

Ekuaizio diferentzial arruntak askatzeko zenbakizko metodoak

Elisabete Alberdi Celaya



© Udako Euskal Unibertsitatea

© Elisabete Alberdi Celaya

ISBN: 978-84-8438-420-5

Lege-gordailua: BI-1218-2012

Inprimategia: PRINTHAUS S.L., Bilbo

Azalaren diseinua: Igor Markaida

Hizkuntza-zuzenketen arduraduna: Ander Altuna Gabiola

Banatzaileak: UEU. Erribera 14, 1. D BILBO telf. 946790546 Faxe. 944793039

Helbide elektronikoa: argitalpenak@ueu.org

www.ueu.org

Elkar Banaketa: Igerabide, 88 DONOSTIA

Galarazita dago liburu honen kopia egitea, osoa nahiz zatikakoa, edozein modutara delarik ere, edizio honen Copyright-jabeen baimenik gabe.

Aurkibidea

1. HASIERAKO KONTZEPTUAK	7
1.1. Ekuazio diferentzial arruntak	7
1.2. Ekuazio diferentzial arrunten zenbait adibide	10
1.3. Hasierako baliodun EDA baten soluzio analitikoa eta zenbakizko soluzioa	17
1.4. Soluzioaren existentzia eta bakartasuna	19
1.5. EDA baten zenbakizko kalkulua: Euler-en metodoak	19
1.5.1. Euler-en metodo esplizitua	19
1.5.2. Euler-en metodo implizitua	25
1.5.2.1. Newton Raphson-en metodoa	26
1.5.2.2. Euler-en metodo implizituaren inplementazioa MATLABen	28
1.6. Metodo trapezoidala	33
1.7. Etapa anitzeko metodoak eta pauso anitzeko metodoak	34
1.7.1. Lehen ordenako EDak askatzeko pauso anitzeko metodo linealak	34
1.8. Zenbakizko metodo baten errorea eta ordena	35
1.9. Egonkortasuna	40
1.10. Euler-en metodoen egonkortasun-eremuak	47
1.10.1. Euler-en metodo esplizituaren egonkortasun-eremua	47
1.10.2. Euler-en metodo implizituaren egonkortasun-eremua	48
1.10.3. Metodo trapezoidalaren egonkortasun-eremua	49
1.11. Errorea vs amplifikazio-faktorea pauso bakarreko metodoetan	54
2. ZURRUNTASUNA	57
2.1. Zurruntasun kontzeptuaren inguruko zenbait kontsiderazio	59
2.1.1. Zurruntasuna ez da bakarrik sistemetan gertatzen	59
2.1.2. Eraitza bera duten bi EDaren portaera	64
2.1.3. Zurruntasun-erradio bera duten bi EDaren portaera	65
2.1.4. Zurruntasunari buruzko zenbait ondorio	66
2.2. Zenbait ekuazio diferentzial zurrunen soluzioak	66

3. RUNGE-KUTTA METODOAK.....	73
3.1. Metodarako sarrera	73
3.2. Ordena-baldintzak.....	75
3.3. Egonkortasun-ezaugarriak	84
3.4. Kateatutako Runge-Kutta metodoak.....	91
 4. PAUSO ANITZEKO ZENBAIT METODO LINEAL EZAGUN	 101
4.1. Ordena-baldintzak.....	103
4.2. Egonkortasun-ezaugarriak	104
4.3. Interpolazio polinomikoa, atzerakako diferentziak.....	105
4.4. Adams Bashforth-en metodoak.....	106
4.5. Adams Moulton-en metodoak.....	109
4.6. Adams-en iragarle-zuzentzaile eskemak.....	114
4.7. Backward Differentiation Formulae metodoak.....	118
 5. PAUSO ANITZEKO ZENBAIT METODO LINEALI EGINDAKO	
HAINBAT ALDAKETA.....	125
5.1. Bigarren deribatua darabilten pauso anitzeko metodoak	125
5.1.1. Enright-en metodoak.....	126
5.1.2. SDBDF metodoak.....	129
5.1.3. Bigarren deribatua darabilen pauso anitzeko metodo	
eraginkor bat.	131
5.2. Pauso anitzeko zenbakizko metodo hedatuak.....	132
5.2.1. EBDF metodoak	132
5.2.2. MEBDF metodoak.....	135
5.3. Pauso anitzeko metodo konbinatuak.....	137
5.4. Pauso anitzeko kokapeneko metodoak	139
5.5. NDF metodoak.....	144
5.6. A-BDF metodoak.....	145
5.7. Bigarren deribatua eta super-etorkizuneko puntuak darabiltzaten	
zenbait metodo	146
5.7.1. Second Derivative Extended Methods (E2BD) familia.....	146
5.7.2. New SDMM metodoak.....	148
5.8. A-EBDF metodoak	149
5.9. EBDFetan zuzentzailea aldatuta lortzen diren beste metodo batzuk	149
5.9.1. ENDF metodoak	149
5.9.2. ENBDF eta EBNDF metodoak.....	152

6. MATLAB-EKO ZENBAIT FUNTZIO EDA-K ASKATZEKO.....	157
6.1. EDAk askatzeko ode45 eta ode23 funtzioak	157
6.1.1. Errorearen estimazioa ode45 eta ode23 funtzioetan	158
6.1.2. Pauso-tamainaren kontrola	161
6.1.3. Lehenengo pauso-tamaina	162
6.1.4. Pauso on baten osteko hurrengo pauso-tamaina	163
6.1.5. Pauso txar baten osteko hurrengo pauso-tamaina.....	165
6.2. Ode15s funtzioa	167
6.2.1. Zenbait ezaugarri konputazional.....	167
6.2.2. Diferentzien matrizea.....	169
6.2.3. Errorearen estimazioa ode15s funtzioan.....	171
6.2.4. Lehenengo pauso-tamaina	173
6.2.5. Pauso on baten osteko hurrengo pauso-tamaina	173
6.2.6. Pauso txar baten osteko hurrengo pauso-tamaina.....	176
6.2.7. Pauso-tamaina aldatu ostean diferentzien matrizearen eguneraketa	177
6.3. Ode45 eta ode15s aplikatuz zenbait adibide.....	180
7. ZENBAIT METODOREN PROGRAMAZIOA MATLAB-EN	191
7.1. EDA linealetarako zenbait metodo inplizituren aplikazioa eta programazioa.....	191
7.1.1. EDA linealetarako Euler inplizitua	193
7.1.2. EDA linealetarako metodo trapezoidala	198
7.1.3. EDA linealetarako Adams Moulton-en metodoa	201
7.1.4. EDA linealetarako BDF metodoak	206
7.1.5. EDA linealetarako NDF metodoak	210
7.1.6. EDA linealetarako EBDF metodoak	215
7.2. Edozein motatako EDAk askatzeko zenbait metodo esplizituren programazioa.....	221
7.2.1. Edozein motatako EDAtarako Adams Bashforth-en metodoak ..	221
7.2.2. Edozein motatako EDAtarako Runge-Kutta metodoak.....	223
7.3. Edozein motatako EDAk askatzeko zenbait metodo inplizituren programazioa.....	226
7.3.1. Metodo trapezoidalaren forma orokorra	227
7.3.2. Adams Moulton-en metodoaren forma orokorra	229
7.3.3. BDFen forma orokorra.....	232
7.3.4. EBDFen forma orokorra	237
BIBLIOGRAFIA.....	241

1. Hasierako kontzeptuak

Eredu matematikoak sistema edo gertakari bat deskribatzeko balio duten tresnak dira. Eredu matematiko baten formulazioa sisteman eragina duten, hau da, sisteman aldaketa sortzen duten aldagaien identifikazioa eginez hasten da. Ondoren, sistemari buruzko arrazoizko hipotesiak ezartzen dira eta erabil daitezkeen lege enpirikoak identifikatzen dira. Hipotesi horietako batzuek aurretiaz zehaztu ditugun aldagai batzuen aldaketaren neurria adieraziko dute. Hipotesi horien enuntziatu matematikoa deribatuak agertzen diren ekuazio bat edo ekuazio-sistema bat izango da, eta horixe da ekuazio diferentziala edo ekuazio diferentzialen sistema bat (Alberdi, 2009).

Zientzia, ingeniari, ekonomia eta beste esparru batzuetan gertatzen diren fenomenoak deskribatzeko eredu matematiko bat darabilgunean, ohikoa da ekuazio diferentzialak agertzea. Gehienetan eredu hori hain da konplexua ezen ia ezinezkoa izaten baita emaitza zehatza zein emaitza hurbildua ordenagailuaren laguntzarik gabe eskuz aurkitzea. Horrelakoetan, ordenagailu bidezko simulazioa beharrezkoa izaten da eta horretara jotzen da.

1.1. EKUAZIO DIFERENTZIAL ARRUNTAK

Ekuazio diferentzial arrunt bat (edo ekuazio diferentzial arrunten sistema bat), aldagai bakarraren menpe dagoen funtzio bat (edota aldagai bakarraren menpe dauden funtzioak) emaitzatzen duen (dituen) eta deribatu arruntak agertzen diren ekuazioa (edo ekuazio-sistema) da. Adibidez, $y = y(t)$ izanik, ondorengo ekuazioa kontsideratuko dugu:

$$t^2 + y \cdot y' = 0 \quad (1.1)$$

(1.1) ekuazioan, t aldagaiarekiko deribatua adierazteko $'$ ikurra erabili dugu, hau da:

$$y' = \frac{dy}{dt}$$

t aldagaiarekiko deribatua kalkulatzeko ari garenez, t aldagai independentea dela esaten da, eta ezezaguna, kasu honetan y , mendeko aldagaia dela esaten da. (1.1) ekuazioan aldagai bakarraren menpe dagoen funtzio bat daukagu eta bertako deribatua arrunta denez, (1.1) ekuazio diferentzial arrunta dela esaten da (Aguirregabiria, 2000: 1). EDA sigla erabiltzen da ekuazio diferentzial arrunta adierazteko.

Ekuazio diferentzial arrunt bakarra izan beharrian, ekuazio diferentzial arruntaren sistema bat ere izan dezakegu. Kasu horretan, aldagai bakarraren menpe dauden funtzioak emaitzatzen dituen eta deribatu arruntak agertzen diren ekuazio-sistema izango dugu. Adibidez, $y_1 = y_1(t)$ eta $y_2 = y_2(t)$ izanik:

$$\begin{cases} y_1' = -y_1 + 4y_2 \\ y_2' = -4y_1 - y_2 \end{cases} \quad (1.2)$$

(1.2) sisteman, t aldagaiaren menpe dauden bi funtzio, $y_1 = y_1(t)$ eta $y_2 = y_2(t)$, eta beraien deribatu arruntak agertzen dira. (1.2) sistema ekuazio diferentzial arruntaren sistema bat da (EDA sistema bat).

Kasu bietan, bai ekuazio diferentzial arrunt bakarra dugunean bai ekuazio diferentzial arruntaren sistema dugunean, ezaugarri komuna da aldagai independente bakarra agertzea, normalean t letraz adierazten dena eta denbora adieraz dezakeena.

Ekuazioan edo ekuazioetan agertzen den ordenarik altueneko deribatuak emango dio ekuazio diferentzial arruntari bere ordena. Adibidez, lehen ordenako deribatua agertzen bada, ekuazio diferentzial arrunt hori lehen ordenako ekuazio diferentziala izango da:

$$y' = y + y^2 - 2 \quad (1.3)$$

Lehen ordenako ekuazio diferentzial arrunt baten formulazioa honela egiten da:

$$y' = f(t, y(t)) \quad (1.4)$$

m ekuazio diferentzial arruntez osatutako sistema, berriz, honela adierazten da matematikoki:

$$\begin{cases} y_1' = f_1(t, y_1, y_2, \dots, y_m) \\ y_2' = f_2(t, y_1, y_2, \dots, y_m) \\ \vdots \\ y_m' = f_m(t, y_1, y_2, \dots, y_m) \end{cases} \quad (1.5)$$

non, $y_i = y_i(t)$ den eta $f_1, f_2, \dots, f_m$ funtzioak $(m+1)$ argumenturen menpe dauden funtzioak diren. Argumentu guztiak funtzio guztietan agertu beharrik ez dago.

(1.4) formulaz emandako formulazioa erabilgarria da EDA sistema baten kasurako ere, mendeko aldagaia y , bektorea izan daitekeela onartzen badugu. Horrela, $\mathbb{R}^m$ eremuan lanean ari garela esango dugu EDA bat edo EDA sistema bat definitzen duen f funtzioa eta haren soluzioa den y funtzioa honela emanda daudela:

$$\begin{aligned} y &: \mathbb{R} \rightarrow \mathbb{R}^m \\ f &: \mathbb{R} \times \mathbb{R}^m \rightarrow \mathbb{R}^m \end{aligned}$$

Ekuazio diferentzial arruntean ordena handiagoko deribatuak ere ager daitezke. Horrela, n . ordenako EDA bat honela emanda legoke:

$$y^{(n)} = f(t, y, y', y'', \dots, y^{(n-1)}) \quad (1.6)$$

Eta (1.6) adierazpeneko n . ordenako ekuazio diferentzial arrunta, posible da lehen ordenako n ekuazio diferentzial arruntetan banatzea aldagai-aldaketak erabiliz. Horretarako aski da n aldagai hauek era honetan definitzea:

$$\begin{cases} y_1 = y \\ y_2 = y' \\ \vdots \\ y_n = y^{(n-1)} \end{cases} \quad (1.7)$$

Eta aldagai-aldaketa horrekin, (1.6) ekuazioaren bidez emandako n . ordenako ekuazio diferentzial arrunta lehen ordenako n ekuazio diferentzial arrunten sistema bilakatzen da:

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \vdots \\ y_{n-1}' = y_n \\ y_n' = f(t, y_1, y_2, \dots, y_{n-1}, y_n) \end{cases}$$

Era berean, n . ordenako ekuazio bakarraren orde, n . ordenako m ekuazio diferentzial arrunten sistema daukagun kasuan, sistemako ekuazio bakoitzarekin ekuazio bakarreko kasuan egin dugun gauza bera errepikatuz, lehen ordenako $n \times m$ ekuazio diferentzial arrunt dituen sistema lortuko dugu.

Normalean, bat ordena baino handiagoko EDAk, lehen ordenako EDAtara laburtzen dira, nahiz eta badauden bat ordena baino handiagoa duten EDAk ordena bakarrera laburtu beharrik gabe zuzenean askatzeko metodo batzuk. Testu honetan, lehen ordenako ekuazio diferentzial arruntez arduratuko gara.

Baina EDA baten soluzioa zehaztu nahi badugu, EDaz gain beste informazio bat ere behar izaten da. Aukera bat hasierako balioa ematea da. Adibidez, lehen ordenako EDaren kasuan hasierako balioa ematea $y(t_0) = y_0$ balioa ematea da. Bigarren ordenako EDaren kasuan $y(t_0) = y_0$ eta $y'(t_0) = y_1$ balioak eman beharko genituzke hasierako balio bezala.

Definizioa: $T = [t_0, t_N]$ tarte finitu batean definitutako hasierako baliodun lehen ordenako EDA bat ondorengo adierazpenaren bidez emana dator:

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0 \quad (1.8)$$

non $y: [t_0, t_N] \rightarrow \mathbb{R}^m$ eta $f: [t_0, t_N] \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ funtzioak jarraituak diren.

1.2. EKUAZIO DIFERENTZIAL ARRUNTEN ZENBAIT ADIBIDE

Atal honetan lehen ordenako ekuazio diferentzial arrunten zenbait adibide ikusiko ditugu. Adibide horietako batzuetan fenomeno bat deskribatzeko erabiliko dugun eredu matematikotik abiatuko gara.

Adibideak

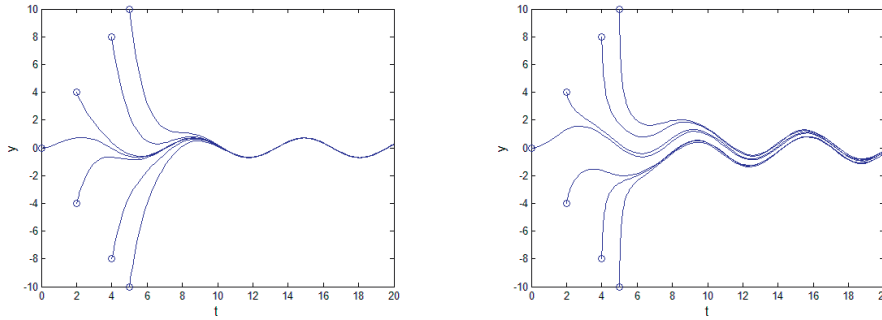
1. Kontsidera dezagun lehen ordenako hurrengo ekuazio diferentzial arrunta (Griffiths eta Highman, 2010: 2-3):

$$y'(t) = \sin(t) - y(t) \quad (1.9)$$

Haren emaitza $y(t) = Ae^{-t} + \frac{1}{2}\sin(t) - \frac{1}{2}\cos(t)$ adierazpenaren bidez emana dator, A zenbaki erreal bat izanik. Beste ekuazio diferentzial arrunt honentzat, ordea, ez da ezagutzen emaitza analitikorik:

$$y'(t) = \sin(t) - 0,1y^3(t) \quad (1.10)$$

Hala eta guztiz, posible da (1.10) ekuazioaren emaitza hurbildu bat aurkitzea zenbakizko metodoez baliatuz. 1.1. irudian (1.9) eta (1.10) ekuazioen soluzio-kurbak irudikatu dira hasierako balio ezberdinetarako, ezkerrean (1.9) ekuazioaren soluzio-kurbak eta eskuinean (1.10) ekuazioarenak. Hasierako balioa 'o' sinboloa erabilia markatu da kasu bietan. Bi EDA horien soluzio-kurbak antzekoak direla ikusten da, bietan hasierako balio berdina hartzen dugunean.



1.1. irudia. (1.9) eta (1.10) ekuazio diferentzialen soluzio-kurbak hasierako balio ezberdinetarako.

2. Kontsidera dezagun ondorengo ekuazio diferentziala (Griffiths eta Highman, 2010: 3-4):

$$y'(t) = \sin(t) - \frac{2}{1+t^4} y(t) \quad (1.11)$$

(1.11) ekuazio diferentzialaren faktore integratzailea ondorengo adierazpenak emana dator:

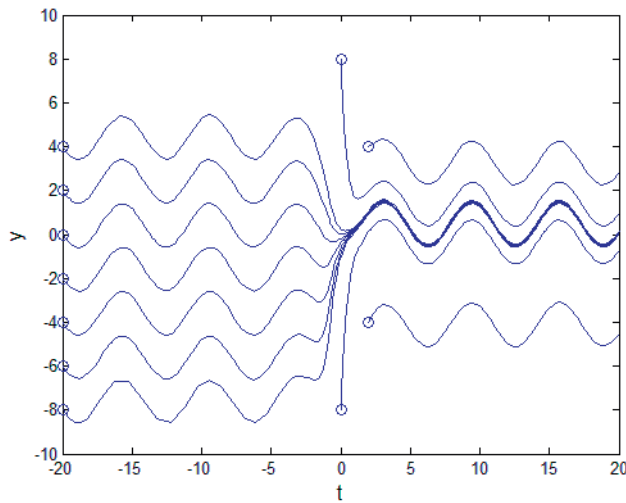
$$g(t) = \left(\frac{\exp\left[2 \tan^{-1}(1 + \sqrt{2}t)\right] t^2 + \sqrt{2}t + 1}{\exp\left[2 \tan^{-1}(1 - \sqrt{2}t)\right] t^2 - \sqrt{2}t + 1} \right)^{1/(2\sqrt{2})}$$

Faktore integratzaile hori (1.11) ekuazioaren soluzio orokorra lortzeko erabiliz, honela lortuko litzateke soluzio orokor hori:

$$y(t) = Ag(t) + g(t) \int_0^t \frac{\sin s}{g(s)} ds \quad (1.12)$$

(1.12) adierazpeneko integrala egin ezik, ez dugu EDaren soluzio orokorra izango. Batzuetan soluzio analitikoa lortzea baino eraginkorragoa izan daiteke zenbakizko metodoak erabiltzea, EDaren emaitza hurbildua lortu eta soluzio-kurbak irudikatzeko.

1.2. irudian (1.11) ekuazioaren emaitza irudikatu da hainbat hasierako balio ezberdinetarako. Berriro ere hasierako balioa irudikatzeko ‘o’ sinboloa erabili da.

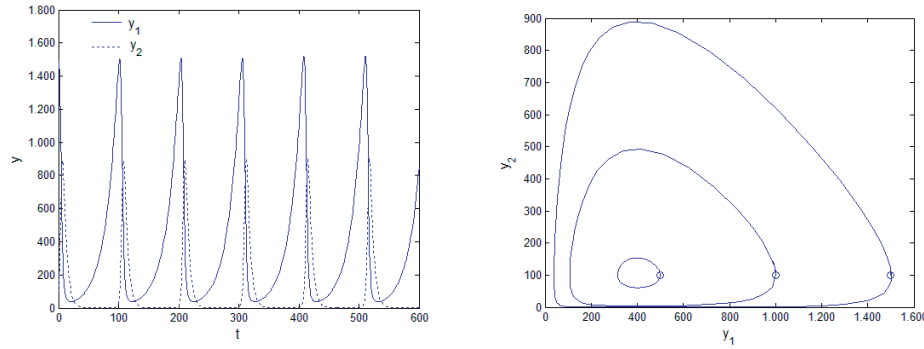


1.2. irudia. (1.11) ekuazio diferentzialaren soluzio-kurbak hasierako balio ezberdinetarako.

3. Lotka Volterra-ren ekuazioak kontsideratuko ditugu (Griffiths eta Highman, 2010: 6). Ekuazio diferentzialen sistema honek piztia harrapatzaileen eta harrapatuen arteko eredu sinplifikatu bat islatzen du. Eman dezagun herrialde bateko untxi eta otsoen kopurua t aldiunean $y_1(t)$ eta $y_2(t)$ direla hurrenez hurren. Hasierako untxi eta otsoen kopuruak $y_1(0)$ eta $y_2(0)$ direla jakinik, t aldiuneko untxi eta otsoen kopurua honako eredu matematiko honen bidez emana dago:

$$\begin{cases} y_1'(t) = 0,05y_1(t)(1 - 0,01y_2(t)) \\ y_2'(t) = 0,1y_2(t)(0,005y_1(t) - 2) \end{cases} \quad (1.13)$$

Eredu horrek zentzua dauka, zeren otso kopurua $y_2(t)$ handitu ahala, untxi kopurua aldatzen den erritmoa, hau da $y_1'(t)$, txikituz baitoa. Gainera, untxi kopurua handitzen bada, untxiak ugaltzen diren abiadura handitu egiten da. 1.500 untxi eta 100 otso dauden hasierako populazio bati dagozkion soluzio-kurbak 1.3. irudiko ez-kerraldean jaso dira. Irudiko eskuinaldean $(y_1(t), y_2(t))$ puntuen kokapena irudikatu da. Irudikatu ditugun hiru kurbak hasieran 100 otso daudela suposatuta egin dira eta hasierako untxi kopurua 500, 1.000 eta 1.500ekoa izan da. Hasierako untxi eta otso kopurua ‘o’ sinboloa erabilia irudikatu da. Eskuineko irudiari begiratuta, populazio horien izaera periodikoa antzeman daiteke.



1.3. irudia. Lotka-Volterra-ren ekuazioen emaitza hasierako balio jakin baterako.

4. Gaixotasun bat zelan zabaltzen den ikusteko ere ekuazio diferentzialak erabiltzea posiblea da (Griffiths eta Highman, 2010: 9-10). Eman dezagun zientzia fikziozko testuinguru batean gaudela eta zonbien izurrite bat dagoela. Erabiliko dugun eredu matematikoan aldiune bakoitzerako funtzio hauen balioak izango ditugu:

- $y_1(t)$ gizakien kopurua izango da.
- $y_2(t)$ zonbien kopurua.
- $y_3(t)$ «hildako» zonbiak dira, eta hauek berriz ere zonbi bihur daitezke.

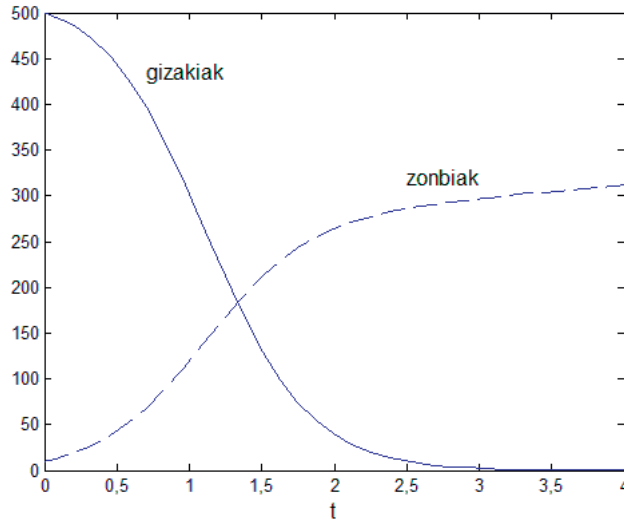
Zonbi batek gizaki bat zonbi bihur dezake. Hau da, gizakiak zonbi egoerara pasa daitezke. Aldiz, zonbi egoetatik gizaki egoerara ezingo da berriz inoiz pasatu. Zonbiak ezin daitezke hil baina gerta daiteke gizaki batek zonbi bat «hiltzea», hau da, denboraldi baterako haren zonbi izaera kentzea. Eredu horren formulazio matematiko sinplifikatu bat hauxe litzateke:

$$\begin{cases} y_1'(t) = -\beta y_1(t)y_2(t), \\ y_2'(t) = \beta y_1(t)y_2(t) + \gamma y_3(t) - \alpha y_1(t)y_2(t), \\ y_3'(t) = \alpha y_1(t)y_2(t) - \gamma y_3(t) \end{cases} \quad (1.14)$$

(1.14) adierazpeneko α , β eta γ konstanteak positiboak dira eta beraien esanahia hauxe da:

- α konstantea gizaki-zonbi erlazioan zonbiak «hiltzea»rekin lotuta dagoen konstantea da.
- β konstantea, gizaki-zonbi erlazioan gizakiak zonbi bihurtzearekin lotuta dagoen konstantea da.
- γ konstantea «hildako» zonbiak berriz ere zonbi egoerara itzultzearekin lotuta dagoen konstantea da.

$\beta = 0,01$, $\alpha = 0,005$ eta $\gamma = 0,02$ kasurako (1.14) sistemaren emaitza irudikatu da 1.4. irudian. Hasierako balioetat hauek hartu dira: $y_1(t) = 500$, $y_2(t) = 10$, $y_3(t) = 500$. Irudi honetan ikus daiteke azkenean gizaki guztiak zonbi bihurtzen direla.



1.4. irudia. (1.14) sistemaren emaitza, gizaki eta zonbi kopuruak.

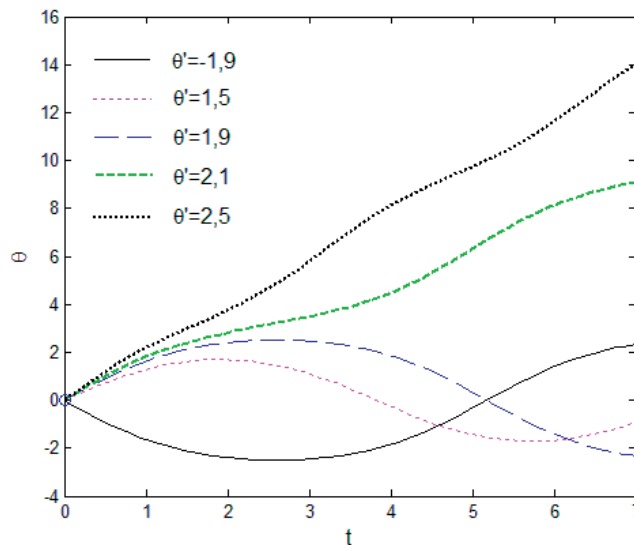
5. Sabaitik bertikalki zintzilik dagoen pendulua hartuko dugu. Soka zurrun batek eta sokaren muturrean dagoen masa batek osatzen dute. Penduluak t aldiunean bertikalarekin osatzen duen angeluari $\theta(t)$ deituko diogu eta beronek bigarren ordenako ondorengo ekuazio diferentzial arrunta betetzen du (Shampine, Gladwell eta Thompson, 2003: 10-11):

$$\theta'' + \sin(\theta) = 0 \quad (1.15)$$

(1.15) ekuazioa, lehen ordenako ekuazio diferentzial arruntak askatzeko dauden zenbakizko metodoak erabiliz askatu nahi badugu, ekuazio hori lehen ordenako bi ekuazio diferentzial arrunt bihurtu beharko dugu. Horretarako aldagai hauek definituko ditugu: $y_1 = \theta$ eta $y_2 = \theta'$. Bi aldagai horiek kontuan izanik, (1.15) ekuazioa lehen ordenako ekuazio diferentzial arruntaren sistema hau bihurtzen da:

$$\begin{cases} y_1' = y_2 \\ y_2' = -\sin(y_1) \end{cases} \quad (1.16)$$

Hasieran penduluak bertikalarekin osatzen duen angelua $\theta(0)=0$ dela joko dugu eta masari bultza egiten diogula $\theta'(0)$ hasierako abiadura emanez. Hasierako abiadura zero denean, pendulua ez da higituko. Hasierako abiadura nulua ez denean eta oso handia ez denean, pendulua ardatz bertikalaren inguruan higituko da atzera eta aurrera. 1.5. irudian jaso dugu $\theta'(0) = -1,9$, $1,5$ eta $1,9$ balioetarako angeluak aldiune ezberdinetan izango lukeen balioa. Hasieran ematen diogun abiadura handiegia bada, pendulua bertikalaren goiko puntaraino igoko da, eta dagoen marruskadura handiegia ez bada, $\theta(t)$ angelua handituz joango da. Hori gertatzen da $\theta'(0) = 2,1$ eta $2,5$ balioetarako adibidez.



1.5. irudia. Penduluak bertikalarekin osatzen duen angelua, (1.5) ekuazioaren emaitza.

6. Prozesu kimiko baten adibidea jarriko dugu azkenik (Butcher, 2008: 14-16). Hiru produktu ditugula, A, B eta C, joko dugu. Beraien arteko hiru erreakzioak hauek izanik:

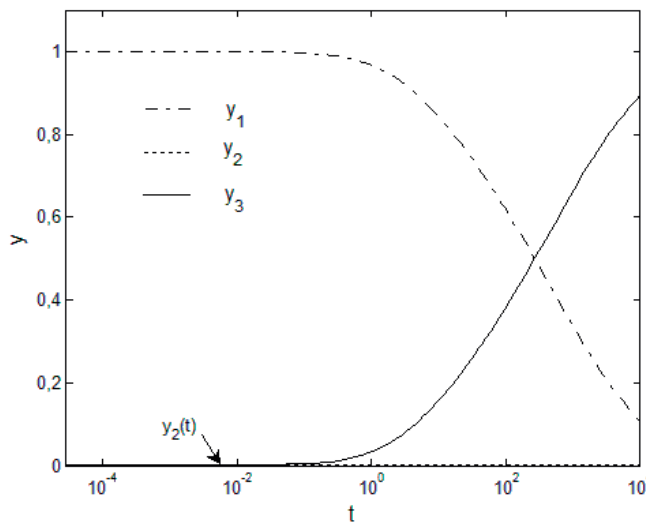


y_1 , y_2 eta y_3 deituko diegu hurrenez hurren A, B eta C produktuen kontzentrazioei. Hiru kontzentrazioen baturak 1 ematen duela suposatuko dugu, hau da, erreakzioak eskalatuta daudela eta erreakzio eratzaileetako bakoitza edozein produkturen kontzentrazioari gehituko zaio, erreaktiboei dagozkien kantitateen

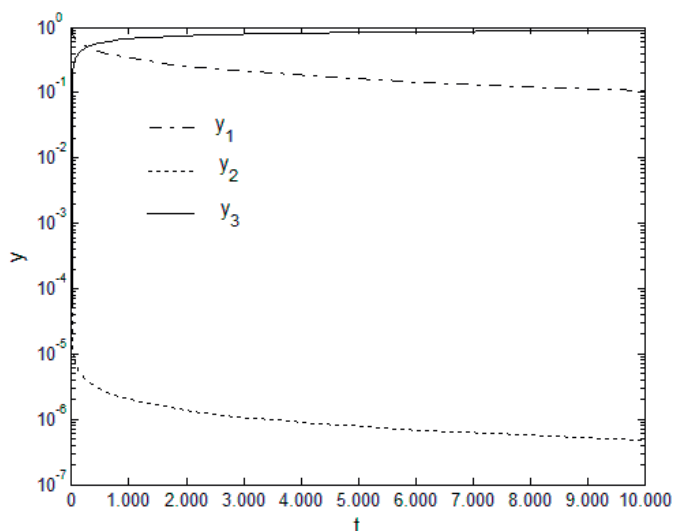
lepotik zehazki. (1.17) sistemako lehenengo erreakzioaren deneko abiadurari k_1 deituko diogu. Horrek esan nahi du, y_1 balioaren txikitze-abiadura eta y_2 balioaren handitze-abiadura, lehenengo erreakzio horren eraginez, $k_1 y_1$ izango dela. (1.17) adierazpeneko bigarren erreakzioan, C-k katalizatzaile moduan eragiten du, B-tik abiatuz A produktuaren sorreran. Erreakzio horren abiadura k_2 -ren bidez adieraziko dugu. Horrek esan nahi du y_1 balioaren handitzea eta y_3 balioaren txikitzea erreakzio horretan $k_2 y_2 y_3$ adierazpenak emana egongo dela. Azkenik, C-ren produkzioa B-tik abiatuz k_3 erritmoan egingo dela suposatuko dugu, hau da, hirugarren erreakzioa $k_3 y_2^2$ abiaduran emango da. Hau guztia bilduz, hiru kontzentrazioen denborarekiko aldaketa ematen duten ekuazio diferentzialetara iristen gara:

$$\begin{cases} \frac{dy_1}{dt} = -k_1 y_1 + k_2 y_2 y_3 \\ \frac{dy_2}{dt} = k_1 y_1 - k_2 y_2 y_3 - k_3 y_2^2 \\ \frac{dy_3}{dt} = k_3 y_2^2 \end{cases} \quad (1.18)$$

Robertson-ek 1966an bezala, balio hauek erabili ditugu (1.18) sistema askatzeko: $k_1 = 0,04$, $k_2 = 10^4$ eta $k_3 = 3 \cdot 10^7$. Eta guk hasierako balio hauek hartu ditugu: $y_1(0) = 1$, $y_2(0) = 10^{-4}$ eta $y_3(0) = 5 \cdot 10^{-5}$. 1.6. eta 1.7. irudietan marraztu ditugu lortu ditugun emaitzak.



1.6. irudia. (1.18) sistemaren soluzio-kurbak
(abszisen ardatza eskala logaritmikoan jarrita).



1.7. irudia. (1.18) sistemaren soluzio-kurbak
(ordenatuen ardatza eskala logaritmikoan jarrita).

1.3. HASIERAKO BALIODUN EDA BATEN SOLUZIO ANALITIKOA ETA ZENBAKIZKO SOLUZIOA

Hasierako baliodun ekuazio diferentzial arrunt baten soluzio analitikoa era analitikoan emandako formula bat da. Emaizta hori, normalean, funtzio elementalak erabiliz ematen da (polinomioak, funtzio esponentzialak, logaritmikoak, trigonometrikoak, etab.). Batzuetan emaitzako batugai kopurua finitua izaten da, eta, beste batzuetan, serie gisa emanda ere egon daiteke. Horrela, EDA baten soluzio analitikoa dugunean, posible da aldagai independenteari balioak emanez soluzioak edozein aldiunetan izango duen balioa kalkulatzeko.

Adibidea

Ondorengo ekuazio diferentzial arrunta emanik:

$$y' = y^2 \quad (1.19)$$

Haren soluzio analitikoa $y(t) = \frac{-1}{t - C_1}$ da, C_1 edozein konstante erreal izanik.

(1.19) ekuazioak infinitu emaitza ditu. Ekuazio diferentzial horrek emaitza bakarra izatea nahi baldin badugu, aski da, adibidez, (1.19) ekuazioan hasierako balio bat finkatzea, $y = y_0$ balioa finkatzea $t = t_0$ denean. Adibidez, $y(1) = 2$ balioa finkatzen

badugu, hau da, $y=2$ balioa finkatzen badugu $t=1$ denean, EDA horren emaitza analitikoa hauxe izango da:

$$y(t) = \frac{-2}{2t-3}$$

Hasierako baliodun ekuazio diferentzial arruntaren zenbakizko soluzio bat, aldiz, puntu diskretu batzuetako balio hurbilduen taula bat da. Balio hurbildu horiek ekuazio diferentziala eta zenbakizko metodo bat erabiliz pausoz pauso lortzen dira, eta mendeko aldagaiak aldagai independentearen balio batzuetan hartzen dituen balioen multzoa da. Hasierako baliodun (1.8) EDAren soluzio analitikoa $y(t)$ baldin bada, zenbakizko metodoek emaitza analitikoaren hurbilpenak emango dituzte $y(t_n) \approx y_n$ puntu batzuetan:

$$a = t_0 < t_1 < t_2 < \dots < t_N = b$$

Integrazioa, $y(a) = y_0$ balioarekin hasten da eta $t_1 = t_0 + h_0$ puntuan soluzioaren hurbilpen bat kalkulatzen da $y_1 \approx y(t_1)$. « h_i » kantitateari pauso-tamaina esaten zaio. Pauso bakarreko metodoek t_n puntuko balioa erabilita (y_n erabilita zein $y_n \approx y(t_n)$) kalkulatzen dute $t_{n+1} = t_n + h_n$ aldiuneko balioa (y_{n+1} balioa). Hau da, y_n erabilita kalkulatzen dute y_{n+1} balioa. Eta y_{n+1} kalkulatzea, t_n aldiunetik t_{n+1} aldiunera pauso bat ematea bezala deskribatzen da. Pauso-tamaina konstantea izan daiteke edo gerta daiteke aldiune bakoitzean pauso-tamaina ezberdina erabiltzea. Guk, testu honetan barrena, pauso-tamaina konstantea erabiliko dugu eta h deituko diogu.

