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

by

Mikel Antonana Otano

Informatika Fakultatea
Euskal-Herriko Unibertsitatea
Donostia

©Mikel Antonana 2017

Gaien Aurkibidea

I. Introduction	3
1. Sarrera.	5
2. Zenbakizko Integratziale Simplektikoak.	13
3. Problemak.	41
4. Koma Higikorreko Aritmetika.	57
5. Zientzia konputazioa.	67
II. Ekarpenak.	87
6. IRK: Puntu-Finkoa.	89
7. IRK: Newton.	127
8. IRK: Eguzki-sistema.	149
III. Ondorioak.	167
9. Eztabaida.	169
10. Konklusioak.	171
IV. Eranskinak	173
A. Ekuazio garapenak.	175
B. Kodea	183

C. Notazioa.	189
D. Nire-Argibideak	193
Bibliografia	195

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 Integratziale Simplektikoak.	13
2.1. Sarrera.	13
2.1.1. Zenbakizko integrazio metodoak.	13
2.1.2. Sistema-Hamiltondarra.	18
2.1.3. Metodo simplektikoak.	20
2.2. Runge-Kutta metodo simplektikoak.	24
2.2.1. Runge-Kutta metodoak.	24
2.2.2. Gauss metodoa.	26
2.2.3. IRK implementazioa.	29
2.3. Konposizio eta Splitting metodoak.	33
2.3.1. Konposizio metodoak.	33
2.3.2. Splitting metodoak.	35
2.3.3. Eguzki-sistemari egokitutako splitting metodoak.	37
2.4. Laburpena.	39
3. Problemak.	41
3.1. Sarrera.	41
3.2. Pendulu bikoitza.	42
3.3. N-Body problema.	45
3.4. Eguzki-sistema.	46
3.5. Laburpena.	56

4. Koma Higikorreko Aritmetika.	57
4.1. Sarrera	57
4.2. <i>IEEE-754</i> estandarra	57
4.3. Biribiltze errorea	60
4.4. Biribiltze errorearen gutxitzeko teknikak	63
4.5. Laburpena	65
5. Zientzia konputazioa.	67
5.1. Sarrera	67
5.2. Eraginkortasuna	68
5.3. Hardwarea	71
5.4. Programazio lengoaiak	76
5.5. Algebra lineal dentsorako liburutegiak	79
5.6. Konpiladorea	85
5.7. Laburpena	86
II. Ekarpenak.	87
6. IRK: Puntu-Finkoa.	89
6.1. Sarrera	89
6.2. Hairer-en implementazioa	89
6.3. Gure implementazioa	92
6.3.1. Metodoaren birformulazioa (1.proposamena)	92
6.3.2. Geratze irizpidea (2.proposamena)	94
6.3.3. Biribiltze errorea gutxitzeko teknikak (3.proposamena)	96
6.3.4. Biribiltze errorearen estimazioa (4.proposamena)	98
6.3.5. Algoritmoa	99
6.4. Implementazio estadarraren xehetasunak	102
6.5. Esperimentuak	104
6.5.1. Sarrera	104
6.5.2. Energia errore jatorriak	105
6.5.3. Errore azterketa estadistikoa	106
6.5.4. Round-off estimation	108
6.6. zaharra	113
6.6.1. Integrazio motak	113
6.6.2. N-Body problema	122
6.6.3. Ondorioak	122
6.7. Laburpena	123

7. IRK: Newton.	127
7.1. Sarrera.	127
7.2. IRK-Newton estandarra.	129
7.3. IRK-Newton eraginkorra.	132
7.4. IRK-Newton estandarra (formulazio berria).	139
7.5. IRK-Newton eraginkorra (formulazio berria).	142
7.6. IRK Sasi-Newton.	146
7.7. Laburpena.	147
8. IRK: Eguzki-sistema.	149
8.1. Sarrera.	149
8.2. Eguzki-sistemaren integraziorako metodoak (review).	149
8.2.1. Efemerideak.	149
8.2.2. Eguzki-sistemaren integrazio luzeak.	152
8.3. Gure implementazioa.	155
8.4. Denbora birparametrizazioa.	163
8.4.1. Adibidea.	164
8.5. Atalen hasieraketa.	164
8.6. Laburpena.	165
III. Ondorioak.	167
9. Eztabaida.	169
9.1. Sarrera.	169
9.2. IRK Puntu finkoa.	169
9.3. IRK Newton.	169
9.4. Laburpena.	170
10. Konklusioak.	171
10.1. Sarrera.	171
10.2. Laburpena.	171
IV. Eranskinak	173
A. Ekuazio garapenak.	175
A.1. Kepler hasierako baliodun problema.	175
A.2. Koordenatu sistemak.	178
A.3. Newton eraginkorraren garapena.	181

B. Kodea	183
B.1. Gauss metodoa	183
B.2. Zientzia konputazioa	183
B.3. Problemak	186
B.4. Fortran kodeak	187
B.5. IRK-Puntu finkoa	188
B.6. IRK-Newton	188
B.6.1. IRK-Newton koefizienteak.	188
B.6.2. IRK-Newton eraginkorra.	188
B.7. Eguzki-sistema	188
C. Notazioa.	189
C.1. Notazioa.	189
C.2. Hitz-zerrenda.	189
D. Nire-Argibideak	193
D.1. IRK-Newton.	193
Bibliografia	195

Itxaropena ez 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 funtsezko galdera batzuk ezagutu ditugu eta berriki, Karplusen taldeak 2013. urteko kimika Nobel saria [57] 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 [25] edo espazioko satelite artifizialen kokapenen [6] 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 [36] izan dateke eta kolisio gertuko egoeren ondorioz, kalkulutan 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, Sussmanek eta Wisdomek [67, 1993] eguzki sistemaren 100 milioiko integrazioarekin, planeten mugimendua kaotikoa zela baiezttatu zuten. Aldi berean, paleoklimatologi-zientziak orain milioika urte gertatutako klima zikloak (epel, hotz eta glaziazio aroak) azaltzeko, lurrauen orbitan izandako aldaketaren eraginez gertatu zirela azaltzen duen teoria (Milankovitch 1941) [5] baieztatzeko, planeten orbiten efemeride zehatzak beharrezkoak 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, Laskarrek [48, 2010] bere azken integrazioa burutzeko 18 hilabete behar izan zituen. Azken urteotako konputagailu berrien arkitekturaren bilakaerak, algoritmo azkarren diseinua aldatu du: simulazioak azkartzeko algoritmoak paralelizazioan oinarritu behar dira. Azpimarratzeko da ere, integrazio luze hauen erronka handienetako bat, biribiltze errorearen metaketa kontu handiz egitea da, non errorearen edozein joera ekiditzen den [47].

1.2. Motibazioa.

Metodo simplektikoen artean, gehien erabiltzen direnak 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 gainkarga 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, baiezta-pen hau eztabaidegarria dela, eta praktikan metodo implizituetan gehiago sakondu behar dela iruditzen zaigu.

Lan honetan, Gauss zenbakizko integrazio metodo simplektikoaren azterketa egin dugu. Hainbat autorek (Hairer [29][30] eta Sanz Serna[38]) metodo honen

potentziala nabarmendu dute. Azken urtetan, espazioko satelite artifizialen arloan ere, Gauss integrazio metodo implizituarekiko interesa azaldu dute [9][6].

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 eraginkorrap direla. Baino metodo implizituk aldiz, Hamiltondar orokorreai 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, orden altuko Gauss metodoak existitzen direla eta hauek nahitaezkoak dira doitasun handiko integrazioak behar ditugunean.

Atal hau bukatzeko, Sanz Sernaren [63, 1992] hitzak berreskuratuko ditugu.

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

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 erakarpen eragina, eta beste zenbait indar ez-grabitacionalak ez dira kontutan hartu. Bestalde era honetako integrazioetan, gorputzen hasierako balio eta parametro zehatzak sateliteen bidez jasotako datu errealekin bat datozena egiaztatze prozesua ez dugu landu [47].

Zeintzuk dira eguzki-sistemaren problemaren ezaugarri bereziak? Batetik, bi gorputzen problemaren soluzio zehatza ezaguna da eta eguzki-sistemaren gorputzen mugimenduaren konputazioaren oinarria. 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 biribitzte errorea gutxitzen duten teknika bereziak aplikatzea ezinbestekoa izaten da. Hala ere, integratio luzeetan maiz doitasun handian lan egiteko beharra azaltzen zaigu baina doitasun altuko aritmetikaren (128-bit) implementazioa software bidezkoa denez, oso motela da. 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*[4] eta *fast multipole method*[14] 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 finkoko 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 konbergentzi ezaugarri onak direla eta. Hirugarren fasean ...

Jarraian, atal bakoitzean egindako ekarprena nagusienak laburtuko ditugu:

1. IRK puntu finkoa.

Gauss metodoaren puntu finkoko implementazioaren azterketa sakon bat egin dugu eta horretarako, Hairer-en implementazioa [30] hartu dugu gure lanaren abiapuntua. Kalitatezkoa implementazio hau hobetzeko aukerak ikusi ditugu eta implementazio sendoago bat proposatu dugu. Gure ekarpenak 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 errezean finkatu daitezkeen.

(b) Geratze erizpide berria.

Orokorrean, Hairer-en implementazioaren puntu finkoko iterazioaren geratze erizpide zuzena dela ikusi dugu baina kasu batzuetan goizegi geratzen dela baiezta dugu. Arrazoia bi direla ikusirik, bere geratze erizpidea bi zentzutan garatu/zorroztu dugu. Lehenik, Hairer-en implementazioan, hobekunza neurtzeko ataletako differentziaren norma batean oinarritzen da. Normaren independientea den geratze erizpidea aplikatzea zuzenagoa da, eta horregaitik, ataletako edozein osagaiaren differentzia txikitzen den bitartean iterazioan jarraitzea finkatu dugu. Bigarrenik, iterazioen hobekunza ez du zertan beherakorra izan behar, eta okertzen diren tarteko iterazioak gerta daitezke. Arazo hau gainditzeko, iterazioren batean osagai guztien differentzia handitzea gertatzen denean, seguridadeko 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 eskeiniko zaio.

2. IRK Newton.

- (a) Ekarpen nagusia.

Newton metodoa, s -ataletako IRK metodoarekin d -dimentsioko ekuazio diferenzial sistemari aplikatzeko, urrats bakoitzean $sd \times sd$ tamaineko hainbat ekuazio sistema (iterazio bakoitzeko bat) askatu behar dira. Atal honetan, jatorrizko sd -dimentsioko ekuazio sistema, $(s+1)d$ sistema baliokide moduan berridatziko dugu eta sistema baliokidea, $d \times d$ tamaineko $[s/2] + 1$ matrize errealeen LU-deskonposaketa bidez askatuko dugu. Tamaina txikiko matrizeen LU-deskonposaketa azkarra denez, konputazionalki eraginkorragoa izatea espero dugu.

- (b) Doitasun laukoitzeko implementazioa.

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

3. IRK Eguzki-sistema.

Hirugarren urratsean, eguzki-sistemaren epe luzeko integrazioan arituko gara. Ekarpen handiena, atalen hasieraketa berri bat aplikatzea da alde Kepleriarraren 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) kapituluetan, zenbakizko integrazio simplektikoak eta zientzia konputazionalaren oinarriak azaldu ditugu. Bigarren kapituluaren lehen zatian, *Zenbakizko integratzaile simplektikoen* inguruko oinarrizko kontzeptuak azaldu ditugu. Kapitulu honen bigarren zatian, Gauss metodo estandarra deskribitu eta bere propietaten nagusienak eman ditugu. Kapitulu honen azken zatian, eguzki-sistemaren integraziorako metodo simplektiko eta esplizitu nagusienak laburtu ditugu. Hirugarren kapituluan, lan honetan zenbakizko esperimentuetan era-bili ditugun hasierako baliodun problemen zehaztasunak eman ditugu. Laugarren kapituluan koma-higikorreko aritmetika murgidu gara eta biribiltze errorearen inguruko gaiak argitu nahi izan ditugu. Bostgarren kapituluan, egungo konputazio zientziaren hardware eta software kontzeptu nagusienak ezagutarazi nahi izan ditugu.

Tesiaren (6 – 8) kapituluetan, gure implementazio berriak landu ditugu eta bakoitzaren konportamendua zenbazki esperimentuetan erakutsi dugu. Lehenik, 6. kapituluan Gauss metodoaren puntu finkoko iterazioaren implementazio berria eman dugu. Ondoren, 7. kapituluan Gauss metodoaren Newton iterazioaren implementazio eraginkorrik azaldu ditugu. 8. kapituluan, eguzki-sistemaren integraziorako implementazio deskribapena egin dugu.

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

Tesiaren bukaeran, hiru eranskinetan lanaren informazio osagarria bildu dugu. A-eranskinean, tesian zehar erabilitako hainbat 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.

Sarrera honen lehen zatian, zenbakizko integrazio metodoen inguruko oinarrizko definizioak eta kontzeptuak azalduko ditugu. Bigarren zatian sistema Hamilton-darrak eta metodo simplektikoen sarrera eman dugu. Sarrera honetako azalpenak ulergarriak emateko, zenbakizko metodo simpletan (Euler metodoak) oinarritu gara eta hauek, hainbat adibideekin lagunduko ditugu.

2.1.1. Zenbakizko integrazio metodoak.

Ekuazio differentzial arruntak (*ODE*), egoera aldaketak aztertzeko problemen eredu matematikoak dira. Zientziaren eta ingeniaritzaren arlo askotan azaltzen zaizkigu: planeten mugimendua astronomian, erreakzio kimikoak, molekulen dinamiken simulazioak, zirkuitu elektronikoak, populazio-hazkunde eta interakzioak biologian, ekonomi azterketak ...

Ekuazio differentzial arrunta, $y(t)$ soluzio ezezagunaren eta bere $\dot{y}(t)$ deribatuaren arteko erlazioa da. t aldagaiaik maiz denbora adierazten du eta $y(t) \in \mathbb{R}^d$ 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 differentzial arruntetarako hasierako baliodun problemetan (*IVP*), $y(t_0) = y_0 \in \mathbb{R}^d$ hasierako balio bat finkatzen da, eta $f(t, y)$ funtzia, $f : \mathbb{R}^{d+1} \rightarrow$

\mathbb{R}^d , t_0 ingurunean differentziagarria bada, orduan hasierako baliodun problema-ren soluzioa existitzen da eta bakarra da [29],

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

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 differentzial arrunten soluzioa analitikoa (funtzio ezagunen araberako soluzio zehatza) eta beraz, ia problemak gehienak zenbakizko integracio metodoen bidez ebatzen dira. Zenbakizko integracioan, soluzioaren hurbilpena,

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

$n = 1, 2, \dots$ 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 $(t_o, y_0), (t_1, y_1), \dots, (t_f, y_f)$ zenbakizko soluzioa definitzen du.

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

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

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

Oinarrizko kontzeptuak.

Jarraian zenbakizko integracioen 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 esleitzen dion funtzioari fluxua deitzen diogu. Izendatzeko φ_t notazioa erabiliko dugu,

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

2. Zenbakizko diskretizazioa.

y_n, y_{n-1}, \dots, y_0 balioak emanda, $y_{n+1} \approx y(t_{n+1})$ soluzioaren hurbilpena kalkulatzeko formulari *zenbakizko fluxua* deritzogu. Honako notazioa era-biliko 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 metodoa) 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. Lerro bakoitzak, hasierako balio ezberdin batzen zenbakizko soluzioa da. Marra bertikal lodi bakoitza errore lokala da eta urratsez-urrats propagatzen dena. Errore globala, t_n unean propagatutako errore lokalaren batura da.

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

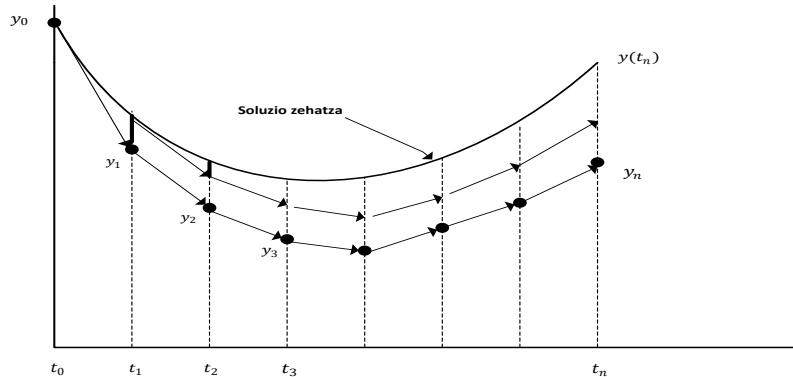
$$\|y_k - y(t_k)\| = \mathcal{O}(h^p), \quad h \rightarrow 0.$$

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,

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



2.1. Irudia: Errore propagazioa.

Euler metodo esplizitua.

Eulerrek 1768. urtean proposatutako zenbakizko metodoa da. Hasierako balio bat emanda (t_n, y_n) eta $h_n > 0$ urrets 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 metodoa esplizitua da eta urratsa emateko dagoen konputazio komplexutasuna 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. Orden altugoko metodoekin, doitasuna handiagoa lortuko dugu lan konputazional gutxiagorekin.

Euler metodo implizitua.

Eulerrek proposatutako beste metodo honetan funtsezko aldaketa dakar, y_{n+1} hurbilketa implizituki definitzen dela. 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 finko 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 finkoko 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. Newton metodoa.

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, Newton 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) $y_{n+1}^{[k+1]} = y_{n+1}^{[k]} + \Delta y_{n+1}^{[k]}.$

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

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

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

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

Puntu finkoko iterazioaren konbergentzia lineala da ($p = 1$) eta Newton iterazioaren konbergentzia koadratikoa da ($p = 2$). 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 implizitua aplikatzea esplizitua baino egokiago izan daiteke, eta hori erakusteko Germund Dahlquist-en (1963) adibidea azalduko dugu,

$$\dot{y} = \lambda y, \quad (2.4)$$

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

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

eta aldiz, metodo esplizituaren bidezkoa,

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

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

Euler implizitua, Euler esplizitua baino eraginkorra den ekuazio differentialei problema *zurrunkak* (*stiff*) esaten zaie. Problema *zurrunen* artean problema garrantzitsuak ditugu, esaterako eskala anitzeko sistemak.

2.1.2. Sistema-Hamiltondarra.

Hamiltondar sistemak, ekuazio differentzial mota garrantzitsu bat dira. $H(p, q)$ funtzio leunari dagokion *Hamiltondar sistema* osatzen duten 2d ekuazio diferen-

tzialak, 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 Hamiltondarren maiz egitura berezia du,

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

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

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

Eta dagokien ekuazio diferenzialak,

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

Adibidea. *Bi-gorputzen problema* 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 dieferentzialak,

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

Planetaren mugimendua orbita eliptiko bat da. Honako hasierako balioei dagoien 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 ezentrizitate ($0 \leq e < 1$) eta $P = 2\pi$ periodoa duen elipsea da.

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

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

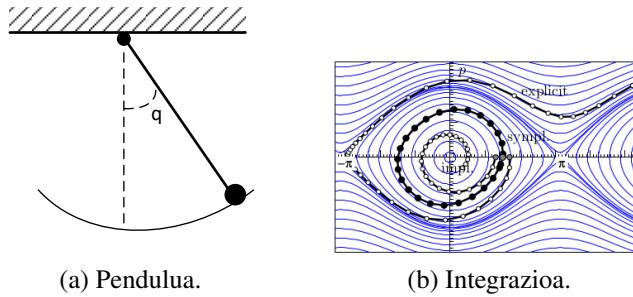
Adibidea. Eguzki-sistemaren probleman, Hamiltondarra modu honetan idatzi daiteke $H = H_k + H_I$, non alde nagusia H_K planeta bakoitzaren eguzki inguruko mugimendu kepleriarra den eta H_I aldiz, planeten arteko interakzioek eragiten duten perturbazio txikia.

2.1.3. Metodo simplektikoak.

Hamiltondar sistemen problemetarako, Euler esplizitua eta Euler implizitua ez dira zenbakizko metodo egokiak. Problema hauen propietate geometrikoak mantentzen dituzte integratziale bereziak beharrezkoak dira. Metodo simplektikoak sistema Hamiltondarrak integratzeko metodo egokiak dira, eta abantaila handiena, epe luzeko integrazioetan azaltzen dute.

Metodo simplektikoak bi taldeetan banatuko ditugu: metodo simplektiko implizituak eta esplizituak. Lehenengo taldean, Runge-Kutta metodo implizituak

(Gauss metodoak) ditugu. Metodo hauek zenbakizko metodo estandarrak dira eta ekuazio diferentzial orokorreai aplika daitezke. Bigarren taldean, konposizioan eta splitting-ean oinarritutako metodoak ditugu. Azken talde honetako metodoak, bakarrik Hamiltondar sistemiei aplika daitezke eta gainera, Hamiltondarak egitura berezia izan behar du baina aplika daitezkenean, oso eraginkorrak dira.



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

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

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

Ekuazio diferentzialak,

$$\dot{p} = -\sin(q), \quad \dot{q} = p. \quad (2.6)$$

Irudian (2.2. Irudia) ikusi daiteke, Euler esplizitua eta implizitua konportamendu koalitatiboki okerra dela. Euler simplektikoa ordea,

$$p_{n+1} = p_n - h \frac{\partial H}{\partial q}(p_{n+1}, q_n), \quad (2.7)$$

$$q_{n+1} = q_n + h \frac{\partial H}{\partial p}(p_{n+1}, q_n). \quad (2.8)$$

sistemaren energia kontserbatzen du integracio luzean zehar.

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 eredu eraikitze-rakoan hartzen diren hubilpenek edo datu ez ziurrek (uncertainty) eragiten duten baino errore maila txikiagoa suposatzen dutela.

Adibidea. Demagun, askatasun maila bakarra duen sistema Hamiltondar bana-garria $H = T(p) + V(q)$ Euler metodo simplektikoaren (2.7) 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 ψ_h^H eta zenbakizko integrazioa ϕ_h^H izendatzen baditugu,

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

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

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

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

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

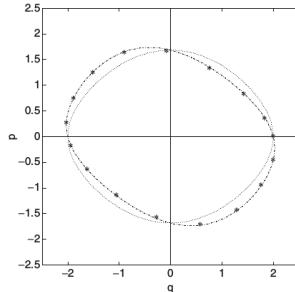
Taylor seriea garatuz, lortuko dugu S_2 sistema ($f = -\partial H/\partial q$, $g = \partial H/\partial p$),

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

non honako Hamiltondarrari dagokion,

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

Irudian (2.3. Irudia), pendulu problema (2.6) hasierako balio hauekin ($p(0) = 0$, $q(0) = 2$) erakutsi dugu. Izarrak, $h = 0.5$ urratsari dagokion zenbakizko soluzioa da. $H = \text{const}$ puntuko lerroak, penduluaren benetako soluzioa irudikatzen



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

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

Kasu orokorra, posible da S_p Hamiltondar sistema perturbatua eraikitzea, zeinakin Euler simplektikoa p ordenekoa den. Beraz, konklusio gisa esango dugu, Hamiltondar sistema bati aplikatzen diogun metodo simplektikoaren zenbakizko soluzioa, perturbatutako Hamiltodar sistema baten soluzio ia zehatza konsideratu behar dugula.

Propietate nagusiak. Metodo simplektikoen propietate hauek azpimarratuko ditugu,

1. Metodo simplektikoeik, energia oso ondo kontserbatzen dute.

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

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

Zenbakizko soluzioak, 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 aritmetikarekin, biribiltze errorearen eraginez energia errorea denboraren erro karratuaren arabera hasiko da,

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

non $k \in \mathbb{R}$.

2. Errore propagazio lineala.

Gehienentan, p eta q osagaien errorea $\mathcal{O}(h^p)$ da eta hazkundea, lineala. Oro har, integratziale arruntek errore hazkunde koadratikoa dute eta ondorioz, integrazio luzeetan konportamendu txarra azaltzen dute.

3. Metodo simplektiko garrantzitsuenak simetrikoak (reversible) dira.
4. Urrats luzera finkoa. Integratziale simplektikoetan, urrats luzera finkoa era-bili behar da metodoaren propietateak ez galtzeko. Metodo gehienak era-ginkorrik izateko, errore estimazio baten arabera integrazioan zehar urrats luzera egokitzen dute. Soluzioaren aldaketa azkarrak gertatzen diren uneetarako urratsa txikia, eta aldi, urratsa handia soluzioa leuna den uneetarako.

2.2. Runge-Kutta metodo simplektikoak.

Gauss metodoa, Runge-Kutta metodo interesgarria da. Metodo simplektikoa, in-plizitua da eta s-ataleko orden altueneko metodoa ($p = 2s$) da.

2.2.1. Runge-Kutta metodoak.

Runge-Kutta metodoak, urrats bakarreko ekuazio diferentzial arrunten zenbakizko integrazio metodoak dira. b_i , a_{ij} eta $c_i = \sum_{j=0}^s a_{ij}$ ($1 \leq i, j \leq s$) koefiziente errealek s-ataleko Runge-Kutta metodoa definitzen dute eta *Butcher* izeneko tau-lan moduan laburtu ohi dira,

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

Hasierako baliodun problema baten (2.2) $y(t)$ soluzioaren $y_n \approx y(t_n)$ hurbilpe-na 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, $(t_n + c_i h)$, $i = 1, \dots, s$ uneetarako 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, $p = 4$ ordenekoa dugu. Orden altuko ERK metodoak aurkitzea konplexua da, koefizienteek bete behar dituzten orden baldintzen kopurua esponentzialki hazten baitira. Esate baterako, orden altuko (ERK) metodo hauek aurkitu dira: $s = 11$ ataletako $p = 8$ ordeneko metodoa, $s = 17$ ataletako $p = 10$ ordeneko metodoa eta $s = 25$ ataletako $p = 12$ ordeneko metodoa.

2.1. Taula: Runge-Kutta metodo implizituak.

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

Orden altuko *IRK* metodoak, *ERK* metodoak baino modu errazagoan eraiki daitezke. *Butcher simplifikazio baldintzen* [13] arabera definitzen dira,

$$\begin{aligned} B(p) : \quad \sum_{i=1}^s b_i c_i^{q-1} &= \frac{1}{q}, & q &= 1, \dots, p. \\ C(\eta) : \quad \sum_{j=1}^s a_{ij} c_j^{q-1} &= \frac{c_i^q}{q}, & i &= 1, \dots, s, \quad q = 1, \dots, \eta. \\ D(\zeta) : \quad \sum_{i=1}^s b_i c_i^{q-1} a_{ij} &= \frac{b_j}{q} (1 - c_j^q), & j &= 1, \dots, s, \quad q = 1, \dots, \zeta. \end{aligned}$$

Taula honetan (2.1. Taula), *IRK* metodo nagusienak laburtu ditugu.

2.2.2. Gauss metodoa.

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

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 abantaila da, zenbakizko soluzioa diskretizazio puntuetaez ezik, polinomio interpolatzale batek modu jarraian emandako soluzioa adierazten duela.

Guillou and Soule 1969, Wright 1970. Kolokazio metodoaren definizioa eta jarraian emandako moduan kalkulatutako s-ataleko Runge-Kutta metodoa balio-kideak dira.

$$a_{ij} = \int_0^{c_i} l_j(\tau) d\tau, \quad b_i = \int_0^1 l_i(\tau) d\tau \quad (2.12)$$

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.

Gauss metodoaren koefizienteak kalkulatzeko bi aukera ditugu. Lehen aukera, *Mathematicak NDSolve* paketean implementatuta dagoen koefizienteak kalkulatzeko funtzioa: `ImplicitRungeKuttaGaussCoefficients[]`. Bigarren aukera, guk koefizienteak kalkulatzeko garatutako funtzioa erabiltzea (B.1.).

Adibidea. Hauek dira, $s = 1$, $s = 2$ eta $s = 3$ ataletako Gauss metodoak [29].

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

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

Gauss metodoaren propietateak.

Jarraian, Gauss metodoaren ezaugarri orokorrak azalduko ditugu:

1. Metodo simplektikoa.

Sanz-Sernak [38] Runge-Kutta metodoaren koefizienteek,

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

baldintza betetzen badute metodoa simplektikoa dela frogatu zuen.

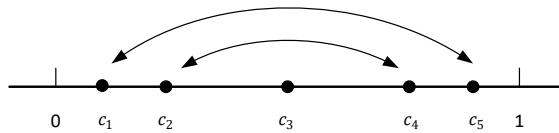
2. Metodo simetrikoa.

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 simetrikoetan, c_i balioak integrazio urratsaren erdigu-nearekiko simetrikoak dira (ikus 2.4. irudia).



2.4. Irudia: Kolokazio metodoen simetria.

3. Orden altuko metodoa. Edozein ordeneko Gauss metodoa eraiki daiteke. Doitasun handiko konputazioetarako orden altuko metodoak beharrezkoak dira: doitasun bikoitzeko aritmetikan ($u \approx 10^{-16}$) $p \geq 8$ ordeneko metodoak gomendarriak dira eta doitasun laukoitzeko aritmetikan ($u \approx 10^{-35}$) orden altuagoko metodoak beharrezkoak dira.
4. Metodo orokorra. Gauss metodoa edozein ekuazio diferenzialari aplika daiteke. Sistema Hamiltondarren problemetan, ez du zertan banagarria izan behar.
5. Paralelizagarria. Ekuazio diferenzial 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 bakotzean polinomio interpolatzale 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.

2.2.3. IRK implementazioa.

IRK algoritmoa.

