

---

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

---

Mikel Antonana Otano

Konputazio zientzia eta Adimen Artifiziala saila  
Informatika fakultatea



Doktorego tesia  
Donostia 2017



# Gaien Aurkibidea

<b>1. Sarrera.</b>	<b>3</b>
<b>2. Zenbakizko integratzaile simplektikoak.</b>	<b>11</b>
<b>3. Problemak.</b>	<b>37</b>
<b>4. Koma higikorreko aritmetika.</b>	<b>51</b>
<b>5. IRK: Puntu-finkoaren iterazioa.</b>	<b>61</b>
<b>6. IRK: Newtonen Iterazioa.</b>	<b>85</b>
<b>7. IRK: Eguzki-sistema.</b>	<b>117</b>
<b>8. Eztabaida.</b>	<b>135</b>
<b>9. Ondorioak.</b>	<b>155</b>
<b>I. Eranskinak</b>	<b>159</b>
<b>A. Konputazio zientzia.</b>	<b>161</b>
<b>B. Ekuazioen garapenak.</b>	<b>179</b>
<b>C. Implementazioak.</b>	<b>191</b>
<b>D. Zerrendak</b>	<b>195</b>
<b>Bibliografia</b>	<b>197</b>



# Gaien Aurkibidea (detailed)

<b>1. Sarrera.</b>	<b>3</b>
1.1. Ikerketaren testuingurua . . . . .	3
1.2. Motibazioa . . . . .	4
1.3. Helburua eta esparrua . . . . .	5
1.4. Ekarpenak . . . . .	7
1.5. Tesiaren egitura . . . . .	10
<b>2. Zenbakizko integratziale sinplektikoak.</b>	<b>11</b>
2.1. Sarrera . . . . .	11
2.2. Zenbakizko integrazio metodoak . . . . .	11
2.3. Runge-Kutta metodo sinplektikoak . . . . .	20
2.4. Konposizio eta Splitting metodoak . . . . .	29
2.5. Laburpena . . . . .	36
<b>3. Problemak.</b>	<b>37</b>
3.1. Sarrera . . . . .	37
3.2. Pendulu bikoitza . . . . .	38
3.3. N-Gorputzen problema . . . . .	41
3.4. Eguzki-sistema . . . . .	43
3.5. Laburpena . . . . .	47
<b>4. Koma higikorreko aritmetika.</b>	<b>51</b>
4.1. Sarrera . . . . .	51
4.2. IEEE-754 estandarra . . . . .	51
4.3. Biribiltze errorea . . . . .	54
4.4. Biribiltze errorea gutxitzeko teknikak . . . . .	57
4.5. Laburpena . . . . .	60
<b>5. IRK: Puntu-finkoaren iterazioa.</b>	<b>61</b>
5.1. Sarrera . . . . .	61
5.2. Hairer-en implementazioa . . . . .	62

5.3. Gure implementazioa . . . . .	64
5.4. Zenbakizko esperimentuak . . . . .	75
5.5. Laburpena . . . . .	83
<b>6. IRK: Newtonen Iterazioa.</b>	<b>85</b>
6.1. Sarrera . . . . .	85
6.2. IRK-Newton estandarra . . . . .	86
6.3. IRK-Newton eraginkorra . . . . .	91
6.4. IRK-Newton estandarra (formulazio berria) . . . . .	99
6.5. IRK-Newton eraginkorra (formulazio berria) . . . . .	104
6.6. IRK Newtonen iterazio mistoa . . . . .	108
6.7. Zenbakizko esperimentuak . . . . .	112
6.8. Laburpena . . . . .	115
<b>7. IRK: Eguzki-sistema.</b>	<b>117</b>
7.1. Sarrera . . . . .	117
7.2. Kepler-en fluxua . . . . .	118
7.3. Kepler Perturbatuaren problema . . . . .	119
7.4. Alde Kepleriar bat baino gehiagoko sistemak . . . . .	122
7.5. Zenbakizko esperimentuak . . . . .	126
7.6. Laburpena . . . . .	132
<b>8. Eztabaida.</b>	<b>135</b>
8.1. Eguzki-sistemaren integrazio luzeak . . . . .	135
8.2. Eredu deskonposaketa . . . . .	138
8.3. IRK implementazioaren oinarriak . . . . .	143
8.4. Doitasun altuko konputazioak . . . . .	147
8.5. Aldagai aldaketa . . . . .	148
8.6. Paralelizazioa . . . . .	149
8.7. Implementazioaren aplikagarritasuna . . . . .	151
<b>9. Ondorioak.</b>	<b>155</b>
<b>I. Eranskinak</b>	<b>159</b>
<b>A. Konputazio zientzia.</b>	<b>161</b>
A.1. Sarrera . . . . .	161
A.2. Eraginkortasuna . . . . .	162
A.3. Hardwarea . . . . .	165
A.4. Programazio lengoaiak . . . . .	170
A.5. Algebra lineal dentsorako liburutegiak . . . . .	173

A.6. Konpiladorea . . . . .	177
A.7. Laburpena . . . . .	178
<b>B. Ekuazioen garapenak.</b>	<b>179</b>
B.1. Kepler hasierako baliodun problema. . . . .	179
B.2. Koordenatu sistemak . . . . .	182
B.3. Newton eraginkorraren garapena . . . . .	185
B.4. Kepler fluxuaren aldagai aldaketa. . . . .	187
<b>C. Inplementazioak.</b>	<b>191</b>
C.1. Egitura orokorra . . . . .	191
C.2. IRK puntu-finkoa. . . . .	193
C.3. IRK Newton. . . . .	193
C.4. IRK Eguzki-sistema. . . . .	194
C.5. Konposizio-Splitting metodoak. . . . .	194
<b>D. Zerrendak</b>	<b>195</b>
<b>Bibliografia</b>	<b>197</b>



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

---

*Vaclav Havel*



# 1. Kapitulua

## Sarrera.

### 1.1. Ikerketaren testuingurua.

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

Garai batean, N-gorputzen problemak teoria analitikoen bidez aztertzen ziren baina konputagailuen sorrerarekin, zenbakizko integrazioak tresna nagusi bilakatu ziren. Azken hamarkadetan, bai konputazio teknologien aurrerapenari esker bai algoritmo berrien sorrerari esker, zenbakizko azterketek garapen handia izan du. Zenbakizko simulazioen laguntzaz, eguzki-sistemaren dinamikaren funtsezko galdera batzuk ezagutu ditugu eta berriki, Karplus-en taldeak 2013. urteko kimika Nobel saria [92] jaso du, kimika konputazionalean egindako lanarengatik.

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

1. Epe motzeko eta doitasun handiko integrazioak. Eguzki-sistemaren efemide zehatzak [41] edo espazioko satelite artifizialen kokapenen [13] kalkuluetarako erabili ohi dira.
2. Epe luzeko baina doitasun txikiko integrazioak. Denbora epe luzean, planeta-sistemen mugimendua ezagutzeko egindako ikerketak dira. Azterketa hauetan, helburua gorputzen mugimenduaren argazki orokorra (zehaztasun handirik gabe) ezagutzea da. Normalean, problema mota hauetan gorputzen arteko kolisioak edota kolisiotik gertuko egoerak ez dira izaten.

3. Gorputz kopurua edozein izanik, hauen arteko kolisioak gerta daitezkeen problemak. Integrazio hauetan, konplexutasun handiari aurre egin behar zaio. Gorputz kopurua miliotakoa [63] izan daiteke eta kolisiotik gertuko egoeren ondorioz, kalkuluetan egindako zenbakizko errore txikiiek soluzioan eragin handia izan dezakete.

Gure helburua, eguzki-sistemaren epe luzeko eta doitasun handiko integracioetarako egokia izango den implementazio eraginkorra garatzea da. Aurreko hamarkadetan, eguzki-sistemaren planeten epe luzeko zenbakizko integrazioa erronka garrantzitsua izan da. Adibidez, Sussman-ek eta Wisdom-ek [108, 1993] eguzki-sistemaren 100 milioiko integrazioarekin, planeten mugimendua kaotikoa zela baiezttatu zuten. Aldi berean, paleoklimatologi-zientzialariak orain milioika urte gertatutako klima zikloak (epel, hotz eta glaziazio aroak) azaltzeko, luraren orbitan izandako aldaken eraginez gertatu zirela azaltzen duen teoria (Milankovitch 1941) [12] baiezttatzeko, planeten orbiten efemeride zehatzetan oinarritu dira.

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

Konputazio-teknologiaren aurrerapenak handiak izan arren, eguzki-sistemaren simulazio hauek konputazionalki oso garestiak dira eta exekuzio denbora luzeak behar dituzte; adibidez, Laskar-ek [80, 2010] bere azken integrazioa burutzeko 18 hilabete behar izan zituen. Azken urteotako konputagailu berrien arkitekturaren bilakaerak, algoritmo azkarren diseinua aldatu du: simulazioak azkartzeko algoritmoak, paralelizazioan oinarritu behar dira. Integrazio luze hauen erronka handienetako bat, biribiltze errorearen garapena zaintza da. Biribiltze errore sistematikoaren hedapenak, errore globalean eragindako joerak ekidin behar dira [79].

## 1.2. Motibazioa.

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

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

Gaussen metodo implizitu simplektikoaren azterketa egingo dugu. Hainbat autorek (Hairer [54, 55] eta Sanz Serna[67]), doitasun altuko integrazioetarako metodo honen potentziala nabarmendu dute. Azken urteetan, espazioko satelite artifizialen arloan ere, Gaussen integrazio metodoarekiko interesa azaldu dute [17, 13].

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

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

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

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

### 1.3. Helburua eta esparrua.

Gure helburua, eguzki-sistemaren epe luzeko integrazorako Gaussen metodo implizituaren implementazio eraginkorra proposatzea edota, bide horretan aurrera-pausoak ematea da. Helburu hau lortzeko, honako aspektu hauek bereziki zainduko ditugu: eguzki-sistemaren problemaren ezaugarriak, biribiltze erroreen garapena eta egungo konputagailuen gaitasunari egokitutako algoritmo azkarren seinua.

N-gorputzeko problema gravitacionalari dagokionez, eguzki-sistemaren eredu simplea integratuko dugu. Eguzki-sistemaren gorputzak masa puntuak kontsideratuko ditugu eta gure ekuazio diferentzialek, gorputz hauen arteko erakarpen

grabitacionalak bakarrik kontutan hartuko dituzte. Beraz, eguzki-sistemaren eredu konplexuagoetako erlatibilitatea efektua, gorputzen formaren eragina, eta beste zenbait indar ez-grabitacional ez ditugu kontutan hartu.

Zeintzuk dira eguzki-sistemaren problemaren ezaugarri bereziak? Batetik, planeten mugimendu orbitala, perturbazio txikiak dituen mugimendu Keplerrira da. Mugimendu Keplerrira zehazki kalkula daiteke eta eguzki-sistemaren planeten orbiten konputazioaren oinarria da. Bestetik, badugu gorputz nagusi bat (eguzkia) eta honen inguruan mugimenduan dauden planetak, bi multzotan bana ditzakegu: barne-planetak, masa txikikoak eta eguzkitik gertu daudenak eta kanpo-planetak, masa handikoak eta eguzkitik urrun daudenak. Kanpo-planeten eboluzioan, barne-planetak eragin oso txikia daukate, eragina, masaren eta distanziaren alderantzikaren proporcionala baita. Eguzki-sistema egonkorra kontsideratzen da, hau da, hurrengo bilioi urteetan planeten arteko talkarik ez da espero gertatzea. Orbiten denbora eskalak anitzak dira; lurraren inguruko ilargiaren orbitaren periodoa 27.32 egunetakoa, eguzkiaren inguruko lurrarena urtebetekoa eta Neptunorena 163 urtekoa. Eguzki-sistemaren egitura aberats 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. Integrazio luzeetan, biribiltze errorea hedatu egiten da eta une batetik aurrera, soluzioen zuzentasuna ezereztatzen da. Zentzu honetan, doitasuna hobetzeko biribiltze errorea gutxitzen duten teknika bereziak aplikatzea ezinbestekoa izaten da. Integrazio luzeetan, maiz doitasun handian lan egiteko aukera aipatzen da, baina doitasun altuko aritmetikaren (128-bit) implementazioa software bidezkoa denez, oso motela da eta ez da erabilgarria. Exekuzio denbora onargarriak lortzeko irtenbideak landu behar dira, esate baterako, doitasun ezberdinak nahasten dituzten implementazioak.

Gauss metodoen *s*-ataletako funtzioen konputazioak, paraleloan exekuta daitzke. *s*-atal askotako metodoak aplikatu nahi ditugu, era honetan paralelizazio gaitasuna honek, abantaila handiagoa suposatuko baitu.

Konputazioko teknologiaren garapenean, algoritmo azkarren diseinua baldintzatzen duten bi ezaugarri azpimarratu behar dira. Batetik, egungo konputagailuak paraleloak dira eta algoritmo azkarra 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 eredu simplea

da eta logikoa da pentsatzea eredu konplexuagoetan, paralelizazioak abantaila handiagoa erakutsiko duela. Bestalde, gorputzen kopurua handia den probleman, hauen arteko interakzio kopuru handia ( $\mathcal{O}(N^2)$ ) kalkulatu behar da eta indar hauen hurbilpena modu eraginkorrean kalkulatzeko metodo ezagunak daude: *tree code*[11] eta *fast multipole method*[24] izeneko metodoak. Baino aplikatuko dugu eguzki-sistemaren eredu simplea gorputz kopurua txikia denez, teknika hauek gure eremutik kanpo utzi ditugu.

## 1.4. Ekarpenak.

Tesiaren lana, hiru urratsetan banatu dugu. Lehen urratsean, puntu-finkoaren iterazioan oinarritutako Gauss metodoaren implementazio estandarra aztertu dugu. Implementazio berriak, biribiltze errorearen garapen optimoa izatea lehenetsi dugu eta zentzu honetan, implementazio estandarraren hobekuntzak proposatu ditugu. Azterketa honen ondorioz, Gauss metodoaren implementazioen oinarriak finkatu ditugu: IRK metodoen formulazio eta geratze irizpidea berriak.

Bigarren urratsean, Newtonen iterazioan oinarritutako Gauss metodoaren implementazioa jorratu dugu. Problema zurruna denean, puntu-finkoaren iterazioa ez da eraginkorra eta Newtonen iterazioa aplikatu behar da. Newtonen iterazioa konputazionalki garestia da eta ekuazio-sistema modu eraginkorrean askatzeko teknika berri bat garatu dugu. Era berean, Newton sinplifikatuaren iterazioaren aldaera bat proposatu dugu.

Hirugarren urratsean, eguzki-sistemaren epe luzeko integrazioak aztertuko ditugu. Eguzki-sistemaren problemaren integrazorako, aurreko bi implementazioen artean, puntu-finkoarena eraginkorragoa dela baiezta dugu. Hori dela-eta, puntu-finkoaren iterazioan oinarritutako IRK implementazioa erabiliko dugu eta eguzki-sistemaren problemarako pentsatutako aldagai aldaketaren bidez, integracio era-ginkorra lortu dugu.

Jarraian, atal bakoitzean egindako ekarpen nagusienak laburtuko ditugu:

1. IRK puntu-finkoaren iterazioa.

Gauss metodoaren puntu-finkoaren implementazioaren azterketa sakon bat egin dugu eta horretarako, Hairer-en implementazioa [55] hartu dugu gure lanaren abiapuntutzat. Implementazio hau hobetzeko aukerak ikusi ditugu eta implementazioa hobetzeko ekarpenak, hauek izan dira:

- (a) Metodoaren birformulazioa.

Gauss metodoa 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 simplekti-

koa izateko baldintza zehazki betetzen eta beraz, izaera simplektikoen propietate onak galtzen dira. Metodoaren birformulazio baliokide bat proposatu dugu, horrela simplektizitatea baldintza zehazki betetzen duten koefizienteak modu errazean finkatu daitezke. Hori dela-eta, Gauss metodoak integral koadratikoak kontserbatuko ditu.

(b) Geratze irizpide berria.

Orokorrean, Hairer-en implementazioaren puntu-finkoaren iterazioaren geratze irizpidea zuzena dela ikusi dugu baina kasu batzuetan goizagi geratzen dela baiezta dugu. Geratze irizpidea bi zentzutan garatu/zorrotzu dugu. Hairer-en implementazioan suposatzen da, iterazio guzietan konbergentzia hobekuntza egon behar dela eta hobekuntzaren tamaina, norma batean oinarritzen da. Normarekiko independentea den geratze irizpidea aplikatzea zuzenagoa da, eta horregatik, ataletako edozein osagaiaren differentzia txikitzen den bitartean iterazioak egiten jarraitza finkatu dugu. Bigarrenik, iterazioetan osagai guztiak hobekuntza ez du zertan beherakorra izan, eta okertzen diren tarteko iterazioak gerta daitezke. Arazo hau gainditzeko, iterazioren batean osagai guztiak differentzia handitza gertatzen denean, seguritateko iterazio gehigarriak emango ditugu, iteraziotik irten aurretik.

(c) Atalen espresioaren aldaketa.

Integrazioan batura konpensatu estandarra aplikatzen dugunean,  $\tilde{y}_n, e_n \in \mathbb{F}^d$  zenbakizko soluzioa lortzen dugu non  $\tilde{y}_n + e_n \approx y(t_n)$  den. Hori dela-eta, IRK metodoaren atalak askatzeko ekuazio-sisteman,

$$Y_{n,i} = \tilde{y}_n + \left( e_n + h \sum_{j=1}^s a_{ij} f(Y_{n,j}) \right),$$

$\tilde{y}_n$ -ren ordez,  $\tilde{y}_n + e_n$  erabiltzea proposatu dugu. Aldaketa honekin, zenbakizko soluzioaren doitasuna zerbait hobetu dela espero da.

(d) Biribiltze errorearen estimazioa.

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

2. IRK Newtonen iterazioa.

(a) Ekarpen nagusia.

*S*-ataletako IRK metodoa, Newton sinplifikatuaren iterazioaren bidez  $d$ -dimentsioko ekuazio differentzial sistemari aplikatzeko, urrats bakoitzean  $sd \times sd$  tamainako era honetako ekuazio-sistemak (iterazio batkoitzeko bat) askatu behar dira,

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

Kapitulu honetan, jatorrizko  $sd$ -dimentsioko ekuazio sistema,  $(s+1)d$  dimentsioko ekuazio-sistema baliokide moduan berridatzi dugu. Ekuazio-sistema baliokidea,  $d \times d$  tamainako  $[s/2]+1$  matrize errealen *LU* deskonposaketa bidez askatuko dugu. Modu honetan, konputazioa eraginkorragoa da.

- (b) Newtonen iterazio mistoa.

Newton sinplifikatuaren iterazioaren aldaera bat proposatu dugu. Implementazio honekin, energia errorearen eboluzioan ez da drift-arik agertzen.

### 3. IRK eguzki-sistema.

- (a) Aldagai aldaketa.

Puntu-finkoaren iterazioan oinarritutako IRK implementazioa erabiliko dugu eta eguzki-sistemaren ekuazio differentzialei Kepler-en fluxuan oinarritutako aldagai aldaketa, aplikatuko dugu: era honetan, alde Kepleriarra ekuazioetatik desagerrazten dugu. Aldagai aldaketaren bidez aplikatzen dugun integrazio metodoa, simplektikoa eta simetrikoa da: neurri batean, Splitting metodoen baliokidea. Aldagai berriekiko ekuazio differentzialak, magnitude txikiko balioak hartzen dituzte eta honek, abantaila hauek ekarriko ditu: integrazioetan urratsa handiak erabiltzea eta puntu-finkoaren iterazioek konbergentzia azkarra izatea.

- (b) Doitasun nahasia.

Doitasun nahasia erabiliko dugu: konputazioaren zati batzuk doitasun altuagoan kalkulatuz, soluzioaren doitasuna hobetzea lortuko dugu.

- (c) Kepler-en fluxua.

Aldagai aldaketa, Kepler-en fluxuan oinarritzen da eta beraz, Kepler-en fluxuaren implementazio eraginkorra garatu dugu. Danby [29] eta J.Wisdom-en [112] implementazioen ezaugarrietan oinarritutako garapenean, hobekuntza bat egin dugu.

## 1.5. Tesiaren egitura.

Tesiaren lehenengo (2 – 4) kapituluetan, zenbakizko integrazio simplektikoak eta koma higikorreko aritmetikaren oinarriak azaldu ditugu. Bigarren kapituluaren lehen zatian, *zenbakizko integratziale simplektikoen* inguruko oinarrizko kontzeptuak azaldu ditugu. Kapitulu honen bigarren zatian, Gauss metodo estandarra deskribatu eta bere propietate nagusienak eman ditugu. Kapitulu honen azken zatian, eguzki-sistemaren integraziorako metodo simplektiko eta esplizitu nagusienak laburtu ditugu. Hirugarren kapituluuan, lan honetan zenbakizko esperimentuetan erabili ditugun hasierako baliodun problemen zehaztasunak eman ditugu. Laugarren kapituluuan, koma-higikorreko aritmetikan murgildu gara eta biribiltze errorearen inguruko gaiak argitu nahi izan ditugu.

Tesiaren (6 – 8) kapituluetan, gure implementazio berriak garatu ditugu eta zenbakizko experimentuen bidez, hauen eraginkortasuna erakutsi dugu. Lehenik, 6. kapituluuan, puntu-finkoaren iterazioaren oinarritutako Gauss metodoaren implementazioa aztertu dugu. Ondoren, 7. kapituluuan, Newtonen iterazioan oinarritutako Gauss metodoa jorratu dugu. 8. kapituluuan, epe luzeko eguzki-sistemaren integraziorako, implementazio berri bat proposatu dugu.

Tesiaren (9 – 10) kapituluetan, eztabaidea eta tesiaren ondorioak idatzi ditugu. Eztabaidaren kapituluuan, ikerketaren kontzeptu nagusienak aztertu ditugu. Ondorioen kapituluuan, gure helburuak zein neurritan bete ditugun eta etorkizunerako lanen laburpena egin dugu.

Tesiaren bukaerako hiru eranskinetan, lanaren informazio osagarria bildu dugu. A. eranskinean, egungo konputazio zientziaren hardware eta software kontzeptu nagusienak ezagutarazi nahi izan ditugu. B. eranskinean, tesian zehar erabilitako hainbat garapen matematikoen zehaztapenak eman ditugu. C. eranskinean, implementazioen kodeak azaldu eta erabiltzaileari erabilgarri izan dakiokeen informazioa laburtu dugu. Azkenik D. eranskinean, erabilitako notazioaren inguruko argibideak eman ditugu.

## **2. Kapitulua**

# **Zenbakizko integratzaile simplektikoak.**

### **2.1. Sarrera.**

Lan honen xede nagusia, N-gorputzeko sistema grabitazionalaren problemen zenbakizko integratorako algoritmoak garatzea edo daudenak hobetzea da. Era horretako problemen artean, eguzki-sistemaren epe luzeko integratorako implementazioak aztertuko ditugu [71, 20]. Gaur egun, era horretako integratorako hainbat zenbakizko metodo erabiltzen dira, bereziki bere izaera Hamiltondarra mantentzen duten metodo simplektikoak [35, 100, 37]. Metodo horien artean, gehien erabiltzen direnak izaera esplizituko algoritmoak diren arren, metodo implizituak, modu egokian implementatz gero, hauek baino eraginkorragoak izan daitezke. Zenbakizko integratorako Gauss metodo implizituaren implementazio eraginkorra aztertuko dugu.

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

### **2.2. Zenbakizko integratorio metodoak**

#### **b) Oinarrizko kontzeptuak.**

Ekuazio diferentzial arruntak, egoera aldaketak dituzten problemen azterketarako eredu matematikoak dira [53]. Zientziaren eta ingeniaritzaren arlo askotan azal-

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

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

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

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

Ekuazio diferentzial arruntetarako hasierako baliodun problemetan,  $y(t_0) = y_0 \in \mathbb{R}^d$  hasierako balio bat finkatuz, eta  $f(t, y)$  funtzioa,  $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$ ,  $t_0$  ingurunean diferentziagarria bada, orduan hasierako baliodun problemaren soluzioa existitzen da eta bakarra da [54],

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

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

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

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

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

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

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

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

**Definizioak.**

Jarraian zenbakizko integrazioen metodoen oinarrizko kontzeptuak eta notazioa finkatuko ditugu [28, 54].

## 1. Fluxua.

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

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

## 2. Zenbakizko diskretizazioa.

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

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

## 3. Metodoaren ordena.

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

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

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

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

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

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

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

## 4. Metodo simetrikoak.

Urrats bakarreko  $\phi_h$  metodoa simetrikoa da, honako baldintza betetzen bantu,

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

### Euler-en metodo esplizitua.

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

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

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

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

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

### Euler-en metodo implizitua.

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

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

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

### Iterazio metodoak

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

#### 1. Puntu-finkoaren iterazioa.

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

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

Iterazioak  $x^*$  soluzioarengana konbergitu dezake.

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

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

## 2. Newtonen iterazioa.

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

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

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

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

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

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

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

### Problema zurruna.

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

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

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

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

eta aldiz, metodo esplizituaren bidezkoa,

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

zerorantz gerturatuko da soilik  $h$  oso balio txikitarako, non  $|1 + h\lambda| < 1$  izan behar duen. Beraz, problema honetan  $\lambda$  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 eraginkorragoa den ekuazio diferentzialei, problema *zurrunak* (*stiff*) esaten zaie [54]. Problema *zurrunden* artean problema garrantzitsuak ditugu, esaterako eskala anitzeko sistemak.

### Sistema-Hamiltondarak.

Hamiltondar sistemak [103], ekuazio diferentzial mota garrantzitsu bat dira.  $H(q, p)$  funtzi leunari dagokion *Hamiltondar sistema* osatzen duten  $2d$  ekuazio diferentzialak, era honetan definitzen dira,

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

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

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

Beste notazio laburtu hau ere erabili ohi da,

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

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

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

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

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

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

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

da eta dagokien ekuazio diferentzialak hauek dira,

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

**Adibidea.** *Kepler problema* [54]. Planoan elkar erakartzen diren bi gorputzen (adibidez eguzkia eta planeta bat) mugimendua adierazten duen problema da. Gorputz baten kokapena, koordenatu sistemaren jatorria konsideratuko dugu eta beste gorputzaren kokapenaren koordenatuak  $q = (q_1, q_2)$  izendatuko ditugu.  $p = (p_1, p_2)$  momentuak dira.

Funtzio Hamiltondarra hauxe da,

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

Dagozkion ekuazio diferentzialak,

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

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

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

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

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

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

**Adibidea.** Eguzki-sistemaren probleman [100, 111], Hamiltondarra modu honetan bana daiteke  $H = H_k + H_I$ , non alde nagusia  $H_K$  planeta bakoitzaren eguzkiaren inguruko mugimendu Keplerrarrari dagokion eta  $H_I$ , aldiz, planeten arteko interakzioek eragiten duten perturbazio txikiari.

## Metodo simplektikoak.

Hamiltondar sistemaren problemetarako, Euler esplizitua eta Euler implizitua ez dira zenbakizko metodo egokiak, ez baitute Hamiltondarra mantentzen. Problema hauen propietate geometrikoak mantentzen dituzten integratzaile bereziak beharrezkoak dira [67, 104]. Integratzaile hauek, metodo simplektikoak dira eta abantaila handiena, epe luzeko integrazioetan azaltzen dute.

Metodo simplektikoen lehen aipamenak, 1950 hamarkadan kokatu behar dira eta 1980 hamarkadan, Feng Kang-ek metodo hauen azterketa sakona burutu zuen. Hastapenetako lan monografiko hau [67] eta ondorengo, azalpen ulergarriak jasotzen dituzten [54] eta [83] lanak azpimarra daitezke.

**Adibidea.** Metodo simplektikoen abantaila azaltzeko, penduluaren problema aukeratu dugu [50]. Penduluaren problema (masa  $m = 1$ ,  $l = 1$  luzerako makila eta  $g = 1$  grabitazioa)  $d = 1$  askatasuneko sistema Hamiltondarra da:

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

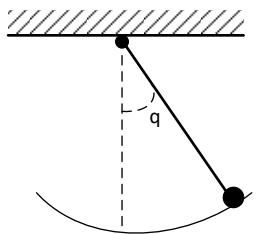
eta ekuazio diferenzialak honakoak dira,

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

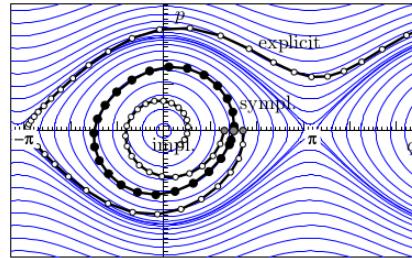
2.1. irudian ikus daitekeenez, Euler esplizitu eta implizituaren portaera kualitatiboki okerra da. Euler simplektikoak ordea, sistemaren energia kontserbatzen du, bere definizioa [54] ikus daiteke:

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

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



(a) Pendulua.



(b) Integrazioa.

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

### Metodo simplektikoen propietateak.

Integratzaile simplektiko baten bidez sistema Hamiltondar baten soluzioa lortzen dugunean, ez dugu lortzen jatorrizko sistemaren soluzio zehatza, baina lortutako soluzio hori bera jatorrizko sistemaren perturbazio baten soluzio zehatza da [104]. Alegia, lortuta soluzioa, beste sistema Hamiltondar baten soluzio zehatza da.

Metodo simplektikoen propietate nagusienak azpimarratuko ditugu [54, 67].

1. Metodo simplektikoek, energia oso ondo kontserbatzen dute.

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

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

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

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

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

2. Errorearen hedapen lineala.

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

3. Metodo simplektiko garrantzitsuenak simetrikoak dira.
4. Izaera simplektikoa mantendu beharrik ez dagoenean, erabiltzen diren metodoek eraginkortasuna hobetzeko, urrats luzera egokitu egiten dute, baina metodo simplektikoak, urrats luzera finkoarekin erabiltzen dira [67].

### 2.3. Runge-Kutta metodo simplektikoak.

1988. urtean, Sanz-Serna [67] *Gauss metodoa* simplektikoa zela ohartu zen. Runge-Kutta metodo inplizitua da eta s-atalekin lor daitekeen ordena altuena du, alegia  $p = 2s$  ordenakoa da.

#### Runge-Kutta metodoak.

Runge-Kutta metodoak [54], urrats bakarreko ekuazio diferentzial arrunten zenbakizko integrazio metodoak dira.  $b_i$ ,  $a_{ij}$  eta  $c_i = \sum_{j=1}^s a_{ij}$  ( $1 \leq i, j \leq s$ ) koefiziente errealek s-ataleko Runge-Kutta metodoa definitzen dute eta *Butcher* izeneko taula moduan laburtu ohi dira,

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

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

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

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

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

Runge-Kutta metodoak, honako integrala hurbiltzen du:

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

hurrengo koadratura formularen bidez:

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

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

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

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

### Metodo esplizituak (ERK) eta implizituak (IRK).

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

1. *ERK* metodoaren adibidea.

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

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

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

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

2. *IRK* metodoaren adibidea.

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

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

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

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

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

Ordena altuko *IRK* metodoak, *ERK* metodoak baino modu errazagoan eraiki daitezke. *Butcher-en simplifikazio baldintzen* [23] arabera definitzen dira.

### Simplektikotasuna.

Sanz-Sernak [67] frogatu zuen, Runge-Kutta metodoa simplektikoa izan dadin honako baldintza betetzea nahikoa dela, baina aldi berean beharrezkoa dela:

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

Baldintza honen arabera, Runge-Kutta metodo esplizituak, simplektikoak ezin daitezkeela izan ondorioztatu daiteke. Bestalde, 1988. urtean, Sanz-Serna [67] ohartu zen *Gauss metodoa* simplektikoa zela.

### Gauss metodoa.

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

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

### Kolokazio metodoak.

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

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

eta soluzioa

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

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

Kolokazio metodoaren definizioa eta modu honetan kalkulatutako s-ataleko Runge-Kutta metodoa baliokideak dira [54],

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

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

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

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

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

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

### Gauss metodoaren koefizienteak.

$s = 1$ ,  $s = 2$  eta  $s = 3$  atalako, Gauss kolokazio metodoaren [54] *Butcher* koefiziente taulak [54] zehaztuko ditugu,

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

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

### Gauss metodoaren propietateak.

Gauss metodoen ezaugarri nagusien artean honakoak daude:

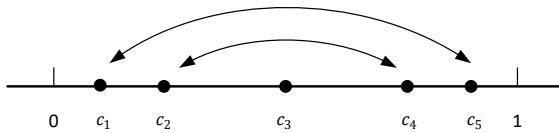
1. Metodo simplektikoa da. Beraz, (2.13) ekuazioak betetzen ditu.
2. Metodo simetrikoa da.

Runge-Kutta metodoa simetrikoa izateko baldintzak hauek dira,

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

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

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



**2.2. Irudia:** Kolokazio metodoen simetria.

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

7. Birparametrizazioa. Metodo simplektiko implizuetan, birparametrizazio teknika zuzenean aplika daiteke. Eguzki-sistemaren integrazioetarako tresna baliagarria dela frogatu da [45].

## IRK implementazioa.

### IRK algoritmoa.

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

```

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

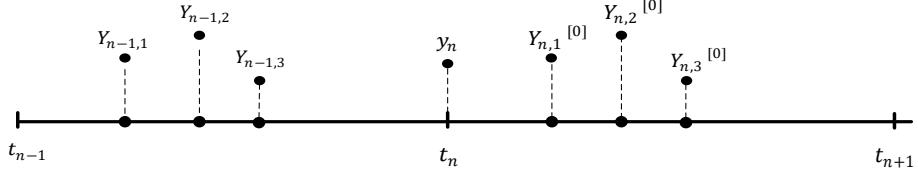
**Algoritmoa 1:** IRK Algoritmo orokorra

### Atalen hasieraketa.

Atalen hasieraketa egokia definitu behar da. Aukera simpleena  $Y_{n,i}^{[0]} = y_n$  hasiera-tzea da, baina aurreko urratseko atalen informazioa erabiliz hurbilketa hobea lor daiteke [54]. Aurreko urratseko atalak interpolatzen dituen polinomioaren bidezko hasieraketa era honetan adieraz dezakegu,

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

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



### 2.3. Irudia: Interpolazioa.

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

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

2. Polinomio interpolatzaila.

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

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

Laburta-ren lanean [77], hasieraketa aurreratuei buruzko informazioa aurki daiteke.

### Iterazio metodoa.

*IRK* metodoen erronka handiena, ekuazio-sistema ez-linealaren zenbakizko soluzioaren implementazio eraginkorra da [54, 104]. Problema ez-zurrenetarako, atallen hasieraketa ( $Y_i^{[0]}$ ) egoki bat duen puntu-finkoaren iterazioa erabil daiteke. Problema zurrenetarako, puntu-finkoaren iterazioan konbergentzia egon dadin, urrats tamaina txikiegia erabiltzea behartuko luke eta ondorioz, Newtonen iterazio sinplifikatua erabili ohi da.

#### 1. Puntu-finkoaren iterazioa.

```

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

**Algoritmoa 2:** Puntu-finkoaren iterazioa.

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

#### 2. Newtonen iterazioa.

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

```

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

**Algoritmoa 3:** Newton metodoaren iterazioa

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

### Hamiltondar banagarriak (Metodo partizionatuak).

Era honetako ekuazio diferenzialak, garrantzitsuak dira,

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

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

Zenbakizko soluzioa  $(p_{n+1}, q_{n+1}) \approx (p(t_{n+1}), q(t_{n+1}))$  era honetan kalkula daiteke [67],

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

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

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

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

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

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

### Bigarren ordenako EDA

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

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

Zenbakizko soluzioa  $(p_{n+1}, q_{n+1}) \approx (p(t_{n+1}), q(t_{n+1}))$  era honetan kalkula daiteke [67],

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

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

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

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

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

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

### Batura Konpensatua.

Zenbakizko integrazioaren urrats bakoitzean,

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

batura kalkulatu behar dugu. Normalean  $|\delta_n| \ll |y_n|$  izango da eta integrazio luzeetan, batura honek soluzioaren doitasun galera eragingo du. Hau ekiditeko *batura konpensatua* izeneko teknika [91, 59, 54] erabili ohi da. Teknika hau 4.atalean deskribatu dugu eta zehaztapenak 11 algoritmoan eman ditugu.

## 2.4. Konposizio eta Splitting metodoak.

Konposizio eta Splitting ideietan oinarrituz, aplikazio eremu ezberdinatarako hainbat integratziale simplektiko [104] garatu dira. Metodo hauek ez dira orokorrak, problema zehatzetan aplikagarriak baizik, eta metodo oso eraginkorrapak dira.

### Konposizio metodoak.

Konposizio metodoak, oinarrizko metodo bat edo gehiago konposatuz eraikitako zenbakizko integrazio metodoak dira [54]. Oinarrizko metodoekin segidan exekutatutako azpi-urrats kopuru batek, konposizio metodoaren integrazioaren urrats bat osatzen du. Helburua, orden baxuko metodo batetik abiatuta, ordena altuko metodoa eraikitzea da; konposizio metodoak, konposatutako oinarrizko metodoaren propietateak (simetrikoa, simplektikoa,...) jasotzen ditu.

### Konposizio orokorrak.

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

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

### Algoritmoa.

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

```

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

**Algoritmoa 6:** Konposizio metodoen implementazioa.

Konposizio metodoen implementazioaren ezaugarri nagusienak hauek dira:

1. Esplizituak dira.

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

2. Sekuentzialak dira.

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

3. Memoria gutxi behar dute.

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

4. Bigarren ordenako ekuazio diferentzialentzat egokiak dira.

Bigarren ordenako ekuazio diferentziala ( $\ddot{q} = f(p)$ ), *Störmer-Verlet (Leap-frog* izenarekin ere ezaguna) metodoan oinarritutako, s-ataletako konposizio metodoarekin integratzeko, urrats bakoitzean ekuazio diferenzialaren  $s$

balioztapen egin behar ditugu:

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

### Konposizio simetrikoak.

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

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

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

### CO1035: 10 ordenako konposizio metodoa.

Sofroniou eta Spaletta-ren [106, 2004]  $s = 35$  eta  $p = 10$  ordenako metodoa, orain arteko ordena altueneko konposizio metodo eraginkorrena bezala har daiteke (2.1. taula). Konposizio metodo simetrikoa: oinarrizko metodoa simetrikoa eta  $p = 2$  ordenakoa da.

**2.1. Taula:** 10 ordenako konposizio metodoa [54, 158.or] (CO1035).

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

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

1. Konposizio metodo orokorra.

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

2. Stömer-Verlet oinarrizko metodoaren konposizio metodoa,

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

3. Jarraian dauden  $\varphi^{H_1}$  fluxuak era honetan elkartu daitezke,

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

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

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

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

## Splitting metodoak.

*Splitting metodoak*,  $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$  sistema osoa integratzeko,  $f^{[i]}$  ( $f = \sum_{i=1}^m f^{[i]}$ ) azpiproblemetan deskonposatu daitezkeen ekuazio diferentzialetarako zenbakizko integrazio metodoak dira [104, 54].

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

### Splitting arruntak.

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

1. Lie-Trotter splitting 1 ordenako metodoak,

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

**Adibidea.** Era honetako Hamiltondar banagarrieta  $H(q, p) = T(p) + V(q)$ ,  $H_1 = T$  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 2 ordenako metodo simetrikoa,

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

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

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

### Fluxu zehatza eta zenbakizko fluxua 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,

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

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

### Splitting orokorrak.

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

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

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

### Algoritmoa.

*Splitting metodoen* implementazio orokorra 7 algoritmoan ikus daiteke:

```

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

```

**Algoritmoa 7:** Splitting metodoak.

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

### Eguzki-sistemari egokitutako splitting metodoak.

Har dezagun, N-gorputzeko problema grabitazionalaren Hamiltondarra,

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

Eguzki-sistemaren integratzaileko erabiltzen diren koordenatu sistema nagusiak, Jacobi eta koordenatu heliozentrikoak dira [111, 37]. Bi koordenatu sistema hauekin, Hamiltondarra beste modu honetan berridatz daiteke,

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

non alde nagusia  $H_A$  planeta bakoitzaren eguzkiaren inguruko mugimendu Kepleriarraren eta  $H_B$  aldiz, planeten arteko interakzioek eragiten duten perturbazio txikia. Jacobi koordenatuetaan  $H_B$ -k ez dauka  $p$ -ren menpekotasunik, heliozentrikoetan, ordea, bi aldagaien menpekotasuna dauka (ikus B.2. eranskinean):

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

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

Eguzki-sistemaren problema grabitazionalari egokitutako zenbakizko bi integratzaile simplektiko azalduko ditugu. Lehena, Laskar-ek eta Robutel-ek [82] definitutako *SABAC*<sub>4</sub> integratzailea eta bigarrena, Blanes-ek eta bere taldeak [16, 37] definitutako *ABAH1064* integratzailea.

***SABAC<sub>4</sub> integratzailea.***

Laskarrek [82, 2001],  $CSABA_n$  eta  $CSBAB_n$  integratzaile simplektikoak proposatu zituen. Metodo hauek koefiziente positiboekin eraikitako metodo simplektikoak dira eta  $\mathcal{O}(h^k\epsilon + h^4\epsilon^2)$  ordenakoak dira,  $n$  bikoitia denean  $k = n + 2$  izanik eta  $n$  bakoitia denean  $k = n + 3$  izanik.

Eguzki-sistemaren epe luzeko integrazioan [80] erabilitako  $CSABA_4$  integratzailea deskribatuko dugu. Hamiltondarra  $H = H_A + \epsilon H_B$  bada, era honetan definituko dugu metodoa,

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

eta koefizienteak 2.2. taulan zehaztu ditugu.

