N-GORPUTZEKO PROBLEMA GRABITAZIONALAREN EBAZPENERAKO ZENBAKIZKO METODOEN AZTERKETA.

by

Mikel Antonana Otano

Informatika Fakultatea
Euskal-Herriko Unibertsitatea
Donostia

Gaien Aurkibidea

Gaien Aurkibidea (detailed)

1. Kapitulua

Sarrera.

1.1. Ikerketaren testuingurua.

Urte luzez, zientziaren arlo ezberdinek N-gorputzeko problema ikertu dute. Astronomoek eguzki-sistemaren planeten mugimendua ulertu nahian egindako lanak edo kimikariek erreakzio kimikoekin esperimentatzeko molekulen dinamikaren azterketak aipatu daitezke. Arlo bakoitzak bere lege fisikoak baditu ere, oinarrian problema berdina lantzen dute eta beraien arteko antzekotasunak handiak dira. Azpimarratu ere, N-gorputzen problemaren azterketak garrantzi berezia izan duela matematikako eremu ezberdinen garapenean, esaterako dinamika ez-lineal eta kaos teorian.

Garai batean, N-gorputzen problemak teori analitikoen bidez aztertzen ziren baina konputagailuen sorrerarekin, zenbakizko integrazioak tresna nagusia bilakatu ziren. Azken hamarkadetan, bai konputazio teknologien aurrerapenari esker bai algoritmo berrien sorrerari esker, zenbakizko azterketek garapen handia izan dute. Zenbakizko simulazioen laguntzaz, eguzki-sistemaren dinamikaren funtsezko galdera batzuk ezagutu ditugu eta berriki, Karplusen taldeak 2013. kimikako Nobel saria [27] jaso du kimika konputazionalean egindako lanarengatik.

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

1. Epe motzeko eta doitasun handiko integrazioak. Eguzki-sistemaren efemeride zehatzak edo espazioko satelite artifizialen kokapenen kalkuluetarako erabili ohi dira.

2. Epe luzeko baina doitasun txikiko integrazioak. Denbora epe luzean planeta-sistemen mugimendu ezagutzeko egindako ikerketak ditugu. Azterketa hauetan, garrantzitsua gorputzen mugimenduaren argazkia orokorra (zehaztasun

handirik gabe) ezagutzea da. Normalean, problema mota hauetan gorputzen arteko kolisio gertuko egoerak ez dira izaten.

3. N-gorputz kopurua edozein izanik, hauen arteko kolisioak gerta daitezkeen problemak. Integrazio hauetan, konplexutasun handiari aurre egin behar zaio. N-gorputz kopurua miliotakoa izan dateke eta kolisio gertuko egoeren ondorioz, kalkulutan egindako zenbakizko errore txikiek soluzioan eragin handia izan ditzakete.

Gure helburua eguzki-sistemaren epe luzeko eta doitasun handikoa algoritmoak garatzea da. Aurreko hamarkadetan, eguzki-sistemaren planeten epe luzeko zenbakizko integrazioa erronka garrantzitsua izan da. Adibidez, Sussmanek eta Wisdomek (1993) [32] eguzki sistemaren 100 milioiko integrazioarekin, planeten mugimendua kaotikoa zela baieztatu zuten. Aldi berean, paleoklimatologizientziak orain milioika urte gertatutako klima zikloak (epel-aroa, hotz-aroa eta glaziazio-aroa) azaltzeko, lurraren orbitan izandako aldaketaren eraginez gertatu zela azaltzen duen teoria (Milankovitch 1941) [2] baieztatzeko planeten orbiten efemeride zehatzak beharrezkoak dira.

Konputazio-teknologi aurrerapenak handiak izan arren, eguzki-sistemaren simulazio hauek konputazionalki oso garestiak dira eta exekuzio denbora luzeak behar dituzte; adibidez, Laskarrek [20, 2010] bere azken integrazioa burutzeko 18 hilabete behar izan zituen. Azken urteotako konputagailu berrien arkitekturaren bilakaerak, algoritmo azkarren diseinua aldatu du: simulazioak azkartzeko algoritmoak paralelizazioan oinarritu behar dira eta eragiketa aritmetikoek baino kostu handiagoa du memorien arteko datu komunikazioak. Beraz, oraindik ere algoritmo eraginkorragoak beharrezkoak dira eta hauek garatzeko bide berriak ikertu behar dira.

1.2. Motibazioa.

Epe luzeko integrazioetarako zenbakizko hainbat metodo erabiltzen dira, bereziki beren izaera Hamiltondarra mantentzen duten metodoak (metodo sinpletikoak). Metodo horien artean, gehien erabiltzen direnak izaera esplizituko algoritmoak dira.

Problema ez-stiffa denean, metodo esplizituak metodo inplizituak baino eraginkorragoak kontsideratzen dira. Metodo inplizituetan ekuazio sistema ez-lineala askatu behar da, eta honek metodo esplizituekiko CPU denbora gainkarga suposatzen du. Horregatik problema ez-stiffa bada, metodo esplizituak erabili ohi dira eta problema stiff-a denean bakarrik jotzen dugu metodo inplizituengana. Baieztapen hau eztabaidagarria da, eta praktikan metodo inplizituetan gehiago sakondu behar dela iruditzen zaigu.

Jarraian, metodo inplizituen ezaugarri interesgarri batzuk nabarmenduko ditugu. Abantaila nagusienetakoa malgutasuna da. Metodo inplizituek inplementazio malgua onartzen dute eta ondorioz, integratu nahi dugun problemari egokitzeko aukera gehiago eskaintzen dizkigu. Aipatzekoa da ere, metodo esplizituak sistema Hamiltondar banagarrietan bakarrik aplika daitezke eta Hamiltondarraren egitura hau aprobetxatuz oso eraginkorrak direla. Metodo inplizituak aldiz, Hamiltondar orokorrei aplika daitezke eta gainera, lehen ordenako ekuazio diferentzialetarako metodo sinpletikoak inplizituak izan behar dira. Azkenik ez dugu ahaztu behar, metodo inplizituen artean orden altuko metodoak existitzen direla eta hauek nahitaezkoak dira doitasun handiko integrazioak behar ditugunean.

Lan honetan, metodo inplizituen artean Gauss zenbakizko integrazio metodoa aukeratu dugu. Hainbat autorek (Hairer [14][15] eta Sanz Serna[19]) metodo honen potentziala nabarmendu dute eta guk ere, iritzi berekoak gara. Laburki aipatuz, s ataletako metodo hau $2s$ ordenekoa da, sinpletikoa da, egonkortasun ezaugarri onak ditu eta paralizatzeko gaitasuna ahaztu gabe.

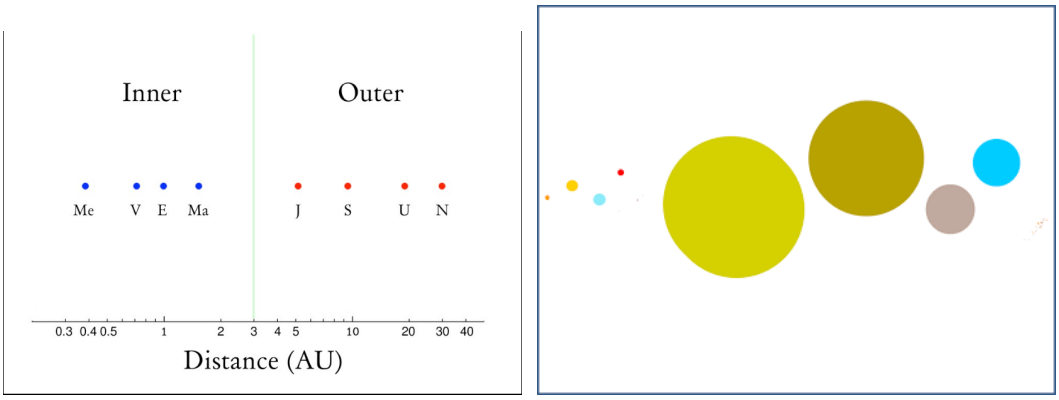
1.3. Helburua eta esparrua.

Gure helburua, eguzki-sistemaren ebazpenerako Gauss inplizituaren inplementazio eraginkorra proposatzea da. Hau lortzeko bereziki honako aspektu hauek kontutan izango ditugu: eguzki-sistemaren problemaren ezaugarriak, konputagailuen koma-higikorreko aritmetika eta algoritmo paraleloen abantailak.

N-gorputzeko problema grabitazionalari dagokionez, eguzki-sistemaren eredu sinplea integratuko dugu. Eguzki-sistemaren gorputzak masa puntualak kontsideratuko ditugu eta gure ekuazio diferentzialek, gorputz hauen arteko erakarpen grabitazionalak bakarrik kontutan hartzen dituzte. Beraz, eguzki-sistemaren eredu konplexuagoetako erlatibitate efektua, gorputzen formaren eragina, eta beste zenbait indar ez-grabitazionalak ez dira kontutan hartu. Bestalde era honetako integrazioetan, gorputzen hasierako balio eta parametro zehatzak sateliteen bidez jasotako datu errealekin bat datozela egiaztatze prozesua ez dugu landu.

Zeintzuk dira eguzki-sistemaren problemaren ezaugarri bereziak? Batetik bi gorputzen problemaren (Keplerren problema) soluzioa zehatza ezaguna da eta eguzki-sistemaren gorputzen mugimenduaren konputazioaren oinarria. Bestetik, badugu gorputz nagusi bat (eguzkia) eta honen inguruan mugimenduan dauden gorputzak: barne planetak, masa txikikoak eta eguzkitik gertu daudenak ; kanpo planetak, masa handikoak eta eguzkitik urrun daudenak (ikus irudia Fig.1.1.). Eguzki-sistemaren egitura honi abantaila handien lortzen duen planteamendua bilatuko dugu.

Konputagailuen koma-higikorreko aritmetika ondo ulertzea garrantzitsua da. Zenbaki errealen adierazpen finitua erabiltzen denez bai zenbakiak memorian gor-



(a) Planeten arteko distantziak. (b) Planeten arteko masa konparaketa.

1.1. Irudia: Eguzki-sistema.

detzerakoan, bai hauen arteko kalkulu aritmetikoak egiterakoan, errore bat egiten dugu. Integrazio luzeetan errore hau propagatzen da eta une batetik aurrera, soluzioen zuzentasuna ezereztatzen da. Ondorioz, integrazioan zehar errore honen monitorizazioa ezagutzea interesgarria da eta integrazio luzeen kasuan, doitasun handian lan egiteko beharra azaltzen zaigu. Doitasun altuko aritmetiken erabilera oso garestia da, inplementazioa software bidezkoa delako. Exekuzio denborak onargarriak lortzeko tarteko irtenbidea, inplementazioan doitasun ezberdinak nahastea izango litzateke.

Sarrera honetan paralelizazioari buruzko ohar bat ematea komeni da. Algoritmo baten kode unitateak paraleloan exekutatzeak badu gainkarga bat eta beraz, algoritmoaren exekuzioa paralelizazioaz azkartzea lortzeko, unitate bakoitzaren tamainak esanguratsua izan behar du. Gure eguzki-sistemaren eredua sinplea da eta logikoa da pentsatzea eredu konplexuagoetan, paralelizazioak abantaila handiagoa erakutsiko duela. Bestalde, N-gorputzen kopurua handia den problemetan, hauen arteko interakzio kopuru $O(N^2)$ handia kalkulatu behar da eta indar hauen hurbilpena modu eraginkorrean kalkulatzeko metodo ezagunak daude: *tree code*[1] eta *fast multipole method*[6]. Baina gure probleman gorputz kopurua txikia denez, teknika hauek gure eremutik kanpo utzi ditugu.

1.4. Ekarpenak.

1.5. Tesiaren egitura.

Gure lanaren abiapuntua Hairer-en IRK metodoaren inplementazioa da [15]. Autoreak IRK puntu-finkoaren inplementazio estandarrean biribiltze errorearen okerreko portaeraz jabetu zen eta arazo hau konpontzeko soluzioa proposatu zuen. Lehen urratsa honetan, biribiltze errorearen arazoari soluzio berri bat eman diogu eta gure IRK inplementazioaren oinarriak finkatu: formulazio, koefizienteak, geratze irizpidea,atalen hasieraketa .... Gure inplementazioak biribiltze errorea propagazioa optimotik gertu dagoela baieztatzeko, *integratzaile idealaren* soluzioarekin konparatu dugu. Aldi berean, integratzailean biribiltze errorearen estimazioa monitorizatzeko aukera garatuko dugu.

Bigarren urratsean, ekuazio sistema ez-lineala ebazteko puntu-finkoaren ordez, Newton sinplifikatuaren metodoa aztertu dugu. Gure ekarpena, Newton sinplifikatua modu eraginkorrean aplikatzeko teknika proposatzea izango da. S-ataletako IRK metodoa eta d-dimentsioko EDA badugu, Newton sinplifikatuaren metodoren iterazio bakoitzean ($sdxsd$) tamainako sistema lineala askatu behar da. Gure proposamena da, jatorrizko sistema lineala blokeka diagonala den sistema baliokide gisa berridaztea eta matrizearen egitura hau aprobetxatu sistema modu eraginkorrean askatzeko.

Hirugarren urratsean, eguzki-sistemaren epe luzeko integrazioan arituko gara. Ekarpen handiena, atalen hasieraketa berri bat aplikatzea da alde Kepleriarraren fluxuan oinarrituz. IRK metodoak eskaintzen digun malgutasunari esker eta N gorputzetako problema grabitazionalaren ezaugarriez baliatuz inplementazio ezberdinak egingo ditugu. Inplementazio hauen eraginkortasuna, egungo integratzaile simplektiko esplizituekin konparatuko ditugu.

Azken urratsean, esperimentalki, eguzki-sistemaren integrazioan denbora birparametrizazio teknikaren aplikazio sinple bat erakutsiko dugu. Integratzaile sinpletikoak luzera finkoko urratsa eduki behar du eta zentzu honetan, birparametrizazioa eraginkortasuna hobetzeko beste bide bat da.

1.6. Laburpena

2. Kapitulua

Zenbakizko Integratzaile Sinplektikoak.

2.1. Sarrera.

2.1.1. Zenbakizko metodoak.

Ekuazio diferentzial arruntetarako (*ODE*) hasierako baliodun problemen (*IVP*) formulazioa,

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad \mathbf{y}(t_0) = \mathbf{y_0}, \tag{2.1}$$

non $\mathbf{y} : \mathbb{R} \longrightarrow \mathbb{R}^{\mathbf{d}}$ soluzioa, $\mathbf{y_0} \in \mathbb{R}^{\mathbf{d}}$ hasierako balioa eta $\mathbf{f} : \mathbb{R} \times \mathbb{R}^{\mathbf{d}} \longrightarrow \mathbb{R}^{\mathbf{d}}$ bektore eremua deskribatzen funtzioa dugun ($\dot{\mathbf{y}}$ notazioa erabiliko dugu $d\mathbf{y}/dt$ adierazteko).

Goiko ekuazio-sistema bektoriala (2.1), ekuazio-sistema modu eskalarrean idatzi daiteke:

$$\begin{aligned}
\dot{y_1}(t) &= f_1(t, (y_1(t), y_2(t), \ldots, y_d(t)), \quad y_1(t_0) = y_{1,0} \\
\dot{y_2}(t) &= f_2(t, (y_1(t), y_2(t), \ldots, y_d(t)), \quad y_2(t_0) = y_{2,0} \\
&\ldots \\
\dot{y_d}(t) &= f_d(t, (y_1(t), y_2(t), \ldots, y_d(t)), \quad y_d(t_0) = y_{d,0}
\end{aligned}$$

Metodo analitikoak (funtzio ezagunen araberako soluzio zehatza) eta erdi-analitikoak, ez dira problema askoren soluzioa bilatzeko teknika egokiak. Zenbakizko metodoak, aldiz, modu errazean aplika daitezke eta horregatik, soluzio metodo nagusiena kontsideratzen da.

Zenbakizko metodo baten bidez, $\mathbf{y}(t)$ soluzioaren $\mathbf{y_n} \approx \mathbf{y}(t_n)$ hurbilpena lortuko dugu $t_n = t_{n-1} + h_n$ ($n = 1, 2, \ldots$) une diskretu ezberdinetarako. Zenbakizko soluzioa urratsez-urrats sekuentzialki eta zehaztutako tarte baterako ($t_0 \leq t \leq$

t_f) kalkulatuko dugu. Beraz, lortutako balio multzoak $(t_o, \mathbf{y_0}), (t_1, \mathbf{y_1}), \ldots, (t_f, \mathbf{y_f})$ zenbakizko soluzioa definitzen du.

Nola jakin zenbakizko soluzioa matematika modeloarekiko zuzena dela? Zenbakizko soluzioaren errorea neurtzeko teknika ezberdinak ditugu.

Azkenik argitu beharra dago ekuazio-sistema beti *sistema autonomo* moduan, hau da, denborarekiko independentea idatz daitekeela. Hori horrela izanik, notazioa sinplifikatzeko era honetako sistemak kontsideratuko ditugu,

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

Jarraian zenbakizko metodoen oinarrizko kontzeptuak eta notazioa finkatuko dugu.

1. Fluxua.

   Fase-espazioko edozein $\mathbf{y_0}$ puntuari, $\mathbf{y}(t_0) = \mathbf{y_0}$ hasierako balio duen $\mathbf{y}(t)$ soluzioa esleitzen dion mapping-ari deitzen diogu. Izendatzeko φ_t notazioa erabiliko dugu,

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

2. Zenbakizko diskretizazioa.

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

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

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

Adibideak. Ekuazio diferentzial arruntaren (2.1), t_0 unetik $t_1 = t_0 + h$ une arteko integrazioak,

$$\mathbf{y}(t_1) = \mathbf{y_o} + \int_{t_0}^{t_1} \mathbf{f}(t, \mathbf{y}(t))dt,$$

Integrala modu ezberdinean hurbilduz, zenbakizko bi metodo defini ditzakegu.

(a) Eurler metodo esplizitua.

$$\mathbf{y_1} = \mathbf{y_0} + h\,\mathbf{f}(\mathbf{t_0}, \mathbf{y_0}).$$

(b) Metodo trapezoidal inplizitua.

$$\mathbf{y_1} = \mathbf{y_0} + \frac{h}{2}\,(\mathbf{f}(t_0, \mathbf{y_0} + \mathbf{f}(t_1, \mathbf{y_1})).$$

3. Metodoaren ordena.

Definizioa. **Errore globala**. Zenbakizko soluzioaren t_0 hasierako unetik t_k une arteko errore globala $ge(t)$,

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

Definizioa. ϕ **metodoaren ordena**. h urrats luzera finkoko ϕ metodoak p ordenekoa dela esaten da, errore globala $ge(t)$ $O(h^p)$ ordenekoa bada $h \to 0$,

$$y_k - y(t_k) = O(h^p), \;\; h \to 0.$$

Definizioa. **Errore lokala**. Zenbakizko soluzioaren urrats bakarreko $[t_k, t_{k+1}]$ errore lokala $le(t)$,

$$le(t_{k+1}) = y_{k+1} - y_k(t_{k+1})$$

non $y_k(t)$, $y(t_k) = y_k$ hasierako balio lokaleko soluzio zehatza den.

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

Asymptotically, that is for small values of h, the local error is Ch^{p+1}, where C depends on the particular problem as well as the method. The value of p is thus a guide to how rapidly errors reduce as a consequence of a reduction in h (Butcher- Gauss method - encyclopedia).

4. Metodo simetrikoak.

Adjoint method. Jatorrizko metodoaren alderantzizko eta kontrako denbora $-h$ esleipenari (map), ϕ_h metodoaren ϕ_h^* *adjoint metodoa* esaten zaio.

$$\phi_h^* = \phi_{-h}^{-1}$$

$\phi_h^* = \phi_h$ betetzen denean, metodoa **simetrikoa** dela esaten da.

2.1.2. Problema motak.

1. Problema kaotikoak.

   Hasierako balio edo parametroen perturbazioekiko, diskretizazio-erroreekiko (trunkatze) edo birbitze erroreekiko esponentzialki sentikorrak diren problemei esaten zaie.

2. Problema stiff.

   Irakurri 12.1 Solution of stiff problems $(555 - 559)$.

   Irakurri 13.2.3 Convergence and consistency and 13.3 Stiffness and Implicitness (600).

   Stiff equations are problems for which explicit methods don't work (Hairer).

2.1.3. Sistema-Hamiltondarrak.

Ekuazio diferentzial arrunten (EDA) formulazio Hamiltondarra erabili ohi da errealitateko sistemak matematikoki adierazteko.

$H(\mathbf{p},\mathbf{q})$ funtzio leuna izanik, non $H : \mathbb{R}^{2d} \longrightarrow \mathbb{R}$ den eta $\mathbf{p} = (p_1, \ldots, p_d)$, $\mathbf{q} = (q_1, \ldots, q_d)$ domeinuaren aldagaiak diren. H funtzioari dagokion *Hamiltondar sistema* osatzen duten $2d$ ekuazio diferentzialak era honetan definitzen dira,

$$\begin{aligned} \frac{d}{dt}\, p_i &= -\frac{\partial H}{\partial q_i}(\mathbf{p},\mathbf{q}), \\ \frac{d}{dt}\, q_i &= \ \ \frac{\partial H}{\partial p_i}(\mathbf{p},\mathbf{q}), \quad i = 1, \ldots, d. \end{aligned}$$

$H(\mathbf{p},\mathbf{q})$ funtzioari *Hamiltondarra* esaten zaio eta integrazioan zehar konstante mantentzen da. $\mathbf{p}$ eta $\mathbf{q}$ bektoreen d dimentsioa sistemaren *askatasun maila* deritzo. Beste notazio laburtu hau ere erabili ohi da,

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

non $\mathbf{y} = (\mathbf{p},\mathbf{q})$, $\nabla H = (\partial H/\partial p_1, \ldots, \partial H/\partial p_d; \partial H/\partial q_1, \ldots, \partial H/\partial q_d)$ eta

$$J = \begin{pmatrix} 0_{dxd} & I_{dxd} \\ -I_{dxd} & 0_{dxd} \end{pmatrix}.$$

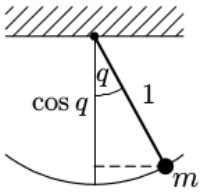
Adibidea.

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

$$H(p, q) = \frac{1}{2}p^2 - cosq.$$

Ekuazio diferentzialak,

$$\dot{p} = -sinq, \;\; \dot{q} = p.$$



(a) Pendulua.

2.1. Irudia: Pendulua.

Kang Feng With the development of the modern mechanics,physics, chemistry, and biology, it is undisputed that almost all physical processes,whether they are classical, quantum, or relativistic, can be represented by an Hamiltonian system. Thus, it is important to solve the Hamiltonian system correctly.

Hamiltondar banagarriak.

Hamiltondar banagarriak egitura bereziko sistema Hamiltondarrak ditugu. Maiz, sistema-mekanikoek era honetako Hamiltondarra dute $H(\mathbf{p}, \mathbf{q}) = T(\mathbf{p}) + U(\mathbf{q})$.

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

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

Beraz, dagokien ekuazio diferentzialak,

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

Adibidea.

