eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Proiektua

Keplerren fluxua Julian N gorputzeko sistema grabitazionaletara aplikatuta

---

Egilea

Ander Salaberria Saizar

Zuzendaria

Ander Murua Uria

informatika fakultatea
facultad de informática

2018

Laburpena

Lan honetan, Keplerren fluxuaren ebazpenaren zehaztasun handiko inplementazio azkar bat aurkezten da, biribiltze-erroreak sor ditzaketen eragiketak eta beharrezkoak ez diren Newton-Rapshonen metodoaren iterazioak ahal izan den moduan ekiditen saiatuz. Gainera, Keplerren fluxua N-gorputzeko problemara aplikatu da energia kontserbatzen duen eta Jacobi koordenatuak erabiltzen dituen Wisdom-Holfman integratzaile bat inplementatuz.

Bi algoritmo hauek Julia programazio-lengoaian kodetu dira, eta inplementazioaren erabilera publiko baterako pakete bat sortu da. Hala ere, Keplerren fluxua C lengoaian kodetu da ere bai —bere eraginkortasunagatik ezagutua—, Juliako kodearekin alderatzeko eta zientziarako konputazioan lengoaia honen bideragarritasuna aztertzeko. Bestalde, Juliako kodean koma higikorreko doitasun ezberdineko datu-motak aztertu dira —*Double*, *BigFloat*—, doitasun handiagoko datu-moten erabilera gomendagarria den ala ez ikusteko —azkartasuna eta zehaztasunak konparatuz—.

Azkenik, lan honetan zehar sistema planetarioen higidurak kalkulatu direnez, eguzki-sistemaren simulazio bat sortu da —Eguzkia, zortzi planetak eta ilargiaren posizioak erakusten dituena sarrerako datu batzuk emanez—. Simulazio hau JavaScript bidez garatu da, Three.JS liburutegia erabiliz.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Gizakiok betidanik izan dugu zerura begiratzeko joera. Hasiera batean, liluraturik, zeruko gorputzak mitologia eta erlijioarekin lotzen ziren, astro[1] horiek zer ziren lehenengo hipotesiak sortzen ari ziren heinean. Filosofo greziarrak —Aristarko Samoskoa, hauen artean, eguzki sistemaren modelo heliozentrikoa proposatu zuen lehena— eklipseak, izarren eta planeten higidurak etab. ulertzen saiatu ziren, garai horietan zituzten baliabide eskasak erabiliz.

Urteak pasa ahala, gure astronomiari buruzko kontzeptu, teknologia eta kalkuluek aurrera egin dute, unibertso ikusgarriko gorputzen ezaugarrien eta higiduren azterketak nabarmenki aurreratu direlarik. Keplerren legeek planeten orbiten ezaugarriak definitu dituzte; horiei esker, Newtonen grabitazio unibertsalaren legea garatu da, eta zenbakizko metodoek —eta, oro har, kalkuluak— ekuazio transzendentalen ebazpenen hurbilpenak ahalbidetu dituzte, besteak beste. Aurrerapen hauek zeruko mekanikaren —Celestial Mechanics— hastapenen parte izan dira, astroen higidurak aztertzen dituen astronomiaren adarra dena.

Gaur egun, zeruko mekanikan ikerketak egin eta aurrerapenak lortzen jarraitzen badira ere, gure unibertsoari buruzko jakituria handiagoa da. Sistema itxi baten gorputzen arteko interakzio eta mugimenduak kalkulatzeko hainbat algoritmo berri sortu dira, horietako hainbat txosten honetan landuko direnak, eta gehienak, ordenagailu batean exekutatzeko diseinatuak eta optimizatuak daudenak.

[1] Lan honetan zehar, gorputz, astro eta partikula hitzak lantzen diren unibertsoko objektu masiboak —planetak eta izarrak, besteak beste— aipatzeko erabiliko dira.

Lan honetan, zeruko mekanikaren bi problema nagusi azaldu eta ebatziko dira:

1. *Keplerren problema*: hasierako uneko posizio eta abiadurak jakinda, bi gorputzen masek sortzen dituzten indar grabitazionalek denboran zehar gorputz horiei eragiten dien mugimendu eta posizio berriak kalkulatzean datza, hots, orbita kepleriarren kalkulua burutzea.

2. *N-gorputzen problema*: Keplerren problemaren generalizazioa, non gorputz guztien arteko interakzioak kontuan izan behar diren.

Programazio-lengoaia berri bat: Julia

Gaur egun, hainbat motako lengoaiak aurki daitezke: objektuei orientatutakoak, modularrak, erregeletan oinarritutakoak edota zenbakizko konputazioak egiteko diseinatuak daudenak, besteak beste. Zeruko mekaniketan egiten diren kalkuluentzat, azkenekoetan jarriko da arreta. Izan ere, programazio-lengoaia batek algoritmoaren exekuzioaren azkartasuna baldintzatzen du.

C, programazio-lengoaia estatikoa, bere eraginkortasunarengatik ezaguna eta erabilia da, maila baxuko lengoaia bat bada ere. Hasiera batean UNIX sistema-eragilea eraikitzeko diseinatu eta erabili bazen ere, bere lekua ezarri du programatzaileen tresna erabilgarri bezala, bai aplikazioak sortzeko garaian eta baita zientziarako konputazioan erabiltzeko ere. Hala ere, maila baxuko lengoaia bat izanik, ez dago programatzaile hasiberrientzat zuzenduta. Izan ere, C-ren kodea ez da oso irakurterraza —maila altuagoko lengoaiekin konparatzen bada behintzat— eta beste pertsona batek idatzitako kodea ulertzea oso zaila izatea gerta liteke.

Horregatik, Julia erabiltzea proposatzen da lan honetan. Julia zenbakizko konputaziorako diseinatuta dagoen errendimendu altuko programazio-lengoaia dinamikoa da, hots, JIT —Just In Time— konpiladoreak egiten dituen eragiketak exekuzio unean egiten ditu eraginkorki —estatikoaren aurkakoa—. Sintaxi aldetik, kodea ulergarriagoa da C-rena baino, eta maila altuagoko lengoaia bat izanik, kodea garatzea eta tratatzea azkarragoa bihurtzen da.

Normalean, programazio-lengoaien errendimendua konparatzeko orduan, C lengoaia erabiltzen da erreferentzi bezala, sortzen duen kode eraginkorra dela eta. Hala ere, Julia ondo defendatzen da alor honetan.

1.1 irudian ikusten denez, Matlab, Python, R eta Octave lengoaiek matrizeen arteko eragiketak azkar egiten dituzte, gehienbat bektore/matrizeen arteko eragiketak egiteko disei-

1.1 Irudia: Zientziarako konputaziorako erabili daitezkeen programazio-lengoaien exekuzio-denborak konparatzen dira, lengoaia bakoitzeko errendimendu erlatiboa aztertzen delarik —C-rekiko— hainbat ataza ezberdinetan. Ardatz horizontalean, aztertu diren 12 lengoaiak agertzen dira, beraien artean C eta Julia daudelarik. Ardatz bertikalean, berriz, ataza bakoitza egiteko behar izan den exekuzio-denbora erlatiboak agertzen dira eskala logaritmikoan, C-k dituen exekuzio-denborekiko. Ataza/funtzio bakoitza lengoaia bakoitzean nola definituta dagoen ikusteko, bisitatu Juliaren web orri ofiziala.

natuak daude eta. Hala ere, errekurtsioak egiterako orduan, adibidez, kosta egiten zaie C-ren errendimendua mantentzea. Hau ez da Julia-ren kasua, exekuzio-denbora baxuak mantentzen ditu eta aztertutako ataza guztietan.

Esan bezala, zientzietara aplikatutako konputazioan, aukeratutako lengoaiak kostu handiko kalkulu eta algoritmoetan dakarren eraginkortasuna baliagarria da. Lan honen kasuan, Keplerren eta N-gorputzen problemei aplikatutako zenbakizko metodoak erabiliko direnez, gomendagarria da programazio-lengoaian aukera on bat egitea. Hori dela eta, Julia proposatzen da C-ren alternatiba bezala.

1.1 Helburuak

Proiektuaren izenburuak lanaren helburu nagusia zein den definitzen badu ere, proiektuan zehar beste hainbat helburu definitu dira:

- Julia lengoaiako pakete bat sortu Keplerren ekuazioa ebazteko, hots, Keplerren fluxua kalkulatzeko.
- Keplerren fluxuaren algoritmoa ahal den heinean optimizatzea, bai exekuzio-denborak eta baita lortutako emaitzen doitasunak ere.
- Keplerren fluxuaren kalkulua, bai C-n eta baita Julian ere inplementatzea, bi lengoaien arteko azterketa egin ahal izateko.
- Paketearen aplikazio gisa, N-gorputzeko problema grabitazionalen zenbakizko ebazpena burutzea.
- N-gorputzeko sistema planetarioen kalkulua egingo denez, Eguzki-sistemaren simulazio interaktibo bat egitea lortutako datuekin —kalkulatutako orbitak modu txukun batean bistaratu ahal izateko—.

1.2 Lanerako plana

Proiektua 2017ko irailatik 2018ko irailera bitartean burutu da, ia lan guztia 2018ko apirilatik aurrera egin bada ere, ikasturteko klaseen lan-karga murriztean. Lau atal nagusitan banatu daiteke proiektuaren garapena:

1. Hainbat argitalpen tekniko eta liburuen irakurketa, gaiari buruzko oinarrizko printzipio eta algoritmoak barneratzeko. Gainera, Julia lengoaiari buruzko hainbat dokumentazio eta bideo jarraitu dira, bai lengoaia bera ikasteko eta baita kodea optimizatzeko ere. Atal hau proiektu osoan zehar burutu da.
2. Kodearen inplementazioa, non problema bakoitzerako zenbait aldaera aztertu diren, lortzen diren exekuzio-denbora eta emaitzak konparatuz. Honetarako, bi programazio lengoaia erabili dira: C eta Julia. C-ko oinarrizko kodea 2018ko apirilerako burutu zen eta data horretatik aurrera kodea ez da asko aldatu. Julia-ko kodea, aldiz, 2018ko apiriletik aurrera landu da.
3. Eguzki-sistemaren simulazioa, 3D grafikoak errenderizatzeko eta JavaScript bidez inplementatutako API bat erabilz, ThreeJS liburutegia hain zuzen ere. Horrela, nabigatzaile batean lokalki exekutatzeko aukera ematen du liburutegi honek. Ataza hau 2018ko maiatzean garatu zen osorik, lan-karga txikiena duen atala baita.

4. Dokumentazioaren idazketa proiektuan zehar burutzen joan da, gehienbat 2018ko ekaina eta abuztua artean.

Hasiera batean bilera gutxi egin baziren ere, apriletik aurrera email bidezko komunikazioa eraiki da zuzendaria eta ikaslearen artean proiektuaren garapenaz hitz egiteko; eta behar izan denean —ia astero—, zuzendariaren bulegoan bilerak egin dira.

1.3 Edukiak

Memoria honetan zehar, honako edukiak azalduko dira:

- Bigarren kapituluan, Keplerren problema zer den eta nola ebatziko den argituko da. Irakurleak zeruko mekanikaren gaiak ez baditu ulertzen, A. eranskina irakurtzea gomendatzen zaio —Oinarrizko kontzeptuak—.
- Hirugarren kapituluan, sistema planetarioen N-gorputzen problema sakonduko da, eta ebazteko erabili den algoritmoa azalduko da.
- Laugarren kapituluan, Eguzki-sistemaren simulazioa nola egin den esplikatuko da, erabili diren teknika eta liburutegiak azalduz.
- Bosgarren kapituluan, bigarren eta hirugarren kapituluan aipatu diren algoritmoetan lortutako emaitzak aztertuko dira, bai denborak eta baita lortutako erroreak ere.
- Azkenik, seigarren kapituluan, emaitzetatik lortzen diren ondorioei buruz hitz egingo da, eta egindako hautaketen zergatiak defendatuko dira.

2. KAPITULUA

Keplerren problema

2.1 Azalpena

Johannes Keplerrek planeten higidurak deskribatzen dituzten legeak lehenengo aldiz formulatu zituenetik, zientzialariak planeten higiduren ekuazioak ebazten saiatu dira. Bere ohoretan, problema honi Keplerren problema deitu zaio.

Zeruko mekanikan, Kepler-en problema bi gorputzen problemaren kasu berezi bat da, non bi gorputzen interakzioa aztertzen den F indar zentral batekiko. Teorian, F indarra erakarlea edo alderatze-indarra[1] izan badaiteke ere, lan honetan zehar indar grabitatorioak aztertzen direnez, aipatzen diren indar guztiek orbitatzen ari diren gorputzak erakarriko dituzte.

Problema honetan, helburua gorputzaren hasierako $\mathbf{r}_0$ posizio eta $\mathbf{v}_0$ abiadura bektoreak izanik, dt denbora-pauso[2] jakin bat igaro ondoren, gorputz horrek izango dituen $\mathbf{r}$ posizio eta $\mathbf{v}$ abiadura bektore berriak kalkulatzea da, 2.1 ekuazioak definitzen duen Hamiltondarraren bidez zehazten diren higidurak jarraituz —non μ bi gorputzen parametro grabitazional estandarren batura den—.

$$\mathcal{H}_{\text{Kepler}} = \frac{1}{2}\mathbf{v}^2 - \frac{\mu}{|\mathbf{r}|} \tag{2.1}$$

[1] Coloumb-en legeak dioenez, adibidez, bi gorputzen artean sortzen diren indarrak alderapen-indarrak izango dira zeinu bereko karga elektrikoak badira.

[2] Denbora-pauso hau negatiboa ala positiboa izan daiteke, gorputzaren posizioa iraganean edota etorkizunean zein den aztertu ahal daitekelarik.

Beraz, bi gorputzen arteko elkar-indarrek definitutako sistema bat ebatziko da, sinplifikazio hauek kontuan hartzen dituena:

1. Sistemako gorputzak esfera perfektuak dira, eta masa-puntuak bezala tratatu daitezke.

2. Ez dago beste kanpo edo barne indarrik eragiten sistema horretan, hots, gorputzen artean sortzen diren indarrak bakarrik kontuan hartuko dira.

2.2 Ebazpena

Problema hau ebazteko bi era ezberdin aztertu dira[3], hasiera batean $\mathbf{r}_0$ eta $\mathbf{v}_0$ bektoreak eskura daudela kontuan izanik.

1. Elementu orbitalak hasierako $\mathbf{r}_0$ eta $\mathbf{v}_0$ bektoreetatik lor daitezke, orbitaren tamaina, forma eta orientazioa definitzen dituztenak —partikularen posizioa orbita horretan zehaztuz ere bai—. Elementu orbitalen balioak dt denbora-pauso bat igaro ondoren izango dituzten balioak erraz kalkulatu daitezke M batezbesteko anomalia eguneratuz eta Keplerren ekuazio tradizionala —A.5 Ek.— ebatziz, A. eranskinean azaldu den bezala. Hala ere, elementu orbital berrietatik $\mathbf{r}$ eta $\mathbf{v}$ bektore berriak lortzeak kostu konputazional handia dakar, batez ere behin eta berriro aplikatu behar denean —hots, sistema hainbat denbora-pausoetan zehar garatu nahi denean—.

2. Hasiera batean problema elementu orbitalak erabiliz kalkulatzen bazen ere, ez da gorputzen higidurak ebazteko era bakarra, ordenagailu bidez kalkulatzeko azkarragoak existitzen dira eta. Funtsean, atal honetan landuko den metodo honekin, ez dira elementu orbitalik kalkulatzen, hau da, $\mathbf{r}$ eta $\mathbf{v}$ bektore berriak $\mathbf{r}_0$ eta $\mathbf{v}_0$ bektoreen konbinazio bidez konputatzen dira —2.2a eta 2.2b Ek.—, Gaussen f eta g koefizienteak erabiliz —aurrerago definituko direnak—.

$$\mathbf{r} = f\mathbf{r}_0 + g\mathbf{v}_0 \tag{2.2a}$$

$$\mathbf{v} = \dot{f}\mathbf{r}_0 + \dot{g}\mathbf{v}_0 \tag{2.2b}$$

[3] Bi era hauek ez dira bakarrak, ekuazio transzendentalik gabeko ebazpenak [Pimienta-Peñalver and Crassidis, 2013] garatu dira eta; baina azkenean aukera hauek ez dira kontuan hartu.

> Koefiziente hauek —eta baita $\dot{f}$ eta $\dot{g}$ koefizienteak ere, f eta g-ren deribatuak direnak denbora-pausoarekiko— elementu orbitalen edo astroaren hasierako $\mathbf{r}_0$ eta $\mathbf{v}_0$ bektoreen, eta dt denbora-pausoaren arabera aldatzen dira. Hala ere, elementu orbitalak ez dira esplizituki kalkulatzen prozesu osoan zehar.

Hori dela eta, $\mathbf{r}$ eta $\mathbf{v}$ bektoreak ahal den era eraginkorrenean kalkulatu nahi direnez, Gaussen f eta g koefizienteak erabiliko dira problema honen ebazpenerako, hots, bigarren metodoa erabiliko da. Hala ere, algoritmoari aldaketa batzuk egitea proposatzen da kapitulu honetan, algoritmoaren bai zehaztasun eta baita azkartasuna ere areagotzeko asmoarekin. Hemendik aurrera, txosten honetan zehar, algoritmo honi Keplerren fluxua deituko zaio, Keplerren legeak jarraituz gorputz hauen higiduren eboluzioa aztertzeko erabiltzen delako.

Keplerren fluxu hau erabiltzeko azkartasuna ez da arrazoi bakarra. Izan ere, A. eranskinean azaldutako ekuazio gehienak —batez ere, A.3 eta A.4 sekzioetan— orbita eliptikoentzat pentsatuta daude, $e < 1$ eszentrikotasuna duten orbitentzat alegia. Hortaz, algoritmo berdina edozein eszentrikotasun edota orbita motarako bideragarria izatea komenigarria izango litzateke. Hauxe bera aldagai unibertsalen formulazioa[4] erabiliz lor daiteke, eta, ikuspuntu hau erabiliz, zenbakizko egonkortasuna eta algoritmoaren azkartasuna areagotu daiteke.

2.2.1 Algoritmoa

Keplerren fluxua lau zati nagusitan banatu daiteke [Curtis, 2010].

1. Lehenik eta behin, hasierako $\mathbf{r}_0$ eta $\mathbf{v}_0$ bektoreetatik hiru balio ezberdin kalkulatu behar dira: β, η eta ζ.

$$\beta = \frac{2\mu}{|\mathbf{r}_0|} - |\mathbf{v}_0|^2 \tag{2.3}$$

$$\eta = \mathbf{r}_0 \cdot \mathbf{v}_0 \tag{2.4}$$

[4] Zeruko mekanikan, *aldagai unibertsalen formulazioa* Keplerren problema ebazteko erabiltzen den metodo bat da. Keplerren ekuazioaren orokortze bat da, orbita eliptiko, paraboliko eta hiperbolikoentzat erabili daitekeena. Beraz, unibertsoan aurkitzen diren egoera askotarako aplikagarria da —non eszentrikotasun ezberdineko orbitak aurki daitezkeen—.

$$\zeta = \mu - \beta|\mathbf{r}_0| \tag{2.5}$$

Kontuan izanik μ bi gorputzen parametro grabitazional estandarren batura dela, β balioak —2.3 ekuazioan definituta dagoena— a ardatz-erdi nagusiarekin erlazioa du, $a = \frac{\mu}{\beta}$ betetzen da eta. Beste hitz batzuetan, μ definizioz positiboa denez, β-ren arabera orbita ea eliptikoa, parabolikoa ala hiperbolikoa den jakin daiteke.

- $\beta > 0$ denean, a ardatz-erdi nagusia positiboa izango da eta orbita, hortaz, eliptikoa.
- $\beta = 0$ denean, definizioz $a = 0$ eta orbita parabolikoa izango da.
- Azkenik, $\beta < 0$ denean, ardatz-erdi nagusia negatiboa izango da, orbita hiperbolikoa izanik.

η —2.4 Ek.— hasierako posizio eta abiadura bektoreen arteko biderketa eskalarra da. ζ-rekin —2.5 Ek.— batera, hiru balioak aldez aurretik kalkulatzea gomendatzen da, algoritmoan zehar behin eta berriro erabili behar dira eta.

2. Ondoren, Keplerren ekuazioa ebatzi behar da. Hori bai, formulazio unibertsala erabiliko denez, 2.6 ekuazioak begi-bistan ez du Keplerren ekuazio tradizionalarekin —A.5 Ek.— antzekotasun handirik, baliokideak badira ere [Mikkola and Innanen, 1999].

$$|\mathbf{r}_0|\chi + \eta G_2(\beta,\chi) + \zeta G_3(\beta,\chi) - dt = 0 \tag{2.6}$$

Formulazio unibertsala erabiltzeak, T benetako anomalia kalkulatzeko M batezbesteko anomalia erabili beharrean, χ anomalia unibertsala zuzenean kalkulatzen dela esan nahi du —gorputzaren kokapena orbitan zehar ezartzeko— [Fukushima, 1999].

C eta G-funtzioak

2.6 ekuazioa Stiefelen G-funtzioen bidez eraikitzen da, eta G-funtzio hauek C-funtzioen menpekoak dira —2.7 Ek.—. Bi funtzio hauek azaltzeko asmotan, parentesi txiki bat egingo da.

$$G_n(\beta,\chi) = \chi^n C_n(\beta\chi^2) \tag{2.7}$$

Zeruko mekanikan, Stumpffen C-funtzioak [Mikkola and Innanen, 1999] aldagai unibertsalen formulazioa erabiliz orbita planetarioak aztertzeko erabiltzen dira —2.8

ekuazioan definitzen direlarik—. Stumpffen funtzioetan, $n \in \mathbb{N}$ betetzen da eta hauek $z \in \mathbb{R}$ multzo osorako konbergitzen dute, $z = \beta\chi^2$ izanik.

$$C_n(z) = \frac{1}{n!} - \frac{z}{(n+2)!} + \frac{z^2}{(n+4)!} \ldots = \sum_{i=0}^{\infty} \frac{(-z)^i}{(n+2i)!} \tag{2.8}$$

Alde batetik, lehenengo bi C-funtzioak sinu eta kosinu funtzioen Tayloren serieen hedapenekin konparatzen badira, erlazio batzuk aurki daitezke $z > 0$ kasuentzako —2.9a eta 2.9b Ek.—, hots, orbita eliptikoentzako.

$$C_0(z) = \cos\sqrt{z} \tag{2.9a}$$

$$C_1(z) = \frac{\sin\sqrt{z}}{\sqrt{z}} \tag{2.9b}$$

Beste aldetik, orbita paraboliko eta hiperbolikoetan —$z \leq 0$ den kasuetan—, sinu eta kosinu hiperbolikoen funtzioei erreparatuz antzeko emaitzak lortzen dira —2.10a eta 2.10b Ek.—. Ikus daitekeenez, C-funtzioak sinu eta kosinu funtzioen generalizazioak dira.

$$C_0(z) = \cosh\sqrt{-z} \tag{2.10a}$$

$$C_1(z) = \frac{\sinh\sqrt{-z}}{\sqrt{-z}} \tag{2.10b}$$

C-funtzioek funtzio errekurtsibo bat betetzen dute —2.11 Ek.—, eta 2.9 eta 2.10 ekuazio multzoetan definitutako ekuazioak errekurtsioaren kasu nabari bezala hartuz, 2.8 ekuazioan definitutako batukaria[5] behin eta berriro kalkulatzea ekidin daiteke, konputazionalki oso pisutsua baita.

$$C_n(z) = \frac{1}{n!} - zC_{n+2}(z) \tag{2.11}$$

Keplerren fluxuan lehenengo lau G-funtzioak kalkulatu behar dira bakarrik, bai Keplerren ekuazio unibertsala —2.6 Ek.— definitzeko eta baita Gaussen f eta g koefizienteak kalkulatzeko ere. Keplerren ekuazio unibertsala transzendentala da, hots,

[5]Berez, batukari honek beti konbergitzen duen serie infinitu bat definitzen du. Hortaz, serie horren gai kopuru finitu bat batu behar da eskatzen den zehaztasuna lortzeko; hots, zehaztasuna handitu nahi bada, gai kopuru hori handitu beharko da.

ezin da aljebraikoki ebatzi χ anomalia unibertsalaren balioa kalkulatzeko. Horretarako, Newton-Raphsonen zenbakizko metodoa erabiliko da, A. eranskineko A.5 atalean azaltzen dena.

3. Behin χ-ren hurbilpen on bat izatean, Gaussen f eta g koefizienteak kalkulatuko dira, bere dt denborarekiko deribatuekin batera —2.12 Ek.—.

$$f = 1 - \frac{\mu G_2(\beta,\chi)}{|\mathbf{r}_0|} \tag{2.12a}$$

$$g = dt - \mu G_3(\beta,\chi) \tag{2.12b}$$

$$\dot{f} = -\frac{\mu G_1(\beta,\chi)}{|\mathbf{r}_0| r} \tag{2.12c}$$

$$\dot{g} = 1 - \frac{\mu G_2(\beta,\chi)}{r} \tag{2.12d}$$

2.12c eta 2.12d ekuazioetan, $r = \mathbf{r}_0 + \eta G_1(\beta,\chi) + \zeta G_2(\beta,\chi)$ aldagaiak dt denbora igaro ondoren orbitaren grabitazio-zentrotik astroaren posizio berrira dagoen distantzia euklidearra adierazten du. Hortaz, $\mathbf{r}$ eta $\mathbf{v}$ bektoreak posizio eta abiadura bektore berriak badira, $r = |\mathbf{r}|$ izango litzateke.

4. Azkenik, $\mathbf{r}$ eta $\mathbf{v}$ bektore berriak kalkulatzea gelditzen da, 2.2a eta 2.2b ekuazioetan definitzen den bezala.

2.2.2 Hobekuntzak

Lan honetan zehar, helburuetako bat N-gorputzeko sistema planetario bat simulatzea da. Simulazio horretan, Keplerren fluxua etengabe exekutatu beharko da, eta algoritmoa ahal den gehien fintzea komeni da.

Keplerren fluxua lau zatitan banatu da 2.2.1 atalean. Lau zati horietatik, bigarren eta hirugarrenak dauzkate mami gehien, eraginkortasunaren ikuspuntu aldetik behintzat. Hortaz, atal horietan egin dira algoritmoaren hobekuntzak.

- Bigarren zatian, Newton-Raphsonen metodo iteratiboa exekutatzen da, eta metodo honek ondo funtzionatzeko, hasierako estimazio on bat jaso behar du, χ_0 deituko dena. Gainera, gelditze-irizpide egoki bat aukeratu behar da, beharrezkoak diren iterazioak bakarrik exekutatzeko.

- Hirugarren zatian, Gaussen f eta g koefizienteak kalkulatzen dira, baina hauen kalkulua egitean biribiltze-erroreak gerta daitezke, ondoren azalduko den bezala. Horretarako, hasierako koefizienteak ordezkatu dituzten $\hat{f}$ eta $\hat{g}$ koefiziente berri batzuk definitzen dira, eta Kahanen batuketa konpentsatua erabiltzea proposatzen da.

Zenbakizko metodoaren hasierako estimazioa

Aipatzekoa da Newton-Raphsonen metodoaren aukeraketaren zergatia. Irakurritako hainbat artikuluetan [Mikkola and Innanen, 1999] [Rein and Tamayo, 2015], metodo iteratibo hau aipatu eta erabiltzen dute, probatutako metodorik azkarrena dela esaten baitute. Izan ere, hauen ustetan, funtzioa eta deribatuaren balioak azkar kalkulatu daitezkeenean, orden handiagoko metodoekin iterazio bakoitzean lortzen diren zehaztasun handiagoak ez dira nahikoak hauen kostu-konputazionalaren handitzearekin konparatuz. Datu hau kontuan harturik, beste zenbakizko metodoekin probak egitea alde batera utzi da eta zuzenean Newton-Raphsonen metodoarekin lan egitea aukeratu da, orden txikiagoko metodoak ere alde batera utziz.

Orbita eliptiko, paraboliko eta hiperbolikoen ezaugarriak ezberdinak izan arren, formulazio unibertsala erabiliz hiru motatako orbiten higidurak definitu daitezke aldagai eta formula berak erabiliz. Beraz, edozein hasierako $\mathbf{r}_0$ eta $\mathbf{v}_0$ bektoreak eta dt denbora-pauso bat emanik, algoritmoa kapaza izan beharko litzateke $\mathbf{r}$ eta $\mathbf{v}$ bektoreak itzultzeko. Horregatik, zenbakizko metodoari hasierako estimazio egoki bat ematea beharrezkoa da, eszentrikotasun eta denbora-pauso ezberdinekin estimazio zuzenak emango dituenak.

Aipatu denez, hasierako estimazioan χ anomalia unibertsala estimatu behar da. χ aldagaia $L^{1/2}$ dimentsioa du, non L ardatz-erdi nagusia neurtzeko erabiltzen den unitatea den. Aldagai hau Sundmanen transformazioaren[6] [Tokis, 2014] [Fernandes, 2003] bidez definitua dago —2.13 Ek.—.

$$\sqrt{\mu}\frac{dt}{d\chi} = |\mathbf{r}| \tag{2.13}$$

Orbita koniko motaren arabera, anomalia eszentrikoa era ezberdin batean definituta dago. A. eranskinean, anomalia eszentrikoa azaltzean, E sinboloarekin izendatu da; baina anomalia eszentriko hori orbita eliptikoentzat bakarrik definitu da eranskin horretan zehar. Orbita parabolikoen anomalia eszentrikoa D eta hiperbolikoena F bezala izendatzen badi-

[6] *Sundmanen transformazioa*: χ aldagai unibertsalak r, dt eta mu aldagaiekin duen erlazioa ezartzen duen ekuazio diferentziala da, orbita koniko mota guztien anomalia eszentriko ezberdinen generalizazioa delarik.

ra, anomalia eszentriko bakoitzak χ-rekin duen erlazioa 2.14 ekuazio multzoan definitzen da.

$$E = \chi\sqrt{\frac{\beta}{\mu}}, \quad \beta > 0 \tag{2.14a}$$

$$D = \chi, \quad \beta = 0 \tag{2.14b}$$

$$F = \chi\sqrt{-\frac{\beta}{\mu}}, \quad \beta < 0 \tag{2.14c}$$

χ beste era honetan definitu daiteke ere bai:

$$\chi = \int_{t_0}^{t_0+dt} \frac{dt'}{r} = dt\ \hat{r}^{-1} \tag{2.15}$$

Non t_0 hasierako unea den eta $\hat{r}$ $[t_0, t_0+dt]$ denbora-tartean orbitatzen ari den grabitazio zentroarekiko batezbesteko distantzia den. $\hat{r}^{-1}$-ren balioa ezin da zehatz-mehatz jakin. Hala ere, eszentrikotasun edota denbora-pauso txikientzako $\chi \approx \frac{dt}{|\mathbf{r}_0|}$ nahikoa izango litzateke, $|\mathbf{r}_0|$ ez baita $\hat{r}$-ren oso ezberdina izango. Hori bai, e eta dt handiagoentzako estimazioa asko okertu daiteke, orbitaren perihelio aldean batez ere. Horregatik, hori konpontzeko [Rein and Tamayo, 2015] artikuluan proposatutako estimazioa erabiliko da —2.16 Ek.—, η aldagaiaz baliatzen delarik perihelio inguruko errore handia txikitu ahal izateko.

$$\chi_0 = \frac{dt}{|\mathbf{r}_0|}\left(1 - \frac{\eta}{2}\right) \tag{2.16}$$

Egindako probetan, estimazio honek ondo funtzionatzen du bai eszentrikotasun handiko orbitetan, hots, orbita hiperbolikoetan, eta baita dt denbora pauso handietan ere, astroaren orbitaren periodoa dt bezala erabiliz azkar konbergitzen duelarik—. Hortaz, hasierako estimazio egoki bat dela erabaki da.

Newton-Raphsonen gelditze-irizpidea

Behin hasierako estimazioa definituta dagoenean, Newton-Raphsonen metodoa habian jar daiteke. Hala ere, metodo hau noiz gelditu behar den jakiteko irizpide bat behar da. Lan honetan zehar, hiru gelditze-irizpide ezberdin landu dira. Hiru irizpideetan, χ aldagaiaren datu-motaren zehaztasuna kontuan hartu da, hots, ez da ε tolerantzia konstante bat definitu datu-mota ezberdin guztientzako.