**2.2. Taula:**  $CSABA_4$  splitting metodoa [82].

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

***ABAH1064 integratzailea.***

Blanes-ek (2013),  $ABA$  eta  $ABAH$  metodo simplektikoak proposatu zituen [16, 37].  $ABA$  metodoak, Jacobi koordenatuetara egokitutako integratzaileak eta  $ABAH$  integratzaileak, koordenatu heliozentrikoetara egokitutako integratzaileak.

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

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

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

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

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

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

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

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

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

## 2.5. Laburpena.

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

**2.4. Taula:** Integrazio metodo simplektikoen laburpena

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

Metodo simplektikoei buruzko liburu monografiko hauek gomendatuko ditugu: [67, 54, 83, 38]. Eguzki-sistemaren epe luzeko simulazioei buruzko lan hauek ere azpimarratuko ditugu: [20, 71, 90, 64, 70].

## **3. Kapitulua**

### **Problemak.**

#### **3.1. Sarrera.**

Gure helburua eguzki-sistemaren simulaziorako zenbakizko integratzailak hobetzea eta hobekuntza horiek barneratzen dituen implementazioa eskaintza da. Eguzki-sistematuz nahi bezain komplexua den sistema har daiteke: planeta guztia, planeta nanoak, planeten ilargiak, kometak, erlatibitate efektua eta abar har daitezke kontutan. Baino sistema sinplifikatuan hobekuntzak egiten badira, sistema konplexuak ere hobeto integratzeko bidea irekiko da. Hori dela-eta, guk eredu simpleekin jardun dugu, eta zenbakizko esperimentuetan, eguzki-sistemaren bi eredu erabili ditugu, bata oso simplea eta bestea osatuagoa: alde batetik kanpo-planeten problema (eguzkia, kanpo-planetak eta Plutonek osatutakoa), eta bestetik, *9-planeten problema* (eguzkia, 8 planetak eta Plutonek osatutakoa). Bi eredu hauetan, gorputzak masa puntuak dira eta gorputz hauen arteko erakarpen gravitacionalak bakarrik hartu ditugu kontutan.

Aipatu beharra dago, zenbakizko metodo simplektiko nagusienak esplizituak direla eta metodo horiek, Hamiltondar banagarria duten problemetan bakarrik erabil daitezkeela. Gainera, problema zurruna bada metodo esplizituak ez dira eraginkorrik eta metodo implizituek abantaila azaltzen dute. Gauss metoda, orokorra eta implizitua izanik, problema zurrutarako eta Hamiltondar banagarria ez den problemetarako aplikagarria dela ere kontuan hartu behar da.

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

N-gorputzeko problema gravitacionalaren Hamiltondarra banagarria da baina pendulu bikoitzarena aldiz, ez da banagarria. Bestalde, pendulu bikoitzari malguki bat gehituz problema zurruna bilakatuko dugu, problema hauen zaitasunei nola

aurre egin erakusteko. Gainera, eguzki-sistema kaotiko [78] kontsideratzen dela jakinik, pendulu bikoitz arruntak izaera kaotikoa azaltzen duen hasierako balio zehatzak aukeratu ditugu. Problema kaotikoak, hasierako balio edo parametroen perturbazioekiko, trunkatze edo birbitze erroreakiko esponentzialki sentikorrik dira.

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

## 3.2. Pendulu bikoitzza.

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

### Pendulu bikoitz arrunta.

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

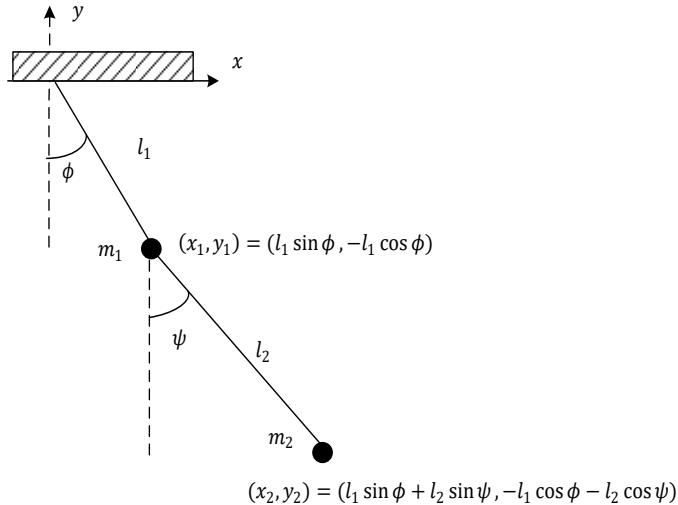
Sistemaren egoera aldagaiak, bi angelu  $q = (\phi, \theta)$  eta dagozkion momentuak  $p = (p_\phi, p_\theta)$  dira.  $\phi$  lehen penduluaren ardatz bertikalarekiko angelua da eta bigarren penduluaren angelua, era honetan definituko dugu  $\psi = \phi + \theta$ .

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

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

**Sistemaren parametroak.** Gure esperimentuetarako honako parametroak kontsideratuko ditugu,

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

**3.1. Irudia:** Pendulu bikoitz arrunta.

**Hasierako balioak.** Bi hasierako balio ezberdin konsideratu ditugu [31]: lehenak, izaera ez-kaotikoa du eta bigarrenak, izaera kaotikoa duen mugimendua agertzen du.

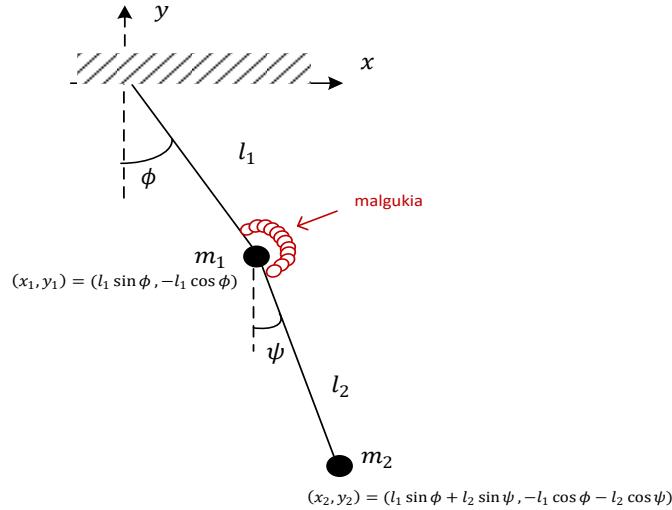
1. Hasierako balio ez-kaotikoak (NCDP):  $q(0) = (1.1, -1.1)$  eta  $p(0) = (2.7746, 2.7746)$ .  $T_{end} = 2^{12}$  segundoko integrazioa egin dugu.
2. Hasierako balio kaotikoak (CDP):  $q(0) = (0, 0)$  eta  $p(0) = (3.873, 3.873)$ .  $T_{end} = 2^8$  segundoko integrazioa egin dugu.

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

### Pendulu bikoitz zurruna.

Pendulu bikoitz arruntari, malguki bat gehitutako sistema da (ikus 3.2. irudia):  $m_1$  eta  $m_2$  masadun bi penduluk eta hauen artean  $k$  parametroaren araberako malgutasuna duen malgukiak osatzen duten sistema mekanikoa.  $k = 0$  balioarentzat, problema ez da zurruna (hau da, pendulu bikoitz arrunta) eta problemaren zurruntasuna,  $k$  balioarekin batera handitzen da.

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



### 3.2. Irudia: Pendulu bikoitza (zurruna).

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

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

**Sistemaren parametroak.** Honako parametroak konsideratu ditugu,

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

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

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

**Hasierako balioak.** Hasierako balioak, era honetan aukeratu ditugu:

1.  $k = 0$  problemarako, [31] artikulutik izaera ez-kaotikoa duen hasierako balioak hartu ditugu:  $q(0) = (1.1, -1.1)$  and  $p(0) = (2.7746, 2.7746)$ .

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

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

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

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

### 3.3. N-Gorputzen problema.

Problemaren formulazioa eman aurretik, K.Tanikawa-k eta T.Ito-k [64] 3-gorputzen problemari buruzko deskribapena aipatuko nahi genuke,

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

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

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

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

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

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

**Ekuazio differentzialak.** Abiaduraren eta kokapenaren araberako ekuazioak hauek dira,

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

**Problemaren integralak.** Integrazioan zehar konstante mantentzen diren kantitateei problemaren integralak edo inbarianteak deitzen zaie. N-gorputzen problemak 10 integral ditu [72]:

1. Masa zentroaren sei integralak.

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

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

Eta ondorioz,

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

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

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

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

2. Momentu angeluarra.

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

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

3. Energia.

Hamitoniar sistema osoaren energia da eta problemaren beste integrala da,

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

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

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

eta era honetan, masa zentroaren kokapen eta abiadurak zero egiten dira:

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

Momentu angeluarra eta energia, zenbakizko integrazioaren doitasuna neur-tzeko erabili ohi dira. Energia zenbakizko integracioen biribiltze errorea neurtze-ko integral egokiena da.

## 3.4. Eguzki-sistema.

### Sarrera.

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

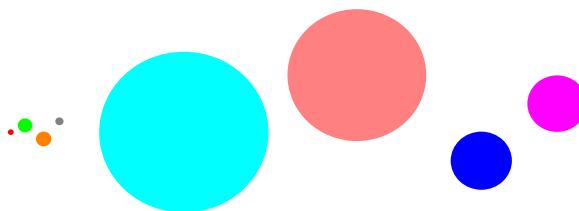
Eguzki sistemaren eredu simplea integratuko dugu. Eguzki-sistemaren gorputzak masa puntualak konsideratuko ditugu eta gure ekuazio diferenzialak defini-tzeko, soilik gorputz hauen arteko erakarpen grabitazionalak hartu ditugu kontu-tan.

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

Eguzkiak, planetak baino 1.000 aldiz masa handiago du eta hauxe da, eguzki-sistemaren ezaugarri garrantzitsuenetakoak: eguzkiaren grabitazio indar nagusiak eta planeten arteko perturbazio txikiak sortzen duten sistema dinamikoa da. Pla-neten eta beren sateliteen mugimenduan kolisiotik gertuko egoerarik edo eszentri-kotasun handiko orbitalik ez dago. Beraz, ikuspuntu honetatik sistema dinamiko simplea dela, esan daiteke. Eguzki-sistema egonkorra konsideratzen da, hau da, epe luze batean planeten arteko talkarik edo planeten kanporatzerik gertatzea, ez da espero [78, 56].

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

Eguzki-sistemaren problemaren eskalak, anitzak dira. [3.3.](#) irudian, planeta nagusien tamainak irudikatu ditugu eta [3.1.](#) taulan, eguzki-sistemaren planeten ezaugarri nagusienak eman ditugu. Aniztasunaren adierazgarri, ilargiaren, lurrauen eta Neptunoren orbitak konpara ditzakegu: ilargiak 27.32 eguneko periodoa duen orbita dauka, lurrauk urtebetekoa eta Neptunok 163 urtekoa.



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

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

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

Hiru dira erabiltzen diren koordenatu sistema nagusienak:

1. Koordenatu cartesiarrak.
2. Koordenatu heliozentrikoak.
3. Koordenatu jacobiarak.

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

## Problemak.

Eguzki-sistemaren simulaziorako test problemak deskribatuko ditugu. Kanpo-planeten problematik abiatuta, gero eta problema konplexuagoak azalduko ditugu. PW, Sharp-ek [105] eguzki-sistemaren problemen bilduma interesgarria egin zuen, eta bertan problema hauek guztiak problema ez-zurrunak konsideratzen dituela nabarmentzekoa da.

### Kanpo-planeten problema.

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

Hasierako balioak [54] liburutik hartu ditugu. Planetei dagokien masak 3.2. taulan eta kokapenak/abiadurak 3.3. taulan laburtu ditugu. planeten masak eguzkiarekiko erlatiboak dira, hau da, eguzkiaren masa 1 da eta grabitazio konstantea  $G = 2.95912208286 \cdot 10^{-4}$ . Barne-planeten masak eguzkiaren masari gehitu zaizkio eta horregatik, eguzkiaren masak,  $m_0 = 1.00000597682$  balioa hartzen du.

### 9-planeten problema.

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

### 3.2. Taula: Kanpo-planeten masak.

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

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

Hasierako balioak *DE-430* (2.014) [41] azken efemeride artikulutik hartu ditugu. Eguzki eta planeten hasierako kokapenak (AU) eta abiadurak (AU/egun), (3.5.) taulan laburtu ditugu. Era berean, planeta bakoitzari dagokion  $Gm$  balioa (3.4.) taulan laburtu dugu.

### Laskar-en eredua.

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

Laskar-ek, 2.011. urteko epe luzeko zenbakizko integratorako [80] eguzki-sistemaren eredua deskribatuko dugu. Hasierako integrazioetan, eguzkia, 8 planetak, Pluton eta ilargia bakarrik konsideratu zituen. Eguzkiaren erlatibilitatea (Saha eta Tremaine-ek [101] finkatutako teknikaren arabera) eta eredu errealistaren indar ez grabitazional garrantzitsuenak aplikatu zituen. Azken integrazioetan, Zeres, Palas, Vesta, Iris eta Bamberga asteroideak gehitu zituen.

Ilargia gorputz independente gisa konsideratu zuen. Ilargiaren lurrarekiko distantzia (380.000 km), beste gorputzekiko distantziekin alderatzen badugu (eguzkiaren 150.000.000 km eta Artizarraren 45.000.000 km) oso txikia da. Horri dela eta, ilargiaren kokapena, eguzki-sistemaren barizentroaren konsideratu ordez, lurrarekiko konsideratz doitasun handiagoa lortuko da. Lurraren eguzkiaren kokapena ( $q_e$ ) eta ilargiaren eguzkiaren kokapena ( $q_m$ ), hurrenez-hurren, lur-ilargi sistemaren barizentroaren eguzkiaren kokapena ( $q_B$ ) eta ilargiaren lu-

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

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

rrarekiko kokapena ( $q_{em}$ ) aldagaietik ordezkatzen dira,

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

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

Argitzea komeni da, ekuazio diferenzialaren eskuin aldeko expresioa ebaluatze-ko ( $q_e, q_m$ ) aldagaiak erabiliko ditugula eta ( $q_B, q_{em}$ ) aldagai berriak integratzeko erabiliko ditugula.

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

**Algoritmoa 8:** Ilargiaren kalkuluak.

## 3.5. Laburpena.

Atal honetan, pendulu bikoitzaren problema eta eguzki-sistema grabitazionalaren eredu ezberdinen zehaztasunak eman ditugu. Batetik, pendulu bikoitzaren proble-

**3.4. Taula:** Planeten  $GM$  balioak.

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

maren hasierako balioak [31] artikulutik hartu ditugu. Bestetik, eguzki-sistema grabitacionalaren problemaren integratorako hasierako balioak lan hauetatik hartu ditugu: kanpo-planeten problemarentzat [54] liburutik eta 9-planeten problema-rentzat 2014. urteko efemerideen [41] artikulutik hartu ditugu. Eguzki-sistemaren problemaren integratorako hasierako balioak jasotzen dituzten beste lan hauek ere aipatu nahi genitzuke: P.W. Sharp-ek eguzki-sistemaren problemen bilduma [105] eta Laskar-en [81] artikuluaren informazio osagarria.

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

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

### 3.6. Taula

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

### 3.7. Taula: Ilargiaren Lurrarekiko hasierako balioak.

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

## 4. Kapitulua

### Koma higikorreko aritmetika.

#### 4.1. Sarrera.

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

Egungo konputagailuen koma-higikorreko aritmetikaren implementazioak, *IEEE-754* estandarrean [62] oinarritzen dira. *IEEE-754* estandarrak, koma-higikorreko aritmetikaren konputaziorako formatu eta metodoak definitzen ditu. Konputazioen fidagarritasuna eta aplikazioen portabilitatea bermatzen ditu.

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

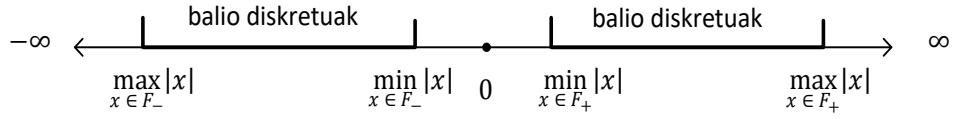
#### 4.2. IEEE-754 estandarra.

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

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

$\mathbb{F}$  zenbaki multzoa, 4.1.irudian laburtu dugu. Bai zenbaki positiboentzat, bai negatiboentzat, adieraz daitekeen zenbaki handienaren eta txikienaren arteko balio

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



#### 4.1. Irudia: Koma-higikorreko zenbakien multzoa.

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

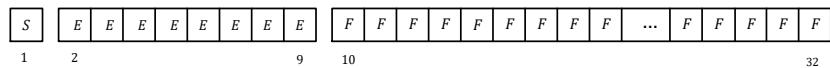
1.  $m$  bitez osatutako zatia, mantisa ( $M$ ) izenekoa. Horietako bit batek ( $S$ ) zeinuadierazten du. Bestalde  $M$  mantisa modu normalizatu honetan emanda,  $\pm 1.F$  eta zati erreala ( $F$ ) bakarrik gorde behar da.
2. Esponentea ( $E$ ),  $(n - m)$  bitez adierazitako zenbaki osoa. Zeinuarentzat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

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

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

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

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



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

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

#### 4.1. Taula: IEEE-754 koma-higikorreko formatuak

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

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

Doitasun bikoitzeko oinarrizko eragiketak (batuketa, kenketa, biderketa, zatiketa, eta erro karratua) hardware bidez exekutatzen dira [91] eta azkarrik dira. Makina ziklo bakoitzeko, 2 eta 4 batuketa, kenketa edo biderketa egin ohi dira; zatiketa eta erro karratua aldiz, eragiketa motelagoak dira. Bestalde, doitasun arruntaren aritmetika, doitasun bikoitza baino azkarragoa da: garraiatu behar den bit kopuru erdia delako eta gainera, hardware bereziei esker (adibidez Intel makinetan SSE moduluak), eragiketa aritmetikoak azkarragoak direlako. 2008. urtean, IEEE-754 estandarrak, 128-biteko koma-higikorreko aritmetika onartu zuen, baina inplementazioa softwarez bidezkoa da eta exekuzioa, gutxi gorabehera, doitasun bikoitzeko aritmetika baino 10-15 aldiz motelagoa da.

Problema batzuk, doitasun bikoitza baino doitasun handiagoa behar dute [68]. Doitasun laukoitza edo altuagoa, software liburutegien bidez emulatu ohi dira. Doitasun altuko zenbakiak adierazteko nagusiki bi modu bereizten dira:

1. *Digitu-anitzeko adierazpena*. Zenbakiak esponente bakarra eta mantisa bat baino gehiagorekin adierazten dira (adb. *GNU MPFR liburutegia* [42]).
2. *Termino-anitzeko adierazpena*. Zenbakiak ebaluatu gabeko hainbat koma-higikorreko makina zenbaki estandarren batura gisa adierazten dira (adb. Bailey QD liburutegia) [58] eta exekuzioaren ikuspegitik, hardware bidezko inplementazioaren abantaila dute.

Doitasun laukoitzeko gure esperimentuetarako, *GCC libquadmath liburutegia* [47] erabili dugu. Doitasun laukoitzean exekutatutako integrazioen zenbakizko soluzioak, soluzio zehatzak konsideratu ditugu eta doitasun bikoitzeko inplementazioaren errorea, soluzio zehatzarekiko diferentzia gisa kalkulatu dugu.

Laskar-ek epe luzeko eguzki-sistemaren simulazioaren ( $-250$  eta  $+250$  milioitako integrazio tartea) konputaziorako kalkuluak [80], kontu handiz eta doitasun handian egin behar ditu. Dena den, era honetako problemak salbuespenak dira eta ez da ohikoa izaten doitasun handian lan egin beharra. Egia da ere, neurri

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

### 4.3. Biribiltze errorea.

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

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

#### Adierazpenaren errorea.

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

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

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

- Errore absolutua,

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

- Errore erlatiboa,

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

- Aurreko bi definizioen ondorioz honako formula erabilgarria dugu,

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

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

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

bermatuta dagoela frogatzen daiteke [28].

## Aritmetikaren errorea.

Koma-higikorreko zenbakien arteko eragiketa baten emaitzak, ez du zertan  $\mathbb{F}$  multzoan adierazpen zehatza izan eta orduan, emaitza biribildu egingo da. Adibidez,  $m$  digituzko bi zenbakien biderketaren emaitza zehatza adierazteko,  $2m$  digituzko mantisa behar dugu ( $m$  digituzko galera) [44]. Salbuespena, biderkagaietako bat 2-ren berretura denean gertatzen da, orduan biderketa zehatza baita.

**Adibidea.** Demagun lau digitu hamartar errealeko aritmetikarekin ari garela lanean.

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

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

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

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

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

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

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

- Errore absolutua,

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

- Errore erlatiboa,

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

- Honako erlazio hau ondorioztatu daiteke,

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

Koma-higikorreko aritmetikan,  $|\delta z| < u$ , non  $u = 2^{-m}$ , beteko dela frog daiteke [28].

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

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

### Biribiltze errorearen hedapena.

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

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

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

$|\delta| < u$  non  $u = 2^{-m}$  beteko denik, ezin daiteke bermatu.

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

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

### Ezabapen arazoa.

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

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

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

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

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

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

## 4.4. Biribiltze errorea gutxitzeko teknikak.

Batuketa eta biderketa eragiketen biribiltze errorea kalkulatzeko algoritmoak ezagunak dira [30, 59]. Algoritmo hauek, *termino-gaitzeko adierazpenetan* oinarritzen dira eta baturaren kasuan, batura konpensatu izeneko algoritmoaren oinarria da. Ikusiko dugun bezala, algoritmo simpleak dira eta konputazio kostu txikia dute.

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

### Batura: Fast2Sum.

*Fast2Sum* algoritmoa, 1971.ean Dekker-ek asmatu zuen [30]. Koma-higikorreko  $\tilde{x}, \tilde{y} \in \mathbb{F}$  non  $|\tilde{x}| \geq |\tilde{y}|$  bi zenbakien arteko  $\tilde{z} = \tilde{x} \oplus \tilde{y}$  batuketari dagokion  $e$  biribiltze errorea era honetan kalkulatu daiteke,

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

**Algoritmoa 9:** Fast2Sum.

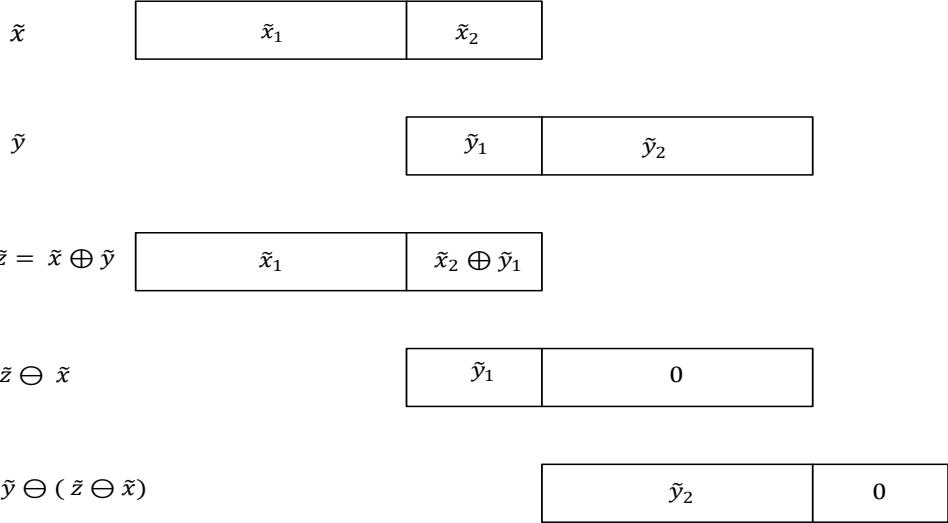
4.3.irudiaren laguntzarekin hobeto uler daiteke batuketaren biribiltze errorearen kalkulua [59].

### Batura konpensatua.

Era honetako batugai askoren arteko batuketan,

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

biribiltze errorea gutxitzeko teknika ezaguna da [59, 91, 54]. Ideia da, bi zenbakien baturan egindako biribiltze errorea lortu, eta errore hau hurrengo baturan erabiltzea. Jarraian azaltzen den moduan, urrats bakoitzaren amaieran  $e_i$  errore



### 4.3. Irudia: Batuketaren biribiltze errorea.

estimazioa kalkulatuko dugu eta hurrengo urratsean, batugaiari gehituko diogu.

```

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

```

**Algoritmoa 10:** Kahan-en batura konpensatua.

Knuth-ek eta Kahan-ek [91] frogatu zuten, batura konpensatuko algoritmoaren bidez kalkulatutako  $z_{n+1}$  baturak honakoa betetzen duela:

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

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

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

zeinek  $\tilde{z}_0, e_0, x_0, x_1, \dots, x_n \in \mathbb{F}^d$  argumentuak emanik,  $\tilde{z}_{n+1}, e_{n+1} \in \mathbb{F}^d$  balioak

itzultzen dituen, eta  $(\tilde{z}_{n+1} + e_{n+1}) \approx (\tilde{z}_0 + e_0 + x_0 + x_1 + \cdots + x_n)$  hurbilketa den.

### Zenbakizko integrazioak.

Zenbakizko integrazioetan,  $n = 1, 2, \dots$  balioentzat era honetako baturak kalkulatu behar ditugu [54],

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

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

$y_{n+1} \in \mathbb{R}^d$ ,  $y_{n+1} = \tilde{y}_n + \tilde{\delta}_n$  batura zehatza izanik eta  $\tilde{y}_{n+1} \in \mathbb{F}^d$ ,  $\tilde{y}_{n+1} = \tilde{y}_n \oplus \tilde{\delta}_n$  koma-higikorreko hurbilpena izanik, batura konpensatuaren bidez lortutako errorearen estimazioa  $e_{n+1}$ , 11 algoritmoa jarraituz lor daiteke eta baturan egindako biribiltze errore zehatza da,

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

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

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

$$\begin{aligned} inc &= \tilde{\delta}_n \oplus e_n; \\ \tilde{y}_{n+1} &= \tilde{y}_n \oplus inc; \\ e_{n+1} &= (\tilde{y}_n \ominus \tilde{y}_{n+1}) \oplus inc; \end{aligned}$$

**end**

**Algoritmoa 11:** Batura konpensatua (zenbakizko integrazioa).

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

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

### Bidekerta: 2MultFMA.

IEEE 754-2008 estandarrean, *FMA* [91] (*fused multiply-add*) instrukzioa gehitu zen eta hurrengo urteetan, ordenagailu arruntetan zabaltzea espero da. Instrukzio honen garrantzia handia da: orokorrean konputazioak azkartzen ditu eta biderketa eskalarren, matrize biderkaduren eta polinomio ebaluazioen biribiltze errorea

txikitzen du. *FMA* instrukzioa, zatiketa eta erro karratuaren algoritmo azkarren diseinuan ere erabiltzen da.

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

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

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

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

**Algoritmoa 12:** 2MultFMA.

### Sterbenz Teorema.

Sterbenz teoremaren arabera [107], bi zenbaki elkarrekiko gertu daudenean, honako baldintza betetzen bada, horien arteko kendura zehatza da.

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

## 4.5. Laburpena.

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

Koma-higikorreko aritmetikan sakontzeko honako bibliografia azpimarratuko dugu: "Numerical computing with IEEE floating point arithmetic", Michael L.Overton [94], "Handbook of floating-point arithmetic", Jean-Michael Muller [91], "Accuracy and stability of numerical algorithms", Nicholas J Higham [59] eta "A graduate introduction to numerical methods", Rober M Corless [28].

## 5. Kapitulua

### IRK: Puntu-finkoaren iterazioa.

#### 5.1. Sarrera.

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

Runge-Kutta implizitu simplektikoaren (Gauss nodoetan oinarritutako Runge-Kutta kolokazio metodoa) implementazioa proposatu dugu, eta biribiltze errorearen garapena txikia izateko ahalegin berezia egin dugu. Implementazioa, problema ez-zurrunetan aplikatzeko garatu dugunez, ekuazio-sistema implizitua, puntu-finkoaren iterazioaren bidez ebatziko dugu (puntu-finkoaren iterazioan eta Newtonen iterazio simplifikatuan oinarritutako implementazioen eraginkortasun azterketak, [54, 67] lanetan kontsulta daitezke).

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

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

gisa kalkulatzen dena.

Lehenengo, Hairer-en IRK metodo simplektikoaren implementazioa [55] aztertuko dugu. Ondoren, IRK implementazio hau hobetzeko gure proposamenak azalduko ditugu eta azkenik, zenbakizko integrazioen bidez, gure implementazio berriaren abantailak erakutsiko ditugu.

## 5.2. Hairer-en implementazioa.

### IRK implementazio estandarra.

Gure abiapuntua, Hairer-ek [55] proposatutako IRK metodoaren implementazioa (batura konpensatuaren teknikarekin [59] garatutakoa) da. Hairer ohartu zen, puntu-finkoaren iterazioan oinarritutako IRK metodo simplektikoaren implementazio estandarrean, biribiltze erroreak energian errore sistematiko bat eragiten zuela, beste metodo simplektiko esplizituetan gertatzen ez zena. Bere azterketaren ondorioen arabera, errore sistematiko honen jatorriak bi dira:

1. Aplikatutako IRK metodoa ez da sinpletikoa, hau da ez du (5.8) baldintza betetzen. Integrazioan  $a_{ij}, b_i \in \mathbb{R}$  koefiziente zehatzak erabili beharrean, biribildutako  $\hat{a}_{ij}, \tilde{b}_i \in \mathbb{F}$  erabiltzen direlako.
2. Geratze irizpide estandarraren ondorioz, urrats bakoitzean errore sistematikoa eragiten da. Geratze irizpide estandarra honakoa da:

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

non  $\text{tol}$  finkatutako tolerantzia den.

### Konponbideak.

Hairer-ek, energiaren errore sistematikoa desagertzeko, implementazio estandarrean honako aldaketak proposatu zituen:

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

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

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

Adibidez, koefizienteak era honetan zehaztu daitezke,

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

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

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

### Hairer-en implementazioaren algoritmoa.

Hairer-ek bere implementazioaren **Fortran kodea** eskuragarri du. Bere funtsa **13** algoritmoan ikus daiteke, eta horrez gain, batura konpensatuaren teknikaren erabileria **14** algoritmoan zehaztu dugu.

```

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

```

**Algoritmoa 13:** Hairer-en IRK implementazioa

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

```

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

```

**Algoritmoa 14:** Hairer-en IRK implementazioaren, batura konpensatua

### Hairer-en implementazioaren arazoak.

Hairer-ek bere implementazioarekin, *Hénon-Helies* eta eguzki-sistemaren kanpo-planeten problemetarako energiaren errore sistematikorik ez zegoela baiezta zuen. Energiaren errorea,  $k\sqrt{t_n}$  espresioaren arabera handitzen dela erakutsi zuen eta implementazioak, *Brouwer legea* [49] betetzen duela ondorioztatu zuen. Integrazioen azterketa estatistikoa egin zuen biribiltze errorearen ausazkotasuna baiez-tatzeko. Problemaren hasierako balio zehatz bati dagokion perturbatutako  $P = 1.000$  integrazio exekutatu zituen eta integrazio horien energia errorearen batez-bestekoa ( $\mu$ ), zero eta desbideratze estandarra ( $\sigma$ ),  $k\sqrt{t_n}$  espresioaren araberakoa zela erakutsi zuen. Era berean, integrazio amaiera uneko energi erroreen histogramak,  $N(\mu, \sigma)$  distribuzio normala betetzen duela erakutsi zuen.

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

### 5.3. Gure implementazioa.

IRK metodoaren puntu-finkoaren iterazioan oinarritutako implementazioa hobetzeko lau proposamen egin ditugu. Lehenik, Runge-Kutta metodoaren birformula-zio bat proposatu dugu, metoda definitzen duten koma-higikorreko koefizienteek izaera simplektikoa zehazki bete dezaten. Bigarrenik, Hairer-ek proposatutako geratze irizpidearen arazoak gainditzeko, geratze irizpide sendoagoa eta normareki-ko independentea garatu dugu.

Hirugarrenik, biribiltze errorea gutxitzeko helburuarekin, koma-higikorreko konputazioak bereziki zaindu ditugu. Kahan-en batura konpensatuaren [69, 59, 91] algoritmoaren aldaera bat aplikatu dugu.

Azkenik, biribiltze errorearen estimazioa monitorizatzeko teknika proposatu dugu. Biribiltze errorearen estimazioa, integracio nagusi eta doitasun txikiagoko bigarren integrazio baten soluzioen arteko differentzia gisa kalkulatu dugu. Zenbakizko soluzio hauek, bi modutara kalkula daitezke: paraleloan, exekuzio independenteak kontsideratuta; edo bi soluzioen integracio sekuentziala kalkulatz, konputazio kostu gehigarri txikiarekin.

Har dezagun, honako hasierako baliodun problema,

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

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

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

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

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

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

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

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

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

IRK metodoa simplektikoa da, honako baldintza betetzen bada [67]:

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

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

## Metodoaren birformulazioa.

Metodoa definitzen duten  $a_{ij}, b_i \in \mathbb{R}$  koefizienteak,  $\tilde{a}_{ij} := fl(a_{ij}), \tilde{b}_i := fl(b_i) \in \mathbb{F}$  biribildutako koefizienteen hurbilpenekin ordezkatzen ditugunean, hauek ez dute simplektikoa izateko baldintza (5.8) beteko. Ondorioz, metodoak ez ditu integral koadratikoak kontserbatuko eta Hamiltondar funtzioren eboluzioan, drift lineala ageriko da [67].

Arazo hau gainditzekeo, IRK metodoa era horretan birformulatuko dugu,

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

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

non

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

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

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

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

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

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

2. Koefiziente matrizearen behe-diagonaleko balioak finkatuko ditugu,

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

3. Koefiziente matrizearen goi-diagonaleko balioak esleituko ditugu,

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

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

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

Bestalde,  $b_i$  koefizienteak eta  $\nu_{ij}$  atalen hasieraketarako interpolazio koefizienteak finkatzeko zehaztapenak honakoak dira.

1.  $hb_i$  koefizienteak.

Gure implementazioak,  $hb_i = h \times b_i$  koefizienteak aurre-kalkulatu egiten ditu. Koefiziente hauek simetrikoak direnez eta  $\sum_{i=1}^s hb_i = h$  berdintza bete behar dela jakinda, modu honetan kalkulatuko ditugu,

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

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

2.  $\nu_{ij}$  interpolazio koefizienteak.

Formulazio estandarraren  $\lambda_{ij}$  koefizienteetatik abiatuta (2.3.atala), formulazio berriari dagozkion  $\nu_{ij}$  interpolazio koefizienteak era honetan definituko ditugu,

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

## Geratze irizpidea.

Puntu-finkoaren iterazioaren metodoaren bidez, (5.9) ekuazio implizituaren soluzioa lortzeko, iterazioaren abiapuntua  $Y_{n,i}^{[0]}$  finkatu eta  $k = 1, 2, \dots$  iterazioetarako  $Y_{n,i}^{[k]}$  hurbilpenak lortu behar dira geratze irizpidea bete arte.

```

Hasieratu  $Y_{n,i}^{[0]}$ ;
for ( $k=1, 2, \dots$  konbergentzia lortu arte) do
     $L_{n,i}^{[k]} = h b_i f(Y_{n,i}^{[k-1]}), \quad i = 1, \dots, s;$ 
     $Y_{n,i}^{[k]} = y_n + \sum_{j=1}^s \mu_{ij} L_{n,j}^{[k]}, \quad i = 1, \dots, s;$ 
end

```

**Algoritmoa 15:** IRK puntu-finkoaren iteraziao

Iterazioa,  $Y_{n,i}^{[0]} = y_n$  balioarekin hasieratu daiteke edo aurreko urratsetako atalen balioen interpolazioz lortutako balioekin [54]. Urrats luzera  $h$  behar bezain txikia aukeratuz gero, iterazioek (5.9) ekuazio aljebraikoen soluzioa den puntu-finkora konbergituko dute. Gure zenbakizko esperimentuetan nahiz Hairer-ek egindako esperimentuetan [55],  $h$  urrats luzera txikiarekin integrazioaren urrats gehienetan, puntu-finkoa lortzen dela baiezta dugu.

IRK metodoaren implementazio estandarraren geratze irizpidea honakoa da,

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

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

Hairer-ek proposatutako geratze irizpidearen arabera bi arrazoi egon daitezke geratzeko; alde batetik  $\Delta^{[k]} = 0$ , hau da, puntu-finkora iristea eta beste aldetik,

$\|\Delta^{[k]}\| \geq \|\Delta^{[k-1]}\|$  non

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

geratzeko bigarren arrazoi honetan, konsideratzen du biribiltze errorearen ondorioz gertatu dela baldintza hori, ondorioz geratzea erabakitzentzu du.

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

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

13.iterazioan geratuko dela konprobatuko dugu. Baino iterazioekin jarraituko bagenu  $\|\Delta^{[13]}\| > \|\Delta^{[14]}\| > \|\Delta^{[15]}\| > \|\Delta^{[16]}\| = 0$  gertatzen da, beraz goizegi geratu dela esan daiteke.

Hairer-en geratze irizpidea aztertu ondoren bi ondorio atera daitezke. Alde batetik, izan bedi

$$\Delta_j^{[k]}, \text{ non } \Delta^{[k]} = (\Delta_1^{[k]}, \dots, \Delta_{sd}^{[k]}) \in \mathbb{F}^{sd} \quad (1 \leq j \leq sd),$$

ezin daiteke suposatu  $\{\Delta_j^{[0]}, \Delta_j^{[1]}, \dots, \Delta_j^{[k]}\}$  segida beherakorra denik. Bigarrenik,  $\|\Delta^{[k]}\| \geq \|\Delta^{[k-1]}\|$  baldintzak, biribiltze errorea nagusia dela adierazten duen arren, gerta daiteke  $\exists j \in \{1, \dots, sd\}$  osagairen batentzat  $|\Delta_j^{[k]}| < |\Delta_j^{[k-1]}|$  izatea. Hau da, osagai horrek hobetzeko tartea izango luke eta horregatik, iterazio gehiago eman beharko genitzuk.

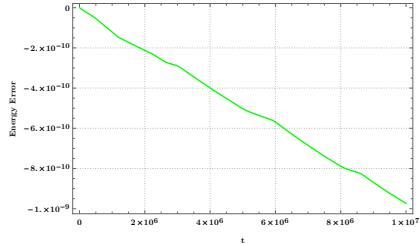
Arazo hauei aurre egiteko, geratze irizpide berri bat proposatu dugu. Iterazioak geratzea,  $\Delta^{[k]} = 0$  denean edo bi aldiz honako baldintza betetzen denean:

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

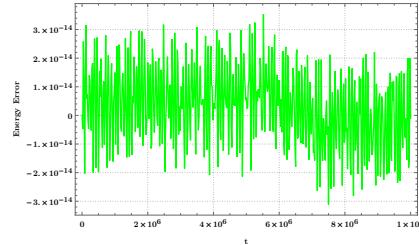
$K$  baldin bada,  $k = K-1$  eta  $k = K$  balioetarako (5.15) baldintza betetzen ez duen lehen zenbaki oso positiboa, orduan  $y_{n+1} \approx y(t_{n+1})$  era honetan kalkulatuko dugu,

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

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



(a) Hairer-en geratze irizpidea



(b) Geratze irizpidea berria

**5.1. Irudia:** Energia errore erlatiboaren eboluzioa,  $h = 1000/3$  urrats luzerarekin eguzki-sistemaren kanpo-planeten problemaren integratziorako [55]. (a) Hairer-en geratze irizpidea, (b) Geratze irizpide berria

### Tolerantzi testa.

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

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

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

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

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

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

### Biribiltze errorea gutxitzeko teknikak.

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

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

#### Batura konpensatua.

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

IRK metodoaren ekuazioetan, implizituki definitutako  $Y_{n,i}$  atalen balioak askatzeko,  $\tilde{y}_n$  balioa erabili beharrean,  $(\tilde{y}_n \oplus e_n)$  espresioa erabiltzea proposatu dugu,

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

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

#### Urratsaren konputazioa.

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

##### 1. Biderketaren biribiltze errorea.

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

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

## 2. Batura konpensatua.

Azkenik, batura konpensatua (4.1) aplikatuko dugu,

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

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

```

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

```

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

## Biribiltze errorearen estimazioa.

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

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

**Function** flr ( $x, r$ )

```

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

```

**Algoritmoa 17:** flr

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

Bigarren zenbakizko soluzioa,  $(\hat{y}_n + \hat{e}_n)$ , funtzio horretan oinarrituz,  $r < m$  finkatu eta soluzioaren bigarren hurbilketa honela lor daiteke:

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

Zenbakizko integrazioaren biribiltze errorearen estimazioa, soluzio nagusia-ren ( $y_n + e_n$ ) eta  $r$  balio txiki baterako (esaterako  $r = 3$ ) kalkulatutako bigarren

zenbakizko soluzioaren  $(\hat{y}_n + \hat{e}_n)$  arteko diferentziaren norma bezala kalkulatuko dugu.

$$estimazioa_n = \|(y_n + e_n) - (\hat{y}_n + \hat{e}_n)\|_2. \quad (5.18)$$

Biribiltze errorearen estimazioa lortzeko, bi integrazioak sekuentzialki eta konputazio kostu txikiarekin kalkula daitezke. Urrats bakoitzean, bi integrazioen  $Y_{n,i}, \hat{Y}_{n,i}$  ( $i = 1, \dots, s$ ) ataletako balioak, biribiltze errorearen estimazioa handiegia ez den artean, antzekoak mantentzen dira. Ondorioz, lehen integrazioaren bukaerako  $Y_{n,i}^{[k]}$  ( $i = 1, \dots, s$ ) atalen balioak, bigarren integrazioaren  $\hat{Y}_{n,i}^{[0]}$  ( $i = 1, \dots, s$ ) atalen hasieraketarako erabiltzen baditugu, bigarren integrazioak, iterazio kopuru txikia beharko du (ikus [18 algoritmoa](#)).

```

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

**Algoritmoa 18:** RKG2: errore estimazioa

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

### Algoritmoa.

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

```

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

**Algoritmoa 19:** IRK (puntu-finkoaren iterazio).

### Zenbakizko soluzioa.

Integrazio tartea  $[t_0, t_{end}]$  eta urrats tamaina  $h$  bada, emandako urrats kopurua  $N = (t_{end} - t_0)/h$  izango da. Bestalde, urrats bakoitzaren ondoren, erabiltzaileak definitutako funtzio bati deitzeko aukera ere eskaintzen du kodeak, baina ohikoa den bezala,  $m$  urratsero, hau da,  $t_i = t_0 + i * (m \cdot h)$ ,  $i = 1, \dots, N/m$  uneetan zenbakizko soluzioa fitxategi bitar batean idazten duen funtzioa ere erabil dezake.