Lan honetan ekuazio diferentzial arruntaren zenbakizko emaitzak aurkitzeko MATLAB® softwarea erabili da. MATLAB kalkulurako eta programaziorako ingurune interaktiboa da. MATLABek, besteak beste, algebra linealeko hainbat problema ebazteko tresnak ditu, ekuazio linealen erroak aurki ditzake, funtzioen integralak kalkula ditzake, polinomioekin lan egin dezake, ekuazio diferentzial arruntak eta algebraikoak askatzeko aukera eskaintzen du, etab. Gainera, erraz hedatu eta pertsonalizatu daiteke erabiltzaileak idatzitako funtzioak erabiliz. Instalatzeko erraza da eta instalatu eta jarraian era interaktibo batean esperimendatzeko has gaitezke. Geroago, posible da arau minimo batzuk kontuan hartuta programatzen hastea, eta ekuazio diferentzial arruntaren soluzioa aurkitzea ahalbidetzen duten hainbat baliabide ere baditu.

1.4. SOLUZIOAREN EXISTENTZIA ETA BAKARTASUNA

Ekuazio diferentzial baten soluzioa existitzen den jakiteko, eta existitzen den kasuan bakarria den jakiteko erabiltzen den irizpiderik arruntena Lipschitz-en baldintza da.

Definizioa. Ondorengo funtzioa emanik, $f:[a,b] \times \mathbb{R}^m \rightarrow \mathbb{R}^m$, funtzio honek bere bigarren aldagaian Lipschitz-en baldintza betetzen duela esaten da, baldin existitzen bada L konstante bat, Lipschitz-en konstante deritzona, ondorengo baldintza beteko duena:

$$\|f(t,Y) - f(t,Z)\| \leq L\|Y - Z\| \quad \forall t \in [a,b], Y, Z \in \mathbb{R}^m$$

Teorema. (1.8) adierazpenaren bidez emana datorren hasierako baliodun ekuazio diferentzial arrunta emanik, $f:[a,b] \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ funtzioa bere lehenengo aldagaian jarraitua izanik eta bere bigarren aldagaian Lipschitz-en baldintza betetzen duena, orduan, problemarentzat soluzio bakarria existitzen da.

Frogapena. Hainbat liburutan aurki daiteke teorema honen frogapena, adibidez: Butcher, 2008: 23 orrialdean.

1.5. EDA BATEN ZENBAKIZKO KALKULUA: EULER-EN METODOAK

Hasierako baliodun EDA bat (edo EDA sistema bat) emanda, zenbakizko metodo bat pauso bakarrekoa dela esaten da baldin zenbakizko soluzioa kalkulatzeko ekuazio diferentziala eta aurreko pausoko balioa erabiltzen badira. Pauso bakarrekoko zenbakizko metodo ezagunenetakoak bi hauek dira: Euler-en metodo esplizitua (Euler-en metodoa ere izendatutakoa) eta Euler-en metodo implizitua (atzerakako Euler-en metodo izenaz ezagutzen dena).

1.5.1. Euler-en metodo esplizitua

(1.8) adierazpenaren bidez emana datorren hasierako baliodun ekuazio diferentzial arruntaren kasuan, t_0 aldiunean $y(t_0)$ eta $y'(t_0)$ balioak ezagutzen ditugu. Taylor-en seriea erabilita, $t_1 = t_0 + h$ aldiunean funtzioak hartzen duen balioa kalkulatzeko dezakegu:

$$y(t_0 + h) = y(t_0) + hf(t_0, y(t_0)) + O(h^2)$$

$f(t_0, y(t_0)) = y'(t_0)$ izanik.

Horrela, pauso-tamaina txiki baterako, $t_1 = t_0 + h$ aldiuneko funtzioaren balioa, $y(t_1)$, ondorengo y_1 balioaren bidez hurbildu daiteke:

$$y_1 = y(t_0) + hf(t_0, y(t_0))$$

Aurreko pausoan lortu dugun t_1 puntuko hurbilpena erabilia, hau da, y_1 erabilia, lehengo prozesua errepika daiteke $y(t_0 + 2h)$ balioaren hurbilpen bat kalkulatzeko. Hau da, y_2 balioa kalkulatu dugu ondorengo formularen bidez:

$$y_2 = y_1 + hf(t_1, y_1)$$

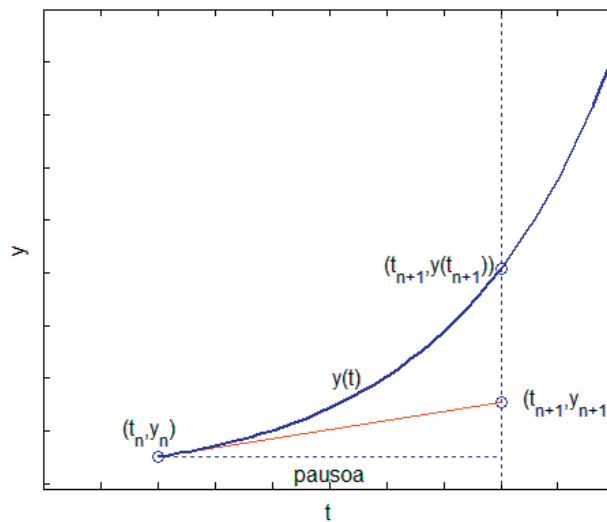
Horrela, pausoz pauso $y(t_i)$ emaitza analitikoaren hurbilpenak diren y_i balioen sekuentzia bat lortuko da, non:

$$\begin{cases} t_i = t_0 + ih, \\ y_{i+1} = y_i + hf(t_i, y_i), \quad i = 0, 1, \dots \end{cases} \quad (1.20)$$

Eta Euler-en metodo esplizitua honela emana dator:

$$y_{n+1} = y_n + hf(t_n, y_n) \quad (1.21)$$

(1.20) adierazpeneko pauso-tamaina guztiak dira berdinak, hau da, pauso-tamaina konstantea da. Baina ez da beharrezkoa pauso guztiak berdinak izatea. Lehen ere esan dugu, testu honetan eta egiten ari garen azterketa sinplifikatzeko asmoz, pauso-tamaina konstantea dela suposatuko dugula eta h letraz izendatuko dugula.



1.8. irudia. Euler-en metodo esplizituko pauso bat.

Euler-en metodo esplizitua erabilia (t_n, y_n) puntutik abiatuta pauso bat ematen da, (t_n, y_n) puntuko zuzen ukitzearen malda erabilia. 1.8. irudian ikus daiteke (t_n, y_n) puntuko zuzen ukitzearen malda erabilia nola ematen den pauso bat metodo honetan eta nola (t_{n+1}, y_{n+1}) puntua lortzen den. Emaiza zehatza edo analitikoa $y(t)$ -z adierazi da eta $(t_{n+1}, y(t_{n+1}))$ litzateke ekuazio diferentzialaren emaitza zehatza t_{n+1} aldiunean.

MATLAB erabilia honela idatziko genuke Euler-en metodo espliziturako kodea:

MATLABek ez dauka implementatuta ekuazio diferentzial arruntak askatzeko Euler-en metodo esplizituaren koderik, baina geuk sor dezakegu ondorengo pausoei jarraituz:

1. pausoa: Euler-en metodo esplizitua aplikatu nahi diogun ekuazio diferentziala sortuko dugu.
2. pausoa: Euler-en metodo esplizitua sortuko dugu. Funtzio honen sarrerako argumentuak izango dira Euler-en metodoa aplikatu nahi diogun ekuazio diferentziala, emaitza kalkulatu dugu denbora-tartea, ekuazio diferentzialaren hasierako balioa, eta metodoa erabiliz emango dugun pauso kopurua.
3. pausoa: Euler-en metodo esplizituaren funtzioak ekuazio diferentzial arruntaren emaitza kalkulatu deneko aldiuneak eta beraietan funtzioak duen balioa bueltatuko dizkigu.

MATLABeko kodea:

Integratuko dugun ekuazio diferentziala definitzeko ondorengo funtzioa sor daiteke. Adibidez, hauxe litzateke lehen adibideko untxi eta otsoen problemari dagokion ekuazio diferentzialen sistema definitzeko funtzioa:

```
function fty=untxi_otso(t,y)
fty(1)=0.05*y(1)*(1-0.01*y(2));
fty(2)=0.1*y(2)*(0.005*y(1)-2);
fty=fty';
```

Ondoren Euler-en metodo esplizituari dagokion funtzioa sortuko dugu:

```
function [tsol,ysol]=euler(fty,tartea,y0,m)
```

Funtzio hori s ekuazio diferentzial arruntetako $y' = f(t, y)$, $y(t_0) = y_0$ hasierako baliodun sistema askatzen duen funtzioa da. Sarrerako datuak hauek izango dira:

- $f(t, y)$: ekuazio diferentzialeko 2. parte definitzen den funtzioaren izena jarri behar da hemen. Adibidez, dimentsio bakarreko EDA hau $y' = -y^2$ daukagun kasuan, MATLABen funtzio hau sortuko genuke:

```
function yprima=p1(t,y)

%Bat dimentsiodun problema:
yprima=-y.^2;
```

Bi dimentsioko EDA sistema hau den kasuan $y_1' = -y_1^2$, $y_2' = -y_2$, MATLABen definituko genukeen funtzioa hau litzateke:

```
function yprima=p2(t,y)

%Bi dimentsiodun problema:
yprima(1)=-y(1)^2;
yprima(2)=-y(2);
```

- t_0 : EDA askatuko dugun denbora-tartea da: $[t_0, t_1]$.
- y_0 : s dimentsioko EDA sistema izanik, hasierako balioa jasotzen den errenkada matrizea da: $[y_1, y_2, \dots, y_s]$. Dimentsio bakarreko EDAREN kasuan balio bakarra izango genuke.
- m : metodoa erabilia emango dugun pauso kopurua da. Beste era batean esanda: $[t_0, t_1]$ denbora-tartea zenbat zatitan zatituko dugun esaten digun zenbakia da.

Irteerako datuak dira:

- $tsol$: $(m+1)$ dimentsioko zutabe matrizea. Bertan $[t_0, t_1]$ tartea diskretizatu dugun t puntuak daude jasota.
- $ysol$: $(m+1) \times s$ dimentsioko matrizea da. Hau da, $(m+1)$ errenkada eta s zutabe dituen matrizea. Errenkada bakoitzean $tsol(i)$ aldiuneko soluzioaren s osagaiak egongo dira $i = 1, \dots, (m+1)$ izanik. Zutabe bakoitzean soluzioaren j osagaiak $(m+1)$ aldiuneetan hartzen dituen balioak egongo dira jasota.

Funtzio nagusia «euler» izango da eta beronek bi funtzio osagarri egingo die: «hasi» eta «paseu» izeneko funtzioei, hain zuzen. Lehenengoak, «hasi»k, integrazioa hasteko beharko dituen parametroak itzuliko dizkigu. Hala nola h pauso-tamaina itzuliko digu eta $tsol$ eta $ysol$ matrizeei dimentsio egokia emango die. Hasiera batean zeroz beteko ditu matrize hauek eta beraien lehen errenkadan badakizkigun hasierako balioak beteta itzuliko dizkigu matrizeok.

«paseu» izeneko funtzioan Euler-en metodoko pausoa definituko dugu eta pauso bakoitzean funtzio horri deitu beharko dio funtzio nagusiak.

Jarraian, «euler» funtzio nagusia eta «hasi» eta «paseu» funtzioen programak txertatu ditugu:

▪ «euler» funtzioa:

```
function [tsol,ysol]=euler(fty,tartea,y0,m)

[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:)' ;

for j1=2:m+1

    [t,y]=paseu(fty,t,y,h);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.
end
```

▪ «hasi» funtzioa:

```
function [h,tsol,ysol]=hasi(tartea,y0,m)
%
% Ekuazio diferentzial arrunten sistema bat integratzeko hasierako
% funtzioa:

t0=tartea(1);
t1=tartea(2);
h=(t1-t0)/m;      % Pausoaren tamaina.
s=length(y0);     % y0 bektorearen dimentsioa.
```

```
% tsol zutabe matrizea zeroekin betetzen da hasieran.
tsol=zeros(m+1,1);
% ysol izeneko soluzio matrizea zeroekin betetzen da hasieran
ysol=zeros(m+1,s);
tsol(1)=t0;
% ysol-eko 1. errenkadan sartzen da hasierako y0 baldintza.
ysol(1,:)=y0';
```

- «paso» funtzioa:

```
function [tn1,yn1] = pasoeu(fty,tn,yn,h)

%
% "s" ekuazio diferentzial arrunteko  $y'=f(t,y)$  sistema askatzeko
% Euler-en metodoko pauso.
% tn aldiunetik tn1=tn+h aldiunerako pauso eman nahi da.
%

%(tn,yn) puntuko maldaren kalkulua:

malda = feval(fty,tn,yn);    % malda = f(tn,yn).

%Euler-en pauso: yn1=yn+h*f(tn,yn)

yn1=yn+malda*h;

%Denbora eguneratzen dugu:

tn1 = tn+h;
```

Behin metodoa eta funtzioa sortuta ditugunean, adibidez untxi eta otsoen problemaren soluzioa $T=[0,600]$ denbora-tartean 100.000 pauso emanda kalkulatu nahi badugu, hasierako balioak $(y_1(0), y_2(0)) = (1.500, 100)$ izanik, MATLABen ondorengo agindua idatzi baino ez dugu:

```
[tsol,ysol]=euler('untxi_otso',[0 600],[1500 100]',100000)
```

Eta MATLABek $T=[0,600]$ denbora-tarteko lehenengo aldiuneari, hau da, $t=0$ aldiuneari, eta distantzia berera dauden beste 100.000 aldiuneri dagozkien soluzioak itzuliko dizkigu.

1.5.2. Euler-en metodo inplizitua

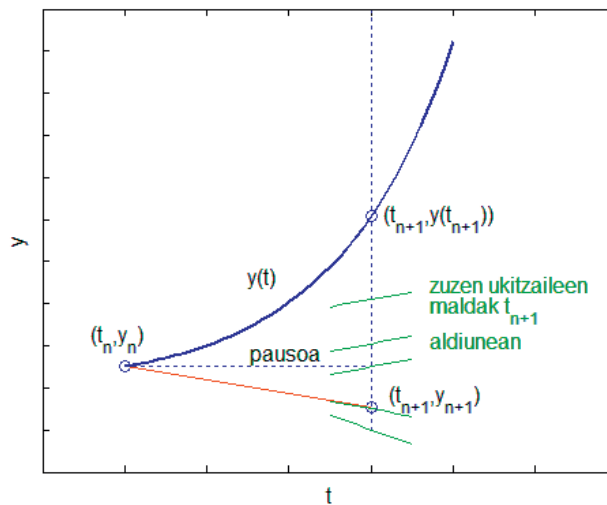
Euler-en metodo inplizituak ahalbidetzen digu (1.8) adierazpenaren bidez emana datorren hasierako baliodun ekuazio diferentzial arruntaren zenbakizko emaitza t_{n+1} aldiunean kalkulatzeko, eta horretarako ondorengo adierazpena erabiltzen da:

$$y_{n+1} = y_n + hf_{n+1} \quad (1.22)$$

y_n ekuazio diferentzial arruntaren t_n aldiuneko emaitza analitikoaren balio hurbildua izanik eta $f_{n+1} = f(t_{n+1}, y_{n+1})$ izanik.

Zenbakizko metodo bat esplizitua da baldin y_{n+1} balioa kalkulatzeko beharrezkoak diren balio guztiak ezagutzen badira, bestela inplizitua da.

Euler-en metodo inplizitua erabilita (t_n, y_n) puntutik abiatuta pauso bat ematen da. Pausoa, t_{n+1} aldiuneko zuzen ukiztailea (t_n, y_n) puntutik pasatzen den y_{n+1} puntua aurkitzean datza. 1.9. irudian ikus daiteke (t_{n+1}, y_{n+1}) puntuko zuzen ukiztailea (t_n, y_n) puntutik pasatzen dela, beraz, (t_{n+1}, y_{n+1}) puntua da Euler-en metodo inplizituak emango digun puntua. Emaitza zehatza edo analitikoa $y(t)$ adierazpenaz adierazi da eta $(t_{n+1}, y(t_{n+1}))$ litzateke ekuazio diferentzialaren emaitza zehatza t_{n+1} aldiunean. Era berean, t_{n+1} aldiuneko zenbait zuzen ukiztaile ere irudikatu dira.



1.9. irudia. Euler-en metodo inplizituko pauso bat.

MATLABen sortuko dugun Euler-en metodo inpliziturako baliagarria den metodo bat ikusiko dugu ondoren, Newton-Raphson-en metodoa, alegia.

1.5.2.1. Newton-Raphson-en metodoa

Newton-Raphson-en metodoak $f(x)=0$ motako ekuazio bat askatzeko balio du, hasierako $x^{(0)}$ balio bat emanda. Beste era batean esanda, $x^{(0)}$ balio batetik abiatuz $f(x)$ funtzioaren erroak kalkulatzeko balio duen metodoa da Newton-Raphson-en metodoa. Lehenik eta behin, $f(x^{(0)})$ balioa kalkulatu da eta $(x^{(0)}, f(x^{(0)}))$ puntutik $f(x)$ funtzioarekiko zuzen ukitzailea marrazten da eta zuzen horrek OX ardatza mozten duen puntua kalkulatu da. $f(x)$ funtzioa lineala balitz, zuzen ukitzaileak OX ardatza mozten dueneko puntua litzateke $f(x)$ funtzioaren erroa. Oro har, lortzen den puntua $x^{(1)}$ izendatu duguna, $f(x)$ funtzioaren erroaren hurbilpen bat da eta hasieran izan dugun $x^{(0)}$ puntua baino hobeago gainera. Puntu berri hori lortzeko analitikoki haxe egin behar da: $(x^{(0)}, f(x^{(0)}))$ puntutik pasatuz $f(x)$ funtzioarekiko zuzen ukitzailearen ekuazioa idatzi:

$$y - f(x^{(0)}) = f'(x^{(0)})(x - x^{(0)}) \quad (1.23)$$

(1.23) zuzenak OX ardatza mozten duen puntua kalkulatzeko $y=0$ egiten dugu ekuazio honetan:

$$0 - f(x^{(0)}) = f'(x^{(0)})(x - x^{(0)}) \Rightarrow x = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})}, \quad f'(x^{(0)}) \neq 0 \quad (1.24)$$

(1.24) adierazpenean lortu dugun x -en balioari $x^{(1)}$ deituko diogu eta aurretik $x^{(0)}$ puntuarekin egin dugun prozesua egingo dugu harekin ere. Hau da, $(x^{(1)}, f(x^{(1)}))$ puntutik $f(x)$ funtzioarekiko zuzen ukitzailea marrazten da eta zuzen horrek OX ardatza mozten duen puntua kalkulatu da, $x^{(2)}$. Oro har, adierazpen hau izango dute $x^{(v+1)}$ puntu horiek:

$$x^{(v+1)} = x^{(v)} - \Delta^{(v)}, \quad v = 0, 1, 2, \dots \quad (1.25)$$

$$\Delta^{(v)} = \frac{f(x^{(v)})}{f'(x^{(v)})} \text{ izanik.}$$

Newton Raphson-en prozesua bukatzen da, hasieran zehaztu den iterazio kopurua bete denean edo/eta $\Delta^{(v)}$ terminoa hasieran definitu den tolerantzia bat baino txikiagoa denean, hau da: $|\Delta^{(v)}| \leq tol$

Goiko emaitza orokortu egin daiteke k ezezaguneko k ekuazio algebrakoa izanik, $R(y) = (0, 0, \dots, 0)$ askatu behar duguneko kasura, $R = (R_1, R_2, \dots, R_k)$ izanik

eta $y = (y_1, y_2, \dots, y_k)$. Goian azaldutakoak berdin jarraitzen du, baina kasu honetan R funtzioaren jacobitarra kalkulatu behar da, $k \times k$ dimentsioko $\partial R / \partial y$ matrizea:

$$\frac{\partial R}{\partial y} = \begin{pmatrix} \frac{\partial R_1}{\partial y_1} & \frac{\partial R_1}{\partial y_2} & \dots & \frac{\partial R_1}{\partial y_k} \\ \frac{\partial R_2}{\partial y_1} & \frac{\partial R_2}{\partial y_2} & \dots & \frac{\partial R_2}{\partial y_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial R_k}{\partial y_1} & \frac{\partial R_k}{\partial y_2} & \dots & \frac{\partial R_k}{\partial y_k} \end{pmatrix}$$

Funtzio bektorialen kasuan, (1.25) adierazpenak forma hau hartzen du:

$$y^{(v+1)} = y^{(v)} - \Delta^{(v)}, \quad v = 0, 1, 2, \dots \quad (1.26)$$

$\Delta^{(v)} = \left(\frac{\partial R}{\partial y}(y^{(v)}) \right)^{-1} R(y^{(v)})$ bektorea izanik.

Adibidea

$f(x) = x^2 - 2$ funtzioaren erro bat aurkitu nahi dugu $x^{(0)} = 3$ puntuan hasita. Badakigu $x^{(0)} = 3$ puntutik hurbil dagoen $f(x)$ funtzioaren erroa $\sqrt{2}$ dela. Newton Raphson-en metodoa erabilita 4 iterazio egin ditugu eta emaitza hauek lortu ditugu:

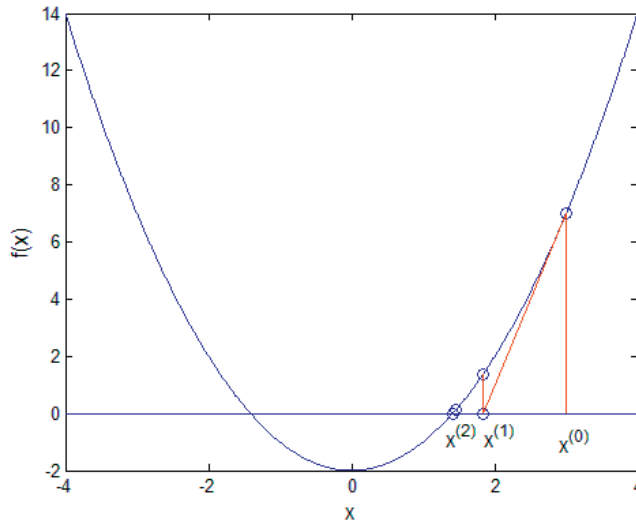
$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})} = 3 - \frac{7}{6} = \frac{11}{6} \approx 1,8333$$

$$x^{(2)} = x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})} = \frac{11}{6} - \frac{49/36}{11/3} = \frac{193}{132} \approx 1,4621$$

$$x^{(3)} = x^{(2)} - \frac{f(x^{(2)})}{f'(x^{(2)})} \approx 1,4150$$

$$x^{(4)} = x^{(3)} - \frac{f(x^{(3)})}{f'(x^{(3)})} \approx 1,4142$$

Eta lortu dugun erroaren hurbilpena $x^{(4)}$, $f(x)$ funtzioaren erroak oso hurbil dago. 1.10. irudian jaso ditugu eman ditugun iterazioak.



1.10. irudia. $f(x) = x^2 - 2$ funtzioaren erro baten kalkulua
Newton-Raphson-en metodoa erabilia.

1.5.2.2. Euler-en metodo inplizituaren inplementazioa MATLABen

Euler inplizituaren koderia itzuliz, (1.22) ekuazioa inplizitua denez, metodoko pauso bakoitzean, Newton Raphson-en metodoa aplikatuko dugu eta y_{n+1} ezezaguna $(\nu + 1)$. iterazioan honela kalkulatu dugu:

$$y_{n+1}^{(\nu+1)} = y_n^{(\nu)} - [J(y_{n+1}^{(\nu)})]^{-1} R(y_{n+1}^{(\nu)}), \quad \nu = 0, 1, 2, \dots \quad (1.27)$$

R eta J hauek dira Euler-en metodo inplizituaren kasuan:

$$\begin{cases} R(y_{n+1}^{(\nu)}) = (y_{n+1}^{(\nu)} - y_n) - hf_{n+1}^{(\nu)} \\ J(y_{n+1}^{(\nu)}) = \frac{\partial R(y_{n+1}^{(\nu)})}{\partial y_{n+1}^{(\nu)}} = I - h \frac{\partial f_{n+1}^{(\nu)}}{\partial y_{n+1}^{(\nu)}} \end{cases} \quad (1.28)$$

(1.27) eta (1.28) formuletak (ν) goi-indizeak iterazioa adierazten du.

$\| [J(y_{n+1}^{(\nu)})]^{-1} R(y_{n+1}^{(\nu)}) \| \leq tol \approx 0$ baldintza betetzen den arte edo zehaztutako iterazio maximora iritsi arte (1.27) prozesua errepikatuko dugu. Hasierako iragarletzat, aurreko pausoko balioa erabiliko dugu. Hau da: $y_{n+1}^{(0)} = y_n$.

MATLAB erabilia honela idatziko genuke Euler-en metodo inpliziturako kodea:

Euler-en metodo inpliziturako funtzio hau sor daiteke MATLABen:

```
function [tsol,ysol]=eulerinp(fty,dfty,tartea,y0,m)
```

Funtzio horrek s ekuazio diferentzial arrunteko $y' = f(t, y)$, $y(t_0) = y_0$ hasierako baliodun sistema askatzeko balioko du. Aurretik Euler-en metodo espliziturako aipatu ditugun sarrerako datuez gain, beste sarrerako datu bat izango du:

- $dfty$: ekuazio diferentzialeko 2. partearen, hau da $f(t, y)$ funtzioaren y aldagaiarekiko deribatua ($\partial f / \partial y$ hain zuzen) definitzen den funtzioaren izena jarri behar da hemen. Adibidez, hauxe litzateke untxi eta otsoen problemari dagokion ekuazio diferentzialeko $f(t, y)$ funtzioaren y aldagaiarekiko deribatua definitzeko MATLABen sortuko genukeen funtzioa:

```
function dyprima=duntxi_otso(t,y)
dyprima(1,1)=0.05*(1-0.01*y(2));%df1/dy1
dyprima(1,2)=0.05*y(1)*(-0.01);%df1/dy2
dyprima(2,1)=0.1*y(2)*0.005;%df2/dy1
dyprima(2,2)=0.1*(0.005*y(1)-2);%df2/dy2
```

Irteerako datuak $tsol$ eta $ysol$ izango dira, lehengo moduan. Funtzio nagusia «eulerinp» izango da eta horrek ere bi funtzio osagarri deituko die: «hasi» eta «pasoeuin» izeneko funtzioei, hain zuzen. Lehenengoa, «hasi» Euler-en metodo espliziturako erabili dugun bera izango da, eta «pasoeuin» izeneko funtzioan Euler-en metodo inplizituko pausoa definituko dugu.

Jarraian, «eulerinp» funtzio nagusiaren eta «pasoeuin» funtzioaren programak jarri ditugu:

- «eulerinp» funtzioa:

```
function [tsol,ysol]=eulerinp(fty,dfty,tartea,y0,m)

s=length(y0);
I=eye(s);           %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:);
```

```

%%Pausoak Euler implizitua metodoa erabilita:
for j1=2:m+1

    [t,y]=pasoeuinp(fty,dfty,t,y,h);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.

end

```

▪ «pasoeuinp» funtzioa:

```

function [tn1,yn1] = pasoeuinp(fty,dfty,tn,yn,h)
% "s" ekuazio diferentzial arrunteko  $y'=f(t,y)$  sistema askatzeko
%Euler implizituko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:
tn1 = tn+h;

s=length(yn);
I=eye(s);

%Newton Raphson metodorako tolerantzia:
tol=0.001;

yn1 = yn;
dyn1=1; %dyn1-en tol baino handiago den balio bat jartzen dugu,
        %buklean sar dadin.
maxiter=0;
while (abs(dyn1)>tol & maxiter<100)
    fn1=feval(fty,tn+h,yn1);
    j=feval(dfty,tn+h,yn1);
    R=yn1-yn-h*fn1;
    %%Newton Raphson-en metodoko jacobitarra:
    J=I-h*j;
    [L,U] = lu ( J );
    dyn1 = -U\ (L\R);
    yn1 = yn1+dyn1;
    maxiter=maxiter+1;
end

```

Berriro ere MATLABen ondorengo agindua idatzita, untxi eta otsoen problemaren soluzioa aurkituko genuke:

```
[tsol,ysol]=eulerinp('untxi_otso','duntxi_otso',[0 600],[1500 100]',100000)
```

Adibideak

1. Aplikatu Euler-en metodo esplizitua ondorengo ekuazio diferentzial arruntari, $h = 0,5$ pauso-tamaina konstantea erabilita:

$$y' = y, \quad y(0) = 1 \quad (1.29)$$

Emaizta analitikoa berehalakoa da: $y = e^t$. Zenbakizko emaitza aurkitzeko $t = 0$ aldiunetik $t = 4$ aldiunera abiatuko gara. Hau da, $0 = t_0 < t_1 < t_2 < \dots < t_N = 4$ aldiuneetako balio hurbilduak kalkulatu ditugu. Erabiliko dugun pauso-tamainarekin 8 pauso eman beharko ditugu balio horiek kalkulatzeko. Hau da, $N = 8$ izango da. Euler-en metodo esplizitua erabiliz lehenengo pausoa hau da:

$$y_1 = y_0 + h y_0' = 1 + 0,5 \cdot 1 = 1,5 \approx y(t_1)$$

Lehen pausoa lortutako zenbakizko emaitza eta emaitza analitikoa ez datozela bat antzeman daiteke:

$$y(t_1) = y(0,5) = e^{0,5} \approx 1,6487$$

Beraz, lehenengo pausoa egindako errorea hau da: $y(t_1) - y_1 = 0,1487$. Bigarren pausoa emango dugu aurreko prozedura bera aplikatuz:

$$y_2 = y_1 + h y_1' = 1,5 + 0,5 \cdot 1,5 = 2,25$$

Bigarren pausoa ere lortutako zenbakizko emaitza eta emaitza analitikoa ez datoz bat:

$$y(t_2) = y(1) = e \approx 2,7183$$

Bigarren pausoko errorea beste hau izango da: $y(t_2) - y_2 = 0,4683$. $t = 4$ aldiunera pausoak emanez jarraitzen badugu, 1.1. taulan jasotako emaitzak lortuko dira.

t_k	y_k	$y(t_k)$	$y(t_k) - y_k$
0	1	1	0
0,5	1,5	1,6487	0,1487
1	2,25	2,7183	0,4683
1,5	3,375	4,4817	1,1067
2	5,0625	7,3891	2,3266
2,5	7,5938	12,1825	4,5887
3	11,3906	20,0855	8,6949
3,5	17,0859	33,1155	16,0295
4	25,6289	54,5982	28,9692

1.1. taula. (1.29) ekuazio diferentzial arruntaren emaitza analitikoa eta zenbakizko emaitza Euler-en metodo esplizitua erabilita.

2. Euler-en metodo esplizitua erabiliz, eman $h = 0,1$ pauso-tamainako lehenengo 2 pausoak ondoko EDA sisteman:

$$\begin{cases} y_1' = -y_1 + 4y_2 \\ y_2' = -4y_1 - y_2 \end{cases} \quad (1.30)$$

Hasierako balioa hauxe izanik: $\vec{y}_0 = (y_1(0), y_2(0)) = (2, -1)$.

(1.30) sistema era honetan idatz daiteke:

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} -1 & 4 \\ -4 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

Esku artean daukagun problema ekuazio diferentzial arrunten sistema bat da. Emaita lortzeko, Euler-en metodoa aplikatu behar diogu aldagai bakoitzari, $\vec{y}_0$ bektoretik abiatuz. Emaita hauek lortzen dira:

$$\begin{cases} \vec{y}_0 = \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \\ \vec{y}_1 = \vec{y}_0 + h \cdot f(t_0, \vec{y}_0) = \begin{pmatrix} 2 \\ -1 \end{pmatrix} + 0,1 \cdot \begin{pmatrix} -1 & 4 \\ -4 & -1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 1,4 \\ -1,7 \end{pmatrix} \\ \vec{y}_2 = \vec{y}_1 + h \cdot f(t_1, \vec{y}_1) = \begin{pmatrix} 1,4 \\ -1,7 \end{pmatrix} + 0,1 \cdot \begin{pmatrix} -1 & 4 \\ -4 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1,4 \\ -1,7 \end{pmatrix} = \begin{pmatrix} 0,58 \\ -2,09 \end{pmatrix} \end{cases}$$

3. Euler-en metodo inplizitua eta $h = 0,5$ pauso-tamaina erabilita, ebatzi ondorengo ekuazio diferentzial arrunta:

$$y' = -y, \quad y(0) = 1 \quad (1.31)$$

Ekuazio diferentzial horren emaitza zehatza edo analitikoa da $y = e^{-t}$. Euler-en metodo inplizitua erabiliz lehenengo pausoa honela geratzen zaigu:

$$y_1 = y_0 + hf(t_1, y_1) = 1 - 0,5y_1$$

Beraz, hauxe da ebatzi behar dugun ekuazioa, y_1 lortu nahi badugu:

$$y_1 = 1 - 0,5y_1 \Rightarrow 0,5y_1 + y_1 = 1 \Rightarrow y_1 = \frac{1}{1,5} = \frac{2}{3} = 0,6667$$

Era berean, bigarren pausoko y_2 balioa lor dezakegu:

$$y_2 = \frac{2}{3} - 0,5y_2 \Rightarrow 0,5y_2 + y_2 = \frac{2}{3} \Rightarrow y_2 = \frac{2/3}{1,5} = \frac{4}{9} = 0,4444$$

$t = 4$ aldiuneraingo pausoak emanez jarraituz gero, 1.2. taulan jasotako emaitzak lortuko dira. Emaitzak lau hamartar erabilita eman dira:

t_k	y_k	$y(t_k)$	$y(t_k) - y_k$
0	1	1	0
0,5	0,6667	0,6065	-0,0601
1	0,4444	0,3679	-0,0766
1,5	0,2963	0,2231	-0,0731
2	0,1975	0,1353	-0,0622
2,5	0,1317	0,0821	-0,0496
3	0,0878	0,0498	-0,0380
3,5	0,0585	0,0302	-0,0283
4	0,0390	0,0183	-0,0207

1.2. taula. (1.31) ekuazio diferentzial arruntaren emaitza analitikoa eta zenbakizko emaitza Euler-en metodo implizitua erabilita.

1.6. METODO TRAPEZOIDALA

Beste metodo implizitu ezagun bat metodo trapezoidala, da eta honela dator emana:

$$y_{n+1} = y_n + \frac{1}{2}h(f_n + f_{n+1}) \quad (1.32)$$

Adibidea

(1.31) ekuazio diferentzial arrunta metodo trapezoidala erabiliz askatu dugu $h = 0,5$ pauso-tamainarekin. 1.3. taulan jaso ditugu lortu diren emaitzak.

t_k	y_k	$y(t_k)$	$y(t_k) - y_k$
0	1	1	0
0,5	0,6000	0,6065	0,0065
1	0,3600	0,3679	0,0079
1,5	0,2160	0,2231	0,0071
2	0,1296	0,1353	0,0057
2,5	0,0778	0,0821	0,0043
3	0,0467	0,0498	0,0031
3,5	0,0280	0,0302	0,0022
4	0,0168	0,0183	0,0015

1.3. taula. (1.31) ekuazio diferentzial arruntaren emaitza analitikoa eta zenbakizko emaitza metodo trapezoidala erabilita.

1.7. ETAPA ANITZEKO METODOAK ETA PAUSO ANITZEKO METODOAK

Pauso bakarreko metodoen bariante bat pauso bakarrekoak izanik etapa anitz dituzten metodoek osatzen dute. Metodo hauek etapa anitzeko metodo edo s etapako metodo izenez ere ezagutzen dira. Pauso bakarreko metodoak dira eta pauso bakoitzean deribatuak tarte horretako hainbat puntutan hartzen dituen balioak erabiltzen ditu pausoaren muturrean zenbakizko soluzioa zein den kalkulatzeko. 3. kapituluan ikusiko ditugun Runge-Kutta metodoak hauetakoak dira.

Pauso bakarreko metodoei egindako beste bariante bat pauso anitzeko edo k pausoko metodoek osatzen dute. Metodo horietan aurreko k pausoetan lortu diren zenbakizko balioak erabiltzen dira kalkulatu nahi den puntuko balioa kalkulatzeko. Hau da, aurreko k pausoetako $t_n, t_n + h, \dots, t_n + (k-1)h$ balioetan lortu diren $y_n, y_{n+1}, \dots, y_{n+k-1}$ balioak edota berauen deribatuak $f_n, f_{n+1}, \dots, f_{n+k-1}$ erabiltzen dituzte $t_n + kh$ puntuko y_{n+k} balioa lortzeko. Backward Differentiation Formulae (BDF) izeneko metodoak pauso anitzeko metodoak dira eta Gear-en liburua ezkerro (Gear, 1971), metodo hauetan oinarritutako ordenagailuko kodeak oso erabiliak dira. Edonola ere, aurrerago ikusiko dugun bezala ordenan aurrera egin ahala egonkortasun-eremu ez oso ona duten metodoak dira eta arrazoi horregatik da, hain zuzen, egonkortasun hobearen metodoak sortzen ahalegin berezia egin izana.

1.7.1. Lehen ordenako EDAk askatzeko pauso anitzeko metodo linealak

Lehen ordenako hasierako baliodun EDA bat emanik (1.8), harentzat soluzio baten bila gabiltza $a \leq t \leq b$ tartean, a eta b finituak izanik. Suposatuko dugu f funtzioak Lipschitz-en baldintzak betetzen dituela, horrela, esku artean dugun ekuazio diferentzialak soluzioa duela eta berau bakarria dela ziurtatuko dugu, $y(t)$ deituko diogun soluzioa. Ondorengo t_n puntu-sekuentzia kontsideratuko dugu: $t_n = a + nh$, non $n = 0, 1, 2, \dots$ eta h pauso-tamaina diren. Izan bedi y_n balioa, t_n puntuan soluzio analitikoaren edo zehatzaren hurbilpen bat, hau da, $y(t_n)$ balioaren hurbilpen bat eta $f_n = f(t_n, y_n)$. y_n balioa lortzea ahalbidetzen duen zenbakizko metodo batek y_{n+j} eta f_{n+j} $j = 0, 1, \dots, k$ balioen arteko erlazio lineal bat betetzen duenean, pauso anitzeko metodo lineal bat dela esaten da. Berau era honetan idatz daiteke:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \sum_{j=0}^k \beta_j f_{n+j} \quad (1.33)$$

α_j, β_j konstanteak izanik, $\alpha_k \neq 0$ eta α_0 eta β_0 batera nuluak ez izanik. Metodoa esplizitua dela esaten da $\beta_k = 0$ denean, eta implizitua dela esaten da $\beta_k \neq 0$ den kasuan. Metodoa pauso bakarrekoa da $k = 1$ betetzen denean.

