x
  return res
```

Algoritmoa 21: flr

Funtzio 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 funtzioa da.

Bigarren zenbakizko soluzio $(\hat{y}_n + \hat{e}_n)$, r ($r < m$) balio bat finkatuta eta implementazioaren kalkulua (6.17) beste era honetan egingo 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 nagusiaren $(y_n + e_n)$ eta r balio txiki baterako (esaterako $r = 3$) kalkulatutako bigarren zenbakizko soluzioaren $(\hat{y}_n + \hat{e}_n)$ arteko differentziaren norma bezala kalkulatuko dugu.

$$\text{estimazio}_n^i = \| (y_n^i + e_n^i) - (\hat{y}_n^i + \hat{e}_n^i) \|_2, \quad i = 1, \dots, d. \quad (6.18)$$

Gure algoritmoan estimazioa lortzeko, bi integrazioak sekuentzialki modu era-ginkorrean kalkulatuko ditugu. Urrats bakoitzean, bi integrazioen Y_i, \hat{Y}_i ($i = 1, \dots, s$) ataletako balioak, biribiltze errorearen estimazioa handiegia ez den artean, antzekoak mantentzen dira. Ondorioz, lehen integrazioaren bukaerako $Y_i^{[k]}$ ($i = 1, \dots, s$) atalen balioak, bigarren integrazioaren $\hat{Y}_i^{[0]}$ ($i = 1, \dots, s$) atalen hasieraketarako erabiltzen baditugu, bigarren integrazioaren iterazio kopuru txikia

izango da (ikus [22](#) 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 22: RKG2: errore estimazioa

non $G()$ interpolazio funtzioa eta $initwithfirst$ egiazko balioa duen aldagaia den, integrazioen arteko differentzia txikian den artean.

6.3.5. Algoritmoa.

Jarraian, formulazio berriari dagokion algoritmoa laburtuko dugu ([23](#) 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 23: IRK (puntu-finkoa).

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, zenbakizko soluzioa fitxategi bitar batean idatziko dugu.

Bi integrazio mota exekutatu daitezke:

1. Integrazio arrunta.

Integrazioaren zenbakizko soluzioa (y_i, e_i) fitxategi batean itzultzen dugu eta lerro bakoitzaren egitura honakoa 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,

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

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

6.4. 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.

6.4.1. Problemak.

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

Pendulu bikoitz arrunta

Pendulu bikoitz arruntaren Hamiltondarra eta parametroak, [3.2.1.](#) atalean definitu ditugu. Sistema Hamiltondar honetarako, hasierako bi balio konsideratu ditugu: hurrenez hurren, izaera ez-kaotikoa (NCDP) eta kaotikoa (CDP) duten mugimenduak eragiten dituztenak. Hasierako baliodun bi problema hauek (NCDP eta CDP), energia errorearen eboluzioaren eta biribiltze errorearen estimazioaren azterketa egiteko konsideratuko ditugu. Energia errorearen jatorria aztertzeko integracio luzerako ordea, problema ez-kaotiko (NCDP) bakarrik konsideratu dugu.

Kanpo-planeten problema

Eguzki-sistemaren kanpo-planeten eredu Newtoniarra, [3.4.2.](#) atalean azaldu dugu. Hamiltondar sistema banagarria da,

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

eta ezaguna da, puntu-finkoaren iterazio estandarraren eraginkortasuna, puntu-finkoaren bertsio partizionatuarekin ([4](#)) hobetu daitekeela [?]. Dena den, zenbakizko esperimentuetarako, Hairer-en [?] lanean bezala, puntu-finkoaren bertsio estandarraren emaitzak erakutsi ditugu. Puntu-finkoaren bertsio partizionatua aplikatu dugunean, antzeko emaitzak lortu ditugu baina urrats bakoitzean iterazio kopuru gutxiagorekin.

6.4.2. Energia errorearen jatorria.

Doitasun bikoitzeko 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 $y(t_n)$ ($n = 1, 2, 3, \dots$), (6.9)-(6.10) metodoa aplikatuz (b_i, μ_{ij} koefiziente zehatzekin) lortutako zenbakizko soluzioa y_n ordezkatzerakoan eragindako errorea.
2. Iterazio errorea: praktikan, puntu-finkoaren (19) K iterazio finitu aplikatzen da, eta (6.9) 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 funtziaren 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).

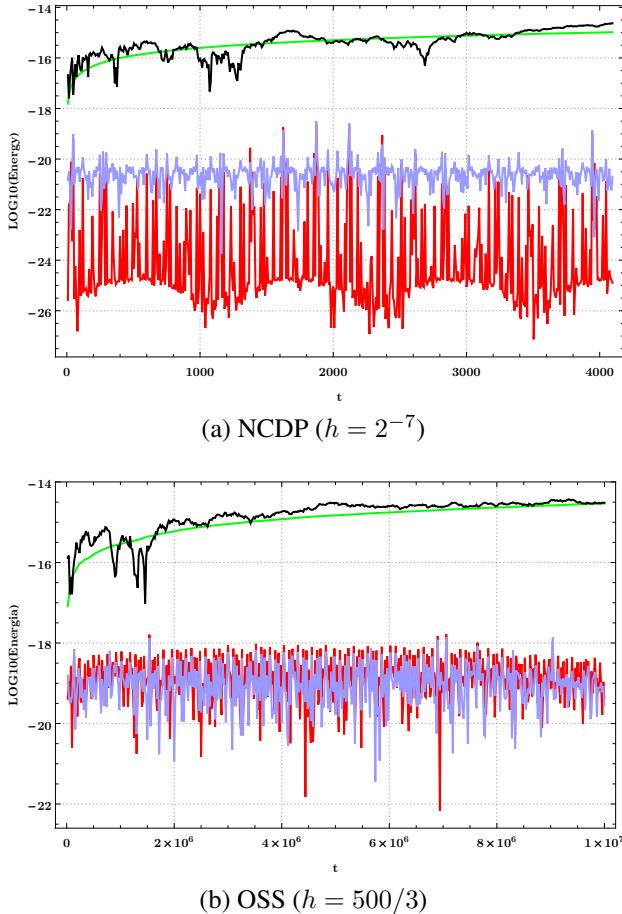
Hasierako baliodun problema bakoitzaren, goian azaldutako A–D implementazio bakoitzari dagokion energia errorearen eboluzioa erakutsi dugu (6.2. irudia). Aukeratutako h urrats luzerarentzat, trunkatze errorea biribiltze errorearen azpitik dagoela baieztago dugu. Metodoaren doitasun bikoitzeko koefizienteak ($\tilde{b}_i, \tilde{\mu}_{ij} \in \mathbb{F}$) erabiltzeak, ez du eraginik biribiltze errorearen garapenean. Iterazio errorea, biribiltze errorearen oso antzoko da, eta energia errorearen drift lineala eragitea espero daiteke.

6.4.3. Errore azterketa estatistikoa.

Biribiltze erroreak eragiten duen zenbakizko erroreen azterketa fidagarriagoa egiteko, analisi estatistikoa aplikatu dugu (Hairer-en [?] 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 [?]. Zenbakizko esperimentuera rako, Hairer-en IRK metodoaren Fortran kodea exekutatu dugu ([Kodea](#)).



6.2. 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).

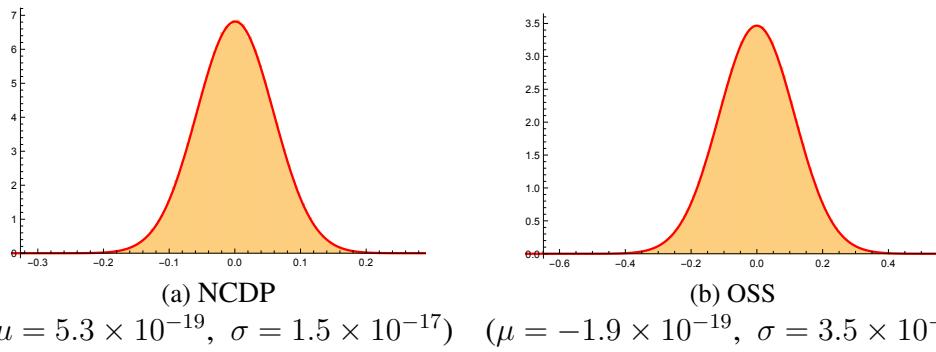
Batetik, DP implementazioaren biribiltze errorearen garapena, FPIEA implementazioaren (aritmetika zehatza) errorearekiko kualitatiboki antzekoa den eta magnitudean gertu dagoen ziurtatu nahi dugu. Bestalde, DP implementazioa, Haireren implementazioarekin konparatu nahi dugu.

Aipatutako hiru implementazioetarako, puntu-finkoa lortu den urratsen por-tzentaia eta iterazio batezbestekoa laburtu ditugu (6.1. taula).

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

	FPIEA		DP		Hairer	
	%	#	%	#	%	#
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 differentzieng banaketa.



6.3. Irudia: DP implementazioarekin lortutako KP energia differentzieng 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 implementazioa ona bada, biribiltze erroreak eragindako energiareng 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 [?] Brouwer legea deritzote, Brouwer-ek [?] Kepler problemarentzat egin zuen zenbakizko integrazioaren birbiltze errorearen azterketa gogoratzu.

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 tartea $[t_0, t_{end}]$ eta P perturbatutako hasierako balioen kopurua bada, KP energia differentziaren balio ditugu, non $K = (t_{end} - t_0)/(mh)$ den. DP implementazioarekin lortutako KP energia differentziaren histograma eta $N(\mu, \sigma)$ banaketa normala irudikatu ditugu (non μ eta σ , balioei dagokien batezbestekoa eta desbideratze tipikoak diren). [6.3.](#) 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.

Errore batezbestekoaren eta desbideratze estandarraren eboluzioa

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

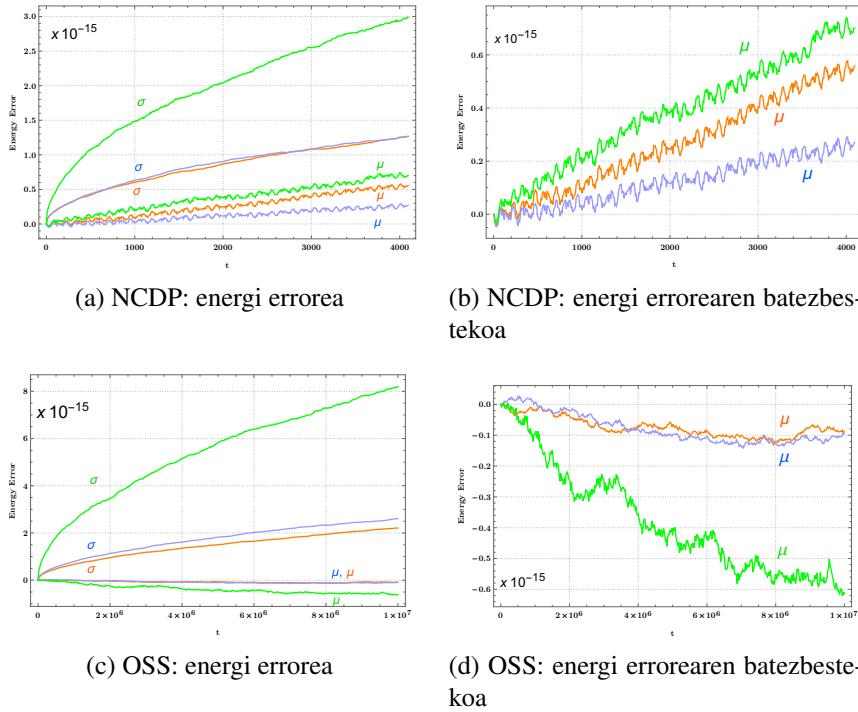
FPIEA, puntu-finkoaren iterazioan oinarritutako IRK implementazio optimoen kontsideratu daiteke, f funtzioari dagokion doitasun bikoitzeko \tilde{f} funtzioa aplikatuko dugula suposatzen badugu. Argitu beharra dago, esperimentu hauetan FPIEA implementazioaren geratze irizpidea DP implementazioarena baino gogorragoa erabili dugula: iterazioa geratu dugu $\Delta^{[k]} = 0$ delako edo [\(6.15\)](#) baldintza hamar iterazio jarraian bete delako. Era honetan, puntu-finkoa lortu ez den urratsetarako, iterazio errorea ekiditen saiatu gara.

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

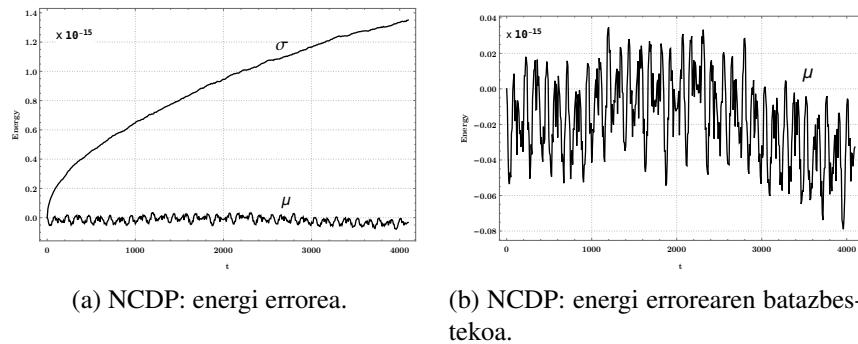
Puntu-finkoaren iterazioan oinarritutako IRK metodoen implementazioan, energi errorearen batezbestekoan drift lineal txiki bat (NCDP problemarentzat) ekidin ezinezkoa dela sinistuta gaude. Esperimentu hauen energi errorearen eboluzioak, [6.2.](#) irudian erakutsitako iterazio errorea, birbiltze errorearen gertu egoteagatik egindako iruzkinarekin kontsistenteari dira.

Energia drift-a, ez da IRK metodo simplektikoen berezko arazoa. [6.5.](#) 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, [6.6.](#) irudian, FPIEA, DP eta Hairer-en implementazioen integrazioen kokapen errorearen eboluzioa (batezbestekoa eta desbideratze estandarra) erakutsi ditugu. Emaitza hauek, DP implementazioa,

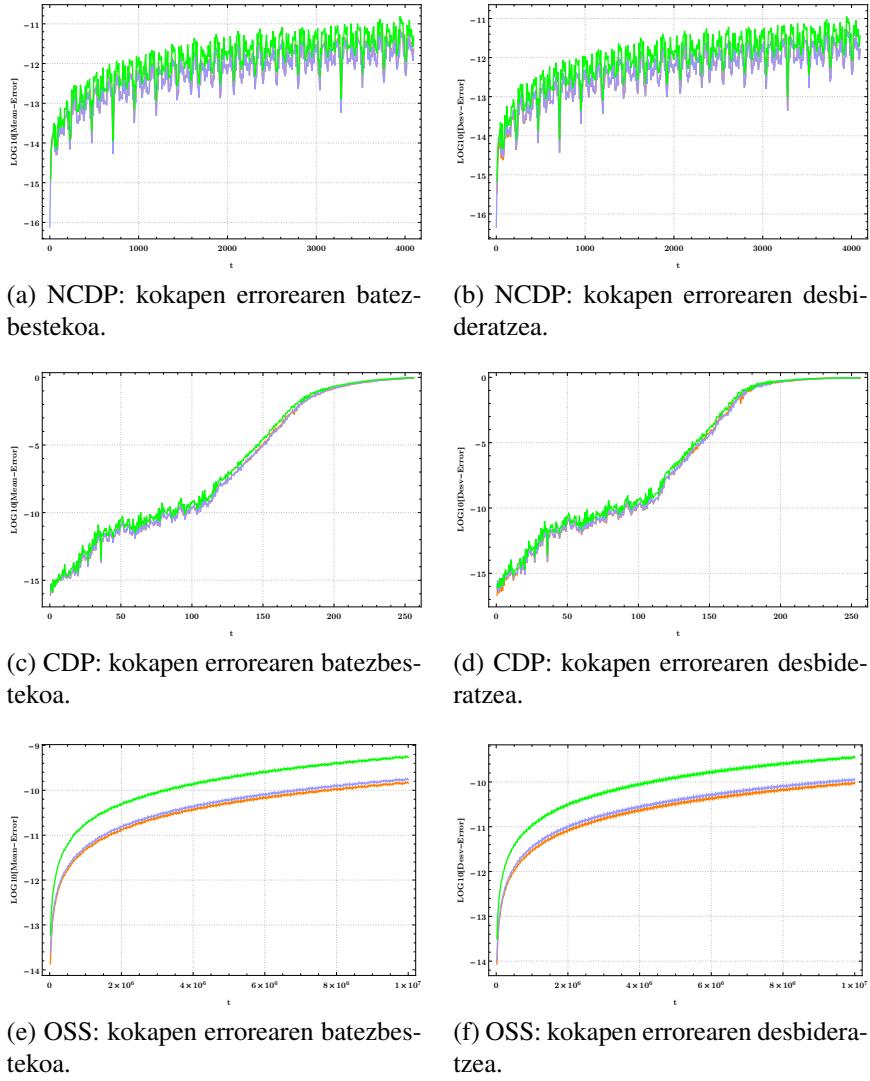


6.4. Irudia: Energi errorearen batezbestekoa (μ) and desbideratze estandarra (σ) (ezkerrean) eta energi errorearen batezbestekoaren zehaztapena (eskubian), DP implementazioarentzat (urdinez), FPIEA implementazioarentzat (laranjaz), eta Hairer-en implementazioarentzat (berdez). Pendulu bikoitzaren problema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



6.5. Irudia: Energi errorearen batezbestekoa (μ) eta desbideratze estandarra (σ), Newton sinplifikatuaren iterazioan oinarritutako IRK implementazioarekin

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



6.6. 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)

6.4.4. Biribiltze errorearen estimazioa

6.3.4. atalean deskribatutako biribiltze errorearen estimazioa kalkulatzeko teknika aplikatu dugu. **6.7.** irudian, hiru problemetarako (perturbatu gabeko hasierako balioa erabiliz) DP implementazioaren integrazioaren kokapen errorea eta kokapen errorearen estimazioa ($r = 3$ aplikatuta) 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.

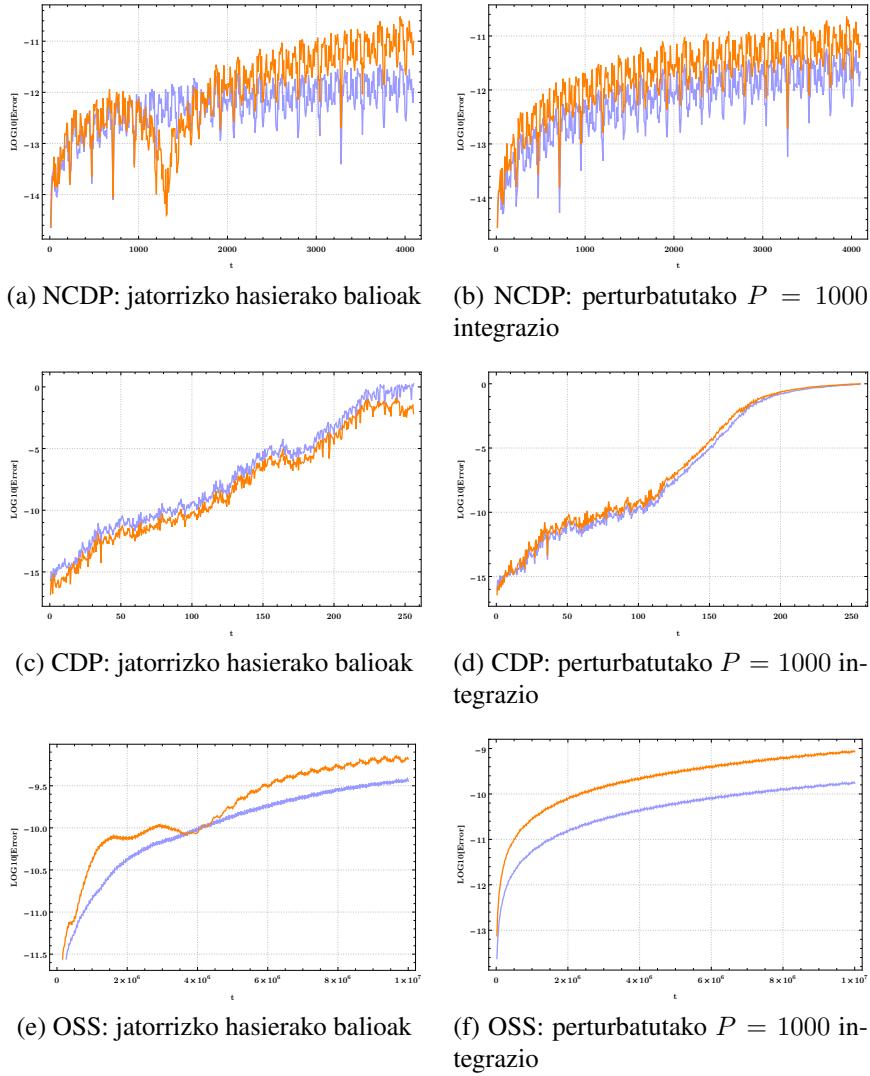
6.5. Laburpena.

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

Proposatutako implementazioan, biribiltze errorearen eragina txikitzeko ahalgina 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.



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

7. Kapitulua

IRK: Newtonen Iterazioa.

7.1. Sarrera.

Runge-Kutta metodo implizituen implementazioetan, ekuazio sistema ez-lineala modu eraginkorrean askatu behar da. Atal honetan, Newton iterazio metodoa modu eraginkorrean aplikatzeko bidea ikertuko dugu. Problema zurruna denean, puntu finkoaren iterazio ez da eraginkorra eta Newton iterazioa aplikatu behar da. Gainera problema ez-zurruna izanik ere, Newton iterazioak interesgarriak izan daitezke; bereziki doitasun altuko (doitasun laukoitzta) konputazioetan iterazio metodoaren konbergentzia ezaugarri onak direla eta.

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

$$(I_d \otimes I_s - h A \otimes J) \in \mathbb{R}^{sd \times sd}, \quad (7.1)$$

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

Lan hauetan [?] [?] [?], (7.1) matrizearen egitura bereziari abantaila atereaz, ekuazio sistema linealak modu eraginkorrean ebazteko proposamena egin zuten. Zehazki, ia blokeka diagonala den matrizea da, s -bloke $I_d - h\lambda_j J \in \mathbb{R}^{d \times d}$ ($j = 1, \dots, s$), A matrizearen λ_j balio propio bakoitzari bloke bat dagokio. Normalean, ordena altuko IRK metodoaren A matrize koefizienteak, $[s/2]$ balio propio konplexu pareak (s bakoitia den kasuan balio propio erreal bat gehituta) ditu.

Gure ekarpenean, sd -dimentsioko ekuazio sistema (7.1) ebazteko, $(s+1)d$ dimentsioko sistema baliokidean berridatziko dugu, eta $d \times d$ tamainako $[s/2] + 1$ matrizeen LU deskonposaketa (eta tamaina bereko matrize batzuen biderketa) kalkulatuz, askatuko dugu. Tamaina txikiko matrizeen LU deskonposaketa azkarra

denez, konputazionalki eraginkorra izatea espero dugu. Algoritmoa, IRK metodoa simetriko eta simplektikoetarako (bi propietateak betetzen dituzten metodoeetarako) garatu dugu. Dena den, bai IRK metodo ez-simetriko simplektikoetan, bai IRK metodo simetriko ez-simplektikoetan aplika daiteke.

Newton iterazio metodoa, $u \in \mathbb{R}^n$ 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, Newton metodoa era honetan definituko dugu (algoritmoa 24).