1. *Balioen konparaketa*: Irizpide honetan azkeneko hiru iterazioetan χ-ren balioa errepikatzen den aztertzen da, hau da, balio bera errepikatzen duen bukle batean sartu den ala ez begiratzen da. Teorikoki, Newton-Raphsonen metodoa eskuz ebatziko balitz, irizpide honek ez luke zentzurik izango, iterazio bakoitzean emaitzen zehaztasuna handitzen joango litzakeelako. Izan ere, konparaketa hau koma higikorreko doitasunarekin egiten dela kontuan izan behar da, eta, hortaz, datu motak duen doitasuna hobetu ezin denean amaituko du algoritmoak.

```
χ_aur2 ← NaN
χ_aur1 ← NaN
χ ← χ_0
while χ ≠ χ_aur1 and χ ≠ χ_aur2 loop
    χ_aur2 ← χ_aur1
    χ_aur1 ← χ
    δ berria kalkulatu          ♢ i. iterazioan, δ = χ_{i+1} − χ_i betetzen da
    χ ← χ − δ                   ♢ χ-ren eguneraketa
end loop
return χ
```

Adibidez, demagun χ-ren balio zehatza $1.495743 \cdot 10^{-10}$ dela. Zenbakizko metodoaren bosgarren iterazioa exekutatu ondoren $\chi_5 = 1.496538 \cdot 10^{-10}$ lortzen da. Seigarren iterazioarekin jarraituz $\chi_6 = 1.495742 \cdot 10^{-10}$ dela antzeman da eta zazpigarren iterazioa exekutatuz $\chi_7 = 1.498745 \cdot 10^{-10}$ balioa lortu da. Ikus daitekeenez, adibidean erabiltzen den koma higikorreko datu motak χ-ren balio zehatza ezin du zehaztu, eta, horregatik, biribiltze-errore bat sortu da. Iterazioa bi balioko begizta batean sartu dela antzeman daiteke zortzigarren iterazioan, non $\chi_8 = 0.1495742$ lortu den. $\chi_6 = \chi_8$ denez, gelditze-irizpidea bete da eta metodo iteratiboa amaitzen da, $\chi = \chi_8$ dela erabakiz.

Kasu honetan, zortzigarren iterazioan bukatu du algoritmoak $\chi = 1.495742 \cdot 10^{-10}$ balioarekin eta balio errealarekiko 10^{-16}-eko errore absolutuarekin. Begiztarik gertatuko ez balitz, hots, $\chi_6 = \chi_7 = \chi_8 = 1.495742 \cdot 10^{-10}$ izango balitz, iterazio bat lehenago bukatuko luke algoritmoak. Hori bai, hiru edo gehiago balio ezberdineko begizta batean sartu ezkero, irizpideak ez luke begiztarik antzemango eta ez litzateke begizta horretatik inoiz aterako.

Badaezpada ere, kodean 50 iterazioko maximo bat ezarri da metodo iteratiboan, konbergitzen ez duen edota irizpideak betetzen ez diren kasuetan bukaera bat ezarri

ahal izateko.

2. *Erroreen konparaketa*: Irizpide honetan, χ_i berriak aurreko iterazioetako χ_{i-1} eta χ_{i-2} berriekin konparatu beharrean, horien artean sortzen diren aldaketak konparatzen dira. Funtzio jarrai batean, Newton-Raphson-en metodoak hasierako estimazio zuzen bat emanda beti konbergitu beharko luke, iterazio bakoitzeko χ_i estimazio berriek errore txikiagoa izanik aurreko χ_{i-1} iterazioen estimazioekiko. $\delta_i = \chi_i - \chi_{i-1}$ bezala definitzen bada, iterazioak exekutatu ahala 2.17 ekuazioko hau bete beharko litzateke.

$$\lim_{i\to\infty} \delta_i = 0 \tag{2.17}$$

Hori bai, koma higikorreko aritmetikarekin lan egiten denez, i iterazio gutxirekin nahikoa izan beharko litzateke $\delta_i \approx 0$ izateko. Hortaz, gelditze-irizpide honetan, $\delta_i = 0$ edota $|\delta_i| \geq E_{max}$ betetzean gelditzuko da metodo iteratiboa, $E_{max} = \max(\delta_i, \delta_{i-1})$ izanik. Beste hitzetan, irizpidea χ bi iterazioetan zehar ez aldatzean edota δ_i ez txikitzean betetzen da.

$E_{aur} \leftarrow NaN$
$E \leftarrow NaN$
$\chi \leftarrow \chi_0$
while true **loop** ♢ Gehienez 50 iterazio
    $E_{max} \leftarrow \max(E, E_{aur})$
    $E_{aur} \leftarrow E$
    δ berria kalkulatu
    $\chi \leftarrow \chi - \delta$
    $E \leftarrow |\delta|$
    if $E = 0.0$ or $E \geq E_{max}$ **then**
        return χ
    end if
end loop
return χ

Lehenengo irizpidea azaltzeko erabili den adibidea berrerabiliz —2.1 taulan kalkulatutako datu guztiak laburbiltzen dira—, bosgarren, seigarren eta zazpigarren iterazioetan $\chi_5 = 1.496538 \cdot 10^{-10}$, $\chi_6 = 1.495742 \cdot 10^{-10}$ eta $\chi_7 = 1.495745 \cdot 10^{-10}$ balioak lortzen dira. Suposatuz $\chi_4 = 2.569317 \cdot 10^{-6}$ dela, $\delta_5 = \chi_5 - \chi_4 = -2.569167 \cdot 10^{-6}$, $\delta_6 = -0.796 \cdot 10^{-14}$ eta $\delta_7 = 3 \cdot 10^{-16}$ dira. Hori horrela izanik, $E_{max} =$

$\max(|\delta_5|,|\delta_6|) = 2.569167 \cdot 10^{-6}$ denez, irizpidea ez da beteko zazpigarren iterazioan, $E = |\delta_7| < E_{max}$ baita.

Iter.	χ	δ	E_{max}
4	$2.569317 \cdot 10^{-6}$	————	————
5	$1.496538 \cdot 10^{-10}$	$-2.569167 \cdot 10^{-6}$	————
6	$1.495742 \cdot 10^{-10}$	$-0.796 \cdot 10^{-14}$	————
7	$1.495745 \cdot 10^{-10}$	$3 \cdot 10^{-16}$	$-2.569167 \cdot 10^{-6}$
8	$1.495742 \cdot 10^{-10}$	$-3 \cdot 10^{-16}$	$-0.796 \cdot 10^{-14}$
9	$1.495745 \cdot 10^{-10}$	$3 \cdot 10^{-16}$	$3 \cdot 10^{-16}$

2.1 Taula: Taula honetan, hiru irizpideak konparatzeko erabiltzen den adibidearen hainbat datu gordetzen dira. Datuak asmatuak direnez, δ eta E_{max} batzuk falta dira, baina adibiderako behar diren datuak behintzat hemen aurkitzen dira.

Zortzigarren iterazioan berdina gertatzen da, $\chi_8 = 1.495742 \cdot 10^{-10}$ eta $\delta_8 = -3 \cdot 10^{-16}$ izanik, $E = 3 \cdot 10^{-16} < E_{max} = 0.796 \cdot 10^{-14}$ baita. Bederatzigarrenean, berriz, $\chi_9 = 1.495745 \cdot 10^{-10}$, $\delta_9 = 3 \cdot 10^{-16}$, $E = \delta_9$ eta $E_{max} = E$ lortzen da, irizpidea betetzen delarik eta azkeneko emaitza $\chi = 1.495745 \cdot 10^{-10}$ izanik.

Biribiltze-errorea dela eta, χ-k izan beharko lukeen balio errealarekiko $2 \cdot 10^{-16}$-eko errore absolutua dauka. Lehenengo irizpidearekin konparatuz emaitzak okerragoak badira ere, kontuan izan behar da χ-ren bi estimazioko begizta duen adibide bat dela. Begizta hori hiru balio edo gehiagoren artekoa izango balitz, lehenengo irizpideak 50 iterazio exekutatuko lituzke, eta bigarrenak, berriz, 10 iterazio inguru bakarrik.

Hasiera batean, proiektuaren zuzendariak gomendatuta probatu zen bigarren irizpide hau, baina biek arazo bat dute. Bi irizpideetan beharrezkoak ez diren iterazioak konputatzen dira. Adibidean, lor daitekeen emaitzarik onena seigarren iteraziora iristen denerako eskuratzen da. Hala ere, iterazio horretan gelditzeko, ondorengo iterazioetan gertatzen dena aurreikusten duen irizpide bat erabili beharko litzateke, eta horretarako diseinatu da hirugarren irizpidea.

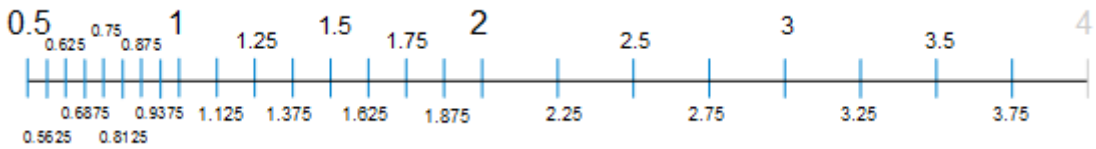
3. *Tolerantzia zuhurra*: Newton-Raphsonen metodoak orden koadratikoa du. δ-ren balioa iterazioz iterazio koadratikoki txikitzen dela jakinda, ε tolerantzia χ-k erabiltzen duen koma higikorreko aritmetikaren ε-*mach*-en[7] araberakoa bezala definitu da —2.18 Ek.—.

[7] *Makinaren epsilona*: koma higikorreko aritmetikan, $1 + \varepsilon$-*mach* > 1 kondizioa betetzen duen ε-*mach* aldagaiak izan dezakeen balio txikiena da, koma higikorreko aritmetika jakin bateko doitasun erlatiboa neurtzeko erabiltzen dena.

$$\varepsilon = \frac{\sqrt{\varepsilon\text{-}mach}}{100} \tag{2.18}$$

2.18 ekuazioan, ε-*mach*-erako doitasuna noiz izatera iritsiko den iragartzeko ε tolerantzia definitu da. Metodo iteratiboa koadratikoa denez, teorikoki, $\sqrt{\varepsilon\text{-}mach}$ balioak metodoak konbergitu aurretik izan beharko lukeen $|\delta|$-ren balioa izan beharko litzateke. Balio horri $/100$ egiten zaio konbergentzia hori kasu gehienetan ziurtatu ahal izateko. Beraz, $|\delta| < \varepsilon$ betetzen den iterazioan metodoak konbergitu egin beharko luke.

Makinaren epsilona azaltzean, doitasun erlatiboa zehazten duela adierazten da. Izan ere, koma higikorreko aritmetikan adierazi daitezkeen zenbaki guztiak ez daude linealki banatuta $\mathbb{R}$ multzoan. Horregatik, ε-*mach* baino txikiagoak diren distantziak aurkitu daitezke koma higikorreko aritmetikako bi zenbakien artean. Zenbaki horien distribuzioa bisualki aztertu daiteke 2.1 irudian.



2.1 Irudia: 4-bit-eko doitasuneko koma higikor bitarreko aritmetikan lor daitezkeen zenbakiak irudikatzen dira lerro urdin batzuen bidez, $\mathbb{R}$ multzoko lerroan zehar. Bi zenbakiren arteko tarteak nola zabaltzen diren ikus daiteke 0 baliotik urruntzen diren heinean. Izan ere, $\pm[2^i, 2^j]$ zehar tarte horiek berdinak izango dira koma higikor bitarreko aritmetikan, $i+1 = j$ eta $i, j \in \mathbb{Z}$ izanik; eta i unitate batez handitzen den heinean, distantzia horiek bikoizten joango dira. Irudi hau https://www.exploringbinary.com/ web orriko [Regan, 2015] artikulutik atera da, gai honetan gehiago sakontzen duelarik.

Gelditze irizpidea definituta izanik, metodo iteratiboaren sasi-kodea horrela gelditu ko litzateke:

```
χ ← χ0
while true loop                    ♢ Gehienez 50 iterazio
    δ berria kalkulatu
    χ ← χ − δ
    δ ← |δ|
    if δ < ε then
        return χ
    end if
end loop
return χ
```

Aurreko irizpideetan erabilitako adibidearekin jarraituz, adibideko koma higikorreko aritmetikaren ε-*mach* $= 10^{-14}$ dela suposatzen bada, $\varepsilon = \frac{\sqrt{\varepsilon\text{-}mach}}{100} = 10^{-9}$ beteko da, eta $|\delta| < 10^{-9}$ betetzen den iterazioan zenbakizko metodoa bukatuko da.

2.1 taulan begiratzen bada, $|\delta| < 10^{-9}$ kondizioa seigarren iterazioan betetzen da, eta, hortaz, aurreko irizpideak baino lehenago bukatzen du metodo iteratiboak. χ-ren balio errealarekiko 10^{-16}-ko errore absolutua lortzen da, koma higikorreko aritmetika erabiltzeak dakarrena. Hainbat iterazio aurrezten direnez, eta, oro har, emaitzetan desberdintasun nabarmenik antzematen ez direnez —ez adibide honetan ezta beste kasuetan ere—, hirugarren gelditze-irizpidea erabili da Keplerren fluxuaren inplementaziorako finalerako.

Gaussen koefizienteak

2.2 ekuazio multzoan azaldu den bezala, Gaussen f eta g koefizienteak, hasierako $\mathbf{r}_0$ posizio eta $\mathbf{v}_0$ abiadura bektoreen konbinazio lineal bidez, $\mathbf{r}$ posizio eta $\mathbf{v}$ abiadura bektore berriak lortu ahal izateko definitu dira. Koefiziente horiek 2.12 ekuazio-multzoan definitu dira, eta koma higikorreko aritmetikan ebatzi behar direnez, biribiltze-erroreekin arazo batzuk sor daitezkeela antzeman daiteke.

Adibidez, f kalkulatzean —2.12a Ek.— bi gai ezberdin aurki daitezke: 1 eta $\frac{\mu G_2(\beta,\chi)}{|\mathbf{r}_0|}$. Bigarren gai hori oso txikia izatera iritsi daiteke. Horregatik, bi gaien kendura egitean informazioa galdu daiteke. Hau bera g eta $\dot{g}$ kalkulatzean gertatzen da ere bai —2.12b eta 2.12d Ek.—; batez ere g-ren kasuan, dt denbora-pausoaren menpekoa delako informazioaren galera.

Hori dela eta, Gaussen f eta g koefiziente tradizionalak erabili beharrean, ia baliokideak

diren $\hat{f}$ eta $\hat{g}$ koefizienteak proposatzen dira —2.19 Ek.—.

$$\hat{f} = \frac{\mu G_2(\beta,\chi)}{|\mathbf{r}_0|} \tag{2.19a}$$

$$\hat{g} = |\mathbf{r}_0| G_1(\beta,\chi) + \eta G_2(\beta,\chi) \tag{2.19b}$$

$$\hat{\dot{f}} = -\frac{\mu G_1(\beta,\chi)}{|\mathbf{r}_0| r} \tag{2.19c}$$

$$\hat{\dot{g}} = \frac{\mu G_2(\beta,\chi)}{r} \tag{2.19d}$$

Koefiziente tradizionalekin konparatzen bada, $\hat{\dot{f}}$ aldatzen ez den koefiziente bakarra da. $\hat{g}$-ren kasuan, 2.6 ekuaziotik hasiz, 2.20 ekuazioko berdintza lor daiteke. $dt - \mu G_3(\beta,\chi)$ azkarrago konputatzen bada ere, $\hat{g}$ berria kalkulatzeko behar diren bi gaien balioen arteko diferentzia askoz txikiagoa da, sortzen den biribiltze-errorea txikituz.

$$dt - \mu G_3(\beta,\chi) = |\mathbf{r}_0| G_1(\beta,\chi) + \eta G_2(\beta,\chi) \tag{2.20}$$

Azkeneko bi koefizienteen kasuan, $1 - \hat{f} = f$ eta $1 - \hat{\dot{g}} = \dot{g}$ betetzen da. Hori dela eta, 2.2 ekuazio-multzoa koefiziente berriak erabiliz berridatzi behar da, emaitza 2.21 ekuazio-multzoa izanik.

$$\mathbf{r} = \hat{f}\mathbf{r}_0 + \hat{g}\mathbf{v}_0 + \mathbf{r}_0 \tag{2.21a}$$

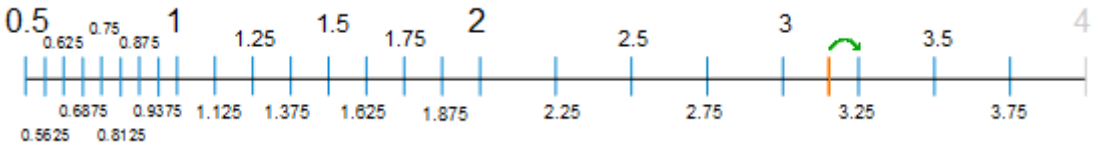
$$\mathbf{v} = \hat{\dot{f}}\mathbf{r}_0 + \hat{\dot{g}}\mathbf{v}_0 + \mathbf{v}_0 \tag{2.21b}$$

2.19 eta 2.21 ekuazio-multzoekin $\hat{f}$ eta $\hat{g}$ koefiziente berrien zehaztapenarekin bukatu da. Oro har, Gaussen koefizienteak oso balio txikiak dituztenez, 2.21a eta 2.21b ekuazioetan, lehendabizi lehenengo bi gaiak batzea komeni da, biribiltze-errorearen galera minimizatzen saiatzeko. Hala ere, bi ekuazio horietako azkeneko batuketan oraindik informazio garrantzitsua galdu daiteke. Horregatik, Kahanen batuketa konpentsatuaren algoritmoa erabiltzea proposatzen da.

Kahanen batuketa konpentsatua

Kapitulu honetan zehar, biribiltze-errorea hainbat alditan azaldu bada ere, nondik datorren ez da zehaztu —irakurleak zer den dakiela kontsideratzen da eta—. Hitz gutxitan,

biribiltze-errorea koma higikorreko aritmetika erabiltzearen ondorio bat da. Izan ere, koma higikorrean lerro errealeko zenbaki gutxi adierazi daitezke. 2.2 irudian π zenbakiarekin adibide bat azaltzen da.



2.2 Irudia: 4-bit-eko doitasuneko koma higikor bitarra erabiliz $\mathbb{R}$ multzoko zenbaki oso gutxi espezifikatu daitezke, eta π ez da horietako bat. Irudi honetan, 2.1 irudia oinarri bezala hartuta, π lerro laranja batez adierazten da. π ezin denez 4-bit-eko koma higikorreko aritmetikan adierazi, ahal den zenbaki hurbilenera hurbiltzen saiatzen da. Kasu honetan, $\pi \approx 3.25$ hurbilpena egiten du, $\xi = 3.25 - \pi \approx 0.1084$ biribiltze-errorea izanik. Izan ere, berdin du zenbat bit dauden eskura, π zenbaki irrazional —eta transzendentala— denez, ezin da koma higikorrean zehatz-mehatz zehaztu. Hori bai, erabilitako bit kopurua handitzean, lortzen den biribiltze-errorea txikitzen joango da. Adibidez, 64-bit erabiliz $\xi \approx 1.2246 \cdot 10^{-16}$ izango litzateke.

Biribiltze-errorearen informazio galerak hasiera batean garrantzi handirik ez duela ematen badu ere, Keplerren fluxua hainbat iterazioetan zehar exekutatzen bada, informazio-galera metatu eta orbita asko desbideratu daiteke. Kahanen batuketa konpentsatuak galtzen den ξ informazio hori gordetzeaz eta hurrengo iterazioetan berrerabiltzeaz arduratzen da. $a+b=c$ batuketa egin nahi dela kontsideratuz, algoritmoak honako itxura du:

INPUT: a, b eta ξ		
$y \leftarrow b - \xi$	$\diamondsuit$	Lehenengo aldian, $\xi = 0$
$t \leftarrow a + y$	$\diamondsuit$	Informazio galera sortzen da batuketan
$\xi \leftarrow (t-a) - y$	$\diamondsuit$	ξ informazio galera eguneratzen da, algebraikoki
$c \leftarrow t$		$\xi = 0$ bada ere
return c, ξ		

Algoritmoa hainbat alditan errepikatu ezkero, $\xi \neq 0$ den beste balio bat edukiko du. Funtsean, ξ aldagaian batuketa nagusian galtzen den hondarra zeinu negatiboarekin gordetzen da, hondar hori iterazio bakoitzean eguneratuz.

Adibide bezala hiru zenbaki batuko dira: $2 \cdot 10^7 + 1.61803 + 3.14159$, 8-bit-eko koma higikor hamartarreko aritmetika erabiliz. Emaitza erreala $2 \cdot 10^7 + 1.61803 + 3.14159 = 20000004.75962$ bada ere, Kahanen algoritmoa erabili gabe 20000004 lortuko litzateke. Kahanen batuketa konpentsatuaren algoritmoa erabili ezkero, 2.2 taulan azalduko lirateke exekuzioan zehar lortutako balioak.

	1. Iterazioa	2. Iterazioa
INPUT	$a = 2 \cdot 10^7$, $b = 1.61803$ $\xi = 0$	$a = 2.0000001 \cdot 10^7$, $b = 3.14159$ $\xi = -0.61803$
Eragiketak	$y \leftarrow 1.61803 - 0$ $t \leftarrow 2 \cdot 10^7 + 1.61803$ $\xi \leftarrow 1 - 1.61803$ $c \leftarrow 2.0000001 \cdot 10^7$	$y \leftarrow 3.14159 + 0.61803$ $t \leftarrow 2.0000001 \cdot 10^7 + 3.75962$ $\xi \leftarrow 3 - 3.75962$ $c \leftarrow 2.0000004 \cdot 10^7$
OUTPUT	$c = 2.0000001 \cdot 10^7$ $\xi = -0.61803$	$c = 2.0000004 \cdot 10^7$ $\xi = -0.75962$

2.2 Taula: Lehenengo iterazioan, sarrerako datuak a eta b batukariak eta $\xi = 0$ hondarra dira. Garrantzitsua da $a \gg b$ izan daitekeela kontuan hartzea, baita $b > a$ ezin dela bete ere. Aurrerago ikusiko denez algoritmoaren diseinuak $\xi \leq 0$ berresten duenez. y aldagaian $b + \xi$ batuketa gordetzen da eta batuketa honetan, berez, ez litzateke informaziorik galdu beharko. Eragiketen hirugarren lerroan $t = a + y$ egiten da, eta, orain bai, y aldagaiko hamartarrak galtzen dira. Hamartar horiek $\xi = (t - a) - y$ eginez berreskuratzen dira, ξ behar bezala eguneratuz. Izan ere, $t - a$ eginez, benetan batu den balioa berreskuratzen da, eta horri $y = b + \xi$ kenduz, $t = a + y$ batuketan eta ξ-rekin kontuan hartzen ez diren hondarrak batu eta lortzen dira.

Beraz, ξ aldagai laguntzailea erabiliz batuketetan galtzen diren hondarrak berreskuratu daitezke. Esan bezala, adibideko bi batuketekin algoritmoa erabiltzeak garrantzirik gabekoa ematen badu ere, batuketak behin eta berriz egin behar direnean desberdintasuna nabarmena da. Kahanen algoritmoa Keplerren fluxuan aplikatuko denez, dt denbora pausoa etengabe simulatu nahi izango da, algoritmoa ezin hobea izanik kasu horretarako.

Beraz, algoritmo hau 2.21 ekuazioan aplikatzeko $\mathbf{a} = \mathbf{r}_0$ eta $\mathbf{b} = \hat{f}\mathbf{r}_0 + \hat{g}\mathbf{v}_0$ izan behar dela kontuan eduki behar da, kasu honetan **a** eta **b** hiru elementuko bektoreak izanik eta $a \gg b$ betetzen delarik.

Azkenik, dokumentuaren hasieran hainbat datu-mota ezberdin erabiliko direla aipatu da. Lan honetan zehar, 64 bit-etik gorako koma higikor bitarreko aritmetikak erabili dira, doitasun altuko datu-motak hain zuzen ere. Hortaz, Kahanen algoritmoa datu-mota guztietarako erabiltzea ez litzateke guztiz komenigarria izango.

Esan bezala, Kahanen algoritmoak kostu konputazionala gutxi areagotzen badu ere, jada doitasun altua duten datu-motek ez dute algoritmo honen behar handirik, galtzen den informazioa oso txikia delako. Horregatik, Kahanen algoritmoa erabili ala ez erabakitzeko datu-mota bakoitzaren ε-*mach* erabiltzea deliberatu da. Eragiketetan doitasun altu bat mantendu nahi denez, ε-*mach* $\geq 10^{-30}$ betetzen duten datu-motak erabiltzen badira, Kahanen algoritmoa erabiliko da. Honek, IEEE-754 estandarra jarraitzen duten 100-bit baino gutxiagoko koma-higikor bitarreko datu-motetan, Kahan-en algoritmoa aplikatuko dela esan nahi du, 100 bit-eko datu-mota barne.

3. KAPITULUA

N-gorputzen problema

3.1 Azalpena

2. kapituluan tratatu den Kepler-en problema sinplifikazio bat da, mundu errealean aurkitu ezin dena. Izan ere, teorikoki, unibertsoko gorputz guztiek elkarri eragiten diote interakzio grabitazionalen bitartez. Interakzio edo **F** indar horiek askotan 0-ra hurbiltzen badira ere —detektagarriak ez badira ere—, gorputzen higidurak indar guzti hauen konbinazioen menpekoak dira.

Zeruko mekanikaren hastapenetatik, Eguzki-sistema eraikitzen duten astro nagusien higidurak aurreikustea izan da helburu nagusietako bat, helburu horrek perturbazio teoria bultzatu duelarik. Funtsean, perturbazio teoriak, problema sinpleago bateko soluzioetatik habiatuz, beste problema baten soluzioen hurbilpenak lortzeko metodo matematikoak batzen ditu matematikako arlo batean —problema bat hainbat zatitan banatuz, adibidez—.

Kasu honetan, Keplerren problematik habiatuz, N-gorputzen problema ebazteko saiakera egingo da. Izan ere, Eguzkia bere inguruan orbitatzen duten planetekiko hain masiboa izanik, planeta horiek beste planeten interakziorik gabe izango luketen orbita Kepleriarrak aztertzea; eta, ondoren, planeten arteko perturbazio txiki horiek orbita Kepleriarrei gehitzeko ikuspuntua ez da batere ideia txarra.

N-gorputzen problema ebazteko bete behar den $\mathcal{H}$ funtzio Hamiltondarra kartesiar koordenatuak erabiliz definitutako sistema planetarioaren energia zinetiko eta potentzialen batura bidez adierazten da —3.1 Ek.—.

$$\mathcal{H} = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} - \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_i m_j}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{3.1}$$

3.1 ekuazioan, sistemako i. gorputzak —$i \in [1,2,\ldots,N]$ izanik— honako ezaugarriak ditu: m_i masa, $\mathbf{r}_i$ posizioa[1] —koordenatu kartesiarretan eta sistemaren barizentroak[2] zentroa definitzen duelarik— eta $\mathbf{p}_i$ momentua. Astro bakoitzaren $\mathbf{p}_i$ momentua bere masa eta $\mathbf{v}_i$ abiaduraren arteko biderkadura eskalarra definitzen da —3.2 Ek.—.

$$\mathbf{p}_i = m_i \mathbf{v}_i \tag{3.2}$$

Problema honetan, Kepler-en probleman bezala, astro bakoitzaren hasierako $\mathbf{r}_i$ eta $\mathbf{v}_i$ bektoreak emanik, dt denbora-pauso jakin bat igaro ondoren gorputz horrek izango dituen $\mathbf{r}_i$ posizio eta $\mathbf{v}_i$ abiadura bektore eguneratuak kalkulatzea da helburua. Gorputz hauen higidurak 3.1 ekuazioan definitutako $\mathcal{H}$ sistema Hamiltondarrak definitzen ditu. Kapitulu honetako ia ekuazio guztiak $\mathbf{r}$ posizio eta $\mathbf{p}$ momentuen menpekoak badira ere —tradizionalki sistema Hamiltondarrak bi aldagai hauekin definitu dira eta—, kodean zehar $\mathbf{r}$ posizio eta $\mathbf{v}$ abiadurekin egingo da lan, 3.2 ekuazioa buruan edukitzea komeni delarik.

3.1 ekuazioan, sistemako astro bakoitzari i indize bat atxikitzen zaio. Hemendik aurrera, sistemako N gorputzak ondorengo arauekin ordenatuko dira.

- Lehenengo astroa sistemako gorputz masiboena izango da, normalean izar bat eta Eguzki-sistemaren kasuan Eguzkia dena. Izar bitarreko sistema bat bada, masa handiena duen izarra izango da lehena eta bestea bigarrena.
- Ondoren, sistemako barizentrotik gertuen dauden astroak etorriko dira eta urrutien daudenak azkenak. Eguzki-sistemako planetak eta Eguzkiak osatzen duten 9 gorputzeko sistemaren kasuan, ordena 3.1 taulan azaltzen da.

3.1 taulan aipatzen denez, sistemaren barizentroa etengabe mugitzen da, sistemako astroen higidurak direla eta. Hori hobeto antzemateko, 3.1 irudian Eguzki-sistemako barizentroa nola mugitzen joan den aztertu daiteke.

[1] Normalean, funtzio Hamiltondar hauetan posizioa $\mathbf{q}_i$ aldagaiarekin izendatzen bada ere, lan honetan zehar $\mathbf{r_i}$ notazioa erabiliko da, funtzio Hamiltondarra astroen $\mathbf{r}$ posizio eta $\mathbf{p}$ momentuen menpekoa izanik, hau da, $\mathcal{H}(\mathbf{r},\mathbf{p})$.

[2] *Barizentroa*: bi gorputz edo gehiagok osatzen duten sistema batean, sistema horren masa-zentroa da, hots, gorputz guztiek puntu honen inguruan orbitatzen dute, sistemaren zentro bezala jokatuz.

i	Astroa	r_i
1	Eguzkia	0.0071391
2	Merkurio	0.4710995
3	Artizarra	0.7273879
4	Lurra	0.9818334
5	Marte	1.3838546
6	Jupiter	4.9579053
7	Saturno	9.1769689
8	Urano	19.9207296
9	Neptuno	30.1189309

3.1 Taula: Taula honetan, Eguzki-sistemako izarra eta 8 planetak azaltzen dira, sistemako barizentroarekiko duten $r_i = |\mathbf{r}_i|$ distantzien arabera ordenatuta. Distantzia hauen unitateak Unitate Astronomikoak dira, non 1 UA $= 1.495978707 \cdot 10^8$ Km diren. Sistemako astroak mugimenduan daudenez, sistemaren barizentroa ere mugitzen doa; eta, hortaz, datu hauek 2000/01/01 0:00 datan neurtu zirenak dira. Hala ere, planeten orbiten eszentrikotasuna oso txikiak direnez, neurketa hauek ez dira asko aldatzen orbita osoan zehar higitzean. Eszentrikotasun handiko orbitak dituzten sistemetan, batezbesteko distantziak edota ardatz-erdi nagusia erabiltzea gomendatzen da.

Hau horrela izanik, Eguzki-sistema bezalako N-gorputzen sistema planetarioen higiduren hurbilpenak egitea da kapitulu honen helburu nagusia. Aipatzekoa da Kepleren fluxuan erabilitako bi sinplifikazioak kontuan hartuko direla N-gorputzen problemaren ebazpenerako ere bai:

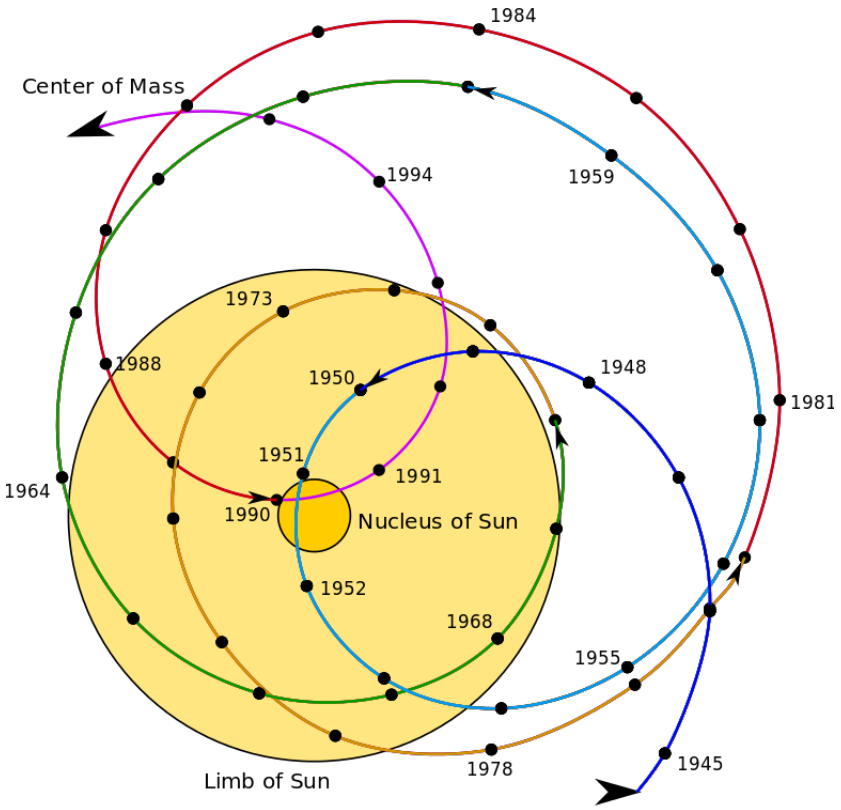
1. Sistemako gorputzak esfera perfektuak edota masa-puntuak bezala tratatu daitezke.