Erabiltzaileak, bi integrazio mota exekutatu ditzake:

1. Integrazio arrunta.

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

nakoa da:

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

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

non

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

## 2. Integrazioa errore estimazioarekin.

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

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

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

## 5.4. Zenbakizko esperimentuak.

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

### Problemak.

Bi Hamiltondar sistema konsideratuko ditugu: pendulu bikoitz arruntaren eta kanpo-planeten problemak [54] [31]. Integrazio guztietañ  $h$  urrats luzera, truncatze errorea biribiltze errorea baino txikiago izan dadin aukeratu dugu.

#### Pendulu bikoitz arrunta

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

#### Kanpo-planeten problema

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

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

eta ezaguna da, puntu-finkoaren bertsio partizionatua (4), puntu-finkoaren iterazio estandarra baino eraginkorragoa dela [102]. Dena den, zenbakizko esperimentuetarako, Hairer-en [55] lanean bezala, puntu-finkoaren bertsio estandarraren emaitzak erakutsi ditugu. Puntu-finkoaren bertsio partizionatua aplikatu dugunean, antzeko emaitzak lortu ditugu baina urrats bakoitzean iterazio kopuru gutxiago behar izan ditu.

#### Energia errorearen jatorria.

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

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

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

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

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

A. Implementazio zehatza.

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

B. Implementazio superideala.

Zenbakizko integracio hau, iterazio errorea estimatzeko erabiliko dugu. Konputazio guztia doitasun laukoitzean egindako implementazioa da baina geratze irizpidea, doitasun bikoitzean neurtuko dugu,

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

### C. Implementazio ideala.

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

### D. Implementazio sasi-idealak.

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

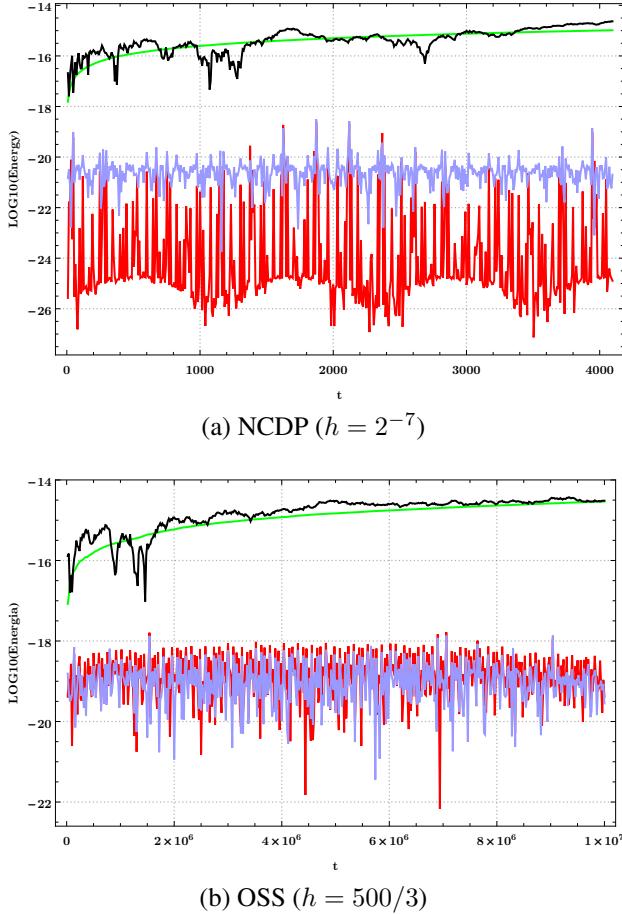
Lau implementazioetan oinarrituz, bi problemen zenbakizko soluzioei dagoien energia-errorearen eboluzioa [5.2.](#) irudian erakutsi dugu. Hainbat ondorio atera daitezke. Aukeratutako  $h$  urrats luzerarentzat, trunkatze errorea biribiltze errorearen azpitik dagoela baiezttatu dugu. Metodoaren doitasun bikoitzeko koefizienteak ( $\tilde{b}_i, \tilde{\mu}_{ij} \in \mathbb{F}$ ) erabiltzeak, ez du eraginik biribiltze errorearen garapenean. Iterazio errorea, biribiltze errorearen oso antzoko da, eta energia errorearen drift lineala eragitea espero daiteke.

## Errore azterketa estatistikoa.

Biribiltze erroreak eragiten duen zenbakizko erroreen azterketa fidagarriagoa egiteko, analisi estatistikoa aplikatu dugu (Hairer-en [\[55\]](#) lanean bezala). Problema bakoitzarentzat, hasierako balioaren osagai bakoitza ausaz perturbatutako ( $\mathcal{O}(10^{-6})$  tamainako errore erlatiboarekin)  $P = 1000$  integracio exekutatu ditugu eta emaitza hauen guztien batezbestekoan oinarritu gara, biribiltze errorearen azterketa zehatzagoa egiteko.

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

1. Implementazio idealak: ekuazio diferenzialaren eskuin aldeko funtzioren ebaluazioa izan ezik, beste eragiketa guztiak doitasun laukoitzean (128-bit) kalkulatzen dituen implementazioa da. Implementazio hau FPIEA (fixed point iteration with exact arithmetic) izendatuko dugu.
2. Doitasun bikoitzeko gure implementazioa berria. Implementazioa hau, DP izendatu dugu.
3. Hairer-ek proposatutako implementazioa [\[55\]](#). Zenbakizko esperimentuera rako, Hairer-en IRK metodoaren [Fortran kodea](#) exekutatu dugu.



**5.2. Irudia:** A-implementazioak trunkatze errorea estimatzen du (gorriz), B-implementazioarekin iterazio errorearen estimazioa lortzen du (berdez), C-implementazioak  $\tilde{f}$  funtzioa doitasun bikoitzean ebaluatzeak eragiten duen errorea (beltzez) eta azkenik, D-implementazioak erakusten du metodoaren koefizienteak  $\tilde{b}_i, \tilde{\mu}_{ij}$  doitasun bikoitzean adierazteak eragindako errorea (urdinez). Goiko irudia, pendulu bikoitzaren problema ez-kaotikoari dagokio eta behekoa, kanpo-planeten problemari

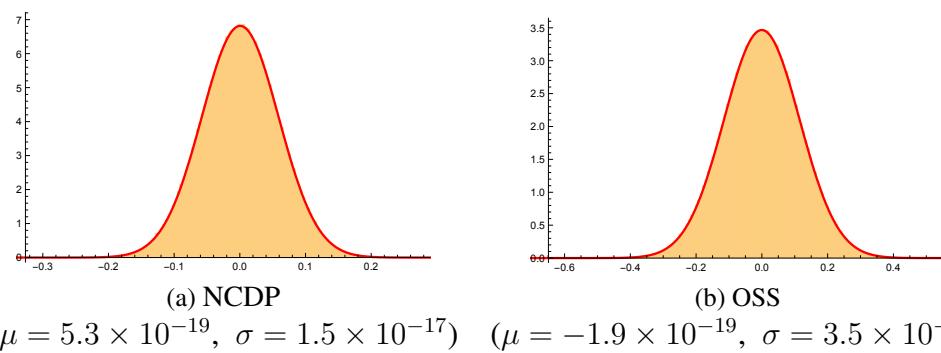
Batetik, DP implementazioaren biribiltze errorearen garapena, FPIEA implementazioaren (aritmetika zehatza) errorearekiko kualitatiboki antzekoa dela eta magnitudean gertu dagoela ziurtatu nahi dugu. Bestalde, DP implementazioa, Hairer-en implementazioarekin konparatu nahi dugu.

Aipatutako hiru implementazioetarako, 5.1. taulan iterazioetan puntu-finkoa lortu den urratsen portzentaia eta urratsetan egin den iterazio kopuruaren batez-bestekoa erakutsi dugu. Hairer-en implementazioarekin baino gehiagotan lortu da puntu-finkoa, hori geratze irizpidearen ondorioa da.

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

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

## Energia differentzien banaketa.



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

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

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

$t_{mk} = t_0 + kmh$  uneetarako,  $k^{1/2}\sigma = (t_{mk}/(mh))^{1/2}\sigma$  desbideratze estandarra duen ausazko ibilbide Gaussiar bat (*random walk*) jarraituko du. Honi, konputazio zientzian [49] Brouwer legea deritzote, Brouwer-ek [18] Kepler problemarentzat egin zuen zenbakizko integracioaren birbilitze errorearen azterketa gogoratu.

Ideia hau jarraituz, doitasun bikoitzeko (DP) implementazioan,  $m$  urrats arteko energiarene differentziari,

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

banaketa Gaussiarra dagokion aztertuko dugu.

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

### **Energia errorearen batezbestekoaren eta desbideratze estandarraren eboluzioa**

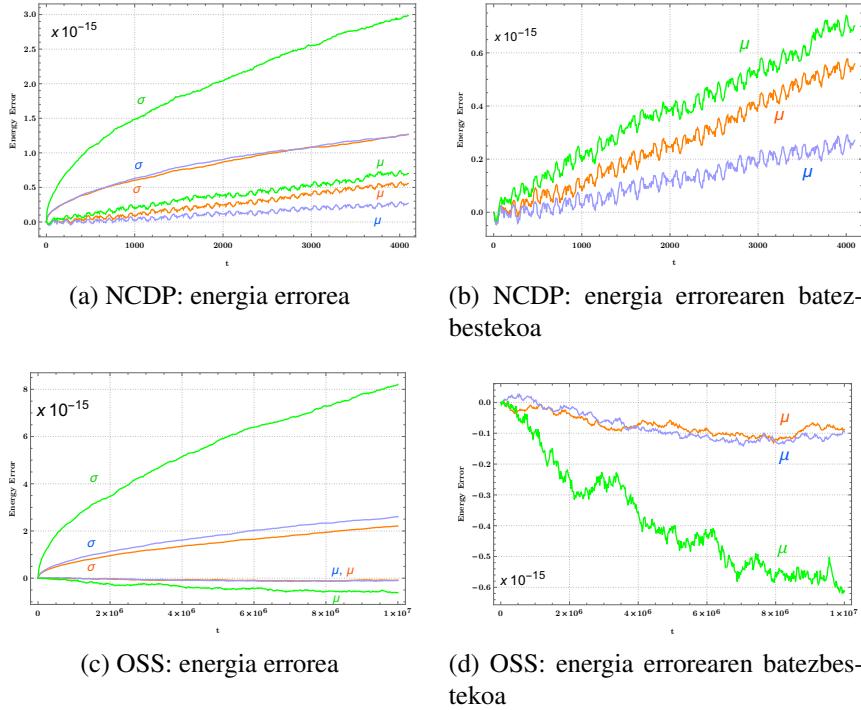
[5.4.](#) irudian, FPIEA, DP eta Hairer-en implementazioetarako, NCDP eta OSS problemen integrazioen energia errorearen batezbestekoa eta desbideratze estandarra irudikatu ditugu.

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

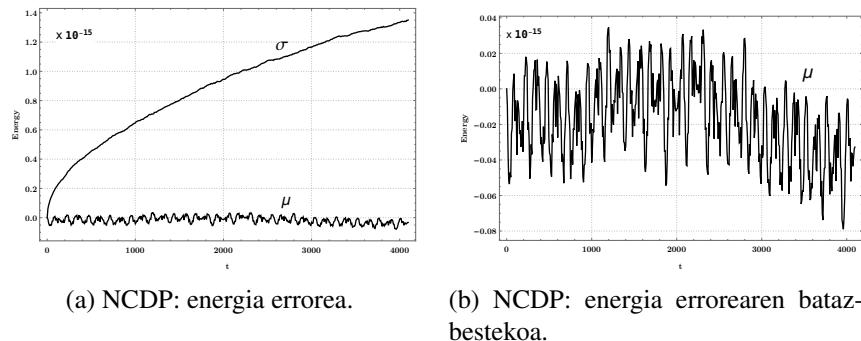
[5.4.](#) irudiko zenbakizko esperimentuetan, DP implementazioaren energia errorearen batezbestekoaren eta desbideratze tipikoaren eboluzioa ia optimoa da (FPIEA implementazioarekiko gertu).

Sinistuta gaude, ekinde ezina dela puntu-finkoaren iterazioan oinarritutako IRK metodoen implementazioan, energia errorearen batezbestekoan drift txiki bat, NCDP probleman gertatzen denaren antzera. [5.2.](#) irudian iterazio errorea eta biribiltze errorea oso antzekoak zirela ikusi dugu, horrek eragin dezake integrazioa aurreratu ahala energia errorea handitzen joatea. Beraz, bi grafikoak konsistenteak dira.

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

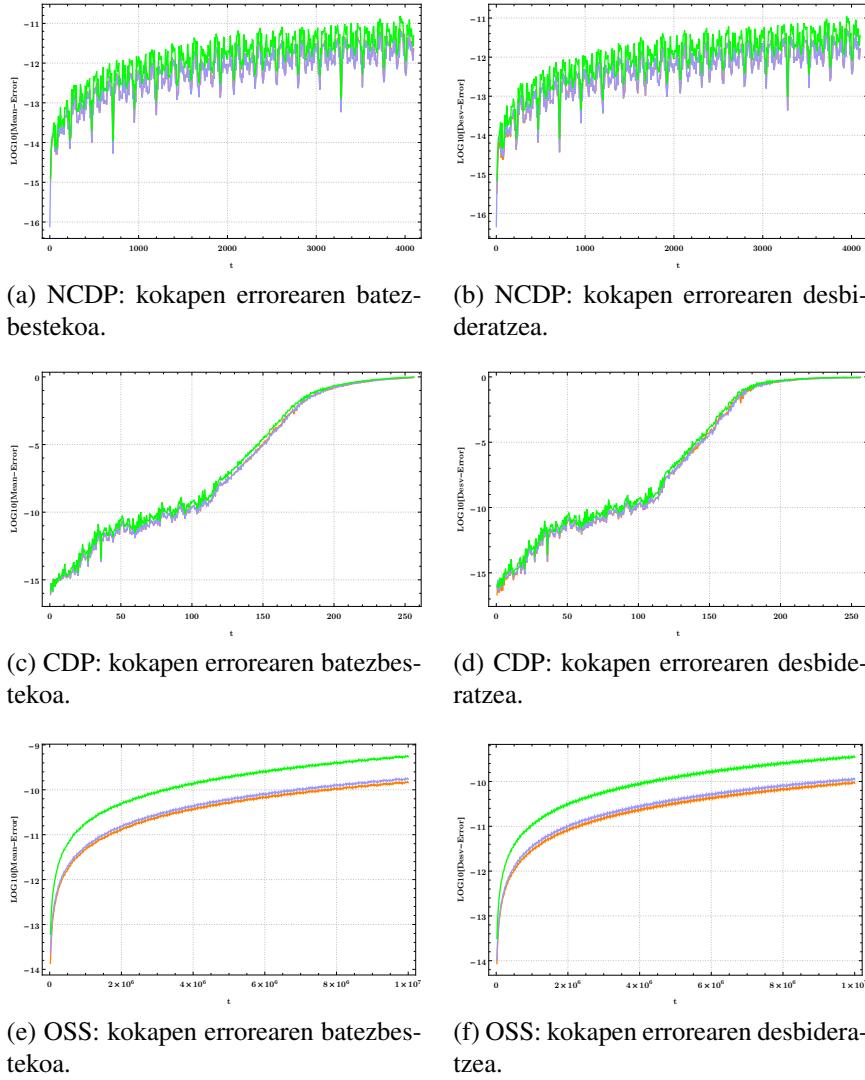


**5.4. Irudia:** Energia errorearen batezbestekoa ( $\mu$ ) eta desbideratze estandarra ( $\sigma$ ) (ezkerrean) eta energia errorearen batezbestekoaren zehaztapena (eskuinean), DP implementazioarentzat (urdinez), FPIEA implementazioarentzat (laranjaz), eta Hairer-en implementazioarentzat (berdez). Pendulu bikoitzaren problema ez-kaotikoa (a,b) eta kanpo-planeten problema (c,d)



**5.5. Irudia:** Energia errorearen batezbestekoa ( $\mu$ ) eta desbideratze estandarra ( $\sigma$ ), Newton simplifikatuaren iterazioan oinarritutako IRK implementazioa aplikatuta

Zenbakizko esperimentuen atala amaitzeko, [5.6.](#) irudian, FPIEA, DP eta Hairer-en implementazioen integrazioen kokapen errorearen eboluzioa (batezbestekoa eta desbideratze estandarra) erakutsi ditugu. Puntu-finkoaren iterazioan oinarritutako gure implementazioa, implementazio optimoarengandik oso gertu dagoenaren ideia indartu egiten dute.



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

## Biribiltze errorearen estimazioa

[5.3.](#) atalean azaldu dugun teknikak, biribiltze errorearen estimazioa ondo egiten duela erakusteko hainbat proba egin ditugu. Hiru problemak hartu ditugu kontutan, hau da, NCDP, CPP eta OSS problemen integrazioak egin ditugu,  $r = 3$  aukeratu dugu eta hasierako balioekin DP implementazioa erabiliz integratu ditugu hiru problemak. [5.7.](#) irudiko ezker aldean, integrazio bakoitzari dagozkion kokapen erroreak eta teknika berriarekin estimatutako erroreak konpara daitezke. Emaitza hauek, proposatutako biribiltze errorearen estimazioa kalkulatzeko teknikaren erabilgarritasuna erakusten dutela uste dugu.

Bestalde, hasierako balioen, 1000 perturbazioekin ere integratu ditugu 3 problemak eta integrazio bakoitzean lortutako errore estimazioen batezbestekoak eta erroreen batezbestekoak erakusten dituzte eskuin aldeko grafikoek.

## 5.5. Laburpena.

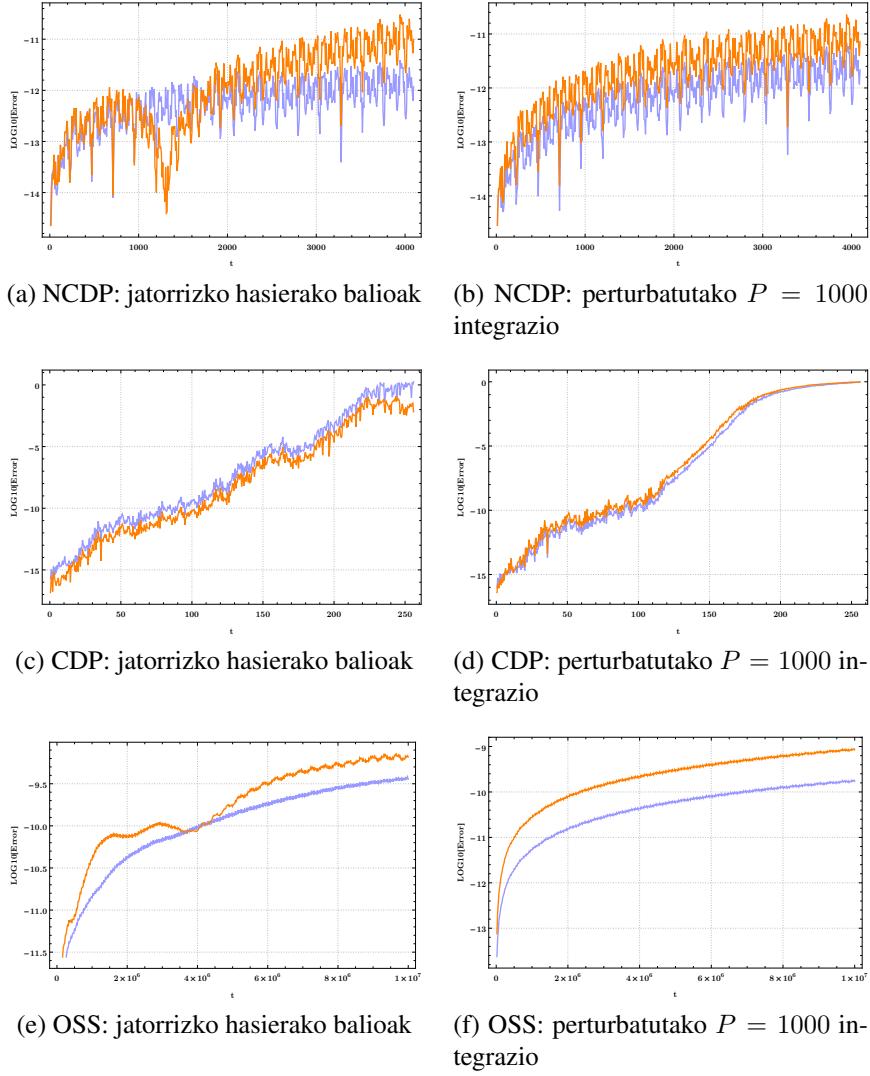
Runge-Kutta metodo implizituak (adibidez, Gauss nodoetan oinarritutako Runge-Kutta kolokazio metodoak) Hamiltondar sistemek doitasun altuko integrazioetarako aproposak dira. Problema ez-zurruntarako, puntu-finkoaren iterazioan oinarritutako implementazioa, Newton metodoaren iterazioan oinarritutako implementazioak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

Implementazio berri honetan, biribiltze errorearen eragina txikitzea ahalegin berezia egin dugu eta gainera, biribiltze errorearen estimazioa kalkulatzeko aukera eman dugu. Gure implementazioak sortzen duen biribiltze errorearen eboluzioaren estimazioa lor daiteke, eta puntu-finkoaren iterazioan oinarritutako implementazio optimoaren, antzeko eboluzioa du. Eta zentzu honetan, gure implementazio ia optimoa da. Zenbakizko esperimentuek hala baieztago dute.

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

Bestalde, puntu-finkoaren iterazioaren implementazioaren zenbakizko esperimentu batzuetan, energia errorearen drift lineal txiki bat, ekidin ezinezkoa, agertu zaigu. Energia errorearen drift-a gainditzea garrantzitsua denenerako, Newtonen iterazioan oinarritutako implementazioa beharrezkoa izango da.

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



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

## 6. Kapitulua

# IRK: Newtonen Iterazioa.

### 6.1. Sarrera.

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

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

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

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

Hainbat lanetan [22, 85, 15], (6.1) matrizearen egitura berezia aprobetxatuz, ekuazio-sistema linealak modu eraginkorrean ebatzeko proposamena egin zuen. Zehazki, ia blokeka diagonala den (6.1) matrizearen antzekoa era honetako  $I_d - h\lambda_j J \in \mathbb{R}^{d \times d}$  ( $j = 1, \dots, s$ )  $s$ -bloke duen matrizea, bloke bat  $A$  matrizearen  $\lambda_j$  balio propio bakoitzeko. Normalean, ordena altuko IRK metodoaren koefizienteen  $A$  matrizeak,  $[s/2]$  balio propio konplexu pare ditu ( $s$  bakoitia denean, balio propio erreal bat gehituta).

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

simplektikoetarako garatu dugu. Dena den, bai IRK metodo ez-simetriko simplektikoetarako, bai IRK metodo simetriko ez-simplektikoetarako garatu daiteke.

Newtonen iterazio metodoaren bidez,  $u \in \mathbb{R}^n$  eta  $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$  emanik,  $F(u) = 0$  betetzen duen  $u^{[*]}$  soluzioa aurkitu nahi dugu. Hasierako soluzioaren  $u^{[0]}$  estimazioa emanik, Newtonen iterazio sinplifikatuaren definizioa 20 algoritmoan ikus daiteke.

```

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

**Algoritmoa 20:** Newton sinplifikatua.

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

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

Newton metodoaren eragiketa konplexuenak doitasun txikiagoan kalkula daitezke [9] eta honek, konputazionalki abantaila interesgarria suposatzen du. 20 algoritmoaren eskuin aldean, implementazioak erabilitako doitasuna 64-biteko dela suposatuz, eragiketa bakoitzarentzat zein doitasun erabili beharko litzatekeen adierazi dugu; Jacobiarraren balioztapena eta aljebra linealeko eragiketak, doitasun arruntean (32-bit) kalkulatu daitezke. Horrela konputazio denbora azkartuko litzateke.

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

## 6.2. IRK-Newton estandarra.

Demagun honako hasierako baliodun problema,

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

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

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

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

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

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

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

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

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

(6.4) kalkulatu ahal izateko,  $Y_i \in \mathbb{R}^d$ ,  $i = 1, \dots, s$  ezezagunak lortu behar ditugu. Bakotza  $d$  dimentsiokoa denez,  $sd$  tamainako ekuazio-sistema ez linea- la adierazten du (6.5) ekuazioak eta sistema hori askatzeko metodo iteratiboak erabil ditzakegu. Iterazio metodo simpleena, puntu-finkoaren iterazioa da. Problema zurruna denean, puntu-finkoaren iterazioa ez da egokia eta orduan, Newtonen iterazioa aplikatu beharra dago. Problema ez-zurruna izanik ere, Newtonen iterazioak interesgarriak izan daitezke; bereziki doitasun altuko (doitasun laukoitza) konputazioetan, doitasun ezberdinak nahasten [9] dituen teknikari esker (algebra lienaleko eragiketak eta Jacobiarraren balioztapena doitasun txikiagoan kalkula- tzea baitago).

Edozein kasutan, Newtonen iterazio bakotzean, Jacobiarraren  $s$  balioztapen eta  $sd \times sd$  neurriko matrizearen LU deskonposaketa kalkulatu behar direnez, aldaera konputazionalki merkeagoak aplikatzen dira.

## Newton iterazioa.

Newtonen iterazioan, (6.5) ekuazio implizituko  $Y_i$  ( $i = 1, \dots, s$ ) atalentzako  $Y_i^{[k]}$   $k = 1, 2, \dots$  hurbilpenak kalkulatzeko algoritmoa, modu honetan defini daiteke,

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

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

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

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

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

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

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

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

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

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

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

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

alderantzizkoarekin aurre-baldintzatuta, [98] iterazio metodo baten bidez ebattea. Praktikan, (6.7) ekuazio-sistemaren soluzioaren hurbilpen bat lortuko dugu, eta metodo hauek, Sasi-Newton (inexact Newton) izenarekin ezagutzen dira.

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

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

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

Modu estandarrean [22, 85, 15],  $A$  matrizearen diagonalizazioa egiten da  $\Lambda = S^{-1}AS = \text{diag}(\lambda_1, \dots, \lambda_s)$  eta ondorioz,

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

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

Beste autore batzuk [19, 65], (6.9) ekuazio-sistema askatzeko, honako alde-rantzizko matrizea proposatzen dute:

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

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

### Newton simplifikatuaren iterazioa.

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

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

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

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

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

non,

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

### Newton super-simplifikatuaren iterazioa.

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

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

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

Nahiz eta,  $Y_i^{[0]} = y_n$  ( $i = 1, \dots, s$ ) ez den beste hasieraketa bat aplikatu, askotan gertatzen da (6.12) sistema lineala, (6.13) sistemarekin ordezkatza. Askotan egokia da [115],  $J = \frac{\partial f}{\partial y}(t + \bar{c} h, \bar{y})$  aplikatzea, non  $\bar{c} = \frac{1}{s} \sum_{i=1}^s c_i$  (metodo simetrikoetan  $\bar{c} = \frac{1}{2}$  da) eta  $\bar{y} = \frac{1}{s} \sum_{i=1}^s Y_i^{[0]}$  den. Maiz,  $\partial f / \partial y$  konputazionalki merkeagoa den hurbilketa batez ordezkatza, nahikoa izango da.

Newtonen iterazioaren bertsio honi super-simplifikatua deitu diogu. Iterazio bakoitzean  $f$  funtzioaren  $s$  ebaluazio eta  $sd$  dimentsioko ekuazio-sistema lineala askatu behar da.  $(I_s \otimes I_d - h A \otimes J)$  matriza iterazio guztiarako berdina da, bere LU deskonposaketa behin bakarrik egin behar da baina konputazionalki garestia da [22, 52]. Hau da aljebra linealari dagokion eragiketen konplexutasuna,

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

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

### Algoritmoa.

```

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

```

**Algoritmoa 21:** IRK (Newton super-simplifikatua).

## 6.3. IRK-Newton eraginkorra.

### Ekuazio-sistema.

Atal honetan, honako ekuazio-sistema lineala modu eraginkorrean askatzeko implementazioa proposatuko dugu,

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

non  $J \in \mathbb{R}^{d \times d}$  eta  $r \in \mathbb{R}^{sd}$  diren.

$S$ -ataletako IRK metodoa, Newton iterazioaren bidez  $d$ -dimentsioko ekuazio diferentzialen sistemari aplikatzeko, urrats bakoitzean  $sd \times sd$  tamainako hainbat ekuazio-sistema (iterazio bakoitzeko bat) askatu behar dira. Atal honetan, jatorrizko  $sd$ -dimentsioko ekuazio-sistema,  $(s+1)d$  dimentsioko ekuazio-sistema balio-kide moduan berridatziko dugu. Ekuazio-sistema baliokide hau,  $d \times d$  tamainako

$[s/2]+1$  matrize errealen LU deskonposaketa bidez askatuko dugu. Tamaina txiki-ko matrizeen LU deskonposaketa azkarra denez, konputazionalki eraginkorragoa izatea espero dugu.

Gauss nodoetan oinarritutako Runge-Kutta kolokazio metodoak, simplektikoak eta simetrikoak [102] dira.

### 1. Simplektikoa.

Runge-Kutta metodoa simplektikoa izateko baldintza,

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

### 2. Simetrikoa.

Runge-Kutta metodoa simetrikoa izateko baldintza,

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

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

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

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

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

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

non

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

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

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

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

Ekuazio-sistemaren adierazpen matriziala lagungarria izan daiteke,

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

Aldi berean, koefiziente notazio berri hau finkatuta,

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

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

### 1. Simplektikoa.

Runge-Kutta metodoa simplektikoa da,

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

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

### 2. Simetrikoa.

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

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

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

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

## IRK metodo simplektikoen garapena.

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

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

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

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

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

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

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

(B.3.) eranskinean, ekuazio baliokideak lortzeko eman diren urratsen zehaztapenak eman ditugu.

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

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

## IRK metodo simetriko simplektikoen garapena.

Atal honetan, (6.15) propietate simplektikoaz gain, (6.16) simetria propietatea ere betetzen duten IRK metodoak konsideratuko ditugu. Lehenengo, garapenean era-biliko ditugun matrize laguntzaileak definituko ditugu.

1. P matrizea.

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

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

## 2. K matrizea.

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

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

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

## 3. D matrizea.

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

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

s bakoitia bada,  $D$  matriza ezkerrekoa eta s bakoitia bada,  $D$  matriza eskuinekoa ( $\sigma_m = 0$ ) da.

## 4. Q matrizea.

(6.24) simetriko izategatik, berdintza hauek baiezta daitezke,

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

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

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

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

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

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

eta metodoa simetrikoa denez, (6.23) baldintzetako lehenagatik ,  $e_s^T B P_2 = 0$  eta  $e_s^T B Q_2 = e_s^T B P_2 V = 0$  berdintasunak aplikatuz, honako ekuazio-sistema baliokidea lortuko dugu,

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

(B.3.) eranskinean , ekuazioak lortzeko urratsen zehaztapenak eman ditugu.

### Matrizearen egitura.

Aldagai aldaketarekin lortutako ekuazio-sistema, blokeka diagonala da eta hau aprobetxatzuz, Newton iterazioaren implementazio eraginkorra lortuko dugu.  $S = 6$  ataletako IRK metodoari dagokion ekuazio-sistemaren matrizearen egitura beretza hau da.

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

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.** (6.31) sistemaren bigarren ekuaziotik  $W''$  askatu,

$$W'' = -h (D^T \otimes J) W' + (Q_2^T B \otimes I_d) r. \quad (6.32)$$

**$W'$  kalkulatzeko ekuazioak.** (6.31) sistemako lehen ekuazioan  $W''$  ordezkatuz, honako ekuazio-sistema lortuko dugu,

$$\begin{aligned} (I_m \otimes I_d + h^2 D D^T \otimes J^2) W' - \frac{1}{2} (Q_1^T B e_s \otimes I_d) \Delta z &= R, \\ -h (e_s^T B Q_1 \otimes J) W' + \Delta z &= 0, \\ \text{non } R &= (Q_1^T B \otimes I_d) r + h (D Q_2^T B \otimes J) r \in \mathbb{R}^{md}. \end{aligned} \quad (6.33)$$

Goiko ekuazio-sistema honako notazioaren arabera,

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

era honetan berridatziko dugu,

$$(I_d + h^2 \sigma_i^2 J^2) W_i - \frac{\alpha_i}{2} \Delta z = R_i, \quad i = 1, \dots, m, \quad (6.34)$$

$$-h J \sum_{i=1}^m \alpha_i W_i + \Delta z = 0, \quad (6.35)$$

non,

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

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

**$\Delta z$  kalkulatzeko ekuazioak.** Aurreko (6.34) ekuaziotik,  $W_i$  askatuz,

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

eta (6.35) ekuazioan ordezkatuz,  $\Delta z \in \mathbb{R}^d$  askatzeko ekuazioak lortuko ditugu,

$$M \Delta z = h J \sum_{i=1}^m \alpha_i (I_d + h^2 \sigma_i^2 J^2)^{-1} R_i, \quad (6.36)$$

non

$$M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1} \in \mathbb{R}^{d \times d}. \quad (6.37)$$

**Ezezagunak askatzeko laburpena.** Sistemaren ezezagunak askatzeko ekuazioak eta ordena laburtuko dugu: lehenengo,  $\Delta z \in \mathbb{R}^d$  (6.36) ekuaziotik askatuko dugu; bigarren,  $W' \in \mathbb{R}^{md}$  (6.34) ekuaziotik askatuko dugu; hirugarren,  $W'' \in \mathbb{R}^{(s-m)d}$  (6.32) ekuaziotik askatuko dugu; eta azkenik,  $\Delta Y$  (6.30) ekuaziotik askatuko dugu.

### IRK Newton: konplexutasun analisia.

$(I_s \otimes I_d - h A \otimes J)\Delta Y = r$  ekuazio sistema modu eraginkorrean askatzeko, implementazio estandarraren eta gure implementazioen konplexutasunak konparatuko ditugu.

#### Implementazio estandarra.

Butcher edota Hairer-en implementazio estandarraren konputazioa bi modutan egin daiteke:

- Zenbaki konplexuak.

$A$  matrizearen diagonalizazioak balio propio konplexuak ditu,

$$P^{-1}AP = \begin{bmatrix} \gamma_1 & & & & & \\ & \bar{\gamma}_1 & & & & \\ & & \gamma_2 & & & \\ & & & \bar{\gamma}_2 & & \\ & & & & \gamma_3 & \\ & & & & & \bar{\gamma}_3 \end{bmatrix},$$

eta ekuazio-sistema, zenbaki konplexuen aritmetika erabiliz ebatzi daiteke.

$$\begin{aligned} (I - h\gamma_j J) X &= b, \quad j = 1, \dots, 3, \\ (I - h\bar{\gamma}_j J) X &= \bar{b}. \end{aligned}$$

- Zenbaki errealkak.

Zenbaki konplexuekin ez bada lana egin nahi, zenbaki errealeko deskonposaketa baliokidea,

$$\begin{aligned} \gamma_j &= \alpha_j + i \beta_j, \\ P^{-1}AP &= \begin{bmatrix} \alpha_1 & -\beta_1 & 0 & 0 & 0 & 0 \\ \beta_1 & \alpha_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_2 & -\beta_2 & 0 & 0 \\ 0 & 0 & \beta_2 & \alpha_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_3 & -\beta_3 \\ 0 & 0 & 0 & 0 & \beta_3 & \alpha_3 \end{bmatrix} \end{aligned}$$

Hairer-en implementazioan,

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

### Gure implementazio berria.

$\bar{A}$  matrizea,  $\bar{A} = P^{-1}DP$  diagonalizatzen dugu eta  $D$  matrizeak, irudikari puruak ditu,

$$\bar{A} = Q^{-1}RQ, \quad R = \begin{bmatrix} 0 & -\gamma_1 & & \\ \gamma_1 & 0 & & \\ & & 0 & -\gamma_2 \\ & & \gamma_2 & 0 \\ & & & 0 & -\gamma_3 \\ & & & \gamma_3 & 0 \end{bmatrix}$$

Gure implementazioan,

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

### Konplexutasun konparaketa

Lehenengo eragiketa aljebraikoen konplexutasunak gogoratuko ditugu,

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

Bi implementazioen konplexutasunen laburpena [6.1](#). taulan ikus daiteke.

## 6.4. IRK-Newton estandarra (formulazio berria).

5. atalean IRK puntu-finkoaren implementazioan erabilitako birformulazioa, IRK-Newton implementazioan ere aplikatuko dugu. Horrela, IRK metodoa simplektikoa izatea ziurtatzen dugu. IRK Newtonen iterazioaren implementazioan ordea,  $L_i$  ( $i = 1, \dots, s$ ) aldagai ezezagunak eta  $Y_i$  ( $i = 1, \dots, s$ ) aldagai laguntzaileak konsideratuko ditugu, biribiltze errorea gutxitzeko helburuarekin [93].

IRK metodoaren formulazio estandarra [\(6.4\)](#), era honetan berridatziko dugu,

$$\Phi(y, t, h) := y + \sum_{i=1}^s L_i, \tag{6.38}$$

**6.1. Taula:**

s	LU		Back Substitution	
	Estandarra	Berria	Estandarra	Berria
2m	$\frac{8m}{3}d^3$	$\frac{2}{3}(2m+1)d^3$	$4m(2d^2)$	$(6m+4)d^2$
$2m+1$	$\left(\frac{8m}{3} + \frac{2}{3}\right)d^3$	$\left(\frac{4m}{3} + \frac{2}{3}\right)d^3$	$(8m+2)d^2$	$(6m+4)d^2$

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

$$L_i = h b_i f(t + c_i h, y + \sum_{j=1}^s \mu_{ij} L_j), \quad i = 1, \dots, s, \quad (6.39)$$

eta

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

**Newton sinplifikatuaren iterazioa.**

Newtonen iterazioa formulazio berriarekin honakoa izango da;  $L_i^{[0]}$  hasieratu eta  $k = 1, 2, \dots$  iterazioetarako,  $L_i^{[k]}$  hurbilpenak era honetan kalkulatuko ditugu,

- 1)  $Y_i^{[k]} := y + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, \quad i = 1, \dots, s.$   
 $g_i^{[k]} := -L_i^{[k-1]} + h b_i f(t + c_i h, Y_i^{[k]}), \quad i = 1, \dots, s,$
  - 2) Askatu  $\Delta L_i^{[k]}$   
 $\Delta L_i^{[k]} - h b_i J_i^{[k]} \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s,$   
non  $J_i^{[k]} = \frac{\partial f}{\partial y}(t + c_i h, Y_i^{[k-1]}), \quad i = 1, \dots, s,$
  - 3)  $L^{[k]} := L^{[k-1]} + \Delta L^{[k]}.$
- (6.40)

Newton sinplifikatuaren iterazioan,  $J_i^{[k]}$  Jacobiarra  $J_i^{[0]} = \partial f / \partial y(t + c_i h, Y_i^{[0]})$   $i = 1, \dots, s$  Jacobiarraz ordezkatzen da eta askatu beharreko ekuazio-sistema hona-

koa da,

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

Modu baliokidean, ekuazio lineala notazio matriziala erabiliz laburtu daiteke,

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

non,

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

### 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,  $J_i^{[0]} \approx J$ ,  $i = 0, \dots, s$  hurbilpenaz ordezkatzuz, ekuazio-sistema lineal hau lortuko dugu,

$$(I_s \otimes I_d - h BAB^{-1} \otimes J) \Delta L = g. \quad (6.41)$$

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

## Algoritmoa.

Formulazio berriari dagokion Newton super-simplifikatuaren implementazioa, (22) algoritmoan laburtu dugu.