```

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 24: Newton sinplifikatua.

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

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

Azpimarratzeko da ere, Newton metodoaren eragiketa konplexuenak doitasun txikiagoan kalkula daitezkeela [?] eta honek, konputazionalki abantaila interesgarria suposatzen duela. Algoritmoaren (Algoritmoa 24) 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.

7.2. IRK-Newton estandarra.

Demagun honako hasierako baliodun problema,

$$\dot{y} = f(t, y), \quad y(t_0) = y_0, \tag{7.2}$$

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, (7.2) hasierako baliodun problemaren $y(t)$ soluzioaren $y_n \approx y(t_n)$, ($n = 1, 2, \dots$) zenbakizko soluzioa, integrazio metodo bat aplikatuz

$$y_{n+1} = \Phi(y_n, t_n, t_{n+1} - t_n), \tag{7.3}$$

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,

$$\Phi(y, t, h) = y + h \sum_{i=1}^s b_i f(t + c_i h, Y_i) , \quad (7.4)$$

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

$$Y_i = y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j) \quad i = 1, \dots, s. \quad (7.5)$$

$Y_i \in \mathbb{R}^d$, $i = 1, \dots, s$ ezezagunen eta sd tamainako ekuazio sistema ez-lineala (7.5) askatzeko, iterazio metodo bat aplikatu behar dugu. Aurreko atalean puntu-finkoaren iterazioa aztertu genuen eta atal honetan, Newton iterazio metoda modu eraginkorrean aplikatzeko bidea ikertuko dugu.

Newton iterazioa.

Newton $k = 1, 2, \dots$ iterazioetarako, (7.5) ekuazio implizituko Y_i ($i = 1, \dots, s$) atalen $Y_i^{[k]}$ hurbilpenak kalkulatzeko algoritmoa, modu honetan definituko dugu,

$$1) \quad r_i^{[k]} := -Y_i^{[k-1]} + y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j^{[k-1]}), \quad i = 1, \dots, s, \quad (7.6)$$

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

$$\Delta Y_i^{[k]} - h \sum_{j=1}^s a_{ij} J_j^{[k]} \Delta Y_j^{[k]} = r_i^{[k]} \quad i = 1, \dots, s, \quad (7.7)$$

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

$$3) \quad Y_i^{[k]} := Y_i^{[k-1]} + \Delta Y_i^{[k]}, \quad i = 1, \dots, s, \quad (7.8)$$

Azpimarratu behar da iterazio bakotzeko, $J_i^{[k]}$ Jacobiaren s ebaluazio eta $sd \times sd$ tamainako matrizearen LU-deskonposaketa kalkulatu behar ditugula. Era-giketa hauek konplexuak dira eta horregatik, Newton osoaren implementazioa konputazionalki oso garestia da. Aukera eraginkorragoen artean, bi aipatuko ditugu:

1. Newton interazio sinplifikatuak aplikatzea. Aukera honetan, (7.7) ekuazioaren $J_i^{[k]}$ Jacobiar matrizeak, $J_i^{[0]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[0]})$ matrizeekin ordezkatuko dira. Era honetan, urratsero behin bakarrik egin beharko litzateke LU deskonposaketa.

2. (7.7) ekuazio sistema, aurrebaldintzatutako matrize honen

$$(I_s \otimes I_d - h A \otimes J) \quad (7.9)$$

alderantzizkoarekin iterazio metodoaren [?] bidez ebatzea.

Dena dela, goiko bi aukeretan era honetako ekuazio sistemak askatu behar dira,

$$(I_d \otimes I_d - h A \otimes J) \Delta Y = r \quad (7.10)$$

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 estandarrenean [?] [?] [?], A matrizearen diagonalizatzen 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. A matrizearen balio propio erreals bakoitzari dagokion $d \times d$ matrize errealen LU deskonposaketak kalkulatu behar dira.

Beste autore batzuk [?] [?], (7.9) ekuazio sistema askatzeko, matrize alderantzizko honen,

$$I_d \otimes I_s - h \bar{A} \otimes J, \quad (7.11)$$

non $\bar{A} \in \mathbb{R}^{s \times s}$ (LU deskonposaketa modu eraginkorragoan askatzeko aukeratua) aurrebaldintzatutako iterazioa aplikatzea proposatzen dute.

Newton simplifikatuaren iterazioa.

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

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

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

Lehen simplifikazio honetan, integrazioaren urrats bakoitzeko, $J_i^{[0]}$ Jacobiarren s-ebaluazio eta $sd \times sd$ tamainako matrizearen LU deskonposaketa behin

bakarrik kalkulatu behar ditugu. Modu baliokidean, ekuazio lineala notazio matriza 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,

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

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

Newton super-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 bakarrekin ordezkatzuz. Era honetako ekuazio sistema lortuko dugu,

$$(I_s \otimes I_d - h A \otimes J) \Delta Y^{[k]} = r^{[k]}. \quad (7.12)$$

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

Problema zurruna denean, atalen hasieraketa $Y_i^{[0]} = y_n$, $i = 1, \dots, s$ erabili ohi da, eta Jacobiarren hurbilpen honekin $J = \partial f / \partial y (t + h/2, y_n)$, ekuazio lineala askatzeak zentzua izango du. Aukera egokia da ere [?], $J = \frac{\partial f}{\partial y}(t + \bar{c}h, \bar{y})$ aplikatzea, non $\bar{c} = \frac{1}{s} \sum_{i=1}^s c_i$ (metodo simetrikoetan $\bar{c} = \frac{1}{2}$ da) eta $\bar{y} = \frac{1}{s} \sum_{i=1}^s Y_i^{[0]}$. Maiz, $\partial f / \partial y$ konputazionalki merkeagoa den hurbilketa batez ordezkatzea nahikoa izango da.

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

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

Jarraian, Newton super-simplifikatuaren implementazioaren algoritmo orokorra laburtu dugu (Algoritmoa 25).

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 25: IRK (Newton super-simplifikua).

7.3. IRK-Newton eraginkorra.

7.3.1. Ekuazio-sistema.

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

$$(I_s \otimes I_d - h A \otimes J) \Delta Y = r, \quad (7.13)$$

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 ODE sistemari aplikatzeko, urrats bakoitzean *sd* × *sd* tamainako hainbat ekuazio sis-

tema (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 *RK* kolokazio metodoak, simplektikoak eta simetrikoak [?] dira.