2. Ez dago beste kanpo edo barne indarrik eragiten N-gorputzeko sistema horretan, hau da, gorputzen artean sortzen diren elkar-indarrak kontuan hartuko dira bakarrik.

3.2 Ebazpena

Lehenik eta behin, problema ebazteko erabiliko den algoritmoa azaldu ahal izateko, Jacobi koordenatuak zer diren argitu behar da.

Jacobi koordenatuetan, erreferentzi-sistemako zentroa ez da izarraren zentroa. Gorputz jakin baten Jacobi koordenatuak, gorputz horrek baino indize txikiagoak dituzten gorputzek osatzen duten masa-zentro edo barizentroarekiko neurtzen dira, barizentro hori erreferentzi-sistemaren zentroa izanik gorputz horrentzat. Beraz, astro bakoitzak bere erreferentzi-sistemaren zentro propioa izango du.



3.1 Irudia: Wikipedia-tik eskuratutako irudi honetan, Eguzki-sistemako barizentroa eguzkiarekiko nola higitu den azaltzen da. Eguzkiak Eguzki-sistemako $\frac{m_1}{M} \approx \%99.86$ masa duenez —M Eguzki-sistemaren masa totala izanik—, barizentroak Eguzkiaren inguruan higitzeko ohitura du. Hala eta guztiz ere, barizentroa Eguzkiaren gainazaletik kanpo egon daiteke, Jupiter eta Saturno bezalako planeta gaseoso erraldoien kokapenaren arabera.

Koordenatu kartesiarretan i. gorputzaren posizioa $\mathbf{r}_i$ aldagaiak adierazten duenez, posizio bera Jacobi koordenatuetan $\mathbf{r'}_i$ bezala definituko litzateke. Momentuarekin berdin, koordenatu kartesiarretan $\mathbf{p}_i$ bada, Jacobi koordenatuetan $\mathbf{p'}_i$ bihurtuko litzateke.

$$\mathbf{r'}_i = \mathbf{r}_i - \mathbf{R}_{i-1} \tag{3.3}$$

3.3 ekuazioan bi koordenatu mota ezberdinetako balioak nola lotzen diren azaltzen da, non $\mathbf{R}_{i-1}$ i. gorputzaren erreferentzi-sistemaren zentroa den, indize txikiagoko astro guztiek osatzen duten barizentroa delarik. $\mathbf{R}_i$ masa-zentroa honela definitzen da matematikoki —3.4 Ek.—:

$$\mathbf{R}_i = \frac{1}{M_i} \sum_{j=1}^{i} m_j \mathbf{r}_j \,, \qquad \text{non } M_i = \sum_{j=1}^{i} m_j \text{ den} \tag{3.4}$$

Abiadura eta azelerazioak modu berean eraldatzen dira sistema batetik bestera. Azken finean, Jacobi koordenatuak koordenatu kartesiarren funtzio linealak dira —3.4 Ek.—, eta abiadura/azelerazioa posizioa deribatuz lortu daiteke bi koordenatu-sistemetan. Momentua, berriz, era ezberdin batean eraldatzen da —3.5 Ek.—.

$$\mathbf{p'}_i = m'_i \dot{\mathbf{r}}'_i = m'_i \mathbf{v}_i \,, \qquad \text{non } m'_i = m_i \frac{M_{i-1}}{M_i} = \frac{m_i M_{i-1}}{m_i + M_{i-1}} \text{ den} \tag{3.5}$$

3.5 ekuazioan agertzen den m'_i-ri Jacobi masa edo masa murriztua deitzen zaio. 3.3, 3.4 eta 3.5 ekuazioetan azaldutako Jacobi koordenatuak koordenatu erlatiboak dira $i \in [2,3,\ldots,N]$ kasuetan, i bakoitzerako $\mathbf{R}_{i-1}$ masa zentroarekiko hain zuzen ere. Hori bai, $i = 1$ kasuan beste modu batean definitzen dira $\mathbf{r}_1$, $\mathbf{p}_1$ eta m'_1 aldagaiak —3.6 Ek.—.

$$\mathbf{r'}_1 = \mathbf{R}_N \tag{3.6a}$$

$$\mathbf{p'}_1 = \sum_{j=1}^{N} \mathbf{p}_j \tag{3.6b}$$

$$m'_1 = M_N \tag{3.6c}$$

Beraz, definizioz, $\mathbf{r}_1$ sistema osoaren barizentroa, $\mathbf{p}_1$ sistemaren momentu totala eta m'_1, berriz, sistemaren gorputz guztien masen batura dira.

Jacobi koordenatuak hobeto azaltzeko, demagun hiru kanikez osatutako sistema bat dagoela, 3.2 irudian agertzen dena. Kanika bakoitzak bere masa eta posizioa dituenez, posizioak koordenatu kartesiarretatik Jacobi koordenatuetara eraldatu daitezke.

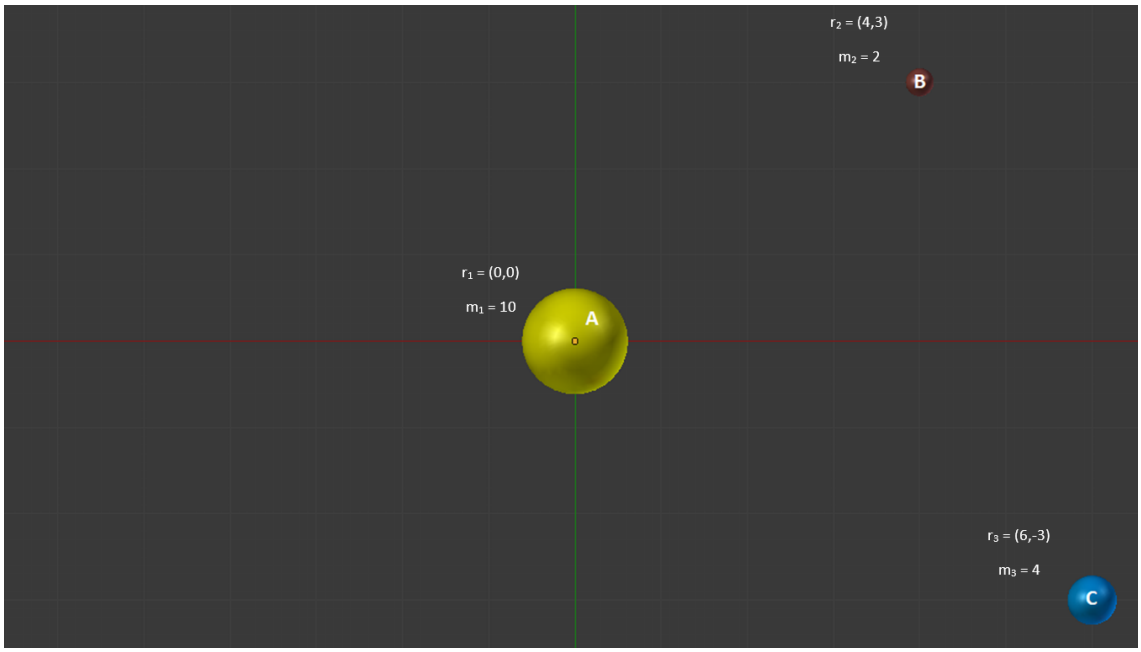
Lehenik eta behin, 3.4 ekuazioa aplikatuz, A eta B kaniken $\mathbf{R}_2$ masa-zentroa kalkulatuko da —3.7a Ek.—, B kanikaren Jacobi posizioa kalkulatu ahal izateko —3.7b Ek.—. Ondoren, hiru kaniken barizentroa kalkulatu beharko da —3.7c Ek.—, C kanikaren Jacobi posizioa deduzitu ahal izateko —3.7d Ek.—.

$$\mathbf{R}_2 = \frac{1}{10+2}\left(10(0,0) + 2(4,3)\right) = \left(\frac{3}{4}, \frac{1}{2}\right) \tag{3.7a}$$

$$\mathbf{r'}_2 = (4,3) - \left(\frac{3}{4}, \frac{1}{2}\right) = \left(\frac{13}{4}, \frac{5}{2}\right) \tag{3.7b}$$

$$\mathbf{R}_3 = \frac{1}{10+2+4}\left(10(0,0) + 2(4,3) + 4(6,-3)\right) = \left(2, \frac{-3}{8}\right) \tag{3.7c}$$

$$\mathbf{r'}_3 = (6,-3) - \left(2, \frac{-3}{8}\right) = \left(4, \frac{-27}{8}\right) \tag{3.7d}$$



3.2 Irudia: Irudi honetan, hiru kanika azaltzen dira bi dimentsioko XY planoan. A kanikak sistemako objektu masibo bezala jokatzen du, hots, izar bat bezala, eta koordenatu kartesiarren erreferentzi-sistemaren zentroan kokatzen da. B eta C kanikek planeta bezala jokatzen dute, masa txikiagoak dituztelarik.

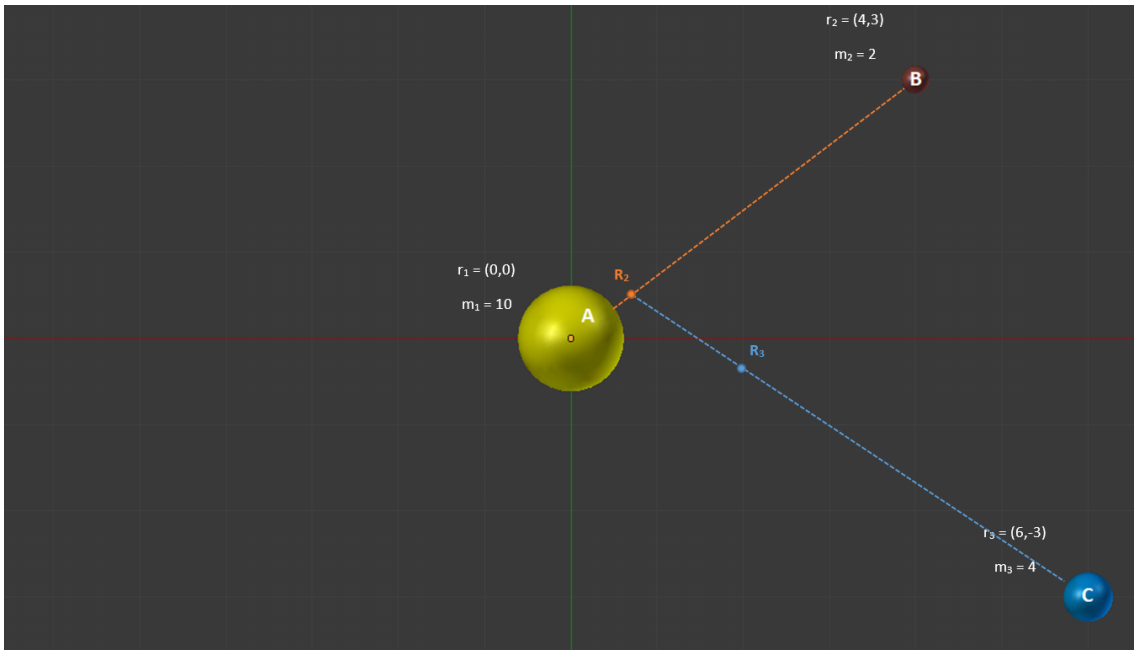
Azkenik,$\mathbf{r'}_1$ posizioa falta da kalkulatzeko, baina 3.6a ekuazioak $\mathbf{r'}_1 = \mathbf{R}_3 = \left(2, \frac{-3}{8}\right)$ dela baieztatzen du. Emaitzak grafikoki ikusi nahi badira, begiratu 3.3 irudia.

3.2.1 Algoritmoa

$\mathcal{H}$ funtzio Hamiltondarrak —3.1 Ek.— ez dauka soluzio analitikorik[3], eta, hortaz, soluzioaren hurbilpen bat bilatu behar da. Hori egiteko era bakarra ez badago ere, integratzaile sinplektiko[4] bat eraikitzea proposatzen da, Hamiltondarra hainbat ataletan zatituz, eta atal horiek osatzen dituzten problema sinpleagoak txandaka ebatziz [Rein and Tamayo, 2015].

[3] *Soluzio analitikoa*: konstante eta funtzio jakin batzuen bidez idatzi daitekeen problema baten soluzioa, operazio kopuru finitu batekin ebatzi daitekeena.

[4] *Integratzaile sinplektikoa*: sistema Hamiltondarrak ebazteko erabiltzen diren zenbakizko hurbilpenak lortzeko metodoak dira, hots, ekuazio diferentzial arruntak ebazten dituzten metodoak sistema Hamiltondarretan aplikatzeko.



3.3 Irudia: Irudi honetan, hiru kanika azaltzen dira bi dimentsioko XY planoan. 3.2 irudiarekin alderatuz, $\mathbf{R}_2$ eta $\mathbf{R}_3$ masa-zentroak azaltzen direla nabari da. Lerro laranja A eta B kaniken barizentroa —puntu laranja— egon litekeen eremua da, lerro horretan m_1 eta m_2 masen arabera barizentroa definitzen delarik. C eta beste bi kaniken masa-zentroa —puntu urdina— kalkulatzeko $\overline{AB}$ lerro laranjan egindako kalkulu berak burutu behar dira $\overline{\mathbf{R}_2 C}$ lerro urdinean.

Beraz, lehenik eta behin, funtzio Hamiltondarra berridatziko da. Hemendik aurrera, sistemaren energia zinetikoa adierazten duen gaia —gai kinetikoa deituko dena— Jacobi koordenatuetan dauden aldagaien bidez idatziko da —3.8 Ek.—.

$$\mathcal{H} = \sum_{i=1}^{N} \frac{\mathbf{p}'^2_i}{2m'_i} - \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_i m_j}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{3.8}$$

Ondoren, $\mathcal{H}_{\pm}$ gaia —3.9 Ek.— $\mathcal{H}$ funtzioari gehitu eta kenduko zaio, non $r'_i = |\mathbf{r}'_i|$ den.

$$\mathcal{H}_{\pm} = \sum_{i=2}^{N} \frac{Gm'_i M_i}{r'_i} \tag{3.9}$$

3.10 ekuazioan ikus daitekeenez, hasierako $\mathcal{H}$ Hamiltondarra Jacobi koordenatuak erabiliz eta gai berri bat sartuz berridatzi da, orain hiru Hamiltondar ezberdinetan banatu daitekeelarik: $\mathcal{H}_0$, $\mathcal{H}_{\text{Kepler}}$ eta $\mathcal{H}_{\text{Interaction}}$.

$$\mathcal{H}_{\text{N-body}} = \underbrace{\frac{\mathbf{p}_1'^2}{2m_1'}}_{\mathcal{H}_0} + \underbrace{\sum_{i=2}^{N} \frac{\mathbf{p}_i'^2}{2m_i'} - \sum_{i=2}^{N} \frac{Gm_i'M_i}{r_i'}}_{\mathcal{H}_{\text{Kepler}}} + \underbrace{\sum_{i=2}^{N} \frac{Gm_i'M_i}{r_i'} - \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_im_j}{|\mathbf{r}_i - \mathbf{r}_j|}}_{\mathcal{H}_{\text{Interaction}}} \tag{3.10}$$

$\mathcal{H}_0$ sistema osoak duen momentua adierazten du, sistema planetarioak duen lerro zuzeneko higidura definitzen duelarik. $\mathcal{H}_{\text{Kepler}}$ gaiak $N-1$ zatitan banatu daitezke, 3.11 ekuazioan azaltzen den itxurarekin.

$$\mathcal{H}_{\text{Kepler}} = \sum_{i=2}^{N} \left(\mathcal{H}_{\text{Kepler}}\right)_i \,, \qquad \text{non } \left(\mathcal{H}_{\text{Kepler}}\right)_i = \frac{\mathbf{p}_i'^2}{2m_i'} - \frac{Gm_i'M_i}{r_i'} \text{ den} \tag{3.11}$$

$\left(\mathcal{H}_{\text{Kepler}}\right)_i$ gai bakoitzak i. gorputzak m_i masarekin burutzen duen orbita kepleriarra deskribatzen du M_{i-1} masa duen $\mathbf{R}_{i-1}$ barizentroarekiko.

Azkenik, $\mathcal{H}_{\text{Interaction}}$ perturbazio ezberdinek osatzen duten Hamiltondarra da, orbita Kepleriarrei eragiten dien perturbazioak alegia. Perturbazio multzo hau bi gaietan banatzen da —3.12 Ek.—. Lehenengo gaia, $\mathcal{H}$ hiru sistematan banatzeko gehitu behar izan den gaia da —$\mathcal{H}_{\text{Interaction}_1} = \mathcal{H}_{\pm}$—, Jacobi koordenatuetan erraz konputatu daitekeena. Bigarren gaia, berriz, astro batek jasan behar dituen beste gorputz guztien arteko interakzioak $\mathcal{H}_{\text{Interaction}_2}$ definitzen dituen perturbazioen batura da, koordenatu kartesiarretan adierazten dena. Hortaz, $\mathcal{H}_{\text{Interaction}_1}$ gaiak i. astroak barne-gorputzekin dituzten $\mathcal{H}_{\text{Interaction}_2}$ gaiko interakzioak deuseztatzen ditu.

$$\mathcal{H}_{\text{Interaction}_1} = \sum_{i=2}^{N} \frac{Gm_i'M_i}{r_i} \tag{3.12a}$$

$$\mathcal{H}_{\text{Interaction}_2} = -\sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_im_j}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{3.12b}$$

$\mathcal{H}_{\text{Interaction}}$ Hamiltondarraren bi gaien erlazioa dela eta, $\mathcal{H}_{\text{Interaction}}$ 3.13 ekuazioan agertzen den bezala sinplifikatu daiteke.

$$\mathcal{H}_{\text{Interaction}} = \sum_{i=3}^{N} \frac{Gm_i'M_i}{r_i} - \sum_{i=1}^{N} \sum_{\substack{j=i+1 \\ j\neq 2}}^{N} \frac{Gm_im_j}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{3.13}$$

Wisdom-Holmanen mapa

Hortaz, $\mathcal{H}_0$ sistema osoaren higidura lineala, $\mathcal{H}_{\text{Kepler}}$ astroen orbitak eta $\mathcal{H}_{\text{Interaction}}$ sistemako gorputzek sortzen dituzten barne perturbazioak definitzen dituzte. $\mathcal{H}_0$ eta $\mathcal{H}_{\text{Interaction}}$ sistementzako soluzio analitikoak aurki daitezke —eta erabiliko dira—. $\mathcal{H}_{\text{Kepler}}$ sistema Hamiltondarrarentzat, berriz, zenbakizko metodoen bidez kalkulatutako hurbilpenak erabiliko dira, hau da, 2. kapituluan azaldutako Keplerren fluxua erabiliko da.

Behin sistema bakoitza nola kalkulatu daitekeen jakinik, 3.10 ekuazioko $\mathcal{H}_{\text{N-body}}$ ebatzi behar da. Hori egiteko, hiru sistemek definitzen dituzten higidurak batzeko era bat aurkitu behar da. Horretarako, integratzaile sinplektiko bat eraiki behar da, kasu honetan eragiketa-banatze metodoak erabiliz. Metodo hauek prozesu osoa sinplifikatu eta optimizatu ditzaketen zenbakizko metodoak dira.

Funtsean, eragiketa-banatze metodoak —edo *operator splitting methods*— ekuazio originala bi zati edo gehiagotan zatitzen ditu dt denbora-pauso baterako. Zati horien soluzioak banaka konputatu, eta, azkenik, soluzio horiek konbinatzen dira ekuazio originaleko soluzioa eraikitzeko.

Kasu honetan, *Drift-Kick-Drift* eragiketa-banatze metodoa erabiliko da, Wisdom-Holmanen mapa izenaz ezagutzen dena ere bai [Rein and Tamayo, 2015]. Metodo honen eskemak honako itxura du, $\mathcal{H}_{\text{N-body}}$ sistema Hamiltondarra dt denbora-pausoan zehar garatu nahi dela kontuan izanik —eragiketa hori egiten duen operadoreari $\hat{\mathcal{H}}(dt)$ deitu zaiolarik—:

- *Drift*: $\hat{\mathcal{H}}_0\left(\frac{dt}{2}\right)$ eta $\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right)$ sistemen eboluzioa burutu.
- *Kick*: $\hat{\mathcal{H}}_{\text{Interaction}}\left(dt\right)$ Hamiltondarraren eboluzioa garatu.
- *Drift*: Berriro ere $\hat{\mathcal{H}}_0\left(\frac{dt}{2}\right)$ eta $\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right)$ sistemen eboluzioa burutu.

Astroen $\mathbf{r}_i$ posizio eta $\mathbf{p}_i$ momentuen balioak dt denbora-pauso jakin bat igaro ondoren zeintzuk diren aurreikusteko $\mathcal{H}_{\text{N-body}}$ sistema Hamiltondar jakin batean, 3.14 ekuazio-multzoan azaltzen diren deribatuak kalkulatu behar dira, non $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$ eta $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_N)$ betetzen den.

$$\frac{d\mathbf{r}_i}{dt} = \frac{\partial \mathcal{H}_{\text{N-body}}}{\partial \mathbf{p}_i}(\mathbf{r}, \mathbf{p}) \tag{3.14a}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}_{\text{N-body}}}{\partial \mathbf{r}_i}(\mathbf{r}, \mathbf{p}) \tag{3.14b}$$

Drift pausoa

Drift pausoetan, $\hat{\mathcal{H}}_0\left(\frac{dt}{2}\right)$ eta $\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right)$ sistemak kalkulatzeko erabiltzen den ordenak ez du garrantzirik, eragiketa hauek trukakorrak baitira.

$\mathcal{H}_{\text{Kepler}}$ sistema dt denbora-pausoan garatzeko, errazena Jacobi koordenatuak erabiltzea da. Azken finean, hainbat gorputzen perturbazioak aztertu beharrean masa-zentroak erabiltzeak perturbazio bera sortzen duen astro hipotetiko batekin bakarrik kalkuluak egitea dakar. Hortaz, Kepler-en fluxua erabiliz, orbita Kepleriarrak kalkulatzeko zentzua dauka Jacobi koordenatuak erabiltzea.

$\mathcal{H}_0$ garatzeko, ordea, erabiltzen den koordenatu-sistemak ez du hainbesteko garrantzirik. Hala ere, pauso berean koordenatu-sistemak behin eta berriro ez aldatzeko, Jacobi koordenatuak erabiliko dira sistemaren eboluzioa kalkulatzeko —3.15 Ek.—. Ekuazio hauetan, $\mathcal{H}_0$-ren higidura lineala antzematen da, $\mathbf{r'}_1$ posizioa $\mathbf{v'}_1$ abiadura konstante batez aldatzen delako.

$$\frac{d\mathbf{r}_1}{dt} = \frac{\partial \mathcal{H}_0}{\partial \mathbf{p}_1}(\mathbf{r}_1, \mathbf{p}_1) = \frac{2\mathbf{p'}_1}{2m'_1} = \mathbf{v'}_1 \tag{3.15a}$$

$$\frac{d\mathbf{p}_1}{dt} = -\frac{\partial \mathcal{H}_0}{\partial \mathbf{r}_1}(\mathbf{r}_1, \mathbf{p_1}) = 0 \tag{3.15b}$$

DKD metodoa hainbat iterazioetan zehar errepikatu nahi bada, iterazio baten azkeneko $\frac{dt}{2}$ denbora-pausoko *Drift* eta hurrengo iterazioko lehen $\frac{dt}{2}$ denbora-pausoko *Drift* pausoak batu egin daitezke, $\hat{\mathcal{H}}_{\text{Kepler}}$ eta $\hat{\mathcal{H}}_0$ trukakorrak direlako eta $\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right) + \hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right) = \hat{\mathcal{H}}_{\text{Kepler}}(dt)$ eta $\hat{\mathcal{H}}_0\left(\frac{dt}{2}\right) + \hat{\mathcal{H}}_0\left(\frac{dt}{2}\right) = \hat{\mathcal{H}}_0(dt)$ betetzen direlako.

Hori bai, *DKD* iterazio kopuru jakin bakoitzero emaitzak koordenatu kartesiarretara pasa nahi badira, bi $\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right)$ eragiketak ezingo dira batu, tarteko emaitzaren koordenatu kartesiarrak kalkulatu nahi dira eta.

Kick pausoa

Kick pausoan, $\hat{\mathcal{H}}_{\text{Interaction}}(dt)$ eragiketa burutu behar da. 3.12 ekuazio-multzoan $\mathcal{H}_{\text{Interaction}} = \mathcal{H}_{\text{Interaction}_1} + \mathcal{H}_{\text{Interaction}_2}$ dela definitu da; eta, denboran zehar Hamiltondar honen eboluzioa hobeto aztertu ahal izateko, gai bakoitzaren eboluzioa bereiztuko da.

Lehenik eta behin, $\hat{\mathcal{H}}_{\text{Interaction}_1}(dt)$ eragiketa aztertuko da —3.16 Ek.—. $\mathcal{H}_{\text{Interaction}_1}$ gaia

ez dagoenez gorputzen $\mathbf{p}'$ momentuen menpe, gai honek astro bakoitzaren momentu eta abiaduran eragiten du, 3.14 ekuazio-multzoan zehaztutakoarekin bat eginez.

$$\frac{d\mathbf{r}'_i}{dt} = \frac{\partial \mathcal{H}_{\text{Interaction}_1}}{\partial \mathbf{p}'_i}\left(\mathbf{r}', \mathbf{p}'\right) = 0 \tag{3.16a}$$

$$\frac{d\mathbf{p}'_i}{dt} = -\frac{\partial \mathcal{H}_{\text{Interaction}_1}}{\partial \mathbf{r}'_i}\left(\mathbf{r}', \mathbf{p}'\right) = \frac{Gm'_i M_i}{|\mathbf{r}'_i|^3}\mathbf{r}'_i \tag{3.16b}$$

Kodean zehar emaitzak eta, oro har, eragiketak $\mathbf{p}$ momentuak erabili beharrean $\mathbf{v}$ abiadurekin kalkulatuko direnez, 3.16b ekuaziotik abiaduraren deribatua dt-rekiko eratorri daiteke —3.17 Ek.—.

$$\frac{d\mathbf{v}'_i}{dt} = \frac{GM_i}{|\mathbf{r}'_i|^3}\mathbf{r}'_i \tag{3.17}$$

Ondoren, $\mathcal{H}_{\text{Interaction}}$ sistemaren bigarren gaiak dakarren $\hat{\mathcal{H}}_{\text{Interaction}_2}(dt)$ eragiketa aztertuko da. Gai honek lan gehiago dakar. Izan ere, 3.10 ekuazioko $\mathcal{H}_{\text{N-body}}$ funtzio Hamiltondarrean azaltzen diren posizio eta momentuak Jacobi koordenatuetan adierazten dira, $\mathcal{H}_{\text{Interaction}_2}$ gaiaren kasuan izan ezik.

Lehenik eta behin, $\mathcal{H}_{\text{Interaction}_2}$ gaiaren posizio eta momentuekiko deribatuak erakusten dira 3.18 ekuazio-multzoan. $\mathcal{H}_{\text{Interaction}_1}$ gaiarekin gertatzen den bezala, gai honek ez du astroen posizioengan eragiten, baina bai bere momentu eta, hortaz, abiaduretan.

$$\frac{d\mathbf{r}_i}{dt} = \frac{\partial \mathcal{H}_{\text{Interaction}_2}}{\partial \mathbf{p}_i}\left(\mathbf{r}, \mathbf{p}\right) = 0 \tag{3.18a}$$

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial \mathcal{H}_{\text{Interaction}_2}}{\partial \mathbf{r}_i}\left(\mathbf{r}, \mathbf{p}\right) = -\sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{Gm_i m_j}{|\mathbf{r}_i - \mathbf{r}_\mathbf{j}|^3}\left(\mathbf{r}_i - \mathbf{r}_j\right) \tag{3.18b}$$

3.18b ekuazioa koordenatu kartesiarretan dagoenez, Jacobi koordenatuetara bihurtu behar da —3.19 Ek.— [Farrés et al., 2012].

$$\frac{d\mathbf{p}'_i}{dt} = \frac{\left(M_{i-1}\frac{d\mathbf{p}_i}{dt} - m_i \sum_{j=1}^{i-1}\frac{d\mathbf{p}_j}{dt}\right)}{M_i} \tag{3.19}$$

Behin momentuaren deribatua Jacobi koordenatuekin kalkulatuta dagoenean, $\mathcal{H}_{\text{Interaction}_2}$

gaiak dazkarren azelerazioak kalkulatu behar dira, 3.2 ekuazioa buruan izanik erraza dena kalkulatzeko —3.20 Ek.—.

$$\frac{d\mathbf{v'}_i}{dt} = \frac{\left(M_{i-1}\frac{d\mathbf{p}_i}{dt} - m_i\sum_{j=1}^{i-1}\frac{d\mathbf{p}_j}{dt}\right)}{M_i} \over m'_i \tag{3.20}$$

Beraz, *Kick* pausoa burutzeko bi koordenatu-sistema aldaketa behar dira, $\hat{\mathcal{H}}_{\text{Interaction}_2}(dt)$ eragiketa burutu ahal izateko. Alde batetik, gai horretan agertzen diren $\mathbf{r}_i$ posizioak aurreko *Drift* pausoan kalkulatu dira Jacobi koordenatuetan. Horregatik, $\mathbf{r'}_i \to \mathbf{r}_i$ posizio bektoreen koordenatu-sistema aldaketa egin behar da. Beste aldetik, 3.18b ekuazioan momentuaren deribatuak kalkulatzen direnean, momentu horiek Jacobi koordenatuetara itzuli behar dira —3.19 ekuazioa—.

Sasikodea

Orain, algoritmoaren atal guztiak azaldu badira ere, pauso guztien sasikodea laburbilduko da. Sarrerako datuak hauexek dira: N gorputzen $\mathbf{r}_i$ posizioak, $\mathbf{v}_i$ abiadurak eta $\mathbf{m}_i$ masak, non $i \in \mathbb{N}$ den. Algoritmoa dt denbora-pausoarekin exekutatuko da, T_{max} denbora simulatu arte.

Algoritmoa erreparatuz, aipatzekoa da *DKD* metodoaren iterazio ezberdineko *Drift* pausoak batzen badira, iterazio arteko posizio eta abiadurak ezingo direla jakin. Gainera, denbora-pauso jakin batean gorputzen posizio eta abiadura bektoreen balioak jakin nahi badira, bektore horiek Jacobi koordenatu-sistematik koordenatu kartesiarren sistemara pasa beharko litzateke.

Koordenatu-sistema aldaketen kopurua minimoa du algoritmoak, *DKD* iterazio bakoitzeko bi koordenatu-sistema aldaketa gertatzen baitira irteerako daturik kalkulatu behar ez badira —lau bestela—:

1. $\mathcal{H}_{\text{Interaction}}$-eko bigarren gaiaren azelerazioak kalkulatzeko behar diren gorputzen posizio bektoreak koordenatu kartesiarretan definituta egotea behar da. Hortaz, posizio bektore horiek Jacobi koordenatuetatik kartesiarretara aldatu behar dira.

2. $\hat{\mathcal{H}}_{\text{Interaction}_2}$ eragiketan kalkulatutako azelerazioak Jacobi koordenatuetara pasa behar dira, *Kick* pausoa aplikatu ahal izateko.

3. Irteerako datuak behar direnean posizio bektoreak Jacobi koordenatuetatik kartesiarretara pasa behar dira erabiltzaileak haiek erabili ahal ditzan.