Adibideak

Orain arte ikusi ditugun Euler-en metodo esplizitua, implizitua eta metodo trapezoidala pauso bakarreko metodo linealak dira. Hiru kasuetan, zenbakizko metodoak (1.33) formari erantzuten dio eta pauso bakarrekoak dira $k=1$ betetzen baita. Hau da, honela idatz daitezke:

$$\sum_{j=0}^1 \alpha_j y_{n+j} = h \sum_{j=0}^1 \beta_j f_{n+j} \quad (1.34)$$

- Euler-en metodo esplizituaren kasuan α_i eta β_i konstanteen balioak hauek dira: $\alpha_1 = 1, \alpha_0 = -1, \beta_1 = 0, \beta_0 = 1$
- Euler-en metodo implizituaren kasuan: $\alpha_1 = 1, \alpha_0 = -1, \beta_1 = 1, \beta_0 = 0$ izanik.
- Eta metodo trapezoidalaren kasuan: $\alpha_1 = 1, \alpha_0 = -1, \beta_1 = 1/2, \beta_0 = 1/2$ izanik.

1.8. ZENBAKIZKO METODO BATEN ERROREA ETA ORDENA

Zenbakizko metodo baten mozketaren errorea globala balio zehatzaren eta zenbakizko metodoa erabiliz kalkulatu den balioaren arteko diferentzia da. GTE letrak erabilita izendatuko dugu (ingelesez, *Global Truncation Error*):

$$GTE = y(t_{n+k}) - y_{n+k} \quad (1.35)$$

Kokapen-bereganaketa esaten zaio kokatuta gauden pausoaren, adibidez $(n+k)$ pausoaren, aurretiko k pausoetan zenbakizko metodoa erabiliz kalkulatu ditugun balioak balio zehatzaren berdina direla kontsideratzeari. Hau da:

$$y_{n+j} = y(t_{n+j}), \quad j = 0, 1, \dots, k-1$$

Kokapen-bereganaketa eginda lortzen den balioari y_{n+k}^* deituko diogu.

Zenbakizko metodo baten mozketaren errorea lokala, balio zehatzaren eta y_{n+k}^* balioaren arteko diferentzia da. Azken balio hori kokapen-bereganaketa eginda kalkulatu den balioa da. LTE deituko diogu (*Local Truncation Error* ingelesez) eta ondorengo formularen bidez emana dator:

$$LTE = y(t_{n+k}) - y_{n+k}^* \quad (1.36)$$

Metodo inplizituen kasuan mozketa-errore lokala kalkulatzeko kokapen-bereganaketaz gain $f(t_{n+k}, y_{n+k}^*) = f(t_{n+k}, y(t_{n+k}))$ berdintza ere bete egiten dela suposatzen da. Hau da, t_{n+k} puntuko deribatua ere zehatza kontsideratzen da mozketa-errore lokala kalkulatzeko.

Definizio hauek (mozketa-errore global eta lokalarenak) baliokoak dira bai pauso bakarreko metodoentzat baita pauso anitzekoentzat ere. $k=1$ denean, pauso bakarreko metodoentzako mozketa-erroreen definizioak izango genituzke, eta $k>1$ den kasuan, pauso anitzeko metodoentzako mozketa-erroreen definizioak. 1.11. irudian jaso dugu mozketa-errore lokal eta globalaren irudikapena.

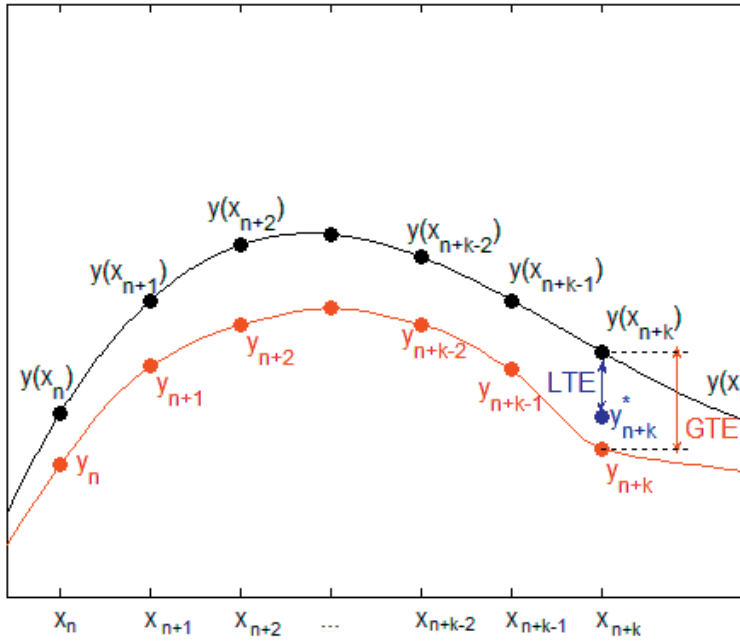
Errorearen metaketaren ondorioz, mozketa-errore globalak mozketa-errore lokalak baino potentzia bat txikiagoa dauka h terminoan. Hau da: $LTE = O(h^{p+1})$ baldin bada, orduan $GTE = O(h^p)$. Zenbakizko metodoa p ordenakoa dela esaten da, mozketa-errore lokala $O(h^{p+1})$ baldin bada, edo gauza bera dena, mozketa-errore globala $O(h^p)$ baldin bada.

Definizioa. Zenbakizko metodo bat emanik, konsistentziak zenbakizko metodo horrek ekuazio diferentziala (edo ekuazio diferentzialen sistema) adierazteko duen gaitasuna neurtzen du. Horretarako, t_{n+k} aldiune bateko mozketa-errore lokala kalkulatu behar da. Eta zenbakizko metodo bat konsistentea dela esaten da, ondorengo baldintza betetzen bada:

$$\lim_{h \rightarrow 0} \left(\frac{1}{h} LTE \right) = 0 \quad (1.37)$$

Definizioa. Zenbakizko metodo bat 0-egonkorra da $h \rightarrow 0$ -ra doanean, perturbazioak edo erroreak anplifikatzen ez badira.

Definizioa. Zenbakizko metodo bat konbergentea izango da konsistentea eta 0-egonkorra denean. Konbergentziak metodoak pauso-tamaina txikiak erabiltzean duen zehaztasuna ziurtatzen du.



1.11. irudia. Mozketa-errore lokal eta globalaren irudikapena.

Hurrengo adibideetan zenbait metodoren mozketa-errore lokalak kalkulatu ditugu.

Adibideak

1. Euler-en metodo esplizituaren mozketa-errore lokalak kalkulatu dugu. Pauso bakarreko metodoa denez, t_{n+1} puntuko balioa kalkulatzeko t_n puntuko balioak erabiliko ditugu eta t_{n+1} puntuko mozketa-errore lokalak ondorengo adierazpenaren bidez emana izango da:

$$LTE = y(t_{n+1}) - y_{n+1}^* \quad (1.38)$$

Euler-en metodo esplizitua erabilita t_{n+1} puntuko balioa honela kalkulatu dela ikusi dugu:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Baina, mozketa-errore lokalaren adierazpenean agertzen den y_{n+1}^* balioa kokapen-bereganaketa egin eta Euler-en metodoa erabiliz kalkulatu dela da. Hau da, $y_n = y(t_n)$ dela suposatuz kalkulatu izan behar du y_{n+1}^* balioak:

$$y_{n+1}^* = y(t_n) + hf(t_n, y(t_n)) \quad (1.39)$$

(1.39) adierazpena (1.38) adierazpenean ordezkatzan badugu, t_{n+1} puntuko mozketa-errore lokala honela geratzen zaigu:

$$LTE = y(t_{n+1}) - [y(t_n) + hf(t_n, y(t_n))] \quad (1.40)$$

$y(t_{n+1})$ balioari dagokion Taylor-en garapena kontsideratuko dugu, ondorengo adierazpenak emana datorrena:

$$y(t_{n+1}) = y(t_n) + (t_{n+1} - t_n)y'(t_n) + \frac{1}{2!}(t_{n+1} - t_n)^2 y''(t_n) + \dots \quad (1.41)$$

(1.41) adierazpena (1.40) adierazpenean ordezkatzuz eta $t_{n+1} - t_n$ diferentziari h deituz, hau da, bi denboren arteko diferentzia zenbakizko metodoan aurrera egiteko erabiliko dugun pauso-tamaina dela kontuan hartuz, (1.40) adierazpena honela geratzen zaigu:

$$\begin{aligned} LTE &= \left[y(t_n) + hy'(t_n) + \frac{1}{2!}h^2 y''(t_n) + \dots \right] - [y(t_n) + hf(t_n, y(t_n))] = \\ &= \left[y(t_n) + hy'(t_n) + \frac{1}{2!}h^2 y''(t_n) + \dots \right] - [y(t_n) + hy'(t_n)] = \frac{1}{2!}h^2 y''(t_n) + O(h^3) \end{aligned}$$

Beraz, Euler-en metodo esplizituko mozketa-errore lokala ondorengo adierazpenaren bidez emana dator:

$$LTE = \frac{1}{2!}h^2 y''(t_n) + O(h^3) \quad (1.42)$$

Eta Euler-en metodo esplizituaren mozketa-errore lokala (1.42) bigarren ordenakoa denez, hau da, $LTE = O(h^2)$, metodo honen ordena $p=1$ da eta haren

errore-konstantea $C = \frac{1}{2}$ da.

2. Euler-en metodo implizitua ere pauso bakarrekkoa da eta t_{n+1} puntuko balioa kalkulatzeko t_n puntuko balioak erabiliko ditugu. t_{n+1} puntuko mozketa-errore lokala ondorengo adierazpenaren bidez emana izango da:

$$LTE = y(t_{n+1}) - y_{n+1}^* \quad (1.43)$$

Euler-en metodo implizitua erabilita t_{n+1} puntuko balioa honela kalkulatzen dela ikusi dugu:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

Baina, mozketa-errore lokalaren adierazpenean agertzen den y_{n+1}^* balioa kokapen-bereganaketa eta metodoa inplizitua denez, $f(t_{n+1}, y_{n+1}^*) = f(t_{n+1}, y(t_{n+1}))$ egin ostean Euler-en metodo inplizitua erabiliz kalkulatzen dena da:

$$y_{n+1}^* = y(t_n) + hf(t_{n+1}, y(t_{n+1})) \quad (1.44)$$

(1.44) adierazpena (1.43) adierazpenean ordezkatzeko badugu, t_{n+1} puntuko mozketa-errore lokala honela geratzen zaigu:

$$LTE = y(t_{n+1}) - [y(t_n) + hf(t_{n+1}, y(t_{n+1}))] \quad (1.45)$$

$y(t_{n+1})$ balioaren Taylor-en garapena hartuko dugu, (1.41) adierazpena. Adierazpen hori (1.40) adierazpenean ordezkatzeko eta, berriz ere, $t_{n+1} - t_n = h$ kontuan izanik, (1.45) adierazpena honela geratzen zaigu:

$$\begin{aligned} LTE &= \left[y(t_n) + hy'(t_n) + \frac{1}{2!}h^2y''(t_n) + \dots \right] - [y(t_n) + hf(t_{n+1}, y(t_{n+1}))] = \\ &= \left[y(t_n) + hy'(t_n) + \frac{1}{2!}h^2y''(t_n) + \dots \right] - [y(t_n) + hy'(t_{n+1})] \end{aligned} \quad (1.46)$$

$y'(t_{n+1})$ lortzeko, (1.41) adierazpena denborarekiko deribatuko dugu:

$$y'(t_{n+1}) = y'(t_n) + (t_{n+1} - t_n)y''(t_n) + \frac{1}{2!}(t_{n+1} - t_n)^2y'''(t_n) + \dots \quad (1.47)$$

Eta (1.47) adierazpena (1.46) adierazpenean ordezkatzeko, Euler-en metodo inplizituko mozketa-errore lokalera iritsiko gara:

$$\begin{aligned} LTE &= \left[y(t_n) + hy'(t_n) + \frac{1}{2!}h^2y''(t_n) + \dots \right] - \left[y(t_n) + h \left(y'(t_n) + hy''(t_n) + \frac{1}{2!}h^2y'''(t_n) + \dots \right) \right] \\ &= -\frac{1}{2}h^2y''(t_n) + O(h^3) \end{aligned}$$

Euler-en metodo inplizituaren ordena ere $p = 1$ da, haren mozketa-errore lokala

$LTE = O(h^2)$ baita. Metodo honen errore-konstantea $C = -\frac{1}{2}$ da.

3. Metodo trapezoidalaren mozketaren errorea lokala kalkulatzeko y_{n+1}^* -en adierazpena kontsideratu behar dugu:

$$y_{n+1}^* = y(t_n) + \frac{1}{2}h(y'(t_n) + y'(t_{n+1}))$$

Mozketa-errorea lokala kalkulatzeko Euler-en metodo implizituan erabili ditugun (1.41) eta (1.47) garapenak erabiliko ditugu:

$$\begin{aligned} LTE &= y(t_{n+1}) - \left[y(t_n) + \frac{1}{2}h(y'(t_n) + y'(t_{n+1})) \right] = \\ &= \left[y(t_n) + hy'(t_n) + \frac{1}{2!}h^2y''(t_n) + \frac{1}{3!}h^3y'''(t_n) \dots \right] \\ &\quad - \left[y(t_n) + \frac{1}{2}hy'(t_n) + \frac{1}{2}h \left(y'(t_n) + hy''(t_n) + \frac{1}{2!}h^2y'''(t_n) + \dots \right) \right] \end{aligned}$$

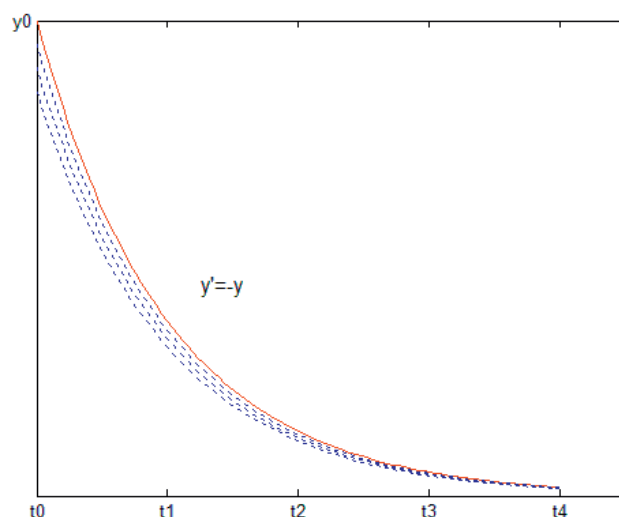
Eta eragiketak eginez zera daukagu:

$$LTE = \left(\frac{1}{6} - \frac{1}{4} \right) h^3 y'''(t_n) + O(h^4) = -\frac{1}{12} h^3 y'''(t_n) + O(h^4)$$

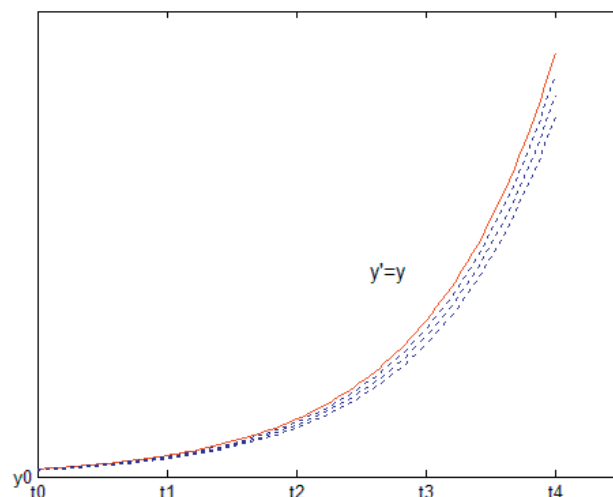
Metodo trapezoidalaren ordena $p = 2$ da eta haren errore-konstantea $C = -\frac{1}{12}$.

1.9. EGONKORTASUNA

Egonkortasunaz ari garenean ekuazio diferentzial baten egonkortasuna eta zenbakizko metodo baten egonkortasuna terminoak bereizi egin behar dira. Ekuazio diferentzial baten egonkortasunak soluzioak perturbazioen aurrean duen sentsibilitatea erakusten du. EDA baten soluzio-kurbak t handitu ahala, hau da, denboran aurrera egin ahala, bata bestearengandik banatzen badoaz, ekuazio diferentziala ezegonkorra da. Matematikoki ekuazio diferentziala egonkorra da bere autobalioen parte erreala negatiboa denean, bestela, ekuazio diferentziala ezegonkorra da. 1.12. eta 1.13. irudietan ikus daiteke, ekuazio diferentziala egonkorra denean edo ezegonkorra denean, perturbazioak nola hedatzen diren. Irudi bietan hasierako balio bera hartu da, y_0 -z adierazi dena.



1.12. irudia. Perturbazioen eraginez sortutako erroreak ekuazio diferentzial egonkor batean. Perturbaziorik gabeko traiektoria lerro solidoz adierazi da eta perturbaziodunak marratxodun lerroz.



1.13. irudia. Perturbazioen eraginez sortutako erroreak ekuazio diferentzial ezegonkor batean. Perturbaziorik gabeko traiektoria lerro solidoz adierazi da eta perturbaziodunak marratxodun lerroz.

Beste kontzeptu bat, zenbakizko metodoaren egonkortasuna da, zeinak zenbakizko emaitzek perturbazioen aurrean duten sentzibilitatea neurtzen baitu. Zenbakizko metodo bat egonkorra da perturbazioak handitzen ez direnean.

Zenbakizko metodo baten egonkortasunaren azterketa egiteko, hauxe da prozesua:

Lehen ordenako ekuazio diferentzial arruntaren sistema bat emanik, sinplifikatzeko jacobitarra diagonalizatu egin daitekeela suposatuko dugu, hau da, $\partial f / \partial y$ diagonalizatu daitekeela. Horrek esan nahi du, existitzen dela autobektoreen matrize bat T , zera betetzen duena:

$$T^{-1} \frac{\partial f}{\partial y} T = \text{diag} \{ \lambda_1, \lambda_2, \dots, \lambda_d \} \quad (1.48)$$

Aldagai-aldaketa bat eginez, posible da ekuazioak desakoplatzea, j osagaiak $\omega_j' = \lambda_j \omega_j$ berdintza beteko duen eran. Ekuazio hauetako bakoitza test-ekuazio izenaz ezagutzen da. Beraz, lehen ordenako ekuazio diferentzial arruntaren kasuan, egonkortasunaren azterketa egiten da test-ekuazioari, hau da, $y' = \lambda y$ ekuazioari zenbakizko metodoa aplikatuz.

Zenbakizko metodoa test-ekuazioari aplikatzen diogunean, diferentziantan emandako ekuazio batera iristen gara, zeina era matrizialean honela adieraz baitaiteke:

$$Y_{n+k} = A Y_{n+k-1} \quad (1.49)$$

Y_{n+k-1} eta Y_{n+k} metodoan agertzen diren aldagaiak edo ezezagunak izanik eta ondorengo adierazpenen bidez emanda datozenak: $Y_{n+k-1} = (y_n, y_{n+1}, \dots, y_{n+k-1})^T$, $Y_{n+k} = (y_{n+1}, y_{n+2}, \dots, y_{n+k})^T$ eta $A = (a_{ij}(\hat{h}))$ $k \times k$ dimentsiodun matrizea, $\hat{h} = h\lambda$ izanik, eta anplifikazio-faktore izenaz ezagutzen dena. Hau da, (1.49) adierazpena era honetan idatz daiteke:

$$\begin{pmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_{n+k} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{pmatrix} \begin{pmatrix} y_n \\ y_{n+1} \\ \vdots \\ y_{n+k-1} \end{pmatrix} \quad (1.50)$$

Pauso bakarreko metodoen kasuan, anplifikazio-faktorea 1×1 ordenako «matrizea» da. Hau da, anplifikazio-faktorea $A = R(\hat{h})$ funtzioa da eta (1.49) ekuazioa era honetan idatz daiteke:

$$y_{n+1} = R(\hat{h}) y_n \quad (1.51)$$

Zenbakizko metodoa pauso anitzekoa denean, anplifikazio faktorea, A , matrize bat da. Kasu horretan, polinomio karakteristikoa edo egonkortasun-polinomioa era honetan definitzen da:

$$p(r) = \det(A - rI) \quad (1.52)$$

A matrizearen autobalioak (1.52) polinomioaren erroak dira. A matrizearen modularik handieneko autobalioak erradio espektral izena dauka eta ρ hizkiaz adierazten da:

$$\rho = \max \{ |\lambda_i| : \lambda_i \text{ } A \text{ - ren autobalioa izanik} \}$$

Lemma. Izan bitez $r_1, r_2, \dots, r_l$ (1.52) polinomioaren erroak, hurrenez hurren $m_1, m_2, \dots, m_l$ anizkoiztasunak dituztenak. Orduan, (1.49) ekuazioaren soluzio orokorra honela emanda dator:

$$y_{n+k} = p_1(n+k)r_1^{n+k} + p_2(n+k)r_2^{n+k} + \dots + p_l(n+k)r_l^{n+k} \quad (1.53)$$

$p_j(n+k)$ polinomioak $(m_j - 1)$ mailakoak izanik, hau da, erroen anizkoiztasuna baino unitate bat txikiagoa den maila izanik. Lemma hori Hairer, Wanner eta Nørsett-en (1993: 379) erreferentzian aurki daiteke.

(1.53) adierazpenak erakusten du, $n \rightarrow \infty$ betetzen denean y_{n+k} balioa bornatua egon dadin, (1.52) polinomioaren erroek unitatedun zirkuluaren barruan egon behar dutela eta justu unitatedun zirkunferentzian jausten diren erroek erro sinpleak izan behar dutela. Erro horiek sinpleak ez balira, anizkoiztasuna baino unitate bat txikiagoko ordena duen polinomio batez bidertuta joango lirатеke eta $n \rightarrow \infty$ kasuan, $y_{n+k} \rightarrow \infty$ beteko litzateke, eta, ondorioz, y_{n+k} balioa ez litzateke bornatuta egongo.

Definizioa. Zenbakizko metodo bat egonkorra dela esango dugu baldin (1.52) polinomioak erroaren baldintza betetzen badu, hau da:

- (1.52) polinomioaren erroak bat unitateko erradioko zirkuluaren barruan edo zirkunferentzian erortzen badira.
- Bat unitateko zirkunferentzian erortzen diren erroak sinpleak direnean.

Definizioa Ondoko eran definituta dagoen S multzoari egonkortasun-eremu esaten zaio:

$$S = \left\{ \hat{h} \in \mathbb{C} : \begin{cases} |r_j(\hat{h})| \leq 1 \quad \forall \hat{h}, r_j \text{ egonkortasun-polinomioaren erro sinplea} \\ |r_j(\hat{h})| < 1 \quad \forall \hat{h}, r_j \text{ egonkortasun-polinomioaren erro anizkoitza} \end{cases} \right\}$$

$\hat{h} = h\lambda$ izanik.

Adibidea

Kontsidera dezagun pauso anitzeko ondorengo zenbakizko metodoa:

$$y_{n+k} - y_{n+k-1} = h \left(\frac{55}{24} f_{n+k-1} - \frac{59}{24} f_{n+k-2} + \frac{37}{24} f_{n+k-3} - \frac{9}{24} f_{n+k-4} \right) \quad (1.54)$$

(1.54) adierazpenaren bidez emana datorren zenbakizko metodoa $k=4$ pausokoa eta $p=4$ ordenakoa da. Hain zuzen, aurrerago ikusiko dugun Adams Bashforth-en $k=4$ pausoko metodoa da. Zenbakizko metodo hau test-ekuazioari aplikatzen diogunean, diferentziantan emandako ondorengo ekuaziora iristen gara:

$$y_{n+4} - y_{n+3} = \hat{h} \left(\frac{55}{24} y_{n+3} - \frac{59}{24} y_{n+2} + \frac{37}{24} y_{n+1} - \frac{9}{24} y_n \right) \quad (1.55)$$

non: $\hat{h} = \lambda h$

(1.55) adierazpenari (1.49) forma eman diezaiokegu edo era baliokidean (1.50) forma:

$$\begin{pmatrix} y_{n+1} \\ y_{n+2} \\ y_{n+3} \\ y_{n+4} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\hat{h}\frac{9}{24} & \hat{h}\frac{37}{24} & -\hat{h}\frac{59}{24} & 1 + \hat{h}\frac{55}{24} \end{pmatrix} \cdot \begin{pmatrix} y_n \\ y_{n+1} \\ y_{n+2} \\ y_{n+3} \end{pmatrix} \Rightarrow Y_{n+4} = AY_{n+3} \quad (1.56)$$

$Y_{n+4} = (y_{n+1}, y_{n+2}, y_{n+3}, y_{n+4})^T$, $Y_{n+3} = (y_n, y_{n+1}, y_{n+2}, y_{n+3})^T$ eta

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\hat{h}\frac{9}{24} & \hat{h}\frac{37}{24} & -\hat{h}\frac{59}{24} & 1 + \hat{h}\frac{55}{24} \end{pmatrix} \text{ izanik.}$$

(1.56) adierazpeneko A matrizearen polinomio karakteristikoa kalkulatu dugu eta zerora berdinduko dugu haren erroak aurkitzeko:

$$p(r) = \det(A - rI) = r^4 - \left(1 + \frac{55}{24}\hat{h}\right)r^3 + \frac{59}{24}\hat{h}r^2 - \frac{37}{24}\hat{h}r + \frac{9}{24}\hat{h} = 0 \quad (1.57)$$

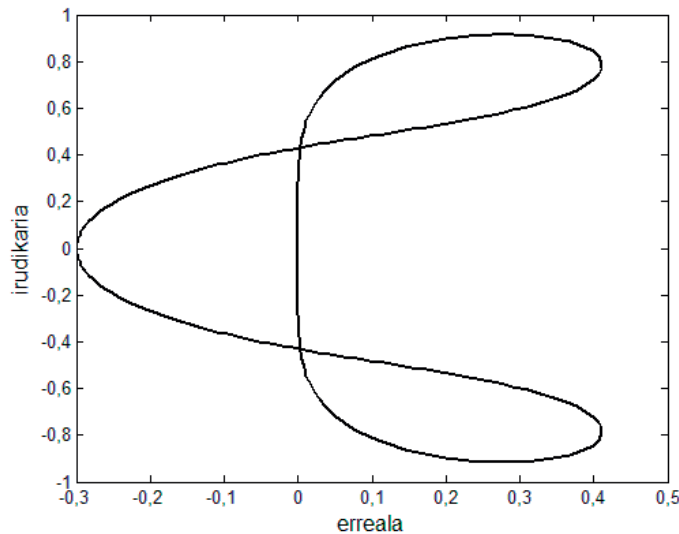
Polinomio karakteristiko berdiner iritsiko ginateke, beste bide honi jarraituz gero: (1.55) diferentzian emandako ekuazio bat denez, $y_p = r^p$ motako soluzioekin egiten dugu proba. Lortzen den adierazpena r^n -z zatituz gero, (1.57) adierazpenarekin bat datorren berdintza honetara iritsiko ginateke:

$$r^4 - \left(1 + \frac{55}{24}\hat{h}\right)r^3 + \frac{59}{24}\hat{h}r^2 - \frac{37}{24}\hat{h}r + \frac{9}{24}\hat{h} = 0 \quad (1.58)$$

$\hat{h}$ direlakoan leku geometrikoa, zeinentzat (1.58) ekuazioaren erroak unitatea baino txikiagoak edo berdinak diren, horixe da zenbakizko metodoaren egonkortasun-eremua osatzen duten elementuen multzoa. (1.58) adierazpenetik $\hat{h}$ askatuko dugu eta erroak unitatearen berdinak direneko lekua marraztuko dugu. Horretarako, $r = e^{i\theta}$ egingo dugu, non $\theta \in [0, 2\pi)$:

$$\hat{h} = \frac{e^{4i\theta} - e^{3i\theta}}{\frac{55}{24}e^{3i\theta} - \frac{59}{24}e^{2i\theta} + \frac{37}{24}e^{i\theta} - \frac{9}{24}} \quad (1.59)$$

θ parametroari balioak emanda gai gara unitatearen berdinak diren erroek osatzen duten leku geometrikoa irudikatzeko eta horrela S multzoaren muga irudikatuko dugu, hau da, egonkortasun-eremuaren muga. Egonkortasun-eremua muga horren barneko edo kanpoko zatia den jakiteko nahikoa da (1.58) adierazpenean $\hat{h}$ -ren balio bat ordezkatzeko eta polinomio horren erroekin zer gertatzen den aztertzea. (1.55) metodoari dagokion egonkortasun-eremua 1.14. irudian marraztu dugu.



1.14. irudia. $k = 4$ pausoko Adams Bashforth-en egonkortasun-eremua (kurbaren barneko aldea).

Definizioa (A-egonkortasuna edo egonkortasun absolutua). Zenbakizko metodo bat A-egonkorra dela esaten da, plano konplexuko alde negatibo osoa egonkortasun-eremuaren parte denean. Hau da, $\mathbb{C}^- \in S$ betetzen denean, S zenbakizko metodoaren egonkortasun-eremua izanik. Ikus 1.15. irudia.

Definizioa (L-egonkortasuna). Pauso bakarreko metodo bat L-egonkorra dela esaten da, A-egonkorra denean eta test ekuazioari metodoa aplikatzerakoan $y' = \lambda y$, $\lambda \in \mathbb{C}$ izanik eta $Re(\lambda) < 0$, $Re(h\lambda) \rightarrow -\infty$ denean orduan $|R(\lambda h)| \rightarrow 0$ betetzen bada, non $y_{n+1} = R(\lambda h)y_n$ den.

Definizioa ($A(\alpha)$ -egonkortasuna). Izan bedi S zenbakizko metodo baten egonkortasun-eremua. Zenbakizko metodoa $A(\alpha)$ -egonkorra izango da, baldin $0 < \alpha < \frac{\pi}{2}$ $S_\alpha = \{\hat{h} : -\alpha < \pi - \arg(\hat{h}) < \alpha\} \subseteq S$ betetzen bada. Ikus 1.15. irudia.

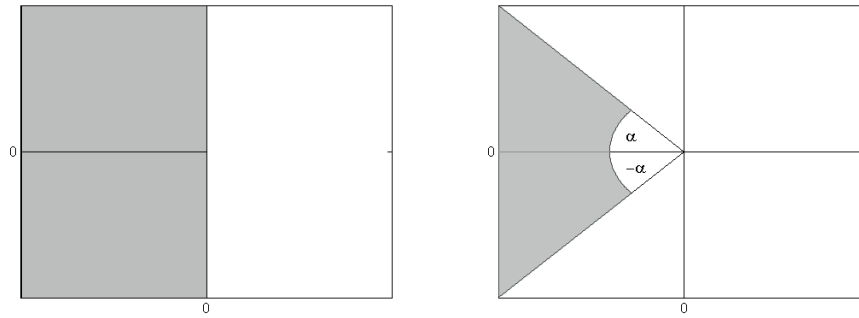
Definizioa. Zenbakizko metodo bat $A(0)$ -egonkorra da, $A(\alpha)$ -egonkorra denean α -ren balio txiki batentzat, $\alpha > 0$ izanik.

Definizioa. Zenbakizko metodo bat *stiff* egonkorra dela esaten da (*stiff* ingelesez zurruna da), ondokoa betetzen denean: $\{\hat{h} : Re(\hat{h}) < -a\} \cup \{\hat{h} : -a \leq Re(\hat{h}) < 0, -c \leq Im(\hat{h}) \leq c\} \subseteq S$ eta $a, c \in \mathbb{R}^+$ izanik. Ikus 1.16. irudia.

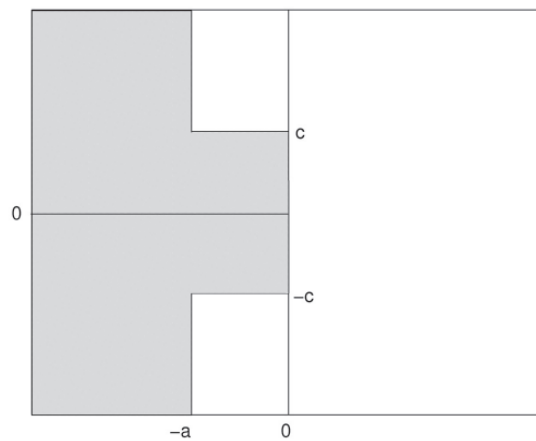
Une honetara iritsita, honako teorema hau formula dezakegu:

Teorema (Dahlquist, 1963). A-egonkorra den pauso anitzeko zenbakizko metodo lineal baten ordena $p \leq 2$ da. Metodo trapezoidala da 2 ordenakoa eta A-egonkorra

izanik, errore-konstanterik txikiena duen metodoa, haren errore-konstantea $C = -\frac{1}{12}$ izanik. Frogapena Hairer eta Wanner-en (1991: 265-266) erreferentzian aurki daiteke.



1.15. irudia. A-egonkortasuna (ezkerrean) eta $A(\alpha)$ -egonkortasuna (eskuinean).



1.16. irudia. *Stiff* egonkortasuna.

1.10. EULER-EN METODOEN EGONKORTASUN-EREMUAK

Euler-en metodo esplizituaren eta inplizituaren egonkortasun-eremuak zeintzuk diren kalkulatu dugu atal honetan. Horretarako, aurreko atalean ikusi dugun teoriaz baliatuko gara.

1.10.1. Euler-en metodo esplizituaren egonkortasun-eremua

Eman dezagun (1.21) adierazpeneko Euler-en metodo esplizitua. Metodo hori aplikatu dugu $y' = \lambda y$ test-ekuazioari:

$$y_{n+1} = y_n + h\lambda y_n = (1 + \hat{h})y_n \quad (1.60)$$

non $\hat{h} = \lambda h$. Aurrez ikusi dugun (1.51) berdintzak (1.60) forma hartzen du Euler-en metodo esplizituaren kasuan, hau da:

$$y_{n+1} = R(\hat{h})y_n \quad (1.61)$$

non $R(\hat{h}) = 1 + \hat{h}$. (1.61) berdintzan $y_p = r^p$ ordezkatzuz eta lortzen den adierazpe-nean r^n sinplifikatuz, zera daukagu:

$$p(r) = r - (1 + \hat{h}) \quad (1.62)$$

$p(r)$ da Euler-en metodo esplizituko polinomio karakteristikoa. Haren erro sinplea $r_1 = 1 + \hat{h}$ da eta Euler-en metodo esplizituaren egonkortasun-eremua osatuko dute $|1 + \hat{h}| \leq 1$ betetzen duten plano konplexuko $\hat{h}$ zenbakiek. Plano konplexuan aurreko baldintza betetzen duten puntuek $(-1, 0)$ zentroan eta bat unitateko erradiodun zirkuluan daude. 1.17. irudiaren ezker aldean marraztu dugu Euler-en metodo esplizituaren egonkortasun-eremua. $\hat{h}$ erreala denean, egonkortasun-tartea haxe litzateke:

$$-1 \leq 1 + \hat{h} \leq 1 \quad (1.63)$$

Beste era batean esanda, $\hat{h}$ erreala denean, $-2 \leq \hat{h} \leq 0$ da Euler-en metodoaren egonkortasun-tartea.

1.10.2. Euler-en metodo implizituaren egonkortasun-eremua

Euler-en metodo implizitua, (1.22) adierazpenaz emandakoa, $y' = \lambda y$ test-ekuazioari aplikatzen diogunean, haxe da lortzen dena:

$$y_{n+1} = y_n + \hat{h}y_{n+1} \quad (1.64)$$

(1.64) adierazpenetik y_{n+1} balioa askatuko dugu:

$$y_{n+1} = \frac{1}{1 - \hat{h}} y_n \quad (1.65)$$

Kasu honetan, Euler-en metodo implizituko polinomio karakteristikoa haxe da.

$$p(r) = r - \frac{1}{1 - \hat{h}} \quad (1.66)$$

(1.66) polinomio karakteristikoaren erro sinplea da:

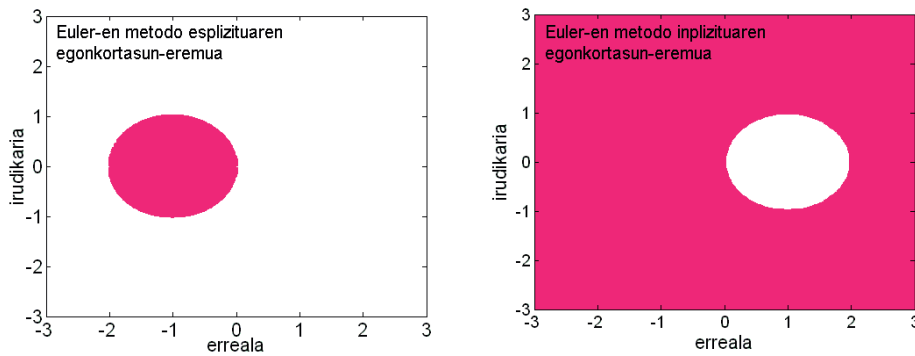
$$r = \frac{1}{1 - \hat{h}}$$

Eta Euler-en metodo inplizituko egonkortasun-eremua osatuko dute ondorengo baldintza betetzen duten plano konplexuko $\hat{h}$ puntuen leku geometrikoek:

$$\left| \frac{1}{1 - \hat{h}} \right| \leq 1 \quad (1.67)$$

(1.67) baldintza betetzen dute (1,0) zentrodun eta bat unitateko erradiodun zirkulutik kanpo dauden plano konplexuko puntuak. 1.17. irudiaren eskuinaldean metodo honen egonkortasun-eremua marraztu da. $\hat{h}$ erreala denean, egonkortasun-tartea hauxe litzateke:

$$\hat{h} \in (-\infty, 0] \cup [2, +\infty) \quad (1.68)$$



1.17. irudia. Euler-en metodo esplizituaren (ezkerrean) eta inplizituaren (eskuinean) egonkortasun-eremuak.

