eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Master Amaierako Lana

Konputazio Ingeniaritza eta Sistema Adimentsuak Unibertsitate Masterra

Klepler-en ekuazioaren ebazpenerako algoritmo bektorizatuen optimizazioa Julia lenguaian

Jon Ruiz de Azua Ruiz

Zuzendariak
Ander Murua Uria

2022ko irailaren 25

Esker onak

Lerro hauetan proiektua gauzatzen lagundu didaten guztioi nire eskerrik beroenak adierazi nahi dizkiet. Aipamen berezi bat, Ander Murua proiektuko zuzendariari, bere ekarpen eta aholkuek proiektua behar bezala garatzen eta biribiltzen laguntzeagatik.

Laburpena

Mekanika klasikoko bi gorputzeko problemaren ebazpena modu bektorizatuan burutzen duen algoritmoa lortzea da lanaren helburua. Bi gorputzen problemaren soluzioa lortzeko Kepler-en ekuazioa ebatzi behar da. Soluzio analitikorik ez duenez zenbakizko metodoak erabiltzen dira. Doitasunez eta modu eraginkorrean ebazteko hainbat algoritmo ezagutzen dira.

Gaur egungo konputagailuetan geroz eta ohikoagoak dira SIMD eragiketak, *Single Instruction Multiple Data*. Kalkuluak azkartzeko hardware-an ahalbidetzen den bektorizazioa da. Eragiketak bi eskalarren (zenbaki) artean burutu ordez bektoreengan egiten ditu, emaitza ere bektore bat izanik. Denbora aldetik antzekoa da bi eskalarren eragiketa edota bektorearena. Behin eta berriz erabiltzen diren algoritmoetan eraginkortasun hobekuntza handia lortu daiteke SIMD erabiliz.

Bektorizazioari etekina ateratzeko algoritmoak ezaugarri jakin batzuk bete behar ditu. Nagusiena bektoreko elementu guztiei eragiketa berdinak aplikatzea da. Kepler-en ekuazioa ebazten duten algoritmo gehienak ez dira bektorizatzeko egokiak. Lan honetan Kepler-en ekuazioa ebazteko algoritmo berri bat optimizazio bidez nola lortu den azaltzen da. Algoritmo hau SIMD bektorizazioaz ebatzi daiteke, sarrera parametroak bi zenbaki izan ordez bakoitza n elementuko bektoreak direlarik. Eragiketa guztiak era bektorizatuan burutu ondoren emaitza ere n elementuko bektorea da. Ondoren beste algoritmo ezagunekin konparatzen da, bai doitasun eta eraginkortasun aldetik. Era sekuentzial eta bektorizatuaren arteko ezberdintasunak aztertzen dira. Azkenik bi gorputzen problema ebazterakoan era bektorialean lortzen den abantaila azaltzen da. Lan hau burutzeko garatu den kodea publikoki eskuragarri jarri da `https://github.com/JonRdA/kepler` estekan, `git` sotwarekin kudeatu daiteke.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

KAPITULUA 1

Sarrera

1.1 Kepler-en ekuazioa

Kepler-en ekuazioak bi gorputzen problemaren emaitza zehatza deskribatzen du. Posizioa denbora jakin batean kalkulatzean datza problemak, lehenik masa, posizioa eta abiadurak jakinak direlarik. Bi gorputzak indar zentral baten bidez erakarriak izan behar dute eta indarrak distantziaren karratuarekin proportzionala izan behar du. Mekanika klasikoa erabiliz Kepler-en orbita definitu daiteke, hau da, gorputzaren mugimendu eliptiko, paraboliko edo hiperbolikoa.

Johannes Kepler-ek 1609 an *Astronomia Nova* liburua argitaratu zuen, 10 urtez Marteko mugimenduak aztertzean lortutakoa azalduz. 60 urte lehenago Copernikok proposaturiko teoria heliozentrikoan oinarrituz planeten orbitak azaltzen ditu. Bertan Keplerren 3 legeetako lehenengo biak agertzen dira. Lehenengo aldiz liburuaren 60. kapituluan azaltzen da, ondoren *Epitome Astronomiae Copernicanae* liburuko 5. tomoan berriro ere azalduz.

- Planeta guztiak Eguzkiaren inguruan higitzen dira orbita eliptikoak eginez. Eguzkia elipsearen bi fokuetako batean dago.
- Planetatik Eguzkira doan irudizko lerroak azalera berdina estaltzen du denbora-tarte berdinean.

Bi lege hauen ondorioz Kepler-ek orbita eliptikoaren mugimendua era zehatzean definitzen duen ekuazioa azaldu zuen. Kepler-en ekuazioa bigarren legetik ondorioztatzen da [1].

$$E - e\sin(E) = M \tag{1.1}$$

non M Batezbesteko anomalia, e eszentrizitatea eta E anomalia eszentrikoa diren.

- **M** Batezbesteko anomalia: iragan den orbita-periodo baten zatikia, angelu gisa adierazia.

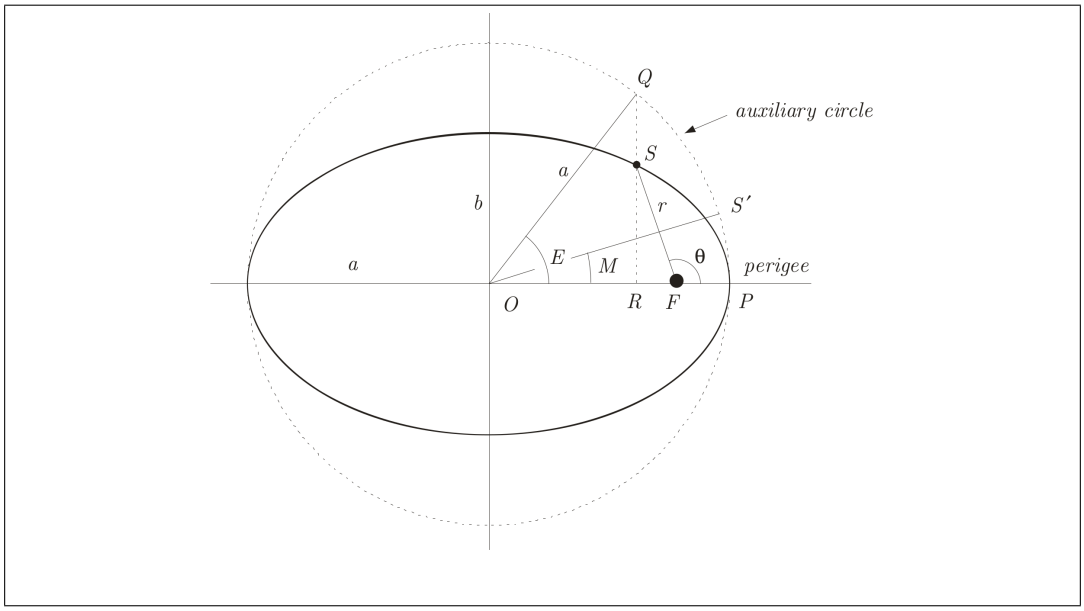
- **E** Anomalia exzentrikoa: planeta elipsea inguratzen duen zirkulu osagarrian proiektatzean, elipsearen zentrotik, perigeoeta proiekzioaren arteko angelus.

- **e** Eszentrikotasuna: kurba konikoa zirkunferentziatik zenbat aldentzen den parametroa. $(0, 1)$ inguruan egonik orbita eliptikoa da. $e = 0$ denean orbita zirkularra da eta $e = 1$ denean parabolikoa. Orbita eliptikoetan $e = \sqrt{1 - \frac{b^2}{a^2}}$ non a eta b elipsearen ardatz-erdi nagusia eta txikia diren.

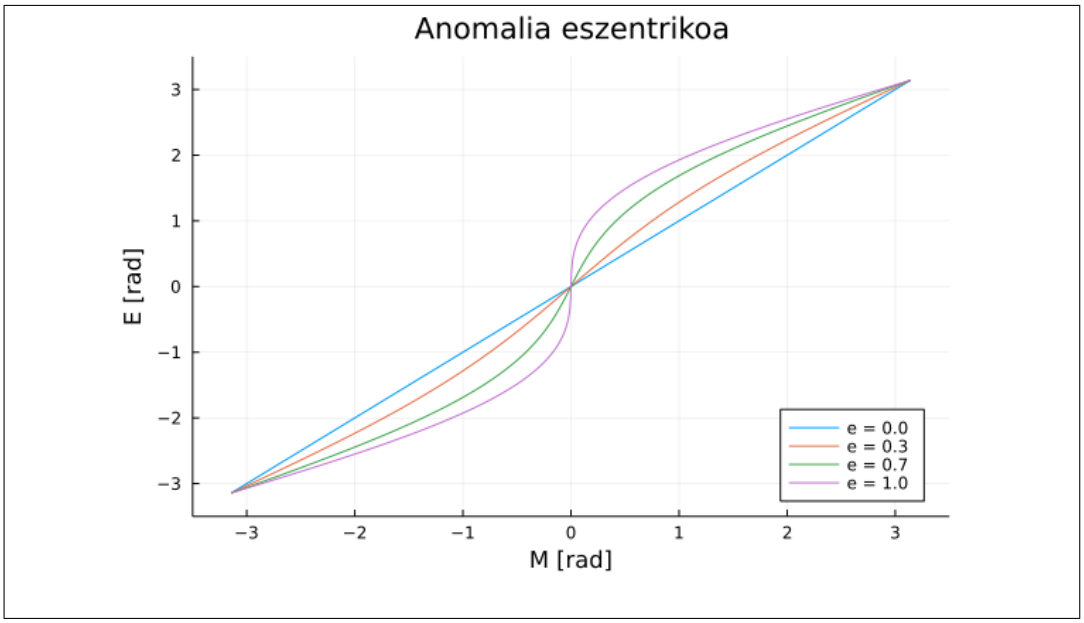
Anomalia eszentrikoa gorputz bat zehazki kokatzeko erabilgarria da. Adibidez gorputz baten posizio eta abiadura jakinik t_0 unean, $t = t_0 + \Delta t$ unean jakiteko lehenik $M = n(t - t_0)$ kalkulatzen da. Ondoren Kepler-en ekuazioa ebatziz E anomalia eszentrikoa lortzen da.

Lurraren (F) inguruan dabilen satelite baten (S)orbita eliptikoa irudikatzen da 1.1 irudian. Lurra elipsearen foko batean dago kokaturik. Zirkulu osagarria ere erakusten da, a (elipsearen ardatz handia) da bere radioa. Q puntua S gorputzaren proiekzioa da zirkulu osagarrian, elipsearen zentroa eta perigeoarekin anomalia eszentriko angelua E osatzen du. S satelitea orbita eliptikoan abiadura ezberdinean mugitzen den bitartean, fikziozko satelite bat S' zirkulu osagarrian dabil abiadura angular konstantean batezbesteko anomalia M osatuz elipsearen zentro eta perigeoarekin. Satelitea perigeoan P denean anomaliek 0 balioa dute. Eszentrizitatea $e = 0$ denean, mugimendua zirkularra baita, beraz Kepler-en ekuazioa $E = M$.

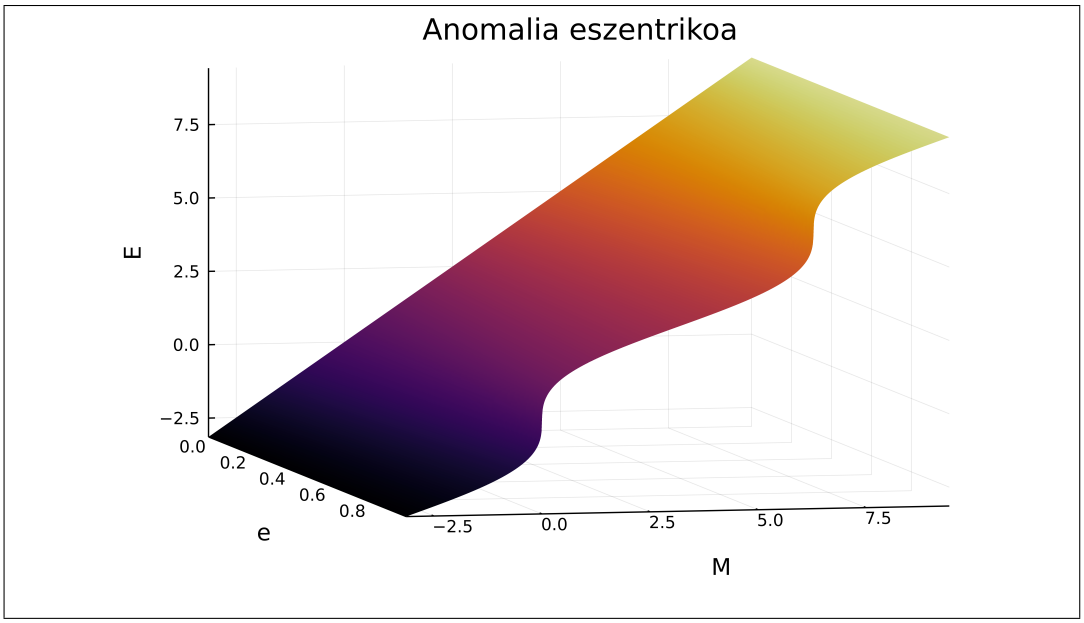
Kepler-en ekuazioa transzendentala da, ez da aljebraikoa. Bertan duen sinu funtzioa transzendentala baita. Ekuazioak ez du E-rentzako soluzio aljebraikorik. Metodo numerikoez eta serie hurbilpenez ebazten da.



1.1 Irudia: Kepler-en ekuazioaren elementuen adierazpen grafikoa



1.2 Irudia: Anomalia eszentrikoa e-ren balio ezberdinentzat



1.3 Irudia: Anomalia eszentrikoaren balioa e eta M-ren balio ezberdinentzat

1.2 Kepler-en ekuazioaren zenbakizko ebazpena

Kepler-en ekuazioa mendeetan zehar aztertua izan da zientzialari askorengandik. 1650 urtetik gaur egunera arte ia hamarkada guztietan lanak argitaratu dira, problema eta bere soluzioa aztertuz [2]. Nahiz eta soluzio bakarra duen, metodo asko daude deskribatu edo hurbiltzeko. Ebazgilearen helburu eta tresnen arabera metodo ezberdinak argitaratu dira.

1860 urtera arte zeruko mekanikarekin erlazionaturik zeuden ikerkuntza guztiak, hortik aurrera matematikako arloaren aurrerapenek metodo berriak ekarri zituzten. Astronomoen nahia ebazpen azkarra lortzea zen, emaitzaren doitasuna erabatekoa izan ez arren. Hainbat

eta hainbatetan ebatzi behar baitzuten. 1890etik aurrera emaitza konputazionalak dira nagusi. XX. mendeko bigarren erdian batez ere, prozedura optimoen bila egon dira ikerlariak, algoritmoen eraginkortasun eta duintasuna hobetsiz. Metodo gehienak iteratiboak dira, hasierako hurbilpen bat aurkituz eta hau iterazio prozesu bidez hobetuz.

$$f(E) = E - e\sin E - M \tag{1.2}$$

Anomali eszentrikoaren E balioa prozesu iteratiboen bidez kalkulatu daiteke, $f(E) = 0$ ekuazioaren soluzio errealak aurkitzeko. Hasierako hurbilpen balio batetatik abiatuz, urrats edo iterazio bakoitzean soluzioaren hurbilpena hobetzea da helburua. 1.3 ekuazioan n iterazio zenbakia da eta Δ_n hurbilpena hobetzen duen balioa. Baliteke prozesu iteratiboak emaitza konbergitzea edo ez. Metodo ezberdinen artean konparatzeko konbergentzi ratioa garrantzitsua da. Sekuentzia limitera nola gerturatzen den adierazten duen ezaugarria da. Aurrerago azaltzen den bezala metodo asko daude bai Δ_n eta lehenengo hurbilpena E_0 aukeratzeko.

$$E_{n+1} = E_n + \Delta_n \tag{1.3}$$

1.2.1 Newton-Raphson-en iterazioa

$f(E) = 0$ ekuazioaren soluzio errealak aurkitzeko Newton-Raphson metodoa erabili daiteke ekuazio iteratibo moduan. Metodo honetan Δ_n funtzioaren balioa eta deribatuarekin kalkulatzen da.

$$E_{n+1} = E_n - \frac{f(E_n)}{f'(E_n)} \tag{1.4}$$

$f'(E)$ Kepler-en ekuazioaren lehenengo deribatua E-rekiko.

$$f'(E) = \frac{d}{dE}f(E) = 1 - e\cos(E) \tag{1.5}$$

Newton-Raphson metodoaren konbergentzia hasierako hurbilpenaren menpe dago. (E_0) balio egokiekin soluziorantz konbergituko du, bestela baliteke emaitza ez aurkitzea naiz eta iterazio asko egin. Aurrerago azalduko den Julia `SatelliteToolbox` paketean dagoen ebazpen metodo iteratiboak konbergentzia ziurtatzen duen hasieraketa balioa erabiltzen du. Kasu hauetan konbergentzia koadratikoa da. Errorearen tamaina iterazio bakoitzean bere karratuarekiko proportzionala da. Iterazio batean 10^{-2} ko errorea 10^{-4} rekiko proportzionala izango da hurrengo urratsean. Lortutako hurbilpenean zuzen dauden digituen kopurua urrats bakoitzean bikoiztu egiten dela esan daiteke gutxi gora behera.

1.2.2 Halley-ren iterazioa

Halley-ren metodoak konbergentzia maila ezberdinetako iterazio Δ_n lortzen ditu. Newton-Raphson metodoaren generalizazioa da.

$$\Delta_n = (k+1)\frac{\left(\frac{1}{f(E_n)}\right)^{(k)}}{\left(\frac{1}{f(E_n)}\right)^{(k+1)}} \tag{1.6}$$

Edozein iterazio zenbakirako $n = 0, 1, 2, ...$ eta $k = 0, 1, 2, ...$ mailarako balio du 1.6 ekuazioak. Bertan k maila da eta deribatua adierazten du (k) zatikietan. Adibidez $k = 0$ rako Newton-Raphson iterazioa lortzen da.

$$\left(\frac{1}{f}\right)' = -\frac{f'}{f^2} \tag{1.7}$$

Beraz lortzen den metodoa, 1.4 ekuazioan adierazitakoa da.

$$E_{n+1} = E_n + \frac{\left(\frac{1}{f(E_n)}\right)}{\left(\frac{-f'(E_n)}{f^2(E_n)}\right)} = E_n - \frac{f(E_n)}{f'(E_n)} \tag{1.8}$$

Halley-ren δ deritzona $k = 1$ denean lortzen da, horretarako bigarren deribatua erabili behar da izendatzailean.

$$\left(\frac{1}{f}\right)'' = -\frac{-f^2 f'' + 2f(f')^2}{f^4} = \frac{2(f')^2 - ff''}{f^3} \tag{1.9}$$

Lorturiko prozesu iteratiboaren Δ_n hurrengoa izanik

$$\Delta_n \frac{2f(E_n)f'(E_n)}{2(f'(E_n))^2 - f(E_n)f''(E_n)} \tag{1.10}$$

Metodo honek konbergentzi kubikoa du. k-ren balio handiagoak erabiliz gero konbergentzia hobeak lortzen dira. Bestalde maila handiagoko deribatuak kalkulatu eta ebatzi behar dira, urrats bakoitza konplexuagoa bihurtuz. 10^{-16} ko doitasuna duten zenbakiak erabiltzean ez da eraginkorra 3. edo 4. mailako konbergentzia baino gehiago erabiltzea. Oso azkar gainditzen baita makinaren doitasun maila.

1.2.3 Iterazio sinplifikatua

Eraginkortasuna kontutan harturik prozesu iteratiboa azkartzeko iterazio sinplifikatua erabil daiteke. Iterazio batean Δ_n balio bat kalkulatu ordez bi urrats burutzean datza. Lehenengo Δ kalkulatzen den arren, bigarrena hurbildu egiten da Taylor-en polinomio bidez. Emaitzeruntz azkarrago hurbiltzen dira balioak (bi iterazio izango balira bezala) baina funtzio eta bere deribatuaren ebaluazio bakarra egiten da. Kepler-en ekuazioaren ebazpenean abantaila du funtzio ebaluaketak $sin(E_n)$ eta $cos(E_n)$ kalkulatzea suposatzen baitu. Funtzio trigonometrikoak ekiditeak prozesua azkartzen du. Ondoren iterazio batean bigarren Δ_n Newton-Raphson-ena nola hurbildu daitekeen azaltzen du. Δ_1 edozein metodo iteratibokoa izan daiteke.

$$E_2 = E_0 + \Delta_1 + \tilde{\Delta}_2 \tag{1.11}$$

$$\Delta_2 = -\frac{f(E_1)}{f'(E_1)} \tag{1.12}$$

Iterazioko bigarren Δ kalkulatzeko funtzioa eta bere deribatua E_1 -en ebaluatu beharko litzateke. Hau ekiditeko Newton-Raphson bidez hurbiltzen da. Kasu honetan Taylore-en polinomioa f -ren hirugarren deribatua erabiltzean trunkatzen da.

$$\Delta_2 = -\frac{f(E_1)}{f'(E_1)} \tag{1.13}$$

Taylor-en serie garapena hurrengoa izanik:

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{1.14}$$

Funtzioaren balioa eta bere deribatuarena E_1 Taylor-en polinomio trunkatu bezala hurbilduz:

$$f(E_1) \approx f(E_0) + f'(E_0)\Delta_1 + f''(E_0)\frac{\Delta_1^2}{2} + f'''(E_0)\frac{\Delta_1^3}{6} \tag{1.15}$$

$$f'(E_1) \approx f'(E_0) + f''(E_0)\Delta_1 + f'''(E_0)\frac{\Delta_1^2}{2} \tag{1.16}$$

Azkenik Δ_1 eta f-ren deribatuak E_0-n kalkulatu ondoren, Δ_2-ren hurbilpena:

$$\tilde{\Delta}_2 = \frac{f(E_0) + f'(E_0)\Delta_1 + f''(E_0)\frac{\Delta_1^2}{2} + f'''(E_0)\frac{\Delta_1^3}{6}}{f'(E_0) + f''(E_0)\Delta_1 + f'''(E_0)\frac{\Delta_1^2}{2}} \tag{1.17}$$

Iterazio batean Δ_1 eta Δ_2 kalkulatzeak funtzio trigonometrikoak lehenengo deltarako soilik kalkulatzea ahalbidetzen du. Bestalde funtzioaren bigarren eta hirugarren deribatuak kalkulatu behar dira E_0 -n. Kepler-en ekuazioan erraza da deribatuak jada kalkulatuak dauden $sin(E_0)$ eta $cos(E_0)$ -rekin lortzen baitira.