1. Simplektikoa.

Runge-Kutta metodoa simplektikoa izateko baldintza,

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (7.14)$$

2. Simetrikoa.

Runge-kutta metodoa simetrikoa izateko baldintza,

$$\begin{aligned} b_{s+1-i} &= b_i, \quad c_{s+1-i} = 1 - c_i, \quad 1 \leq i, j \leq s, \\ b_j &= a_{s+1-i, s+1-j} + a_{i,j}, \quad 1 \leq i, j \leq s. \end{aligned} \quad (7.15)$$

Bi propietate hauetan oinarrituko gara implementazio eraginkorra garatzeko.

S-ataletako IRK metodoa (7.4), baliokidean den modu honetan berridatzi daiteke,

$$\Phi(y, t, h) := y + z, \quad (7.16)$$

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

$$Y_i = y + \frac{z}{2} + h \sum_{j=1}^s \bar{a}_{ij} f(t + c_j h, Y_j) \quad i = 1, \dots, s, \quad (7.17)$$

$$z = h \sum_{i=1}^s b_i f(t + c_i h, Y_i), \quad (7.18)$$

non

$$\bar{a}_{ij} = a_{ij} - \frac{b_j}{2}, \quad 1 \leq i, j \leq s \text{ den.} \quad (7.19)$$

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

$$\begin{aligned} (I_s \otimes I_d - h \bar{A} \otimes J) \Delta Y - \frac{1}{2} (e_s \otimes I_d) \Delta z &= r, \\ (-h e_s^T B \otimes J) \Delta Y + \Delta z &= 0, \end{aligned} \quad (7.20)$$

non $e_s = (1, \dots, 1)^T \in \mathbb{R}^s$, eta $\bar{A} = (\bar{a}_{ij})_{i,j=1}^s$ den. Argi dago, $(\Delta Y, \Delta z)$ ekuazio-sistema honen (7.20) soluzioa bada, orduan ΔY gure jatorrizko ekuazio sistemaren (7.13) 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 simplektikotasun (7.14) eta simetrikotasun (7.15) propietateak berrida-tziko ditugu,

1. Simplektikoa.

Runge-kutta metodoa simplektikoa da,

$$(B\bar{A}) \text{ antisimetriko bada,} \quad (7.21)$$

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,

$$\begin{aligned} b_{s+1-i} &= b_i, \quad \bar{c}_{s+1-i} = -\bar{c}_i, \quad 1 \leq i \leq s, \\ \bar{a}_{s+1-i, s+1-j} &= -\bar{a}_{ij}, \quad 1 \leq i, j \leq s. \end{aligned} \quad (7.22)$$

Implementazio berrian, matrizeen dimentsioak zehazteko, parametro berri haue-ten 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 $s = 7 \rightarrow m = 4, s - m = 3$).

7.3.2. IRK metodo simplektikoen garapena.

Lehenengo, 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} diagonalizagarrria dela eta irudikari balio propio puruak dituela suposatzen du. Beraz, Q , $s \times s$ tamainako matrize ortogonala existitzen da,

$$Q^{-1}\bar{A}Q = \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \quad (7.23)$$

non D , balio erreal positiboen matrize diagonala eta $m \times (s-m)$ tamainako den.

Ekuazio-sistemari (7.20), aldagai aldaketa hau aplikatuz,

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

honako ekuazio sistema baliokidea lortuko dugu,

$$\begin{aligned} \begin{pmatrix} I_m \otimes I_d & -h D \otimes J \\ h D^T \otimes J & I_{s-m} \otimes I_d \end{pmatrix} W - \frac{1}{2} (Q^{-1} e_s \otimes I_d) \Delta z &= (Q^{-1} \otimes I_d) r, \\ -h (e_s^T B Q \otimes J) W + \Delta z &= 0, \end{aligned} \quad (7.24)$$

Eranskinean (A.3.), ekuazio baliokideak lortzeko eman diren urratsen zehaztapenak eman ditugu.

Sistemaren (7.24) bloke egiturari esker, bere LU deskonposaketa konputatze-ko, $d \times d$ tamainako matrizeen biderkadurak eta $[s/2] + 1$ matrize errealen ($d \times d$) LU deskonposaketak kalkulatuko ditugu:

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 balio diagonalak diren.
2. Aurreko matrizeetatik lortutako $d \times d$ dimentsioko matriza.

7.3.3. IRK metodo simetriko simplektikoen garapena.

Atal honetan, simplektikotasun propietateaz (7.14) gain, simetria propietate (7.15) ere betetzen dituzten IRK metodoak konsideratuko ditugu. Lehenengo, garape-nean erabiliko ditugun matrize laguntzaileak definituko ditugu.

1. P matriza.

Kontsideratu $P = (P_1 \ P_2) \in \mathbb{R}^{s \times s}$ matrize ortogonala, non $P_1 \in \mathbb{R}^{s \times m}$ eta $P_2 \in \mathbb{R}^{s \times (s-m)}$. 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, (7.22) simetria propietateak, $P_i^T B^{\frac{1}{2}} \bar{A} B^{-\frac{1}{2}} P_i = 0$, $i = 1, 2$ eta bestetik, propietate simplektikoak $B^{1/2} \bar{A} B^{-1/2}$ antisimetrikoa dela ziurtatzentz dutenez, \bar{A} matrizea honako matrizearen antzekoa dela ondorioztatu daiteke,

$$P^T B^{\frac{1}{2}} \bar{A} B^{-\frac{1}{2}} P = \begin{pmatrix} 0 & K \\ -K^T & 0 \end{pmatrix} \quad (7.25)$$

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 singularraren 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 diagonalala da,

$$D = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \sigma_{s-m} \end{pmatrix}, \quad D = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \sigma_{s-m} \\ 0 & 0 & \dots & 0 \end{pmatrix}. \quad (7.26)$$

s bikoitia bada, D matrizea ezkerrean eta s bakoitia bada, D matrizea eskuian ($\sigma_m = 0$) irudikatu dugu.

4. Q matrizea.

Simetrikoak den (7.23) propietateari esker, berdintza hauek baiezttatu daitezke,

$$Q = (Q_1 \ Q_2) = B^{-1/2} (P_1 \ P_2) \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} = B^{-1/2} (P_1 U \ P_2 V), \quad (7.27)$$

$$Q^{-1} = Q^T B. \quad (7.28)$$

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

Ekuazio-sistemari (7.20), aldagai aldaketa hau aplikatuz,

$$\Delta Y = (Q \otimes I_d)W = (Q_1 \otimes I_d)W' + (Q_2 \otimes I_d)W'', \quad (7.29)$$

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

eta metodoa simetrikoa izatearen lehen baldintzagatik (7.22) $e_s^T B P_2 = 0$, eta orduan $e_s^T B Q_2 = e_s^T B P_2 V = 0$ berdintasunak aplikatuz, honako ekuazio-sistema baliokidea lortuko dugu,

$$\begin{aligned} W' - h(D \otimes J)W'' - \frac{1}{2}(Q_1^T B e_s \otimes I_d)\Delta z &= (Q_1^T B \otimes I_d)r, \\ h(D^T \otimes J)W' + W'' &= (Q_2^T B \otimes I_d)r, \\ -h(e_s^T B Q_1 \otimes J)W' + \Delta z &= 0. \end{aligned} \quad (7.30)$$

Eranskinean (A.3.), ekuazioak lortzeko urratsen zehaztapenak eman ditugu.

Matrizearen egitura. $S = 6$ ataletako IRK metodoari dagokion ekuazio sistemaren matrizearen egitura berezia ikus daiteke (7.31). Aldagai aldaketarekin lortutako ekuazio sistema, blokeka diagonala da eta hau aprobetxatz, Newton iterazioaren implementazio eraginkorra lortuko dugu.

$$\left[\begin{array}{ccc|ccc} I_d & & & -h\sigma_1 J & & -\frac{\alpha_1}{2} I_d \\ & I_d & & & -h\sigma_2 J & -\frac{\alpha_2}{2} I_d \\ & & I_d & & & -\frac{\alpha_3}{2} I_d \\ \hline h\sigma_1 J & & & I_d & & 0 \\ h\sigma_2 J & & & & I_d & 0 \\ h\sigma_3 J & & & & & I_d \\ \hline -h\alpha_1 J & -h\alpha_2 J & -h\alpha_3 J & 0 & 0 & 0 \end{array} \right] = \left[\begin{array}{c|c} W' & R' \\ \hline W'' & R'' \\ \hline \Delta z & 0 \end{array} \right] \quad (7.31)$$

non

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

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

W'' kalkulatzeko ekuazioak. Sistemaren (7.30) bigarren ekuaziotik W'' askatu,

$$W'' = -h(D^T \otimes J)W' + (Q_2^T B \otimes I_d)r. \quad (7.32)$$

W' kalkulatzeko ekuazioak. W'' lehen ekuazioan (7.30) ordezkatz, honako ekuazio-sistema lortuko dugu,

$$\begin{aligned} (I_m \otimes I_d + h^2 DD^T \otimes J^2) W' - \frac{1}{2}(Q_1^T B e_s \otimes I_d) \Delta z &= R, \\ -h (e_s^T B Q_1 \otimes J) W' + \Delta z &= 0, \\ \text{non } R &= (Q_1^T B \otimes I_d) r + h (DQ_2^T B \otimes J) r \in \mathbb{R}^{md}. \end{aligned} \quad (7.33)$$

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,

$$(I_d + h^2 \sigma_i^2 J^2) W_i - \frac{\alpha_i}{2} \Delta z = R_i, \quad i = 1, \dots, m, \quad (7.34)$$

$$-h J \sum_{i=1}^m \alpha_i W_i + \Delta z = 0, \quad (7.35)$$

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 (7.26).

Δz kalkulatzeko ekuazioak. Aurreko (7.34) 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 (7.35) ekuazioan ordezkatz, $\Delta z \in \mathbb{R}^d$ askatzeko ekuazioak lortuko ditugu,

$$M \Delta z = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i, \quad (7.36)$$

non

$$M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1} \in \mathbb{R}^{d \times d}. \quad (7.37)$$

Ezezagunak askatzeko laburpena. Sistemaren ezezagunak askatzeko ekuazioak eta ordena laburtuko dugu: lehen, $\Delta z \in \mathbb{R}^d$ (7.36) ekuaziotik askatuko dugu; bigarren, $W' \in \mathbb{R}^{md}$ (7.34) ekuaziotik askatuko dugu; hirugarren, $W'' \in \mathbb{R}^{(s-m)d}$ (7.32) ekuaziotik askatuko dugu; eta azkenik, ΔY (7.29) ekuaziotik askatuko dugu.

7.4. IRK-Newton estandarra (formulazio berria).

IRK puntu-finkoaren implementazioan erabilitako birformulazio (6.atala), IRK-Newton implementazioan ere aplikatuko dugu. IRK-Newton implementazioan ordea, L_i ($i = 1, \dots, s$) aldagai ezezaguna eta Y_i ($i = 1, \dots, s$) aldagai laguntzailea kontsideratzea [?] izango da egokiena.

IRK metodoa era honetan berridatziko dugu

$$\Phi(y, t, h) := y + \sum_{i=1}^s L_i, \quad (7.38)$$

non $L_i \in \mathbb{R}^d$, $i = 1, \dots, s$ implizituki era honetan definitzen diren,

$$L_i = h b_i f(t + c_i h, y + \sum_{j=1}^s \mu_{ij} L_j), \quad i = 1, \dots, s, \quad (7.39)$$

eta

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

Newton sinplifikatuaren 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
- $$\Delta L_i^{[k]} - h b_i J_i^{[k]} \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s, \quad (7.40)$$
- 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 iterazioa aplikatzerakoan, $J_i^{[k]}$ Jacobiarra $J_i^{[0]} = \partial f / \partial y (t + c_i h, Y_i^{[0]})$ $i = 1, \dots, s$ Jacobiarraz ordezkatzen da eta askatu beharreko ekuazio sistema honakoa da,

$$\Delta L_i^{[k]} - h b_i J_i^{[0]} \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s.$$

Modu baliokidean, ekuazio lineala notazio matriziala erabiliz laburtu daiteke,

$$\left(I_s \otimes I_d - h \begin{bmatrix} b_1 \mu_{11} J_1^{[0]} & \dots & b_1 \mu_{1s} J_1^{[0]} \\ b_2 \mu_{21} J_2^{[0]} & \dots & b_2 \mu_{2s} J_2^{[0]} \\ \vdots & \ddots & \vdots \\ b_s \mu_{s1} J_s^{[0]} & \dots & b_s \mu_{ss} J_s^{[0]} \end{bmatrix} \right) \Delta L^{[k]} = g^{[k]},$$

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-simplifikatuaren iterazioa.

Honako bigarren sinplifikazioarekin, $J_i^{[0]} = \partial f / \partial y (t + c_i h, Y_i^{[0]})$, $i = 1, \dots, s$ matrizeak, $J_i^{[0]} \approx J$, $i = 0, \dots, s$ hurbilpenarekin ordezkatuz, ekuazio sistema lineal hau lortuko dugu,

$$(I_s \otimes I_d - h B A B^{-1} \otimes J) \Delta L = g. \quad (7.41)$$

non I_d, I_s identitate matrizeak eta B , (b_1, b_2, \dots, b_s) koefizienteen matrize diagonala diren.

Algoritmoa.

Formulazio berrian Newton super-simplifikatua deitu dugun algoritmoa laburtu dugu (Algoritmoa 26).

```

 $\tilde{y}_0 = fl(y_0);$ 
 $e_0 = fl(y_0 - \tilde{y}_0);$ 
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 26: IRK (Newton super-simplifikatua).

Interpolazio koefizienteak. $L_{n,i}^{[0]}$ atalen hasieraketarentzat dagokien koefizienteak era honetan definituko ditugu. IRK puntu finkoaren implementazioan finkatu genituen interpolazio koefizienteetatik abiatuta (6.13) modu errazean definituko

ditugu formulazio honi dagozkion interpolazio koefizienteak.

$$\begin{cases} Y_{n,i}^{[0]} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[0]} \\ Y_{n,i}^{[0]} = y_n + \sum_{j=1}^s \nu_{ij} L_{n-1,j} \end{cases} \Rightarrow L_n^{[0]} = (Mu^{-1}Nu)L_{n-1},$$

$$\Rightarrow (Mu^{-1}Nu)_{i,j}^s = \lambda_{ij}/a_{ij}. \quad (7.42)$$

Geratze irizpidea. Puntu-finkoaren iterazioan oinarritutako implementazioaren-tzat definitutako geratze irizpide berdina (6.15) 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,

$$\forall j \in \{1, \dots, sd\}, \quad \min \left(\{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} / \{0\} \right) \leq |\Delta_j^{[k]}|. \quad (7.43)$$

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,

$$(\tilde{y}_{n+1}, e_{n+1}) = S_{s,d}(\tilde{y}_n, \delta_n, L_{n,1}^{[k-1]}, \dots, L_{n,s}^{[k-1]}) \quad (7.44)$$

```
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 27: BaturaKonpensatua

7.5. IRK-Newton eraginkorra (formulazio berria).

Formulazio berrian, modu eraginkorrean askatu behar dugun ekuazio-lineala honakoa da,

$$(I_s \otimes I_d - h BAB^{-1} \otimes J) \Delta L = g, \quad (7.45)$$

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

Ekuazio-lineala (7.45) ebazteko, aurreko (7.3.2. atala eta 7.3.3. atala) ataletan (7.13) moduko sistemak askatzeko deskribatutako teknikak egokituko ditugu. Jarrain, IRK metodo simetriko simplektikoetarako (7.3.3. atala) garatutako teknika, formulazio berriko (7.45) sistema ebazteko nola aplikatu daitekeen deskribatuko dugu.