Bi-gorputzen problema edo *Kepler problema*. Planoan elkar erakartzen diren bi gorputzen (adibidez eguzkia eta planeta bat) mugimendua kalkulatzeko, horietako gorputz baten kokapena koordenatu sistemaren jatorria kontsideratuko dugu eta beste gorputzaren kokapenaren koordenatuak $\mathbf{q} = (q_1, q_2)$ izendatuko ditugu. Normalizatutako Newton legearen araberako ekuazio diferentzialak,

$$\dot{p}_1 = -\frac{q_1}{(q_1^2 + q_2^2)^{3/2}}, \;\; \dot{p}_2 = -\frac{q_2}{(q_1^2 + q_2^2)^{3/2}}. \tag{2.3}$$

$$\dot{q}_i = p_i, \;\; i = 1, 2. \tag{2.4}$$

Baliokide den sistema Hamiltondarra,

$$H(p_1, p_2, q_1, q_2) = \frac{1}{2}(p_1^2 + p_2^2) - \frac{1}{\sqrt{q_1^2 + q_2^2}}. \tag{2.5}$$

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

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

e ezentrizidade ($0 \leq e < 1$) duen elipsea da, eta $P = 2\pi$ periododuna.

Hamiltondar perturbatuak.

Hamiltondar perturbatuak, honako egitura duten sistemak ditugu.

$$H = H_A + \epsilon H_B \;\; (|H_B| \ll |H_A|).$$

Adibidea.

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

2.1.4. Metodo sinplektikoak.

Zenbakizko integratzaile tradizionalak ez dituzte ekuazio diferentzialen hainbat propietate zehazki mantentzen: energia, momentua, momentu angeluarra, simetriak,...

Metodo sinpletikoak sistema Hamiltondarren soluzioen hurbilpena kalkulatzeko zenbakizko metodo egokiak dira eta eguzki-sistemaren epe luzeko integrazioetan oso erabiliak dira. Metodo sinplektikoen ezaugarri hauek azpimarratuko ditugu,

1. Metodo simetrikoak dira.

2. Urrats luzera finkoa. Metodo gehienak eraginkorrak izateko, errore estimazio baten arabera integrazioan zehar urrats luzera egokitzen dute. Integratzaile sinplektikoetan, urrats luzeera finkoa erabili behar da metodoaren propietateak ez galtzeko.

Störmer-Verlet metodoa.

$p = 2$ ordeneko metodo sinplektikoa garrantzitsua da. Izen ezberdinekin ezagguna da: *Störmer metodoa* astronomian, *Verlet metodoa* molekula dinamikoan edo *Leap-Frog metodoa* ekuazio diferentzial partzialetan. Era honetan definitzen da,

$$\begin{aligned} p_{n+1/2} &= p_n - \frac{h}{2}\nabla_q H(p_{n+1/2}, q_n) \\ q_{n+1} &= q_n + \frac{h}{2}\big(\nabla_p H(p_{n+1/2}, q_n) + \nabla_p H(p_{n+1/2}, q_{n+1})\big) \\ p_{n+1} &= p_{n+1/2} - \frac{h}{2}\nabla_q H(p_{n+1/2}, q_{n+1}) \end{aligned} \tag{2.6}$$

edo

$$\begin{aligned} q_{n+1/2} &= q_n + \frac{h}{2}\nabla_p H(p_n, q_{n+1/2}) \\ p_{n+1} &= p_n - \frac{h}{2}\big(\nabla_q H(p_n, q_{n+1/2}) + \nabla_q H(p_{n+1}, q_{n+1/2})\big) \\ q_{n+1} &= q_{n+1/2} + \frac{h}{2}\nabla_p H(p_{n+1}, q_{n+1/2}) \end{aligned} \tag{2.7}$$

Bigarren ordeneko ekuazio diferentziala denean, metodoa esplizitua da eta modu honetan labur daiteke,

$$\begin{aligned} p_{n+1/2} &= p_n + \frac{h}{2} f(q_n) \\ q_{n+1} &= q_n + h\, p_{n+1/2} \\ p_{n+1} &= p_{n+1/2} + \frac{h}{2}\, f(q_{n+1}) \end{aligned} \tag{2.8}$$

edo

$$
\begin{aligned}
q_{n+1/2} &= q_n + \frac{h}{2} p_n \\
p_{n+1} &= p_n + h\, f(q_{n+1/2}) \\
q_{n+1} &= q_{n+1/2} + \frac{h}{2}\, p_{n+1}
\end{aligned}
\tag{2.9}
$$

2.2. Gauss metodoak.

Runge-Kutta metodo sinplektiko interesgarriena *Gauss* metodoa da. Metodo inplizitua da eta s-ataleko orden altueneko metodoa ($p = 2s$). ++

2.2.1. Runge-Kutta metodoak.

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

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

Hasierako baliodun problema (2.1) baten $\mathbf{y}(t)$ soluzioaren $\mathbf{y}_n \approx \mathbf{y}(t_n)$ hurbilpena era honetan kalkulatzen da,

$$
\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i\, \mathbf{f}(t_n + c_i h, \mathbf{Y}_{n,i}) \ , \tag{2.11}
$$

non $\mathbf{Y}_{n,i}$ atalak era honetan definitzen diren,

$$
\mathbf{Y}_{n,i} = \mathbf{y}_n + \ h \sum_{j=1}^{s} a_{ij}\, \mathbf{f}(t_n + c_j h, \mathbf{Y}_{n,j}) \quad i = 1, \dots, s. \tag{2.12}
$$

Metodo esplizituak (ERK) eta inplizituak (IRK).

Runge-Kutta bi mota nagusi bereizi ditzakegu: esplizituak (*ERK*) non $\forall i \geq j,\ a_{ij} = 0$ eta inplizituak (*IRK*) non $\exists i \geq j\ , a_{ij} \neq 0$.

Adibidea: ERK lau-ataletako metodo klasikoa.

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

Atalak $\mathbf{Y}_{n,i}$ esplizituki kalkula daitezke,

$$\mathbf{Y}_{n,i} = \mathbf{y}_n + \ h \sum_{j=1}^{i-1} a_{ij}\ \mathbf{f}(t_n + c_j h, \mathbf{Y}_{n,j}) \qquad i = 1, \ldots, s.$$

Adibidea: IRK metodoa (Implicit Midpoint method).

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

$\mathbf{Y}_{n,1}$ atalaren balioa kalkulatzeko, honako ekuazio ez-linealaren ebazpena egin behar da,

$$\mathbf{Y}_{n,1} = \mathbf{y}_n + \ \frac{h}{2}\ \mathbf{f}(t_n + \frac{h}{2}, \mathbf{Y}_{n,1}).$$

ERK lau-ataletako metodo klasikoa, $p = 4$ ordeneko dugu. Orden altuko *ERK* metodoak aurkitzea konplexua da, koefizienteek bete behar dituzten baldintza kopurua esponentzialki hazten baitira. Orden altuko *(*ERK) metodo hauek aurkitu dira: $p = 8$ ordeneko metodoa $s = 11$ atalekin, $p = 10$ ordeneko metodoa $s = 17$ atalekin eta $p = 12$ ordeneko metodoa $s = 25$ atalekin.

IRK metodoak *ERK* metodoak baino modu errazagoan eraiki daitezke. *Butcher sinplifikazio baldintzen* [5] arabera definitzen dira,

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

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

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

2.1. Taula: Runge-Kutta metodo inplizituak.

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

Gauss metodoa.

Aurreko taulan (Taula 2.1.) *IRK* metodo ezagunenak laburtu ditugu. Lehenik, Gauss metodoaren bi ezaugarri azpimarratu nahi ditugu: Runge-Kutta metodo sinplektiko bakarra eta s-ataletako orden altueneko ($p = 2s$) IRK da.

Jarraian, Gauss metodoaren ezaugarri orokorrak azalduko ditugu:

1. Metodo sinplektikoa da.

   Sanz-Sernak [19] Runge-Kutta metodoaren koefizienteek,

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

   baldintza betetzen badute metodoa sinplektikoa dela frogatu zuen.

2. Metodo simetrikoa.

   $$b_i = b_{\sigma(i)}, \; c_{\sigma(i)} = 1 - c_i, \quad i = 1, 2, \dots \lceil s/2 \rceil \tag{2.14}$$

   $$b_j = a_{\sigma(i),\sigma(j)} a_{i,j}, \quad i = 1, 2, \dots \lceil s/2 \rceil \tag{2.15}$$

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

3. Orden altuko metodoa. Gauss metodoa edozein ordenekoa izan daiteke. Doitasun handiko konputazioetarako orden altuko metodoak behar dira: doitasun bikoitzeko aritmetikan ($u \approx 10^{-16}$) $p \geq 8$ ordeneko metodoak eta doitasun laukoitzeko aritmetikan ($u \approx 10^{-35}$) oraindik orden altuagoko metodoak gomendagarriak dira.

4. Metodo orokorra. Gauss metodoa edozein ekuazio diferentzialari aplika daiteke. Sistema Hamiltondarren problemetan, ez du zertan banagarria izan behar.

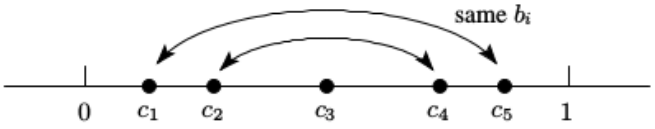


Fig. 2.1. Symmetry of collocation methods

(a) Kolokazio metodoen simetria.

2.2. Irudia: Kolokazio metodoen simetria.

5. Paralelizagarria. Ekuazio diferentzial garestiak ditugunean, s-ataletako funtzio konputazioak ($f(Y_i),\ i = 1, \ldots, s$) paraleloan kalkula daitezke.

6. Kolokazio metodoa. Zenbakizko soluzioa diskretizazio puntuetan ez ezik, integrazio tarte bakoitzean polinomio interpolatzaile batek modu jarraian emandako soluzioa adierazten du.

7. A-stability and B-stability. A-stable methods have a central role in the numerical solution of stiff problems and Gauss methods are likely candidates.

Gauss metodoaren desabantailak?

Adibidea. $s = 1$, $s = 2$ eta $s = 3$ ataletako Gauss metodoak.

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

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

IRK algoritmo-I orokorra.

IRK metodoen erronka handiena ekuazio-sistema ez-linealaren zenbakizko soluzioaren inplementazio eraginkorra da. Nonstiff problemetarako, atalen hasieraketa ($Y_i^{[0]}$) egoki bat duen puntu-finkoko iterazioa erabil daiteke. Stiff probleme-

tarako, puntu-finkoa iterazioak urrats tamaina txikiegia izatea behartuko luke eta ondorioz, Newton sinplifikatua erabili ohi da.

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

ALGORITHM 1: IRK Algoritmo orokorra

Algoritmo nagusiko agindu bakoitzari ohar moduko egingo diogu, IRK metodoaren hainbat zehaztapen emateko helburuarekin.

1. Hasieratu $Y_{n,i}^{[0]}$.

   Atalen hasieraketa egokia definitu behar da. Aukera sinpleena $Y_{n,i}^{[0]} = y_{n-1}$ hasieratzea da baina aurreko urratseko atalen informazioa erabiliz hurbilketa hobea lortu daiteke. Aurreko urratseko atalen polinomio interpolatzailearen bidezko hasieraketa era honetan adierazi dezakegu $Y_{n,i}^{[0]} = g(Y_{n-1,i})$, $i = 1, \ldots, s$.

2. $F_{n,i} = f(t_n + c_i h, Y_{n,i})$.

   Atal bakoitzaren ($i = 1, \ldots, s$) ekuazio diferentzialaren balioztapena independentea da eta paraleloan exekutatu daiteke.

3. Askatu $(Y_{n,i} = y_n + h \sum_{i=1}^{s} a_{ij} F_{n,j}$, $i = 1, \ldots, s)$.

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

(a) Puntu-finkoko iterazioa.

for *(k=1,2,...)* **do**

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

end

ALGORITHM 2: Puntu-finkoko iterazioa.

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

(b) Newton metodoaren iterazioa.

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

for *(k=1,2,...)* **do**

$$r_i^{[k]} = -Y_i^{[k-1]} + y_n + h \sum_{j=1}^{s} a_{ij} f(t_n + c_i h, Y_j^{[k-1]});$$
$$\text{Askatu } \triangle Y_i^{[k]} - h \sum_{j=1}^{s} a_{ij} \frac{\partial f}{\partial y}(t_n + c_j h, Y_j^{[k-1]}) \triangle Y_i^{[k]} = r_i^{[k]};$$
$$Y_i^{[k]} = Y_i^{[k-1]} + \triangle Y_i^{[k]}, \quad i = 1, \ldots, s;$$

end

ALGORITHM 3: Newton metodoaren iterazioa

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

4. $y_{n+1} = y_n + \delta_n$

Zenbakizko integrazio luzeetan urrats asko ematen direnez, urratsero batura errekurtsibo honen konputazioa beharrezkoa da. Normalean $|\delta_n| \ll |y_n|$ izango da eta batura honetan doitasun galera gertatuko da. Hau ekiditeko *batura konpensatu* teknika erabili ohi da.

IRK algoritmoa-II (Hamiltondar banagarriak).

Era honetako ekuazio diferentzialak garrantzitsuak dira,

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

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

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

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

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

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

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

for *(k=1,2,...)* **do**

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

end

ALGORITHM 4: Puntu-finkoaren iterazioa (Gauss-Seidel).

IRK algoritmoa-III (bigarren ordeneko EDA).

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

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

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

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

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

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

IRK algoritmoa-III (bigarren ordeneko EDA).

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

Hasieratu $Q_{n,i}^{[0]}\ ,\ \ i = 1, \ldots, s$;

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

$F_{n,i} = f(t_n + c_i h, Q_{n,i})\ ,\ \ i = 1, \ldots, s$;

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

end

$\delta p_n = h \sum_{i=1}^{s} b_i F_{n,i}$;

$\delta q_n = h^2 \sum_{i=1}^{s} \beta_i F_{n,i}$;

$p_{n+1} = p_n + \delta p_n$;

$q_{n+1} = q_n + h\gamma_i p_n + \delta q_n$;

end

ALGORITHM 5: IRK algoritmoa-III (bigarren ordeneko EDA)

Bigarren ordeneko ekuazio diferentzialak ditugunean, puntu-finkoko iterazioa,

for *(k=1,2,...)* **do**

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

end

ALGORITHM 6: Puntu-finkoko iterazioa (bigarren ordeneko EDA)

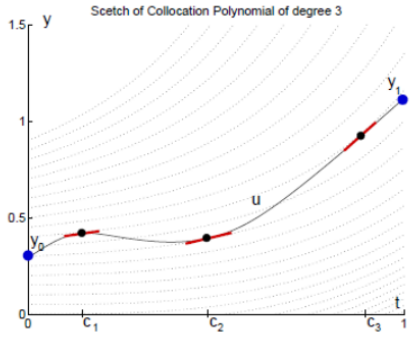
2.2.2. Kolokazio metodoak.

Kolokazio metodoak ekuazio diferentzialen zenbakizko soluzioa azaltzeko beste modu bat dira. Gauss metodoak kolokazio metodoak ditugu eta hauen abantaila da, zenbakizko soluzioa diskretizazio puntuetan ez ezik, polinomio interpolatzaile batek modu jarraian emandako soluzioa adierazten duela. Honako definizioa emango dugu,

2..1 Definizioa $c_1, c_2, \ldots, c_s \ \ (0 \leq c_i \leq 1)$ *zenbaki errealak izanik, s-mailako* $u(t)$ kolokazio polinomioak *honakoa betetzen du,*

$$u(t_0) = y_0,$$
$$\dot{u}(t_0 + c_i h) = f(t_0 + c_i h, u(t_0 + c_i h)), \ \ i = 1, \ldots, s.$$

Orduan soluzioa $y_1 = u(t_0 + h) \approx y(t_0 + h)$ *da.*



(a) kolokazio metodoak.

2.3. Irudia: kolokazio metodoak.

2..1 Teorema ***Theorem 1.4 (Guillou and Soule 1969, Wright 1970).*** *Kolokazio metodoaren definizioa eta jarraian emandako moduan kalkulatutako s-ataleko Runge-Kutta metodoa baliokideak dira.*

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

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

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

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

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

Gauss metodoaren koefizienteak kalkulatzeko bi aukera ditugu. Lehen aukera, *Mathematicak* NDSolve paketean inplementatuta dagoen koefizienteak kalkulatzeko funtzioa: ImplicitRungeKuttaGaussCoefficients[]. Bigarren aukera, guk koefizienteak kalkulatzeko garatutako funtzioa erabiltzea.

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

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

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

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

      glist = Table[g[i] = Collect[ff, f[_],
                          Simplify[∫_0^c[i] #1 dx] &], {i, 1, s}];
      yy = Collect[ff, f[_], Simplify[∫_0^1 #1 dx] &];