Δ_1 Newton-Raphson-en metodoarekin kalkulatu ezkero Δ_2 sinplifikatu daiteke. 1.17 ekuazioan izendatzaile eta zenbakitzaileari $f'(x_0)$ zatitzean, hurrengo ekuazioa lortzen da.

$$\tilde{\Delta}_2 = \frac{\frac{f(E_0)}{f'(E_0)} + \Delta_1 + ...}{\frac{f'(E_0)}{f'(E_0)} + ...} \tag{1.18}$$

Bertan ikusten den $\Delta_1 = \frac{f(E_0)}{f'(E_0)}$ denez sinplifikatu daiteke:

$$\tilde{\Delta}_2 = \frac{\frac{f''(E_0)\Delta_1^2}{2f'(E_0)} + \frac{f'''(E_0)\Delta_1^3}{6f'(E_0)}}{1 + \frac{f''(E_0)\Delta_1}{f'(E_0)} + \frac{f'''(E_0)\Delta_1^2}{2f'(E_0)}} \tag{1.19}$$

1.3 Bi gorputzen problema

Kepler-en ekuazioa ebazten duen algoritmo berri bat garatu den arren, bi gorputzen problema ebazten duen programa batean txertatzea da lanaren helburua. Prozesu guztia bektorizatzeko egokia izango da, ebazteko garaian lortzen den abantaila neurtu ahal izateko.

Mekanika klasikoan bi gorputzen problema sinplifikatuari Kepler-en problema deritzo. $t0$ unean masa, kokapena eta abiadura emanik, t unean kokapen eta abiadurak kalkulatzean datza. Gorputzen arteko interakzioa indar zentral batek definitzen du eta bere balioa distantziaren karratuarekin erlazionaturik dago. Mekanika klasikoa erabiliz soluzioa Kepler-en orbita bezala definitu daiteke.

Kepler-en problema arlo askotan agertzen da, Newton-ek definituko grabitate legean indarrak distantziaren karratuarekin erlazionaturik daudenez, zeruko mekanika problemetan oso ohikoa da. Adibidez, satelite batek planeta baten inguruan duen orbita deskribatzen du, planeta baten orbita bere eguzkiaren inguruan edota bi izarren orbitak beraien inguruan.

Zeruko mekanikan Kepler-en bi gorputzen problemna aldagai unibertsalekin adierazten da, lehenengo atalean azterturiko Kepler-en ekuazioaren generalizazioa da. Orbita eliptiko ($e < 1$), paraboliko ($e = 1$), eta hiperbolikoak ($e < 1$) deskribatzen ditu. Horretarako aldagai berri bat definitu behar da s. Bi gorputzen arteko indarra 1.20 ekuazioan definitzen da, distantziaren karratuarekiko alderantzizko proportzionala da. Bertan k konstante bat da, $k > 0$ indar erakorra, eta r bi gorputzen arteko distantzia. $\hat{r}$, unitate bektorea da, bi gorputzen arteko norabide eta noranzkoa duena.

$$F = \frac{k}{r^2}\hat{r} \tag{1.20}$$

Hurrengo ekuazio direrentzialean definitzen da s aldagaia. $r = r(t)$ denboraren menpe dagoen bi gorputzen masa zentroaren arteko distantzia da.

$$\frac{df}{dt} = \frac{1}{r} \tag{1.21}$$

Bi gorputzen problema sinplifikatuaren soluzioa deskribatzeko **f** eta **g** funtzioen formulazioa erabiliko da. Ondorengo baldintzak bete behar dira.

- Gorputz txikiaren (erakarria dena) masa mespretxagarria da gorputz handiarenaren aldean.
- Aukeraturiko erreferentzia sistema koordinatua inertziala da eta gorputz handiaren masa zentruan kokaturik dago.
- Gorputz handiak masa banaketa uniformea du, esferikoki simetrikoa izanik. Honek masak bi puntuetan kontzentraturik irudikatzea ahalbidetzen du.
- Ez dago beste indarrik sisteman eragintzen.

Gorputza orbita jakin batean dagoenean bere kokapen (q) eta abiadura (v) bektoreak definitzen dira. Hiru dimentsioetako bektoreak, $q, v \in \mathbb{R}^3$. v abiadura kokapenaren deribatua da denborarekiko. $t = 0$ hasierako unean ezagunak dira bi bektoreen balioak. Indarra erakargarria denez $k > 0$ da eta $\|\cdot\|$ -ek norma euklideoa definitzen du.

$$\frac{d}{dt}q = v, \qquad \frac{d}{dt}v = -\frac{k}{\|q\|^3}q, \qquad q(0) = q_0, \qquad v(0) = v_0 \tag{1.22}$$

Jakina da denbora eta posizioa kalkulatzeko hurengo erlazioak:

$$q(t) = f q_0 + g v_0 \tag{1.23}$$

$$v(t) = \dot{f} q_0 + \dot{g} v_0 \tag{1.24}$$

f eta g funtzioak G (1.32 ekuazioan definiturik) funtzioen menpe daude hurrengo erlazioarekin.

$$f = 1 - \frac{M G_2(X, \beta)}{r_0} \tag{1.25}$$

$$\dot{f} = -\frac{M G_1(X, \beta)}{r_0 r_t} \tag{1.26}$$

$$g = t - M G_3(X, \beta) \tag{1.27}$$

$$\dot{g} = 1 - \frac{M G_2(X, \beta)}{r(t)} \tag{1.28}$$

Hasierako kokapena eta abiadura (r_0, v_0) jakinik, β, η_0 eta ζ_0 balioak kalkulatzen dira.

$$r_0 = \|q_0\|, \quad \eta = q_0 \cdot v_0, \quad \beta = \frac{2k}{r_0} - \|v_0\|^2, \quad \zeta = k - \beta\, r_0. \tag{1.29}$$

Ondoren Kepler-en ekuazioa aldagai unibertsaletan ebazten da X -en balioarentzat $t(X) = t - t_0$ -rako.

$$t(X) = r_0\, X + \eta\, G_2(\beta, X) + \zeta\, G_3(\beta, X) \tag{1.30}$$

β energiarekin erlazionaturik dagoenez, energia negatiboko balioentzako ebatziko da $\beta > 0$. Kasu honetan 1.22 ekuazioaren soluzioak orbita eliptikoa du eta bere eszentrititatea e honakoa da:

$$0 \leq e = \frac{1}{k}\sqrt{\zeta^2 + \beta\, \eta^2} < 1. \tag{1.31}$$

Orbita eliptikoaren kasu honetarako G funtzioak hauek dira.

$$G_0(\beta, X) = \cos(\sqrt{\beta} X) \tag{1.32}$$

$$G_1(\beta, X) = \frac{\sin(\sqrt{\beta} X)}{\sqrt{\beta}} \tag{1.33}$$

$$G_2(\beta, X) = \frac{1 - G_0(\beta, X))}{\beta} \tag{1.34}$$

$$G_3(\beta, X) = \frac{X - G_1(\beta, X))}{\beta} \tag{1.35}$$

Erreferentzia gorputz handiaren grabitate zentruan kokaturik dagoenez, gorputz txikiaren kokapenaren norma distantzia da. $r(t) = \|q(t)\|$ kalkulatuz.

$$r(t) = r_0 + \eta\, G_1(\beta, X) + \zeta\, G_2(\beta, X). \tag{1.36}$$

f eta g funtzioetatik aldagai aldaketa baten bidez b koefizienteak erabiltzen dira.

$$b_{11} = f(\beta, X) - 1 \tag{1.37}$$
$$b_{12} = g(\beta, X) \tag{1.38}$$
$$b_{21} = \dot{f}(\beta, X) \tag{1.39}$$
$$b_{22} = \dot{g}(\beta, X) - 1 \tag{1.40}$$

b_{ij} koefizienteak kalkulatzen dira abiadura eta kokapen berriak lortzeko

$$b_{11} = -\frac{k}{r_0}\, G_2(\beta, X) \tag{1.41}$$
$$b_{12} = r_0\, G_1(\beta, X) + \eta\, G_2(\beta, X) \tag{1.42}$$
$$b_{21} = -\frac{k}{r_0\, r(t)}\, G_1(\beta, X) \tag{1.43}$$
$$b_{22} = -\frac{k}{r(t)}\, G_2(\beta, X) \tag{1.44}$$

Kepler-en ekuazioa aldagai unibertsaletatik lehenengo atalean azaldutako Kepler-en ekuazio sinplifikatuara eraldatu behar da. $\beta, r_0, \eta, t \in \mathbb{R}$, jakinik, 1.30 ekuazioa $F(X) = 0$ itxuran berridatziz, $\zeta = k - \beta\, r_0$ delarik.

$$F(X) := r_0\, X + \eta\, G_2(\beta, X) + \zeta\, G_3(\beta, X), \tag{1.45}$$

Kepler-en ekuazio sinplifikatua hau delarik

$$E - e\, \sin(E) = M \tag{1.46}$$

Ondorengo aldagai aldaketa eginik.

$$E = E_0 + \sqrt{\beta}\, X \tag{1.47}$$

$f(X)$ funtzioa lortzen da

$$F(X) = k\, \beta^{-3/2} f(E_0 + \sqrt{\beta}\, X) \tag{1.48}$$

Bertako elementuak hauek direlarik

$$r_0 = k\, \beta^{-1}\, (1 - e\, \cos(E_0)), \tag{1.49}$$
$$\eta = k\, \beta^{-1/2}\, e\, \sin(E_0), \tag{1.50}$$
$$M_0 = E_0 - e\, \sin(E_0), \tag{1.51}$$
$$t = (M - M_0)\, k\, \beta^{-3/2}. \tag{1.52}$$

Kepler-en ekuazio estandarrari $E = M + x$ aldagai aldaketa aplikatu ezkero hurrengoa lortzen da, $a = e\ \sin(M)$ eta $b = e\ \cos(M)$ direlarik.

$$x - a\ \cos(x) - b\ \sin(x) = 0 \tag{1.53}$$

1.30 ekuazioa 1.53 bezala berridatzi daiteke ondorengo aldagaiak aldaturik.

$$x = \sqrt{\beta}\, X - x_0, \quad \text{where} \quad x_0 = \frac{\beta\, t - \eta}{k}\, \sqrt{\beta}. \tag{1.54}$$

Zehazki, $t, \eta, \zeta \in \mathbb{R}$ eta $k, \beta, r_0 > 0$ jakinik, a and b kalkulatu daitezke hurrengo ekuazioan adierazten den bezala.

$$a = \frac{\sqrt{\beta}\, \eta\ \cos(x_0) + \zeta\ \sin(x_0)}{k}, \tag{1.55}$$

$$b = \frac{-\sqrt{\beta}\, \eta\ \sin(x_0) + \zeta\ \cos(x_0)}{k}. \tag{1.56}$$

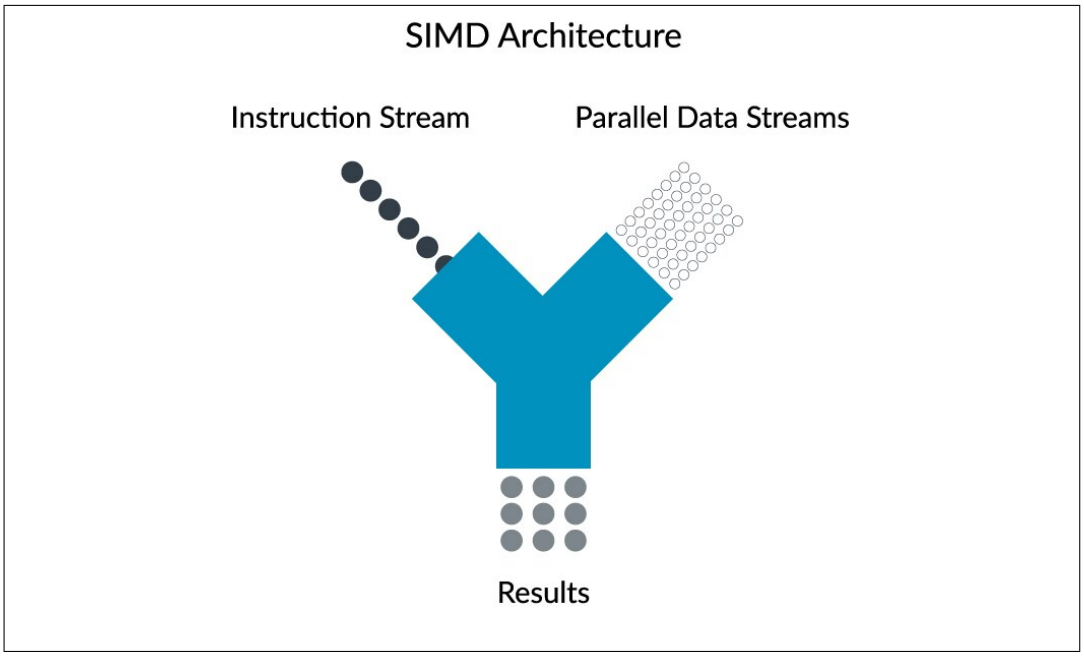
Behin a eta b lortu direlarik, Kepler-en ekuazio estandarra ebazten da, bertatik X lortzen delarik. Honekin azken pausoa b koefizienteak lortzea da, abiadura eta kokapen berriak kalkulatzeko.

1.4 Single Instruction Multiple Data

SIMD, ingelesez *Single Instruction Multiple Data*-k bektoreengan operatzen duen exekuzio modelo bat deskribatzen du. Eragiketa tradizionalak era sekuentzialean burutzean, instrukzio bakoitzak balioak banaka jasotzen ditu eta emaitza ere balio bakarra da. SIMD modeloan ordea operazioak bektoreak jasotzen ditu eta lorturiko emaitzak ere bektoreak dira.

Operazio mota hauek ahalbidetzen duen atala hardware-a da. Hain zuzen ere prozesagailuaren ISA *Instruction Set Architecture*. Gaur egungo CPU gehienek (bai ordenagailu, smartphone, tablet...) SIMD instrukzioak burutzeko hardware-a dute. Ohikoak dira AVX moduluak *Advanced Vector Extension* dituzten prozesagailuak. Lan honetako bektorizazio probak Intel AVX-512 SIMD instrukzio set-arekin egin dira. Ondorengo zerrendan garatzen joan diren prozesagailuen atalak, saltze datarekin batera.

- Intel MMX (64 bit , 1996)
- AMD 3DNow! (128 bit , 1998)
- Intel Streaming SIMD Extensions (SSE, 128 bit, 1999)
- Freescale/IBM/Apple AltiVec (also VMX, 128 bit, 1999)
- ARM NEON (128 bit, 2005)
- Intel Advanced Vector Extensions (AVX, 256 bit, 2011)
- Intel Larrabee New Instructions (LRBni, 512 bit, 2013).



1.4 Irudia: Bektorialki buruturiko eragiketa fluxua

- Intel AVX-512 (512 bit, 2015)

Eragiketa bektorizatuak SIMD memoria erregistroetan burutzen dira. Doitasun soileko zenbakien tamaina (32 bit) baina handiagoak dira (2 eta 16 aldiz handiagoak, ohikoenak 128, 256 eta 512 bit-ekoak). Hala ere, instrukzio bektorizatuak burutzea eskalar batenak burutzea bezain azkarra da [3]. Adibidez, doitasun bikoitzeko bi eskalar biderkatzea eta bi bektore (doitasun bikoitzeko 8 eskalarrekin bakoitza, 512 bit) biderkatzeak denbora antzekoa behar dute. Eragiketa modelo hau programa batean txertatzeak eraginkortasunean inpaktu handia dauka. Modu sekuentzialean egiten diren kalkuluen memoria kudeaketa hobetzen du SIMD unitateak. Memoria eragiketak (gorde eta kargatzea) ez du elementuka burutzen, bektore osoarena aldi berean baizik.

1.4.1 LLVM IR kodea

LLVM IR (*Low Level Virtual Machine Intermediate Repressentation* maila baxuko kodea da, *assembler* estilokoa. Hardware-ak ulertzen duen makina kodetik gertu dago. Normalean automatikoki sortzen da konpilazio prozesuetan, adibidez, Julian. Bertan ez daude lengoai ezaugarri konplexurik (if-else, klaseak ...). LLVM IR kodean instrukzio bakoitzak operazio bat deskribatzen du "\%"sinboloarekin hasita. LLVM IR kodean definituriko datu motetan SIMD memoria erregistroetan gordeta dauden aldagaiak azaltzen dira. 1.1 taulan agertzen diren bektoreak erabiliko dira funtzioa bektorizatzeko garaian, batez ere `<8 x float>` eta `<8 x double>`.

1.4.2 SIMD baldintzak

SIMD bidez lortu daitezken abantaila guztiak erabiltzeko kodeak baldintza jakin batzuk bete behar ditu. Eragiketa guztiak ez dira hain eraginkorrak eta baliteke kasu batzuetan modu

sekuentzialean burutu beharra. Garatu beharreko kodea bektorizatzeko egokia izan dadin baldintzak betetzea garrantzitsua da. Kepler-en ekuazioa ebazteko metodologia egokia aukeratzea behartu du. 2 atalean azaltzen den bezala argitaratutako algoritmo gehienak ez dira bektorizatzeko egokiak.

- Bektoreko elementu guztien gain eragiketa berdina burute behar da. Elementu bakoitzean eragiketa ezberdina burutzea ezin baita SIMD bidez lortu.
- *Branch-free* programak. Adarretan banatutako programa. Programatzerakoan ohikoak diren `if-else` operazio batzuk adarretan banatzen dute exekuzioa, hau ezin da modu eraginkorrean SIMD bidez bete [4]. Programan baldintza bat ezartzean eragiketa sekuentzia bitan banatzen da, demagun "A"eta "B"adarrak. Arkitekturaren arabera baliteke era bektorizatuan baldintza burutzea, *Mask* erabiliz. SIMD-ek booleano bektore berri bat sortzen du baldintzaren emaitzarekin. Bertan ze elementuk "A"adarra hartu duen eta zeinek "B"gordeta geratzen da. Ondoren elementu guztiei bi adarren eragiketak aplikatzen dizkie separaturik, bi emaitza bektore lorturik. Azkenik bektore booleanoa erabiliz elementu bakoitzari "A"adarreko emaitza edo "B-koa dagokion ikusten du. Beste aukera baldintza bektorea modu sekuentzialean exekutatzea da. Bi aukerak eraginkortasun galera bat suposatzen dute, eragiketen fluxua moteldu egiten da, nahiz eta bektorialki operatzen jarraitu. Bektore guztiari ez zaizkio eragiketa berdinak aplikatzen, beraz lehenengo baldintzaren ondorio bat da bigarren hau.

1.4.3 Julia

Master amaierako lanean erabiliko den programazio lengoaia Julia da. Kepler-en ekuazioa ebazten duen funtzioa (era sekuentzial eta bektorialean) eta ebaluazio proba guztiak Julia erabiliz programatu dira. Julia lengoaian idatziriko kodea, exekuzio garaian, JIT *Just in Time* konpilatzen da. Programaturiko instrukzioak makina kodera transformatzen dira, hardware-ak ulertu eta exekutatu dezan. Konpilazio prozesua konplexua da eta ez da lan honen helburua aztertzea, hala ere, atal batzuetan konprobaketak egin dira. Konpilazioa lau pausotan ematen da, bakoitzean kode berri bat sortuz.

LLVM	Julia	Azalpena
i1	bool	true edo false
i8	char	karakter bakarra
i32	Int32	32bit osoa
i64	Int64	64bit osoa
float	Float32	32bit doitasun soileko dezimala
double	Float64	64bit doitasun bikoitzeko dezimala
<4 x i32>	VecElement{4, Int32}	4 doitasun soileko osoen bektorea
<4 x float>	VecElement{4, Float32}	4 doitasun soileko hamartarren bektorea
<4 x double>	VecElement{4, Float64}	4 doitasun bikoitzeko hamartarren bektorea
<8 x i32>	VecElement{8, Int32}	8 doitasun soileko osoen bektorea
<8 x float>	VecElement{8, Float32}	8 doitasun soileko hamartarren bektorea
<8 x double>	VecElement{8, Float64}	8 doitasun bikoitzeko hamartarren bektorea

1.1 Taula: LLVM eta Julia-ren data motak.

- Lowered code
- Typed code
- LLVM IR code
- Native code

Konpilazioaren urrats bakoitzeko kodea aztertzeko aukerak ematen ditu Juliak. Eragiketak SIMD modelo bidez burutzen ari diren ikusteko LLVM IR kodean ikusi daitezke. `@code_llvm` makroa funtzio bati aplikatu ezkero, Juliak LLVM kodea testualki eta era ulergarri batean azaltzen du. LLVM kodea ez da programazio lengoaia bezala irakurtzen, maila baxuko instrukzioak baitira. 1.1 taulan aipaturiko bektoreak azaltzen dira eta bektorizazioa konpilazio garaian burutu dela konprobatzeko LLVM IR kodea aztertu da. Hurrengo pausoaren kodea ikusi ahal izateko `@code_native` makroa erabili daiteke. Instrukzioak maila baxuagokoak dira, bertan hardware-ak onartzen duen kodea dago. Makina batetik bestera Native Code ezberdina izan daiteke, eta bertan aurkitzen da benetan bektorizadu den edo ez.

Julian idatzitako kodea bektorizatu daiteke. Konpilatzaileak automatikoki burutu dezaken optimizazio bat da. Hala ere, ziur egoteko esplizituki programatzea komenigarria da. Bektorizazioa burutzeko data mota berezi bat du, `VecElement{T}`. Konpilazio garaian LLVM `vector` batera bihurtzen du. Bektore hauen arteko eragiketak SIMD erabiliz burutzen direla ziurtatzeko, LLVM kode egokia sortzen duten paketeak erabili behar dira.

Pakete horietako bat `SIMD.jl` da. Bertan idatzitako kodean esplizituki adierazten da bektorizatua izan behar duela. SIMD bektore mota definitzen du `Vec{}`, erabiltzerakoan konpilatzaileari adieraziz eragiketak era bektorialean egiteko.