IRK metodoen erronka handiena, ekuazio-sistema ez-linealaren zenbakizko soluzioaren implementazio eraginkorra da. Problema ez-zurretarako, atalen hasieraketa ($Y_i^{[0]}$) egoki bat duen puntu finkoko iterazioa erabil daiteke. Problema zurretarako, puntu-finkoa iterazioak urrats tamaina txikiegia erabiltzea behartuko luke eta ondorioz, Newton iterazio sinplifikatua erabili ohi da.

Jarraian, *IRK* metodoaren implementazioaren algoritmo orokorraren (Algoritmo 1) laguntzarekin, implementazioaren alde 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
```

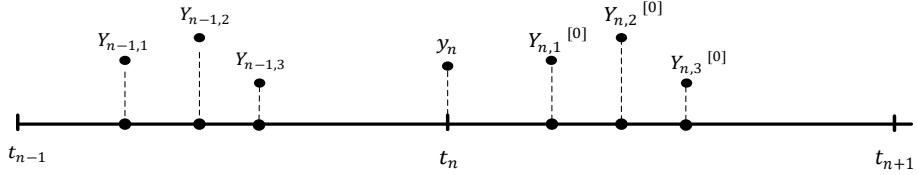
Algoritmo 1: IRK Algoritmo orokorra

Atalen hasieraketa.

Atalen hasieraketa egokia definitu behar da. Aukera simpleena $Y_{n,i}^{[0]} = y_n$ hasiera-tzea da, baina aurreko urratseko atalen informazioa erabiliz hurbilketa hobea lortu daiteke. Aurreko urratseko atalen polinomio interpolatzailaren bidezko hasieraketa era honetan adierazi dezakegu,

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

Interpolazio bidezko hasieraketa. Aurreko urratseko $Y_{n-1,i}$, $i = 1, \dots, s$ eta $(t_{n-1} + h, y_n)$ uneetako balioei dagokien polinomio interpolatzailaren bidez, urrats berriaren atalen hasieraketa $Y_{n,i}^{[0]}$ kalkulatzea (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. Polinomio interpolatzaila.

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

non $l_i(t)$ Lagrangiar polinomioa dugun,

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

3. Atalen hasieraketa.

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

Modu honetan s-ataletako IRK metodo bakoitzari dagokion λ_{ij} koefiziente interpolatzailen lortu daitezke (B.1. atala). Polinomio interpolatzailaren 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.

Laburtaren lanean aurkitu daiteke hasieraketa aurreratuei buruzko informazioa [45].

Iterazio metodoa.

Ekuazio-sistema ez lineala metodo iteratibo bat erabiliz askatu beharra dago. Aplikatutako metodoa problemaren araberako izango da: puntu-finkoaren metodoa edo Newton metodoa izan daiteke. Azpimarratu, Newton metodoaren iterazioak, puntu finkoko metodoaren iterazioak baino azkarrago konbergitzen duela baina konputazionalki garestiagoa dela.

1. Puntu-finkoko iterazioa.

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

Algoritmoa 2: Puntu-finkoko iterazioa.

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

2. Newton metodoaren iterazioa.

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

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

Algoritmoa 3: Newton metodoaren iterazioa

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

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 integracio luzeetan, batura honek soluzioaren doitasun galera eragingo du. Hau ekiditeko *batura konpensatu* teknika erabili ohi da.

Urrats askotako integrazioetan, batura honetan gertatutako biribiltze erroreak doitasun galera garrantzitsua sortzen du. Beraz, zenbakizko integrazioetan biribiltze errorea gutxitzeko batura konpensatu estandarra aplikatzea oso erabilgarria zaigu (ikus 4 algoritmoa).

```

 $y_0 = y(0); e_0 = 0;$ 
for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
    Hasieratu  $Y_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
    Puntu-finkoko-iterazioak  $(y_n, Y_{n,i})$ ;
     $\delta_n = \left( \sum_{i=1}^s h b_i f(Y_{n,i}^{[k]}) \right) + e_n$ ;
     $y_{n+1} = y_n + \delta_n$ ;
     $e_{n+1} = (y_n - y_{n+1}) + \delta_n$ ;
end
```

Algoritmoa 4: Batura konpensatua estandarra.

Batugaien ordena. Batugaien ordenak ere, batura errekurtsiboen emaitzaren doitasunean eragina du [33]. Zenbaki positiboen batura kalkulatzeko, egokiena magnitude txikienetik handienerako batugaien ordena jarraitza da. Horregatik, honako batura

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

irizpide honi jarraituz, prekalkulatutako hb_i koefizienteen txikienetik handieneko ordenaren arabera batuko dira.

2.3. Konposizio eta Splitting metodoak.

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

2.3.1. 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, orden 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 errealk emanik, urrats luzera hauen $\gamma_1 h, \gamma_2 h, \dots, \gamma_s h$ konposaketari dagokion konposizio metodoa,

$$\Psi_h = \phi_{\gamma_s h} \circ \dots \circ \phi_{\gamma_1 h}. \quad (2.14)$$

Algoritmoa.

s-ataletako konposizio metodoen algoritmo orokorra honakoa izango litzateke:

```

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 5: Konposizio metodoak.

Konposizio metodoei buruzko hainbat ohar 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. Oinarrizko metodoa: Störmer-Verlet.

Bigarren ordeneko ekuazio diferentzialak (2.5) ditugunean, Störmer-Verlet

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

integratzialean oinarritzen diren konposizio metodoekin urrats bakoitzean s ekuazio diferentzialaren balioztapena egin behar ditugu.

Konposizio simetrikoak.

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

$$\Psi_h = \phi_{\gamma_s h} \circ \phi_{\gamma_{s-1} h} \circ \dots \circ \phi_{\gamma_{2h}} \circ \phi_{\gamma_1 h} \quad (2.15)$$

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

CO1035: 10 ordeneko konposizio metodoa.

Sofroniou eta Spalettaren [65, 2004] $s = 35$ eta $p = 10$ ordeneko metodoa, orain arteko lortutako orden altueneko konposizio metodo eraginkorrena konsideratu daiteke (Taula 2.2.). Konposizio metodo simetrikoa da. Oinarrizko metodoa simetrikoa eta $p = 2$ ordenekoa dela baliatuz eraikitako metodoa da.

Hamiltondarra $H = H_1 + H_2$ izanik, eta Stömer-Verlet $\phi_h = \varphi_{h/2}^{H_1} \circ \varphi_h^{H_2} \circ \varphi_{h/2}^{H_1}$ metodoan oinarritutako konposizio metodoaren implementazioaren zehaztasunak emango ditugu.

1. Konposizio metodo orokorra.

$$\Psi_h = \phi_{\gamma_s h} \circ \phi_{\gamma_{s-1} h} \circ \dots \circ \phi_{\gamma_{2h}} \circ \phi_{\gamma_1 h}$$

2.2. Taula: 10 ordeneko metodoa konposizio metodoa [29, 158.or] (CO1035).

Koefiziente	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

2. Stömer-Verlet oinarritzko metodoaren konposizio metodoa,

$$\Psi_h = (\varphi_{h\gamma_s/2}^{H_1} \circ \varphi_{h\gamma_s}^{H_2} \circ \varphi_{h\gamma_s/2}^{H_1}) \circ \cdots \circ (\varphi_{h\gamma_2/2}^{H_1} \circ \varphi_{h\gamma_2}^{H_2} \circ \varphi_{h\gamma_2/2}^{H_1}) \circ (\varphi_{h\gamma_1/2}^{H_1} \circ \varphi_{h\gamma_1}^{H_2} \circ \varphi_{h\gamma_1/2}^{H_1}).$$

3. Jarraian dauden φ^{H_1} fluxuak 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. Era berean, integrazioaren tarteko urratsetan, lehen atala $\varphi_{ha_{s+1}}^{H_1}$ eta azkena $\varphi_{ha_1}^{H_1}$ bakar batean elkartu daitezke,

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

2.3.2. Splitting metodoak.

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

Splitting arruntak.

Splitting arrunta aplika daiteke, H Hamiltondarra, $H = H_1 + H_2$ banatu eta Hamiltondar bakoitzari dagokion sistema modu esplizituan integratu daitezkeenean. Sistemen fluxu zehatzak, $\varphi_t^{[H_2]}$ eta $\varphi_t^{[H_1]}$ izendatzen baditugu,

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

$$\phi_h = \varphi_h^{[H_2]} \circ \varphi_h^{[H_1]} \quad \text{edo} \quad \phi_h^* = \varphi_h^{[H_1]} \circ \varphi_h^{[H_2]}. \quad (2.16)$$

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

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

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

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

2. Strang-Marchuk splitting. $p = 2$ ordeneko metodo simetrikoa,

$$\phi_h = \varphi_{h/2}^{[H_1]} \circ \varphi_h^{[H_2]} \circ \varphi_{h/2}^{[H_1]}. \quad (2.17)$$

Adibidea. Hamiltondar banagarrieta rako $H(p, q) = T(p) + V(q)$, *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 honetan ere, splitting teknika aplika daiteke metodo simplektikoak eraikitzeko. 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, $\varphi_t^{[H_2]}$ zenbakizko metodoaren propietateak mantentzen ditu.

Splitting orokorrak.

Aurreko splitting metodoen orokorpenea,

$$\phi_h = \varphi_{\beta_s h}^{[H_2]} \circ \varphi_{\alpha_s h}^{[H_1]} \circ \varphi_{\beta_{s-1} h}^{[H_2]} \cdots \varphi_{\beta_1 h}^{[H_2]} \circ \varphi_{\alpha_1 h}^{[H_1]}. \quad (2.18)$$

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

Algoritmoa.

Splitting metodoen algoritmo orokorra honakoa izango litzateke:

```

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 6: Splitting metodoak.

Splitting metodoen algoritmoan, konposizio metodoen algoritmoei buruz aipatutako ezaugarri berdinak errepikatu beharko genituzke.

2.3.3. Eguzki-sistemari egokitutako splitting metodoak.

Honakoa dugu N-gorputzeko problema grabitazionalaren Hamilondarra,

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

Eguzki-sistemaren integrazioarako erabiltzen diren koordenatu sistema nagusiak *Jacobi* edo koordenatu Heliozentrikoak dira. Bi koordenatu sistema hauekin Hamilondarra 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 kepleriarra den eta H_B aldiz, planeten arteko interakzioek eragiten duten perturbazio txikia. Bi koordenatu sistematan berridatzitako Hamilondar funtzioen (ikus A.2. eranskina) arteko differentzia handiena da, *Jacobi* koordenatuetan H_B bakarrik kokapenen araberako dela eta *Heliozentrikoetan* H_B kokapen naiz abiaduraren arabera-koa dela.

Eguzki-sistemaren N-gorputzeko problema grabitazionalari egokitutako zenbakizko bi integratzaile simplektiko azalduko ditugu. Lehena, Laskarrek eta Ro butelek [49, 2001] definitutako *SABAC*₄ integratzailea eta bigarrena, Blanes-ek [8, 2013] [22] definitutako *ABAH1064* integratzailea.

***SABAC*₄ integratzailea.**

Laskarrek (2001) *SABA*_n eta *SBAB*_n integratzaile simplektikoak [49] proposatu zituen. Metodoen ordena $K_{SABA_n} = K_{SBAB_n} = \mathcal{O}(h^{2n}\epsilon + h^2\epsilon^2)$ da; urrats luzera

2.3. Taula: $CSABA_4$ splitting metodoa [49].

Koeficiente	Balioa	Koeficiente	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		

handitarako eta ϵ balio txikitarako, gai nagusia ($h^{2n}\epsilon$) izango da.

Bigarren integratzaila famili bat ere $CSABA_n$ eta $CSBAB_n$ (*Corrected integrators*) definitzen ditu. Hauetan, urrats bat gehitutako integratzailaak 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 [48] erabilitako $CSABA_4$ integratzaila 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 (Taula 2.3.) taulan zehaztu ditugu.

ABAH1064 integratzaila.

Blanes-ek (2013), ABA eta $ABAH$ metodo simplektikoak proposatu zituen [8]. ABA metodoak, Jacobi koordenatuak erabiltzen direnean egokitutako integratzailaak eta $ABAH$ integratzailaak, koordenatu Heliozentrikoei erabiltzen direnean egokitutako integratzailaak.

Atal honetan, koordenatu Heliozentrikoei egokitutako $ABAH1064$, $p = 10$ ordeneko metodoa deskribatuko dugu. Eguzki sistemaren integrazioarako koordenatu Heliozentrikoei dagokion Hamiltondarra era honetakoa dugu,

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

2.4. Taula: ABAH1064 splitting metodoa [8].

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

2.5. Taula: Integrazio metodoen laburpena

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

$H_I(p, q)$ 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,

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

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

2.4. Laburpena.

Hurrengo taulan (2.5. Taulan) orden altuko hiru integratzaile simplektikoen ezau-garriak laburtu ditugu.

Metodo simplektikoei buruzko liburu monografiko hauek gomendatuko ditugu: Sanz-Serna and M.P. Calvo's Numerical Hamiltonian Problems (1994) [38]; E. Hairer, C. Lubich and G. Wanner's Geometrical Numerical Integration (2001) [29]; B. Leimkuhler and S. Reich's Simulating Hamiltonian Dynamics (2004) [50]; Feng, Kang and Qin, Mengzhao Symplectic geometric algorithms for hamiltonian systems (2010) [23].

3. Kapitulua

Problemak.

3.1. Sarrera.

Tesiaren erdigunea N-gorputzeko problema grabitazioanala da eta gure helburua, eguzki sistemaren problemaren simulaziorako zenbakizko integratzaile eraginkorra implementatzea da. Gure experimentuetan, eguzki sistemaren bi eredu simple kontsideratu ditugu: lehena, *kanpoko planeten* izeneko problema (eguzkia, kanpoko planetak eta pluto) eta bigarrena eredu konplexuagoa, *9-planeten problema* izeneko problema (eguzkia, 8 planetak eta pluto). Bi eredu hauetan, gorputzak masa puntualak dira eta gorputz hauen arteko erakarpen grabitazionalak bakarrik kontsideratu 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 eraginkorrak 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, zenbakizko esperimentuak modu aberatsago eta zabalago batean egiteko, pendulu bikoitzaren problema osagarri gisa aukeratu dugu. Pendulu bikoitzaren bi bertsio kontsideratu ditugu: pendulu bikoitz arrunta eta pendulu bikoitz zurruna. N-gorputzeko problemaren Hamiltondarra banagarria da baina pendulu bikoitzarena aldiz, ez da banagarria. Bestalde, pendulu bikoitzari malguki bat gehituz problema zurruna izatea lortuko dugu eta problema hauen zaitasunei aurre egitea. Gainera, eguzki sistema kaotiko kontsideratzen dela [46] jakinik, pendulu bikoitz arruntak izaera kaotiko nabarmena azaltzen duen hasierako balio zehatzak aukeratu ditugu. Problema kaotikoak, hasierako balio edo parametroen perturbazioekiko, diskretizazio-errorekiko (trunkatze) edo birbi-

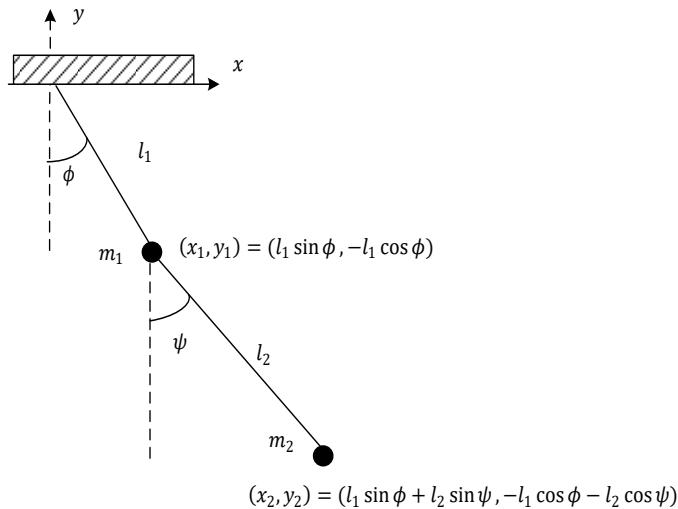
tze errorekiko esponentzialki oso sentikorrik dira.

3.2. Pendulu bikoitza.

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

Pendulu bikoitza 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 konsideratuko 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 bikoitza.

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

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

Sistemaren parametroak. Gure esperimentuetarako honako parametroak konsideratuko ditugu,

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

Hasierako balioak. Bi hasierako balio ezberdin konsideratu ditugu [19]: 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 $p(0) = (2.7746, 2.7746)$. Urrats luzera $h = 2^{-7}$ aukeratz, $T_{end} = 2^{12}$ segunduko integrazioa egin dugu. Zenbakizko soluzioa, $m = 2^{10}$ urratsero itzuli dugu.
2. Hasierako balio kaotikoak (CDP): $q(0) = (0, 0)$ eta $p(0) = (3.873, 3.873)$. Urrats luzera $h = 2^{-7}$ aukeratz, $T_{end} = 2^8$ segunduko integrazioa egin dugu. Zenbakizko soluzioa, $m = 2^8$ urratsero itzuli dugu.

Pendulu bikoitza (zurruna).

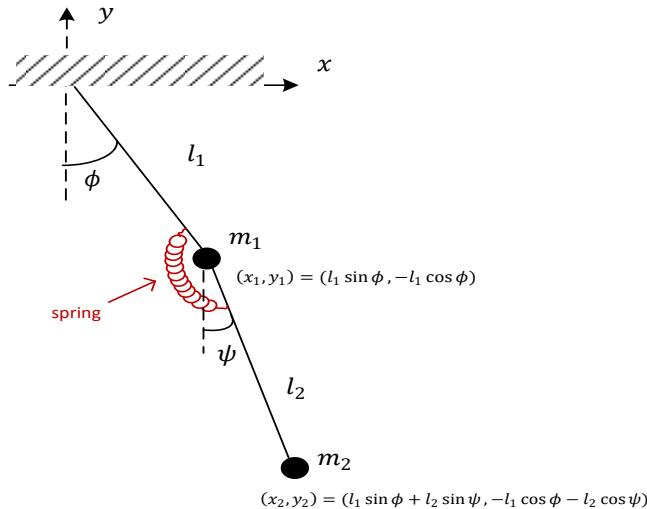
Pendulu bikoitz arruntari malguki bat gehitutako sistema da (ikus 3.2. Irudia). Malgukiaren zurruntasuna k parametroaren araberakoa da; k -ren balioa zenbat eta handiago izan orduan eta malgukia zurrunagoa da.

Formulazio Lagrangiarrean,

$$L = T - V$$

energia potentzialari $1/2 k \theta^2$ gaia gehituz, dagokion $H(q, p)$ funtzio Hamilton-darra lortuko dugu,

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



3.2. Irudia: Pendulu bikoitza (zurruna).

Sistemaren parametroak. Gure esperimentuetarako honako parametroak fin-koak konsideratuko ditugu,

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

eta pendulu bikoitz zurruntasun maila ezberdinak aztertzeko, k parametroari balio ezberdinak emango dizkiogu,

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

Hauek dira izaera ez-zurreneko hasierako balioak [19] : $q(0) = (1.1, -1.1)$ and $p(0) = (2.7746, 2.7746)$ eta k balio bakoitzari dagokionak, era honetan kalkulatu ditugu,

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

Urrats luzera $h = 2^{-7}$ aukeratuz, $T_{end} = 2^{12}$ segunduko integrazioa egin dugu. Zenbakizko soluzioa, $m = 2^{10}$ urratsero itzuli dugu.

3.3. N-Body 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,

$$\dot{q}_i = v_i, \quad i = 0, 1, \dots, N, \quad (3.5)$$

$$\dot{v}_i = \sum_{j=0, j \neq i}^N \frac{G m_j}{\|q_j - q_i\|^3} (q_j - q_i), \quad i = 0, 1, \dots, N \quad (3.6)$$

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 honetan definitzen dugun P , konstantea dela modu errezean frogatutako,

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

Eta ondorioz,

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

P eta B bektoreen osagaiei masa zentroaren sei integralak esaten zaie. Masa zentruaren kokapen (Q) eta abiadura (V) era honetan definitzen dira,

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

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

2. Momentu angeluarra.

Momentu angeluarra (L) 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.

Hamitonarra 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 daitezen, modu honetan finkatuko ditugu,

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

Momentu angeluarra eta energia zenbakizko integrazioaren doitasuna neurtze-ko erabiltzen dira. Energia zenbakizko integracioen biribiltze errorea neurtzeko integral egokiena da.

3.4. Eguzki-sistema.

Eguzki-sistemaren planeten orbiten mugimenduaren eredu matematikoa, Hamitonarrean oinarritutako ekuazio diferentzial arrunten bidez formulatzen da. Ekuazio diferentzial arrunten sistema, $6N$ ordenekoa da (N planeten kopurua izanik).

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 konplexuagoetako 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, $N = 9$ eta $N = 16$ (ilargia, Pluto eta asteroide nagusienak ere konsideratzen ditugunean) gorputz kopuruen artekoa.

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 txikiiek sortzen duten sistema dinamikoa da. Planetak eta bere sateliteen mugimenduan kolisio gertuko egoerarik edo eszentrizitate handiko orbitalik ez dago. Beraz, ikuspuntu honetatik, sistema dinamiko simplea konsideratu daiteke.

Eguzki sistema egonkorra konsideratzen da, hau da, hurrengo bilioi urteetan planeten arteko talkarik edo planeten kanporatzerik ez da espero [46] [31]. Egonkortasunaren azterketa zehatzago batean, sistema erregularrak eta kaotikoak bereiz ditzakegu. Sistema erregularretan, hasierako balioen perturbazio txikiak eragina, denboran poliki hasiko da eta beraz, epe luzeko predikzio zehatzak posible dira. Sistema kaotikoetan aldiz, hasierako balioekiko oso sentikorra da eta gertuko soluzioen differentzia esponentzialki hasiko da. Hasierako unean, differentzia $d(0) = d_0$ bada orduan,

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

non λ Lyapunov exponentziala deitzen zaio. Eguzki sistema kaotikoa konsideratzen da eta Laskarrek [46] Lyapunov denbora $\lambda^{-1} \approx 5$ milioi urtetan finkatzen du. Honako espresioa kalkulatzeko modu zuzenena proposatzen du,

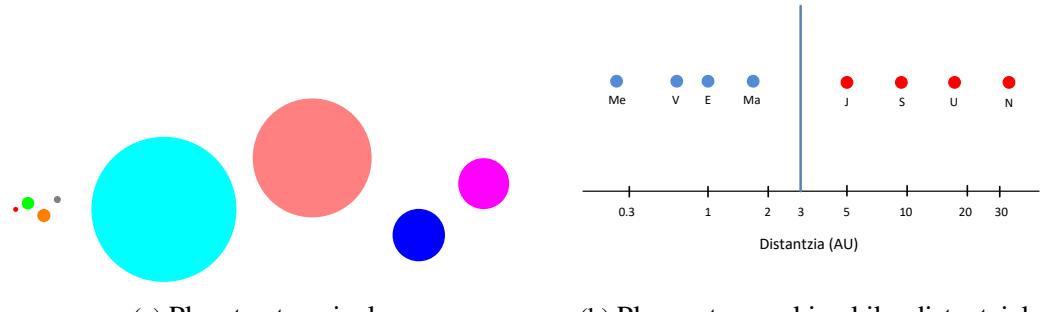
$$d(t) \approx d(0)10^{t/10}.$$

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

Eguzki sistemaren epe luzeko integrazioak aztertu nahi ditugu eta beraz, urrats kopurua oso handia da. Eguzki sistemaren bizi iraupena 5×10^9 urtetakoak izanik eta eguzki sistema osoaren integrazioetako ohiko urratsa $h = 0.0025$ urteko bida (orbita txikieneko periodoaren %1), eman beharreko urrats kopurua 2×10^{12} izango da. Era berean, kanpo planeten integrazioan erabilitako urrats tamaina handiago denez, 5×10^{10} urrats kopuru ingurukoa da. Gainera ilargia konsideratuko bagenu, lurrateko distantzia $D_M = 3.844 \times 10^8$ eta periodoa $P_M = 27.32$ egunekoa.

Eguzki sistemaren probleman eskalak oso zabalak dira. Denbora eskalari da-gokionez, ilargiak luraren inguruko orbita 27.32 egunetakoak da, lurruk eguzkiarekiko inguruko orbitaren periodoa 1 urtekoa eta Neptunorena 163 urtekoa. (3.3.)

irudian, planeta nagusien tamainak konparatu eta eguzkiarekiko distantziak irudikatu ditugu .



(a) Planeten tamainak.

(b) Planetenet eguzkiarekiko distantziak.

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

3.1. Taula

Planeta	Masak kg	Distantzia AU	Periodoa urteak
Eguzkia	1.99×10^{30}		
Merkurio	3.30×10^{23}	0.39	0.24
Venus	4.87×10^{24}	0.72	0.007
Earth	5.97×10^{24}	1.00	1.007
Mars	6.42×10^{23}	1.52	1.88
Jupiter	1.90×10^{27}	5.20	11.86
Saturn	5.68×10^{26}	9.54	29.42
Uranus	8.68×10^{25}	19.19	83.75
Neptune	1.02×10^{26}	30.06	163.72
Pluto	1.31×10^{22}	39.53	248.02

Koordenatu sistemak.

Hiru dira erabiltzen diren koordenatu sistema nagusienak:

1. Koordenatu kartesiarrak.
2. Koordenatu Heliozentrikoak.

3. Koordenatu Jakobiarrak.

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

Problema motak.

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 [64], 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,

$$H(p, q) = \frac{p^2}{2m} - \frac{\mu}{\|q\|}, \quad (3.7)$$

non m eta μ konstanteen balioak formulazioaren araberakoak diren.

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

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

eta ekuazio diferentzialak era honetan definitzen dira,

$$\dot{q} = p, \quad \dot{p} = -\frac{k q}{\|q\|^3}, \quad (3.8)$$

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, funsezkoa dela eguzki sistemaren integrazio metodoentzat, kepler problemaren doitasun handiko eta kalkulu eraginkorra konputatzea. Erreferentziazko kepler problemaren implementazioak, Danby (1992) [17] eta J.Wisdom (2015) [69] ditugu.

Gure aldetik, kepler problemaren implementazioa garatu dugu eta ideia nagusia, hitzez azalduko dugu. Lehenik, koordenatu cartesiarretatik koordenatu eliptikoetara (a, e, i, Ω, E) itzulpena egingo dugu. Koordenatu berriean E (*eccentric anomaly*) aldagai izan ezik, beste aldagaien konstante mantentzen dira: beraz E_0 balioa emanda, Δt denbora tartea aurrera egin eta E_1 balioa berria kalkulatuko dugu. Azkenik, koordenatu eliptikoetatik koordenatu cartesiarretara itzulpena eginez, kokapen eta abiadura berriak eskuratuko ditugu.

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

Implementazioaren garapenaren zehaztasun guztiak eranskinaren (A.) atalean eman ditugu.

Hiru gorputzen problema.

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

- Ohikoa den moduan Newton eredu grabitazionalean, gorputzak masa puntualak konsideratuko ditugu.
- Horietako bi gorputz nagusiak (m_1, m_2), bere masa zentroaren inguruan orbita zirkularrean mugitzen dira. Adibidez, lurra eta ilargia konsideratu daitezke. Ilargiak ezentrizidate 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 denez, 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 beste gorputzen eragina baztertu da. Lurra-Ilargi sistemaren, eguzkiaren grabitazio indarra ez dagoela konsideratu da.

(x, y) masa txikiko gorputzaren kokapen koordenatuak izanik, hauek dira erro-tazio koordenatu sisteman 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, $r_1 = ((x + \mu)^2 + y^2)^{1/2}$, $r_2 = ((x - umu)^2 + y^2)^{1/2}$, $\mu = 0.01277471$, $-\mu = (-m/M + m)$ eta $umu = 1 - \mu$.

Problema ezberdinak deskribatzen dira μ parametroaren arabera. Guk aztertutako problema zehatzean, $\mu = 0.01277471$ balioa Lurra-Ilargia modeloari dago-kio. Gorputz nagusi handiena, Lurra, $(-\mu, 0)$ eta gorputz nagusi txikiena, Ilargia, $(1 - \mu, 0)$ posizioan kokatzen dira. Ilargiak orbita zirkularra du luraren inguruan eta ekuazio diferentzialak satelitearen mugimendua deskribatzen dute.

Sistema dinamikoaren energia, soluzioan zehar konstante mantentzen da,

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

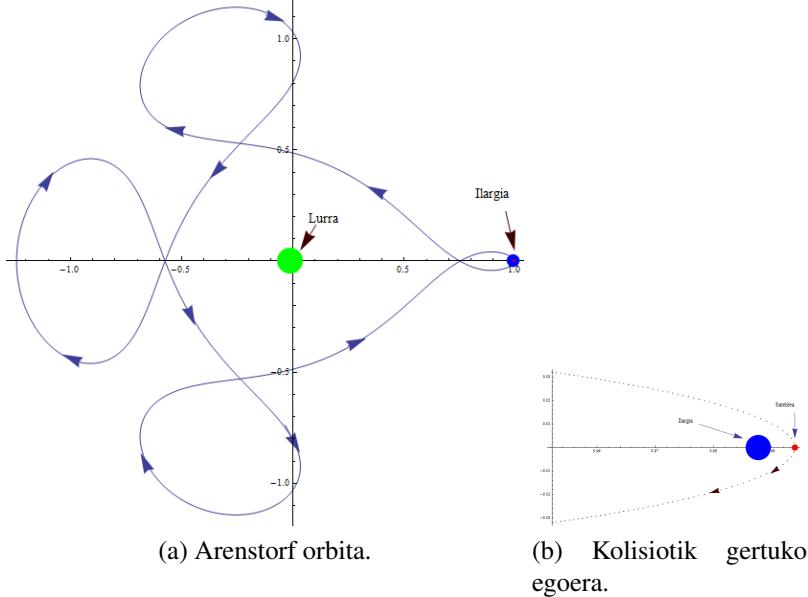
Ezaguna da, hasierako balio hauetarako (3.2. taula) soluzioa $t = 17.0652165601579625588917206249$ periodikoa dela (3.4. irudia),

3.2. Taula: R3BP problemaren hasierako balioak.

Kokapenak	x, y	0.994	0
Momentuak	p_x, p_y	0.	$-2.00158510637908252240537862224 + 0.994$

Kanpoko planeten problema.

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 integrazioarengan oso eragin txikia du. Bigarrenik, urrats luzera handi erabili daiteke eta beraz, epe luzeko integrazioak errazten dira (konputazio denbora gutxiago behar delako). Hirugarrenik, Pluto orbitaren berezitasunak ikertzea, 1960 – 1980 urteetan interes handikoa izan zen.



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.

Kanpo planeten problemaren ereduan, eguzkia, lau planeta nagusiak (Jupiter, Saturno, Urano, Neptuno) eta Pluto konsideratuko ditugu. Unitateei dagokionez, planeten masak eguzkiarekiko erlatiboak dira, hau da, eguzkiaren masa 1 da eta grabitazio konstantea $G = 2.95912208286 \cdot 10^{-4}$. Barne planeten masak eguzkiaren masari gehitu zaio eta horregatik, eguzkiaren masak, $m_0 = 1.00000597682$ balioa hartzen du.

Hasierako balioak Hairer [29] liburutik hartu ditugu. Planetei dagokien masak (3.3. taula) eta kokapenak/abiadurak (3.4. taula) tauletan laburtu ditugu.

3.3. Taula: Kanpo planeten masa parametroak.

Gorputza	Masa
Eguzkia	1.000005976823
Jupiter	0.000954786104043
Saturn	0.000285583733151
Uranus	0.0000437273164546
Neptune	0.0000517759138449
Pluto	$1/(1.3 \cdot 10^8)$

3.4. Taula: Kanpoko planeten problema.

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
Saturn	x, y, z	9.0755314	-3.0458353	-1.6483708
	v_x, v_y, v_z	0.00168318	0.00483525	0.00192462
Uranus	x, y, z	8.3101420	-16.2901086	-7.2521278
	v_x, v_y, v_z	0.00354178	0.00137102	0.00055029
Neptune	x, y, z	11.4707666	-25.7294829	-10.8169456
	v_x, v_y, v_z	0.00288930	0.00114527	0.00039677
Pluto	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, kanpoko planeten problemak baino konplexutasun handiago du. Planeten eta eguzkiaren arteko interakzio kopurua 45 (kanpoko planeten probleman 15) da. Orbita-periodoa txikiena 50 aldiz txikiagoa (Merkuriok 0.24 urtetakoa eta Jupiterrek 11.86 urtetakoa) da. Merkuriok eszentrizitate handieneko orbita $e = 0.206$ (Jupiterrek $e = 0.048$) du. Konplexutasun handiago honen ondorioz, problema honen zenbakizko integrazioaren konputazioak zaitasun handiagoa du.

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

Hasierako balioak *DE-430* (2.014) [25] 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 ,

Laskaren eredu.

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

Laskarrek, 2.011. urteko epe luzeko zenbakizko integraziorako [48] erabilita-

3.5. Taula: Eguzki eta 9 planeten hasierako balioak integrazio jatorriarekiko.

Eguzkia	x, y, z	0.00450250878464055477	0.00076707642709100705	0.00026605791776697764
	v_x, v_y, v_z	-0.00000035174953607552	0.00000517762640983341	0.00000222910217891203
Mercury	x, y, z	0.36176271656028195477	-0.09078197215676599295	-0.08571497256275117236
	v_x, v_y, v_z	0.00336749397200575848	0.02489452055768343341	0.01294630040970409203
Venus	x, y, z	0.61275194083507215477	-0.34836536903362219295	-0.19527828667594382236
	v_x, v_y, v_z	0.01095206842352823448	0.01561768426786768341	0.00633110570297786403
EMB	x, y, z	0.12051741410138465477	-0.92583847476914859295	-0.4015402264531522236
	v_x, v_y, v_z	0.01681126830978379448	0.00174830923073434441	0.00075820289738312913
Mars	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
Saturn	x, y, z	7.89439068290953155477	4.59647805517127300705	1.55869584283189997764
	v_x, v_y, v_z	-0.00321755651650091552	0.00433581034174662541	0.00192864631686015503
Uranus	x, y, z	-18.26540225387235944523	-1.16195541867586999295	-0.25010605772133802236
	v_x, v_y, v_z	0.00022119039101561468	-0.00376247500810884459	-0.00165101502742994997
Neptune	x, y, z	-16.05503578023336944523	-23.94219155985470899295	-9.40015796880239402236
	v_x, v_y, v_z	0.00264276984798005548	-0.00149831255054097759	-0.00067904196080291327
Pluto	x, y, z	-30.48331376718383944523	-0.87240555684104999295	8.91157617249954997764
	v_x, v_y, v_z	0.00032220737349778078	-0.00314357639364532859	-0.00107794975959731297

ko eredua deskribatuko dugu. Hasierako integrazioetan, eguzkia, 8 planetak, Pluto eta ilargia bakarrik kontsideratu zituen. Eguzkiaren erlatibilitate efektua (Saha eta Tremainen [62] teknikaren arabera) eta eredu errealistaren indar ez grabitazional garrantzitsuenak aplikatu zituen. Azken integrazioetan, Ceres, Pallas, Vesta, Iris eta Bamberga asteorideak gehitu zituen zituen.

Ilargia gorputz independente gisa kontsideratu zuen. Ilargiaren lurrarekiko distantzia (380.000km), beste gorputzekiko distantziekin alderatzen badugu (eguzkiaren 150.000.000 km eta Venus-ekiko 45.000.000 km) oso txikia da. Hori dela eta, ilargiaren zenbakizko integrazioa, eguzki-sistemaren barizentroarekiko kalkulatu beharrean, lurrarekiko kalkulatzu doitasun handiago lortuko da. Lur-eguzkiaren q_e kokapen eta ilargi-eguzkiaren q_m kokapen aldagaiak, hurrenez-hurren q_B lur-ilargi sistemaren barizentroa eta q_{em} ilargia-lurrarekiko kokapen oaldagaiekin ordezkatzen dira,

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

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

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

3.6. Taula: Planeten masa parametroak.

Gorputza	GM (au^3/day^3)
Eguzkia	$0.295912208285591100e - 03$
Mercury	$0.491248045036476000e - 10$
Venus	$0.724345233264412000e - 09$
Earth	$0.888769244512563400e - 09$
Mars	$0.954954869555077000e - 10$
Jupiter	$0.282534584083387000e - 06$
Saturn	$0.845970607324503000e - 07$
Uranus	$0.129202482578296000e - 07$
Neptune	$0.152435734788511000e - 07$
Pluto	$0.217844105197418000e - 11$
Moon	$0.109318945074237400e - 10$

3.7. Taula

Planeta	Distantzia AU	Periodoa urte	GM ($au^3/egun^3$)	Ezentrizitatea
Sun			$0.2959e - 03$	
Mercure	0.39	0.24	$0.4912e - 10$	0.205
Venus	0.72	0.007	$0.7243e - 09$	0.007
Earth	1.00	1.007	$0.8887e - 09$	0.017
Moon			$0.1093e - 10$	0.055
Mars	1.52	1.88	$0.9549e - 10$	0.094
Jupiter	5.20	11.86	$0.2825e - 06$	0.049
Saturn	9.54	29.42	$0.8459e - 07$	0.057
Uranus	19.19	83.75	$0.1292e - 07$	0.046
Neptune	30.06	163.72	$0.1524e - 07$	0.011
Ceres	2.77	4.6	$0.1400e - 12$	0.07
Pallas	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
Pluto	39.53	248.02	$0.2178e - 11$	0.244

tzeko erabiliko ditugula.

Lurra, Ilargia = $\{q_B, q_{em}\}$;

for $i \leftarrow 1$ **to** endstep **do**

$\{q_e, q_m\} \leftarrow \{q_B, q_{em}\}$;
 Ebaluatu $\dot{y} = f(y)$;
 $\{q_B, q_{em}\} \leftarrow \{q_e, q_m\}$;
 Integrazioa (q_B, q_{em}) ;;

end

Algoritmoa 7: Ilargiaren kalkuluak.

3.8. Taula: Ilargiaren Lurrarekiko hasierako balioak.

Ilargia	x, y, z	-0.0008017735147818490	-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 aritmetikoen (+, -, *, /) kalkulu gehienetan ere, emaitzaren hurbilpena egin beha da. \mathbb{R} sistematik \mathbb{F} sistemara bihurtzeko funtzioari biribiltzea esaten zaio. Oro har konputazio zientzietan, 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 formato 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.

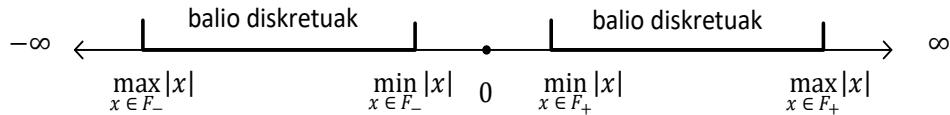
4.2. IEEE-754 estandarra.

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 (Irudia 4.1. Irudia) irudian laburtu dugu. Bai zenbaki positiboentzat, bai negatiboentzat, adierazi daitekeen zenbaki handienaren eta txikie-

naren arteko balio 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 adierazpena.

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

1. m bitez osatutako zatia, mantisa izenekoa eta M zenbaki erreala adierazten duena. Horietako bit bat (S) zeinua adierazten du. Bestalde M mantisa modu normalizatu honetan emana da, $\pm 1.F$ eta zati erreala (F) bakarrik gordetzen 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,

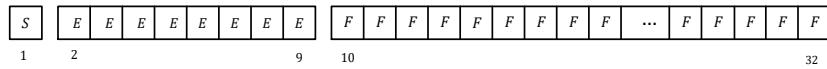
$$M \times b^E, \quad b = 2,$$

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

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

IEEE-754 estandarrean, oinarri bitarreko koma-higikorreko hiru formato definitzen dira: bata doitasun arrunta (*single precision*), bestea doitasun bikoitza (*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 [56] eta azkarra 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



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.

4.1. Taula

Formatoa	Tamaina	m	e	Tartea	$u = 2^{-m}$
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}

bit kopuru erdia delako eta gainera, hardware bereziei (*Intel* makinetan *SSE* moduloak) esker, eragiketa aritmetikoak azkarragoak direlako. 2008. urtean, IEEE-754 estandarrak 128-biteko koma-higikorreko aritmetika onartu zuen bainan bere implementazioa softwarez bidezkoa da eta exekuzioa gutxi gorabehera doitasun bikoitzeko aritmetika baino 10-15 aldiz motelagoa da.

Problema batzuk, doitasun bikoitza bainon doitasun handiagoa behar dute [39]. Doitasun laukoitza edo altuagoa, software liburutegien bidez emulatzen ohi dira. Doitasun altuko zenbakiak adierazteko nagusiki bi modu bereizten dira:

1. *Digito-anitzeko adierazpena.* Zenbakiak exponente bakarra eta mantisa bat baino gehiagorekin adierazten dira (adbz *GNU MPFR liburutegia*).
2. *Termino-anitzeko adierazpena.* Zenbakiak ebaluatu gabeko hainbat koma-higikorreko makina zenbaki estadarren batura gisa adierazten dira (adbz *Bailey QD liburutegia*) eta exekuzioaren ikuspegitik, hardware bidezko implementazioaren abantaila dute.

GCC libquadmath liburutegia erabili dugu doitasun laukoitzeko gure experimentuetarako. Doitasun laukoitzeko integrazioak, soluzio zehatzak konsideratu ditugu eta doitasun bikoitzeko implementazioaren errorea, soluzio zehatzaren differentzi gisa kalkulatu dugu.

Laskar-ek epe luzeko eguzki-sistemaren simulazioaren (-250 eta $+250$ milioitako integrazio tartea) konputaziorako kalkuluak, kontu handiz eta doitasun

handian egin behar ditu. Dena den, era honetako problemak salbuezpenak dira. Ez da ohikoa izaten doitasun handian lan egin beharra eta egia da ere, neurri fisiko oso gutxi ezagutzen direla hain doitasun handian (adibidez 50-bitexin, lurra eta ilargiaren arteko distantzia bakteria baten neurriaren bainon errorea txikiagoarekin 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 integrazioaren 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 adierazpen errorea eta beste eragiketa (aritmetika) errorea.

Adierazpen errorea.

Zenbaki erreal 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 esleitzten dion funtzioari deituko diogu. $fl(x)$, \mathbb{F} multzoan x -en gertuen dagoen balioa bezala definitzen da.

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.

Eragiketen 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 mantisen biderketaren emaitza zehatza adierazteko, $2m$ digituzko mantisa behar dugu (m digituzko galera) [26]. Salbuespena, biderkagaietako bat 2ren berredura 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 $\tilde{x}_1, \dots, \tilde{x}_n \in \mathbb{F}$,

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

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

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.

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

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

>> z=1.+y;
>> InputForm[z]
>> 1.000000003141594 # 16-digitu hamartar zuzenak.

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

4.4. Biribiltze errorearen gutxitzeko teknikak.

Batura eta biderketa eragiketen biribiltze errorearen konputaziorako algoritmoak ezagunak dira [18][33]. Algoritmo hauek, *termino-anitzeko adierazpen* implemen-tazioetan erabiltzen dira eta baturaren kasuan, batura konpensatu izeneko algorit-moaren oinarria da. Ikusiko dugun bezala, algoritmo simpleak dira eta konputazio kostu txikia dute.

Ideia, teknika hauek zenbakizko integrazioaren implementazioaren kalkulu "kritikoetan" erabiltzea da, soluzioaren doitasuna handitzeko asmoarekin.

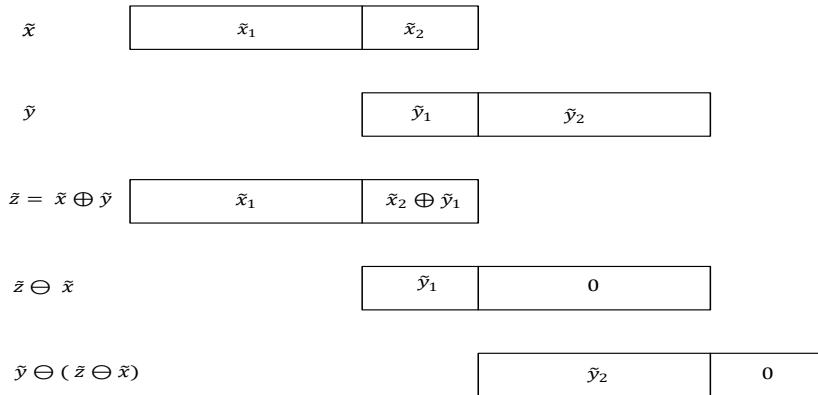
Batura: Fast2Sum.

Fast2Sum algorithmoa, 1971an Dekker-ek [18] sortu zuen. Koma-higikorreko $\tilde{x}, \tilde{y} \in \mathbb{F}$ non $|\tilde{x}| \geq |\tilde{y}|$ bi zenbakien, 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 8: Fast2Sum.

Irudiaren (4.3. Irudia) laguntzarekin hobeto uler daiteke baturaren biribiltze errorearen kalkulua.



4.3. Irudia: Batuketaren biribiltze errorea.

Batura konpensatua.

Era honetako batura errekurtsiboetan,

$$z = \sum_{i=1}^n \tilde{x}_i,$$

biribiltze errorea gutxitzeko teknika ezaguna da [33]. 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) kalkulatuko dugu eta hurrengo urratsean, batugaiari gehituko diogu.