4. Abiadura bektoreak ere eskatzen badira, Jacobi koordenatuetatik kartesiarretara pasa beharko dira.

$\mathbf{r}_i \rightarrow \mathbf{r'}_i$ eta $\mathbf{v}_i \rightarrow \mathbf{v'}_i$ Jacobi koordenatuak kalkulatu
$t \leftarrow \frac{dt}{2}$
$\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right)$ garatu Jacobi koordenatuak erabiliz, ***Drift***
while $t < T_{max}$ **loop**
    $\hat{\mathcal{H}}_{\text{Interaction}_1}(dt)$ garapenaren azelerazioak kalkulatu Jacobi koordenatuak erabiliz
    $\mathbf{r'}_i \rightarrow \mathbf{r}_i$ koordenatu kartesiarrak kalkulatu
    $\hat{\mathcal{H}}_{\text{Interaction}_2}(dt)$ garapenaren azelerazioak kalkulatu koor. kartesiarrak erabiliz
    $\mathcal{H}_{\text{Interaction}_2}$-eko azelerazioak Jacobi koordenatuetara pasa
    $\mathcal{H}_{\text{Interaction}}$-ko bi azelerazioak abiadurei aplikatu dt denbora-pausoan zehar, ***Kick***
    if *not* azken iterazioa *and not* output-a kalkulatzen den iterazioa **then**
        $t \leftarrow t + dt$
        $\hat{\mathcal{H}}_{\text{Kepler}}(dt)$ garatu Jacobi koordenatuak erabiliz, ***Drift***
    else if output-a kalkulatzen den iterazioa **then**
        $t \leftarrow t + \frac{dt}{2}$
        $\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right)$ garatu Jacobi koordenatuak erabiliz, ***Drift***
        $\mathbf{r'}_i \rightarrow \mathbf{r}_i$ eta $\mathbf{v'}_i \rightarrow \mathbf{v}_i$ koordenatu kartesiarrak kalkulatu output-a lortzeko
        $t \leftarrow t + \frac{dt}{2}$
        $\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right)$ garatu Jacobi koordenatuak erabiliz, ***Drift***
    end if
end loop
$t \leftarrow t + \frac{dt}{2}$
$\hat{\mathcal{H}}_{\text{Kepler}}\left(\frac{dt}{2}\right)$ garatu Jacobi koordenatuak erabiliz, ***Drift***
$\mathbf{r'}_i \rightarrow \mathbf{r}_i$ eta $\mathbf{v'}_i \rightarrow \mathbf{v}_i$ koordenatu kartesiarrak kalkulatu output-a lortzeko

3.2.2 Aldaketak kodeari begira

Kapituluarekin bukatzeko, kodean egin diren bi ohar aipatzea komeni da.

Alde batetik, *Drift-Kick-Drift* metodoaren *Kick*-pausoan, abiadurak eguneratzean gehitzen diren azelerazioak oso txikiak izan daitezke. Azelerazio horiek astroen arteko perturbazio edo elkar-eraginek sortzen dituzte, eta objektu horiek oso urruti egon daitezkeenez, indar horiek ez dira ia nabaritzen.

Kasu hau 2. kapituluan Gaussen f eta g koefizienteekin gertatzen den parekoa da, eta,

hortaz, biribiltze-erroreen metaketa ekiditeko soluzio bera proposatzen da. Kahanen batuketa konpentsatuaren algoritmoa 3.21 ekuazioko bi gaien baturak sor ditzakeen informazio galerak saihesteko nahikoa izan beharko litzateke, non $\mathbf{a'}_{i_1}$ azelerazioa $\hat{\mathcal{H}}_{\text{Interaction}_1}$ Hamiltondarra dt denboran zehar garatzeak sortzen duen eta $\mathbf{a'}_{i_2}$ azelerazioa $\hat{\mathcal{H}}_{\text{Interaction}_2}$ garatzearen ondorioa den.

$$\mathbf{v'}_i \leftarrow \mathbf{v'}_i + \left(\mathbf{a'}_{i_1} + \mathbf{a'}_{i_2}\right) dt \tag{3.21}$$

Beste hitz batzuetan, 3.21 ekuazioak *Kick* pausoa adierazten du, eta $\mathbf{v'}_i$ gaia $\left(\mathbf{a'}_{i_1} + \mathbf{a'}_{i_2}\right) dt$ baino askoz handiagoa denez, Kahan-en algoritmoa erabiltzea komeni da.

Beste aldetik, koordenatu-sistemen aldaketak egiteak errore handiak sor ditzake algoritmoaren exekuzio luze baten ondoren. Koordenatu-sistemen aldaketa hauek algoritmoaren kostu konputazionala igotzen dute, eta hori gutxi ez balitz, hauen inplementazio oker batek erroreak sor ditzake, koma higikorreko aritmetika erabiltzeagatik.

Hori ekiditeko, [Rein and Tamayo, 2015] artikuluan proposatzen den algoritmoak erabili dira, bai koordenatu kartesiarretatik Jacobi koordenatuak konputatzeko, eta baita alderantziz ere:

Algoritmo berriak

Kartesiar → Jacobi

$\mathbf{R} \leftarrow m_1 \cdot \mathbf{r}_1$
for $i : 2, \ldots, N$ **loop**
    $\mathbf{r'}_i \leftarrow \mathbf{r}_i - \mathbf{R}/M_{i-1}$
    $\mathbf{R} \leftarrow \mathbf{R} \cdot (1 + m_i/M_{i-1}) + m_i \cdot \mathbf{r'}_i$
end loop
$\mathbf{r'}_1 \leftarrow \mathbf{R}/M_N$ ♢ Masa-zentroa

Jacobi → Kartesiar

$\mathbf{R} \leftarrow \mathbf{r'}_1 \cdot \mathbf{M}_N$
for $i : N, \ldots, 2$ **loop**
    $\mathbf{R} \leftarrow (\mathbf{R} - m_i \cdot \mathbf{r'}_i)/M_i$
    $\mathbf{r}_i \leftarrow \mathbf{r'}_i + \mathbf{R}$
    $\mathbf{R} \leftarrow \mathbf{R} \cdot M_{i-1}$
end loop
$\mathbf{r}_1 \leftarrow \mathbf{R}/m_0$ ♢ 1. partikularen pos.

Algoritmo berri hauek, koordenatu-sistemak aldatzeko modu tradizionalaren baliokideak dira, hau da, ondoren azaltzen direnen baliokideak.

Hala ere, sistema planetarioren astroen masen artean ezberdintasun handiak egon daitezkeenez, koordenatuak aldatzerakoan eragiketa arriskutsu batzuk azaltzen dira, emaitzen zuzentasun eta egonkortasunak arriskuan jartzen dituztenak. Bi konbertsioetarako algoritmo tradizionalak eta berriak alderatu dira, eta, ondoren ikusiko den bezala, emaitza egonkorrenak algoritmo berriek ematen dituztenez, horiek erabiliko dira.

Algoritmo tradizionalak

Kartesiar → Jacobi

$sum \leftarrow m_1 \cdot \mathbf{r}_1$
for $i : 2, \ldots, N$ **loop**
  $\mathbf{R} \leftarrow sum / M_{i-1}$
  $\mathbf{r'}_i \leftarrow \mathbf{r}_i - \mathbf{R}$
  $sum \leftarrow sum + m_i \cdot \mathbf{r}_i$
end loop
$\mathbf{r'}_1 \leftarrow sum / M_N$ ♢ Masa-zentroa

Jacobi → Kartesiar

$\mathbf{R} \leftarrow 0$
for $i : N, \ldots, 2$ **loop**
  $\mathbf{r}_i \leftarrow \mathbf{r'}_1 + M_{i-1}/M_i \cdot \mathbf{r'}_i - \mathbf{R}$
  $\mathbf{R} \leftarrow \mathbf{R} + m_i/M_i \cdot \mathbf{r'}_i$
end loop
$\mathbf{r}_1 \leftarrow \mathbf{r'}_1 - \mathbf{R}$ ♢ 1. partikularen pos.

Bi algoritmo motak alderatzeko Eguzki-sistemako izar eta planeten koordenatu kartesiarretako $\mathbf{r}_A$ posizio jakin batzuk $\mathbf{r}'$ Jacobi koordenatuetara bihurtu eta berriro $\mathbf{r}_B$koordenatu kartesiarretara itzuli dira, $\mathbf{r}_A \rightarrow \mathbf{r}' \rightarrow \mathbf{r}_B$ eragiketak burutuz. Ondoren, astro bakoitzaren $|\mathbf{r}_A - \mathbf{r}_B|$ posizioen ezberdintasunak aztertuko dira 3.2 taulan, teorian 0 izan beharko lukeena.

	Tradizionala			Berria		
	X	Y	Z	X	Y	Z
Eguzkia	$1.734e-18$	$8.673e-19$	$2.710e-20$	$8.705e-19$	$2.609e-19$	$7.139e-21$
Merkurio	$2.775e-17$	0.0	$3.469e-18$	0.0	0.0	0.0
Artizarra	0.0	$3.469e-18$	0.0	0.0	0.0	0.0
Lurra	0.0	$1.110e-16$	$2.964e-20$	0.0	0.0	$6.776e-21$
Marte	$2.220e-16$	0.0	0.0	0.0	0.0	0.0
Jupiter	0.0	0.0	0.0	0.0	0.0	0.0
Saturno	$8.881e-16$	0.0	0.0	0.0	0.0	0.0
Urano	0.0	$1.776e-15$	0.0	0.0	0.0	0.0
Neptuno	0.0	0.0	0.0	0.0	0.0	0.0

3.2 Taula: Taula honetan, Eguzki-sistemako astro nagusien posizioak koordenatu-sistema aldaketen bidez nola aldatzen den erakusten da, hots, $|\mathbf{r}_A - \mathbf{r}_B|$-ren balioak agertzen dira XYZ ardatz bakoitzeko balioak ezberdintzen direlarik. Algortimo tradizionaletan $|\mathbf{r}_A - \mathbf{r}_B|$ errorea handiagoa dela nabarmentzen da. Errore hauen handitzeak koma higikorreko aritmetikako batuketa eta keneketak egiteak dakar. Izan ere, batuketa hauetan informazioa galdu daiteke bi batugaien artean diferentzia nabarmenak baldin badaude. Algoritmo tradizionaleko bi konbertsioetan, iterazioko azkeneko lerroetan dauden batuketek dute erroreen handitzearen errua —$sum \leftarrow sum + m_i \cdot \mathbf{r}_i$ eta $\mathbf{R} \leftarrow \mathbf{R} + m_i/M_i \cdot \mathbf{r'}_i$ eragiketek—, astroen masen artean desberdintasun handiak daudelako eta.

4. KAPITULUA

Eguzki-sistemaren simulazioa

Orain arte, sistema planetarioen higidurak deskribatzen dituzten sistema Hamiltondarrak —2.1 eta 3.1 Ek.— ebazteko saiakera egin da, bai Keplerren problema eta baita N-gorputzen problema ebatzi ahal izateko ere. Hasierako balio batzuk emanik, sistema hauetako astroen posizio eta abiaduren hurbilpenak kalkulatu dira denboran zehar. Hortaz, kalkuluak egin ondoren lortu diren emaitzak bisualki aztertzeko asmoarekin, Eguzki-sistemaren 3D-errepresentazio bat egingo da eskala errealean.

Simulazioa egiten hasi aurretik, kalkuluak simulazio kanpo egingo direla erabaki da. Beraz, simulazioaren datuak Julian kodetu den N-gorputzen problema ebazteko erabili den algoritmoaren bidez kalkulatu, eta datu horiek simulazioaren sarrera bezala eman behar dira —*.txt* testu fitxategi bat irakurriz—. Hau horrela izanik, kalkuluak edozein modutan egin daitezke —ez da 3. kapituluan azaldu den algoritmoa erabili beharrik—, sarrerako datuen formatua zuzena den bitartean.

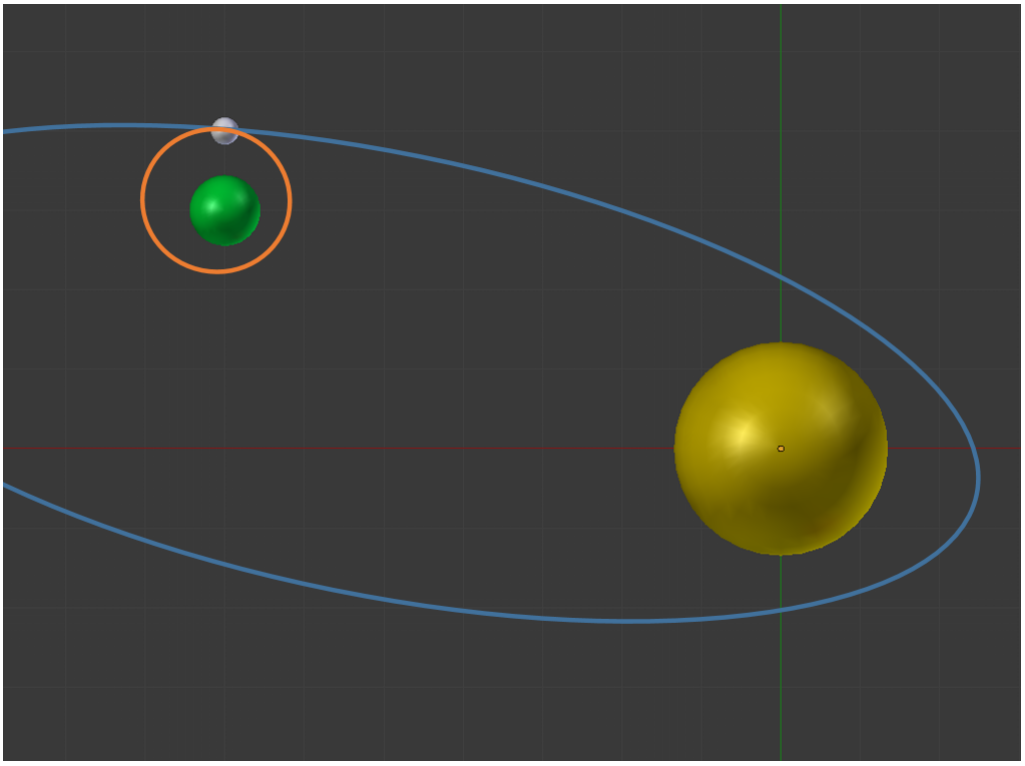
Eguzki-sisteman dauden gorputz kopurua oso handia da, handiegia gorputz guztiak simulazio barruan sartu ahal izateko. Hori dela eta, Eguzki-sistemako hamar gorputz ezagunenak sartuko dira simulazioan: Eguzkia, zortzi planetak eta Lurraren satelite natural bakarra, Ilargia. Ilargia baino gorputz masiboagoak diren sateliteak existitzen badira ere —horien artean Eguzki-sistemako sateliterik handiena, Gaminedes—, Lurraren satelitea aukeratu da bere ospea beste sateliteena baino handiagoa delako.

Eguzki-sistemaren simulazioaren arazoa

Simulazioan kontuan hartuko diren gorputzen artean, Ilargiak beste astroekiko berezitasun bat dauka. Zortzi planetek Eguzkiaren inguruan bira ematen badute ere, Ilargiak

Lurraren inguruan orbitatzen du. Hasiera batean honek arazo handirik ez dituela sortuko pentsatu badaiteke ere, erabilitako algoritmoan Ilargiaren orbita Kepleriarraren foku batean Eguzkia egongo litzatekeela esan nahi du, hots, Ilargia Eguzkiaren inguruan biratzen duela suposatuko litzateke. Kasu honetan, Lurrak Ilargiari eragiten dioen indarrak perturbazio bezala jokatuko luke, *DKD* metodoaren *Kick* pausoan Ilargiaren orbita zuzenduko litzatekeelarik.

Kasu berezi hau hobeto ulertzeko, 4.1 irudian aipatutakoaren diagrama txiki bat azaltzen da.



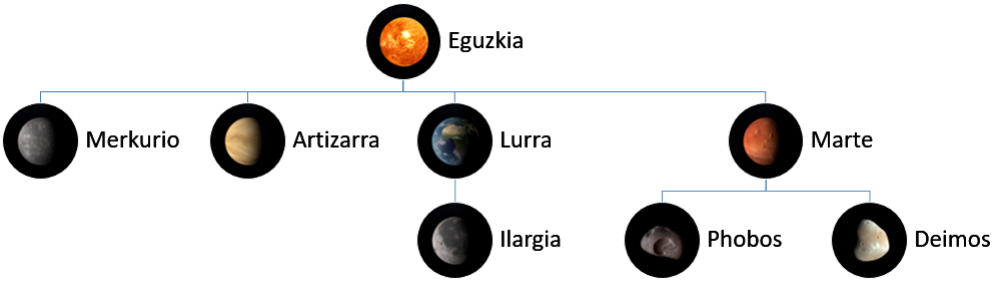
4.1 Irudia: Eguzkia esfera horia, Lurra esfera berdea eta Ilargia esfera zuria izanik, diagrama sinple honetan elipse laranjak Ilargiaren orbita zuzena adierazten du. Elipse urdinak, berriz, Ilargiak Eguzkiaren inguruan orbitatuko balu, posizio eta abiadura berdinekin izango lukeen orbita adierazten du. Esan bezala, *Kick* pausoan Lurraren perturbazioarengatik orbita laranjaren antzeko ibilbidea hartuko luke Ilargiak, *dt* denbora-pauso txiki bat hartu dela kontuan izanik.

Berez, 3. kapituluan jardun den algoritmoa ez dago prest berezitasun hauek dituzten sistemak ebazteko, hau da, izarren inguruan orbitatzen duten planeten inguruan biratzen duten sateliteak dituzten sistemak ez ditu ondo ebazten. Arazo hau konpondu ez bada ere —sistema sinpleagoak ebazteko algoritmoa diseinatu baita—, atal honetan soluzio bat proposatzen da.

DKD metodoan, arazoa *Drift* pausoak dakar. Izan ere, *Kick* pausoan gorputzen artean sor-

tzen diren indarrek sortzen dituzten abiadura aldaketak konputatzen dira, orbitaren ezaugarriak kontuan hartzen ez direlarik.

Adibidez, Eguzki-sistemako barne-planetek, bere satelite naturalek eta Eguzkiak sortzen duten sistema bat ebatzi nahi da. Lurrak satelite bat du: Ilargia; eta Martek berriz bi satelite ditu: Phobos eta Deimos. 4.2 irudiko hierarkia betetzen du sistemak, non Eguzkiaren inguruan lau planetek orbitatzen duten eta satelite bakoitza bere planetaren inguruan higitzen den.

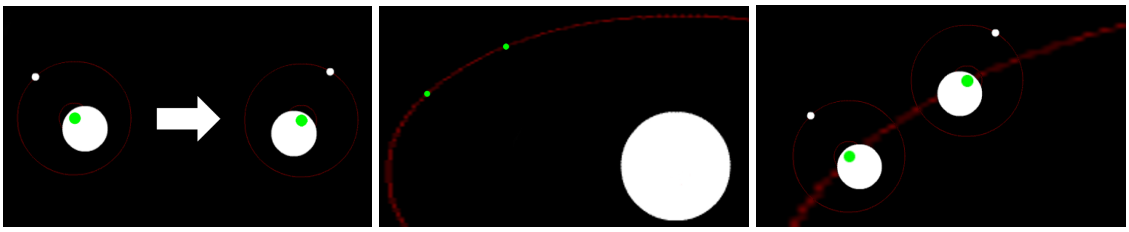


4.2 Irudia: Barne Eguzki-sistemaren hierarkia

Gakoa hemen *Drift* pausoa txandaka exekutatzea da. *Drift* pausoetan Eguzki-sistema sinplifikatu hau hainbat zatitan banatu behar da. Hierarkiako maila baxueneko gorputzekin hasi behar da, satelite bakoitzak eta bere planetak sortzen duten sistemak ebatziz. Beraz, lehenik eta behin, Ilargiak eta Lurrak osatzen duten Keplerren problema ebatzi behar da —lehen pausoa—, baita Martek eta bere bi sateliteek sortzen duten 3-gorputzen problemaren *Drift* pausoa ere —pauso guztiak Jacobi koordenatuak erabiliz kalkulatzeak asko laguntzen du—. Ondoren, landutako bi sistema hauen hasierako barizentroek Lurra eta Marteren kokapenak ordezkatuko dituzte Eguzkia eta lau planeten *Drift* pausoa garatzeko —bigarren pausoa—. Azkenik, bi pausoak "itsatsi"behar dira, masa-zentroen kokapen berriek satelite eta planetaren kokapen berriak definitzen dituztelarik —hirugarren pausoa—.

Adibidea hobeto bistaratzeko, begiratu 4.3 irudiak; Eguzkia, Lurra eta Ilargiak osatzen duten sistemaren *Drift* pausoaren garapena irudikatzen delarik.

Hala ere, N-gorputzen problema ebazteko algoritmo hierarkiko hau ez da inplementatu dena, ez baita proiektu honen helburuetan sartzen. Hortaz, arazo hau konpontzeko simulazioa garatzerakoan *dt* denbora-pauso txikiak erabili dira. *dt* txikiak erabiltzeak *Kick* pausoa maizago aplikatzea dakar, Ilargiaren abiadura bektorea gehiagotan zuzenduz. Soluzio hau matematikoki guztiz zuzena ez bada ere, Ilargia Lurraren inguruan orbitatzen mantentzeko nahikoa da —simulazioa martxan jartzeko nahikoa dena—.



(a) Lehenengo pausoa **(b)** Bigarren pausoa **(c)** Hirugarren pausoa

4.3 Irudia: Hiru irudi hauek sateliteak dituzten sistemen ebazpena hobeto ulertzen laguntzen dute. 4.3a irudian Lurra eta Ilargiak osatzen duten sistemaren *Drift* pausoa ikus daiteke $\frac{dt}{2}$ denbora-pausoa igaro ondoren, puntu berdea sistema horren barizentroa izanik. 4.3b irudian 4.3a irudiko barizentroak Eguzkia nola orbitatzen duen azaltzen da, $\frac{dt}{2}$ denbora-pauso bera igaro delarik. Azkenik, 4.3c irudian Lurra eta Ilargiaren posizio berriak azaltzen dira —eskubian— hasierako posizioekin batera —ezkerrean—. Kontuan eduki behar da 4.3b eta 4.3c irudiko barizentroen kokapenak berdinak direla.

Simulazioa garatzeko ingurunea

Behin kalkuluen arazoa konponduta dagoela, simulazioa zein tresna erabiliz eraikiko den definituko da. Aukeran hainbat erreminta badaude ere, ataza hau burutzeko *WebGL*[1] espezifikazio estandarra bidez eraikitako *Three.js* liburutegia erabili da, ordenagailu bidez 3D-animazioak sortu eta Web nabigatzaile batean renderizatzeko erabiltzen dena. Liburutegi hau *JavaScript* lengoaiarekin dago idatzita eta simulazioa kodetzeko programazio-lengoaia bera erabili da ere bai. Sarrerako datuak jasotzeko *HTML5* lengoaia erabili da, eta *CSS* programazio-lengoaia sortutako web orrian zehar simulazioaren atal ezberdinak kokatzeko.

Beraz, simulazioa web nabigatzaile batean irekitzen da zerbitzari baten beharrik gabe —dena lokalki exekutatzen da—. Erabiltzaileak sarrerako datuak kargatuko ditu, eta simulazioa martxan jarriko da. Simulazioa interaktiboa izatea nahi denez, *Dat.GUI* liburutegia ere erabili da, *JavaScript* lengoaiaz idatzita dagoena eta erabiltzaileak simulazioarengan kontrola edukitzeko tresnak definitzeko erabiliko dena.

Three.js liburutegiak simulazioaren garapenean pisu handia duenez, ingurune honetan liburutegi honek eta baita simulazioaren eszena nola eraiki eta funtzionatzen duen azalduko da ere.

[1] *Web Graphics Library*: edozein web nabigatzailetan 3D grafikoak renderizatzeko *JavaScript*-eko API baten inplementazioa definitzen duen espezifikazio estandarra da.

4.1 Three.js

Liburutegi hau erabiliz 3D objektuak edo animazioak bistaratu nahi badira, hasiera batean hiru gauza definitu behar dira: eszena, kamera eta renderizatzailea; eszena kameraren ikuspegitik renderizatu ahal izateko. Hitz gutxitan, renderizatzaileak eszenari dagokion modelo batetik kamera baten ikuspuntuari dagokion eszenaren irudia sortzeko prozesua burutzen du.

Hainbat kamera mota aurki daitezke liburutegi honetan, nagusiki bi erabiltzen direlarik:

- Perspektiba kamera: Simulazioaren eszenan erabiliko den kamara da, gizakien begiak ikusten duen modua kopiatzen saiatzen dena. Beraz, gero eta urrunago dauden gorputzak txikiagotzen joaten dira.
- Kamera ortografikoa: Kamera honetan ez da sakonera hautematen, objektuen dimentsioak urruntasunaren arabera aldatzen ez direlarik. Ikuspuntu-mota hau marrazketa-teknikoan erabiltzen da gehienbat, objektuen arteko distantzia errealak mantentzen ditu eta.

Kamera ortografikoa Eguzki-sistemaren gorputzen orbitak hobeto aztertzeko erabili badaiteke ere, perspektiba kamera erabiltzea erabaki da simulazio errealistago bat lortzeko ahaleginean.

Perspektiba kameraren *view frustum*-a[2] definitzeko lau ezaugarri behar dira:

- *Field of view* (Fov): *Field of view*-ak angelu bat adierazten du, kameraren ikuspuntutik antzeman daitekeen eszenaren zabalera adierazten duena. Graduak erabiltzen dira hau neurtzeko.
- *Aspect ratio*: kameraren ikuspuntutik lortzen diren irudien h altuera eta w zabalearen arteko $ratio$ proportzioa da —4.1 Ek.—.

$$ratio = \frac{w}{h} \tag{4.1}$$

- *Near*: kameratik gorputz batera dagoen distantzia n *near* balioa baino txikiagoa bada, gorputz hori ez da renderizazioan kontuan hartuko.

[2] *View frustum*: objektuak renderizatzeko, objektu horiek *view frustum* eremu barruan kokatu behar dira, eremu hau honako parametroek definitzen dituztelarik: Fov, $ratio$, n, f, kameraren posizioa eta kamerak duen ikuspuntuaren norabidea.

- *Far*: *Near* aldagaiaren aurkakoa, kameratik gorputz batera dagoen distantzia f *far* balioa baino handiagoa bada, gorputz hori ez da kameraren *view frustum* barruan egongo.

Near eta *far* balioak definitzeko, simulazioan zehar erabiliko diren distantzia unitateak jakin behar dira. Gainera, simulazioaren denbora-unitatea definitu behar da ere bai.

Eguzki-sistemako gorputzen kokapenak neurtzeko Unitate Astronomikoak erabiliko dira, $1\text{UA} = 1.495978707 \cdot 10^8\text{km}$ delarik. Izan ere, Eguzki-sistemako gorputzen arteko distantziak aipatzen direnean unitate hau erabiltzen da. Gainera, Eguzki-sistemako bi gorputzen arteko batezbesteko distantziak definitzen du unitate hau; hots, urte lurtar batean zehar Eguzkia eta Lurraren artean izaten den batezbesteko distantzia da unitate astronomiko bat.

Behin distantzia unitatea definituta dagoela, simulazioaren defektuzko abiadura erabaki behar da. Hurrengo kapituluan astroen $\mathbf{v}$ abiadura bektoreak $UA/egun$ unitateen bidez definitu direnez, denbora errealeko segundo bakoitzero egun lurtar bat simulatzea erabaki da, $1egun = 86400seg$ direla kontuan izanik.

Beraz, Unitate Astronomikoak erabiliz, $n = 5 \cdot 10^{-5}\text{UA}$ ezarri da kamera astroei asko hurbildu ahal izateko, eta $f = 500\text{UA}$ bezala definitu da, Eguzki-sistema osoa urrutitik ikusi ahal izateko. Erreferentzia bat edukitzeko, simulazioan aztertu daitekeen distantzia txikiena Lurra eta ilargiaren arteko da, batezbeste $3.844 \cdot 10^5\text{Km} \approx 0.0025\text{UA}$-ko distantzia dena. Kontrako aldean, eszenan aurkitu daitekeen gorputzik urrunena Neptuno da, honen orbitaren ardatz-erdi nagusia 30.11UA izanik. f-ren balioa azken hau baino askoz handiagoa da, simulazioa inguratzen duen testura[3] kameraren edozein kokapenetik ikusi ahal izateko.

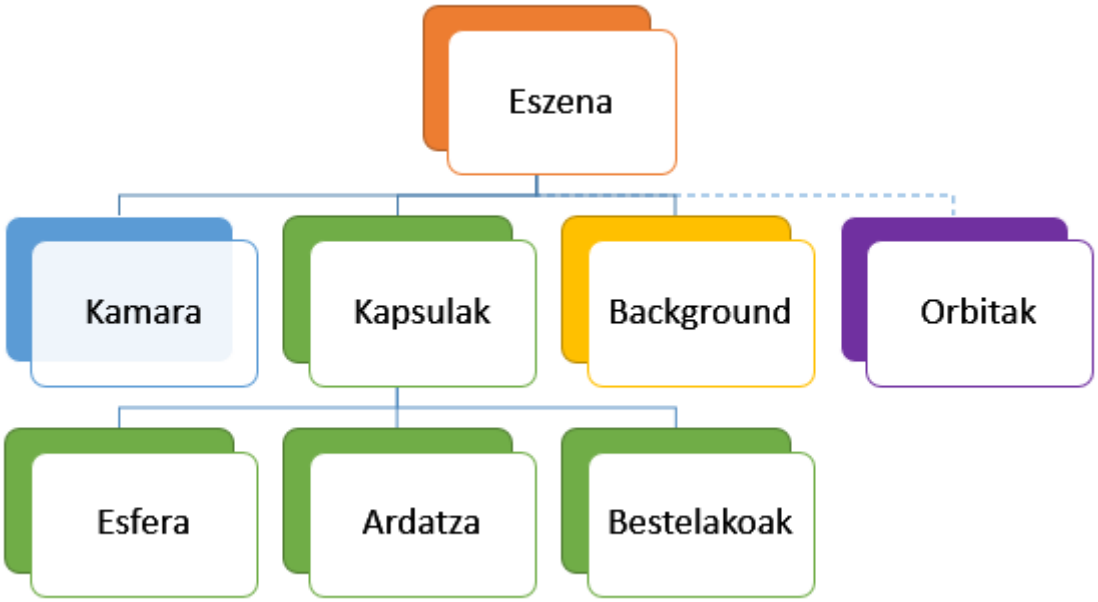
Renderizatzailearen aldetik, *Three.js* liburutegiak dakarren *WebGL* renderizatzailea erabiliko da, *WebGL* espezifikazio estandarra aplikatuz definituta dagoena. Honen instantzia egitean, web nabigatzailearen leihoaren tamainako renderizazioak egiteko espezifikazioa egin behar da, eta leiho horren tamaina aldatzerakoan, renderizatzaileari leiho berriaren tamaina zein den esan behar zaio.

Behin eszena hutsa, kamera eta renderizatzailea definituta daudelarik, Eguzki-sistema osatzen duten gorputz nagusiak eszenara sartuko dira, baita behar diren argi-iturriak eta ardatz-laguntzaileak ere.

[3]Testura honetan Esne Bidea ordezkatzen duten hainbat izar ageriko dira, simulazioaren murgilketa handituz.

4.1.1 Eszenaren hierarkia

Simulazioaren eszena elementu askok osatzen dute, eta elementu horiek hierarkia zuzen bat izatea beharrezkoa da renderizazio eraginkor bat burutzeko. Hori dela eta, 4.4 irudian hierarkia horren sinplifikazioa azaltzen da.



4.4 Irudia: Simulazioaren eszenaren hierarkia.

Eszena lau ataletan banatu daitezkeen objektuetaz osatuta dago:

- Kamera: Eszenan kokatzen den kamera perspektiboak renderizazioak nondik eta zer noranzkoan egiten diren definitzen du. Kamera eszenan zehar mugitu dezake erabiltzaileak, bai arratoiarekin eta baita teklatuko ←↑→↓ teklekin ere.

- Kapsulak: Kapsula bakoitzak astro bat eraikitzen duten objektuak multzokatzen ditu. Saturno adibide bezala hartuz, hiru objektuk osatzen dute planeta hau: planeta bera, Saturnoren eraztunak eta planeta bi poloetatik zeharkatzen duen ardatza. Hiru objektu hauek kapsularen zentroan kokatzen dira, eta planetaren kokapena aldatu nahi denean, hiru objektuak mugitu beharrean kapsula bakarrik mugitu beharko da.

  Eguzkiaren kasuan kapsulak esfera/planeta bat eta argi-iturri bat batzen ditu. Beste 9 gorputzen kapsulek esfera bat eta planeta horren ardatza gordetzen ditu, Saturnoren kasua bere eraztunengatik berezia izanik.

Aipatutako argi-iturria puntu-argi bat da, Eguzkiaren zentroan kokatzen dena. Objektuek itzalik proiektatzen ez dutenez —itzalak ez dira simulazioan aktibatu kostu konputazionala asko igotzen dute eta—, argi-puntua Eguzkiaren barruan egoteak ez du axola.

- Atzealdea: Eguzki-sistema gorputzak oso txikiak dira, beraien artean dauden distantziekin konparatzen bada. Renderizazioaren atal nagusi bat eszenaren *background* izango da. Hori dela eta, eszena osoa esfera erraldoi batez inguratu da, eta esfera horren barneko aurpegiari izarrez betetako testura bat itsatsi zaio.