`SIMD.jl` paketeak Julia lengoaian programaturiko instrukzioak era bektorizatuan betetzeko LLVM kodea sortzen du. Era egokian programatzen bada, `Vec{}` bektoreak erabiliz, konpilazio prozesuan maila baxuko instrukzioak bektorizatzen ditu. LLVM kodea era bektorizatuan egotea ziurtatzen du. Hala ere baliteke makinak SIMD hardware-a ez eduki eta paketeak LLVM bektorizatua sortzea Hurrengo pausoan, Native Code sortzerakoan egokitzen da kodea hardware-era. Beraz LLVM kodean Juliaren konpilazio egokia konproba daiteke, ea ondo programatu den. Native Code-n ordea, programa hori nola exekutatzen ari den.

- `Vec{}` bektoreetan erabil daitezken data motak: `Bool Int{8, 16, 32, 64, 128} UInt{8, 16, 32, 64, 128} Float{16, 32, 64}`
- Bektorialki bete daitezkeen oinarrizko eragiketak: `+ - * / % ^ ! ~ & $ << >> >>> == != < <= > >=`
- Bektorialki exekutatu daitezkeen funtzioak: `abs ceil copysign cos div exp exp10 exp2 flipsign floor fma inv isfinite isinf isnan issubnormal log log10 log2 muladd rem round sign signbit sin sqrt trunc vifelse`

1.5 Zenbakizko ebazpen bektorizatua

Lan honen helburua Kepler-en ekuazioa era bektorizatuan ebazten duen algoritmo bat garatzea eta bi gorputzen problemaren soluzioa kalkulatzen duen programa batean integratzea da. Kapitulu honen ataletan Kepler-en ekuazioa eta bi gorputzen problema azaldu dira. Baita erabiliko diren tresnak, Julia lengoaia eta SIMD bektorizazioa.

Kepler-en ekuazioaren ebazpena zenbakizko prozesu baten bidez burutzen denez, eragiketa anitz burutu behar dira. Problemen soluzioa bilatzerakoan, adibidez orbitak kalkulatzerakoan, ekuazioa ez da behin ebazten, askotan baizik. Era bektorizatuan burutu ezkero, funtzioaren exekuzio bakarrean bektoreen emaitza lortu daiteke. 512 bit eko SIMD unitateetan:

- 8 doitasun bikoitzeko (64 bit) zenbakiak dituen bektoreekin.
- 16 doitasun soileko (32 bit) zenbakiak dituen bektoreekin.

Bektorizatu ahal izateko ebazpen funtzioan sartzen diren elementu guztiek eragiketa berdinak egin behar dituzte. Kodearen exekuzioa "adar"ezberdinetan banatzen duen baldintzarik ezin da onartu, SIMD eten edota eraginkortasuna galtzen baita. Bai ordea elementu guztiak banatzen dituen baldintzak (adibidez iterazioak eteten dituen doitasun baldintzak). Algoritmoak jarraia izan behar du, programatzerakoan ezin da `if-else` instrukziorik egon parametroen menpe exekuzioa zatitzen duenik.

Hurrengo kapituluan ikusiko den bezala ebazpen metodo asko iteratiboak dira. Iterazio bakoitzean emaitzaren ontasuna konprobatu ondoren baldintzak dituzte (emaitza ona da eta eten prozesua edo jarraitu iteratzen). Sarrera parametroen arabera (e eta M) eragiketa kopurua ezberdina da. Kasu hau bektorizadu daiteken arren ez da optimoa, bektorearen elementurik okerrenak baldintza bete arte guztiak iteratzen baitira.

Kepler-en ekuazioa ebazteko beste ikuspuntu bat, e eta M -k sortzen duten planoa eremutan banatzean datza. Eremu bakoitzean ahalik eta eragiketa gutxienak eginez soluzioa lortzen da. Kasu hauetan eragiketak desberdinak dira parametroen arabera. 1.5 irudian eskematikoki azaltzen den bezala, e eta M parametroak doitasun bikoitzeko 8 elementuko bektoreak dira. SIMD eraginkorra izan dadin elementu guztiak eragiketa berdinak bete behar dituzte. Bektorearen 3. eta 5. elementuek sorturiko (e, M) pareak eremu ezberdinetakoak badira ezin da bektorizaziorik lortu.

Metodo honen bidez lorturiko instrukzioak bektorizagarriak dira. Kepler-en ekuazioaren emaitzaren konplexutasuna ikusirik, (e, M) balio guztientzat emaitza onak ematen dituen baldintza gabeko prozesuak eragiketa asko edukiko ditu. Ez da metodo iteratibo edo eremua zatitzen dutenak bezain azkarra izango kalkulua behin egin ezkero. Hala ere, askotan ebaztea behar denetan bektorizazioari esker abantaila izango du beste soluzioekiko.

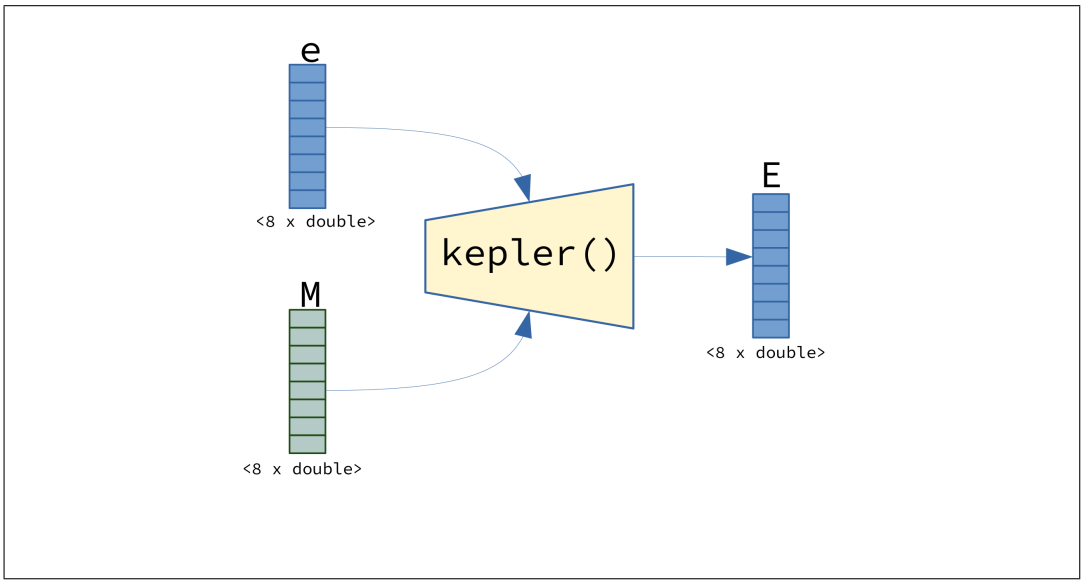
Soluzioa modu eraginkorrean eta ahalik eta eragiketa gutxien erabiliz lortzeko bi erara planteatu daiteke algoritmoa.

- Hasieraketa balio ona topatu eta ondorengo hobekuntza sinplea izatea. Bertan lehenengo urratsak kalkulu asko beharko ditu, bigarrenean eraldaketa gutxiekin emaitza onera lortzeko. Emaitzaren lehen hurbilpenetik iterazio gutxirekin lortuko litzate-

ke emaitza, iterazio bat edo bi gehienez. Hasieraketa oso ona izan ezkero baliteke bigarren urratsaren beharrik ez izatea.

- Hasieraketa balio nahiko on bat definitu (emaitzara gehiegi gerturatu gabe) eta bigarren hobekuntza urratsa konplexuagoa. Bertan iterazioak eragiketa gehiago edukiko ditu eta zenbaki finkoa altua izango da.

Hurrengo kapituluetan Kepler-en ekuazioa era bektorizatuan ebazten duen algoritmoa deskribatzen da, baita optimizazio bidez nola lortu den ere. Ohikoenak diren metodoetan oinarritzea ez da posible izan, hauek SIMD-k eskatzen dituen baldintzak ez baitituzte betetzen.



1.5 Irudia: Kepler-en ekuazioa bektorialki ebazten duen funtzioaren eskema.

KAPITULUA 2

Ebazpen algoritmoak

2.1 Metodo argitaratuak

Gaur egun erabiltzen diren ebazpen metodoetan doitasuna eta eraginkortasuna dira ezaugarririk garrantzitsuenak. Algoritmo analitikoak lortu daitezken arren [5], metodo gehienak zenbakizko ebazpenean oinarritzen dira.

- Mortari eta Elipek [6] proposaturiko metodoan 2 eremu definitzen dira batazbesteko anomaliarentzat. Eremu bakoitzaren arabera metodoa desberdina da ebazteko. Eremu baxuena bi funtzio inplizituen bidez lortzen da. Balio altuko eremua ordea, Newton-Raphson metodoa erabiliz, eremuaren limite baxua hasierako baliotzat hartuz lortzen da. Makinaren doitasun maila ez bada lortzen iterazioa errepikatzen da lortu arte.

- Markley [7] metodoan hasieraketa balioa ekuazio kubiko bat ebatziz lortzen da, bertako $\sin(E)$ Pade-ren hurbilpenarekin ebaluatuz. Orduan 5. mailako metodo baten urrats batekin emaitza hobetzen da. Soilik 4 funtzio transzendental ebaluatzen ditu metodoak, erro karratua, erro kubikoa eta bi funtzio trigonometriko.

- Serafin-en [8] ikerketan proposaturiko metodoak anomalia eszentrikoarentzako limiteak definitzen ditu, inekuazio batzuen baldintzak betez. Interbaloaren limiteak hobetzen ditu bi metodo linealak iteratiboki erabiliz.

- Gooding eta Odell-ek [9] proposaturiko metodoan, hasierako E_0 balioa ekuazio kubiko bat ebatziz lortzen da. Ondoren hurbilpena hobetzeko 2 iterazio burutzen dira. Iterazio bakoitzean Halley-ren δ eta Newton-Raphson en δ erabiltzen dira. Azken hau, Taylor serie bidez hurbilduz, $\sin$ funtzioak berriro ez ebaluatzeko.

- Fukushima-k [10] proposaturiko metodoan hasieraketako E_0 balioa soluzioaren balio maximoa da, π. Hurbilpen hau hobetzeko Newton-Raphson metodoa transformaturik erabiltzen da. E ren eremu jarraitik j -ko eremu diskretu batera j eramanik. j-ren balioak soluzioaren indizeak dira. Emaitza hobetzeko Newton-Raphson metodoa erabiltzen da Taylorren serie garapen trunkatuak erabiliz iteratiboki. Ez da funtzio transzendentalik erabiltzen metodo guztian.

- Raposo-Pulido eta Pelaez-ek [11] argitaratutako ikerketan hasieraketa lortzeko 12 eremu ezberdin definitzen dituzte anomalia eszentrikoarentzat. Eremu bakoitzean 5. mailako polinomio bidez interpolatzen da hasierako E_0 balioa. Ondoren Newton-Raphson metodoarekin hurbilpena hobetzeko.

- Nijenhuis [12] proposatutako metodo numerikoa iteratiboa da. Bertan hasierako E_0 balioa kalkulatzen da. (M, e)k sorturiko espazioan lau ingurune desberdintzen dira eta hauetako bakoitzak hasierako balioa lortzeko prozesu bat erabiltzen du. 3 inguruetan hasierako balioa hobetzeko Halley-ren iterazio bat erabiltzen da, bertako $\sin$ funtzioa hurbilpen batekin ordezkatuz. 4. inguruak puntu singularra du barne eta hasierako balioa Mikkola-ren ekuazio kubikoak ematen du, Newton-en iterazioz hobetua. Hasieraketa hobetzeko Newton-en m mailako metodoa erabiltzen du, m lortu nahi den doitasunarekin definitzen da.

Aztertutako metodoetan eraginkortasunari eta doitasunari erreferentzi asko egiten zaio, hauek baitira algoritmoak ebaluatzeko irizpide nagusiak. SIMD edo bektorizazioa aipatzen den metodorik ez da aurkitu. Are gehiago, metodoak aztertzerakoan guztiz bektorizatzeko egokiak ez direla ikusi da. Nahiz eta zati batzuk bektorizagarriak izan, kasu gehienetan baldintzak erabiltzen dituzte. Goodin eta Odell-en ikerketak [13][9] salbuespena dira, proposatzen duten ebazpen metodoa ez baita zatitan banatzen (e, M) planoa, ezta iterazio kopuru aldakorrik erabiltzen. Bektorizatzeko egokitasun hau ikusita sakon aztertu dira proposatutako metodoak, hasieraketa balio eta iterazio motak.

2.2 Ekepl

Gooding eta Odell-ek argitaratutako ikerketek [13][9] Kepler-en ekuazio eliptiko eta hiperbolikoaren ebazpena aztertzen dute. **Ekepl** deituriko funtzioak bi urratsetan lortzen du Kepler-en ekuazioaren soluzioa. Prozesu iteratibo baten eskema jarraituz lehenik hasieraketa balio bat kalkulatzen da, E_0, ondoren balio hau hobetuz emaitza lortuz. Hasieraketa bi iterazioz hobetzen den arren, iterazio zenbakia finkoa da, beraz ez dago baldintzarik. Hasieraketa balioak lortzeko metodo ezberdinak aztertzen dituzte, baita hau hobetzeko prozesuak (Newton-en iterazioa, Halley-rena...) Matematikoki metodoa garatzeaz gain programatua ere argitaratua dago, `Fortran 77` lengoaian idatzita. Kodea garatzean konputazioari dagokion biribiltze erroreak aztertzen dituzte. Zenbakizko errepresentazio finituak eragiketetan sar ditzakeen erroreak ekiditeko prestatua dute kode eta metodoa.

$$E - e\sin(E) = M \tag{2.1}$$

Gooding eta Odell-ek proposaturiko metodoan Kepler-en ekuazio eliptikoa 2.1 ebazten da. Bertan M $[0, \pi]$ eremuan dago eta e $[0, 1)$. Prozesua edozein M baliorentzako baliogarria da, $M > \pi$ edota $M < 0$ bada lehenik transformazio bat egin beharra dago $M[0, \pi]$ -ra murrizteko. Kepler-en ekuazioa periodikoa denez, kalkulua egin ondoren transformazioa desegiten da. Lehenik E_0 hasieraketa balio E_0 egokia lortu behar da. Horretarako Kepler-en ekuazioa hurbiltzen duen ekuazio kubiko bat 2.2 ebazten da. Horretarako erraz prestatua dagoen bat proposatzen dute. Koefiziente koadratikoa zero bihurtuz ekuazio kubiko deprimitua erabiltzen dute. Definitzeko erabiltzen diren koefizienteak prestaturik daude Cardanoren formula aplikatzean eragiketa sinpleak lortzeko. 2.2 ekuazioan q eta r definitu

behar dira, ekuazio kubikoa Kepler-ren ekuazioaren hurbilpen egokia izan dadin.

$$E_0^3 + 3qE_0 - 2r = 0 \tag{2.2}$$

Proposaturiko balioak honakoak dira:

$$q = 2\frac{e_1}{e} \qquad r = 3\frac{M}{e} \qquad e_1 = 1 - e \tag{2.3}$$

Ekuazioa ebazteko s aldagai auxiliarra erabiltzen da.

$$s = \sqrt[3]{\sqrt{r^2 + q^3} + r} \tag{2.4}$$

$$E_0 = \frac{2r}{s^2 + q + (\frac{q}{s})^2} \tag{2.5}$$

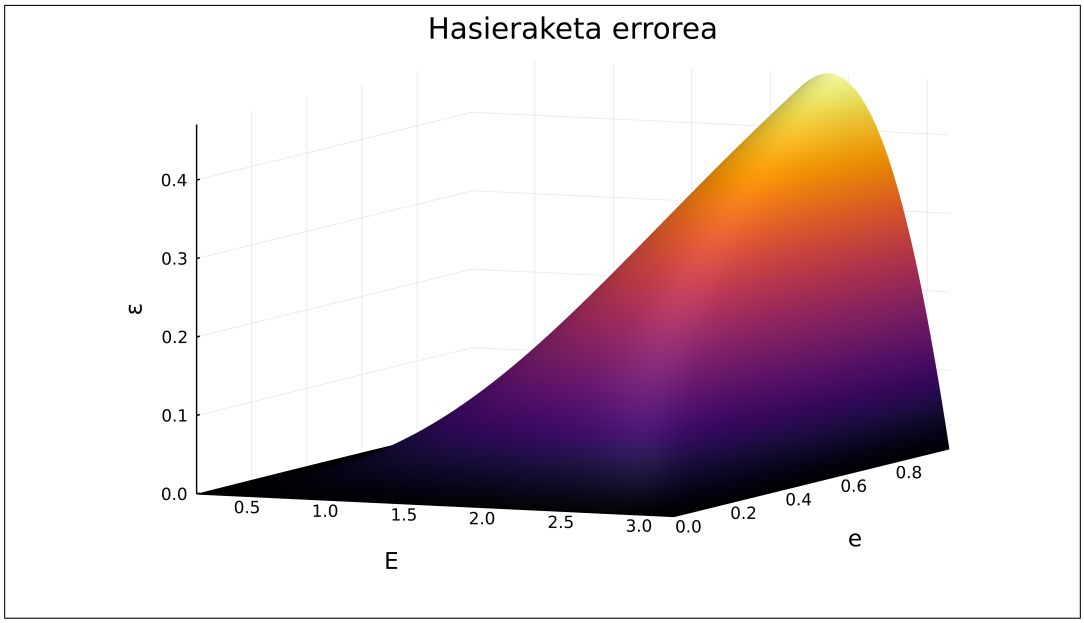
Hasieraketa balioa, 2.5 ekuazioan lortua, hurbilpen ona den jakiteko aztertu egin da. Ebaluaketetarako bi parametroekin dituen emaitzak kalkulatu nahi izan dira, beraz e eta M-k osatzen duten planoa diskretuki zatitu da. Parametro bakoitzari 1000 baliotan zatitu da, guztira 101 000 (e, M) balio ezberdin sorturik. Bikote bakoitzarentzat E_0 kalkulatu da eta ondoren errorea $\epsilon = |E - E_0|$. 2.1 irudian dauden balioak, hurrengo urratsean emaitzara gerturatzeko balioa zenbat hobetu behar den dira.

$$e_i = \frac{i}{1000} \quad i = 1, 2, ..., 1001 \tag{2.6}$$

$$M_j = \frac{\pi}{1000} \quad j = 1, 2, ..., 1000 \tag{2.7}$$

Hasieraketa balioa lortu ondoren, hurbilpen hau hobetzean datza bigarren urratsak. Horretarako metodo iteratiboetan (1.3 ekuazioa) oinarrituriko Δ_n balioak aztertzen dituzte. Kasu honetan iterazio kopuru finko bat erabiliz. Δ_n eta iterazio kopuru ezberdinak aztertu ondoren lorturiko konbinaziorik eraginkorrena proposatzen dute, Halley-en eta Newton-Raphson-en metodoak konbinatuz bi iterazio bidez hobetzen dute hasieraketa. Iterazio bakoitzean bi urrats daude, lehenik Halley-ren metodoaren Δ_n (1.10 ekuazioa) kalkulatzen da eta ondoren Newton-Raphson-ena. Halley-ren δ kalkulatu ondoren ez dira berriro $\sin(E)$ eta $\cos(E)$ kalkulatzen (eraginkortasuna hobetzeko), Taylor-en hurbilpen bat erabiltzen da Newton-Raphson-en Δ_n hurbiltzeko. Beraz, funtzio trigonometrikoak soilik bi aldiz kalkulatzen diren arren, 4 urratseko iterazioa izango balitz bezala jokatzen da. Δ_1 1.10 ekuazioan definiturik dago eta Δ_2-ren hurbilpena 1.17 ekuazioan.

Deskribaturiko prozesua programatu ondoren Kepler-en ekuazioaren emaitza zehatzekin alderatu dira hurbilpenak. Errore absolutua erabili da $\epsilon = |E_2 - E|$ Errore absoluturik handiena $2.35 * 10^{-14}$ da. Doitasuna aztertu eta konparatzen den hurrengo kapituluan (e, M) balio diskretuentzat lorturiko erroreak irudikatuta daude 4.3).



2.1 Irudia: Gooding eta Odell-ek proposaturiko hasieraketa balioaren errore absolutua

2.3 Satt

Sakon azterturiko eta konparaketetan erabilitako algoritmo bat Julia lengoaiaren Satellite-Toolbox (*https://juliaspace.github.io/SatelliteToolbox.jl/stable/*) paketean programaturikoa da. Vallado-ren [14] liburuan oinarriturik dago funtzioa. Metodo iteratiboa erabiltzen du Newton-Raphson-en iterazioak erabiliz.

Hasieraketa balioa aukeratzen ez du konplexutasunik, konputazio gehienak iterazioetan burutzen dira. Hasieraketa hautatzean konbergentzia ziurtatzen duen balioa aukeratzen du. M eta e -ren arabera bi aukera erabiltzen ditu.

- $M > \pi$ denean $E_0 = M - e$
- $M < \pi$ denean $E_0 = M + e$

E_0 balioa ez dago beti soluziotik oso gertu baina Newton-en metodoa aplikatuz konbergentzia ziurtatua du. Eraginkortasun aldetik iterazioan eragiketa asko kalkulatzea gerta daiteke. Kasu batzuetan segituan amaituko duen arren, kasurik okerrenean baliteke hainbat iterazio behar izatea. Eremu singularra ($e = 1 eta M = 0$) inguruan, funtzioaren eta deribatuaren balioak txikiak direnez Newton-Raphson-en metodoa motela da konbergitzen.

2.4 Kepler

"Kepler"Julian garatutako algoritmo optimizatuari deritzogu. Ekepl algoritmoaren estruktura berdina jarraitzen du. Bi urratsekin lortzen da emaitza, lehenik hasieraketa balio bat lortzen da ondoren hobetzeko. 3 kapituluan prozesua azaltzen da, bertan ikus daitezke egindako probak. Lehenik hasieraketa balio sinple batekin lorturiko emaitzak, ondoren hasieraketa konplexuago batekin lorturikoak. Behin betiko algoritmoak eragiketa askoren

bidez lorturiko hasieraketa balioa erabiltzen du. Honek ekuazioaren hurbilpen oso ona ematen du eta ondorengo hobetze prozesua sinplea da. Iterazio sinplifikatuaren (1.11 ekuazioa) urrats bakarrean emaitza lortzen da. Bertan, Δ_1 eta Δ_2 Newton-Raphson-en metodoarekin kalkulatzen dira (1.4).