```

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

```

**Algoritmoa 22:** IRK (Newton super-simplifikatua).

**Interpolazio koefizienteak.**  $L_{n,i}^{[0]}$  atalen hasieraketarentzat dagokien koefizienteak era honetan definituko ditugu: IRK puntu-finkoaren implementazioan finkatu genituen (5.13) interpolazio koefizienteetatik abiatuta modu errazean definituko

ditugu formulazio honi dagozkion interpolazio koefizienteak.

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

$$\Rightarrow (Mu^{-1}Nu)_{i,j}^s = \lambda_{ij}/a_{ij}.$$
(6.42)

**Geratze irizpidea.** Puntu-finkoaren iterazioan oinarritutako implementazioarentzat definitutako geratze irizpide berdina (5.15) erabiliko dugu baina  $L_{n,i}$ ,  $i = 1, \dots, s$  aldagaiei aplikatuta.

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

Honako notazioa finkatuko dugu,

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

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

$$\forall j \in \{1, \dots, sd\}, \quad \min \left( \{|\Delta_j^{[1]}|, \dots, |\Delta_j^{[k-1]}|\} / \{0\} \right) \leq |\Delta_j^{[k]}|. \quad (6.43)$$

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

1.  $\Delta L^{[k]}$  gaiak gehitu.

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

2. Batura konpensatua.

Azkenik, batura konpensatua aplikatuko dugu,

$$(\tilde{y}_{n+1}, e_{n+1}) = S_{s,d}(\tilde{y}_n, \delta_n, L_{n,1}^{[k-1]}, \dots, L_{n,s}^{[k-1]}) \quad (6.44)$$

**Function** BaturaKompensatua ( $y_n, \delta_n, L_n^{[k-1]}$ )

```

 $s_0 = y_n$ 
 $ee = \delta_n$ 
for  $i \leftarrow 1$  to ( $s$ ) do
     $s_1 = s_0$ 
     $inc = L_{n,i}^{[k-1]} + ee$ 
     $s_0 = s_1 + inc$ 
     $ee = (s_1 - s_0) + inc$ 
end
 $y_{n+1} = s_0$ 
 $e_{n+1} = ee$ 
return ( $y_{n+1}, e_{n+1}$ )

```

**Algoritmoa 23:** BaturaKompensatua  $S_{s,d}(\tilde{y}_n, \delta_n, L_{n,1}^{[k-1]}, \dots, L_{n,s}^{[k-1]})$  funtziaren implementazioa da

## 6.5. IRK-Newton eraginkorra (formulazio berria).

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

$$(I_s \otimes I_d - h BAB^{-1} \otimes J) \Delta L = g, \quad (6.45)$$

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

(6.45) ekuazio-lineala ebazteko, aurreko 6.3. atalean (6.14) moduko sistemak askatzeko deskribatutako teknika egokituko dugu. Jarraian, IRK metodo sime-triko simplektikoetarako (6.3. atala) garatutako teknika, formulazio berriko (6.45) sistema ebazteko nola aplikatu daitekeen deskribatuko dugu.

### Formulazio estandarretik formulazio berrirako urratsa.

Formulazio berriaren implementazio eraginkorra, (6.3. atala) formulazio estanda-rean emandako ekuazioak moldatuz zehaztuko dugu. Aurreko ataleko ekuazioetan, bi formulazioen aldagaien arteko erlazioak ordezkatuz,

$$\Delta Y = \Delta L (B \otimes I_d)^{-1} \quad (6.46)$$

$$r = (B^{-1} \otimes I_d) g, \quad (6.47)$$

formulazio berrirako ekuazio baliokideak lortuko ditugu.

1. Aldagai aldaketa. Formulazio estandarraren (6.30) aldagai aldaketari, (6.46) ekuazioa ordezkatuz,

$$\Delta L = (BQ_1 \otimes I_d) W' + (BQ_2 \otimes I_d) W''. \quad (6.48)$$

2.  $R \in \mathbb{R}^{md}$  matrizea. Formulazio estandarreko  $R$  matrizearen (6.33) ekua-zioan, (6.47) ekuazioa ordezkatzu,

$$\begin{aligned} R &= (Q_1^T \otimes I_d) g + h (DQ_2^T \otimes J) g, \\ R &= Q_1^T g + h DQ_2^T g J^T. \end{aligned} \quad (6.49)$$

3.  $W'' \in \mathbb{R}^{(s-m)d}$  matrizea. Formulazio estandarraren  $W''$  matrizearen (6.32) ekuazioan, (6.47) ekuazioa ordezkatzu,

$$W'' = -h (D^T \otimes J) W' + (Q_2^T \otimes I_d) g. \quad (6.50)$$

Formulazio berrian, IRK Newton simplifikatuaren implementazioaren urratsak hauek dira,

1. LU deskonposaketak.

- (a)  $\mathbb{R}^{d \times d}$  matrizeen LU deskonposaketa,

$$I_d + h^2 \sigma_i^2 J^2, \quad i = 1, \dots, [s/2].$$

- (b)  $M \in \mathbb{R}^{d \times d}$  matriza kalkulatu,

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

- (c)  $M$  matrizearen LU deskonposaketa

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

2. (6.45) ekuazio-sistemaren soluzioa ebatzi.

- $R \in \mathbb{R}^{md}$  kalkulatu,

$$R = (Q_1^T \otimes I_d) g + h (DQ_2^T \otimes J) g.$$

- $d$  kalkulatu,

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

- $\Delta z \in \mathbb{R}^d$ , ekuazio-sistematik askatu,

$$M \Delta z = d.$$

- $W_1, \dots, W_m \in \mathbb{R}^d$  kalkulatu,

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

- $W'' \in \mathbb{R}^{sd}$  kalkulatu,

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

- $\Delta L \in \mathbb{R}^{sd}$  kalkulatu,

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

IRK Newton sinifikatuaren implementazioa, [24](#) algoritmoan laburtu dugu.

```

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

```

**Algoritmoa 24:** IRK (NSS-Eraginkorra).

## 6.6. IRK Newtonen iterazio mistoa.

### Sasi-Newton iterazioa.

Newton iterazio bakoitza, (6.40) eskema jarraituz konputatzea da, eta egin beharretako bat ekuazio-sistema askatzea da. IRK metodoaren implementazio berria ekuazio-sistema hori askatzeko metodoa aldatuko dugu, hau da, honako sistema,

$$\Delta L_i^{[k]} - h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k]} = g_i^{[k]}, \quad i = 1, \dots, s, \quad (6.51)$$

non

$$g_i^{[k]} = -L_i^{[k-1]} + h b_i f\left(t + c_i h, y + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}\right), \quad i = 1, \dots, s, \quad (6.52)$$

eta

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

zehazki askatu beharrean, modu iteratiboan askatuko dugu. Horretarako, 25 algoritmoa aplikatuko dugu eta  $\Delta L^{[k]} \in \mathbb{R}^{sd}$  soluzioaren  $\Delta L_i^{[k,0]}, \Delta L_i^{[k,1]}, \Delta L_i^{[k,2]}, \dots$  hurbilpenak kalkulatuko ditugu.

```

 $\Delta L^{[k,0]} = (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} g^{[k]};$ 
while GeratzeErizpidea ( $fl_{32}(\Delta L^{[k,0]}), \dots, fl_{32}(\Delta L^{[k,l]})$ ) do
    
$$\left| \begin{array}{l} l = l + 1; \\ G_i^{[k,l]} = g_i^{[k]} - \Delta L_i^{[k,l-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,l-1]}, \quad i = 1, \dots, s; \\ \Delta L^{[k,l]} = \Delta L^{[k,l-1]} + (I_s \otimes I_d - h BAB^{-1} \otimes J)^{-1} G^{[k,l]}; \end{array} \right.$$

end
```

**Algoritmoa 25:** Barne iterazioa.

non  $fl_{32}(x)$ ,  $x \in \mathbb{R}$  zenbakitik gertuen dagoen 32-biteko IEEE doitasun arrunteko balioa den.

IRK implementazio berri honen 26 algoritmoan, (6.51) ekuazio-sistema linealaren  $J_i$  Jacobiar matrizeen ebaluazioak, doitasun arrunta duten  $Y_i$  atalekin kalkulatuko ditugu. Beraz, 25 algoritmoaren iterazioen geratze irizpidea,  $fl_{32}(\Delta L^{[k,l]}) = fl_{32}(\Delta L^{[k,l-1]})$  doitasun arruntean betetzen dela aztertzea nahiakoa izango dugu.

### IRK Newton Mistoia

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

Zehazki,  $(\tilde{y}_{n+1}, e_{n+1}) = \tilde{\Phi}(\tilde{y}_n, e_n, t_n, h)$  IRK metodoaren urrats berriaren zenbakizko soluzioa, bost faseetan kalkulatuko dugu:

1.  $L^{[0]} = 0 \in \mathbb{R}^{sd}$  atalak hasieratu, eta Newton super-simplifikatuaren iterazioak aplikatu ((6.40) iterazioaren ekuazio-sistema, (6.41) sistemarekin ordezkatuz),

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

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

2.  $L^{[k]}$  berriari dagokion Jacobiarak ebaluatu

$$J_i = \frac{\partial f}{\partial y} \left( t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k]} \right), \quad i = 1, \dots, s.$$

3. Lehen fasean lortutako  $\Delta L^{[k]} \in \mathbb{R}^{sd}$  balioa, (6.51) ekuazio-sistema linealaren  $\Delta L^{[k]}$  soluzio zehatzaren  $\Delta L^{[k,0]}$  hurbilpena kontsideratu eta 25 algoritmoa aplikatu,  $\Delta L^{[k]}$  soluzioaren  $\Delta L^{[k,\ell]}$  hurbilpena (gutxienez doitasun arruntarekin) lortzeko.
4.  $L^{[k]} = L^{[k-1]} + \Delta L^{[k,\ell]}$ , eta  $k = k+1$  eguneratu ondoren, Sasi-Newton iterazio bat aplikatuko dugu bigarren urratsean kalkulatutako  $J_i$  Jacobiarren balioak erabiliz. Ekuazio-sistema linealaren (6.51)-(6.52),  $\Delta L^{[k]}$  soluzioaren  $\Delta L^{[k,\ell]}$  hurbilpenak (berriz ere doitasun arruntean) 25 algoritmoa aplikatuz kalkulatuko ditugu.
5. Azkenik,  $(\tilde{y}_{n+1}, e_{n+1}) = \tilde{\Phi}(\tilde{y}_n, e_n, t_n, h)$  urrats berriaren zenbakizko soluzioa kalkulatuko dugu,

$$\tilde{\Phi}(\tilde{y}_n, e_n, t_n, h) = (\tilde{y}_n + e_n) + \sum_{i=1}^s (L_{n,i}^{[k-1]} + \Delta L_{n,i}^{[k,\ell]}).$$

Horretarako, Kahan-en batura konpensatua (10 algoritmoa) modu honetan aplikatuko dugu:

- (a)  $\Delta L^{[k]}$  gaien batura (magnitude txikiko bektoreen batura).

$$\delta_n := e_n + \sum_{i=1}^s \Delta L_{n,i}^{[k,\ell]}$$

- (b) Batura konpensatua.

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

Implementazio honen hainbat zehaztasun azpimarratuko ditugu:

- Algoritmoaren era honetako sistema linealak ( $I_s \otimes I_d - h B A B^{-1} \otimes J$ ), 7.5.ataleko Newton implementazio eraginkorrarekin ([24 algoritmoa](#)) askatuko ditugu.
- $\mu_{ij} \in \mathbb{F}$  koefizienteek, zehazki ([5.8](#)) propietate simplektikoa eta simetria propietatea  $\mu_{j,i} = \mu_{s+1-i,s+1-j}$  betetzen dituzte.
- ([6.52](#)) ekuazioaren  $g_i^{[k]}$  ( $i = 1, \dots, s$ ) hondarren konputaziorako,  $y \in \mathbb{R}^d$  balioaren ordez,  $\tilde{y} + e$  ( $\tilde{y}, e \in \mathbb{F}^d$ ) espresioa erabili beharko litzateke. Hala ere, hori horrela egitearen eragina, oso txikia izango litzateke, eta azken Sasi-Newton iterazioan bakarrik (4.fasea) kontutan hartzea erabaki dugu. Gainera, azken Sasi-Newtonen iterazioan, ([6.52](#)) ekuazioan  $y$ -ren ordez  $\tilde{y} + e$  erabili beharrean,  $J_i$  Jacobiarak erabili ditugu honako hurbilketa egiteko:

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

non  $f_i^{[k]} = f \left( t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]} \right)$ .

Implementazioaren algoritmo osoa, 26 algoritmoan laburtu dugu.