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

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

    ]
```

2.4. Irudia: Code Gauss.

2.3. Konposizio eta Splitting metodoak.

2.3.1. Sarrera.

Konposizio eta Splitting ideietan oinarrituz, aplikazio eremu ezberdinetarako hainbat integratzaile sinplektiko garatu dira. Metodo hauek ez dira orokorrak, problema zehatzetan aplikagarriak baizik, eta metodo oso eraginkorrak dira.

2.3.2. Konposizio metodoak.

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

Definizioa orokorra.

ϕ_h oinarrizko metodoa eta $\gamma_1, \ldots, \gamma_s$ zenbaki errealak emanik, urrats luzera hauen $\gamma_1 h, \gamma_2 h, \ldots, \gamma_s h$ konposaketari dagokion konposizio metodoa,

$$\Psi_h = \phi_{\gamma_s h} \circ \cdots \circ \phi_{\gamma_{1h}}. \tag{2.17}$$

Algoritmoa.

s-ataletako konposizio metodoen algoritmo orokorra honakoa izango litzateke:

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

ALGORITHM 7: Konposizio metodoak.

Konposizio metodoei buruzko hainbat ohar azpimarratuko ditugu:

1. Esplizitua.

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

2.2. Taula: 10 ordeneko metodoa konposizio metodoa (CO1035).

Koefiziente	Balioa	Koefizientea	Balioa
$\gamma_1 = \gamma_{35}$	0.078795722521686419263907 68	$\gamma_{10} = \gamma_{26}$	-0.399105630136035897878629 81
$\gamma_2 = \gamma_{34}$	0.313096103415108527764812 47	$\gamma_{11} = \gamma_{25}$	0.103087398527471077315802 77
$\gamma_3 = \gamma_{33}$	0.027918383235078066109520 27	$\gamma_{12} = \gamma_{24}$	0.411430873955890237820704 12
$\gamma_4 = \gamma_{32}$	-0.229592841593907094151213 40	$\gamma_{13} = \gamma_{23}$	-0.004866360583135261762195 66
$\gamma_5 = \gamma_{31}$	0.130962061077164863174656 86	$\gamma_{14} = \gamma_{22}$	-0.392033353708639906448081 94
$\gamma_6 = \gamma_{30}$	-0.269733405654510714344609 73	$\gamma_{15} = \gamma_{21}$	0.051942502962449647037182 90
$\gamma_7 = \gamma_{29}$	0.074973343135891435666137 11	$\gamma_{16} = \gamma_{20}$	0.050665090759924496335874 34
$\gamma_8 = \gamma_{28}$	0.111993423999810204889575 08	$\gamma_{17} = \gamma_{19}$	0.049674370639729879054568 80
$\gamma_9 = \gamma_{27}$	0.366133449546226751193148 12	γ_{18}	0.049317735759594537917680 01

2. Sekuentziala.

   Azpi-urrats bakoitzaren kalkulua modu sekuentzialean egin behar dugu.

3. Memoria gutxi.

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

4. Oinarrizko metodoa: Störmer-Verlet.

   Bigarren ordeneko ekuazio diferentzialak ditugunean, Störmer-Verlet integratzailean oinarritzen diren konposizio metodoekin urrats bakoitzean s ekuazio diferentzialaren balioztapena egin behar ditugu.

CO1035: 10 **ordeneko konposizio metodoa.**

Sofronio eta Spalettaren (2004), $s = 35$ eta $p = 10$ ordeneko metodoa [30], orain arteko lortutako orden altueneko konposizio metodo eraginkorrena kontsideratu daiteke (taula 2.2.). *IRK* metodoa konposizio metodo honekin alderatuko dugu.

Konposizio metodo simetrikoa da. Oinarrizko metodoa simetrikoa eta $p = 2$ dela baliatuz eraikitako metodoa da. Integrazio metodo hau aplika daitekeen problemak, Hamiltondarra $H(q, p) = T(p) + U(q)$ modukoa izan behar dira (????).

Gure inplementazioa. Gure abiapuntua Haireren konposizio metodoaren Fortran kodea izan da. Konposizio metodoaren azalpenak liburuko [14] *II.4* eta *V.3* ataletan ematen ditu. *GNI-Comp* izeneko kodea eskuragarri dago [**?**] helbidean. Kodearen C-lengoaiako bertsioa garatu dugu.

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

$$\Psi_h = \phi_{\gamma_s h} \circ \phi_{\gamma_s - 1 h} \circ \cdots \circ \phi_{\gamma_{2h}} \circ \phi_{\gamma_{1h}} \tag{2.18}$$

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

Koefizienteak. Konposizio metodoaren oinarrizko metodoa *Stömer-Verlet* $\phi_h = \varphi^1_{h/2} \circ \varphi^2_h \circ \varphi^1_{h/2}$ dugula kontutan hartuta,

$$\Psi_h = (\varphi^1_{h\gamma_s/2} \circ \varphi^2_{h\gamma_s} \circ \varphi^1_{h\gamma_s/2}) \circ \cdots \circ (\varphi^1_{h\gamma_2/2} \circ \varphi^2_{h\gamma_2} \circ \varphi^1_{h\gamma_2/2}) \circ (\varphi^1_{h\gamma_1/2} \circ \varphi^2_{h\gamma_1} \circ \varphi^1_{h\gamma_1/2})$$

Beraz jarraian dauden φ^1 fluxuak elkartuz,

$$\Psi_h = \varphi^1_{ha_{s+1}} \circ \varphi^2_{hb_s} \circ \cdots \circ \varphi_{ha_3} \circ \varphi^2_{hb_2} \circ \varphi^1_{ha_2} \circ \varphi^2_{hb_1} \circ \varphi^1_{ha_1}$$

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

Tarteko urratsetan, lehen atala $\varphi^1_{ha_1}$ eta azkena $\varphi^1_{ha_1}$ bakar batean elkar daitezke,

$$\Psi_h = \varphi^1_{h2a_{s+1}} \circ \varphi^2_{hb_s} \circ \cdots \circ \varphi_{ha_3} \circ \varphi^2_{hb_2} \circ \varphi^1_{ha_2} \circ \varphi^2_{hb_1}.$$

2.3.3. Splitting metodoak.

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

Maiz, jatorrizko $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$ problema era honetan bana daiteke,

$$\dot{\mathbf{y}} = \mathbf{f}^{[\mathbf{1}]}(\mathbf{y}) + \mathbf{f}^{[\mathbf{2}]}(\mathbf{y}), \tag{2.19}$$

non $\dot{\mathbf{y}} = \mathbf{f}^{[\mathbf{1}]}(\mathbf{y})$ eta $\dot{\mathbf{y}} = \mathbf{f}^{[\mathbf{2}]}(\mathbf{y})$ sistemen fluxu zehatzak, $\varphi^{[1]}_t$ eta $\varphi^{[2]}_t$ esplizituki kalkula daitezkeen.

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

$$\phi_h = \varphi^{[1]}_h \circ \varphi^{[2]}_h \quad edo \quad \phi^*_h = \varphi^{[2]}_h \circ \varphi^{[1]}_h. \tag{2.20}$$

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

$$\phi_h = \varphi^{[1]}_{h/2} \circ \varphi^{[2]}_h \circ \varphi^{[1]}_{h/2} \tag{2.21}$$

Splittig metodo orokorrak. Konposizio metodoen modu berean, oinarrizko Splitting metodoak konposatuz orden altuagoko metodoak lortzen dira,

$$\Psi_h = \varphi^{[1]}_{a_{s+1}h} \circ \varphi^{[2]}_{b_s h} \circ \varphi^{[1]}_{a_s h} \circ \cdots \circ \varphi^{[1]}_{a_2 h} \circ \varphi^{[2]}_{b_1 h} \circ \varphi^{[1]}_{a_1 h}. \tag{2.22}$$

a_i, b_i koefizienteek metodoaren ordena definitzen dute. Metodoa simetrikoa bada $\Psi_h = \Psi^*_h$,

$$a_1 = a_{s+1},\ b_1 = b_s,\ a_2 = a_s, b_2 = b_{s-1}, \ldots$$

Algoritmoa.

Splitting metodoen algoritmo orokorra honakoa izango litzateke:

for $n \leftarrow 0$ **to** *(endstep-1)* **do**
  $Y_{0,n} = y_{n-1}$;
  for *i=1,2,...,s* **do**
    $Y_{i,n} = (\varphi^{[2]}_{b_i h} \circ \varphi^{[1]}_{a_i h})(Y_{i-1,n})$;
  end
  $y_{n+1} = Y_{s,n}$;
end

ALGORITHM 8: Splitting metodoak.

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

Fluxu zehatza eta zenbakizko fluxua konbinatuz.

Demagun sistema $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$ era honetan banatzen dugula,

$$\dot{\mathbf{y}} = \mathbf{f}^{[\mathbf{1}]}(\mathbf{y}) + \mathbf{f}^{[\mathbf{2}]}(\mathbf{y}), \tag{2.23}$$

Suposatu bakarrik $\dot{\mathbf{y}} = \mathbf{f}^{[\mathbf{1}]}(\mathbf{y})$ sistemaren fluxua zehatza $\varphi^{[1]}_t$ kalkulatu daitekeela eta $\phi^{[2]}_t$, $\dot{\mathbf{y}} = \mathbf{f}^{[\mathbf{2}]}(\mathbf{y})$ sistemari aplikatutako zenbakizko integrazio dugula. Orduan konposizio metodoaren oinarrizko metodoa honakoa kontsideratu daiteke,

$$\phi_h = \varphi^{[1]}_h \circ \phi^{[2]}_h, \qquad \phi^*_h = \phi^{[2]*}_h \circ \varphi^{[1]}_h.$$

Splitting metodoak konposizio metodoen interpretazioa emanez,

$$\Psi_h = \varphi^{[1]}_{a_s h} \circ \phi^{[2]}_{a_s h} \circ \phi^{[2]*}_{b_s h} \circ \varphi^{[1]}_{(b_s + a_s - 1)h} \circ \phi^{[2]}_{a_s h} \circ \cdots \circ \phi^{[2]*}_{b_1 h} \circ \varphi^{[1]}_{b_1 h}. \tag{2.24}$$

2.3. Taula: $SABA_4$ splitting metodoa.

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

Eguzki-sistemari egokitutako splitting metodoak.

Honakoa dugu N-gorputzeko problema grabitazionalaren Hamiltondarra,

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

Koordenatu sistema egokia (*Jacobi* edo koordenatu Heliozentrikoak) erabiliaz, Hamiltondarra beste modu honetan berridatzi daiteke,

$$H = H_K + H_I, \;\; |H_I| \ll |H_K|,$$

non alde nagusia H_K planeta bakoitzaren eguzki inguruko mugimendu kepleriarra den eta H_I aldiz, planeten arteko interakzioek eragiten duten perturbazio txikia.

Eguzki-sistemaren N-gorputzeko problema grabitazionalari egokitutako zenbakizko bi integratzaile sinplektiko azalduko ditugu. Lehena, Laskarrek eta Robutelek [21] definitutako $SABAC_4$ integratzailea eta bigarrena, Blanes-ek [3] [10] definitutako $ABAH1064$ integratzailea.

1. Laskarren (2001) $SABAC_4$ zenbakizko integratzailea [21]. $SABA_4$ integratzailea definitzen duten koefizienteak (taula 2.3.).

   Hamiltondarra, $H = H_A + \epsilon H_B$ izanik eta goiko notazioa erabiliz, era honetan definituko dugu metodoa,

   $$SABA_4 = \varphi^{[A]}_{c_1 h} \circ \varphi^{[B]}_{d_1 h} \circ \varphi^{[A]}_{c_2 h} \circ \varphi^{[B]}_{d_2 h} \circ \varphi^{[A]}_{c_3 h} \circ \varphi^{[B]}_{d_2 h} \circ \varphi^{[A]}_{c_2 h} \circ \varphi^{[B]}_{d_1 h} \circ \varphi^{[A]}_{c_1 h}.$$

   Corrected integrator. Urrats bat gehitutako integratzailea $SABAC_4$,

   $$SABAC_4 = \varphi[B]_{\frac{-c}{2}} \circ SABA_4 \circ \varphi[B]_{\frac{-c}{2}},$$

   non $c = 0.00339677504820860133153215778349214 4$.

2.4. Taula: $ABAH1064$ splitting metodoa.

Koefiziente	Balioa	Koefiziente	Balioa
$a_1 = a_9$	0.047319086976533822704043717963208132509 88	$b_1 = b_9$	0.119688462458532203531286429748989214385 2
$a_2 = a_8$	0.265110523574878515953948003618569320107 8	$b_2 = b_8$	0.375295585537937425042012853768750319945 1
$a_3 = a_7$	−0.009976522883811240843267468164812380613143	$b_3 = b_7$	−0.468459341832599378365082040980538174060 5
$a_4 = a_6$	−0.059929199734941551263952479877296760040 16	$b_4 = b_6$	0.335139734275589701039309894294956904927 5
a_5	0.257476112067340453449228226460331688035 6	b_5	0.276671119121080097504945726335683469605 5

2. $ABAH1064$ (Blanes, 2013).

   Eguzki sistemaren integraziorako koordenatu Heliozentrikoei dagokion Hamiltondarra era honetakoa dugu,

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

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

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

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

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

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

2.3.4. Kepler fluxua.

Bi gorputzen problema edo Kepler problemari dagokion Hamiltondarra,

$$H(\mathbf{p},\mathbf{q}) = \frac{\mathbf{p^2}}{\mathbf{2m}} - \frac{\mu}{\|\mathbf{q}\|}. \tag{2.25}$$

Elkar erakartzen diren bi gorputzen mugimendua kalkulatzeko, gorputz baten kokapena koordenatu sistemaren jatorria kontsideratuko dugu. Notazio hau finkatuko dugu,

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

Hamiltondarrari dagokion bigarren ordeneko ekuazio diferentzialak,

$$\ddot{\mathbf{q}} = -\frac{k\mathbf{q}}{\|\mathbf{q}\|^{\mathbf{3}}}, \tag{2.26}$$

non $k = \mu/m$ eta $\mathbf{q} \in \mathbb{R}^3$.

Inplementazioa

Ideia nagusia. Koordenatu cartesiarretatik koordenatu eliptikoetara (a, e, i, Ω, E) itzulpena egingo dugu. Kontutan hartuta E (izena??) aldagai ezik beste aldagaiek konstante mantentzen direla, E_0 abiapuntua harturik, $\triangle t$ denbora tartea aurrera egingo dugu E_1 balioa berria kalkulatzeko. Azkenik, koordenatu eliptikoetatik koordenatu cartesiarrak berreskuratuko ditugu kokapen eta abiadura berriekin.

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

$$\downarrow \triangle t$$

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

Newton metodoa. Kepler-en ekuazioan oinarrituz ($E - e \sin E = n(t - t_p)$), $E_1 = \triangle E + E_0$ balioa kalkulatuko Newtonen metodoa aplikatuz,

$$f(\triangle E) = \triangle E - ce\sin(\triangle E) - se(\cos(\triangle E) - 1) - n\triangle t = 0$$
$$\triangle E^{[k+1]} = \triangle E^{[k]} - \frac{f(\triangle E^{[k]})}{f'(\triangle E^{[k]})} \tag{2.27}$$

Ekuazioak. Gure inplementazioan erabilitako ekuazio guztien azalpenak eta definizoak eranskinean eman ditugu.

2.4. Laburpena.

Metodo sinplektikoei buruzko liburu monografiko hauek gomendatuko ditugu: Sanz-Serna and M.P. Calvo's Numerical Hamiltonian Problems (1994) [19]; E. Hairer, C. Lubich and G. Wanner's Geometrical Numerical Integration (2001) [14]; B. Leimkuhler and S. Reich's Simulating Hamiltonian Dynamics (2004) [22]; Feng, Kang and Qin, Mengzhao Symplectic geometric algorithms for hamiltonian systems (2010) [11].

h

2.5. Taula: Integrazio metodoen laburpena

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

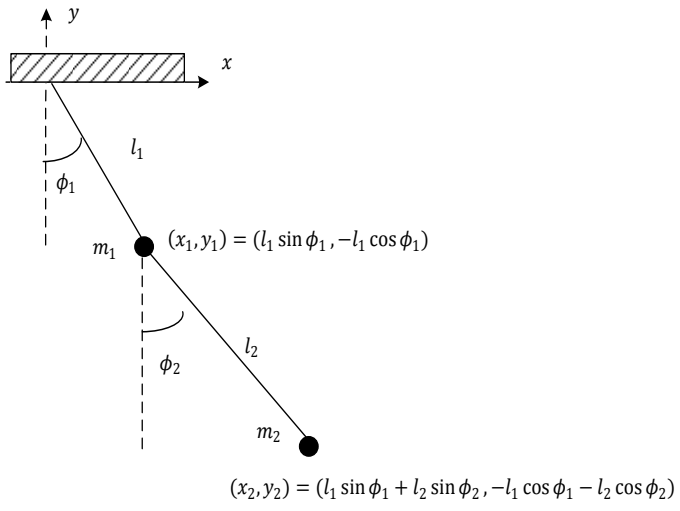
3. Kapitulua

Problemak.

3.1. Sarrera.

3.2. Pendulu bikoitza.

Planoan pendulu bikoitzaren problema era honetan definitzen da: m_1,m_2 masadun bi pendulu eta l_1, l_2 luzerako makilez (masa gabekoak kontsideratuko ditugunak) elkar lotuta. Penduluen aldagai-egoerak bi angelu (Θ_1,Θ_2) eta dagokion momentuak (P_1,P_2) dira.



3.1. Irudia: Pendulu bikoitza.

3.2.1. Ekuazioak.

Hamiltondarra.

$$q = (\Theta_1, \Theta_2) \ , \ p = (P_1, P_2) \ ,$$

$$H(q,p) = \left(\frac{C1\ P_1^2 + C2\ P_2^2 + C3\ P_1\ P_2\ \cos(\Theta_1 - \Theta_2)}{(C4 + C5\ \sin^2(Q_1 - Q_2))} \right) - C6\ \cos(\Theta_1) - C7\ \cos(\Theta_2),$$

non,

$$\begin{aligned} C1 &= l_2^2 * m_2, & C2 &= l_1^2 * (m_1 + m_2), \\ C3 &= -2 * l_1 * l_2 * l_2, & C4 &= 2 * l_1^2 * l_2^2 * m_2 * m_1, \\ C5 &= 2 * m_1^2 * l_2^2 * m_2^2, & C6 &= g * l_1 * (m_1 + m_2), \\ C7 &= g * l_2 * m_2. \end{aligned}$$

Ekuazio diferentzialak.

$$\begin{aligned} \dot{\Theta}_1 &= \frac{2 * C1 * P1 + C3 * \cos(Q1 - Q2) * P2}{aux1}, \\ \dot{\Theta}_2 &= \frac{(2 * C2 * P2 + C3 * \cos(Q1 - Q2) * P1)}{aux1}, \\ \dot{P}1 &= -(aux4 + C6 * \sin(Q1)), \\ \dot{P}1 &= (aux4 - C7 * \sin(Q2)). \end{aligned}$$

non,

$$\begin{aligned} aux1 &= C4 + C5 * \sin(Q1 - Q2) * \sin(Q1 - Q2), \\ aux2 &= C3 * \cos(Q1 - Q2), \\ aux3 &= 2 * C5 * \sin(Q1 - Q2) * \cos(Q1 - Q2), \end{aligned}$$

$$\begin{aligned} aux4 = (-1/aux2) * (C1 * P1^2 + C2 * P2^2 + P1 * P2 * aux2 * aux3) \\ - (C3 * P1 * P2 * \sin(Q1 - Q2))/aux1. \end{aligned}$$

3.2.2. Hasierako balioak.

Sistemaren parametroak. Gure esperimentuetarako honako parametroak kontsideratuko ditugu,

$$\begin{aligned} g &= 9.8\ m/sec^2, \\ l_1 &= 1\ m\ ,\ l_2 = 1\ m\ , \\ m_1 &= 1\ kg\ ,\ m_2 = 1\ kg. \end{aligned}$$

Hasierako balioak. Pendulu bikoitza izaera kaotikoa duen sistema ez-lineala da. Zentzu honetan bi hasierako balio ezberdin kontsideratu ditugu [29]:

1. Hasierako balio ez-kaotikoak:
   $q(0) = (1.1,\ 0)\ ,\ p(0) = (0,\ 2.7746).$
2. Hasierako balio kaotikoak:
   $q(0) = (0,\ 0)\ ,\quad p(0) = (0,\ 3.873).$

3.2.3. Kodeak.

Mathematican DoublePendulum.m paketean honako funtzioak inplementatu ditugu:

1. Hamiltondarra: DoublePendulumHam.
2. EDA: DoublePendulumODE.
3. Jakobiarra: DoublePendulumJAC.

C-lengoaian GaussUserProblem.c fitxategian honako funtzioak inplementatu ditugu:

1. Hamiltondarra: HamPendulum().
2. EDA: OdePendulum().
3. Jakobiarra: JacPendulum().

3.3. N-Body problema.

N-gorputzeko problema grabitazionalari dagokionez, Eguzki sistemaren eredu sinplea integratuko dugu. Eguzki-sistemaren gorputzak masa puntualak kontsideratuko ditugu eta gure ekuazio diferentzialek, soilik gorputz hauen arteko erakarpen grabitazionalak kontutan hartu ditugu. Beraz, eguzki-sistemaren eredu konplexuagoetako erlatibitate efektua, gorputzen formaren eragina, eta beste zenbait indar ez-grabitazionalak ez dira kontutan hartu.

$(N+1)$ gorputz kopurua izanik, $\mathbf{q_i}, \mathbf{p_i} \in \mathbb{R}^3$, $m_i \in \mathbb{R},\ \ i = 0, \ldots, N$ gorputz bakoitzaren kokapena, momentua eta masa dira. Bestalde , momentua era honetan definituko dugu $\mathbf{p_i} = m_i * \mathbf{v_i}$ non $\dot{\mathbf{q}}_\mathbf{i} = \mathbf{v_i}$ den.

3.3.1. Zailtasunak.

Kontutan hartzekoa da eguzki-sistemaren integrazioaren urrats kopuru handia. Eguzki-sistemaren bizi iraupena 5×10^9 urtetakoa izanik eta integrazioetako ohiko urratsa $h = 0.0025$ urteko bada (orbita txikieneko periodoaren $\%1$), eman beharreko urrats kopurua 2×10^{12} izango da. Era berean, kanpo planeten integrazioa 5×10^{10} urrats kopuru ingurukoa da.

Eguzki-sistemaren gorputzen denbora eskala oso ezberdinak dira. Gainera ilargia kontsideratuko bagenu, lurrarekiko distantzia $D_M = 3.844 \times 10^8$ eta periodoa $P_M = 27.32$ egunekoa.

3.1. Taula

Planeta	Axis a AU	Periodoa years
Merkurio	0.39	0.24
Venus	0.72	0.007
Earth	1.00	1.007
Mars	1.52	1.88
Jupiter	5.20	11.86
Saturn	9.54	29.42
Uranus	19.19	83.75
Neptune	30.06	163.72
Pluto	39.53	248.02

3.3.2. Ekuazio barizentrikoak.

Hamiltondarra.

$$H(q,p) = \frac{1}{2}\sum_{i=0}^{N}\frac{\|p_i\|^2}{m_i} - G\sum_{0\le i<j\le N}^{N}\frac{m_i m_j}{\|q_i - q_j\|} \tag{3.1}$$

Ekuazio diferentzialak.

$$\dot{q_i} = v_i, \;\; i = 0, 1, \ldots, N \tag{3.2}$$

$$\dot{v_i} = \sum_{j=0, j\neq i}^{N}\frac{Gm_j}{\|q_j - q_i\|^3}(q_j - q_i), \;\; i = 0, 1, \ldots, N \tag{3.3}$$

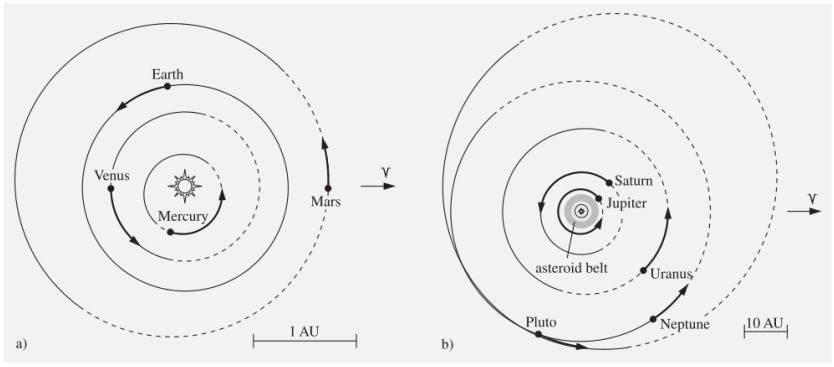


Fig. 7.1. (a) Planetary orbits from Mercury to Mars. The dashed line represents the part of the orbit below the ecliptic; the arrows show the distances travelled by the planets during one month (January 2000). (b) Planets from Jupiter to Neptune and the dwarf planet Pluto. The arrows indicate the distances travelled by the planets during the 10 year interval 2000–2010.

(a) Eguzki-sistema.

3.2. Irudia: Eguzki-sistema.

Ekuazio diferentzialak. Eguzkiaren erlatibitate efektua kontutan hartzen duten ekuazio diferentzialak azalduko ditugu.

$$\dot{q}_i = v_i,\ i = 0, 1, \dots N \tag{3.4}$$

$$\begin{aligned}\dot{v}_i = \sum_{j=0, j\neq i}^{N} \frac{Gm_j}{\|q_j - q_i\|^3}(q_j - q_i)\Bigg(1 - \frac{2(\beta+\gamma)}{c^2}\sum_{k=0,k\neq i}^{N}\frac{Gm_k}{\|q_k - q_i\|} - \frac{2\beta-1}{c^2}\sum_{k=0,k\neq j}^{N}\frac{Gm_k}{\|q_k - q_j\|} \\ + \gamma\Big(\frac{v_i}{c}\Big)^2 + (1+\gamma)\Big(\frac{v_j}{c}\Big)^2 - \frac{2(1+\gamma)}{c^2}v_i\, v_j \\ - \frac{3}{2c^2}\Big(\frac{(q_i - q_j)v_j}{\|q_j - q_i\|}\Big)^2 + \frac{1}{2c^2}(q_j - q_i)\dot{v}_i\Bigg) \\ + \frac{1}{c^2}\sum_{j=0,j\neq i}^{N}\frac{Gm_j}{\|q_j - q_i\|^3}((q_i - q_l)((2+2\gamma)v_i - (1+2\gamma)v_j))(v_i - v_j) \\ + \frac{3+4\gamma}{2c^2}\sum_{j=0,j\neq i}^{N}\frac{Gm_j\dot{v}_j}{\|q_j - q_i\|}\end{aligned} \tag{3.5}$$