Bektorizatzeko egokia da eta Ekepl funtzioaren estruktura jarraitzen du. Anomalia eszentrikoa E balioa zuzenean lortu ordez, aldagai aldaketa baten bidez lortzen da. Hasieraketa balio bat ekuazio kubiko bat askatuz lortzen da, ondoren balio hau hobetuz. Hobetzeko erabiltzen den Δ_n balioa Newton-Raphson-en metodoaren bi iterazio sinplifikatu bidez lortzen da. Bi urrats hauen bitartez funtzio eraginkor bat lortzea da helburua eta doitasuna biribiltze errorearen tamainakoa izatea. Hurrengo atalean algoritmoa lortzeko erabili den optimizazio prozesua azaltzen da.

KAPITULUA 3

Optimizazioa

3.1 Algoritmoaren antolamendua

Aurreko atalean deskribaturiko Ekepl metodoa bektorizatzeko egokia da. Programatzeko behar dituen baldintza bakarrak eremua murriztekoak dira, atal honek bektorizaziorako arazo bat suposatzen du. M $[0, \pi]$ eremutik kanpo badago, murrizpen bat egin eta amaitutakoan jatorrizko baliora bueltatuz. Ebazpen metodo berria garatzerakoan baldintzarik gabekoa izan behar du M-ren balio guztientzat. Errore maila txikiagoa lortzea da helburu, doitasun bikoitzeko (`Float64`) zenbakiekin eginez gero, biribiltze errorea baino txikiagoa izan dadin. Metodoa hobetzeko bi elementuetan egin daitezke aldaketak:

- E_0 hasierako balioa. Hurbilpena geroz eta hobea izanez, ondorengo prozesu iteratiboa sinpleagoa izango da. Iterazio gutxiago beharko dira, baliteke bakarra edota inongorik behar ez izatea. Bestalde, e eta M-k sortzen duten planoko eremu guztietan hurbilpen ona ematen duen hasieraketa funtzio bat kalkulatzea eraginkortasun aldetik garestia izango da.
- Hasierako hurbilpena hobetzen duen balioa. Metodo egokia aukeratu beharko da eta zenbat aldiz aplikatu behar den definitu beharra dago.

Bi elementuek ez dute modu independentean funtzionatzen. Funtsezkoa da bien interakzioa ona izatea. Hasieraketa balio on batek ez du ziurtatzen ondorengo iterazioetan metodoa errore txikietara helduko denik. Bai hasieraketa funtzioa eta iterazioak bat izanik errore txikia lortu behar dute.

Lehenik eta behin, ekuazioa ebazteko aldagai aldaketa bat erabili da, 3.1 ekuazioa. Honela ez da M-ren murrizketarik beharko. 3.2 ekuazioan a eta b definiturik Kepler-en ekuazioa 3.3 agertzen den bezala definitzen da. x-ren balioa aurkitzea da helburua.

$$E = M + x \tag{3.1}$$

$$a = e\sin(M) \qquad b = e\cos(M) \tag{3.2}$$

$$f(x) = x - a\cos(x) - b\sin(x) \tag{3.3}$$

Aldagai aldaketa honen ondoren soluzioa den x aldagaiak abantaila handi bat du. Periodikoa da eta bere balioa beti 0 eta 1 en artean dago. Hurbilpenak egitea errazten du eta (e, M) balio guztientzat prozesu bera erabiltzea ahalbideratzen du, baldintzak deuseztatuz.

Lehen urratsa ekuazioaren hasierako hurbilpena lortzea izan da. Ekepl metodoan deskribaturiko ekuazio kubikoan oinarrituz egin da. Ekuazio kubiko hori nola eraldatu eta hobetu aztertu da. Horretarako Ansatz bat planteatu da. Ansatz bat problema fisiko edo matematiko bat deskribatzen duen hasierako ekuazio baten soluzioa da. Ansatz bat ezarri ondoren, ekuazioaren muga-baldintzak kontuan hartuta ebazten dira. Antsatz-ak emaitzaren kalitatea ebaluatuz eguneratu daitezkeen parametroak ditu. Ekuazio kubikoa λ parametro batzuen menpe planteatze izan da Ansatz-a. Parametro jakin batzuk erabiliz emaitza bat lortuko da, hau ebaluatuz parametro horiek ahalik eta emaitza onena emateko optimizatuko dira.

Ekuazio kubikoa aurkitzeko Ansatz-a erabiliko da. Gooding eta Odell-ek aztertutako hasieraketa funtzio ezberdinetan oinarriturik ekuazio kubikoa (3.4 ekuazioa) definitzen duten p, q, r parametroak λ parametroen menpe definitzen dira., non e, M eta

$$(x_0 - p)^3 + 3q(x_0 - p) - 2r = 0 \tag{3.4}$$

Non $p(e, M, \lambda)$, $q(e, M, \lambda)$, $r(e, M, \lambda)$ e, M eta λ -ren funtzio diren.

Optimizazio prozesuan zehar λ balio ezberdinekin eraikiko dira ekuazio kubikoak. Baliteke kasu batzuetan soluzio bakarra ez duten ekuazioak sortzea. Kasu hauek baztertu beharrekoak dira, ez baitute Kepler-en ekuazioa ebazteko balio. Programatzerako garaian ekuazio kubiko deprimituaren diskriminantea erabili da, hain zuzen ere $\kappa = r^2 + q^3$. Positiboa denean ekuazioak soluzio bakarra du, beraz optimizazio prozesuak normal jarrai dezake. Negatiboa bada ordea, prozesuak ezin du jarraitu, beraz helburu funtzioari balio altu bat jarri zaio (10^6). Horrela optimizazio prozesua λ minimoen bila dabilenean, soluzioa ez duten kubikoak sortzen hasten bada, helburu funtzioa haztean ez du ontzat ematen.

Hiru Ansatz ezberdin planteatu dira hasierako hurbilpena lortzeko. Ondoren, algoritmoaren bigarren urratsa aztertu da, hau da, hurbilpena hobetzen duen prozesua. Horretarako metodo iteratiboen Δ_n gehitzen zaio x_0 ri hainbatetan. Δ_n -ren balioa eta zenbatetan aplikatu behar den optimizatu da. Newton-Raphson eta Halley-ren metodoaren urrats kopuru ezberdinak probatu dira. Gehienez bi iterazioko limitea ezarriz, funtzio trigonometrikoen ebaluazioa murrizteko. Gooding eta Odell-en Ekepl metodoan bi iterazio bidez lortzen dute emaitza, beraz hasieraketa hobetzea lortu ezkero, eragiketa gutxiagorekin lortzea espero da. Azkenik funtzio trigonometrikoak behin bakarrik kalkulatzea erabaki da, beraz iterazio urrats bakarra emanez. Hala ere, 1.11 ekuazioan definitzen den bezala bigarren iterazio bat hurbilduz egin da. Bai Δ_1 eta Δ_2 Newton-Raphson-en metodoarenak erabili dira, lehena zehazki kalkulaturik eta bigarrena 1.17 ekuazioan definituriko hurbilpenarekin. Aldagai auxiliarrak erabiliz programatu da urratsa (3.10 ekuazioan definitua), behin $\tilde{x}$ kalkulaturik 3.1 ekuazioarekin aldagai aldaketa burutuz $\tilde{E}$ lortzen da.

$$f(x) = x - a\cos(x) - b\sin(x) \tag{3.5}$$
$$f'(x) = 1 - b\cos(x) + a\sin(x) \tag{3.6}$$
$$f''(x) = b\sin(x) + a\cos(x) \tag{3.7}$$
$$f'''(x) = b\cos(x) - a\sin(x) \tag{3.8}$$

$$\delta = \frac{f(x_0)}{f'(x_0)} \qquad \eta = \frac{f''(x_0)}{f'''(x_0)} \qquad \nu = \frac{f'(x_0)}{f'(x_0)} \tag{3.9}$$

$$\tilde{x} = x_0(\lambda) + \delta\left(1 - \frac{(\frac{\eta\delta}{2} + \frac{\eta\delta^2}{6}}{(1+\eta\delta)}\right) \tag{3.10}$$

3.2 Ebaluazio sarea

Gooding eta Odell-en Ekepl metodoan oinarriturik, ebazpen metodoa bi urratsean planteatua dago. Lehenik hasieraketa balioa eta jarraian hurbilpen honen hobekuntza. Hasieraketa balioaren kalkulua parametrizaturik dago λ balioen menpe. Ebazpena ahalik eta ondoren burutzen duten parametroen bilaketa optimizazio problema bat da. Horretarako λ bakoitzarekin lorturiko emaitzak ebaluatu beharrean daude, hau da, $h(\lambda)$ helburu funtzio bat definitu beharra dago.

λ jakin batzuek finkatutakoan, hasieraketa balioa e eta M askotarako ebatzi daiteke. Balio guztietarako emaitza onak lortu nahi dira, beraz helburu funtzioak (e, M) balio askotarako kalkulatu beharko du emaitza. Lehenik helburu funtzioak kalkulatu behar duen eremua definitu da. Hain zuzen ere $e \in (0, 1),\ M \in [-\pi, \pi]$ osatzen duten plano jarraia da eremua. Optimizazio prozesuan erabiltzeko, planoa diskretizatu egin da, e aldagaia m zatitan banatuz eta M n zatitan. Lehenengo optimizazio probak egiterakoan sareak emaitzarengan eragin handia duela ikusi da. Sare txiki bat eginez (m eta n 20 baino txikiagoak) balio gutxiegi aztertzen dira, beraz, baliteke errore handiko inguruak ez sartzea optimizazio prozesuan. m eta n handiak erabili ezkero, Kepler-en ekuazioa $m \times n$ aldiz ebaluatu behar da, optimizazio denborak asko luzatuz. Sare optimoa elementu tamaina ertaina ($30 < m, n < 150$) duena, baina inguru arazotsu gehienak elementu askoz betetzen dituenak. Horretarako sare ez uniformeekin egin dira probak, elementu gehiago kontzentratuz ($e = 1$) inguruan. Hiru sare mota erabili dira optimizazio probleman, bakoitza aurrekoaren emaitzak hobetzeko garatua.

3.2.1 Sare uniformea

Lehenengo Ansatz-ean probak egiten hasterakoan, geometrikoki aukerarik errazena hartu eta sare uniforme batetan banandu zen $e,\ M$ planoa.

$$e_i = \frac{1}{m-1}(i-1) \qquad M_j = \frac{\pi}{n-1}(j-1) \tag{3.11}$$

$$i = 1, 2, ..., m \qquad j = 1, 2, ..., n$$

3.2.2 Sare aldakorra

Optimizazio prozesuaren lehenengo saiakeretan errorerik handienak eremu singularraren ($e = 1, M = 0$) inguruan topatu dira. Helburu funtzioaren emaitza, sareko puntu guztien errorearen gehiketa (hurrengo atalean sakonki azaldua) da. Beraz, izkina batean errore handia edukitzea ez du eragin askorik. Optimizazioaren emaitzek ez dute eremu horretako errore handia txikitzen. Helburu funtzioaren balioa minimoa izan arren, lorturiko ebazpen metodoak errore handiak ditu eremu singularrean. Argitaratu diren ikerketa askotan, errore konprobaketak $e = 1$ denean soilik kalkulatzen direla ikusi da. Hau ikusirik oinarriturik sarea erabiltzen jarraitzea erabaki da, baina errore asko dagoen inguruan ebaluazio puntu gehiago kontzentratuz. M -ren diskretizazioak lineala izaten jarraituz, e -rena eraldatu egin da dentsitate handiagoa izan dezan $e = 1$ inguruan. 3.12 ekuazioak aurreko $_{ij}$ -ren definizioa ordezkatzen du. Adibide bezala 3.1 irudian sarea ikusten da m eta n balio txikientzat (optimizazioan erabili direnak dentsitate handia dute irudikatzeko).

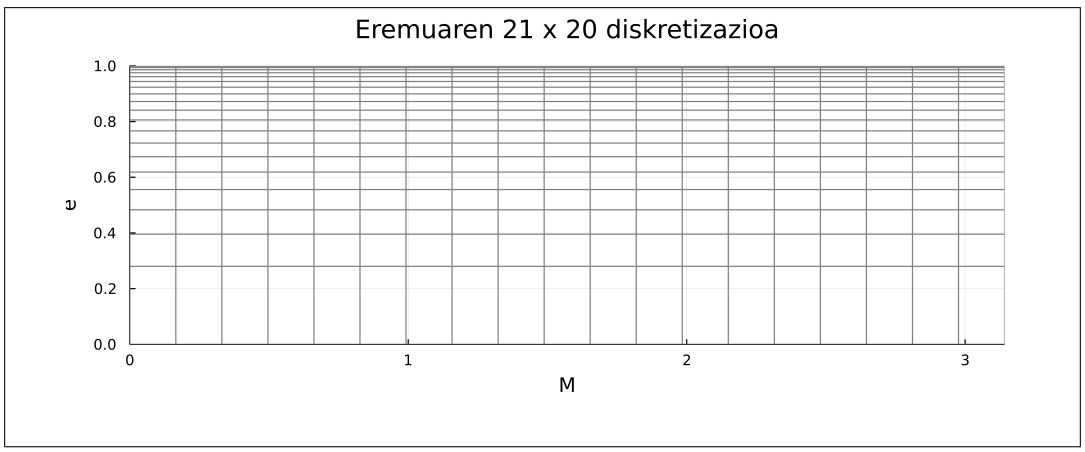
$$e_i = \sqrt{\sin\left(\frac{\pi(i-1)\pi}{2m-2}\right)} \tag{3.12}$$

3.2.3 Anomalia eszentrikoaren sarea

Ansatz-aren geroz eta optimizazio prozesuko emaitza onak lortzean erroreak asko txikitu dira. Bertan biribiltze errorearen eragina ikusi da. Hau murriztu nahian sare mota berri bat garatu da. e eta M zatitu ordez , e eta E -ren balioak erabili dira. Aurreko sareetan bezala e aldakorra erabili da ($e = 1$ inguruan konzentratuz) eta E uniformeki zaturu da (3.13 ekuazioa). M -ren balioak Kepler-en ekuaziotik askatu dira 3.14 ekuazioan definiturik.

$$e_i = \sqrt{\sin\left(\frac{\pi(i-1)\pi}{2m-2}\right)} \qquad E_j = \frac{\pi}{n-1}(j-1) \tag{3.13}$$

$$M_{ij} = E_j - e_i \sin(E_j) = \frac{\pi}{m-1}(j-1) - e_i \sin(\frac{\pi}{m-1}(j-1)) \tag{3.14}$$



3.1 Irudia: Eremuaren diskretizazioaren adibidea, $e = 1$ inguruan balio konzentrazoa nabari da.

$$i = 1, 2, ..., m \qquad j = 1, 2, ..., n$$

Sarea honela definituz M -ren balioak ez dira errepikatzen baina ondorengo bi abantailak ditu.

- E -ren balioa jada kalkulaturik dago. Aurreko bi sareetan (e, M) definitu ondoren anomalia eszentrikoaren balio zehatza kalkulatzen zen. Bertan biribiltze errorearen eragina nabaria zen balio optimizatu onetan. Errazago eta eraginkorragoa da optimizazio prozesuan E -tik M kalkulatzea alderantziz baina.
- Kepler-en ekuazioan $e = 1$ inguruan E eta M -ren arteko erlazioak (1.2 eta 1.3 irudiak) eraginda sarean puntuak $e = 1$ eta $M = 0$ inguruan kontzentratzen ditu. Eremu singularrean datu gehiago edukiz, helburu funtzioaren balioan eragin handiagoa edukitzea lortzen da. Honela optimizazio prozesuak, izkinako errorea gutxitzen dituen λ balioak hobesten ditu.

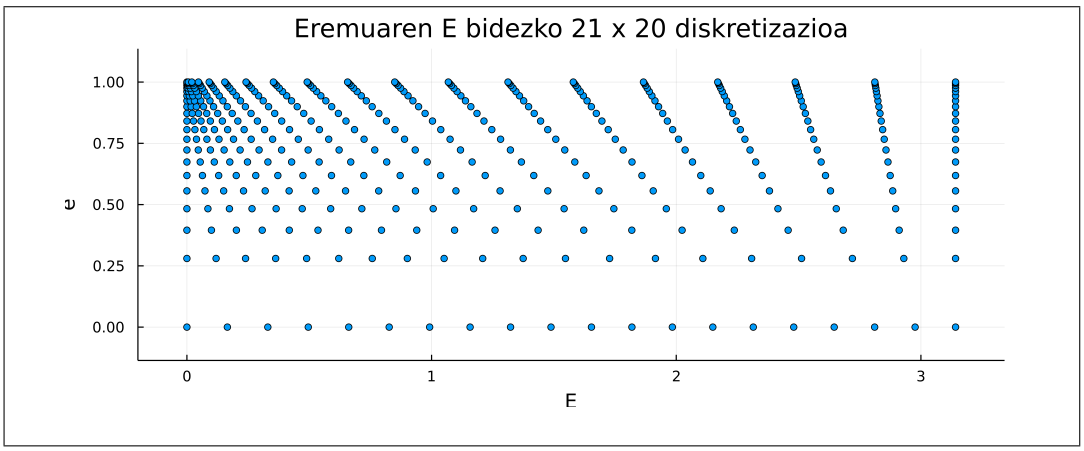
3.3 Helburu funtzioa

Optimizazio prozesua burutzeko λ parametroen menpe dagoen $h(\lambda)$ helburu funtzioa definitu da. Funtzio honek lorturiko balioak "ebaluatu"egite ditu, zeinen onak edo txarrak diren. Errorearekin erlazionaturik dagoen funtzioa aukeratu da, beraz $h(\lambda)$ balio altuek errore handiak dituzte. Helburu funtzioa kalkulatzeko bi errore erabili dira.

- Lehenengoa hasieraketa balioaren errore absolutua da (3.15 ekuazioa). Emaitza zehatza eta λ -kin kalkulatutako hasieraketako balioaren arteko diferentzia da.

$$\epsilon_{ij} = |E_0(\lambda, e_i, M_{ij}) - E_j| \tag{3.15}$$

- Bigarrena Kepler-en ekuazioa ebazten duen funtzio osoaren $\tilde{E}$ balioa eta ebazpen zehatzaren E balioaren arteko diferentzia da. $\tilde{E}$ kalkulatzeko hasieraketa balioaren



3.2 Irudia: Eremuaren E eta e bidez eginiko diskretizazioa, $e = 1$, $M = 0$ inguruan balio konzentrazoa lortuz.

ondoren Newton-Raphsonen iterazio sinplifikatu bat kalkulatzen da hobetzeko $\tilde{E} = E_0(\lambda) + \Delta_1 + \tilde{\Delta}_2$ 1.17 ekuazioan definitu bezala.

$$\epsilon_{ij} = |\tilde{E}_{ij}(\lambda, e_i, M_{ij}) - E_j| \tag{3.16}$$

Optimizazioa helburu funtzioaren balioa minimizatzen dituen parametroak aurkitzea da. Aurreko atalean sareak aipatzean azaldu den bezala, parametro balio bat probatzeko hasieraketa balioa sareko elementu guztietan probatu behar da. Sareko elementu guztien emaitzak balio bakar batean konbinatzeko hurrengo aukerak erabili dira.

- **Sarearen maximoa.** Sare guztian E_0 hurbilpen ona ematen duen λ balioak aurkitu nahi dira. Sarearen %99-an oso emaitza onak dituzten parametroak eta puntu batean txarra direnak ez dira egokiak. Beraz, zentzuzkoa da sare guztian ematen den errore maximoa minimizatzea.

  $$h(\lambda) = \max(\epsilon_{ij}^2(\lambda)) \tag{3.17}$$

  Optimizazio probak egiterakoan ez da funtzioaren minimo egonkorrik aurkitu. Emaitza txarrak edota oso aldakorrak topatu dira. Balio guztien maximoa aukeratzean helburu funtzioak leuntasuna galtzen du, optimizaziorako zaila bihurtuz.

- **Sarearen gehiketa.** Sarea ebaluatzean lorturiko errore guztiak gehituz lortzen da helburu funtzioaren balioa. Kasu honetan ez da leuntasuna galtzen eta optimizazio prozesuan emaitza hobeak lortzen dira.

  $$h(\lambda) = \sum_{i=1}^{m} \sum_{j=1}^{n} \epsilon_{ij}^2(\lambda) \tag{3.18}$$

  Hala ere, lorturiko emaitzetan eremu txikietan errore handiak aurkitu dira. Helburu funtzioaren minimoak sarearen atalik handiena errore txikiarekin du baina inguru batzuetan errore handiekin. Arazo hau ekiditeko inguru horietan sare finagoa erabili da, aurreko atalean azaldu bezala.

Optimizazio problema planteaturik dagoela, Julia lengoaia erabili da ahalik eta parametrorik onenak aurkitzeko. Aldagai bakarra edo multzoak optimizatzeko programaturik dagoen `Optim` paketea erabili da. Paketea erabiltzeko helburu funtzio bat programatu beharra dago eta bertako minimizatze algoritmoa aukeratu. Kasu batzuetan deribatua erabiltzen bada baita gehitu daiteke optimizazio parametro bezala. BFGS optimizazio algoritmoa erabili da. *Broyden, Fletcher, Goldfarb, eta Shanno* izenekoa, tokiko bilaketak burutzen dituen algoritmoa da. Helburu funtzioa eta bigarren mailako deribatua erabiltzen ditu minimoa aurkitzeko. Bigarren deribatua esplizituki definitzen ez den kasuetan *Quasi-Newton* deritzon metodoa erabiltzen du kalkulatzeko. Bigarren deribatua Hessiano matrizea da, bigarren deribatu partzialak dituen matrize simetrikoa. Honi esker optimizatzeko prozesu iteratiboan hurrengo urratsaren norabide eta tamaina kalkulatzen dira. Norabide honetan λ parametroak aldatzen dira urrats berrian helburu funtzioa eta Hessianoa kalkulatzeko.