1.10.3. Metodo trapezoidalaren egonkortasun-eremua

Metodo trapezoidala, $y' = \lambda y$ test-ekuazioari aplikatzen diogunean, hauxe da lortzen dena:

$$y_{n+1} = y_n + \frac{1}{2} \hat{h} (y_n + y_{n+1}) \quad (1.69)$$

(1.69) adierazpenetik y_{n+1} balioa askatuz, zera daukagu:

$$y_{n+1} = \frac{1 + \frac{1}{2}\hat{h}}{1 - \frac{1}{2}\hat{h}} y_n \quad (1.70)$$

Metodo trapezoidaleko polinomio karakteristikoa hauxe da:

$$p(r) = r - \frac{1 + 1/2\hat{h}}{1 - 1/2\hat{h}} \quad (1.71)$$

Eta polinomio karakteristikoaren erroa $r = \frac{1 + 1/2\hat{h}}{1 - 1/2\hat{h}}$ da. Metodo trapezoidalaren egonkortasun-eremua osatuko dute ondorengo baldintza betetzen duten $\hat{h}$ puntuek:

$$\left| \frac{1 + 1/2\hat{h}}{1 - 1/2\hat{h}} \right| \leq 1 \quad (1.72)$$

$\lambda < 0$ denean, beti betetzen da (1.72) adierazpena, kasu horretan izendatzailea zenbakitzailea baino handiagoa baita. Beraz, metodo trapezoidalaren egonkortasun-eremua da plano konplexuaren ezker alde osoa. $\hat{h}$ erreal denean, egonkortasun-tartea hauxe litzateke:

$$\hat{h} \in (-\infty, 0)$$

Adibideak

1. Demagun ondorengo ekuazio diferentziala Euler-en metodo esplizitua erabilita askatu nahi dugula: $y' = -16y$. Ekuazio diferentzial horretako autobalioa $\lambda = -16$ da. Eta autobalio errealetarako daukagun egonkortasun absolutuaren (1.63) baldintza honela geratuko litzateke kasu honetan:

$$-1 \leq 1 - 16h \leq 1 \Rightarrow -2 \leq -16h \leq 0 \quad (1.73)$$

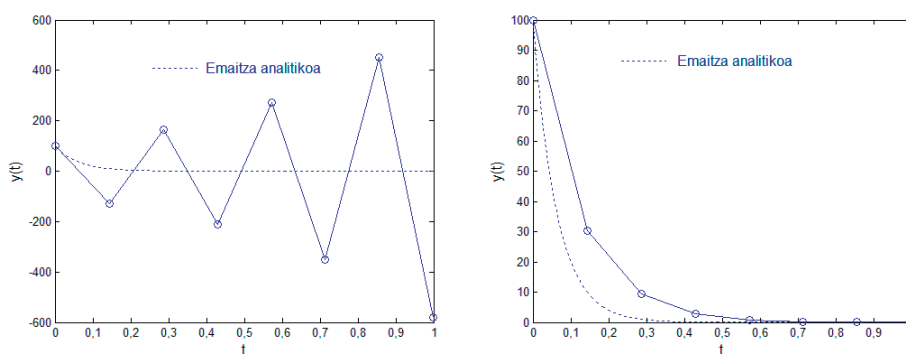
(1.73) inekuazioko $-16h \leq 0$ edozein pauso-tamainak beteko du, hau da, edozein h -k. Baina $-2 \leq -16h$ betetzeko, pauso-tamainak $h \leq 1/8$ bete beharko du. Horrek esan nahi du ezen ekuazio diferentzial hau Euler-en metodo esplizituarekin askatzean egonkortasun-baldintza betetzeko $h \leq 1/8$ betetzen duten pauso-tamainak aukeratu beharko ditugula. Era berean, $y' = -160y$ ekuazio diferentziala Euler-en metodo esplizituarekin askatzeko $h \leq 1/80$ betetzen duten pauso-tamainak erabili beharko genituzke.

Ekuazio diferentzial berdina Euler-en metodo implizituarekin askatu nahi bagenu, erabili beharko genukeen pauso-tamaina kalkulatzeko (1.67) baldintzari erreparatu beharko genioke. Horrela, $y' = -16y$ ekuazioarentzat honela geratuko litzateke baldintza hori:

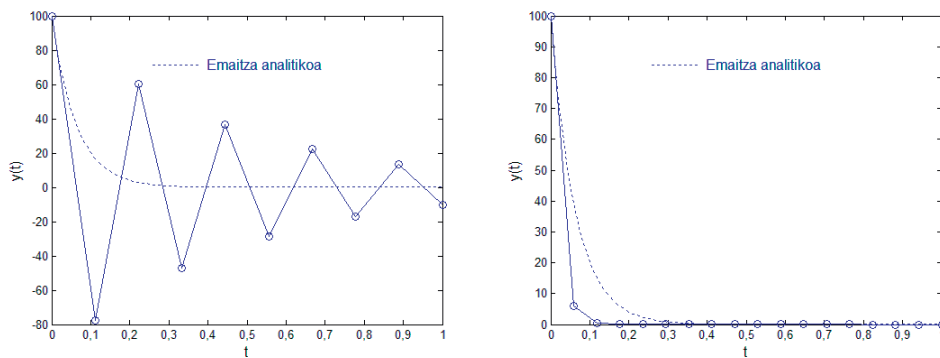
$$-1 \leq \frac{1}{1+16h} \leq 1 \quad (1.74)$$

Eta (1.74) baldintza edozein pauso-tamainak betetzen du. Beraz, Euler-en metodo implizituarekin $y' = -16y$ ekuazioa askatu nahi badugu, edozein pauso-tamaina erabilia ere, egonkortasun absolutuaren baldintza bete egingo da. Gauza bera gertatzen da $y' = -160y$ ekuazioa Euler-en metodo implizituarekin askatu nahi dugunean, pauso-tamainak ez baitu inolako baldintzarik bete beharko egonkortasun baldintzari dagokionez.

$y' = -16y$ ekuazio diferentzialaren soluzio-kurbak irudikatu ditugu 1.18. irudian. Hasierako balio moduan $y(0)=100$ aukeratu da eta $[0, 1]$ tartean aurkitu da ekuazio diferentzialaren soluzioa. Honetarako 7 pauso eman dira bai Euler-en metodo esplizituarekin, baita implizituarekin ere. Hau da, $h=1/7$ pauso-tamaina erabili da. 1.18. irudian antzeman daitekeenez, Euler-en metodo esplizituarekin lortzen den emaitza ez da ona. Problema bera 9 pausorekin ebatzita lortzen diren emaitzak 1.19. irudian jaso dira. Kasu horretan, pauso-tamaina $h=1/9$ da eta egonkortasun-baldintza betetzen du, $h \leq 1/8$. Kasu horretan Euler-en metodo esplizituarekin lortzen den emaitza emaitza analitikoaren inguruan bibratzen du. Emaitza ez da oso ona baina behintzat ez doa anplifikatuz. Euler-en metodo esplizituan gutxienez 16 pauso behar dira emaitza hurbildua «bibratzen» egon ez dadin. 1.19. irudiaren ezkerrean 17 pausorekin lortu den emaitza irudikatu da. Ekuazio diferentzial honen emaitza analitikoa ere irudikatu da kasu bietan.



1.18. irudia. $y' = -16y$ ekuazio diferentzialaren emaitza Euler esplizituaz (ezkerrean) eta Euler implizituaz (eskuinean).



1.19. irudia. $y' = -16y$ ekuazio diferentzialaren emaitza Euler-en metodo esplizitua erabilia (ezkerrean 9 pausorekin eta eskuinean 17 pausorekin).

2. $y' = \lambda y$ ekuazio diferentziala emanik eta $\lambda = -6 + 8i$ izanik, Euler-en metodo esplizitua erabiliz kalkulatu nahi dugu zein den egonkortasun absolutuaren baldintza beteko duen pauso-tamainarik handiena. Ondokoa bete behar da horretarako:

$$|1 + h(-6 + 8i)| \leq 1 \quad (1.75)$$

$1 + h(-6 + 8i)$ zenbakiaren modulua kalkulatu dugu:

$$\sqrt{(1 - 6h)^2 + (8h)^2} = \sqrt{1 + 36h^2 - 12h + 64h^2} = \sqrt{1 - 12h + 100h^2}$$

$\lambda = -6 + 8i$ baliorako Euler-en metodo esplizituko egonkortasun absolutuko baldintza honela geratzen zaigu:

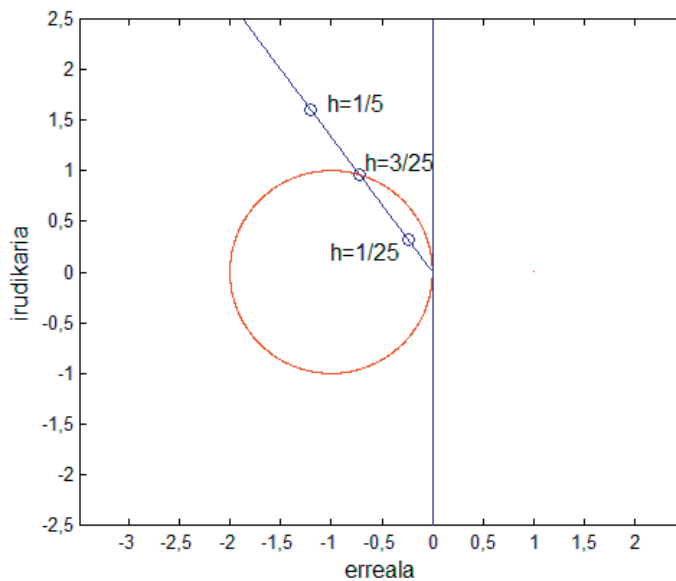
$$1 - 12h + 100h^2 \leq 1 \Rightarrow -12h + 100h^2 \leq 0 \Rightarrow h(100h - 12) \leq 0 \Rightarrow (100h - 12) \leq 0 \quad (1.76)$$

Eta egonkortasun absolutuko baldintza betetzeko, pauso-tamainak $h \leq 3/25$ izan beharko du. Geometrikoki honela interpreta daiteke emaitza hori: h -ri balioak eman ahala, $h\lambda = h(-6 + 8i)$ puntuen kokapena plano konplexuko zuzen bat

da. Konkretuki $\hat{h} = \hat{x} + i\hat{y}$ bezala idazten badugu, zuzenaren ekuazioa $\hat{y} = \frac{-4}{3}\hat{x}$

ekuazioaren bidez emana egongo da. 1.20. irudian irudikatu dugun zuzena eta bertan markatu ditugun puntuak $h = 1/25$, $h = 3/25$ eta $h = 1/5$ puntuei dagozkie. $h = 3/25$ denean, $h\lambda = h(-6 + 8i)$ puntua Euler-en metodoaren egonkortasun-eremuaren mugan jausten da; $h = 1/5$ denean, $h\lambda = h(-6 + 8i)$ puntua egonkortasun-eremutik kanpo dago, eta $h = 1/25$ denean, $h\lambda$ puntua egonkortasun-eremuaren barruan

dago. Euler-en metodo esplizitua erabiliz ekuazio diferentzial hau askatu nahi badugu, egonkortasun absolutuko baldintza beteko da $h \leq 3/25$ betetzen duten pauso-tamainak aukeratzen ditugunean.



1.20. irudia. Euler-en metodo esplizituko egonkortasun-eremua eta $h(-6+8i)$ puntuen kokapena.

3. MATLAB erabilia era honetan marraztu ahal izango genituzke Euler-en metodo esplizituaren eta implizituaren egonkortasun-eremuak. Euler-en metodo esplizituaren kasuan, polinomio karakteristikoa $p(r) = r - (1 + \hat{h})$ zerora berdinduko dugu eta $\hat{h}$ askatuko dugu:

$$\hat{h} = r - 1 \quad (1.77)$$

Eta $|r|=1$ betetzen duten puntuen muga marraztuko dugu. Horixe izango da egonkortasun-eremuaren muga, eta egonkortasun-eremua muga horren alde batera edo bestera kokatuta egongo da. $|r|=1$ egiteko kodean $r = e^{i\theta}$, $\theta \in [0, 2\pi)$ egingo dugu.

```
%Euler-en metodo esplizituko egonkortasun eremua.
%n deituko diogu egonkortasun eremua marrazteko erabiliko dugun
%puntu kopuruari:

n=1000;
fi=0:2*pi/n:2*pi;
for j1=1:n+1
    r(j1)=exp(fi(j1)*i);
    h(j1,:)=r(j1)-1;
    plot(h(j1,:), '-r')
    hold on
end
```

Euler-en metodo inplizituaren kasuan ere berdin egiten dugu. (1.66) adierazpena zerora berdindu eta $\hat{h}$ askatuko dugu:

$$\hat{h} = 1 - \frac{1}{r} \quad (1.78)$$

$|r|=1$ betetzen duten puntuen muga marrazten dugu eta horixe izango da egonkortasun-eremuaren muga. Berriz ere kodean $r = e^{i\theta}$, $\theta \in [0, 2\pi)$ egingo dugu $|r|=1$ ziurtatzeko. MATLABen lerro hauek idatzi beharko ditugu egonkortasun-eremuaren muga hori marrazteko:

```
%Euler-en metodo inplizituko egonkortasun eremua.
%n deituko diogu egonkortasun eremua marrazteko erabiliko dugun
%puntu kopuruari:
n=1000;
fi=0:2*pi/n:2*pi;
for j1=1:n+1
    r(j1)=exp(fi(j1)*i);
    h(j1,:)=1-1/r(j1);
    plot(h(j1,:), '-r')
    hold on
end
```

1.11. **ERROREA VS ANPLIFIKAZIO-FAKTOREA PAUSO BAKARREKO METODOETAN**

Garrantzitsua da anplifikazio-faktorearen eta mozketa-errore globalaren arteko erlazioa zein den argi uztea. Pauso bakarreko metodoen mozketa-errore globala era honetan kalkulatzen dela ikusi dugu:

$$GTE_{n+1} = y(t_{n+1}) - y_{n+1} \quad (1.79)$$

(1.79) adierazpenean y_{n+1}^* terminoa batzen eta kentzen sartuz, hauxe daukagu:

$$GTE_{n+1} = y(t_{n+1}) - y_{n+1}^* + y_{n+1}^* - y_{n+1}$$

Eta mozketa-errore lokala hau dela kontuan izanda $LTE_{n+1} = y(t_{n+1}) - y_{n+1}^*$ zera daukagu:

$$GTE_{n+1} = LTE_{n+1} + y_{n+1}^* - y_{n+1} \quad (1.80)$$

Nomenklatura sinplifikatuko dugu era honetan: $G \equiv GTE$, $L \equiv LTE$. Beraz, (1.80) ekuazioa era honetan idatziko dugu nomenklatura berria erabilita:

$$G_{n+1} = L_{n+1} + y_{n+1}^* - y_{n+1} \quad (1.81)$$

y_{n+1}^* eta y_{n+1} balioentzat (1.49) erabiliz, zera daukagu: $y_{n+1}^* = Ay(t_n)$ eta $y_{n+1} = Ay_n$:

$$G_{n+1} = L_{n+1} + Ay(t_n) - Ay_n = L_{n+1} + A(y(t_n) - y_n) \quad (1.82)$$

Berriz ere $G_n = y(t_n) - y_n$ betetzen dela kontuan hartuta, zera daukagu:

$$G_{n+1} = L_{n+1} + AG_n \quad (1.83)$$

Eta (1.83) adierazpena kontuan hartuz errorea zelan hedatzen den ikus dezakegu:

$$\begin{aligned} G_{n+1} &= L_{n+1} + AG_n = L_{n+1} + A(L_n + AG_{n-1}) = \\ &= A^2G_{n-1} + AL_n + L_{n+1} = \dots = A^nG_1 + \sum_{i=0}^{n-1} A^i \cdot L_{n-i+1} \end{aligned} \quad (1.84)$$

Lehenengo pausoan sortzen den errore globala errore lokalaren berdina da. $(t_0, y(t_0))$ puntutik abiatzen bagara, zera baitaukagu:

$$G_1 = y(t_1) - y_1 = y(t_1) - y_1^* + y_1^* - y_1 = L_1 + y_1^* - y_1 \quad (1.85)$$

non: $y(t_0) = y_0$ denez, $y_1^* = y_1$ betetzen baita. Beraz, $G_1 = L_1$ dela kontuan hartuz, errore globalaren (1.84) adierazpena honela geratzen zaigu:

$$G_{n+1} = \sum_{i=0}^n A^i \cdot L_{n-i+1} \quad (1.86)$$

Eta (1.86) adierazpenak pauso bakarreko metodoetako mozketa-errore globala anplifikazio-faktorearen eta mozketa-errore lokalaren menpe ematen digu.

2. Zurruntasuna

Ekuazio diferentzial arruntak (EDAk) zenbakizko metodoak erabilia askatzerakoan zurruntasuna (ingelesez, *stiffness*) kontzeptu garrantzitsua da. Hainbat autore bat datoz zurruntasunari buruzko definizio zehatz bat ez dela existitzen diotenean. Bai esan dezakegu, zurruntasuna askatzen ari garen EDAREN mendekoa dela, EDako hasierako baldintzen mendekoa ere, baita EDA askatzeko darabilgun zenbakizko metodoaren mendekoa eta emaitza bilatzen ari garen denbora-tartearen mendekoa ere.

Lambert-en (1973: 231-232) erreferentzian zurruntasunari buruzko definizio hau ematen da: $y' = Ay + \Phi(t)$ sistema lineala, $y, \varphi \in \mathbb{R}^m$ eta A $m \times m$ ordenako matrizea izanik, zurruna (*stiff*) da, ondorengo baldintzak betetzen badira:

- A matrizearen autobalio guztien parte erreala negatiboa da. Hau da: $Re(\lambda_i) < 0$, $\lambda_i, i = 1, 2, \dots, m$ A matrizearen autobalioak izanik.
- A matrizearen autobaliorik handienaren eta txikienaren parte errealean artean diferentzia handia dago. Hau da: $\max_{i=1,2,\dots,m} |Re(\lambda_i)| \gg \min_{i=1,2,\dots,m} |Re(\lambda_i)|$

Erreferentzia berean (Lambert, 1973: 231-232), zurruntasun-erradioa definitzen da: zurruntasun-erradio esaten zaio balio absolutuan parte errealik handiena eta txikiena dituzten autobalioen parte errealean balio absolutuen zatidurari:

$$\rho = \frac{\max_{i=1,2,\dots,m} |Re(\lambda_i)|}{\min_{i=1,2,\dots,m} |Re(\lambda_i)|} \quad (2.1)$$

Eta zurruntasun-erradioaren (2.1) definizioa erabilia, $y' = Ay + \Phi(t)$ motako sistema zurruna dela esaten da, baldin eta A matrizearen autobalio guztien parte erreala negatiboa bada eta zurruntasun-erradioa *handia* bada.

EDA sistema ez-lineal bat daukagun kasurako $y' = f(t, y)$, zurruntasunaren definizio hau ematen du (Lambert, 1973: 231-232) erreferentziak:

Sistema zurruna dela esaten da jacobitarraren autobalioek, $J = \partial f / \partial y$, ekuazio diferentzial arruntan sistema lineal bateko A matrizearen autobalioen portaera bera dutenean. Sistema ez-linealaren kasuan, autobalioak t aldagaiaren mendekoak izango dira. Horrela, sistema ez-lineal bat T denbora-tarte batean zurruna dela esango dugu baldin $\lambda_i(t)$ autobalioek sistema lineal zurrun batekoek bete behar dituzten baldintza berak betetzen badituzte: beraien parte erreala negatiboa izatea eta sistemako zurruntasun-erradioa *handia* izatea, hain zuzen.

Argi dago orain arte emandako definizioak zehatzak ez direla, ez baitute esaten autobalioen parte errealean arteko diferentzia *handia* zenbatekoa den edota zurruntasun-erradio *handia* zenbatekoa den. Gainera zurruntasunaren definizioan dagoen zehaztasun-gabezia horretaz gain, bibliografian zurruntasunari buruz agertzen diren zenbait baieztapen ere elkarren artean kontrajarriak dira. Besteak beste, hauek dira bibliografian zurruntasunari buruz aurki ditzakegun baieztapen batzuk:

Lehenengo baieztapena: Kostelich eta Armbruster-en (1996: 479-482) erreferentzian soluzioko batugai batzuk beste batzuk baino arinago erortzen edo txikitzen direnean gertatzen dela zurruntasuna esaten da. Erreferentzia honetan ziurtzat jotzen da problemaren emaitza zehatzean edo analitikoan batugai bat baino gehiago existitzen dela.

Bigarren baieztapena: Hoffman-en (2001: 401-408) erreferentzian zenbakizko metodoan erabili beharreko pauso-tamaina zehaztasunak mugatu beharrean egonkortasunak mugatzen duenean gertatzen dela zurruntasuna esaten da. Baieztapen honekin ez datorrela bat adierazten du, ordea, Lambert-ek, (1991: 219) egonkortasuna eta zehaztasuna bereizezinak baitira.

Hirugarren baieztapena: Hubbard eta West-en (1991: 238-239) erreferentzian zenbakizko emaitzak ekuazio diferentzialaren portaera kualitatiboa errepikatzea nahi baldin badugu, pauso-tamaina egonkortasun-mugekin zerikusia duen balio bat baino txikiagoa aukeratu behar dela esaten da.

Laugarren baieztapena: Heath-en (1997: 282) erreferentzian zurruntasuna magnitude ezberdinetako autobalioen existentziaren ondorio bezala definitzen da, ezberdintasun hau autobalioen parte errealean edota irudikarian eman daitekeelarik. Hau da, erreferentzia honetan hasieran aipatu dugun zurruntasunaren definizioaren hedapen bat egiten da, magnitude-ezberdintasuna problemako autobalioen parte errealean ezezik parte irudikarietan ere ager daitekeela esanez.

Bostgarren baieztapena: Lambert-en (1991: 213-224) erreferentzian zurruntasun kontzepturako definizio praktiko bat ematen da. Hasierako baliodun EDA bat (edo EDA sistema bat) zurruna dela esaten da, zenbakizko metodo bat aplikatzen diogunean eta emaitza aurkitzeko pauso-tamaina oso txikiak erabili behar dituenean. Esaten da zurruntasuna sortzen dela bereziki egonkortasun-eremu txikiko zenbakizko metodoak erabiliz zenbait EDA askatzean, eta, beraz, egonkortasun-eremu txikia duten zenbakizko metodoek EDA zurrunen soluzioa aurkitzerakoan arazoak izango dituztela.

Kapitulu honen helburua, zurruntasun kontzeptua argitzea izango da.

2.1. ZURRUNTASUN KONTZEPTUAREN INGURUKO ZENBAIT KONTSIDERAZIO

Lambert-en (1973) erreferentziaren ondoren zurruntasunari buruz egindako azterketek erakusten dute erreferentzia horretan emandako definizioak ez direla zehatzak. Autore berak (Lambert, 1991: 213-224) kontradibideak aurkitzen ditu, berak (Lambert, 1973) emandako zurruntasunaren definizioa betetzen duten EDA sistemak zurrunak ez direnekoak. Kontradibide horiek, Lambert-ek (1973) emandako zurruntasunaren definizioa zuzentzera eramango dute.

2.1.1. Zurruntasuna ez da bakarrik sistemetan gertatzen

Lambert-ek (1973) zurruntasunari buruz emandako definizioak problemako autobalioen parte errealean arteko konparazioa egiten du. Ziurtzat jotzen du EDAk autobalio bat baino gehiago izango dituela, beraz, ekuazio diferentzial bat baino gehiago izan beharko ditugu. Hau da, badirudi zurruntasuna gertatzeko EDA sistema bat izan beharko dugula. Baina hau ez da egia, zurruntasuna ekuazio diferentzial arrunt bakarra daukagunean ere gerta baitaiteke. Hau da, zurruntasuna gerta dadin ez da beharrezkoa magnitude ezberdinetakoak diren autobalioak dituen EDA sistema bat izatea. Adibide batzuen bidez ikusiko dugu hori.

Adibideak

1. Campbell-en (1990: 472-475) erreferentzian, hurrengo hasierako baliodun EDA ez-homogeneoa proposatzen da (guk hasierako balioa aldatuta kontsideratu duguna, erreferentzian bertan $y(0) = 4$ hasierako balioarekin baitago):

$$y' = -40y + 40t + 1, \quad y(0) = 1 \quad (2.2)$$

(2.2) ekuazioaren emaitza zehatza $y(t) = t + e^{-40t}$ da. Emaita zehatzak bi batugai ditu: alde batetik, t batugaia, EDAREN emaitza partikularra dena eta oso geldo aldatzen dena, eta, beste aldetik, e^{-40t} batugaia, EDAREN emaitza homogeneoa dena

eta oso azkar zerorantz jotzen duena. Era honetako soluzioak zirkuitu elektrikoetan, erreakzio kimikoetan, eta abarretan gertatzen dira.

Problema honetako e^{-40t} batugaia oso azkar zero egiten denez, soluzioa oso azkar $y(t) \approx t$ bihurtzen da. Hala eta guzti, problema honen soluzioa zenbakizko metodoak erabiliz aurkitu nahi badugu, behartuta gaude pauso-tamaina oso txikiak erabiltzera, zeren $\lambda = -40$ autobalioak problemaren egonkortasunean eta zehaztasunean agintzen baitu.

MATLABek EDak askatzeko zenbait funtzio eskaintzen dizkigu. Horietako funtzio bi ode45 eta ode15s dira (aurrerago ikusiko ditugunak). Funtzio horiek erabilia (2.2) ekuazioa $T = [0, 10]$ tartean askatzen badugu, ode15s funtzioak 49 pauso ematen ditu eta ode45 funtzioak 127 pauso. EDA bera $T = [0, 30]$ tartean askatzen badugu, ode15s-k 51 pauso beharko ditu eta ode45-ek 368 pauso. MATLABek eskaintzen dituen bi algoritmo horietatik ode15s da, hain zuzen, ekuazio diferentzial zurrunk askatzeko gomendatzen dena.

(2.2) EDaren emaitzean ageri diren batugaien txikitze-erritmoak ezberdinak dira, autobalio bakarra duen problema daukagu eta, adibidez, ode45 funtzioaren bidez EDA hau askatzean honek pauso asko eman behar ditu. (2.2) EDA zurrunka da, nahiz eta aurretik aipatu ditugun baieztapen guztiak betetzen ez dituen. Kasu honetan, EDA bakarra daukagu eta autobalioa ere bakarra da: $\lambda = -40$. Daukagun EDA, aldiz, zurrunka da, zeren emaitza zehatzeko bi batugaien (emaitza homogeneoa eta emaitza partikularra) txikitze-erritmoak ezberdinak baitira.

2. EDA bakarra zurrunka gertatzen deneko beste adibide bat sugarren hedatze-ereduan aurki dezakegu. Eredu horri dagokion EDA ez-lineala da. Problema honen emaitza MATLABeko ode45 eta ode15s funtzioak erabilia kalkulatzeko bada, argi geratzen da ode45 funtzioak ode15s-k baino askoz pauso gehiago behar dituela EDA honen zenbakizko soluzioa aurkitzeko.

Pospolo bat pizten dugunean, su-bola oso azkar hazten da tamaina kritiko bat hartzen duen arte. Tamaina hori hartzen duen momentutik aurrera, su-bolaren tamaina egonkortu egiten da, errektuntzaren ondorioz sugarraren barruan kontsumitzen den oxigeno kantitatea sugarraren gainazalean libre dagoen oxigeno kantitatearekin orekatzen baita. Fenomeno horri dagokion eredu sinplifikatu bat hauxe litzateke:

$$y' = y^2 - y^3, \quad y(0) = \delta \quad (2.3)$$

$y(t)$ aldagai eskalarra su-bolaren erradioa da, y^2 eta y^3 terminoak bolaren gainazalera eta bolumenetik eratortzen diren terminoak dira, hurrenez hurren. Problema honetako parametro kritikoa hasieran bolak duen erradioaren balioa da, δ letraz adierazi duguna eta oso balio txikia duena. Eta EDaren soluzioa aurkitu nahi

da δ -rekiko alderantziz proportzionala den denbora-tarte batean, hau da, $T = \left[0, \frac{2}{\delta}\right]$ denbora-tartean hain zuzen ere. EDA horren soluzio analitikoa aldagai-banaketaren metodoa erabiliz kalkula daiteke, eta ondorengo da:

$$\ln\left(\frac{1}{y}-1\right) + \frac{1}{y} = \ln\left(\frac{1}{\delta}-1\right) + \frac{1}{\delta} - t \quad (2.4)$$

MATLAB erabilita problema honen soluzioa lortu dugu su-bolaren hasierako balio ezberdinetarako. Hasierako balio bezala, $\delta = 10^{-2}, 10^{-3}, 10^{-4}$ balioak aukeratu ditugu eta ode45 eta ode15s funtzioek problema askatzeko ematen dituzten pauso kopuruak 2.1. taulan jaso ditugu. Bertan ikusten da, zenbat eta δ balioa txikiagoa izan, orduan eta pauso gehiago eman dituztela ode45 eta ode15s funtzioek problema askatzeko, bereziki ode45 funtzioak:

δ -ren balioak	ode45-ek ematen dituen pauso kopurua	ode15s-k ematen dituen pauso kopurua
	39	49
	314	77
	3.028	107

2.1. taula. MATLABeko ode45 eta ode15s funtzioek problema askatzen ematen dituzten pauso kopurua.

Lortu ditugun zenbakizko emaitzak ulertzeko eta problema ez-lineal bat denez, jacobitarra aztertu beharra daukagu. Kasu honetan, jacobitarraren balioa hau da:

$$J = 2y - 3y^2 \quad (2.5)$$

Eta jacobitarrak autobalio bakarra dauka, kasu honetan bera dena:

$$|J - \lambda I| = 0 \Rightarrow \lambda = J = 2y - 3y^2$$

λ autobalioaren ikurra aztertzen badugu, hau $y(t)$ -ren mendekoa dela ikusiko dugu:

- EDA ezgonkorra izango da $\lambda > 0$ denean, eta hori $y(t) \in \left(0, \frac{2}{3}\right)$ denean betetzen da.
- EDA egonkorra izango da, aldiz $\lambda < 0$ betetzen denean, eta hori $y(t) > \frac{2}{3}$ denean betetzen da.

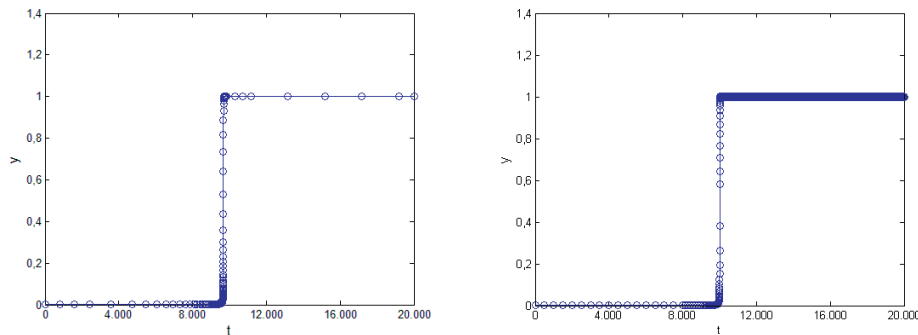
Shampine, Gladwell eta Thompson-en (2003: 78-80) lanean esaten da δ^{-1} ordenako t -ren balio baterako ekuazio diferentziala ez dela egonkorra, baina ezegonkortasun moderatua dela esaten da eta posible dela EDA zehaztasun nahikoaz ebaztea. Hori gertatzen dela ikusteko nahikoa da $\lambda = 2y - 3y^2$ autobalioak hartzen duen balioa aztertzea. Horrela, $y(t) \in \left(0, \frac{2}{3}\right)$ denean, $\lambda \in \left[0, \frac{1}{3}\right]$ betetzen da eta argi dago ez dela ezegonkortasun handiegia. Guk $\delta = 10^{-4}$ den kasurako, $y = \frac{2}{3}$ betetzen den aldiunea kalkulatu dugu, ikusi nahi izan baitugu δ^{-1} ordenakoa den aldiuneren baterako gertatzen dela ezegonkortasuna. Horretarako, EDAren emaitza zehatza ematen digun (2.4) adierazpenean $y = \frac{2}{3}$ eta $\delta = 10^{-4}$ balioak ordezkatu ditugu:

$$\ln\left(\frac{3}{2}-1\right) + \frac{3}{2} = \ln(10^4 - 1) + 10^4 - t \quad (2.6)$$

Eta (2.6) adierazpenetik t aldiunearen balioa askatu dugu:

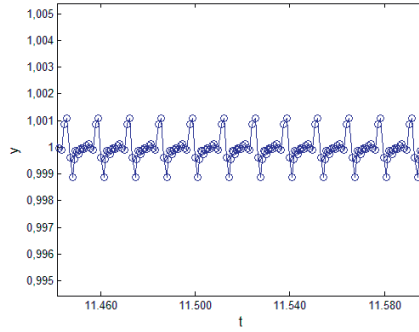
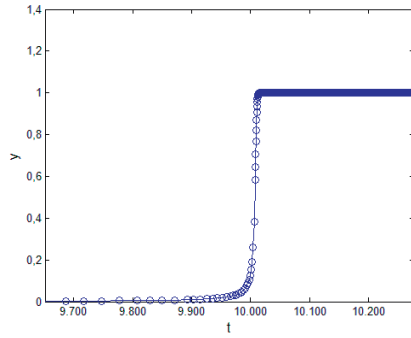
$$t = \ln(10^4 - 1) + 10^4 - \ln\left(\frac{1}{2}\right) - \frac{3}{2} = 10.008,403 \approx O(\delta^{-1}) \quad (2.7)$$

(2.7) adierazpenean ikusten den moduan, $\delta = 10^{-4}$ denean $y = \frac{2}{3}$ betetzen den aldiunea $t = 10.008,403 \approx \frac{1}{\delta}$ da. Hau da, δ^{-1} balioaren ordenakoa den t -ren balio batentzat da ezegonkorra EDA.



2.1. irudia. Ode15s (ezkerrean) eta ode45 (eskuinean) erabilia

(2.3) problemaren emaitza $\delta = 10^{-4}$ denean.



2.2. irudia. (2.3) problemaren emaitzaren xehetasuna ode45 erabilia (zoom-a aplikatuta).

2.1. irudian jaso ditugu ode15s eta ode45 funtzioek problema hau askatzean ematen dituzten pausoak. 2.2. irudian, ode45 funtzioak ematen digun zenbakizko emaitzaren xehetasuna irudikatu dugu. Horretarako, 2.1. irudiko ode45 funtzioaren emaitzari zoom-a aplikatu diogu: alde batetik, $T = [0, 2\delta^{-1}]$ denbora-tartearen lehenengo partean, eta, bestetik, bigarrenean. Nabaria da, problema honen zurruntasuna $t \in [\delta^{-1}, 2\delta^{-1}]$ denbora-tartean gertatzen dela, hau da, integrazioko denbora-tartearen bigarren partean. Portaera horren azalpena (2.4) adierazpenak emana datorren EDaren emaitza zehatzean aurki dezakegu:

$$\ln\left(\frac{1}{y} - 1\right) + \frac{1}{y} = \ln\left(\frac{1}{\delta} - 1\right) + \frac{1}{\delta} - t \Rightarrow S_1 = S_2 - t \quad (2.8)$$

$$\text{non: } \begin{cases} S_1 = \ln\left(\frac{1}{y} - 1\right) + \frac{1}{y} \\ S_2 = \ln\left(\frac{1}{\delta} - 1\right) + \frac{1}{\delta} \end{cases}$$

2.1. irudian ikusten den moduan, denbora-tarte bat igaro ostean problemaren soluzioak y -k baterantz jotzen du. Ikusi nahi duguna da ea S_1 eta S_2 batugaien artean ezberdintasunik badagoen eta, egotekotan, ea zenbatekoa den. Horretarako bi limite hauek kalkulatu ditugu:

$$\begin{cases} \lim_{y \rightarrow 1} S_1 = \lim_{y \rightarrow 1} \ln\left(\frac{1}{y} - 1\right) + \frac{1}{y} = -\infty \\ \lim_{\delta \rightarrow 0} S_2 = \lim_{\delta \rightarrow 0} \ln\left(\frac{1}{\delta} - 1\right) + \frac{1}{\delta} = \infty \end{cases} \quad (2.9)$$

(2.9) adierazpeneko limiteek erakusten dute ezen hasierako balioa zerotik zenbat eta hurbilago egon, emaitza zehatzeko bi batugaien artean ezberdintasun handiagoa dagoela. Emaitza zehatzean t -ren balioa askatzen badugu eta $(\delta, y) \rightarrow (0, 1)$ denean limitearen balioa kalkulatzeko badugu, zera daukagu:

$$\lim_{(\delta, y) \rightarrow (0, 1)} t = \lim_{(\delta, y) \rightarrow (0, 1)} \ln\left(\frac{1}{\delta} - 1\right) + \frac{1}{\delta} - \ln\left(\frac{1}{y} - 1\right) - \frac{1}{y} = \infty - (-\infty) = +\infty \quad (2.10)$$

(2.10) limiteak erakusten du, $t \rightarrow \infty$ betetzen denean, emaitza zehatzeko bi batugaien arteko diferentzia handia egiten dela, $S_1 \rightarrow -\infty$ eta $S_2 \rightarrow +\infty$ betetzen baita. Beraz, esku artean daukagun problema honetan zurruntasuna integrazioko denbora-tartearen bigarren erdialdean gertatzen da eta problemaren hasierako balioen mendekoa da: hasierako balioa eta emaitza zehatzeko bi batugaien arteko diferentzia handia existitzea eragiten duena. Hau da, zenbat eta hasierako balioa txikiagoa izan, problema orduan eta zurrunago bihurtzen da.

2.1.2. Emaitza bera duten bi EDAREN portaera

Zurruntasuna aztertzerakoan, EDAREN emaitzak ez duela zerikusirik ikusiko dugu, hau da, bi EDAK emaitza bera izan arren, gerta daitekeela berauetako bat zurruna izatea eta bestea ez. Horretarako, Lambert-en (1991: 213-216) erreferentzian agertzen diren bi EDA sistema hauek kontsideratuko ditugu:

Lehenengo sistema:

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} -2 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} 2\sin t \\ 2(\cos t - \sin t) \end{pmatrix}, \quad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad (2.11)$$

Lehenengo sistemaren emaitza orokorra hau da:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = k_1 e^{-t} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + k_2 e^{-3t} \begin{pmatrix} 1 \\ -1 \end{pmatrix} + \begin{pmatrix} \sin t \\ \cos t \end{pmatrix} \quad (2.12)$$

Eta (2.11) EDAK dituen hasierako balioetarako, hau da lehenengo sistemaren emaitza:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = 2e^{-t} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} \sin t \\ \cos t \end{pmatrix} \quad (2.13)$$

Bigarren sistema:

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} -2 & 1 \\ 998 & -999 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} 2\sin t \\ 999(\cos t - \sin t) \end{pmatrix}, \quad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad (2.14)$$

Bigarren sistemaren emaitza orokorra honakoa da:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = k_1 e^{-t} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + k_2 e^{-1.000t} \begin{pmatrix} 1 \\ -998 \end{pmatrix} + \begin{pmatrix} \sin t \\ \cos t \end{pmatrix} \quad (2.15)$$

Eta bigarren sistemak dituen hasierako balioetarako, lehenengo sistemak duen emaitza bera dauka, hau da, (2.13) adierazpenak emana dator bi sistema hauen emaitza.