- Orbitak: simulazioaren exekuzio hasieran orbitak ez badira erakusten ere, erabiltzaileak aukera izango du orbita horien ibilbideen lerroak pantailaratzeko. Hortaz, lerro hauek exekuzio denboran sortu/ezabatu daitezke. Astroen orbita guztiak pantailaratu daitezke Eguzkiarena izan ezik, astro honen posizioa ez baita ia aldatzen.

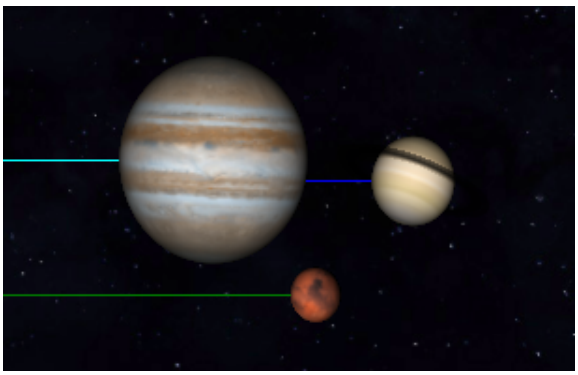
Objektuak sortzeko lau geometria ezberdin erabili dira: lerroak orbita eta ardatzentzat, esferak astroentzat, *grid* motako geometria erreferentzi-koadrikularentzat eta eraztun motako geometria Saturnoren eraztunentzat. Hortaz, objektuak eraikitzeko bi gauza definitu behar dira: geometria eta geometriari aplikatuko zaion materiala.

Lerroen geometria hiru dimentsioko erpinen *array* bidez osatzen da. Array hauek hasiera batean hutsik definitzen dira orbiten kasuan, eta orbita pantailaratu nahi denean, astroen kokapenak lerroen erpin berriak izango dira. Ardatza, berriz, bi erpinez osatutako lerro bertikala da, eta planetaren kapsularekin batera biratzen da ardatzak astroaren inklinazioa definitu dezan —4.5 irudian Lurra eta bere ardatza azaltzen dira—.

4.5 Irudia: Simulazioan zehar Lurrak duen itxura azaltzen da irudian. Lurra zeharkatzen duen lerro zuria astro honen ardatza da, eta ardatz honen inguruan egiten du bere buruarekiko errotazioa. Lurra eta ardatzarekin batera aipatutako koadrikula grisaren atal batzuk azaltzen dira.

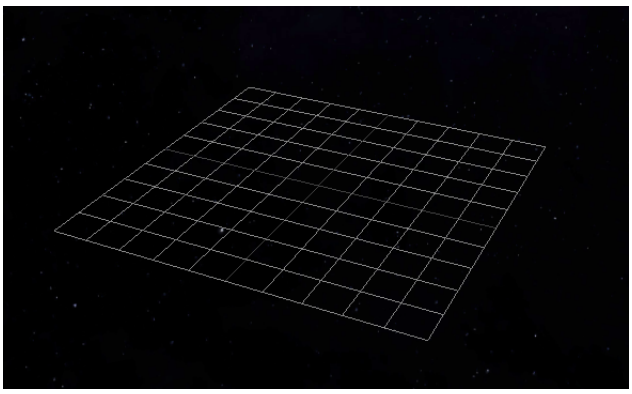
Aipatu den bezala, ardatz guztiei kolore zuria ezarri zaie, eta orbiten lerroei beste hainbat kolore ezberdin esleitu zaie.

Esferen geometria definitzerakoan, esferaren erradioa eta esferaren sareak izango dituen aurpegi/triangelu kopurua ezarri behar dira. Kasu guztietan, 1984 aurpegi erabili dira —liburutegiak uzten duen aurpegi kopuru maximoa—, erradioa astroaren araberakoa izanik. Material aldetik, hamar gorputzei edo esferei testurak ezarri zaizkie —4.1.2 sekzioan nola ezarri diren azaltzen da—. Hainbat planeten itxura 4.6 irudian antzeman daiteke.



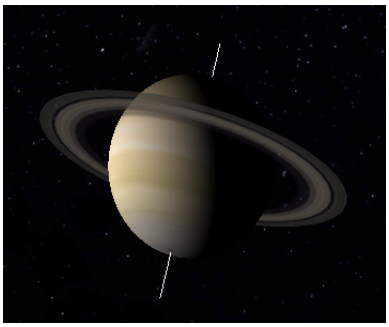
4.6 Irudia: Irudi honetan, Marte, Jupiter eta Saturno planetak azaltzen dira ia lerrokatuta. Bakoitzak bere testura propioa dauka eta Saturnoren eraztunak antzeman daitezke. Irudiko lerro berdea Martek eraman duen ibilbidea definitzen du, lerro urdin argiak Jupiterrena eta urdin ilunak, berriz, Saturnorena.

Erreferentzi-koadrikularen kasuan, koadrikula karratuaren alde baten tamaina eta alde bakoitza zenbat zatitan banatu behar den definitu behar da. Kasu honetan, koadrikulak $10\text{UA} \cdot 10\text{UA} = 100\text{UA}^2$-ko azalera du, eta koadrikularen gelaxka bakoitzaren alde batek 1UA-ko luzeera du. Koadrikula hau eszenaren zentroan kokatzen da, orbita eliptikoen ia plano berean, eta material bezala kolore gris bat du —4.7 irudia—.



4.7 Irudia: Simulazioaren erreferentzi-koadrikula.

Azkenik, Saturnoren eraztunen geometria definitzeko *Three.js* liburutegitik kanpoko klase bat erabili da *XRingGeometry* deiturikoa[4]. Saturnoren eraztunari testura bat jarri zaio ere bai, 4.8 irudian ikusi daitekena.



4.8 Irudia: Saturnoren eraztunak simulazioan. Itzalik ez direnez konputatzen, eraztun osoak Eguzki-izpiak jasotzen dituela ematen du, esfera hor egongo ez balitz bezala.

4.1.2 Testuren erabilera

Kapitulu honetan aipatu den bezala, hamar gorputzen esferei eta Saturnoren eraztunei kolore bat ezarri beharrean, testurak erabiltzea erabaki da. Eguzki-sistemako planeten, Ilargiaren eta Eguzkiaren gainazalaren testura hauek aurkitzea ez da zaila, NASA-k eskura jartzen baititu planeta hauen 3D modeloak egin ahal izateko.

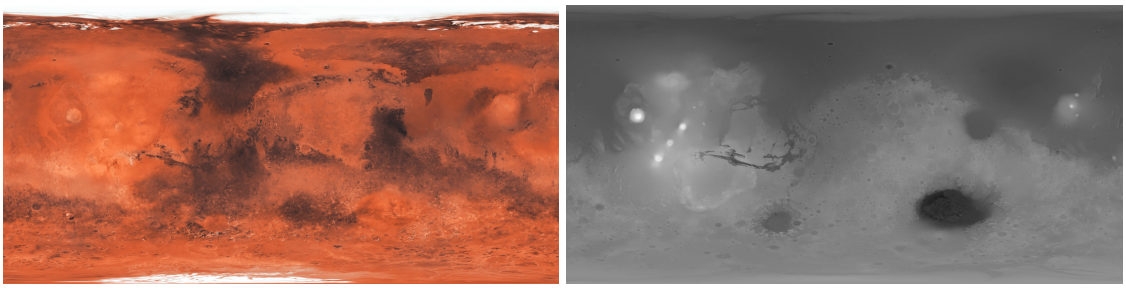
Gainera, Eguzki-sistemako barne planetak gaseosoak ez direnez, hots, gainazal solidoa dutenez, planeta hauen testurekin batera testura hauen normalak[5] aurki daitezke. 4.9 irudian Marteren gainazalaren testura azaltzen da bere normalekin batera.

Hasiera batean, simulazio foto-errealista bat garatzea pentsatu zen, *Blender* progrmaren *shader*-ak[6] erabiliz, eta, ondoren, ideia *shader* horiek *WebGL* bidez renderizatzea zen. Hala ere, *Blender*-en *shader* hauek ezin dira era azkar batean *WebGL*-era eraman. Horretaz konturatzerako planeten gainazalei itxura emango zioten *shader*-ak eginak zeudenez,

[4] Kodea https://cdn.rawgit.com/bubblin/The-Solar-System/master/js/shared/xRingGeometry.js orrian aurkitu daiteke.

[5] *bump-mapping*: konputagailu bidezko grafikoetan, teknika hau 3D objektuetan zimurrak eta erosioa simulatzeko erabiltzen da, objektuaren normalak aldatuz eta, hortaz, argi-izpien isla eraldatuz. Beraz, normalen irudiak testuraren tamaina bera du, eta normalen irudiaren pixel bakoitzak objektuan testuraren pixel hori dagoen normala definitzen du.

[6] *Shader* bat objektu baten gainazalak argi-izpiekin duen interakzioa deskribatzen duen programa da —gainazalaren kolorea zuzenean zehaztu beharrean—, *GLSL* bidez idatzita eta GPUan exekutatzen dena. *Blender* 3D modeloak garatzeko, animatzeko eta renderizatzeko erabiltzen den software-librea da, eta programa honekin erabili daitezkeen *shader*-ak oso onak dira gainazal foto-errealistak sortzeko.



(a) Marteren testura **(b)** Marteren sakonera-mapa

4.9 Irudia: Testura hauek Marte-ren itxura definitzen dute, batek kolorea —4.9a irudia— eta besteak sakonera emanik —4.9b irudia—. Kuriositate bezala, *Olympus Mons*, Eguzki-sistemako mendirik altuena Marten aurkitzen da. 4.9b irudiko ezker aldean mendi horrek duen 22.5 Km-ko altuera nabaritzen da, konkor hori punto zuri lodi batek adierazten du eta.

beste erabilera bat eman zaie, B. eranskinean nola eraiki eta zertarako erabili diren azaltzen delarik.

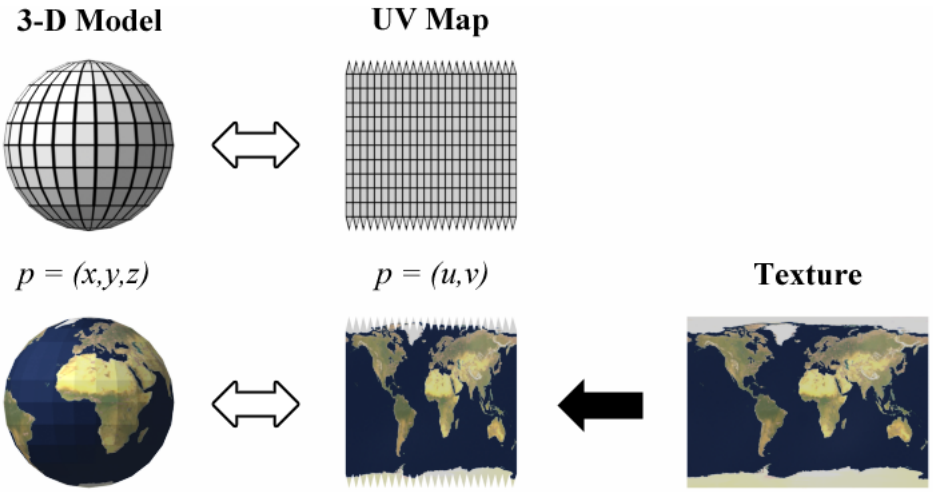
Three.js liburutegian *shader*-ak ez dira zuzenean kodetu behar nahi diren ezaugarriak dituzten gainazalak lortzeko, ezta *Blender*-en ere. Horren ordez, materialen propietateak aldatu daitezke materiala bera definitzerakoan.

Lurraren kasua adibide bezala hartuz, ondorengo espezifikazioak egin dira: Lurraren gainazalari 4.10a irudiko oinarrizko kolorea eman zaio, 4.10b irudian adierazten diren normalek lehenengo irudiko testurari sakonera ematen die —*bump-mapping* aplikatuz—, eta 4.10c irudian azaltzen den mapa-espekularrak argi-izpiak gainazaleko zein puntuetan islatuko diren espezifikatzen du, islatutako argi-izpien kolorea zuria dela adierazi delarik.

(a) Lurraren testura **(b)** Lurraren sakonera-mapa **(c)** Lurraren mapa-espekularra

4.10 Irudia: Testura hauek Lurraren itxura definitzen dute, batek kolorea —4.10a irudia—, bigarrenak sakonera emanik —4.10b irudia— eta azkenak islapen-espekularra gainazalaren ureremuetan egin behar dela esaten duelarik —4.10c irudia—.

Mapa-espekularrean kolore zuriak argi-izpiak islatu daitezkeen zonaldeak zehazten ditu, hau ozeano eta itsasoetan betetzen da, urak propietate hau du eta. 4.10 irudi-multzoko testurak erabiliz, 4.5 irudiko renderizazioa lortzen da. Irudi horretan, Lurraren esferaren erdikaldean islapen-espekularra ozeanoan bakarrik gertatzen dela ikusi daiteke, efektu hau mapa-espekularraren bidez lortzen delarik.



4.11 Irudia: Wikipedia-tik eskuratutako eskema honetan, testura bidimentsional bat esfera batean nola proiektatzen den azaltzen da. UV proiekzioan, U eta V testuraren ardatzak eta X, Y eta Z objektuaren ardatzak badira, proiekzioa $p = (u, v) \rightarrow p = (x, y, z)$ aldaketa bezala definitzen da.

Astro guztiek oinarrizko kolorea ematen dioten testurak dituzte. Eguzki-sistemaren barne planeta eta Ilargiak *bump*-mapak dituzte gorputz hauek gaseosoak ez baitira —Artizarraren kasuan atmosfera oso lodia denez ez da planetaren gainazala ikusten espaziotik—; eta Lurra da mapa-espekularra duen gorputz bakarra, itsasoak dituen astro bakarra delako. Astro guztiak opakoak definitu dira, Saturnoren eraztunak izan ezik, eraztunen tarteetatik planeta ikusi ahal izateko erdi-transparenteak egin dira eta.

Azkenik, testurak irudi bidimentsionalak direla eta hiru dimentsioetako esferetan proiektatu behar direla aipatzekoa da. Ataza honi UV proiekzioa edo *UV Mapping* deitzen zaio, eta 4.11 irudian testuraren proiekzioa deskribatzen duen eskema bat azaltzen da.

Kasu honetan, gorputzak sortzeko erabili diren esfera eta eraztunen datu motek proiekzio hau automatikoki egiten dute —klase bakoitzean bere proiekzioa kodetuta dago—, baina *Blender* bezalako programetan ataza hau eskuz egin beharko litzateke.

4.2 Simulazioaren logika

Behin eszena osatzen dituzten objektuak prest daudelarik, simulazioa nola eraiki den azalduko da. Simulazioa renderizazio bakoitzak sortzen duen irudi jarraien bidez osatzen da, animazio bat lortzen delarik. Uneko irudia renderizatzetik hurrengo *frame*-a renderizatzera δt denbora jakin bat pasatzen da, eta Δt-ren arabera astroen posizioak gehiago ala

gutxiago mugituko dira, hots, simulatu beharko den eszena renderizazioen denboren araberakoa da.

Hala ere, simulazioaren denbora nola definitzen den aipatu baino lehenago, aurretik sarrerako datuak nola jasotzen diren eta zer nolako arauak jarraitu behar dituen jakitea beharrezkoa da.

Simulazioa martxan jartzeko bi gauza eskatzen dira. Alde batetik, denboran zehar hamar astroen posizioak dituen fitxategia, non lerro bakoitzean 30 datu ezberdin azaltzen diren —hamar astroen posizioen hiru koordenatuak izanik—. Simulazioak ondo funtzionatzeko, fitxategi horretan, lerro bakoitzeko posizioetatik hurrengo lerroko kokapenetara egun lurtar bateko desberdintasuna egon behar du. Beraz, fitxategiak n lerro baditu, n egun simulatu ahal izango dira; eta fitxategiak $n \cdot 30$ tamainako matrizea gordeko du. Sarrerako fitxategi hauek ondo irakurri ahal izateko *.txt* formatua izan behar dute. Matrize handietan fitxategi honek memoria asko beharko duenez, hobekuntza bat fitxategi bitarrak irakurri ahal izatea izan litzateke.

Beste aldetik, simulazioari hasierako-data bat pasa behar zaio. Datu errealekin lan egiten denez, simulazioko une bakoitzean zein egun den jakitea ondo legoke; eta, horretarako, gutxienez fitxategiko lehen lerroko datuen data zein den jakin behar da.

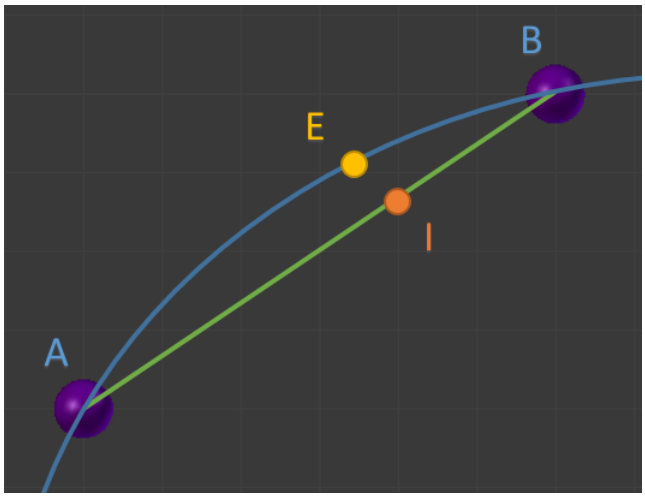
Behin behar diren datu guztiak definitzean, animazioa nola egiten den definitu daiteke. Animazioa hastean, simulazioa begizta batean sartzen da. Ahal duen bezain laster renderizazioekin hasi, eta renderizazio bat bukatzean hurrengoarekin hasten da. Renderizazioen artean, astroen posizioak aldatu behar dira igaro den Δt denboraren arabera.

Adibidez, segundo bat pasatzen denean, simulazioak egun bat simulatzen badu $v_{sim} = 1\text{egun}/\text{seg}$ eta segunduko 24 renderizazio egiten badira, *frame* batetik hurrengora igarotzeko batez-beste $\Delta t = 0.042$ egun aurreratu beharko da simulazioa irudi bakoitzean. Renderizazio denborak konstanteak ez direnez, *timer* bat erabiltzea erabaki da. Renderizazio baten hasieran, *timer* hori 0-ra jartzen da eta iterazio honen irudia lortzean Δt *timer*-aren balioa gordetzen da. 4.2 ekuazioan renderizazio hori kalkulatzeko behar izan den t_{sim} egun kopurua lortzen da, simulazioan simulatu den denbora hain zuzen ere.

$$t_{sim} = v_{sim} \cdot \Delta t \tag{4.2}$$

Adibidearekin jarraituz, $\Delta t = 0.042$ egun bada, irudi bakoitzean $0.042\text{egun} = 1\text{ordu}$ simulatzen direla esan nahi du. Arazoa gorputzen eguneko posizioak bakarrik direla ezagunak da, posizio horiek egun bakoitzeko 0:00 AM ordukoak direla suposatzen delarik.

Astroen edozein uneko posizioak definitzeko, une jakin bateko bi egun gertuenen **A** eta **B** posizioen arteko interpolazio lineala kalkulatzen da. Kalkulu hau 4.12 irudian bistaratu daiteke.



4.12 Irudia: **A** puntutik **B** puntura iristeko egun lurtar bat behar dela eta bi puntuek gorputzaren posizioa 0:00 AM-etan definitzen dutela suposatzen bada, interpolazio lineala erabiliz egun horretako arratsaldeko laurak direnean **I** puntuan egongo dela ondorioztatzen da. Hala ere, orbita lerro urdinak definitzen duenez, une horretan gorputzak izango duen posizioa **E** puntuak definituko du. $|\mathbf{E}-\mathbf{I}|$ interpolazio lineala erabiltzeagatik lortzen den errorea da.

Interpolazio lineala erabiltzeak errore nabarmenak ekartzen baditu ere, konputazionalki baliabide gutxi behar ditu, eta horrek segundoko konputatu daitezkeen irudi kopurua —*FPS* edo *frames per second*— handitzen du. 4.12 irudiko notazioa erabiliz, 4.3 ekuazioak interpolazio lineala definitzen du, $h = T_{sim} - \lfloor T_{sim} \rfloor$ egun horretako ordua eta T_{sim} simulazioaren uneko denbora dela kontuan izanik —egun lurtarreko unitatetan neurtzen dira bai h eta baita T_{sim} ere—.

$$I = (1-h)\cdot A + h\cdot B \tag{4.3}$$

Interpolazio errorea handiagoa bihurtzen da astroaren abiadura eta orbitaren eszentrikotasunak handitzen diren heinean. Simulazio honetako gorputzek ez dituzte orbita oso eszentrikoak eta, hortaz, lortzen diren erroreak ez dira ia nabaritzen. Astro guztietatik, errore handienak sor ditzakeen gorputza Merkurio da, planeten artean orbita eszentrikoena duena $e = 0.205$ eta azkarrena dena. Hala ere, sortzen den errore hau ez da metatzen, egun guztietan 0:00 AM denean $|\mathbf{E}-\mathbf{I}| = 0$ betetzen da eta.

Erabiltzaileak uneko data *YYYY/MM/DD* formatuan ikusten badu ere —2000/02/20 adibidez—, eragiketak T_{sim} aldagaiarekin konputatzen dira. Biekin lan egiteko, konbertsio

funtzio batzuk definitu dira, *frame* guztietan uneko-data lortzeko, eta erabiltzaileak data aldatzen duenean T_{sim} automatikoki eguneratzeko. Data baten T_{sim} konputatzeko uneko data eta hasierako dataren arteko egun kopurua besterik ez da hartu behar kontuan. Kalkuluak zerbait zailtzen dira orduak eta minutuak kontuan hartu behar direnean, erabiltzaileak balio hauek idatzi ditzake eta; baina egun batek 24 ordu eta ordu batek 60 minutu dituela kontuan izatearekin T_{sim} arazo handirik gabe kalkulatu daiteke.

Gorputzen erradioa, errotazio abiadura eta inklinazioaren konstanteak aurretik definitu dira. Errotazio-abiadura garrantzitsua da; izan ere, v_{sim}-en arabera astroen errotazioa azkartu ala moteldu behar da. Horregatik, kalkulu hau erraz egiteko, gorputzen rot_i errotazioa $rad/egun$ unitatea erabiliz definitu da. Beraz, simulazioaren i. gorputzaren $rot_{i_{sim}}$ errotazio-abiadura 4.4 ekuazio bidez definitzen da.

$$rot_{i_{sim}} = rot_i \cdot v_{sim} \tag{4.4}$$

Azkenik, aipatzekoa da simulatu daitezkeen egunak bukatzen direnean, hots, bukaerako-data pasatzerakoan, $v_{sim} = 0$ ezartzen dela erroreak ekiditeko, simulazioa gelditzen delarik. Hala ere, erabiltzaileak uneko data eta v_{sim} aldatu ditzake simulazioa berriz habian jartzeko.

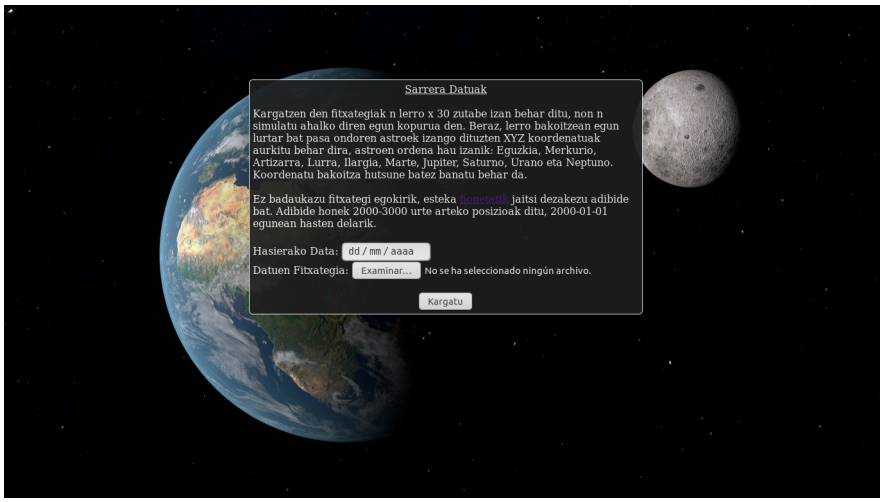
4.3 Funtzionalitatea

Sekzio honetan simulagailuaren gida txiki bat egiten da, honek dituen funtzionalitateak azalduz. Simulagailuak *Mozilla Firefox* web-nabigatzailean arazorik gabe funtzionatzen du, baina beste nabigatzaileetan ez da web-orri honen funtzionamendu zuzena ziurtatzen.

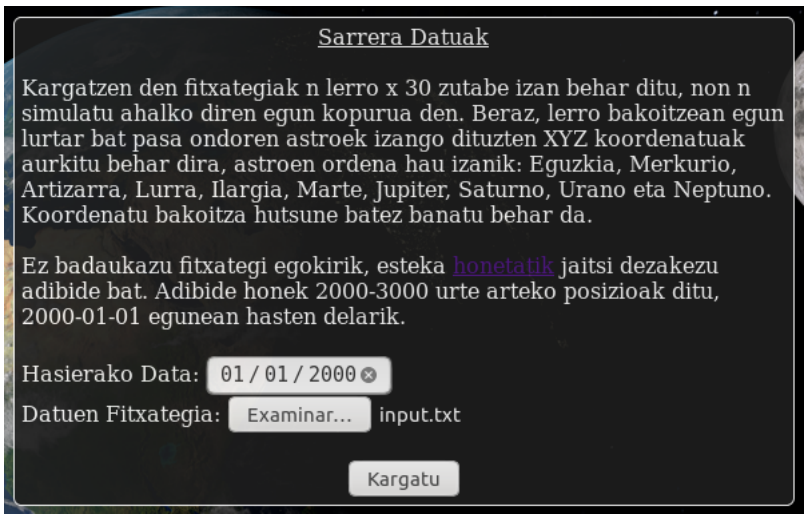
Hasiera batean, sarrerako orri batek erabiltzailea agurtzen du, sartu behar diren sarrerako datuak nola definitu behar diren azalduz eta datu horiek kargatzeko baliabideak emanez —4.13 irudia—. Sarrerako datuak behin sartu direnean kargatu botoiari klikatu behar zaio simulazioa habian jartzeko —4.14 irudia—.

Simulazioa kargatzean, hainbat elementu berri aurki daitezke —4.15 irudia—.

4.15 irudiaren goi-ezkerraldean laukizuzen urdin txiki bat azaltzen da, simulazioaren *FPS* kontadorea dena. Eskuinaldean, berriz, simulazioaren kontrolak azaltzen dira, simulazioaren hainbat elementu kendu/jarri, simulazioaren abiadura aldatu eta astroen tamaina aldatzeko erabili daitekeena. Eszenaren zentroan, koadrikula bat azaltzen da, koadrikularen



4.13 Irudia: Simulagailuaren sarrera orria.



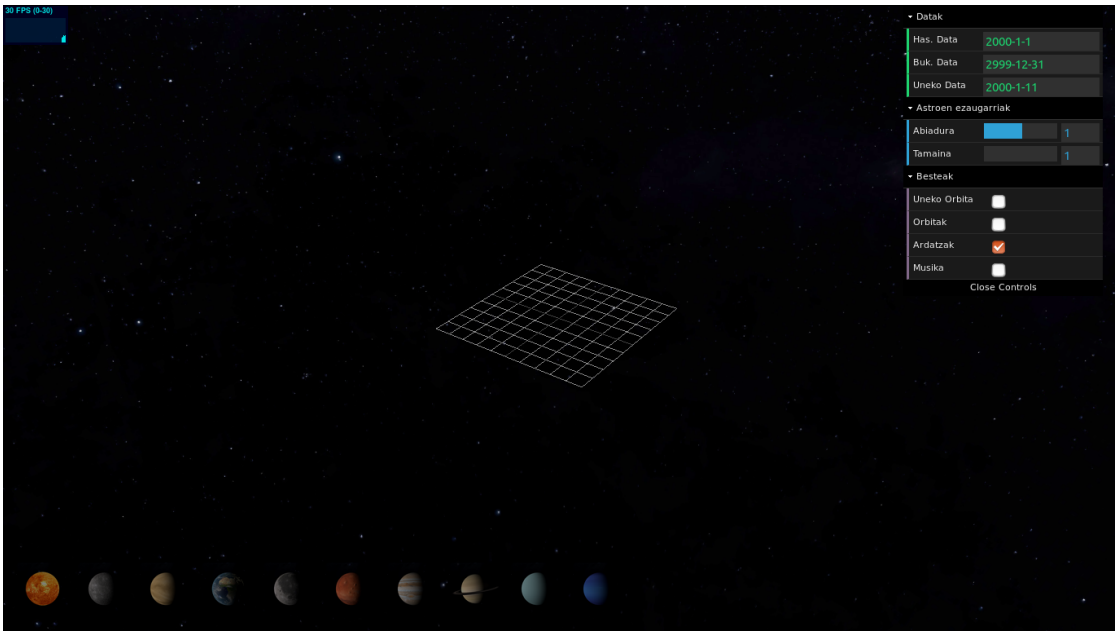
4.14 Irudia: Simulagailuaren sarrera orriko sarrera-datuak sartzen.

gelaxkaren alde bakoitzak unitate astronomiko bateko luzera duelarik. Azkenik, beheezkerraldeko izkinean, astroen ikono batzuk azaltzen dira, bati klikatzean gorputz horren posizioa azaltzen delarik —4.16 irudian adibidea—.

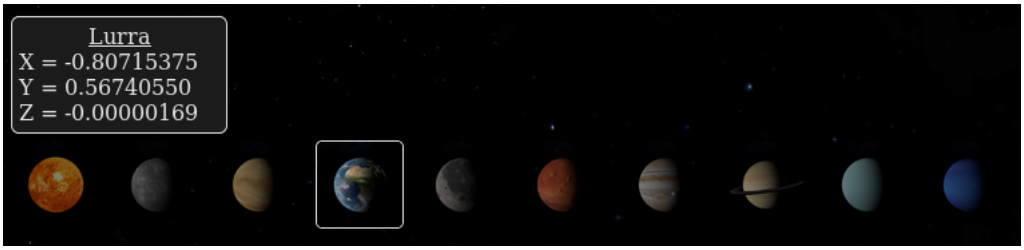
Simulazioaren kontroletan, erabiltzaileak hainbat gauza aldatu ditzake —4.17 irudia—.

Kontrol-panelaren ordena jarraituz, honako kontrol edo informazioa azaltzen da:

- *Hasierako data*: simulazioa hasten den data da. Erabiltzaileak sartzen duen sarrerako datu bat da eta simulazio osoan zehar konstante mantentzen da.
- *Bukaerako data*: simulazioa bukatuko den data da. Hasierako datarekin batera kons-



4.15 Irudia: Simulazioaren eszena nagusia.

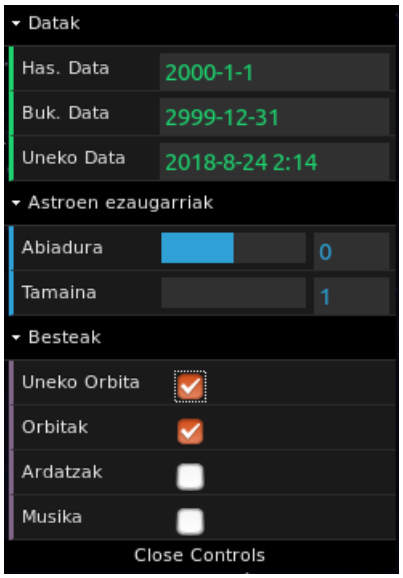


4.16 Irudia: Simulazioko Lurraren kokapena Unitate Astronomikoetan.

tantea da, eta simulazioa data honetara iristean, simulazioaren abiadura 0-ra eslei-tuko da automatikoki.

- *Uneko data*: simulazioaren uneko data, erabiltzaileak edozein momentutan eskuz aldatu dezakeena. Uneko data honako formatua jarraituz aldatu daiteke bakarrik: yyyy-mm-dd hh:mm. Orduak eta minutuak —hh:mm— definitzea hautazkoa da. Bestela, aldaketa ez da burutuko. Gainera, uneko data hasierako eta bukaerako daten arteko data bat ez bada, uneko data ez da aldatuko eta simulazioak aurrera jarraituko du, esleituta dagoen v_{sim} abiadurarekin.