Formulazio estandarretik formulazio berrirako urratsa.

Formulazio berriaren implementazio eraginkorra, formulazio estandarrean eman-dako ekuazioak (7.3.3. atala) moldatuz zehaztuko dugu. Aurreko ataleko ekuazioetan, bi formulazioen aldagaien arteko erlazioak ordezkatuz,

$$\Delta L = (B \otimes I_d) \Delta Y, \quad (7.46)$$

$$r = (B^{-1} \otimes I_d) g, \quad (7.47)$$

formulazio berrirako ekuazio baliokideak lortuko ditugu.

1. Aldagai aldaketa. Formulazio estandarraren aldagai aldaketari (7.29) goiko lehen erlazioa aplikatuz,

$$\Delta L = (BQ_1 \otimes I_d) W' + (BQ_2 \otimes I_d) W''. \quad (7.48)$$

2. $R \in \mathbb{R}^{md}$ matrizea. Formulazio estandarraren R matrizearen ekuazioan (7.33) goiko bigarren erlazioa aplikatuz,

$$\begin{aligned} R &= (Q_1^T \otimes I_d) g + h (DQ_2^T \otimes J) g, \\ R &= Q_1^T g + h DQ_2^T g J^T. \end{aligned} \quad (7.49)$$

3. $W'' \in \mathbb{R}^{(s-m)d}$ matrizea. Formulazio estandarraren W'' matrizearen ekuazioan (7.32) goiko bigarren erlazioa aplikatuz,

$$W'' = -h (D^T \otimes J) W' + (Q_2^T \otimes I_d) g. \quad (7.50)$$

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) $M \in \mathbb{R}^{d \times d}$ matrizea 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 (7.45) 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 gure implementazioaren algoritmoa laburtuko dugu (Algoritmoa [28](#)),

```

 $\tilde{y}_0 = fl(y_0);$ 
 $e_0 = fl(y_0 - \tilde{y}_0);$ 
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  $\Delta z = d$ );
         $W_i = (I_d + h^2 \sigma_i^2 J^2)^{-1} (R_i + \frac{\alpha_i}{2} \Delta z)$ ,  $i = 1, \dots, m$ ;
         $W^{\prime} = (-h D^T) W^{\prime} J^T + Q_2^T g;$ 
         $\Delta L = B Q_1 W^{\prime} + B Q_2 W^{\prime \prime};$ 
         $L_n^{[k]} = L_n^{[k-1]} + \Delta L_n^{[k]};$ 
        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 28: IRK (NSS-Eraginkorra).

7.6. IRK-Newton Mixtoa.

7.6.1. Sasi-Newton iterazioa.

Hurrengo 7.6.2. ataleko IRK metodoaren implementazio berrirako, Newton iterazio metodoaren (7.40) aldaera bat konsideratuko dugu. Newton iterazio bakotzaren ekuazio sistema hau,

$$\Delta L_i^{[k]} - h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s, \quad (7.51)$$

non

$$g_i^{[k]} = -L_i^{[k-1]} + h b_i f\left(t + c_i h, y + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}\right), \quad i = 1, \dots, s, \quad (7.52)$$

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 aplikatzuz (Algoritmoa 29), $\Delta L^{[k]} \in \mathbb{R}^{sd}$ soluzioaren $\Delta L_i^{[k,0]}, \Delta L_i^{[k,1]}, \Delta L_i^{[k,2]}, \dots$ hurbilpenak 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
    
$$\left| \begin{array}{l} l = l + 1; \\ G_i^{[k,l]} = g_i^{[k]} - \Delta L_i^{[k,l-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,l-1]}, \quad i = 1, \dots, s; \\ \Delta L^{[k,l]} = \Delta L^{[k,l-1]} + (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} G^{[k,l]}; \end{array} \right.$$

end
```

Algoritmoa 29: Barne iterazioa.

non $fl_{32}(x)$, $x \in \mathbb{R}$ zenbakiaren gertuen dagoen IEEE doitasun arrunteko (32-bit) balioa adierazten duen.

IRK implementazio berri honetan (Algoritmoa 30), ekuazio sistema linealaren (7.51) J_i Jacobiar matrizeen ebaluazioa, doitasun arrunta duten Y_i atalekin kalkulatuko dugu. Beraz, barne iterazioen geratze irizpidea (Algoritmoa 29), $fl_{32}(\Delta L^{[k,l]}) = fl_{32}(\Delta L^{[k,l-1]})$ doitasun arruntean betetzen dela aztertzea nahi-koia izango dugu.

7.6.2. IRK-Newton Mixtoa

Zenbakizko soluzioa $y_n \approx y(t + hn) \in \mathbb{R}^d$, $n = 1, 2, \dots$, 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 faseetan kalkulatuko dugu:

1. $L^{[0]} = 0 \in \mathbb{R}^{sd}$ atalak hasieratu, eta Newton super-simplifikatuaren iterazioak aplikatu ((7.40) iterazioaren ekuazio-sistema, (7.41) sistemarekin ordezkatuz),

$$L^{[1]} = L^{[0]} + \Delta L^{[1]}, \quad L^{[2]} = L^{[1]} + \Delta L^{[2]}, \dots$$

geratze irizpidean, $fl_{32}(L^{[k]}) = fl_{32}(L^{[k-1]})$ bete arte.

2. $L^{[k]}$ berriari dagokion Jacobiarra 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, (7.51) ekuazio sistema linealaren $\Delta L^{[k]}$ soluzio zehatzaren $\Delta L^{[k,0]}$ hurbilpena konsideratu eta barne iterazioak (Algoritmoa 29) 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 (7.51)-(7.52), $\Delta L^{[k]}$ soluzioaren $\Delta L^{[k,\ell]}$ hurbilpenak (berriz ere doitasun arruntean) Algoritmoa 29 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 (Algoritmoa 11) 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 eraginkorratekin (Algoritmoa 28) askatuko ditugu.
- $\mu_{ij} \in \mathbb{F}$ koefizienteek, zehazki (6.8) 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 (7.52) 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,

$$\begin{aligned} h b_i f \left(t + c_i h, \tilde{y} + e + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]} \right) - L_i^{[k-1]} &\approx \\ \left(h b_i f_i^{[k]} - L_i^{[k-1]} \right) + h b_i J_i e, \end{aligned}$$