3.3.3. Ekuazio Heliozentrikoak.

Ohikoa da ekuazio diferentzialak koordenatu heliozentrikoen (eguzkiaren zentroarekiko) arabera definitzea. Argitu koordenatu heliozentrikoak ere modu ezberdi-

3.2. Taula: Konstanteak

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

nean eman daitezkeela eta guk koordenatu *heliozentriko kanonikoak* deiturikoak erabili ditugula.

$$\dot{\mathbf{Q}}_\mathbf{i} = \mathbf{V_i} + \sum_{j\neq i,\ j=1}^{N} \frac{\mathbf{V_j}\, m_j}{(m_0 + m_j)},$$

$$\dot{\mathbf{V}}_\mathbf{i} = -G\,\frac{(m_0+m_i)}{\|\mathbf{Q_i}\|^3}\,\mathbf{Q_i} - G\,\frac{(m_0+m_i)}{m_0}\sum_{j\neq i,\ j=1}^{N}\left(\frac{m_j}{\|\mathbf{Q_i}-\mathbf{Q_j}\|^3}(\mathbf{Q_i}-\mathbf{Q_j})\right),\ \ i=1,\ldots,N.$$

Ekuazioen garapen osoa eranskinean eman dugu.

3.3.4. Hasierako balioak.

Eguzki eta planeten hasierako kokapenak (au) eta abiadurak (au/day), Julian data (TDB) 2440400.5 (1969. ekainaren 28) eta ICRFR2 (International Celestial Reference Frame) koordenatu sisteman [12],

Masa zentrua. Integrazio hasieran, gorputzen kokapen eta abiadurak masa zentruaren kokapen eta abiadurak zero izateko aldatzen dira.

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

Masa zentruaren kokapena (Q) eta abiadura (V),

$$Q = \frac{(\sum_{i=0}^{N} Gm_i * q_i)}{M},\ V = \frac{(\sum_{i=0}^{N} Gm_i * v_i)}{M}$$

Eta integrazio hasierako balioak,

$$qnew_i = q_i - R,\ \ vnew_i = v_i - V,\ \ i = 0,\ldots,N.$$

3.3. Taula: Eguzki eta planeten hasierako balioak integrazio jatorriarekiko.

Eguzkia	x, y, z	0.00450250878464055477	0.00076707642709100705	0.00026605791776697764
	v_x, v_y, v_z	−0.00000035174953607552	0.00000517762640983341	0.00000222910217891203
Mercury	x, y, z	0.36176271656028195477	−0.09078197215676599295	−0.08571497256275117236
	v_x, v_y, v_z	0.00336749397200575848	0.02489452055768343341	0.01294630040970409203
Venus	x, y, z	0.61275194083507215477	−0.34836536903362219295	−0.19527828667594382236
	v_x, v_y, v_z	0.01095206842352823448	0.01561768426786768341	0.00633110570297786403
EMB	x, y, z	0.12051741410138465477	−0.92583847476914859295	−0.40154022645315222236
	v_x, v_y, v_z	0.01681126830978379448	0.00174830923073434441	0.00075820289738312913
Mars	x, y, z	−0.11018607714879824523	−1.32759945030298299295	−0.60588914048429142236
	v_x, v_y, v_z	0.01448165305704756448	0.00024246307683646861	−0.00028152072792433877
Jupiter	x, y, z	−5.37970676855393644523	−0.83048132656339789295	−0.22482887442656542236
	v_x, v_y, v_z	0.00109201259423733748	−0.00651811661280738459	−0.00282078276229867897
Saturn	x, y, z	7.89439068290953155477	4.59647805517127300705	1.55869584283189997764
	v_x, v_y, v_z	−0.00321755651650091552	0.00433581034174662541	0.00192864631686015503
Uranus	x, y, z	−18.26540225387235944523	−1.16195541867586999295	−0.25010605772133802236
	v_x, v_y, v_z	0.00022119039101561468	−0.00376247500810884459	−0.00165101502742994997
Neptune	x, y, z	−16.05503578023336944523	−23.94219155985470899295	−9.40015796880239402236
	v_x, v_y, v_z	0.00264276984798005548	−0.00149831255054097759	−0.00067904196080291327
Pluto	x, y, z	−30.48331376718383944523	−0.87240555684104999295	8.91157617249954997764
	v_x, v_y, v_z	0.00032220737349778078	−0.00314357639364532859	−0.00107794975959731297

3.4. Taula: Ilargiaren Lurrarekiko hasierako balioak.

Ilargia	x, y, z	−0.00080817735147818490	−0.00199462998549701300	−0.00108726268307068900
	v_x, v_y, v_z	0.00060108481561422370	−0.00016744546915764980	−0.00008556214140094871

3.3.5. Kodeak.

Mathematicako NBodyProblem.m paketean honako funtzioak garatu ditugu.

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

-

C-lengoaiako inplementazioa:

1. Hamiltondarra: HamNBody().
2. EDA: OdeNbody().
3. Jakobiarra: JacNBody().

3.5. Taula: Planeten masa parametroak.

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

3.4. Laburpena.

4. Kapitulua

Koma Higikorreko Aritmetika.

4.1. Sarrera.

Gaur-egungo konputagailuetan, *IEEE-754* estandarraren araberako koma-higikorreko aritmetika erabiltzen da. Koma-higikorreko aritmetikaren gaiak ez dira zenbaki errealak, koma-higikorreko zenbakiak baizik. Zenbaki errealak bit kopuru finituen bidez adierazten dira eta adierazpen finitu honek, biribiltze errorea eragiten du. Zenbakizko integrazio luzeetan biribiltze errorearen garapenak garrantzia handia du eta errore honen gaineko ahalegin berezia beharrezkoa da.

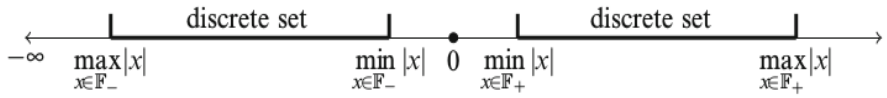
Atal honetan, koma-higikorreko oinarria eta *IEEE-754* estandarraren biribiltze errorea azalduko dugu. Ondoren, konputazioetan gertatu ohi diren biribiltze errore nagusiak azaldu eta horiek gutxitzeko teknikak emango ditugu.

4.2. Adierazpena.

Definizioa. Koma-higikorreko adierazpen zehatza duen zenbaki errealei koma-higikorreko zenbakiak deritzogu. Koma-higikorreko zenbakien multzoa $\mathbb{F}$ izendatuko dugu eta $\phi : \mathbb{F} \to W$ koma-higikorreko adierazpen funtzioa.

$$\mathbb{F} \subset \mathbb{R},$$
$$\mathbb{F} = \{x \in \mathbb{R} \mid \phi(x) \in W\}. \tag{4.1}$$

$\mathbb{F}$ zenbaki multzoa finitua da. Bai zenbaki positiboentzat,bai negatiboentzat, adieraz daitekeen zenbaki handienaren eta txikienaren arteko balio bakanez osatuta dago. Multzoaren kanpoaldean zenbaki hauek guztiak ditugu: batetik overflow tartean $(-\infty, \max_{x\in\mathbb{F}_-} |x|)$ eta $(\max_{x\in\mathbb{F}_+} |x|, \infty+)$ daudenak; bestetik underflow tartean $(\min_{x\in\mathbb{F}_-} |x|, 0)$ eta $(0, \min_{x\in\mathbb{F}_+} |x|)$ daudenak.



4.1. Irudia: Floating-point number line.

IEEE-785 estandarraren arabera, n-biteko koma-higikorrezko adierazpenak bi zati ditu,

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

2. Esponentea (E), $(n-m)$ bitez adierazitako zenbaki osoa. Zeinuarentzat ez da bit zehatz bat erabiltzen, baizik *bias izeneko balio bat gehituz adierazten dira zenbaki positiboak eta negatiboak.*

Eta beraz, oinarri bitarrean koma-higikorreko zenbaki hauek adierazten dira,

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

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

Bi koma-higikor zenbaki jarraituen arteko diferentzia,

$$\triangle x = x_1 - x_2 = 2^{E-m+1}$$

Machine epsilon (ϵ_M), $E = 0$ deneko koma-higikorrezko bi zenbakien arteko diferentzia bezala definitzen da,

$$\epsilon_M = 2^{-m+1}$$

Azkenik, roundoff definituko dugu zenbaki erreal batek koma-higikorrean adierazpenean izan dezakeen errore erlatibo maximoa bezala,

$$u = \frac{\epsilon_M}{2} = 2^{-m}$$

IEEE-785 estandarrean honako formatu bitarrak definitzen dira:

4.1. Taula

Mota	Tamaina	m	e	Tartea	$u = 2^{-m}$
Single	32 bit	24	8	$10^{\pm 38}$	6×10^{-8}
Double	64 bit	53	11	$10^{\pm 308}$	1×10^{-16}
Quadruple	128 bit	113	15	$10^{\pm 11356}$	1×10^{-35}

Gaur egungo konputagailuetan, Single eta Double koma-higikorreko aritmetika Hardwarez inplementatuta dago eta oso azkarra da. Single koma-higikorreko aritmetikak Double baino azkarragoa da: batetik garraiatu behar den bit kopuru erdia da eta bestetik, Inteleko txipen SSE modulo bereziei esker eragiketa aritmetikoak azkarragoak dira.

2008. urtean, IEEE-785 estandarrak 128 biteko koma-higikorreko aritmetika onartu zuen. Quadruple aritmetika softwarez inplementatuta dago eta horregatik exekuzio motela da, gutxi gorabehera Double aritmetika baino 10 aldiz garestiago. Horretarako hainbat liburutegi daude, guk GCC libquadmath liburutegia aukeratu dugu gure garapenetarako.

Bestalde, badaude doitasun arbitrariotan lan egiteko beste lan-ingurune (Problem Solving Enviroment [18]) batzuk ere. Doitasun altuetako kalkulu hauekin, soluzio zehatzak lortzen dira eta horrela, algoritmoen testak egiteko bidea ematen dute. Matlab eta Mathematica bezalako softwareetan doitasun handian lan egiteko aukera ematen dute eta beraz, algoritmo berri baten garapenean oso tresna erabilgarriak izan daitezke.

4.3. Biribiltze errorea.

Zenbakizko integrazioen errorea, trunkatze eta biribiltze erroreez osatuta dago. Urrats tamaina adina txikia aukeratuz, trunkatze errorea biribiltze errorea baino txikiago izango da eta beraz, zenbakizko integrazioaren errorean biribiltze errorearen eragina nagusituko da. Epe luzeko integrazioetan errorea gutxitzea funtsezkoa izango da.

Bi motako biribiltze errorea dugu, bata adierazpen errorea eta beste eragiketa (aritmetika) errorea.

Adierazpen errorea.

$x \in \mathbb{R}$ *eta* $fl : \mathbb{R} \to \mathbb{F}$ *koma-higikorreko gertuen dagoen zenbakia esleitzen duen funtzioa emanik,*

errore absolutua,

$$\triangle x = fl(x) - x = \tilde{x} - x,$$

eta errore erlatiboa,

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

Aurreko bi definizioen ondorioz honako formula erabilgarria dugu,

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

IEEE-785 *estandarrak,* $|\delta x| < u$ *dela bermatzen du.*

Eragiketen errorea.

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

$$* \in \{+, -, x, /\}.$$

Modu berean, hauek dira koma-higikorrezko zenbakien arteko funtsezko eragiketak $\circledast : \mathbb{F}^2 \to \mathbb{F}$*,*

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

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

$$\triangle z = \tilde{z} - z = (\tilde{x} \circledast \tilde{y}) - (\tilde{x} * \tilde{y}),$$

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

Modu berean honako erlazio dugu,

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

IEEE-785 estandarrak $|\delta z| < u$ *dela bermatzen du.*

Adibidea. *Errore erlatiboak emaitzaren digitu zuzenak neurtzen du:*

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

Jarraian, zenbakizko konputazioetan gertatu ohi diren biribiltze errore nagusiak azalduko ditugu. Doitasun handiko kalkuluetan, arreta berezia izatea nahitaezkoa izango da.

Biribiltze errorearen propagazioa.

Ohiko konputazioetan, eragiketa aritmetiko kopuru handia egin behar dugu emaitza lortzeko eta orduan biribiltze errorea metatu daiteke. Batzuetan, eragiketa bakoitzeko biribiltze errorea elkar ezereztatzen da baina kasu txarrenean, biribiltze errorea metatu eta magnitude handikoa izan daiteke.

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

$$\bigoplus_{i=1}^{n}(\tilde{x}_i) = (\sum_{i=1}^{n} \tilde{x}_i)(1+\delta),\ |\delta| < u.$$

Zehazki $n = 3$ deneko batura badugu,

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

Ezabapen arazoa.

Algoritmoen kalkuluetan, doitasuna galera azkarra gerta daiteke. Horren adibidea ezabapen arazoa dugu: oso antzekoak diren bi zenbaki arteko kendura egiten dugunean gerta daitekeena; edo magnitude oso ezberdinak diren bi zenbaki gehitzen ditugunean.

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

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

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

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

Biderketa.

Orokorrean, m digituzko bi mantisen biderketaren emaitza zehatza adierazteko, $2m$ digituzko mantisa behar dugu (m digituzko galera) [13]. Biderkagaietako bat 2ren berredura bat bada, orduan biderketa zehatza da baina oro har, m digitu galduko ditugu.

Adibidea. Demagun hiru digitu hamartar errealeko aritmetikarekin ari garela lanean.

Emaitza zehatza,

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

Hiru digitu hamartar errealeko aritmetika,

$$1,343 \times 2,103 \approx 2.824.$$

4.4. Biribiltze errorea gutxitzeko teknikak.

Batura konpensatua.

Era honetako batura errekurtsiboetan,

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

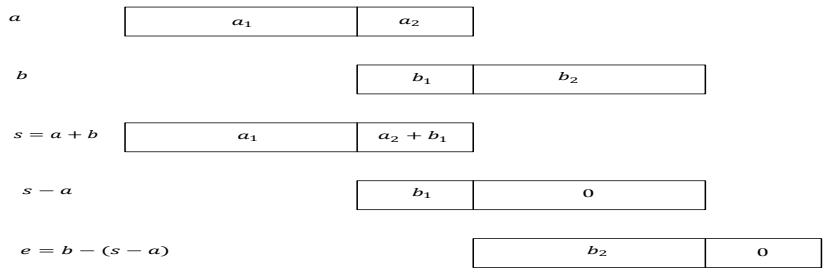
biribiltze errorea gutxitzeko metodoa dugu [17]. Ideia da, bi zenbakien baturan egindako biribiltze errorearen estimazioa lortu eta estimazio hau hurrengo baturan erabiltzea.

Estimazioa nola kalkulatu azaltzeko ikus irudia (Irudia 4.3.). Koma-higikorreko bi zenbaki baditugu, $\tilde{x}, \tilde{y} \in \mathbb{F}$ non $|\tilde{x}| \geq |\tilde{y}|$, eta $\tilde{z} = \tilde{x} \oplus \tilde{y}$,

$$\tilde{e} = -\Big(\big((\tilde{x} \oplus \tilde{y}) \ominus \tilde{x} \big) \ominus \tilde{y} \Big) = (\tilde{x} \ominus \tilde{z}) \oplus \tilde{y}$$

Lortutako errore estimazioa, koma-higikorreko aritmetikan zehazki benetako biribiltze errorea da,

$$\tilde{x} + \tilde{y} = \tilde{z} + \tilde{e}.$$



4.3. Irudia: Biribiltze errorea.

Batura konpensatu algoritmoa biribiltze errore honen estimazioan oinarritzen da. Honako batura $z = \sum_{i=1}^{n} \tilde{x}_i$ *kalkulatzeko , urrats bakoitzaren amaieran errore estimazioa* (e) *kalkulatuko dugu eta hurrengo urratsean, batugaiari gehituko diogu.*

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

ALGORITHM 9: Batura konpensatua.

Zenbakizko integrazioetan, era honetako batura errekurtsiboa kalkulatu behar dugu,

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

non $|\delta_n| < |y_n|$ *izan ohi da. Beraz, batura honetan dugun birbiltze errorea gutxitzeko batura konpensatua erabiliko dugu.*

Biderketaren errore estimazioa.

Konputagailu berrietan , hardware unitate bereziak bermatzen du biribiltze errore bakarra era honetako eragiketean,

$$\tilde{x} \otimes \tilde{y} \oplus \tilde{z} = (\tilde{x} \times \tilde{y} + \tilde{z})(1 + \delta), \ \delta < u.$$

Orduan konputagailuak FMA (fused multiply-add) eragiketak exekutatzen dituela esan ohi da. Hau honela den kasuetan, modu errezan estimatu daiteke biderketa baten biribiltze errorea,

$$\tilde{z} = \tilde{x} \otimes \tilde{y}, \;\; \tilde{e} = (\tilde{x} \otimes \tilde{y}) \ominus \tilde{z}$$

Sterbenz Teorema.

Sterbenz teoremaren arabera [31], bi zenbaki elkarrekiko gertu daudenean, horien arteko kendura zehatza da.

$$x, y \in \mathbb{F}, \;\; \frac{y}{2} \leq x \leq 2y \quad \Rightarrow \quad x - y \in \mathbb{F} \tag{4.2}$$

4.5. Brouwer legea.

Zenbakizko integrazioaren biribiltze errorea hausazkoa dela ziurtatzeko, metodoak Brouweren legea *[4] betetzen duela konprobatu ohi izan da.* Brouwerren legeak dioenez, *t urrats finkoko integrazioaren ($t_n = t_{n-1} + h$) denbora bada, energia errorea $t^{1/2}$ proportzionalki hasiko da. See also Hairer [14][VIII.5]. Figure 7.5. plots the histogram of the Local energy error against the normal distribution $N(\mu, \delta)$.*

4.6. Laburpena.

Koma-higikorreko aritmetikan sakontzeko honako biografia azpimarratuko dugu, Overton [28], Muller [26] eta Corless [8].

5. Kapitulua

Zientzia konputazioa.

Processor speed doubles every 18 months.

Moore's Law (1965)

Number of cores per chip can double every two years.

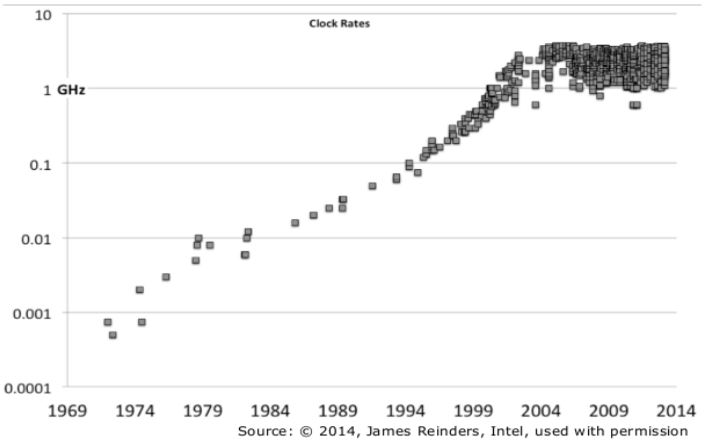
Moore's Law Reinterpreter (2006)

5.1. Sarrera.

Gaur-egungo konputagailuak (super-konputagailu, eramangarri,...) orokorrean paraleloak dira. 1986-2002 urteen artean, txip barruan transistore dentsitatea handitzen zen heinean, prozesadore bakarreko konputagailuen eraginkortasuna hobetuz joan zen. Baina teknologi-garapena muga fisikoetara iritsita, bide honetatik konputagailuen abiadura hobetzea ezinezkoa bilakatu zen (Irudia 5.1.). Horrela, 2005.urtetik aurrera fabrikatzaileek konputagailuen gaitasuna hobetzeko, txipan prozesadore bat baino gehiago erabiltzea erabaki zuten.

Konputagailuen eredu aldaketa honen ondorioz, algoritmo azkarrak garatzeko kodearen paralelizazio gaitasunari heldu behar zaio. Programazio paralelo teknikak inplementatzeko, beharrezko da prozesadore berrien hardware arkitekturak nahiz software ingurune berriak ulertzea. Gaia nahiko konplexua izanik, ikuspegi orokorra eman ondoren, gure inplementazioan erabilitako hardware arkitektura eta software teknika zehatzak azalduko ditugu: memoria-konpartitutako sistemak eta OpenMP programazio eredua.

Bi dira, algoritmo azkarrak diseinatzeko erronkak:



5.1. Irudia: Processor clock rate growth halted around 2005.

1. *Paralizatzeko pisuko lana identifikatzea.*

2. *Memoria eta prozesadorearen arteko datu mugimendua gutxitzea.*

Bestalde, inplementazio berrien garapenean optimizatutako liburutegiak erabiltzea komeni da. Horien artean, LAPACK eta BLAS algebra linealeko liburutegiak erabilgarriak izan zaizkigu. Liburutegi hauen gaineko azalpenak emango ditugu.

5.2. Eraginkortasuna.

Zein azkarrak dira konputagailuak?

Gaur egungo prozesadoreen maiztasun-abiadura hertzetan neurtzen da, hau da, makina ziklo segundoko *kopuruaren arabera. Une honetako prozesadoreak gigahertz mailakoak dira.*

Kilo = mila (10^3*).*

Mega = milioi (10^6*).*

Giga = bilioi (10^9*).*

Tera = trilioi (10^{12}*).*

Peta = 10^{15}.

Exa = 10^{18}.

Koma-higikorreko oinarrizko eragiketa bat egiteko ($\oplus, \ominus, \otimes, \oslash$) ziklo gutxi batzuk behar dira. Honek esan nahi du, 1 *GHz-ko prozesadore batek,* $> 100.000.000$ *koma-higikorreko eragiketa segundoko egiten dituela (*> 100 *megaflops).*

Adibidea. *Demagun* A, B *eta* C $(n \times n)$ *dimentsioko matrizeak ditugula eta* $C = AB$ *matrize arteko biderketa egiteko behar dugun denbora jakin nahi dugula.*

$$c_{ij} = \sum_{i,j=1}^{n} a_{ij} * b_{ji}.$$

- c_{ij} *gai bakoitza kalkulatzeko* n *biderketa eta* $(n-1)$ *batura egin behar ditugu.*
- C *matrizeak* n^2 *osagaia ditu* $\Rightarrow O(n^3)$ *koma-higikorrezko ariketak exekutatu behar dira.*

Adibidez, $n = 100$ *bada* $\Rightarrow$ $n^3 = 10^6$ *eragiketa egin behar ditugu.* 1*GHz prozesadorean exekutatzeko,* $> 10^(- 2)$ *segundo beharko genituzke.*

*Zientzia konputazioaren eraginkortasuna neurtzeko, koma-higikorreko eragiketa kopurua (*flops*) erabili ohi zen. Problema handia denean, datuen mugimendua koma-higikorreko eragiketak baino garestiagoa da, eta beraz eraginkortasuna eragiketa kopuruaren arabera neurtzea okerra izan daiteke. Kodearen exekuzioa azkartzeko derrigorrezkoa da konputagailuan datuen mugimendua minimizatzea.*

Adibidea, $n = 400$ *tamainako matrizeak hartzen baditugu,* $15,6$ MB *memoria behar dugu (suposatuz konputagailuaren CACHE memoria baino handiagoa) eta datuen mugimenduaren eragina nabarituko da exekuzio denboran.*

Timing code.

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