- *Abiadura*: simulazioaren uneko v_{sim} abiadura da, segundo batean pasatzen diren egun lurtar kopurua definitzen duelarik. Defektuz, segundo batean egun lurtar bat simulatuko da. $v_{sim} < 0$ bada, denboran atzera higituko dira astroak; eta $v_{sim} = 0$ betetzen bada, simulazioa gelditu egingo da, *pause* tekla batek egingo lukeen beza-



4.17 Irudia: Simulazioaren kontrol-panela.

la. Simulazioaren abiadura $-20 \leq v_{sim} \leq 20$ artean mantenduko da beti, simulazioa azkarregi joan ez dadin.

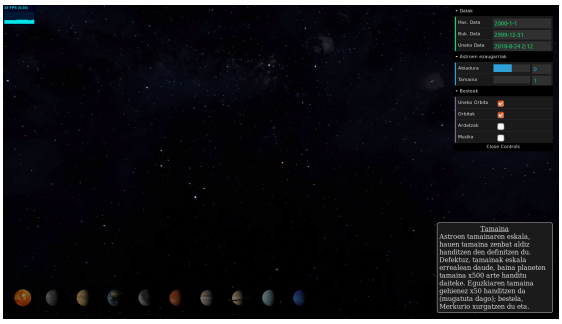
- *Astroen tamaina*: Astroen erradioaren tamaina definitzen du. Defektuz, Eguzki-sistemaren eskala erreala mantentzen du 1 balioarekin, baina gorputzen tamaina 500 aldiz arte handitu daiteke. Hala ere, Eguzkiaren tamaina 50 aldiz bakarrik handitu daiteke, kontrakoan barne planetak xurgatzen ditu eta. Aldaketak nabariak direla ikusteko 4.18 irudi-multzoa begiratu.

- *Uneko orbita*: Simulazioa exekutatzen ari den une bereko astroen posizioak azaltzen dira aukera hau hautatzen bada. Honi sakatzean, simulazioaren abiadura automatikoki $v_{sim} = \frac{1}{7}86400$ baliora ezartzen da, astroek bizitza errealeko kokapen eta abiadura berdinak izanik.

- *Orbitak*: Hau klikatzean astroen orbitak lerroekin irudikatzen hasten da, 4.19 irudian ikusi daitekeen lerroak agertzen direlarik.

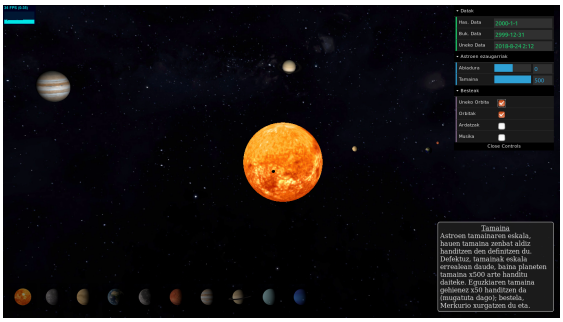
- *Ardatzak*: Astroen ardatzak azaltzen dira aukera hau hautatzean, gorputzen bi poloak gurutzatzen dituena. Aukera berarekin 4.7 irudiko koadrikula eraikitzen da ere.

- *Musika*: Azken aukera honekin simulazioari laguntzen dion *copyright* gabeko musika entzuten hasten da, v_{sim} abiaduraren arabera entzuten den abestia aldatzen delarik.



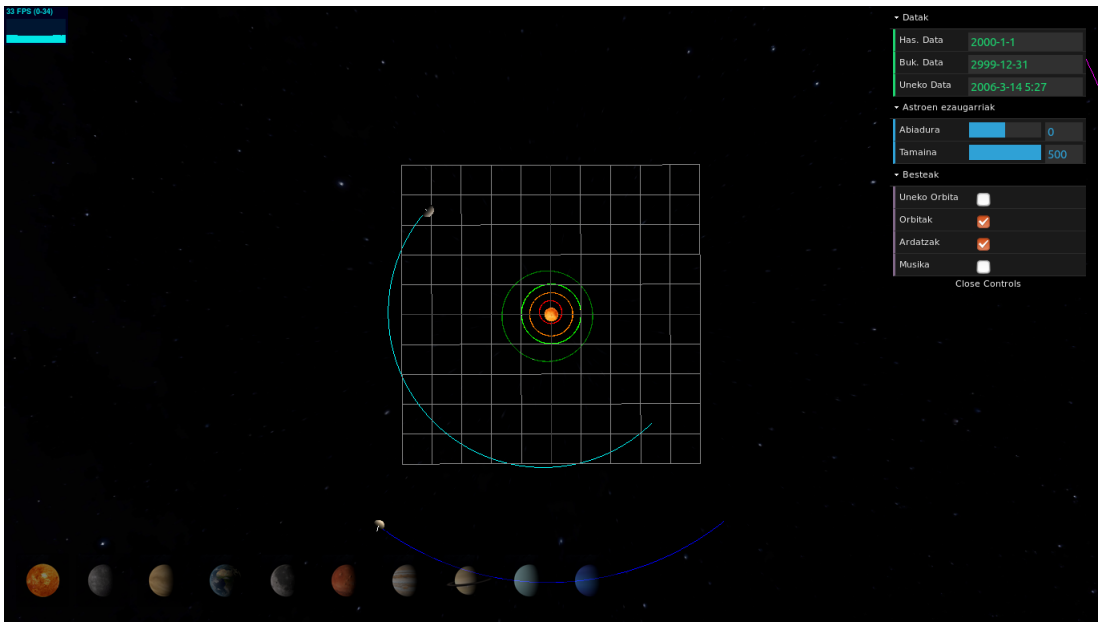
(a) Eguzki-sistemaren eskala erreala

(b) Eguzki-sistemaren astroak x50 handiagoak



(c) Eguzki-sistemaren astroak x500 handiagoak

4.18 Irudia: Simulazioaren eskala ezberdinak azaltzen dira irudi-multzo honetan. 4.18a irudian Eguzkia bakarrik ikusi daiteke, Eguzki-sistemaren eskala erreala mantentzen baita. Astroen erradioa 50 aldiz handituz, Eguzkiak irudiaren zati nabarmen bat hartzen du, Jupiter irudiaren ezkerraldean agertzen delarik tamaina hutsal batekin —4.18b irudia—. Azkenik, 4.18c irudian, Eguzkiaren tamaina bera mantenduz, barne-planetak, Jupiter eta Saturno agertzen dira. Merkurio Eguzkiaren aurrean azaltzen da; Artizarra, Lurra eta Marte kontrolen ezkerrean azaltzen dira, Jupiter 4.18b irudiko leku berean aurkitzen da eta Saturno irudiaren goikaldean ikus daiteke.



4.19 Irudia: Simulazioaren astroen orbitak.

5. KAPITULUA

Lortutako emaitzak

Keplerren fluxua eta N-gorputzen problema ebazten duen algoritmoa —hemendik aurrera N-gorputzen fluxua deituko dena— Juliako pakete batean sartu dira. Pakete hori *KeplerFlow* deitzen da eta GitHub-en aurkitu daiteke —bere dokumentazioarekin batera—, edozeinek deskargatu dezakeelarik honako komandoa exekutatuz.

```
(v1.0) pkg> add https://github.com/salanueva/KeplerFlow.jl
```

Pakete hau Juliako 1.0 bertsioan ondo funtzionatzeko prestatu da, eta hau erabiltzen hasteko, Juliari pakete hau erabiliko dela esatea ez da ahaztu behar:

```
julia> using KeplerFlow
```

Behin *KeplerFlow* paketea erabiltzeko prest dagoelarik, algoritmoen azkartasun eta doitasunak aztertzeko ordua iritsi da. Algoritmoen ezaugarri hauek aztertzeko hainbat proba egin dira.

Doitasuna analizatzeko ezin dira lortutako posizio eta abiadura bektoreak datu errealekin konparatu, problema hauen soluzio zehatza ezin baita konputatu. Posizio eta abiadura bektoreak erabili beharrean, aztertutako sistemen energia errore erlatiboak begiratuko dira. Izan ere, N-gorputzeko sistema itxi batean, energia zinetiko eta potentzialen batura ez da aldatzen, hots, energia kontserbatzen da —kanpo indarrak kontuan hartzen ez dira eta—.

Problemak definitzen dituzten sistema Hamiltondarren ekuazioek E sistemen energiak definitzen dituzte. Keplerren fluxuaren kasuan, ebazten den sistemaren Hamiltondarra 2.1 ekuazioak definitzen du. Hori dela eta, E_{Kepler} Hamiltondar horrek definitzen duen sistemaren energia bada, 5.1 ekuazioko berdintza beteko da.

$$E_{\text{Kepler}} = \mathcal{H}_{\text{Kepler}} = \frac{1}{2}\mathbf{v}^2 - \frac{\mu}{|\mathbf{r}|} \tag{5.1}$$

N-gorputzen fluxuan, berriz, sistema Hamiltondarra 3.10 ekuazioak definitzen du. $E_{\text{N-body}}$ energia kalkulatzeko, ekuazio hori sinplifikatu daiteke gai batzuk beraien artean deusezta tuz —5.2 ekuazioa—.

$$E_{\text{N-body}} = \mathcal{H}_{\text{N-body}} = \sum_{i=1}^{N} \frac{\mathbf{p}_i'^2}{2m_i'} - \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_i m_j}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{5.2}$$

Sistemaren t uneko $E(t)$ energiaren E_{err} errore erlatiboa E_0 energia teorikoarekiko 5.3 ekuazio bidez kalkulatu da, zeinua kontuan hartzen ez delarik.

$$E_{\text{err}} = \frac{|E(t) - E_0|}{|E_0|} \tag{5.3}$$

5.1 Keplerren fluxua

Keplerren problemaren algoritmoarekin hasiz, algoritmo honi ahal diren hobekuntza gehienak aplikatzea izan da helburuetako bat. Izan ere, algoritmo hau iterazio luzeetan zehar exekutatu behar izaten da; eta, hortaz, Keplerren fluxua azkarra izatea funtsezkoa da. Keplerren fluxuaren sekzio honetan agertzen diren denborak 10000 exekuzioetan zehar lortu diren denboren medianak dira.

5.1.1 C vs Julia

Alde batetik, *KeplerFlow* paketea Julia lengoaiarako idatzi bada ere, programazio-lengoaia honetan C-kodea konpilatu eta exekutatu daiteke kanpo-liburutegirik erabili gabe. Honek, Kepler-en fluxua C lengoaian kodetzea eta kode hori eraikitako paketera gehitzea ahalbidetu du, hauen arteko konparaketa egiteko gauzak erraztuz.

Beste aldetik, *SatelliteToolbox* sateliteen higidurak simulatzeko liburutegian, A.4 sekzioan azaltzen den prozedura jarraitzen da Kepler-en problema ebazteko. Hori dela eta, bi algoritmoen arteko ezberdintasunak aztertu daitezke.

	Kepler (C)	**Kepler (Julia)**	**Kepler + Kahan**	**SatelliteToolbox**
64 bit	450,727ns	769,824ns	806,378ns	1,184μs
128 bit	————-	55,508μs	59,834μs	182,042μs

5.1 Taula: Keplerren problema ebazteko behar izan diren denborak azaltzen dira —denbora-pauso bat exekutatuz bakarrik—, bai 64 bit-eko koma higikorreko aritmetikarekin, eta baita 128 bit-eko datu motarekin ere. Algoritmoak ondorengoak dira, hurrenez hurren: Keplerren fluxua C lengoaian kodetuta, Keplerren fluxua Julian, Keplerren fluxua Kahanen algoritmoa erabilita Julia-n eta *SatelliteToolbox* paketeak dakarren algoritmo tradizionala.

	Kepler (C)	**Kepler (Julia)**	**SatelliteToolbox**	***ε-mach***
64 bit	0,0	2,67E-16	2,67E-16	2,22E-16
128 bit	————-	3,53E-39	8,84E-19	5,88E-39

5.2 Taula: 5.1 taulako algoritmoek egin dituzten energiaren errore erlatiboak azaltzen dira. Denbora-pauso bat exekutatzen denez, Keplerren fluxuak errore erlatibo bera izaten du Kahanen algoritmoa erabiltzen bada ere, soluzio berak lortzen dira eta. Konparaketa egin ahal izateko, datu mota bakoitzaren ε-*mach* balioa ere azaltzen da.

5.1.2 Newton-Raphsonen metodoaren gelditze-irizpidea

2.2.2 sekzioan, Keplerren fluxua hobetzeko hainbat ideia azaltzen dira. Ideia horietan, Newton-Raphsonen metodoarentzat hainbat gelditze-irizpide probatu direla aipatzen da:

- *Balioen konparaketa*: zenbakizko metodoaren azken hiru iterazioen emaitzak konparatzen dira, eta metodoa ez amaitzen hiru iterazio horien bi emaitza berdinak izan arte.
- *Erroreen konparaketa*: iterazio bakoitzeko emaitzen aldaketak aztertzen dira, eta aldaketa horiek txikitzen diren bitartean metodoaren iterazio berriak konputatzen jarraitzen da.
- *Tolerantzia zuhurra*: iterazio bakoitzeko emaitzen aldaketak tolerantzia batekin konparatzen dira, tolerantzia emaitzen aldaketa baino handiagoa den arte. Tolerantzia erabiltzen ari den datu-motaren ε-mach-en araberakoa da.

Hiru geldize-irizpide horiek Keplerren fluxuaren Juliako inplementazioan probatu dira, Kahanen batuketa konpentsatua erabili gabe.

	Balioen konparaketa	Erroreen konparaketa	Tolerantzia zuhurra
64 bit	1,026μs	898,767ns	769,824ns
128 bit	70,350µs	71,748µs	55,508μs

5.3 Taula: 2.2.2 sekzioan aipatu diren gelditze-irizpideen arteko Keplerren fluxuaren denborak dira, 64 eta 128 bit-eko koma higikorreko aritmetikarekin. Probetarako erabili diren Kepler-en fluxuaren bertsioetan ez da Kahanen algoritmoa erabili.

	Balioen konparaketa	Erroreen konparaketa	Tolerantzia zuhurra
64 bit	2,27E-15	2,27E-15	2,67E-16
128 bit	2,82E-38	2,82E-38	3,53E-39

5.4 Taula: 2.2.2 sekzioan aipatu diren gelditze-irizpideen arteko energiaren errore erlatiboak dira, 64 eta 128 bit-eko koma higikorreko aritmetikarekin. Probetarako erabili diren Keplerren fluxuaren bertsioetan ez da Kahanen algoritmoa erabili.

5.2 N-gorputzen fluxua

N-gorputzen problema konplexuagoa da. Algoritmoa exekutatzeak denbora gehiago dakarrenez, sekzio honetan agertzen diren denborak exekuzio batean lortutako denborak dira —hainbat exekuzio konparatu beharrean—. Denboren konsistentzia galtzen bada ere, denbora hauek algoritmoaren eraginkortasunaren ideia nagusi bat ematen dute.

5.2.1 Datu-mota ezberdinak

N-gorputzen fluxuaren azkartasuna aztertzeko, lau datu mota ezberdin erabili dira: *double* (64 bit), *bigfloat* (96 bit), *bigfloat* (128 bit) eta *bigfloat* (256 bit). 5, 10 eta 15 astroetako sistemak erabili dira —Eguzki-sistemako astro errealak direlarik—, astroen kopuruaren arabera denborak nola aldatzen diren aztertzeko.

Sistema eta datu mota konbinazio bakoitzetik, 5 exekuzio desberdin egin dira. Bostetan 100 urte lurtar simulatzen dira, urtero *output* bat kalkulatuz; baina pausu bakoitzean denbora-tarte ezberdinak aukeratuz —365, 5, 1, 0.5 eta 0.1 egunekoak—, iterazio kopuru ezberdinekin lortutako emaitzak alderatzeko.

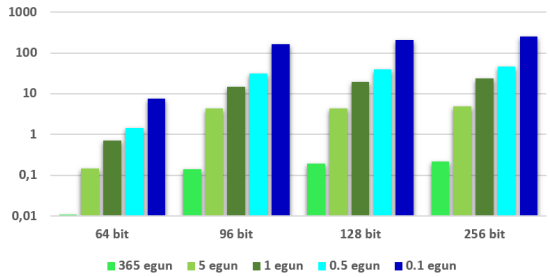
Hiru N-gorputzeko sistema ezberdin ebatzi dira probetan zehar:

1. *Barne Eguzki-sistema*: Bost gorputzez osatuta: Eguzkia, Merkurio, Artizarra, Lurra eta Marte. Eguzkia beste astroekiko konparatuta oso masiboa da, eta beste lau planeten orbiten eszentrikotasunak txikiak dira, $e \leq 0.2$.

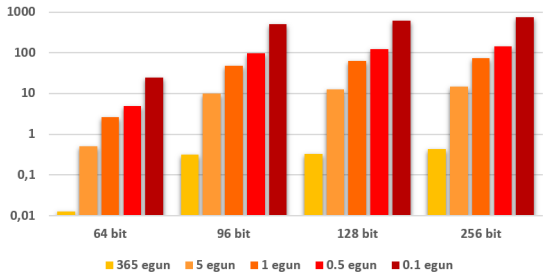
2. *Eguzki-sistemaren astro nagusiak*: Barne Eguzki-sistemari Jupiter, Saturno, Urano, Neptuno eta Pluton gehituz lortzen den sistema. Lehenengo laurak eszentrikotasun oso txikia eta barne Eguzki-sistemako planetak baino masiboak dira. Pluton, berriz, txikia eta eszentrikotasun handiagokoa da, 10 astro izateko sartu delarik.

3. *Eguzki-sistemaren astro nagusiak + objektu txikiak*: Hamar gorputzeko sisteman Ceres, Vesta, Halley kometa, Eris eta Haumea gorputzak gehitu dira. Bostek masa oso txikia dute, eta, gainera, hauen orbitek $e \leq 0.85$ inguruko eszentrikotasuna dute.

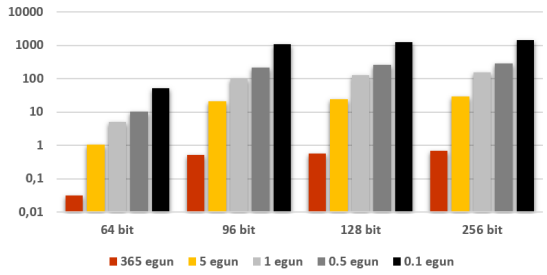
Sekzio honetako grafikoetan, irakurmena errazteko denbora-pauso eta sistemen arteko 15 konbinazioei kolore jakin batzuk esleitu zaizkie.



(a) Exekuzio-denborak 5 gorputzekin



(b) Exekuzio-denborak 10 gorputzekin



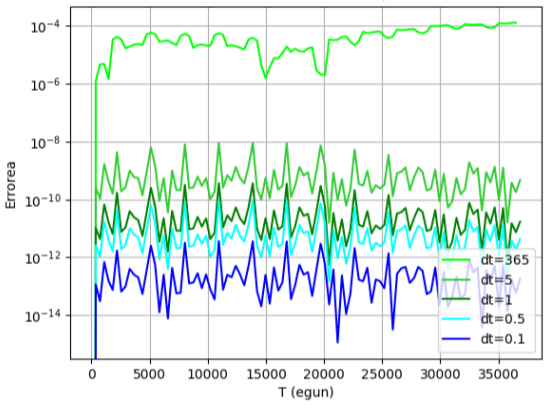
(c) Exekuzio-denborak 15 gorputzekin

5.1 Irudia: Grafiko bakoitzean, kolore ezberdin bakoitzak sistema-planetario bakoitzaren 100 urteko simulazioetan erabili diren denbora-pauso ezberdinak definitzen dituzte. Denborak eskala logaritmikoan azaltzen dira, hauen unitateak segundoak izanik. Denboren barra hauek datu-motaren arabera multzokatu dira, eta grafiko bakoitzean sistema ezberdin bat ebazten da.

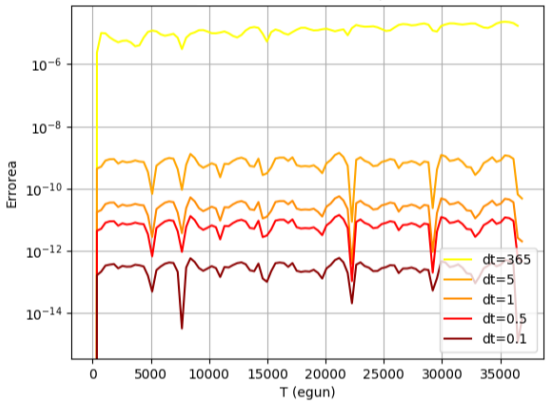
5.1 irudi multzoko *double* edo 64 bit-eko koma higikorreko aritmetika erabiliz lortu diren emaitzekin, energiaren errore erlatiboak kalkulatu dira —5.2 irudi multzoa—.

5.2.2 Kahanen batuketa konpensatua

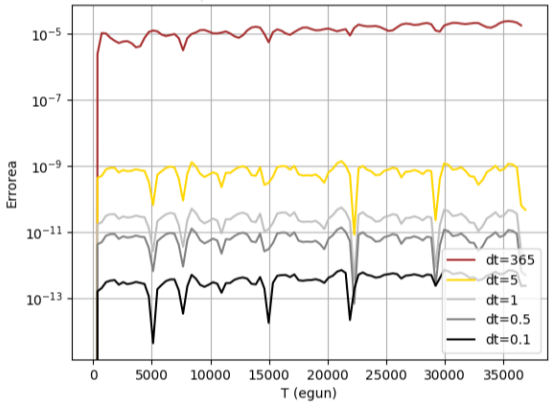
Kahanen algoritmoa N-gorputzen fluxua hainbat unetan erabiltzen da, exekuzioko datu-motaren doitasunaren arabera. Alde batetik, $\mathcal{H}_{\text{Kepler}}$ garatzean, Keplerren fluxuan aplika-



(a) Energiaren errore erlatiboak 5 gorputzekin



(b) Energiaren errore erlatiboak 10 gorputzekin



(c) Energiaren errore erlatiboak 15 gorputzekin

5.2 Irudia: Grafiko bakoitzean, 5.1 irudietako 64 bit-eko exekuzioen *output*etan lortzen diren energiaren errore erlatiboak azaltzen dira. Beraz, grafikoetako lerro bakoitzean denbora-pauso jakin bat erabiliz lortu diren energia errore erlatiboak azaltzen dira.

tzen da; eta bestetik, *Kick*-pausoan abiadura eguneratzean. Hortaz, N-gorputzen fluxua iterazio askotan zehar exekutatu behar denean, biribiltze-errorearen informazio-galera ahal den heinean ekidingo da. Dena den, honek algoritmoa mantsotzen du.

Horregatik, N-gorputzen fluxua moldatu da, Kahanen algoritmorik gabe zer nolako emaitzak eta denborak lortzen diren ikusteko. Kalkulu hauek 64 bit-eko datu motekin egin dira.

5.3 irudi-multzoan, N-gorputzen fluxuaren bi bertsio ezberdinetan lortutako emaitzak konparatzen dira. Kasu honetan, gorputzen **r** posizio eta **v** abiadura bektoreak kontuan hartu dira, eta energia ez bezala, hainbat baliok osatzen dituzte bektore hauek. Beraz, bektore horiek errepresentatzen dituzten x_r eta x_v ordezko bi balioak lortzeko, 5.4 ekua-

zioko kalkuluak egin dira, non N sistema horren gorputz kopurua den:

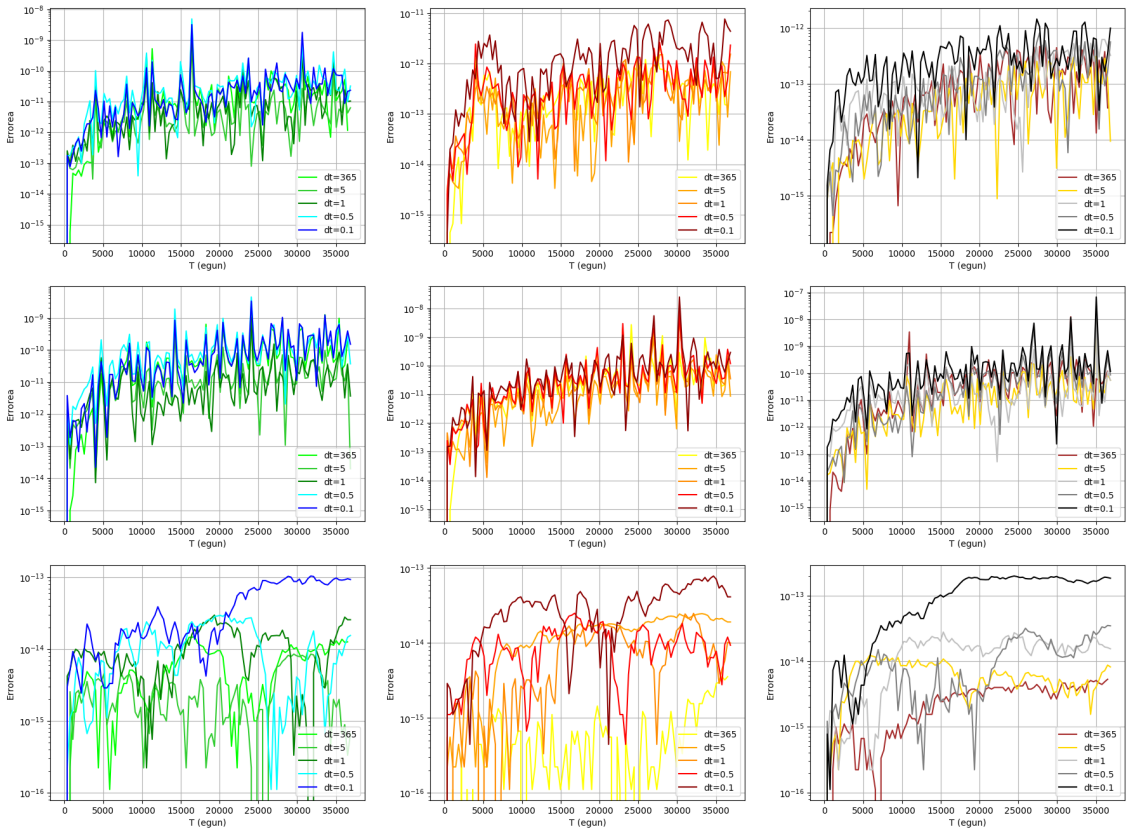
$$x_r = \frac{\sum_{i=1}^{N} |\mathbf{r}_i|}{N} \tag{5.4a}$$

$$x_v = \frac{\sum_{i=1}^{N} |\mathbf{v}_i|}{N} \tag{5.4b}$$

Azkenik, 5.3 irudi-multzoan azaltzen diren balioak algoritmoen bi bertsioen arteko energia, x_r eta x_v-ren energiaren errore erlatiboak dira, 5.3 ekuazioan bezala konputatuta.

	365 egun	**5 egun**	**1 egun**	**0.5 egun**	**0.1 egun**
5 Gorputz	38,24%	7,82%	-0,62%	7,03%	12,40%
10 Gorputz	10,88%	21,57%	22,01%	19,35%	20,83%
15 Gorputz	22,74%	21,36%	20,33%	17,74%	15,52%

5.5 Taula: Taula honetan, 64 bit-eko aritmetika erabiliz, Kahanen algoritmoa erabili gabe N-gorputzen fluxuaren denboren hobekuntzen portzentaiak azaltzen dira. Zutabeetan erabilitako denbora-pauso ezberdinak azaltzen dira, eta ilaretan, berriz, ebatzi den sistemaren astro kopurua. Portzentaia positibo batek Kahanen algoritmoa erabili gabe denbora aurreztu dela esan nahi du.



5.3 Irudia: Irudi multzo honetan, 64 bit-eko aritmetika erabiliz lortu diren posizio, abiadura eta energien arteko diferentziak azaltzen dira. Alde batetik, N-gorputzeko fluxuan Kahanen algoritmoa erabiltzen duen bertsioaren emaitzak, eta bestetik, Kahanen algoritmoa erabiltzen ez duen bertsioaren emaitzak lortu dira. Bi emaitza hauen desberdintasunak grafiko hauetan azaltzen dira. Lehenengo zutabean, bost gorputzek osatzen duten sisteman lortu diren emaitzak azaltzen dira; bigarrenean hamar gorputzena eta hirugarrenean, berriz, hamabost gorputzena. Lerro bakoitzean, konparatzen ari den datu bat azaltzen da: x_r, x_v eta energia balioen errore erlatiboak, hurrenez hurren.

6. KAPITULUA

Ondorioak

Lortu diren emaitzak bi zatitan aztertuko dira, algoritmo bakoitza bi atal horietan aztertuz. Kontuan izan behar da, lortutako denbora eta errore erlatiboak argitaratu den paketeko kodea exekutatuz lortu dela. Pakete hau gama erdiko ordenagailu batean exekutatu da, honako ezaugarriak dituenak:

- Prozesadorea: Intel® Core™ i5-4210U CPU @ 1.70GHz × 4
- RAM memoria: 8 GB
- Sistema eragilea: Ubuntu 16.04 64-bit

6.1 Keplerren fluxua

Keplerren fluxuaren hiru bertsio aurki daitezke paketean. Lehena C lengoaian kodetuta dago eta ez du Kahanen algoritmoa erabiltzen. Bigarrenean, Julia lengoaia erabiliz kodetu da, lehenengoaren parea dena. Azkenik, hirugarren bertsioak Kahanen algoritmoa inplementatzen du, Julian inplementatuta dagoelarik. 5.1 taulan hiru bertsioen arteko denborak antzeman daitezke. C lengoaia azkarragoa dela ikus badaiteke ere —kode bera %41,45 azkarrago exekutatu du—, Julian sarrerako datuen formatua errespetatzen den bitartean, kode bera koma higikorreko aritmetika ezberdinekin erabili daiteke. Gainera, Julia-n kodea garatzea azkarragoa izan da, abstrakzio maila altuagoak erabiltzen ditu eta. C lengoaiako bertsioa 64 bit-eko aritmetika erabiltzeko prestatu da.

Algoritmo honen exekuzioak azkarra izan behar du, sistema konplexuagoetan askotan ebatzi behar baita. Azkartasun hori aztertzeko algoritmoaren exekuzio oso batek zenbat irauten duen aztertu da 5.1 taulan.

Esan bezala, taula horretan, kodetutako algoritmoaren hainbat bariante azaltzen dira, baita *SatelliteToolbox* paketeak dakarren Keplerren ekuazio tradizionala ebazten duen algoritmoa ere.

Beste paketeak erabiltzen duen algoritmoa motelagoa da, elementu orbitalekin ebazten duelako ekuazioa, eta, ondoren, elementu orbital horiek posizio eta abiadura bektoreetan bihurtzen dituelako —azkeneko pauso hau konputazionalki garestia izanik—. *KeplerFLow* paketean kodetutako algoritmoa, berriz, azkarragoa dela ikus daiteke, zuzenean posizio eta abiadura bektoreekin lan egiten duelako.

Gainera, aipatzekoa da *SatelliteToolbox* paketeak orbita eliptikoak bakarrik ebatzi ditzakeela, hots, $e < 1$ eszentrikotasunak dituzten orbitak. Izan ere, Kepler-en ekuazio tradizionala orbita koniko motaren arabera aldatzen da. Ezagunena den bertsioa A.4 sekzioan azaltzen dena bada ere —orbita eliptikoena—, 6.1 ekuazioetan orbita eliptiko eta hiperbolikoenak agertzen dira, hurrenez hurren —$e = 1$ duten orbitak *Barker*ren metodoaren[1] bidez ebatzi daitezke zuzenean—.

$$M = E - e\sin E \tag{6.1a}$$

$$M = e\sin H - H \tag{6.1b}$$

6.1 ekuazioetan, M batezbesteko anomalia, E anomalia eliptiko eszentrikoa eta H anomalia hiperboliko eszentrikoa dira. Azkeneko biak modu ezberdin batean kalkulatzen dira, eta, *SatelliteToolbox* paketean E da kalkulatu daitekeen anomalia eszentriko bakarra. Keplerren fluxuak χ anomalia unibertsala erabiltzen duenez, edozein orbita mota ebatzi dezake.

Bestalde, Kahanen algoritmoa gehitzeak ez du algoritmoa askoz motelagoa bihurtzen. Algoritmoaren exekuzio batean lortzen diren emaitzak berdinak badira ere, hainbat exekuzio jarrai egitean informazio galera asko txikituko litzateke.

5.2 taulan, 5.1 taulako algoritmoek lortu dituzten energiaren errore erlatiboak azaltzen dira. Kontuan izan behar da, algoritmoaren exekuzio baterako Kahanen algoritmoak ez

[1] *Barkerren metodoa*: orbita parabolikoan dagoen gorputzaren posizioa kalkulatzeko erabiltzen den funtzioa, denboraren arabera gorputzaren posizio zehatza definitzen duelarik.

duela datuetan hobekuntzarik eragiten —galtzen den informazioa beste aldagai batean gordetzen du bakarrik—. Horregatik, Kahanen algoritmoa erabiltzen ez duen bertsioaren errore berdinak lortzen ditu. Bestela, ε-mach balioaren inguruko errore erlatiboak lortzen dira, *SatelliteToolbox* paketean *BigFloat*ak erabiltzen direnean izan ezik. Pakete hau 64 bit-eko aritmetika erabiltzeko dago prestatuta, hots, funtzioetan egiten diren kalkuluak aritmetika horrekin egiten dira; bit kopuru handiagoko aritmetikak erabiltzeak onurarik ematen ez duelarik.