Optimizazio prozesuan hiru Ansatz ezberdin erabili dira, sinpleagotik konplexuagora joanez. Bakoitzean sare mota eta tamaina ezberdinak erabili dira azken emaitzakoa lortu arte. Ondoren Ansatz bakoitza azaltzen da bere optimizazio prozesua eta emaitza adieraziz.

3.4 Ansatz 1

Lehenengo Ansatz-a Gooding eta Odell -en ikerketan [13] azterturiko hasieraketa balioetan oinarriturik dago. Ekuazio kubikoa definitzen duten parametroen(r, q eta p) hainbat balio aztertzen dituzte. Ansatz-a balio hauek era parametrikoan zabaltzen dituen formula bezala planteatu da.

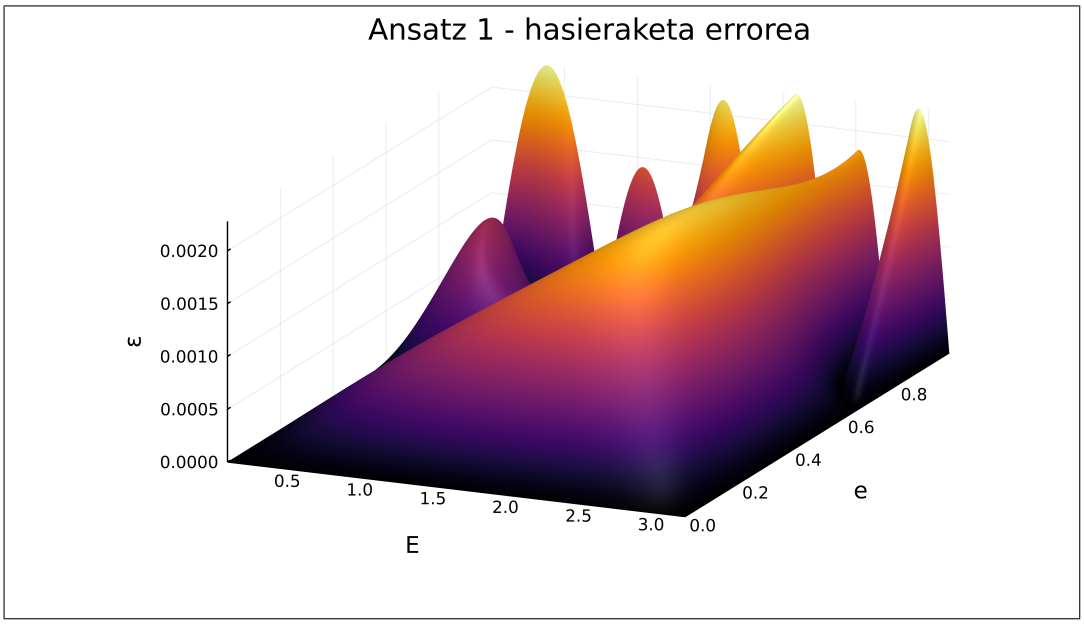
$$r = \frac{\lambda_0 + \lambda_1 b}{1 + \lambda_2 b} a \tag{3.19}$$

$$q = \frac{(\lambda_3 + \lambda_4 b)(1 - b) + (\lambda_5 + \lambda_6 b)a^2}{1 + \lambda_2 b} \tag{3.20}$$

$$p = \frac{\lambda_7 + \lambda_8 b}{1 + \lambda_2 b} a \tag{3.21}$$

Optimizatzeko garaian 151×150 elementuko sare aldakorra erabili da. Prozesua gune jakin batean hasten da, λ balio batzuekin. Ondoren algoritmoa doa pixkanaka minimoaren bila. Hasierako λ balio onak ematea ezinbestekoa da minimizazio prozesuarentzat. Goodin eta Odell-ek proposaturiko ekuazio kubikoa sortzen duten λ parametroak erabili dira. Hau da, 3.22 ekuazioan hasieraketa balio on baten q, r, p parametroak daude. Emaitza hau ematen duten λ parametroak erabili dira optimizazioaren abiapuntu bezala.

Helburu funtzioa kalkulatzeko sare guztiaren erroreen gehiketa erabili da, 3.18 ekuazioa. Hasieraketa baliotik emaitza zehatzera errorea zenbatekoa den neurtu da, 3.15 ekuazioan definiturikoa. Optimizazioan hasieraketa balio onena lortu ondoren emaitzak aztertu dira. Kepler-en ekuazioan errore txikia (10^{-16} ingurukoa) lortzeko sare osoan iterazio asko behar direla ikusi da. Bi Newton-Rapshonenak baino gehiago, funtzio trigonometrikoak askotan ebaluatuz. Beraz, Ansatz berri bat planteatzea eta helburu funtzioa Kepler-en emaitzarekin konparatzea erabaki da.



3.3 Irudia: Lehenengo Ansatz-aren E_0 hasieraketaren errorea.

$$p = 1 \qquad r = \frac{3a}{3-2b} \qquad q = \frac{2(1-b)}{3-2b} \tag{3.22}$$

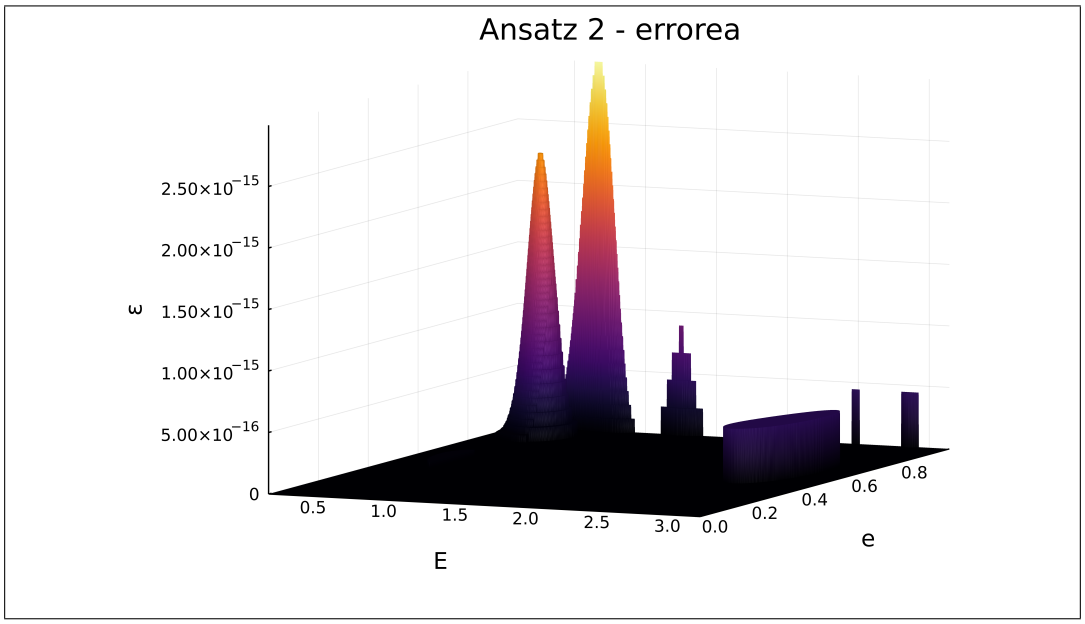
3.5 Ansatz 2

Bigarren Ansatz-ean λ parametroen kopurua handitu egin da hasieraketak konplexutasun gehiago eduki dezan. 15 balio ezberdin optimizatu dira. Zatiketa egin ordez, a eta b -rekin egindako polinomioekin osatu dira parametroak.

$$r = (\lambda_0 + \lambda_1 b + \lambda_2 b^2)a + (\lambda_3 + \lambda_4 b)a^3 \tag{3.23}$$
$$q = (\lambda_5 + \lambda_6 b + \lambda_7 b^2) + (\lambda_8 + \lambda_9 b)a^2 \tag{3.24}$$
$$p = (\lambda_{10} + \lambda_{11} b + \lambda_{12} b^2)a + (\lambda_{13} + \lambda_{14} b)a^3 \tag{3.25}$$

Kasu honetan 81×80 -eko sare aldakorra erabili da, $e = 1$ inguruan balio gehiago ebatziz. Sareko puntu guztietan hasieraketa balioa kalkulatu da, ondoren Newton-en iterazio sinplifikatu bat eginez hurbilpena hobetzeko, 3.10 ekuazioa. Errorea kalkulatzeko E -ren balio zehatzarekin konparatu dira 3.16 ekuazioan definitu bezala. Helburu funtzioa errore guztien baturarekin lortu da, 3.18 ekuazioan adierazia. Optimizazioan lorturiko λ balioek Kepler-en ekuazioa ebazteko garaian 3×10^{-15} -eko errorea du. Emaitza ona izanik Ansatz-a zabaltzea eta optimizazio metodoa erabaki da.

Bigarren Ansatz-arekin lehen optimizazio probak egiterakoan errorerik handienak eremu singularrean topatu dira. Lorturiko λ parametroen balioek ez dute errorea txikitzen izkina inguruan. Nahiz eta sare tamaina eta mota aldatu ez da erabat murrizterik lortu. Optimizazio bidez lortu ezin denez, hasieratik baldintza batzuk finkatu beharrean daude emaitza onak lortzeko. Beraz, hasieraketa balioaren definizioa aztertu da. Ondorengo



3.4 Irudia: Bigarrengo Ansatz-aren errorea.

formuletan azaltzen dena azterturik hasierako hurbilpena izkinan errore txikikoa izan dadin lortu da. λ balioetan eman behar diren hiru baldintza lortu dira, beraz parametro kopurua 15 etik 12-ra jaitsi da.

Lehenik, hasieraketa balioa x_0 definitzen duen ekuazio kubikoa adierazten da. Aldagai aldaketan θ erabiliz baita M eta E, e -ren arteko erlazioekin 3.28 ekuazioa lortzen da.

$$g(e, E, \lambda) = (x - p(e, M, \lambda))^3 + 3q(x - p(e, M, \lambda)) - 2r(e, M, \lambda) \tag{3.26}$$

$$M = E - e \sin E, \qquad x = E - M = e \sin E \tag{3.27}$$

$$g(1 - \theta, E, \lambda) = (x - p(\theta, M, \lambda))^3 + 3q(x - p(\theta, M, \lambda)) - 2r(\theta, M, \lambda) \tag{3.28}$$

Taylor-en garapena eginez θ eta E -rekiko hurrengo ekuazioa lortzen da. Lehenengo terminoak soilik idazten dira, garrantzirik handiena dutenak emaitzan. g_{mn} adieraztean lehenengo azpindizeak deribatuaren maila adierazten du, bigarrenak ordea aldagaia. g_{212} deribatu partziala da, lehenengo eta bigarren aldagaiarekiko.

Inguruan bertan ekuazioaren balioa 0 denez, termino bakoitza 0 izatera behartzen da, honela λ balioen baldintzak lortuz.

$$g(1 - \theta, E, \lambda) = g_0(\lambda)E + g_{32}(\lambda)E + g_{32}(\lambda)E^3 + g_{212}(\lambda)E\,\theta + \ldots \tag{3.29}$$

Ansatz-ak dagoeneko g_0 terminoa zero du, edozein λ baliorentzako. Baldintza hau jada beterik dago. Gainontzekoak deribatuetan zabalduz eta 0 izatera behartuz.

$$g_{12}(\lambda) = \frac{\partial g}{\partial E}(1, E, \lambda)|_{E=0} = 0 \tag{3.30}$$

$$g_{32}(\lambda) = \frac{\partial^3 g}{\partial E^3}(1, E, \lambda)|_{E=0} = 0 \tag{3.31}$$

$$g_{212}(\lambda) = \frac{\partial^2 g}{\partial E\,\partial e}(e, E, \lambda)|_{(e,E)=(1,0)} = 0 \tag{3.32}$$

3.30 ekuaziotik lortzen diren baldintza hurrengoak dira:

$$\lambda_0 + \lambda_1 + \lambda_2 = 0 \tag{3.33}$$

$$\lambda_5 + \lambda_6 + \lambda_7 = 0 \tag{3.34}$$

$$\lambda_6 + 2(1 + \lambda_7) = 0 \tag{3.35}$$

3.6 Ansatz 3

Hirugarren Ansatz-ak aurrekoaren estruktura berdina du, parametro batzuk gehituz Keplerren ekuazioaren konplexutasuna hobeto adierazi dezan.

$$r = (\lambda_0 + \lambda_1 b + \lambda_2 b^2 + \lambda_3 b^3)a + (\lambda_4 + \lambda_5 b + \lambda_6 b^2)a^3 \tag{3.36}$$
$$q = (\lambda_7 + \lambda_8 b + \lambda_9 b^2 + \lambda_{10} b^3) + (\lambda_{11} + \lambda_{12} b + \lambda_{13} b^2)a^2 \tag{3.37}$$
$$p = (\lambda_{14} + \lambda_{15} b + \lambda_{16} b^2 + \lambda_{17} b^3)a + (\lambda_{18} + \lambda_{19} b + \lambda_{20} b^2)a^3 \tag{3.38}$$

Lehenik sarea E eta e banatuz lortu da, anomalia eszentrikoaren sarea deiturikoa aurreko atalean. Puntu gehiago kontzentratzen dira $e = 1, \quad M = 0$ izkinan. Kasu honetan optimizaziorako sare handiagoa erabili da 151×150 -eko sarea. Bigarren Ansatzean erabilitako baldintza berdinak ezarri dira λ balioetan. Honek izkinan errorea txikia edukitzea ziurtatzen du. 21 parametro erabili ordez 18 erabili dira, 3 baldintza betez. 3.30 ekuaziotik lortzen diren λ balioen arteko erlazio berriak hurrengoak dira.

$$\lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 = 0 \tag{3.39}$$
$$\lambda_7 + \lambda_8 + \lambda_9 + \lambda_1 0 = 0 \tag{3.40}$$
$$\lambda_8 + 3\lambda_1 0 + 2\lambda_9 - 2 = 0 \tag{3.41}$$

Helburu funtziorako hasieraketa balioa eta Newton-Raphson-en iterazio sinplifikatuaz hurbilpena hobetu da. Balio hau emaitza zehatzarekin konparatuz lortu da errorea. Helburu funtzioa errore guztien karratuaren batura izan da. Errorea 3.16 ekuazioan eta helburu funtzioa 3.18 ekuaziokoak direlarik. Optimizaturiko λ parametroekin lorturiko errorerik handiena 9.4×10^{-19} da. Erroreen banaketa sare guztian 4 kapituluan azaltzen dira, 4.1 eta 4.2 irudietan.

3.7 Biribiltze errorea

Oso antzekoak diren zenbakien arteko kenketa egiterakoan biribiltze errorea ematen da, informazioa galduz. Eragiketa hauek identifikatu eta eraldatu egin behar dira prozesuan errorerik sar ez ditzaten . Eremu arazotsuan $e \approx 1$ eta $M \approx 0$ denean $f(x)$ funtzioa eta bere deribatuak kalkulatzean zenbaki txikien arteko kenketak ematen dira. Honako aldaketak egin dira ekuazioetan errorea murrizteko.

$$f(x) = (1 - b)x - a + a(1 - \cos(x)) + b(x - \sin(x)) \tag{3.42}$$
$$f'(x) = (1 - b) + b(1 - \cos(x)) + ax - a(x - \sin(x)) \tag{3.43}$$

Gune arazotsuak honako hauek dira:

- $1 - b = 1 - e\cos(M)$. $M \approx 0$ denean b batetik oso gertu dago kenketan doitasuna galduz. Angelu erdien identitate trigonometrikoa erabiliz eraldatu daiteke.

$$\cos(M) = 2(\cos\left(\frac{M}{2}\right)^2 - 1 \tag{3.44}$$

Kenketa honela kalkulatzen da.

$$1 - b = 1 - e + e\left(2\sin\left(\frac{M}{2}\right)\right) \tag{3.45}$$

- $x - \sin(x)$. Biak zerotik gertu daudenean antzekoak dira. Taylor-en 18 mailako polinomio baten bidez hurbildu dira $[-1, 1]$ eremuan errorea biribiltze errorea baino txikiagoa izan dadin.
- $1-\cos(x)$. Biak 1 etik oso gertu daude $x \approx 0$ denean. Taylor-en 18 mailako polinomio baten bidez hurbildu dira $[-1, 1]$ eremuan errorea biribiltze errorea baino txikiagoa izan dadin.

Taylor-en polinomioak alde batetik kenketa eragiketak hurbiltzen ditu (biribiltze errorea ekidinez), bestetik bi funtzio trigonometrikoen balioa lortzea (funtzio deialdia egin gabe) ahalbidetzen du. Beraz $cos(x) = 1 - (1 - cos(x)$ bezala eta $sin(x) = x - (x - sin(x))$ bezala kalkulatu dira, parentesi artekoa hurbilpena izanik.

Taylor-en 18 mailako polinomioa modu eraginkorrean ebaluatzeko Horner-en algoritmoa erabili da. Bertan polinomioaren balioa puntu jakin batean kalkulatzeko eragiketa kopurua optimizatu egiten da. Gainera egin beharreko eragiketak gehiketa eta biderketa bat errepikatzean datza. Programatzeko garaian bi eragiketa ezberdin egin ordez FMA erabili da. *Fused multiply-add* operazioak dira, ziklo bakarrean biderketa bat eta honen emaitzari balio bat gehitzen zaio. Hardware-an inplementaturiko aukera bat da, gaur egun oso ohikoa izanik Intel eta AMD prozesagailuetan. FMA bidez egindako eragiketak errore txikiagoa dute. Funtzioa bektorialki erabiltzeko egokia da.

$p(x)$ polinomioa definiturik.

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n \tag{3.46}$$

Hurrengo moduan berridatzi daiteke, x faktore komun ateraz aldiro. Eskuinetik ezkerrerantz ebaluatzen hasi ezkero biderketa eta gehiketa eragiketak errepikatuz lortzen da balioa.

$$p(x) = a_0 + x\Big(a_1 + x\Big(a_2 + x\big(a_3 + ... + x(a_{n-1} + xa_n)...\big)\Big)\Big) \tag{3.47}$$

KAPITULUA 4

Emaitzak

Kepler-en ekuazioaren emaitza lortzeko aztertu eta programatu diren hiru algoritmoak ebaluatzeko hainbat irizpide erabili dira. Julia lengoaian eginiko proben emaitza eta ondorioak azaltzen dira.

4.1 Doitasuna

Kepler-en ekuazioa ebaztean lorturiko anomalia eszentrikoaren balioa ($\tilde{E}$) emaitza zehatzetik (E) zein hurbil dagoen neurtu da. Honetarako errore funtzioa ezarri da, bi balioak kontutan hartuz doitasun balio bat ematen duena (ϵ). Emaitzak errore funtzioaren menpe daude eta ezberdinak izan daitezke. Horregatik ebaluatzeko garaian garrantzitsua da helburu bakoitzerako funtzio egokia aukeratzea.

Azterturiko ikerketa aunitzetan errore erlatiboa erabiltzen da (4.1 ekuazioa). Nahiz eta hasieraketa balioak konprobatzean eta programazio prozesuan erabili den, errore absolutua izan da doitasunak konparatzeko funtzioa (4.2 ekuazioa). Errore honen bidez edozein e eta M balio pareentzat emaitza zein hurbil dagoen ikustea lortzen da. Errore erlatiboan ordea balioaren menpe dago, E -ren balio handiagoentzat errore handiagoa onartzen da txikientzat baino, 0 -tik gertu dauden balioetan asko handituz.

$$\epsilon_r = \frac{|\tilde{E} - E|}{E} \tag{4.1}$$

$$\epsilon = |\tilde{E} - E| \tag{4.2}$$

Kepler-en ekuazioa ebazten duten hiru metodoek bi parametro dituztenez (e, M) hauen balio askotarako egin dira probak. $e[0, 1)$ eta $M[0, \pi]$ eremu barruko balioekin. (e, M) bikoteekin sare bat osatu da 2001×2000 elementukoa. e -ren balioak uniformeki sakabanaturik daude 0 -tik $1 - 1 \times 10^{-15}$-era. M -ren balioak lortzeko ordea E -ren balioak uniformeki ezarri dira 1×10^{-15} -tik π -ra. Ondoren M kalkulatu da 1.1 ekuaziotik zuzenean. M -ren eremua honela definitzeak bi abantaila dauzka: errorearen kalkulurako emaitza zehatza ezaguna da, eta $e = 1$ inguruan balio gehiago kontzentratzen dira $M = 0$ inguruan (hau

eremu singularra izanik puntu gehiagotan ebaluatzea komeni da). Sare guztiko elementu bikote guztientzat (40.002 miloi) errorea kalkulatu da algoritmo bakoitzarekin. Lorturiko errore absolutu maximoak 4.1 taulan daude. Ondoren algoritmo bakoitzaren emaitzak azaltzen dira sarearen erroreak erakutsiz.