```

 $z = 0; e = 0;$ 
for  $i \leftarrow 1$  to  $n$  do
     $x = z;$ 
     $y = \tilde{x}_i + e;$ 
     $z = x + y;$ 
     $e = (x - z) + y;$ 
end
```

Algoritmoa 9: Batura konpensatua.

Knuth-ek eta Kahan-ek [56], batura konpensatuko algoritmoaren bidez kalkulatutako z batura honakoa betetzen duela,

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

frogatu zuten.

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.

Bidekerta: 2MultFMA.

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

orokorrean konputazioak azkartzen ditu eta biderketa eskalarren, matrize biderkaturen 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 10: 2MultFMA.

Sterbenz Teorema.

Sterbenz teoremaren arabera [66], bi zenbaki elkarrekiko gertu daudenean, honako baldintza betetzen bada horien arteko kendura zehatza da.

$$x, y \in \mathbb{F}, \quad \frac{y}{2} \leq x \leq 2y \quad \Rightarrow \quad x - y \in \mathbb{F}. \quad (4.1)$$

4.5. Laburpena.

Koma-higikorreko aritmetikan sakontzeko honako biografia azpimarratuko dugu: Numerical computing with IEEE floating point arithmetic (Michael L. Overton) [59], Handbook of floating-point arithmetic (Jean-Michel Muller) [56], Accuracy and stability of numerical algorithms (Nicholas J Higham) [33] eta A graduate introduction to numerical methods (Rober M Corless) [16].

5. Kapitulua

Zientzia konputazioa.

Optimizing a program can be a better solution than buying a faster computer!

Performance optimization of Numerically Intensive Codes(2001)

5.1. Sarrera.

Azken hamarkadetan, zientzia konputazionalaren hazkundea oso handia izan da eta bere erabilera ia zientzia arlo guzietara zabaldu da. Zientzialiek 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 garatuz eta konputagailuen gaitasunei etekina atereaz 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 lortzeko helburuarekin 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 [68] eta zentzu honetan hiru ezaugarri hauek bereziki zaindu behar dira:

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

5.2. Eraginkortasuna

Zientzia konputazioaren eraginkortasuna neurtzeko, koma-higikorreko eragiketa kopurua (*flops*) erabili ohi da baina problema handia denean, datuen mugimendua koma-higikorreko eragiketak baino garestiagoa da, eta beraz 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 egiten ditu (> 100 megaflops).

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 genitzke. Baina matrize honek, 3.9 MB memoria beharrezkoa du eta suposatuz konputagailuaren Cache memoria baino handiagoa dela, 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 gainera, etorkizunean diferentzia 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_cpu_time;

clock0= clock();

<neurtu nahi den kodea>

clock1=clock();

elapsed_cpu_time=(clock1 - clock0)/CLOCKS_PER_SEC;
```

2. `time()`.

Funtzioaren bi deien artean igarotako denbora neurtzeko erabiliko dugu (**elapsed-time**).

```
#include <time.h>

time_t time0, time1;
double elapsed_time;

time(&time0);

<neurtu nahi den kodea>

time(&time0);

elapsed_time=difftime(time1, time0);
```

CPU denborari dagokion zehaztasun 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.

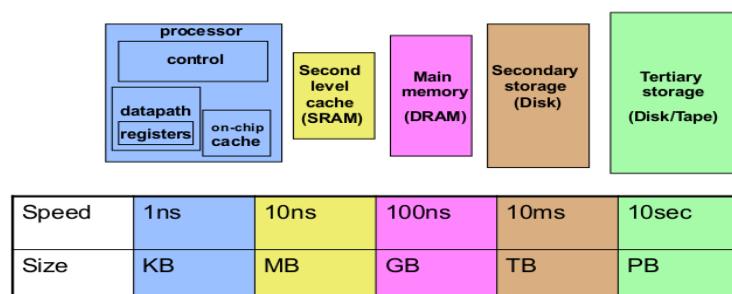
Orokorean, gaur-egungo konputagailuak (super-konputagailu, eramangarri,...) paleloak dira. 1986 – 2002 urteen artean, txip barruko transistore dentsitatea handitzen zen heinean, prozesadore bakarreko konputagailuen eraginkortasuna hobetuz joan zen. Baino teknologi honen garapena muga fisikoetara iritsi zenean, bide honetatik konputagailuen abiadura hobetzea ezinezkoa bilakatu zen. Horrela, 2005.urtetik aurrera fabrikatzaleek konputagailuen gaitasuna hobetzeko, txipan prozesadore bat baino gehiago erabiltzea erabaki zuten.

- Moore's Law (1965). Processor speed doubles every 18 months.
- Moore's Law Reinterpreter (2006). Number of cores per chip can double every two years.

Konputagailuen eredu aldaketa honen ondorioz, algoritmo azkarrak garatzeko kodearen paralelizazio gaitasunari heldu behar zaio. Programazio paralelo teknikak implementatzeko, beharrezko da prozesadore berrien hardware arkitekturak sistema 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 eta funtzionamendua deskribatuko dugu.



5.1. Irudia: Memoria hierarkia.

CPU-k koma-higikorreko eragiketak exekutatzen ditu: erregistroetatik datuak irakurri, eragiketak egin eta emaitza erregistroetan idazten ditu. Memoria nagusia eta erregistroen artean, 2 edo 3 mailako cache memoria dugu: lehen cache memoria (L_1) txikieta eta azkarrena da, eta beste mailak (L_2, L_3, \dots), handiagoak eta motelagoak dira. Memoria nagusian, exekutatzen diren programak eta datuak gordetzen dira (1 – 4 GB artekoa). Azkenik, disko gogorrean konputagailuko datu (argazki, bideo,...) eta erabilgarri ditugun programa guztiak gordetzen dira.

CPUk 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 konsistentzia 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. Programatzaleak, algoritmo eraginkorrak 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 lengoaian 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 11): matrizearen lehen osagaia $a(1, 1)$ behar dugunean, memoria nagusitik Cachera osagai honetaz gain, jarraiko 16 osagaiak 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 berrikin egin gabe. Bigarren implementazioan, kanpo iterazioa zutabeka (Algoritmoa 12): 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 11: Memoria atzipena eraginkorra.

```

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

```

Algoritmoa 12: Memoria atzipena ez-eraginkorra.

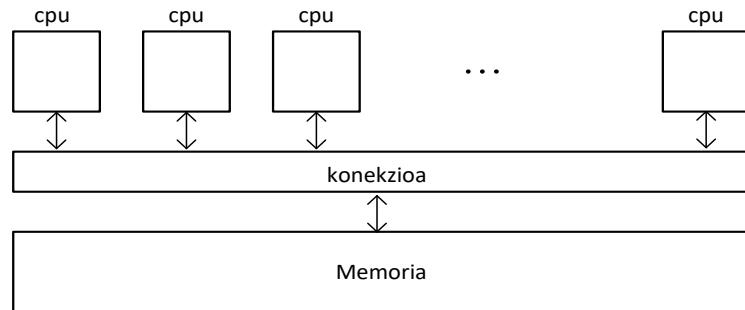
Arkitektura motak.

Prozesadore anitzeko konputagailu hardware berriak, konplexuak eta heterogeneoak dira. Egoera honetan, programatzaleak zaitasun handiak ditu arkitektura berrieik eskaintzen dituzten gaitasunak ondo kudeatzeko [52].

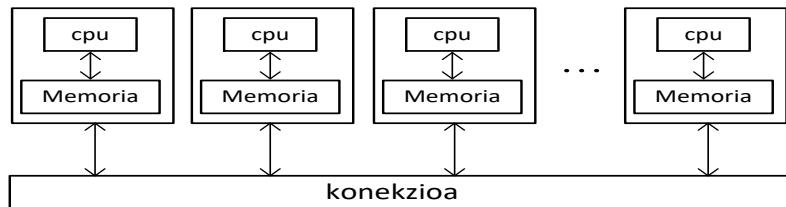
Lehen hurbilpen modura bi sistema nagusi bereiziko ditugu: memoria konpartitutako eta memoria banatutako sistemak. Memoria konpartitutako sistemetan, prozesadore guztiak memoria osoa konpartitzen dute eta inplizituki konpartitutako datuen atzipenaren bidez komunikatzen dira (5.2. irudia). Memoria banatutako sistemetan aldiz, prozesadore bakoitzak bere memoria pribatua du eta esplizituki bidalitako mezuen bidez komunikatzen dira (5.3. irudia). Aipatzekoa da, sistema handietan bi memoria motak nahasten direla, hau da, batetik goiko mailan memoria banatuta alde bat eta bestetik, konputazio unitate bakoitzak memoria konpartitutako aldea.

Hirugarren sistema osagarria ere aipatuko dugu, general purpose GPU computing (Graphical Processor Unit). Jokoak eta animazio industrian, grafiko oso azkarrak beharrak bultzatuta sortutako teknologia da. Oinarrian, imajinak pan-

tailaratzeko prozesagailu asko paraleloan lan egiten dute eta azken hamarkadan, *GPU* unitate hauek zientzia konputaziora zabaldu dira.



5.2. Irudia: Memoria konpartitutako sistemak.



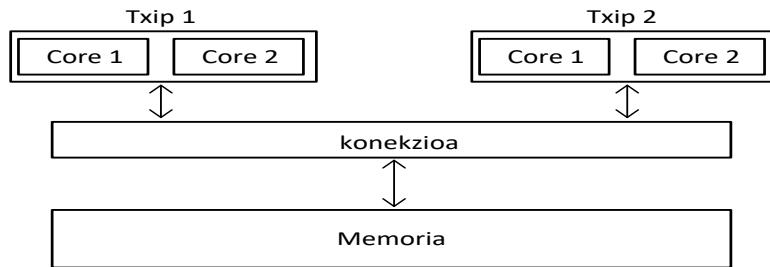
5.3. Irudia: Memoria banatutako sistemak.

Oinarrizko konputagailu paraleloa.

Gure lanerako, oinarrizko konputagailu paraleloak kontsideratuko 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).

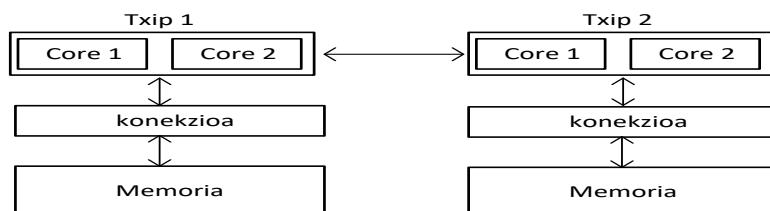
Memoria konpartitutako sistemek artean bi mota bereiziko ditugu:

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 bakotza, hardware berezi baten bidez zuzenean memoria bloke bati konektatuta 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 eskeintzen 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 orohar, 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 13: SIMD (bektorizazioa).

5.4. Programazio lengoaiak.

Fortran eta C, aplikazio zientifikoetan gehien erabiltzen diren programazio lengoaiak dira [34]. Fortran (formula translation) 1950 hamarkadan garatutako goi-mailako lehen lengoiaia 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 lengoiaia, 1970 hamarkadan jaio zen eta "hardwarekiko hurbiltasun"ezagarri nagusiak, konpiladoreari kode eraginkorra sortzeko aukera ematen dio. C lengoiaia *UNIX* sistema eragileari lotuta jaio zen eta hurrengo garapenetan bere izaera askea mantendu du. C lengoaiaren estandardrak 1989, 1999 eta 2011 dira.

Fortran eta C lengoaiak, ez dituzte kodea paraleloan exekutatzeko tresnarik, hau da, ez dago konputazio banatu, eta prozesadore ezberdinaren artean aldi berean exekuzioak zehazteko modurik. Konputazioa paraleloa implementatzeko, bi dira interfaze aplikazio programa (*API*) moduan implementatuta dauden sistema nagusiak [60]:

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

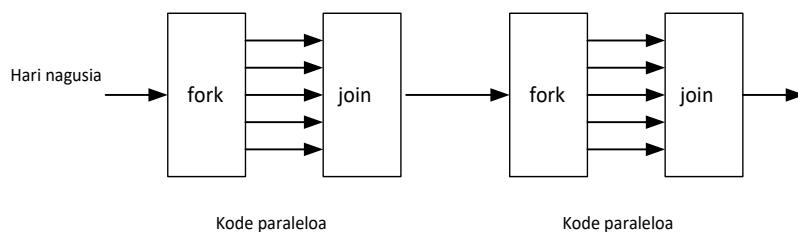
Eraginkortasun altuko konputazioaren (*high performance computing*) programazioa konplexua da: espezializazio handikoa eta konputagailuen hardware-n jakin baterako egokitutakoa. Zaitasun hauek, proiektu zientifikoak aurrera ateratzeko eta mantentzeko arazo asko eragiten ditu. Azken urteotan, eragozpen hau gainditzezko, programazio lengoiaia interesgarriak sortu dira (adibidez, Julia [7] edo Chapel [3]) baina oraindik, hauen arrakasta ikustekoa da.

Azkenik, zientzia konputazioan *problemak ebazteko inguruneak* (Problem Solving Enviroments) deituriko softwareak aipatuko ditugu. Ingurune hauek, programazio leihoko interaktiboa batean, goi-mailako lengoai batean implementazioen garapena eta emaitzen azterketa egiteko aukera eskaintzen dute. Matlab eta Mathematica [71] programazio ingurune nagusienak dira eta guk bi modutara erabili dugu. Lehenik, prototipoak garatzeko tresna gisa: idei berriak garatu eta probatu, implementazioa C lengoaien 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 sistematan 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 zaio.

- Kode paralelizagarria adierazi.
- Hariaren datu pribatuak zehaztu.
- Harien arteko sinkronizazioa.