Gainera, C eta Julia lengoaien artean errore erlatiboak ezberdinak dira. Berez, honek ez luke gertatu beharko, baina emaitzak aztertuz abiadura bektorearen bigarren koordenatuan azkeneko digitua ezberdina dutela antzeman da. Ez dago garbi zergatik gertatzen den hau, bietan Newton-Raphsonen metodoak iterazio kopuru bera eman direlako, baina lengoaia bakoitzak koma higikorreko aritmetikaren biribilketak ezberdin egiten dituztela da litekeena.

Emaitza hauei erreparatuz, alde batetik, oraingoz 64 bit edo *double* zehaztasuna duten datu-motak erabiltzea gomendatzen da. Izan ere, gaur egungo ordenagailuen *hardware*-ak 64 bit-ekin lan egiten dute, 64 bit-eko datuen arteko operazioak *hardware* bidez optimizatuta daudelarik. Bit gehiago dituzten datuekin lan egitea ez dago hain optimizatuta, eta horrek exekuzio-denbora luzeak lortuko direla esan nahi du, Newtonen fluxuaren kasuan aztertuko den bezala.

Beste aldetik, Julia eta C lengoaien arteko exekuzio-denboretan ezberdintasun nabarmenak lortzen badira ere —C azkarragoa izanik—, Juliako kodea nahi den datu-motarekin berrerabili daiteke. Gainera, lengoaia honetan kodea garatzea azkarragoa eta erroreak antzematea errazagoa izan da, abstrakzio handiagoko lengoaia da eta.

Newton-Raphsonen gelditze irizpidea

2. kapituluan zehar, zenbakizko metodoan hainbat gelditze-irizpide probatu direla esan da. Hiruetatik, azkarrena tolerantzia zuhurraren irizpidea dela antzeman daiteke 5.3 taulan, bai 64 bit-eko aritmetikarekin eta baita 128-koekin ere. Ezberdintasuna handia da, metodoan hainbat begizta aurrezten direlako, beste bi irizpideekin konparatuz behintzat. Gainera, aurrezten diren iterazioak beharrezkoak ez direla ikus daiteke 5.4 taulan, energiaren errore erlatiboetan ez baita aldaketa nabarmenik ikusten —kasu honetan hobeagoak izanik—.

Gelditze irizpide hau aukeratzeak Keplerren eta N-gorputzen fluxuak azkartu ditu, erabiltzen den aritmetikaren arabera lortzen den zehaztasuna galdu gabe. Sekzio honetan bi aritmetika ezberdin azaldu badira ere —N-gorputzen fluxuan lau erabili dira—, zehaz-

tasun ezberdineko beste hainbat aritmetika probatu dira; baina lortzen diren emaitzek ez dituzte algoritmoari buruzko informazio gehiago erakusten, N-gorputzen fluxuan ez bezala.

6.2 N-gorputzen fluxua

Keplerren fluxuan ez bezala, N-gorputzen fluxuaren exekuzio batean hainbat denbora-pauso exekutatu dira. Aurreko sekzioan, Keplerren problema ebazteko beharrezko denbora eta erroreak aztertu nahi izan dira; baina, oraingo honetan, N-gorputzen problema hainbat denbora-pausoetan zehar exekutatuko da. Honen arrazoi nagusia, algoritmo honen iterazio kopurua aldatzeak eta datu mota ezberdinak erabiltzeak dakarren denbora eta energiaren errore erlatiboen aldaketak aztertzea da. Bide batez, N-gorputzaren fluxuaren begizta batean Keplerren fluxua hainbat aldiz exekutatzen denez, exekuzio luzeetan zehar energia errore erlatiboetan eragina duen aztertuko da.

5 kapituluan esan bezala, $N = 5$, $N = 10$ eta $N = 15$ gorputzeko sistemak ebatzi dira. Sistema bakoitza koma higikorreko lau datu mota ezberdin erabiliz ebatzi da, eta kasu guztietan, 64 bit-eko datu moten eragiketak askoz azkarrago burutu dira. Keplerren fluxuan berdina gertatzen da, *hardware*a 64 bit-eko datuekin lan egiteko prestatuta daude, eta bit gehiagoko eragiketak burutzeko *software*aren laguntza behar da —kalkuluak mantsotuz—.

Hala ere, 96, 128 eta 256 bit-eko datuekin lan egitean lortzen diren exekuzio-denborak ia berdinak dira. Igoera bat nabaritzen da, baina aldaketa txiki hauek ezin dira 64 bit-eko datuen denborekin konparatu. Energiaren errore erlatiboekin ezberdintasun handiak baleude, *BigFloat* motako zehaztasuna duten datuak erabiltzeak merezi izango luke; baina aurrerago ikusiko denez, hori ez da kasua.

5.1 irudi multzoan esandakoa garbi ikus daiteke. Gainera, sistema eta datu mota bakoitzeko denbora-pauso ezberdinak erabili dira, simulatzen den T_{tot} denbora totala berdin mantenduz —$T_{tot} = 36524\text{egun} \approx 100\text{urte}$—. Koordenatu kartesiarretan kalkulatzen diren emaitzak, hots, *output*-ak, urtero kalkulatu dira kasu guztietan. Beraz, denbora-pausoa 5 aldiz txikitzeak algoritmoaren begizta kopurua 5 aldiz handitzea dakar.

Hori kontuan harturik, zentzua du begizta kopurua 5 aldiz handitzean exekuzio-denborak 5 aldiz handitzen direla. 5.1 irudi-multzoan, edozein datu mota edo sistemetan ikus daiteke hori.

5.2 irudietako energiaren errore erlatiboei erreparatuz, N-gorputzen fluxuak sistemaren energia kontserbatzen duela ikus daiteke, metodo honen propietate bat delarik. Errore erlatibo hauek 5.1 irudietan aztertu diren denboren exekuzioen emaitzetatik lortu dira; eta hauetan, Keplerren fluxua ebazteko erabiltzen den Newton-Raphsonen metodoa orden koadratikokoa dela antzeman daiteke.

Orden koadritikoa duela esateak, errore erlatiboa jaisten den r ratioa denbora-pausoen karratuen proportzionala dela esan nahi du —6.2 Ek.—, non k balio konstante bat den.

$$r = k \cdot dt^2 \tag{6.2}$$

Hortaz, 0.5 eguneko denbora-pauso eta 0.1 eguneko denbora-pausoen artean $\left(\frac{0.5}{0.1}\right)^2 = 25$ aldiz txikitzen da errore erlatiboa. 365 egunekin ez da hori guztiz betetzen. Izan ere, denbora-pauso handia denez —Lurraren orbitaren periodoa—, hasiera batean propietate hau betetzen bada ere, energiaren errorea ez da konstante mantentzen eta pixkanaka handitzen doa, eskala logaritmikoan asko ez bada nabaritzen ere.

Sistema ezberdinen arteko emaitzak aztertzen badira, 10 eta 15 gorputzen sistemen emaitzetan gorabehera txikiagoak lortzen direla antzematen da. Hau, masa handiko gorputzak sartu direlako gertatzen dela suposatzen da. 10 eta 15 gorputzen arteko sistemen energiak oso antzekoak dira; izan ere, bien arteko diferentzia masa txikiko gorputzek osatzen dute, energiaren kalkuluari aldaketa nabarmenik sortu gabe. Beste aldetik, 5 gorputzeko sisteman, barne-planetek beste sistemetan baino askoz gehiago eragiten dute energiaren kalkuluan; eta barne-planeta hauek azkarragoak eta orbita txikiagoak izanik, energiaren kalkuluan oszilazio gehiago aurkitzen dira.

Kahanen batuketa konpentsatuaren eragina

N-gorputzen fluxuan, Kahanen algoritmoa zehaztasun ez oso handiko datu motekin erabili da, $\varepsilon - \text{mach} \leq 10^{-30}$ dituzten datuekin hain zuzen ere; baina algoritmo hori erabiltzeak zer diferentzia ekartzen duen aztertzea komeni da. Izan ere, algoritmo hau erablitzen ez bada, lortzen diren denboren hobekuntzak 5.5 taulan ikus daitezke —64 bit-eko datu motekin lortzen diren denbora hobekuntzak izanik—.

Denborak %20 inguru hobetzen dira batezbeste, baina Kahanen algoritmorik gabe emaitza ezberdinak lortzen dira. 5.3 irudietan, bi algoritmoetan lortzen diren emaitzak konparatzen dira; bien arteko posizio, abiadura eta energia aldaketak konparatzen direlarik. Hasiera batean, sistema bakoitzeko energiaren errore erlatiboak bistaratzean, desberdintasun oso txikiak lortzen dira, baina ezberdintasun horiek ez dute patroi oso konstanterik

jarraitzen. Denbora-pauso konstanteak simulatzen direnez, Kahanen algoritmorik gabe begizta bakoitzean biribiltze-errore jakin bat sortzen da —biribiltze-errore hori antzekoa izanik iterazio guztietan—; beraz, energiaren errore erlatiboa gorakorra izatea eta linealki haztea espero zen.

Baina, noski, Kahanen algoritmoa ez da energiaren kalkuluan erabiltzen, posizio eta abiadura bektoreetan baizik. Horregatik, *output* iterazio bakoitzeko posizioa eta abiadura bektoreen erroreak ordezkatzen dituzten x_r eta x_v aldagaiak erabili dira —5.4 ekuazioan definitzen direnak—, sortzen diren erroreen batezbestekoak aztertzeko.

Orain bai, x_r eta x_v aldagaien hazkuntza lineala dela ikus daiteke, grafiko guztietan Y ardatza eskala logaritmikoan dagoela kontuan izanik. 5.3 taulan, aztertutako errore erlatiboak kontuan hartzekoak direla erabaki da, Kahanen algoritmoak diferentzia nabarmena ekartzen du eta.

Sistema ezberdinen arteko ezberdintasunak antzekoak dira, baina denbora-pausoak txikitzen diren heinean, desberdintasun hauek handitzen doaz. Azkeneko hau iterazio kopurua handitzen delako gertatzen da, honek egin behar diren kalkulua handitzen du eta, biribiltze-errore kopurua handiagotuz.

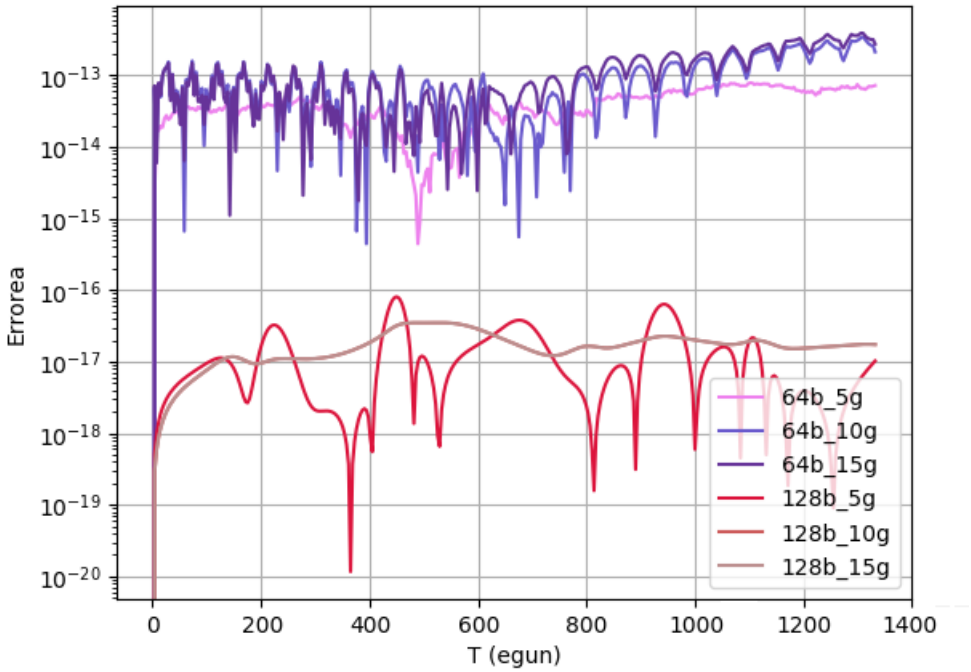
Azkenik, energiaren errore erlatiboak lau datu mota ezberdinentzako kalkulatu badira ere, 64 bit-ekoak bakarrik azaldu dira, bai emaitza eta baita ondorioetan ere. Izan ere, kalkulatu diren emaitzen kasuan, energiaren errore erlatibo berdinak lortzen dira 64 bit edo zehatzagoak diren datu motak erabiliz. Hau aukeratutako denbora-pausoengatik gertatzen da.

5.2 irudiei erreparatuz, 0.1 eguneko denbora pausoak erabiltzean errore erlatiboak *double* aritmetikako ε-mach baliora nahiko ez direla hurbiltzen ikus daiteke. Denbora-pauso txikiagoak erabiliz, $dt = 0.001$egun ≈ 86seg adibidez, ezberdintasunak hobetu antzeman daitezke.

6.1 irudian, denbora-pauso oso txiki bat erabiliz lortu diren energiaren errore erlatiboak aurkitzen dira. Sistemen gorputz kopurua handitzean errore erlatiboa zerbait handitu dela ikus daiteke; eta, batez ere, 128 bit-eko aritmetika erabiltzean lortzen den desberdintasuna antzeman daiteke.

Hala ere, hain denbora-pauso txikiak erabiltzea ez da gomendagarriena, kostu konputazionalarekin alderatuz, emaitzak ez baitira hainbeste hobetzen.

Errore erlatibo txikiagoak lortzeko beste aukera —denbora-pausoa txikitu gabe—, DKD metodoaren orden koadratikoa handitzea da. Hori zuzenean ezin badaiteke egin, zuzen-



6.1 Irudia: 0.001 eguneko denbora-pausoak erabiltzean, energiaren errore erlatiboa 64 bit-eko datuen ε-mach baino txikiagoa izan beharko luke. Bestalde, 128 bit-ekin kalkulu berak eginez lortu beharko litzatekeen errore erlatiboak betetzen dira. Datu mota bakoitza 5, 10 eta 15 gorputzeko sistemetan probatu da, *output-a* astero jasoz.

tzaile sinplektikoak [Wisdom, 2006] erabili daitezke DKD metodoarekin lortutako emaitzak hobetzeko.

6.3 Azkeneko konklusioak

Sortu den *KeplerFlow* paketeak bi algoritmo nagusi ditu. Horietatik, oinarrizkoena eta paketearen euskarria Keplerren fluxua da. Algoritmo hau optimizatzeari garrantzi handia eman zaio, eta, ondorioz, bere barruan erabiltzen den Newton-Raphsonen metodoarentzat hainbat gelditze-irizpide probatu dira exekuzio-denborak hobetzeko. Gainera, Gauss-en f eta g koefizienteak berridatzi eta Kahan-en batuketa konpentsatuaren algoritmoa kodetu da biribiltze-errorea txikitu ahal izateko.

Aldaketa hauek algoritmoan nola eragiten duten ikusteko hainbat proba burutu dira, eta lortu diren emaitzak espero bezain onak dira. Hala ere, orbita parabolikoak -edo radialak, momentu angeluarrik gabeko orbita, $e = 1$ eszentrikotasuna duena- ebazteko Barkerren metodoa inplementatu liteke emaitza zehatza zuzenean lortu ahal izateko, baina ataza hau

ez da sartu proiektuan eraiki den Keplerren fluxua ez baitu aplikatzen.

Julia eta C lengoaiak konparatuz, eraginkortasunean C nagusitzen da. Hala ere, Julia flexibleagoa da aritmetika ezberdineko sarrerako datuak eta kalkuluak egin ditzake eta. Gainera, Julia-n kodetzea azkarragoa da, azken finean maila altuagoko lengoaia bat da eta. Baliabide gutxiko gailu batean exekutatu edo exekuzio-denbora azkarrak behar badira, C lengoaia erabiltzea gomendatzen da, baina bestela Julia erabiltzeak lana errazten du eta lortzen diren emaitzak ez dira batere txarrak.

Bestalde, N-gorputzen fluxuan hainbat hobekuntza egin daitezke. Algoritmo hau [Rein and Tamayo, 2015] artikuluan aipatzen den metodoaren inplementazioa da, Julia lengoaian. Inplementazio honek ez du satelite-planeta-izar erlazioa kontuan hartzen, eta horrek Eguzki-sistemako kasuan arazoak sor ditzake Ilargia bezalako satelite bat N-gorputzeko sisteman sartu nahi bada. Algoritmoaren bertsio berri hori 4. kapituluan azaldu da, eta kostu-konputazionala handitu badezake ere, sistema konplexuago hauen ebazpena zuzenagoa izango litzateke.

Hori gutxi ez balitz, zuzentzaile sinplektikoak —edo *symplectic correctors*— Wisdom-Holfman-en mapan erabili daitezke, integrazio hauen erroreak txikitu ahal izateko —metodo honen ordena handiagotuz—. Hala ere, zuzentzaile hauek ez dira aplikatu, Wisdom-Holfman-en mapa ondo inplementatzea eta aztertzeari lehentasuna eman zaio eta. Dena den, zuzentzaile sinplektikorik gabe, algoritmoak lortu diren energiaren errore erlatiboetan ondo funtzionatzen duela ikus daiteke.

Egin diren kalkulu guztien ondoren, simulazio interaktibo bat garatu da lortu diren emaitzak bistaratzeko. Gainera, simulazio honetan edozein algoritmorekin kalkulatutako datuak bistaratu ditzake, sarrerako datuen formatua errespetatzen den bitartean. Indar konputazional handiko ordenagailuentzat, egun-arteko posizioak zehazteko interpolazio lineala erabili beharrean beste interpolazio zehatzago bat inplementatzea hobeto izango litzateke; baina interpolazio linealarekin nahikoa da astroen orbitak aztertu ahal izateko.

Oro har, hasiera batean ezarritako helburu guztiak bete dira, eta osatutako paketea edonork erabili dezake bere erabilera pertsonalerako. *KeplerFlow* paketea *GitHub*-en aurkitu daiteke, ondorengo estekan: https://github.com/salanueva/KeplerFlow.jl

Eranskinak

A. ERANSKINA

Oinarrizko kontzeptuak

Eranskin honetan, proiektuan zehar landuko diren oinarrizko kontzeptuak jorratuko dira, zeruko mekanikaren funtsezko kontzeptu eta elementuak alegia.

A.1 Keplerren legeak

- *Lehen legea*: Planeta guztiak bere izarraren inguruan mugitzen dira, orbita eliptikoak eginez. Izar hori elipsearen bi fokuetako batean dago, foku nagusia deritzona.
- *Bigarren legea*: Planetatik izarrera doan irudizko lerroak azalera bera estaltzen du denbora-tarte berdinean.
- *Hirugarren legea*: Planetak orbita osatzeko behar duen T periodoaren karratua, izarraren r batezbesteko distantziaren kuboarekiko proportzionala da.

A.2 Newtonen grabitazio-legea

Grabitazio unibertsalaren legea masa duten gorputz ezberdinen arteko elkarreragin grabitatorioak deskribatzen duen lege fisiko klasiko bat da, Isaac Newton-ek definitua bere *Philosophiae Naturalis Principia Mathematica* liburuan.

Legea honakoa da: elkarrengandik r distantziara dauden m_1 eta m_2 masa duten bi gorputzen artean eragiten den F indarra euren masen produktuarekiko proportzionala eta bana-

tzen dituen distantziaren karratuarekiko alderantzizko proportzionala da -A.1 ekuazioa—, non G grabitazio unibertsalaren konstantea den.

$$F = G\frac{m_1 m_2}{r^2} \tag{A.1}$$

Lege hau planeten arteko elkarreragin grabitatorioak definitzen dituenez, N-gorputzen probleman etengabe azaltzen da, gorputzen azelerazioekin erlazioa du eta.

A.3 Elementu orbitalak

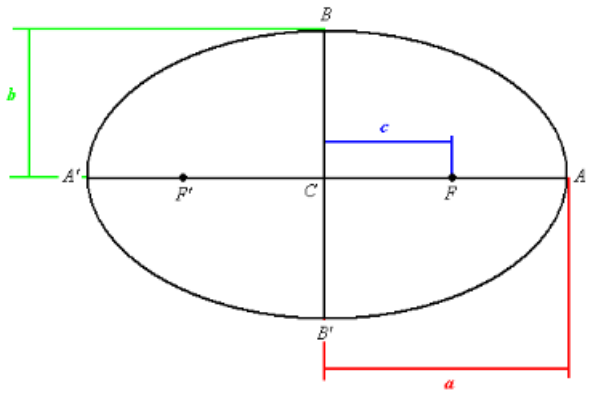
Keplerren legeek definitzen dituzten orbita eliptikoei orbita kepleriarrak deritze. Orbita hauek 6 elementuen bitartez definitu daitezke, elementu orbitalak deitzen direnak. Lehenengo biek —ardatz-erdi nagusiak eta eszentrikotasunak—, orbitaren tamaina eta forma definitzen dute. Ondorengo hirurek —inklinazioak, nodo gorakorraren longitudeak eta periastroaren argumentuak—, orbitaren orientazioa definitzen dute. Azkenik, seigarrenak une horretan astroa orbitaren zein puntutan kokatua dagoen definitzen du. Hala ere, hainbat elementu ezberdin erabili ahal dira gorputzaren posizioa definitzerako orduan [Madariaga, 2006].

1. *Ardatz-erdi nagusia (a)*: Elipsearen ardatz luzeenaren erdia da. Orbitaren tamaina definitzen duen elementua da. Ardatz nagusi honek elipsearen bi fokuak zeharkatzen ditu —A.1 irudia—.

2. *Eszentrikotasuna (e)*: Orbitaren forma definitzen duen elementua. Orbitaren itxura zirkunferentzia batetik zenbat desbideratzen den definitzen du, $[0,\infty)$ tartean egon daitekeelarik. Balioaren arabera, hainbat forma lortu ditzake orbitak. Kontuan izan behar da, orbita itxia izateko $e < 1$ bete behar dela.

   - $e = 0$: zirkunferentzia
   - $0 < e < 1$: elipsea
   - $e = 1$: parabola
   - $e > 1$: hiperbola

   Eszentrikotasuna ardatz-erdi nagusi eta txikiarekin kalkula daiteke, A.2 ekuazioan ikus daitekeen bezala.



A.1 Irudia: Elipsea definitzen duten hainbat elementu agertzen dira hemen. F eta F' elipsearen bi fokuak dira, elipsearen C zentroarekiko distantzia berera daudenak eta $\overline{AA'}$ ardatz nagusian aurkitzen direnak. Beraz, $\overline{AC} = \overline{CA'} = \mathbf{a}$ lerroa ardatz-erdi nagusia da, eta $\overline{BC} = \overline{CB'} = \mathbf{b}$, berriz, ardatz-erdi txikia. Azkenik, $\mathbf{c} = \overline{CF'} = \overline{FC}$ foku eta zentroaren arteko distantzia definitzen du —aldagai hauek A.2 ekuazioan erabiltzen dira—.

$$e = \frac{c}{a} = \frac{\sqrt{a^2 - b^2}}{a} \tag{A.2}$$

3. *Inklinazioa (i)*: Kanpoko erreferentzia-plano batekiko orbitaren orientazioa adierazten du. Eguzki Sistemaren kasuan, plano hori Lurraren orbitaren planoa da.

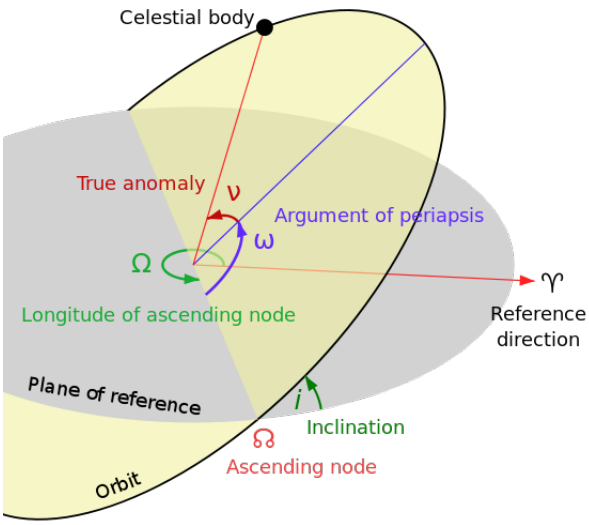
4. *Nodo gorakorraren longitudea (Ω)*: Orbitaren Aries puntutik nodo gorakorrera lortzen den angelua da, inklinazioarekin batera orbitaren planoa definitzen du. Hobeto ulertzeko, ikusi A.2 irudia.

    - Aries puntua (♈): Lurreko zeruan Eguzkia hego hemisferiotik iparraldekora pasatzen den puntua. Beste izarren sistemetan, erreferentzi-planoan definitutako edozein bektore erabili daiteke puntu hau ezartzeko.
    - Nodo gorakorra: Erreferentzi planoa zeharkatzen duen puntua, erreferentzia plano azpitik gora mugitzen ari delarik.

5. *Periastroaren argumentua (ω)*: nodo gorakorraren puntuak eta periastroak [1] sortzen duten angelua.

6. *Seigarren elementua*: batezbesteko anomalia, anomalia eszentrikoa, benetako anomalia eta perihelio-denborak orbitan dagoen gorputzaren posizioa definitzen dute.

[1] *Periastroa*: fokuan dagoen gorputzetik orbitaren puntu hurbilena da.



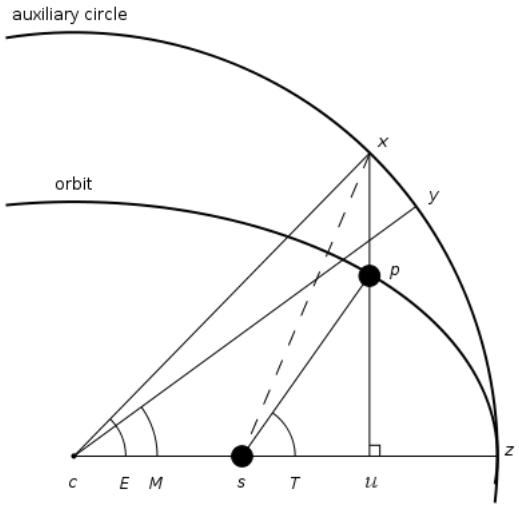
A.2 Irudia: Elementu orbitalak irudikatzen dira: orbitaren i inklinazioa erreferentzia planoarekiko, Ω nodo gorakorraren longitudea —♈ Aries/erreferentzia puntutik nodo gorakorrera—, ω periastroaren argumentua eta ν benetako anomalia. Irudi honetan, gorputza erlojuaren aurkako noranzkoan dabil higitzen.

Lau hauek, era ezberdin batean, gorputza orbitaren zein puntutan dagoen definitzen dute; hortaz, lauak beraien artean erlazionatu daitezke, A.3 irudian ageri den bezala.

- *Batezbesteko anomalia (M)*: gorputza zirkulu nagusian[2] zehar higituko balitz, ardatz nagusiak —periastrotik hasita— eta gorputz horrek sortzen duten angelua izango litzateke —angelua elipsearen zentroan neurtzen delarik—.
- *Anomalia eszentrikoa (E)*: zirkulu nagusian gorputzaren uneko posizioa ardatz nagusiarekiko perpendikularki proiektatuko balitz, zirkulu nagusiko puntu horrek eta ardatz nagusiak zentroan eratzen duten angelua da. Kepler-en ekuazioaren bidez —A.5 ekuazioa—, batezbesteko anomalia eta anomalia eszentrikoaren arteko erlazioa definitzen da.
- *Benetako anomalia (T)*: foku nagusian periastroaren noranzkoa eta gorputzaren posizioak sortzen duen angelua. Benetako anomalia, anomalia eszentrikoarekin erlazionatu daiteke A.3 ekuazioan.

$$\tan\frac{T}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} \tag{A.3}$$

[2] *Zirkulu nagusia*: orbitaren zentro bera eta ardatz-erdi nagusiaren luzera berdineko erradioa duen zirkunferentzia da. Orbitak periheliotik hasita abiadura-angeluar konstante bat mantenduko balu, orbita horrek izango lukeen itxura izango litzateke.



A.3 Irudia: M batezbesteko anomalia, E anomalia eszentrikoa eta T benetako anomalia irudikatzen dira diagrama honetan, non c orbitaren zentroa, s foku nagusia eta p higitzen ari den gorputza diren. Periastro-denbora gorputza z periastroan dagoen unea da.

- *Periastro-denbora*: gorputza orbitaren periastrotik pasa den unea definitzen du. A.4 ekuazioak batezbesteko anomalia eta T_p perihelio denboraren arteko erlazioa definitzen du, T_{orb} orbitaren periodoa eta t gorputzaren posizioa kalkulatu nahi den unea izanik.

$$\frac{t - T_p}{T_{orb}} = \frac{M}{2\pi} \tag{A.4}$$

A.4 Keplerren ekuazio tradizionala

Keplerren ekuazio honek batezbesteko anomalia eta anomalia eszentrikoaren arteko erlazio trigonometrikoa ezartzen du eszentrikotasunaren bitartez —A.5 ekuazioa—. Adibidez, orbitaren eszentrikotasuna 0-koa balitz —$e = 0$—, gorputzaren abiadura-angeluarra konstantea izango litzateke, orbitaren forma zirkunferentzia batena izanik —zirkulu nagusian orbitatuko luke gorputzak—. Hortaz, batezbesteko anomalia, anomalia eszentrikoaren berbera izango litzateke: $M = E$.

$$M = E - e \sin E \tag{A.5}$$

Eszentrikotasuna handitzen doan heinean, M eta E-ren arteko diferentzia handitzen joan-

go da, ardatz-erdi txikiaren luzera txikitzen joango baita. Diferentzia hori, matematikoki, $e \sin E$ izango litzateke, non anomalia eszentrikoaren sinua A.3 irudiko $\overline{XP}$ lerroa izango litzatekeen eta P planetaren kokapena A.6 ekuazioak definituko lukeen, kokapen horrek $\overline{XU}$ lerroan egon beharko lukeelarik.

$$e = \frac{||\overline{XP}||}{||\overline{XU}||} \tag{A.6}$$

A.4.1 Ekuaziotik gorputzaren posizioa lortzen

A.5 ekuazioa transzendentala izanik —aljebraikoki ebatzi ezin den ekuazioa—, zenbakizko metodo iteratiboak erabili behar dira hurbilpen zehatz bat lortzeko —Newton-Raphsonen metodoa, adibidez—.

Behin anomalia eszentrikoaren hurbilpen zehatz bat edukita, plano orbitaleko[3] planetaren XY koordenatuak kalkula daitezke —A.7 eta A.8 ekuazioak— [Madariaga, 2006].

$$x = a(\cos E - e) \tag{A.7}$$

$$y = b \sin E \tag{A.8}$$

Behin plano orbitaleko XY koordenatuak izanik, koordenatuak hiru dimentsioko erreferentzi-sistemara eraman ditzakegu, hiru biraketa-matrize aplikatuz —ardatz bakoitzeko biraketa bat—.

$$\begin{pmatrix} X_{3D} \\ Y_{3D} \\ Z_{3D} \end{pmatrix} = M_3\, M_2\, M_1 \begin{pmatrix} X_{orb} \\ Y_{orb} \end{pmatrix} \tag{A.9}$$

Lehen aipatu denez, hiru elementu orbitalek plano orbitalaren orientazioa definitzen dute: i inklinazioak, Ω nodo-gorakorraren longitudea eta ω periastroaren argumentua. Beraz, angelu hauek erabili ditzakegu biraketa-matrizeak definitzeko. Beraz, egin beharreko eragiketa A.9 ekuazioan ikus daiteke, non M_1, M_2 eta M_3 biratze-matrizeak A.10 ekuazioan definitzen diren.

[3] *Plano orbitala*: planeta higitzen den elipsearen planoa. Plano orbitalaren koordenatu-sistema honela definitzen da: jatorria orbitaren zentroa da, X ardatza elipsearen ardatz-nagusian zehar definitzen da, eta Y ardatza, berriz, ardatz txikian zehar.

$$M_1 = \begin{pmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{pmatrix} \quad M_2 = \begin{pmatrix} 1 & 0 \\ 0 & \cos i \\ 0 & \sin i \end{pmatrix} \quad M_3 = \begin{pmatrix} \cos\Omega & -\sin\omega & 0 \\ \sin\Omega & \cos\omega & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{A.10}$$

A.5 Newton-Raphsonen metodoa

Esaldi batean, Newton-Raphsonen metodoa $f(x)$ funtzio erreal deribagarri baten erroen hurbilpenak aurki ditzakeen algoritmo bat da. Metodoaren konbergentzia ezin da guztiz bermatu, erabiltzen den hasierako hurbilpenaren arabera konbergitzen baitu algoritmoak.