```

 $L^{[0]} = 0;$ 
 $J = \frac{\partial f}{\partial y}(t + h/2, \tilde{y});$ 
 $M = I_d + J \frac{h}{2} \sum_{i=1}^m \alpha_i^2 (I_d + h^2 \sigma_i^2 J^2)^{-1};$ 
Compute the LU decomposition of M;
 $***** 1\text{-Fasea} *****;$ 
 $k = 0;$ 
while ContFcn( $fl_{32}(L^{[0]}), \dots, fl_{32}(L^{[k]})$ ) do
     $k = k + 1;$ 
     $Y_i^{[k]} = \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, i = 1, \dots, s;$ 
     $f_i^{[k]} = f(t + c_i h, Y_i^{[k]}), i = 1, \dots, s;$ 
     $g_i^{[k]} = h b_i f_i^{[k]} - L_i^{[k-1]}, i = 1, \dots, s;$ 
     $\Delta L^{[k]} = (I_s \otimes I_d - h B A B^{-1} \otimes J)^{-1} g^{[k]};$ 
     $L^{[k]} = L^{[k-1]} + \Delta L^{[k]};$ 
end
 $***** 2\text{-Fasea} *****;$ 
 $J_i = \frac{\partial f}{\partial y} \left( t + c_i h, \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k]} \right), i = 1, \dots, s;$ 
 $***** 3\text{-Fasea} *****;$ 
 $\ell = 0;$ 
 $\Delta L^{[k,0]} = \Delta L^{[k]};$ 
while ContFcn( $fl_{32}(\Delta L^{[k,0]}), \dots, fl_{32}(\Delta L^{[k,\ell]})$ ) do
     $\ell = \ell + 1;$ 
     $G_i^{[k,\ell]} = g_i^{[k]} - \Delta L_i^{[k,\ell-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,\ell-1]}, i = 1, \dots, s;$ 
     $\Delta L^{[k,\ell]} = \Delta L^{[k,\ell-1]} + (I_s \otimes I_d - h B A B^{-1} \otimes J)^{-1} G^{[k,\ell]};$ 
end
 $L^{[k]} = L^{[k-1]} + \Delta L^{[k,\ell]};$ 
 $***** 4\text{-Fasea} *****;$ 
 $k = k + 1;$ 
 $Y_i^{[k]} = \tilde{y} + \sum_{j=1}^s \mu_{ij} L_j^{[k-1]}, i = 1, \dots, s;$ 
 $f_i^{[k]} = f(t + c_i h, Y_i^{[k]}), i = 1, \dots, s;$ 
 $g_i^{[k]} = (h b_i f_i^{[k]} - L_i^{[k-1]}) + h b_i J_i e, i = 1, \dots, s;$ 
 $\ell = 0;$ 
 $\Delta L^{[k,0]} = (I_s \otimes I_d - h B A B^{-1} \otimes J)^{-1} g^{[k]};$ 
while ContFcn( $fl_{32}(\Delta L^{[k,0]}), \dots, fl_{32}(\Delta L^{[k,\ell]})$ ) do
     $\ell = \ell + 1;$ 
     $G_i^{[k,\ell]} = g_i^{[k]} - \Delta L_i^{[k,\ell-1]} + h b_i J_i \sum_{j=1}^s \mu_{ij} \Delta L_j^{[k,\ell-1]}, i = 1, \dots, s;$ 
     $\Delta L^{[k,\ell]} = \Delta L^{[k,\ell-1]} + (I_s \otimes I_d - h B A B^{-1} \otimes J)^{-1} G^{[k,\ell]};$ 
end
 $***** 5\text{-Fasea} *****;$ 
 $\delta = e + \sum_{i=1}^s \Delta L_i^{[k,\ell]};$ 
 $(\tilde{y}^*, e^*) = S_{s,d}(\tilde{y}, \delta, L_1^{[k-1]}, \dots, L_s^{[k-1]});$ 

```

**Algoritmoa 26:** IRK Newtonen iterazio mistoa

## 6.7. Zenbakizko esperimentuak.

Newton iterazioan oinarrtitutako 6-ataletako Gauss kolokazio metodoaren implementazioarekin egindako zenbakizko esperimentuak azalduko ditugu. Esperimentu hauen konputaziorako, 64-biteko doitasuneko IEEE koma-higikorreko aritmetika erabili dugu.

### Problemak

#### Pendulu bikoitz zurruna

Pendulu bikoitz zurrunaren problemaren Hamiltondarra eta parametroak, [3.2.](#) atalean definitu ditugu.  $k$  parametroak malgukiaren zurruntasun maila finkatzen du:  $k = 0$  balioarentzat, problema ez da zurruna eta problemaren zurruntasuna,  $k$  balioarekin batera handitzen da.

Hasierako balioak, era honetan aukeratu ditugu:  $k = 0$  problema ez zurrunarentzat, [\[31\]](#) artikulutik izaera ez-kaotikoa duen hasierako balioa hartu dugu:  $q(0) = (1.1, -1.1)$  eta  $p(0) = (2.7746, 2.7746)$ .  $k \neq 0$  problemen hasierako balioak,

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

espresioen bidez finkatu ditugu, horrela sistemaren energia  $k \rightarrow \infty$  handitzen den heinean, bornatua dago.

Integrazio guztiak,  $h = 2^{-7}$  urrats luzera erabiliz egin ditugu; problema ez zurrunarentzat ( $k = 0$ ) trunkatze errorea biribiltze errorea baino txikiago izan dadin aukeratu dugu. Trunkatze errorea, biribiltze errorea baino handiagoa izango da  $k > 0$  zurruntasun balio handietarako.  $T_{end} = 2^{12}$  segundoko integrazioak egin ditugu eta zenbakizko soluzioa,  $m = 2^{10}$  urratsero itzuli dugu.

#### Biribiltze errorearen azterketa.

Lehenengo, Newtonen iterazioan oinarrtitutako IRK implementazio berriaren biribiltze errorearen hedapena egokia dela aztertuko dugu. Aurreko artikuluan [\[7\]](#), biribiltze errorearen hedapena gutxitzeko ahalegin berezia eginez, puntu-finkoaren iterazioan oinarrtitutako IRK implementazioa proposatu genuen. Bi implementazioen, 6-ataleko Gauss kolokazio metodoaren biribiltze erroreak konparatuko ditugu.

Pendulu bikoitzaren  $k$  parametroaren hiru balioetarako, energia errorearen azterketa zehatza egin dugu:  $k = 0$ , non biribiltze errorea trunkatze erroreari nagusitzen zaion;  $k = 2^{10}$ , non bi erroreak tamaina berekoak diren; eta  $k = 2^{12}$ , non trunkatze errorea biribiltze erroreari nagusitzen zaion. Biribiltze errorearen

konparaketa sendoago izan dadin ([55] lanean bezala), azterketa estatistikoa egin dugu. Problema bakoitzarentzat, hasierako balioak  $\mathcal{O}(10^{-6})$  errore tamainako au-saz perturbatutako  $P = 1000$  integrazio konputatu ditugu.

**6.1.** irudian, zenbakizko integrazioek gure implementazio berriaren biribiltze errorearen propagazio ona erakusten dute. Alde batetik,  $k = 0$  eta  $k = 2^{10}$  balioetarako, puntu-finkoaren iterazioan oinarritutako implementazioak energiaren grapenaren batezbestekoak, drift lineala agertzen du eta Newton iterazioan oinarritutako implementazioak, ordea ez du energia driftarik agertzen. Beste alde batetik, bi implementazioetan, energiaren desbideratze estandarrak antzekoak dira eta  $t^{1/2}$  espresioaren proportzionalak dira.

## Puntu-finkoa versus Newton iterazioa

**6.2.** taulan,  $k$  parametroaren lau balioetarako, bi implementazioen eraginkortasunaren adierazle nagusienak laburtu ditugu.

### 6.2. Taula

$k$	0	$2^3$	$2^6$	$2^8$
$E_0$	-14.39	-5.75	-5.64	-5.64

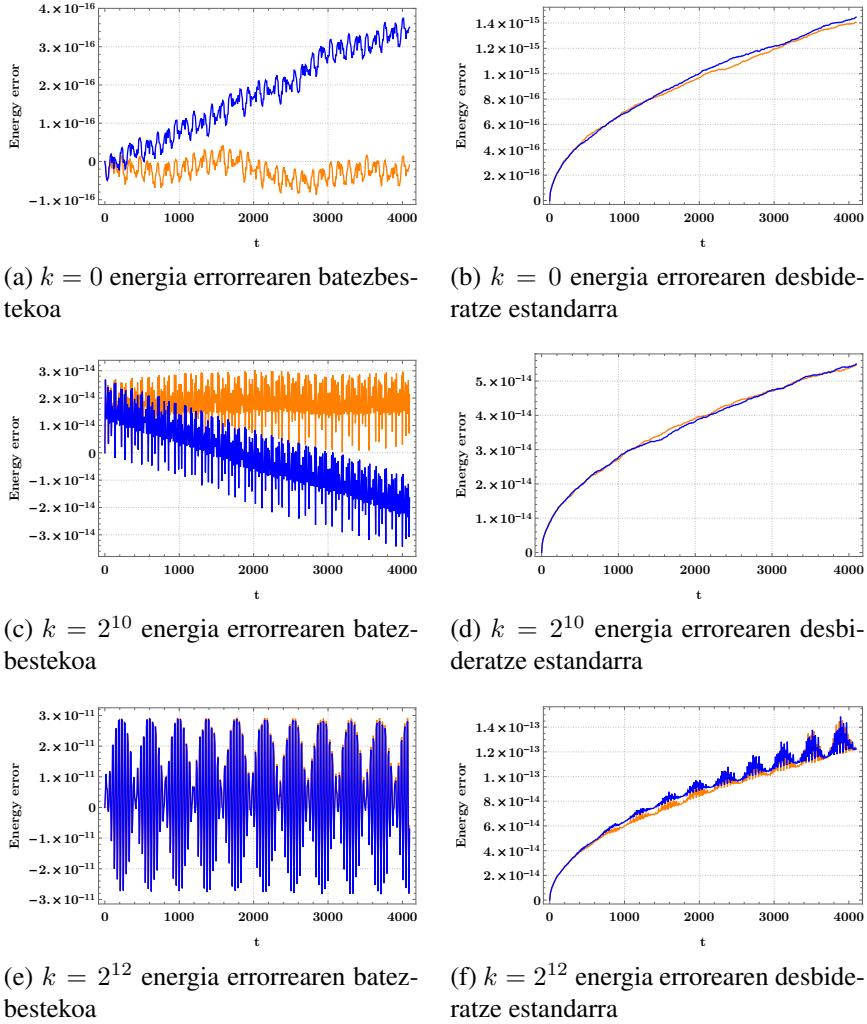
  

Fixed-points it.				
Elapsed-time (sec.)	10	12	19	51
It. per step	8.58	11.1	22.	64.2
Energy	$2.96 \times 10^{-15}$	$1.81 \times 10^{-14}$	$2.94 \times 10^{-11}$	$6.33 \times 10^{-5}$

Newton it.				
Elapsed-time (sec.)	18	20	19	18
It. per step	5.09	5.53	5.58	5.01
L. solves per step	11.37	12.92	12.72	11.04
Energy	$1.6 \times 10^{-15}$	$1.74 \times 10^{-14}$	$2.94 \times 10^{-11}$	$6.33 \times 10^{-5}$

Eraginkortasuna neurtzeko, bi implementazioen exekuzio sekuentzialen cputenborak konparatu ditugu. Horrez gain, bi implementazioen urratseko iterazioen batezbestekoak (It. per step) alderatu ditugu eta Newton implementazioan, urratseko sistema linealen ebazpenen batezbestekoa (L.solves per step) eman dugu.



**6.1. Irudia:** Energia errorearen batezbestekoaren (ezkerrean) eta desbideratze estandarraren eboluzioa (eskubian), puntu-finkoaren implementazioa (urdinez), eta Newton implementazioa (laranjaz).  $k = 0$  problema ez-zurruna (a,b),  $k = 2^{10}$  lehen problema zurruna (c,d) eta  $k = 2^{12}$  bigarren problema zurruna (e,f)

Zenbakizko soluzioaren doitasuna neurtzeko, energiaren errore erlatiboaren maximoa eman dugu,

$$\max \left| \frac{E(t_n) - E(t_0)}{E(t_0)} \right|, \quad t_n = t_0 + nh, \quad n = 1, 2, \dots$$

$k$  balio txikienetarako, puntu-finkoaren implementazioa Newton implementazioa baino eraginkorragoa da. Baina, pendulu bikoitzaren zurruntasun maila han-

ditzen dugunean, puntu-finkoaren iterazio kopurua gero eta handiagoa den bittarlean, Newton implemtazioaren iterazio kopurua mantendu egiten den, eta  $k$  handietan txikitu ere bai. Beraz, zurruntasuna handitzen dugunean, Newton implementazioa gero eta eraginkorragoa bilakatzen da.  $k = 2^{18}$  baliotik aurrera, puntu-finkoak ez du konbergitzen eta Newton implemtazioak, antzeko iterazio kopuruarekin konbergitzen du.

## 6.8. Laburpena.

IRK metodoen Newton simplifikatuaren iterazioen ekuazio-sistema, modu eraginkorrean askatzeko implemtazioa proposatu dugu. Newtonen iterazioan oinarritutako IRK metodoaren implemtazio berria aurkeztu dugu eta implemtazio honen biribiltze errorearen hedapena egokia dela baiezta dugu. Problema zurruñetarako, Newton simplifikatuaren iterazioa, puntu-finkoaren iterazioa baino eraginkorragoa dela ikusi dugu.

Newtonen iterazioan oinarritutako IRK metodoen implemtazioen inguruko lan hauek [22, 54] gomendatuko ditugu .

Azkenik, aipatu nahi dugu, atal honen edukiak [Numerical Algorithms](#) aldizkarian publikatutako [6] artikuluan aurki daitezkeela eta implemtazioaren [kodea](#) eskuragarri jarri dugula.



## 7. Kapitulua

### IRK: Eguzki-sistema.

#### 7.1. Sarrera.

Kapitulu honetan, eguzki-sistemaren ekuazio diferentzialei Kepler-en fluxuan oinarritutako aldagai aldaketa aplikatzea proposatuko dugu. 5. kapituluan puntu-finkoaren iterazioan oinarrituz eta 6. kapituluan Newton sinplifikatuaren iterazioan oinarrituz, IRK implementazioak garatu ditugu; eguzki-sistemaren problemaren integratorako, bi implementazioen artean, puntu-finkoarena eraginkorragoa dela baiezttu dugu. Ondorioz, puntu-finkoaren iterazioan oinarritutako IRK implementazioa erabiliko dugu eta ekuazio diferentzialetako aldagai eragingo diegun aldagai aldaketaren bidez, integracio eraginkorra lortzea espero dugu.

Aplikatzen dugun integracio metodoa simplektikoa eta simetriko da: neurri batean, Splitting metodoen baliokidea. Aldagai berriekiko ekuazio diferenzialak, magnitude txikiko balioak hartzen dituzte eta honek, hiru abantaila eragingo ditu. Lehenik, eguzki-sistemaren problemaren trunkatze errore nagusiena ezabatzen dugunez, urrats luzera handiagoak erabili ahal izango ditugu. Bigarrenik, batura konpensatuaren konputazioan, informazio gutxiago galduko dugu. Hirugarrenik, puntu-finkoaren iterazioek konbergentzia azkarra izango dute.

Lehenengo, Kepler-en fluxuaren implementazioa azalduko dugu. Bigarrenik, aldagai aldaketa definitu eta metodoa integratzeko zehaztapenak emango ditugu. Hirugarrenik, eguzki-sistemaren problemaren zenbakizko integrazioak egingo ditugu: implementazio honen eta doitasun altuko beste metodo simplektikoen (konposizio eta Splitting metodoak) eraginkortasunak, alderatuko ditugu.

## 7.2. Kepler-en fluxua.

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

$$H(q, p) = \frac{p^2}{2m} - \frac{\mu}{\|q\|}, \quad (7.1)$$

non  $m$  eta  $\mu$  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,

$$\dot{q} = p, \quad \dot{p} = -\frac{k q}{\|q\|^3}, \quad (7.2)$$

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

Kepler problemaren soluzio zehatza kalkula daiteke: une bateko kokapen eta abiadurak emanik,  $\Delta t$  denbora tarte bat igarotakoan (positiboa ala negatiboa), kokapen eta abiadura zehatzak konputatu daitezke. Eguzki-sistemaren integrazioetarako, Kepler problema doitasun handian eta era eraginkorrean kalkulatzea, funtsezkoa da. Kepler problemaren erreferentziazko implementazioak, Danby [29] eta J.Wisdom-enak [112] ditugu.

Kepler-en fluxua, era honetan kalkulatzen da. Lehenik, koordenatu cartesiarretatik ( $q, p \in \mathbb{R}^3$ ), koordenatu eliptikoetara ( $a, e, i, \Omega, E$ ) itzulpena egingo dugu. Koordenatu eliptikoetan,  $E$  (*eccentric anomaly*) aldagai izan ezik, beste aldagaiak konstante mantentzen dira: beraz,  $E_0$  balioa emanda,  $\Delta t$  denbora tartea aurrera egin eta  $E_1$  balio berria kalkulatuko dugu. Azkenik, koordenatu eliptikoetatik koordenatu cartesiarretara itzulpena eginez, kokapen eta abiadura berriak eskuratuko ditugu.

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

$$\downarrow \Delta t$$

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

Gorputz baten orbita Kepleriarrak hiru motakoa izan daiteke:  $H(q_0, p_0) < 0$  denean orbita eliptikoa da,  $H(q_0, p_0) > 0$  orbita hiperbolikoa eta  $H(q_0, p_0) = 0$  orbita parabolikoa. Kepler fluxuaren C implementazioa, orbita eliptikoetarako

garatu dugu eta zehaztasunak, [B.1.](#) eranskinean eman ditugu. [\(7.2\)](#) problemari dagokion fluxua, era honetan defini daiteke,

$$\begin{aligned}\varphi_{\Delta t}^k : \quad \mathbb{R}^6 &\longrightarrow \quad \mathbb{R}^6, \\ u_0 &\rightsquigarrow u_1.\end{aligned}$$

non  $u = (q, v) \in \mathbb{R}^6$  den.

### 7.3. Kepler Perturbatuaren problema.

Kepler problemaren Hamiltondarra perturbatzen badugu, ezingo dugu aurreko atalean erabili dugun fluxua erabili problema ebazteko. Kasu honetan Hamilton-darra bi zatitan banatuta egongo da;

$$H(q, p, t) = H_K(q, p) + H_I(q, p, t) \quad (7.3)$$

non  $H_K$  mugimendu Kepleriarrari dagokion Hamiltondarraren aldea den, hau da, [\(7.1\)](#) ekuazioko eskuin aldea, eta  $H_I$  perturbazioei dagokien Hamiltondarraren aldea den.

Problema berri honetan aldagai aldaketa bat egingo dugu, aldaketaren helburua da Kepleren fluxua erabili ahal izatea problemaren ebazpenean.

#### Aldagai aldaketa.

[\(7.3\)](#) problemari dagozkion ekuazioetan, Kepleren fluxuan oinarritutako aldagai aldaketa bat egingo dugu, baina horretarako notazioa finkatuko dugu: jatorrizko aldagaiak  $u = (q, p) \in \mathbb{R}^{2d}$  izango dira eta aldagai berriak  $U = (Q, P) \in \mathbb{R}^{2d}$  letra larriz adieraziko ditugu. Jatorrizko aldagaien bidez adierazitako problema, alegia, ebatzi beharreko hasierako baliodun problema, honakoa da:

$$\frac{du}{dt} = k(u) + g(u, t), \quad u(0) = u_0 \quad (7.4)$$

non  $k(u)$  alde Kepleriarrari dagokion eta  $g(u, t)$  perturbazioari. Problema horretan honako aldagai aldaketa egingo dugu, kontuan izan urrats bakoitzean egingo dugula aldagai aldaketa, hau da  $j = 0, 1, 2, \dots$  indizeak  $j$ . urratsean aplikatu beharreko aldaketa adierazten du:

$$u(t) = \varphi_{t-(j+\frac{1}{2})h}^k \left( U_j^{j+\frac{1}{2}}(t) \right) \quad (7.5)$$

$\varphi_{\Delta t}^k$  fluxua  $\Delta t > 0$  eta  $\Delta t < 0$  balioentzat definitzen da, eta  $u = \varphi_{-t}(\varphi_t(u))$  betetzen dela kontutan hartuz honako alderantzizko aldaketa ere egin dezakegu:

$$U^{j+\frac{1}{2}}(t) = \varphi_{-t+(j+\frac{1}{2})h}^k(u(t)) \quad (7.6)$$

Aldaketa hauekin asmoa da  $i+1$  urratsa emateko  $u_i \approx u(hi)$  zenbakizko soluzioan oinarrituz, aldagai aldaketaren bidez  $U_i^{i+\frac{1}{2}} = \varphi_{\frac{h}{2}}^k(u_i)$  lortu, hau da, fluxuan  $\frac{h}{2}$  aurrera egin aldagai berriak lortzeko, aldagai berri hauetan ebatzi jatorrizko problemaren urrats bati dagokion zenbakizko soluzioa (ikusiko dugun bezala, aldagai berriean alde Kepleriarri dagokion espresioak ez du eraginik eta, azken finean perturbazioari dagokion aldaketa da hemen kalkulatuko dena) eta azkenik, aldagai berri hauen balio berriak jatorrizko aldagaietara itzuli behar dira, baina  $i+1$  urratsari dagozkion unera pasa behar dira aldagaiak, hau da, fluxuan aurrera  $\frac{h}{2}$  egin behar da. Atzera egingo bagenu urratsaren hasierako balioei perturbazioak zein aldaketa eragiten dien kalkulatuko baikenuke, baina guk urratsaren bukaerako balioak nahi ditugu. Laburbilduz:

$$\begin{array}{ccc} U_0^{\frac{1}{2}} & \implies & U_1^{\frac{1}{2}} \\ \nearrow \varphi_{\frac{h}{2}}(u_0) & & \searrow \varphi_{\frac{h}{2}}(U_1) \\ u_0 & & u_1 \end{array}$$

Aldagai aldaketak fluxuan aurrera egiten du urratsaren luzeraren erdia. Hori horrela egiteak badu arrazoi bat: urratsa bere osotasunean simetrikoa da. Aurrera  $h$  luzerako urratsa ematea  $-h$  luzerako urratsa ematearekin desegiten baita.

### Aldagai berriean ekuazio differentzialak.

(7.5) aldagai aldaketa abiapuntutzat hartuz, aldagai berriei dagozkien ekuazio differentzialak lortu behar ditugu. Horrela, problemaren integrazioa aldagai berrien arabera egin ahal izango dugu. Irakur erraztasunagatik (7.5) ekuazioko  $\varphi(U)$  indizerik gabe idatziko dugu, eta  $\dot{U}$ ri dagozkion ekuazioak lortze aldera bi aldeak  $t$  aldagaiarekiko deribatuko ditugu:

$$\frac{d}{dt}u = \frac{d}{dt}(\varphi(U)), \quad (7.7)$$

Eskuin aldeari katearen erregla aplikatuz,

$$\dot{u} = \dot{\varphi}(U) + \varphi'(U)\frac{d}{dt}U. \quad (7.8)$$

$\varphi$  Kepler problemaren fluxua da, hau da,  $\dot{u} = k(u)$  problemaren fluxua da, eta fluxuaren definizioz  $\dot{\varphi}(U) = k(\varphi(U))$  da. Aldaketa horrekin, eta (7.4) ekuazioarekin berdinduz,

$$k(u) + g(u, t) = k(u) + \varphi'(U)\dot{U}. \quad (7.9)$$

Bi aldeetan  $k(u)$  kenduz,  $U$  aldagaieneko ebatzi beharreko ekuazio diferentziala lortuko dugu:

$$\dot{U} = (\varphi'(U))^{-1} g(u, t). \quad (7.10)$$

Alderantzizko matrizeak kalkulatu beharrik gabe idatz ditzakegu (7.10) ekuazioak. Horretarako  $\varphi$  fluxuaren izaera simplektikoa erabiliko dugu, hau da,  $(\varphi')^t J \varphi' = J$  propietatea betetzen du fluxuak, ondorioz,

$$\dot{U} = J^{-1}(\varphi'(U))^t J g(u, t). \quad (7.11)$$

(7.11) ekuazioak  $\varphi'(U)$  kalkulatzea eskatzen du, eta horretarako deribazio automatikoko teknikak erabil ditzakegu.

**Algoritmoa.**  $U$  aldagaietan oinarritutako ekuazio diferentzialen integratorako (7.11) espresioaren konputazioa hiru urratsetan egingo dugu:

1.  $\{u, aux\} \leftarrow KeplerFlowGen(t, U, mu).$

Kepler-en fluxua  $u = \varphi_t(U)$  aplikatuko dugu eta fluxuaren kalkulutarako erabilitako tarteko balioak,  $aux \in \mathbb{R}^{16}$  aldagaien itzuliko ditugu.

2.  $g \leftarrow g(u, t).$

Jatorrizko problemako ekuazio diferentzialetan perturbazioei dagokien espresioa kalkulatuko dugu.

3.  $KeplerFlowGFcnaux(aux, U, t, g).$

Urrats honetan  $\varphi'_t()$  kalkulatu behar da. Deribazio automatikoaren tekniken bidez, Kepler fluxuaren  $U$  aldagaieneko deribatuaren konputazio eraginkorra definitu dugu.

Hirugarren urratsak (7.11) espresioa konputatzeko behar dugun azken zatia kalkulatzen du, beraz, bere emaitza  $\dot{U}$ ren konputazioa izango da:

$$\dot{U} \leftarrow KeplerFlowGFcnaux(aux, U, t, g) = (\varphi'(U))^{-1} g.$$

## 7.4. Alde Kepleriar bat baino gehiagoko sistemak.

Alde Kepleriar bat baino gehiago dituzten problemetan ere Kepler perturbatuaren egindako aldagai aldaketa egin dezakegu. Hainbat gorputzeko sisteman gorputz bakoitzari eragingo diogu aldagai aldaketa, bakoitzak bere fluxu Kepleriar perturbatua izango du, eta horretan oinarrituz egingo diogu aldagai aldaketa. Problemaren alde Kepleriarren kopurua  $k$  bada, era honetako ekuazio diferenzialak ditugu,

$$\frac{d}{dt} \begin{pmatrix} u \\ w \end{pmatrix} = \begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \vdots \\ \dot{u}_k \\ \dot{w} \end{pmatrix} = \begin{pmatrix} k(u_1) \\ k(u_2) \\ \vdots \\ k(u_k) \\ 0 \end{pmatrix} + \begin{pmatrix} g_1(u_1, u_2, \dots, u_k, w, t) \\ g_2(u_1, u_2, \dots, u_k, w, t) \\ \vdots \\ g_k(u_1, u_2, \dots, u_k, w, t) \\ g_{k+1}(u_1, u_2, \dots, u_k, w, t) \end{pmatrix} \quad (7.12)$$

Gorputz bakoitzari dagokion aldagai aldaketa bere fluxu Kepleriarren arabera da,

$$u_j = \varphi_t^{\mu_j}(U_j), \quad j = 1, \dots, k. \quad (7.13)$$

Bi aldeak  $t$  aldagaiarekiko deribatuz eta katearen erregele aplikatuz,

$$\begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \vdots \\ \dot{u}_k \\ \dot{w} \end{pmatrix} = \begin{pmatrix} \dot{\varphi}^{\mu_1}(U_1) \\ \dot{\varphi}^{\mu_2}(U_2) \\ \vdots \\ \dot{\varphi}^{\mu_k}(U_k) \\ 0 \end{pmatrix} + \begin{pmatrix} (\varphi^{\mu_1})'(U_1) \frac{d}{dt} U_1 \\ (\varphi^{\mu_2})'(U_2) \frac{d}{dt} U_2 \\ \vdots \\ (\varphi^{\mu_k})'(U_k) \frac{d}{dt} U_k \\ g_{k+1}(u_1, u_2, \dots, u_k, w, t) \end{pmatrix},$$

ekuazioetan fluxuen propietate eta definizioak erabiliz,

$$\begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \vdots \\ \dot{u}_k \\ \dot{w} \end{pmatrix} = \begin{pmatrix} k(u_1) \\ k(u_2) \\ \vdots \\ k(u_k) \\ 0 \end{pmatrix} + \begin{pmatrix} (\varphi^{\mu_1})'(U_1) \dot{U}_1 \\ (\varphi^{\mu_2})'(U_2) \dot{U}_2 \\ \vdots \\ (\varphi^{\mu_k})'(U_k) \dot{U}_k \\ g_{k+1}(u_1, u_2, \dots, u_k, w, t) \end{pmatrix},$$

eta, azkenik, (7.12) ekuazioarekin berdinduz eta sinplifikatuz,

$$\begin{pmatrix} \dot{U}_1 \\ \dot{U}_2 \\ \vdots \\ \dot{U}_k \\ \dot{w} \end{pmatrix} = \begin{pmatrix} ((\varphi^{\mu_1})'(U_1))^{-1} g_1 \\ ((\varphi^{\mu_2})'(U_2))^{-1} g_2 \\ \vdots \\ ((\varphi^{\mu_k})'(U_k))^{-1} g_k \\ g_{k+1} \end{pmatrix},$$

Aldagai berriekiko ekuazio diferentzialak balioztatzeko, fluxuen propietateei esker,  $(\varphi')^{-1} = J^{-1}(\varphi')^t J$  kalkula dezakegu eta, Kepler perturbatuaren bezalaxe, alderantzikozko matrizerik kalkulatu beharrik ez dugu izango. Deribazio automatikoko teknikei esker, kalkulatu ahal izango ditugu. Bestalde,  $\dot{w}$  aldagaien ekuazioak balioztatzeko  $u_i$  aldagaiak behar ditugu, baina  $U_i$  aldagaietatik lor ditzakegu, gainera,  $g_i$  funtzioetarako ere behar ditugu. Ondorioz, Kepler perturbatuaren probleman bezala hiru urratsetan balioztatu ahal izango ditugu ekuazioak.

## Metodo simetrikoa.

Azpimarratu behar dugu aldagai aldaketa urratsero egiten dugula, eta ekuazio diferentziala aldagai berriekiko ebazten dugula. Aldagai berriak eta jatorrizko aldagaiak fluxuak erlazionatzen ditu: jatorrizko aldagaiak fluxuan  $\frac{h}{2}$  aurrera eginez aldagai berriak lor ditzakegu. Ebatzi beharreko problema aldagai berrietan ebatziko dugu, eta  $h$  luzerako urratsa emanez aldagai berriak aldatuko ditugu. Jatorrizko aldagaietara igarotzeko fluxuaren bidez mugitu behar ditugu balio horiek:  $\frac{h}{2}$  atzeratzen badugu jatorrizko aldagaiak urratsaren hasieran kokatuko ditugu, baina perturbazioari dagokion aldaketa bere baitan daramate, izan ere, aldagai berriei metodoaren urratsa kalkulatu diegu, hau da, perturbazioari dagokion  $h$  luzerako urratsa eman dugu. Bestalde, fluxuan  $\frac{h}{2}$  aurrera egiten badugu hurrengo urratsaren hasierako egoerara eramango ditugu balioak.

Ondorioz integracioko urrats bat hiru azpi-urratsen konbinazioa da:

1.  $U_i^{i+\frac{1}{2}} = \varphi_{\frac{h}{2}}(u_i)$ : fluxuaren arabera  $\frac{h}{2}$  aurreratu.
2. Gauss metodoaren  $h$  luzerako urratsa:  $U_i^{i+\frac{1}{2}}$  aldagaietatik  $U_{i+1}^{i+\frac{1}{2}}$  balioetara pasako gara.
3.  $u_{i+1} = \varphi_{\frac{h}{2}}(U_{i+1}^{i+\frac{1}{2}})$  fluxuaren araberako  $\frac{h}{2}$  aurreratu.

Hiru azpiurratsak simetrikoak dira, eta ondorioz,  $u_{i+1}$  abiapuntutzat hartuz,  $-h$  luzerako urratsa ematen badugu  $u_i$  lortuko dugu:

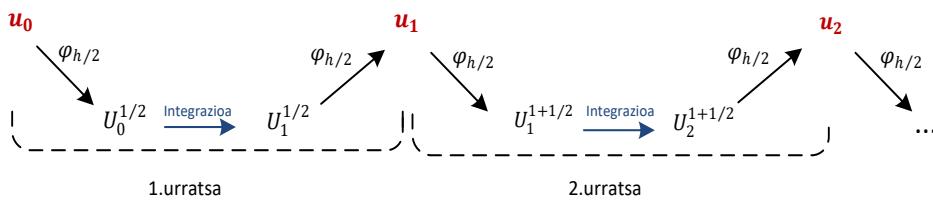
1.  $\varphi_{-\frac{h}{2}}(u_{i+1})$  kalkulatu behar da, baina  $u_{i+1}$  hirugarren azpiurratsaren emaitza denez, bere expresioa jarriko dugu, eta ikusiko dugu  $U_{i+1}^{i+\frac{1}{2}}$  aldagaietatik hasi eta fluxuan aurrera eta atzera egitearen parekoa dela, alegia, ez aldatzearen parekoa:

$$U_{i+1}^{i+1+\frac{1}{2}} = \varphi_{-\frac{h}{2}}(u_{i+1}) = \varphi_{-\frac{h}{2}}\left(\varphi_{\frac{h}{2}}(U_{i+1}^{i+\frac{1}{2}})\right) = U_{i+1}^{i+\frac{1}{2}}$$

2. Gauss metodoaren  $-h$  luzerako urratsa: Gaussen metodoa simetrikoa denez, aurreko urratsean bukaerako egoera zenari  $-h$  luzerako urratsa eragiteak hasierako egoera itzultzen du, beraz,  $(U_{i+1}^{i+\frac{1}{2}})$  balioetatik abiatuz  $U_i^{i+\frac{1}{2}}$  balioetara itzuliko gara.
3. Fluxuan urrats luzeraren erdia egin behar da: lehenengo azpiurratsean beza, fluxuan  $\frac{-h}{2}$  mugitu behar gara, baina fluxuan  $\frac{h}{2}$  mugitutako balioekin egin behar dugu, hau da:

$$\varphi_{\frac{-h}{2}}(U_i^{i+\frac{1}{2}}) = \varphi_{\frac{-h}{2}}(\varphi_{\frac{h}{2}}(u_i)) = u_i$$

Metodoaren integrazio eskema orokorra [7.1.](#) irudian laburtu dugu, bere simetria ere bertan ikus daiteke,



**7.1. Irudia:** Metodoaren integrazio eskema orokorra. Metodoa simetrikoa eta simplektikoa da.

Integrazioaren urrats guztietan ez baditugu emaitzak itzuli behar, bi urratsen arteko,  $\varphi_{h/2}$  fluxuaren bi konputazioak,  $\varphi_h$  fluxuaren konputazio bakarrarekin konputatuko dugu. Horretarako, proiekzio kontzeptua sortuko dugu ([7.2.](#) irudia).

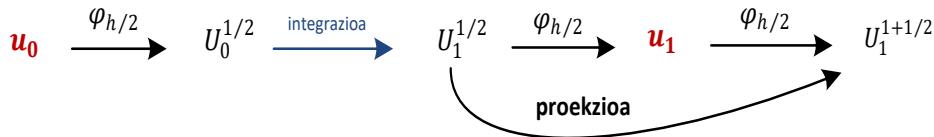
Azkenik, emaitzak behar ditugun urratsetarako fluxua  $\varphi_{-h/2}$  aplikatuko dugu ([7.3.](#) irudia).

$u_i$  jatorrizko aldagaiak eta  $U_i$  aldagai berriak adierazten duten notazioa erabiliako dugu. Hauek dira, integratzeko emango ditugun urratsak:

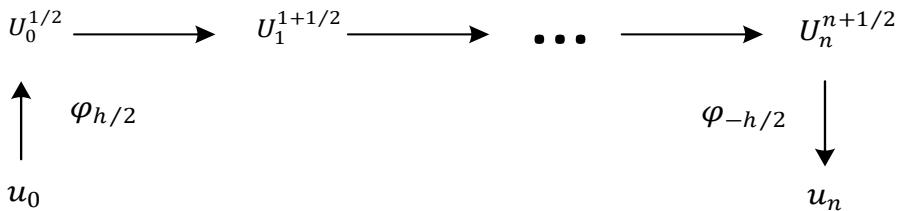
1. *Startfun* funtzioa.

Lehenengo,  $u_0$  jatorrizko aldagaien hasierako baliotik abiatuta,  $\varphi_{h/2}$  fluxuaren konputazioaren bidez, aldagai berrieta dagokion hasierako balioa lortuko dugu.

$$u_0 \rightarrow U_0^{\frac{1}{2}}.$$



**7.2. Irudia:** Proiekzioa: bi urratsen arteko,  $\varphi_{h/2}$  fluxuaren bi konputazioak,  $\varphi_h$  fluxuaren konputazio bakarrarekin konputatuko dugu



**7.3. Irudia:**  $u_i$  jatorrizko aldagaiak eta  $U_i$  aldagai berriak adierazten dute. Lehenengo,  $u_0$  jatorrizko aldagaien hasierako baliotik abiatuta, aldagai berriei dagokion  $U_0^{1/2}$  hasierako balioa finkatuko dugu. Urrats bakoitza, integrazio eta proiekzioaren konposaketa da, 7.2. irudian zehaztu dugun bezala. Erabiltzaileak definitutako urratsetarako,  $u_n$  jatorrizko aldagaietan zenbakizko soluzioa itzuliko dugu

## 2. Urratsa.

Urratsa bi azpiurratsen konbinazio bezala ikusiko dugu: aldagai berriei Gaussen metodoaren bidezko integrazioaren urrats bat eta lortutako mai-  
tzei  $\varphi_h$  bidez fluxuan  $h$  aurrera egitea. Bigarren azpiurratsa fluxuaren bidez  
proiektatzea da. 7.2. irudian zehaztapenak eman ditugu.

$$U_0^{\frac{1}{2}} \rightarrow U_1^{\frac{1}{2}} \rightarrow U_1^{1+\frac{1}{2}}.$$

Biribiltze errorea txikitzeko, proiekzioa doitasun altuan konputatzea garra-  
tzitsua da. Modu honetan, batura konpensatua aplikatzerakoan zifra batzuk  
irabaziko ditugu.

### 3. Outputfun funtzioa.

Erabiltzaileak,  $t$ -ren balio jakin batzuetan  $u(t)$  balioen zenbakizko soluzioak nahiko ditu, kasu horietan  $\varphi_{-h/2}$  fluxuaren konputazioaren bidez,  $U_i^{i+\frac{1}{2}}$  balioetatik  $u_i$  jatorrizko aldagaien balioak lortuko dira:

$$U_n^{n+\frac{1}{2}} \rightarrow u_n.$$

Gauss metodoa, neurri batean Splitting eta konposizio metodoen baliokideak dira.

Konposizio metodoa  $\equiv$  Gauss metodoa aldagai aldaketa gabe.

Splitting metodoa  $\equiv$  Gauss metodoa aldagai aldaketarekin.

Splitting metodoekiko antzekotasuna azaltzeko, (2.21) *Störmer-Verlet* Splitting metodoarekin konparatuko dugu. *Störmer-Verlet* metodoa, era honetan aplikatzen da:  $h/2$  fluxua aplikatu, perturbazioak kalkulatu eta berriz  $h/2$  fluxua aplikatu. Fluxuaren aldagai aldaketarekin, gauza bera egiten ari gara:  $h/2$  fluxua aurreratu, perturbazioak kalkulatu (aldagai berriean eta beraz, hobeto kalkulatzen dugu),  $h/2$  fluxua aurreratu.

## 7.5. Zenbakizko esperimentuak.

Zenbakizko esperimentuetarako, puntu-finkoaren iterazioan oinarritutako Gauss metodoaren implementazioa (5. kapitulua) erabili dugu eta eguzki-sistemaren ekua-zio differentzialei, Kepler-en fluxuan oinarritutako aldagai aldaketa aplikatu diegu.  $s = 6, 8, 9, 16$  atalako Gauss metodoak exekutatu ditugu eta metodo eraginkorrena aukeratu dugu, *COI035* konposizio eta *ABAH1064* Splitting metodoekin konparatzeko.

Esperimentu hauen konputaziorako, 64-biteko (bikoitza) eta 80-biteko (*long double*) doitasunak nahasi ditugu. Konputazioaren zati nagusiena, 64-biteko doitasunean egin dugu eta proiekzioa kalkulatzeko, 80-biteko doitasuna aplikatu dugu. Era honetan, modu merkean soluzioaren doitasuna hobetzea lortu dugu.

Gauss metodoaren exekuzio sekuentziala eta paraleloak egin ditugu.  $s$  atalen funtzioen balioztapena,

$$F_{n,i} = f(Y_{n,i}), \quad i = 1, \dots, s,$$

independenteak dira eta paraleloan kalkula daitezke. Exekuzio paraleloak  $s = 8$  metodoarentzat egin ditugu, eta bi kasu aztertu ditugu: hari kopuruak 2 eta 4.

## Problemak.

9-planeten problema (3.4. atala) erabili dugu integrazioetarako. Hasierako balioak *DE-430* efemerideen artikulutik hartu ditugu: planeten masak 3.4. taulan laburtu ditugu; eta hasierako kokapen eta abiadurak 3.5. taulan aurki daitezke.

Koordenatu heliozentrikoei dagokien Hamiltondar sistema (B.3),

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

integratu dugu.  $H_K(q, p)$  mugimendu Keplerrari dagokion Hamiltondarraren aldea da eta  $H_I(q, p)$ , perturbazioei dagokien Hamiltondarraren aldea da.

Integrazioen tarteak,  $t_{end} = 10^6$  egunetako da eta zenbakizko integrazioetan,  $h$ -ren balio ezberdinak erabili ditugu.  $s = 6$  metodoarentzat urrats luzerak aukeratu ditugu eta gainontzeko metodoentzat,  $s$ -atalen araberako urrats luzera proportzionalak finkatu ditugu:

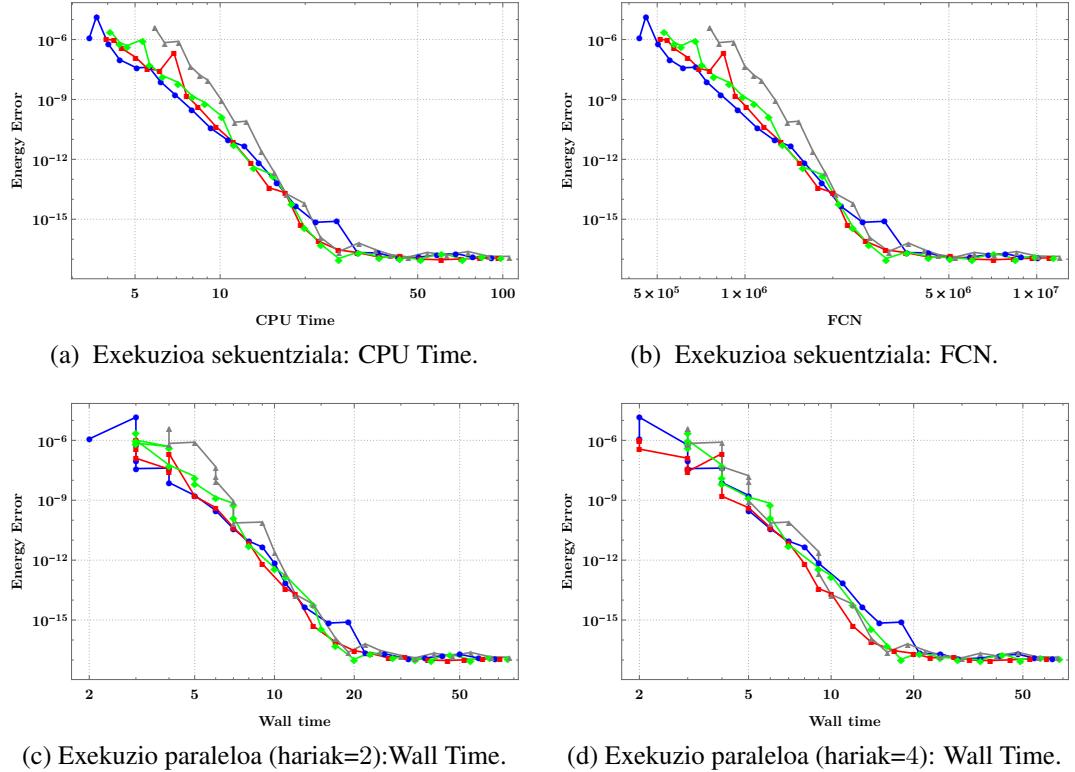
$$\begin{aligned}s = 6 : \quad h &= 2^{k/4}, \quad k = 4, \dots, 28, \\ s = 8 : \quad h &= (8/6)h, \\ s = 9 : \quad h &= (9/6)h, \\ s = 16 : \quad h &= (16/6)h.\end{aligned}$$

Zenbakizko esperimentuetarako, aldagai aldaketa planeta guziei aplikatzea erabaki dugu. 9-planeten probleman, gorputz kopurua txikia denez, Kepler fluxuaren gainkarga esanguratsua da eta barne-planetei bakarrik aplikatzea, eraginkorragoa izan daiteke. Baino, gorputz gehiago konsideratzen baditugu (esaterako ilargia eta asteroide nagusienak) edo eguzki-sistemaren eredu konplexuagoetan (esaterako erlatibilitatea efektua gehitzerakoan), perturbazio aldearen konputazioa nagusituko da eta Kepler fluxuaren kalkuluak pisua galduko luke.

## Lehen esperimentua.

7.4. irudian,  $s = 6, 8, 9, 16$  ataletako metodoen eraginkortasun grafikoak irudikatu ditugu. Eraginkortasuna, energia errore erlatibo maximoaren arabera neurtu dugu: lehen kasuan, *CPU*-denborarekiko (exekuzio paraleloetan *Wall time*) eta bigarren kasuan, ekuazio diferenzialen ebaluazio kopuruarekiko (*FCN*). Eraginkortasuna *CPU*-rekiko, problema zehatz honetarako gertatzen dena azaltzen digu eta era-ginkortasuna *FCN*-rekiko, metodoak problema erreald batean nola jokatuko luke erakusten digu.

Exekuzio sekuentzialak eta exekuzio paraleloak aztertuz, Gauss metodo era-ginkorrena aukeratu nahi dugu. Horretarako, biribiltze errorea nagusitzen hasten



**7.4. Irudia:** Eraginkortasun grafikoak eskala logaritmiko bikoitzean irudikatu ditugu. Batetik, ardatz bertikalean, energiaren errore erlatibo maximoa eman dugu. Bestetik, ardatz horizontalean, (a),(c),(d) irudietan CPU denbora (exekuzio paralelotan Wall-Time) eta (b) irudian, ekuazio diferentzialen ebaluazio kopurua (FCN) erakutsi dugu. (a) konputazioa modu sekuentzialean egin dugu. (c) eta (d) modu paraleloan: (c) kasuaren exekuzioa hari kopurua 2 da eta (d) kasuan hari kopurua 4 izan dira. Irudi bakoitzean, Gauss metodoaren  $s$  ataletako lau integrazio konparatu ditugu:  $s = 6$  urdinez,  $s = 8$  gorriz,  $s = 9$  berdez, eta  $s = 16$  grisez.

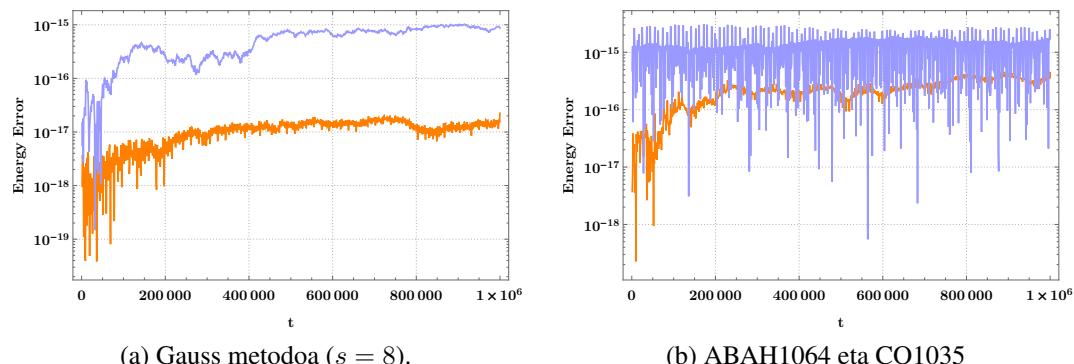
den inguruko unean gertatutakoa aztertu dugu:  $s = 8, 9, 16$  metodoak,  $s = 6$  metodoa baino eraginkorragoak azaldu zaizkigu.  $s = 8, 9, 16$  metodoak beraien artean oso antzekoak izanik,  $s = 8$  ataleko Gauss metodoa aukeratu dugu. Bestalde, exekuzio paraleloa, sekuentziala baino eraginkorragoa da eta 4 hari erabili dugun integrazioa, eraginkorrena azaldu zaigu.

## Bigarren esperimentua.

### Energiaren eboluzioa

[7.5.](#) irudian energiaren errore erlatiboaren eboluzioa erakutsi dugu lau integrazio metodo hauetarako:

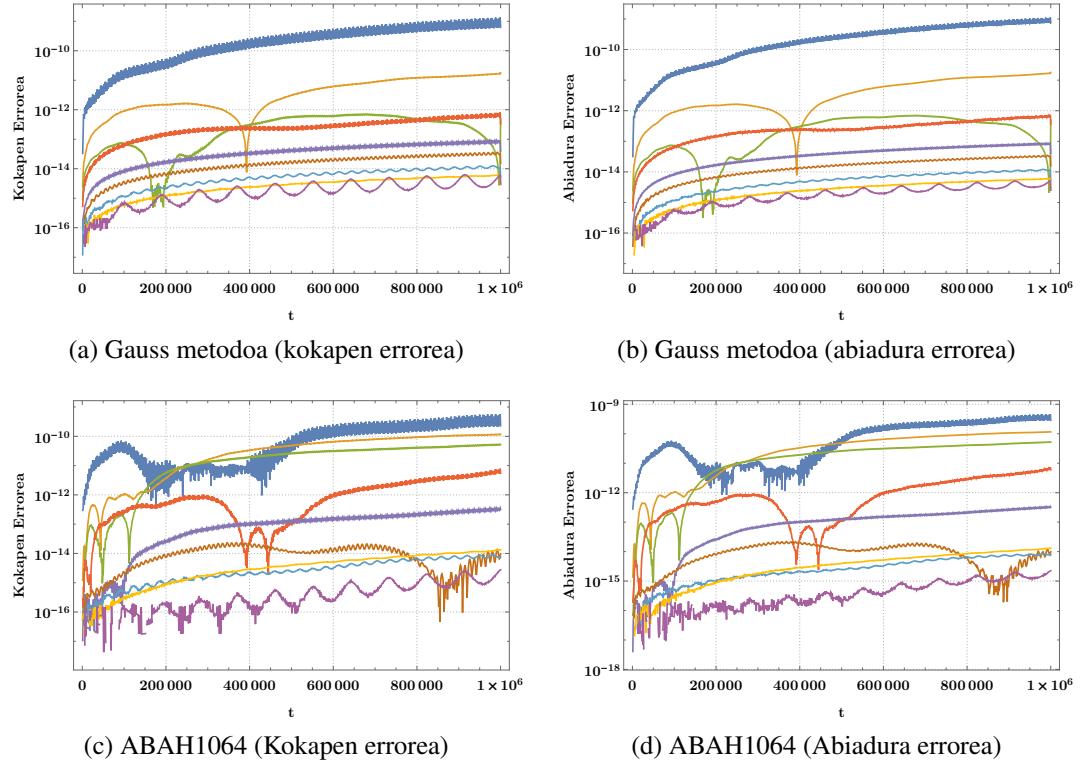
1. Gauss metodoa  $s = 8$  (proiekzioa *long double* doitasunarekin). Integrazio-rako  $h = 10.63$  urrats luzera erabili dugu.
2. Gauss metodoa  $s = 8$  (proiekzioa *double* doitasunarekin). Integraziorako  $h = 10.63$  urrats luzera erabili dugu.
3. *ABAH1064* Splitting metodoa. Integraziorako  $h = 4.76$  urrats luzera erabili dugu.
4. (CO1035) Konposizio metodoa. Integraziorako  $h = 4.76$  urrats luzera era-bili dugu.



**7.5. Irudia:** Energia errorearen eboluzioa lau integrazio metodoetarako erakutsi dugu. Ezkerreko irudian,  $s = 8$  atalako Gauss metodoaren  $h = 10,667$  urrats luzerarekin egindako integrazioak erakutsi ditugu: proiekzioa *long double* doitasunarekin (laranjaz) eta proiekzioa *double* doitasunarekin (urdinez). Eskueko irudian, Splitting/Konposizio metodoak erakutsi ditugu: ABAH1064 (laranjaz) eta CO1035(urdinez)

### Kokapen eta abiadura errore estimazioak

[7.6.](#) irudian, Gauss ( $s = 8$ ) eta *ABAH1064* metodoentzako, kokapen eta abiadura-ren errorearen estimazioak erakutsi ditugu. Erroreak estimatzeko, urrats txikiagoako integrazioaren soluzioarekiko differentzia gisa kalkulatu dugu. Gauss metodoaren integrazioan, urrats luzera handiago erabili arren, barne-planeten kokapen eta abiaduren errore estimazioak txikiagoak izan dira.

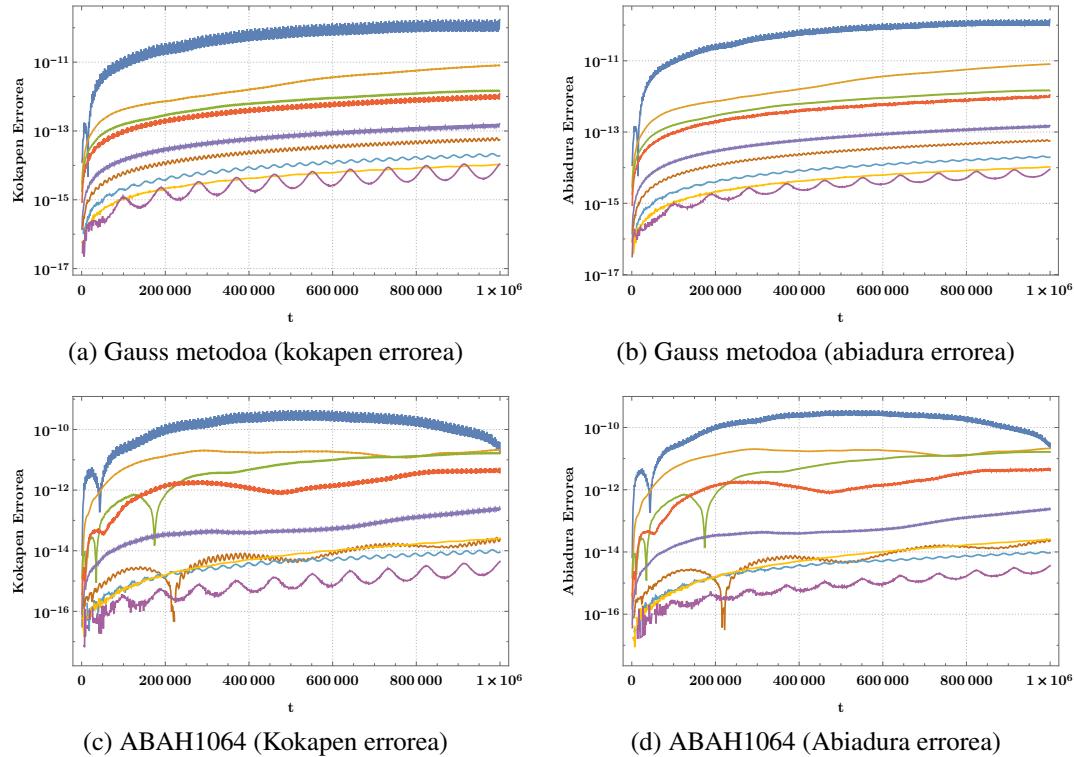


**7.6. Irudia:** Kokapen eta abiaduraren errorearen estimazioak erakutsi ditugu. (a) eta (b) irudietan,  $s = 8$  ataletako Gauss metodoaren errore estimazioak eman ditugu,  $h = 10,667$  urrats luzerarekin integratuz. (c) eta (d) irudietan, *ABAH1064* Splitting metodoaren errore estimazioak eman ditugu,  $h = 4.76$  urrats luzerarekin integratuz. Kolore bakoitza planeta bakoitzari dagokion errorea da: Merkurio (urdin ilunez), Artizarra (marroi argiz), Lurra (berdez), Marte (gorriz), Jupiter (more argiz), Saturno (marroi ilunez), Urano (urdin argiz), Neptuno (laranja argiz), Pluto (morez)

irudi berriak !!!!!

## Hirugarren esperimentua.

Hirugarren esperimentu honetan, biribiltze errorearen azterketa egin dugu eta horretarako, momentu angeluarren errore erlatiboaren eboluzioa eman dugu. Momentu angeluarren trunkatze errorea beti zero da, metodo simplektikoek invariante koadratikoa zehazki mantentzen dituzte. [7.8.](#) irudian, Gaussen  $s = 8$  ataletako metodoarekin,  $h = 9.0$  eta  $h = 10.63$  urrats luzerarekin integrazioen soluzioen momentu angeluarren errorea erakutsi dugu.

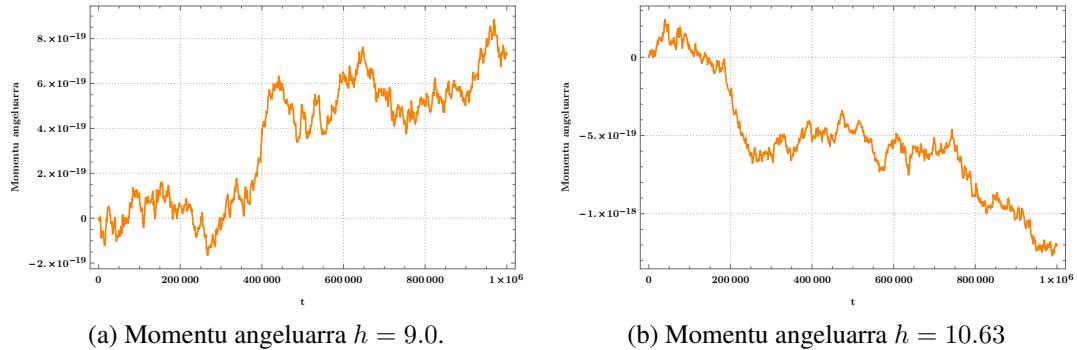


**7.7. Irudia:** Kokapen eta abiaduraren errorearen estimazioak erakutsi ditugu. (a) eta (b) irudietan,  $s = 8$  ataletako Gauss metodoaren errore estimazioak eman ditugu,  $h = 4,48$  urrats luzerarekin integratuz. (c) eta (d) irudietan, ABAH1064 Splitting metodoaren errore estimazioak eman ditugu,  $h = 2.38$  urrats luzerarekin integratuz. Kolore bakoitza planeta bakoitzari dagokion errorea da: Merkurio (urdin ilunez), Artizarra (marroi argiz), Lurra (berdez), Marte (gorriz), Jupiter (more argiz), Saturno (marroi ilunez), Urano (urdin argiz), Neptuno (laranja argiz), Pluto (morez)

## Laugarren esperimentua.

7.9. irudian, Gauss metodoa eta konposizio/Splitting metodoen arteko eraginkortasunaren konparaketa egin dugu.  $s = 8$  ataletako Gauss metodoa, modu sekuentzialean eta modu paraleloan exekutatu dugu. Esperimentu hauetan, eguzkisistemaren eredu simplearen integratziorako, Splitting metodoak oso eraginkorrik azaldu zaizkigu. Gauss metodoaren exekuzioa paralelizatzeak abantaila erakusten du baina hala ere, Splitting metodoak eraginkorragoak dira.

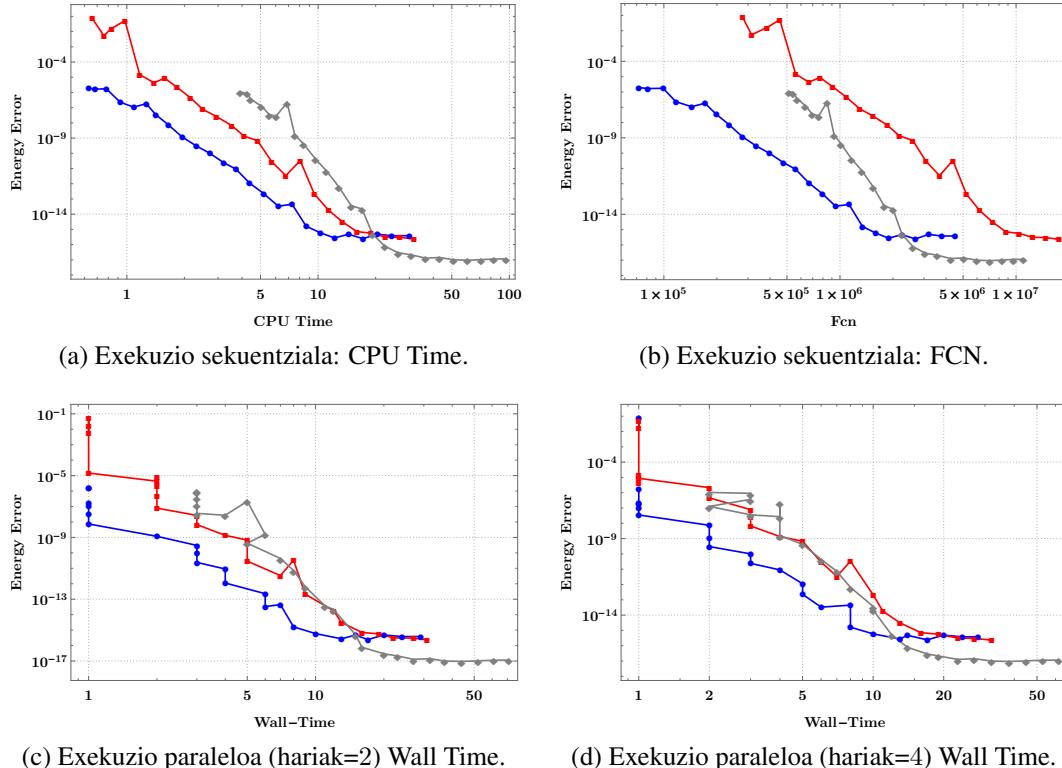
Dena den, eredu errealistagoak (gorputz kopurua handitzen delako edota erlatibilitatea kontutan hartzen delako) integratzeko, Gauss metodoak eraginkorragoak bilakatzea espero da. Splitting metodoen konputazioak, modu trinkoan

(a) Momentu angeluarra  $h = 9.0$ .(b) Momentu angeluarra  $h = 10.63$ 

**7.8. Irudia:** Momentu angeluarren errore erlatiboaren eboluzioa erakutsi dugu. Gaussen  $s = 8$  ataletako metodoa integratu dugu  $h = 9.0$  eta  $h = 10.63$  urrats luzerarekin.

kalkulatu behar dira, hau da, atalen konputazioak sekuentzialki exekutatzen dira eta ez ditu aldaerak onartzen. Gauss metodoaren ekuazio implizituak ebazteko, ordea, teknika ezberdinak konbina daitezke eta eraginkortasuna hobetzeko aukera asko eskaintzen dizkigu. Adibidez, iterazio gehienak problemaren eredu simple batekin, doitasun baxuan kalkula daitezke [13] eta bukaerako iterazio pare bat eredu osoarekin, doitasun altuan. Gauss metodoaren  $s$ -ataletako funtzioen ebaluazioak modu paraleloan exekutatu daitezke eta eguzki-sistemaren eredu komple-xuagoa kontutan hartzen den neurrian, paralelizazioak abantaila handiagoa suposatuko du.

## 7.6. Laburpena.



**7.9. Irudia:** Eraginkortasun grafikoak eskala logaritmikoa bikoitzean irudikatu ditugu. Batetik, ardatz bertikalean, energiaren errore erlatibo maximoa eman dugu. Bestetik, ardatz horizontalean, (a),(c),(d) irudietan CPU denbora (exekuzio paralelotan Wall-Time) eta (b) irudian, ekuazio differentzialen ebaluazio kopurua (FCN) erakutsi dugu. (a) konputazioa modu sekuentzialean egin dugu. (c) eta (d) modu paraleloan: (c) kasuaren exekuzioa hari kopurua 2 da eta (d) kasuan hari kopurua 4 izan dira. Irudi bakoitzean, hiru integrazio metodoa konparatu ditugu:  $s = 6$  Gauss metodoa grisez, ABAH1064 urdinez eta CO1035 gorriz



## **8. Kapitulua**

### **Eztabaida.**

Kapitulu honetan, ikerketan bildutako informazio baliagarria azaldu dugu.

#### **8.1. Eguzki-sistemaren integrazio luzeak.**

Gure helburua eguzki-sistemaren epe luzeko eta doitasun handiko implementazio eraginkorra proposatzea da. Atal honetan, egungo eguzki-sistemaren simulazioetarako erabiltzen diren metodo eta implementazioen laburpena egingo dugu.

##### **Implementazioen garapena.**

Astronomi arloaren ikerketetan, eguzki-sistemaren eredu errealisten integrazio luzeak konputatzen dituzte. A. Morbidellik [90], eguzki-sistemaren zenbakizko integrazioen implementazioen azterketa egin zuen: urteak aurrera joan ahala, eguzki-sistemaren ereduak gero eta konplexuagoak bilakatzen dira, eta integrazio tarteak handiago da. Garai hauek bereizten ditu:

###### **1. Garai klasikoa.**

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

###### **2. Garai simplektikoa.**

Wisdom eta Holman-en [108, 1991] lanarekin, eguzki-sistemaren azterketa erabiltzen integratzaile simplektikoen erabilera zabaldu zen. Garai honetan, ( $10^8 - 10^9$ ) urte arteko eguzki-sistemaren integrazioak egin ziren.

3. Garai estatistikoak.

Planeten, eta asteroideen edota meteoritoen moduko gorputz txikien arteko kolisiotik gertuko egoerak kalkulatzen dituzten algoritmoak garatu ziren. Implementazio berri hauetan, milaka gorputzen integrazio azkarra egin daiteke. Asteroide eta meteoritoen orbiten distribuzio azterketa estatistikoak egin zitzuten.

4. Planeten sorrera garaiko azterketak.

Garai honetan, eguzki-sistemaren sorrerari buruzko simulazioak nagusi dira; masa handiko gorputzen arteko kolisiotik gertuko egoerak gertatzen diren problemak konputatzen dira.

Astronomi arloko eguzki-sistemaren integrazioetarako, nagusiki bi integratzaile famili bereiz daitezke [64]; metodo simetrikokoak eta simplektikoak. Bestalde, integrazio metodo orokorrak, efemerideen konputazioetarako aplikatzen dira.

1. Metodo simetrikokoak.

Metodo simetrikoen artean nagusiena, 4 ordenako *Hermite* integratzailea [3] dugu. Urrats luzera tamaina aldakorreko integratzaile da, eta modu errazean implementatu daiteke. *Hermite* integratzailea konputazionalki garestia da eta gorputz kopuru handia duten eta kolisiotik gertuko egoerak maiz gertatzen diren problemetan aplikatzen da; esate baterako, eguzki-sistemaren sorreraren azterketarako.

2. Metodo simplektikoak.

Egungo eguzki-sistemaren epe luzeko integrazioetarako, integratzaile sinplektikoak nagusitu dira.

3. Metodo orokorrak.

Efemerideen doitasun altuko integrazioetarako, metodo orokorrak erabilten dituzte: *Multistep Adams* metodoa (NASA), *Adams-Cowell* metodoa (IMCCE, Paris Observatory) eta *Radau* (IAA, St. Petersburg) metodoa.

## Metodo simplektikoak.

Wisdom eta Holman-en [108, 1991] eguzki-sistemaren epe luzeko simulazioetarako integratzaile sinplektikoak (*WH*), arrakasta izan zuen. Eguzki-sistema, mugimendu perturbatua duen sistema dinamikoa da eta ezaugarri honi egokitutako integratzaile eraginkorra garatu zuten. Jacobi koordenatuak aplikatuz, N-gorputzen problemaren Hamiltondarra, bi zatitan banatu zuten,

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

non  $H_K$ , Hamiltondarraren alde Keplériarra (planeten eguzkiarekiko mugimendu Kleperiarra) eta  $H_I$ , Hamiltondarraren perturbazioa (planeten arteko interakzioak) diren. Integrazioaren urrats bakoitzean, Hamiltondar bakoitzaren soluzioa tarte-katuz, problema osoaren soluzioa lortzen da.

$WH$  implementazioaren erabilgarritasuna mugatua da. Batetik, izar anitzeko planeten sistemak edo planeta-ilargiak sistemak integratzeko ez da egokia. Bestetik,  $WH$  metodo simplektikoa denez, urrats luzera finkoa aplikatu behar da eta hau, gorputzen arteko kolisiotik gertuko egoerak dituzten problemak modu eraginkorrean integratzeko eragozpen bat da. Arazo hauek gainditzeko, hurrengo urteetan algoritmo honen aldaerak proposatu dira.

Levinson eta Duncan-ek [84, 1994],  $WH$  implementazioa, integratzaile ez simplektiko batekin konbinatu zuten, kolisiotik gertuko egoeren kalkulua hobetzeko. *SWIFT* paketean, *RMVS3* izeneko integratzailea implementatu zuten. Duncan, Levinson eta Lee-k [34, 1998], koordenatu heliozentrikoak erabiliz, Hamiltondarra beste modu honetan banatu zuten,

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

eta kolisiotik gertuko egoerei, urrats luzera txikituz aurre egin zioten. Implementazio honek, *SYMBA* izena du. Chambers [25], koordenatu heliozentrikoetan oinarritu zen eta kolisiotik gertuko egoerak gertatzen diren uneetan,  $WH$  implementazioa, beste integrazio metodo batekin (Bulirsch-Stoer metodoa) konbinatu zuen. Implementazio honek, *MERCURY* izena du. Kvaerno eta Leimkuhler [76] eta beste autore batzuk ere, antzeko ideiak landu dituzte.

Exzentrizitate handiko orbitadun sistementzako hainbat “erregularizazio simpletiko integratzaileak” aztertu dira: Levinson and Duncan (1994). “RMVS”: Regularized mixed variable symplectic integrator”, Mikkola (1997), Fukushima (2001), Beus(2003). Erregularizazioa aplikatzen dituzte metodo simplektiko ezbedinen “review” honakoa dugu: “Rauch and Holman (1999). Dynamical chaos in the Wisdom-Holman integrator”.

Wisdom eta Holmanek proposatutako Hamiltondarraren banaketa, (2.21) *Leapfrog* metodoaren bidez integratzen da eta beraz,  $p = 2$  ordenako da. Ordena altuagoko ( $p > 2$ ) metodo simplektikoak definitzeko, koefiziente negatiboak erabili behar zirela uste zen [116, 82] eta modu honetan definitutako metodoak, ez dira *Leapfrog* metodoa baino eraginkorragoak.

McLachlan-ek [88, 1995] eta Laskar-ek [82, 2001] koefiziente negatiboen arazoa gainditu zuten eta koefiziente positiboekin definitutako ordena altuko Splitting eskemak aurkitu zituzten. Berriki, Blanes-ek [16, 2012] ordena altuko Splitting eskema eraginkorrapak aurkitu ditu.

Hernandez eta Bertschinger-ek [57, 2015] N-gorputzen problema grabitazionala eta kolisiotik gertuko egoerak integratzeko, 2 ordenako metodo simplektiko

berri bat proposatu dute. Hernandez eta Bertschinger-ek [57] koordenatu cartesiarretan oinarrituz, N-gorputzen problema, 2-gorputzen azpiproblemetan banatzen dute.

### Konposizio eta Splitting metodoak.

Konposizio metodoak, era honetako Hamiltondar sistemak integratzeko aplika daitezke,

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

Konposizio metodoa, eguzki-sistemaren eredu grabitazionalari aplikatzerakoan oso eraginkorra da, baina eredu konplexuagoetarako bere abantaila galtzen du. Batetik, gorputzen kopurua handitzen bada bere eraginkortasuna gutxitzen da eta bestetik, beste indar ez grabitazionalak (erlatibilitate efektua,...) ezin daitezke era-bili.

Splitting metodoak, era honetako Hamiltondar sistemak integratzeko aplika daitezke,

$$H = H_A + \epsilon H_B, \quad (8.1)$$

non  $H_A$  eta  $H_B$  independenteki integratu daitezkeen. Eguzki-sistemaren eredu errealistetan, hainbat indar ez grabitazional modu honetan gehitzeko, zaitasunak izan ditzakegu.

Laskar-ek [80, 2011], epe luzeko zenbakizko integraziozko,  $1/c^2$  ordenako eguzkiaren erlatibilitate efektua konsideratu zuen, Saha eta Tremain-ek [101] fin-kututako teknikan oinarrituz. Teknika honen bidez, Hamiltondar banagarriei egokitzen zaien erlatibilitate efektuaren expresioa lortzen da eta gainera modu eraginkorrean kalkulatzen da. Baina Splitting metodoetan, beste planeten erlatibilitate efektuak ezin daitezke erabili. Adibidez, bi planeta handienen (Jupiter eta Saturno) erlatibilitate efektua ere kontutan hartuko balira, ekuazioak ez dira gehiegikonplikatzen eta integrazio hobea lor daiteke, errorea txikituz.

Eguzki-sistemaren integrazioetarako, (8.1) Hamiltondarraren banaketa, Jacobi koordenatuak edo heliozentrikoak aplikatz lortzen da. Koordenatu cartesiarrak, ezin daitezke erabili.

IRK metodoekin lan egiteko ordea, ez daukagu halako mugarik. Ekuazio dife-rentzial orokorreai aplika dakizkienez, askatasun osoa dugu behar diren ekuazioak erabiltzeko eta interesatzen zaigun koordenatu sistema aplikatzeko.

### 8.2. Eredu deskonposaketa.

IRK metodoen implementazio eraginkorraren azterketaren hasieran, eredu deskonposaketan oinarritutako ideia ikertu dugu. Atal honen bukaeran emango ditugun

arra佐oiengatik, beste bide batzuk ikertzea zuzenago dela ikusi dugu eta planteamendu hau alde batera utzi dugu. Dena den, planteamendu honen laburpen bat emango dugu eta implementazio bide honetan sortutako hainbat ideia interesgarri jaso ditugu.

Eguzki-sistemaren eredua bi zatitan bana daiteke: problema simplea (konputazionalki merkea) eta problema konplexua (konputazionalki garestia). Newton sinplifikatuaren iterazio berezi bat aplikatu dugu, problema konplexuaren baliozta-pen kopurua txikitzeo eta era honetan, implementazio eraginkorra lortzeko. Newton sinplifikatuaren iterazio berezi honetan, ekuazio-sistema lineala LU deskonposaketaren bidez askatu beharrean, puntu-finkoaren bidez ebatzi dugu.

## Implementazioa.

Eguzki-sistema, perturbatutako mugimendu Kepleriar gisa har daiteke,

$$f(y) = k(y) + \epsilon g(y), \quad g \ll k. \quad (8.2)$$

$k(y)$  sistema dinamikoaren alde sinplifikatua da eta konputazio kostu txikia du.  $g(y)$ , ordea, sistemaren alde konplexua da eta konputazio kostu handia du. (8.2) motako ekuazio diferentzialen sistema, IRK metodoen bidez integratzeko, atalen ekuazio ekuazio-sistema implizitua ebatzi behar da:

$$Y_i = y_n + h \sum_{j=1}^s a_{ij} f(Y_j).$$

Ekuazio implizituaren sistema hau, honako iterazioaren bidez ebaaztea proposatzen dugu:

$$\begin{aligned} k &= 1, 2, \dots \\ Y_i^{[k]} &= y_n + h \sum_{j=1}^s a_{ij} (k(Y_j^{[k]}) + g(Y_j^{[k-1]})) \end{aligned} \quad (8.3)$$

Iterazio hau aplikatzea, Jacobiaren hurbilpentzat,  $J_i = \partial k / \partial y(Y_i)$ ,  $i = 1, \dots, s$  hartzen duen Newton sinplifikatua aplikatzearen baliokidea da. 27 algoritmoa.

ritmoan, iterazioa aplikatzeko zehaztasun gehiago ematen ditugu,