Bi sistemen autobalioak ezberdinak dira, ordea. Lehenengo sistemaren autobalioak $\lambda_{1,1} = -1, \lambda_{1,2} = -3$ dira, eta bigarren sistemarenak $\lambda_{2,1} = -1, \lambda_{2,2} = -1.000$. Bigarren sistemako $\lambda_{2,2} = -1.000$ autobalioa *handia* da eta autobalio honetatik eratortzen da zurruntasuna. Aukeratu ditugun hasierako balioetarako, bigarren sistemako $\lambda_{2,2} = -1.000$ autobalioari dagokion batugaia emaitzan agertzen ez den arren, bigarren sistema zurrunda dela ikusiko dugu, zurruntasun-erradioa handia duen sistema baita bigarrena: $\rho = 1.000$. Hau da, Lambert-ek (1973) ematen duen sistema zurrundaren definizioa bete egiten du sistema honek, baina emaitzan ez dira ageri magnitude ezberdinetako bi batugai ezberdin, emaitzan ageri den autobalioa bakarra baita eta, gainera, bigarren sistemaren kasuan autobalio bietatik txikiena da emaitzan agertzen dena.

Bi sistema hauen emaitza $T = [0, 10]$ denbora-tartean aurkitu dugu MATLABeko ode45 eta ode15s funtzioak erabilita. Ode15s funtzioak 41 pauso ematen ditu lehenengo sistemaren soluzioa aurkitzeko eta 48 pauso bigarren sistemarena aurkitzeko. Ode45 funtzioak 25 pauso behar izan ditu lehenengo sistemaren emaitza kalkulatzeko eta 3015 bigarren sistemarena aurkitzen. Beraz, nahiz eta sistema biek emaitza bera duten, bigarren sistema zurrunda da eta lehenengo sistema ez da zurrunda. Horrek esan nahi du gerta daitekeela emaitza bera duten EDak bata zurrunda izatea eta bestea ez.

2.1.3. Zurruntasun-erradio bera duten bi EDaren portaera

Lambert-ek (1973) dio ere, zurruntasun-erradio handia duten EDA sistema bat zurrunda dela. Baina, Lambert-en lanean (1991: 217-218) ematen den EDA sistema bikote batean argi ikusten da zurruntasun-erradio bereko bi EDA emanik bata zurrunda izatea eta bestea ez izatea gerta daitekeela. Zurruntasun-erradio bereko bi EDA sistema sortzea geuk egin dezakegun lana da. Sistema bat $\lambda_{1,1} = -1$ eta $\lambda_{1,2} = -1.000$ autobalioak sortuko dugu, eta bigarrena $\lambda_{2,1} = -1$ eta $\lambda_{2,2} = -0,001$. Biek daukate zurruntasun-erradio bera: $\rho = 1.000$. Aipatu autobalioak dituzten EDA sistemak hauek izan daitezke:

Hirugarren sistema:

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -0,001 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad (2.16)$$

Laugarren sistema:

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1.000 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad (2.17)$$

MATLABeko ode45 eta ode15s funtzioak erabiliz, bi sistema hauen emaitza $T=[0,10]$ denbora-tartean kalkulatzeko badugu ikusiko dugu ode15s funtzioak 42 pauso behar dituela hirugarren sistema deitu duguna askatzeko, eta ode45-ek 14 pauso. Laugarren sistema askatzeko, aldiz, ode15s-k 109 pauso eman ditu eta ode45-ek 3.027. Nahiz eta biek zurruntasun-erradio bera eduki, laugarren sistema izendatu duguna zurruna da eta hirugarrena ez.

2.1.4. Zurruntasunari buruzko zenbait ondorio

Ikusi ditugun adibideak kontuan hartuz, zurruntasuna ondorengo kasuetakoren bat ematen denean gertatzen dela ondoriozta daiteke:

- Soluzio zehatzean magnitude ezberdinetako batugaiak agertzen direnean.
- EDAk magnitude ezberdinetako autobalioak dituenen. Magnitude-ezberdintasun hori autobalioen parte errealean edo parte irudikarian gerta daiteke (nahiz eta parte irudikarietako haren ez dugun landu).
- Pauso-tamainaren eta autobalioaren arteko biderketa, $h\lambda$, zenbakizko metodoaren egonkortasun-eremuan eror dadin, autobalioa pauso-tamaina txiki batez biderkatzea eskatzen duen zenbakizko metodo bat darabilgunean.
- Askatzen ari garen EDAREN emaitza aurkitzeko darabilgun zenbakizko metodoak pauso-tamaina oso txikiak erabili behar dituenen.

2.2. ZENBAIT EKUAZIO DIFERENTZIAL ZURRUNEN SOLUZIOAK

Normalean zenbakizko metodo esplizituek metodo inplizituek baino egonkortasun-eremu txikiagoa dute. Adibide gisa, hor daukagu lehenengo kapituluaren ikusi ditugun Euler-en metodo esplizituaren eta inplizituaren egonkortasun-eremuen kasua, zeinetan Euler-en metodo inplizituak esplizituak baino egonkortasun-eremu handiagoa duen. Zurruntasunari buruz ikusi ditugun ondorioak kontuan hartuta, beraien egonkortasun-eremu ez hain handiak tarteko, zenbakizko metodo esplizituek arazoak izaten dituzte EDA zurrunak askatzerakoan.

Kontsidera dezagun, ondoko EDA:

$$y' = -40(y - \cos t), \quad y(0) = 0 \quad (2.18)$$

EDA horren autobalioa $\lambda = -40$ da eta EDAREN emaitza analitikoa hau da:

$$y(t) = \frac{1.600}{1.601} \cos t + \frac{40}{1.601} \sin t - \frac{1.600}{1.601} e^{-40t}$$

Zenbakizko metodo baten egonkortasun absolutuko eremua, plano konplexuko eremua da, zeinak metodoa test-ekuazioari aplikatu ostean egonkortasun absolutuko baldintza beteko duen. Horrela, Euler-en metodo espliziturako, ondorengo baldintza lortu genuen lehenengo kapituluan:

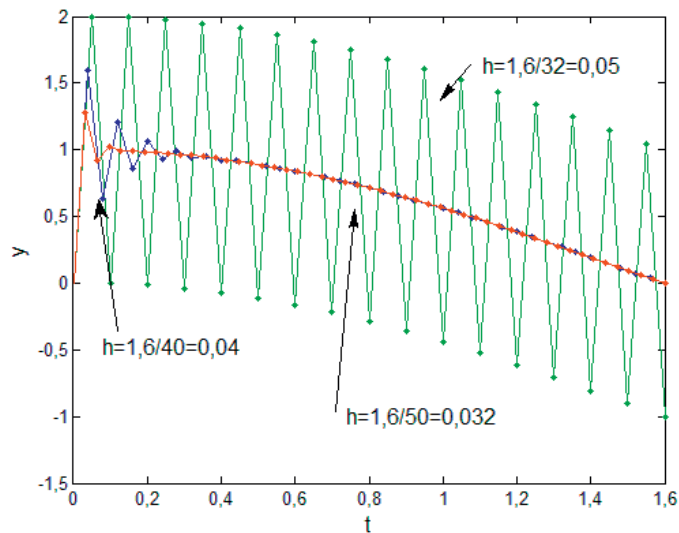
$$|1 + h\lambda| \leq 1 \quad (2.19)$$

λ negatiboa den kasuan, h pauso-tamainak bete beharreko egonkortasun absolutuko baldintza honela geratzen zaigu:

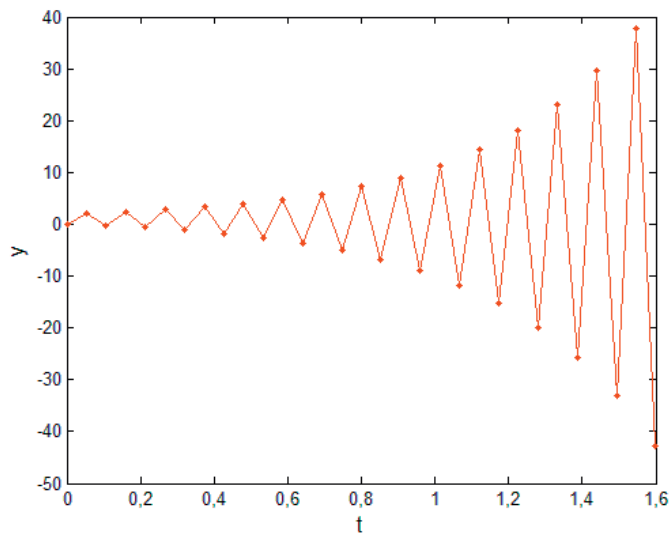
$$h \leq -\frac{2}{\lambda} \quad (2.20)$$

(2.18) ekuazioa $T = [0, 1, 6]$ tartean Euler-en metodo esplizitua erabilia ebatzi nahi badugu, egonkortasun absolutuko baldintza bete dezan pauso-tamainak bete beharreko baldintza $h \leq 0,05$ da. 2.3. irudian Euler-en metodo esplizituan $h = 0,05$, $h = 0,04$ eta $h = 0,032$ pauso-tamainak erabilia lortu ditugun (2.18) EDAREN zenbakizko emaitzak irudikatu ditugu. Hurrenez hurren 32, 40 eta 50 pauso emanda lortu ditugun emaitzak dira hauek. $h = 0,032$ pauso-tamainarekin lortzen den emaitzak 2.5. irudian marraztu dugun emaitza zehatzaren antz handia dauka. $h = 0,05$ balioarekin lortzen den emaitzak emaitza zehatzaren inguruan oszilatzen duela ikusten da, eta antzekoa gertatzen da $h = 0,04$ pauso-tamainarekin, nahiz eta oraingoan oszilazioak bakarrik denbora-tartearen hasieran gertatzen diren eta txikiagoak diren.

Egonkortasun absolutuko baldintza ez betetzeak, hau da $h > 0,05$ aukeratzeak, emaitzan duen eragina zein den 2.4. irudian ikus daiteke. Irudi honetako zenbakizko emaitza $h = 0,053$ pauso-tamaina erabilia lortu da. Era horretan lortu dugun emaitza oszilakorra da, denboran aurrera egin ahala oszilazioak gero eta handiagoak izanik eta emaitza ezegonkortzea lortuz.



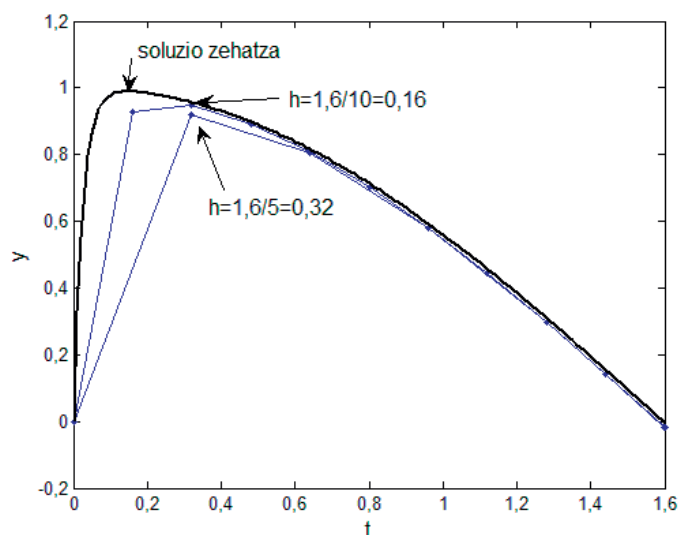
2.3. irudia. (2.18) ekuazio diferentzial arruntaren zenbakizko emaitza Euler-en metodo esplizitua erabilia.



2.4. irudia. (2.18) ekuazio diferentzial arruntaren zenbakizko emaitza Euler-en metodo esplizituan $h = 0,053$ pauso-tamaina erabilia.

Problema bera Euler-en metodo implizituarekin ebatz dezakegu. Metodo hori egonkorra da plano konplexuko ezkerreko osoan eta, edozein pauso-tamaina aukeratuta ere, egonkortasun absolutuko baldintza bete egingo da. Gertatuko dena da, zenbat eta pauso gehiago erabili lortuko dugun emaitza zehatzagoa izango dela

baina, egonkortasun-eremu zabalari esker pauso gutxi behar ditu emaitza hurbildua emaitza analitikoaren antzekoa izan dadin. 2.5. irudian (2.18) ekuazio diferentzial arruntaren zenbakizko emaitza irudikatu dugu Euler-en metodo inplizituan 5 pauso eta 10 pauso emanda. Hurrenez hurren $h = 0,32$ eta $h = 0,16$ pauso-tamainak erabili ditugu problema hau askatzeko Euler-en metodo inplizituan. Era horretan lortu dugun zenbakizko emaitza, emaitza zehatzaren oso antzekoa da.

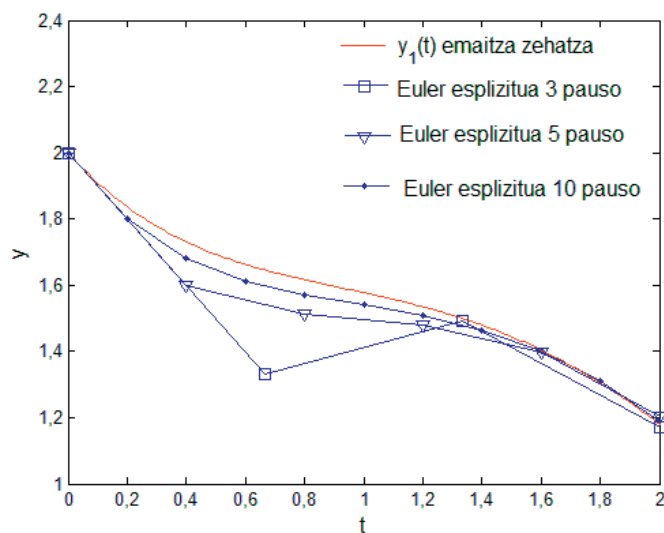


2.5. irudia. (2.18) ekuazio diferentzial arruntaren emaitza Euler-en metodo inplizitua erabilia.

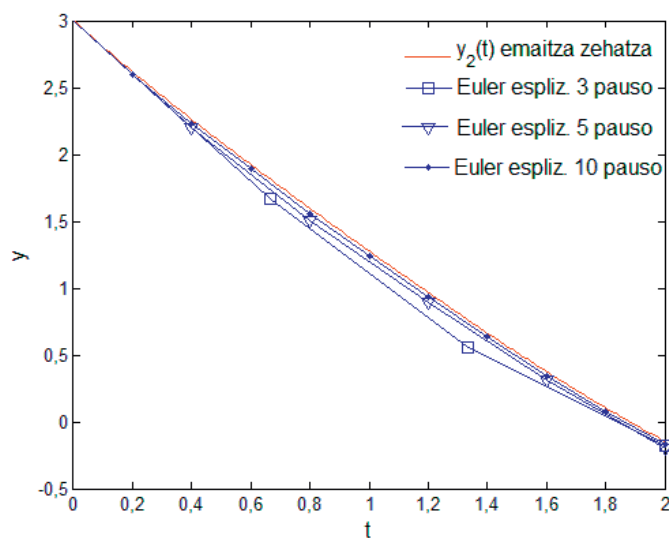
Euler-en bi metodoen gaitasuna berriz neurtu dugu, oraingoan (2.11) eta (2.14) sistemak ebatziz. (2.11) eta (2.14) sistemak aztertu ditugunean esan dugu lehenengoa ez dela zurruna eta bigarrena, aldiz, zurruna dela. Bi sistema horiek Euler-en metodo esplizitua eta inplizitua erabilia ebatzi ditugu $T = [0, 2]$ denbora-tartean.

(2.11) sistemaren autobalioak $\lambda_{1,1} = -1$ eta $\lambda_{1,2} = -3$ direla ikusi dugu. Egonkortasun absolutuko baldintza beteko duen pauso-tamaina kalkulatzeko balio absolutuan handiena den autobalioari erreparatu behar diogu, hau da, $\lambda_{1,2} = -3$ balioari eta pauso-tamaina $h \leq 2/3$ izan beharko da Euler-en metodo esplizituko egonkortasun absolutuko baldintza bete dadin. Bigarren sistemaren autobalioak $\lambda_{2,1} = -1$ eta $\lambda_{2,2} = -1.000$ dira eta Euler-en metodo esplizituko egonkortasun absolutuko baldintza beteko duen pauso-tamaina $h \leq 2/1.000$ izango da kasu honetan. Beraz, Euler-en metodo esplizituarekin sistema hauek askatzean, lehenengo kasuan gutxienez 3 pauso eman beharko ditugu, eta bigarren kasuan 1.000 pauso. Euler-en metodo inplizituko egonkortasun-eremua plano konplexuko ezkerralde osoa denez, metodo honetan, edozein pauso-tamaina aukeratuta ere, egonkortasun absolutuko baldintza bete egingo da.

2.6. eta 2.7. irudietan (2.11) sistemaren emaitzako lehen eta bigarren osagaiak irudikatu ditugu. Irudi bietan emaitza zehatza eta Euler-en metodo esplizituarekin 3, 5 eta 10 pauso emanda lortu ditugun zenbakizko emaitzak irudikatu dira. Ikusten denez, kasu honetan Euler-en metodo esplizituko 3 pauso ere nahikoak dira emaitza zehatzaren itxura handia duen zenbakizko emaitza lortzeko.

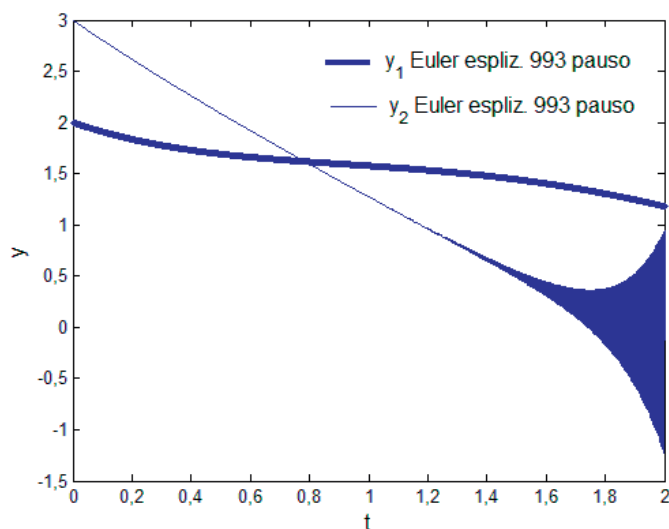


2.6. irudia. (2.11) sistemaren y_1 emaitza Euler-en metodo esplizituarekin 3, 5 eta 10 pauso.

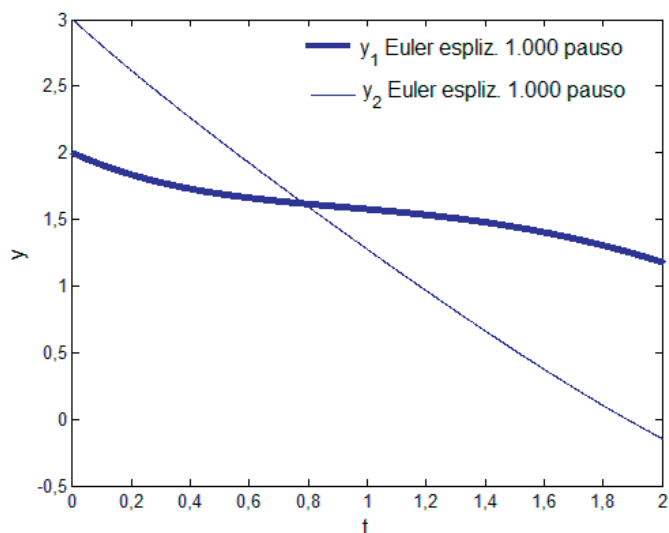


2.7. irudia. (2.11) sistemaren y_2 emaitza Euler-en metodo esplizituarekin 3, 5 eta 10 pauso emanda.

2.8. irudian Euler-en metodo esplizituarekin 993 pauso emanda (2.14) problema askatzean lortu dugun zenbakizko emaitza irudikatu dugu, eta 2.9. irudian problema bera Euler-en metodo esplizituarekin 1.000 pauso emanda askatzean lortu ditugunak. 1.000 pausok egonkortasun absolutuko baldintza betetzen dutenez, 2.9. irudiko emaitzak ondo errepikatzen du problema honen emaitza zehatza. Aldiz, 993 pausorekin lortu dugun zenbakizko emaitzak ez du emaitza zehatza behar den besteko zehaztasunez errepikatzen.

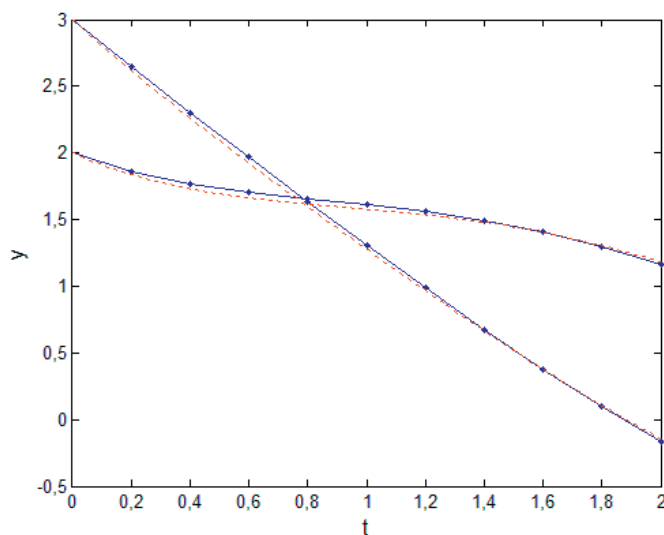


2.8. irudia. (2.14) sistemaren emaitza Euler-en metodo esplizituarekin 993 pauso emanda.



2.9. irudia. (2.14) sistemaren emaitza Euler-en metodo esplizituarekin 1.000 pauso emanda.

(2.14) problema askatzeko Euler-en metodo inplizitua darabilgunean, berriz, 10 pauso erabilia emaitza zehatzaren oso antzekoa den zenbakizko emaitza lortzen dugula ikus dezakegu 2.10. irudian. Irudi horretan (2.14) problemaren emaitzaren osagai biak irudikatu dira: marratxodun lerroz emaitza zehatzaren osagaiak eta lerro jarraituz zenbakizko emaitzaren osagaiak.



2.10. irudia. (2.14) sistemaren emaitza Euler-en metodo inplizituarekin 10 pauso emanda.

3. Runge-Kutta metodoak

Runge-Kutta metodoak pauso bakoitzean pauso horretako tarteko zenbait puntutako deribatua darabilten metodoak dira. Horri etapa anitz izatea deritzo eta Runge-Kutta metodoak pauso bakarrekoak eta etapa anitzeko metodoak dira. Azken ezaugarri hori da, hain zuzen, Runge-Kutta metodoei izena ematen diena, etapa anitzeko metodo edo s etapako metodo izenez ezagutzen baitira. Ideia hau Runge-ri (1895) zor diogu. Ekarpen gehiago Heun-ek (1900) eta Kutta-k (1901) egin zituzten. Kutta izan zen laugarren ordenako Runge-Kutta metodoak guztiz karakterizatu zituen eta bosgarren ordenako lehen Runge-Kutta metodoak proposatu zituen. Huta-k, aldiz, besteak beste seigarren ordenako Runge-Kutta metodoak proposatu zituen. Ordenagailu digitalak gure artean barneratu direnetik, interes handia dago Runge-Kutta metodoetan eta ikertzaile askok lagundu dute metodo hauen hedapenean. Hasiera baten ikerketa guztiak Runge-Kutta metodo esplizituetan zentratu baziren ere, gaur egun Runge-Kutta metodo implizituak ere interesekoak dira. Azken horiek, egonkortasun-eremu handiak dituztenez, ekuazio diferentzial zurrinak ebazteko oso onak baitira.

3.1. METODORAKO SARRERA

Hasierako baliodun EDA bat askatzeko s etapako Runge-Kutta metodoak adierazpen hauxe dauka:

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i k_i \quad (3.1)$$

Non:

$$k_i = f(t_n + c_i h, y_n + h \sum_{j=1}^s a_{ij} k_j), \quad i = 1, 2, \dots, s \quad (3.2)$$

Runge-Kutta metodoetako a_{ij}, b_i, c_i konstanteak koefizienteen taula batean jasotzen dira, eta taula horri Butcher-en taula esaten zaio (ikus 3.1. taula).

c_1	a_{11}	a_{12}	$\cdots$	a_{1s}
c_2	a_{21}	a_{22}	$\cdots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\cdots$	a_{ss}
	b_1	b_2	$\cdots$	b_s

3.1. taula. Runge-Kutta metodoko Butcher-en taula.

Butcher-en taula era matritzialean ere idatz daiteke. Horretarako, s dimentsiodun b eta c bektoreak eta $s \times s$ dimentsiodun A matrizea definitzen dira ondoko eran:

$$b = [b_1, b_2, \dots, b_s]^T, c = [c_1, c_2, \dots, c_s]^T, A = [a_{ij}]$$

Eta era matritzialean emandako Butcher-en taula hauze litzateke:

c	A
	b^T

3.2. taula. Era matritzialean emandako Butcher-en taula.

c bektoreko osagai bakoitzak pausoaren barneko etapa adierazten du eta b^T bektorea pisu-bektore bat da. A matrizeak etapa bakoitzak aurreko etapekin (metodoa esplizitua denean) edo etapa guztiekin (metodo implizituaren kasuan) duen mendekotasuna adierazten du. Runge-Kutta metodo bat esplizitua, implizitua edo erdi-implizitua izan daiteke A matrizeko elementuen arabera:

- Metodoa esplizitua da A matrizea hertsiki behe-triangeluarra denean. Hau da: $a_{ij} = 0$ $j \geq i$ eta $i = 1, 2, \dots, s$ balioetarako.
- Metodoa erdi-implizitua da A matrizea behe-triangeluarra denean. Hau da: $a_{ij} = 0$ $j > i$ eta $i = 1, 2, \dots, s$ balioetarako.
- Metodoa implizitua da A matrizea ez denean behe-triangeluarra. Hau da: $a_{ij} \neq 0$ $j > i$ balioen batentzat betetzen denean.

Runge-Kutta metodoa esplizitua denean, (3.2) adierazpenak forma hau hartzen du:

$$k_i = f(t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j), \quad i = 1, 2, \dots, s \quad (3.3)$$

Adibidez, 4 etapako Runge-Kutta metodo esplizitu batek 4 konstante c_i , 4 konstante b_i eta hamasei konstante a_{ij} ditu, zeinetatik hamar metodoa esplizitua izateagatik nuluak izango diren. Era honetako metodoari dagokion Butcher-en taula 3.3. taula litzateke:

0	0	0	0	0
c_2	a_{21}	0	0	0
c_3	a_{31}	a_{32}	0	0
c_4	a_{41}	a_{42}	a_{43}	0
	b_1	b_2	b_3	b_4

3.3. taula. 4 etapako Runge-Kutta metodo esplizitu baten Butcher-en taula.

Oro har, a_{ij}, c_i koefizienteek errenkadaka ondoko erlazioa betetzen dute:

$$c_i = \sum_{j=1}^s a_{ij}, \quad i = 1, 2, \dots, s \quad (3.4)$$

(3.4) erlazioaz gain, hurrengo atalean ikusiko ditugun ordenari lotutako zenbait erlazio ere bete behar dituzte a_{ij}, b_i, c_i koefizienteek.

3.2. ORDENA-BALDINTZAK

Runge-Kutta metodoen mozketak-errore lokalari dagokion adierazpena nahiko konplexua da. 3 etapako Runge-Kutta metodo esplizitu batek hirugarren ordena izan dezan bete behar dituen ordena-baldintzak zeintzuk diren ikusiz hasiko gara. Horretarako, pauso bakarreko zenbakizko metodoen mozketak-errore lokalaren adierazpena gogoratuko dugu:

$$LTE = y(t_{n+1}) - y_{n+1}^* \quad (3.5)$$

y_{n+1}^* balioa kokapen-bereganaketa eginda kalkulatzeko delarik. $y(t_{n+1})$ terminoaren Taylor-en garapena kontsideratuko dugu:

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2} y''(t_n) + \frac{h^3}{6} y'''(t_n) + O(h^4) \quad (3.6)$$

$y' = f(t, y)$ funtzioa nahi adina aldiz deribagarria dela suposatuko dugu, haren deribatu partzialak existitu eta jarraituak izateko eran. Eta ondoko idazkera laburtuak erabiliko ditugu bai $f(t, y)$ funtzioarentzat baita haren deribatu partzialentzat ere:

$$f = f(t, y), f_t = \frac{\partial f}{\partial t}, f_y = \frac{\partial f}{\partial y}, f_{tt} = \frac{\partial^2 f}{\partial t^2}, f_{yy} = \frac{\partial^2 f}{\partial y^2}, f_{ty} = f_{yt} = \frac{\partial^2 f}{\partial y \partial t}, \text{ etab.} \quad (3.7)$$

Eta (3.6) adierazpenean agertzen diren deribatuak hauexek dira:

$$\begin{aligned} y'(t) &= f \\ y''(t) &= f_t + f_y y' = f_t + f_y f \\ y'''(t) &= f_{tt} + f_{ty} f + f(f_{yt} + f_{yy}) + f_y(f_t + f_y f) \\ &= f_{tt} + 2f_{ty} f + f^2 f_{yy} + f_y(f_t + f_y f) \end{aligned} \quad (3.8)$$

(3.8) adierazpenean erabili dugun notazioa F eta G funtzioak erabilita laburtuko dugu:

$$F = f_t + f_y f, \quad G = f_{tt} + 2f_{ty} f + f^2 f_{yy} \quad (3.9)$$

Era horretan (3.6) adierazpenak emandako Taylor-en garapena honela geratzen da:

$$y(t_{n+1}) = y(t_n) + hf + \frac{1}{2} h^2 F + \frac{1}{6} h^3 (F f_y + G) + O(h^4) \quad (3.10)$$

3 etapa dituen Runge-Kutta metodo baten kasuan, y_{n+1}^* balioa honela emana dator:

$$y_{n+1}^* = y(t_n) + h \sum_{i=1}^s b_i k_i \quad (3.11)$$

non:

$$k_i = f(t_n + c_i h, y(t_n) + h \sum_{j=1}^{i-1} a_{ij} k_j), \quad i = 1, 2, 3 \quad (3.12)$$

(3.12) adierazpenaren bidez emanak dauden k_i terminoen garapenak eginez eta beraietan F eta G funtzioen adierazpenak erabiliz, ondoko adierazpen laburtuetara iristen gara:

$$\begin{cases} k_1 = f \\ k_2 = f + h c_2 F + \frac{1}{2} h^2 c_2^2 G + O(h^3) \\ k_3 = f + h c_3 F + h^2 \left(c_2 a_{32} F f_y + \frac{1}{2} c_3^2 G \right) + O(h^3) \end{cases} \quad (3.13)$$

(3.13) adierazpenak (3.11) adierazpenean ordezkatzeko baditugu, zera daukagu:

$$y_{n+1}^* = y(t_n) + h(b_1 + b_2 + b_3)f + h^2(b_2c_2 + b_3c_3)F + \frac{1}{2}h^3[2b_3c_2a_{32}Ff_y + (b_2c_2^2 + b_3c_3^2)G] + O(h^4) \quad (3.14)$$

(3.10) eta (3.14) adierazpenak, mozketaren errore lokalaren adierazpenean ordezkatzeko, zera daukagu:

$$LTE = hf(1 - (b_1 + b_2 + b_3)) + h^2F\left(\frac{1}{2} - (b_2c_2 + b_3c_3)\right) + h^3\left(\frac{1}{6}(Ff_y + G) - \frac{1}{2}(2b_3c_2a_{32}Ff_y + (b_2c_2^2 + b_3c_3^2)G)\right) + O(h^4) \quad (3.15)$$

Eta (3.15) adierazpeneko hirugarren batua berrordenatzen badugu, hauxe daukagu:

$$LTE = hf(1 - (b_1 + b_2 + b_3)) + h^2F\left(\frac{1}{2} - (b_2c_2 + b_3c_3)\right) + h^3\left(Ff_y\left(\frac{1}{6} - b_3c_2a_{32}\right) + G\left(\frac{1}{6} - \frac{1}{2}(b_2c_2^2 + b_3c_3^2)\right)\right) + O(h^4) \quad (3.16)$$

Eta (3.16) adierazpenetik Runge-Kutta metodoetako ordena-baldintzetara iristen gara:

- Metodoaren ordena bat izango da, $1 - (b_1 + b_2 + b_3) = 0 \Rightarrow \sum_{i=1}^3 b_i = 1$ betetzen denean.
- Metodoa bigarren ordenakoa izango da, aurreko baldintzaz gain, beste hau ere betetzen bada:

$$\frac{1}{2} - (b_2c_2 + b_3c_3) = 0 \Rightarrow \sum_{i=1}^3 b_i c_i = \frac{1}{2}$$

- Metodoa hirugarren ordenakoa izango da aurreko bi adierazpenez gain (lehen eta bigarren ordenetarako bete behar direnez gain), beste bi baldintza hauek ere betetzen badira:

$$\begin{cases} \frac{1}{6} - \frac{1}{2}(b_2c_2^2 + b_3c_3^2) = 0 \Rightarrow \sum_{i=1}^3 b_i c_i^2 = \frac{1}{3} \\ \frac{1}{6} - b_3c_2a_{32} = 0 \Rightarrow \sum_{i=1}^3 b_i \left(\sum_{j<i} a_{ij}c_j \right) = \frac{1}{6} \end{cases} \quad (3.17)$$

Oro har, s etapa dituen Runge-Kutta metodo baten mozketaren erroreak lokalak forma hau dauka:

$$LTE = \sum_{i=1} h^{i+1} \left(\sum_{r=i+1} \frac{1}{\sigma_{r,q}} \left(\frac{1}{\gamma_{r,q}} - \phi_{r,q} \right) D_{r,q}(y(t_n)) \right) \quad (3.18)$$

$\sigma_{r,q}$ eta $\gamma_{r,q}$ konstanteak izanik, $\phi_{r,q}$ konstantea a_{ij} , b_i eta c_i koefizienteen mendekoa izanik eta $D_{r,q} y(t)$ funtzioaren r ordenako deribatua agertzen diren batugaiak izanik. 3.4. taulan 1-5 ordenei dagozkien $\sigma_{r,q}$, $\gamma_{r,q}$, $\phi_{r,q}$ eta $D_{r,q}$ -ren balioak jaso dira.

Runge-Kutta metodo bat p ordenakoa da, ondorengo baldintza betetzen bada:

$$\frac{1}{\sigma_{r,q}} \left(\frac{1}{\gamma_{r,q}} - \phi_{r,q} \right) = 0, \quad \forall q, r = 1, 2, \dots, p \quad (3.19)$$

Eta hori gertatzen denean, metodoaren mozketaren erroreak lokala $(p+1)$ ordenakoa da eta hauxe da:

$$LTE = h^{p+1} \sum_{r=p+1} \frac{1}{\sigma_{r,q}} \left(\frac{1}{\gamma_{r,q}} - \phi_{r,q} \right) D_{r,q}(y(t_n)) + O(h^{p+2}) \quad (3.20)$$

$\sigma_{r,q}$, $\gamma_{r,q}$ eta $\phi_{r,q}$ konstanteen balioak (3.19) berdintzan ordezkatzuz gero, Runge-Kutta metodoa p ordenakoa izan dadin a_{ij} , b_i eta c_i koefizienteek bete behar dituzten baldintzetara iristen gara. Hauxe da, hain zuzen, metodoa bosgarren ordenakoa izateko bete behar dituen (3.21) adierazpeneko baldintzak lortzeko egin duguna.

$$\begin{aligned}
1. \phi_{1,1} &= 1 & 5. \phi_{4,1} &= \frac{1}{4} & 9. \phi_{5,1} &= \frac{1}{5} & 13. \phi_{5,5} &= \frac{1}{20} \\
2. \phi_{2,1} &= \frac{1}{2} & 6. \phi_{4,2} &= \frac{1}{8} & 10. \phi_{5,2} &= \frac{1}{10} & 14. \phi_{5,6} &= \frac{1}{20} \\
3. \phi_{3,1} &= \frac{1}{3} & 7. \phi_{4,3} &= \frac{1}{12} & 11. \phi_{5,3} &= \frac{1}{15} & 15. \phi_{5,7} &= \frac{1}{40} \\
4. \phi_{3,2} &= \frac{1}{6} & 8. \phi_{4,4} &= \frac{1}{24} & 12. \phi_{5,4} &= \frac{1}{30} & 16. \phi_{5,8} &= \frac{1}{60} \\
& & & & & & 17. \phi_{5,9} &= \frac{1}{120}
\end{aligned} \tag{3.21}$$

r	q	$\sigma_{r,q}$	$\gamma_{r,q}$	$\phi_{r,q}$	$D_{r,q}$
1	1	1	1	$\sum_{i=1}^s b_i$	f
2	1	1	2	$\sum_{i=1}^s b_i c_i$	ff
3	1	2	3	$\sum_{i=1}^s b_i c_i^2$	$f''(f, f)$
3	2	1	6	$\sum_{i,j=1}^s b_i \left(\sum_{j<i} a_{ij} c_j \right)$	fff
4	1	6	4	$\sum_{i=1}^s b_i c_i^3$	$f'''(f, f, f)$
4	2	1	8	$\sum_{i,j=1}^s b_i c_i \left(\sum_{j<i} a_{ij} c_j \right)$	$f''(f, ff)$
4	3	2	12	$\sum_{i,j=1}^s b_i \left(\sum_{j<i} a_{ij} c_j^2 \right)$	$ff''(f, f)$
4	4	1	24	$\sum_{i,j,k=1}^s b_i \left(\sum_{j<i} a_{ij} \left(\sum_{k<j} a_{jk} c_k \right) \right)$	$ffff$

3.4. taula. 1-5 ordenen dagozkien $\sigma_{r,q}$, $\gamma_{r,q}$, $\phi_{r,q}$ eta $D_{r,q}$ -ren balioak.

r	q	$\sigma_{r,q}$	$\gamma_{r,q}$	$\phi_{r,q}$	$D_{r,q}$
5	1	24	5	$\sum_{i=1}^s b_i c_i^4$	$f^{(4)}(f, f, f, f)$
5	2	2	10	$\sum_{i,j=1}^s b_i c_i^2 \left(\sum_{j<i} a_{ij} c_j \right)$	$f^{(3)}(f, f, ff)$
5	3	2	15	$\sum_{i,j=1}^s b_i c_i \left(\sum_{j<i} a_{ij} c_j^2 \right)$	$f''(f, f''(f, f))$
5	4	1	30	$\sum_{i,j,k=1}^s b_i c_i \left(\sum_{j<i} a_{ij} \left(\sum_{k<j} a_{jk} c_k \right) \right)$	$f''(f, fff)$
5	5	2	20	$\sum_{i,j=1}^s b_i \left(\sum_{j=1}^s a_{ij} c_j \right)^2$	$f''(ff, ff)$
5	6	6	20	$\sum_{i,j=1}^s b_i \left(\sum_{j<i} a_{ij} c_j^3 \right)$	$fff'(f, f, f)$
5	7	1	40	$\sum_{i,j,k=1}^s b_i \left(\sum_{j<i} a_{ij} c_j \left(\sum_{k<j} a_{jk} c_k \right) \right)$	$ff''(f, ff)$
5	8	2	60	$\sum_{i,j,k=1}^s b_i \left(\sum_{j<i} a_{ij} \left(\sum_{k<j} a_{jk} c_k^2 \right) \right)$	$fff''(f, f)$
5	9	1	120	$\sum_{i,j,k,m=1}^s b_i \left(\sum_{j<i} a_{ij} \left(\sum_{k<j} a_{jk} \left(\sum_{m<k} a_{km} c_m \right) \right) \right)$	$fffff$

3.4. taula. jarraipena.