Metodo iteratibo honetan, x_0 hasierako hurbilpen bat definitzen da, bilatzen ari den errotik gertu dagoela estimatzen dena. Esan bezala, estimazio hau ez bada errorera asko hurbiltzen, minimo/maximo erlatibo batean trabatu daiteke. Gainera, $f'(x_0)$ 0-ra gerturatzen den kasuan hurbilpena asko urruntzen da hasierako hurbilpenetik, eta $f'(x_0) = 0$ kasuak erroreak ekar ditzake. Behin x_0 definitu delarik, A.11 ekuazioan definitutako $n \in \mathbb{N}$ iterazioak kalkulatuko dira gelditze irizpide batera iritsi arte.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{A.11}$$

Normalean, gelditze-irizpide bat $|x_{n+1} - x_n| < \varepsilon$ erako kondizio bat izaten da, non ε gehienez onartzen den errorea den, hots, tolerantzia. Metodo honek orden koadratiko bat du, A.12 ekuazioan $q = 2$ denean τ balio positibo batera konbergitzen duelako. Hau horrela izanik, metodoak koadratikoki konbergitzen duela esan daiteke.

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|^q} = \tau \tag{A.12}$$

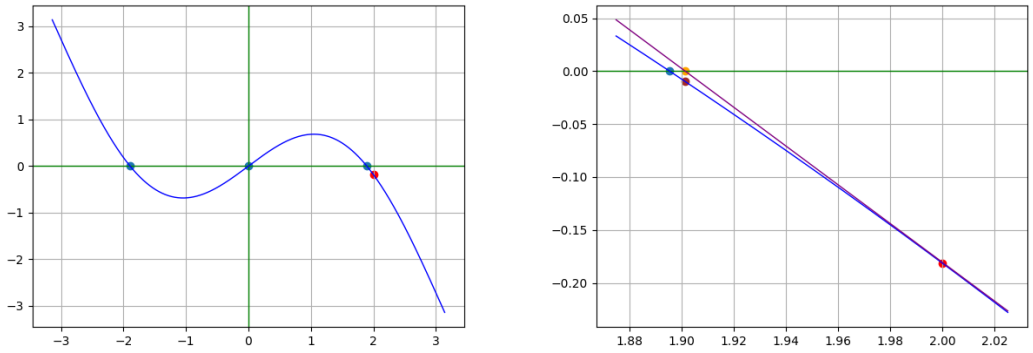
A.5.1 Metodoaren aplikazioa

Metodo honen funtzionamendua eta konbergentzia ordena ulertzeko adibide bat proposatzen da. $f(x) = 2\sin x - x$ funtzioa eta $f'(x) = 2\cos x - 1$ bere deribatua izanik, $x_0 = 2$ hasierako estimazio bezala hartzen bada, honela garatzen da hurbilpena Newton-Raphsonen metodoarekin —A.1 taula—. Gelditze-irizpidearen $t = 10^{-15}$ tolerantzia definitu da.

Iter.	x	$f(x)$	$f'(x)$
0	2.00000000000000000	-0.18140514634863660	-1.83229367309428477
1	1.90099559420390903	-0.00904008714061051	-1.64846307473352480
2	1.89551164537959469	-0.00002846679920560	-1.63807798874337045
3	1.89549426720871319	-2.8621929e-10	-1.63804504861644169
4	1.89549426703398094	1.17023e-17	-1.63804504828523771
5	1.89549426703398094	1.17023e-17	-1.63804504828523771

A.1 Taula: Taula honetan, algoritmoaren garapen azkarra ikus daiteke. Iterazio bakoitzean lortzen den x zehatzagoa da. x zutabean azpimarratuta agertzen diren hamartarrak ez dira aldatzen iterazioak aplikatzen diren heinean.

A.1 taulan ikusten denez, 6. iterazioan gelditze-irizpidea betetzen da, $|x_6 - x_5| = 0 < 10^{-15}$ baita. Gainera, metodoaren orden koadratikoa begi-bistan dago, iterazio bakoitzean lortzen den zehaztasuna bikoizten baita.



(a) Aztertutako $f(x)$ funtzioa **(b)** Metodoaren lehen iterazioa

A.4 Irudia: A.4a irudian Newton-en metodoarekin aztertu den $f(x) = 2\cos(x) - x$ funtzioa irudikatzen da —lerro urdina—, funtzio horrek dituen hiru erroak —puntu urdinak— azaltzen direlarik, hasierako estimazioarekin batera —puntu gorria—. A.4b irudian, berriz, metodoaren lehen iterazioan egindako kalkuluak geometrikoki azaltzen dira, non lerro morea hasierako puntuko lerro ukitzailea, puntu laranja lerro ukitzailea eta X ardatzaren ebakidura-puntua eta puntu morea hurrengo iterazioko estimazioa den

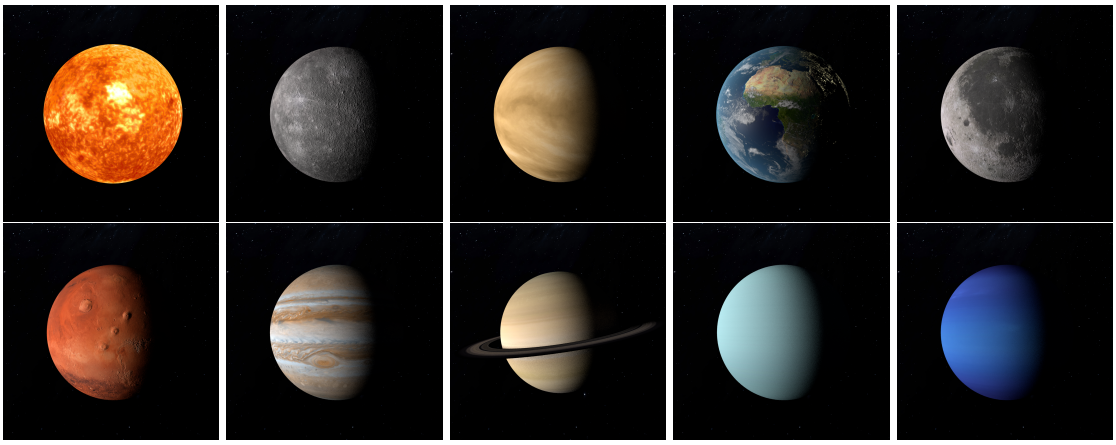
Azkenik, A.4 irudietan $f(x) = 2\sin(x) - x$ funtzioak hiru erro dituela ikus daiteke, baina amaierako estimazioa hasierako estimaziotik gertuen dagoen erroa dela antzeman daiteke. Hemen ikusten da nola hasierako estimazioak lortzen den erroa baldintzatzen duela.

B. ERANSKINA

Blender

Hasiera batean, Eguzki-sistemako simulazioan *Blender*-en definitutako *shader*-ak erabili nahi ziren, 3D modelatze irakasgaian landutakoa praktikan sartzeko. *Blender* aplikazioan simulazio interaktibo bat garatzea ahal bada ere, aplikazio honek eskaintzen duen ingurunea ez da hoberena Eguzki-sistemaren simulazioa egiteko. Horregatik, *Three.js* liburutegia erabiliz, web nabigatzaile batean exekutatzen den simulazioa egin da.

Ideia *Blender*-en definitutako *shader*-ak erabiltzea bazen ere, *shader* pertsonalizatu hauek ezin dira *Three.js*-ra zuzenean exportatu. Hau jakiteko ordurako *shader*-ak eginak zeudenez, simulazioan erabili diren ikono batzuk egiteko erabili dira, baita sarrerako datuak jasotzeko orriaren atzealdea egiteko ere.

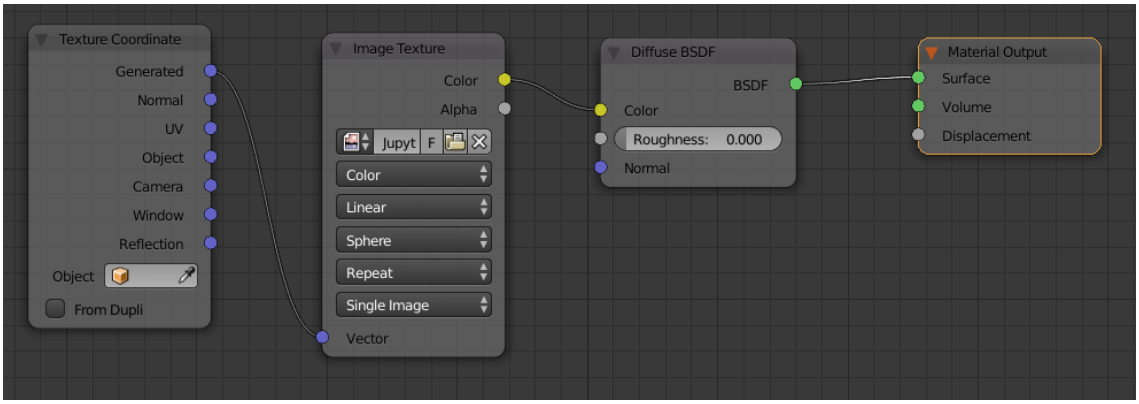


B.1 Irudia: Eguzki-sisteman agertzen diren gorputzen ikonoak. Lehenengo ilaran: Eguzkia, Merkurio, Artizarra, Lurra eta Ilargia. Bigarrenean, berriz: Marte, Jupiter, Saturno, Urano eta Neptuno.

4. kapituluko hainbat irudietan sortutako ikono eta atzealde hauek agertzen dira: 4.13 eta 4.16 irudietan, besteak beste. Ikono hauek B.1 irudietan hobeto ikus daitezke.

Ikono hauek astroen ezaugarrien arabera definitu dira, eta *Cycles* renderizatzailearen bidez lortzen den foto-errealismoa antzeman daiteke. Hamar gorputzen artean, bost gaseosoak —Eguzkia, Jupiter, Saturno, Urano eta Neptuno— dira, eta besteak, berriz, solidoak. *Shader*-ak egitean, astro gaseosoen *shader*-ak dira errazenak egiten, sarrerako testura bat erabiltzen dute eta. B.2 irudian astro gaseosoen *shader*-a nola definitzen den azaltzen da.

Shader hauek zuzenean programatu beharrean, *Blender*-ek nodo editore bat dakar aurredefinitutako nodo batzuen bidez nahi den *shader* edo materiala definitzeko.

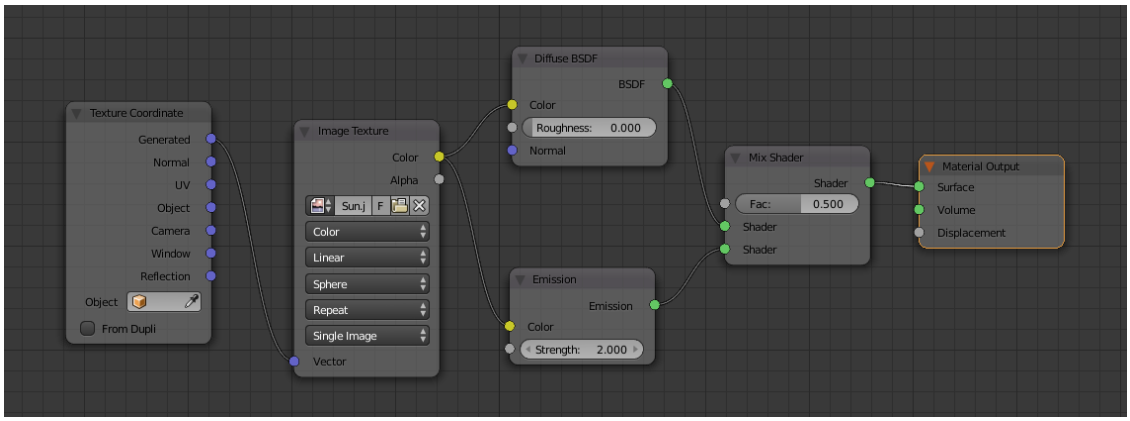


B.2 Irudia: Astro gaseosoen shaderra. Astro hauen gainazala testura bezala sartzen da *Image Texture* nodoaren bidez, eta gorputza errepresentatzen duen esferaren geometria hartzen da testura hori esferan mapeatzeko —*Texture Coordinate* nodoa—. Ondoren, material honen erreflexua definitzen da *Diffuse BSDF* nodoaren bidez, gorputz hori argiztatzen bada gorputzak islatzen duen kolorea definitzen duena. Shader horrek kalkulatzen duen kolorea renderizazioan ikusten den gorputzaren materiala definitzen du.

B.2 irudiko nodo multzoa Artizarra, Jupiter, Saturno, Urano eta Neptuno planetentzat. Artizarraren atmosfera oso lodia denez eta bere gainazala espaziotik ezin denez ikusi, gorputz gaseoso bat bezala tratatu da.

Eguzkiaren kasua aurreko planeten ezberdina da, argi-iturri bat baita. Argia igortzen duenez, *shader* berezi bat definitu da Eguzkiarentzat, B.3 irudian.

Lau gorputzen *shader*-ak definitzea falta da. Aurrerago Lurrari tratamendu berezi bat emango zaionez, gelditzen diren hiru gorputzen materiala definitzea falta da. Merkurio, Marte eta Ilargia astro solidoak direnez, *Bump-mapping* teknika erabiliko da, gorputzen gainazalei zimurtasun efektu bat emateko. Astro hauen *shader*-a B.4 irudian definitzen da.



B.3 Irudia: Gorputz gaseosoen antzeko *shader*-a da, baina oinarrizko bi *shader* nahasten ditu *Mix Shader* nodoa erabiliz. Horietako batek gorputzak islatzen duen kolorea definitzen du *Diffuse BSDF*, eta besteak igortzen duen argitasuna —*Emission* nodoa—. Berez Eguzkiak argia igortzen badu ere —eta ez islatu—, bien nahasketa egiteak nahi den efektua ematen dio Eguzkiari.

B.1 Lurraren materiala

Lurraren kasua berezia da. Izan ere, itsasoak ditu —zona hauetan isla espekularrak sortuz—, atmosfera bat dauka —esferaren ertzetan atmosfera hau ikusgarria izanik—, atmosferan hodeiak daude eta gauez gizakiak argi artifiziala sortzen du.

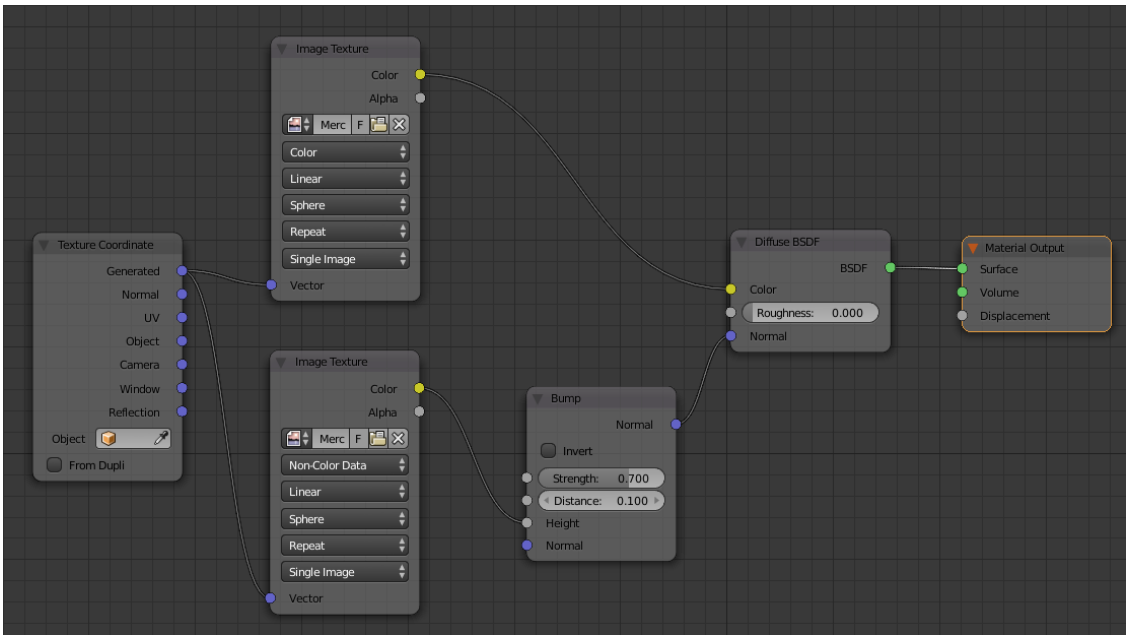
Hau guztia batzeko hiru *shader* ezberdin definitu dira. *Shader* bakoitza hiru esfera ezberdinetan aplikatzen da; kanpo-esfera atmosfera izanik, erdiko esfera hodeiak jartzeko erabiliko delarik eta barne-esfera Lurraren gainazala izanik.

Atmosferarekin hasiz, *shader* hau B.5 irudian definitzen da. Kontuan izanik ia bere osotasunean gardena izan behar dela, hodeiak eta gainazala ikusi nahi badira behintzat.

Erdiko esferarekin jarraituz, atmosfera eta hodeien materialak oso antzekoak dira. Hodeien esfera edo geruzan, hodeirik ez dagoenean gardena izan behar du. Gainera, hodeiei ukitu errealista emateko sakonera emango zaie *bump-mapping* erabiliz. *Shader* honetan ez dira nodo berririk agertzen, beraz B.6 irudiko nodo multzoa interpretatzea erraza bihurtzen da.

Azkenik, Lurraren gainazala definitu behar da. Gainazalaren materiala definitzen duen nodo multzoa da konplexuena. Horregatik, nodoz nodo azaldu beharrean, B.7 irudian nodo multzoak egiten duena esplikatuko da.

B.7 irudiko nodo multzoa bi zatitan banatu daiteke, goikoa eta behekoa deituko direnak testua sinplifikatzeko.



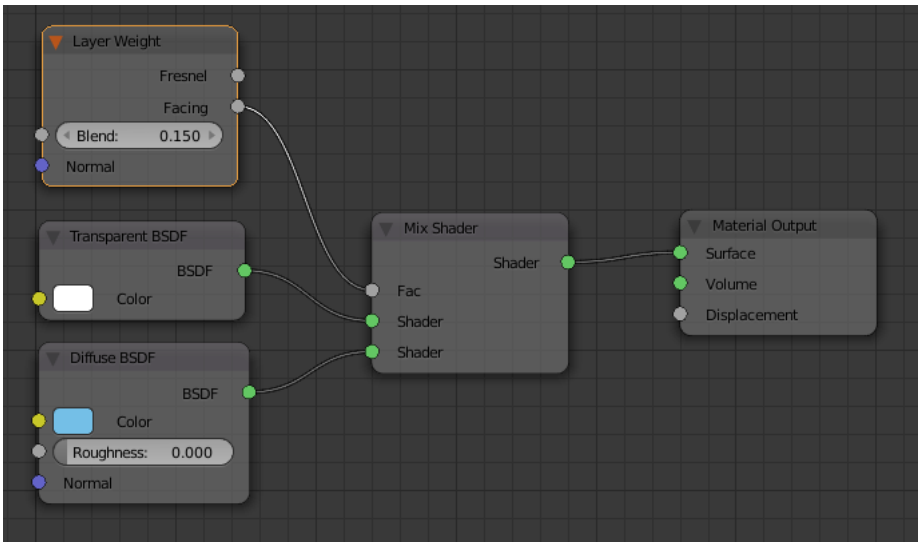
B.4 Irudia: B.2 irudian definitutako nodo multzoari bi gehikuntza egin zaie. Alde batetik, *bump-mapping*-a egiteko beharrezkoa den testura eman behar zaio —azpiko *Image Texture* nodoa erabiliz—, kolorea ezartzen ez duen testura bat dela zehaztu behar delarik. Beste aldetik, *Bump* nodoa gehitu behar da. Nodo honek *bump* testura hartu eta gorputzaren normalak aldatzen dizkio, gorputz hori argiztatzean argiaren isla aldatzeko eta materialari sakonera efektua emateko, honen geometria aldatu gabe.

Lurrari kolorea ematen dioten bi testura erabili dira: Lurraren gainazalaren testura eta gizakiak sortutako argi-artifizialena. Biak Lurraren eguneko eta gaueko testura osatzen dute.

Beheko nodo multzoan, berriz, guztira lau testura erabiltzen dira: Lurraren gainazala, argi-artifiziala, gainazalaren mapa-espekularra eta lurrazalaren *bump*-mapa. Alde batetik, *Bump*-mapa planeta solidoen kasuan bezala aplikatu da —*Bump* nodo bat erabiliz—. Sakonera hori Lurraren gainazalari ezartzen zaio —ez argi-artifizialari, ez du behar eta—.

Ondoren mapa-espekularrak isla-espekularrak Lurraren gainazalaren zein zonaldetan sortzen diren zehazten du —itsasoetan, alegia—. Isla-espekularra *Glossy BSDF* nodoaren bitartez simulatzen da; eta hori Lurraren gainazalaren kolorea definitzen duen *Diffuse BSDF* nodoarekin nahasten da *Mix Shader* bat erabiliz. Beste aldetik, argi-artifizialaren testurak Lurraren barne-esferan argia zein puntuetan igortzen den definitzen du, argia *Emission* nodoa erabiliz definitzen delarik. B.8 irudian beheko nodo multzoaren eskema azaltzen da.

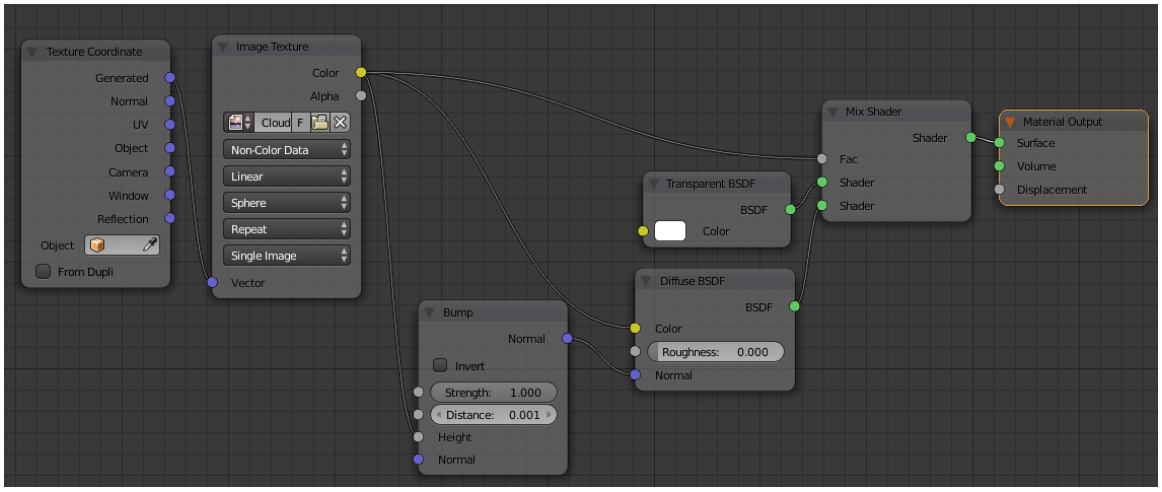
Goiko nodo multzoak testura bakoitzari dagokion esferaren erdia definitzen du, argi-



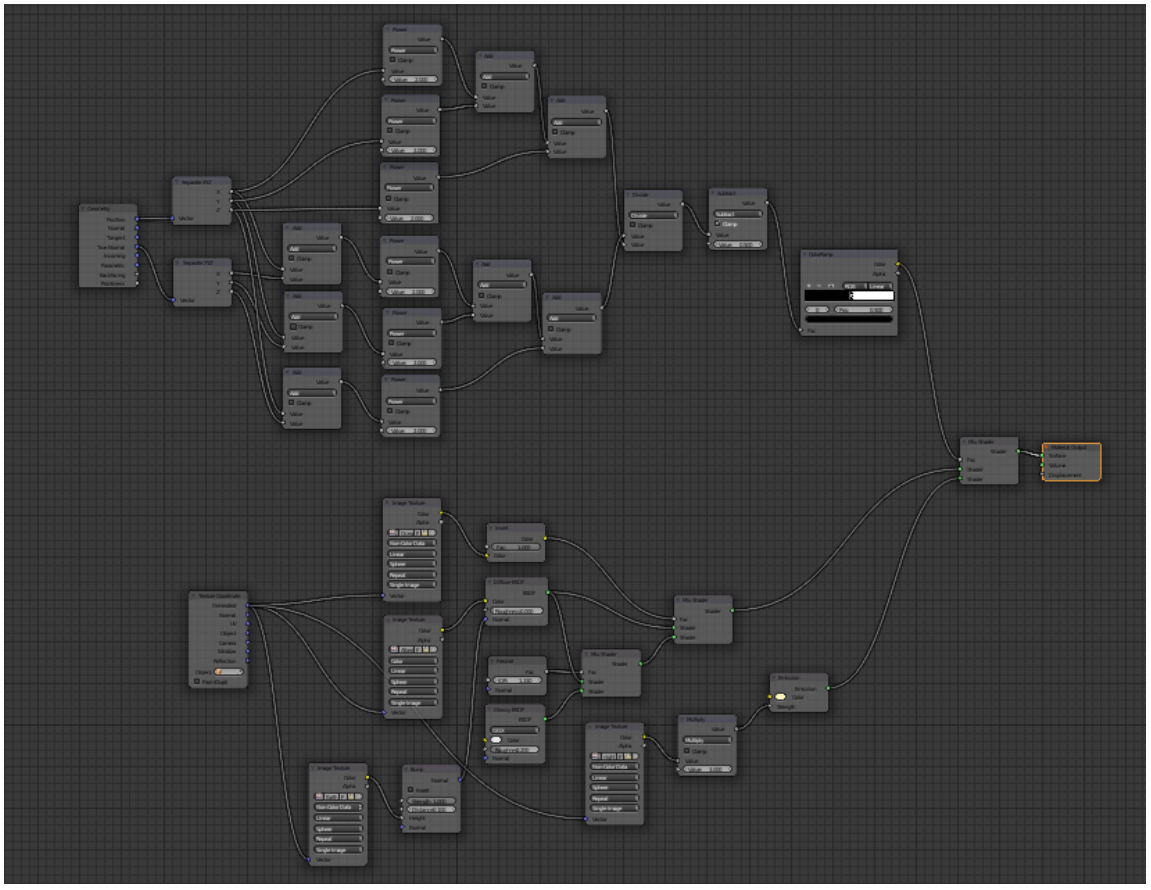
B.5 Irudia: *Mix Shader* baten bidez, atmosferak islatzen duen kolore urdin argia —Diffuse BSDF nodo bidez definitua— eta kanpo-esferaren gardentasuna —Transparent BSDF nodoa— nahsten dira. Nahasketa hau kameraren ikuspuntuaren araberakoa da; hots, esferaren aurpegi baten normala kameraren norabidearekiko paraleloa bada, atmosfera ez da ikusiko; normal horrek angelu ia perpendikularra osatezen den kasuan, berriz, atmosferak urdin-argi kolorea izango du —Layer Weight nodo bidez definitua—.

iturriaren posizioa eta esferaren geometriaren posizio eta normalekin baliatuz esfera-erdi horiek definitzen dituena. Bi nodo multzoekin Lurraren gainzalaren koloreak definitzen dituen *shader* erraldoia lortzen da.

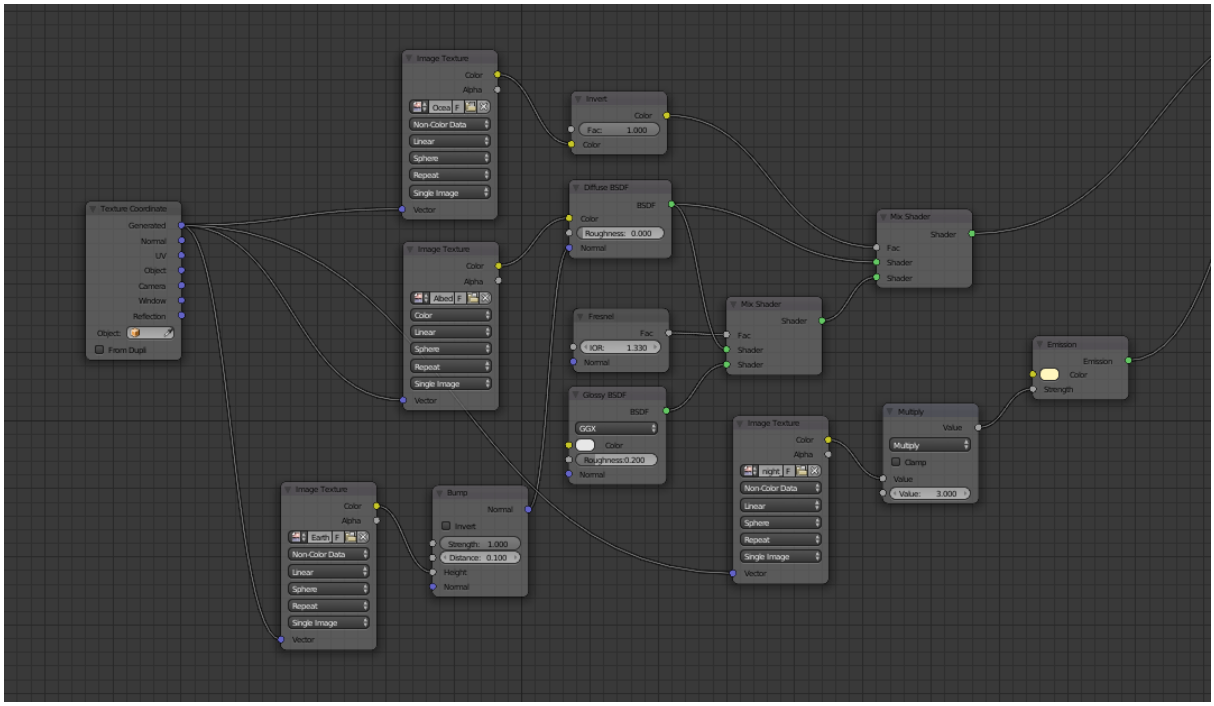
Aipatutako hiru esferek Lurra osatzen dute, eta definitutako *shader* hauek batera ipiniz B.9 irudia bezalako renderizazioak lor daitezke.



B.6 Irudia: Lurraren hodeien shader-a.



B.7 Irudia: Lurraren gainazalaren shader-a.



B.8 Irudia: Lurraren gainazalaren shader-aren beheko zatia.

B.9 Irudia: Lurra eta Ilargiaren renderizazioa. Bi gorputz hauek ez daude eskalan.

Bibliografia

[Curtis, 2010] Curtis, H. D. (2010). *Orbital mechanics for engineering students*.

[Farrés et al., 2012] Farrés, A., Laskar, J., Blanes, S., Casas, F., Makazaga, J., and Murua, A. (2012). High precision Symplectic Integrators for the Solar System.

[Fernandes, 2003] Fernandes, S. (2003). Universal Closed-Form of Lagrangian Multipliers for Coast-Arcs of Optimum Space Trajectories. XXV(4):336–340.

[Fukushima, 1999] Fukushima, T. (1999). Fast Procedure Solving Universal Kepler's Equation. *Celestial Mechanics and Dynamical Astronomy*, 75(3):201–226.

[Madariaga, 2006] Madariaga, D. (2006). *Orienta ezazu zeure burua zeruko esferan*. Udako Euskal Unibertsitatea.

[Mikkola and Innanen, 1999] Mikkola, S. and Innanen, K. (1999). Symplectic Tangent map for Planetary Motions. *Celestial Mechanics and Dynamical Astronomy*, 74(1):59–67.

[Pimienta-Peñalver and Crassidis, 2013] Pimienta-Peñalver, A. and Crassidis, J. (2013). Accurate Kepler equation solver without transcendental function evaluations. *Advances in the Astronautical Sciences*, 147:233–247.

[Regan, 2015] Regan, R. (2015). The Spacing of Binary Floating-Point Numbers.

[Rein and Tamayo, 2015] Rein, H. and Tamayo, D. (2015). WHFast: A fast and unbiased implementation of a symplectic Wisdom-Holman integrator for long term gravitational simulations.

[Tokis, 2014] Tokis, J.Ñ. (2014). A Solution of Kepler ' s Equation. (December):683–698.

[Wisdom, 2006] Wisdom, J. (2006). Symplectic Correctors for Canonical Heliocentric n-Body Maps. *The Astronomical Journal*, 131(4):2294–2298.