Adibidea1. C lengoainan, *OpenMP* konpilazio direktibak adierazteko *pragma* hitza lerroaren hasieran idatziko dugu. Adibide honetan, programaren *for-iterazioa* paraleloan exekutatu daitekeela eta hari kopurua bi dela zehaztu dugu.

```
#   include <omp.h>

int thread_count=2;

# pragma omp parallel for num_threads(thread_count)
for (i = 0; i<n; i++)
{
    ! Aginduak
}
```

Konpilazioan, *-fopenmp* aukera zehaztu behar dugu,

```
$ gcc -g -Wall -fopenmp adibidea.c -o adibidea.o.
```

Orokorrean, defektuz aldagai guztiak harien artean konpartituta daude eta aldagaiak pribatuak direla zehazteko, esplizituki adierazi behar da. Goiko adibidea salbuespena da; *for* iterazioaren kontagailua (adibideko *i* aldagai) pribatua da.

Algoritmo batean, kodearen zati bat bakarrik 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, $(1 - 1/S)$ paralizagarria konsideratz, 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 paralelo batean, T_p denborari paralelizazioak duen gainkarga gehitu behar zaio. Gainkarga hau, kontzeptu ezberdinez osatuta dago eta milisegunduko mailako erakigetak izan daitezke.

- Prozesuak edo hariak sortzeko denbora.
- Sinkronizazio denbora.
- Datu konpartitutako komunikazioa.

5.5. Algebra lineal dentsorako liburutegiak.

Zenbakizko integrazioetan, algebra lineala konputazio denboraren eta memoria komunikazioaren zati nagusienetako da.

1. Matrize-Matrize biderketak, Matrize-bektore biderketak, ...
2. Ekuazio sistemek *LU-deskonposaketak*, forward and backward substitution,
...

Algebra linealeko eragiketak implementatzen dituzten kalitate handiko liburutegiak daude ([Netlib](#)) eta algoritmo berriak, liburutegi hauetan oinarritzea gomen-dagarria da [35]. Honako abantailak azpimarratuko ditugu:

1. Garapen berriak egiteko denbora aurrezten da.
2. Problema askotan ondo probatutako softwareak dira.
3. Konplexutasun handikoak dira, modu seguruan eta azkarrean exekutatzeko diseinatu direlako.

Hauek dira, algebra linealerako liburutegi aipagarrienak:

- BLAS : Basic Linear Algebra Subroutines.
- LAPACK: Linear Algebra Package.
- ScaLAPACK: Parallel Distributed Dense Linear Algebra.

Implementazio hauen funtzioak *Fortran* lengoaien garatuta daude eta beste lengoitatik (C, C++, Java, Python) ere deitu daitezke.

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.

Algebra linealeko aplikazioen exekuzio denboraren zati garrantzitsua, behe-mailako eragiten konputazioak ematen du. Behe-mailako eragiketen hauen optimizazioak, konputagailu bakoitzaren araberakoak dira eta espezializazio handia eskatzen du. Konputagailu araberako, algebra linealeko oinarrizko funtziotako eraginkorrik, *BLAS* liburutegia, makina plataforma bakoitzarentzat eskuragarri daude. *BLAS* (www.netlib.org/blas) liburutegian, bektore eta matrizeen arteko funtziotako estandarrak implementatuta daude. *BLAS* liburutegiaren eraginkortasuna matrizeen biderkaduraren funtzioaren (*cblas_dgemm*) bidez aztertu dugu eta gure implementazio arrunta baino $10 \times$ azkarragoa dela baiezztatu dugu (Taula 5.1.).

Denetara 142 errutinaz osatzen da 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: $\approx 2n^2/n^2 = 2$.

5.1. Taula: BLAS liburutegiaren eraginkortasuna. Portatilean egindako esperimentua: Ubuntu 16.04 bertsioa eta 4-CPU. Gure C lengoaiako garapena (C-arrunta) eta BLAS liburutegiaren `cblas_dgemm()` implementazioak konparatu ditugu. n tamainako ezberdineko matrizeak biderkatu ditugu eta $ntest$ kopurua eragiketa aritmetikoa berdinak izan daitezen finkatu dugu

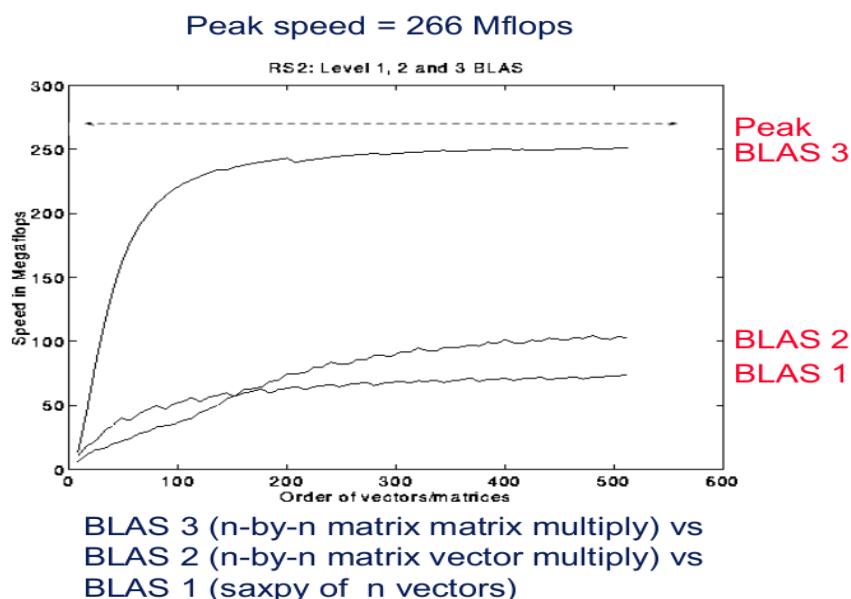
n	ntests	C-Arrunta		cblas_dgemm	
		Wall T.	CPU T.	Wall T.	CPU T.
10	5.00×10^8	478.	478.	205.	206.
20	6.25×10^7	491.	491.	92.	91.
30	1.85×10^7	474.	474.	78.	78.
40	7.81×10^6	523.	523.	66.	66.
50	4.00×10^6	493.	493.	64.	64.
60	2.31×10^6	479.	479.	58.	58.
70	1.45×10^6	475.	475.	43.	170.
80	9.76×10^5	469.	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.

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 aldiz, konputazio intentsitatea handiagoa da eta ezaugarri honi esker, konputagailuaren konputazio gaitasuna ondo aprobetxatu ahal izango da (irudia 5.8.).



5.7. Irudia: BLAS speeds.

Fabrikatzaile bakoitzak optimizatutako BLAS liburutegia du (AMD-ACML, Intel-MKL) eta hauek, paraleligarritasun gaitasuna (multi-threaded) dute. Bestalde, optimizatutako BLAS instalazioa, *ATLAS* (Automatically Tuned Linear Algebra Software) izeneko aplikazioaren bidez ere egin daiteke. Implementazio guztiekin, interfaze berdina erabiltzen dute eta beraz, BLAS-en oinarritutako garapena edozein konputagailuan erabili daiteke (portable).

Aipatu dugunez, *BLAS* Fortran lengoaiaren implementatuta dago eta C-lengoaiatik deitzeko kontu handiz egin beharra dago. C lengoaiatik *BLAS* funtzioen erabilpena errezteko, *BLAS* funtzioei deitzeko *cblas* interfazea erabiltzea gomendagarria da.

LAPACK.

LAPACK 1992. urtean garatu zen [33] eta algebra linealaren problemak ebazteko programen liburutegia da. Jatorrizko bertsioa *Fortran 77* lengoaiaren implemen-tatuta dago eta liburutegiaren dokumentazioa nahiz kodea *Netlib* errepositorioan (<http://www.netlib.org/lapack/>) aurkitu daiteke. Matrize dentsoeta-rako garatuta dago eta honako problemetarako errutinak biltzen ditu:

- Ekuazio sistema lineal: $AX = b$.
- Linear least square problems: choose x to minimize $\|Ax - b\|$.
- Eigenvalues problems.
- Balio singularren deskonposaketa (SVD).

LAPACK liburutegiaren diseinua, konputagailu sekuentzial eta memoria kon-partitutako konputagailuetan bateragarria izateko oinarritzen da, zeinetan eragin-kortasuna *BLAS* funtzió optimizatuen menpe dagoen. LAPACK liburutegiaren funtzióen implementazioa, gehien bat *BLAS-3* taldeko funtzióen erabilpenean oinarritzen da.

LAPACK liburutegiaren hainbat garapen daude eta horien artean, *LAPACKE* liburutegia C-lengoaiatik LAPACK funtzióei deitzeko interfazea da.

LAPACK liburutegiaren errutinak hiru taldeetan banatzen dira:

1. Problema osoa ebazten dituzten errutinak (drivers). Talde honetan funtzió arruntak eta funtzió espezializatuak daude.

Adibidea.

- (a) LAPACKE-dgelsv (LAPACK-ROW-MAJOR, n, nrhs, A, lda, ipiv, B, ldb);

Computes the solution to system of linear equations $A * X = B$ for GE matrices.

- (b) LAPACKE-dgelsvx (...);

Computes the solution to the system of linear equations with a square coefficient matrix A and multiple right-hand sides, and provides error bounds on the solution.

- (c) LAPACKE-dgelsvxx (...)

Uses extra precise iterative refinement to compute the solution to the system of linear equations with a square coefficient matrix A and mul-tiple right-hand sides

2. Konputazio errutinak. Lan zehatz bat exekutatzen duen errutinak.

Adibidea.

(a) LAPACKE-dgetrf (LAPACK-ROW-MAJOR, n, m, A, lda, ipiv);

Errutina honek A ($n \times m$) tamainako matrizearen LU faktorizazioa kalkulatzen du, $A = P * L * U$.

(b) LAPACKE-dgetrs (LAPACK-ROW-MAJOR,trans,n,nrhs,A,lda,ipiv,B,ldb);

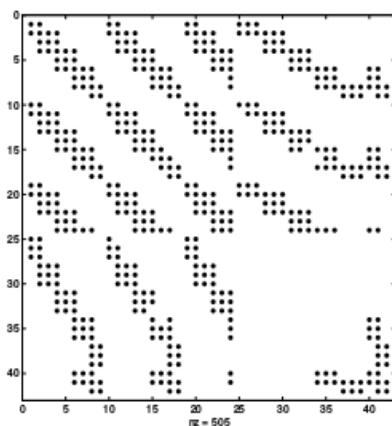
Errutina honek $A * X = B$ ekuazio sistemaren X soluzioa kalkulatzen du.

3. Errutina auxiliarrak.

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} ,

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



5.8. Irudia: Matrize bakanak.

Bakantasunak memoria eta exekuzio denbora gutxitu dezake.

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) Bakna: $\mathcal{O}(n_{nz})$, eragiketa aritmetikoak.

Matrize bakanentzako oraindik *BLAS* liburutegia estadarra ez dagoela <http://www.netlib.org/blas/blast-forum> konprobatu dugu. Aldiz, Intel-en *Math Kernel Library* izeneko liburutegian matrize bakanentzako *BLAS* eta *LAPACK* funtziak garatuta dituzte <https://software.intel.com/en-us/intel-mkl-support/documentation>.

5.6. Konpiladorea.

Konpiladorearen zereginan 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 erraztu.

Konpiladore aukera 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. Hainbat konpiladore komertzialak daude (adibidez intel-en *icc* konpiladore).

Konpilazio optimizazioak. Konpiladore guztiekin optimizazio maila estandarrak eskaintzen dituzte. Orokorrean, hurrengo kode optimizazioak izango ditugu:

1. *-O0*. Kode optimizazio nagusienak aplikatzen deneko aukera da. Kodea *debugger* moduan aztertzen ari garenean gomendatzen da.
2. *-O2*. Kode azkarra sortzeko aukera gomendarriena. Maila gehiagoko optimizazioak aplika daitezke, baina optimizazio ez egokiak gerta daitezke eta beraz, arriskutsuak dira.

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
```

```
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.

Eranskinean *Makefile* legoaiaren oinarrizko adibideak eman ditugu.

5.7. Laburpena.

Best practices for code vectorization and parallelization, and additional tips and tricks.

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. Badaude kodearen exekuzio denboraren analisia egiteko tresnak (adibidez gprof). Algoritmoaren funtzio bakoitzaren exekuzio denborari buruzko informazio erabilgarria lortuko dugu. Zenbait gauza modu simplean azkartu daitezke baina zenbait beste gauza azkartzeko esfuertzu handia eskatu de-zake.
4. Optimizatutako beste hainbat kode erabiltzea komenigarria da. LAPACK aljebra lineal paketea Fortran eta C-lengoaitatik deitu daiteke. Eraberean, LAPACKek BLAS subrutinak erabiltzen ditu. Subrutinak hauek matrizen arteko biderketak, "inner product", ... BLAS konputagailu arkitektura ez-berdinatarako optimizatutako bertsioak daude.

Atal honi dagokion gomendatutako bibliografia. Introduction to High Performance Scientific Computing [21] Best Practices for Scientific Computing [68]

II. Atala

Ekarpenak.

6. Kapitulua

IRK: Puntu-Finkoa.

6.1. Sarrera.

Puntu-finkoan oinarritutako sistema Hamiltondar ez-zurrunen doitasun altuko zenbakizko integrziotarako, IRK metodoaren implementazioa proposatuko dugu. Konputazioetan koma-higikorreko aritmetika erabiltzen denez, doitasun altuko integratioetan, biribiltze errore nagusitzen da. Hortaz, epe luzeko doitasun altuko zenbakizko integratioetarako implementazioetan, biribiltze errorearen eragina txikia izatea eta biribiltze errorearen estimazioa ezagutzea interesgarria da.

Integratioaren 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 zehatztuko duela suposatuko dugu.

Lehenengo, Hairer-en IRK metodoaren implementazioa [30] aztertuko dugu eta implementazio horretan aurkitu ditugun hainbat arazo azalduko ditugu. Ondoren, IRK implementazioa hobetzeko gure proposamenak egingo ditugu eta azkenik, zenbakizko integracioen bidez bere abantailak erakutsiko ditugu.

6.2. Hairer-en implementazioa.

IRK implementazio estandarra.

Gure abiapuntua, Hairer-ek [30] proposatutako IRK metodoaren implementazio da. Lan honetan, puntu-finkoan oinarritutako IRK metodo simplektikoaren implementazio estandarrean, biribiltze erroroeak energian errore sistematiko bat eragi-

ten zuela ohartu ziren. Metodo sinplektiko esplizituetan, ordea, ez da horrelakorik gertatzen. Bere azterketaren ondorioaren arabera, errore sistematiko honen jatorriak bi direla aipatzen da:

1. Praktikan aplikatutako IRK metodoa ez da sinpletikoa, integrazioan $a_{ij}, b_i \in \mathbb{R}$ koefiziente zehatzak erabili ordez, biribildutako $\tilde{a}_{ij}, \tilde{b}_i \in \mathbb{F}$ erabiltzen direlako.
2. Puntu-finkoaren geratze irizpide estandarrak,

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

(finkatutako δ tolerantzia baten araberakoa), urrats bakoitzean errore sistematikoa eragiten du.

Konponbideak.

Biribiltze errorearen eragina gutxitzeko helburuarekin, implementazio estandardearren honako aldaketak proposatu zituzten:

1. Doitasun handiagoko koefizienteak erabiltzea, hauetako bakoitza bi komahigikorreko koefizienteen batura kontsideratzu,

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

non $a_{ij}^* > \tilde{a}_{ij}$ eta $b_i^* > \tilde{b}_i$ diren.

Adibidez, koefizienteak era honetan zehaztu daitezke,

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

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 algoritmoa.

Hairer-ek bere *Fortran* implementazioa eskuragarri du ([Fortran kodea](#)). Jarraian, bere implementazioaren algoritmoa ([Algoritmoa.14](#)) eta erabilitako batura konpensatuaren teknika ([Algoritmoa.15](#)) 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
         $k = k + 1;$ 
         $F_{n,i}^{[k]} = f(Y_{n,i}^{[k-1]});$ 
         $Y_{n,i}^{[k]} = y_n + h \left( \sum_{j=1}^s a_{ij}^* F_{n,j}^{[k]} \right) + h \left( \sum_{j=1}^s \tilde{a}_{ij} F_{n,j}^{[k]} \right);$ 
         $\Delta^{[k]} = \max_{i=1,\dots,s} \|Y_{n,i}^{[k]} - Y_{n,i}^{[k-1]}\|_\infty;$ 
    end
     $(y_{n+1}, e_{n+1}) \leftarrow BaturaKonpensatua(y_n, e_n, F_n^{[k]});$ 
end

```

Algoritmoa 14: Hairer (IRK)

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

```

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

```

Algoritmoa 15: Hairer (Batura konpensatua)

Hairer-en implementazioaren arazoak.

Hairer-ek bere implementazio berria, *Hénon-Helies* eta eguzki-sistemaren kanpo planeten problemetarako energian errore sistematikorik ez zegoela baieztago zuen. Energia errorea, $k\sqrt{t_n}$ espresioaren arabera handitzen dela erakutsi zuen eta beraz, implementazio berriak *Brouwer legea* [27] betetzen duela ondorioztatu zuen. Era

berean, integrazioen azterketa estatistikoa egin zuen biribiltze errorearen ausazko-tasuna baieztatzeko. Perturbatutako 1.000 hasierako balio ezberdinien integrazio egin zituen eta integrazio horien energia errorearen batezbestekokoa (μ) zero eta desbiazio tipikoa (σ) $\sqrt{t_n}$ espresioaren proportzionala zirela erakutsi zuen. Azkenik, integrazioen bukaerako energi erroreen histogramak, $N(\mu, \sigma)$ distribuzio normala betetzen duela erakutsi zuen.

Gure aldetik Hairer-en implementazioarekin egindako zenbakizko integracio berriak egin ditugu, eta zenbait kasuetan geratze erizpidea ez duela ondo funzionatzen konprobatu dugu. Gainera, bere implementazioaren biribiltze errorearen propagazioa optimoa ez dela uste dugu.

6.3. Gure implementazioa.

IRK metodoaren puntu-finkoaren implementazioan lau proposamen berri egingo ditugu. Lehen bi proposamenak Hairer-ek bere lanean proposatutako konponbideen hobekuntzak dira. Batetik, IRK-ren birformulazio bat erabiliz, IRK metodoaren koma-higikorreko koefizienteak simplektizidade baldintza zehazki betetzea lortuko dugu. Bestetik, puntu-finkoaren geratze irizpidean arazo batzuk topatu ditugu eta arazo hauek gainditzen dituen geratze irizpide sendoagoa garatu dugu. Azken bi proposamenetan, batura-konpensatuaren hobekuntza eta biribiltze errorea monitorizatzeko teknika proposatzen dira.

6.3.1. Metodoaren birformulazioa (1.proposamena).

IRK metodoa definitzen duten $a_{ij}, b_i \in \mathbb{R}$ koefizienteen ordez (2.9), aplikatzen ditugun biribildutako koefizienteen $\tilde{a}_{ij}, \tilde{b}_i \in \mathbb{F}$ hurbilpenek ez dute simplektizidade baldintza [38] betetzen,

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (6.4)$$

Hairer-ek doitasun handiagoko koefizienteak (6.2) erabiltzea proposatu zuen baina era honetan, ez da simplektizidade baldintza betetzen. Arazo hau gainditze-ko, IRK metodoa era honetan birformulatuko dugu,

$$Y_{n,i} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}, \quad L_{n,i} = h b_i f(Y_{n,i}), \quad i = 1, \dots, s, \quad (6.5)$$

$$y_{n+1} = y_n + \sum_{i=1}^s L_{n,i}, \quad (6.6)$$

non

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

Eta simplekzidate baldintza (6.4) modu honetan berridatziko dugu,

$$\mu_{ij} + \mu_{ji} - 1 = 0, \quad 1 \leq i, j \leq s. \quad (6.7)$$

Birformulazio berri honen simplektizidate baldintzaren espresioan, ez da biderketarik agertzen eta ezaugarri honek, baldintza zehazki betetzen duten $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak aurkitzeko bidea errezten du. Jarraian, koefizienteak finkatzeko urratsak azalduko ditugu:

1. μ_{ij} koefizienteak.

S -ataleko Gauss metodoetan, diagonaleko koefizienteek balio finkoa dute ($\tilde{\mu}_{ii} := 1/2$, $i = 1, \dots, s$) eta balio honek, koma-higikorreko adierazpen zehatza du.

Gainontzeko koefizienteak, bi urratsetan kalkulatuko ditugu:

- Lehenengo, koefiziente matrizearen behe-diagonaleko balioak finkatu 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 (ikus. 4.1), $1/2 < |\mu_{ij}| < 2$ denez, $1 - \tilde{\mu}_{ij}$ balioak koma-higikorreko adierazpen zehatza izango duela ziurtatzen du.

Laburtuz, hauek ditugu birformulatutako simplektizitate baldintza (6.7) zehazki betetzen duten koma-higikorreko $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak,

$$\tilde{\mu} = \begin{pmatrix} 1/2 & 1 - fl(\mu_{21}) & \dots & 1 - fl(\mu_{s1}) \\ fl(\mu_{21}) & 1/2 & \dots & 1 - fl(\mu_{s2}) \\ \vdots & \ddots & & \vdots \\ fl(\mu_{s1}) & fl(\mu_{s2}) & \dots & 1/2 \end{pmatrix}. \quad (6.8)$$

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} koefizienteetik abiatuta (ikus 2.2.3.atala), formulazio berriari dagozkion interpolazio ν_{ij} koefizienteak era honetan definituko ditugu,

$$Y_{n,i}^{[0]} = y_n + h \sum_{j=1}^s \nu_{ij} L_{n-1,j}, \quad \nu_{ij} = \lambda_{ij}/b_j \quad 1 \leq i, j \leq s. \quad (6.9)$$

6.3.2. Geratze irizpidea (2.proposamena).

Ekuazio implizituaren (7.23) soluzioaren hurbilpena lortzeko puntu-finkoko iterazioa era honetan definituko dugu: iterazioaren abiapuntua $Y_i^{[0]}$ finkatu eta $k = 1, 2, \dots$ iterazioetarako $Y_i^{[k]}$ hurbilpenak lortu dagokigun geratze irizpidea bete arte.

```

for ( $k=1,2,\dots$  konbergentzia lortu arte) do
     $L_i^{[k]} = hb_i f(Y_i^{[k-1]});$ 
     $Y_i^{[k]} = y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k]}, \quad i = 1, \dots, s;$ 
end

```

Algoritmoa 16: Puntu-finkoko iterazioa.

IRK metodoaren implementazio estandarrean, geratze irizpidea honakoa da,

$$\Delta^{[k]} = (Y_1^{[k]} - Y_1^{[k-1]}, \dots, Y_s^{[k]} - Y_s^{[k-1]}) \in \mathbb{F}^{sd},$$

$$\|\Delta^{[k]}\| \leq tol \quad (6.10)$$

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 $Y_i^{[k]}$ hurbilpenak biribiltze errorea baino errore handiagoa izango du. Gogorarazi ere, Hairer-ek [30] iterazio errorea modu sistematikoan metatzeko delako.

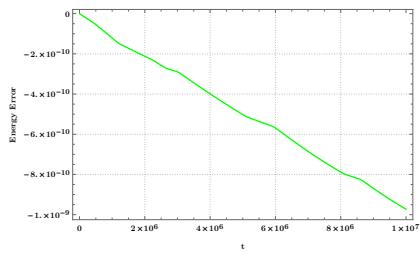
Hairer-ek proposatu zuen geratze irizpidea gogoratuko dugu; $\Delta^{[k]} = 0$ (puntu-finkora iritsi delako); edo $\Delta^{[k]} \geq \Delta^{[k-1]}$ (biribiltze errorea nagusi delako). Orokorrean, geratze irizpide honek ondo funtzionatzen du baina esperimentalki zenbait kasutan, iterazioak goizegi geratzen direla ikusi dugu. Hairer-ek, eguzki-sistemaren kanpo planeten problemaren $h = 500/3$ eguneko urrats luzeerarekin egindako integrazioan ondo funzionatzen du baina $h = 1000/3$ urrats luzeera aukeratuz, energia errorearen tamaina oso handia da. Energia errore erlatiboaren

garapena (6.1.Irudia) irudikatu dugu. Integrazioaren lehen urratsaren iterazioak aztertuz,

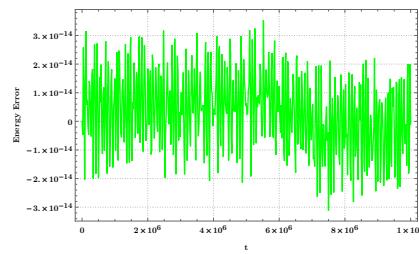
$$\|\Delta^{[1]}\| > \|\Delta^{[2]}\| \dots > \|\Delta^{[12]}\| = 3.91 \times 10^{-14} \leq \|\Delta^{[13]}\| = 4.35 \times 10^{-14}$$

iterazioa 13.iterazioan geratzen dela konprobatu dugu. Puntu-finkoko iterazioa goizegi geratu da, hurrengo iterazioetan, $\|\Delta^{[13]}\| > \|\Delta^{[14]}\| > \|\Delta^{[15]}\| > \|\Delta^{[16]}\| = 0$ gertatzen baita.

Hairer-en geratze erizpidea aztertu ondoren bi ondorio atera daitezke. Batetik, ezin daiteke suposatu $\Delta^{[k]}$ segida beherakorra denik. Bigarrenik, $\Delta^{[k]} \geq \Delta^{[k-1]}$ baldintzak, biribiltze errorea nagusia dela adierazten duen arren, badago $j \in \{1, \dots, sd\}$ osagairik, $|\Delta_j^{[k]}| < |\Delta_j^{[k-1]}|$ hobetzeko tartea duena.



(a) OSS: Hairer's stopping criterion



(b) OSS: New stopping criterion

6.1. Irudia: Evolution of relative error in energy for the outer solar system problem (OSS) with the original unperturbed initial values in [30] and doubled step-size ($h = 1000/3$ days). (a) Hairer's stopping criterion, (b) New stopping criterion

Arazoari soluzioa emateko, geratze erizpidea berri bat proposatuko dugu. Honako notazioa finkatuko dugu,

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

Iterazioak $k = 1, 2, \dots$ jarraitza, $\Delta^{[k]} = 0$ bete arte edo honako baldintza bi iterazio jarraietan betetzen den artean,

$$\forall j \in \{1, \dots, sD\}, \quad \min \left(\{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} / \{0\} \right) \leq |\Delta_j^{[k]}|. \quad (6.11)$$

Hairer-ek, eguzki-sistemaren kanpo planeten problemaren $h = 1000/3$ eguneko urrats luzeerarekin egindako integrazioa errepikatu dugu baina oraingoan geratze erzipide berria aplikatuz. Energia errorearen eboluzioa (6.1.Irudia) irudikatu dugu.

Tolerantzi testa.

Iterazio gehienetan, j guztietaurako $\Delta_j^{[k]} = 0$ delako geratuko da. Dena den j batetarako $\Delta_j^{[k]} \neq 0$ izanik iterazio geratzen bada, orduan puntu-finkoaren iterazioa onargarria den ala ez erabaki behar dugu. Iterazioa geratu daiteke biri

orduan iterazio gehigarri bat emango dugu. Iterazio gehigarri bat emanet, esperimentanki puntu-finkoa lortzen den urratsen portzentaia nabarmen handitzen dela konprobatu dugu.

Puntu-finkoaren iterazioak amaitzerakoan, urratsa eman aurretik erabiltzaileak definitutako tolerantzia lortu den ala ez aztertuko dugu. Horretarako iterazioaren azken bi hurbilpenak konparatuko ditugu eta honako notazioa finkatuko dugu,

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

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 integrazio amaituko dugu.

Tolerantzia ez dugu erabiliko puntu-finkoko iterazio geratzeko, behin iterazioa geratu denean, emaitza onargarria den ala ez erabakitzeko baizik. Puntu-finkoko iterazioa konbergentzia lortu duelako edo konbergentzia arazoak izan direlako gera daiteke. Doitasun zehatzean, biribiltze errorearen eragina handia izan daiteke eta beraz, ez dugu tolerantzia estua erabiliko. Diskriminatu nahi dugu, aritmetika zehatzean konbergitu duen ala ez. Biribiltze errorea ez balitz konbergitu duen erabaki nahi dugu.

6.3.3. Biribiltze errorea gutxitzeko teknikak (3.proposamena).

Koma higikorreko aritmetika atalean (ikus 4.4.atala), baturan nahiz biderketan egindako biribiltze errore zehatza modu errazezan kalkulatu daitekeela ikusi genuen. Eragiketa hauen biribiltze erroreak, ondorengo konputazioetan erabiliko ditugu soluzioaren doitasuna hobetzeko.

Batura errekurtsiboetan, konputazioaren doitasuna hobetzeko teknikari *batura konpensatua* esaten zaio eta zenbakizko integrazioetan erabili ohi da. Atal

honetan, 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 kalkulatza proposatuko dugu.

Goian aipatutako ideia, beste ikuspegi batetik ere azaldu daiteke. Ikuspegi honen arabera, gure zenbakizko soluzioa, bi *double* 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,

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

Batura konpensatua (aportazioa).

$y_{n+1} \in \mathbb{R}^d$, $y_{n+1} = \tilde{y}_n + \tilde{\delta}_n$ 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 $n = 1, 2, \dots$ **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 17: Batura konpensatua.

baturan egindako biribiltze errore zehatza da,

$$y_{n+1} = \tilde{y}_{n+1} + e_{n+1}. \quad (6.12)$$

Horregatik, IRK metodoaren implementazioan, implizituki $Y_{n,i}$ atalak askatzeko ekuazioetan, \tilde{y}_n ordez ($\tilde{y}_n \oplus \tilde{e}_n$) erabiltzea proposatzen dugu,

$$L_{n,i}^{[k]} = hb_i f(Y_{n,i}^{[k-1]}), \quad Y_{n,i}^{[k]} = \tilde{y}_n \oplus \left(\tilde{e}_n \oplus \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]} \right). \quad (6.13)$$

Aldaketa honekin, lortutako zenbakizko soluzioaren doitasuna batura konpensatu estandarrarekin baino pixka bat hobea izango dela espero dugu.

Biderketaren biribiltze errorea.

Urratsa emateko unean, $L_{n,i} = hb_i f(Y_{n,i})$, $i = 1, \dots, s$ biderketen biribiltze errorea kalkulatu eta e_n gaiari gehituko diogu. Biderketaren biribiltze errorea jasotzeko konputagailuaren *FMA* eragiketan oinarrituko gara eta (4.4.)ataleko (8.algoritmoa) aplikatuko dugu.

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

Baturaren biribiltze errorea.

$$\delta_n = \left(\sum_{i=1}^s L_{n,i}^{[k]} \right) + \beta_n \text{ baturaren biribiltze errorea.}$$

Function BaturaKonpensatua ($y_n, \beta_n, L_n^{[k]}$)