3.5. taulan jaso dira y funtzioaren r . ordenako deribatu bakoitzak bete behar dituen baldintza kopurua eta guztira zenbat baldintza bete behar diren Runge-Kutta metodo bat ordena konkretu batekoa izateko. 3.5. taula era honetan irakurtzen da: metodoa hirugarren ordenakoa izateko, y funtzioaren hirugarren ordenako deribatuak bi baldintza bete behar ditu (3 ordenaren azpian dagoen zenbakia da bi), lehen ordenako deribatuak baldintza bat, eta bigarren ordenako deribatuak ere baldintza bat. Guztira, metodoa hirugarren ordenakoa izan dadin, 4 baldintza

bete behar dira. Kopuru osoari baldintza kopuru osoa deitu diogu taulan, deribatu bakoitzean bete beharreko baldintza kopurutik bereizteko.

Ordena	1	2	3	4	5	6	7	8	9	10
Deribatu bakoitzeko baldintza kopurua	1	1	2	4	9	20	48	115	286	719
Baldintza kopuru osoa	1	2	4	8	17	37	85	200	486	1.205

3.5. taula. 1-10 ordenako Runge-Kutta metodoek bete beharreko baldintza kopurua.

Gainera, Runge-Kutta metodo esplizitueta ordena eta etapa kopurua erlazionatuta daude. Horrek esan nahi du, ezen ordena bakoitzerako badagoela gutxieneko etapa kopuru bat. Ordena eta gutxieneko etapa kopurua 3.6. taulan jaso ditugu.

Runge-Kutta metodoaren ordena	1	2	3	4	5	6	7	8	9	10
Metodoaren etapa kopuru minimoa	1	2	3	4	6	7	9	11	12-17	13-17

3.6. taula. Runge-Kutta metodo esplizitu bateko ordenaren eta etapa kopuruaren arteko erlazioa.

Beraz, Runge-Kutta metodo bat sortzeko ezinbestekoa da ordenari dagozkion baldintzak kontuan izatea. Adibidez, etapa bakarreko eta lehen ordenako Runge-Kutta metodo esplizitu bat sortu nahi badugu, $b_1 = 1$ bete beharko da. Euler-en metodo esplizitua da hain zuzen, etapa bakarreko eta lehen ordenako Runge-Kutta metodoa. Bi etapako eta bigarren ordenako Runge-Kutta metodo esplizitu baten koefizienteek ondoko baldintza biak bete beharko dituzte:

$$\begin{cases} b_1 + b_2 = 1 \\ b_2 c_2 = \frac{1}{2} \end{cases} \quad (3.22)$$

(3.22) sistema bi ekuazio eta 3 ezezaguneko sistema da eta hainbat soluzio ditu. Haren soluzio orokorra parametro baten menpe dagoen familia bat da.

$$c_2 = \lambda \neq 0, \quad b_1 = 1 - \frac{1}{2\lambda}, \quad b_2 = \frac{1}{2\lambda} \quad (3.23)$$

Eta (3.22) sistemaren soluzio partikular bat hauxe litzateke: $b_1 = b_2 = \frac{1}{2}, c_2 = 1$. Bi etapa eta bigarren ordenako Runge-Kutta metodo honi dagokion Butcher-en taula 3.7. taula da.

0	0	0
1	1	0
	$\frac{1}{2}$	$\frac{1}{2}$

3.7. taula. (3.22) sistemaren soluzio partikular bati dagokion Butcher-en taula.

Hiru etapa eta hirugarren ordenako Runge-Kutta metodo esplizituko koefizienteek lau baldintza hauek bete behar dituzte:

$$\begin{cases} b_1 + b_2 + b_3 = 1 \\ b_2 c_2 + b_3 c_3 = \frac{1}{2} \\ b_2 c_2^2 + b_3 c_3^2 = \frac{1}{3} \\ b_3 c_2 a_{32} = \frac{1}{6} \end{cases} \quad (3.24)$$

(3.24) sistemak emaitza bezala familia ezberdinak ditu: alde batetik, bi parametroren menpe dagoen familia bat, eta, bestetik, parametro bakarraren mendeko bi familia.

- Bi parametroren mendeko familia hauxe da:

$$c_2 = a_{21} = \lambda, \quad c_3 = \mu, \quad b_1 = \frac{6\lambda\mu - 3(\lambda + \mu) + 2}{6\lambda\mu}, \quad b_2 = \frac{2 - 3\mu}{6\lambda(\lambda - \mu)}, \quad b_3 = \frac{3\lambda - 2}{6\mu(\lambda - \mu)}$$

$$a_{31} = \frac{\mu(3\lambda(\lambda - 1) + \mu)}{\lambda(3\lambda - 2)}, \quad a_{32} = \frac{\mu(\lambda - \mu)}{\lambda(3\lambda - 2)}$$

$$\lambda \neq 0, \frac{2}{3} \text{ eta } \mu \neq 0 \text{ izanik.}$$

- Parametro bakarraren menpe dagoen lehenengo familia:

$$c_2 = \frac{2}{3}, \quad c_3 = 0, \quad b_1 = \frac{1}{4} - \nu, \quad b_2 = \frac{3}{4}, \quad b_3 = \nu, \quad a_{31} = -a_{32} = \frac{-1}{4\nu}, \quad \nu \neq 0$$

- Parametro bakarraren menpe dagoen bigarren familia:

$$c_2 = c_3 \frac{2}{3}, \quad b_1 = \frac{1}{4}, \quad b_2 = \frac{3}{4} - \omega, \quad b_3 = \omega, \quad a_{31} = \frac{2}{3} - \frac{1}{4\omega}, \quad a_{32} = \frac{1}{4\omega}, \quad \omega \neq 0$$

(3.24) sistemaren bi soluzio partikular oso ezagunak diren hirugarren ordenako bi Runge-Kutta metodo lirateke. Alde batetik 3.8. Butcher-en tauladun ohiko hirugarren ordenako Runge-Kutta metodoa:

$$\begin{cases} b_1 = \frac{1}{6}, b_2 = \frac{2}{3}, b_3 = \frac{1}{6} \\ c_1 = 0, c_2 = \frac{1}{2}, c_3 = 1 \\ a_{21} = \frac{1}{2}, a_{31} = -1, a_{32} = 2 \end{cases} \quad (3.25)$$

0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0
1	-1	2	0
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

3.8. taula. 3 etapa eta hirugarren ordenako ohiko Runge-Kutta esplizituaren Butcher-en taula.

Eta bestetik, 3.9. Butcher-en tauladun Heun-en metodo izenaz ezagutzen den eta hirugarren ordenako Runge-Kutta metodoa den beste hau:

$$\begin{cases} b_1 = \frac{1}{4}, b_2 = 0, b_3 = \frac{3}{4} \\ c_1 = 0, c_2 = \frac{1}{3}, c_3 = \frac{2}{3} \\ a_{21} = \frac{1}{3}, a_{31} = 0, a_{32} = \frac{2}{3} \end{cases} \quad (3.26)$$

0	0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	0	0
$\frac{2}{3}$	0	$\frac{2}{3}$	0
	$\frac{1}{4}$	0	$\frac{3}{4}$

3.9. taula. Heun-en metodoari dagokion Butcher-en taula.

Era berean jarraituz, 4 etapako eta laugarren ordenako Runge-Kutta esplizitu bateko koefizienteek bete beharreko 8 baldintzak (3.21) adierazpeneko lehenengo zortziak lirateke. Eta sistema horren emaitza bi parametrodun familia bat eta parametro bakarraren mendeko lau familia dira. Horrela jarraitu ahal izango genuke bosgarren ordenarekin eta hurrengoekin. Honekin adierazi nahi izan duguna zera da, ordena eta etapa kopurua finkaturik, Runge-Kutta horri dagozkion konstanteak ez direla bakarrak, baizik eta metodo ezberdinak izan daitezkeela ordena eta etapa kopuru berarentzat.

3.3. EGONKORTASUN-EZAUGARRIAK

Runge-Kutta metodo baten egonkortasun-ezaugarriak aztertzeke metodoa test-ekuazioari, $y' = \lambda y$ ekuazioari, aplikatu behar zaio. Hori egindakoa, diferentzietan emandako lehenengo ordenako ekuazioa lortzen da:

$$y_{n+1} = R(\hat{h})y_n, \quad (3.27)$$

$\hat{h} = h\lambda$ izanik. Nahi duguna da kalkulatzeko zein den $R(\hat{h})$ bezala adierazi dugun egonkortasun-funtzioa. Test-ekuazioari (3.1) adierazpena aplikatu ondoren, ekuazio honetara iristen gara:

$$y_{n+1} = y_n + \hat{h} \sum_{i=1}^s b_i Y_i \quad (3.28)$$

$Y_i = y_n + \hat{h} \sum_{j=1}^s a_{ij} Y_j$, $i = 1, 2, \dots, s$ izanik eta s metodoaren etapa kopurua izanik.

$Y = [Y_1, Y_2, \dots, Y_s]^T \in \mathbb{R}^s$ eta $e = [1, 1, \dots, 1]^T \in \mathbb{R}^s$ definitzen ditugu eta era honetan (3.28) adierazpena eta Y_i -ren adierazpenak honela idatziko ditugu aldagai berri hauen menpe:

$$\begin{cases} Y = y_n e + \hat{h} A Y \\ y_{n+1} = y_n + \hat{h} b^T Y \end{cases} \quad (3.29)$$

(3.29) adierazpeneko lehenengo berdintzatik Y -ren balioa askatuz eta bigarren berdintzan ordezkaturaz, hauxe daukagu:

$$y_{n+1} = y_n \left(1 + \hat{h} b^T \left(I_s - \hat{h} A \right)^{-1} e \right) \quad (3.30)$$

Bukatzeko (3.27) eta (3.30) adierazpenak konparatuz, badaukagu s etapako Runge-Kutta metodoei dagokien egonkortasun-funtzioaren adierazpena:

$$R(\hat{h}) = 1 + \hat{h} b^T \left(I_s - \hat{h} A \right)^{-1} e \quad (3.31)$$

Dekker eta Verwer-ek egonkortasun-funtzioaren beste adierazpen hau ematen dute, (3.31) adierazpenarekin guztiz baliokidea dena:

$$R(\hat{h}) = \frac{\det(I_s - \hat{h} A + \hat{h} e b^T)}{\det(I_s - \hat{h} A)} \quad (3.32)$$

Lambert-en (1991: 199-200) erreferentzian, 2 etapako Runge-Kutta metodo bati dagokion egonkortasun-funtzioa (3.32) adierazpenak emana dagoela frogatuta dago.

Runge-Kutta medoen egonkortasun funtzioaren (3.32) adierazpenari erreparatzen badiogu, egonkortasun-funtzioaren inguruko zenbait ondorio atera daitezke:

- Runge-Kutta metodoa esplizitua denean, hau da, A matrizea hertsiki behe-triangeluarra denean, $(I_s - \hat{h} A)$ matrizea ere behe-triangeluarra da eta 1 zenbakia da matrize horren diagonal nagusian agertzen den elementua. Beraz, $\det(I - \hat{h} A) = 1$ beteko da eta, kasu honetan, egonkortasun-funtzioa $\hat{h}$ aldagaidun polinomioa izango da: $R(\hat{h}) = \det(I_s - \hat{h} A + \hat{h} e b^T)$.

- Runge-Kutta metodo esplizituen kasuan, A-egonkortasuneko baldintza ez da beteko. $|\hat{h}| \rightarrow \infty$ betetzen denean, y_{n+1} ez baita bornatua egongo. Horrek esan nahi du, Runge-Kutta metodo esplizituek egonkortasun-eremu finitua izango dutela.
- Runge-Kutta metodo ez-esplizituen kasuan, $(I - \hat{h}A)$ matrizearen determinantea $\hat{h}$ aldagaidun polinomioa da eta egonkortasun-funtzioa $\hat{h}$ aldagaidun funtzio arrazionala izango da.
- Runge-Kutta metodo ez-esplizituen kasuan, A-egonkortasuneko baldintza betetzea posible da. Gerta baitaiteke $|\hat{h}| \rightarrow \infty$ betetzen denean y_{n+1} bornatua egotea eta, beraz, posible da egonkortasun-eremu infinitu bat izatea metodo hauek.

Zenbait Runge-Kutta metodo esplizituen kasurako (3.31) edo (3.32) adierazpenek hartzen duten forma zein den ikusiko dugu, Lambert-en (1991:201-205) erreferentzia aurki daiteke beraien frogapen bat:

- $s = p$ betetzen duen s etapa eta p ordenako Runge-Kutta metodo esplizitu bat daukagunean (eta hau gertatzen da $s = 1, 2, 3, 4$ kasuetan), egonkortasun-eremua beti da berdina eta egonkortasun-funtzioa (kasu honetan polinomioa, metodo esplizituekin baikabiltza) honela emana dator:

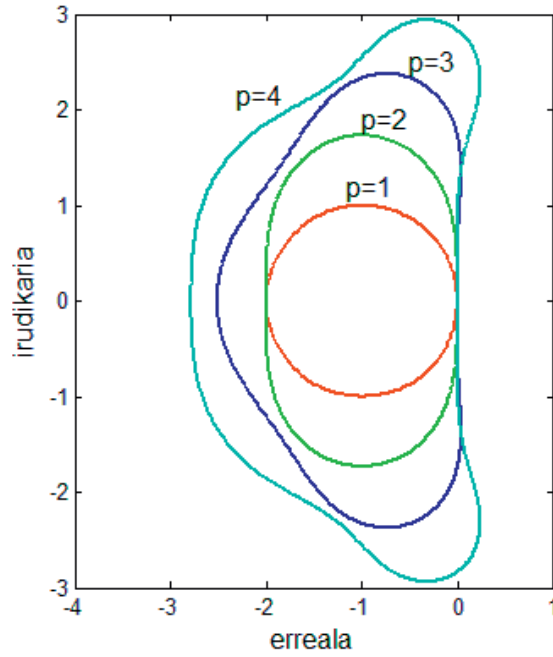
$$R(\hat{h}) = 1 + \sum_{j=1}^s \frac{\hat{h}^j}{j!} \quad (3.33)$$

3.1. irudian, ordena eta etapa kopuru bera duten Runge-Kutta metodo esplizituen egonkortasun-eremuak irudikatu dira.

- $s > p$ betetzen duen s etapa eta p ordenako Runge-Kutta metodo esplizitua badaukagu (eta hau gertatzen da $p > 4$ denean metodo esplizituen kasuan), egonkortasun-polinomioak forma hau dauka:

$$R(\hat{h}) = 1 + \hat{h} + \frac{\hat{h}^2}{2!} + \frac{\hat{h}^3}{3!} + \dots + \frac{\hat{h}^p}{p!} + \sum_{q=p+1}^s \gamma_q \hat{h}^q \quad (3.34)$$

γ_q koefizienteak Runge-Kutta metodoko koefizienteen mendekoak dira eta orain, bai, existitzen da aukera txiki bat, behin etapa eta ordena finkatuta egonkortasun-eremu ezberdinak dituzten metodoak lortzeko eta, beraz, ezaugarri hobeak dituzten metodoak sortzeko. Baina esan beharra dago, aukera hau ustiatzeko probek ere ez dutela emaitza ikusgarriak sorrarazten.



3.1. irudia. Etapa eta ordena berdinak dituzten Runge-Kutta metodo esplizituen egonkortasun-eremuak (kurben barruko aldeak).

Adibidea

s etapa eta $p = (s-1)$ ordenako Runge-Kutta metodo esplizitu baten egonkortasun-funtzioaren adierazpena zein den ondorioztatu dugu. $s > p$ kasuan gaudenez, kasu honetarako daukagun egonkortasun-polinomioaren (3.34) adierazpena honela geratzen zaigu:

$$R(\hat{h}) = 1 + \hat{h} + \frac{\hat{h}^2}{2!} + \frac{\hat{h}^3}{3!} + \dots + \frac{\hat{h}^{s-1}}{(s-1)!} + \gamma_s \hat{h}^s \quad (3.35)$$

Egonkortasun-polinomioa guztiz definituta izateko, γ_s koefizientea aurkitzea besterik ez zaigu geratzen. Horretarako, egonkortasun-funtzioak kalkulatzeko daukagun (3.31) adierazpena erabiliko dugu. Adierazpen horretako idazkera apur bat sinplifikatzeko $d = (I_s - \hat{h}A)^{-1}e$ definituko dugu, eta ondorioz, (3.31) adierazpenak forma hau hartuko du:

$$R(\hat{h}) = 1 + \hat{h}b^T d \quad (3.36)$$

$p = (s-1)$ kasuko (3.35) adierazpeneko γ_s koefizientea kalkulatzeko, (3.36) adierazpenean d aldagaian $\hat{h}^{s-1}$ terminoari dagokion koefizientea gorde beharko dugu. Hau da, $(I_s - \hat{h}A)d = e$ sistemako pauso bakoitzean $\hat{h}$ -ri dagokion potentziarik handienaren terminoak gorde beharko ditugu.

Gogoratu, Runge-Kutta metodo bateko koefizienteek errenkadaka (3.4) baldintza bete behar dutela. Metodoa esplizitua den kasurako, baldintza horrek forma hau hartzen du:

$$\sum_{j=1}^{i-1} a_{ij} = c_i, \quad i = 1, 2, \dots, s \quad (3.37)$$

$(I_s - \hat{h}A)d = e$ sistema planteatuz haxe daukagu:

$$(I_s - \hat{h}A)d = e \Rightarrow \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\hat{h}a_{21} & 1 & 0 & \cdots & 0 \\ -\hat{h}a_{31} & -\hat{h}a_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\hat{h}a_{s1} & -\hat{h}a_{s2} & -\hat{h}a_{s3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_s \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \quad (3.38)$$

(3.38) sistemako pauso bakoitzean $\hat{h}$ -ren potentziarik handienak gordeko ditugu eta P. T. hizkiak erabiliko ditugu $\hat{h}$ -ren potentziarik handienak ez direnak adierazteko (Potentzia Txikiak adierazi nahi dugu hizki horiekin). Zera daukagu:

$$\begin{cases} d_1 = 1 \\ d_2 = 1 + \hat{h}a_{21}d_1 \\ d_3 = 1 + \hat{h}a_{31}d_1 + \hat{h}a_{32}d_2 \\ \vdots \\ d_s = 1 + \hat{h}a_{s1}d_1 + \hat{h}a_{s2}d_2 + \dots + \hat{h}a_{s,s-1}d_{s-1} \end{cases} \quad (3.39)$$

(3.39) sisteman, d_1 balioa badaukagunez, d_2 kalkula dezakegu. Behin d_2 daukagunez, d_3 kalkula dezakegu, etab. Hau da, d_i kalkulatzeko $d_1, d_2, \dots, d_{i-1}$ balioak ordezkatzeko ditugu d_i -ren adierazpenean eta beti ordenarik altueneko $\hat{h}$ -ren potentzia bakarrik hartuko dugu:

$$\begin{cases} d_1 = 1 \\ d_2 = 1 + \hat{h}a_{21}d_1 = c_2\hat{h} + P.T. \\ d_3 = 1 + \hat{h}a_{31}d_1 + \hat{h}a_{32}d_2 = a_{32}c_2\hat{h}^2 + P.T. \\ \vdots \\ d_s = 1 + \hat{h}a_{s1}d_1 + \hat{h}a_{s2}d_2 + \dots + \hat{h}a_{s,s-1}d_{s-1} = a_{s,s-1}a_{s-1,s-2}\dots a_{32}c_2\hat{h}^{s-1} + P.T. \end{cases} \quad (3.40)$$

$\hat{h}^{s-1}$ -ren koefizientea $a_{s,s-1}a_{s-1,s-2}\dots a_{32}c_2$ da eta $R(\hat{h}) = 1 + \hat{h}b^T d$ adierazpeneko $(\hat{h}b^T d)$ biderketa egiten dugunean, $\hat{h}^s$ -ren koefiziente bilakatuko da. Beraz, γ_s koefizientea hauxe da s etapa eta $p = (s-1)$ ordenako Runge-Kutta metodo esplizitu baten kasuan:

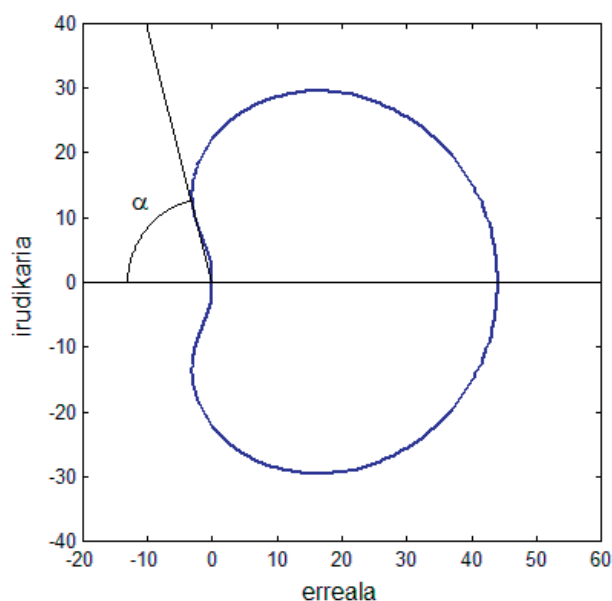
$$\gamma_s = b_s a_{s,s-1} a_{s-1,s-2} \dots a_{32} c_2 \quad (3.41)$$

Edozelan ere, adibide honetan egin ditugun eragiketa guztiak egin beharrik ez dago, Dekker eta Verwer-ek proposatutako egonkortasun-eremuaren (3.32) adierazpena erabilia asko errazten baita kalkulua.

Runge-Kutta metodo inplizituen abantaila da, ordena konkretu bat lortzeko metodo esplizituetan baino etapa gutxiago nahikoa izatea eta esplizituek baino egonkortasun-eremu handiagoa edukitzea. Desabantaila nagusia, ordea, gutxienez etaparen batek duen izaera inplizitua da. Izaera inplizitu horrek metodo iteratiboak erabiltzera behartzen gaitu pauso bakoitzean. Adibidez, 3.10. taulako koefizienteak dituen Runge-Kutta metodo inplizitua $A(75,5996^\circ)$ -egonkorra da $\lambda \approx 0,158984$ baliorako. Metodo horri dagokion egonkortasun-eremua 3.2. irudian marraztu da.

λ	λ	0	0
$\frac{1+\lambda}{2}$	$\frac{1-\lambda}{2}$	λ	0
1	$\frac{-(1-\lambda)(1-9\lambda+6\lambda^2)}{1-3\lambda+6\lambda^2}$	$\frac{2(1-\lambda)(1-6\lambda+6\lambda^2)}{1-3\lambda+6\lambda^2}$	λ
	$\frac{1+3\lambda}{6(1-\lambda)^2}$	$\frac{2(1-3\lambda)}{3(1-\lambda)^2}$	$\frac{1-3\lambda+6\lambda^2}{6(1-\lambda)^2}$

3.10. taula. Runge-Kutta metodo inplizitu baten taula
(Butcher, 2008: 230).



3.2. irudia. 3.10. Butcher-en tauladun Runge-Kutta metodoaren egonkortasun-eremua $\lambda = 0,158984$ kasurako (eremuaren kanpoko aldea).

Runge-Kutta metodo inplizituen adibide dira Gauss-en metodo izenekoak, zeinetan $s = 1, 2, 3$ etapetarako $p = 2, 4, 6$ ordenak lortzen ditugun hurrenez hurren. Gauss-en metodo hauei dagozkien koefizienteak 3.11., 3.12. eta 3.13. tauletan jaso ditugu eta berriro azpimarratu nahi dugu metodo inplizituen desabantaila beren izaera inplizitua dela.

$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$
	1

3.11. taula. Etapa bat eta bi ordena dituen Runge-Kutta metodo inplizitu bat.

$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4} - \frac{\sqrt{3}}{6}$
$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{4} + \frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

3.12. taula. Bi etapa eta lau ordena dituen Runge-Kutta metodo implizitu bat.

$\frac{1}{2} - \frac{\sqrt{15}}{10}$	$\frac{5}{36}$	$\frac{2}{9} - \frac{\sqrt{15}}{15}$	$\frac{5}{36} - \frac{\sqrt{15}}{30}$
$\frac{1}{2}$	$\frac{5}{36} + \frac{\sqrt{15}}{24}$	$\frac{2}{9}$	$\frac{5}{36} - \frac{\sqrt{15}}{24}$
$\frac{1}{2} + \frac{\sqrt{15}}{10}$	$\frac{5}{36} + \frac{\sqrt{15}}{30}$	$\frac{2}{9} + \frac{\sqrt{15}}{15}$	$\frac{5}{36}$
	$\frac{5}{18}$	$\frac{4}{9}$	$\frac{5}{18}$

3.13. taula. Hiru etapa eta sei ordena dituen Runge-Kutta metodo implizitu bat.

3.4. KATEATUTAKO RUNGE-KUTTA METODOAK

Runge-Kutta metodoetako mozketak-errore lokalaren termino nagusia kalkulatzeko zaila da. Hori dela-eta, Merson-ek (1957an) pausoa kalkulatuak k_i konstanteen menpe zegoen errorea estimatzeko era bat proposatu zuen. Merson-en ideia p eta $(p+1)$ ordenako Runge-Kutta metodo bi kontsideratzean oinarritzen zen, zeinek c_i, a_{ij} konstante berdinak baitzituzten, metodo bakoitzak bere b^T eta $\hat{b}^T$ koefizienteak izanik. Era horretan sortutako Runge-Kutta metodoei kateatutako Runge-Kutta metodo deritze eta 3.14. Butcher-en taula dagokie, non b^T eta $\hat{b}^T$ bektoreen diferentzia ere jasota dagoen: $E^T = \hat{b}^T - b^T$.

c	A
	b^T
	$\hat{b}^T$
	E^T

3.14. taula. Kateatutako Runge-Kutta metodo baten Butcher-en taula.

c , b^T bektoreek eta A matrizeak definitzen duten Runge-Kutta metodoa p ordenakoa da, eta c , $\hat{b}^T$ bektoreak eta A matrizeak definitzen dutena $(p+1)$ ordenakoa da. Kateatutako Runge-Kutta metodoetan k_i koefizienteak berdinak dira p ordena duen metodoan, baita $(p+1)$ ordenakoan ere, k_i koefizienteek c_i, a_{ij} -ren mendekotasuna baitaukate bakarrik. p eta $(p+1)$ ordenako metodoen arteko ezberdintasun bakarra metodoaren soluzioa diren $\hat{y}_{n+1}$, $(p+1)$ ordenakoa, eta y_{n+1} , p ordenakoa, kalkulatzeko egin beharreko azken batuketan dago, zeinetan $\hat{b}^T$ eta b^T bektoreetako osagaiak erabiltzen diren:

- $(p+1)$ ordenako metodoaz lortzen dena: $\hat{y}_{n+1} = y_n + h \sum_{i=1}^s \hat{b}_i k_i$
- p ordenako metodoaz lortutakoa: $y_{n+1} = y_n + h \sum_{i=1}^s b_i k_i$

Merson-ek mozketa-errore lokala $\hat{y}_{n+1}$ eta y_{n+1} balioen diferentzia bezala estimatzea proposatu zuen:

$$\hat{y}_{n+1} - y_{n+1} = h \sum_{i=1}^s (\hat{b}_i - b_i) k_i = h \sum_{i=1}^s E_i k_i \quad (3.42)$$

Non: $E_i = \hat{b}_i - b_i$.

Hurrengo pausoko hasierako balio gisa p ordenaz kalkulatu dugun y_{n+1} balioa erabiltzen dugunean, kateatutako Runge-Kutta metodoa $(p, p+1)$ bezala idazten da. Kasu horretan, p ordenaz kalkulaturako emaitzak kalkulatu ditugu eta $(p+1)$ ordenaz kalkulaturakoak bakarrik errorearen estimazioan erabiltzen ditugu.

Aldiz, $(p+1)$ ordenako metodoak ematen digun emaitza, $\hat{y}_{n+1}$, hurrengo pausoko hasierako balio bezala darabilgunean eta p ordenak ematen digun emaitza bakarrik estimaziorako darabilgunean, kateatutako Runge-Kutta metodoa $(p+1, p)$ bezala izendatzen da.

Asko dira kateatutako Runge-Kutta metodoak. Besteak beste:

- Fehlberg-ek hainbat kateatutako Runge-Kutta metodo proposatu izan ditu.

Fehlberg-en kateatutako Runge-Kutta (4,5) metodorik ezagunenetakoa, RKF45 izenaz ezagutzen da. Metodo horrek laugarren ordenako metodoa erabilia lortzen den emaitza ematen du eta bosgarren eta laugarren ordenetako soluzioen diferentzia erabiltzen du errorearen estimazioa kalkulatzeko. Metodo horretako koefizienteak mozketaren erroreak txikia izateko eran aukeratu zituen Fehlberg-ek. Metodo horren Butcher-en taula 3.15. taula da. Sei etapa dituen metodoa da eta laugarren ordenari dagokion egonkortasun-polinomioa eta bosgarren ordenari dagozkionak hauek dira:

$$\text{– Laugarren ordenakoa: } R(\hat{h}) = 1 + \hat{h} + \frac{\hat{h}^2}{2!} + \frac{\hat{h}^3}{3!} + \frac{\hat{h}^4}{4!} + \frac{\hat{h}^5}{104}$$

$$\text{– Bosgarren ordenakoa: } R(\hat{h}) = 1 + \hat{h} + \frac{\hat{h}^2}{2!} + \frac{\hat{h}^3}{3!} + \frac{\hat{h}^4}{4!} + \frac{\hat{h}^5}{5!} + \frac{\hat{h}^6}{2.080}$$

Bosgarren ordenako egonkortasun-polinomioko $\hat{h}^6$ -ren koefizientea (3.41) adieraz-pena erabilia kalkula dezakegu:

$$\gamma_6 = \hat{b}_6 a_{65} a_{54} a_{43} a_{32} a_{21} = \frac{2}{55} \cdot \left(-\frac{11}{40}\right) \cdot \left(-\frac{845}{4.104}\right) \cdot \frac{7.296}{2.197} \cdot \frac{9}{32} \cdot \frac{1}{4} = \frac{1}{2.080}$$

Egonkortasun-eremuak 3.3. irudian ikus daitezke.

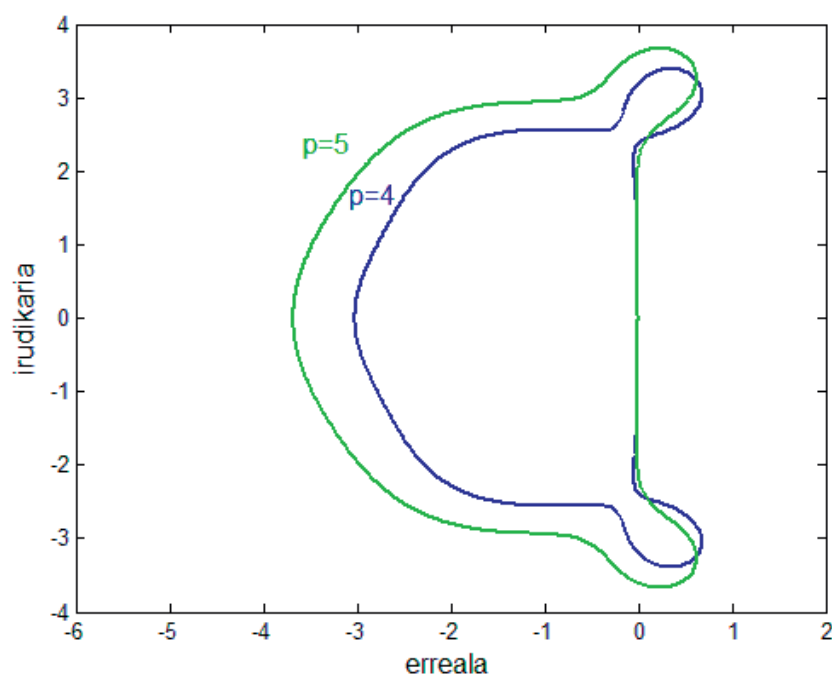
c_i	a_{i1}	a_{i2}	a_{i3}	a_{i4}	a_{i5}	a_{i6}
0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1.932}{2.197}$	$-\frac{7.200}{2.197}$	$\frac{7.296}{2.197}$			
1	$\frac{439}{216}$	-8	$\frac{3.680}{513}$	$-\frac{845}{4.104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3.544}{2.565}$	$\frac{1.859}{4.104}$	$-\frac{11}{40}$	
b^T	$\frac{25}{216}$	0	$\frac{1.408}{2.565}$	$\frac{2.197}{4.104}$	$-\frac{1}{5}$	0
$\hat{b}^T$	$\frac{16}{135}$	0	$\frac{6.656}{12.825}$	$\frac{28.561}{56.430}$	$-\frac{9}{50}$	$\frac{2}{55}$
E^T	$\frac{1}{360}$	0	$-\frac{128}{4.275}$	$-\frac{2.197}{75.240}$	$\frac{1}{50}$	$\frac{2}{55}$

3.15. taula. RKF45 metodoaren Butcher-en taula.

- Kateatutako beste Runge-Kutta metodo ezagun bat Dormand eta Prince-k sortu zutena da (Dormand eta Prince, 1980). Zazpi etapako metodoa da, $\hat{b}^T$ konstanteak darabiltzagunean metodoa bosgarren ordenakoa da, eta b^T konstanteekin, aldiz, laugarren ordenakoa. Praktikan 6 etapako metodoa da, $\hat{b}^T$ bektoreko azken osagaia zero baita. Baldintza hori betetzen duten metodoei ingelesez FSAL (*First Same As Last*) esaten zaie, lehenengoa azkena bezalakoa esan nahi duena. Horrek zera esan nahi du: gauden pausoko deribatuaren azken ebaluazioa hurrengo pausoko lehen etapako deribatuaren berdina dela. Metodo horretako egonkortasun-polinomioak hauek dira:

– Laugarren ordenakoa:

$$R(\hat{h}) = 1 + \hat{h} + \frac{\hat{h}^2}{2} + \frac{\hat{h}^3}{6} + \frac{\hat{h}^4}{24} + \frac{149\hat{h}^5}{16.299} + \frac{41\hat{h}^6}{30.559} + \frac{\hat{h}^7}{24.000}$$



3.3. irudia. RKF45 metodoaren egonkortasun-eremuak (kurben barneko aldeak).

– Bosgarren ordenakoa:
$$R(\hat{h}) = 1 + \hat{h} + \frac{\hat{h}^2}{2} + \frac{\hat{h}^3}{6} + \frac{\hat{h}^4}{24} + \frac{\hat{h}^5}{120} + \frac{\hat{h}^6}{600}$$

Metodo horretako koefizienteak 3.16. taulan jaso ditugu, eta metodoari dagokion egonkortasun-eremua 3.4. irudian dago eginda.

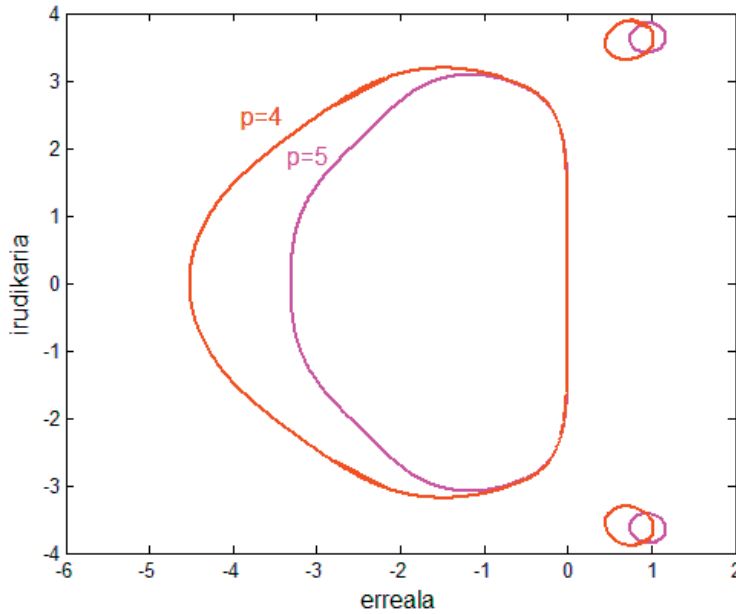
c_i	a_{i1}	a_{i2}	a_{i3}	a_{i4}	a_{i5}	a_{i6}	a_{i7}
0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19.372}{6.561}$	$-\frac{25.360}{2.187}$	$\frac{64.448}{6.561}$	$-\frac{212}{729}$			
1	$\frac{9.017}{3.168}$	$-\frac{355}{33}$	$\frac{46.732}{5.247}$	$\frac{49}{176}$	$-\frac{5.103}{18.656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1.113}$	$\frac{125}{192}$	$-\frac{2.187}{6.784}$	$\frac{11}{84}$	
b^T	$\frac{5.179}{57.600}$	0	$\frac{7.571}{16.695}$	$\frac{393}{640}$	$-\frac{92.097}{339.200}$	$\frac{187}{2.100}$	$\frac{1}{40}$
$\hat{b}^T$	$\frac{35}{384}$	0	$\frac{500}{1.113}$	$\frac{125}{192}$	$-\frac{2.187}{6.784}$	$\frac{11}{84}$	0
E^T	$\frac{71}{57.600}$	0	$-\frac{71}{16.695}$	$\frac{71}{1.920}$	$-\frac{17.253}{339.200}$	$\frac{22}{525}$	$-\frac{1}{40}$

3.16. taula. DOPRI(5,4) metodoaren Butcher-en taula.