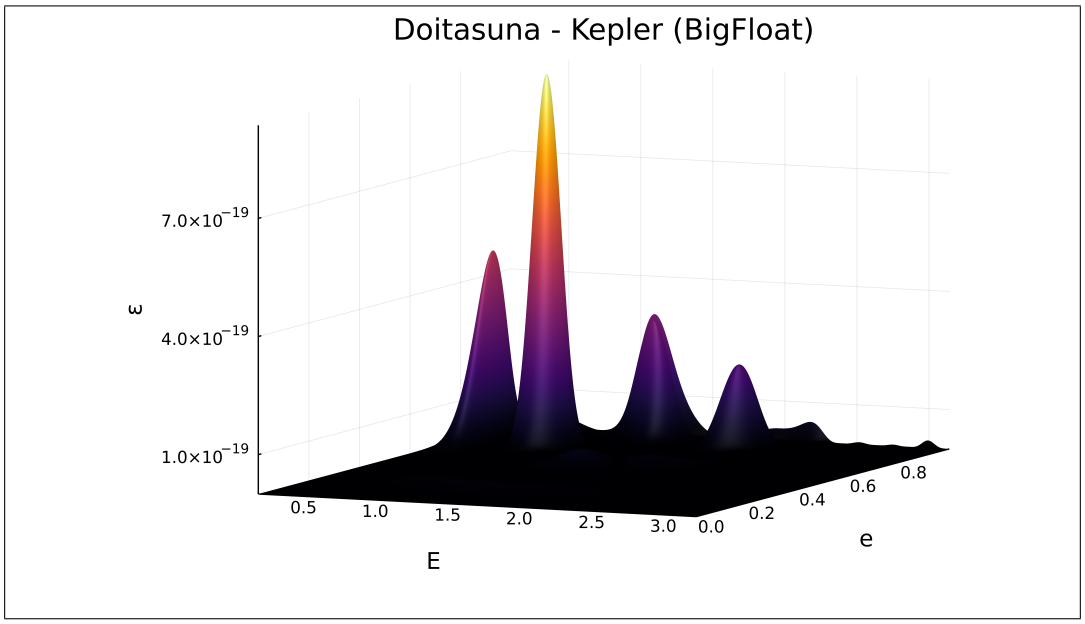
Algoritmoa	Float64	BigFloat
Kepler	4.44×10^{-16}	9.36×10^{-19}
Ekepl	2.35×10^{-14}	2.35×10^{-14}
Satt	2.67×10^{-14}	8.78×10^{-18}

4.1 Taula: Algoritmo bakoitzaren errore maximoa datu motaren arabera

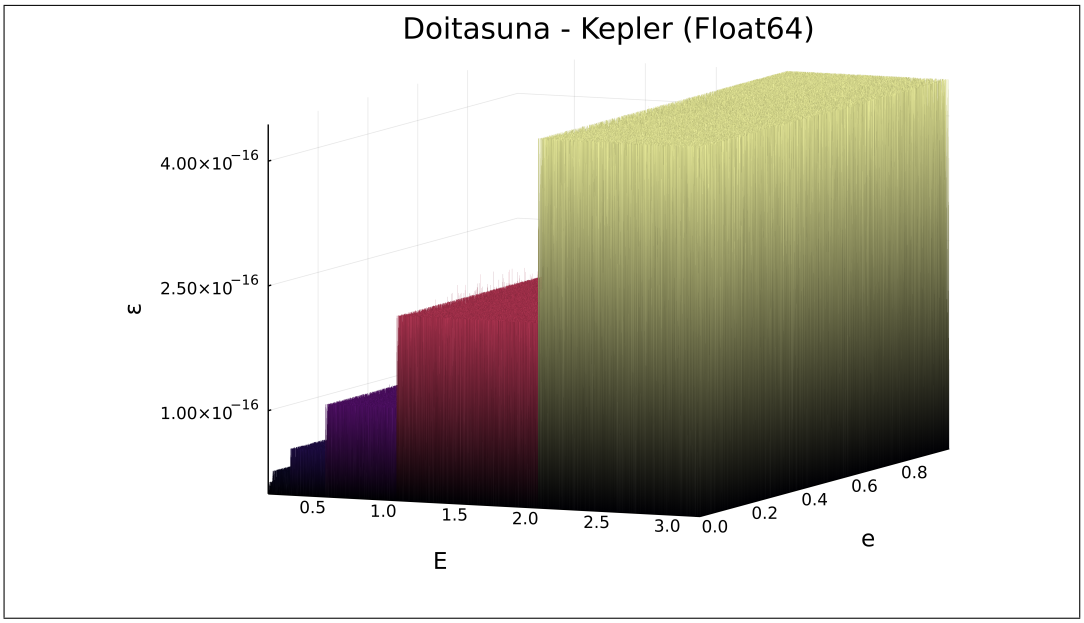
Algoritmo bakoitzean doitasuna ondorioztatzeko Julia lengoaiako probak zenbakiak bi era desberdinetan irudikatuz egin dira. Alde batetik doitasun bikoitzeko zenbakiekin, hau da, 64 bit eko *floating point number 64*, Julian `Float64` datu mota. Bestetik doitasun handiagoko zenbaki dezimalen errepresentazioarekin, Julian `BigFloat`. Doitasun handiko zenbakiak erabiltzean formulazio matematikoaren emaitza aztertzen da. Zein den diseinaturiko algoritmoaren doitasuna konputagailuaren zenbaki errepresentazioa kontutan hartu gabe. Kalkuluak ez dira eraginkorrak baina ez dago biribiltze errorerik. Doitasun bikoitzeko zenbakiekin ordea, algoritmoaren inplementazioak lortu dezaken limitea aztertzen dugu. Zenbakiak 64 bit-etan gordetzen direnez eragiketa batzuetan biribiltze errorea eman daiteke. Programatzerako garaian hau kontutan izatea ezinbestekoa da errorea saihesteko.

4.1.1 Kepler

`kepler()` funtzioan programaturiko algoritmo optimizatuak 10^{-18} inguruko errore absolutua du doitasun handiko zenbakiz kalkulaturik. Algoritmoa garatzerakoan ezarritako helburu bat doitasun bikoitzeko zenbakien biribiltze errorea baino errore txikiagoa lortzea da eta betetzen du. Azterturiko ebazpen metodo askoetan eremu singularrean ($e = 1$ eta



4.1 Irudia: Kepler algoritmoaren errore absolutua doitasun handiko zenbakiz kalkulatua.



4.2 Irudia: Kepler algoritmoaren errore absolutua doitasun bikoitzeko zenbakiz kalkulatua.

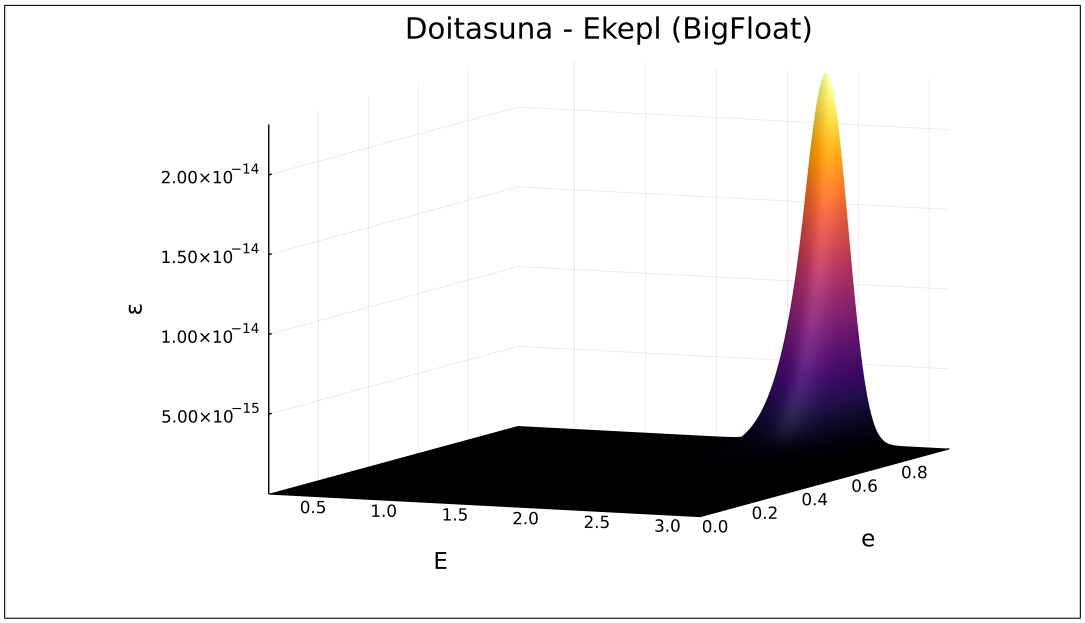
$M = 0$ inguruan) arazoak dituzte, errorea asko handituz. Kasu honetan ez da inongo puntarik nabari 4.1 irudiaren ezkerreko izkinan.

`kepler()` funtzioan doitasun bikoitzeko zenbaki errepresentazioarekin errorea kalkulatu ezkero 10^{-16} inguruko balio maximoak lortzen dira. Aurreko atalean baino errore handiagoa dago, biribiltze erroreak eragina baitu. 3 azaltzen diren eragiketa aldaketak funtzioan inplementatuak daude biribiltze errorea ahalik eta txikiena izan dadin. 4.2 irudian ikusten den bezala erroreak balio diskretuetan banatzen dira. Hau da, ez dute ϵ -en eremu guztia hartzen z ardatzean baizik eta saltoka. Hau doitasun bikoitzeko zenbaki errepresentazioaren limitearengatik sortzen da. Julia lengoaian `eps()` funtzioak makinaren epsilon balioa bueltatzen du, hau da, 1.0 -ri gehitu daitekeen zenbaki dezimalik txikiena. Kasu honetan Juliak `eps() = 2.220446049250313e-16` du, lorturiko errore maximoaren erdia.

4.1.2 Ekepl

Gooding eta Odell [9] -en ikerketetan azaldutako EKEPL metodoa `ekepl()` funtzioan programatua da. Metodoaren errore maximoa 2.32×10^{-14} da. Ez du arazorik eremu singularrean baina errorearen balioa biribiltze errorea baina handiagoa da.

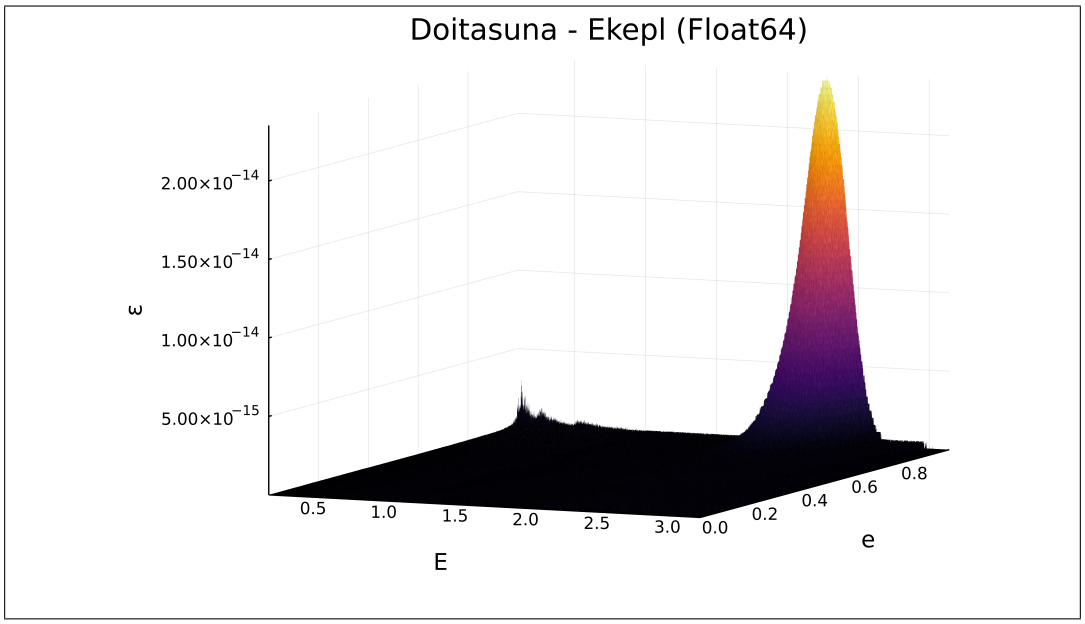
`ekepl()` funtzioa doitasun bikoitzeko zenbaki (Float64) errepresentazioarekin kalkulatzean antzeko irudi eta balioak lortzen dira. 4.4 irudian daude sareko errore guztiak. Biribiltze errorearen eragina sumatzen da, balioak ez baitira erabat leunak, ϵ salto diskretuka baitoa. Eremu singularrean errore balio batzuk sumatzen dira, doitasun handiz buruturiko kalkuluan ez daudenak. M-ren balio oso txikiak eta e 1 etik oso gertu dagoenean funtzioak oso zenbaki txikien arteko kenketak egiten ditu. Hori dela eta, baliteke sare guztia balio gehiagorekin bete ezkero erroreak handitzea eremu horretan.



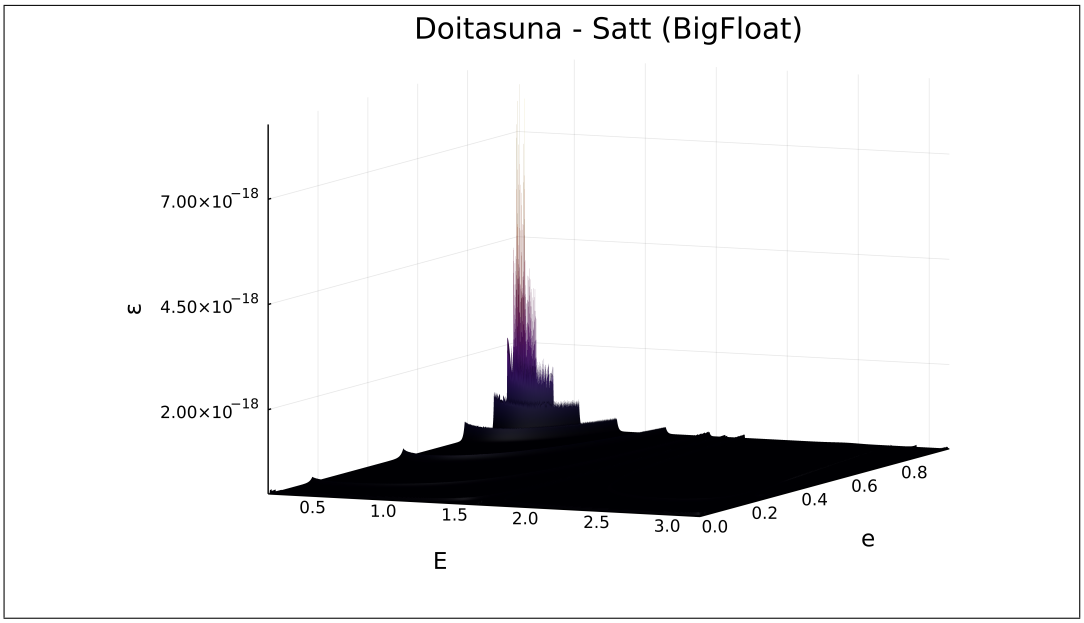
4.3 Irudia: Gooding eta Odell-en Ekepl algoritmoaren errore absolutua doitasun handiko zenbakiz kalkulatua.

4.1.3 Satt

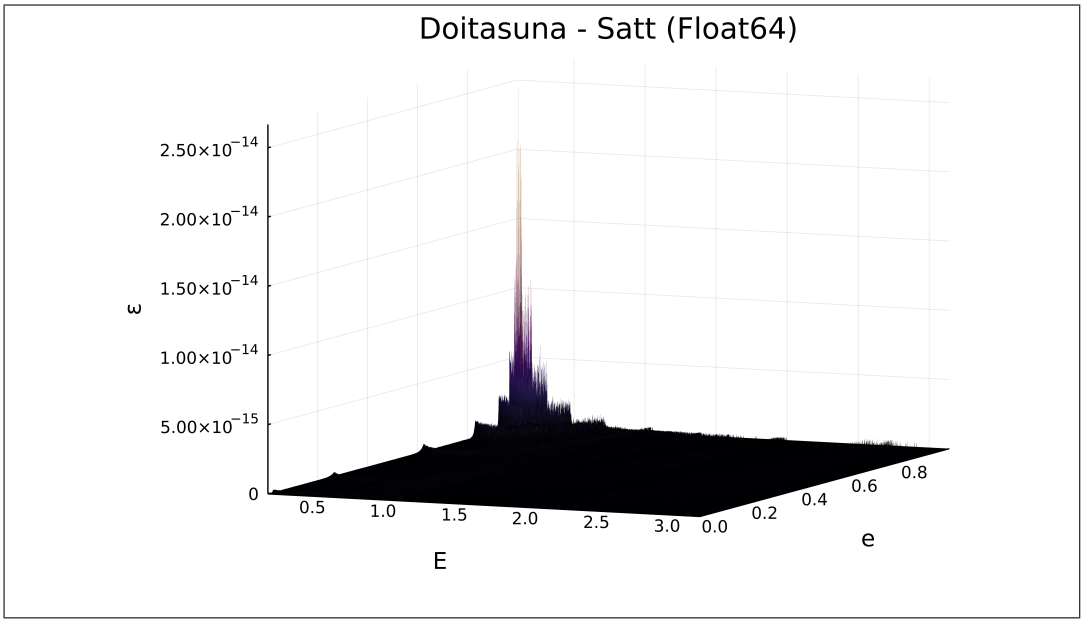
SatelliteToolbox paketearen funtzioa erabili da doitasuna ebaluatzeko. Funtzioan doitasun balioa parametro bezala sar daiteke. Funtzio honen ebazpena prozesu iteratiboa denez, esandako doitasunera iristean amaitzen du iterazio prozesua. Baita iterazio kopuru maximoa ezarri daiteke. Errore konprobaketa honetarako iterazio zenbaki maximoa 10 erabili da eta lortu nahi den doitasun balioa ezberdina Float64 eta BigFloat-entzat.



4.4 Irudia: Gooding eta Odell-en Ekepl algoritmoaren errore absolutua doitasun bikoitzeko zenbakiz kalkulatua.

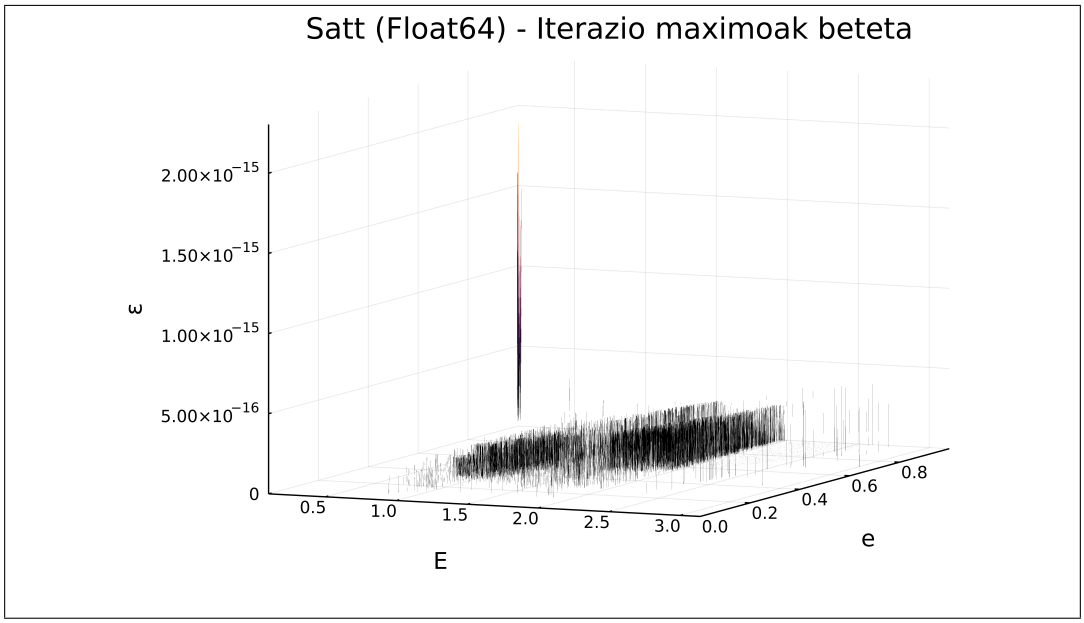


4.5 Irudia: SatteliteTools paketearen algoritmoaren errore absolutua doitasun handiko zenbakiz kalkulatua.



4.6 Irudia: SatteliteTools paketearen algoritmoaren errore absolutua doitasun bikoitzeko zenbakiz kalkulatua.

Doitasuna parametro bezala ematen denez eta hau lortu arte prozesu iteratiboa amaitzen ez denez, errore balio txikiak espero dira. Hala ere, eremu singularrean funtzioaren eta deribatuaren balioak oso txikiak direnez, konbergentzia oso motela izan daiteke. Kasu hauen emaitzak eta biribiltze errorearenak aztertzekotan egin dira probak. Lehenik doitasun handiko zenbaki errepresentazioarekin. Kasu honetan funtzioari eskaturiko doitasuna biribiltze errorea baino handiagoa izan da 1×10^{-19}. Lorturiko errore maximoa 8.78×10^{-18} da. 4.5 irudian azaltzen den bezala errore altuenak eremu singularrean kokaturik daude.



4.7 Irudia: SatteliteTools paketearen algoritmoan 10 iterazio betetako erroreak doitasun bikoitzeko zenbakiz kalkulatua.

Errorea eskaturiko doitasun minimoa baino handiagoa da batzuetan, iterazioak geratzeko irizpidea ez baita zuzenki errorean oinarritzen. Baita ere, baliteke 10 iterazioak bete izana. Erroreak txikiak diren arren, funtzioa ez da bate eraginkorra, hurrengo atalean aztertuko den bezala.

Float64 data motarekin erroreak kalkulatzerakoan funtzioaren doitasun parametroa makinaren epsilon erabili da, 2.22×10^{-16}. 4.6 irudian errore balioak sare osoak ikusten dira, guztien maximoa 1.28×10^{-14} da. Prozesua iteratiboa denez ezin da jakin soluziora 2 iterazio edota 300 egin ondoren heldu den. Horregatik 10 iterazio (ezarritako limitea) baino gehiago dituzten erroreak irudikatu dira (4.7). Sare osoaren %8 ak 10 iterazioetara iritsi da doitasun baldintza bete gabe. Iterazio balioak gehitu ondoren beti limitera heltzen direla ikusten da, beraz, biribiltze errorearen eraginarengatik da.

4.2 Eraginkortasuna

Lan honetan garaturiko algoritmoak errore txikia eta bestalde eraginkorra izatea du helburu. Horretarako, bai programatze eta formulazioa burutzean, kalkulua ahalik eta gutxienak egitea eta hauek optimizatuak izatea eduki da kontutan. Ansatz-eko λ balioen eragiketak FMA instrukzio bidez egiten dira. Eragiketa guztietatik, funtzio trigonometrikoak garestienak dira. Kepler funtzioan soilik hiru kalkulatzen dira. Behar diren beste balioak Taylor-en garapenean burutzen dira (biribiltze errorea gutxitzeko), edo bertatik eragiketa sinpleen bidez lortzen dira.

Eraginkortasuna ebaluatzeko funtzioen exekuzio denbora neurtu da. Funtzio deialdi bakarrak oso azkarrak dira eta ordenagailuen denbora doitasuna ez da oso fidagarria nanosegundo inguruan (nahiz eta balio horik eskaini). Arrazoi honengatik, behin baino gehiagotan neurtu da exekuzio bakoitza, batezbesteko balioak erabiltzeko. Neurturiko denborak ez dute garrantzi handirik (erabilitako ordenagailuaren eta funtzio deialdi kopuruaren menpe

daude), baina bai algoritmoen arteko konparaketak.