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 algoritmoa laburtu dugu Algoritmoa 30.

```

 $L^{[0]} = 0;$ 
 $J = \frac{\partial f}{\partial y}(t + h/2, \tilde{y});$ 
 $M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1};$ 
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 30: IRK implementazio Mixtoa

7.7. Zenbakizko esperimentuak.

Atal honetan, Newton iterazioan oinarritutako 6-ataletako Gauss kolokazio metodoaren implementazioarekin egindako zenbakizko esperimentuak azalduko ditugu. Esperimentu hauen konputaziorako, 64-biteko doitasuneko IEEE komahigikorreko aritmetika erabili dugu.

7.7.1. Problemak

Pendulu bikoitz zurruna

Pendulu bikoitz zurrunaren problemaren Hamiltondarra eta parametroak, 3.2.2. 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: $k = 0$ problemarako, [?] artikulutik izaera ez-kaotikoa duen hasierako balioak hartu ditugu: $q(0) = (1.1, -1.1)$ and $p(0) = (2.7746, 2.7746)$. $k \neq 0$ problematarako 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 $k \rightarrow \infty$ bornatua dagoen.

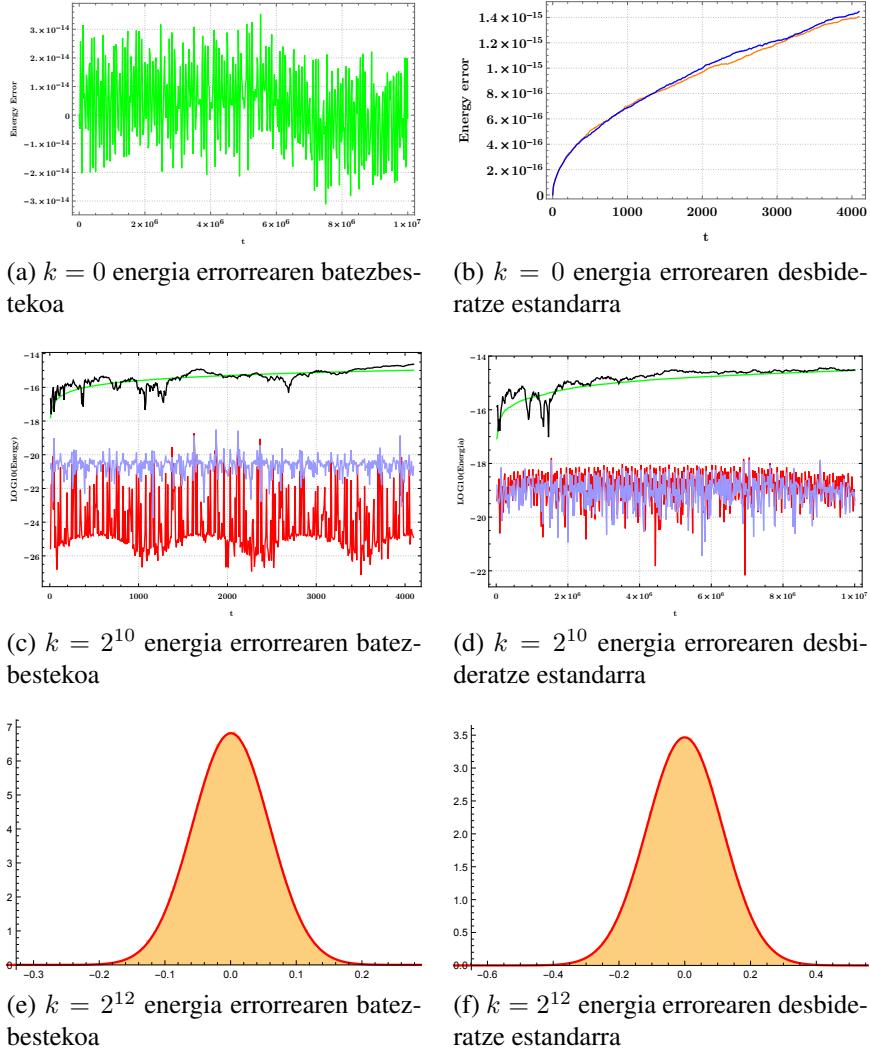
7.7.2. Biribiltze errorearen azterketa.

Lehenengo, gure Newton iterazioan oinarritutako IRK implementazio berriaren biribiltze errorearen propagazioa egokia dela aztertuko dugu. Aurreko artikuluan [?], biribiltze errorearen propagazioa gutxitzeko ahalegin berezia eginez, puntu-finkoa iterazioan oinarritutako IRK implementazioa proposatu genuen. Bi implementazioen, 6-ataleko Gauss kolokazio metodoaren biribiltze erroreak konparatu-ko 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 biribiltze erroreari nagusitzen zaion. Biribiltze errorearen konparaketa sendoago izan dadin ([?]), azterketa estatistikoa egin dugu. Problema bakoitzarentzat, $\mathcal{O}(10^{-6})$ errore tamainako ausaz perturbatutako $P = 1000$ hasierako balio ezberdin kontsideratu ditugu.

7.1. 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 proportzionalak dira.



7.1. 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)

7.7.3. Puntu-finkoa versus Newton iterazioa

[7.1.](#) taulan, k parametroaren lau balioetarako, bi implementazioen eraginkortasunaren adierazle nagusienak laburtu ditugu.

7.1. 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 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 implementazioa baino eraginkorragoa da. Baina, 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.

7.8. Laburpena.

8. Kapitulua

IRK: Eguzki-sistema.

8.1. Sarrera.

Koordenatu kartesiarrak erabiltzearen abantaila.

8.2. Eguzki-sistemaren integrazorako metodoak (review).

8.2.1. Efemerideak.

Konputagailuen aurreko garaian, efemerideak teoria analitikoetan oinarritutako serie funtzioen bidez kalkulatzen ziren. Soluzio hauetan, Fourier serie trigonométriko luzeen ebaluazioa egin behar zen. 1960 urteetan eguzki-sistemaren ezagutza hobetu zenean (accuracy of new observation types and space missions), serie oso luzeak kalkulatu behar zituzten, eta zenbakizko integrazioen bidezko soluzioak eraginkorragoak bilakatu ziren [?].

Eguzki-sistemaren gorputzen efemeride modernoak, mugimenduaren ekuazio differentzialen zenbakizko integrazioaren bidez kalkulatzen dira. Integrazioaren hasierako balioak eta ereduaren parametroak, sateliteen bidez jasotako datuei egokitzen zaizkie.

Efemerideetarako eguzki-sistemaren eredu konplexua erabiltzen da. N-gorputz nagusien arteko indar grabitazionalez gain, erlatibilitatea, asteroideek era-gindako grabitazio indarrak, gorputzen formen eragina eta beste hainbat indar ez grabitacionalak kontutan hartzen dira. Mugimenduaren ekuazio differentzialak (Einstein-Imfeld-Hoffmann, c^{-4} PPN hurbilketa) hauek dira,

$$\ddot{x}_{\text{Planet}} = \sum_{A \neq B} \mu_B \frac{r_{AB}}{\|r_{AB}\|^3} + \ddot{x}_{GR}(\beta, \gamma, c^{-4}) + \ddot{x}_{AST,300} + \ddot{x}_{J_2}$$

- 8 planetak, Ilargia, Pluto eta 300 asteroide.
- GR: erlatibilitate efektua.
- J_2 : eguzkia esferikoa ez izatearen eragina.
- Urrats luzeera, $h = 0.055$ egunekoa da.

Ekuazio differentzialak (erlatibilitate efektua). Eguzkiaren erlatibilitate efektua kontutan hartzen duten ekuazio differentzialak azalduko ditugu.

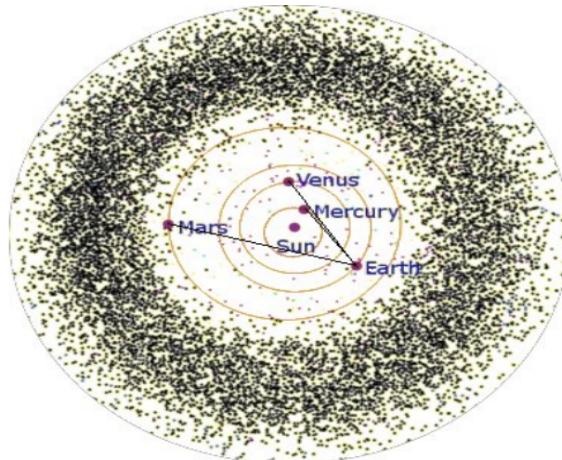
$$\dot{q}_i = v_i, \quad i = 0, 1, \dots, N \quad (8.1)$$

$$\begin{aligned} \dot{v}_i = & \sum_{j=0, j \neq i}^N \frac{Gm_j}{\|q_j - q_i\|^3} (q_j - q_i) \left(1 - \frac{2(\beta + \gamma)}{c^2} \sum_{k=0, k \neq i}^N \frac{Gm_k}{\|q_k - q_i\|} - \frac{2\beta - 1}{c^2} \sum_{k=0, k \neq j}^N \frac{Gm_k}{\|q_k - q_j\|} \right. \\ & + \gamma \left(\frac{v_i}{c} \right)^2 + (1 + \gamma) \left(\frac{v_j}{c} \right)^2 - \frac{2(1 + \gamma)}{c^2} v_i v_j \\ & \left. - \frac{3}{2c^2} \left(\frac{(q_i - q_j)v_j}{\|q_j - q_i\|} \right)^2 + \frac{1}{2c^2} (q_j - q_i) \dot{v}_i \right) \\ & + \frac{1}{c^2} \sum_{j=0, j \neq i}^N \frac{Gm_j}{\|q_j - q_i\|^3} ((q_i - q_l)((2 + 2\gamma)v_i - (1 + 2\gamma)v_j))(v_i - v_j) \\ & + \frac{3 + 4\gamma}{2c^2} \sum_{j=0, j \neq i}^N \frac{Gm_j \dot{v}_j}{\|q_j - q_i\|} \end{aligned} \quad (8.2)$$

8.1. Taula: Konstanteak

c	299792.458 km/s	Argiaren abiadura
au	149597870.700 km	Astronomical unit
β	1.0	PPN parametroa
γ	1.0	PPN parametroa

Astroideek, bereziki Marte planetaren mugimenduarengan eragina dute eta kontutan hartzekoak, barne planeten mugimenduaren doitasun handiko emaitzak behar ditugunean. Bost astroide nagusiren masak (Ceres, Pallas, Vesta, Iris eta Bamberga) Merkurio eta Pluto planeten mailakoak direnez, integrazioetan gehitzen dira. Beste astroide txikien talde handia, estimazioen bidez simulatzen dira.



8.1. Irudia: Asteroideak.

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 aritmetika erabiltzeko aukera oso garestia delako bere erabilera baztertzen da eta 64-bit doitasuneko aritmetika hobetzeko teknika konputazionalki merkeak aplikatzen dira.

Hiru dira planetak efemerideak,

1. Jet Propulsion Laboratory (*EEBB*) NASA-ko erakundeak DE (Development Ephemerides) izeneko efemerideak.

1984. urtean kalkulatu zen lehen efemeridea (DE-200) eta 2.014. urteko da *DE-430* [?] publikatutako azken efemeridea. Kalkulatutako integrazio tartea, (1550 – 2650) izan da.

Zenbakizko integrazio metodoa. Urrats luzera eta ordena aldakorreko *Multistep Adams* metodoa, *DIVA/QIVA* (Krogh, 1997). *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. arte, teori analitikoetan oinarritutako efemerideak garatu zituzten. 2.003. urtean kalkulatu zuten lehen zenbakizko integrazio bidezko efemeridea eta *INPOP13c* (2.014) publikatutako azkena da.

Zenbakizko integrazio metodoa. Urrats finkoa eta 12 ordeneko *Adams-Cowell* metodoa da.

Doitasuna. C lengoain implementatuta dago eta *Intel* makinetako 80-biteko doitasuna erabiltzen du. Era berean, modu merkean doitasun handitzeko, doitasun laukoitza simulatzu urrats zuzentzaile (corrector step) bat aplikatzten zaio [?].

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* (2.014) [?] publikatutako azken efemeridea da.

Zenbakizko integrazio metodoa. *Everhart* izeneko *IRK* metodoa (Gauss-Radau) da. 23 ordeneko metodoa eta urrats luzera finkoa erabiltzen du.

Doitasuna. Implementazioak (software package ERA), *Intel* makinetako 80-biteko doitasuna erabiltzen du.

(8.2.) taulan, planeten efemerideen doitasunaren eboluzioa ikus daiteke. Hiru efemerideak antzeko doitasuna azaltzen dutela aipatu beharra dago.

	Le Verrier		DE200		DE421, INPOP08, EPM2008		DE430, INPOP15a, EPM2014	
	c. 1900		1980		2008		2014-2015	
	angle "	distance km	angle "	distance km	angle "	distance km	angle "	distance km
Mercury	1	450	0.020	5	0.0030	0.40	0.0002	0.020
Venus	0.5	100	0.020	2	0.0004	0.02	0.0002	0.004
Earth			0.010	1	0.0004	0.01	0.0002	0.002
Mars	0.5	150	0.010	3	0.0004	0.01	0.0002	0.002
Jupiter	0.5	1400	0.1	50	0.0050	2.	0.0040	1.5
Saturn	0.5	3000	0.2	350	0.0010	0.2	0.0002	0.2
Uranus	1	12700	0.4	5000	0.0100	100.	0.0050	50.
Neptune	1	22000	1.0	8000	0.0100	300.	0.0050	200.
Pluto			2.5	80000	0.0200	1200.	0.0200	500.

8.2. Irudia: Efemerideen doitasunaren eboluzioa.

8.2.2. Eguzki-sistemaren integrazio luzeak.

A.Morbidellik [?] eguzki-sistemaren zenbakizko integrazioen algoritmoen garapenaren azterketan, garai hauet bereizten ditu:

1. Garai klasikoa (The classical period).

90. hamarkada hasiera arte, urrats luzera aldakorreko integratzaileak erabilten dira: Runge-Kutta (Dormand et al. 1987), Bulirsch and Stoer (1966), Radau (Everharht, 1985), eta Störmer (1990). Garai honetan integrazio tarreak $10^4 - 10^6$ dira.

2. Garai simplektikoa (The symplectic period).

Wisdom eta Holmanen [?, 1991] lanarekin, eguzki-sistemaren azterketarako integratzaile sinpletikoen erabilera zabaldu zen. Garai honetan, ($10^8 - 10^9$) integrazio tarteko eguzki-sistemaren azterketak egin ziren.

3. Garai estatistikoa (The statistic period).

Planeten eta gorputz txikiren (asteroide, meteoritoak) arteko kolisio gertuko egoerak kalkulatzen dituzten algoritmoak garatu ziren. Inplementazio berri hauetan, milaka gorputzen integrazio azkarra egin daiteke. Horrela, asteroide eta meteoritoen orbiten distribuzio azterketa estatistikoak egin ziren.

4. Planeten sorrera azterketen garaia (The planetary accretion period).

Masa handiko gorputzen arteko kolisio gertuko egoerak kalkulatu daitezke eta beraz, eguzki-sistemaren sorrerari buruzko simulazioak nagusituko dira.

Mekanika zelestean bi integratzaile famili agertzen dira nagusi: integratzaile simetrikoak eta simplektikoak. Metodo simetrikoetan urrats luzera tamaina aldakorreko integratzaileak, modu errezean implementatu daitezke. Metodo simetrikoen artean nagusiena, 4 ordeneko *Hermite* integratzailea (Aarseth [?]) da. Hermite integratzailea konputazionalki garestia da eta bereziki, gorputz kopuru handia dituzten eta kolisio gertuko egoerak maiz gertatzen diren problemetan (eguzki-sistemaren sorrera, stellar dynamics, ...) aplikatzen da.

Gure lan eremua, eguzki-sistemaren epe luzeko integrazioak dira eta gaur egun arlo honetan, nagusi bilakatu diren integratzaile simplektikoak aztertuko ditugu. Eguzki-sistemaren integrazio luzeei buruzko beste errebizio dokumentu interesgarri hauek aholkatu nahiko genituzke: [?] (Kholshevnikov 2007, 2.007), [?] (Brumberg, 2.013) eta [?] (Ito eta Tanikawa, 2.007).

Eguzki-sistemari egokitutako integratzaile simplektikoak.

Wisdomek eta Holmanenek bere lanean [?, 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. Metodoan n-planeten Hamiltondarra Jacobi koordenatuak erabiliz, bi zatitan banatu zuten

$$H(q, p) = H_K(p) + H_I(q) \quad \text{non} \quad H_K \gg H_I,$$

H_K , Hamiltondar Kepleriarra (planeten eguzkiarekiko mugimendu kleperiarra) eta H_I , Interakzioen Hamiltondarra (planeten arteko grabitazio interakzioak). Integrazioaren urrats bakoitzean, Hamiltondar bakoitzaren soluzioa tartekatuz, problema osoaren ebazpena kalkulatzen da.

WH integratzailea, ondorengo metodoen aurrekaria kontsideratu bada ere, berre aplikagarritasuna mugatua da. Batetik, izar anitzeko planeten sistemak edo planeta-ilargiak sistemak integratzeko ez da egokia. Bestetik, *WH* metodo sinplektikoa denez urrats luzera finkoarekin integratu beharra, eta hau, gorputzen arteko kolisio gertuko egoerak dituzten problemak modu eraginkorream integratzeko eragozpen bat da. Arazo hauek gainditzeko, urteetan zehar algoritmo honen aldaerak proposatu dira eta jarraian, nagusienak aipatuko ditugu.

Levinson eta Duncan-ek [?, 1994], *WH* implementazioa, integratzaile ez sinplektiko batekin konbinatu zuten kolisio egoeren kalkulua hobetzeko. *SWIFT* paketean *RMVS3* izeneko integratzailea implementatu zuten. Duncan, Levinson eta Lee-k [?, 1998], koordenatu Heliozentrikoak erabiliz, Hamiltondarra beste modu batean banatu zuten,

$$H(q, p) = H_K(p) + H_C(p) + H_I(q)$$

eta kolisio egoerak gertatzen diren unean, urratsa luzera txikituz aurre egin zioten. Implementazioa *SYMBA* izenekoa da. Chambers-ek [?] koordenatu Heliozentrikoetan oinarritu zen eta kolisio egoeretan, beste integratzaile (Bulirsch-Stoer metoda) batera aldatuz kalkulatzen ditu. Implementazio honi *MERCURY* izena eman zion. Levinson eta Duncan-ek (2000), aurreko implementazioaren arazo batzuk konponduz, *modified SYMBA* izeneko garapen berria egin zuten. Kvaerno eta Leimkuhler [?] eta beste autore batzuk ere, antzeko ideiak landu dituzte.

Wisdom eta Holmanek proposatutko Hamiltondarraren banaketa, *leapfrog* metodoaren bidez integratzen da eta beraz, 2 ordeneko da. Orden altuagoko ($p > 2$) metodoak definitzeko, koefiziente negatiboak erabili behar dira [?] [?] eta ez dira interesgarriak, *leapfrog* metoda hauek baino eraginkorragoa baita.

Adibidea. Yoshidaren 4 ordeneko metoda. ϕ_h oinarrizko metoda *leapfrog* izanik, metodoaren konposaketari dagokion 4 ordeneko konposizio metoda,

$$\Psi_h = \phi_{\gamma_1 h} \circ \phi_{\gamma_0 h} \circ \phi_{\gamma_1 h}.$$

non $\gamma_0 = -2^{1/3}/(2 - 2^{1/3})$ eta $\gamma_1 = 1/(2 - 2^{1/3})$.

Beranduago, McLachlan-ek [?, 1995], Laskar-ek eta Robutel-ek [?, 2001] koeffiziente negatiboen arazoa gainditu zuten eta orden altuko splitting eskemak aurkitu zitzuten. Laskar-Nature: "The stepsize is 2.5×10^{-2} years, unless the eccentricity of the planets increases beyond about 0.4, in which case the step size is reduced to preserve numerical accuracy". Berriki, Blanes et al [?, 2012] orden altuko splitting eskema berriak eta eraginkorrapak aurkitu ditu.

Hernandez eta Bertschinger-ek [?, 2015] N-body problema gravitacional eta kolisiōdunetarako 2 ordeneko integratzaile simplektiko berri bat proposatu dute. Hernandez eta Bertschinger-ek [?, 2015] koordenatu kartesiarretan oinarrituz, N-Body problema 2-gorputzen problemetan banatzen dute. Honako Hamiltondarren banaketa proposatzen dute,

$$\begin{aligned} H &= T + V, \\ H &= T + \sum_i \sum_{i>j} V_{ij}, \\ H &= T + \sum_i \sum_{i>j} (K_{ij} - T_{ij}) \end{aligned}$$

8.3. Gure implementazioa.

Sarrera.

Demagun honako Hamiltondar banagarria dugula,

$$H(y) = H_A(y) + H_B(y), \text{ non } H_A \gg H_B.$$

Eta beraz, dagokion hasierako problema orokorra,

$$\dot{y} = J^{-1} \nabla H(y) = f(y), \quad y(t_0) = y_0.$$

$\dot{y} = f(y)$, eredu simple $k(y) = J^{-1} \nabla H_A(y)$ eta eredu konplexu $g(y) = J^{-1} \nabla H_B(y)$ baten arteko batura gisa deskonposatu daiteke,

$$\dot{y} = f(y) = k(y) + g(y).$$

Eredu simplea konputazionalki merkea da eta eredu konplexua aldiz, garestia. Banaketa honekin, zati konplexuaren ebaluazio kopuru txikienarekin integratu nahi dugu, zenbakizko integrazio eraginkorra lortzeko.

Eguzki-sistemaren ekuazioak.

N-gorputzen problemaren Hamiltondarra, alde Kepleriarra (eguzkiarekiko interakzioa) eta planeten interakzioen batura gisa bana daiteke,

$$H(q, p) = H_k + H_I, \quad H_k \gg H_I.$$

Eguzkiari dagokion azpindizea $i = 0$ kontsideratzen badugu,

$$\begin{aligned} H_k(q, p) &= \frac{1}{2} \sum_{i=0}^N \frac{p_i^2}{m_i} - \sum_{i=1}^N \frac{G m_0 m_i}{\|q_i - q_0\|}, \\ H_I(q) &= \sum_{1 \leq i < j \leq N}^N \frac{G m_i m_j}{\|q_j - q_i\|} \end{aligned}$$

Banaketa honi dagokion ekuazio diferentzialak, $f(y) = k(y) + g(y)$ modu honetan laburtuko ditugu. Honako notazioa erabiliz,

$$\dot{y} = f(y) = \begin{pmatrix} \dot{q} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} f_q(y) \\ f_v(y) \end{pmatrix} = \begin{pmatrix} k_q(y) \\ k_v(y) \end{pmatrix} + \begin{pmatrix} g_q(y) \\ g_v(y) \end{pmatrix}$$

Batetik, $\dot{q} = f_q(y)$ ekuazio diferenzialen deskonposaketa honakoa da,

$$\begin{aligned} \dot{q} &= f_q(y) = v, \\ \dot{q} = f_q(y) &\Rightarrow \begin{cases} k_q(y) = v_i, & i = 0, \dots, N. \\ g_q(y) = 0, & i = 0, \dots, N. \end{cases} \end{aligned}$$

Bestetik, $\dot{v} = f_v(y)$ ekuazio diferenzialen deskonposaketa honakoa da,

$$\begin{aligned} \dot{v} = f_v(y) &= \sum_{j=0, j \neq i}^N \frac{G m_j}{\|q_j - q_i\|^3} (q_j - q_i), \\ \dot{v} = f_v(y) &\Rightarrow \begin{cases} k_v(y) = \begin{cases} \dot{v}_0 = \sum_{j=1}^N \frac{G m_j}{\|q_j - q_0\|^3} (q_j - q_0). \\ \dot{v}_i = \frac{G m_0}{\|q_0 - q_i\|^3} (q_0 - q_i), & i = 1, \dots, N. \end{cases} \\ g_v(y) = \begin{cases} \dot{v}_0 = 0. \\ \dot{v}_i = \sum_{j=1, j \neq i}^N \frac{G m_j}{\|q_j - q_i\|^3} (q_j - q_i), & i = 1, \dots, N. \end{cases} \end{cases} \end{aligned}$$

Meta-algoritmoa.

IRK metodoaren formulazioa gogoratuz,

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

S-ataleko IRK metodoaren iterazio bakoitzean, $Y \in \mathbb{R}^{s \times d}$ ezezagunetako ekuazio-sistema askatu behar dugu:

$$Y_i - y_n - \sum_{j=1}^s \mu_{ij} h b_j \left(k(Y_j) + g(Y_j) \right) = 0, \quad i = 1, \dots, s.$$

Gure planteamenduan ekuazio-sistema Newton-simplifikatuaren bidez askatuko dugu, baina jakobiarraren kalkulurik gabe. Lortzen dugun metodoa, jakobiarraren hurbilpena alde kepleriarra kontsideratzen duen ($J = k'(Y_i)$) Newton-simplifikatuaren baliokidea da.

Garapena.

Ekuazio-sisteman Newton metodoa aplikatuz. Soluziotik gertu dagoen balio batetik abiatuta, $(Y_i^{[0]})$ eta $k = 1, 2, \dots$,

$$\Delta Y^{[k]} = -\frac{F(Y^{[k]})}{F'(Y^{[k]})},$$

$$Y^{[k+1]} = Y^{[k]} + \Delta Y^{[k]}.$$

IRK metodoaren ekuazio-sistemari aplikatuz,

$$\Delta Y_i^{[k]} = -\frac{\left(Y_i^{[k]} - y_n - \sum_{j=1}^s \mu_{ij} h b_j \left(k(Y_j^{[k]}) + g(Y_j^{[k]}) \right) \right)}{\left(1 - \sum_{j=1}^s \mu_{ij} h b_j \left(k'(Y_j^{[k]}) + g'(Y_j^{[k]}) \right) \right)}$$

Ekuazio laburtzeko $\delta_i^{[k]}$ aldagai laguntzailea erabiliz hau da askatu beharreko ekuazio,

$$\Delta Y_i^{[k]} = \sum_{j=1}^s \mu_{ij} h b_j \left(k'(Y_j^{[k]}) + g'(Y_j^{[k]}) \right) \Delta Y_j^{[k]} + \delta_i^{[k]}, \quad (8.3)$$

non,

$$\delta_i^{[k]} = -Y_i^{[k]} + y_n + \sum_{j=1}^s \mu_{ij} h b_j (k(Y_j^{[k]}) + g(Y_j^{[k]})).$$

Lortutako espresioa garatuko dugu.

1. Lehen hurbilpena.

$g'(Y_j^{[k]}) << k'(Y_j^{[k]})$ eta $g'(Y_j^{[k]}) < \Delta Y_j^{[k]}$ denez,

$$\Delta Y_i^{[k]} \approx \sum_{j=1}^s \mu_{ij} h b_j k'(Y_j^{[k]}) \Delta Y_j^{[k]} + \delta_i^{[k]} \quad (8.4)$$

2. Bigarren hurbilpena (Linealizazioa).

$$\begin{aligned} k(Y_j^{[k+1]}) &= k'(Y_j^{[k]})(Y_j^{[k+1]} - Y_j^{[k]}) + k(Y_j^{[k]}) + O(\|\Delta Y_j^{[k]}\|^2) \\ \Delta Y_j^{[k]} &= Y_j^{[k+1]} - Y_j^{[k]} \end{aligned}$$

Honako hurbilpena,

$$k'(Y_j^{[k]}) \Delta Y_j^{[k]} \approx k(Y_j^{[k]} + \Delta Y_j^{[k]}) - k(Y_j^{[k]})$$

ordezkatuz eta garatuz,

$$\Delta Y_i^{[k]} = \sum_{j=1}^s \mu_{ij} h b_j k(Y_j^{[k]} + \Delta Y_j^{[k]}) - \sum_{j=1}^s \mu_{ij} h b_j k(Y_j^{[k]}) + \delta_i^{[k]},$$

$$\Delta Y_i^{[k]} = -Y_i^{[k]} + y_n + \sum_{j=1}^s \mu_{ij} h b_j k(Y_j^{[k]} + \Delta Y_j^{[k]}) + \sum_{j=1}^s \mu_{ij} h b_j g(Y_j^{[k]}). \quad (8.5)$$

3. Ekuazioak berridatiziz.

$\Delta Y_i^{[k]} = Y_i^{[k+1]} - Y_i^{[k]}$ definizioa erabliaz,

$$Y_i^{[k+1]} = y_n + \sum_{j=1}^s \mu_{ij} h b_j k(Y_j^{[k+1]}) + \sum_{j=1}^s \mu_{ij} h b_j g(Y_j^{[k]}). \quad (8.6)$$

Meta-Algoritmoa.

IRK metodoa aplikatzeko meta algoritmoa planteatuko dugu.