Adibidea

Runge-Kutta metodo bati dagokion egonkortasun-eremua MATLAB erabilita zelan irudikatu daitekeen ikusiko dugu adibide baten bidez. DOPRI(5,4) kateatutako metodoaren kasuan esan dugunez, bosgarren ordenako metodoaren egonkortasun-polinomioa honela dator emana:

$$R(\hat{h}) = 1 + \hat{h} + \frac{\hat{h}^2}{2!} + \frac{\hat{h}^3}{3!} + \frac{\hat{h}^4}{4!} + \frac{\hat{h}^5}{5!} + \frac{\hat{h}^6}{600}$$



3.4. irudia. DOPRI(5,4) metodoaren egonkortasun-eremuak (kurben barneko aldeak).

Metodoa egonkorra izango da, $R(\hat{h}) \leq 1$ baldintza betetzen den kasuan. MATLAB erabilia egonkortasun-eremu hau konputatu nahi badugu, egonkortasun-polinomioa unitatearekin berdinduko dugu. Horrela, egonkortasun-polinomioa unitatearen berdina betetzen duten $\hat{h}$ plano konplexuko zenbakiak lortuko ditugu, eta horiek egonkortasun-eremuaren muga osatuko dute:

$$R(\hat{h}) = 1 + \hat{h} + \frac{\hat{h}^2}{2!} + \frac{\hat{h}^3}{3!} + \frac{\hat{h}^4}{4!} + \frac{\hat{h}^5}{5!} + \frac{\hat{h}^6}{600} = e^{i\theta}, \quad \theta \in [0, 2\pi)$$

Gauza bera dena, θ parametroari $[0, 2\pi)$ tarteko balioak eman ahala, $p(\hat{h})$ polinomioaren erroak aurkituko ditugu:

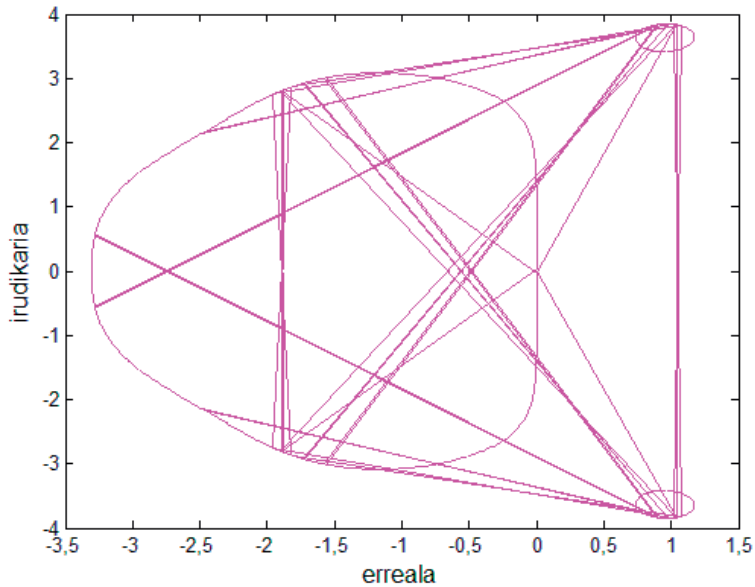
$$p(\hat{h}) = 1 - e^{i\theta} + \hat{h} + \frac{\hat{h}^2}{2!} + \frac{\hat{h}^3}{3!} + \frac{\hat{h}^4}{4!} + \frac{\hat{h}^5}{5!} + \frac{\hat{h}^6}{600}, \quad \theta \in [0, 2\pi)$$

MATLABen era honetan programatuko genuke:

```
%R(h)-ren koefizienteak maila handienekotik 1. Mailakoraino
%jarrita:
c=[1/600 1/120 1/24 1/6 1/2 1];
%Egonkortasun-eremuaren muga zenbat puntu erabilita irudikatu
%dugun adierazten dugu. Adibidez, 100 puntu erabilita:
n=100;
%Angeluarentzat tartea sartzen dugu. Fi deitu diogu angeluari eta
%[0,pi] artean balioak hartzea eragingo dugu. Gainontzeko erroak
%konjokatua erabiliz lortuko ditugu:
fi=0:pi/n:pi;
%Erroak jasoko ditugun matrizeari zeroekin betez ematen diogu
%hasiera. 6. mailako polinomio bat daukagunez, angeluaren balio
%bakoitzerako 6 erro lortuko ditugu:
xc=zeros(n+1,6);
%Angeluaren balioa aldatuz goaz:
for j1=1:n+1
    cc=[c (1-exp(fi(j1)*i))];
    xc(j1,:)=roots(cc)';
end
%Konjokatua kalkulatu dugu, 3. eta 4. Koadranteetako erroak
%izateko:
yc=conj(xc);
plot(xc,'m');
hold on
plot(yc,'m');
```

Goiko programaren emaitza 3.5. irudian adierazi duguna litzateke. Bertan ikus daiteke $p(\hat{h})$ polinomioaren erroak ez direla agertzen ordenatuta.

Goiko programan agindu-zerrenda luzea sartu beharko genuke, egonkortasun-eremua ordenatuta ateratzea nahiko bagenu.



3.5. irudia. DOPRI(5,4) metodoko bosgarren ordenari dagokion egonkortasun-eremua desordenatuta.

Ohikoa da zenbait metodoren egonkortasun-polinomioari dagozkion erroak era ordenatuan ez agertzea eta, ondorioz, lortuko dugun egonkortasun-eremuaren grafikoa ere ez da ordenatuta agertuko. Erroak ordenatzeko funtzio bat sortu dugu, MATLABen «ordenatu_egonkortasun_eremua» izena jarri diogu eta azpian jaso dugu:

```
function h= ordenatu_egonkortasun_eremua(h)
%Zenbakizko metodo bateko egonkortasun-eremuko muga osatzen duten
%erroak ordenatzeko balio duen programa. Egonkortasun-eremuaren
%erro guztiak eman behar zaizkio h deitu dugun matrize batean:
    realh(:, :)=real(h(:, :));
    imagh(:, :)=abs(imag(h(:, :)));
    h(:, :)=realh(:, :)+i*imagh(:, :);
    ff=size(h,1);%ff: h-ren errenkada kopurua
    cc=size(h,2);%cc: h-ren zutabe kopurua
%Lehenengo zutabetik azkenengora arte, eta zutabe bakoitzean
%lehenengo errenkadatik azkenengora arte, h(m,n) elementuari
%hurrengo errenkadako elementu guztiak kentzen dizkiogu:
```

```

for n=1:cc-1
    for m=1:ff-1
        A=h(m,n)-h(m+1,n:end);
        %kenketa hauetako baliorik txikiena aurkitzen dugu
        %kenketaren balio minimoa dagoen indizea hartzen dugu:
        a=min(A);
        kk=find(abs(A)<=abs(a));
        s2=size(kk,2);
        %Zutabeetan aurrera egin ahala, definitu dugun A matrizeak gero eta
        %zutabe gutxiago ditu eta bere dimentsioa hasierako dimentsiora
        %egokitu beharra dago:
        kk=kk+n-1;
        %Aurreko baldintza leku baten baino gehiagotan gertatzen bada,
        %era ordenatuan egiteko baldintza betetzen den indizerik txikiena
        %hartzen dugu:
        if s2>1
            kk=kk(1);
        end
        %Diferentziarik txikiena gertatzen zen posizioan geneukan balioa
        %hartzen dugu:
        a2=h(m+1,kk);
        %h(m+1,n)-en geneukana posizio berrira eramaten dugu:
        h(m+1,kk)=h(m+1,n);
        %a2 balioa h(m+1,n) posiziora eramango dugu:
        h(m+1,n)=a2;
    end
end

```

Erroen ordenamendurako funtzio hau aplikatzeak aipatu dugun aginduzerrenda idaztea ekiditen du eta egonkortasun-eremua era ordenatuan irudikatzea ahalbidetzen du, 3.4. irudiko grafikoak lortuz.

4. Pauso anitzeko zenbait metodo lineal ezagun

Pauso anitzeko zenbakizko metodoak aurreko hainbat pausotako informazioa darabilten metodoak dira. Zenbakizko metodo bat pauso anitzeko metodoa edo k pausoko metodoa dela esaten da aurreko k pausoetako $t_n, t_{n+1}=t_n+h, \dots, t_{n+k-1}=t_n+(k-1)h$ balioetan lortu diren $y_n, y_{n+1}, \dots, y_{n+k-1}$ balioak edota berauen deribatuak $f_n, f_{n+1}, \dots, f_{n+k-1}$ erabiltzen dituztenean $t_{n+k}=t_n+kh$ puntuko y_{n+k} balioa lortzeko. Pauso anitzeko zenbakizko metodo bat era honetan idatz daiteke:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \phi(f_{n+k}, f_{n+k-1}, \dots, f_n, t_n; h) \quad (4.1)$$

$f_{n+j} = f(t_{n+j}, y_{n+j})$ izanik.

y_{n+j}, f_{n+j} balioen arteko erlazioa lineala denean, pauso anitzeko zenbakizko metodoa lineala da, eta (4.1) adierazpenak forma hau hartzen du:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \sum_{j=0}^k \beta_j f_{n+j} \quad (4.2)$$

(4.2) adierazpenaz emandako zenbakizko metodo lineala esplizitua da $\beta_k = 0$ bada, eta inplizitua $\beta_k \neq 0$ bada. Zenbakizko metodoa pauso bakarrekoa da $k=1$ denean.

Pauso anitzeko metodo linealik ezagunenetakoak Adams Bashforth eta Adams Moulton-en metodoak eta *Backward Differentiation Formulae* (BDF) metodoak dira.

- Pauso anitzeko metodo linealen (4.2) adierazpenean $\alpha_k = 1, \alpha_{k-1} = -1, \alpha_j = 0 \quad j=0,1,\dots,k-2$ balioetarako, $\beta_j = \tilde{\beta}_{k-j} \quad j=0,1,\dots,k-1$ balioetarako eta $\beta_k = 0$ hartzen baditugu, Adams Bashforth-en metodoak ditugu eta hauxe da beraien adierazpena:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=1}^k \tilde{\beta}_j f_{n+k-j} \quad (4.3)$$

- Adams Bashforth-en kasuan egin dugun antzera, (4.2) adierazpenean $\alpha_k = 1$, $\alpha_{k-1} = -1$, $\alpha_j = 0$ $j = 0, 1, \dots, k-2$ balioetarako eta $\beta_j = \beta_{k-j}^*$ $j = 1, 2, \dots, k$ balioetarako eta $\beta_0 = 0$ hartzen baditugu, Adams Moulton-en metodoak lortzen ditugu. Metodo horiek inplizituak dira eta hauxe da beraien adierazpena:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=1}^k \beta_j^* f_{n+k+1-j} \quad (4.4)$$

$\tilde{\beta}_j$ eta β_j^* Adams Bashforth eta Adams Moulton-en metodoetako konstanteak izanik hurrenez hurren, aurrerago ikusiko ditugunak. Adams-en metodoek Euler-en metodoei egin zitzairen lehenengo hedapena suposatu zuten eta Euler-ek baino informazio gehiago darabilte pauso bat emateko. Metodo hauek Adams eta Bashforth-i zor dizkiegu (Bashforth eta Adams, 1883). Balistikako (Moulton, 1926) erreferentzian hasierako baliodun ekuazio diferentzial arruntak askatzeko erabiltzen dira metodo hauek. Konkretuki proiektile baten mugimenduaren propietateak ezagutzeko darabiltzate Adams-en metodoak.

- Pauso anitzeko beste metodo ezagun batzuk BDFak dira, Gear-ek sartu zituenak (Gear, 1971). Metodo horiek asko erabili izan dira, egonkortasun-ezaugarri onak baitituzte. Metodo horiek ere Euler-en metodoek baino informazio gehiago darabilte pauso bat eman ahal izateko. Beraien adierazpena lortzeko, (4.2) adierazpenean $\alpha_j = \hat{\alpha}_j$ $j = 0, 1, \dots, k$ balioetarako, $\beta_k = 1$ eta $\beta_j = 0$ egiten dugu $j = 0, 1, \dots, k-1$ balioetarako. Hori egin ostean lortzen dugun adierazpena hauxe da:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = h f_{n+k} \quad (4.5)$$

$\hat{\alpha}_j$ konstanteak BDF metodoko konstanteak izanik, eta aurrerago BDFei buruzko atal berezian ikusiko ditugu.

4.1. ORDENA-BALDINTZAK

Pauso anitzeko zenbakizko metodo linealen mozketak-errore lokala kalkulatzeko izango da gure hurrengo helburua. Horretarako, mozketak-errore lokalaren adierazpena gogoratuko dugu:

$$LTE_{n+k} = y(t_{n+k}) - y_{n+k}^* \quad (4.6)$$

y_{n+k}^* balioa kokapen-bereganaketa eginda kalkulatzen delarik. $y(t_{n+k})$ eta y_{n+k}^* terminoen Taylor-en garapenak (4.6) adierazpenean ordezkatzuz eta eragiketak eginez, pauso anitzeko metodo linealen mozketak-errore lokalaren adierazpena lortzen da:

$$LTE_{n+k} = C_0 y(t_n) + C_1 h y'(t_n) + C_2 h^2 y''(t_n) + \dots + C_q h^q y^{(q)}(t_n) + \dots \quad (4.7)$$

C_i konstanteak hauek izanik:

$$\begin{cases} C_0 = \sum_{i=0}^k \alpha_i \\ C_1 = \sum_{i=0}^k i \alpha_i - \sum_{i=0}^k \beta_i \\ C_q = \frac{1}{q!} \sum_{i=0}^k i^q \alpha_i - \frac{1}{(q-1)!} \left(\sum_{i=0}^k i^{q-1} \beta_i \right), \quad q \geq 2 \end{cases} \quad (4.8)$$

Pauso anitzeko metodo lineal bat p ordenakoa da, baldin ondorengo baldintza betetzen bada:

$$C_0 = C_1 = \dots = C_p = 0, C_{p+1} \neq 0 \quad (4.9)$$

Eta hori gertatzen denean, metodoaren mozketak-errore lokala $(p+1)$ ordenakoa da:

$$LTE_{n+k} = C_{p+1} h^{p+1} y^{(p+1)}(t_n) + O(h^{p+2}) \quad (4.10)$$

Pauso anitzeko zenbakizko metodo linealen kasurako ohikoa da, (4.10) adierazpeneko C_{p+1} errore-konstantea normalizatuta hartzea (Hairer, Nørsett eta Wanner, 1993: 372-373; eta Butcher, 2000):

$$C = \frac{C_{p+1}}{\rho'(1)} = \frac{C_{p+1}}{\sigma(1)} \quad (4.11)$$

$\rho(r)$ eta $\sigma(r)$ metodoari lotutako polinomio karakteristikoak izanik:

$$\begin{cases} \rho(r) = \sum_{j=0}^k \alpha_j r^j \\ \sigma(r) = \sum_{j=0}^k \beta_j r^j \end{cases} \quad (4.12)$$

4.2. EGONKORTASUN-EZAUGARRIAK

Pauso anitzeko metodo linealen kasurako ere baliagarria da aurretik egonkortasun-eremuak kalkulatzeko azaldu den prozesua. Baina pauso anitzeko metodo linealen kasuan, aurretik azaldu den prozesu hura oraintxe azalduko dugun metodologia errazago honek ordezkatu dezake. Pauso anitzeko metodo linealen kasuan, behin metodoa test-ekuazioari aplikatu ostean, diferentzian emandako ondorengo ekuaziora iristen gara:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h\lambda \sum_{j=0}^k \beta_j y_{n+j} \quad (4.13)$$

(4.13) ekuazioa, Lagrange-ren metodoa erabilia ebazten da (Hairer, Nørsett eta Wanner, 1993: 378-380). Lagrange-ren metodoa honetan datza: (4.13) ekuazioarentzat $y_p = r^p$ motako soluzioak aurkitu nahi ditugu. (4.13) ekuazioan $p = n, n+1, \dots, n+k$ balioetarako $y_p = r^p$ ordezkatzuz eta lortzen dugun ekuazioa r^n terminoaz zatituz, pauso anitzeko metodo linealenzako egonkortasun-polinomioa lortzen da, zeina lehenengo kapituluan azaldu dugun metodologia erabilia lortuko genukeenaren berdina baita:

$$p(r, \hat{h}) = \sum_{j=0}^k \alpha_j r^j - \hat{h} \sum_{j=0}^k \beta_j r^j \quad (4.14)$$

$\hat{h} = \lambda h$ izanik.

(4.14) adierazpena zenbakizko metodoari lotutako bi polinomio karakteristikoaren menpe, honela idatz dezakegu:

$$p(r) = \sum_{j=0}^k \alpha_j r^j - \hat{h} \sum_{j=0}^k \beta_j r^j = \rho(r) - \hat{h} \sigma(r) \quad (4.15)$$

non: $\rho(r)$ eta $\sigma(r)$ (4.12) adierazpenak emanak datozen.

4.3. INTERPOLAZIO POLINOMIKOA, ATZERAKAKO DIFERENTZIAK

$P(t)$ polinomio bat, $\{(t_i, g_i) : i = 0, 2, \dots, p-1\}$ datu-multzoa interpolatzen duena, hau da, puntu horietatik igarotzen dena, oinarri bat osatzen duten $\{\phi_j(t) : j = 0, 1, \dots, p-1\}$ funtzioen konbinazio lineal bezala honela idatz daiteke:

$$P(t) = \sum_{j=0}^{p-1} x_j \phi_j(t) \quad (4.16)$$

x_j konstanteak zehaztu beharreko konstanteak izanik.

$\{(t_i, g_i) : i = 0, 2, \dots, p-1\}$ datu-multzoko p balioak h distantzia berak banatzen ditutenean, hau da: $h = t_{i+1} - t_i$ betetzen denean $i = 0, 1, 2, \dots, p-2$ balioetarako, (4.16) polinomioko $\phi_j(t)$ funtzioak eta x_j konstanteak hauek dira:

$$\phi_j(t) = \prod_{m=0}^{j-1} (t - t_{(p-1)-m}) \quad (4.17)$$

$$x_j = \nabla^j g_{p-1} \frac{1}{j! h^j} \quad (4.18)$$

$\nabla^j g_{p-1}$ terminoei atzerakako diferentzia deritze eta honela kalkulatu dira: $\nabla^j g_{p-1} = \nabla(\nabla^{j-1} g_{p-1})$, $\nabla g_{p-1} = g_{p-1} - g_{p-2}$. Atzerakako diferentziek hurrengo berdintza ere betetzen dute:

$$\nabla^{j+1} g_s = \nabla^j g_s - \nabla^j g_{s-1}, \quad \forall s \quad (4.19)$$

Eta kasu honetan lortzen den polinomio interpolatzaileari Newton-en atzerakako polinomio interpolatzaile deritzo, eta honela emana dator:

$$P(t) = \sum_{j=0}^{p-1} x_j \phi_j(t) = \sum_{j=0}^{p-1} \nabla^j g_{p-1} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t - t_{p-1-m}) \quad (4.20)$$

Newton-en atzerakako polinomio interpolatzaileak zenbait forma hartzen ditu interpolatu behar ditugun puntu kopuruaren arabera.

- Polinomioa $(k+1)$ puntutatik igarotzea nahi badugu, hau da, adibidez $\{(t_{n+i}, y_{n+i}) : i = 0, 1, 2, \dots, k\}$ puntuetatik igarotzea nahi badugu, (4.20) polinomioak forma hau hartzen du:

$$P(t) = y_{n+k} + \sum_{j=1}^k \nabla^j y_{n+k} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t - t_{n+k-m}) \quad (4.21)$$

- Polinomioa ondorengo k puntuetatik igarotzea nahi badugu $\{(t_{n+i}, f_{n+i}) : i = 1, 2, \dots, k\}$, $f_{n+i} = f(t_{n+i}, y_{n+i})$ izanik, (4.20) polinomioak forma hau hartzen du:

$$P(t) = f_{n+k} + \sum_{j=1}^{k-1} \nabla^j f_{n+k} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t - t_{n+k-m}) \quad (4.22)$$

- Eta polinomioa ondorengo k puntuetatik igarotzea nahi badugu $\{(t_{n+i}, f_{n+i}) : i = 0, 1, \dots, k-1\}$, $f_{n+i} = f(t_{n+i}, y_{n+i})$ izanik, (4.20) polinomioak forma hau hartzen du:

$$P(t) = f_{n+k-1} + \sum_{j=1}^{k-1} \nabla^j f_{n+k-1} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t - t_{n+k-1-m}) \quad (4.23)$$

4.4. ADAMS BASHFORTH-EN METODOAK

$y' = f(t, y)$, $y(t_0) = y_0$ hasierako baliodun ekuazio diferentzial arrunta emanik eta $t_n, t_{n+1}, \dots, t_{n+k-1}$ aldiuneetako balio zehatzen $y(t_n), y(t_{n+1}), \dots, y(t_{n+k-1})$ hurbilpenak, $y_n, y_{n+1}, \dots, y_{n+k-1}$ ezagunak ditugula suposatuz, Adams-ek hasierako baliodun ekuazio diferentzial arrunta era honetan kontsideratzen du:

$$y(t_{n+k}) = y(t_{n+k-1}) + \int_{t_{n+k-1}}^{t_{n+k}} f(t, y(t)) dt \quad (4.24)$$

$y_n, y_{n+1}, \dots, y_{n+k-1}$ zenbakizko balioak ezagutzen ditugunez, $t_n, t_{n+1}, \dots, t_{n+k-1}$ aldiuneetako deribatuen balio hurbilduak ere kalkula ditzakegu:

$$f_{n+j} = f(t_{n+j}, y_{n+j}), \quad j = 0, 1, \dots, k-1$$

Eta (4.24) adierazpeneko $y(t_{n+k-1})$ balio zehatza zenbakizko balio hurbilduarekin ordezkatzuz eta $f(t, y(t))$ funtzioa $\{(t_{n+j}, f_{n+j}) : j = 0, 1, \dots, k-1\}$ puntuetatik igarotzen den $P(t)$ polinomio interpolatzailearekin ordezkatzuz, hauxe dugu:

$$y_{n+k} = y_{n+k-1} + \int_{t_{n+k-1}}^{t_{n+k}} P(t) dt \quad (4.25)$$

$P(t)$ polinomio interpolatzaile hau (4.23) adierazpenak eman dago. Eta (4.25) adierazpeneko integrala egin ostean, Adams Bashforth-en metodoa lortzen da:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=0}^{k-1} \tilde{\gamma}_j \nabla^j f_{n+k-1} \quad (4.26)$$

$\tilde{\gamma}_j$ koefizienteen balioak 4.1. taulan jaso direnak izanik. (4.26) adierazpeneko atzerakako diferentzien garapena eginez gero, Adams Bashforth-en metodoetako beste adierazpide baliokide honetara iristen gara:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=1}^k \tilde{\beta}_j f_{n+k-j} \quad (4.27)$$

$\tilde{\beta}_j = (-1)^{j-1} \sum_{s=j-1}^{k-1} \tilde{\gamma}_s \binom{s}{j-1}$ izanik. $\tilde{\beta}_j$ konstanteen balioak 4.2. taulan jaso ditugu.

Adams Bashforth-en metodoetako formula batzuek hauek dira:

$$k=1: y_{n+1} = y_n + h f_n$$

$$k=2: y_{n+2} = y_{n+1} + h \left(\frac{3}{2} f_{n+1} - \frac{1}{2} f_n \right)$$

$$k=3: y_{n+3} = y_{n+2} + h \left(\frac{23}{12} f_{n+2} - \frac{4}{3} f_{n+1} + \frac{5}{12} f_n \right)$$

$$k=4: y_{n+4} = y_{n+3} + h \left(\frac{55}{24} f_{n+3} - \frac{59}{24} f_{n+2} + \frac{37}{24} f_{n+1} - \frac{9}{24} f_n \right)$$

j	0	1	2	3	4	5	6	7	8
$\tilde{\gamma}_j$	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$	$\frac{19.087}{60.480}$	$\frac{5.257}{17.280}$	$\frac{1.070.017}{3.628.800}$

4.1. taula. Adams Bashforth-en metodoko $\tilde{\gamma}_j$ konstanteen balioak.

k	$\tilde{\beta}_1$	$\tilde{\beta}_2$	$\tilde{\beta}_3$	$\tilde{\beta}_4$	$\tilde{\beta}_5$	$\tilde{\beta}_6$	$\tilde{\beta}_7$	$\tilde{\beta}_8$	C
1	1								$-\frac{1}{2}$
2	$\frac{3}{2}$	$-\frac{1}{2}$							$\frac{5}{12}$
3	$\frac{23}{12}$	$-\frac{4}{3}$	$\frac{5}{12}$						$-\frac{3}{8}$
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{3}{8}$					$\frac{251}{720}$
5	$\frac{1.901}{720}$	$-\frac{1.387}{360}$	$\frac{109}{30}$	$-\frac{637}{360}$	$\frac{251}{720}$				$-\frac{95}{288}$
6	$\frac{4.277}{1.440}$	$-\frac{2.641}{480}$	$\frac{4.991}{720}$	$-\frac{3.649}{720}$	$\frac{959}{480}$	$-\frac{95}{288}$			$\frac{19.087}{60.480}$
7	$\frac{198.721}{60.480}$	$-\frac{18.637}{2.520}$	$\frac{235.183}{20.160}$	$-\frac{10.754}{945}$	$\frac{135.713}{20.160}$	$-\frac{5.603}{2.520}$	$\frac{19.087}{60.480}$		$-\frac{5.257}{17.280}$
8	$\frac{16.083}{4.480}$	$-\frac{1.152.169}{120.960}$	$\frac{242.653}{13.440}$	$-\frac{296.053}{13.440}$	$\frac{2.102.243}{120.960}$	$-\frac{115.747}{13.440}$	$\frac{32.863}{13.440}$	$-\frac{5.257}{17.280}$	$\frac{1.070.017}{3.628.800}$

4.2. taula. Adams Bashforth-en metodoko $\tilde{\beta}_j$ konstanteen balioak.

k pausoko Adams Bashforth-en metodoek k ordena daukate eta beraien mozketa-errore lokala honela emana dator:

$$LTE = (-1)^k \tilde{\gamma}_k h^{k+1} y^{(k+1)}(t_n) + O(h^{k+2})$$

Adams Bashforth-en metodoak test-ekuazioari aplikatu ostean, diferentziant emanda dagoen ondorengo ekuazioa lortzen da:

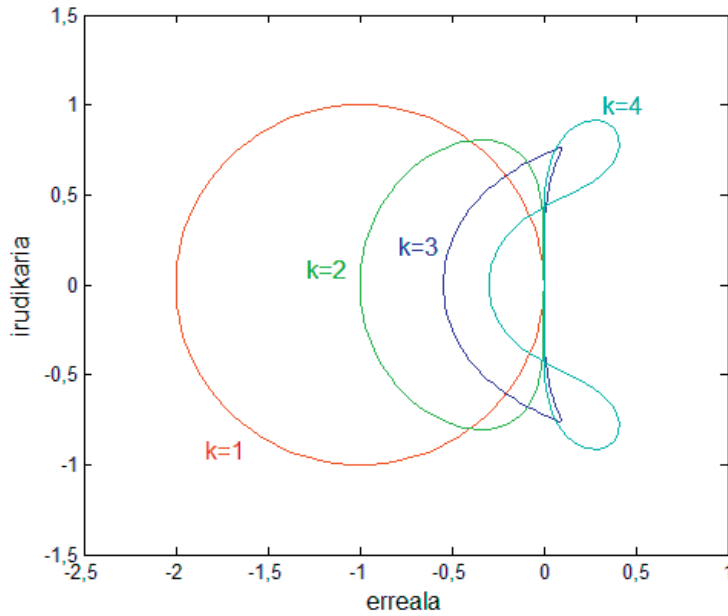
$$y_{n+k} = y_{n+k-1} + \hat{h} \sum_{j=1}^k \tilde{\beta}_j y_{n+k-j}, \quad \hat{h} = h\lambda \quad (4.28)$$

(4.28) ekuazioan $y_p = r^p$ ordezkatu, r^n -z zatitu eta $\hat{h}$ askatu ondoren, adierazpen honetara iristen gara:

$$\hat{h} = \frac{r^k - r^{k-1}}{\sum_{j=1}^k \tilde{\beta}_j r^{k-j}} \quad (4.29)$$

Eta (4.29) ekuazioan $r = e^{i\theta}$ eginez eta $\theta \in [0, 2\pi)$ balioak emanez, egonkortasun-eremuk mugaz izango dugu. 4.1. irudian Adams Bashforth-en

metodoen egonkortasun-eremuak ikus daitezke. $k=1$ kasuko Adams Bashforth-en metodoa Euler-en metodo esplizitua da eta $(-1,0)$ zentroan eta unitatedun erradioa duen zirkunferentzia da haren egonkortasun-eremuaren muga. Ordenan aurrera egin ahala, egonkortasun-eremuak gero eta txikiagoak direla ikus daiteke irudi honetan.



4.1. irudia. Adams Bashforth-en zenbait egonkortasun-eremu (kurben barruko aldeak).

4.5. ADAMS MOULTON-EN METODOAK

Berriz ere $y' = f(t, y)$, $y(t_0) = y_0$ hasierako baliodun ekuazio diferentzial arrunta emanik eta $t_{n+1}, t_{n+2}, \dots, t_{n+k-1}$ aldiuneetako balio zehatzen $y(t_{n+1}), y(t_{n+2}), \dots, y(t_{n+k-1})$ zenbakizko hurbilpenak $y_{n+1}, y_{n+2}, \dots, y_{n+k-1}$ ezagunak ditugula suposatuz, (4.24) adierazpenean $y(t_{n+k-1})$ balio zehatza zenbakizko balio hurbilduarekin ordezkatzuko dugu eta $f(t, y(t))$ funtzioa $\{(t_{n+j}, f_{n+j}) : j = 1, 2, \dots, k\}$ puntuetatik igarotzen den $P(t)$ polinomio interpolatzailearekin ordezkatzuko dugu:

$$y_{n+k} = y_{n+k-1} + \int_{t_{n+k-1}}^{t_{n+k}} P(t) dt \quad (4.30)$$

Kasu honetan, (4.30) adierazpenean ordezkaturiko dugun $P(t)$ polinomio interpolatzailearen adierazpena (4.22) izango da. (4.30) adierazpeneko integrala egin ondoren, Adams Moulton-en formulak lortzen dira:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=0}^{k-1} \gamma_j^* \nabla^j f_{n+k} \quad (4.31)$$

γ_j^* koefizienteen balioak 4.3. taulan jaso dira. (4.31) adierazpeneko atzerakako diferentzien garapenak eginez, Adams Moulton-en metodoetako beste adierazpide baliokide honetara iristen gara:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=1}^k \beta_j^* f_{n+k+1-j} \quad (4.32)$$

$\beta_j^* = (-1)^j \sum_{s=j}^{k-1} \gamma_s^* \binom{s}{j}$ izanik. β_j^* konstanteen balioak 4.4. taulan jaso ditugu.

Adams Moulton-en metodoetako formula batzuk hauek dira:

$$\begin{aligned} k=1: y_{n+1} &= y_n + hf_{n+1} \\ k=2: y_{n+2} &= y_{n+1} + h \left(\frac{1}{2} f_{n+2} + \frac{1}{2} f_{n+1} \right) \\ k=3: y_{n+3} &= y_{n+2} + h \left(\frac{5}{12} f_{n+3} + \frac{8}{12} f_{n+2} - \frac{1}{12} f_{n+1} \right) \\ k=4: y_{n+4} &= y_{n+3} + h \left(\frac{9}{24} f_{n+4} + \frac{19}{24} f_{n+3} - \frac{5}{24} f_{n+2} + \frac{1}{24} f_{n+1} \right) \end{aligned}$$

$k=1$ ordenako Adams Moulton-en metodoa Euler-en metodo implizitua da, eta $k=2$ denean, Adams Moulton-en metodoa, berriz, metodo trapezoidala da.

k -pausoko Adams Moulton-en metodoek ere k ordena dute eta beraien mozketa-errore lokalaren adierazpena hauxe da:

$$LTE = (-1)^k \gamma_k^* h^{k+1} y^{(k+1)}(t_{n-k+1}) + O(h^{k+2})$$

j	0	1	2	3	4	5	6	7	8
γ_j^*	1	$-\frac{1}{2}$	$-\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{19}{720}$	$-\frac{3}{160}$	$-\frac{863}{60.480}$	$-\frac{275}{24.192}$	$-\frac{33.953}{3.628.800}$

4.3. taula. Adams Moulton-en metodoko γ_j^* konstanteen balioak.

k	β_1^*	β_2^*	β_3^*	β_4^*	β_5^*	β_6^*	β_7^*	β_8^*	C
1	1								$\frac{1}{2}$
2	$\frac{1}{2}$	$\frac{1}{2}$							$-\frac{1}{12}$
3	$\frac{5}{12}$	$\frac{2}{3}$	$-\frac{1}{12}$						$\frac{1}{24}$
4	$\frac{3}{8}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$					$-\frac{19}{720}$
5	$\frac{251}{720}$	$\frac{323}{360}$	$-\frac{11}{30}$	$\frac{53}{360}$	$-\frac{19}{720}$				$\frac{3}{160}$
6	$\frac{95}{288}$	$\frac{1.427}{1.440}$	$-\frac{133}{240}$	$\frac{241}{720}$	$-\frac{173}{1.440}$	$\frac{3}{160}$			$-\frac{863}{60.480}$
7	$\frac{19.087}{60.480}$	$\frac{2.713}{2.520}$	$-\frac{15.487}{20.160}$	$\frac{586}{945}$	$-\frac{6.737}{20.160}$	$\frac{263}{2.520}$	$-\frac{863}{60.480}$		$\frac{275}{24.192}$
8	$\frac{5.257}{17.280}$	$\frac{139.849}{120.960}$	$-\frac{4.511}{4.480}$	$\frac{123.133}{120.960}$	$-\frac{88.547}{120.960}$	$\frac{1.537}{4.480}$	$-\frac{11.351}{120.960}$	$\frac{275}{24.192}$	$-\frac{33.953}{3.628.800}$

4.4. taula. Adams Moulton-en metodoko β_j^* konstanteen balioak.

Adams Moulton-en metodoen egonkortasun-eremuak kalkulatzeko metodoa aplikatzen zaio test-ekuazioari eta diferentziatan emanda dagoen ondoko ekuazioa lortzen da:

$$y_{n+k} = y_{n+k-1} + \hat{h} \sum_{j=1}^k \beta_j^* y_{n+k+1-j}, \quad \hat{h} = h\lambda \quad (4.33)$$

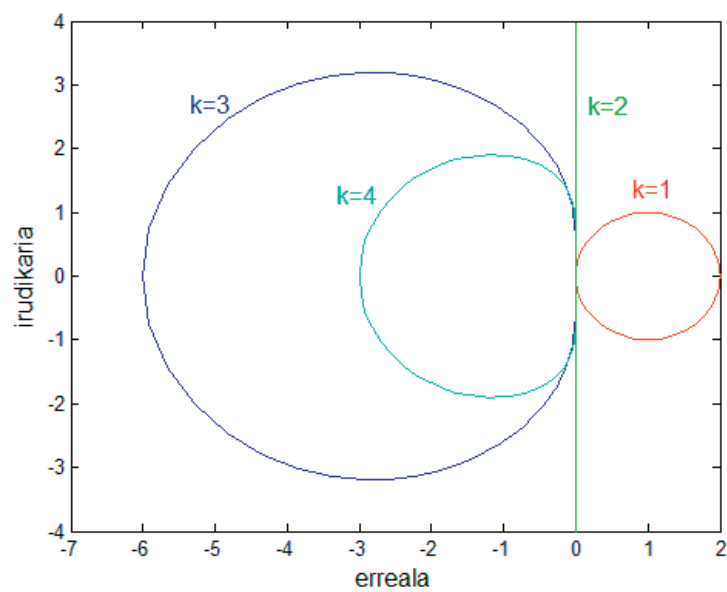
Adams Bashforth-en metodoekin egin dugun bezala, (4.33) ekuazioan $y_p = r^p$ ordezkatu, r^n -z zatitu eta $\hat{h}$ askatu ondoren, adierazpen honetara iristen gara:

$$\hat{h} = \frac{r^k - r^{k-1}}{\sum_{j=1}^k \beta_j^* r^{k+1-j}} \quad (4.34)$$

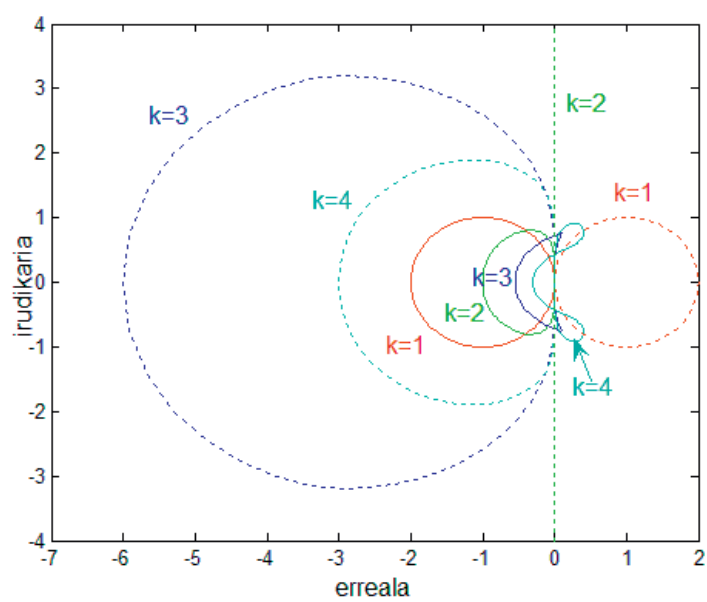
Eta (4.34) adierazpenean $r = e^{i\theta}$ eginez eta $\theta \in [0, 2\pi)$ balioak emanez, egonkortasun-eremuko muga lortuko dugu. Adams Moulton-en egonkortasun-eremuak 4.2. irudian marraztu dira. $k=1$ denean daukagun Adams Moulton-en metodoa Euler-en metodo inplizitua denez, $(1,0)$ zentroan eta bat unitateko erradiodun zirkuluaren kanpoko aldean da metodoaren egonkortasun-eremua. $k=2$ denean daukagun Adams Moulton-en metodoa metodo trapezoidal da, eta haren egonkortasun-eremua plano konplexuaren ezker alde osoa da. $k=3$ eta $k=4$ kasuetan, irudikatu diren kurben barneko aldeak dira metodoari dagozkien egonkortasun-eremuak. Adams Moulton-en metodoen egonkortasun-eremuak Adams Bashforth-enak baino handiagoak direla ikus daiteke 4.3. irudian, zeinean lerro jarraituz Adams Bashforth-en egonkortasun-eremuak irudikatu ditugun eta marraztutako lerroz Adams Moulton-enak. Hala eta guztiz, bakarrik $k=1$ eta $k=2$ kasuetan da Adams Moulton-en egonkortasun-eremuaren parte plano konplexuaren ezker alde osoa. Hau da, bakarrik $k=1$ eta $k=2$ kasuetan dira Adams Moulton-en metodoak A-egonkorak.