```
S time ./a.out
<kodearen irteera>

real 0m38.856s
user 0m38.789s
sys  0m0.004s
```

Agindu honekin, ./a.out *C programa exekutatuko da eta ondoren, programa exekutatzeko behar izan duen denboraren informazioa pantailaratuko du:* real *hasi eta bukatu arteko denbora (*wall-time*);* user *prozesadoreak gure programa exekutatzen erabili duen denbora (*CPU-time*);* sys *programa exekutatu ahal izateko, sistema eragile lanetan emandako denbora.*

*C lengoaian badaude, denbora neurtzeko funtzioak. Jarraian, exekuzio denbora (*Wall-time*) eta cpu denborak* CPU-time *nola kalkulatu azaldu dugu.*

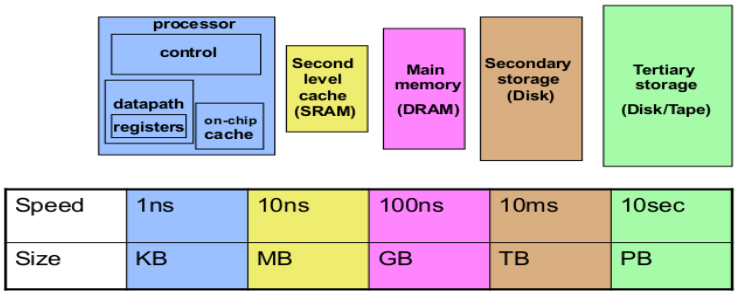
```
#include <time.h>

    time_t  wtime0,wtime1;
    clock_t clock0, clock1;

    wtime0= time(NULL);
    clock0= clock();

    <neurtu nahi den kodea>

    wtime1= time(NULL);
    clock1=clock();