```

 $y_0 = \text{float}(y(t_0));$ 
 $e_0 = \text{float}(y(t_0) - y_0);$ 
for  $n \leftarrow 0$  to ( $\text{endstep} - 1$ ) do
    Hasieratu  $Y_{n,i}^{[0]}, W_{n,i}^{[0]}$  ,  $i = 1, \dots, s$ ;
     $k = 1;$ 
     $K_{n,i}^{[k]} = k(Y_{n,i}^{[k-1]});$ 
    Askatu  $Y_{n,i}^{[k]} = y_n + \left( \sum_{j=1}^s \mu_{ij} h b_j K_{n,j}^{[k]} \right) + (e_n + W_{n,i}^{[k-1]});$ 
    while (konbergentzia lortu) do
         $k = k + 1;$ 
         $W_{n,i}^{[k-1]} = \sum_{j=1}^s \mu_{ij} h b_j g(Y_{n,j}^{[k-1]});$ 
         $K_{n,i}^{[k]} = k(Y_{n,i}^{[k-1]});$ 
        Askatu  $Y_{n,i}^{[k]} = y_n + \left( \sum_{j=1}^s \mu_{ij} h b_j K_{n,j}^{[k]} \right) + (e_n + W_{n,i}^{[k-1]});$ 
    end
    if (NormalizeDistance( $Y^{[k]}, Y^{[k-1]}$ ) > 1) then
        | exit;
    else
         $F_{n,i}^{[k]} = f(Y_{n,i}^{[k]});$ 
         $L_{n,i}^{[k]} = h b_i F_{n,i}^{[k]};$ 
         $(y_{n+1}, e_{n+1}) \leftarrow \text{BaturaKonpensatua}(y_n, e_n, L_n^{[k]}, F_n^{[k]});$ 
    end
end

```

Algoritmoa 31: Main Algorithm

Meta-algoritmoari buruzko hainbat ohar:

1. Barne iterazioak.

Barne iterazioan, metodo egokiena aplikatu daiteke. Problema ez-stiffa bada, puntu finkoaren bidez askatu daiteke eta problema stiffa bada Newton sinplifikatuaren bidez.

Puntu finkoaren bidezko barne-iterazioa :