```

 $s_0 = y_n$ 
 $ee = \beta_n$ 
for  $i \leftarrow 1$  to ( $s$ ) do
     $s_1 = s_0$ 
     $\delta = L_{n,i}^{[k]} + ee$ 
     $s_0 = s_1 + \delta$ 
     $ee = (s_1 - s_0) + \delta$ 
end
```

```

 $y_{n+1} = s_0$ 
 $e_{n+1} = ee$ 
return ( $y_{n+1}, e_{n+1}$ )
```

Algoritmoa 18: BaturaKonpensatua

6.3.4. Biribiltze errorearen estimazioa (4.proposamena).

Zenbakizko integrazioaren biribiltze errorearen estimazioa, bigarren zenbakizko integrazio baten soluzioaren differentzia gisa kalkulatuko dugu. Bigarren integrazio honetan, $\tilde{L}_{n,i}$ terminoak mantisa txikiagoko zenbakira biribilduko ditugu, doitasun gutxiagoko soluzioa lortzeko.

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

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

Algoritmoa 19: floatR

Funtzio honek itzultzen duen balioa, $(m-r)$ -biteko doitasuneko koma-higikorreko zenbakia da. Beste modu batera esanda, m biteko koma-higikorreko x zenbakia-ren azken r bitak zeroan jartzen dituen funtzioa da.

Biribiltze errorearen estimazioa, zenbakizko soluzio nagusiaren $(y_n + e_n)$ eta r ($r < m$) balio txiki baterako (adibidez $r = 3$) kalkulatutako bigarren zenbakizko soluzioaren $(\tilde{y}_n + \tilde{e}_n)$ arteko diferentziaren norma bezala kalkulatuko dugu.

$$\text{estimazioa}_n^i = \| (y_n^i + e_n^i) - (\tilde{y}_n^i + \tilde{e}_n^i) \|_2, \quad i = 1, \dots, d. \quad (6.14)$$

Gure algoritmoan estimazioa zuzenean lortzeko, bi integrazioak sekuentzialki modu eraginkorrean kalkulatuko ditugu. Urrats bakoitzean, bi integrazioen Y_i, \tilde{Y}_i ($i = 1, \dots, s$) ataletako balioak, biribiltze errorea estimazio handiegia ez den artean, antzekoak mantentzen dira. Beraz, bigarren integrazioan iterazio kopuru txikia beharko dugu, lehen integrazioaren bukaerako $Y_i^{[k]}$ ($i = 1, \dots, s$) atalen balioak, bigarren integrazioaren $\tilde{Y}_i^{[0]}$ ($i = 1, \dots, s$) atalen hasieraketarako erabiltzen baditugu (ikus 20.algoritmoa).

6.3.5. Algoritmoa.

Formulazio berriari dagokion algoritmo orokorra laburtuko dugu 21.algoritmoa.

Zenbakizko soluzioa.

Integrazio tarte $[t_0, t_{end}]$ eta urrats tamaina h bada, emandako urrats kopurua $N = (t_{end} - t_0)/h$ izango da. Zenbakizko soluzioa fitxategi bitar batean idatziko da eta erabiltaileak definitutako m urratsero itzuliko da.

k. integrazioan N urrats eman baditugu eta m finkatutako soluzioen frekuentzia bada, $t_i = t_0 + i * (m \cdot h)$, $i = 1, \dots, N/m$ uneetarako zenbakizko soluzioa lortuko dugu. Bi integrazio mota exekutatu daitezke:

1. Integrazio arrunta.

Integrazioan itzultzen den fitxategiaren egitura honakoa da:

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

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

```

for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $Y_n^{[0]} = G(Y_{n-1}, h);$ 
    Lehen Integrazioa;
     $(y_{n+1}, e_{n+1}) \leftarrow BaturaKonpensatua(y_n, \beta_n, L_n^{[k]});$ 
    if ( $initwithfirst$ ) then
         $\tilde{Y}_n^{[0]} = Y_n^{[k]} + (\tilde{y}_n - y_n);$ 
    else
         $\tilde{Y}_n^{[0]} = G(\tilde{Y}_{n-1}, h);$ 
    end
    Bigarren Integrazioa;
     $(\tilde{y}_{n+1}, \tilde{e}_{n+1}) \leftarrow BaturaKonpensatua(\tilde{y}_n, \tilde{\beta}_n, floatR(\tilde{L}_n^{[k]}, r));$ 
     $estimation_{n+1} = \|(y_{n+1} + e_{n+1}) - (\tilde{y}_{n+1} - \tilde{e}_{n+1})\|_2;$ 
end

```

Algoritmoa 20: RKG2: errore estimazioa

2. Integrazioa errore estimazioarekin.

Errorearen estimazioa itzultzen duen integrazioaren emaitza honakoa da,

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

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

Zenbakizko soluzioa doitasun bikoitzeko bi zenbakien batura gisa adieraziko dugu (urretik azaldutakoaren erreferentzia),

$$(y_i, e_i) = (q_i, p_i, eq_i, ep_i), \quad y_i, e_i \in \mathbb{R}^d.$$

$$(q_i^{[k]} + eq_i^{[k]}, p_i^{[k]} + ep_i^{[k]}) \approx (q(t_i)^{[k]}, p(t_i)^{[k]}), \quad i = 1, \dots, N/m.$$

```

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

```

Algoritmoa 21: IRK (puntu-finkoa).

6.4. Implementazio estadarraren xehetasunak.

Hamiltondar banagarriak.

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 ordeneko ekuazio diferentzialak $\ddot{q} = f(q)$ era honetako ekuazio diferentzialen kasu partikularrak dira.

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

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

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

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

Era honetako problemetan, funtsean *IRK* algoritmo orokorra (alg.1) aplikatuko dugu. Baino Hamiltondarraren egituraren abantaila aprobetxatuz, puntu-finkoaren iterazioaren konbergentzia hobetuko dugu,

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

Algoritmoa 22: Puntu-finkoaren iterazioa (Gauss-Seidel).

Bigarren ordeneko EDA.

Bigarren ordeneko ekuazio diferentzialen $\ddot{q} = f(q)$ (*Runge-Kutta-Nyström*) azterketa egiteko, modu baliokide honetan idatziko dugu,

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

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

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

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

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

IRK algoritmoa-III (bigarren ordeneko 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 23: IRK algoritmoa-III (bigarren ordeneko EDA)

Bigarren ordeneko ekuazio diferenzialak ditugunean, puntu-finkoko iterazioa,

```

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 24: Puntu-finkoko iterazioa (bigarren ordeneko EDA)

6.5. Esperimentuak.

6.5.1. Sarrera.

Lau problemetarako esperimentuetan, integrazio parametroak bateratu ditugu.

1. $s = 6$ ataletako Gauss IRK metodoa aplikatu dugu.
2. Epe luzeko integrazioak aztertu ditugu.
3. Urratsa, trunkatze errorea biribiltze errorea baino txikiagoa izan dadin aukeratu dugu.
4. Integrazio mota bakoitza $P = 1.000$ perturbatutako hasierako balio ezberdinekin integratu dugu. Perturbazioen kalkulurako $k = 2^{-20}$ balioa finkatu dugu, hau da, $\mathcal{O}(10^{-16})$ mailako perturbazioak aplikatu ditugu problema guztiatarako.
5. Kokapen errorearen estimazioaren kalkuluan, integrazio nagusian $rdigits1 = 0$ eta bigarren integrazioan $rdigits2 = 3$ balioak erabili ditugu.

Lau probema ezberdinatarako egin ditugu zenbakizko esperimentuak eta baikotzean zehazki erabilitako integrazio tartea eta urratsa tamainak hauek izan dira:

1. Pendulu bikoitza ez-kaotikoa (NCDP).

$$\begin{aligned} t_0 &= 0, \quad t_{end} = 2^{12}, \quad h = 2^{-7}, \\ m &= 2^{10}, \quad L = 512. \end{aligned}$$

2. Pendulu bikoitza kaotikoa (CDP).

$$\begin{aligned} t_0 &= 0, \quad t_{end} = 2^8, \quad h = 2^{-7}, \\ m &= 2^6, \quad L = 512. \end{aligned}$$

3. Outer solar system (6-Body).

$$\begin{aligned} t_0 &= 0, \quad t_{end} = 10^7, \quad h = 500/3, \\ m &= 120, \quad L = 500. \end{aligned}$$

4. Solar system (10-Body).

$$\begin{aligned} t_0 &= 0, \quad t_{end} = 10^6, \quad h = 2, \\ m &= 2000, \quad L = 250. \end{aligned}$$

6.5.2. Energia errore jatorriak.

Gure doitasun bikoitzeko implementazioaren zenbakizko soluzioaren $\tilde{y}_n + e_n \approx y(t_n)$ ($n = 1, 2, \dots$) errorea, jatorri ezberdineko errore mota ezberdinen konbinazioa da:

1. Trunkatze errorea. Hasierako baliodun problemaren soluzio zehatza $y(t_n)$, ($n = 1, 2, 3, \dots$), (6.5) metodoa aplikatzuz (b_i, μ_{ij} koefiziente zehatzekin) lortutako zenbakizko soluzioa y_n ordezkatzerakoan eragindako errorea.
2. Iterazio errorea. Praktikan, puntu-finkoko iterazioa (16) K kopuru finitua aplikatzen da, eta $L_{n,i}, Y_{n,i}$ ($i = 1, \dots, s$) soluzioa, $L_{n,i}^{[K]}, Y_{n,i}^{[K]}$ hurbilpenarekin ordezkatzen da. Hurbilpen honen arabera dagokion zenbakizko soluzioa kalkulatzen da,

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

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 eragindako bi mailetako errorea. Batetik, K iterazio finituetan puntu-finkora iristeagatik eragindako ekidin ezineko iterazio errorea. Bestetik, \tilde{f} konputazioan gertatutako 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, doitasun bikoitzean kalkulatzeagatik eragindako errorea.

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

1. Implementazio zehatza.

Konputazio guztia (ekuazio diferentzialaren funtzioaren ebaluazioa barne) doitasun laukoitzeko (128-bit) koma higikorreko aritmetikan egindako integrazioa. Zenbakizko integrazio hau, trunkatze errorea estimatzeko eta errore globala kalkulatzeko erreferentzi gisa (soluzio zehatza) kontsideratuko dugu.

2. Implementazio superideala.

Zenbakizko integrazio hau iterazio errorea estimatzeko erabiliko dugu. Konputazio guztia doitasun laukoitzean egindako integrazioa baina geratze erizpidea, doitasun bikoitzean neurtuko dugu,

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

3. Implementazio idealia.

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

4. Implementazio sasi-idealai.

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

We next plot (Fig. 6.2.), for each of the three considered initial value problems, the evolutions of the energy errors corresponding to the items A–D in previous list. We confirm that for the time-step h considered, truncation error is below round-error. The use of double precision coefficients ($\tilde{b}_i, \tilde{\mu}_{ij}$) is do not effect in the propagation of round-off. The iteration error is very similar to the round-off error and it could be cause an energy drift.

6.5.3. Errore azterketa estadistikoa.

Errorearen konparaketa fidagarriagoa egiteko helburuarekin, azterketa estadistika egin dugu. Ausaz perturbatutako $P = 1000$ hasierako balio ezberdinatarako integrazioak exekutatu ditugu eta emaitza hauen guztiak batezbestekoan oinarritu gara, biribiltze errorearen azterketa egokia egiteko.

Ausazko perturbazioak. Perturbaziorik gabeko hasierako balioa ($u0, e0$) eta $\mathcal{O}(10^{-6})$ ordeneko k perturbazio tamaina finkatuta, era honetan kalkulatuko dugu ($u1D, e1$) perturbatutako hasierako balioa,

```

k = 2^(-20);
u1 = u0 + u0*(k*RandomReal[{-1,1}]);
u1D = N[u1];
u1DD = SetPrecision[u1D, prec];
e1 = N[u1 - u1DD];

```

Puntu-finko interazio bidezko 6-ataletako Gauss kolokazio metodoaren hiru implementazio konparatu ditugu:

6.1. Taula: Percentage of steps that reach a computational fixed-point and the number of fixed-point iterations per step for the computations of non-chaotic double pendulum (NCDP), chaotic double pendulum (CDP), and the outer solar system (OSS) problems. In columns, we compare three different implementations: FPIEA (ideal), DP (double precision) and Hairer's Fortran code.

	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

1. Implementazio ideala. Ekuazio diferenzialaren eskuin aldeko funtzioren ebaluazioa izan ezik, beste eragiketa guztiak doitasun laukoitzean egiten dituen implementazioa da.
2. Doitasun bikoitzeko gure implementazioa berria (DP).
3. Hairer-en implementazioa [30], jatorrizko bere IRK metodoaren Fortran koda exekutatu dugu ([Hairer's preprints page](#)).

Distribution of energy jumps.

Integratzailearen implementazio egoki batean, biribiltze erroreak eragindako energiaren errore lokala $H(y_n) - H(y_{n-1})$, ausazkoa izatea espero da. Zenbakizko soluzioa m urratsero jasotzen dugula jakinik, energi diferentzia $H(y_{km}) - H(y_{km-m})$ ausazkoa izango da, μ batazbestekoa ($\mu = 0$ ideal) eta σ duen distribuzio Gaussiararekin. Beraz, metatutako energia diferentzia

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

$t_{mk} = t_0 + kmh$ uneetarako, $k^{1/2}\sigma = (t_{mk}/(mh))^{1/2}\sigma$ desbiazo estandarra duen ausazko ibilbide Gausiar bat (*random walk*) jarraituko du. Brouwer-ek Kepler problemarentzat zenbakizko integracioaren birbiltze errorearen azterketa egin zuen [10] eta horregatik, aurretik aipatutakoa Brouwer legea bezala ezaguna da konputazio zientzian [27].

Integracio tartea $[t_0, t_{end}]$ bada eta P perturbatutako hasierako balio kopurua bada, $L P$ energia diferentzia balioak ditugu,

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

non $L = (t_{end} - t_0)/(mh)$ den. (6.3.) irudian, L P balioen histogramak eta $N(\mu, \sigma)$ distribuzio normalak irudikatu ditugu (non μ eta σ , balioei dagokien batazbestekoa eta desbiazio tipikoak diren). Gure helburua, gure DP implementazioari dagozkion histogramak distribuzio Gausiarra ondo egokitzen zaion ikustea da.

Evolution of mean and standard deviation of errors

(6.4.) irudian, energiaren errorearen batazbestekoa eta desbiazio tipikoa irudikatu ditugu.

$$\Delta E_i^{[k]} = \frac{(E_i^{[k]} - E_0^{[k]})}{E_0^{[k]}}, \quad i = 1, \dots, N/m \text{ eta } k = 1, \dots, P.$$

$$\bar{\Delta E}_i = \frac{1}{P} \sum_{k=1}^P \Delta E_i^{[k]}, \quad \sigma_i = \sqrt{\frac{1}{P} \sum_{k=1}^P (\Delta E_i^{[k]} - \bar{\Delta E}_i)^2}.$$

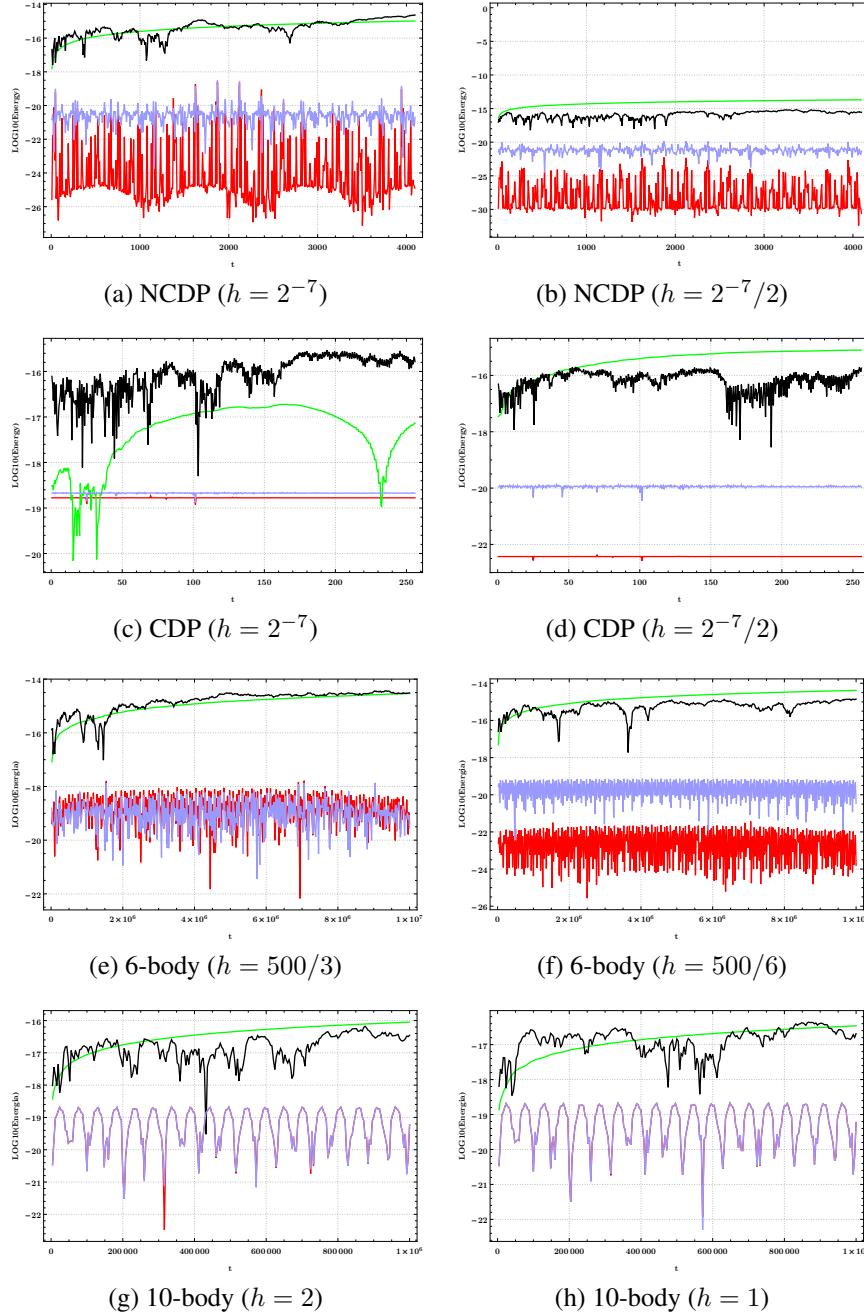
eta kokapen erroreak: doitasun laukoitzean lortutako soluzioa, soluzio zehatzak kontsideratuko dugu,

$$y_{exact}^{[k]} = \hat{y}_i^{[k]} = (\hat{q}_i^{[k]} + \hat{eq}_i^{[k]}, \hat{p}_i^{[k]} + \hat{ep}_i^{[k]}), \quad i = 1, \dots, N/m.$$

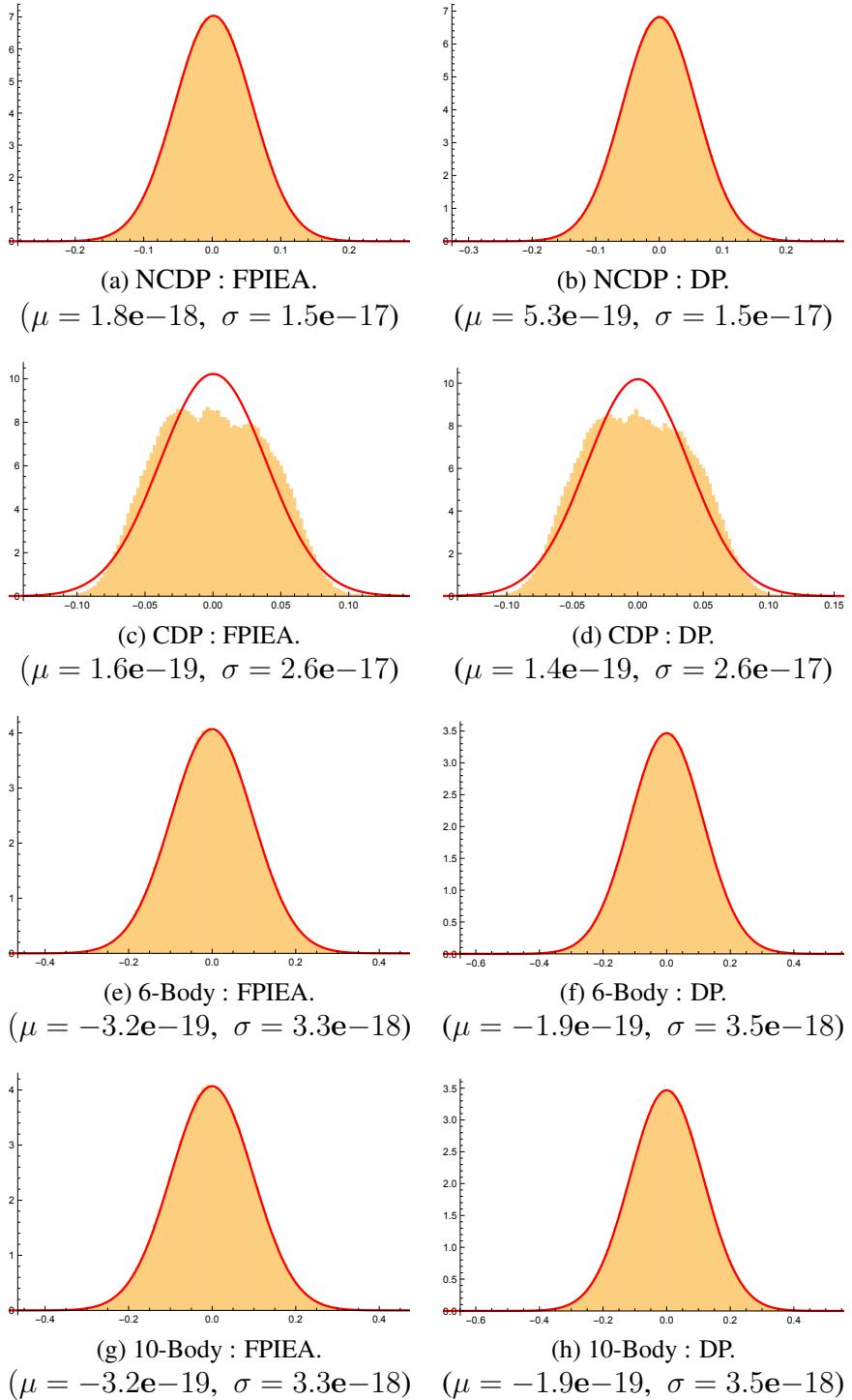
eta $k.$ soluzioari $y_i^{[k]} = (q_i^{[k]} + eq_i^{[k]}, p_i^{[k]} + ep_i^{[k]})$ dagokion kokapen errorea,

$$Ge_i^{[k]} = \|(\hat{q}_i^{[k]} + \hat{eq}_i^{[k]}) - (q_i^{[k]} + eq_i^{[k]})\|_2.$$

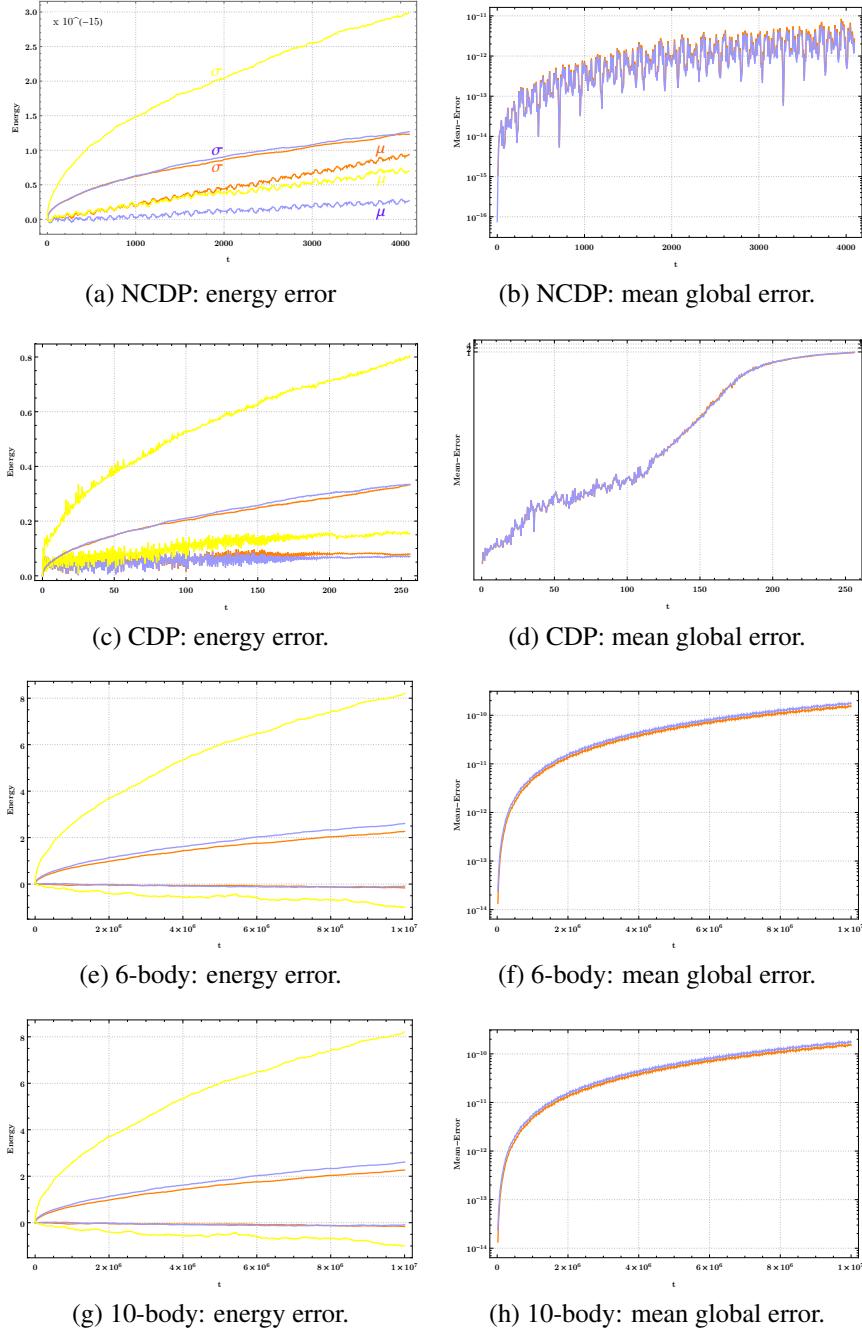
6.5.4. Round-off estimation



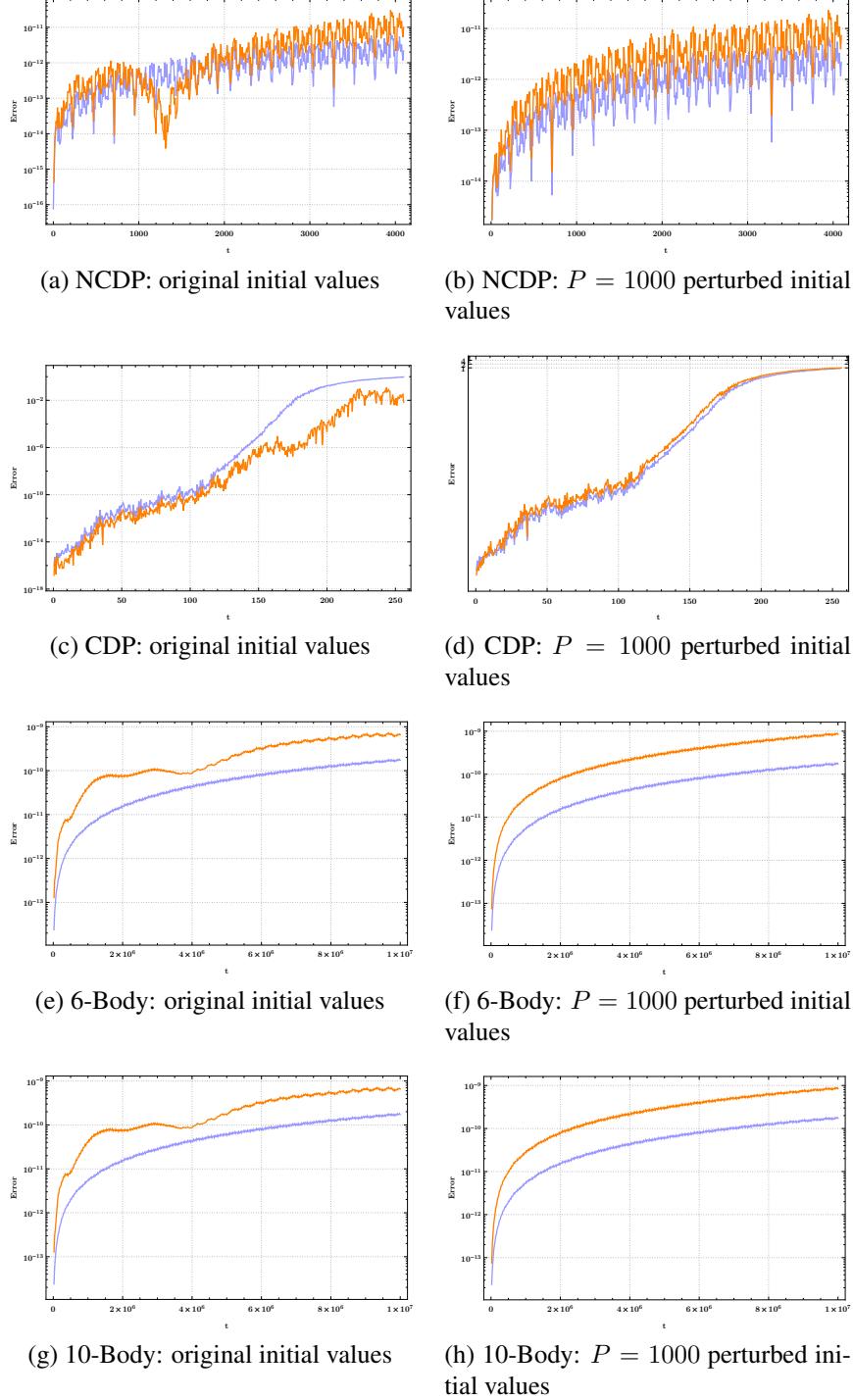
6.2. Irudia: We plot the evolution of energy relative error in logarithmic scale of the next algorithms implementations: A-algorithm as estimation of truncation error (red), B-algorithm as estimation of iteration error (green), C-algorithm as estimation of the effect of the computation of double precision \tilde{f} function (black) and D-algorithm as estimation of the effect of using double precision $\tilde{b}_i, \tilde{\mu}_{ij} \in \mathbb{F}$ coefficients (blue). We show an figure for each initial value problem: Double Pendulum Non-Chaotic case (a), Double Pendulum Chaotic case (b) and outer solar system case (e).