```

for  $n \leftarrow 1$  to endstep do
    Hasieraketa  $Y_i^{[0]}, W_i^{[0]}$ ;
    k=1;
    Askatu  $Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} k(Y_j^{[k]}) + W_i^{[k-1]}$  ;
    while (konbergentzia ez lortu) do
        k++;
         $W_i^{[k-1]} = h \sum_{j=1}^s a_{ij} g(Y_j^{[k-1]})$ ;
        Askatu  $Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} k(Y_j^{[k]}) + W_i^{[k-1]}$  ;
    end
     $y_{n+1} = y_n + h \sum_{j=1}^s b_i f(Y_j^{[k]})$  ;
end
```

**Algoritmoa 27:** Meta-algoritmoa

Urrats bakoitzean, hainbat aldiz  $k = 2, 3, \dots$  balioentzat ekuazio-sistema inplizitua askatu behar da eta horretarako, metodo egokiena aplikatzeko askatasuna dugu. Pentsa liteke, problema zurruna ez bada puntu-finkoaren bidez askatzea eta problema zurruna bada, berriz, Newton sinplifikatuaren iterazioaren bidez.

Barne ekuazio-sistema,

$$Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} k(Y_j^{[k]}) + W_i^{[k-1]},$$

puntu-finkoaren iterazioaren bidez askatzeko, [28](#) barne-iterazioa era honetan aplikatuko dugu:

```

 $l = 0;$ 
 $Y_i^{[k,0]} = Y_i^{[k-1]}$ ;
while (konbergentzia ez lortu) do
     $l = l + 1$ ;
     $K_i^{[k,l]} = k(Y_i^{[k,l-1]})$ ;
     $Y_i^{[k,l]} = y_n + h \sum_{j=1}^s a_{ij} K_j^{[k,l]} + W_i^{[k-1]}$ ;
end
```

**Algoritmoa 28:** Barne-iterazioa: puntu-finkoaren iterazioa

## Orokorpena.

### Eredu konplexuak.

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

problema emanik,

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

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

...

**Adibidea.** Eguzki-sistemaren eredu konplexuaren ekuazio-sisteman, alde Kepleriarra eta perturbazio maila ezberdinak bereiz daitezke. Meta-algoritmoa, deskonposaketaren maila bakoitzari modu errekurtsiboan aplika daiteke. Demagun, eguzki-sistemaren problemaren ekuazio diferentzial hauek ditugula,

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

non

$k(y)$  : kepleriarra.

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

$r(y)$  : planeten eguzkiarekiko erlatibilitate efektua.

Meta-algoritmoa modu honetan aplika daiteke,

1. maila

$$Y_i = y_n + h \sum_{j=1}^s a_{ij} f(Y_j).$$

2. maila,  $k = 1, 2, \dots$

$$Y_i^{[k]} = y_n + h \sum_{j=1}^s a_{ij} k(Y_j^{[k]}) + \delta_1^{[k-1]},$$

$$\text{non } \delta_1^{[k-1]} = h \sum_{j=1}^s a_{ij} (g(Y_j^{[k-1]}) + r(Y_j^{[k-1]})).$$

3. maila,  $, l = 1, 2, \dots$

$$Y_i^{[k,l]} = y_n + h \sum_{j=1}^s a_{ij} \left( k(Y_j^{[k,l]}) + g(Y_j^{[k-1,l]}) \right) + \delta_2^{[l-1]},$$

$$\text{non } \delta_2^{[l-1]} = h \sum_{j=1}^s a_{ij} r(Y_j^{[k-1,l-1]}).$$

### **Problema independenteak.**

Batzuetan, problemaren eredu simplifikatua, azpiproblema independentetan bana daiteke,

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

Azpiproblema bakoitzari dagokion barne-iterazioak independenteak dira eta paleoloan kalkula daitezke. Eguzki-sistema eredu grabitacionala, azaldutakoaren adibide argia da; planeta bakoitzaren  $k(y)$  eguzkiarekiko interakzioa, azpiproblema independentea da. Zenbakizko esperimentuetarako aukera eraginkorrena, eredu simplifikatua bi azpiproblemetan banatzea dela konprobatu dugu: alde batean, barne-planeten eredu simplifikatuak osatutako azpiproblema eta beste aldetik, kanpo-planeten eredu simplifikatuak osatutakoa.

### **Tolerantzia aldakorra.**

Kanpo eta barne-iterazioetarako geratze irizpide berdinak definitu ditugu. Kanpo-iterazioetarako, tolerantzia finkoa erabili dugu eta barne-iterazioetarako, ordea, tolerantzia aldakorra. Tolerantzia aldakorra, kanpo-iterazio (konputazionalki garrestia) bakoitzarentzat, barne-iterazio (konputazionalki merkea) kopuru nahikoak eta beharrezkoak eman daitezen, definitu dugu.

### **Eragozpenak.**

Esan dugunez, planteamendu honetan Newton simplifikatua aplikatu dugu eta ekuazio-sistema lineala LU deskonposaketaren bidez askatu beharrean, puntu-finkoaren iterazioaren bidez askatu dugu. Planteamendu honi IRK-Newton simplifikatuaren implementazio estandarrarekiko, hiru desabantaila aurkitu dizkiogu:

1. Barne kalkuluen doitasuna.

Planteamendu honen barne-iterazioak, implementazioaren oinarrizko doitasunean kalkulatu behar dira. IRK-Newton simplifikatuaren implementazioaren eragiketa konplexuenak, doitasun txikiagoan kalkula daitezke [9].

2. Jacobiarraren balioztapena.

Planteamendu honetan, Jacobiarra iterazio bakoitzean aldatzen denez, iterazio guztietai balioztatu behar da. IRK-Newton simplifikatu estandarrean, Jacobiarra urrats bakoitzean behin bakarrik kalkulatu behar da.

3. Ereduen deskonposaketa.

Ereduen deskonposaketak, nolabaiteko konplexutasuna gehitzen du. Hiru eredu ezberdin bereizi behar ditugu: eredu osoa  $f(y)$ , eredu simplifikatua  $\tilde{f}(y)$  eta perturbazioa  $g = f(y) - \tilde{f}(y)$ .

## 8.3. IRK implementazioaren oinarriak.

*IRK* implementazioan, ekuazio-sistema iterazio metodo bat aplikatuz askatu behar da. Funtsezkoa da, iterazioaren atalen hasieraketa eta geratze irizpide sendoak aplikatzea. Laburki, gure lanean aztertutako aukera ezberdinak deskribatuko ditugu.

### Atalen hasieraketa.

Iterazio metodoetan atalen hasieraketa kalkulatzeko teknika ezberdinak ikertu ditugu.

1. Metodo esplizituak.

Atalen hasieraketa metodo esplizitu bat aplikatuz lor daiteke. Hurbilpen merkea lortze aldera, ordena txikiko metodoak aplikatu ditugu: Euler  $\mathcal{O}(h)$  eta Euler hobetuaren  $\mathcal{O}(h^2)$  metodoa. Atalen hasieraketa metodo hauetan, ekuazio diferencialaren balioztapena beharrezkoa da eta era honetan aplikatu ditugu:

Euler :

$$Y_i = Y_{i-1} + h(c_i - c_{i-1})f(Y_{i-1}).$$

Euler hobetua :

$$Y_i = Y_{i-1} + h(c_i - c_{i-1})f(k_i),$$

$$\text{non } k_i = Y_{i-1} + \frac{h}{2}(c_i - c_{i-1})f(Y_{i-1}).$$

2. Kepler-en fluxua.

Eguzki-sistema, perturbatutako sistema Keplériarra denez, atalak Kepler fluxuaren bidez hasieratu daitezke. Kepler fluxuaren funtzioak,  $y(t_n)$  une bateko kokapen eta abiadurak emanik,  $\Delta t_n$  denbora igarotakoan kokapen eta abiadura zehatzak itzultzen ditu,

$$\text{Keplerflow}(\Delta t_n, y(t_n)) \rightarrow y(t_n + \Delta t_n).$$

Beraz,  $y_n$  egoeratik abiatuta, atal bakoitzari dagokion hasieraketa,

$$Y_{n,i}^{[0]} = y(t_n + hc_i), \quad i = 1, \dots, s,$$

Kepler fluxua  $\Delta t_n = hc_i$  denbora aurrera eginez lortuko dugu.

### 3. Interpolazio polinomioak.

Aurreko urratseko ataletako informazioa erabiliz, urrats berriaren atalen hasieraketa lortzen dugu. Problema ez-zurrunetarako interesgarria da. Era honetako hasieraketa merkea da, ez baita funtzió balioztapen berririk egiten. Era honetan aplikatzen da,

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

non  $\lambda_{ij}$  aurre-kalkulatutako interpolazio koefizienteak diren.

Bide honetatik, teknika aurreratuagoak ere aztertu ditugu (maila txikiko polinomioen bidezko hasieraketak). Era berean, B-Serieak teknikan [26] oinarrituz, interpolazio estandarra [77] hobetzeko saiakera egin dugu baina ez dugu arrakastarik izan. Interpolazio metodo estandarrarekin, urrutien da goen ataletarako hasieraketa txarra lortzen da eta atal askotako metodoen-tzat, arazo hau, eragozpen handia izan daiteke.

Problema ez-zurrunetarako, interpolazio polinomioen bidezko hasieraketa ona eta merkea lortzen da. Problema zurrunetarako, ordea, atalak aurreko urratseko soluzioarekin hasieratuko ditugu,

$$Y_{n,i}^{[0]} = y_n, \quad i = 1, \dots, s.$$

## Geratze irizpidea.

Iterazio metodo bat aplikatzeko, geratze irizpide sendoa finkatzea funtsezkoa da. Iterazioak, biribiltze errorearen eragina azaltzen denean geratu behar dira. Bateziki, iterazioak beranduegi geratzen baditugu, alferrikako iterazioak emango ditugu, eraginkortasunaren kalterako. Bestetik, iterazioak goizegi geratzen baditugu, biribiltze errorea handitu edota trunkatze errorea eragin ditzake.

Geratze irizpide estandarrak aplikatzerakoan, hainbat arazo aurkitu ditugu eta ondorioz, geratze irizpide berri bat definitzeko beharra ikusi dugu. Geratze irizpide berria definitzeko, aukera ezberdinak aztertu ditugu eta eskuartean ibili ditugun bertsio nagusienak azalduko ditugu.

**Norma.**

Iterazio bakoitzaren hobekuntza ( $\Delta^{[k]}$ ) neurtzeko, norma ezberdinak aplikatzen dira. Honakoa da, Hairer-ek puntu-finkoaren iterazioan oinarritutako IRK implementazioaren geratze irizpidean aplikatzen duen norma,

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

Norma hau zalantzat jarri dugu eta iterazioaren hobekuntza hobeto neurtzen duen norma finkatzen saiatu gara. Honako aukerak aztertu ditugu:

## 1. Lehen bertsioa.

Diferentziak, modu erlatiboan neurtzeko beharrak bultzatuta, norma era honetan definitu dugu,

$$\Delta = \max_{1 \leq j \leq d} \frac{\max_{1 \leq i \leq s} |\Delta Y_i^j|}{(\max_{1 \leq i \leq s} |Y_i^j|) + \delta},$$

non  $\delta \approx 10^{-20}$ , zatitzalea zero ez izateko finkatzen dugun balio txikia eta  $y = (y^1, \dots, y^d)$  den.

## 2. Bigarren bertsioa.

Kokapenen ( $Q_i \in \mathbb{R}^d$ ) eta abiaduren ( $V_i \in \mathbb{R}^d$ ) izaera ezberdina dela jabetuta, norman bi kontzeptu hauek banatu ditugu.  $Y = Y^{[k]}$ , eta  $\tilde{Y} = Y^{[k-1]}$  notazioa finkatuta,

$$\Delta =$$

$$\max \left( \max_{1 \leq j \leq d} \frac{\max_{1 \leq i \leq s} \|Q_i^j - \tilde{Q}_i^j\|^2}{\max_{1 \leq i \leq s} \|Q_i^j\|^2}, \max_{1 \leq j \leq d} \frac{\max_{1 \leq i \leq s} \|V_i^j - \tilde{V}_i^j\|^2}{\max_{1 \leq i \leq s} \|V_i^j\|^2} \right),$$

non,

$$Y = \begin{pmatrix} Q_1^1 & \dots & Q_1^d & V_1^1 & \dots & V_1^d \\ Q_2^1 & \dots & Q_2^d & V_2^1 & \dots & V_2^d \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Q_s^1 & \dots & Q_s^d & V_s^1 & \dots & V_s^d \end{pmatrix} \quad \tilde{Y} = \begin{pmatrix} \tilde{Q}_1^1 & \dots & \tilde{Q}_1^d & \tilde{V}_1^1 & \dots & \tilde{V}_1^d \\ \tilde{Q}_2^1 & \dots & \tilde{Q}_2^d & \tilde{V}_2^1 & \dots & \tilde{V}_2^d \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \tilde{Q}_s^1 & \dots & \tilde{Q}_s^d & \tilde{V}_s^1 & \dots & \tilde{V}_s^d \end{pmatrix}.$$

## 3. Hirugarren bertsioa.

Iterazioaren hobekuntza, norma jakin bat aplikatuz zenbaki eskalar bakarra-rekin neurtu beharrean,

$$\Delta^{[k]} = |Y^{[k]} - Y^{[k-1]}| \in \mathbb{R}^{sd} \quad (8.4)$$

matrizea erabiliko dugu. Modu honetan, geratze irizpidea normaren independentea izango da eta gainera, geratze irizpide segurua eraikitzeo aukera emango digu.

### **Geratze irizpidea.**

Geratze irizpidearen abiapuntua Hairer-en geratze irizpidea izan da,

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

Geratze irizpide hau hobetzeko proposamen batzuk garatu ditugu,

1. Lehen bertsioa.

Hairer-ek definitutako geratze irizpidea, batzuetan goizegi geratzen da eta ziurtasuna handitu beharra dago. Horregatik, geratze irizpidean,  $k - 2$ . iterazioaren informazioa erabiltzea aztertu dugu,

$$(\Delta^{[k]} = 0) \text{ edo } (\Delta^{[k]} \geq \Delta^{[k-1]} \text{ eta } \Delta^{[k]} \geq 0.81 * \Delta^{[k-2]}).$$

2. Bigarren bertsioa.

Bertsio honetan,  $k - 3$ . iterazioaren informazioa ere erabiliz saiakera berria proposatu dugu:

$$(\Delta^{[k]} \leq tol) \text{ edo } (zat^{[k]} \geq koeff * \max(zat^{[k-1]}, zat^{[k-2]})),$$

non  $zat^{[k]} = \Delta^{[k]} / \Delta^{[k-1]}$ ,  $tol \approx 10^{-16}$  eta  $koeff = 10$  den.

3. Hirugarren bertsioa.

$$(\Delta^{[k]} \leq tol) \text{ edo } \left( \frac{zat^2}{(1 - zat)} \Delta^{[k-1]} \leq tol \right) \text{ edo } \left( \frac{zat}{(1 - zat)} \Delta^{[k-2]} \leq tol \right),$$

non  $zat = \max(zat^{[k-1]}, zat^{[k-2]})$ ,  $tol \approx 10^{-16}$  den.

Geratze irizpide honetan, iteraziotik irteten denean,  $z \geq 1$  betetzen bada,

- (1).  $\Delta^{[k]} > c tol \rightarrow$  konbergentzia arazoak (exekuzioa geratu),
- (2).  $\Delta^{[k]} \leq c tol \rightarrow$  birbiltze errorea (exekuzioa jarraitu),

non  $c \approx 10^4$  konbergentzia koefizientea den.

#### 4. Laugarren bertsioa.

Azkenik, honakoa proposatu dugu: (8.4)  $\Delta^{[k]} \in \mathbb{R}^{sd}$  izanik, iterazioak  $k = 1, 2, \dots$  jarraitzea,  $\Delta^{[k]} = 0$  bete arte edo honako baldintza bi iterazio jarrietan betetzen den artean,

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

Geratze irizpide honetan, iterazioak bukatu ondoren  $\exists j, \Delta_j^{[K]} \neq 0$  betetzen bada, orduan urratsa onargarria den ala ez erabaki behar dugu.

## 8.4. Doitasun altuko konputazioak.

Zientzia-konputazioan, nagusiki doitasun bikoitzeko aritmetika (64-bit) aplikatzen da baina eguzki-sistemaren epe luzeko integratorako, doitasun bikoitza ez da nahikoa. Jarraian, zenbakizko integracio hauetan [79], doitasuna hobetzeko bi bide azalduko ditugu.

### 80-biteko doitasuna.

Egungo IEEE-754 (2008) estandarrak, koma-higikorreko hiru formatu definitzen ditu: arrunta (32-bit), bikoitza (64-bit) eta laukoitza (128-bit). IEEE-754 (1985) estandar zaharrak, 80-biteko formatuaren (*double-extended* izenekoa) implementazioa aholkatzen zuen [94] baina egungo estandarretik kanpo utzi dute. 80-biteko doitasunak, 19 zifra hamartarreko doitasuna eskaintzen du.

C lengoaiaren *long double* datu motak, 80-biteko koma-higikorreko aritmetika sustatzen du. IA-32 koma-higikorreko eragiketa multzoak [91], 80-biteko zenbaki formatua onartzen du; 64-biteko mantisa eta 15-biteko esponentearekin. IA-64 eragiketa multzoak, 82-biteko formatu onartzen du; 64-biteko mantisa eta 17-biteko esponentearekin.

Intel x86 prozesadoreek, esaterako Intel Xeon Phi [27], doitasun arrunta eta bikoitza hardwarez implementatzen dute; laukoitza, softwarez; eta 80-biteko doitasuna ere hardwarez.

Laskar-ek [80, 79], eguzki-sistemaren epe luzeko zenbakizko integratorako, batura konpensatua eta 80-biteko koma-higikorra erabili zituen. Integratzaille simplektikoa C lengoian berridatzi zuen, Intel-en 80-biteko aritmetika erabili ahal izateko. Era berean, efemerideen kalkuluetarako antzekoa gertatu da: EPM [96] (aplikazioa berridatzi zuten) eta INPOP [40] efemerideen kalkulutarako, 80-biteko aritmetika erabiltzen dute.

Honek, astronomoentzat, soluzioen doitasuna hobetzeak duen garrantzia era-kusten digu. 128-biteko aritmetikaren implementazioa software bidezkoa da eta

erabiltzeko garestiegia. 80-biteko aritmetikarekin eta batura konpensatuarekin, zifra hamartar batzuk irabazten dituzte, konputazio kostu nahiko txikiarekin.

80-biteko doitasuna abantailak ditu baina doitasun hau erabiltzeak, hainbat errezero sortzen ditu. *IEEE-754* estandarretik kanpo dago eta hardwarearen menpekoa da. Gainera, Laskar-en [80] zenbakizko integracioen arabera, konputazioak  $x2$  garestitu zaizkio.

### **Doitasun nahasia.**

Azken urteetan, beharrezko doitasuna lortzeko, koma-higikorreko doitasun ezberdinak nahasteko joera dago.

Doitasun arrunteko aritmetikak (32-bit), doitasun bikoitzarekiko (64-bit)  $x2$  aldeko eraginkortasun erratioa du [32]. Aplikazio batuetarako, doitasun arrunteko doitasuna nahikoa da eta beste batuetarako, konputazioaren funtsa doitasun arruntean egin daiteke eta galdu tako digituak modu merkean berreskuratu.

Doitasun laukoitza (128-bit), software bidez modu errazean implementatu daiteke. Doitasun laukoitzeko zenbaki bat, doitasun bikoitzeko bi zenbakiren bidez adieraz daiteke. Ideia da, konputazioa doitasun bikoitzean egitea eta doitasun laukoitzean integrazioaren kalkulu zehatz batzuk egitea. Zenbakizko integracioetarako batura konpensatua, era honetan interpretatu daiteke.

Batura konpensatua merke da. Eguzki-sistemaren gure implementazioan batura konpensatua aplikatzeak abantaila berezia dakar: urrats bakoitzean gehitzen dugun balioa txikia denez, batura konpensatuarekin zifra hamartar asko mantentzen ditugu.

## **8.5. Aldagai aldaketa.**

### **Atalen hasieraketa.**

Kepler-en fluxuan oinarritutako aldagai aldaketa, atalen hasieraketa ona lortzeko aplika daiteke. Honako ekuazio diferenziala badugu,

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

non  $k(y)$  alde Kepleriarri dagokion eta  $g(y, t)$  perturbazioari. Alde Kepleriarren fluxua ezaguna dugu,

$$\begin{aligned}\varphi_{\Delta t}^k : \mathbb{R}^d &\longrightarrow \mathbb{R}^d \\ y_0 &\rightsquigarrow y_1.\end{aligned}$$

Kepler-en fluxuan oinarritutako aldagai aldaketa aplikatzen badugu,

$$\begin{aligned} y(t_0 + \Delta t) &= \varphi_{\Delta t}^k(z(t_0 + \Delta t)), \quad y(t_0) = z(t_0), \\ z(t_0 + \Delta t) &= \varphi_{-\Delta t}^k(y(t_0 + \Delta t)), \end{aligned}$$

aldagai berriarekiko ekuazio diferenziala mantso aldatzen den funtzioa da,

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

$z$  aldagai berria erabiliz, atalen hasieraketa era honetan egin daiteke (8.1. irudia):

1.  $Y_{n-1}$  atalei, Kepler-en fluxuan oinarritutako aldagai aldaketa aplikatu.  $Z_{n-1}$  aldagai berriarekiko atalak lortuko ditugu.

$$Z_{n-1,i} = \varphi_{-\Delta t}^k(Y_{n-1,i}), \quad i = 1, \dots, s.$$

2.  $Z_{n-1}$  atalak interpolatzuz,  $Z_n^{[0]}$  hasieraketak lortuko ditugu.

$$Z_n^{[0]} = G(Z_{n-1}, h)$$

3.  $Z_n^{[0]}$  ataletatik abiatuta aldagai aldaketa deseginez,  $Y_n^{[0]}$  hasieraketak lortuko ditugu.

$$Y_{n,i}^{[0]} = \varphi_{\Delta t}^k(Z_{n,i}^{[0]}), \quad i = 1, \dots, s.$$

Ohiko polinomio interpolatzailaren bidez hasieraketa ona izateko, urratsa txikia izan behar du. Teknika hau erabiliz, interpolazioaren errorea  $\mathcal{O}(\epsilon)$  mailakoa izango da eta urratsa handiagoarekin hasieraketa ona izango dugu.

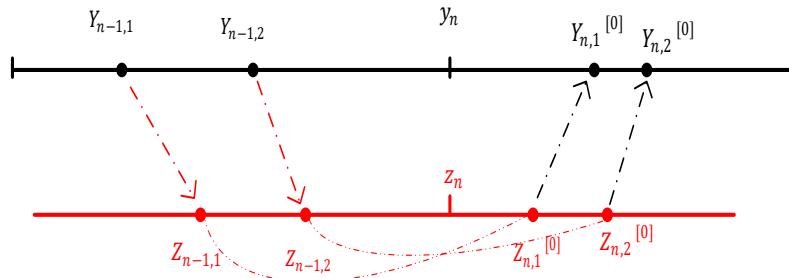
## 8.6. Paralelizazioa.

Hasierako baliodun problemen,

$$\dot{y}(t) = f(t, y(t)), \quad f : \mathbb{R}^{d+1} \longrightarrow \mathbb{R}^d, \quad y(t_0) = y_0, \quad (8.5)$$

integrazioetan, denboran zehar zenbakizko soluzioa sekuentzialki lortzen goaz eta beraz, ez dago paralelizazioa modu zuzenean aplikatzerik. Hala ere, hainbat teknika proposatu dira [21], eta etorkizunean, paralelizazio garapen berriak espero dira.

Saha eta Tremaineek [99], eguzki-sistemaren epe luzeko integrazioetarako, metodo paraleloa proposatu zuten. Konputazio paraleloa lortzeko, integrazio tarteen



### 8.1. Irudia: Atalen hasieraketa3.

(1-astea, 2-astea,...) integrazioak prozesadoreen artean banatzen dituzte. Ondoren, teknika hau garatuz, Laskar-ek [66] bere proposamen berria egin zuen. Hairer-ek [46], Hamiltondar sistemetan implementazio hauen analisia egin zuen eta aplikatzeko, hainbat eragozpen zeudela ondorioztatu zuen.

Ebatzi nahi den problemaren ekuazio diferentzialen balioztapena, konputazio-nalki garestia bada edota dimentsio handikoa, paralelizazioa beste maila batean aplika daiteke. Paralelizazioa aplikatzeko bide bat, dimentsio handiko ekuazio-sistemaren balioztapena, osagai independentetan banatzea eta hauen konputazioa, prozesadore batzuen artean modu paraleloan kalkulatzea da. Problema zurruna denean, metodo implizituak aplikatzen direnez, urrats bakoitzean, dimentsio handiko ekuazio-sistemak ebatzi behar dira eta horretarako, paralelizazioan oinarritutako implementazio (*LAPACK*) eraginkorrik ditugu. Era berean, N-gorputzeko problemetan,  $N$  handia denean, gorputzen arteko interakzioak ( $\mathcal{O}(N^2)$ ) kalkulatzeko paralelizazioan oinarritutako hainbat implementazio [11, 24, 33] aplika daiteke.

IRK metodoei dagokienez,  $f(Y_i)$ ,  $i = 1, \dots, s$  ataletako funtziotako balioztapena paraleloan exekuta daiteke. Eguzki-sistemaren eredu errealistetan,  $f$  funtzioa konplexua izango da eta beraz, paralelizazioa aplikatzeak abantaila izango du. Bestalde, doitasun bikoitzeko integrazioetarako, Gauss metodoaren implementazio estandar eraginkorrena,  $s = 6$  atalekoa kontsideratzen da [54]. Eguzki-sistemaren integratziorako gure implementazioa, ordea,  $s = 8$  ataletako metodoa eraginkorra da eta honek, paralelizazioak abantaila handiago suposatzen du.

## 8.7. Implementazioaren aplikagarritasuna.

Implementazioa, eguzki-sistemaren doitasun altuko epe luzeko integrazioetan aplikatzearaz gain, beste N-gorputzeko problema batzuetan aplikatzeko eraginkorra izan daitekeen hausnartuko dugu.

### Efemerideak.

Eguzki-sistemaren efemerideak, eguzki-sistemaren eredu konplexuak eta epe tarte txikietarako (ehunka urtekoak) integrazioak konputatzen dituzte. Konputagailuen aurreko garaian, efemerideak teoria analitikoetan oinarritutako serie funtzioen bidez kalkulatzen ziren. Soluzio hauetan, Fourier-en serie trigonometriko luzeen ebaluazioa egin behar zen. 1960 hamarkadan, eguzki-sistemaren ezagutza hobetu zenean (espazio bidaiaiak eta behatoki astronomikoen aurrerapenak medio), serie oso luzeak kalkulatu behar zitzuten, eta orduan zenbakizko integrazioen metodoak, eraginkorragoak bilakatu ziren [70].

Eguzki-sistemaren gorputzen efemeride modernoak, aldaketa adierazten duten (3.5) ekuazio differentzialen zenbakizko integrazioaren bidez kalkulatzen dira. Integrazioaren hasierako balioak eta ereduaren parametroak, sateliteen bidez jasotako datuekin kontrastatu egiten dira.

Efemerideak, Chebyshev polinomio moduan adierazten dira. Zenbakizko integrazio hauetan, biribiltze errorea gai garrantzitsua da. 128-biteko aritmetikaren aukera baztertzen da, konputazioa oso garestia delako eta 64-biteko doitasuneko aritmetika hobetzen dituzten teknika konputazionalki merkeagoak, aplikatzen dira.

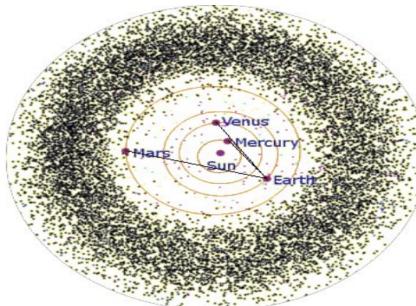
Efemerideetarako, eguzki-sistemaren eredu konplexua aplikatzen da. Gorputz nagusien arteko indar gravitacionalez gain, erlatibilitatea, asteroideek era gindako gravitazio indarrak, gorputzen formen eragina eta beste hainbat indar ez gravitacionalak kontutan hartzen dituzte. Mugimenduaren ekuazio differentzialak, era honetakoak izaten dira [39],

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

Hauek dira, ekuazio hauen konplexutasuna erakusten duten ezaugarri batzuk:

- Gorputz kopurua: 8 planetak, ilargia, Pluton eta 300 asteroide. Asteroideek, bereziki Marte planetaren mugimenduarengan dute eragina (8.2. irudia) eta kontutan hartzekoak dira, barne-planeten mugimenduaren doitasun handiko emaitzak behar ditugunean. Bost asteroide nagusiren masak (Ceres, Pallas, Vesta, Iris eta Bamberga) Merkurio eta Pluton planeten mailakoak direnez, integrazioetan gehitzen dira. Gainontzeko asteroide txikien talde handiak, estimazioen bidez simulatzen dira.

- Erlatibilitatea efektua (GR): Einstein-Imfeld-Hoffmann,  $c^{-4}$  PPN hurbilketa.
- $J_2$ : eguzkia esferikoa ez izatearen eragina.
- Integrazioaren urrats luzera,  $h = 0.055$  egunekoa da.



**8.2. Irudia:** Asteroideak.

### Hiru efemerideak.

Gaur-egun, eguzki-sistemaren planeten hiru efemeride kalkulatzen dira.

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

1984.urtean kalkulatu zen lehen efemeridea (DE-200) eta 2.014.urteko *DE-430* [41] efemeridea publikatutako azkena da. Azken efemeride honen integracio tarteak, 1550 – 2650 artekoa izan da.

Zenbakizko integrazio metodoa, urrats luzera eta ordena aldakorreko *Multistep Adams* metodoa [75] (*DIVA/QIVA*) da. *QIVA* doitasun laukoitzeko (128-bit) bertsioari deitzen zaio: mugimenduaren ekuazioen Newton zatia, doitasun laukoitzean kalkulatzen da eta ekuazioaren gainontzeko zatia, doitasun bikoitzean.

2. Institut de Méchanique Céleste et de Calcul des Ephémérides (IMCCE, Paris Observatory) INPOP (Intégrateur Numerique Planétaire de l'Observatoire de Paris) izeneko efemerideak.

2.000.urteraino, efemerideen kalkulua teori analitikoetan oinarritzen zituzten. 2.003.urtean, zenbakizko integrazioaren bidezko lehen efemeridea kalkulatu zuten eta *INPOP13c* efemeridea [40, 2.014] publikatutako azkena da.

Zenbakizko integrazio metodoa: 12 ordenako *Adams-Cowell* metodoa da eta urrats finkoa aplikatzen dute.

Doitasuna: C lengoiaian implementatuta dago eta *Intel* makinetako 80-biteko doitasuna erabiltzen du [40].

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 *EPM2013* efemeridea [96, 2.014] publikatutako azkena da.

Zenbakizko integrazio metodoa, *Everhart* izeneko *IRK* metodoa (Gauss-Radau) da. 23 ordenako metodoa eta urrats luzera finkoa aplikatzen dute.

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

## Satelite artifizialen problemak.

Satelite artifizialen problemak, gure implementazioaren aplikagarritasunaren adibide argia dira. Problema hauetan, eredu matematiko konplexuak erabiltzen dira, konputazionalki garestiak, eta sateliteen kokapenaren eboluzioa, zehaztasun handiarekin ezagutu behar da. Ondorioz, integrazio metodo eraginkorrak eta ordena altuko metodoak aplikatu behar dira [13].

## Exoplanetak.

Eguzki-sistemari egokitutako splitting metodo simplektikoak [113, 82], eguzki-sistemaren problemaren bi ezaugarriei abantaila atereaz diseinatzen dira. Lehenik, planeten eguzkiarekiko orbitak ia Kepleriarrak dira. Bigarrenik, N-gorputzen problemaren formulazio Hamiltondarrean oinarritzen dira eta simplektikoak direnez, Hamiltondarra zehazki mantentzen dute.

Exoplaneta sistemak integratzeko metodo berriak behar dira [36]. Exoplaneten planeta-sistemen mugimenduak ez dira eguzki-sistemaren bezain egonkorrik. Exoplaneta sistemetan, eszentrikotasun handiko orbitak, periodo txikikoak, eta izar anitzeko planeta-sistemak aurkituko ditugu. Splitting metodo simplektikoak, ez dira ondo egokitzen era honetako sistemak integratzeko. IRK Gauss kolokazio metodo orokorrean oinarritutako gure implementazioa, edozein sistemarako egokia da. Eguzki-sistemaren ekuazioei aldagai aldaketa modu egokian aplikatuz metodoa lehiakorra da gurea eta exoplaneta-sistemak integratzeko ere, baliagarria izan daitekeela pentsatzen dugu. Dena den, exoplaneta sistemaren datuen doitasuna txikia da eta beraz, ez du zentzu handirik doitasun handiko integrazioak egitea.

## Molekulen sistema dinamikoak.

Molekulen dinamikaren problema, Hamiltondarra da eta ekuazio diferentzialak era honetan idatzi daitezke,

$$\begin{aligned}\dot{q}_i &= v_i, \\ \dot{v}_i &= g(q_i), \quad i = 1, \dots, N,\end{aligned}$$

non  $N$  partikulen kopurua eta  $q_i, v_i \in \mathbb{R}^3$  diren.

Molekulen sistema dinamikoen integrazioak konputazionalki garestiak dira,  $N$  balioa handia delako,  $g$  funtzioaren konplexutasunagatik eta integrazioa epea luzea izaten delako. Molekulen sistema dinamikoen integrazioetarako, *Störmer-Verlet* (2.21) da metodo nagusienetakoa. Sanz Sernak [86], molekulen sistema dinamikoen integratorako Gauss metodoen eraginkortasuna aztertu zuen, eta *Störmer-Verlet* metoda eraginkorragoa zela ondorioztatu zuen. Molekulen sistema dinamikoen integrazioetarako gure planteamenduak, ez duela ekarpenik egiten dirudi.

## **9. Kapitulua**

### **Ondorioak.**

Gure helburua, eguzki-sistemaren doitasun altuko eta epe luzeko integratorako, Gaussen metodo implizituaren implementazio eraginkorra lortzea da. Helburua lortu dugun ala ez galderari erantzuteko, balorazioa bi ikuspegietatik eman behar dela iruditzen zaigu. Batetik, lortutako implementazioaren aldeko argumentuak azalduz, eraginkorra dela defenditu daiteke. Bestetik, eraginkortasuna, implementazioak etorkizunean izan ditzakeen hobekuntzen arabera neurtu beharko litzateke.

Gakoetako bat, integratu nahi den eguzki-sistemaren ereduaren konplexutasuna da. Eguzki-sistemaren ezagutza gero eta zehatzagoa denez, eredu gero eta konplexuagoak integratuko direla suposatu daiteke. Zentzu honetan, integrazio metodoa problema konplexuen integratorako eraginkorra izan beharko luke. Edozein kasutan, argi dago, integrazio metodoak problemaren formulazio matematikoa ez duela mugatu behar eta askatasuna behar dela, ekuazioetan nahi diren efektuak gehitzeko.

Algoritmoen eraginkortasunaren balorazioa, konputagailu hardware ingurune bati lotuta dago. Ezaguna da, egungo konputagailuen ahalmen handia, konputazio paraleloan oinarritzen dela eta algoritmo eraginkorrak garatzeko, idei berriak aplikatu behar direla. Ausartuko ginateke esaten, Gauss metodo implizituak, implementatzeko aukera asko eskaintzen dituenez, egungo idei berri hauek aplikatzeko bidea errazten duela [32].

#### **Eguzki-sistemaren integratorako implementazioa.**

Eguzki-sistemaren integratorako Gauss-en metodo orokorra aplikatu dugula azpimarratu behar dugu. Hori dela-eta, implementazioa simplea da eta implementazioak, metodoaren ezaugarri on guztiak heredatzen ditu. Azken finean, puntu-finkoaren iterazioan oinarrtitako IRK implementazio estandarra aplikatu dugu,

formulazio eta geratze irizpide bereziekin. Epe luzeko integrazioetarako, metodoa simplektikoa izatea garrantzitsua da; IRK metodoaren formulazio estandarrak ez du simplektikotasuna ziurtatzen, eta horregatik, IRK metodoa aplikatzeko formulazio berria proposatu genuen. Modu beraean, geratze irizpide berria implementazio estandarrarena hobetzeko aplikatu dugu. Beraz, implementazio berri batek izan ditzakeen konplexutasunak ekidin ditugu.

Kepler-en fluxuan oinarritutako aldagai aldaketa, ebatzi nahi dugun problema-ri lotuta dago. Aldagai aldaketa hau, teknika orokor baten aplikazioa da; ekuazio differentzialetatik zati bat desagertzeko helburuarekin definitutako aldagai aldaketa aplikatzea. Aldagai aldaketa honekin, ekuazio differentzialetatik alde Kepleriarrari dagokion zatia desagertzea lortzen dugu eta mantso aldatzen den ekuazio differentzialak lortzen ditugu. Aldagai berriekiko ekuazio differentzialak, hiru abantaila ditu. Lehenik, eguzki-sistemaren trunkatze errore nagusiena ezabatu dugunez, integrazioan urrats luzera handiak erabili daitezke. Bigarrenik, integrazioaren urra-tsaren konputazioaren baturan,

$$(\tilde{y}_{n+1}, e_{n+1}) \leftarrow \text{batura konpensatua}(\tilde{y}_n, e_n, \delta_n),$$

$\delta_n$  magnitude txikia du eta beraz, batuketa honetan informazio gutxiago galdu-ko dugu. Gainera,  $\delta_n$  doitasun altuan kalkulatzeak abantaila areagotu egiten du. Hirugarrenik, puntu-finkoaren konbergentzia azkarra da.

Eguzki-sistema eredu konplexuetan, Kleperiarrak ez diren indarrak aplika dai-tezke. Aldagai hauen konbergentzia azkartzeko, Newton simplifikatuaren itera-zioan oinarritutako implementazioa aplikatzea komeniko da.

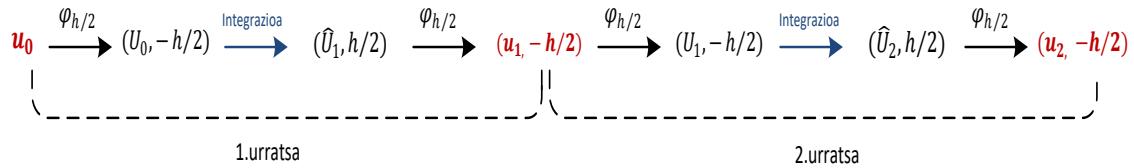
Gauss metodoa, neurri batean Splitting eta konposizio metodoen baliokideak dira.

Konposizio metodoa  $\equiv$  Gauss metodoa aldagai aldaketa gabe.

Splitting metodoa  $\equiv$  Gauss metodoa aldagai aldaketarekin.

Splitting metodoekiko antzekotasuna azaltzeko, (2.21) *Störmer-Verlet* Splitting metodoarekin konparatuko dugu. *Störmer-Verlet* metodoa, era honetan apli-katzen da:  $h/2$  fluxua aplikatu, perturbazioak kalkulatu eta berriz,  $h/2$  fluxua aplikatu. Gauss metodoan, Kepler-en fluxuan oinarritutako aldagai aldaketa apli-katzen dugunean, gauza bera egiten ari gara:  $h/2$  fluxua aurreratu, perturbazioak kalkulatu (aldagai berrieta eta beraz, hobeto kalkulatzen dugu),  $h/2$  fluxua au-rreratu (9.1. irudia).

Splitting metodoak oso eraginkorrak dira eta eguzki-sistemaren eredu simplearen integrazioen esperimentuek, hori erakutsi digute. Baino eredu errealistagoak (gorputz kopurua handitzen delako edota erlatibilitate efektua kontutan hartzen delako) integratzeko, Kepler fluxuan oinarritutako Gauss metodoak, Splitting metodoekiko abantailak hauek izango dituela azpimarratuko dugu:



**9.1. Irudia:** Gauss metodoan, Kepler-en fluxuan oinarritutako aldagai aldaketa aplikatzenten dugunean, urrats baten konputazioa

1. Ordena altuko metodoak.

Konposizio eta Splitting metodo optimoenak,  $p = 10$  ordenakoak dira. Gauss metodoei dagokienez, edozein ordenako metodoak eraiki daitezke. Adibidez, Gaussen  $s = 16$  atalaetako metodoa ( $p = 32$  ordenako metodoa) eta 80-biteko doitasuneko koma higikorra aplikatzen baditugu, implementazioaren zati batzuk 128-biteko doitasunean kalkulatuz, soluzioaren doitasuna hobetuko dugu.

2. Paralelizagarria.

Gauss metodoaren  $s$ -ataletako funtzioen ebaluazioak independenteak dira eta modu paraleloan exekutatu daitezke. Eguzki-sistemaren eredu konplexuagoa kontutan hartzen den neurrian, paralelizazioak abantaila handiagoa suposatuko du. Gauss metodoaren paralelizazio gaitasun hau azpimarratzeko da, egungo algoritmo azkarren diseinua, kodearen paralelizazioan oinarritzen baita.

3. Hamiltondar orokorrak.

Gauss metodoa, edozein sistema Hamiltondarri aplika dako. Splitting metodoa, ordea, sistema Hamiltondar banagarrietan bakarrik aplika daitezke; eguzki-sistemaren eredu errealistak integratzeko, Hamiltondarraren egitura mantendu behar da eta hainbat indar ez grabitacionalak gehitzeko, zailtasunak izan ditzakegu.

4. Malgutasuna.

Splitting metodoen konputazioak, modu trinkoan kalkulatu behar dira, hau da, atalen konputazioak sekuentzialki exekutatzen dira eta metodo esplizituak ez ditu aldaerak onartzen. Gauss metodoaren ekuazio implizituak ebazteko, ordea, teknika ezberdinak konbina daitezke eta eraginkortasuna ho-

betzeko aukera asko eskaintzen dizkigu. Adibidez, iterazio gehienak problemaren eredu simple batekin, doitasun baxuan kalkula daitezke [13] eta bukaerako iterazio pare bat eredu osoarekin, doitasun altuan.

### 5. Birparametrizazioa.

Birparametrizazio, eguzki-sistemaren integrazioetarako tresna baliagarria dela frogatu da [45, 97] eta Gauss metodoan, teknika hau zuzenean aplikatzen daiteke. Splitting metodoetan, ordea, birparametrizazioa ezin daiteke erabili eta integrazioetan zailtasunak agertzen direnean (esaterako, planeta baten eszentrikotasuna handitzen denean), orduan integrazioaren tarte batean urratsa txikitu behar dute soluzioaren doitasuna mantentzeko [81].

## Etorkizuneko lanak.

Etorkizunerako, hobekuntza aukera asko izan ditzake:

1. Eredu deskonposaketen aplikatutako teknikak.
2. Oszilazio teknikak errorea txikitzea.
3. Kepler fluxuaren hobekuntzak.
  - Kepler-en fluxua: orain iterazio guztietai hasieratik aplikatzen dugu, aurreko iterazioen informazioa erabili gabe. Iterazio berri batean, aurreko iterazioaren egoeratik abiatuta, nahikoa izango da fluxuaren iterazio bakarra egitea.
  - Kepler fluxuaren implementazioa h txikitarako hobetu daiteke.
  - Paralelizazioa eta bektorizazioa.
4. Eredu konplexuak. Zenbat eta gorputz gehiago izan edo erlatibilitate efektua,... Era honetako sistemak ditugunean, lehen iterazioetan eredu simplearekin iteraziok eman daitezke eta azken iterazio pare bat eredu konplexuarekin.

**I. Atala**  
**Eranskinak**



## **A. Eranskina**

### **Konputazio zientzia.**

#### **A.1. Sarrera.**

Azken hamarkadetan, konputazio zientziaren hazkundeak oso handia izan da eta bere erabilera ia zientzia arlo guztieta zabalduta dago [48]. Zientzialariek ahalmen handiko tresna berria (zenbakizko simulazioa) eskuragarri dute, neurri handi batean konputagailuen teknologiaren garapen handiari esker. Egungo oinarrizko konputagailuek, orain urte gutxitako superordenagailuen ahalmen berdina dute eta superordenagailuen konputazio gaitasuna ere, maila berdinean hazi da. Zenbakizko algoritmoek ere, garapen handia izan dute; algoritmo eraginkorragoak, idei berriak sortuz eta konputagailuen gaitasunei egokituz, garatu dira.

Konputazioaren alde garestiena, memoria eta prozesadorearen arteko datu mugimendua da. Prozesadoreak gero eta azkarragoak dira, baina memori atzipenaren abiaduraren hobekuntza mugatuagoa dago. Horregatik algoritmo eraginkorrapak, prozesadorearen konputazio arik eta handiena, memoria komunikazio arik eta txikiarenarekin, diseinatu behar dira.

Implementazio baten eraginkortasuna ez da exekuzio denboraren arabera bakanrik neurtu behar. Hori bezain garrantzitsua da kode ona idaztea [110] eta zentzu honetan hiru ezaugarri hauek bereziki zaindu behar dira:

1. Errorerik gabeko kodea.
2. Kode argia idaztea.
3. Etorkizunean erraz aldatu daitekeen kodea.

Atal honetan, konputazio zientziaren funtsezko osagaiak azalduko ditugu. Lehenengo, konputazioaren eraginkortasuna aztertzeko eta exekuzioak neurtzeko tresnen zehaztapenak emango ditugu. Ondoren, konputagailu hardware berriak eta paralelizaio gaitasunak laburtu dugu. Jarraian software ikuspegitik, programazio

lengoaiak eta aljebra linealerako (LAPACK) liburutegia landu dugu. Azkenik, konpiladoreari buruzko argibide batzuk eman ditugu.

## A.2. Eraginkortasuna

Konputazio zientziaren implementazio berri baten eraginkortasuna neurtzeko, koma-higikorreko eragiketa kopurua (*flops*) erabili ohi da. Problema handia denean, datuen mugimendua koma-higikorreko eragiketak baino garestiagoa da eta era-ginkortasuna, eragiketa kopuruaren arabera neurtzea okerra izan daiteke.

Prozesadoreen maiztasun-abiadura hertzetan neurtzen da, hau da, *makina ziklo segundoko* kopuruaren arabera. Une honetako prozesadoreak gigahertz (Giga =  $10^9$ ) mailakoak dira. Koma-higikorreko oinarrizko eragiketa bat ( $\oplus, \ominus, \otimes, \oslash$ ) exekutatzeko ziklo gutxi batzuk behar dira eta beraz, 1 GHz-ko prozesadore batek  $> 10^8$  koma-higikorreko eragiketa segundoko exekutatzen ditu ( $> 100$  megaflops) [95].

**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 bakoitza 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 genitzuke. Baino matrize honek, 3.9 MB memoria beharrezkoa du eta konputagailuaren Cache memoria baino handiagoa dela suposatuz, exekuzio denboran datuen mugimenduaren eragina nabarmena izango da.

Konputazio gaitasuna (*peak*), hardwareak fisikoki exekutatu dezakeen eragiketa kopuru maximoa bada, aplikazio gehienak, konputagailuaren konputazio gaitasunaren %10 baino gutxiagorekin exekutatzen dira. Eraginkortasun horren txikia, memoria irakurketa/idazketetan galtzen da. Azpimarratu nahi dugu,  $t_f$  eragiketa aritmetiko bat egiteko denbora bada eta  $t_m$ , datu bat memoria nagusitik cache memoriara mugitzeko denbora bada,

$$t_f \ll t_m,$$

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

### Exekuzio denboren neurketa.

Unixeko *time* agindua, konputazioen denborak ezagutzeko erabili daiteke [95]:

```
S time ./a.out
<kodearen irteera >

real 0m38.856 s
user 0m38.789 s
sys 0m0.004 s
```

Agindu honekin, *./a.out* C programa exekutatuko da eta ondoren, programa exekutatzeko behar izan duen denboraren informazioa pantailaratuko du:

- *real*: hasi eta bukatu arteko denbora (*wall-time* edo *elapsed-time*).
- *user*: prozesadoreak gure programa exekutatzen erabili duen denbora (*CPU-time*).
- *sys*: programa exekutatu ahal izateko, sistema eragile lanetan emandako denbora.

Programa osoaren konputazio denborak ezagutu beharrean, kodearen zati bat neurtu nahi dugunean, C lengoaiaren bi funtzio hauek erabilgarriak ditugu:

1. *clock()*.

Funtzioaren bi deien arteko CPU denbora neurtzeko erabiliko dugu (**CPU time**).

```
#include <time.h>

clock_t clock0, clock1;
double elapsed_cpu_time;

clock0= clock();

<neurtu nahi den kodea>

clock1=clock();

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

2. `time()`.

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

```
#include <time.h>

time_t time0, time1;
double elapsed_time;

time(&time0);

<neurtu nahi den kodea>

time(&time0);

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

CPU denborari buruzko argibide bat ematea komeni da. Neurtzen ari garen kodea sekuentzialki exekutatzen bada, hau da, hari (*thread*) bakarrekoa, orduan kode zati hori exekutatzeko erabili duen CPU denbora itzuliko du. Aldiz, kodea paraleloan exekutatzen bada, orduan , hari guztien CPU denboren batura itzuliko ditu.

**Adibidea.** Argiago azaltzeko,  $C = AB$  matrize biderketaren kodearen bi exekuzioen denboren neurketak zehaztuko ditugu:

1.  $200 \times 200$  tamainako, bi matrizeen biderketa sekuentzialaren denborak hauek izan dira:

```
elapsed-time=2.1 s
elapsed-cpu-time=2.07 s
```

2.  $200 \times 200$  tamainako, bi matrizeen biderketa paraleloaren (hariak=2) denborak hauek izan dira:

```
elapsed-time=1.0 s
elapsed-cpu-time=2.35 s
```

*Elapsed-time* deiturikoa, kode paraleloen denborak neurtzeko irizpidea da. Programazio paraleloan, algoritmoen exekuzio denborak egokien neurtzen duen aldagaia da baina, une berean programa bakarra exekutatzea behartuta gaude.

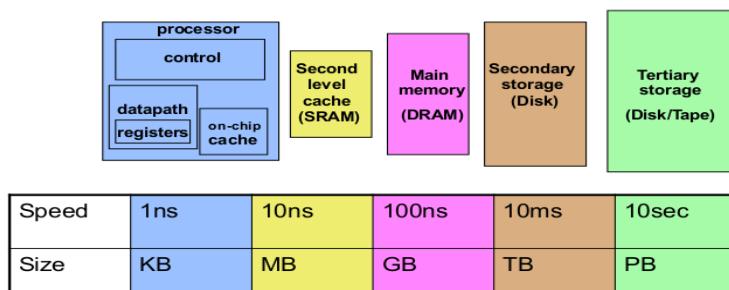
## A.3. Hardwarea.

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

Konputagailuen eredu aldaketa honen ondorioz, algoritmo azkarrak garatzeko kodearen paralelizazio gaitasunari heldu behar zaio. Programazio paralelo teknikak implementatzeko, beharrezko da prozesadore berrien hardware arkitektura berriak ulertzea. Gaia nahiko konplexua izanik, ikuspegi orokor bat ematera mugatuko gara.

### Memori hierarkia.

Memorien arteko datuen komunikazioak, algoritmoaren eraginkortasuna baldintzatuko du eta zentzu honetan, konputagailuaren memoria hierarkiaren kudeaketa egokia egitea funtsezkoa da. Konputagailuaren memoria mota ezberdinaren hierarkia ([A.1.irudia](#)) eta funtzionamendua deskribatuko dugu.



**A.1. Irudia:** Memoria hierarkia.

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

CPU-k datu bat behar duenean, memoria hierarkian zehar bilatuko du: lehenik  $L1$  cachean, ondoren  $L2$  cachean,...eta hauetan ez badago, memoria nagusira joko du. Memoria nagusi eta cache memoria arteko irakurketa eta idazketa guzti hauetan, informazio konsistentzia mantentzeko hainbat arau aurrera ematen dira.

Cache memoria lerroka egituratuta dago eta lerro bakoitza 64 edo 128 bytetz (8 edo 16 doitasun bikoitzeko zenbaki) osatuta dago. Programa batek datu bat behar duenean, memoria nagusitik lerro tamainako datu taldea (memorian jarraian gordetako datuak) irakurriko du eta cachean idatziko du. Programatzaleak, algoritmo eraginkorrik implementatzeko memorien arteko komunikazio hau minimizatzen saiatu behar du eta horretarako, implementazioaren diseinua datuen memoria atzipen jarraian oinarritu behar du. Ezaugarri hau, *spatial/data locality* izenaz ezaguna da eta helburua, cacheria ekartzen diren datuak, memoria nagusian idatzi aurretik gutxienez behin erabiltzea da.

**Adibidea.** Adibide honetan,  $A = (a_{ij})_{i,j}^{n,m}$  matrize baten osagaien batura ( $\text{sum} = \sum_{i,j=0}^{n,m} a_{ij}$ ) kalkulatzeko bi implementazio aztertuko ditugu. C lengoainen matrizeak lerroka gordetzen dira ( $n = m = 100$ ),

$$A = \begin{pmatrix} 1 & 2 & 3 & \dots & 100 \\ 101 & 102 & 103 & \dots & 200 \\ 201 & 202 & 203 & \dots & 300 \\ \dots & \dots & \dots & \dots & \dots \\ 9.901 & 9.902 & 9.903 & \dots & 10.000 \end{pmatrix}.$$

eta horregatik, lehen aukera bigarrena baino eraginkorragoa izango da. Lehen implementazioan, kanpo iterazioa lerroka (Algoritmoa 29): matrizearen lehen osagaia  $a(1, 1)$  behar dugunean, memoria nagusitik Cacheria osagai honetaz gain, jarraiko 16 osagaia ekarriko dira ( $a(1, 1), a(1, 2), \dots, a(1, 16)$ ). Honela, hurrengo 15 batura egiteko behar ditugun datuak Cachean eskura izango ditugu memoria irakurketa berririk egin gabe. Bigarren implementazioan, kanpo iterazioa zutabeka (Algoritmoa 30): bigarren osagaia ( $a(2, 1)$ ) gehitzeko memoria irakurketa berri bat egin behar dugu.

## Arkitektura motak.

Prozesadore anitzeko konputagailu hardware berriak, konplexuak eta heterogeneoak dira. Egoera honetan, programatzaleak zaitasun handiak ditu arkitektura berrieik eskaintzen dituzten gaitasunak ondo kudeatzeko [87].

Lehen hurbilpen modura bi sistema nagusi bereiziko ditugu: memoria konpartitutako eta memoria banatutako sistemak. Memoria konpartitutako sistemetan,

```

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

```

**Algoritmoa 29:** Memoria atzipena eraginkorra.

```

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

```

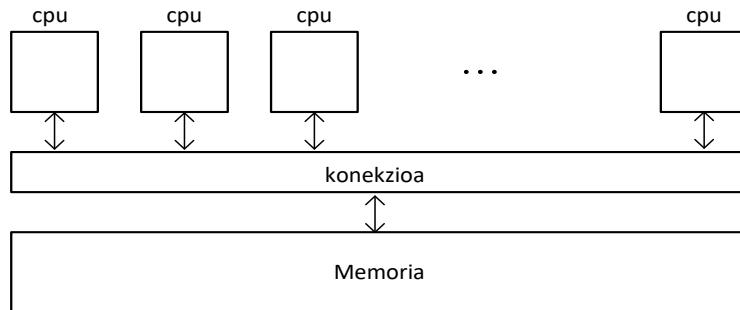
**Algoritmoa 30:** Memoria atzipena ez-eraginkorra.

prozesadore guztiekin memoria osoa konpartitzen dute eta implizituki konpartitutako datuen atzipenaren bidez komunikatzen dira ([A.2.](#) irudia). Memoria banatu-takotako sistemetan aldiz, prozesadore bakoitzak bere memoria pribatua du eta esplizituki bidalitako mezuen bidez komunikatzen dira ([A.3.](#) irudia). Aipatzeko da, sistema handietan bi memoria motak nahasten direla, hau da, batetik goiko mailan memoria banatuta alde bat eta bestetik, konputazio unitate bakoitzak memoria konpartitutako aldea.

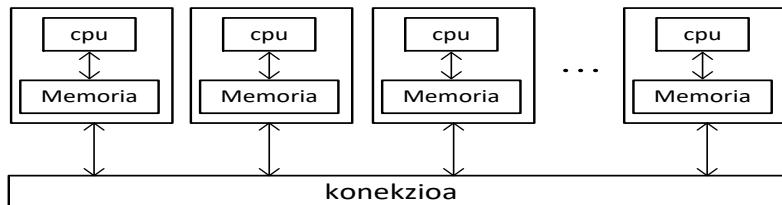
Hirugarren sistema osagarria ere aipatuko dugu, GPU (Graphical Processor Unit) unitateetan oinarritutako konputazioa. Joko-en eta animazio industrian, gra-fiko oso azkarrak beharrak bultzatuta sortutako teknologia da. Oinarrian, imajinak pantailaratzeko prozesagailu asko paraleloan lan egiten dute eta azken hamarkadan, *GPU* unitate hauek konputazio zientziara zabaldu dira.

### Oinarrizko konputagailu paraleloa.

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



**A.2. Irudia:** Memoria konpartitutako sistemak.

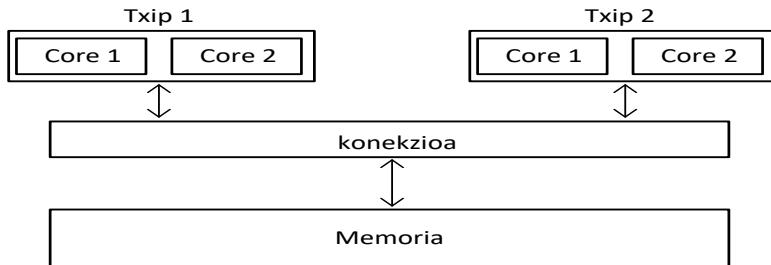


**A.3. Irudia:** Memoria banatutako sistemak.

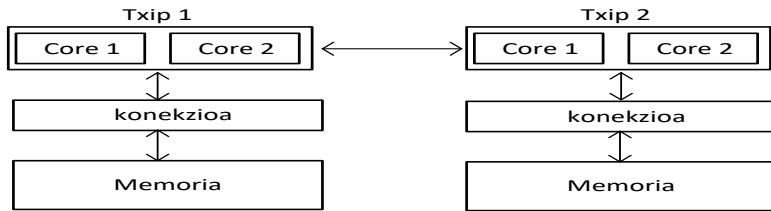
[95, 109].

Memoria konpartitutako sistemen artean bi mota bereiziko ditugu:

1. UMA sistemak (uniform memory access). Txip prozesadore guztiak zuzean memoria konektatuta daude eta guztiak atzipen denbora berdina dute ([A.4..irudia](#)).
2. NUMA sistemak (nonuniform memory access). Txip prozesadore bakotza, hardware berezi baten bidez zuzenean memoria bloke batikonektatuta dago. Zuzenean konektatuta dagoen memori blokearen atzipen denbora, beste txipan zehar konektatutako memoriaren atzipena baina azkarragoa da ([A.5..irudia](#)).



**A.4. Irudia:** Memoria konpartitutako sistemak (UMA).



**A.5. Irudia:** Memoria konpartitutako sistemak (NUMA).

### Bektorizazioa (SIMD).

Prozesadore berriei hardware bektore unitateak gehitu zaizkie, eta (AVX) bektore instrukzioetan oinarritutako paraleлизazioa eskaintzen dute [91, 5]. CPU-ak bektore erregistroan gorde diren zenbaki multzo bati, eragiketa berdina aplikatzen die (*Single Instruction Multiple Data*, SIMD). Bektore erregistro hauek 512-biteko tamainakoak ( $512/64 = 8$  doitasun bikoitzeko zenbaki) izan daitezke eta oro har, oinarrizko eragiketa aritmetikoak (batuketa, kenketa, biderketa eta zatiketa) aploka daitezke. Zentzu honetan ulertu behar da hardware bidezko bektorizazioak, doitasun bikoitzeko konputazioen eraginkortasuna  $x8$  hobetzen duela eta doitasun arrunteko konputazioetan, berriz,  $x16$ .

Adibide moduan, hurrengo pseudokodearen bidez, iterazio bakoitzean 8 osa-

gaien bektore baten batura erakutsiko nahi dugu,

```
for  $i = 0; i < n; i+ = 8$  do
     $A[i : (i + 8)] = A[i : (i + 8)] + B[i : (i + 8)];$ 
end
```

**Algoritmoa 31:** SIMD (bektorizazioa).

## A.4. Programazio lengoaiak.

Fortran eta C, aplikazio zientifikoetan gehien erabiltzen diren programazio lengoaiak dira [60]. Fortran (formula translation) 1950 hamarkadan garatutako goi-mailako lehen lengoia izan zen eta oraindik ere, oso zabaldua da. Fortran estandarraren hainbat bertsio sortu dira: Fortran 66, 77, 90, 95, 2003 eta 2008. Hauetako bertsio bakoitzean funtzionalitate berriak eta C lengoaiarekin lan egiteko bateragarritasuna gehitu zaizkio. C lengoia, 1970 hamarkadan jaio zen eta hardwarearekiko hurbiltasun "ezaugarri nagusiak, konpiladoreari kode eraginkorra sortzeko aukera ematen dio. C lengoia UNIX sistema eragileari lotuta jaio zen eta hurrengo garapenetan bere izaera askea mantendu du. C lengoaiaren estandarrak 1989, 1999 eta 2011 dira.

Fortran eta C lengoaiak, ez dituzte kodea paraleloan exekutatzeko tresnarik, hau da, ez dago konputazio banatu, eta prozesadore ezberdinaren artean aldi berean exekuzioak zehazteko modurik. Konputazioa paraleloa implementatzeko, bi dira interfaze aplikazio programa (*API*) moduan implementatuta dauden sistema nagusiak [95]:

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

Eraginkortasun altuko konputazioaren (*high performance computing*) programazioa konplexua da: espezializazio handikoa eta konputagailuen hardware jakin baterako egokitutakoa. Zaitasun hauek, proiektu zientifikoak aurrera ateratzeko eta mantentzeko arazo asko eragiten ditu. Azken urteotan, eragozpen hau gainditzeke, programazio lengoia interesgarriak sortu dira (adibidez, Julia [14] edo Chapel [10]) baina oraindik, hauen arrakasta ikustekoa da.

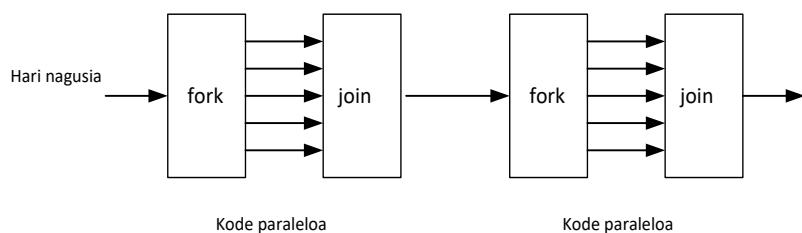
Azkenik, konputazio zientzian *problemak ebazteko inguruneak* (Problem Solving Environments) deituriko softwareak aipatuko ditugu. Ingurune hauek, programazio leihoko interaktibo batean, goi-mailako lengoai batean implementazioen gara-

pena eta emaitzen azterketa egiteko aukera eskaintzen dute. Matlab eta Mathematica [114] programazio ingurune nagusienak dira. Guk Mathematica bi modutara erabili dugu. Lehenik, prototipoak garatzeko tresna gisa: idei berriak garatu eta probatu, implementazioa C lengoian egin aurretik. Bigarrenik, gure C implementazioen esperimentuak Mathematica ingurunetik exekutatu eta emaitzak grafikoki aztertu ditugu. Era honetan, irakurleari Mathematicako dokumentuetan esperimentuen zehaztasun guztiak eta esperimentu berdina errepikatzeko aukera ematen diogu.

## OpenMP

*OpenMP* [2] memoria konpartitutako sistemetan programazio paraleloa exekutatzeko interfaze aplikazio programa (*API*) da. *OpenMP* programazioan, memoria osoaren atzipena duten prozesadore multzo batek osatzen du konputazio sistema.

Hasieran programaren hari bakarra prozesadore batean exekutatuko da, kode-paraleloko unera iritsi arte. Orduan, hari multzo independenteak exekutatuko dira une paraleloa bukaera iritsi arte. Exekuzio kontrolari, *fork-join* eredu deitzen zaio eta grafikoki (A.6. irudian) adierazi dugu.



**A.6. Irudia:** OpenMp programazio eredu.

- OpenMP programen hasieran prozesu bakarra dago, hari (thread) nagusia.
- FORK: hari nagusiak, hari talde paraleloa sortzen du.
- JOIN: kode-paraleloko hari guztiak bukatzen dutenean (sinkronizazioa), hari nagusiak soilik jarraitzen du.

Paralelizazioan hari kopurua zehaztu behar da, eta ohikoa izaten da hari bat prozesadore bakoitzeko sortzea. Konpilazio direktiben bidez (C kodean *pragma* izeneko preprozesadore aginduak), paralelizazioa nola exekutatu behar den zehazten da.

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

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

```
# include <omp.h>

int thread_count=2;

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

Konpilazioan, *-fopenmp* aukera zehaztu behar dugu,

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

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

Algoritmo batean, kodearen zati bat da paralelizagarria [95]. Suposa dezan gun, konputazioaren %50 sekuentziala dela eta beste %50 paraleloan exekutatu daitekeela. Zati sekuentzialak, lortu daitekeen konputazio optimoena (zati paralelizagarriaren exekuzio denbora zero dela konsideratzea) mugatuko du eta beraz, gehienez exekuzio sekuentziala baino bi aldiz azkarrago izango da. Kontzeptu hau orokortzen badugu, konputazioaren  $(1/S)$  sekuentziala eta gainontzekoa,  $(1 - 1/S)$  paralizagarria 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 paraleloan,  $T_p$  denborari paralelizazioak duen gainkarga gehitu behar zaio. Gainkarga hau, kontzeptu ezberdinez osatuta dago eta milisegunduko mailako eragiketak izan daitezke.

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

## A.5. Aljebra lineal dentsorako liburutegiak.

Zenbakizko integrazioen aljebra lineala, konputazioaren alde konplexua da. Aljebra linealeko eragiketak implementatzen dituzten kalitate handiko liburutegiak daude eta implementazio berriak, liburutegi hauetan oinarritzea gomendagarria da [61]. Liburutegi hauek, ondo probatutako softwareak dira, konplexutasun handikoak, modu seguruan eta azkarrean exekutatzeko diseinatu dira.

Hauek dira, aljebra linealerako liburutegi aipagarrienak:

1. BLAS (Basic Linear Algebra Subroutines): matrize eta bektoreen arteko eragiketa aritmetikoak biltzen dituen liburutegia.
2. LAPACK (Linear Algebra Package): aljebra linealaren problemak ebazteko liburutegia.

Implementazio hauen funtzioak *Fortran* lengoaiaren garatuta daude eta ezaugarri hauek dituzte:

1. Datu-mota huetarako aplika daitezke:
  - (a) S: doitasun arrunta (*float*, 32-bit).
  - (b) D: doitasun bikoitza (*double*, 64-bit).
  - (c) C: zenbaki konplexua doitasun arruntean (*complex*).
  - (d) Z: zenbaki konplexua doitasun bikoitzean (*complex double*).
2. Matrize dentsoetarako liburutegiak dira. Matrize egitura hauek zehaztu daitezke.
  - (a) Matrize orokorrak.  
GE=General; GB=General Band.
  - (b) Matrize simetrikoak.  
SY=SYmmetric ; SB=Symmetric Band; SP=Symmetric Packed.
  - (c) Hermitiar matrizeak.  
HE=HErmitian ; HB=Hermitian Band; HP=Hermitian Packed.
  - (d) Matrize triangularrak.  
TR=TRiangular ; TB=TRiangular Band; TP=Triangular Packed.

## BLAS.

**BLAS** liburutegiak [1], bektore eta matrizeen arteko funtziotako estandarrak biltzen ditu. Liburutegia, 142 errutinaz osatuta dago eta hauek, hiru taldeetan sailkatzen dira:

1. BLAS-1:  $\mathcal{O}(n)$  bektore-bektore eragiketak.

Adibidea.  $y = \alpha * x + y$ , non  $\alpha \in \mathbb{R}$ , eta  $x, y \in \mathbb{R}^n$ .

2n eragiketa aritmetiko eta  $3n$  irakurketa/idazketa.

Konputazio intentsitatea:  $2n/3n = 2/3$ .

2. BLAS-2:  $\mathcal{O}(n^2)$  matrize-bektore eragiketak.

Adibidea.  $y = \alpha * A * x + \beta * y$ , non  $\alpha, \beta \in \mathbb{R}$ ,  $x, y \in \mathbb{R}^n$  eta  $A \in \mathbb{R}^{n \times n}$ .

$\mathcal{O}(n^2)$  eragiketa aritmetiko eta  $\mathcal{O}(n^2)$  irakurketa/idazketa.

Konputazio intentsitatea:  $\approx 2n^2/n^2 = 2$ .

3. BLAS-3:  $\mathcal{O}(n^3)$  matrize-matrize eragiketak.

Adibidea.  $C = \alpha * A * B + \beta * C$ , non  $\alpha, \beta \in \mathbb{R}$  eta  $A, B \in \mathbb{R}^{n \times n}$ .

$\mathcal{O}(n^3)$  eragiketa aritmetiko eta  $\mathcal{O}(n^2)$  irakurketa/idazketa.

Konputazio intentsitatea:  $\approx 2n^3/4n^2 = n/2$ .

BLAS-1 eta BLAS-2 funtzioen konputazio intentsitatea txikia da eta beraz, talde hauetako funtzioetan, datuen komunikazioa nagusia da. BLAS-3 funtzioetan aldiz, konputazio intentsitatea handiagoa da eta ezaugarri honi esker, tamaina handiko matrizeen kalkuluetan, konputagailuaren konputazio gaitasuna ondo aprobetxatu ahal izango da.

Aljebra linealeko aplikazioen exekuzio denboraren zati garrantzitsuena, behe-mailako eragiketa hauen konputazioak ematen du. Behe-mailako eragiketen hauen optimizazioak, konputagailu bakoitzaren araberakoak dira eta espezializazio handia eskatzen du. Fabrikataile bakoitzak optimizatutako bere BLAS liburutegia du (AMD-ACML,Intel-MKL). Bestalde, optimizatutako BLAS instalazioa, ATLAS (Automatically Tuned Linear Algebra Software) izeneko aplikazioaren bidez ere egin daiteke.

Implementazio guztiak, interfaze berdina erabiltzen dute eta beraz, BLAS-en oinarritutako garapena edozein konputagailuan erabili daiteke (portabilitatea). BLAS liburutegia, Fortran lengoian implemtatuta dago eta C lengoiaiatik BLAS funtzioen erabilpena errazteko, *cblas* interfazea erabiltzea gomendagarria da.

**Adibidea.** BLAS liburutegiaren eraginkortasuna, *cblas\_dgemm()* matrizeen biderkadura funtzioren bidez aztertu dugu eta gure implementazio arrunta baino  $10 \times$  azkarragoa dela baiezta dugu (Taula A.1.).

**A.1. Taula:** BLAS liburutegiaren eraginkortasuna. C lengoaiako gure garapena (C-arrunta) eta BLAS liburutegiaren cblas\_dgemm() implementazioak konparatu ditugu.  $n$  tamainako ezberdineko matrizreak biderkatzu ditugu, eta biderketa bakoitzak  $n_{test}$  alditan errepikatu dugu, kasu guztiak eragiketa aritmetiko kopuru berdina izan ditzaten

$n$	nests	C-Arrunta		cblas_dgemm	
		Wall T.	CPU T.	Wall T.	CPU T.
10	$5.00 \times 10^8$	478.	478.	205.	206.
20	$6.25 \times 10^7$	491.	491.	92.	91.
30	$1.85 \times 10^7$	474.	474.	78.	78.
40	$7.81 \times 10^6$	523.	523.	66.	66.
50	$4.00 \times 10^6$	493.	493.	64.	64.
60	$2.31 \times 10^6$	479.	479.	58.	58.
70	$1.45 \times 10^6$	475.	475.	43.	170.
80	$9.76 \times 10^5$	469.	469.	45.	177.
90	$6.85 \times 10^5$	491.	491.	47.	186.
100	$5.00 \times 10^5$	466.	466.	39.	156.
200	$6.25 \times 10^4$	504.	504.	34.	138.
400	$7.81 \times 10^3$	657.	657.	35.	140.

## LAPACK.

**LAPACK**, 1992. urtean garatu zen [4, 59] eta aljebra linealaren problemak ebazteko funtzioen liburutegia da. Jatorrizko bertsioa *Fortran 77* lengoian implementatuta dago eta liburutegiaren dokumentazioa nahiz kodea *Netlib* software bilgunean eskuratu daiteke. Matrize dentsoetarako garatuta dago eta problema hauetarako errutinak biltzen ditu:

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

LAPACK liburutegia, konputagailu sekuentzial eta memoria konpartitutako konputagailuetan erabilgarria izateko diseinatuta dago. Eraginkortasuna *BLAS* funtzio optimizatuen menpe dago eta funtzioen implementazioa, gehien bat *BLAS-3* taldeko funtzioetan oinarritzen da.

*LAPACKE* liburutegia C-lengoaiatik LAPACK funtzioei deitzeko interfazea da. LAPACK liburutegiaren funtzioak, hiru taldeetan banatzen dira [4, 1]:

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

**Adibidea.**

LAPACKE-dgelsv (LAPACK-ROW-MAJOR, n, nrhs, A, lda, ipiv, B, ldb);  
Matrize orokoren (GE),  $A * X = B$  sistema lineala ebazten du,

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

**Adibidea.**

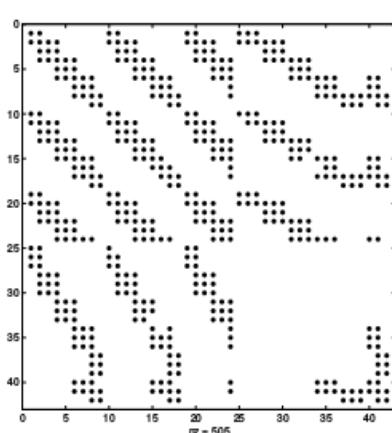
LAPACKE-dgetrf (LAPACK-ROW-MAJOR, n, m, A, lda, ipiv);  
Errutina honek  $A$  ( $n \times m$ ) tamainako matrizearen  $LU$  faktorizazioa kalkulatzen du,  $A = P * L * U$ .  
LAPACKE-dgetrs (LAPACK-ROW-MAJOR,trans,n,nrhs,A,lda,ipiv,B,ldb);  
Errutina honek  $A * X = B$  ekuazio sistemaren  $X$  soluzioa kalkulatzen du.

3. Errutina laguntzaileak.

### Matrize bakanak.

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

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



**A.7. Irudia:** Matrize bakanak.

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

1. Doitasun bikoitzeko  $A \in \mathbb{R}^{m \times n}$  matriza,
  - (a) Dentsoa:  $8mn$  byte.
  - (b) Bakana:  $\approx 16n_{nz}$  (gordetzeko teknikaren arabera).
2.  $y = y + Ax$ ,  $y, x \in \mathbb{R}^n$  eta  $A \in \mathbb{R}^{m \times n}$ ,
  - (a) Dentsoa:  $\mathcal{O}(mn)$ , eragiketa aritmetikoak.
  - (b) Bakana:  $\mathcal{O}(n_{nz})$ , eragiketa aritmetikoak.

## A.6. Konpiladorea.

Konpiladorearen zeregina konplexua da, goi mailan idatzitako programari dagoen makina kodea sortzea (konputagailuaren errekurtoak modu eraginkorrean erabiltzen dituenak) [109]. Konpiladoreak heuristikotan oinarritutako kode alda-ketak eragiten ditu, eraginkortasuna hobetzeko asmoarekin. Horregatik, programatzaileak konpiladorearen optimizazio automatiko hauek kontutan hartu behar ditu eta ahal duen neurrian, bere kodean konpiladorearen optimizazioak erraztu.

**Konpiladoreak.** Konpiladore ezberdinak daude:

1. *gcc* (GNU open source compiler).

```
$ gcc -v
$ gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.2)
```

2. Konpiladore komertzialak: Intel (*icc*),...

**Optimizazioak.** Konpiladoreek, optimizazio maila estandarrak eskaintzen dituzte. Orokorean, hurrengo kode optimizazioak izango ditugu:

1. *-O0*. Kode optimizazio nagusienak aplikatzeko aukera da. Kodea *debugger* moduan aztertzen ari garenean gomendatzen da.
2. *-O2*. Kode eraginkorra sortzeko aukera gomendarriena.

Maila altuagoko optimizazioak aplika daitezke, baina optimizazio hauek arriskutsuak izan daitezke. Kodearen exekuzio denboraren analisia egiteko tresnak (*gprof*) daude. Algoritmoaren funtzio bakoitzaren exekuzio denborari buruzko informazio erabilgarria lortuko dugu.