```

 $l = 0;$ 
 $Y_{n,i}^{[k,0]} = Y_{n,i}^{[k-1]};$ 
while (konbergentzia lortu) do
     $l = l + 1;$ 
     $K_{n,i}^{[k,l]} = k(Y_{n,i}^{[k,l-1]});$ 
     $Y_{n,i}^{[k,l]} = y_n + \left( \sum_{j=1}^s \mu_{ij} h b_j K_{n,j}^{[k,l]} \right) + (e_n + W_{n,i}^{[k-1]});$ 
end

```

Algoritmoa 32: Main Algorithm

2. Problema independenteak.

Era honetako deskonposaketa bat dugunean,

$$f \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} k_1(y_1) \\ k_2(y_2) \end{pmatrix} + \begin{pmatrix} g_1(y_1, y_2) \\ g_2(y_1, y_2) \end{pmatrix},$$

eredu simplifikatua problema independenteak osatzen dituzte eta barne iterazioak modu independentean kalkula daitezke. N-gorputzen eguzki-sistemaren adibidean, eredu simplifikatua $k(y)$ (eguzkiarekiko interakzioa) planeta bat koitzarentzat problema independentea dugu. Kasu honetan urrats bat finkatuta, kanpo planeten $k(y)$ problema barruko planeta baino azkarrago konbergituko du.

Orokorpenea.

Aurreko atalean, maila bakarreko eredu deskonposaketa aztertu dugu. Ideia orokortuz, eredu deskonposaketa maila ezberdinetan egin daiteke. Problema bat emanda $\dot{y} = f(y)$,

$$1. \text{ maila } \begin{cases} \text{Eredu osoa. } f(y) \\ \text{Eredu simplea. } \tilde{f}(y) \end{cases} \Rightarrow f = \tilde{f} + (f - \tilde{f}) \quad (8.7)$$

$$2. \text{ maila } \begin{cases} \text{Eredu osoa. } \tilde{f}(y) \\ \text{Eredu simplea. } \tilde{\tilde{f}}(y) \end{cases} \Rightarrow \tilde{f} = \tilde{\tilde{f}} + (\tilde{f} - \tilde{\tilde{f}}) \quad (8.8)$$

Adibidea.

Demagun ekuazio diferentzialak, m perturbazio funtzio dituela,

$$\dot{y} = f(y) = k(y) + g_1(y) + g_2(y) + \cdots + g_m(y)$$

non $k(y) \ll g_k(y)$, $k = 1, \dots, m$.

$m = 2$ deneko kasu partikulara aztertuko dugu,

$$\dot{y} = f(y) = k(y) + g_1(y) + g_2(y).$$

Askatu behar dugun ekuazio,

$$\text{Askatu } Y_{n,i} = y_n + \sum_{j=1}^s \mu_{ij} h b_j (k(Y_{n,j}) + g_1(Y_{n,j}) + g_2(Y_{n,j}))$$

$$y_0 = \text{float}(y(t_0));$$

$$e_0 = \text{float}(y(t_0) - y_0);$$

for $n \leftarrow 0$ **to** ($\text{endstep} - 1$) **do**

Hasieratu $Y_{n,i}^{[0]}, W_{n,i}^{[0]}$, $i = 1, \dots, s$;

$k = 1$;

$K_{n,i}^{[k]} = k(Y_{n,i}^{[k-1]})$;

Askatu $Y_{n,i}^{[k]} = y_n + \left(\sum_{j=1}^s \mu_{ij} h b_j K_{n,j}^{[k]} \right) + (e_n + W_{n,i}^{[k-1]})$;

while (*konbergentzia lortu*) **do**

$k = k + 1$;

$W_{n,i}^{[k-1]} = \sum_{j=1}^s \mu_{ij} h b_j g_2(Y_{n,j}^{[k-1]})$;

$K_{n,i}^{[k]} = k(Y_{n,i}^{[k-1]})$;

Askatu $Y_{n,i}^{[k]} = y_n + \left(\sum_{j=1}^s \mu_{ij} h b_j K_{n,j}^{[k]} \right) + (e_n + W_{n,i}^{[k-1]})$;

end

if (*NormalizeDistance*($Y^{[k]}, Y^{[k-1]}$) > 1) **then**

exit;

else

$F_{n,i}^{[k]} = f(Y_{n,i}^{[k]})$;

$L_{n,i}^{[k]} = h b_i F_{n,i}^{[k]}$;

$(y_{n+1}, e_{n+1}) \leftarrow \text{BaturaKonpensatua}(y_n, e_n, L_n^{[k]}, F_n^{[k]})$;

end

end

Algoritmoa 33: Main Algorithm

Erlatibilitatea efektua gehitzerakoan, N-gorputzen problemari dagokion ekuazio differentziala,

$$\dot{y} = f(y), \quad f(y) = k(y) + g(y) + rs(y) + rn(y),$$

$k(y)$: kepleriarra .

$g(y)$: planeten arteko gravitazio interakzioak .

$rs(y)$: eguzkiaren erlatibilitatea efektua .

$rn(y)$: planeten arteko erlatibilitatea efektuak .

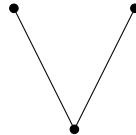
Adierazpena.

Ekuazio differentzialen deskonposaketak, zuhaitz moduan adieraz daitezke.

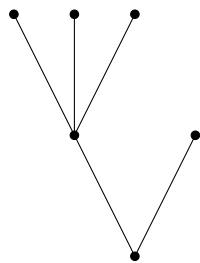
$$\dot{y} = f(y).$$

•

$$f \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} f_1(y_1) \\ f_2(y_2) \end{pmatrix} + \begin{pmatrix} g(y) \end{pmatrix},$$



$$f_1 \begin{pmatrix} y_1 \end{pmatrix} = \begin{pmatrix} f_{11}(y_{11}) \\ f_{12}(y_{11}, y_{12}) \\ f_{13}(y_{11}, y_{12}, y_{13}) \end{pmatrix} + \begin{pmatrix} g_1(y_1) \end{pmatrix},$$



8.4. Denbora birparametrizazioa.

Sarrera.

Uurrats luzera finkoa, ez da kolisio gertuko egoerak dituzten sistema dinamikoak edo denbora maila oso ezberdinak dituzten problemak integratzeko eraginkorra.

Erregularizazio da arazo honen aurrean teknika arrakastatsuena. Erregularizazioaren bidez, gorputzen arteko distantzia zerorantz hurbildu arren, mugimenduaren ekuazioak ez singularrak mantentzen dira.

Demagun jatorrizko ekuazio diferenziala,

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

non y menpeko aldagai eta t aldagai askea den.

Aldagai askeari aldaketa bat aplikatuz ($s()$ izeneko funtziotan), ekuazio diferenziala leuntzea lortuko dugu.

$$y = z,$$

$$\frac{dt}{d\tau} = s(z)$$

Honako garapena egingo dugu aldagai berriarekiko (τ) ekuazioak lortzeko.

$$y = z \Rightarrow \frac{dy}{dt} = \frac{dz}{d\tau} \frac{d\tau}{dt} \Rightarrow \frac{dz}{d\tau} = s(z) f(y(t))$$

Sistema berrian, mugimendua $z(\tau)$ funtziok deskribatzen du: z aldagai berria τ aldagai askearen menpekoa da.

8.4.1. Adibidea.

Esperimentu honetan, IRK metodoan denbora birparametrizazioa modu errezean aplika daitekeela erakutsi nahi dugu. N9-Body probleman, merkurio ezentriziitate handiena duen planeta da: merkurio araberako denbora birparametrizazioa planteatuko dugu.

$$s(q) = r_{10}^{3/2} \tag{8.9}$$

$$r_{10} = \|q_1 - q_0\|_2 \tag{8.10}$$

non $q_1 = (q_{1x}, q_{1y}, q_{1z})$ merkurio plantearen kokapena eta $q_0 = (q_{0x}, q_{0y}, q_{0z})$ eguzkiaren kokapena den.

8.5. Atalen hasieraketa.

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,

$$\dot{z} = \epsilon r(z, t).$$

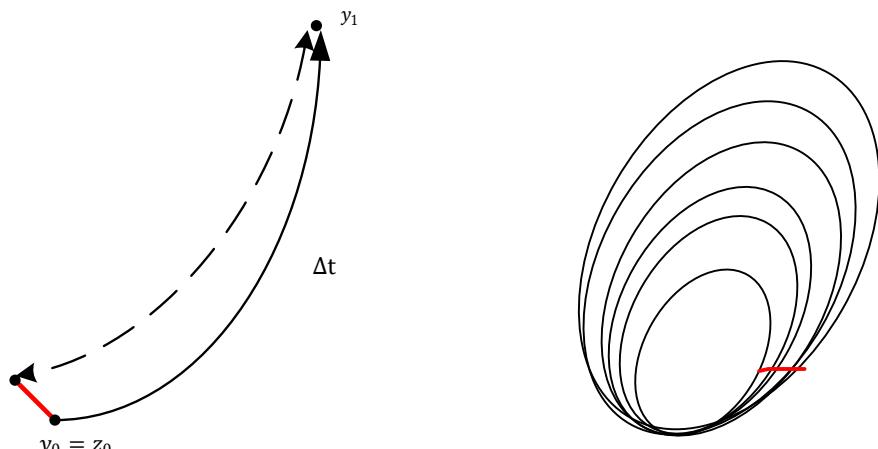
Idea hau bi modutara aplika daiteke,

1. Gauss implizituaren 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.

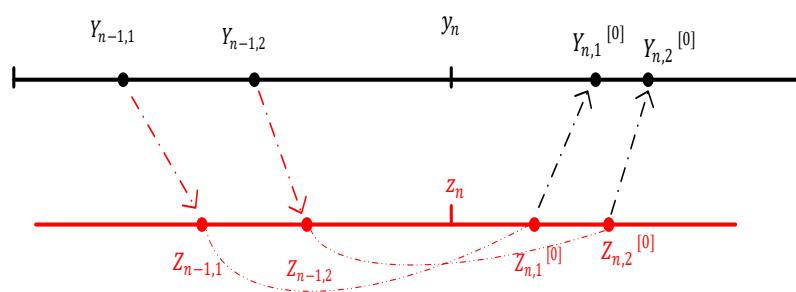
8.6. Laburpena.



(a) Atalen hasieraketa1.

(b) Atalen hasieraketa2.

8.3. Irudia: (a) irudian, (b)



8.4. Irudia: Atalen hasieraketa3.

**III. Atala
Ondorioak.**

9. Kapitulua

Eztabaida.

9.1. Sarrera.

9.2. IRK Puntu finkoa.

9.3. IRK Newton.

$(I_s \otimes I_d - h A \otimes J) \Delta Y = r$ ekuazio sistema askatzeko, Hairer-en implementazio estandarrean, A matrizearen,

$$A = P^{-1}DP, \quad D = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & & \sigma_s \end{bmatrix}$$

diagonalizazioa proposatzen da. D matrizeak, balio propio konplexuak ditu. Zenbaki konplexuekin lana ez bada egin nahi, zenbaki errealeko deskonposaketa baliokidea,

$$A = Q^{-1}RQ, \quad R = \begin{bmatrix} \sigma_{1A} & \sigma_{1B} & 0 & 0 & \cdots & 0 & 0 \\ \sigma_{1C} & \sigma_{1D} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma_{2A} & \sigma_{2B} & \cdots & 0 & 0 \\ 0 & 0 & \sigma_{2C} & \sigma_{2D} & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \sigma_{sA} & \sigma_{sB} \\ 0 & 0 & 0 & 0 & \cdots & \sigma_{sC} & \sigma_{sD} \end{bmatrix}$$

Honen arabera, Hairer-en implementazioan,

- s bikoitia $\rightarrow (2d \times 2d)$ tamainako $[s/2]$ LU deskonposaketa.

- s bakoitia $\rightarrow (2d \times 2d)$ tamainako $(s + 1)/2$ LU deskonposaketa.

Gure implementazioan, \bar{A} matrizearen,

$$\bar{A} = P^{-1}DP$$

diagonalizatzen dugu eta D matrizeak, irudikari puruak ditu. Eta ondorioz, gure implementazioaren bertsio errealean,

$$\bar{A} = Q^{-1}RQ, \quad R = \begin{bmatrix} 0 & -\sigma_1 & 0 & 0 & \cdots & 0 & 0 \\ \sigma_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & -\sigma_2 & \cdots & 0 & 0 \\ 0 & 0 & \sigma_2 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & -\sigma_s \\ 0 & 0 & 0 & 0 & \cdots & \sigma_s & 0 \end{bmatrix}$$

zeroak agertzen dira. Honen arabera, gure implementazioan,

- s bikoitia $\rightarrow (d \times d)$ tamainako $[s/2] + 1$ LU deskonposaketa.
- s bakoitia $\rightarrow (d \times d)$ tamainako $(s + 1)/2$ LU deskonposaketa.

9.4. Laburpena.

10. Kapitulua

Konklusioak.

10.1. Sarrera.

New Worlds, New Horizons in Astronomy and Astrophysics. ISBN 978-0-309-15802-2 | DOI 10.17226/12951 /.../PIC/dokumentazioa/bibliografia/nap/3-New Words, New Horizons-2016

Discovery Through the Power of Mathematics, Physics, and the Imagination

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. Examples range from the prediction that the chemical elements heavier than hydrogen and helium must have been created inside nuclear furnaces in the cores of stars, to the idea that the infant universe underwent a period of extremely rapid expansion called inflation, to the prediction of exotic objects like black holes, neutron stars, and white dwarfs, and the prediction that planets are a typical by- product of normal star formation. 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.

10.2. Laburpena.

IV. Atala

Eranskinak

A. Eranskina

Ekuazio garapenak.

A.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 integrazio 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,

$$b_{11} = (C - 1) \frac{a}{\|q\|}, \rightarrow b_{11} = -\frac{\sin^2(\Delta E)}{1 + \cos(\Delta E)} \frac{a}{\|q\|}$$

A.2. Koordenatu sistemak.

Sarrera.

Lehenik koordenatu barizentrikoei $q_i, p_i \in \mathbb{R}^3$, $i = 0, \dots, N$ dagokien Hamilton-darra gogoratuko dugu,

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

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

$$H = H_K + T_1 + U_1.$$

1. H_K .

$$H_K = \sum_{i=1}^N \left(\frac{\|P_i\|^2}{2\mu_i} - \frac{Gm_0m_i}{\|Q_i\|} \right), \quad \mu_i = \frac{m_0m_i}{(m_0 + m_i)}.$$

2. T_1 .

$$T_1 = \frac{1}{m_0} \left(\sum_{0 < i < j \leq N}^N P_i P_j \right).$$

3. U_1 .

$$U_1 = - \sum_{0 < i < j \leq N}^N \frac{Gm_i m_j}{\|Q_i - Q_j\|}.$$

Ekuazio differentzialak.

Hamiltondar bakoitza independientekei konsideratuta dagokio ekuazio differentzialak lortzen ditugu:

1. H_K .

$$\begin{aligned}\dot{Q}_i &= \nabla_p H_k \Rightarrow \dot{Q}_i = P_i \left(\frac{m_0 + m_i}{m_0 m_i} \right), \\ \dot{P}_i &= -\nabla_q H_k \Rightarrow \dot{P}_i = -\frac{G m_0 m_i}{\|Q_i\|^3} Q_i, \quad i = 1, \dots, N.\end{aligned}$$

2. T_1 .

$$\begin{aligned}\dot{Q}_i &= \nabla_p T_1 \Rightarrow \dot{Q}_i = \sum_{j \neq i, j=1}^N \frac{P_j}{m_0}, \\ \dot{P}_i &= -\nabla_q T_1 \Rightarrow \dot{P}_i = 0, \quad i = 1, \dots, N.\end{aligned}$$

3. U_1 .

$$\begin{aligned}\dot{Q}_i &= \nabla_p U_1 \Rightarrow \dot{Q}_i = 0, \\ \dot{P}_i &= -\nabla_q U_1 \Rightarrow \dot{P}_i = \sum_{j \neq i, j=1}^N \left(\frac{-G m_i m_j}{\|Q_i - Q_j\|^3} (Q_i - Q_j) \right), \quad i = 1, \dots, N.\end{aligned}$$

Azkenik, $V_i = P_i/\mu_i$ aplikatuta integrazioan erabiliko ditugun ekuazioak laburtuko 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 barizentrikoetara bihurtuko dugu . Hauek dira koordenatu heliozentrikoetatik abiatuta (Q_i, V_i) , koordenatu barizentrikoak (q_i, v_i) kalkulatzeko ekuazioak,

$$1. \ q_i, \ i = 0, \dots, N.$$

$$\begin{aligned} q_0 &= - \sum_{i=1}^M \frac{m_i Q_i}{M}, \quad M = \sum_{i=0}^N m_i, \\ q_i &= q_0 + Q_i, \quad i = 1, \dots, N. \end{aligned}$$

$$2. \ v_i, \ i = 0, \dots, N.$$

$$\begin{aligned} v_i &= \frac{m_0}{m_0 + m_i} V_i, \quad i = 1, \dots, N \\ P_0 = \sum_{i=0}^N p_i &= \sum_{i=0}^N m_i v_i = 0 \Rightarrow m_0 v_0 + \sum_{i=1}^N m_i v_i = 0 \Rightarrow v_0 = -\frac{1}{m_0} \sum_{i=1}^N m_i v_i. \end{aligned}$$

Koordenatu Jacobiarrak.

Koordenatu barizentrikoetatik abiatuta eta aldagai aldaketa bat aplikatuz ekuazioak koordenatu Jacobiaren $Q_i, P_i \in \mathbb{R}^3$, $i = 0, \dots, N$ arabera berridatziko ditugu.

Aldagai aldaketa.

Lehenik honako aldagai aldaketa aplikatuko dugu,

$$\begin{aligned} Q_0 &= (m_0 q_0 + \dots + m_n q_n) / \eta_N, \quad Q_i = q_i - \left(\sum_{j=0}^{i-1} m_j q_j \right) / \eta_{i-1} \\ P_0 &= \sum_{i=0}^N p_i, \quad P_i = \left(\eta_{i-1} p_i - m_i \sum_{j=0}^{i-1} p_j \right) / \eta_i, \quad i = 1, \dots, N. \end{aligned}$$

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,

$$H = H_K + H_I.$$

1. H_K .

$$H_K = \sum_{i=1}^N \left(\frac{\|P_i\|^2}{2m'_i} - \frac{\mu'_i}{\|Q_i\|} \right).$$

2. H_I .

$$H_I = \left(\frac{\mu'_i}{Q_i} - \frac{Gm_0m_i}{q_i} \right) - \sum_{0 < i < j \leq N}^N \frac{Gm_im_j}{\|Q_i - Q_j\|}.$$

A.3. Newton eraginkorraren garapena.

Formulazio estandarrean honako ekuazio sistema askatzeko metodoa proposatzen da,

$$(I_s \otimes I_d - h A \otimes J) \Delta Y = r. \quad (\text{A.2})$$

Lehenengo modu orokorrean eta ondoren, metodoa simetrikoa dela kontutan harturik garapenaren zehaztasunak emango ditugu.

Kasu orokorra

Honako ekuazio sistemari,

$$(I_s \otimes I_d - h \bar{A} \otimes J) \Delta Y - \frac{1}{2}(e_s \otimes I_d) \Delta z = r, \quad (\text{A.3})$$

$$(-he_s^T B \otimes J) \Delta Y + \Delta z = 0, \quad (\text{A.4})$$

aldagai aldaketa hau, aplikatuko doigu.

$$\Delta Y = (Q \otimes I_d) W. \quad (\text{A.5})$$

1. Lehen urratsa.

Ekuazioa sistemaren lehen ekuazioari (A.3), aldagai aldaketa (A.5) aplikatu eta $(Q^{-1} \otimes I_d)$ gaia ezkerretik biderkatuz,

$$\begin{aligned} (Q^{-1} \otimes I_d) (I_s \otimes I_d - h \bar{A} \otimes J) (Q \otimes I_d) W \\ - (Q^{-1} \otimes I_d) \left(\frac{1}{2} e_s \otimes I_d \right) \Delta z = (Q^{-1} \otimes I_d) r. \end{aligned}$$

Eta garatuz,

$$(I_s \otimes I_d - hQ^{-1}\bar{A}Q \otimes J) W - \frac{1}{2}(Q^{-1}e_s \otimes I_d)\Delta z = (Q^{-1} \otimes I_d)r.$$

2. Bigarren urratsa.

Metodoa simetrikoa bada,

$$Q^{-1}\bar{A}Q = \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \quad (\text{A.6})$$

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,

$$I_s \otimes I_d = \begin{pmatrix} I_m \otimes I_d & 0 \\ 0 & I_{s-m} \otimes I_d \end{pmatrix} \quad (\text{A.7})$$

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

3. Hirugarren urratsa.

Bigarren ekuazioari (A.3) aldagai aldaketa (A.5) aplikatuz,

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

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. Eranskina

Kodea

Kodeari buruzko hainbat ohar emateko.

B.1. Gauss metodoa

Metodoaren koefizienteak.

GaussCollocationCoefficients.nb funtzioa.

Atalen hasieraketa koefizienteak.

B.2. Zientzia konputazioa.

FMA.

Nola jakin gure *linux* konputagailu batek *FMA* instrukzioak dituen ala ez ? Honako 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
```

OpenMP

Adibidea2. *Reduction* direktiba $+, -, *, \min, \max$ funtziokin erabili daiteke. Direktiba honen adibidea emateko, trapezio erregelaren bidezko zenbakizko inte-

```

GaussCollocationCoefficients
[s_Integer, doi_, a_Symbol, b_Symbol, c_Symbol] :=

Module
[{f, g, ff, glist, B, A},

Do[c[i] = N[(Root[LegendreP[s, #] &, i] + 1) / 2, doi]
 // Simplify, {i, s}
];

ff = Collect[InterpolatingPolynomial
[Table[{c[i], f[i]}, {i, s}], x], f[_]];

glist = Table[g[i] = Collect[ff, f[_],
Simplify[Integrate[#, x]] &], {i, 1, s}];
yy = Collect[ff, f[_], Simplify[Integrate[#, x]] &];

B = Table[b[i] = D[f[i], x], {i, 1, s}];
A = Table[a[i, j] = D[g[i], f[j]], {i, 1, s}, {j, 1, s}];

{Array[c, s], B, A}

]

```

B.1. Irudia: Code Gauss.

grazioaren implementazioa erakutsiko dugu.

$$approx = h * (f(x_0)/2 + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + f(x_n))$$

```
#     include <omp.h>

     int thread_count=2;

     h= (b-a)/n;
     approx = (f(a)+f(b))/2.0;

# pragma omp parallel for num_threads(thread_count) \
reduction(+: approx)
     for (i = 0; i<n; i++) approx+= f(a+i*h);

     approx=h*approx;
```

Makefile adibideak.