6.3. Irudia: Histograms of KP samples of energy jumps against the normal distribution $N(\mu, \delta)$. The horizontal axis is multiplied by 10^{15} and vertical axis ??? for the FPIEA and DP implementations for the three considered problems (NCDP, CDP, and OSS).



6.4. Irudia: Evolution of mean (μ) and standard deviation (σ) of errors in energy (left) and evolution of Mean of global errors in positions (right), for DP implementation (blue), FPIEA implementation (orange), and Hairer's implementation (yellow). Non-Chaotic case (a,b), Chaotic case (c,d) and outer solar system case (e, f).



6.5. Irudia: Left plots: estimation of the round-off error with the original unperturbed initial values. We compare the evolution of our error estimation (orange) with the evolution of the global error (blue). Right plots: evolution of the mean error in positions (blue) of the application of our DP algorithm to $P = 1000$ perturbed initial value problems, together with the evolution of the mean of the estimated errors in positions (orange).

6.6. zaharra

6.6.1. Integrazio motak.

Erabiltzaileak ekuazio diferentzialaren eskuin aldeko funtzioa, doitasun bikoitzean definituko duela suposatu dugu eta gure implementazioaren biribiltze errorearen propagazioa idealetik gertu dagoela erakutsi nahi dugu. Gure implementazioaren zenbakizko soluzioaren $\tilde{y}_n + e_n \approx y(t_n)$ ($n = 1, 2, \dots$) errorea, jatorri ezberdineko erroreen konbinazioa da.

1. Integrazio zehatza.

Konputazio guztia (ekuazio diferentzialaren funtzioaren ebaluazioa barne) doitasun laukoitzeko (128-bit) koma higikorreko aritmetikan egindako integrazioa. Zenbakizko integrazio hau, trunkatze errorea estimatzeko eta errore globala kalkulatzeko erreferentzi gisa (soluzio zehatza) konsideratuko dugu.

2. Integrazio superidealra.

Zenbakizko integrazio hau iterazio errorea estimatzeko erabiliko dugu. Konputazio guztia doitasun laukoitzean egindako integrazioa baina geratze erizpidea, doitasun bikoitzean neurrtuko dugu,

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

3. Integrazio idealra.

Ekuazio diferentzialaren eskuin aldeko funtzioaren ebaluazioa izan ezik, beste eragiketa guztiak doitasun laukoitzean egiten dituen implementazioa da. Ekuazio diferentziala doitasun bikoitzean kalkulatzeak, eragiten duen errorea neurrtzeko erabiliko dugu eta integrazio hau hobetu ezin daitekeen integrazioa konsideratuko dugu.

4. Integrazio sasi-idealra.

Doitasun bikoitzeko koefizienteak ($\tilde{\mu}_{i,j}, \tilde{b}_i$) erabiltzeak eragiten duen errorea neurrtzeko integrazioa da. Konputazio guztia doitasun laukoitzean kalkulatzen da baina doitasun bikoitzeko koefizienteen balioak erabiliz (doitasun bikoitzeko koefiziente koadrifikatuak esaten diogu).

5. Double-berria (batura konpensatu hobetua).

Guk proposatutako implementazioa da. Doitasun bikoitzeko koma higikorreko aritmetika (64-bit) eta gure proposamenak jarraituz implementatutako *batura konpensatua hobetua* erabiltzen duen implementazioa, hau da, atalen

eguneraketan, $(\tilde{y}_n \oplus \tilde{e}_n)$ (ekuazioa 6.13) espresioa erabiltzen duen integrazioa.

6. Double-estandarra (batura konpensatu estandarra).

Doitasun bikoitzeko koma higikorreko aritmetika (64-bit) eta *batura konpensatu estandarra* erabiltzen duen implementazioa (atalen eguneraketan \tilde{e}_n gaia kontsideratzen ez duen integrazioa).

The implementation given in [30], we will refer to it as Hairer's implementation. We have run their original implicit Runge-Kutta Fortran code ([Hairer's preprints page](#)).

Integrazio bakarra egin ordez, ausaz perturbatutako $P = 1000$ hasierako balio ezberdinatarako integrazioak exekutatu ditugu eta emaitza hauen guztien batez-bestekoan oinarritu gara, biribiltze errorearen azterketa egokia egiteko.

Ausazko perturbazioak.

Perturbaziorik gabeko hasierako balioa (u_0, e_0) eta $\mathcal{O}(10^{-6})$ ordeneko k perturbazio tamaina finkatuta, era honetan kalkulatuko dugu (u_{1D}, e_1) perturbatutako hasierako balioa,

```
k = 2^(-20);
u1 = u0 + u0*(k*RandomReal[{-1,1}]);
u1D = N[u1];
u1DD = SetPrecision[u1D, prec];
e1 = u1 - u1DD;
```

Neurtzeko faktoreak.

Sistema Hamiltoniarretan energia kontserbatzen da,

$$E_i^{[k]} = H(q_i^{[k]} + eq_i^{[k]}, p_i^{[k]} + ep_i^{[k]}) = \text{konst}, \quad i = 1, \dots, N/m.$$

Energia eta kokapen erroreak honako faktoreen bidez neurtuko ditugu.

1. Energia errorea.

$$\begin{aligned}\Delta E_i^{[k]} &= \frac{(E_i^{[k]} - E_0^{[k]})}{E_0^{[k]}}, \quad i = 1, \dots, N/m \text{ eta } k = 1, \dots, P. \\ \bar{\Delta E}_i &= \frac{1}{P} \sum_{k=1}^P \Delta E_i^{[k]}, \quad \sigma_i = \sqrt{\frac{1}{P} \sum_{k=1}^P (\Delta E_i^{[k]} - \bar{\Delta E}_i)^2}, \\ \bar{MaxE} &= \max_{i=1, \dots, N} |\bar{\Delta E}_i|.\end{aligned}$$

2. Energia differentzia.

P integrazio baditugu eta $L = N/m$, integrazio bakoitzean lortutako emaitza kopurua bada, $L \cdot P$ balio guztiak energiaren differentzien batezbesteko ($\bar{\mu}$) eta desbideratze estandarra ($\bar{\sigma}$) kalkulatuko ditugu.

$$\begin{aligned}Dif_i^{[k]} &= \frac{(E_i^{[k]} - E_{i-1}^{[k]})}{E_0^{[k]}}, \quad i = 1, \dots, N/m \text{ eta } k = 1, \dots, P, \\ \bar{\mu} &= \frac{1}{L \cdot P} \left(\sum_{k=1}^P \sum_{i=1}^L Dif_i^{[k]} \right), \quad \bar{\sigma} = \sqrt{\frac{1}{L \cdot P} \left(\sum_{k=1}^P \sum_{i=1}^L (Dif_i^{[k]} - \bar{\mu})^2 \right)}\end{aligned}$$

3. Kokapen errore globala.

Doitasun laukoitzean lortutako soluzioa, soluzio zehatza konsideratuko dugu,

$$yexact_i^{[k]} = \hat{y}_i^{[k]} = (\hat{q}_i^{[k]} + \hat{e}q_i^{[k]}, \hat{p}_i^{[k]} + \hat{e}p_i^{[k]}), \quad i = 1, \dots, N/m.$$

eta k . soluzioari $y_i^{[k]} = (q_i^{[k]} + eq_i^{[k]}, p_i^{[k]} + ep_i^{[k]})$ dagokion kokapen errorea,

$$\begin{aligned}Ge_i^{[k]} &= \|(\hat{q}_i^{[k]} + \hat{e}q_i^{[k]}) - (q_i^{[k]} + eq_i^{[k]})\|_2 \\ \bar{Ge}_i &= \left(\frac{1}{P} \sum_{k=1}^P Ge_i^{[k]} \right), \quad \bar{MaxGe} = \max_{i=1, \dots, N/m} (Ge_i)\end{aligned}$$

4. Puntu-finkorainoko konbergentzia portzentaien batezbesteko ($\bar{\Lambda}$).

$\Lambda^{[k]}$ k . integrazioan, puntu-finkorainoko konbergentzia lortutako urratsen ehuneko portzentaia izanik,

$$\bar{\Lambda} = \frac{1}{p} \sum_{k=1}^P \Lambda^{[k]}.$$

5. Kokapen errore estimazioa ($\mu\bar{Q}_i$, $\sigma\bar{Q}_i$).

Biribiltze errorearen estimazioaren gogoratuz,

$$Est_i^{[k]} = \|(q_i^{[k]} + eq_i^{[k]}) - (\tilde{q}_i^{[k]} + \tilde{eq}_i^{[k]})\|_2 \quad (6.15)$$

Errore estimazioaren kalitatea neurtzeko,

$$Q_i^{[k]} = \log_{10} \left(\frac{Est_i^{[k]}}{Ge_i^{[k]}} \right), \quad (6.16)$$

$$\mu\bar{Q}_i = \frac{1}{P} \sum_{k=1}^p Q_i^{[k]}, \quad \sigma Q_i = \sqrt{\frac{1}{P} \sum_{k=1}^P (Q_i^{[k]} - \mu\bar{Q}_i)^2} \quad (6.17)$$

Emaitza nagusiak.

Esperimentuen emaitzak (??) taula orokorrean eta (??) irudian adierazi ditugu.

1. (??) taula orokorra.

Taula honetan esperimentuen ikuspegi orokorrena laburtu dugu. Emaitza hauetan, gure implementazioaren (*Double-berria*) eta *integrazio idealaren* artean diferentzia oso txikia dela baiezttatu daiteke. Batetik, bi integracioetan energia errorearen eta kokapen errorearen maximoek ia balio berdina hartzen dute. Bestalde, puntu-finkorainoko konbergentzia batezbesteko porzentaia altuak, urratsa gehienetan puntu-finkoaren iterazioaren bidez lortu daitekeen emaitza hoberena lortzen ari garela adierazten digu.

2. (??) irudia.

(a) **(a) irudia: energiareneko eboluzioa.** Irudi honetan *integrazio idealaren*, *Double-berria* eta *Double-estandarra* integrazioak konparatu ditugu. Batetik, espero genuen moduan *Double-berria* integrazioak, *Double-estandarra* integrazioak baino emaitza hobea du. Bigarrenik, *Double-berria* integracioren eta *integrazio idealaren* energiareneko eboluzioak antzekoak dira.

Integrazio mota guztien energiareneko eboluzioan, drift-a nabarmenzen da. Hurrengo atalean (6.6.1.) drift honen jatorria aztertuko dugu.

(b) **(b) irudia: kokapen errorearen eboluzioa.** Integrazio tartean zehar, *Double-berri* integrazioaren eta *integrazio idealaren* kokapen errorea parekoak dira (irudian ia ez dira bereizten). Energia errorearekin geratzen zaigun moduan, *Double-berri* integrazioan kokapen errorea pixka bat handiagoa da.

- (c) **(c) eta (d) irudiak: energia histogramak.** Energia errorearen ausazkotasuna aztertu dugu. Esperimentu hauetan, $P = 1.000$ perturbatutako hasierako balio ezberdinen kopurua bada eta $N_{out} = 512$ integrazio bakoitzean itzulitako diskretizazio kopurua bada, integrazio hauen guztien 521.000 energia differentziuen laginen histogramak irudikatu ditugu, lagin hauen batezbesteko eta desbideratze estandarreko banaketa normalarekin konparatz. Bai ((c) irudia) *integrazio idealaren*, bai ((d) irudia) *Double-berri* integrazioaren energia differentziuen laginak bat datozen banaketa normalarekin.
- (d) **(e) eta (d) irudiak: errorearen estimazioa.** (e) irudian errorearen estimazioa kokapen errorearekin konparatu dugu: gure estimazioa kokapen errorearen gainetik mantentzen da integrazioan zehar. Estimazioaren kalitatea neurtu dugu (d) irudian: 0 eta 2.1 tartean mantentzen da, eta beraz gehienez estimazioa, kokapen errorea baino bi aldiz handiago da.

Energia Drift-a.

(?? a) irudian energia batezbestekoak zero izan behar luke integrazio osoan baina drift-a azaldu zaigu. Lehenik, gure implementazioaren arazoren batek drift hori ez duela sortzen baiezttatu behar dugu. Horretarako bi esperimentu berri egin ditugu:

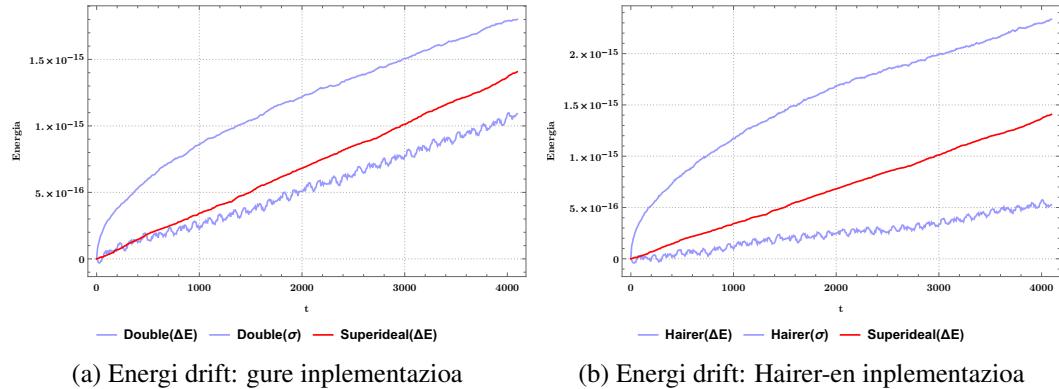
1. *Integratio superidealaren*. Horretarako, integratio mota berri bat exekutatu dugu *superidealaren* deitu duguna: exekuzio osoa doitasun laukoitzean baina geratze irizpidea *double moduan* aplikatua,

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

Integratio superidealaren emaitzak (6.6.(a) irudia) ikus daitezke eta bertan, *superidealaren* integrazioak ere energia drift-a azaltzen duela ikusi daiteke.

2. Hairer-en implementazioaren azterketa. Hairer-ek bere lanean [30] bi problema (Hénon-Helies eta kanpo eguzki-sistema) aztertu zituen eta energia drift-aren arazoa konpondutzat eman zuen. Guk Hairer-en kodearekin Pendulu bikoitzaren problema integratu dugu eta bere implementazioan ere, energiaren drift-a badagoela baiezttatu dugu. Hala ere, 6.6.(b) irudian energia drift-a gure implementazioa baino txikiago duela adierazi behar dugu.

Bigarrenik, energiaren drift-aren arrazoia bilatu nahi dugu. Gure ustez, energiaren drift-aren jatorria, puntu-finkoaren iterazioen erroreak eragindakoa da: puntu-finkoaren iterazioen konbergentzia $O(h)$ da. Zentzu honetan bi esperimentu berri egin ditugu:



6.6. Irudia: Energia Drift-a (I) (Pendulu-bikoitza ez-kaotikoa).

- (a) irudian**, gure implementazioaren emaitzak ikusi daitezke: *superidealaren* integrazio bakarraren energia errore erlatiboa eta *Double-Berri* bertsioaren $P = 1.000$ integrazioen energia batezbestekoa eta desbideratze estandarra.
- (b) irudian**, Haire-en implementazioaren $p = 1000$ integrazioen energia batezbestekoa eta desbideratze estandarra, gure implementazioaren *superidealaren* integrazio bakarraren energia errore erlatiboarekin konparatu ditugu.

1. Newton simplifikatua. Newton simplifikatuaren iterazioen konbergentzia $O(h^2)$ da. Newton simplifikatuan oinarritutako implementazio berri batekin Pendulu bikoitzaren integrazioa errepikatu dugu eta energiaren drift-a ez dagoela baiezta dugu (ikus 6.7. (a) irudia). Beraz, puntu-finkoaren metodoak berezkoa duen errorea konsidera daiteke.
2. Sasi-Newton. Hala ere, azken urratsa bezala, LU-deskonposaketa eragiketa eta jakobiarrak behar ez duen implementazio eskaini nahi diogu erabiltzai-leari. Esperimentalki frogatu dugu modu honetan ere, energiaren drift-a desagertu dela (ikus 6.7. (b) irudia).

Jarraian, IRK puntu-finkoaren eta Newton Simplifikatuaren arteko algoritmoa azalduko dugu. Newton metodo simplifikatua, $k = 1, 2, \dots, L_i^{[k]}$ hurbilpenak kalkulatzeko algoritmoa hau bada,

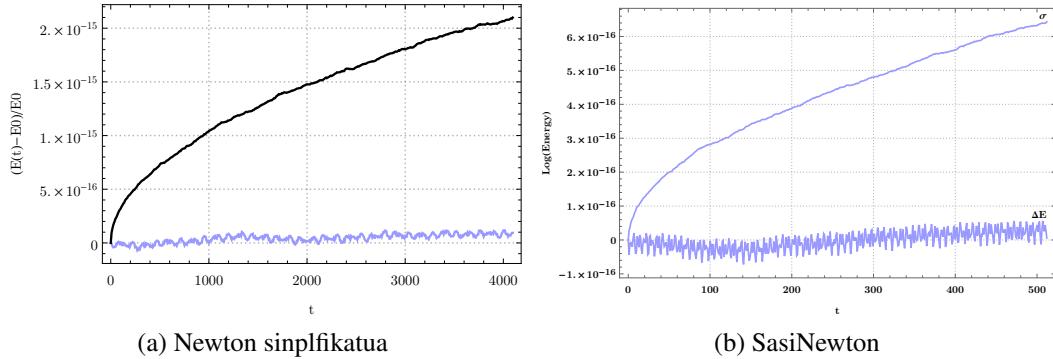
- (a) $g_i^{[k]} = -L_i^{[k-1]} + hb_i f(t + c_i h, y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}), \quad i = 1, \dots, s.$
- (b) Solve $\Delta L_i^{[k]}$,

$$\Delta L_i^{[k]} - hb_i \sum_{j=1}^s \mu_{ij} J_j \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s.$$
- (c) $L_i^{[k]} = L_i^{[k-1]} + \Delta L_i^{[k]}, \quad i = 1, \dots, s.$

Modu merkean, $\Delta L_i^{[k]}$ era honetan hurbildu daiteke,

$$\Delta L_i^{[k]} = \frac{hb_i}{\lambda} \left(f(Y_i^{[k]} + \sum_{j=1}^s \lambda \mu_{ij} g_j^{[k]}) - f(Y_i^{[k]}) \right)$$

non $\lambda = 2^{10}$.



6.7. Irudia: Energia Drift-a (II) (Pendulu-bikoitza ez-kaotikoa).

(a) irudian puntu-finkoaren iterazioaren ordez Newton sinplifikatua aplikatuz integrazioen energia batezbestekoa eta desbideratze estandarra.

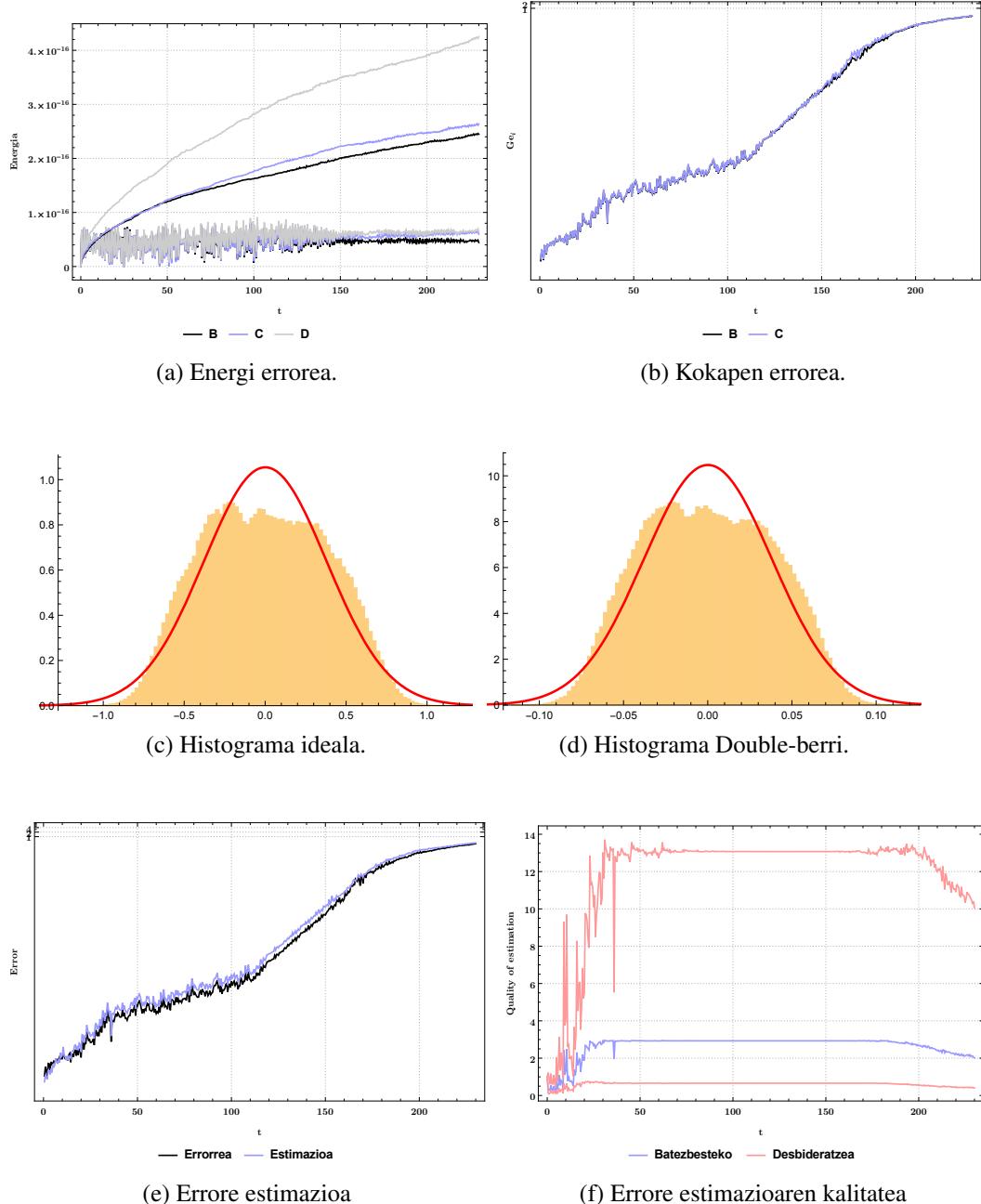
(b) irudian, puntu-finkoaren iterazioaren ordez SasiNewton aplikatuz integrazioen energia energiaren errorearen batezbestekoa eta desbideratze estandarra. Esperimentu honetan integrazio tarte txikiagoa exekutatu dugu ($t = 2^9$).

Emaitza nagusiak.

Esperimentuen emaitzak (6.2.) taulan eta (6.8.) irudian adierazi ditugu. Problema kaotikoa denez, kokapen errorea esponentzialki handitzen da eta horregatik integracio tarte txikiagoa aukeratu da. Orokorean, Pendulu bikoitza ez-kaotikoaren esperimentuen ezaugarriak mantentzen dira. Esan, esperimentu hauetan ez dela energia drift-arik azaltzen (integracio tarte txikia delako) eta energia differentzien histogramak ez dato zela erabat dagokion banaketa normalarekin.

6.2. Taula: Pendulu Bikoitza kaotikoaren laburpena.

Arithmetic	$\bar{\Lambda}$	$MaxE$	$\bar{\mu}$	$\bar{\sigma}$	$MaxGe$
	%				
Zehatza	94	$2e10^{-19}$	$4e10^{-22}$	$9e10^{-21}$	
Ideala	98	$8e10^{-17}$	$1e10^{-19}$	$3e10^{-17}$	0.33
Double-berria	95	$8e10^{-17}$	$1e10^{-19}$	$3e10^{-17}$	0.34
Double-estandarra	95	$9e10^{-17}$	$1e10^{-19}$	$3e10^{-17}$	0.35



6.8. Irudia: Pendulu-bikoitza kaotikoa.

(a) irudian, energi errorearen batezbestekoaren ($\Delta\bar{E}_i$) eta desbideratze estandarraren (σ_i) eboluzioa. **(b) irudian**, kokapen errorearen eboluzioa $\bar{G}e_i$. Honako laburdurak erabili ditugu: B=Integrazio ideal, C=Double-Berria, D=Double-estandar.

(c) eta (d) iruditan $P = 1.000$ integrazio ezberdinaren 521.000 energi diferentziaren lagineen histogramak irudikatu ditugu, batezbesteko eta desbideratze estandar bereko banaketa normalarekin alderatuz. Ardatza horizontalak 10^{-16} unitatea adierazten du.

(e) irudian errorearen estimazio kokapen errorearekin konparatu dugu. **(f) irudian** estimazioaren kalitatea aztertu dugu: (6.16) definizioaren arabera, estimazio-kalitatearen batezbestekoa $\mu\bar{Q}_i$ eta desbideratze estandarra $\sigma\bar{Q}_i$ irudikatu ditugu.

6.6.2. N-Body problema.

Problema honetan erabilitako integrazio tarteak eta urratsa luzera hauek izan dira,

$$\begin{aligned}t_0 &= 0, \quad t_{end} = 10^6, \\h &= 2, \\sampling &= 1000, \quad step = 1. \\P &= 1.000, \quad N = 500.\end{aligned}$$

(integrazioen iraupena= xx egun).

Esperimentuen emaitzak (6.3.) taulan eta (6.9.) irudian adierazi ditugu. Azpimarratu nahi dugu N-Body problemaren ez zaigula energia Drift-arik agertu. N-Body problema bigarren ordeneko ekuazio diferentzial arrunta da eta *Gauss-Seidel* moduko iterazioa aplikatu dugu (konbergentzia azkarra goa).

6.3. Taula: N-Body problemaren laburpena.

Arithmetic	$\bar{\Lambda}$	\bar{MaxE}	$\bar{\mu}$	$\bar{\sigma}$	\bar{MaxGe}
	%				
Zehatza	98.	$2e10^{-19}$	$5e10^{-24}$	$2e10^{-20}$	
Ideala	99.	$6e10^{-18}$	$6e10^{-21}$	$3e10^{-19}$	$7e10^{-12}$
Double-berria	98.	$5e10^{-18}$	$4e10^{-21}$	$3e10^{-19}$	$1e10^{-11}$
Double-estandarra	98.	$5e10^{-18}$	$4e10^{-21}$	$3e10^{-19}$	$1e10^{-11}$

Momentu angeluarra.