Funtzioen parametroak, sarrerako (e, M) balioak ez du eraginik exekuzio denboran, Sat funtzioan izan ezik. Hau iteratiboa denez urrats gehiago edo gutxiago ematen ditu. Beraz exekuzio denbora e eta E balioen sare guztirako kalkulatu dira, sarea eremu diskretuetan bananduz. $e(0, 1)$ 29 zatitan banatu da, $E(0, pi)$ 40 zatitan 4.8 irudian adierazi bezala. Eremu bakoitzean 351×350 elementuko sub-sare bat sortzen da. Adibidez, lehenengo eremua $e = 0.0, e = 0.1, E = 0.0, E = 0.1963$ balioek mugatzen dute, non $\pi/16 = 0.1963$. Bertan 122850 balio bikoteko sarea sortzen da eta hauen ebaluazioaren denbora neurtzen da. Denbora neurketak fidagarriak izan daitezen, 30 aldiz errepikatu dira eta guztien batezbestekoa gorde da. Algoritmo bakoitzeko 135 denbora balio neurtu dira. Balio hauen batezbestekoa, maximoa, minimoa eta desbideratze estandarra 4.2 taulan daude.

Algoritmoa	Max	Min	Mean	Std
Kepler	0.016	0.0088	0.0119	0.0013
Ekepl	0.0244	0.0153	0.0199	0.0019
Satt	0.0448	0.0067	0.0134	0.0056

4.2 Taula: Algoritmo bakoitzaren denbora sarearen balio estatistikoak

Denborak neurtzeko erabilitako ordenagailuaren CPU-a textbf11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz × 8 16 BG eko RAM eta **Intel AVX-512 "Instruction Set Extension** duelarik. Batezbesteko eta desbiderazio estandarrari begira ondoriozta daiteke:

- Kepler eta Ekepl funtzioak denbora uniformea behar dute eremu guztian zehar, desbiderazio estandar txikia.
- Kepler eta Ekepl funtzioak denbora antzekoa behar dute, batezbesteko balio antzekoa.
- Satt funtzioak ez du exekuzio denbora uniformea eremu guztian, desbiderazio estandar handia baitu.

4.2.1 Kepler

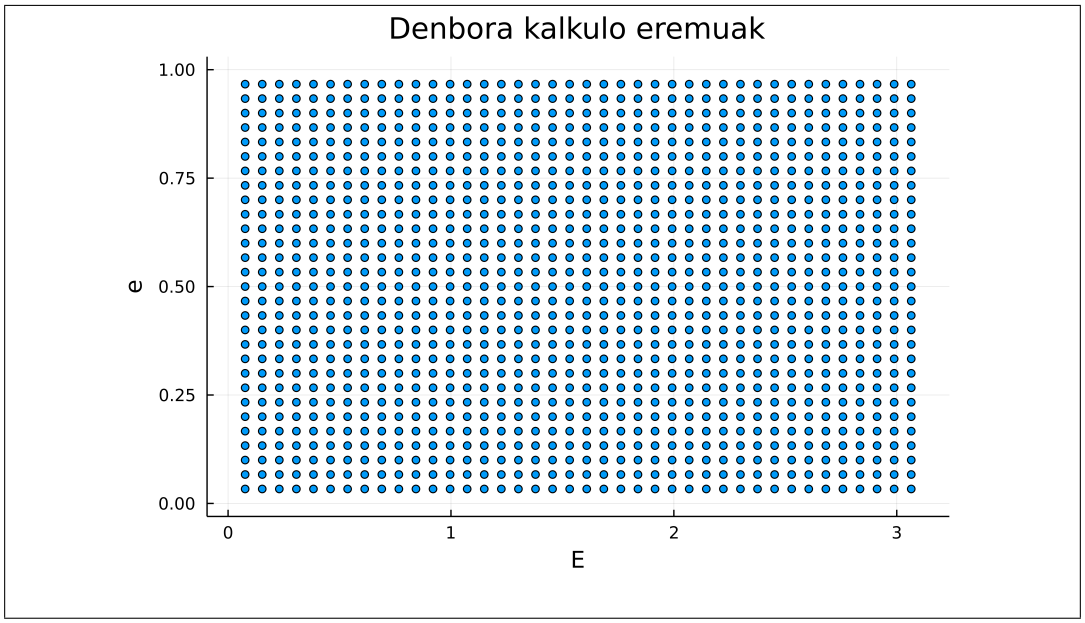
Kepler funtzioaren exekuzio denbora uniformea da eremu guztietan. 4.9 irudian ikusten den bezala 16 ms behar ditu 351×350 tamainako sarean ekuazioa ebazteko, hau da 122850 aldiz.

4.2.2 Ekepl

Ekepl funtzioaren exekuzio denbora baita uniformea da eremu guztietan, parametroen balioak ez du eragin handirik exekuzio denboran. 4.9 irudian eremu bakoitzeko exekuzio denbora irudikatu da. Kepler metodoan bezala 16 ms behar ditu sub-sarea ebaluatzeko.

4.2.3 Satt

SatelliteToolbox paketeko funtzioaren ebaluazio denborak ez dira uniformeak. Eremu batzuen kalkuluak iterazio kopuru handiago behar duenez denbora gehiago behar du soluzioa kalkulatzeko. Iterazioen limitea 10-etik handitu ezkero denbora ezberdintasunak nabarmenagoak dira. Sub-eremu bat kalkulatzeko 8 ms behar ditu batezbeste, baina maximoa

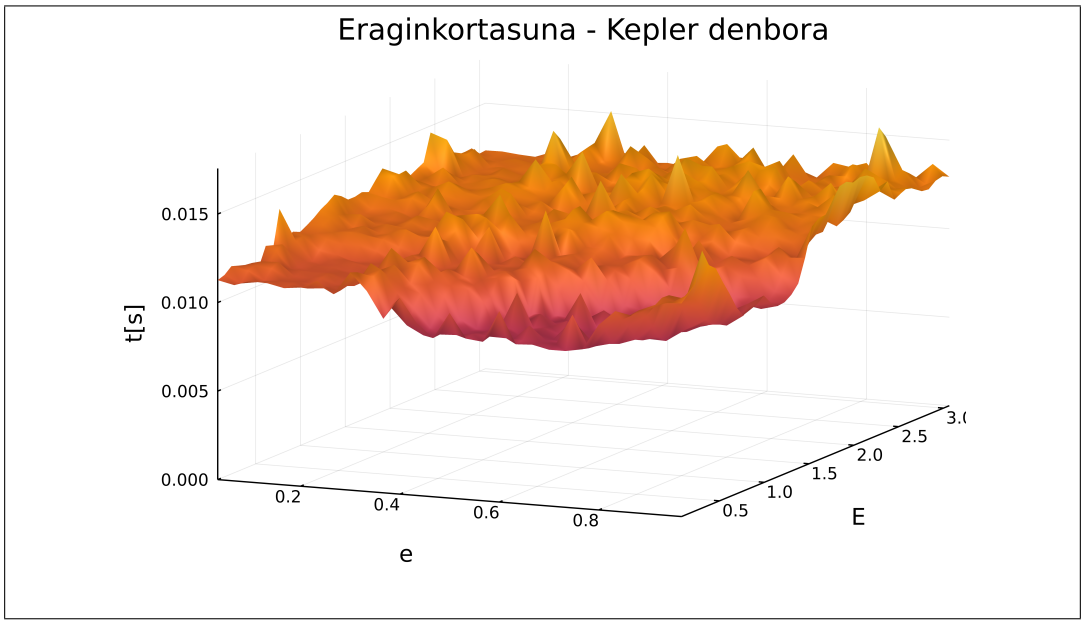


4.8 Irudia: Exekuzio denbora kalkulatzeko erabilitako eremuak.

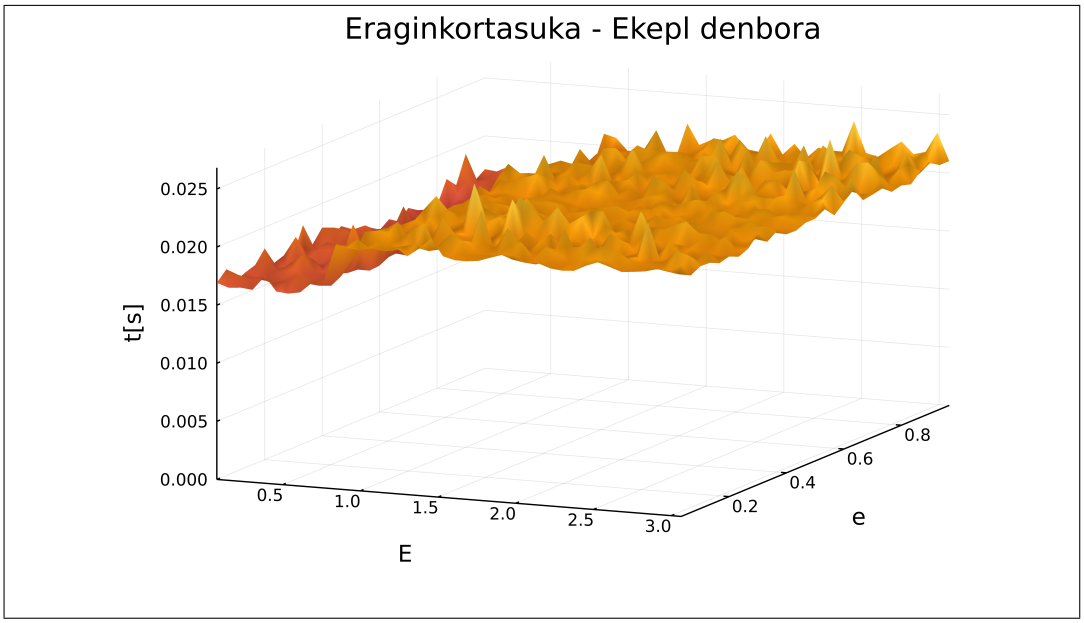
29 ms-koa da. $e = 1$ eta $E = 0$ -tik gertu dauden balioek iterazio gehiago behar dituzte emaitza lortzeko eta ondorioz denbora gehiago.

4.2.4 Konparaketa

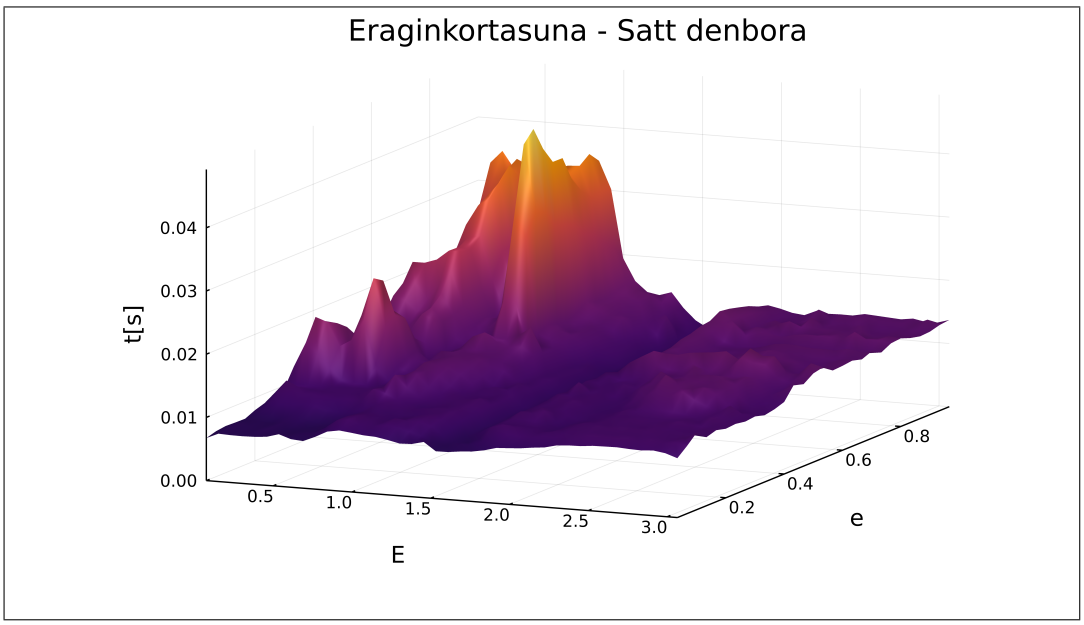
Denbora neurketak erabilitako sub-sarearen tamaina eta ordenagailuaren eragina dute. Bi aldagai hauen efektua deuseztatu nahian erlazio adimentsionala kalkulatu da. Hau da Kepler funtzioaren ebaluazio denbora Ekepl eta Satt funtzioarenarekin erlazionatu.



4.9 Irudia: Kepler funtzioaren exekuzio denbora eremuka lortua (351 x 350 sareetan).

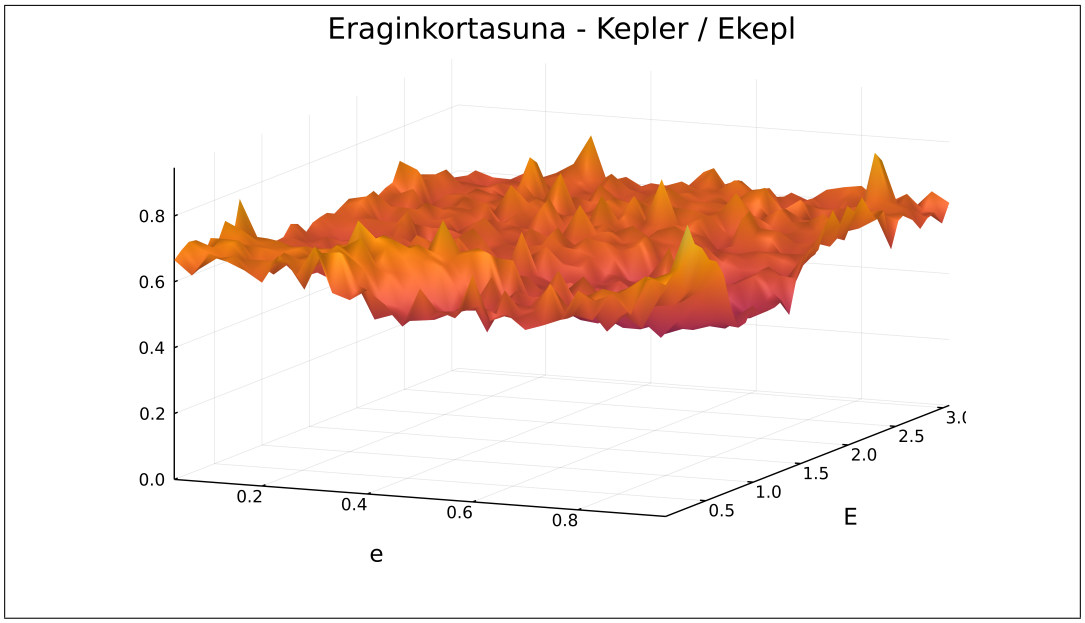


4.10 Irudia: Kepler funtzioaren exekuzio denbora eremuka lortua (351 x 350 sareetan).



4.11 Irudia: Satt funtzioaren exekuzio denbora eremuka lortua (351 x 350 sareetan).

- Kepler eta Ekepl metodoen exekuzio denborak uniformeak direla ikusi da. Funtzio bakoitzaren exekuzio denbora sarrera parametroengandik independentea da. Honek konparaketa zuzenean egitea ahalbidetzen du. Horretarako (e, M) sare osoa 5000 ataletan diskretizatu da. Guztira 25 milioi aldiz egin da kalkulua, honen denbora neurtuz funtzio bakoitzarekin. Prozesu hau 10 aldiz errepikatu da, batezbesteko denborak lortuz. Kepler funtzioak 2.485 s behar izan ditu eta Ekepl funtzioak 4.059 s. Kepler funtzioaren exekuzio denbora Ekepl-enare %60 a. 4.12 irudian ikusten den bezala bi exekuzio denboren arteko erlazioa uniformea da, 0.6 balio inguruan.

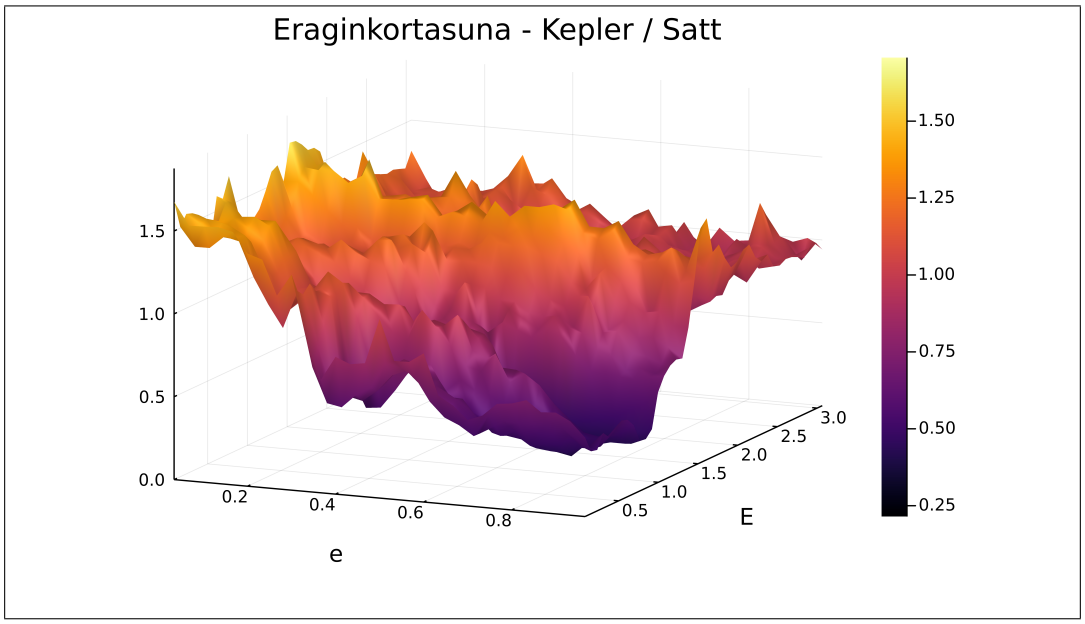


4.12 Irudia: Exekuzio denbora kalkulatzeko erabilitako eremuak.

- Satt funtzioaren exekuzio denborak uniformeak ez direnez, konparaketa eremuka egin behar izan da. Eremu osoko denborak zatituz 4.13 irudiko erlazio baloreak lortzen dira. Ingurune singularrean 0.25 eko erlazioa dago, hau da, Kepler lau aldiz azkarragoa da ebazten Satt baino. Beste inguru batzuetan ordea 1.50 balioak daude, Kepler motelagoa delarik konbergentzi azkarra dagoen inguruetan.

.

Eremuka eginiko konparaketaren laburpena 4.3 taulan idatzia dago. Bertan Kepler



4.13 Irudia: Exekuzio denbora kalkulatzeko erabilitako eremuak.

Algoritmo erlazioa	Max	Min	Mean	Std
Kepler/Ekepl	0.8605	0.43	0.6023	0.0531
Kepler/Satt	1.7066	0.2147	0.9993	0.3026

4.3 Taula: Kepler funtzioaren exekuzio denbora erlazioak Satt eta Ekepl-enekin.

eta Ekepl-ren arteko balioak uniformeak direla eta %60-ko denbora murrizketa dagoela ikusten da. Kepler eta Ekepl denboren arteko erlazioan balioak uniformeak ez direla ikusten da, batzuetan azkarragoa eta besteetan motelagoa izanik. Hala ere, kasurik okerrenean eraginkorra izatea garrantzitsua denez, Kepler-en funtzionamendua hobesten da.

4.3 Bektorizazioa

Optimizazio bidez lorturiko algoritmoaren doitasuna modu sekuentzialean aztertu ondoren bektorizaturik egin da. Lehenik, programaturiko kodea bektorialki eta sekuentzialki exekutatzen direla egiaztatu da. Ondoren biak konparatu dira. Programatu diren bi algoritmoak konparatu dira, Kepler-en ekuazioa ebazten duena eta bi gorputzen problema kalkulatzen duena.

4.3.1 Kepler-en ekuazioa

Julian programaturiko `kepler(e, M)` funtzioa erabili da SIMD bidez exekutaturiko probak egiteko. Modu bektorizatuan e eta M eskalarrak izan ordez bakoitza bektore bat izango da, funtziotik irteten den emaitza bezala. Doitasun bikoitzeko zenbakiekin errore txikia lortzeko sortua denez algoritmoa, SIMD bektoreak `Float64` zenbakiekin beteko dira. Probak egiteko erabili den ordenagailuaren CPU-a "11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz $\times$ 8"da eta Intel AVX-512 "Instruction Set Extension"du. Beraz 64 bit eko 8 elementuko bektoreak erabil ditzake SIMD instrukzioetan.

- `e = Vec{8, Float64}`
- `M = Vec{8, Float64}`
- `E = Vec{8, Float64}`

Eraginkortasuna neurtzerakoan sare handi bat ebaluatzeko behar den denbora neurtu da, $m = 4000, \quad n = 4000$. Alde batetik era sekuentzialean egin dira 16 milioi kalkuluak, bestetik era bektorialean datuak zortzinaka sartuz. Eraginkortasuna neurtzerakoan funtzioaren deialdiak duen eragina ez neurtzeko eraldatu egin da. Esate baterako, ez da 16 miloi aldiz deitu `kepler()` funtzioa. Neurketarako beste funtzio bat programatu da, sare osoa kalkulatzen duena `for` baten barruan. Kalkulu sekuentzialean sareko elementuak bezainbeste iterazio egin dira, era bektorialean ordea zortzinaka. Iterazio bakoitzean bi `Vec{8, Float64}`bektoree jasotzen ditu, hirugarren bat kalkulatuz emaitza bezala.

Konparaketan memoria kudeaketak eragin ahalik eta gutxiena izan dezan ez da erreserbarik egin. Lehendik sorturiko sarea erabiltzen dute (e, M) parametroak jasotzeko, eta baita lehendik erreserbaturiko memoria zatian gordetzen dira emaitzak, sekuentzialak banaka eta bektorizatuak zortzinaka. Eraginkortasun froga 10 aldiz errepikatu da emaitzak aldakorrak ez direla ikusteko.

Lehen probak egin eta gero soilik %30 inguruko denbora murriztea sumatu da. Bektorizazioak abantaila handiagoa ematea espero zenez, kodea aztertu da arrazoiaren bila.