    Wall_time=(wtime1 - wtime0);
    CPU_Time=(clock1 - clock0)/CLOCKS_PER_SEC);
```

Wall-time *deiturikoa izango da algoritmo baten denborak neurtzeko gure irizpidea. Programazio paraleloan, algoritmoen exekuzio denborak egokien neurtzen duen aldagaia da. Dena den, une berean programa bakarra exekutatzea behartuta gaude.*

5.3. Hardwarea.

5.3.1. Memori hierarkia.

Lehenik, konputagailuan dauden memoria mota ezberdinen hierarkia azalduko dugu.



5.2. Irudia: Memoria hierarkia.

CPU-k koma-higikorrezko eragiketak egiten ditu: erregistroetatik datuak irakurri, eragiketak egin eta emaitza erregistroetan idazten ditu. Memoria nagusia eta erregistroen artean, 2 edo 3 mailako Cache memoria dugu: lehen Cache memoria (L1) txikiena eta azkarrena da, eta beste mailak (L2,L3,...), handiagoak eta motelagoak. Memoria nagusian, exekutatzen diren programak eta datuak gordetzen dira ($1-4$ *GB artekoa). Azkenik, disko gogorrean konputagailuko datu (argazki, bideo,...) eta erabilgarri ditugun programa guztiak gordetzen dira.*

Cache memoria lerroka egituratuta dago eta lerro bakoitza 64 *edo* 128 *bytez (*8 *edo* 16 *double zenbaki) osatuta dago. Programa batek datu bat behar duenean, memoria nagusitik lerro tamainako datu taldea irakurriko du eta Cachean idatziko ditu. Komunikazio hau minimizatzeko memorian datuak gordetzeko ordenak badu garrantzia. Beraz, datu-egiturak diseinatzen direnean, kontutan hartu behar da une berean beharko diren datuak memorian gertu gordetzea*

Adibidea. *Badakigunez, C-lengoaian matrizeak lerroka gordetzen dira. Beheko adibidean, matrizearen lehen osagaia* $a(1,1)$ *behar dugunean, memoria nagusitik Cachera osagai honetaz gain jarraiko 16 osagaiak ekarriko dira (*$a(1,1), a(1,2), \ldots, a(1,16)$*). Honela, hurrengo* 15 *batura egiteko behar ditugun datuak Cachean eskuara izango ditugu memoria irakurketa berririk egin gabe.*

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

ALGORITHM 10: Memoria atzipena.

$$a = \begin{pmatrix} 1 & 2 & 3 & \ldots & 1000 \\ 1001 & 1002 & 1003 & \ldots & 2000 \\ 2001 & 2002 & 2003 & \ldots & 2000 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 9001 & 9002 & 9003 & \ldots & 10000 \end{pmatrix}.$$

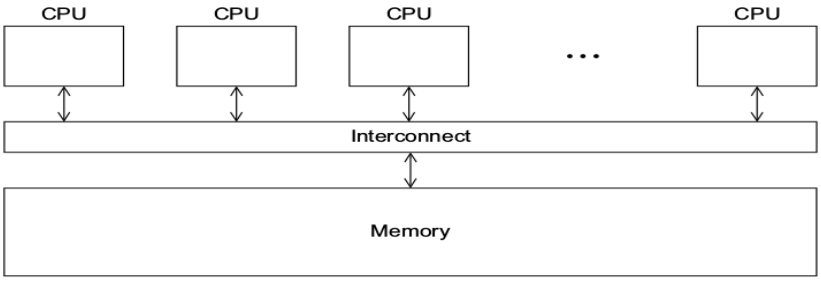
CPUk datu bat behar duenean, memoria hierarkian zehar bilatuko du: lehenik $L1$ *cachean, ondoren* $L2$ *cachean,...eta hauetan ez badago, memoria nagusira*

joko du. Memoria nagusi eta cache memoria arteko irakurketa eta idazketa guzti hauetan, informazio konsistentzia mantentzeko hainbat arau aurrera ematen dira.

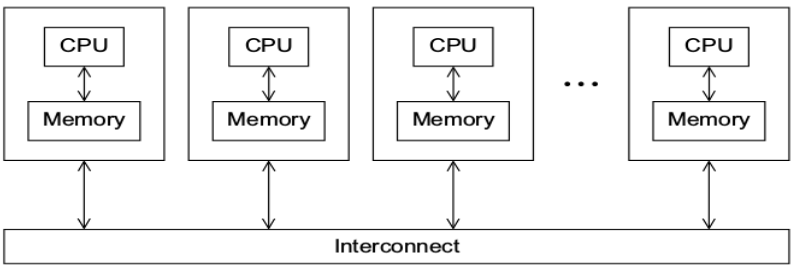
5.3.2. Hardware motak.

MIMD (Multiple instruction, multiple data) sistemak, guztiz independenteak diren prozesadore multzoak osatzen dituzte. Bi dira MIMD sistema nagusiak: memoria konpartitutako eta memoria banatutako sistemak. Memoria konpartitutako sistemetan, prozesadore guztiek memoria osoa konpartitzen dute eta inplizituki konpartitutako datuen atzipenaren bidez komunikatzen dira. Memoria banatutakotako sistemetan aldiz, prozesadore bakoitzak bere memoria pribatua du eta esplizituki bidalitako mezuen bidez komunikatzen dira.

Hirugarren hardware arkitektura ere aipatuko dugu, general purpose GPU computing (Graphical Processor Unit). Jokoen eta animazio industrian, grafiko oso azkarrak beharrak bultzatuta sortutako teknologia da. Oinarrian, imajinak pantailaratzeko prozesagailu asko paraleloan lan egiten dute. Azken hamarkadan, GPU unitate hauek zientzia konputaziora zabaldu dira.

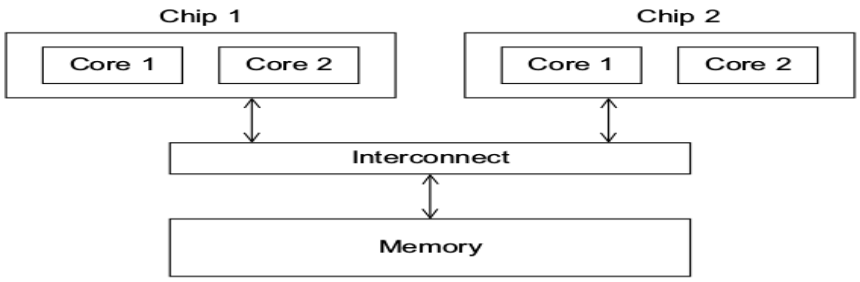


5.3. Irudia: Memoria konpartitutako sistemak.



5.4. Irudia: Memoria banatutako sistemak.

Memoria konpartitutako sistemak. *Multicore bat edo gehiagoz osatutako sistema dugu. Multicore prozesadore bakoitzak txipean CPU bat baino gehiago ditu. Normalean CPU bakoitzak $L1$ bere cache memoria du. Aipatzeko da, era honetako sistemetan prozesadore kopurua ezin dela nahi adina handitu eta mugatua dela (normalean ≤ 32).*



5.5. Irudia: Memoria konpartitutako sistemak (UMA).

5.4. Softwarea.

5.4.1. Software liburutegiak.

Matematika bi software errekurtso nagusienak aipatuko ditugu; BLAS (Basic Linear Algebra Subroutines) eta LAPACK (Linear Algebra Package). Kalitate handiko software orokorrak dira eta hauek erabiltzea abantaila asko ditu:

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

Konputagailu hardware bakoitzerako optimizatutako bertsioak daude. Inplementazioa Fortranen egina dago eta datu-motei dagokionez:

1. *S: float (32 bit).*
2. *D: double (64 bit).*
3. *C: complex.*
4. *Z: complex double.*

BLAS.

BLAS liburutegian, bektore eta matrizeen arteko funtzio estandarrak inplementatuta daude. Hiru mailetan banatuta dago:

1. *BLAS-1: bektore-bektore eragiketak.*

   Adibidez: $y = \alpha * x + y$ *,* $2n$ *flop eta* $3n$ *irakurketa/idazketa.*

   Konputazio intentsitatea: $\frac{2n}{3n} = \frac{2}{3}$.

2. *BLAS-2: matrize-bektore eragiketak.*

   Adibidez: $y = \alpha * A * x + \beta * x$*,* $O(n^2)$ *flop eta* $O(n^2)$ *irakurketa/idazketa.*

   Konputazio intentsitatea: $\approx \frac{2n^2}{n^2} = 2$.

3. *BLAS-3: matrize-matrize eragiketak.*

   Adibidez: $C = \alpha * A * B + \beta * C$*,* $O(n^3)$ *flop eta* $O(n^2)$ *irakurketa/idazketa.*

   Konputazio intentsitatea: $\approx \frac{2n^3}{4n^2} = \frac{n}{2}$.

Azpimarratu, BLAS-1 eta BLAS-2 funtzioen konputazio intentsitatea txikia dela eta beraz, datuen komunikazioa nagusia dela. BLAS-3 aldiz, konputazio intentsitatea handiagoa da eta ezaugarri honi esker, konputagailuaren konputazio gaitasuna ondo aprobetxatu ahal izango da.

Fabrikatzaile bakoitzak optimizatutako BLAS liburutegiak (AMD ACML,Intel MKL) dituzte eta beraz, multi-threaded dira. Beste aukera bat, optimizatutako BLAS instalazioa ATLAS (Automatically Tuned Linear Algebra Software) bidez egitea.

LAPACK

.

Zenbakizko aljebra linealaren liburutegia da.

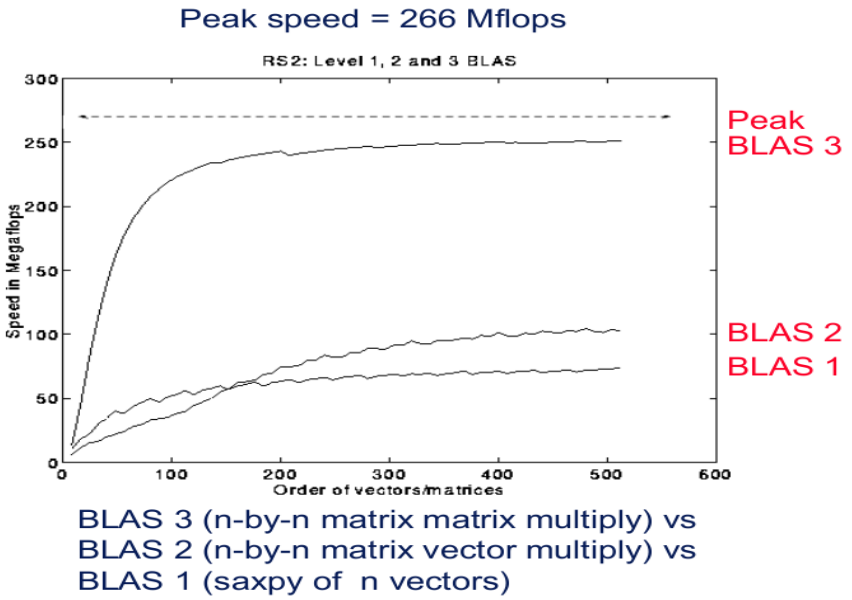
1. *Sistema linealak:* $AX = b$.

2. *Least Square: choose* x *to minimize* $\|Ax - b\|$.

3. *Eigenvalues.*

4. *Balio singularren deskonposaketa (SVD).*

Posible den guztietan, BLAS-3 funtzioetan oinarritzen da.



5.6. Irudia: BLAS speeds.

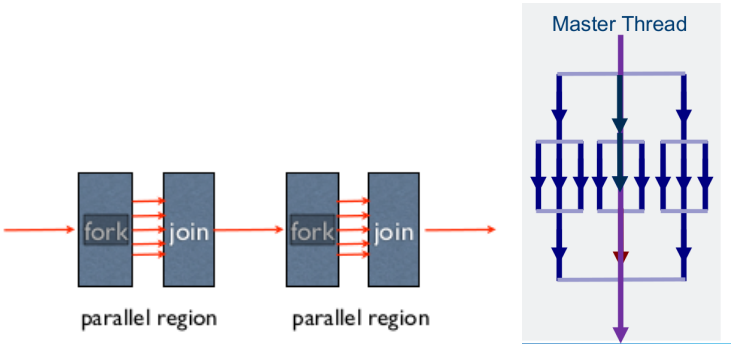
5.4.2. Programazio paraleloa.

C-lengoaia programazio paraleloan erabiltzeko, lengoaiaren bi extensio dira nagusienak: bata memori-banatutako sistemetarako diseinatuta MPI (Message-Passing Inteface) eta bestea, memoria-konpartitutako sistemetarako diseinutakoa OpenMP (Open Specifications for MultiProcessing). MPI datu moten definizio, funtzio eta makroen liburutegia da. OpenMP liburutegia bat eta C konpiladorearen aldaketa batzuk. OpenMP erabili dugu gure inplementaziorako eta jarraian honi buruzko idei nagusienak emango ditugu.

OpenMP. *Memoria konpartitutako programazio paraleloaren estandarra dugu. Programazioan paralelizazio kontrola, "fork-join"modeloa jarraituz egiten da.*

1. *OpenMP programen hasieran prozesu bakarra dago, hari (thread) nagusia.*
2. *FORK: hari nagusiak hari talde paraleloa sortzen du.*
3. *JOIN: hariak kode paraleloa bukatzen dutenean, behin sinkronizatuta amaitzen dute eta hari nagusiak bakarrik jarraitzen du.*

Aldagai batean (threadcount) paralelizazioan zenbat hari erabili adierazten da eta ohikoa izaten da hari bat prozesadore bakoitzeko sortzea. Konpilazio direktiben bidez, paralelizazioa nola exekutatu behar den zehazten zaio.



(a) Fork-Join. (b) Fork-Join.

5.7. Irudia: OpenMp programazio modeloa.

Adibidea .

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

OpenMP Version 4.5.
gcc -v (gcc version 4.8.4 (Ubuntu 4.8.4-2ubuntu1 14.04.3))
A number of compilers from various vendors or open source communities implement the OpenMP API:

1. *From GCC 4.7.0, OpenMP 3.1 is fully supported.*

2. *From GCC 6.1, OpenMP 4.5 is fully supported in C and C++.*

5.4.3. Konpiladorea.

Sarrera.

Erabiliko dugun konpiladorea,

1. gcc - GNU *open source compiler.*

   ```
   $ gcc -v
   $ gcc version 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04.3)
   ```

2. *Several comercial compilers also are avalaible.*

C11 (formerly C1X) is an informal name for ISO/IEC 9899:2011,[1] the current standard for the C programming language.It replaces the previous C standard, informally known as C99. This new version mainly standardizes features that have already been supported by common contemporary compilers, and includes a detailed memory model to better support multiple threads of execution.

gcc requires you specify -std=c99 or -std=c11

Optimizations *(-Olevel).*

Optimizes the code for execution speed according to the level specified by level , which can be 1, 2, or 3. If no level is specified, as in –O , then 1 is the default. Larger numbers indicate higher levels of optimization.

-O2 (default) Optimize for code speed. This is the generally recommended optimization level. -O3 Enable -O2 optimizations and in addition, enable more aggressive optimizations such as loop and memory access transformation, and prefetching.

Every compiler offers a collection of standard optimization options (-O0,-O1,. . .). However, all compilers refrain from most optimizations at level -O0, which is hence the correct choice for analyzing the code with a debugger. At higher levels, optimizing compilers mix up source lines, detect and eliminate "redundant" variables, rearrange arithmetic expressions, etc.,

Konpilazioa.

Oinarrizko erabilpena.

1. *Compiles and links and creates an executable adibidea.exe.*

```
$ gcc adibidea.c -o adibidea.exe
```

```
$ ./adibidea.exe
```

2. *Compile and link steps.*

```
$ gcc adibidea.c  # creates adibidea.o
$ gcc adibidea.o -o adibidea.exe
```

```
gcc -O2 -Wall -std=c99 -fno-common adibidea.c
```

Makefile.

A common way of automating software builds and other complex tasks with dependencies.

A Makefile is itself a program in a special language.

Adibidea. *Demangun programa bat hiru fitxategieten banatuta dugula,*

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

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

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

Programa exekutagarria lortzeko makefile fitxategia,

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

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

Typical element in the simple Makefile:

```
target: dependencies
>TAB>  command(s) to make the target
```

Typing "make target"means:

- *Make sure all dependencies are update (those that are also targets).*
- *If target older than any dependency, recreate it using specified commands.*
- *The rules are applied recursively.*

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

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

%.o: %.c
	${CC} ${FLAGS} -c $<

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

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

5.5. Kode Optimizazioak.

Applications have two general challenges:

1. *Numerical Method.*

   Performance required computation the shortest account of time.

2. *. Computer Program.*

   Express the algorithm as fast computer problem, you have realized computer hardware eficiently.

Optimization areas are:

1. *Vectorization.*
2. *Paralelization.*
3. *Memory trafic control.*

Optimization areas:

1. *scalar optimization (compiler friendly practices).*

2. *vectorization (must use 16 or 8 wide vectors).*

3. *multi-threading (must scale to* $100+$ *threads).*

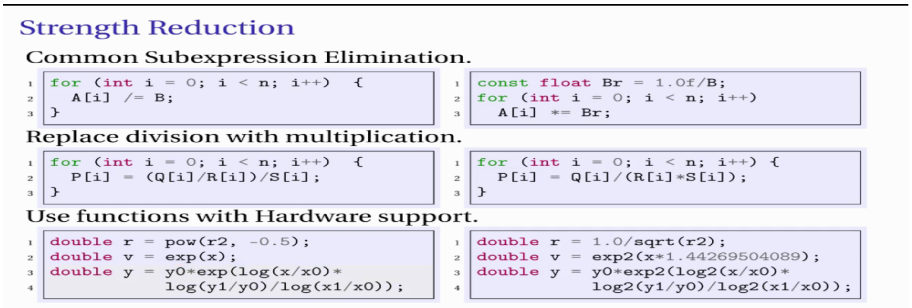
4. *memory access (streaming acces).*

5. *communication (offload, MPI traffic control).*

Scalar Tuning and General Optimization.

Optimization of scalar arithmetics.

One of the most important scalar optimizazion techincs is ***strength reduction*** *: replace expensive operation for less expensive operations.*



5.8. Irudia: Optimization.

Precision control.

1. *Precision Control for transcendental functions.*

2. *Floating-point semantics.*

3. *Consistency of precision: constants and constants.*

   Using incorrect function names in the single precsion is a common mistake

Optimization of vectorization.

1. *Preferably unid-stride access to data. Very important step in the optimization.*

   Artikulua "Auto-Vectorization with the Intel Compilers".

   Most CPU architectures today include Single Instruction Multiple Data (SIMD) parallelism in the form of a vector instruction set. Serial codes (i.e., running with a single thread), as well as instruction-parallel cal- culations (running with several threads) can take advantage of SIMD instructions and significantly increase the performance of some computations. Each CPU core performs SIMD operations on several numbers (in- tegers, single or double precision floating-point numbers) simultaneously, when these variables are loaded into the processor's vector registers, and a vector instruction is applied to them. SIMD instructions include common arithmetic operations (addition, subtraction, multiplication and division), as well as comparisons, reduction and bit-masked operations (see, e.g., the list of SSE 2 intrinsics). Libraries such as the Intel Math Library provide SIMD implementations of common transcendental functions, and other libraries provide vectorized higher-level operations for linear algebra, signal analysis, statistics, etc.

2. *Data Alignment and Padding.*

   An important consideration for efficient vectorization is data alignment.

Multi-threading.

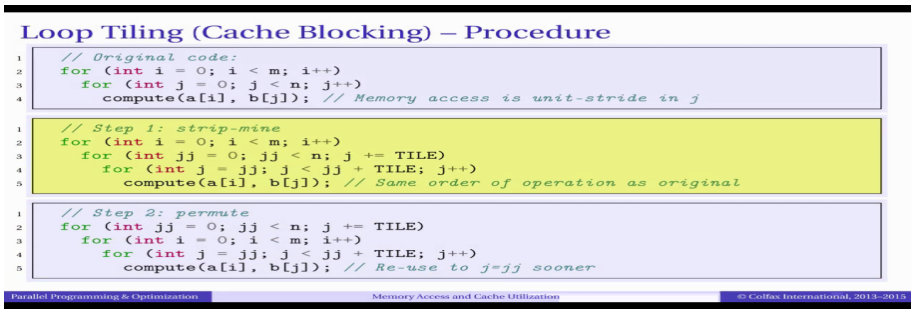
Do you have enough parallelism in your code?
Expanding iteration space, if it is no enough iterations in parallel loop.
Three layers of parallelism: MPI processes, OpenMP threads, vectorization.

Memory access.

Memory access and Cache utilization.
loop tiling technics

5.6. Laburpena.

Best practices for code vectorization and parallelization, and additional tips and tricks.
Algoritmo bat inplementatzen dugunean kontutan hartu beharrekoa:



5.9. Irudia: Optimization.

1. *Lerro edo zutabe araberako iterazioak exekuzio denboran eragin handia du.*

2. *Kodea garbia eta ulergarria mantendu behar da.*

3. *Badaude kodearen exekuzio denboraren analisia egiteko tresnak (adibidez gprof). Algoritmoaren funtzio bakoitzaren exekuzio denborari buruzko informazio erabilgarria lortuko dugu. Zenbait gauza modu sinplean azkartu daitezke baina zenbait beste gauza azkartzeko esfuertzu handia eskatu dezake.*

4. *Optimizatutako beste hainbat kode erabiltzea komenigarria da. LAPACK aljebra lineal paketea Fortran eta C-lengoaitetatik deitu daiteke. Eraberean, LAPACKek BLAS subrutinak erabiltzen ditu. Subrutinak hauek matrizen arteko biderketak, "inner product", ... BLAS konputagailu arkitektura ezberdinetarako optimizatutako bertsioak daude.*

6. Kapitulua

Eguzki-sistemaren integraziorako metodoak (review).

6.1. Sarrera.

6.2. Efemerideak.

Hiru dira planetak efemerideak,

1. *Jet Propulsion Laboratory, DE (Development Ephemerides).*

   Integrazio tartea: $1550 - 2650$.

   Zenbkizko metodoa: "DIVA"(Krogh,1997).A variable order Adams method.

   Doitasuna: "QIVA"a quaruple precision of "DIVA": the equations of motion , the newtonian part is computed in quadruple precision; all of the rest are computed in double precision.

2. *Paris Observatory, INPOP (Intégrateur Númerique Planétaire de l'Observatoire de Paris).*

   Integrazio tartea: .

   Zenbakizko metodoa: The integrator is an Adams-Cowell method with fixed step-size.

   Doitasuna: the programming is done in C language, thus allowing to use the extended precision (80 bits). Integrating in quadruple precision would of course reduce the round off error in a very large amount, but the CPU time is about 15 time larger than for double precision arithmetic (or extended arithmetic) on our machine (Itanium II with Intel C++ compiler). Nevertheless, it was possible to obtain an additional order of magnitude

im- provement by using a single addition in simulated quadruple precision in the corrector step with a very small over- head.

Hardware: Intel Itanium II processors.

3. *St. Petersburg, EPM (Ephemerides Planets-Moon).*

   Integrazio tartea: .

   Zenbakizko metodoa: Everhart. Implicit RK method (Gauss-Radau). (An efficient integrator that uses Gauss-Radau Spacings)

   Doitasuna: double precision. The change of ERA system integrator (19 decimal digits instead of 15 ones) with the aim to reduce the round-off error. (Extended precision).

Influence of the methods of constructing ephemerides... *It is obvious that such ephemerides in themselves must have 128 bits, that is, comprise the coefficients calculated with quadruple precision.*

6.3. Eguzki-sistemaren integrazio luzeak.

A.Morbidellik [25] eguzki-sistemaren zenbakizko integrazioen algoritmoen garapenaren azterketan, gaia hauek sailkatzen ditu:

1. *The classical period.*

   90. *hamarkada hasiera arte,*

2. *The symplectic period.*

3. *The statistic period.*

4. *The planetary accretion period.*

Wisdomek eta Holmanenek bere lanean [32, 1991], eguzki-sistemaren epe luzeko simulazioetarako integratzaile sinplektikoen erabilerak arrakasta izan zuen. N-planeta eta masa nagusiko gorputza bat dugula kontsideratuta, problemaren Hamiltondarra bitan banatu zuten: Hamiltondar Kleperiarra eta interakzioen Hamiltondarra. Metodo honetan, Hamiltondar bakoitzaren soluzioa tartekatuz, problema osoaren ebazpena kalkulatuko da.

Wisdom eta Holmanen inplementazioak ez ditu kolisio gertuko egoerak onartzen. Arazo hau gainditzeko, urteetan zehar algoritmo honen hainbat aldaera proposatu dira: Levinson eta Duncan-ek [23, 1994] SWIFT *softwarea garatu zuten; Duncan, Levinson eta Lee-k [9, 1998]* SYMBA *softwarea garatu zuten;*

Chambers-ek [7] MERCURY *softwarea garatu zuen. Berriki, Hernandez eta Bertschinger-ek [16,* 2015*] garapen berri bat proposatu dute.*

Koordenatu sistema aukera ezberdinak erabili dira Hamiltondarraren banaketa lortzeko. Jacobi koordenatuak eta koordenatu Heliozentrikoak erabili ohi dira bakoitzak bere abantaila eta desbaintailekin.

Problema integratzeko oinarrizko metodoa leapfrog *metodoa dugu. Metodo hau 2 ordeneko da. Orden altuagoko splitting eskemak : McLachlan [24,* 1995*], Laskar eta Robutel [21, 2001], Blanes [3].*

6.4. Laburpena.

7. Kapitulua

IRK: Puntu-Finkoa.

7.1. Sarrera.

Puntu-finkoan oinarritutako non-stiff *sistema Hamiltondarren zenbakizko integraziotarako IRK metodoaren inplementazioa proposatuko dugu. Zientzian, konputazioak koma-higikorreko aritmetikarekin egien direnez, biribiltze erroreak soluzioen doitasuna mugatzen du. Hortaz, inplementazioa epe luzeko doitasun altuko zenbakizko integrazioetarako aplikagarria izateko, biribiltze errorearen eraginak txikia izan behar du. Era berean, integrazioan zehar biribiltze errorearen estimazioa ezagutzea interesgarria da.*

Integrazioaren exekuzio denborak onargarriak izan daitezen , honako aurrebaldintza finkatu dugu: ekuazio diferentzialaren eskuin aldeko funtzioaren sarrera eta irteera argumentuak makina zenbakiak izatea, hau da, konputagailuan hardware bidezko exekuzioa (azkarra) duen koma-higikorreko aritmetika erabiltzea. Gaur-egun, zientzia-konputazioan 64*-biteko koma-higikorreko aritmetikarekin (*double *datu-mota) lan egiten da eta beraz, praktikan erabiltzaileak ekuazio diferentziala datu-mota honetan zehaztuko duela suposatuko dugu.*

Lehenengo Hairer-en inplementazioa [15] aztertuko dugu. Ondoren, IRK inplementazioa hobetzeko gure proposamenak azalduko ditugu. Azkenik, zenbakizko gure inplementazioaren emaitzak erakutsiko ditugu.

7.2. Hairer-en inplementazioa.

Gure abiapuntua, Haier-ek [15] proposatutako inplementazio hartu dugu. Lan honetan, IRK metodo sinplektikoaren puntu-finkoaren inplementazio estandarrean biribiltze errorearen garapen okerraz jabetu ziren eta gainera, metodo sinplektiko esplizituetan agertzen ez zena. Hauen ustez, bi ziren errore honen jatorriak:

1. *Integrazioan* $a_{ij}, b_i \in \mathbb{R}$ *koefiziente zehatzak erabili ordez, biribildutako* $\tilde{a}_{ij}, \tilde{b}_i \in \mathbb{F}$ *erabiltzeak, aplikatutako IRK metodoa zehazki sinpletikoa ez izatea eragiten du.*

2. *Puntu-finkoaren geratze irizpide estandarra dela eta, urrats bakoitzean errore sistematikoa gertatzen da.*

$$\triangle^{[k]} = \max_{i=1,\ldots,s} \|Y_i^{[k]} - Y_i^{[k-1]}\|_\infty \leq \delta$$

Arrazoi hauek aztertu ondoren, honako konponbideak proposatu zituzten:

1. *Doitasun handiagoko koefizienteak erabili, hauetako bakoitza bi koma-higikorreko koefizienteen batura kontsideratuz,*

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

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

Zehazki era honetan kalkulatu daitezke,

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

2. *Iterazioak geratu, definitutako norma txikitzeari uzten dionean.*

$$\triangle^{[k]} = 0 \ \ or \ \triangle^{[k]} \geqslant \triangle^{[k-1]}.$$

Hairer-ek bere Fortran *inplementazioa eskuragarri du (Fortran kodea). Jarraian Hairer-en algoritmoa laburtu dugu (alg.11). Aipatutako hobekuntzaz gain, batura konpensatua erabiltzen duela azpimarratu nahi dugu (notazioa sinplifikatze aldera* $Y_{n,i}$ *gaiaren ordez,* Y_i *adierazpena erabiliko dugu).*

7.3. Gure inplementazioa.

IRK metodoaren puntu-finkoaren inplementazioan lau proposamen berri egin ditugu. Lehen bi proposamenak Hairer-ek bere lanean proposatutako konponbideen hobekuntzak dira. Batetik, IRK-ren birformulazio bat erabiliz, IRK metodoaren koma-higikorreko koefizienteak sinplektizidade baldintza zehazki betetzea lortuko dugu. Bestetik, geratze irizpidean arazo batzuk topatu ditugu eta arazo hauek gainditzen dituen geratze irizpide sendoagoa garatu dugu. Beste bi proposamenak dagokionez, bata batura-konpensatuari erlazionatuta dago eta bestea biribiltze errorea monitorizatzeko proposamena da.

Bestalde kapitulu honen bukaeran, interpolazio bidezko atalen hasieraketa azalduko dugu. Bukatzeko, gure algoritmoa azalduko dugu.

$e = 0$;
for $n \leftarrow 0$ **to** $(endstep - 1)$ **do**
  $k = 0$;
  $Y_i^{[0]} = y_n + h\, c_i\, f(y_n)$;
  while $(\triangle^{[k]} \,!= 0 \;\; and \; \triangle^{[k]} < \triangle^{[k-1]})$ **do**
    $k = k + 1$;
    $F_i^{[k]} = f(Y_i^{[k-1]})$;
    $Y_i^{[k]} = y_n + h\, \big(\sum_{j=1}^{s} a_{ij}^* F_j^{[k]} \big) + h\, \big(\sum_{j=1}^{s} \tilde{a}_{ij} F_j^{[k]} \big)$;
    $\triangle^{[k]} = \max_{i=1,\ldots,s} \|Y_i^{[k]} - Y_i^{[k-1]}\|_\infty$;
  end
  $\delta_n^* = h\, \big(\sum_{i=1}^{s} b_i^* F_i^{[k]} \big)$;
  $\tilde{\delta}_n = h \big(\sum_{i=1}^{s} \tilde{b}_i F_i^{[k]} \big)$;
  $ee = \delta_n^* + e$;
  $yy = y_n + ee$;
  $ee = (y_n - yy) + ee$;
  $ee = \tilde{\delta}_n + ee$;
  $y_{n+1} = y_n + ee$;
  $e = (yy - y_{n+1}) + ee$;
end

ALGORITHM 11: Hairer IRK

7.3.1. Metodoaren birformulazioa (1.proposamena).

IRK metodoa definitzen duten a_{ij}, b_i koefizienteak, biribildutako $\tilde{a}_{ij}, \tilde{b}_i \in \mathbb{F}$ ordezkatzerakoan, sinpletizide baldintza ez da beteko,

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \;\; 1 \leqslant i, j \leqslant s. \tag{7.1}$$

Arazo hau gainditzeko asmoarekin, IRK metodoa era honetan birformulatuko

dugu,

$$Y_{n,i} = y_n + \sum_{j=1}^{s} \mu_{ij} L_{n,j}, \;\; L_{n,i} = hb_i f(Y_{n,i}), \;\; i = 1, \ldots, s, \tag{7.2}$$

$$y_{n+1} = y_n + \sum_{i=1}^{s} L_{n,i}, \tag{7.3}$$

non

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

Eta sinplekzidade baldintza modu honetan berridatziko dugu,

$$\mu_{ij} + \mu_{ji} - 1 = 0, \quad 1 \leq i, j \leq s. \tag{7.4}$$

Formulazio berri honek estandarrarekiko duen abantaila handiena , sinplektizidade baldintzan biderketarik agertzen ez denez, baldintza hau betetzen duten $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak aurkitzeko bidea errazten zaigula da. Honako izango da koefiziente hauek finkatzeko bidea:

1. *μ_{ij} koefizienteak.*

   Batetik s-ataleko Gauss metodoetan, diagonaleko koefizienteek ($\tilde{\mu}_{ii} := 1/2$, $i = 1, \ldots, s$) koma-higikorreko adierazpen zehatza dute.

   Bestetik gainontzeko koefizienteak erabakitzeko, lehengo $\tilde{\mu}_{ij} := fl(\mu_{ij})$, $1 \leq j < i \leq s$ koefizienteak balioekin finkatuko ditugu. Eta bigarrenik, $\tilde{\mu}_{ji} := 1 - \tilde{\mu}_{ij}$, $1 \leq j < i \leq s$ koefizienteei balioa esleituko diegu; $1/2 < |\mu_{ij}| < 2$ denez, eta Sterbenz-en Teoremaren (ikus. 4.2) arabera $1 - \tilde{\mu}_{ij}$ koma-higikorreko adierazpen zehatza du.

   Beraz, hauek ditugu simplektizitate baldintza zehazki betetzen duten koma-higikorreko $\tilde{\mu}_{ij} \in \mathbb{F}$ koefizienteak.

$$\tilde{\mu} = \begin{pmatrix} 1/2 & 1 - fl(\mu_{21}) & \ldots & 1 - fl(\mu_{s1}) \\ fl(\mu_{21}) & 1/2 & \ldots & 1 - fl(\mu_{s2}) \\ \vdots & \ddots & & \vdots \\ fl(\mu_{s1}) & fl(\mu_{s2}) & \ldots & 1/2 \end{pmatrix} \tag{7.5}$$

2. *b_i koefizienteak.*

   Gure inplementazioan, hbi koefizienteak aurre-kalkulatuko ditugu. Batetik, koefiziente hauek simetrikoak direla eta bestetik, $\sum_{i=1}^{s} hb_i = h$ berdintza bete behar dela kontutan hartuz,

$$hb_1 = hb_s := h - \sum_{i=2}^{s-1} hb_i \tag{7.6}$$

3. $y_{n+1} = y_n + \sum_{i=1}^{s} L_{n,i}$ *baturan, batugaien ordena.*

   Batugaien ordenak, batura errekurtsiboen emaitzaren doitasunean eragina du [17]. Zenbaki positiboen batura dugunean, egokiena magnitude txikienetik handienerako batugaien ordena erabiltzea da. Horregatik, honako batura $y_{n+1} = y_n + \sum_{i=1}^{s} hb_i f(Y_{n,i})$ *irizpide honi jarraituz,* b_i *koefizienteen txikienetik handieneko ordenaren arabera kalkulatuko dugu.*

7.3.2. Geratze irizpidea (2.proposamena).

Ekuazio inplizituaren (9.2.) soluzioaren hurbilpena lortzeko puntu-finkoko iterazioa era honetan definituko dugu. Iterazioaren abiapuntua $Y_i^{[0]}$ *finkatu eta* $k = 1, 2, \ldots$ *iterazioetarako* $Y_i^{[k]}$ *hurbilpenak lortu dagokigun geratze irizpidea bete arte.*

for *(k=1,2,...)* **do**

$$L_i^{[k]} = hb_i f(Y_i^{[k-1]});$$

$$Y_i^{[k]} = y_n + \sum_{j=1}^{s} \mu_{ij} L_j^{[k]}, \quad i = 1, \ldots, s;$$

end

ALGORITHM 12: Puntu-finkoko iterazioa.

IRK metodoaren inplementazio estandarrean geratze irizpidea honakoa da,

$$\triangle^{[k]} = (Y_1^{[k]} - Y_1^{[k-1]}, \ldots, Y_s^{[k]} - Y_s^{[k-1]}) \in \mathbb{R}^{sd},$$

$$\|\triangle^{[k]}\| \leq tol \tag{7.7}$$

non $\|.\|$ *aurre-finkatutako bektore norma eta* tol *tolerantzia errorea den . Tolerantzia txikiegia aukeratzen bada, gerta daiteke tolerantzia hori ez lortzea eta infinituki iterazioak exekutatzea. Baina tolerantzia ez bada behar adina txikia aukeratzen, iterazioa puntu-finkora iritsi aurretik geratuko da eta lortutako* $Y_i^{[k]}$ *hurbilpenak biribiltze errorea baino errore handiago izango du.*

Hairer-ek proposatu zuen geratze irizpidea gogoratuko dugu; $\triangle^{[k]} = 0$ *(puntu-finkora iritsi delako) ; edo* $\triangle^{[k]} \geqslant \triangle^{[k-1]}$ *(biribiltze errorea nagusi delako). Orokorrean, geratze irizpide honek ondo funtzionatzen du baina esperimentalki zenbait kasuetan, iterazioak goizegi geratu direla konprobatu dugu. Gure iritziz, honen arrazoia da* $\triangle^{[k]} \geqslant \triangle^{[k-1]}$ *biribiltze errorea nagusia dela adierazten duen arren, badago* $j \in \{1, \ldots, sd\}$ *osagairik,* $|\triangle_j^{[k]}| < |\triangle_j^{[k-1]}|$ *hobetzeko tartea duena.*

Gure proposamena azaldutako arazoari soluzioa emateko, iterazioak jarraitzea honako baldintza betetzen den artean,

$$\exists j \in \{1,\dots,sd\}\ ,\ |\triangle_j^{[1]}| > |\triangle_j^{[2]|} > \cdots > |\triangle_j^{[k]}| > 0. \tag{7.8}$$

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

Koma higikorreko aritmetika atalean (ikus 4..atala) birbiltze errorea gutxitzeko bi teknika aipatu genituen: batura konpensatua eta biderketaren biribiltze errorea jasotzeko modua. Batetik, IRK metodoetan batura konpensatuaren aplikazio estandarra hobetzeko proposamena azalduko dugu eta bestetik, biderketaren biribiltze errorea nola aplikatu urratsaren gehikuntzan.

Batura konpensatua.

Integrazioaren zenbakizko soluzioa $y_{n+1} \approx y(t_{n+1})$ *lortzeko, urrats bakoitzean honako batura dugu,*

$$y_{n+1} = y_n + \phi(y_{n,h}),\ \ n = 0,1,2,\dots.$$

IRK metodoetan, $\phi : \mathbb{R}^{[d+1]} \to \mathbb{R}^d$ *gehikuntza,*

$$\phi(y_{n,h}) = \sum_{i=1}^{s} L_{n,i},$$

non $L_{n,i}$ $(i = 1,\dots,s)$ *inplizituki definitzen den.*

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

$y_{n+1} \in \mathbb{R}^d$, $y_{n+1} = \tilde{y}_n + \tilde{\delta}_n$ *soluzioa zehatza izanik eta* $\tilde{y}_{n+1} \in \mathbb{F}^d$, $\tilde{y}_{n+1} = \tilde{y}_n \oplus \tilde{\delta}_n$ *koma-higikorreko hurbilpena izanik, lortutako errore estimazioa* $\tilde{e}_{n+1}$ *, baturan egindako biribiltze errorea zehatza da,*

$$y_{n+1} = \tilde{y}_{n+1} + \tilde{e}_{n+1}. \tag{7.9}$$

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

$$L_{n,i}^{[k]} = hb_i f(Y_{n,i}^{[k-1]}),\quad Y_{n,i}^{[k]} = \tilde{y}_n \oplus \big(\tilde{e}_n \oplus \sum_{j=1}^{s} \mu_{ij} L_{n,j}^{[k]}\big). \tag{7.10}$$

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

$e_0 = 0$;
for $n \leftarrow 0$ **to** $(endstep - 1)$ **do**
  Hasieratu $Y_{n,i}^{[0]}$, $i = 1, \ldots, s$;
  Puntu-finkoko-iterazioak $(y_n, Y_{n,i})$;
  $\delta_n = \big(\sum_{i=1}^{s} L_{n,i}^{[k]}\big) + e_n$;
  $y_{n+1} = y_n + \delta_n$;
  $e_{n+1} = (y_n - y_{n+1}) + \delta_n$;
end

ALGORITHM 13: Batura konpensatua estandarra.

Biderketaren biribiltze errorea.

Urratsa emateko unean, $Ln, i = hb_i\ f(Y_{n,i})$ *biderketaren biribiltze errorea kalkulatuko dugu eta* e_{n-1} *gaiari gehituko diogu. Biderketaren biribiltze errorea jasotzeko konputagailuaren* Fused Multiply Add *(FMA) eragiketan oinarritutko gara.*

$e_0 = 0$;
for $n \leftarrow 0$ **to** $(endstep - 1)$ **do**
  Hasieratu $Y_{n,i}^{[0]}$, $i = 1, \ldots, s$;
  Puntu-finkoko-iterazioak $(y_n, Y_{n,i})$;
  $EL_i = hb_i\ f(Y_{n,i}^{[k-1]}) - L_{n,i}^{[k]}$, $i = 1, \ldots, s$;
  $e_n = e_n + \sum_{j=1}^{s} EL_j$;
  $\delta_n = \big(\sum_{i=1}^{s} L_{n,i}^{[k]}\big) + e_n$;
  $y_{n+1} = y_n + \delta_n$;
  $e_{n+1} = (y_n - y_{n+1}) + \delta_n$;
end

ALGORITHM 14: Biderketaren biribiltze errorea eta batura konpensatua.

7.3.4. Biribiltze errorearen estimazioa (4.proposamena).

Zenbakizko integrazioaren biribiltze errorearen estimazioa, bigarren zenbakizko integrazio baten soluzioaren diferentzia gisa kalkulatuko dugu. Bigarren inte-

grazio honetan, δ_n gehikuntza mantisa txikiagoko zenbakira biribilduko dugu eta horrela doitasun gutxiagoko soluzioa lortuz.

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

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

ALGORITHM 15: floatR

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

$r < m$ zenbaki osoa finkatuta, bigarren integrazioaren urratsa honela kalkulatuko dugu,

$\hat{e}_0 = 0;$
for $n \leftarrow 0$ **to** $(endstep - 1)$ **do**
  $\ldots;$
  $\hat{\delta}_n = floatR((\sum_{i=1}^{s} L_{n,i}^{[k]}) \oplus \hat{e}_n, r);$
  $\hat{y}_{n+1} = \hat{y}_n + \hat{\delta}_n;$
  $\hat{e}_{n+1} = (\hat{y}_n - \hat{y}_{n+1}) + \hat{\delta}_n;$
end

ALGORITHM 16: Batura konpensatua estandarra.

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

$$estimazioa_n = \|(y_n + e_n) - (\tilde{\tilde{y}}_n + \tilde{\tilde{e}}_n)\|_2 \tag{7.11}$$

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

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

$Y_n^{[0]} = G(Y_{n-1}, h)$;
$\ldots$;
$y_{n+1} = y_n + \delta_n$;
$e_{n+1} = (y_n - y_{n+1}) + \delta_n$;

if $(initwithfirst)$ **then**
$\tilde{\tilde{Y}}_n^{[0]} = Y_n^{[k]} + (\tilde{\tilde{y}}_n - y_n)$;
else
$\tilde{\tilde{Y}}_n^{[0]} = G(\tilde{\tilde{Y}}_{n-1}, h)$;
end
$\ldots$;
$\tilde{\tilde{y}}_{n+1} = \tilde{\tilde{y}}_n + \tilde{\tilde{\delta}}_n$;
$\tilde{\tilde{e}}_{n+1} = (\tilde{\tilde{y}}_n - \tilde{\tilde{y}}_{n+1}) + \tilde{\tilde{\delta}}_n$;

$estimation_{n+1} = \|(y_{n+1} + e_{n+1}) - (\tilde{\tilde{y}}_{n+1} - \tilde{\tilde{e}}_{n+1})\|_2$;

end

ALGORITHM 17: RKG2: errore estimazioa

7.3.5. Atalen hasieraketa.

Ideia da, aurreko urratseko $(t_{n-1} + hc_i, Y_{n-1,i}),\ i = 1, \ldots, s$ *eta* $(t_{n-1} + h, y_n)$, *uneetako balioei dagokien polinomio interpolatzailea erabiliz, urrats berriaren atalen hasieraketa* $(t_n + hc_i, Y_{n,i}^{[0]}),\ i = 1, \ldots, s$ *kalkulatzea.*

$(n-1)$. urratseko informazioa erabiliz,

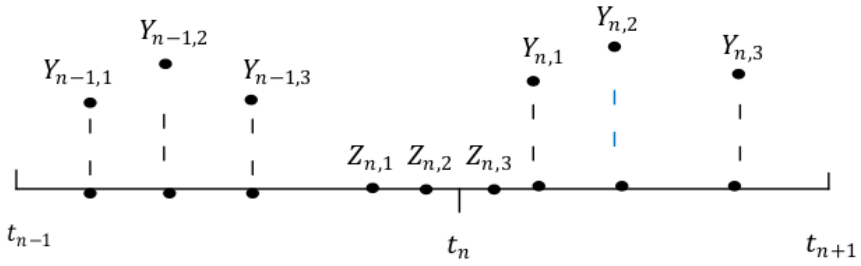
$$\begin{cases} Y_{n-1,i} = y_{n-1} + h \sum_{j=1}^{s} a_{ij} f(Y_{n-1,j}), \\ y_n = y_{n-1} + h \sum_{j=1}^{s} b_j f(Y_{n-1,j}), \end{cases} \Rightarrow \quad Y_{n-1,i} = y_n + h \sum_{j=1}^{s} (a_{ij} - b_j) f(Y_{n-1,j}).$$

Dagokion polinomio interpolatzailea,

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

non $l_i(t)$ *Lagrangiar polinomioa dugun,*

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



7.1. Irudia: Interpolazioa.

Eta beraz,

$$Y_{n,i} \approx Y_{n,i}^{[0]} = P(t_n + hc_i) = y_n + h\sum_{j=1}^{s} \lambda_{ij} f(Y_{n-1,j}) \tag{7.12}$$

Modu honetan s-ataletako IRK metodo bakoitzari dagokion λ_{ij} koefiziente interpolatzaileak lortu daitezke. Polinomio interpolatzailearen bidezko hasieraketa ona izango da, emandako urratsa ez bada oso handia eta problema stiff ez denean. Era berean aipatu nahi genuke, atal askotako metodoetan (adibidez $s = 16$) interpolaziozko koefizienteen kalkuluan ezabapen arazoak, doitasun handian lan egitea behartzen gaituela interpolaziozko hasieraketa ona izateko.

7.3.6. Algoritmoa.

Formulazio berriari dagokion algoritmo orokorra,
Eta puntu-finkoa erabiliz,

7.4. Esperimentuak.

7.4.1. Integrazio motak.

Biribiltze erroreari dagokionez gure inplementazioa optimotik gertu dagoela erakutsi nahi dugu. Esperimentuetan lau integrazio mota egingo ditugu:

1. *Koadruple doitasunezko (128-bit) integrazioa.*

   Zenbakizko integrazio hau soluzio zehatza kontsideratuko dugu eta errore globala kalkulatzeko erreferentziazko soluzioa izango da.

$e = 0$;
for $n \leftarrow 0$ **to** $(endstep - 1)$ **do**
  $k = 0$;
  Hasieratu $Y_{n,i}^{[0]}$, $i = 1, \ldots, s$;
  while *(konbergentzia lortu)* **do**
    $k = k + 1$;
    $L_{n,i}^{[k]} = hb_i f(Y_{n,i}^{[k-1]})$;
    $Y_{n.i}^{[k]} = y_n + \left(e + \sum_{j=1}^{s} \mu_{ij} L_{n,j}^{[k]}\right)$;
  end
  $EL_i = hb_i\ f(Y_{n,i}) - L_{n,i}, \quad i = 1, \ldots, s$;
  $e = e + \sum_{j=1}^{s} EL_j$;
  $\delta_n = \sum_{i=1}^{s} L_{n,i}^{[k]} + e$;
  $y_{n+1} = y_n + \delta_n$;
  $e = (y_n - y_{n+1}) + \delta_n$;
end

ALGORITHM 18: IRK (puntu-finkoa).

2. *Integrazio ideala.*

   Koadruple doitasunezko integrazioa, ekuazio diferentzialaren eskuin aldeko funtzioaren ebaluazioa izan ezik. Ekuazio diferentziala double doitasunean (64-bit) definituko dela aurre baldintza harturik, integrazio ideala kontsideratuko dugu, hau da, hobetu ezin daitekeen integrazioa.

3. *Double doitasuna (batura konpensatu hobetua).*

   64-*biteko koma higikorreko aritmetika eta* batura konpensatua hobetua *erabiltzen duen inplementazioa, hau da, atalen eguneraketan, ($\tilde{y}_n \oplus \tilde{e}_n$) (ekuazioa 7.10) espresioa erabilitako integrazioa.*

4. *Double doitasuna (batura konpensatu estandarra).*

   64-*biteko koma higikorreko aritmetika eta* batura konpensatu estandarra *erabiltzen duen inplementazioa (atalen eguneraketan $\tilde{e}_n$ gaia kontsideratzen ez duen integrazioa).*

7.4.2. Errore azterketa.

Integrazio bakarra egin ordez, ausaz perturbatutako P hasierako balio ezberdinetarako integrazioak exekutatu ditugu eta emaitza guzti hauen batazbestekoan oinarritu gara, biribiltze errorearen azterketa egokia egiteko.

Adibidea.

Ausazko perturbazioak kalkulatzeko funtzio bat definituta,