### Konpilazio aginduak .

1. Konpilazioa, esteka egin eta adibidea.exe exekutagarria sortzeko

```
$ gcc adibidea.c -o adibidea.exe
```

2. Konpilazio eta esteka urratsak banatuta.

```
$ gcc adibidea.c # creates adibidea.o
$ gcc adibidea.o -o adibidea.exe
```

Gure esperimentuetarako era honetan burutu dugu konpilazioa,

```
gcc -O2 -Wall -std=c99 -fno-common adibidea.c
```

**Makefile.** Normalean, aplikazioaren kodea fitxategi ezberdinetan egituratzen da eta konpilazio prozesua konplexua izan daiteke. *Makefile* fitxategia, lengoaia berri bat erabiliz, konpilazio prozesua automatizatzeko programa moduko bat dugu [109]. ?? eranskinean, *Makefile* lengoaiaren oinarrizko adibideak eman ditugu.

## A.7. Laburpena.

Algoritmo bat implementatzen dugunean kontutan hartu beharrekoa:

1. Lerro edo zutabe araberako iterazioak exekuzio denboran eragin handia du.
2. Kodea garbia eta ulergarria mantendu behar da.
3. LAPACK eta BLAS liburutegiak oso eraginkorrik dira, eta implementazio berriean erabiltzea komenigarria da.

Atal honi dagokion gomendatutako bibliografia: "An introduction to parallel programming", P. Pacheco [95], "Introduction to High Performance Scientific Computing", V. Eijkhout [109], "Performance optimization of numerically intensive codes", Goedecker [48], "Handbook of linear algebra", Leslie Hogben [61].

LAPACK eta BLAS liburutegiak erabiltzeko informazio interesgarria "Intel Math Kernel Library. Reference Manual" [1] dokumentuan aurki daiteke.

## B. Eranskina

### Ekuazioen garapenak.

#### B.1. Kepler hasierako baliodun problema.

Keplerren ekuazioa, kokapen eta abiadura berriak kalkulatzeko oinarrizkoa da eta era honetan definitzen da,

$$E - e \sin E = M,$$

non  $M = n(t - T)$  (*mean anomaly*),  $n = k a^{-3/2}$  (*mean motion*) eta  $T, M = 0$  deneko integracio konstantea da.  $E$  (*eccentric anomaly*) eta  $t$ -ren arteko erlazio hau erabiliz kalkulatzen da mugimendua. Mugimendu eliptikoaren kasura mugatuko gara ( $0 \leq e < 1$ ) eta Keplerren ekuazioa transentalala denez, zenbakizko metodo baten bidez ebatziko dugu.

#### Garapena.

Gure abiapuntua, honakoa da,

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

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

Bi ekuazioen arteko kendura eginez,

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

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

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

Eta beraz, Keplerren ekuazioak hauek izango dira,

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

Eta  $(q_1, v_1)$  balioak kalkulatzeko,

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

## B.2. Koordenatu sistemak.

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

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

### Koordenatu Heliozentrikoak.

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

#### Aldagai aldaketa.

Lehenik honako aldagai aldaketa aplikatuko dugu,

$$\begin{aligned} Q_0 &= q_0, \quad Q_i = q_i - q_0, \\ P_0 &= \sum_{i=0}^N p_i, \quad P_i = p_i, \quad i = 1, \dots, N. \end{aligned}$$

Hamiltondarra era honetan deskonposatu daiteke,

$$H = H_K + (T_1 + U_1)$$

non

$$\begin{aligned} H_K &= \sum_{i=1}^N \left( \frac{\|P_i\|^2}{2\mu_i} - \frac{Gm_0m_i}{\|Q_i\|} \right), \quad \mu_i = \frac{m_0m_i}{(m_0 + m_i)}, \\ T_1 &= \frac{1}{m_0} \left( \sum_{0 < i < j \leq N} P_i P_j \right), \\ U_1 &= - \sum_{0 < i < j \leq N} \frac{Gm_i m_j}{\|Q_i - Q_j\|}. \end{aligned} \quad (\text{B.2})$$

Hamiltondarra abiaduraren arabera idazteko  $V_i = P_i/\mu_i$  berdintza ordezkatu-

ko dugu,

$$\begin{aligned} H_K &= \sum_{i=1}^N \left( \frac{\|V_i\|^2 \mu_i}{2} - \frac{Gm_0 m_i}{\|Q_i\|} \right), \quad \mu_i = \frac{m_0 m_i}{(m_0 + m_i)}, \\ T_1 &= m_0 \left( \sum_{0 < i < j \leq N}^N \frac{m_i m_j V_i V_j}{(m_0 + m_i)(m_0 + m_j)} \right), \\ U_1 &= - \sum_{0 < i < j \leq N}^N \frac{Gm_i m_j}{\|Q_i - Q_j\|}. \end{aligned} \quad (\text{B.3})$$

### Ekuazio differentzialak.

Hamiltondar bakoitza independenteki kontsideratuta dagokio ekuazio differentzialak lortzen ditugu:

1.  $H_K$ .

$$\begin{aligned} \dot{Q}_i &= \nabla_p H_k \Rightarrow \dot{Q}_i = P_i \left( \frac{m_0 + m_i}{m_0 m_i} \right), \\ \dot{P}_i &= -\nabla_q H_k \Rightarrow \dot{P}_i = -\frac{Gm_0 m_i}{\|Q_i\|^3} Q_i, \quad i = 1, \dots, N. \end{aligned}$$

2.  $T_1$ .

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

3.  $U_1$ .

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

Azkenik,  $V_i = P_i / \mu_i$  aplikatuta integrazioan erabiliko ditugun ekuazioak la-burtuko ditugu.

1.  $H_k$ .

$$\begin{aligned} \dot{Q}_i &= V_i \\ \dot{V}_i &= -\frac{G(m_0 + m_i)}{\|Q_i\|^3} Q_i, \quad i = 1, \dots, N. \end{aligned}$$

2.  $T_1$ .

$$\begin{aligned}\dot{Q}_i &= \sum_{j \neq i, j=1}^N \frac{V_j m_j}{(m_0 + m_j)}, \\ \dot{V}_i &= 0, \quad i = 1, \dots, N.\end{aligned}$$

3.  $U_1$ .

$$\begin{aligned}\dot{Q}_i &= 0, \\ \dot{V}_i &= -\frac{G(m_0 + m_i)}{m_0} \sum_{j \neq i, j=1}^N \left( \frac{m_j}{\|Q_i - Q_j\|^3} (Q_i - Q_j) \right), \quad i = 1, \dots, N.\end{aligned}$$

### Energia.

Koordenatu heliozentrikoetan integrazioak egiten ditugunean, sistemaren energia kalkulatzeko, soluzioa koordenatu sistema barizentrikoetara bihurtuko dugu. Hauek dira koordenatu heliozentrikoetatik abiatuta ( $Q_i, V_i$ ), koordenatu barizentrikoak ( $q_i, v_i$ ) kalkulatzeko ekuazioak,

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

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

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

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

### Koordenatu Jacobiarak.

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

### Aldagai aldaketa.

Lehenik honako aldagai aldaketa aplikatuko dugu,

$$Q_0 = (m_0 q_0 + \cdots + m_n q_n) / \eta_N, \quad Q_i = q_i - \left( \sum_{j=0}^{i-1} m_j q_j \right) / \eta_{i-1}$$

$$P_0 = \sum_{i=0}^N p_i, \quad P_i = \left( \eta_{i-1} p_i - m_i \sum_{j=0}^{i-1} p_j \right) / \eta_i, \quad i = 1, \dots, N.$$

non  $\eta_i = \sum_{j=0}^i m_j$  den.

Era berean, Jacobi masak  $m'_i = (\eta_{i-1} m_i) / \eta_i$  eta  $\mu'_i = m_i \eta_{i-1}$  ekuazioetan ordezkatuko ditugu. Hamiltondarra era honetan deskonposatu daiteke,

$$H = H_K + H_I,$$

non

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

$$H_I = \left( \frac{\mu'_i}{Q_i} - \frac{Gm_0 m_i}{q_i} \right) - \sum_{0 < i < j \leq N} \frac{Gm_i m_j}{\|Q_i - Q_j\|}.$$

## B.3. Newton eraginkorraren garapena.

Formulazio estandarrean honako ekuazio sistema askatzeko metodoa proposatzen da,

$$(I_s \otimes I_d - h A \otimes J) \Delta Y = r. \quad (\text{B.4})$$

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

### Kasu orokorra

Honako ekuazio sistemari,

$$(I_s \otimes I_d - h \bar{A} \otimes J) \Delta Y - \frac{1}{2} (e_s \otimes I_d) \Delta z = r, \quad (\text{B.5})$$

$$(-h e_s^T B \otimes J) \Delta Y + \Delta z = 0, \quad (\text{B.6})$$

aldagai aldaketa hau, aplikatuko doigu.

$$\Delta Y = (Q \otimes I_d) W. \quad (\text{B.7})$$

1. Lehen urratsa.

Ekuazioa sistemaren lehen ekuazioari (B.5), aldagai aldaketa (B.7) aplikatu eta  $(Q^{-1} \otimes I_d)$  gaia ezkerretik biderkatuz,

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

Eta garatuz,

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

2. Bigarren urratsa.

Metodoa simetrikoa bada,

$$Q^{-1} \bar{A} Q = \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \quad (\text{B.8})$$

eta beraz,

$$(I_s \otimes I_d - h \begin{pmatrix} 0 & D \\ -D^T & 0 \end{pmatrix} \otimes J) W - \frac{1}{2} (Q^{-1} e_s \otimes I_d) \Delta z = (Q^{-1} \otimes I_d) r.$$

Eta  $I_s \otimes I_d$  bloke moduan idatzia,

$$I_s \otimes I_d = \begin{pmatrix} I_m \otimes I_d & 0 \\ 0 & I_{s-m} \otimes I_d \end{pmatrix} \quad (\text{B.9})$$

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

3. Hirugarren urratsa.

Bigarren ekuazioari (B.5) aldagai aldaketa (B.7) aplikatuz,

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

eta garatuz,

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

## Kasu simetrikoa

Kasu orokorraren emaitzan, kasu simetrikoa ordezkatz

$$W = \begin{pmatrix} W' \\ W'' \end{pmatrix}, \quad Q = (Q_1 \ Q_2)$$

dagokion ekuazioak lortuko ditugu ekuazioak.

Honako berdintza hauek kontutan hartuz,  $Q^{-1} = Q^T B$ ,  $es^T B Q_2 = 0$ ,  $Q_2^T B es = 0$  honako ekuazioak lortuko ditugu,

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

## B.4. Kepler fluxuaren aldagai aldaketa.

### Alde Kepleriar bakarra.

Satelite baten orbitaren ekuazio diferentzialak, era honetan idatz daitezke,

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} q \\ v \end{pmatrix} &= k(t, q, v) + g(t, q, v), \\ k(t, q, v) &= \begin{pmatrix} v \\ -\frac{\mu}{\|q\|^3} q \end{pmatrix}, \end{aligned} \tag{B.10}$$

non  $q, v \in \mathbb{R}^3$  den.

**Aldagai aldaketa.** Honako aldagai aldaketa aplikatuko dugu,

$$\begin{pmatrix} q \\ v \end{pmatrix} = \varphi_t(Q, V) \tag{B.11}$$

**Aldagai berriekiko ekuazio diferentzialak.** Aldagai berriarekiko ekuazio differenzialak definituko ditugu eta horretarako (B.11) ekuazioak, katearen erregela aplikatuz deribatuko ditugu,

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} q(t) \\ v(t) \end{pmatrix} &= \frac{d}{dt} (\varphi_t(Q(t), V(t))) \Rightarrow \\ \frac{d}{dt} \begin{pmatrix} q(t) \\ v(t) \end{pmatrix} &= \dot{\varphi}_t(Q(t), V(t)) + \varphi'_t(Q(t), V(t)) \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix}. \end{aligned}$$

Fluxuaren definizioa erabiliz, honako berdintza

$$\dot{\varphi}_t(Q(t), V(t)) = k(q(t), v(t)),$$

betetzen da eta beraz,

$$g(t, q, v) = g(t, \varphi_t(Q, V)) = \varphi'_t(Q(t), V(t)) \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix}.$$

Azkenik,  $Q(t), V(t)$  aldagairekiko deribatu partzialen espresioak lortuko ditugu,

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix} &= \varphi'_t(Q(t), V(t))^{-1} g(t, \varphi_t(Q, V)) \Rightarrow \\ G(t, Q, V) &= \varphi'_t(Q(t), V(t))^{-1} g(t, \varphi_t(Q, V)). \end{aligned} \quad (\text{B.12})$$

**Perturbazio Hamilondarra.** Lehenik, perturbazioa Hamilondarra den kasua kontsideratuko dugu,

$$g(t, q, v) = \begin{pmatrix} \nabla_p r(t, q, v) \\ -\nabla_q r(t, q, v) \end{pmatrix} = J^{-1} \nabla r(t, q, v),$$

non,

$$J = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}, \quad J^{-1} = -J.$$

Bestalde fluxu Kepleriarrak, Hamilondarra simplektikoa izateagatik, honako propietatea betetzen du,

$$(\varphi'_t(Q, V))^T J \varphi'_t(Q, V) = J. \quad (\text{B.13})$$

Eta goiko (B.13) garatuz, honakoa ere betetzen dela badakigu,

$$J^{-1}(\varphi'_t(Q, V))^T = (\varphi'_t(Q, V))^{-1} J^{-1}.$$

Notazio hau finkatuz,

$$R(t, Q, V) = r(t, \varphi_t(Q, V)),$$

eta  $\nabla R(t, Q, V)$  deribatuari katearen erregela aplikatuz, berdintza hau,

$$J^{-1}(\varphi'_t(Q, V))^T \nabla r = J^{-1} \nabla R(t, Q, V). \quad (\text{B.14})$$

betetzen dela baieztagatzen da.

**Perturbazio Ez-Hamiltondarra.** Kepler-en fluxua simplektikoa da eta propietate horretan oinarritzen gara fluxuaren alderantzikoaren kalkulua saihesteko.

$$\begin{cases} \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix} = (\varphi'_t(Q, V))^{-1} g(t, \varphi_t(Q, V)) \\ J^{-1}(\varphi'_t(Q, V))^T J = (\varphi'_t(Q, v))^{-1} \end{cases} \Rightarrow \frac{d}{dt} \begin{pmatrix} Q(t) \\ V(t) \end{pmatrix} = J^{-1}(\varphi'_t(Q, V))^T J g. \quad (\text{B.15})$$

**Algoritmoa.** (B.15) modu eraginkorrean konputatzeko, deribazio automatikoen prozedura aplikatuz kalkulatuko dugu.  $(Q, V)$  aldagaien integrazioa hiru urratsetan egingo dugu,

1. Kepler-en fluxua aplikatu, aldagai zaharretara itzultzeko (KeplerFlowGen). Funtzio honek, gero erabiliko ditugun tarteko aldagaiak ere itzuliko ditu.

$$(q, v) = \varphi_t(Q, V).$$

2. Perturbazioak kalkulatu (GFcn).

$$g = GFcn(t, q, v)$$

3. Fluxuaren deribatua eta honako espresioa konputatu (KeplerFlowGFcnaux).

$$J^{-1}(\varphi'_t(Q, V))^T J g.$$

### Alde Kepleriar bat baino gehiago.

Alde Kepleriar bat baino gehiago den kasurako azterketa egingo dugu.  $k$ , alde Kepleriarren kopurua esango diogu.

$$\frac{d}{dt} \begin{pmatrix} q \\ v \end{pmatrix} = \begin{pmatrix} q_1 \\ v_1 \\ q_2 \\ v_2 \\ \vdots \\ q_k \\ v_k \\ w \end{pmatrix} = \begin{pmatrix} v_1 \\ -\mu_1 q_1 / \|q_1\|^3 \\ v_2 \\ -\mu_2 q_2 / \|q_2\|^3 \\ \vdots \\ v_k \\ -\mu_k q_k / \|q_k\|^3 \\ 0 \end{pmatrix} + g(t, q_1, v_1, \dots, q_k, v_k, w),$$

non

$$g(t, q_1, v_1, \dots, q_k, v_k, w) = \begin{pmatrix} g_1(t, q_1, v_1, \dots, q_k, v_k, w) \\ g_2(t, q_1, v_1, \dots, q_k, v_k, w) \\ \vdots \\ g_k(t, q_1, v_1, \dots, q_k, v_k, w) \\ g_{k+1}(t, q_1, v_1, \dots, q_k, v_k, w) \end{pmatrix}.$$

Gorputz bakoitzari dagokion aldagai aldaketa lokala da,

$$\begin{pmatrix} q_j \\ v_j \end{pmatrix} = \varphi_t^{\mu_j}(Q_j, V_j), \quad j = 1, \dots, k, \quad (B.16)$$

$w = W.$

Fluxuaren Jacobiarra diagonala da,

$$\begin{pmatrix} \varphi_t^{\mu_1'}(Q_1, V_1) & & & \\ & \ddots & & \\ & & \varphi_t^{\mu_k'}(Q_k, V_k) & \\ & & & I \end{pmatrix}.$$

(B.15) aplikatzen badugu,

$$\frac{d}{dt} \begin{pmatrix} Q_1 \\ V_1 \\ \vdots \\ Q_k \\ V_k \\ w \end{pmatrix} = \begin{pmatrix} \varphi_t^{\mu_1'}(Q_1, V_1)^{-1} g_1 \\ \varphi_t^{\mu_2'}(Q_2, V_2)^{-1} g_2 \\ \vdots \\ \varphi_t^{\mu_k'}(Q_k, V_k)^{-1} g_k \\ g_{k+1} \end{pmatrix}$$

## C. Eranskina

### Inplementazioak.

Lan honetan, ekarpen bakoitzari dagokion C lengoaiako implementazio bat garatu dugu eta [kode bilgunean](#) eskuragarri jarri dugu. Kodea eskuragarri jartzeari, lehentasuna eman diogu [8, 110]. Batetik, garapenak modu argian dokumentatzea behartu gaitu; bertan zehaztasun guztiak aztertu daitezke eta gure lanaren baliagarritasuna baiezztatu daiteke. Bestetik, ikerlariei gure lana erabiltzeko eta hobetzeko aukera eskaintzen diegu.

Konputazio zientzian, idatzitako kodearen 200 lerro oro, errore bat izaten dela [43] estimatzen da. Gure implementazioak ondo frogatu baditugu ere, errorerik ez denik izango ezin dugu ziurtatu. Akatsen bat izatekotan, larria ez izatea espero dugu eta edozein kasutan erabiltzaileak jakinaraztea eskertu genuke.

Implementazioen kdea antolatzeko, irizpide berdinak erabili ditugu. Lehenengo, implementazioen egitura orokorra azaldu dugu eta ondoren, garapen bakotzaren argibideak emango ditugu.

#### C.1. Egitura orokorra.

Implementazioen edukia, direktorio hauetan banatu dugu:

##### 1. CoefficientsData.

Mathematican, IRK Gauss kolokazio metodoaren koefizienteak sortzeko aplikazio bat garatu dugu. Gure esperimentuetarako  $s = 6, s = 8$  eta  $s = 16$  ataletako Gaussen metodoak aplikatu ditugu eta metodo bakoitzari dagokien doitasun bikoitze (64-bit) koefizienteak, azpidirektorio batean bildu ditugu.

##### 2. PerturbationsData.

Errorearen azterketa estatistikoak egiteko, hasierako balio ezagun baten  $P = 1000$  hasierako balio perturbatuak sortu ditugu; hasierako balio originalaren

osagai bakoitza ausaz perturbatu dugu ( $\mathcal{O}(10^{-6})$  tamainako errore erlatiboarekin). Problema bakoitzari dagokion,  $P = 1000$  hasierako balio perturbatuak, fitxategi bitar batean idatzi dugu eta horrela, fitxategi hauetatik hasierako balioak irakurrita, esperimentu bera errepikatu daiteke.

### 3. Packages.

Esperimentuak, Mathematica ingurunetik exekutatu ditugu eta direktorio honetan, soluzioen azterketak egiteko hainbat funtzi garatu ditugu. Batez, problema bakoitzari dagokion ekuazio diferentzialak eta Hamiltondarra implementatu ditugu. Bestetik, esperimentuen grafikoak irudikatzeko funtzioak garatu ditugu.

### 4. Examples.

Mathematica erabiliz egindako zenbakizko integrazioen adibideak eman ditugu. Adibide bakoitza, azpidirektorio batean bildu dugu: zenbakizko integrazioa eta honen analisia, modu independentean exekutatu daitezke. Horretarako, integrazioaren soluzioak fitxategi bitarretan idazten dira eta analisiak, fitxategi hauek irakurriko ditu.

### 5. Code.

C lengoian garatutako gure implementazioa.

## Exekuzioa.

Code/Readme.txt fitxategian, implementazioaren argibideak eman ditugu. Implementazioa bi modutan exekutatu daiteke.

### 1. Linux terminala.

Exekuzioa Linux terminaletik exekutatu daiteke eta exekutatzeko adibidea bat "terminal.c" fitxategian eman dugu.

### 2. Mathematica ingurunetik.

Bestalde, implementazioa Mathematicatik exekutatzeko prest dago eta "Examples" direktorioan hainbat adibide eman ditugu. Mathematicatik gure C implementazioaren funtzi nagusia deitu ahal izateko, 'math-IRK.tm' eta 'math-IRK.c' fitxategiak definitu ditugu. Exekutarria, Mathematicatik bateragarria izan dadin, konpilazio eta esteka egiteko moduak *makefile* fitxategian kontsultatu daitezke.

Erabiltzaileak, bere problemaren ekuazio diferentzialak eta integrazioen emaitzak definituko dituela espero da.

### Paralelizazioa.

IRK metodoen s-ataletako funtzioen konputazioak ( $f(Y_i)$ ,  $i = 1, \dots, s$ ), paraleloan exekutatu daitezke. Gure implementazioan, OpenMP erabili dugu: *PARALLEL* konpilazio parametroarekin aktibatu daiteke eta orduan, prozesadore kopuruoa 'threadcount' aldagaien, zehaztu behar da.

## C.2. IRK puntu-finkoa.

Puntu-finkoaren iterazioan oinarritutako IRK metodoaren implementazioa, [kodea](#) helbidean eskuragarri dago. Bi modutara exekutatu daiteke: puntu-finkoaren iterazio estandarra edo puntu-finkoaren iterazio partizionatua.

Code/Readme.txt fitxategian, IRK puntu-finkoaren implementazioaren argibide guztiak eman ditugu. Integrazioaren funtzi nagusiaren deia honakoa da:

```
IRKFPI ( t0 , t1 , h,& method ,& u,& system ,& options ,& thestat );
```

Exekuzioa Linux terminaletik exekutatu daiteke eta exekutatzeko adibidea bat "terminal.c" fitxategian eman dugu. Bestalde, implementazioa Mathematicatik exekutatzeko prest dago eta "Examples" direktorioan hainbat adibide eman ditugu.

Implementazio honetan, ez dugu *BLAS* liburutegia erabili.

## C.3. IRK Newton.

Newton simplifikatuaren iterazioan oinarritutako IRK metodoaren implementazioa, [kodea](#) helbidean eskuragarri dago. Bi implementazio exekutatu daitezke: Newton simplifikatuaren implementazio eraginkorra eta artikuluan, proposatutako Newton iterazioan oinarritutako IRK implementazio berria.

Code/Readme.txt fitxategian, IRK puntu-finkoaren implementazioaren argibide guztiak eman ditugu. Integrazioaren funtzi nagusiaren deia honakoa da:

```
IRKNEWTON ( t0 , t1 , h,& method ,& u,& system ,& options ,& thestat );
```

Exekuzioa Linux terminaletik exekutatu daiteke eta exekutatzeko adibidea bat "terminal.c" fitxategian eman dugu. Bestalde, implementazioa Mathematicatik exekutatzeko prest dago eta "Examples" direktorioan hainbat adibide eman ditugu.

Implementazio honetan, *BLAS* eta *LAPACK* liburutegiak erabili ditugu.

## **C.4. IRK Eguzki-sistema.**

## **C.5. Konposizio-Splitting metodoak.**

Newton simplifikatuaren iterazioan oinarritutako IRK metodoaren implementazioa, [kodea](#) helbidean eskuragarri dago. Bi implementazio exekutatu daitezke: konposizio metodoak (CO1035) eta splitting metodoak (ABAH1064).

Code/Readme.txt fitxategian, konposizio/Splitting implementazioaren argibide guztiak eman ditugu. Integrazioaren funtziogun nagusiaren deia honakoa da:

```
Solve_Comp ( t0 , t1 , h,& method , basic ,& system ,& options ,& u,& thestat );
```

Exekuzioa Linux terminalatik exekutatu daiteke eta exekutatzeko adibidea bat "terminal.c" fitxategian eman dugu. Bestalde, implementazioa Mathematicatik exekutatzeko prest dago eta "Examples" direktorioan hainbat adibide eman ditugu.

Eguzki-sistemari egokitutako Splitting metodoak aplikatzeko, Kepler fluxuan implementazio berri bat garatu dugu. Implementazio honen C kodean, 'Kepler.c' fitxategian aurkitzen da.

**D. Eranskina**

**Zerrendak**

**Hitz-zerrenda.**

### D.1. Taula: Hitz-zerrenda.

Euskaraz	Ingelesez	Laburdura	Adibidea
Ekuazio diferentzial arrunta	Ordinary Differential equation	<i>ODE</i>	$\dot{y} = f(t, y)$
Hasierako baliodun problema	Initial value problem	<i>IVP</i>	
Zurruna	Stiff		
Simplektikoa	Symplectic		
Drift	Drift		
Splitting metodoak	Splitting methods		
Konposizio metodoak	Conposition methods		
Runge-Kutta Esplizitua	Explicit Runge-Kutta	<i>ERK</i>	
Runge-Kutta Implizitua	Implicit Runge-Kutta	<i>IRK</i>	
Batura konpensatua	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		
Balio singularen deskonposaketa	Graphical Processor Unit	<i>GPU</i>	
Eszentrikotasuna	Least Eigenvalues problems		
	Linear least square problems		
	Eigenvalues problems		
	Singular values descomposition	<i>SVD</i>	
Unitate astronomikoa	Eccentricity		
LU deskonposaketa	Eccentric anomaly		
	Astronomical unit	<i>AU</i>	
	LU Decomposition		
	Backward substitution		
Koma-higikorreko eragiketa segunduko	Float operations per second	<i>flops</i>	
Peak	Exekuzio gaitasuna		
	Wall-time, elapsed-time		
CPU denbora	CPU-time		
Cache memoria	Cache memory		
	Spatial/data locality		
Single Instruction Multiple Data	Single Instruction Multiple Data	<i>SIMD</i>	
	Fork-join		
Interfaze aplikazio programa	Application programming interface	<i>API</i>	
Erredimendu altuko konputazioa	High performance computing	<i>HPC</i>	
Problemak ebazteko inguruneak	Problem solving enviroments	<i>PSE</i>	Mathematica
Matrize bakanak	Sparse matrices		

# Bibliografia

- [1] *Intel Math Kernel Library. Refrence Manual*. Intel, 2015.
- [2] The openmp api specification for parallel programming, 2017.
- [3] Sverre Aarseth, Christopher Tout, and Rosemary Mardling. *The Cambridge n-body lectures*, volume 760. Springer, 2008.
- [4] Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, et al. *LAPACK Users' guide*. SIAM, 1999.
- [5] KV Vladimirov Andrey. *Test-driving intel xeon phi coprocessors with a basic n-body simulation*. *Coflax International*, 3:2, 2013.
- [6] Mikel Antonana, Joseba Makazaga, and Ander Murua. Efficient implementation of symplectic implicit runge-kutta schemes with simplified newton iterations. *arXiv preprint arXiv:1703.07697*, 2017.
- [7] Mikel Antonana, Joseba Makazaga, and Ander Murua. Reducing and monitoring round-off error propagation for symplectic implicit runge-kutta schemes. *Numerical Algorithms*, pages 1–20, 2017.
- [8] Harald Atmanspacher and Sabine Maasen. *Reproducibility: Principles, Problems, Practices, and Prospects*. John Wiley & Sons, 2016.
- [9] Marc Baboulin, Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Julie Langou, Julien Langou, Piotr Luszczek, and Stanimire Tomov. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications*, 180(12):2526 – 2533, 2009.
- [10] Pavan Balaji. *Programming models for parallel computing*. MIT Press, 2015.
- [11] Josh Barnes and Piet Hut. *A hierarchical o (n log n) force-calculation algorithm*. *nature*, 324(6096):446–449, 1986.

- [12] André Berger. *A brief history of the astronomical theories of paleoclimates*. Springer, 2012.
- [13] Gregory Beylkin and Kristian Sandberg. *Ode solvers using band-limited approximations*. *Journal of Computational Physics*, 265:156–171, 2014.
- [14] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. *Julia: A fresh approach to numerical computing*. *SIAM*, 2014.
- [15] Theodore A Bickart. *An efficient solution process for implicit runge–kutta methods*. *SIAM Journal on Numerical Analysis*, 14(6):1022–1027, 1977.
- [16] Sergio Blanes, Fernando Casas, Ariadna Farres, Jacques Laskar, Joseba Makazaga, and Ander Murua. *New families of symplectic splitting methods for numerical integration in dynamical astronomy*. *Applied Numerical Mathematics*, 68:58–72, 2013.
- [17] Ben K Bradley, Brandon A Jones, Gregory Beylkin, Kristian Sandberg, and Penina Axelrad. *Bandlimited implicit runge–kutta integration for astrodynamics*. *Celestial Mechanics and Dynamical Astronomy*, 119(2):143–168, 2014.
- [18] Dirk Brouwer. *On the accumulation of errors in numerical integration*. *The Astronomical Journal*, 46:149–153, 1937.
- [19] Luigi Brugnano, Gianluca Frasca Caccia, and Felice Iavernaro. *Efficient implementation of gauss collocation and hamiltonian boundary value methods*. *Numerical Algorithms*, 65(3):633–650, 2014.
- [20] VA Brumberg. *Celestial mechanics: Past, present, future*. *Solar System Research*, 47(5):347–358, 2013.
- [21] Kevin Burrage. *Parallel methods for initial value problems*. *Applied Numerical Mathematics*, 11(1-3):5–25, 1993.
- [22] John C Butcher. *On the implementation of implicit runge-kutta methods*. *BIT Numerical Mathematics*, 16(3):237–240, 1976.
- [23] John Charles Butcher. *Numerical Methods for Ordinary Differential Equations*. Second edition, Wiley, 2008.
- [24] 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.

- [25] 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.
- [26] Philippe Chartier, Ander Murua, and Jesus Maria Sanz-Serna. [Higher-order averaging, formal series and numerical integration i: B-series](#). *Foundations of Computational Mathematics*, 10(6):695–727, 2010.
- [27] Martyn Corden. [Differences in floating-point arithmetic between intel® xeon® processors and the intel® xeon phi™ coprocessor x100 product family](#), 2013.
- [28] Robert M Corless and Nicolas Fillion. [A graduate introduction to numerical methods](#). *AMC*, 10:12, 2013.
- [29] John Danby. Fundamentals of celestial mechanics. *Richmond: Willman-Bell, | c1992, 2nd ed.*, 1, 1992.
- [30] Theodorus Jozef Dekker. [A floating-point technique for extending the available precision](#). *Numerische Mathematik*, 18(3):224–242, 1971.
- [31] DumitruN. Deleanu. [Fast detection of chaotic or regular behavior of double pendulum system: application of the fast norm vector indicator method](#).
- [32] Jack Dongarra, Stanimire Tomov, Piotr Luszczek, Jakub Kurzak, Mark Gates, Ichitaro Yamazaki, Hartwig Anzt, Azzam Haidar, and Ahmad Abdelfattah. [With extreme computing, the rules have changed](#). *Computing in Science & Engineering*, 19(3):52–62, 2017.
- [33] Michael Driscoll, Evangelos Georganas, Penporn Koanantakool, Edgar Solomonik, and Katherine Yelick. [A communication-optimal n-body algorithm for direct interactions](#). In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, IEEE, 2013, pages 1075–1084.
- [34] 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.
- [35] Siegfried Eggl and Rudolph Dvorak. [An introduction to common numerical integration codes used in dynamical astronomy](#). In *Dynamics of small solar system bodies and exoplanets*, Springer, 2010, pages 431–480.
- [36] Daniel C Fabrycky. [Non-keplerian dynamics](#). *arXiv preprint arXiv:1006.3834*, 2010.

- [37] 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.
- [38] Kang Feng and Mengzhao Qin. [Symplectic geometric algorithms for hamiltonian systems](#). Springer, 2010.
- [39] A. Fienga. [International workshop in astronomy and dynamics](#). In *INPOP planetary ephemerides: from 2013 to 2015.*, Observatoire de Paris, 2015.
- [40] A Fienga, H Manche, J Laskar, and Mickael Gastineau. [Inpop06: a new numerical planetary ephemeris](#). *Astronomy & Astrophysics*, 477(1):315–327, 2008.
- [41] 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.
- [42] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. [Mpfr: A multiple-precision binary floating-point library with correct rounding](#). *ACM Transactions on Mathematical Software (TOMS)*, 33(2):13, 2007.
- [43] Daan Frenkel and Berend Smit. [Understanding molecular simulation: from algorithms to applications](#), volume 1. Academic press, 2001.
- [44] Toshio Fukushima. [Reduction of round-off error in symplectic integrators](#). *The Astronomical Journal*, 121(3):1768, 2001.
- [45] Toshio Fukushima. [Numerical comparison of two-body regularizations](#). *The Astronomical Journal*, 133(6):2815, 2007.
- [46] Martin J Gander and Ernst Hairer. [Analysis for parareal algorithms applied to hamiltonian differential equations](#). *Journal of Computational and Applied Mathematics*, 259:2–13, 2014.
- [47] GNU. [libquadmath v 1.3](#), 2008.
- [48] Stefan Goedecker and Adolfy Hoisie. [Performance optimization of numerically intensive codes](#). SIAM, 2001.
- [49] 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.

- [50] E. Hairer and C. Lubich. *Numerical solution of ordinary differential equations*. In *The Princeton Companion to Applied Mathematics*, Nicholas J. Higham, Mark R. Dennis, Paul Glendinning, Paul A. Martin, Fadil Santosa, and Jared Tanner, editors, Princeton University Press, Princeton, NJ, USA, 2015, pages 293–305.
- [51] E Hairer, SPNørsett, and G Wanner. *Solving ordinary differential equations I: nonstiff problems*, vol. 8. 1993.
- [52] E Hairer and G Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer, Berlin, 1996.
- [53] E. Hairer and G. Wanner. *Initial value problems*. In *Encyclopedia of Applied and Computational Mathematics*, B Engquist, editor, Springer, 2015, pages 691–694.
- [54] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.
- [55] 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.
- [56] Wayne B Hayes. *Is the outer solar system chaotic?* *Nature Physics*, 3(10): 689–691, 2007.
- [57] 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.
- [58] Yozo Hida, Xiaoye S Li, and David H Bailey. *Algorithms for quad-double precision floating point arithmetic*. In *Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium on*, IEEE, 2001, pages 155–162.
- [59] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. Siam, 2002.
- [60] Nicholas J Higham. *Programming languages: An applied mathematics view*. 2015.
- [61] Leslie Hogben. *Handbook of linear algebra*. Chapman and Hall/CRC, 2013.
- [62] IEEE. *Ieee standard for floating-point arithmetic*. 2008.

- [63] 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.
- [64] Takashi Ito and Kiyotaka Tanikawa. Trends in 20th century celestial mechanics. *Publ. Natl. Astron. Obs. Jpn*, 9:55–112, 2007.
- [65] Laurent O. Jay. Preconditioning of implicit runge-kutta methods. *Scalable Computing: Practice and Experience*, 10, 2009.
- [66] Hugo Jiménez-Pérez and Jacques Laskar. [A time-parallel algorithm for almost integrable hamiltonian systems](#). *arXiv preprint arXiv:1106.3694*, 2011.
- [67] MP Calvo JM Sanz-Serna. *Numerical Hamiltonian problems*. Chapman and Hall, 1994.
- [68] 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.
- [69] W Kahan. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.
- [70] 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.
- [71] 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.
- [72] Sergei A Klioner. [Basic celestial mechanics](#). *arXiv preprint arXiv:1609.00915*, 2016.
- [73] D Knuth. The art of computer programming: Vol 2/seminumerical algorithms, 1969.
- [74] Sergei Kopeikin, Michael Efroimsky, and George Kaplan. *Relativistic celestial mechanics of the solar system*. John Wiley & Sons, 2011.

- [75] Fred T Krogh. [An adams guy does the runge-kutta](#). 1997.
- [76] 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.
- [77] MP Laburta. [Construction of starting algorithms for the rk-gauss methods](#). *Journal of computational and applied mathematics*, 90(2):239–261, 1998.
- [78] 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.
- [79] 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.
- [80] 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.
- [81] Jacques Laskar and Mickael Gastineau. [Existence of collisional trajectories of mercury, mars and venus with the earth](#). *Nature*, 459(7248):817–819, 2009.
- [82] Jacques Laskar and Philippe Robutel. [High order symplectic integrators for perturbed hamiltonian systems](#). *Celestial Mechanics and Dynamical Astronomy*, 80(1):39–62, 2001.
- [83] Benedict Leimkuhler and Sebastian Reich. *Simulating hamiltonian dynamics*, volume 14. Cambridge University Press, 2004.
- [84] Harold F Levison and Martin J Duncan. [The long-term dynamical behavior of short-period comets](#). *Icarus*, 108(1):18–36, 1994.
- [85] Werner Liniger and Ralph A Willoughby. [Efficient integration methods for stiff systems of ordinary differential equations](#). *SIAM Journal on Numerical Analysis*, 7(1):47–66, 1970.
- [86] Sanz Serna J.M. Diaz J.C Lopez Marcos, M.A. [Are gauss-legendre methods useful in molecular dynamics ?](#) *Journal of Computational and Applied Mathematics*, 1996.

- [87] 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.
- [88] Robert I McLachlan. [Composition methods in the presence of small parameters](#). *BIT Numerical Mathematics*, 35(2):258–268, 1995.
- [89] Robert I McLachlan and Pau Atela. [The accuracy of symplectic integrators](#). *Nonlinearity*, 5(2):541, 1992.
- [90] A Morbidelli. [Modern integrations of solar system dynamics](#). *Annual Review of Earth and Planetary Sciences*, 30(1):89–112, 2002.
- [91] Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, and Serge Torres. [Handbook of floating-point arithmetic](#). Springer Science & Business Media, 2009.
- [92] Nobelprize.org. [The nobel prize in chemistry 2013](#), May 2014.
- [93] Hans Olsson and Gustaf Sderlind. [The approximate runge-kutta computational process](#). *BIT Numerical Mathematics*, 40(2):351–373, 2000.
- [94] Michael L Overton. [Numerical computing with IEEE floating point arithmetic](#). Siam, 2001.
- [95] Peter Pacheco. [An introduction to parallel programming](#). Elsevier, 2011.
- [96] EV Pitjeva and NP Pitjev. [Development of planetary ephemerides epm and their applications](#). *Celestial Mechanics and Dynamical Astronomy*, 119(3-4):237–256, 2014.
- [97] Kevin P. Rauch and M. Holman. [Dynamical chaos in the wisdom-holman integrator: origins and solutions](#). 1998.
- [98] Yousef Saad. [Iterative methods for sparse linear systems](#). SIAM, 2003.
- [99] Prasenjit Saha, Joachim Stadel, and Scott Tremaine. [A parallel integration method for solar system dynamics](#). *arXiv preprint astro-ph/9605016*, 1996.
- [100] Prasenjit Saha and Scott Tremaine. [Symplectic integrators for solar system dynamics](#). *The Astronomical Journal*, 104:1633–1640, 1992.
- [101] Prasenjit Saha and Scott Tremaine. [Long-term planetary integration with individual time steps](#). *arXiv preprint astro-ph/9403057*, 1994.

- [102] Jesús María Sanz-Serna. [Symplectic integrators for hamiltonian problems: an overview](#). *Acta numerica*, 1:243–286, 1992.
- [103] JM. Sanz-Serna. [Hamiltonian systems](#). In *Encyclopedia of Applied and Computational Mathematics*, B Engquist, editor, Springer, 2015, pages 617–624.
- [104] JM. Sanz-Serna. [Symplectic methods](#). In *Encyclopedia of Applied and Computational Mathematics*, B Engquist, editor, Springer, 2015, pages 1451–1458.
- [105] 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.
- [106] Mark Sofroniou and Giulia Spaletta. [Derivation of symmetric composition constants for symmetric integrators](#). *Optimization Methods and Software*, 20(4-5):597–613, 2005.
- [107] Pat H Sterbenz. [Floating-point computation](#). Prentice Hall, 1973.
- [108] Gerald J Sussman and Jack Wisdom. [Chaotic evolution of the solar system](#). Technical report, DTIC Document, 1992.
- [109] Robert van de Geijn Victor Eijkhout and Edmond Chow. [Introduction to High Performance Scientific Computing](#). lulu.com, 2011.
- [110] 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.
- [111] Jack Wisdom. [Symplectic correctors for canonical heliocentric n-body maps](#). *The Astronomical Journal*, 131(4):2294, 2006.
- [112] 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.
- [113] Jack Wisdom and Matthew Holman. [Symplectic maps for the n-body problem](#). *The Astronomical Journal*, 102:1528–1538, 1991.
- [114] Wolfram Research, Inc. [Mathematica](#).

- [115] Dexuan Xie. A new numerical algorithm for efficiently implementing implicit runge-kutta methods. *Department of Mathematical Sciences. University of Wisconsin, Milwaukee, Wisconsin, USA*, 2009.
- [116] 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.