N-Body problemaren momentu angeluarra,

$$O = \sum_{i=0}^N \mathbf{p}_i \times \mathbf{q}_i.$$

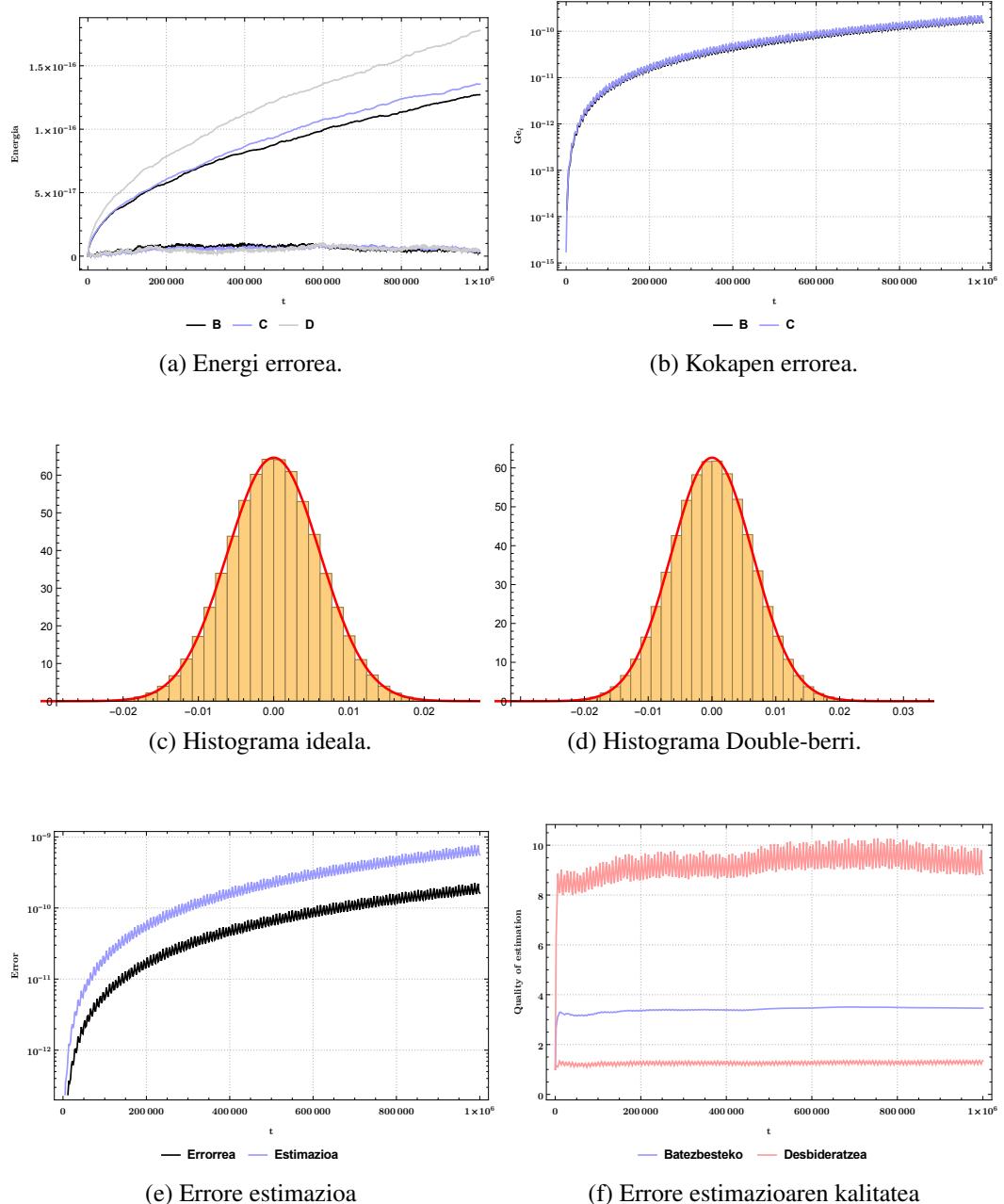
metodo simplektikoek mantentzen duten inbariantea da. Hurrengo grafikoetan (irudia 6.10.) horrela baieztau daiteke.

6.6.3. Ondorioak.

Lehen atal honetan, IRK metodoaren puntu-finkoko iterazioan oinarritutako implementazioa hiru problema ez-stiff-etalako aztertu dugu. Gogoan eduki behar dugu, erabiltzaileak ekuazio diferentzialaren eskuin aldeko funtzioa doitasun bikoitzean definituko duela suposatu dugula. Ondorio hauek azpimarratuko ditugu:

1. Hairer-en implementazioan antzeman ditugun hainbat arazo konpondu ditugu, implementazioaren hobekuntzak proposatuz eta implementazio sendoagoa lortu dugula uste dugu.
2. Doitasun laukoitzeko aritmetika ($128 - bit$) erabiltzen duen implementazioa (*ideal*) ez duela merezi ikusi dugu. Doitasun bikoitzeko aritmetika (64 bit) eta biribiltze errorea gutxitzeko teknika zehatzak (batura konpensatua, ...) erabiltzen dituen implementazioarekin, *integrazio idealaren* errore mailaren oso antzekoa lortu dugu. Ondorioz, doitasun laukoitzeko aritmetikarekin (*ideal*) lortzen diren emaitzak, eskatzen duen CPU kostuak ez duela merezi esan dezakegu.
3. Problema ez-stiff-tarako, puntu-finkoko iterazioa Newton simplifikatua baino merkeagoa da eta beraz, iterazio metodo hau aplikatzea gomendatzen da [29]. Pendulu bikoitzaren (hasierako balio ez-kaotikoak) epe luzeko integrazioan, energia drift-aren arazoaz ohartu gara. Puntu-finkoko iterazioaren errorea energi drift honen jatorria da. Puntu-finkoko iterazioaren konbergentzia $O(h)$ izanik, Newton simplifikatua iterazioak $O(h^2)$ konbergentziarekin ez du energia drift-aren arazorik.
4. Puntu-finkoak doitasun laukoitzean lan egiterakoan muga batzuk ditu. Esperimentua definitu ?? Atal askotako metodoak??
5. Aurreko bi arrazoi hauen ondorioz, problema ez-stiff-tarako ere, Newton simplifikatua aplikatzea egokia izan daitekeela pentsa liteke. Newton simplifikatua garestia da, iterazio bakoitzean *LU-deskonposaketa* eta jakobiarra kalkulatu behar direlako. Guzti hau kontutan hartuta, Newton iterazio metodoen implementazio eraginkorrik, abantaila asko izan ditzakeela iruditzen zaigu.

6.7. Laburpena.

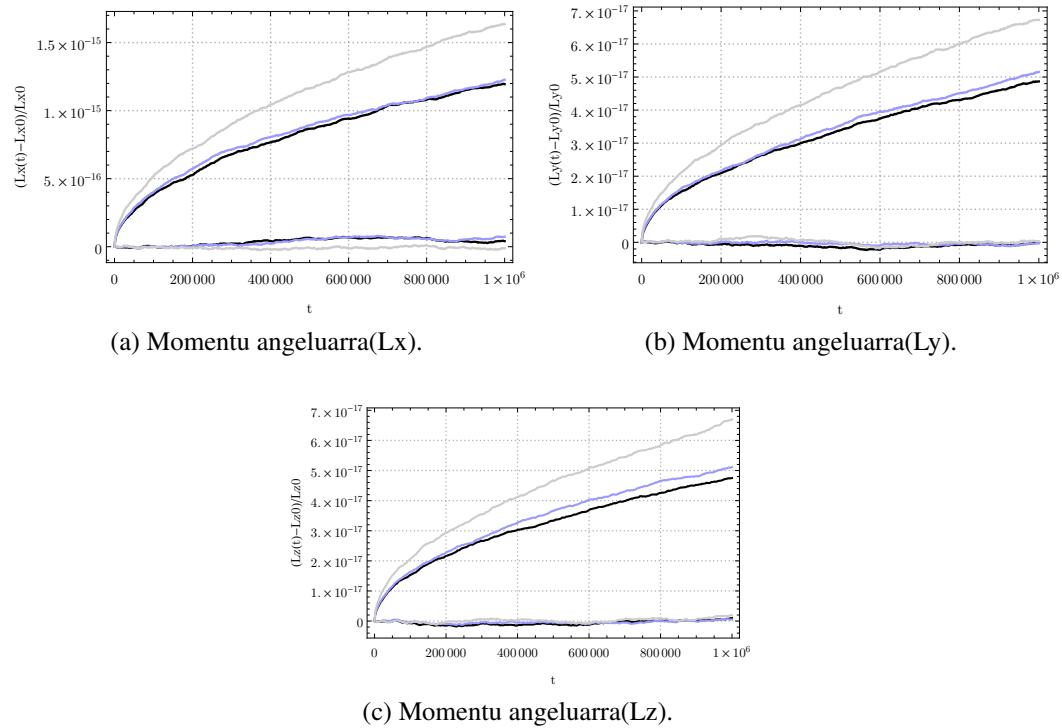


6.9. Irudia: N-Body.

(a) irudian, energi errorearen batezbestekoaren ($\bar{\Delta E}_i$) eta desbideratze estandarraren (σ_i) eboluzioa. **(b) irudian**, kokapen errorearen eboluzioa \bar{G}_e . Honako laburdurak erabili ditugu: B=Integrazio ideal, C=Double-Berria, D=Double-estandarra.

(c) eta (d) iruditan $P = 1.000$ integrazio ezberdinaren 521.000 energi differentziengaten lagineen histogramak irudikatu ditugu, batezbesteko eta desbideratze estandar bereko banaketa normalarekin alderatuz. Ardatza horizontalak 10^{-16} unitatea adierazten du.

(e) irudian errorearen estimazio kokapen errorearekin konparatu dugu. **(f) irudian** estimazioaren kalitatea aztertu dugu: (6.16) definizioaren arabera, estimazio-kalitatearen batezbestekoa $\mu \bar{Q}_i$ eta desbideratze estandarra $\sigma \bar{Q}_i$ irudikatu ditugu.



6.10. Irudia: N-Body (Momentu angeluarra).

(a) irudian,(b) irudian eta (c) irudian momentu angeluarraren errorearen batezbestekoaren ($\bar{\Delta L}_i$) eta desbideratze estandarraren (σ_i) eboluzioa.

7. Kapitulua

IRK: Newton.

The top 10 algorithms in Applied Mathematics: 1.Newton and quasi-Newton methods.

Nick Higham (2016)

7.1. Sarrera.

Runge-Kutta metodo implizituen implementazio eraginkorren arazo handiena, ekua-zio sistema ez-lineala metodo iteratibo baten bidez askatzea da. Problema zurruna denean, puntu finkoaren iterazio ez da eraginkorra eta Newton interazioa aplikatu behar da. Gainera problema ez-zurruna izanik ere, Newton iterazioak interesgarriak izan daitezke; bereziki doitasun altuko (doitasun laukoitza) konputazioetan iterazio metodoaren konbergentzi ezaugarri onak direla eta.

Newton metodoa iteratiboa, ekua-zio sistema ez-linealen zenbakizko soluzioa aurkitzeko metodoa da. $u \in \mathbb{R}^n$ 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 25).

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

Algoritmoa 25: Newton metodoa.

non $J(u^{[k-1]})$, $n \times n$ tamainako matrize jacobiarra den ($J_{ij}(u) = \partial f_i / \partial u_j(u)$, $1 \leq i, j \leq n$). Newton metodoak, $u^{[*]}$ soluzioarekiko konbergentzia koadratikoa du,

$$\|u^{[k+1]} - u^{[*]}\| \leq C \|u^{[k]} - u^{[*]}\|^2.$$

Newton metodo interesgarria da, baina konputazionalki garestia; iterazio bakoitzean $J(u^{[k-1]})$ jacobiarren ebaluazioa eta $n \times n$ tamainako matrizearen LU deskonposaketa kalkulatu behar da. Horregatik, konputazionalki merkeagoak diren Newton metodoaren aldaerak erabili ohi dira. Newton metodoaren aldaera horien artean, Newton simplifikatua (Algoritmoa 26) dugu aukera nagusienetako bat.

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

Algoritmoa 26: Newton simplifikatua.

Argia da Newton simplifikatua konputazionalki merkeagoa dela: urrats bakoitzean, jacobiarren ebaluazio $J^{[0]} = \frac{\partial F}{\partial u}(u^{[0]})$ eta dagokion LU deskonposaketa behin bakarrik kalkulatu behar dira.

Azpimarratzeko da ere, Newton metodoaren eragiketa konplexuenak doitasun txikiagoan kalkula daitezkeela [2] eta honek, konputazionalki abantaila interesgarria suposatzen duela. Algoritmoaren (Algoritmoa 26) eskubi aldean, doitasun bikoitzeko (64-bit) implementazioa balitz, eragiketa bakoitzaren doitasuna zehaztu dugu: jacobiarren balioztapena eta algebra linealeko eragiketak, doitasun arruntean (32-bit) kalkulatu daitezke.

S -ataletako IRK metodoa, Newton iterazioaren bidez d -dimentsioko ODE sistemari aplikatzeko, urrats bakoitzean $sd \times sd$ tamaineko hainbat ekuazio sistema (iterazio bakoitzeko bat) askatu behar dira. Atal honetan, jatorrizko sd -dimentsioko ekuazio sistema, $(s+1)d$ sistema baliokide moduan berridatziko dugu eta sistema baliokidea, $d \times d$ tamaineko $[s/2] + 1$ matrize errealen LU -deskonposaketa bidez askatuko dugu. Tamaina txikiko matrizeen LU-deskonposaketa azkarra denez, konputazionalki eraginkorragoa izatea espero dugu.

7.2. IRK-Newton estandarra.

Demagun honako hasierako baliodun problema,

$$\dot{y} = f(t, y), \quad y(t_0) = y_0, \quad (7.1)$$

non $y = [q^1, \dots, q^n, p^1, \dots, p^n]^T \in \mathbb{R}^{d=2n}$ eta $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$ diren.

S-ataletako IRK metodoa gogoratz, hasierako baliodun problemaren $y(t)$ soluzioaren $y_n \approx y(t_n)$ hurbilpena, era horretan kalkulatzen da,

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i f(t_n + c_i h, Y_{n,i}), \quad (7.2)$$

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

$$Y_{n,i} = y_n + h \sum_{j=1}^s a_{ij} f(t_n + c_j h, Y_{n,j}) \quad i = 1, \dots, s. \quad (7.3)$$

$Y_{n,i} \in \mathbb{R}^d$, $i = 1, \dots, s$ ezezagunen eta $s d$ tamainako ekuazio sistema ez-lineala askatzeko (7.2), iterazio metodo bat aplikatu behar dugu. Aurreko atalean puntu-finkoaren iterazioa aztertu genuen eta atal horretan, Newton metodoa modu eraginkorrean aplikatzeko bidea ikertuko dugu.

Newton iterazioa.

Newton iterazioa, $k = 1, 2, \dots$ $Y_i^{[k]}$ atalen hurbilpenak kalkulatzeko algoritmoa, modu horretan definituko dugu,

$$1) \quad r_i^{[k]} := -Y_i^{[k-1]} + y + h \sum_{j=1}^s a_{ij} f(t + c_j h, Y_j^{[k-1]}), \quad i = 1, \dots, s, \quad (7.4)$$

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

$$\Delta Y_i^{[k]} - h \sum_{j=1}^s a_{ij} J_j^{[k]} \Delta Y_j^{[k]} = r_i^{[k]} \quad i = 1, \dots, s, \quad (7.5)$$

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

$$3) \quad Y_i^{[k]} := Y_i^{[k-1]} + \Delta Y_i^{[k]}, \quad i = 1, \dots, s, \quad (7.6)$$

Azpimarratu behar da iterazio bakoitzeko, $J_i^{[k]}$ jakobiarraren s -ebaluazio eta $sd \times sd$ tamainako matrizearen LU-deskonposaketa kalkulatu behar ditugula. Era-giketa hauek konplexuak dira eta beraz, Newton osoaren implementazioa konputazionalki garestia da.

Newton sinplifikatuaren iterazioa.

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

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

non,

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

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

Newton super-sinplifikatuaren iterazioa.

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

$$(I_s \otimes I_d - h A \otimes J) \Delta Y^{[k]} = r^{[k]}. \quad (7.7)$$

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

Ohikoa da $J = \partial f / \partial y(t + h/2, y_n)$ hurbilpena erabiltzea. Problema zurruna denean, atalen hasieraketa $Y_i^{[0]} = y_n$, ($i = 1, \dots, s$) erabili ohi da, eta jakpiarren hurbilpen honekin ekuazio lineala askatzeak zentzua izango du.

Newton iterazio bertsio honi super-simplifikatua deitu diogu. Iterazio bakotzean f funtzioren s ebaluazio eta sd tamainako ekuazio-sistema lineala askatu behar da. $(I_s \otimes I_d - h A \otimes J)$ matriza iterazio guztietarako berdina da, bere LU-deskonposaketa behin bakarrik egin behar da eta konputazionalki garestia da [12] [28],

LU-deskonposaketa, $2s^3d^3/3 + \mathcal{O}(d^3)$,
back substitution, $2s^2d^2 + \mathcal{O}(d)$.

Jarraian, Newton super-simplifikatuaren implementazioaren algoritmo orokorra laburtu dugu (Algoritmoa 27).

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
         $k = k + 1;$ 
         $r_i^{[k]} = -Y_i^{[k-1]} + y + h \sum_{j=1}^s a_{ij}f(t + c_j h, Y_j^{[k-1]});$ 
        Solve ( $M \Delta Y^{[k]} = r^{[k]}$ );
         $Y^{[k]} = Y^{[k-1]} + \Delta Y^{[k]};$ 
        konbergentzia  $\leftarrow$  GeratzeErizpidea( $Y^{[k]}$ ,  $Y^{[k-1]}$ ,  $\Delta_{min}$ );
    end
    if ( $\exists j$  non  $\Delta_j^{[k]} \neq 0$ ) then
        if ( $NormalizeDistance(Y^{[k]}, Y^{[k-1]}) > 1$ ) then
            | fail convergence;
        end
    end
     $(\tilde{y}_{n+1}, e_{n+1}) \leftarrow$  baturakonpensatua( $y_n, e_n, Y_n^{[k]}$ );
end

```

Algoritmoa 27: IRK (Newton super-simplifikatua).

7.3. IRK-Newton eraginkorra.**Sarrera.**

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

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

$J \in \mathbb{R}^{d \times d}$ 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 $sd \times sd$ tamaineko hainbat ekuazio sis-

tema (iterazio bakoitzeko bat) askatu behar dira. Atal honetan, jatorrizko sd -dimentsioko ekuazio sistema, $(s+1)d$ sistema baliokide moduan berridatziko dugu eta sistema baliokidea, $d \times d$ tamaineko $[s/2] + 1$ matrize errealeen LU-deskonposaketa bidez askatuko dugu. Tamaina txikiko matrizeen LU-deskonposaketa azkarra denez, konputazionalki eraginkorragoa izatea espero dugu.

Gauss nodoetan oinarritutako *IRK* metodoak simplektikoak eta simetrikoak dira.

1. Simplektikoa.

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad 1 \leq i, j \leq s. \quad (7.8)$$

2. Simetrikoak.

$$b_{s+1-i} = b_i, \quad c_{s+1-i} = 1 - c_i, \quad 1 \leq i, j \leq s, \quad (7.9)$$

$$b_j = a_{s+1-i, s+1-j} + a_{i,j}, \quad 1 \leq i, j \leq s. \quad (7.10)$$

Proposamen berria, bi propietate hauetan oinarrituz garatuko dugu.

Notazio alternatiboa.

Koefiziente notazioa berri bat finkatuta,

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

simplektikotasun (7.8) eta simetrikotasun (7.9) propietateak modu baliokidean era honetan adieraziko ditugu,

1. Simplektikoa.

$$(B\bar{A}) \text{ antisimetrikoa da,} \quad (7.11)$$

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

2. Simetrikoak.

$$b_{s+1-i} = b_i, \quad \bar{c}_{s+1-i} = -\bar{c}_i, \quad 1 \leq i \leq s, \quad (7.12)$$

$$\bar{a}_{s+1-i, s+1-j} = -\bar{a}_{ij}, \quad 1 \leq i, j \leq s. \quad (7.13)$$

$$(7.14)$$

Implementazio berri honen matrizeen dimentsioak zehazteko, balio berri hauean oinarrituko gara, $m = [(s+1)/2]$, eta $s-m = [s/2]$. Beraz metodoaren s -atal kopurua bikoiti ala bakoiti den arabera,

- s bikoitia.

Adibidea $s = 6 \rightarrow m = 3, s - m = 3$.

- s bakoitia.

Adibidea $s = 7 \rightarrow m = 4, s - m = 3$.

Simplektikoa.

$(B\bar{A})$ antisimetrikoa bada, $B^{1/2}\bar{A}B^{-1/2}$ antisimetrikoa da. Era berean, honek \bar{A} diagonalizagarria dela eta irudikari balio propio puruak dituela suposatzen du. Eta beraz, Q ($s \times s$) tamaineko matrize ortogonala existitzen da,

$$Q^{-1}\bar{A}Q = \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \quad (7.15)$$

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

Simetrikoa.

Simplektikotasun eta simetri propietate hauetan oinarrituz jarraiko matrizeak definituko ditugu.

1. P matrizea.

Kontsideratu $(s \times s)$ tamainako $P = (P_1 \ P_2)$ matrize ortogonala. $x = (x_1, \dots, x_s)^T \in \mathbb{R}^s$, $P_1^T x = (y_1, \dots, y_m)^T$, eta $P_2^T x = (y_{m+1}, \dots, y_s)^T$ non,

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

Matrizearen dimentsioak laburtuz, $P = (P_1 \ P_2) \in \mathbb{R}^{s \times s}$, $P_1 \in \mathbb{R}^{s \times m}$ eta $P_2 \in \mathbb{R}^{s \times (s-m)}$.

2. K matrizea.

Batetik, simetri propietateak (7.12), $P_i^T B^{\frac{1}{2}} \bar{A} B^{-\frac{1}{2}} P_i = 0$, $i = 1, 2$ eta bestetik, propietate simplektikoak $B^{1/2}\bar{A}B^{-1/2}$ antisimetrikoa dela ziurtatzen

duenez, \bar{A} matrizea honako matrizearen antzekoa dela ondorioztatu daiteke,

$$P^T B^{\frac{1}{2}} \bar{A} B^{-\frac{1}{2}} P = \begin{pmatrix} 0 & K \\ -K^T & 0 \end{pmatrix} \quad (7.16)$$

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

3. D matrizea.

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

$$D = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \sigma_{s-m} \end{pmatrix}, \quad D = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \sigma_{s-m} \\ 0 & 0 & \dots & 0 \end{pmatrix}. \quad (7.17)$$

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

4. Q matrizea.

Simetrikoa den (7.15) propietateari esker,

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

$$Q^{-1} = Q^T B.$$

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

(s+1) ekuazio-sistema.

Aldagai berri bat konsideratuko dugu $z \in \mathbb{R}^d$ eta honako ekuazio sistema konsideratuko dugu,

$$Y_{n,i} = y_n + \frac{z}{2} + h \sum_{j=1}^s \bar{a}_{ij} f(t_n + c_j h, Y_{n,j}) \quad i = 1, \dots, s,$$

$$z = h \sum_{i=1}^s b_i f(t_n + c_i h, Y_{n,i}).$$

Ekuazio berri hau, sistemari gehituz, $(s+1) \times d$ dimentsioko ekuazio sistema baliokidea dugu,

$$(I_s \otimes I_d - h \bar{A} \otimes J) \Delta Y - \frac{1}{2}(e_s \otimes I_d) \Delta z = r, \quad (7.18)$$

$$(-h e_s^T B \otimes J) \Delta Y + \Delta z = 0, \quad (7.19)$$

non $e_s^T = (1, \dots, 1) \in \mathbb{R}^{1 \times s}$ den. Argi dagoenez, $(\Delta Y, \Delta z)$ ekuazio-sistema honen (7.18) soluzioa bada, ΔY gure jatorrizko ekuazio sistemaren (7.11) soluzioa da.

Ekuazio-sistemaren adierazpen matriziala lagungarria izan daiteke,

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

IRK metodo orokoren garapena.

Lehengo, garapena metodo simplektikoentzat egingo dugu eta ondoren, metodoa simetrikoentzat. Honako aldagai aldaketa aplikatuz,

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

eta metodoa simetrikoa izateagatik (7.15), ekuazio sistema baliokidea lortuko dugu,

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

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

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

IRK metodo simetriko eta simplektikoen garapena.

Honako aldagai aldaketarekin,

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

non,

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

$$Q = (Q_1 \ Q_2), \quad Q_1 \in \mathbb{R}^{s \times m}, \quad Q_2 \in \mathbb{R}^{s \times (s-m)},$$

eta metodoa simetrikoa nahiz simplektikoa izateagatik $e_s^T B P_2 = 0$, $e_s^T B Q_2 = e_s^T B P_2 V = 0$ berdintasunak aplikatuz, honako ekuazio-sistema baliokidea lortuko dugu,

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

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

Matrizearen egitura. Ekuazio sistemaren adierazpen matrizialarekin ($s = 6$), matrizearen egitura berezia ikus daiteke. Aldagai aldaketarekin lortutako ekuazio sistema baliokideak, blokeka diagonala da eta hau aprobetxatzuz, Newton iterazioaren implementazio eraginkorra lortuko dugu.

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

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. Bigarren ekuaziotik W'' askatu,

$$W'' = -h (D^T \otimes J) W' + (Q_2^T B \otimes I_d) r. \quad (7.20)$$

W' kalkulatzeko ekuazioak. Eta lehen ekuazioan ordezkatuz,

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

$$\text{non } R = (Q_1^T B \otimes I_d) r + h (D Q_2^T B \otimes J) r \in \mathbb{R}^{md}.$$

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,

$$\begin{aligned} (I_d + h^2 \sigma_i^2 J^2) W_i - \frac{\alpha_i}{2} \Delta z &= R_i, \quad i = 1, \dots, m, \\ -h J \sum_{i=1}^m \alpha_i W_i + \Delta z &= 0, \end{aligned}$$

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 (7.17).

Δz kalkulatzeko ekuazioak. Lehen 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 bigarren ekuazioan ordezkatzuz, $\Delta z \in \mathbb{R}^d$ askatzeko ekuazioak lortuko ditugu,

$$M \Delta z = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i, \quad (7.21)$$

non

$$M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1} \in \mathbb{R}^{d \times d}. \quad (7.22)$$

7.4. IRK-Newton estandarra (formulazio berria).

Sarrera.

IRK-Newton implementazioarentzat, IRK puntu-finkoaren implementazioan erabilako birformulazio (6.atala) berdina aplikatuko dugu. Newton iterazio metodoan ordea, L_i ($i = 1, \dots, s$) aldagai ezezagunatzat eta Y_i ($i = 1, \dots, s$) aldagai laguntzailea konsideratzea [58] izango da egokiena,

$$L_{n,i} = hb_i f(Y_{n,i}), \quad Y_{n,i} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}, \quad i = 1, \dots, s, \quad (7.23)$$

$$y_{n+1} = y_n + \sum_{i=1}^s L_{n,i}, \quad (7.24)$$

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

Newton sinplifikatuaren iterazioa.

Newton metodo sinplifikatua, $k = 1, 2, \dots$ $L_i^{[k]}$ hurbilpenak kalkulatzeko,

$$1. \quad g_i^{[k]} = -L_i^{[k-1]} + hb_i f(t + c_i h, y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}), \quad i = 1, \dots, s.$$

2. Askatu $\Delta L_i^{[k]}$,

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

3. $L_i^{[k]} = L_i^{[k-1]} + \Delta L_i^{[k]}, \quad i = 1, \dots, s.$

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

Modu baliokidean, ekuazio lineala notazio matriziala erabiliz laburtu daiteke,

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

non,

$$L^{[k]} = \begin{bmatrix} L_1^{[k]} \\ \vdots \\ L_s^{[k]} \end{bmatrix} \in \mathbb{R}^{sd}, \quad g^{[k]} = \begin{bmatrix} g_1^{[k]} \\ \vdots \\ g_s^{[k]} \end{bmatrix} \in \mathbb{R}^{sd},$$

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

Newton super-simplifikatuaren iterazioa.

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

$$(I_s \otimes I_d - h BAB^{-1} \otimes J) \Delta L = g. \quad (7.25)$$

non

$$I_s, I_d \text{ identitateak eta } B = \begin{bmatrix} b_1 & 0 & \dots & 0 \\ 0 & b_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & b_s \end{bmatrix}.$$

Algoritmoa.

Formulazio berrian Newton super-simplifikatua deitu dugun algoritmoa laburtu du-gu (Algoritmoa 28).

Interpolazio koefizienteak. $L_{n,i}^{[0]}$ atalen hasieraketarentzat dagokien koefizienteak era honetan definituko ditugu. IRK puntu finkoaren implementazioan finkatu