```
Pert[e0_,k_]:={e0*(k*RandomReal[{-1,1}])};
```

Perturbaziorik gabeko hasierako balioa (u_0, e_0) eta k perturbazio tamaina finkatuta, era honetan kalkulatuko dugu (up_0, ep_0) perturbatutako hasierako balioa,

```
k = 2^35;
aux = Pert[e0, k];
up0 = N[u0] + aux;
aux2=up0-N[u0];
ep0=aux-aux2;
```

Notazioa.

k. integrazioan N urrats eman baditugu, $t_i = t_0 + i * h,\ i = 1, \ldots, N$ uneetarako lortuko dugu zenbakizko soluzioa,

$$(q_i^{[k]}, p_i^{[k]}) \approx (q(t_i)^{[k]}, p(t_i)^{[k]}), \quad i = 1, \ldots, N.$$

Sistema Hamiltondarretan energia kontserbatzen da, energiaren definizioa hau izanik $H(q(t), p(t)) = E(t)$,

$$E_i^{[k]} = H(q_i^{[k]}, p_i^{[k]}) = konst, \quad i = 1, \ldots, N.$$

Neurtzeko faktoreak.

Energia eta kokapen erroreak honako faktoreen bidez neurtuko ditugu.

1. *Energia errore globala.*

$$\triangle E_i^{[k]} = \frac{(E_i^{[k]} - E_0^{[k]})}{E_0^{[k]}}, \quad i = 1, \ldots, N \ eta \ k = 1, \ldots, P.$$

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

$$\bar{MaxE} = \max_{i=1,\ldots,N} |\bar{\triangle E_i}|.$$

2. *Energia errore lokala.*

   P integrazio guztietarako, bi urratsen arteko energia lokalaren batazbestekoa (μ) eta desbiazio estarrada (σ).

$$\blacktriangle E_i^{[k]} = \frac{(E_i^{[k]} - E_{i-1}^{[k]})}{E_0^{[k]}}, \quad i = 1, \ldots, N \; eta \; k = 1, \ldots, P,$$

$$\bar{\mu} = \frac{1}{N \cdot P}\left(\sum_{k=1}^{P} \sum_{i=1}^{N} \blacktriangle E_i^{[k]} \right), \;\; \bar{\sigma} = \sqrt{\frac{1}{N \cdot P}\left(\sum_{k=1}^{P} \sum_{i=1}^{N} (\blacktriangle E_i^{[k]} - \bar{\mu})^2 \right)}$$

3. *Kokapen errore globala.*

   Doitasun laukoitzean lortutako soluzioa, soluzio zehatza kontsideratuko dugu,

$$yexact_i^{[k]} = \hat{y}_i^{[k]} = (\hat{q}_i^{[k]}, \hat{p}_i^{[k]}).$$

   eta k. soluzioari dagokion kokapen errorea,

$$Ge_i^{[k]} = \|\hat{q}_i^{[k]} - q_i^{[k]}\|_2,$$

$$\bar{Ge}_i = (\frac{1}{P} \sum_{k=1}^{P} Ge_i^{[k]}) \;,\; \bar{Max}Ge = \max_{i=1,\ldots,N} (\bar{Ge}_i)$$

4. *Puntu-finkoa lortutako urratsen batazbesteko portzentajea ($\bar{\triangle}0$).*

   $\triangle 0^{[k]}$, k. integrazioan puntu-finkoa lortutako urratsen portzentaia izanik,

$$\bar{\triangle}0 = \frac{1}{p} \sum_{k=1}^{P} \triangle 0^{[k]}.$$

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

   Biribiltze errorearen estimazioaren gogoratuz,

$$Est_i^{[k]} = \|(q_i^{[k]} + eq_i^{[k]}) - (\tilde{\tilde{q}}_i^{[k]} + e\tilde{\tilde{q}}_i^{[k]})\|_2 \tag{7.13}$$

   Errore estimazioaren kalitatea neurtzeko,

$$Q_i^{[k]} = \log_{10}\left(\frac{Est_i^{[k]}}{Ge_i^{[k]}}\right), \tag{7.14}$$

$$\mu\bar{Q}_i = \frac{1}{P} \sum_{k=1}^{p} Q_i^{[k]} \;,\;\; \sigma\bar{Q}_i = \sqrt{\frac{1}{P} \sum_{k=1}^{P} (Q_i^{[k]} - \mu\bar{Q}_i)^2} \tag{7.15}$$

7.4.3. Integrazio parametroak.

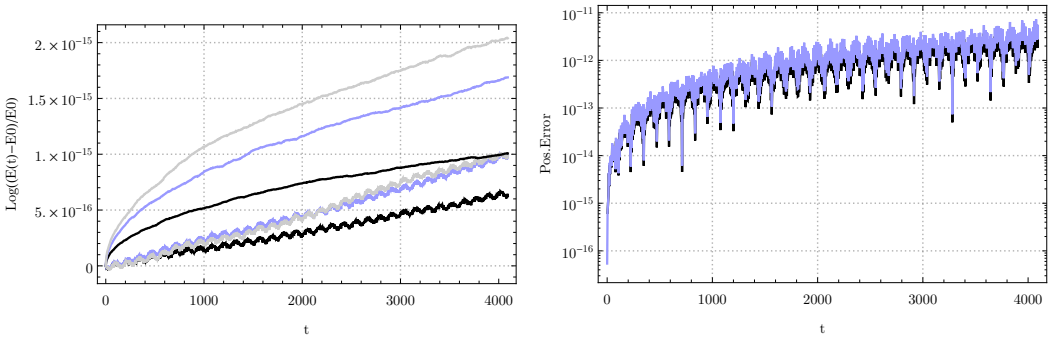
Hiru problemetarako esperimentuetan, integrazio parametroak bateratu ditugu.

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

7.4.4. Pendulu bikoitza ez-kaotikoa.

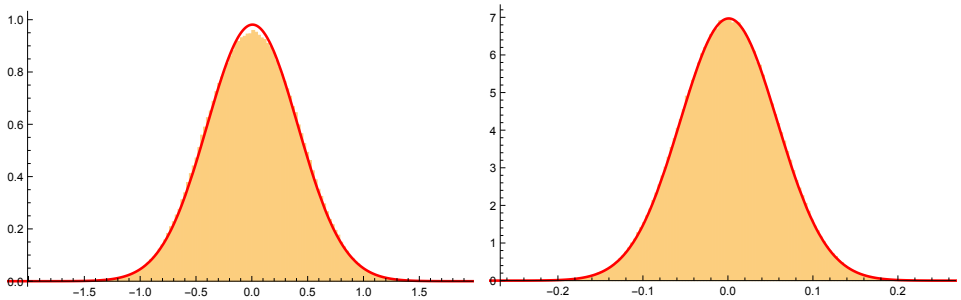
Problema honetan zehazki erabilitako integrazio tartea eta urratsa hauek izan dira,

$$t_0 = 0, \;\; t_{end} = 2^{12},$$
$$h = 2.^{-12},$$
$$sampling = 2^7.$$



(a) Energi errorea. (b) Kokapen errorea.

7.2. Irudia: Pendulu-bikoitza ez-kaotikoa. Ezkerreko grafikoan, energi errorearen batazbestekoa $\bar{\triangle E_i}$ eta desbiazio tipikoa σ_i eboluzioa. Eskubiko grafikoan, kokapen errorearen eboluzioa $\bar{Ge_i}$. Integrazio ideala kolore beltzez, Double integrazioa (hobetua) kolore urdinez eta Double integrazioa (estandarra) gris kolorez.

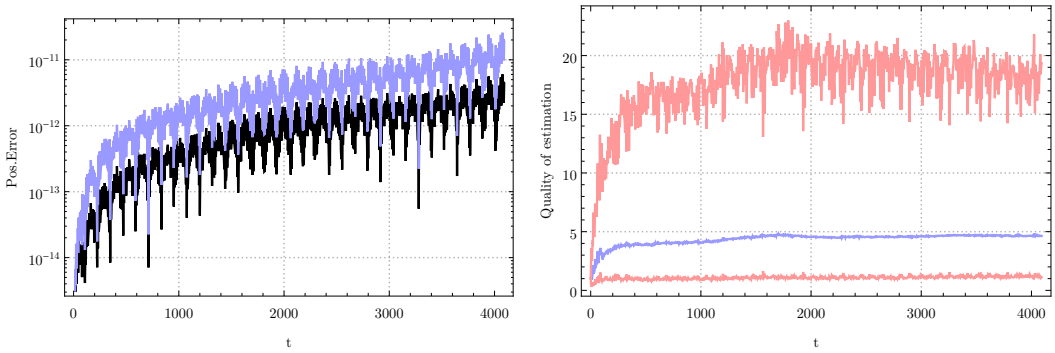


(a) Non-Chaotic: Ideal. (b) Non-Chaotic: rdigits=0.

7.3. Irudia: Histogram of energy errors for Non-Chaotic case (a,b) and for Chaotic case (c,d).

7.1. Taula: Pendulu Bikoitza ez-kaotikoaren laburpena.

Integrazio Mota	$\bar{\triangle}0$ %	$M\bar{a}xE$	$\bar{\mu}$	$\bar{\sigma}$	$M\bar{a}xGe$
Koadruple	93.6	$2e10^{-19}$	$1e10^{-25}$	$2e10^{-20}$	
Ideala	98.3	$7e10^{-16}$	$6e10^{-19}$	$2e10^{-17}$	$5e10^{-12}$
Double	94.8	$1e10^{-15}$	$9e10^{-19}$	$2e10^{-17}$	$7e10^{-12}$



(a) Non Chaotic: estimation

(b) Non Chaotic: quality of estimation

7.4. Irudia:

Pendulu-bikoitza ez-kaotikoa: biribiltze errorearen estimazio. We compare evolution of our estimation error (blue) with evolution of global error (black). Estimation Quality. We show mean, $\mu\bar{Q}_i$ (blue) and standard deviation, $\sigma\bar{Q}_i$ (red) of the quality according our definition of (7.14).

7.4.5. Pendulu bikoitza kaotikoa.

Problema honetan zehazki erabilitako integrazio tartea eta urratsa hauek izan dira,

$$t_0 = 0, \;\; t_{end} = 230,$$
$$h = 2.^{-12},$$
$$sampling = 2^6.$$

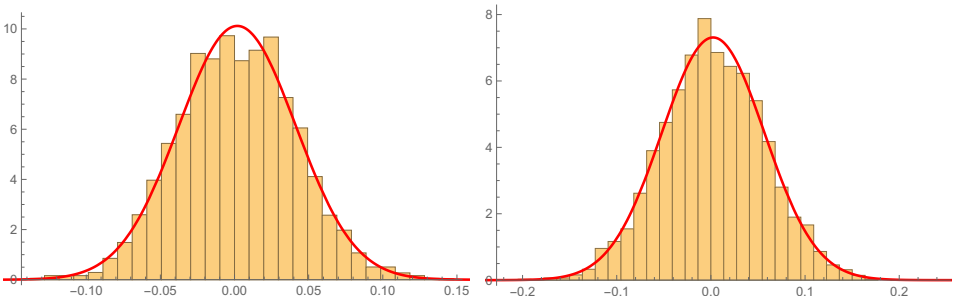
7.2. Taula: Summary of Chaotic case.

Arithmetic	$\bar{\triangle}0$ %	$M\bar{a}xE$	$\bar{\mu}$	$\bar{\sigma}$	$M\bar{a}xGe$
Quadruple prec	93.6	$2e10^{-19}$	$7e10^{-22}$	$1e10^{-20}$	
Ideal Integrator	98.3	$3e10^{-16}$	$1e10^{-18}$	$9e10^{-18}$	0.18
Double prec	94.7	$3e10^{-16}$	$1e10^{-18}$	$1e10^{-17}$	0.23

Energia eta kokapen erroreak.

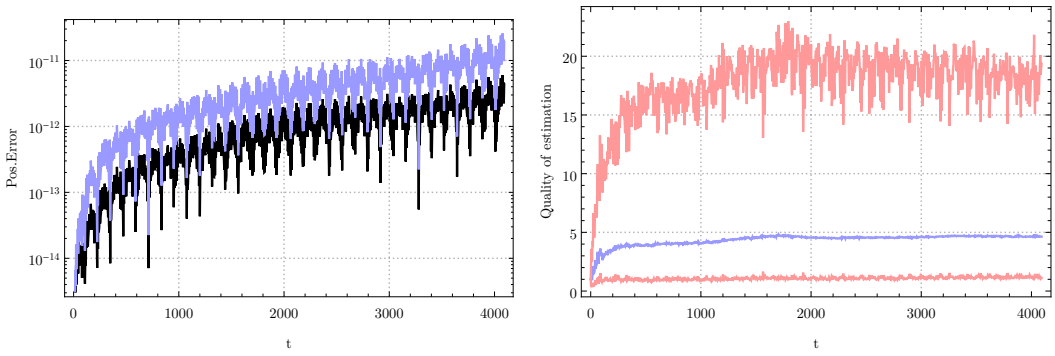
Brower legea.

Errore estimazioa.

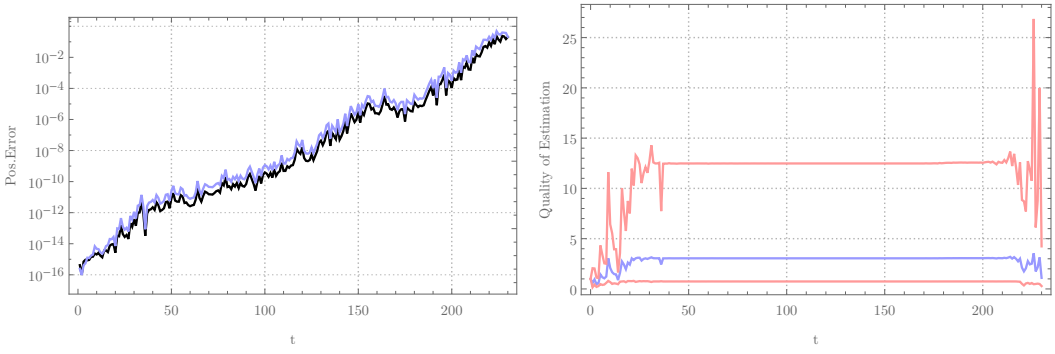


(a) Kaotikoa: Ideal. (b) Kaotikoa: rdigits=0.

7.5. Irudia: Histogram of energy errors for Non-Chaotic case (a,b) and for Chaotic case (c,d).



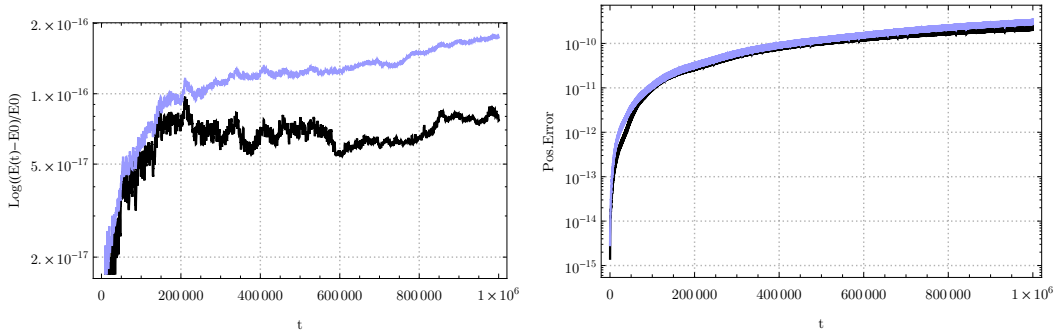
(a) Non Chaotic: estimation (b) Non Chaotic: quality of estimation



(c) Chaotic: estimation (d) Chaotic: quality of estimation

7.6. Irudia: Estimation round-off error. We compare evolution of our estimation error (blue) with evolution of global error (black). Estimation Quality. We show mean (blue) and standard deviation (red) of the quality according our definition of (7.14).

7.4.6. N-Body problema.

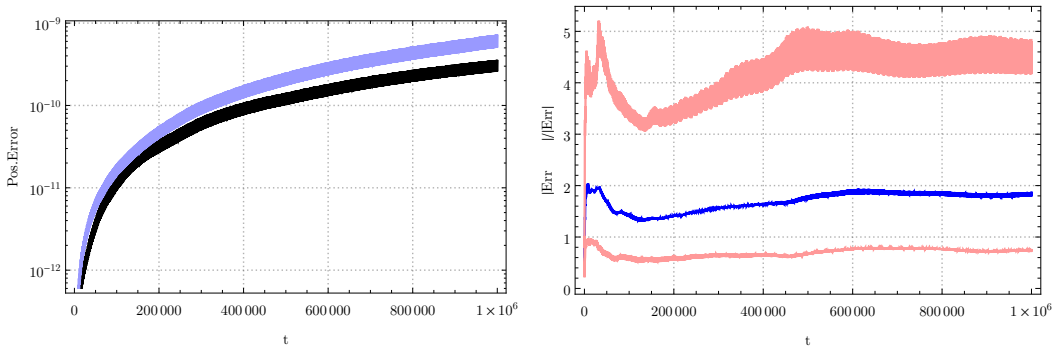


(a) Energy error.

(b) Global error.

7.7. Irudia: N-body: left figure mean energy error evolution $\bar{\triangle E_i}$ and right figure mean Global error evolution $\bar{Ge_i}$ of the 100 integrations for Ideal Integrator (black) and Double prec(blue).

7.5. Laburpena.



(a) Estimation. (b) Quatlity.

7.8. Irudia: Left estimation round-off error, we compare evolution of our estimation error (blue) with evolution of global error (black). Right estimation Quality ,we show mean (blue) and standard deviation (red) of the quality according our definition of (7.14). We use rdigits1=0 and rdigits2=3.

8. Kapitulua

IRK: Newton.

8.1. Sarrera.

Newton metodoa, $G(X) = 0, X \in \mathbf{R}^{n\times m}$ *ekuazio sistema ez-linealen zenbakizko soluzioa aurkitzeko metodoa da. Hasierako soluzioaren estimazioa* $X^{[0]}$ *emanda, benetako Newton metodoa,*

for *(k=1,2,...)* **do**
$M^{[k]} = G'(X^{[k]})$;
$Solve(\triangle X^{[k]} = -G(X^{[k]})/M^{[k]},\ \triangle X^{[k]})$;
$X^{[k]} = X^{[k-1]} + \triangle X^{[k]}$;
end

ALGORITHM 19: Benetako Newton metodoa

Benetako Newton metodoa konputazionalki garestia da , iterazio bakoitzean $M^{[k]} = G'(X^{[k]})$ *jakobiarraren ebaluazioa eta* $(nm \times nm)$ *matrizearen* LU *deskonposaketa kalkulatu behar delako. Horregatik konputazionalki merkeagoa diren Newton metodoaren aldaerak erabili ohi dira.*

Jarraian Newton sinplifikatua definituko dugu, nagusiki erabili den metodoa: jakobiarraren ebaluazio $M = G'(X^{[0]})$ *eta dagokion* LU *deskonposaketa behin bakarrik kalkulatu behar da.*

8.2. IRK-Newton formulazio orokorroa.

Demagun hasierako baliodun problema,

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y_0}, \tag{8.1}$$

non $\mathbf{y} = (q_1, \ldots, q_n, p_1, \ldots, p_n) \in \mathbb{R}^{d=2n}$ *eta* $\mathbf{f}: \mathbb{R}^{\mathbf{d}} \longrightarrow \mathbb{R}^{\mathbf{d}}$.

$M = G'(X^{[0]})$;
for *(k=1,2,...)* **do**
  $Solve(\triangle X^{[k]} = -G(X^{[k]})/M,\ \triangle X^{[k]})$;
  $X^{[k]} = X^{[k-1]} + \triangle X^{[k]}$;
end

ALGORITHM 20: Newton sinplifikatua (L_i).

IRK metodoaren ekuazio sistema ez-lineala,

$$r_i = -Y_i + y_n + h\sum_{j=1}^{s} a_{ij} f(Y_j),\ \ i = 1, \ldots, s.$$

Modu baliokidean notazio hau erabiliko dugu,

$$R(Y) = -Y + y_n + h\ A\ F(Y),$$

–

non,

$$Y^T = (Y_1, \ldots, Y_s),\quad A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1s} \\ a_{21} & a_{22} & \ldots & a_{2s} \\ \vdots & \ddots & & \vdots \\ a_{s1} & a_{s2} & \ldots & a_{ss} \end{bmatrix},\quad F(Y) = \begin{bmatrix} f(Y_1) \\ f(Y_2) \\ \ldots \\ f(Y_s) \end{bmatrix}.$$

Newton sinplifikatuaren metodoa,

$M = R'(Y^{[0]})$;
for *(k=1,2,...)* **do**
  $Solve(\triangle Y^{[k]} = -R(Y^{[k]})/M,\ \triangle Y^{[k]})$;
  $Y^{[k]} = Y^{[k-1]} + \triangle Y^{[k]}$;
end

ALGORITHM 21: Newton sinplifikatua

non

$$R'(Y^{[0]}) = I_s \otimes I_d - h \begin{bmatrix} a_{11} f'(Y_1^{[0]}) & \ldots & a_{1s} f'(Y_1^{[0]}) \\ a_{21} f'(Y_2^{[0]}) & \ldots & a_{2s} f'(Y_2^{[0]}) \\ \ldots & \ldots & \ldots \\ a_{s1} f'(Y_s^{[0]}) & \ldots & a_{ss} f'(Y_s^{[0]}) \end{bmatrix},$$

eta

$$f'(y) = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \dots & \frac{\partial f_1}{\partial q_n} & \frac{\partial f_1}{\partial p_1} & \dots & \frac{\partial f_1}{\partial p_n} \\ \dots & \dots & \dots & \dots & \dots & \\ \frac{\partial f_d}{\partial q_1} & \dots & \frac{\partial f_d}{\partial q_n} & \frac{\partial f_d}{\partial p_1} & \dots & \frac{\partial f_d}{\partial p_n} \end{bmatrix}$$

Problema stiff denean, erabili ohi den atalen hasieraketa $Y_i^{[0]} = y_n$, *(*$i = 1, \dots, s$*) da eta horregatik ekuazio sistema linealaren* M *matrizea beste modu honetan adierazten da ([14]),*

$$M = R'(Y^{[0]}) = I_s \otimes I_d - hA \otimes J, \;\; J = f'(y_n).$$

8.3. IRK-Newton gure formulazioa.

Lehen urratsa.

Gure IRK formulazioan, Newton metodoa aplikatzeko orduan egokiagoa da L_i *(*$i = 1, \dots, s$*), ezezagunak eta* Y_i *(*$i = 1, \dots, s$*), aldagai auxliarrak kontsideratzea (zergaitik?),*

$$r_i = -L_i + hb_i f(Y_i), \;\; Y_i = y_n + h \sum_{j=1}^{s} m_{ij} L_j, \;\; i = 1, \dots, s.$$

Modu baliokidean notazio hau erabiliko dugu,

$$R(L) = -L + h \; B \; F(L),$$

non,

$$L^T = (L_1, \dots, L_s), \quad B = \begin{bmatrix} b_1 & 0 & \dots & 0 \\ 0 & b_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & b_s \end{bmatrix}, \quad F(Y) = \begin{bmatrix} f(Y_1) \\ f(Y_2) \\ \dots \\ f(Y_s) \end{bmatrix},$$

$$Y_i = y_n + h \sum_{j=1}^{s} m_{ij} L_j, \;\; i = 1, \dots, s.$$

Newton sinplifikatuaren metodoa,
non

$$R'(Y^{[0]}) = I_s \otimes I_d - h \begin{bmatrix} b_1 \, m_{11} \, f'(Y_1^{[0]}) & \dots & b_1 \, m_{1s} \, f'(Y_1^{[0]}) \\ b_2 \, m_{21} \, f'(Y_2^{[0]}) & \dots & b_2 \, m_{2s} \, f'(Y_2^{[0]}) \\ \dots & \dots & \dots \\ b_s \, m_{s1} \, f'(Y_s^{[0]}) & \dots & b_s \, m_{ss} \, f'(Y_s^{[0]}) \end{bmatrix}$$

$M = R'(L^{[0]})$;
for *(k=1,2,...)* **do**
  $Solve(\triangle L^{[k]} = -R(L^{[k]})/M,\ \triangle L^{[k]})$;
  $L^{[k]} = L^{[k-1]} + \triangle L^{[k]}$;
end

ALGORITHM 22: Newton sinplifikatua

Newton sinplifikatuaren metodoa problema stiff-etarako, non $Li^{[0]} = 0$ $(i = 1, \ldots, s)$,

$$M = R'(L^{[0]}) = I_s \otimes I_d - h\ (BAB^{-1} \otimes J),\ \ J = f'(y_n).$$

Bigarren urratsa.

Ekuazio sistema-lineala askatzeko ezezagun $Z \in \mathbf{R}^d$ *berri bat gehituko dugu,*

$$Z = \frac{1}{2}(y_n + y_{n+1}) \ \ \Rightarrow \ \ Z = y_n + \frac{1}{2}\sum_{i=1}^{s} L_i.$$

Beraz, ekuazio sistema-lineala,

$$r_i = L_i - hb_i f(Y_i),$$
$$0 = -Z + y_n + \frac{1}{2}\sum_{i=1}^{s} L_i$$

non

$$Y_i = y_n + h\sum_{j=1}^{s} m_{ij} L_j,\ \ i = 1, \ldots, s.$$

Notazio orokorra $W = \{L, Z\}$,

$$R(W) = \begin{cases} R(L) = -L + h\ B\ F(L) \\ 0 = -Z + y_n + \frac{1}{2}\sum_{i=1}^{s} L_i \end{cases}$$

Newton sinplifikatuaren metodoa,

$M = R'(W^{[0]})$;
for *(k=1,2,...)* **do**
  $Solve(\triangle W^{[k]} = -R(W^{[k]})/M,\ \triangle W^{[k]})$;
  $L^{[k]} = L^{[k-1]} + \triangle L^{[k]}$;
end

ALGORITHM 23: Newton sinplifikatua

non

$$R'(W^{[0]}) = \begin{bmatrix} I_s \otimes I_d - h\alpha_{11}\, f'(Y_1^{[0]}) & \dots & I_s \otimes I_d - h\alpha_{1s}\, f'(Y_1^{[0]}) & -hb_1 f'(Y_1^{[0]}) \\ I_s \otimes I_d - h\alpha_{21}\, f'(Y_2^{[0]}) & \dots & I_s \otimes I_d - h\alpha_{2s}\, f'(Y_2^{[0]}) & -hb_2 f'(Y_2^{[0]}) \\ \dots & \dots & \dots & \\ I_s \otimes I_d - h\alpha_{s1}\, f'(Y_s^{[0]}) & \dots & I_s \otimes I_d - h\alpha_{ss}\, f'(Y_s^{[0]}) & -hb_s f'(Y_s^{[0]}) \\ & & & \\ 1/2 I_d & \dots & 1/2 I_d & -I_d \end{bmatrix}$$

eta

$$\alpha_{ij} = (b_i\, m_{ij} - b_i/2).$$

Newton sinplifikatuaren metodoa problema stiff-etarako,

$$M = R'(W^{[0]}) = \begin{bmatrix} & & & -hb_1\, J \\ & & & -hb_2\, J \\ & I_s \otimes I_d - h((BAB^{-1} - b/2) \otimes J) & & \dots \\ & & & \dots \\ & & & -hb_s\, J \\ 1/2 I_d & 1/2 I_d \quad \dots & 1/2 I_d & -I_d \end{bmatrix}$$

Algoritmoa.

Algoritmoaren hainbat zehaztapen emango ditugu,

1. *LAPACK.*

   LU deskonposaketa *egiteko funtzioa,*

   ```
   GETRF(LAPACK_ROW_MAJOR, n, m, MM, lda, ipiv);
   ```

   *Ekuazio sistemaren ebazpena (*Solve*) egiteko,*

   ```
   GETRS(LAPACK_ROW_MAJOR, trans, n, nrhs, MM, lda, ipiv, fl, ldb);
   ```

$e = 0$;
for $n \leftarrow 0$ **to** *(*$endstep - 1$*)* **do**
  $k = 0$;
  Init $(L_n^{[0]})$;
  $Y_{n,i}^{[0]} = y_n + \left(e + \sum_{j=1}^{s} \mu_{ij} L_{n,j}^{[0]}\right)$;
  $M = \mathrm{LU}(R'(W_n^{[0]}))$;
  while *(konbergentzia lortu)* **do**
    $k = k + 1$;
    Solve $(\triangle W^{[k]} = -R(W^{[k]})/M,\ \triangle W^{[k]})$;
    $L^{[k]} = L^{[k-1]} + \triangle L^{[k]}$;
    $Y_{n,i}^{[k]} = y_n + \left(e + \sum_{j=1}^{s} \mu_{ij} L_{n,j}^{[k]}\right)$;
  end
  $\delta_n = \sum_{i=1}^{s} L_{n,i}^{[k]} + e$;
  $y_{n+1} = y_n + \delta_n$;
  $e = (y_n - y_{n+1}) + \delta_n$;
end

ALGORITHM 24: IRK (Newton-sinplifikatua).

2. *Geratze erizpidea.*

   Newton metodoan, L *atalen araberako geratze irizpide baliokidea erabili dugu,*

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

   Iterazioak jarraitu, honako baldintza betetzen den artean,

$$\exists j \in \{1, \ldots, sd\}\ ,\ |\triangle_j^{[1]}| > |\triangle_j^{[2]|} > \cdots > |\triangle_j^{[k]}| > 0. \qquad (8.2)$$

8.4. Laburpena.

9. Kapitulua

IRK: Eguzki-sistema.

9.1. Sarrera.

Koordenatu kartesiarrak erabiltzearen abantaila.

9.2. Meta-Algoritmoa.

Demagun Hamiltondar banagarria,

$$H(y) = H_A(y) + H_B(y),$$

non $H_A \gg H_B$.

Hau izanik dagokion hasierako problema orokorra,

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

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

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

Adibidea.

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

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

Eguzkiari dagokion azpindizea $i = 0$ *kontsideratzen badugu,*

$$H_k(q,p) = \frac{1}{2}\sum_{i=0}^{N}\frac{p_i^2}{m_i} - Gm_0\sum_{i=1}^{N}\frac{m_i}{\|q_i - q_0\|} \tag{9.1}$$

$$H_I(q) = \sum_{1\le i<j\le N}^{N}\frac{G\, m_i m_j}{\|q_j - q_i\|} \tag{9.2}$$

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

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

Batetik, $f_q(y)$ ekuazio diferentzialen deskonposaketa honakoa da,

$$\dot{q} = f_q(y) = v_i,\ i = 0,\dots,N.$$

$$\dot{q} = f_q(y) \ \Rightarrow\ \begin{cases} k_q(y) = v_i, & i = 0,\dots,N. \\ g_q(y) = 0, & i = 0,\dots,N. \end{cases} \tag{9.3}$$

Bestetik, $f_v(y)$ ekuazio diferentzialen deskonposaketa honakoa da,

$$\dot{v} = f_v(y) = \sum_{j=0,j\neq i}^{N}\frac{Gm_j}{\|q_j - q_i\|^3}(q_j - q_i) \quad i = 0,\dots,N.$$

$$\dot{v} = f_v(y) \ \Rightarrow\ \begin{cases} k_v(y) = \begin{cases} \dot{v}_0 = \sum_{j=1,j\neq i}^{N}\frac{Gm_j}{\|q_j - q_0\|^3}(q_j - q_0). \\ \dot{v}_i = \frac{Gm_0}{\|q_0 - q_i\|^3}(q_0 - q_i), \quad i = 1,\dots,N. \end{cases} \\ g_v(y) = \begin{cases} \dot{v}_0 = 0. \\ \dot{v}_i = \sum_{j=1,j\neq i}^{N}\frac{Gm_j}{\|q_j - q_i\|^3}(q_j - q_i), \quad i = 1,\dots,N. \end{cases} \end{cases} \tag{9.4}$$

Meta-algoritmoa.

IRK metodoaren formulazioa gogoratuz,

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

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

s-ataleko IRK metodoaren iterazio bakoitzean, ($s*d$) ezezagunetako (Y_i) ekuazio-sistema askatu behar dugu:

$$Y_i - y_n - \sum_{j=1}^{s} \mu_{ij}\ hb_j \Big(k(Y_j) + g(Y_j) \Big) = 0, \quad i = 1, \ldots, s.$$

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

Garapena.

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

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

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

IRK metodoaren ekuazio-sistemari aplikatuz,

$$\triangle Y_i^{[k]} = -\frac{\big(Y_i^{[k]} - y_n - \sum\limits_{j=1}^{s} \mu_{ij}\ hb_j\big(k(Y_j^{[k]}) + g(Y_j^{[k]})\big)\big)}{\big(1 - \sum\limits_{j=1}^{s} \mu_{ij}\ hb_j\big(k'(Y_j^{[k]}) + g'(Y_j^{[k]})\big)\big)}$$

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

$$\triangle Y_i^{[k]} = \sum_{j=1}^{s} \mu_{ij}\ hb_j\big(k'(Y_j^{[k]}) + g'(Y_j^{[k]})\big) \triangle Y_j^{[k]} + \delta_i^{[k]}, \tag{9.5}$$

non,

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

Lortutako espresioa garatuko dugu.

1. ***Lehen hurbilpena.***

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

$$\triangle Y_i^{[k]} \approx \sum_{j=1}^{s} \mu_{ij}\ hb_j\ k'(Y_j^{[k]})\ \triangle Y_j^{[k]} + \delta_i^{[k]} \tag{9.6}$$

2. ***Bigarren hurbilpena (Linealizazioa).***

$$k(Y_j^{[k+1]}) = k'(Y_j^{[k]})(Y_j^{[k+1]} - Y_j^{[k]}) + k(Y_j^{[k]}) + O(\|\triangle Y_j^{[k]}\|^2)$$
$$\triangle Y_j^{[k]} = Y_j^{[k+1]} - Y_j^{[k]}$$

   Honako hurbilpena,

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

   ordezkatuz eta garatuz,

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

$$\triangle Y_i^{[k]} = -Y_i^{[k]} + y_n + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ k(Y_j^{[k]} + \triangle Y_j^{[k]})\ + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ g(Y_j^{[k]}). \tag{9.7}$$

3. ***Ekuazioak berridatiziz.***

   $\triangle Y_i^{[k]} = Y_i^{[k+1]} - Y_i^{[k]}$ *definizioa erabiliaz,*

$$Y_i^{[k+1]} = y_n + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ k(Y_j^{[k+1]})\ + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ g(Y_j^{[k]}). \tag{9.8}$$

Meta-Algoritmoa.

IRK metodoa aplikatzeko meta algoritmoa planteatuko dugu.

$e = 0$;
for $n \leftarrow 1$ **to** $endstep$ **do**
  $k = 0$;
  Hasieratu $Y_i^{[0]}$, $i = 1, \ldots, s$;
  while *(konbergentzia lortu)* **do**
    $k = k + 1$;
    if $k == 1$ **then**
      Hasieratu $G_i^{[0]}$;
    else
      $G_i^{[k-1]} = \sum_{j=1}^{s} \mu_{ij}\ hb_j\ g(Y_j^{[k-1]})$;
    end
    Askatu $Y_i^{[k]} = y_{n-1} + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ k(Y_j^{[k]})\ + G_i^{[k-1]}$;
  end
  $L_i = hb_i f(Y_i^{[k]})$;
  $\delta_n = \sum_{i=1}^{s} L_i + e$;
  $y_n = y_{n-1} + \delta_n$;
  $e = (y_{n-1} - y_n) + \delta_n$;
end

ALGORITHM 25: Main Algorithm

Meta-algoritmoari buruzko hainbat ohar:

1. *Barne iterazioak.*

   Kanpo iterazioa Newton metodoa aplikatuz zehaztu dugu. Barne iterazio aldiz, aukera ezberdinak ditugu.

Barne-interazioa *puntu finkoaren bidez:*

$l = 0$;
$Y_i^{[k,0]} = Y_i^{[k-1]}$;
while *(konbergentzia lortu)* **do**
  $l = l + 1$;
  $K_i^{[k,l]} = k(Y_j^{[k,l-1]})$;
  $Y_i^{[k,l]} = y_{n-1} + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ K_j^{[k,l]}\ + g_i^{[k-1]}$;
end

ALGORITHM 26: Main Algorithm

2. *Problema independenteak.*

   Era honetako deskonposaketa bat dugunean,

$$f\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} f_1(y_1) \\ f_2(y_2) \end{pmatrix} + \begin{pmatrix} g_1(y_1, y_2) \\ g_2(y_1, y_2) \end{pmatrix},$$

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

Orokorpena.

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

$$\textit{1. maila} \begin{cases} \textit{Eredu osoa. } f(y) \\ \textit{Eredu sinplea. } \tilde{f}(y) \end{cases} \Rightarrow \quad f = \tilde{f} + (f - \tilde{f}) \tag{9.9}$$

$$\textit{2. maila} \begin{cases} \textit{Eredu osoa. } \tilde{f}(y) \\ \textit{Eredu sinplea. } \tilde{\tilde{f}}(y) \end{cases} \Rightarrow \quad \tilde{f} = \tilde{\tilde{f}} + (\tilde{f} - \tilde{\tilde{f}}) \tag{9.10}$$

Adibidea.

Demagun ekuazio diferentzialak, m perturbazio funtzio dituela,

$$\dot{y} = f(y) = k(y) + g1(y) + g2(y) + \cdots + gm(y)$$

non $k(y) << gl(y)$, $l = 1, \ldots, m$.

$m = 2$ *deneko kasu partikulara aztertuko dugu,*

$$\dot{y} = f(y) = k(y) + g1(y) + g2(y).$$

Askatu $Y_i = y_{n-1} + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ (k(Y_j) + g1(Y_j) + g2(Y_j))$;

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

  $k = k + 1$;

  $g2_i^{[k-1]} = \sum_{j=1}^{s} \mu_{ij}\ hb_j\ g2(Y_j^{[k-1]})$;

  Askatu $Y_i^{[k]} = y_{n-1} + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ (k(Y_j^{[k]}) + g1(Y_j^{[k]}))\ + g2_i^{[k-1]}$;

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

    $l = l + 1$;

    $g1_i^{[k,l-1]} = \sum_{j=1}^{s} \mu_{ij}\ hb_j\ g1(Y_j^{[k,l-1]})$;

    Askatu $Y_i^{[k,l]} = y_{n-1} + \sum_{j=1}^{s} \mu_{ij}\ hb_j\ k(Y_j^{[k,l]}) + g1_i^{[k,l-1]}\ + g2_i^{[k-1]}$;

  end

end

ALGORITHM 27: Main Algorithm

Erlatibitate efektua gehitzerakoan, N-gorputzen problemari dagokion ekuazio diferentziala,

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

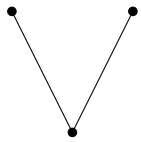
```
k(y): kepleriarra.
g(y): planeten arteko grabitazio interakzioak.
rs(y): eguzkiarekiko erlatibitate efektua.
rn(y): planeten arteko erlatibitate efektuak.
```

Adierazpena.

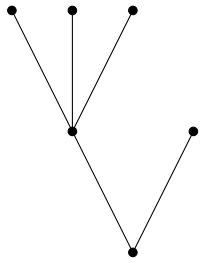
Ekuazio diferentzialen deskonposaketak, zuhaitz moduan adieraz daitezke.

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

- $$f \left(\begin{array}{c} y_1 \\ y_2 \end{array} \right) = \left(\begin{array}{c} f_1(y_1) \\ f_2(y_2) \end{array} \right) + \left(\begin{array}{c} g(y) \end{array} \right),$$



$$f_1 \left(\begin{array}{c} y_1 \end{array} \right) = \left(\begin{array}{c} f_{11}(y_{11}) \\ f_{12}(y_{11}, y_{12}) \\ f_{13}(y_{11}, y_{12}, y_{13}) \end{array} \right) + \left(\begin{array}{c} g_1(y_1) \end{array} \right),$$



9.3. Denbora birparametrizazioa.

9.3.1. Denbora birparametrizazioa.

Demagun jatorrizko ekuazio diferentziala,

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

non y menpeko aldagaia eta t aldagai askea den.

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

$$y = z,$$

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

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

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

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

9.3.2. Adibidea.

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

$$s(q) = r_{10}^{3/2} \tag{9.11}$$

$$r_{10} = \|q1 - q0\|_2 \tag{9.12}$$

non $q_1 = (q1_x, q1_y, q1_z)$ merkurio plantearen kokapena eta $q_0 = (q0_x, q0_y, q0_z)$ eguzkiaren kokapena den.

9.4. Laburpena.

10. Kapitulua

Eztabaida.

10.1. Sarrera.

10.2. Laburpena.

11. Kapitulua

Konklusioak.

11.1. Sarrera.

11.2. Laburpena.

Bibliografia

[1] *Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm.* nature, *324(6096):446–449, 1986.*

[2] *André Berger.* A brief history of the astronomical theories of paleoclimates. *Springer, 2012.*

[3] *Sergio Blanes, Fernando Casas, Ariadna Farres, Jacques Laskar, Joseba Makazaga, and Ander Murua. New families of symplectic splitting methods for numerical integration in dynamical astronomy.* Applied Numerical Mathematics, *68:58–72, 2013.*

[4] *Dirk Brouwer. On the accumulation of errors in numerical integration.* The Astronomical Journal, *46:149–153, 1937.*

[5] *John Charles Butcher.* Numerical Methods for Ordinary Differential Equations. *Second edition, Wiley, 2008.*

[6] *J Carrier, Leslie Greengard, and Vladimir Rokhlin. A fast adaptive multipole algorithm for particle simulations.* SIAM journal on scientific and statistical computing, *9(4):669–686, 1988.*

[7] *John E Chambers. A hybrid symplectic integrator that permits close encounters between massive bodies.* Monthly Notices of the Royal Astronomical Society, *304(4):793–799, 1999.*

[8] *Robert M Corless and Nicolas Fillion. A graduate introduction to numerical methods.* AMC, *10:12, 2013.*

[9] *Martin J Duncan, Harold F Levison, and Man Hoi Lee. A multiple time step symplectic algorithm for integrating close encounters.* The Astronomical Journal, *116(4):2067, 1998.*

[10] *Ariadna Farrés, Jacques Laskar, Sergio Blanes, Fernando Casas, Joseba Makazaga, and Ander Murua. High precision symplectic integrators for*

the solar system. Celestial Mechanics and Dynamical Astronomy, *116(2): 141–174, 2013.*

[11] Kang Feng and Mengzhao Qin. Symplectic geometric algorithms for hamiltonian systems. *Springer, 2010.*

[12] William M Folkner, James G Williams, Dale H Boggs, Ryan S Park, and Petr Kuchynka. *The planetary and lunar ephemerides de430 and de431*. Interplanet. Netw. Prog. Rep, *196:1–81, 2014.*

[13] Toshio Fukushima. *Reduction of round-off error in symplectic integrators*. The Astronomical Journal, *121(3):1768, 2001.*

[14] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Geometric numerical integration: structure-preserving algorithms for ordinary differential equations, *volume 31. Springer Science & Business Media, 2006.*

[15] Ernst Hairer, Robert I McLachlan, and Alain Razakarivony. *Achieving brouwer's law with implicit runge–kutta methods*. BIT Numerical Mathematics, *48(2):231–243, 2008.*

[16] David M Hernandez and Edmund Bertschinger. *Symplectic integration for the collisional gravitational n-body problem*. Monthly Notices of the Royal Astronomical Society, *452(2):1934–1944, 2015.*

[17] Nicholas J Higham. Accuracy and stability of numerical algorithms. *Siam, 2002.*

[18] Nicholas J Higham. *Programming languages: An applied mathematics view*. *2015.*

[19] MP Calvo JM Sanz-Serna. Numerical Hamiltonian problems. *Chapman and Hall, 1994.*

[20] Jacques Laskar, Agnes Fienga, Mickael Gastineau, and Herve Manche. *La2010: a new orbital solution for the long-term motion of the earth*. Astronomy & Astrophysics, *532:A89, 2011.*

[21] Jacques Laskar and Philippe Robutel. *High order symplectic integrators for perturbed hamiltonian systems*. Celestial Mechanics and Dynamical Astronomy, *80(1):39–62, 2001.*

[22] Benedict Leimkuhler and Sebastian Reich. Simulating hamiltonian dynamics, *volume 14. Cambridge University Press, 2004.*

[23] Harold F Levison and Martin J Duncan. The long-term dynamical behavior of short-period comets. Icarus*, 108(1):18–36, 1994.*

[24] Robert I McLachlan. Composition methods in the presence of small parameters. BIT Numerical Mathematics*, 35(2):258–268, 1995.*

[25] A Morbidelli. Modern integrations of solar system dynamics. Annual Review of Earth and Planetary Sciences*, 30(1):89–112, 2002.*

[26] Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, and Serge Torres. Handbook of floating-point arithmetic*. Springer Science & Business Media, 2009.*

[27] Nobelprize.org. The nobel prize in chemistry 2013, May 2014.

[28] Michael L Overton. Numerical computing with IEEE floating point arithmetic*. Siam, 2001.*

[29] M Papadrakakis, M Kojic, and I Tuncer. Fast detection of chaotic or regular behavior of double pendulum system: application of the fast norm vector indicator method.

[30] Mark Sofroniou and Giulia Spaletta. Derivation of symmetric composition constants for symmetric integrators. Optimization Methods and Software*, 20(4-5):597–613, 2005.*

[31] Pat H Sterbenz. Floating-point computation*. Prentice Hall, 1973.*

[32] Gerald J Sussman and Jack Wisdom. Chaotic evolution of the solar system. Technical report, DTIC Document, 1992.

[33] Jack Wisdom and Matthew Holman. Symplectic maps for the n-body problem. The Astronomical Journal*, 102:1528–1538, 1991.*