Adibidea1. Adibide simple baten bidez azalduko dugu bere erabilpena. Hauxe da, *Makefile*-aren oinarrizko elementua,

```
helburua: dependentziak
>TAB> helburua lortzeko aginduak
```

Demagun aplikazioa bat hiru fitxategietan banatuta dugula,

```
/* file: main.c */
void main()
{
    printf("Main program");
    sub1();
    sub2();
}
```

```
/* file: sub1.c */
void sub1()
{
    printf("sub1");
}
```

```
/* file: sub2.c */
void sub2()
{
    printf("sub2");
}
```

Konpilazioa automatizatzeko *make* fitxategia,

```
main.exe: main.o sub1.o sub2.o
        gcc main.o sub1.o sub2.o -o main.exe
main.o: main.c
        gcc -c main.c
sub1.o: sub1.c
```

```
gcc -c sub1.c
sub2.o: sub2.c
    gcc -c sub2.c
```

Eta erabilpena,

```
$ make main.exe
gcc -c main.c
gcc -c sub1.c
gcc -c sub2.c
gcc main.o sub1.o sub2.o -o main.exe
```

Adibidea2. Aurreko adibidea, modu egokiagoan idatzitako *make* fitxategia,

```
CC = /usr/bin/gcc
FLAGS=-O2 -Wall -std=c99 -fno-common
OBJECTS= main.o sub1.o sub2.o
.PHONY: clean help

main.exe: $(OBJECTS)
    $(CC) $(OBJECTS) -o main.exe

%.o: %.c
    $(CC) $(FLAGS) -c $<

clean:
    rm -f $(OBJECTS) main.exe

help:
    @echo "Valid targets:"
    @echo " main.exe"
    @echo " main.o"
    @echo " sub1.o"
    @echo " sub2.o"
    @echo " clean"
```

B.3. Problemak

Pendulu bikoitza

Mathematican, DoublePendulum.m eta DoublePendulumSTIFF.m paketetan, hurrenez-hurren pendulu bikoitz arruntaren eta pendulu bikoitz zurrunaren honako funtzioen implementazioak garatu ditugu:

1. Hamiltondarra: DoublePendulumHam eta DoublePendulumSTIFFHam.

2. EDA: DoublePendulumODE eta DoublePendulumSTIFFODE.
3. Jakobiarra: DoublePendulumJAC eta DoublePendulumSTIFFJAC.

C-lengoaiako implementazio baliokidea, GaussUserProblem.c fitxategian implementatu ditugu. Pendulu bikoitz arruntaren problema $problem = 5$ gisa (ode5, jac5, ham5) eta pendulu bikoitz zurrunaren problema $problem = 6$ gisa (ode6, jac6, ham6) izendatu dugu.

N-body problema

Mathematicako NBodyProblem.m paketean honako funtzioak garatu ditugu.

1. Hamiltondarra: NBodyHAM.
2. EDA: NBodyODE.
3. Jakobiarra: Ez dut garatu.

C-lengoaiako implementazioa:

1. Hamiltondarra: HamNBody().
2. EDA: OdeNbody().
3. Jakobiarra: JacNBody().

B.4. Fortran kodeak

Haireren konposizio metodoaren Fortran kodearen C-lengoaiako itzulpena egin dugu. Konposizio metodoaren azalpenak liburuko [?] II.4 eta V.3 ataletan ematen dira. *GNI-Comp* izeneko kdea eskuragarri dago [?] helbidean.

B.5. IRK-Puntu finkoa

B.6. IRK-Newton

B.6.1. IRK-Newton koefizienteak.

GaussCoefficients(s).nb mathematicako fntzioaren bidez aurrekalkulatzen ditugu koefizienteak.

B.6.2. IRK-Newton eraginkorra.

Algoritmoaren hainbat zehaztapen emango ditugu,

1. LAPACK.

LU deskonposaketa egiteko funtzioa,

```
GETRF(LAPACK_ROW_MAJOR, n, m, MM, lda, ipiv);
```

Ekuazio sistemaren ebazpena (*Solve*) egiteko,

```
GETRS(LAPACK_ROW_MAJOR, trans ,n ,nrhs ,MM, lda ,ipiv ,f1 ,ldb);
```

B.7. Eguzki-sistema

C. Eranskina

Notazioa.

C.1. Notazioa.

C.2. Hitz-zerrenda.

C.1. Taula: Notazioa.

Notazioa	Esanahia	Adibidea
\mathbb{R}	Zenbaki errealen multzoa	
\mathbb{F}	Koma-higorreko zenbakien multzoa	
$\mathbb{R}^n, \mathbb{R}^{m \times n}$	n dimentsioko bektore eta $m \times n$ dimentsioko matrizeak	
\approx	Hurbilpena	$y_n \approx y(t_n)$
$fl()$	Koma-higikorreko balioa esleitzen duen funtzioa	$\tilde{y} = fl(y)$
\otimes	Biderketa tensoriala	
$\lceil \dots \rceil$	Round up to nearest integer	$\lceil 2.4 \rceil = 3$
$\lfloor \dots \rfloor$	Round down to the nearest integer	$\lfloor 2.6 \rfloor = 2$
$Const$	Balio konstantea	$H(p(t), q(t)) = Const$

C.2. 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		

D. Eranskina

Nire-Argibideak

Niretzako dokumentatutako argibide batzuk. Ez ditut tesian sartu behar.

D.1. IRK-Newton.

Newton sinplifikatuaren iterazioa.

Newton sinplifikatuaren iterazioan formulazio berrian ekuazio honen jatorria:

$$\begin{aligned} \text{Askatu } \Delta L_i^{[k]} \\ \Delta L_i^{[k]} - hb_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s. \end{aligned}$$

Frogapena. Abiapuntua.

$$\begin{aligned} \Delta L_i &= L_i - L_i^{[k]}, \\ L_i - hb_i f(y_n + \sum_{j=1}^s \mu_{ij} L_j) &= 0 \end{aligned}$$

Honako garapena egingo dugu.

$$L_i - hb_i f(y_n + \sum_{j=1}^s \mu_{ij} L_j) = L_i^{[k]} + \Delta L_i - hb_i f(y_n + \sum_{j=1}^s \mu_{ij} (L_j^{[k]} + \Delta L_j)).$$

Eta $f(y_n + \sum_{j=1}^s \mu_{ij}(L_j^{[k]} + \Delta L_j))$ linealizatuz,

$$\begin{aligned} &\approx L_i^{[k]} + \Delta L_i - hb_i f(y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k]}) - hb_i f'(y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k]})(\sum_{j=1}^s \mu_{ij} \Delta L_j) = \\ &\Delta L_i - hb_i J_i (\sum_{j=1}^s \mu_{ij} \Delta L_j) = -L_i^{[k]} + hb_i f(y_n + \sum_{j=1}^s \mu_{ij} L_j) \\ \text{non } J_i &= f'(y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k]}) = f'(Y_i). \end{aligned}$$

Bibliografia

- [1] Sverre Aarseth, Christopher Tout, and Rosemary Mardling. *The Cambridge n-body lectures*, volume 760. Springer, 2008.
- [2] 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.
- [3] 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.
- [4] Pavan Balaji. *Programming models for parallel computing*. MIT Press, 2015.
- [5] Josh Barnes and Piet Hut. A hierarchical $O(n \log n)$ force-calculation algorithm. *nature*, 324(6096):446–449, 1986.
- [6] André Berger. *A brief history of the astronomical theories of paleoclimates*. Springer, 2012.
- [7] Gregory Beylkin and Kristian Sandberg. Ode solvers using band-limited approximations. *Journal of Computational Physics*, 265:156–171, 2014.
- [8] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *arXiv preprint arXiv:1411.1607*, 2014.
- [9] Theodore A Bickart. An efficient solution process for implicit runge–kutta methods. *SIAM Journal on Numerical Analysis*, 14(6):1022–1027, 1977.

- [10] 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.
- [11] 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.
- [12] Dirk Brouwer. [On the accumulation of errors in numerical integration](#). *The Astronomical Journal*, 46:149–153, 1937.
- [13] 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.
- [14] VA Brumberg. [Celestial mechanics: Past, present, future](#). *Solar System Research*, 47(5):347–358, 2013.
- [15] John C Butcher. [On the implementation of implicit runge-kutta methods](#). *BIT Numerical Mathematics*, 16(3):237–240, 1976.
- [16] John Charles Butcher. [Numerical Methods for Ordinary Differential Equations](#). Second edition, Wiley, 2008.
- [17] 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.
- [18] 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.
- [19] Robert M Corless and Nicolas Fillion. [A graduate introduction to numerical methods](#). *AMC*, 10:12, 2013.
- [20] John Danby. Fundamentals of celestial mechanics. *Richmond: Willman-Bell, c1992, 2nd ed.*, 1, 1992.
- [21] Theodorus Jozef Dekker. [A floating-point technique for extending the available precision](#). *Numerische Mathematik*, 18(3):224–242, 1971.
- [22] DumitruN. Deleanu. [Fast detection of chaotic or regular behavior of double pendulum system: application of the fast norm vector indicator method](#).

- [23] 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.
- [24] Victor Eijkhout. [Introduction to High Performance Scientific Computing](#). lulu.com, 2011. Also available for download from <http://www.tacc.utexas.edu/~eijkhout/istc/istc.html>. ISBN 978-1-257-99254-6.
- [25] 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.
- [26] Kang Feng and Mengzhao Qin. [Symplectic geometric algorithms for hamiltonian systems](#). Springer, 2010.
- [27] A Fienga, H Manche, J Laskar, and Mickael Gastineau. [Inpop06: a new numerical planetary ephemeris](#). *Astronomy & Astrophysics*, 477(1):315–327, 2008.
- [28] 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.
- [29] Toshio Fukushima. [Reduction of round-off error in symplectic integrators](#). *The Astronomical Journal*, 121(3):1768, 2001.
- [30] 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.
- [31] E Hairer and G Wanner. [Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems](#). Springer, Berlin, 1996.
- [32] Ernst Hairer, Christian Lubich, and Gerhard Wanner. [Geometric numerical integration: structure-preserving algorithms for ordinary differential equations](#), volume 31. Springer Science & Business Media, 2006.
- [33] 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.
- [34] Wayne B Hayes. [Is the outer solar system chaotic?](#) *Nature Physics*, 3(10):689–691, 2007.

- [35] 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.
- [36] Nicholas J Higham. [Accuracy and stability of numerical algorithms](#). Siam, 2002.
- [37] Nicholas J Higham. [Programming languages: An applied mathematics view](#). 2015.
- [38] Leslie Hogben. [Handbook of linear algebra](#). Chapman and Hall/CRC, 2013.
- [39] 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.
- [40] Takashi Ito and Kiyotaka Tanikawa. Trends in 20th century celestial mechanics. *Publ. Natl. Astron. Obs. Jpn*, 9:55–112, 2007.
- [41] Laurent O. Jay. Preconditioning of implicit runge-kutta methods. *Scalable Computing: Practice and Experience*, 10, 2009.
- [42] MP Calvo JM Sanz-Serna. [Numerical Hamiltonian problems](#). Chapman and Hall, 1994.
- [43] 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.
- [44] W Kahan. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.
- [45] 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.
- [46] 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.
- [47] D Knuth. The art of computer programming: Vol 2/seminumerical algorithms, 1969.

- [48] 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.
- [49] MP Laburta. **Construction of starting algorithms for the rk-gauss methods.** *Journal of computational and applied mathematics*, 90(2):239–261, 1998.
- [50] Jacques Laskar. **The limits of earth orbital calculations for geological time-scale use.** *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 357(1757):1735–1759, 1999.
- [51] 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.
- [52] 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.
- [53] Jacques Laskar and Philippe Robutel. **High order symplectic integrators for perturbed hamiltonian systems.** *Celestial Mechanics and Dynamical Astronomy*, 80(1):39–62, 2001.
- [54] Benedict Leimkuhler and Sebastian Reich. *Simulating hamiltonian dynamics*, volume 14. Cambridge University Press, 2004.
- [55] Harold F Levison and Martin J Duncan. **The long-term dynamical behavior of short-period comets.** *Icarus*, 108(1):18–36, 1994.
- [56] 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.
- [57] 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.
- [58] Robert I McLachlan. **Composition methods in the presence of small parameters.** *BIT Numerical Mathematics*, 35(2):258–268, 1995.
- [59] Robert I McLachlan and Pau Atela. **The accuracy of symplectic integrators.** *Nonlinearity*, 5(2):541, 1992.

- [60] A Morbidelli. *Modern integrations of solar system dynamics*. *Annual Review of Earth and Planetary Sciences*, 30(1):89–112, 2002.
- [61] Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiod, Nathalie Revol, Damien Stehlé, and Serge Torres. *Handbook of floating-point arithmetic*. Springer Science & Business Media, 2009.
- [62] Nobelprize.org. *The nobel prize in chemistry 2013*, May 2014.
- [63] Hans Olsson and Gustaf Sderlind. *The approximate runge-kutta computational process*. *BIT Numerical Mathematics*, 40(2):351–373, 2000.
- [64] Michael L Overton. *Numerical computing with IEEE floating point arithmetic*. Siam, 2001.
- [65] Peter Pacheco. *An introduction to parallel programming*. Elsevier, 2011.
- [66] EV Pitjeva and NP Pitjev. *Development of planetary ephemerides epm and their applications*. *Celestial Mechanics and Dynamical Astronomy*, 119(3-4):237–256, 2014.
- [67] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
- [68] Prasenjit Saha and Scott Tremaine. *Long-term planetary integration with individual time steps*. *arXiv preprint astro-ph/9403057*, 1994.
- [69] Jesús María Sanz-Serna. *Symplectic integrators for hamiltonian problems: an overview*. *Acta numerica*, 1:243–286, 1992.
- [70] 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.
- [71] Mark Sofroniou and Giulia Spaletta. *Derivation of symmetric composition constants for symmetric integrators*. *Optimization Methods and Software*, 20(4-5):597–613, 2005.
- [72] Pat H Sterbenz. *Floating-point computation*. Prentice Hall, 1973.
- [73] Gerald J Sussman and Jack Wisdom. *Chaotic evolution of the solar system*. Technical report, DTIC Document, 1992.

- [74] 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 Plumley, et al. [Best practices for scientific computing](#). *PLoS Biol*, 12(1):e1001745, 2014.
- [75] 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.
- [76] Jack Wisdom and Matthew Holman. [Symplectic maps for the n-body problem](#). *The Astronomical Journal*, 102:1528–1538, 1991.
- [77] Wolfram Research, Inc. [Mathematica](#).
- [78] Dexuan Xie. A new numerical algorithm for efficiently implementing implicit runge-kutta methods. *Department of Mathematical Sciences. University of Wisconsin, Milwaukee, Wisconsin, USA*, 2009.
- [79] 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.