```

 $\tilde{y}_0 = fl(y_0);$ 
 $e_0 = fl(y_0 - \tilde{y}_0);$ 
for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $k = 0;$ 
    Hasieratu  $L_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
     $Y_{n,i}^{[0]} = y_n + (e_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[0]});$ 
     $J = \frac{\partial f}{\partial y}(y_n);$ 
     $M = LU(I_s \otimes I_d - h BAB^{-1} \otimes J);$ 
    while (not konbergentzia) do
         $k = k + 1;$ 
         $g_i^{[k]} = -L_i^{[k-1]} + hb_i f(t + c_i h, y_n + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]});$ 
        Solve ( $M \Delta L^{[k]} = g^{[k]}$ );
         $L^{[k]} = L^{[k-1]} + \Delta L^{[k]};$ 
         $Y_{n,i}^{[k]} = y_n + (e_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]});$ 
        konbergentzia  $\leftarrow$  GeratzeErizpidea( $L^{[k]}$ ,  $L^{[k-1]}$ ,  $\Delta_{min}$ );
    end
    if ( $\exists j$  non  $\Delta_j^{[K]} \neq 0$ ) then
        if ( $NormalizeDistance(Y^{[k]}, Y^{[k-1]}) > 1$ ) then
            fail convergence;
        end
    end
     $(\tilde{y}_{n+1}, e_{n+1}) \leftarrow$  baturakonpensatua( $\tilde{y}_n, e_n, L_n^{[k]}$ );
end

```

Algoritmoa 28: IRK (Newton super-simplifikatua).

genituen interpolazio koefizienteetik abiatuta (6.9) modu errezean definituko ditugu formulazio honi dagozkion interpolazio koefizienteak.

$$\begin{cases} Y_{n,i}^{[0]} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[0]} \\ Y_{n,i}^{[0]} = y_n + \sum_{j=1}^s \nu_{ij} L_{n-1,j} \end{cases} \Rightarrow L_n^{[0]} = (Mu^{-1}Nu)L_{n-1}, \quad (Mu^{-1}Nu)_{i,j}^s = \lambda_{ij}/a_{ij}. \quad (7.26)$$

Geratze erizpidea. Puntu-finkoan definitutako geratze erizpide berdina aplikatuko dugu baina $L_{n,i}$, $i = 1, \dots, s$ aldagei.

$$\Delta^{[k]} = (L_1^{[k]} - L_1^{[k-1]}, \dots, L_s^{[k]} - L_s^{[k-1]}) \in \mathbb{R}^{sd},$$

Iterazioak jarraitu, honako baldintza betetzen den artean,

$$\exists j \in \{1, \dots, sd\} \text{ non} \\ \min \left(\{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} / \{0\} \right) > |\Delta_j^{[k]}| \text{ and } \Delta_j^{[k]} \neq 0. \quad (7.27)$$

Batura konpensatua. Batura konpesatua egiteko modua zehaztuko dugu. Lehenik ΔL_i gaiak gehituko ditugu,

$$\beta_n = e_n + \sum_{j=1}^s \Delta L_{n,j}^{[k]};$$

Bigarrenik, $y_{n+1} = y_n + \sum_{i=1}^s L_{n,i}^{[k-1]} + \beta_n$ batuketa egiteko

Function BaturaKonpensatua (y_n , β_n , $L_n^{[k-1]}$)

```

 $s_0 = y_n$ 
 $ee = \beta_n$ 
for  $i \leftarrow 1$  to ( $s$ ) do
     $s_1 = s_0$ 
     $\delta = L_{n,i}^{[k-1]} + ee$ 
     $s_0 = s_1 + \delta$ 
     $ee = (s_1 - s_0) + \delta$ 
end
```

$y_{n+1} = s_0$

$e_{n+1} = ee$

return (y_{n+1}, e_{n+1})

Algoritmoa 29: BaturaKonpensatua

7.5. IRK-Newton eraginkorra (formulazio berria).

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

$$(I_s \otimes I_d - h BAB^{-1} \otimes J) \Delta L = g, \quad (7.28)$$

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

Formulazio estandarretik formulazio berrirako urratsa.

Formulazio berriaren implementazio eraginkorra, formulazio estandarrean eman-dako algoritmoaren (7.3. atala) moldatuz zehaztuko dugu.

Hauek izango dira bi formulazioen aldagaien arteko erlazioak,

1. Aldagai aldaketa.

$$\begin{aligned}\Delta L &= (B \otimes I_d) \Delta Y, \\ \Delta L &= (BQ \otimes I_d) W, \\ \Delta L &= (BQ_1 \otimes I_d) W' + (BQ_2 \otimes I_d) W''.\end{aligned}$$

2. R matrizea.

$$g \in \mathbb{R}^{sd} \rightarrow r = (B^{-1} \otimes I_d)g.$$

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

3. W'' matrizea.

$$\begin{aligned}W'' &= -h (D^T \otimes J) W' + (Q_2^T B \otimes I_d) r, \\ W'' &= -h (D^T \otimes J) W' + (Q_2^T \otimes I_d) g,\end{aligned}$$

Formulazio berrian IRK-Newton sinplifikatuaren implementazioaren urratsak hauek dira,

1. Δz askatu.

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

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

2. W' askatu.

$$W'_i = (I_d + h^2 \sigma_i^2 J^2)^{-1} (R_i + \frac{\alpha_i}{2} \Delta z) \in \mathbb{R}^d, \quad i = 1, \dots, m$$

3. W'' .

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

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

$$W'' = (-hD^T) W' J^T + Q_2^T g,$$

4. ΔL .

$$\Delta L = (BQ_1 \otimes I_d) W' + (BQ_2 \otimes I_d) W'',$$

AlgoritmoaV1.

IRK-Newton gure implementazioaren algoritmoa laburtuko dugu (Algoritmoa 30),

```

 $\tilde{y}_0 = fl(y_0);$ 
 $e_0 = fl(y_0 - \tilde{y}_0);$ 
for  $n \leftarrow 0$  to ( $endstep - 1$ ) do
     $k = 0;$ 
    Hasieratu  $L_{n,i}^{[0]}$ ,  $i = 1, \dots, s$ ;
     $Y_{n,i}^{[0]} = y_n + (e_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[0]})$ ;
     $J = \frac{\partial f}{\partial y}(y_n);$ 
     $M = I_d - J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1};$ 
    LUM = LU( $M$ );
    while (not konbergentzia) do
         $k = k + 1;$ 
         $g = (hBF(L) - L);$ 
         $R = Q_1^T g + (hDQ_2^T) g J^T;$ 
         $d = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i;$ 
        Solve( $M \Delta z = d$ );
         $W' = (I_d + h^2 \sigma_i^2 J^2)^{-1} (R_i + \frac{\alpha_i}{2} \Delta z)$ ,  $i = 1, \dots, m$ ;
         $W'' = (-hD^T) W' J^T + Q_2^T g;$ 
         $\Delta L = BQ_1 W' + BQ_2 W'';$ 
         $L^{[k]} = L^{[k-1]} + \Delta L^{[k]}$ ;
         $Y_{n,i}^{[k]} = y_n + (e_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]})$ ;
        konbergentzia  $\leftarrow$  GeratzeErizpidea( $L^{[k]}$ ,  $L^{[k-1]}$ ,  $\Delta_{min}$ );
    end
    if ( $\exists j$  non  $\Delta_j^{[K]} \neq 0$ ) then
        if ( $NormalizeDistance(Y^{[k]}, Y^{[k-1]}) > 1$ ) then
            | fail convergence;
        end
    end
     $(\tilde{y}_{n+1}, e_{n+1}) \leftarrow$  baturakonpensatua( $\tilde{y}_n, e_n, L_n^{[k]}$ );
end

```

Algoritmoa 30: IRK (NSS-EraginkorraV1).

7.6. IRK Sasi-Newton.

Sarrera.

Doitasun laukoitzeko abantaila asko izango dituen proposamen berria egingo dugu. Doitasun laukoitzeko exekuzioetan, funtzioko ebaluaztapena oso garestia da eta funtzioko ebaluaztapen kopurua gutxitzea bilatuko dugu.

Newton osoa aplikatzen dugunean pena mereziko du (Newton osoak ez du atalen hasieraketa ona behar eta Newton sinplifikatuak bai).

Ideia nagusia hauxe da; Newton osoaren iterazioaren ekuazio sistema, modu iteratiboan askatzea,

$$\begin{aligned}\Delta L_i^{[k]} - hb_i J_i^{[k]} \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} &= g_i^{[k]}, \quad i = 1, \dots, s, \\ J_i^{[k]} &= \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[k]}), \quad i = 1, \dots, s,\end{aligned}$$

Notazio berria finkatzen badugu, hau da askatu beharreko ekuazio sistema,

$$M^{[k]} \Delta L^{[k]} = g^{[k]},$$

eta $G()$ funtzioa era honetan definituz,

$$G(\Delta L) := g - M \Delta L,$$

askatu nahi dugun ekuazioa,

$$G(\Delta L) = 0.$$

Aurreko atalean, $M^{[k]} \approx \tilde{M}$ hurbilketa, modu eraginkorrean askatzen ikusi dugu,

$$\tilde{M} = (I_s \otimes I_d - h B A B^{-1} \otimes J).$$

Eta beraz,

Sasi-Newton.

Non

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

$$G_i^{[k,l]} = J_i^{[k]} \cdot \left(\sum_{j=1}^s hb_i \mu_{ij} \Delta L_j^{[k,l-1]} \right) + (g_i^{[k]} - \Delta L_i^{[k,l-1]}), \quad i = 1, \dots, s.$$

```

 $\Delta L^{[0]} = 0;$ 
for  $l = 0, 1, 2, \dots$  do
     $\Delta L^{[l+1]} = \Delta L^{[l]} + \tilde{M}^{-1} G^{[l]},$ 
end

```

Algoritmoa 31: .

```

 $\Delta L^{[k,1]} = \tilde{M}^{-1} g^{[k]};$ 
for ( $l=2,3,\dots$ ) do
     $G^{[k,l]} = g^{[k]} - M^{[k]} \Delta L^{[k,l-1]},$ 
     $\Delta L^{[k,l]} = \Delta L^{[k,l-1]} + \tilde{M}^{-1} G^{[k,l]},$ 
end

```

Algoritmoa 32: Sasi-Newton Benetakoa.

Proposamen berri honetan,

- Iterazioa=1: *Newton Super-Simplifikatua* (NSS-eraginkorra).
- Iterazioa= ∞ : *Newton Benetako* (NB-eraginkorra).
- Iterazioa=2 edo 3: *Sasi-Newton Benetakoa* (SNB-eraginkorra).

7.7. 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 [41].

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.

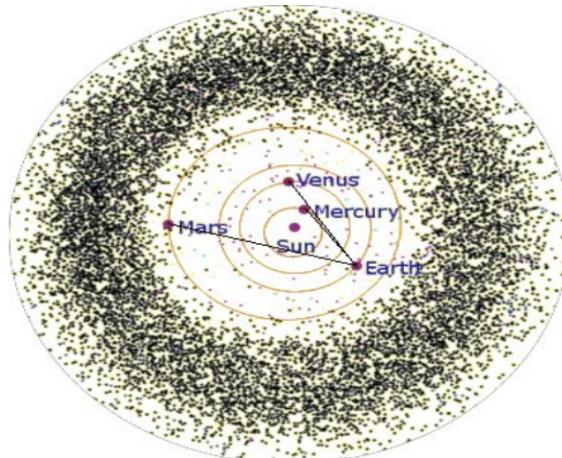
$$\dot{q}_i = v_i, \quad i = 0, 1, \dots, N \quad (8.1)$$

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

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* [25] 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 lengoian 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 [24].

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) [61] 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 [55] eguzki-sistemaren zenbakizko integrazioen algoritmoen garapenaren azterketan, garai hauek 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 tartreak $10^4 - 10^6$ dira.

2. Garai simplektikoa (The symplectic period).

Wisdom eta Holmanen [67, 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 [1]) 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 errebibio dokumentu interesgarri hauek aholkatu nahiko genituzke: [42] (Kholshevnikov 2007, 2.007), [11] (Brumberg, 2.013) eta [37] (Ito eta Tanikawa, 2.007).

Eguzki-sistemari egokitutako integratzaile simplektikoak.

Wisdomek eta Holmanenek bere lanean [67, 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, bere 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 eraginkorrean integratzeko eragozpen bat da. Arazo hauek gainditzeko, urteetan zehar algoritmo honen aldaerak proposatu dira eta jarraian, nagusienak aipatuko ditugu.

Levinson eta Duncan-ek [51, 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 [20, 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 [15] koordenatu Heliozentrikoetan oinarritu zen eta kolisio egoeretan, beste integratzaile (Bulirsch-Stoer metodoa) 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 [44] 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 [72] [49] eta ez dira interesgarriak, *leapfrog* metodoa hauek baino eraginkorragoa baita.

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

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

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

Beranduago, McLachlan-ek [53, 1995], Laskar-ek eta Robutel-ek [49, 2001] koefiziente negatiboen arazoa gainditu zuten eta orden altuko splitting eskemak aurkitu zituzten. 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 [8, 2012] orden altuko splitting eskema berriak eta eraginkorrapak aurkitu ditu.

Hernandez eta Bertschinger-ek [32, 2015] N-body problema gravitacional eta kolisiōdunetarako 2 ordeneko integratzaile simplektiko berri bat proposatu dute. Hernandez eta Bertschinger-ek [32, 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 $i = 0$ konsideratzen 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  $Y_{n,i}^{[k]} = y_n + \left( \sum_{j=1}^s \mu_{ij} \ hb_j \ K_{n,j}^{[k]} \right) + (e_n + W_{n,i}^{[k-1]});$ 
    while (konbergentzia lortu) do
         $k = k + 1;$ 
         $W_{n,i}^{[k-1]} = \sum_{j=1}^s \mu_{ij} \ hb_j \ g(Y_{n,j}^{[k-1]});$ 
         $K_{n,i}^{[k]} = k(Y_{n,i}^{[k-1]});$ 
        Askatu  $Y_{n,i}^{[k]} = y_n + \left( \sum_{j=1}^s \mu_{ij} \ hb_j \ K_{n,j}^{[k]} \right) + (e_n + W_{n,i}^{[k-1]});$ 
    end
    if ( $\text{NormalizeDistance}(Y^{[k]}, Y^{[k-1]}) > 1$ ) 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

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 34: 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 sinplifikatua problema independenteak osatzen dituzte eta barne iterazioak modu independentean kalkula daitezke. N-gorputzen eguzki-sistemaren adibidean, eredu sinplifikatua $k(y)$ (eguzkiarekiko interakzioa) planeta bakoitzarentzat 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 den 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  $Y_{n,i}^{[k]} = y_n + \left( \sum_{j=1}^s \mu_{ij} \ hb_j \ K_{n,j}^{[k]} \right) + (e_n + W_{n,i}^{[k-1]});$ 
    while (konbergentzia lortu) do
         $k = k + 1;$ 
         $W_{n,i}^{[k-1]} = \sum_{j=1}^s \mu_{ij} \ hb_j \ g_2(Y_{n,j}^{[k-1]});$ 
         $K_{n,i}^{[k]} = k(Y_{n,i}^{[k-1]});$ 
        Askatu  $Y_{n,i}^{[k]} = y_n + \left( \sum_{j=1}^s \mu_{ij} \ hb_j \ K_{n,j}^{[k]} \right) + (e_n + W_{n,i}^{[k-1]});$ 
    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 35: 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 grabitazio interakzioak .
rs(y): eguzkiarekiko erlatibilitatea efektua .
rn(y): planeten arteko erlatibilitatea efektua .

```

Adierazpena.

Ekuazio diferentzialen deskonposaketak, zuhaitz moduan adieraz daitezke.

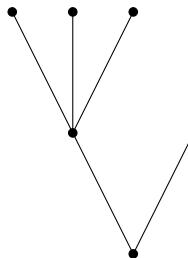
$$\dot{y} = f(y).$$

•

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



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



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 diferentziala,

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

non y menpeko aldagai eta t aldagai askea den.

Aldagai askeari aldaketa bat aplikatuz ($s()$ izeneko funtzio), ekuazio differenziala 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 $z(\tau)$ funtzioak deskribatzen du: z aldagai berria τ aldagai askearen menpeko 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.

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

$$r_{10} = \|q_1 - q_0\|_2 \quad (8.10)$$

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 differenziala dugularik,

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

Alde kepleriarren 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,

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

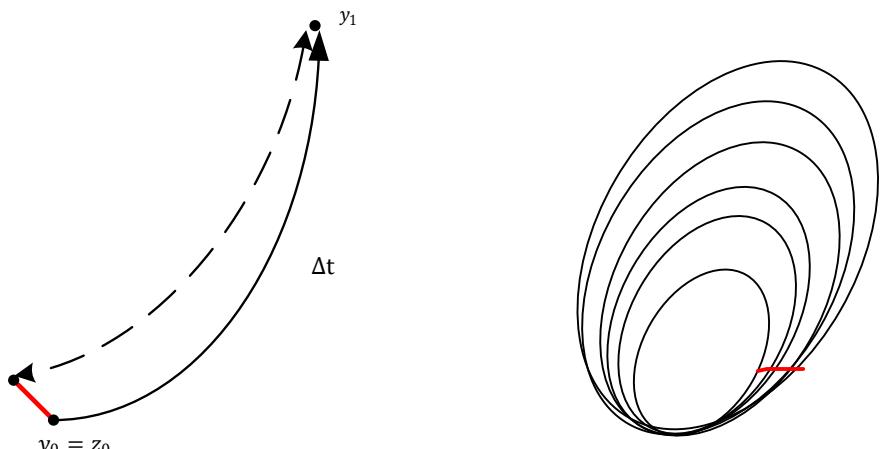
Idea hau bi modutara aplika daiteke,

1. Gauss implizituaren integrazio 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.

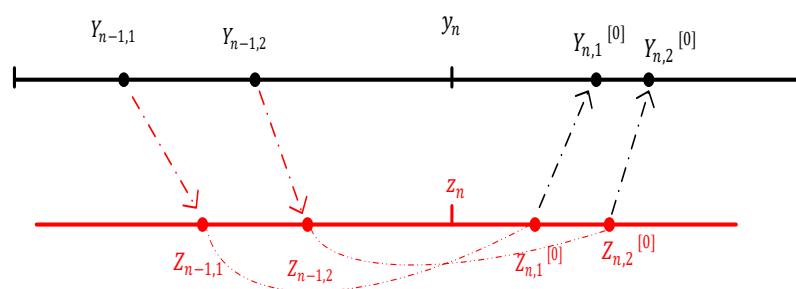
8.6. Laburpena.



(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 integracio konstantea da. E (*eccentric anomaly*) eta t -ren arteko erlazio hau erabiliz kalkulatzen da mugimendua. Mugimendu eliptikoaren kasura mugatuko gara ($0 \leq e < 1$) eta Keplerren ekuazioa transentalala denez, zenbakizko metodo baten bidez ebatziko dugu.

Garapena.

Gure abiapuntua, honakoa da,

$$\begin{aligned} E_0 - e \sin E_0 &= n(t_0 - t_p), \\ E_1 - e \sin E_1 &= n(t_1 - t_p) \end{aligned}$$

non $n = 2\pi/P$ eta P periodoa diren.

Bi ekuazioen arteko kendura eginez,

$$E_1 - E_0 - e(\sin(E_1) - \sin(E_0)) = n\Delta t \longrightarrow \Delta E - e(\sin(E_0 + \Delta E) - \sin(E_0)) = n\Delta t$$

non $E_1 = E_0 + \Delta E$ den.

Honako notazioa erabiliz adieraziko dugu,

$$\Delta E - ce \sin(\Delta E - se(\cos(\Delta E) - 1)) = n\Delta t,$$

non $ce = e \cos(E_0)$ eta $se = e \sin(E_0)$ den.

Newton metodoa. Ekuazio ebatzeko, Newton metodoa aplikatuko dugu,

1. $f(\Delta E) = \Delta E - ce \sin(\Delta E) - se(\cos(\Delta E) - 1) - n\Delta t = 0.$
2. $f'(\Delta E) = 1 - ce \cos(\Delta E) + se \sin(\Delta E).$
3. $\Delta E^{[k+1]} = \Delta E^{[k]} - \frac{f(\Delta E^{[k]})}{f'(\Delta E^{[k]})}.$

Hasierako balioa. $\Delta E^{[0]}$ hasierako balioa, finkatzea da dugun zaitasun handiena. Horretarako honako garapena egingo dugu,

$$\begin{aligned}\Delta E - ce \sin(\Delta E - se (\cos(\Delta E) - 1)) &= n\Delta t, \\ x = \Delta E - n\Delta t,\end{aligned}$$

eta beraz,

$$x - ce \sin(n\Delta t + x) - se(\cos(n\Delta t + x) - 1) = 0.$$

Honako baliokidetasun trigonometrikoak ordezkatzuz,

$$\begin{aligned}\cos(A + B) &= \cos(A)\cos(B) - \sin(A)\sin(B), \\ \sin(A + B) &= \sin(A)\cos(B) + \cos(A)\sin(B),\end{aligned}$$

berdintza hau lortzen dugu,

$$\begin{aligned}x - (se \cos(n\Delta t) + ce \sin(n\Delta t)) \cos(x) \\ + (se \sin(n\Delta t) - ce \cos(n\Delta t)) \sin(x) + se = 0.\end{aligned}$$

x txikia dela suposatuz, honako hurbilpenak ordezkatuko dugu,

$$x \approx \sin(x), \cos(x) \approx 1 - \frac{x^2}{2}$$

eta honako berdintza lortuko dugu,

$$\begin{aligned}(se \cos(n\Delta t) + ce \sin(n\Delta t)) \frac{x^2}{2} \\ + (1 + se \sin(n\Delta t) - ce \cos(n\Delta t))x - (se) = 0.\end{aligned}$$

Azkenik, goiko ekuazio hau askatuz ($Ax^2 + Bx + C = 0 \rightarrow x = -B \pm \sqrt{B^2 - 4AC}/2A$) lortuko dugu,

$$\Delta E^{[0]} = x + n\Delta t.$$

Koordenatu kartesiarren kalkulua ekuazio hauen bidez egindo dugu,

$$(q_1, v_1) = (q_0, v_0) + (q_0, v_0) \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$\begin{aligned} b_{11} &= (C - 1) \frac{a}{\|q\|}, \\ b_{21} &= \Delta t + (S - \Delta E) \frac{a^{\frac{3}{2}}}{\mu^{\frac{1}{2}}} \\ b_{12} &= \frac{s}{\|q\| \sqrt{a(1 - ce C + se S)}}, \\ b_{22} &= \frac{C - 1}{1 - ce C + se S}. \end{aligned}$$

Non osagai bakoitzaren definizioa,

$$\begin{aligned} C &= \cos(\Delta E), \quad S = \sin(\Delta E), \\ ce &= e \cos(E_0) = \|q\| \|v\|^2 - 1, \\ se &= e \sin(E_0) = \frac{(q \cdot v)}{\sqrt{\mu a}}, \\ a &= \frac{\mu \|q\|}{2\mu - \|q\| \|v\|^2}, \\ n &= \frac{\mu^{\frac{1}{2}}}{a^{\frac{3}{2}}}. \end{aligned}$$

Ezabapen arazoa. ΔE txikia denean, $\cos(\Delta E) - 1$ espresioaren kalkuluaren ezabapen arazoak biribiltze errore handia eragin dezake. Hori konpontzeko balio-kidetasun trigonometriko hau erabiliko dugu,

$$\cos(\Delta E) - 1 = -\frac{(\sin^2(\Delta E))}{1 + \cos(\Delta E)}.$$

Eta beraz, Keplerren ekuazioak hauek izango dira,

$$f(\Delta E) = \Delta E - ce \sin(\Delta E) + se \left(\frac{(\sin^2(\Delta E))}{1 + \cos(\Delta E)} \right) - n \Delta t = 0$$

Eta (q_1, v_1) balioak kalkulatzeko,

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

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 .

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

3. U_1 .

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

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[#1 dx] &], {i, 1, s}];
yy = Collect[ff, f[_], Simplify[Integrate[#1 dx] &];

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 [29] 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] 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.
- [3] Pavan Balaji. *Programming models for parallel computing*. MIT Press, 2015.
- [4] Josh Barnes and Piet Hut. A hierarchical $O(n \log n)$ force-calculation algorithm. *nature*, 324(6096):446–449, 1986.
- [5] André Berger. *A brief history of the astronomical theories of paleoclimates*. Springer, 2012.
- [6] Gregory Beylkin and Kristian Sandberg. Ode solvers using band-limited approximations. *Journal of Computational Physics*, 265:156–171, 2014.
- [7] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *arXiv preprint arXiv:1411.1607*, 2014.
- [8] 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.
- [9] 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.

- [10] Dirk Brouwer. [On the accumulation of errors in numerical integration](#). *The Astronomical Journal*, 46:149–153, 1937.
- [11] VA Brumberg. [Celestial mechanics: Past, present, future](#). *Solar System Research*, 47(5):347–358, 2013.
- [12] John C Butcher. [On the implementation of implicit runge-kutta methods](#). *BIT Numerical Mathematics*, 16(3):237–240, 1976.
- [13] John Charles Butcher. [Numerical Methods for Ordinary Differential Equations](#). Second edition, Wiley, 2008.
- [14] 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.
- [15] 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.
- [16] Robert M Corless and Nicolas Fillion. [A graduate introduction to numerical methods](#). *AMC*, 10:12, 2013.
- [17] John Danby. [Fundamentals of celestial mechanics](#). *Richmond: Willmann-Bell, | c1992, 2nd ed., 1*, 1992.
- [18] Theodorus Jozef Dekker. [A floating-point technique for extending the available precision](#). *Numerische Mathematik*, 18(3):224–242, 1971.
- [19] DumitruN. Deleanu. [Fast detection of chaotic or regular behavior of double pendulum system: application of the fast norm vector indicator method](#).
- [20] 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.
- [21] 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.
- [22] 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.

- [23] Kang Feng and Mengzhao Qin. *Symplectic geometric algorithms for hamiltonian systems*. Springer, 2010.
- [24] A Fienga, H Manche, J Laskar, and Mickael Gastineau. Inpop06: a new numerical planetary ephemeris. *Astronomy & Astrophysics*, 477(1):315–327, 2008.
- [25] 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.
- [26] Toshio Fukushima. Reduction of round-off error in symplectic integrators. *The Astronomical Journal*, 121(3):1768, 2001.
- [27] 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.
- [28] E Hairer and G Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer, Berlin, 1996.
- [29] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.
- [30] 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.
- [31] Wayne B Hayes. Is the outer solar system chaotic? *Nature Physics*, 3(10):689–691, 2007.
- [32] 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.
- [33] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. Siam, 2002.
- [34] Nicholas J Higham. Programming languages: An applied mathematics view. 2015.
- [35] Leslie Hogben. *Handbook of linear algebra*. Chapman and Hall/CRC, 2013.

- [36] 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.
- [37] Takashi Ito and Kiyotaka Tanikawa. Trends in 20th century celestial mechanics. *Publ. Natl. Astron. Obs. Jpn*, 9:55–112, 2007.
- [38] MP Calvo JM Sanz-Serna. *Numerical Hamiltonian problems*. Chapman and Hall, 1994.
- [39] 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.
- [40] W Kahan. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.
- [41] 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.
- [42] 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.
- [43] D Knuth. The art of computer programming: Vol 2/seminumerical algorithms, 1969.
- [44] 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.
- [45] MP Laburta. [Construction of starting algorithms for the rk-gauss methods](#). *Journal of computational and applied mathematics*, 90(2):239–261, 1998.
- [46] 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.

- [47] 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.
- [48] 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.
- [49] Jacques Laskar and Philippe Robutel. High order symplectic integrators for perturbed hamiltonian systems. *Celestial Mechanics and Dynamical Astronomy*, 80(1):39–62, 2001.
- [50] Benedict Leimkuhler and Sebastian Reich. *Simulating hamiltonian dynamics*, volume 14. Cambridge University Press, 2004.
- [51] Harold F Levison and Martin J Duncan. The long-term dynamical behavior of short-period comets. *Icarus*, 108(1):18–36, 1994.
- [52] 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.
- [53] Robert I McLachlan. Composition methods in the presence of small parameters. *BIT Numerical Mathematics*, 35(2):258–268, 1995.
- [54] Robert I McLachlan and Pau Atela. The accuracy of symplectic integrators. *Nonlinearity*, 5(2):541, 1992.
- [55] A Morbidelli. Modern integrations of solar system dynamics. *Annual Review of Earth and Planetary Sciences*, 30(1):89–112, 2002.
- [56] 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.
- [57] Nobelprize.org. The nobel prize in chemistry 2013, May 2014.
- [58] Hans Olsson and Gustaf Sderlind. The approximate runge-kutta computational process. *BIT Numerical Mathematics*, 40(2):351–373, 2000.
- [59] Michael L Overton. *Numerical computing with IEEE floating point arithmetic*. Siam, 2001.
- [60] Peter Pacheco. *An introduction to parallel programming*. Elsevier, 2011.

- [61] EV Pitjeva and NP Pitjev. [Development of planetary ephemerides epm and their applications.](#) *Celestial Mechanics and Dynamical Astronomy*, 119(3-4):237–256, 2014.
- [62] Prasenjit Saha and Scott Tremaine. [Long-term planetary integration with individual time steps.](#) *arXiv preprint astro-ph/9403057*, 1994.
- [63] Jesús María Sanz-Serna. [Symplectic integrators for hamiltonian problems: an overview.](#) *Acta numerica*, 1:243–286, 1992.
- [64] 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.
- [65] Mark Sofroniou and Giulia Spaletta. [Derivation of symmetric composition constants for symmetric integrators.](#) *Optimization Methods and Software*, 20 (4-5):597–613, 2005.
- [66] Pat H Sterbenz. [Floating-point computation.](#) Prentice Hall, 1973.
- [67] Gerald J Sussman and Jack Wisdom. [Chaotic evolution of the solar system.](#) Technical report, DTIC Document, 1992.
- [68] 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.
- [69] 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.
- [70] Jack Wisdom and Matthew Holman. [Symplectic maps for the n-body problem.](#) *The Astronomical Journal*, 102:1528–1538, 1991.
- [71] Wolfram Research, Inc. [Mathematica.](#)
- [72] 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.