Lehenik berriro optimizatu da kode guztia. Ahalik eta eragiketa gehienak FMA instrukzio bidez programatu dira. Baita funtzio trigonometrikoen kalkulua optimizatu ere. Honek bai era sekuentzialean eta bektorialean emaitzak hobetzen ditu, baina aldaketa handirik gabe.

Ondoren eragiketa guztiak aztertu dira ea zein den beraien eragina exekuzio denboran. Sinpleenak ez dira aztertu baina bai funtzio trigonometrikoak eta eragiketa konplexuagoak. Hauen artean ekuazio kubikoa ebazte prozesuan erabiltzen den erro kubikoa. Bai funtzio trigonometrikoak eta bai erro kubikoak eragin handia dute eraginkortasun eta bektorizazioan.

Funtzio trigonometrikoen eraginkortasunean duten inpaktua ikusirik, funtzioa optimizatu egin da. Hasiera batean hiru funtzio trigonometriko deitzetik, soilik bi deialdietara murriztu da. $\sin M$, $\cos M$, $\sin \frac{M}{2}$ lortu behar dira. Identitate trigonometrikoak erabiliz angelu erdiaren funtzioak kalkulatuz guztiak lortzen dira. Funtzio trigonometrikoak ebaluatuz ondorengoak jakinik.

$$\sin \frac{M}{2}, \qquad \cos \frac{M}{2} \tag{4.3}$$

$\sin M$, $\cos M$ ondorioztatu daitezke hurrengo ekuazioetan bezala. .

$$\sin M = 2 \sin \frac{M}{2} \cos \frac{M}{2} \tag{4.4}$$

$$\cos M = 1 - 2 \sin^2 \frac{M}{2} = 1 - 2 \sin \frac{M}{2} \sin \frac{M}{2} \tag{4.5}$$

Ekuazio kubikoa ebazterakoan erro kubiko bat erabiltzen da. Hau $x^{\frac{1}{3}}$ bezala programaturik dago. Eragiketa honek eragin handia du eraginkortasunean, bai modu sekuentzial eta bektorizatuan. Hau dela eta, berridatzi egin da logaritmoak erabiliz kalkulatzeko.

$$\sqrt[3]{x} = x^{\frac{1}{3}} = e^{\frac{1}{3} \ln x} \tag{4.6}$$

Ondoren SIMD bektorizazioan programaturiko funtzioek duten jokabidea aztertu da. Horretarako lehenengo kapituluan azaldutako LLVM eta Native kodeak aztertu dira. `kepler()` funtzioaren eta baita eragiketa isolatuen konpilazio urratsen kodea aztertu da, ea bektorizazioa egokia den jakiteko. LLVM kodean funtzioko instrukzio guztiak era bektorizatuan sortzen dira (hau baita SIMD.jl paketearen eginkizuna), ez ordea Native kodean. Hau da, $\sin x$ soilik kalkulatzen duen funtzio batean LLVM instrukzioak 8 Float64-eko bektore batekin burutzen ditu, Native kodean ordea, 8 aldiz ikusten da instrukzio bera. Batuketa, biderketa edota FMA-k burutzerakoan ordea, bai LLVM eta Native kodean bektorizaturik ikusten dira instrukzioak. Konpilazioaren azken urratsa hardware-ak baldintzaturik dago, bertan SIMD bidez funtzio trigonometrikoen instrukzioak ez badaude ez dago exekutatzerik. Hala ere, LLVM kodea era bektorizatuan egina dago eta SIMD.jl paketean erabiltzea dagoela argitaraturik. Beraz probak egin ondoren hurrengoa ondorioztatu da.

- Funtzio trigonometriko eta erro kubikoa ebaluatzeak eraginkortasun galera suposatzen du modu sekuentzialean.
- Funtzio trigonometriko eta erro kubikoa LLVM kodean bektorizaturik dauden arren, ez dira hardware-ko instrukzioetara bektorizatuta pasatzen. Probak egin diren makinan ez ditu SIMD bidez funtzio trigonometrikoak kalkulatzeko instrukzioak.

Probak egiteko `kepler()` funtzioa eraldatu egin da hiru bertsio sorturik. Baten ez da erro kubikorik erabiltzen, hurrengoan funtzio trigonometrikorik eta azkenekoan ez erro kubiko eta ezta funtzio trigonometrikorik.

- `kepler()` funtzio normala.
- `kepler2()` ez du erro kubiko instrukziorik baizik eta logaritmo bidez kalkulatzen da, 4.6 ekuazioa.
- `kepler3()` funtzioak ez du funtzio trigonometrikorik. 10 FMA instrukzioekin ordezkatu dira. Doitasun aldetik fidagarria ez den arren, funtzio trigonometrikoak kentzearen eragina aztertzeko egin da. Bertan Taylor-en hurbilpenaz kalkulatu dira.
- `kepler4()` funtzioan `kepler2()` eta `kepler3()` -ren aldaketak gehitzen dira. Ez du ez funtzio trigonometriko ezta erro kubikorik erabiltzen. Bi eraldaketen efektua neurtzeko egina da. Honen emaitzak ere ez dira doitasun handikoak izango.

Hiru funtzioak modu sekuentzialean eta bektorizatuan exekutatu dira denborak neurtuz. Emaitzak 4.4 taulan daude.

Funtzioa	t sekuentziala	t bektorizatua	hobekuntza
`kepler()`	1.922	1.271	1.513
`kepler2()`	0.889	0.598	1.487
`kepler3()`	1.548	1.119	1.384
`kepler4()`	0.725	0.413	1.756

4.4 Taula: Kepler funtzioen exekuzio denborak era sekuentzial eta bektorizatuan.

Emaitzetan ikusten den bezala, erro kubikoa ez erabiltzeak eragin handia du. Denborak erdira jaisten dira modu bektorizatu eta sekuentzialean. Hala ere bektorizatzeko garaian abantaila oso antzekoa da %50 eko azkartasuna lortzen da. Funtzio trigonometrikoak ez erabiltzeak abantaila txikia du modu sekuentzialean, eta ez du asko aldatzen bektorizatzeak %50 inguruko azkartasuna lortzen da. Ez funtzio trigonometriko ezta erro kubikorik ez erabiltzean emaitzarik azkarrenak lortzen dira. Denbora sekuentziala erdira baino gehiago jaisten da (bi aldaketen gehiketa). Baita bektorizatzerakoan gehiago hobetzen dira denborak. Bektorizatuta kalkulaturik sekuentzialki baino 1.75 aldiz azkarragoa da.

4.3.2 Bi gorputzen problema

Julian programaturiko `KeplerFlow` funtzioa erabili da SIMD bidez exekutaturiko probak egiteko. 1 atalean azaldu den bi gorputzen problema ebazten du. Hasierako abiadura eta posizio bektoreak, $q0, v0 \in \mathbb{R}^3$, emanik, t unean abiadura eta posizio berriak kalkulatzen ditu, qt, vt. k parametroa ere definiturik behar du.

Modu sekuentzialean eragiketak eginez, `q0, v0, qt, vt` 3 elementuko bektoreak dira. Era bektorialean burutzeko ordea dimentsio bat gehiago gehitu behar da. Kepler-en ekuazioaren ebazpen bektorizatuan bezala SIMD bektoreak `Float64` zenbakiekin beteko dira. Beraz kasu honetan 3 elementuko bektoreak 8×3 -ko bektoreetara bihurtu behar ditugu. Posizio bektoreak 3 koordinatuen posizioa du `q0 = [x, y, z]` . Bektorizatzeko matrizean ordea `q0 = [[x], [y], [z]]` elementu bakoitza 8 zutabeko `Vec{8, Float64}` SIMD bektorea da. Elementu mota berri hau `Vec8Vector` bezala izendatu da. Modu bektorialean kalkuluak egiterakoan zutabeka hartuko dira elementuak eta hauek SIMD bektoreak izanik eragiketa danak era bektorizatuan burutzen ditu. Era bektorizatuan sarturiko parametroak hauek dira

- `q0v = Vec8Vector{Float64}(8 x 3)`
- `v0v = Vec8Vector{Float64}(8 x 3)`
- `qtv = Vec8Vector{Float64}(8 x 3)`
- `vtv = Vec8Vector{Float64}(8 x 3)`
- `qtv = Vec8Vector{Float64}(8 x 3)`
- `qtv = Float64`

Eraginkortasuna neurtzeko `BenchmarkRools` paketea erabili da. Funtzio deien eraginkortasuna neurtzeko aukerak ditu. Kepler-en ekuazioaren konparaketan bezala, funtzio berriak sortu dira deialdi asko egiten dituztenak. Kasu honetan 10000 aldiz kalkuluak egiten dituena. 10000 balio sortu dira `q0, v0, k` bektoreentzat. Auskazko metodoekin sortu diren arren, orbita eliptikoak dituztela ziurtatu da. Horretarako $\|v| \in [0, \sqrt{\frac{2k}{\|q\|}}]$ ziurtatu da. Ondoren modu sekuentzialean `for` baten bidez parametro guztiak kargatu eta exekutatu dira. Modu bektorialean ordea `Vec8Vector` elementuetan, 8×3 tamainakoetan. Memoria erreserbarik ez egoteko kalkuluak egiteko `qt, vt` bektoreak sortuta pasatu dira, baita emaitzak idazteko matrizeak.

Eraginkortasun frogak Kepler-en ekuazioarentzako erabili diren 4 funtzioen bertsioekin egin dira.

270 aldiz errepikatu du `BenchmarkTools` paketeak. Bertako `keplerFlow` funtzioetan 10000 aldiz kalkulatu da bi gorputzen problemaren soluzioa. `keplerFlow` funtzioak `kepler` funtzioa erabiltzen du barne, `keplerFlow2` -k `kepler2`. Batezbesteko denborak 4.5 taulan daude milisegundoetan adierazua

Funtzioa	t sekuentziala [ms]	t bektorizatua [ms]	hobekuntza
`keplerFlow()`	21.12	8.60	2.46
`keplerFlow2()`	16.41	4.77	3.44
`keplerFlow3()`	18.89	7.23	2.61
`keplerFlow4()`	13.90	4.04	3.44

4.5 Taula: KeplerFlow funtzioen exekuzio denborak era sekuentzial eta bektorizatuan.

Kasu honetan bektorizazioaren abantaila handiagoa da. Baliteke eragiketa gehiago burutzen direnez, hauen bektorizazio egokiak prozesua azkartzea. `keplerFlow()` funtzioan

2.46 aldiz azkarrago burutzen ditu kalkuluak era bektorialean. Erro kubikoa logaritmo bidez kalkulatzean era sekuentzialean %25 eko azkartzea lortzen da eta baektorizatzean 3.44 aldiz azkarrago burutzen ditu kalkuloak. Funtzio trigonometrikoak kentzeak abantaila txikia du era sekuentzialean eta funtzio originalaren antzeko abantaila du bektorizatzean. Azkenik funtzio trigonometrikoak eta erro kubikoa kentzerakoan funtziorik azkarrena lortu da. Sekuentzialki denbora erdira jaitsi da eta bektorizatzerakoan 3.5 aldiz azkarragoa da. Bi gorputzen problema kalkulatzerakoan eragiketa gehiago daude bektorizagarriak, beraz funtzio trigonometriko eta erro kubikoen eragina txikiagoa da, bektorizazioak gehiago azkartuz exekuzio denbora, 2 eta 3 aldiz azkarragoak izanez.

KAPITULUA 5

Ondorioak

Algoritmo optimizatuaren garatzeko erabili den metodologia eta optimizazio probleman honako ondorioak lortu dira:

- Gooding eta Odell-ek proposaturiko metodoan oinarriturik Kepler-en ekuazioa ebazten duen algoritmo berri bat garatu da. Hasieraketa balio on bat bilatu eta urrats batean hobetzean datza.
- Algoritmoa SIMD bidez bektorizatzeko egokia izan dadin garatu da.
- Algoritmoaren hasieraketa balioa bilatzeko soluzio parametriko batetik (Ansatz) abiatu da. Helburu funtzio bat definituz parametroak optimizatzeko problema bat ebatzi da.
- Hiru Ansatz ezberdin probatu dira, algoritmoak doitasun maila egokia lortu arte (biribiltze errorearen tamainakoa $1 \times 10^{-}16$)
- Doitasun bikoitzeko zenbakizko errepresentazioa erabiltzean biribiltze errorea gutxitzeko eragiketak eraldatu dira. Zenbaki txikien arteko kenketak Taylor-en garapenez kalkulatuz.
- Eraginkortasuna zainduz eragiketa asko FMA-k erabiliz bete dira. Soilik bi funtzio trigonometriko ebaluatzen dira ebazte prozesu guztian.

Algoritmoa Juliako `kepler()` funtzioan programatu ondoren beste bi metodoekin argitaratuekin konparatu da. Doitasuna eta eraginkortasuna aztertuz honako ondorioak lortu dira:

- `kepler()` metodoak emaitza zehatzarekiko duen errorea biribiltze errorea bezain txikia izatea lortu da doitasun bikoitzeko zenbakiak erabiltzerakoan. Konparaturiko beste bi funtzioek baino errore absolutu txikiagoa du.
- Sarrera parametro, (e, M) guztientzako exekuzio denbora berdina du, ez da bizkorragoa gune batzuetan besteetan baino. Gooding eta Odell-ek proposaturiko metodoa baino eraginkorragoa da, exekuzio denbora batezbestekoz %60 azkarragoa da.

Algoritmoaren eraginkortasuna era bektorizatuan eta sekuentzialean neurtu da, SIMD bidez kalkulatzeak duen abantaila neurtzeko.

- `kepler()` funtzioa era bektorizatuan ebazterakoan 1.51 aldiz azkarragoa da.
- Eraginkortasunean inpaktu handiena duten eragiketak bilatu dira. Funtzio trigonometrikoak eta $1/3$ -ren berredura (erro kubikoa kalkulatzeko). Hauen eragina gutxitzeko programazioa eraldatu egin da, identitate trigonometrikoak erabiliz (soilik bi funtzio trigonometriko ebaluatzeko) eta berredura logaritmo bidez kalkulatuz.
- Julia-ren konpilazio kodeak aztertzerakoan SIMD instrukzioak hardware mailara iristen ez direla ikusi da. Nahiz eta horretarako programaturik egon, probak egin diren makinetan ez dago funtzio trigonometrikoak modu bektorizatuan ebazterik. Honek bektorizazioaren abantaila mugatu du.
- Bektoriztzerakoan abantaila handiak lortzeko, (modu sekuentzialetik bektorizatura denborak erdira baino gehiago jaistea) funtzio trigonometriko eta erro kubikoak ekidin behar dira, beste era batera (SIMD bidez burutu daitezkeenak) kalkulatu behar dira.
- `kepler()` funtzioa bi gorputzen problema ebazteko erabili da, prozesu guztia bektorizatuz. Ebazpen bektorizatua 3.44 aldiz azkarragoa izatea lortu da. Eragiketa kopurua handia denez, funtzio trigonometrikoen eragina txikiagoa izan da abantaila handiagoa lortuz.

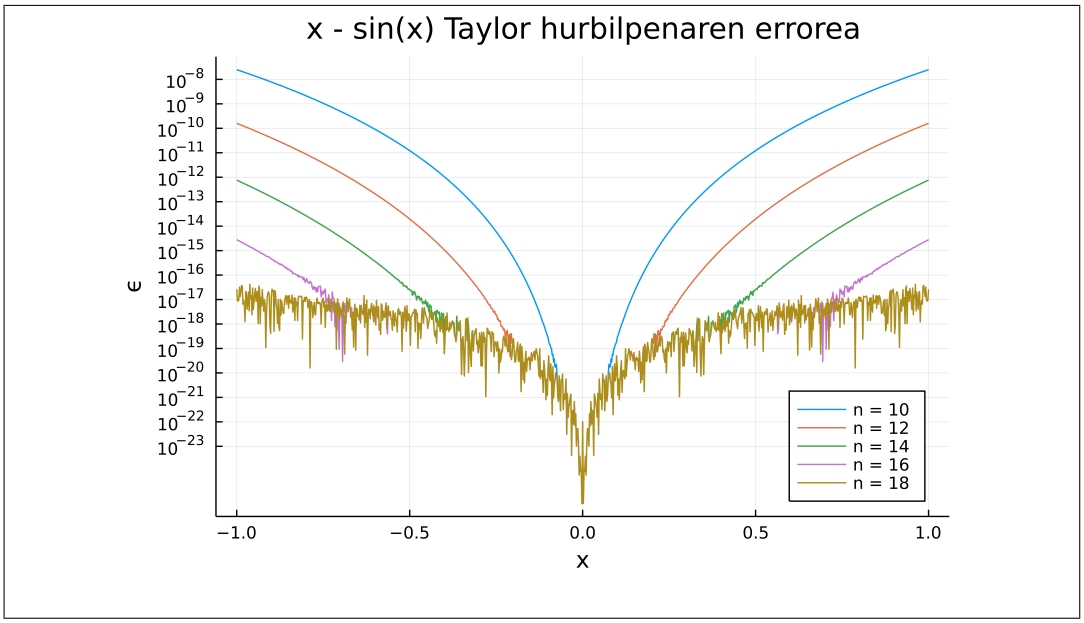
5.1 Etorkizunerako lanak

Master amaierako lanean zehar esperimentu asko egin badira ere, hainbat proba eta hobekuntza bertan behera utzi behar izan dira denbora falta dela eta. Probetan ondorioztatu denean oinarriturik hiru hobekuntza proposatzen dira aztertzeko.

Bektorizatzeko eraginkortasun esperimentuetan erro kubikoa kalkulatzeak eragin handia duela ikusi da. Ekuazio kubikoa zehazki ebazteko beharrezkoa den pausua da, beraz ezin da ekidin. Exekuzio denboran eragin handiena duen pausua dela ikusi da. Ikerketa zabaltzerakoan urrats hau nola ekidin aztertzea proposatzen da. Bai ekuazio kubikoa ebazteko modu berri bat bilatuz edota ekuazio kubikorik erabili gabeko metodo bat garatuz.

Algoritmo optimizatuak 18 λ balio erabiltzen ditu hasieraketa balioa lortzeko. Baliteke algoritmoa zehatzegia izatea. Hau da, potentzialki (BigFloat bidez kalkulaturik) 10^{-19} inguruko errorea du baina doitasun bikoitzeko zenbakiekin 10^{-16} -era bakarrik iristen da. Ansatz-a murriztuz eragiketa kopurua txikitzea aztertu daiteke, beti ere doitasun bikoitzeko zenbakiz kalkuluak eginez errorea mantentzen delarik. Baita ere, Ansatz-a gehiago murriztu daiteke, edo beste bat planteatu, doitasun soileko zenbakizko ebazpenerako, bertan 16 elementuko SIMD bektoreak erabiliz.

Biribiltze errorea gutxitzeko Taylor-en garapena erabili da, bai $x - \sin x$ eta $1 - \cos x$ eragiketetan. $x \in [-1, 1]$ dagoenez, errorea eremu guztian 10^{-18} baino txikiagoa izan dadin ziurtatu da. Honek definitu du polinomioaren maila, 18. Taylor-en hurbilpenak errore oso txikia du $x = 0$ inguruan, baina urrutiratzean asko hazten da, 5.1 irudia. Errorea uniformeki banatzen duen hurbilpen polinomio bat aztertu daiteke bi eragiketa hauen



5.1 Irudia: Maila ezberdineko Taylor-en polinomioen bidezko hurbilpenaren erroreak $x - \sin x$ eragiketarako.

emaitzak lortzeko. Taylor-en polinomio trunkatuarekin hurbildu ordez beste hurbilpen bat erabili ezkero maila txikiagokoa erabili daiteke, beraz eragiketa kopurua murriztuz.

Bibliografia

[1] R.E. Deakin. *Satellite Orbits*. School of Mathematical & Geospatial Sciences, RMIT University, Melbourne, Australia, 2007. Ikusi 1 orrialdea.

[2] P. Colwell. *Solving Kepler's Equation over Three Centuries*. Willmann-Bell, Richmond, Virginia, 1993. Ikusi 3 orrialdea.

[3] Karrenberg. R. *Automatic SIMD Vectorization of SSA-based Control Flow Graphs*. PhD thesis, Saarlanders Universitat, 2014. Ikusi 11 orrialdea.

[4] H. Sun, S. Gorlatch, and R. Zhao. Vectorizing programs with if-statements for processors with simd extensions. *J Supercomput*, 76:4731–4746, 2020. Ikusi 12 orrialdea.

[5] A. Pal. An analytical solution for kepler's problem. *Monthly Notices of the Royal Astronomical Society*, 396:1737–1742, 2009. Ikusi 17 orrialdea.

[6] D. Mortari and A. Elipe. Solving kepler's equation using implicit functions. *Celestial Mechanics and Dynamical Astronomy*, 118:1–11, 2014. Ikusi 17 orrialdea.

[7] F.L. Markley. Kepler equation solver. *Celestial Mechanics and Dynamical Astronomy*, 63:101–111, 1995. Ikusi 17 orrialdea.

[8] R. Serafin. Bounds on the solution to kepler's equation. *Celestial Mechanics and Dynamical Astronomy*, 70:131–146, 1998. Ikusi 17 orrialdea.

[9] R.H. Gooding and A.W. Odell. The hyperbolic kepler equation (and the elliptic equation revisited). *Celestial Mechanics*, 44:267–282, 1988. Ikusi 17, 18, and 37 orrialdeak.

[10] T. Fukushima. A method solving kepler's equation without transcendental function evaluations. *Celestial Mechanics and Dynamical Astronomy*, 66:309–319, 1996. Ikusi 17 orrialdea.

[11] V. Raposo-Pulido and J Pelaez. An efficient code to solve the kepler equation. elliptic case. *Monthly Notices of the Royal Astronomical Society*, 467:1702–1713, 2017. Ikusi 18 orrialdea.

[12] A.Ñijenhuis. Solving kepler's equation with high efficiency and accuracy. *Celestial Mechanics and Dynamical Astronomy*, 51:319–330, 1998. Ikusi 18 orrialdea.

[13] R.H. Gooding and A.W. Odell. *Procedures for solving Kepler's equation*. Royal Aircraft Establishment, Hants, England, 1986. Ikusi 18, 29 orrialdeak.

[14] David A. Vallado. *Fundamentals of Astrodynamics and Applications, 4th ed.* Microcosm Press, Hawthorn, CA, USA, 2013. Ikusi 20 orrialdea.