Adams Bashforth eta Adams Moulton-en metodoetako diferentzia nagusiak hauek dira:

- Adams Bashforth-en metodoak esplizituak dira eta Adams Moulton-enak inplizituak.
- Adams Moulton-en metodoetako koefizienteak Adams Bashforth-en metodoetakoak baino txikiagoak dira. Hau da: $|\gamma_i^*| < |\tilde{\gamma}_i|$. Familia bietako mozketaren errore lokalak koefiziente hauen menpe daudenez, ordena bererako Adams Moulton-en metodoek Adams Bashforth-en metodoek baino mozketaren errore lokal txikiagoa daukate.
- Ordena bererako, Adams Moulton-en metodoei dagozkien egonkortasun-eremuak Adams Bashforth-enak baino handiagoak dira. Metodo inplizituek esplizituekiko izan ohi duten abantaila izaten da hori askotan.



4.2. irudia. Adams Moulton-en metodoen zenbait egonkortasun-eremu.



4.3. irudia. Adams Bashforth eta Adams Moulton-en egonkortasun-eremuak.

4.6. ADAMS-EN IRAGARLE-ZUZENTZAILE ESKEMAK

Adams-en metodoak iragarle-zuzentzaile forman ere erabil daitezke. Iragarle-zuzentzaile eskema bi PECE (*Predict Evaluate Correct Evaluate*) eta PEC (*Predict Evaluate Correct*) dira. Lehenengoan, Adams Bashforth-en metodoa erabilita $\tilde{y}_{n+k}$ iragartzen da, iragarpen horren deribatua $f(t_{n+k}, \tilde{y}_{n+k})$ ebaluatzen da, Adams Moulton-en metodoa erabilita aurreko balioa zuzentzen da y_{n+k}^* eta azkenik zuzenketaren deribatua ebaluatzen da $f(t_{n+k}, y_{n+k}^*)$. Bigarreanean, PEC eskeman, Adams Bashforth-en metodoa erabilita $\tilde{y}_{n+k}$ iragartzen da, iragarpen horren deribatua $f(t_{n+k}, \tilde{y}_{n+k})$ ebaluatzen da eta Adams Moulton-en metodoa erabilita aurreko balioa zuzentzen da y_{n+k}^* lortuz. Adams Bashforth-en metodoaz kalkulatu diren balioei $\tilde{y}_{n+i}$ deituko diegu eta Adams Moulton-en metodoaz kalkulatu y_{n+i}^* .

PECE izenaz ezagutzen den eskemako pausoak ikusiko ditugu. Iragartzeko darabilgun Adams Bashforth-en metodoa eta zuzentzeko darabilgun Adams Moulton-en metodoa, biak, k pausokoak kontsideratuko ditugu. Hauek dira PECE eskemako pausoak:

- Aurreko pausoetan Adams Moulton-en metodoaz kalkulatuak $y_n^*, y_{n+1}^*, \dots, y_{n+k-1}^*$ balioak ditugu. Balio horiek erabiliz, Adams Bashforth-en k ordenako (4.27) adierazpenaz $\tilde{y}_{n+k}$ balioa iragarriko dugu:

$$\tilde{y}_{n+k} = y_{n+k-1}^* + h \sum_{j=1}^k \tilde{\beta}_j f_{n+k-j}^* \quad (4.35)$$

$f_{n+k-j}^* = f(t_{n+k-j}, y_{n+k-j}^*)$ izanik.

- (4.35) adierazpena erabilita kalkulatu dugun $\tilde{y}_{n+k}$ balioaren deribatua ebaluatzen da: $\tilde{f}_{n+k} = f(t_{n+k}, \tilde{y}_{n+k})$.
- Adams Bashforth-ekin egin dugun iragarpena zuzenduko dugu, Adams Moulton-en k -ordenako (4.32) adierazpena erabilita:

$$y_{n+k}^* = y_{n+k-1}^* + h\beta_1^* \tilde{f}_{n+k} + h \sum_{j=2}^k \beta_j^* f_{n+k+1-j}^* \quad (4.36)$$

y_{n+k}^* balioa da prozesu honetan lortu dugun balioa.

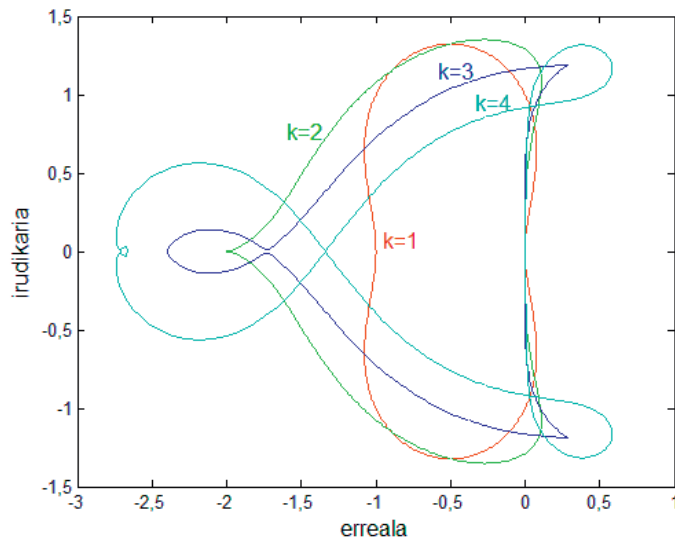
- (4.36) adierazpena erabilia kalkulatu dugun y_{n+k}^* balio zuzenduaren deribatua ebaluatzen da: $f(t_{n+k}, y_{n+k}^*)$. Eta berriro iragarpeneko pausora pasatuko ginateke.

Metodo hauen egonkortasun-azterketa egiteko, metodoa test-ekuazioari aplikatzen zaio, $y_p = r^p$ ordezkatzan dugu eta r^n terminoaz zatituz $\hat{h}$ aldagaian koadratikoa den ekuazioa lortzen da:

$$A\hat{h}^2 + B\hat{h} + C = 0 \quad (4.37)$$

$$\begin{cases} A = \left(\sum_{j=0}^{k-1} \gamma_j^* \right) \left(\sum_{j=0}^{k-1} \tilde{\gamma}_j \left(1 - \frac{1}{r} \right)^j \right) \\ B = (1-r) \sum_{j=0}^{k-1} \gamma_j^* + r \sum_{j=0}^{k-1} \gamma_j^* \left(1 - \frac{1}{r} \right)^j \\ C = 1 - r \end{cases} \quad (4.38)$$

PECE eskemari dagozkion egonkortasun-eremuen grafikoak 4.4. irudian daude ikusgai.



4.4. irudia. PECE eskemen egonkortasun-eremuak (kurben barruko aldeak), Adams Bashforth eta Adams Moulton-en metodoak biak k ordenakoak izanik.

Deribatuaren azken ebaluazioa ez egiteaz gain, PEC eskemako pauso iragarlean eta zuzentzailean ez dira PECEko balio berak erabiltzen. PEC prozesu osoa honexetan datza:

- Aurreko pausoetan Adams Bashforth-en metodoaz kalkulaturako $\tilde{y}_n, \tilde{y}_{n+1}, \dots, \tilde{y}_{n+k-1}$ balioak ditugu. Balio horietako deribatuaren ebaluazioa erabiliz eta aurreko pausoan Adams Moulton-en metodoaz lortutako y_{n+k-1}^* balioa erabiliz, Adams Bashforth-en k ordenako (4.27) adierazpenaz $\tilde{y}_{n+k}$ balioa iragarriko dugu:

$$\tilde{y}_{n+k} = y_{n+k-1}^* + h \sum_{j=1}^k \tilde{\beta}_j \tilde{f}_{n+k-j} \quad (4.39)$$

$\tilde{f}_{n+k-j} = f(t_{n+k-j}, \tilde{y}_{n+k-j})$ izanik.

- (4.39) adierazpena erabilia kalkulatu dugun $\tilde{y}_{n+k}$ balioaren deribatua ebaluatzen da: $\tilde{f}_{n+k} = f(t_{n+k}, \tilde{y}_{n+k})$.
- Iragarri dugun balioari zuzenketa egingo diogu Adams Moulton-en k ordenako (4.32) adierazpena erabilia:

$$y_{n+k}^* = y_{n+k-1}^* + h\beta_1^* \tilde{f}_{n+k} + h \sum_{j=2}^k \beta_j^* \tilde{f}_{n+k+1-j} \quad (4.40)$$

Eta y_{n+k}^* balioa da prozesu honetan lortu dugun balioa.

PEC eskemako egonkortasun-eremuak formula honen bidez emanak datoz:

$$\hat{h} = \frac{-r^k \rho_k^*(r)}{\tilde{\rho}_k(r) \sigma_k^*(r) - \rho_k^*(r) \tilde{\sigma}_k(r) - r^k \sigma_k^*(r)} \quad (4.41)$$

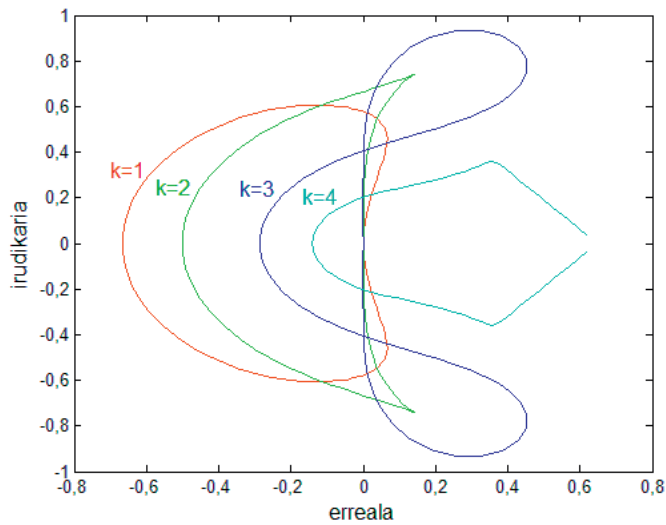
$\tilde{\sigma}_k$ eta $\tilde{\rho}_k$ Adams Bashforth-en metodoko polinomio karakteristikoak izanik eta σ_k^* eta ρ_k^* Adams Moulton-en metodokoak. (4.2) adierazpenaz emanda dagoen pauso anitzeko metodo lineal baten polinomio karakteristikoak (4.12) erabiliz kalkula daitezkeela esan da arestian. Adams Bashforth eta Adams Moulton-en kasuan polinomio karakteristiko horiek forma hau hartzen dute:

- Adams Bashforth-en kasuan: $\tilde{\rho}_k(r) = r^k - r^{k-1}$ eta $\tilde{\sigma}_k(r) = \sum_{j=1}^k \tilde{\beta}_j r^{k-j}$
- Adams Moulton-en kasuan: $\rho_k^*(r) = r^k - r^{k-1}$ eta $\sigma_k^*(r) = \sum_{j=1}^k \beta_j^* r^{k+1-j}$

Hauek lirateke, $\tilde{\sigma}_k$ eta σ_k^* polinomioetako batzuk:

$$\left\{ \begin{array}{l} \tilde{\sigma}_1(r) = 1, \\ \tilde{\sigma}_2(r) = \frac{3}{2}r - \frac{1}{2}, \\ \tilde{\sigma}_3(r) = \frac{23}{12}r^2 - \frac{4}{3}r + \frac{5}{12}, \\ \tilde{\sigma}_4(r) = \frac{55}{24}r^3 - \frac{59}{24}r^2 + \frac{37}{24}r - \frac{3}{8} \\ \vdots \end{array} \right. \quad \left\{ \begin{array}{l} \sigma_2^*(r) = r, \\ \sigma_2^*(r) = \frac{1}{2}r^2 + \frac{1}{2}r, \\ \sigma_3^*(r) = \frac{5}{12}r^3 + \frac{2}{3}r^2 - \frac{1}{12}r, \\ \sigma_4^*(r) = \frac{3}{8}r^4 + \frac{19}{24}r^3 - \frac{5}{24}r^2 + \frac{1}{24}r \\ \vdots \end{array} \right. \quad (4.42)$$

4.5. irudian ikus daitezke PEC eskemen zenbait egonkortasun-eremu.



4.5. irudia. PEC eskemen egonkortasun-eremuak (kurben barruko aldeak), Adams Bashforth eta Moulton biak k ordenakoak direnean.

PEC eta PECE iragarle-zuzentzaile eskemez gain, PECEC edota PECECE eskemak izatea ere posiblea da. Oro har, Adams-en metodoek onartzen dituzten iragarle-zuzentzaile eskemek $P(EC)^n$ eta $P(EC)^n E$ formak izan ditzakete. $P(EC)^n$ eskeman egiten dena da, Adams Bashforth erabilia aurreikusi, lortu den balioaren deribatua ebaluatu eta Adams Moulton erabilia zuzendu. Lortu den balioaren deribatua ebaluatzea eta Adams Moulton-ekin zuzentzea n aldiz egin daiteke eta horregatik adierazten da forma hau $P(EC)^n$ bezala. $P(EC)^n E$ eskeman, Adams Bashforth-ekin iragarriko dugu eta lortutako balioaren deribatua ebaluatzea eta Adams Moulton-ekin zuzentzea n aldiz egingo dugu. Azkenik, lortzen den y_{n+k}^* balioari dagokion deribatua $f(t_{n+k}, y_{n+k}^*)$ kalkulatzen da. PEC eskema $P(EC)^n$ eskemaren kasu berezia da $n=1$ izanik eta PECE eskema $P(EC)^n E$ eskemaren kasu berezia da $n=1$ baliorako.

4.7. BACKWARD DIFFERENTIATION FORMULAE METODOAK

BDFak pauso anitzeko metodo linealak dira, deribatuaren ebaluazio bakarra darabiltenak. Adams-en metodoek y_{n+k} balioa kalkulatzeko bakarrik y -ren balio bat erabiltzen dute eta gainerakoak deribatuaren balioak dira. BDFetan alderantzizkoa gertatzen da: deribatuaren ebaluazio bakarra darabilte eta gainerakoak y -ren balioak dira. BDF metodoen formula ondorioztatzeko, $\{(t_{n+i}, y_{n+i}) : i = 0, \dots, k\}$ puntuetatik igarotzen den $P(t)$ polinomioa behar dugu, (4.21) formularen bidez emana datorrena. Kontuan izan aurreko k pausoko zenbakizko balioak baditugula, $y_n, \dots, y_{n+k-1}$ eta y_{n+k} balioa kalkulatu nahi dela. Ezezaguna den y_{n+k} balioa $P(t)$ polinomioak t_{n+k} aldiunean ondoko baldintza betetzeko eran kalkulatzen da:

$$P'(t_{n+k}) = f(t_{n+k}, y_{n+k}) \quad (4.43)$$

$P'(t)$ kalkulatuz eta bertan $t = t_{n+k}$ ordezkatzuz, BDFen formulara iristen gara:

$$\sum_{j=1}^k \frac{1}{j} \nabla^j y_{n+k} = h f_{n+k} \quad (4.44)$$

(4.44) adierazpeneko atzerakako diferentziak garatzen baditugu, BDFen beste adierazpen baliokide hau lortzen da:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = hf_{n+k} \quad (4.45)$$

Non $\hat{\alpha}_j$ konstanteak 4.5. taulan jaso diren.

BDFei dagozkien zenbait formula:

$$k=1: y_{n+1} - y_n = hf_{n+1}$$

$$k=2: \frac{3}{2}y_{n+2} - 2y_{n+1} + \frac{1}{2}y_n = hf_{n+2}$$

$$k=3: \frac{11}{6}y_{n+3} - 3y_{n+2} + \frac{3}{2}y_{n+1} - \frac{1}{3}y_n = hf_{n+3}$$

$$k=4: \frac{25}{12}y_{n+4} - 4y_{n+3} + 3y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{4}y_n = hf_{n+4}$$

$$k=5: \frac{137}{60}y_{n+5} - 5y_{n+4} + 5y_{n+3} - \frac{10}{3}y_{n+2} + \frac{5}{4}y_{n+1} - \frac{1}{5}y_n = hf_{n+5}$$

$$k=6: \frac{147}{60}y_{n+6} - 6y_{n+5} + \frac{15}{2}y_{n+4} - \frac{20}{3}y_{n+3} + \frac{15}{4}y_{n+2} - \frac{6}{5}y_{n+1} + \frac{1}{6}y_n = hf_{n+6}$$

k	$\hat{\alpha}_0$	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\alpha}_3$	$\hat{\alpha}_4$	$\hat{\alpha}_5$	$\hat{\alpha}_6$	C
1	-1	1						$-\frac{1}{2}$
2	$\frac{1}{2}$	-2	$\frac{3}{2}$					$-\frac{1}{3}$
3	$-\frac{1}{3}$	$\frac{3}{2}$	-3	$\frac{11}{6}$				$-\frac{1}{4}$
4	$\frac{1}{4}$	$-\frac{4}{3}$	3	-4	$\frac{25}{12}$			$-\frac{1}{5}$
5	$-\frac{1}{5}$	$\frac{5}{4}$	$-\frac{10}{3}$	5	-5	$\frac{137}{60}$		$-\frac{10}{6}$
6	$\frac{1}{6}$	$-\frac{6}{5}$	$\frac{15}{4}$	$-\frac{20}{3}$	$\frac{15}{2}$	-6	$\frac{147}{60}$	$-\frac{1}{7}$

4.5. taula. BDF metodoetako koefizienteak eta C errore-konstantea.

k pausoko BDFak k ordena dauka eta haren mozketaren errorea lokalak honela emana dator:

$$LTE = C_{k+1} h^{k+1} y^{(k+1)}(t_n) + O(h^{k+2}) \quad (4.46)$$

$$\text{Non: } C_{k+1} = \frac{-1}{\gamma_k (k+1)}, \quad \gamma_k = \hat{\alpha}_k = \sum_{j=1}^k \frac{1}{j} = \begin{cases} 1, & k=1 \\ 3/2, & k=2 \\ 11/6, & k=3 \\ 25/12, & k=4 \\ 137/60, & k=5 \\ \vdots \end{cases} \quad (4.47)$$

Eta (4.46) adierazpeneko C_{k+1} errore-konstantea normalizatzen badugu, (4.11) adierazpenak emana datorren C errore-konstantea hauxe litzateke BDFen kasuan:

$$C = \frac{C_{k+1}}{\rho'(1)} = \frac{-1}{k+1} \quad (4.48)$$

$$\rho'(1) = \frac{1}{\gamma_k} \text{ baita.}$$

Adibidea

$k=2$ kasuko BDFaren mozketaren errorea lokalak kalkulatu dugu. BDF2 metodoaren adierazpena hauxe da:

$$y_{n+k} = \frac{4}{3} y_{n+k-1} - \frac{1}{3} y_{n+k-2} + \frac{2}{3} h f_{n+k} \quad (4.49)$$

$k=2$ kasuan gaudenez eta $f = y'$ kontuan hartuta, (4.49) adierazpena honela idatziko dugu:

$$y_{n+2} = \frac{4}{3} y_{n+1} - \frac{1}{3} y_n + \frac{2}{3} h y'_{n+2}$$

Kasu honetan konstanteen balioak $\alpha_0 = \frac{1}{3}, \alpha_1 = -\frac{4}{3}, \alpha_2 = 1, \beta_2 = \frac{2}{3}$ dira eta BDF2 metodoaren bi polinomio karakteristikoak honela datoz emanak:

$$\begin{cases} \rho(r) = \sum_{j=0}^k \alpha_j r^j = \frac{1}{3} - \frac{4}{3}r + r^2 \\ \sigma(r) = \sum_{j=0}^k \beta_j r^j = \frac{2}{3}r^2 \end{cases} \quad (4.50)$$

Mozketa-errore lokalaren definizioa erabilita zera daukagu:

$$LTE = y(t_{n+2}) - y_{n+2}^* \quad (4.51)$$

$$y_{n+2}^* = \frac{4}{3}y(t_{n+1}) - \frac{1}{3}y(t_n) + \frac{2}{3}h(y_{n+2}^*)' \text{ izanik.}$$

Hainbat autorek diote ezen mozketa-errore lokala hobeto karakterizatua geratzen dela y_{n+2}^* balioaren adierazpeneko deribatua kalkulatzeko $(y_{n+2}^*)'$ erabiltzen dugunean eta ez $y'(t_{n+2})$ darabilgunean. y_{n+2}^* balioaren adierazpenean erabiltzen dugun deribatuaren arabera, mozketa-errore lokalean agertzen zaigun konstantea ezberdina da:

- $(y_{n+2}^*)'$ darabilgunean mozketa-errore lokalean azaltzen den konstantea $C = \frac{C_{k+1}}{\rho'(1)}$ da.
- $y'(t_{n+2})$ darabilgunean mozketa-errore lokalean agertzen den konstantea, aldiz, C_{k+1} da.

C_3 eta $\rho'(1)$ (4.47) adierazpena erabiliz kalkula badaitezke ere, guk mozketa-errore lokala kalkulatzeko prozesu osoa garatuko dugu hemen.

Mozketa-errore lokala kalkulatzeko t_n puntuan zentratutako Taylor-en garapen hauek erabiliko ditugu:

$$\begin{cases} y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{1}{2!}h^2 y''(t_n) + \frac{1}{3!}h^3 y'''(t_n) + \dots \\ y(t_{n+2}) = y(t_n) + 2hy'(t_n) + 2h^2 y''(t_n) + \frac{4}{3}h^3 y'''(t_n) + \dots \\ y'(t_{n+2}) = y'(t_n) + 2hy''(t_n) + 2h^2 y'''(t_n) + \dots \end{cases} \quad (4.52)$$

Guk $y_{n+2}^* = \frac{4}{3}y(t_{n+1}) - \frac{1}{3}y(t_n) + \frac{2}{3}hy'(t_{n+2})$ adierazpena erabilia kalkulatu

dugu mozketaren errorea lokal eta ondoren dituko dugu C konstantearen balioa. Behar ditugun Taylor-en garapenak mozketaren erroreak lokalaren (4.51) adierazpenarekin ordezkatzuz, zera daukagu:

$$\begin{aligned} LTE = & \left(y(t_n) + 2hy'(t_n) + 2h^2y''(t_n) + \frac{4}{3}h^3y'''(t_n) \dots \right) \\ & - \frac{4}{3} \left(y(t_n) + hy'(t_n) + \frac{1}{2!}h^2y''(t_n) + \frac{1}{3!}h^3y'''(t_n) + \dots \right) \\ & + \frac{1}{3}y(t_n) - \frac{2}{3}h \left(y'(t_n) + 2hy''(t_n) + 2h^2y'''(t_n) + \dots \right) \end{aligned} \quad (4.53)$$

(4.53) adierazpenarekin terminoak berrordenatuz, hauxe daukagu:

$$\begin{aligned} LTE = & y(t_n) \left(1 - \frac{4}{3} + \frac{1}{3} \right) + hy'(t_n) \left(2 - \frac{4}{3} - \frac{2}{3} \right) \\ & + h^2y''(t_n) \left(2 - \frac{2}{3} - \frac{4}{3} \right) + h^3y'''(t_n) \left(\frac{4}{3} - \frac{4}{18} - \frac{4}{3} \right) + O(h^4) \end{aligned} \quad (4.54)$$

Eta (4.54) adierazpenarekin eragiketarik egitea baino ez zaigu geratzen BDF2 metodoaren mozketaren errorea lokal izateko:

$$LTE = -\frac{2}{9}h^3y'''(t_n) + O(h^4) \quad (4.55)$$

Lortu dugun mozketaren errorea lokal hirugarren ordenakoa denez, metodoa bigarren ordenakoa da. (4.55) adierazpenarekin azaltzen zaigun errorea-konstantea $C_3 = -2/9$ da. C konstantea kalkulatzeko $\rho'(1)$ edota $\sigma(1)$ kalkulatu behar ditugu. (4.50) adierazpenetik zera daukagu:

$$\begin{cases} \rho'(r) = -\frac{4}{3} + 2r \Rightarrow \rho'(1) = \frac{2}{3} \\ \sigma(1) = \frac{2}{3} \end{cases} \quad (4.56)$$

Eta C errorea-konstante normalizatua kalkulatzeko behar ditugun osagai guztiak ditugu:

$$C = \frac{C_3}{\rho'(1)} = \frac{C_3}{\sigma(1)} = \frac{-2/9}{2/3} = -\frac{1}{3}$$

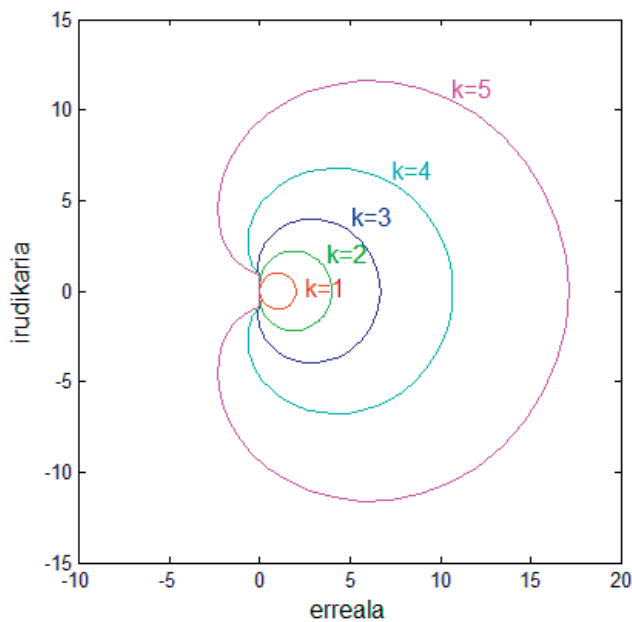
Test-ekuazioari BDF metodoak aplikatu ostean lortzen den ekuazioa hauxe da:

$$\hat{h} = \sum_{j=1}^k \frac{1}{j} \left(1 - \frac{1}{r}\right)^j$$

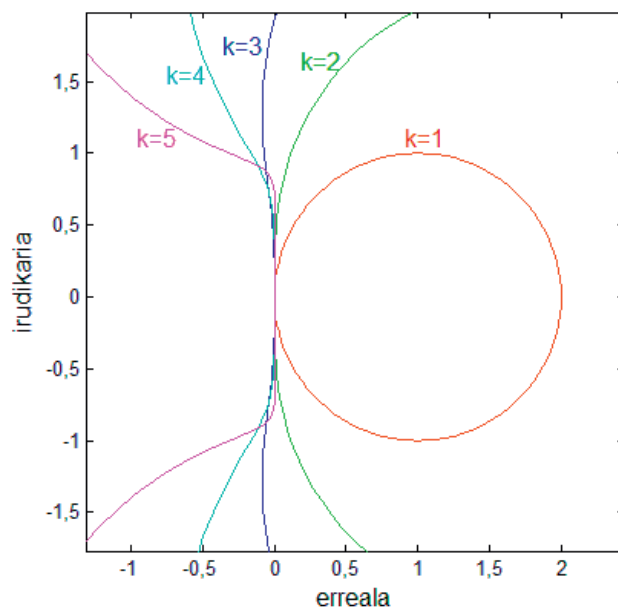
Beti bezala, $r = e^{i\theta}$ eginez eta θ angeluari balioak emanez, egonkortasun-eremuko muga marraz daiteke. 4.6. taulan jaso dira BDF metodoen egonkortasun-ezaugarriak eta 4.6. eta 4.7. irudietan marraztu dira metodo horien egonkortasun-eremuak.

k	1	2	3	4	5	6
$A(\alpha)$	90°	90°	86,03°	73,35°	51,84°	17,84°

4.6. taula. BDF metodoen egonkortasun-ezaugarriak.



4.6. irudia. BDF metodoen egonkortasun-eremuak (kurben kanpoko aldeak).



4.7. irudia. BDF metodoen egonkortasun-eremuak hurbilagotik (kurben kanpoko aldeak).

5. Pauso anitzeko zenbait metodo lineali egindako hainbat aldaketa

Hirugarren kapituluaren Runge-Kutta metodoak errebisatu ditugu eta 4.ean pauso anitzeko zenbait metodo lineal aurkeztu ditugu. Azken urteetan lan asko egin izan da ordena altuko eta egonkortasun-eremu zabaleko zenbakizko metodoak lortzeko. Guk pauso anitzeko metodo linealetan egindako aurrerapenei begiratuko diegu. Zehaztasun- eta egonkortasun-ezaugarri oneko pauso anitzeko metodoak lortzeko, bi norabideri jarraitu zaie bereziki:

- Alde batetik, ordena altuagoko deribatuak erabili dira zenbakizko metodoaren formulaz.
- Eta bestetik, super-etorkizuneko puntuak erabili dira.

Kapitulu honetan norabide biei jarraituz lortu diren metodoetako batzuk ikusiko ditugu.

5.1. BIGARREN DERIBATUADARABILTEN PAUSO ANITZEKO METODOAK

$y' = f(t, y)$ EDA emanik, Taylor-en bigarren ordenako serie gisako garapena erabiliz, honela kalkulatu ahal izango litzateke y_{n+1} balioa:

$$y_{n+1} = y_n + hf_n + \frac{h^2}{2!} g_n \quad (5.1)$$

$f_n = f(t_n, y_n)$, $g(t, y) = y'' = f_t + f_y y' = f_t + f_y f$ eta $g_n = g(t_n, y_n)$ izanik.

Bigarren deribatua erabiltzen duen pauso anitzeko zenbakizko metodo lineal baten adierazpen orokorra honela emana dator:

$$\sum_{i=0}^k \alpha_i y_{n+i} = h \sum_{i=0}^k \beta_i f_{n+i} + h^2 \sum_{i=0}^k \gamma_i g_{n+i} \quad (5.2)$$

Bakarrrik lehen deribatua darabilten pauso anitzeko metodoen kasuan erabilitako antzeko prozedura erabilia, 2. deribatua darabilten metodoen kasuan ere, metodoa p ordenakoa izateko α_i , β_i eta γ_i parametroek bete behar dituzten baldintzetara iristen gara:

$$\sum_{i=0}^k \alpha_i i^q = q \sum_{i=0}^k \beta_i i^{q-1} + q(q-1) \sum_{i=0}^k \gamma_i i^{q-2}, \quad 0 \leq q \leq p \quad (5.3)$$

2. deribatua darabilten metodoen egonkortasun-azterketa ere metodoa test-ekuazioari aplikatuz egiten da. Hau da, $y' = \lambda y$ ekuazioari zenbakizko metodoa aplikatzen diogu, zeinentzat $y'' = (\lambda y)' = \lambda^2 y$ betetzen den. Adierazpen hauek (5.2) ekuazioan ordezkaturata egonkortasun-eremua ematen digun adierazpenera iristen gara:

$$\sum_{i=0}^k (\alpha_i - \hat{h}\beta_i - \hat{h}^2\gamma_i) r^i = 0, \quad \hat{h} = \lambda h \quad (5.4)$$

(5.4) ekuazioa ekuazio koadratiko bat da, eta $r = e^{i\theta}$ ordezkatu eta $\theta \in [0, 2\pi)$ balioak emanez lortzen den ekuazioaren bi erroek egonkortasun-eremuaren muga deskribatuko dute. (5.2) metodoaren errore-konstantea (Hairer eta Wanner, 1991: 280-281) ondorengo da:

$$C = \frac{C_{p+1}}{\sigma(1)} = \frac{1}{\sigma(1)(p+1)!} \left[\sum_{i=0}^k \alpha_i i^{p+1} - (p+1) \sum_{i=0}^k \beta_i i^p - p(p+1) \sum_{i=0}^k \gamma_i i^{p-1} \right] \quad (5.5)$$

Bigarren deribatua erabiltzen duten zenbait metodo aurkeztuko ditugu jarraian.

5.1.1. Enright-en metodoak

Enright-ek 1974. urtean α_i , β_i , γ_i parametroen aukeraketa oso ona egin zuen. Aukeraketa horretan ondorengo ideiak izan zituen kontuan:

- $\alpha_k = 1, \alpha_{k-1} = -1, \alpha_{k-2} = \dots = \alpha_0 = 0$ balioak ezartzen ditu, jatorriaren inguruan metodoak egonkortasun-ezaugarri onak izan ditzan. Adams-en metodoetako konstanteek ere baldintza hau betetzen dutela eta jatorriaren inguruan egonkortasun-ezaugarri onak dituztela kontuan hartuta ezarri zituen konstante hauen balioak.

- $\gamma_k \neq 0, \gamma_{k-1} = \dots = \gamma_0 = 0$ balioak ezartzen ditu, infinituan egonkortasuna ziurtatzeko. Adibidez, BDFak egonkorak dira infinituan.
- Gainerako $(k+2)$ konstanteak, hau da, $\gamma_k, \beta_k, \beta_{k-1}, \dots, \beta_0$ konstanteak (5.3) ekuazioak erabilia kalkulatu zituen metodoak $(k+2)$ ordena izateko eran. Kontuan izan behar da $q=0$ balioari dagokion baldintza beteko dela aukeraketa hau egiten bada: $\alpha_k = 1, \alpha_{k-1} = -1, \alpha_{k-2} = \dots = \alpha_0 = 0$.

Idea hauen emaitza k pausoko eta $(k+2)$ ordenako zenbakizko metodo bat izan zen, honela emana dagoena:

$$y_{n+k} = y_{n+k-1} + h \sum_{i=0}^k \beta_i f_{n+i} + h^2 \gamma_k g_{n+k} \quad (5.6)$$

Familia honetako zenbait ordenari dagozkion adierazpenak hauek dira:

$$k=1: \quad y_{n+1} = y_n + h \left(\frac{2}{3} f_{n+1} + \frac{1}{3} f_n \right) - \frac{1}{6} h^2 g_{n+1}$$

$$k=2: \quad y_{n+2} = y_{n+1} + h \left(\frac{29}{48} f_{n+2} + \frac{5}{12} f_{n+1} - \frac{1}{48} f_n \right) - \frac{1}{8} h^2 g_{n+2}$$

$$k=3: \quad y_{n+3} = y_{n+2} + h \left(\frac{307}{540} f_{n+3} + \frac{19}{40} f_{n+2} - \frac{1}{20} f_{n+1} + \frac{7}{1.080} f_n \right) - \frac{19}{180} h^2 g_{n+3}$$

$$k=4: \quad y_{n+4} = y_{n+3} + h \left(\frac{3.133}{5.760} f_{n+4} + \frac{47}{90} f_{n+3} - \frac{41}{480} f_{n+2} + \frac{1}{45} f_{n+1} - \frac{17}{5.760} f_n \right) - \frac{3}{32} h^2 g_{n+4}$$

Adibidea

Enright-en metodoetako $k=2$ kasuari dagokion egonkortasun-eremua honela kalkulatu genuke. Test-ekuazioari metodoa aplikatzen diogu:

$$y_{n+2} = y_{n+1} + \hat{h} \left(\frac{29}{48} y_{n+2} + \frac{5}{12} y_{n+1} - \frac{1}{48} y_n \right) - \frac{1}{8} \hat{h}^2 y_{n+2} \quad (5.7)$$

$\hat{h} = h\lambda$ izanik. (5.7) ekuazioan $y_p = r^p$ ordezkatuz eta r^n terminoaz zatituz, hauxe daukagu:

$$r^2 = r + \hat{h} \left(\frac{29}{48} r^2 + \frac{5}{12} r - \frac{1}{48} \right) - \frac{1}{8} \hat{h}^2 r^2 \quad (5.8)$$

Eta $r = 1$ betetzen duten $\hat{h}$ balioen leku geometrikoa izango denez egonkortasun-eremuko muga, (5.8) ekuazioan $r = e^{i\theta}$ ordezkatzeko dugu eta ekuazioa ordenatuko dugu:

$$\hat{h}^2 \frac{1}{8} e^{2i\theta} + \hat{h} \left(-\frac{29}{48} e^{2i\theta} - \frac{5}{12} e^{i\theta} + \frac{1}{48} \right) + (e^{2i\theta} - e^{i\theta}) = 0 \quad (5.9)$$

(5.9) ekuazioa ekuazio koadratikoa da $\hat{h}$ aldagaian.

MATLAB erabilia script gidoi honen bidez irudikatu ahal izango genuke $k = 2$ Enright-en metodoari dagokion egonkortasun-eremua:

```
%Enright-en metodoko k=2 pausori dagokion metodoaren egonkortasun
%eremua irudikatzeko script gidoia.
k=2;
%n puntu erabilia irudikatuko dugu egonkortasun eremua.
n=1000;
h=zeros(n+1,1);p=zeros(n+1,1);
fi=0:2*pi/n:2*pi;

m=1; k=-1; %m=alfa2; k=alfa1;

%Beta konstanteak B matrizean jaso ditugu:
B=[-29/48 -5/12 +1/48];
%g=gamma
g=1/8;

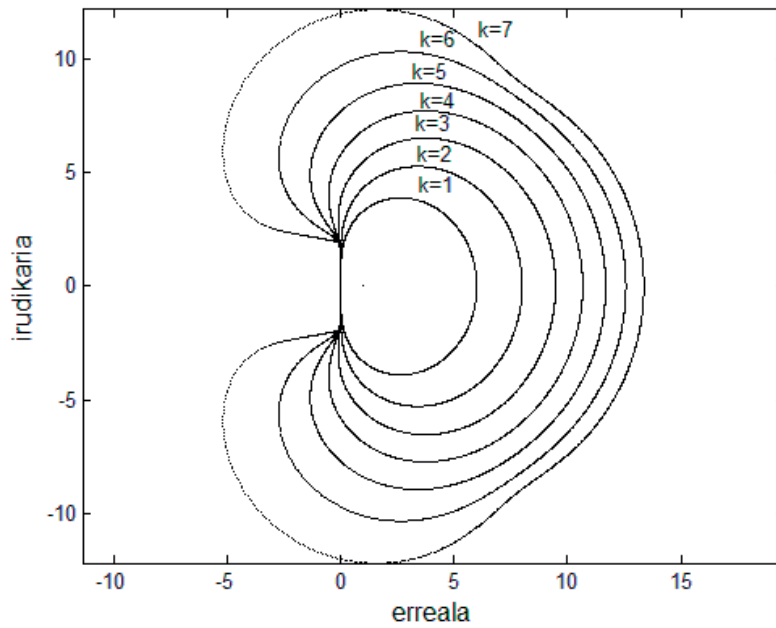
%Egonkortasun-eremuaren muga 2. mailako ekuazio baten erroek
%emana dator: a*h^2+b*h+c=0

for j1=1:n+1
    r(j1)=exp(fi(j1)*i);
    c=m*(r(j1))^2+k*r(j1);
    b=(B(1)*(r(j1))^2+B(2)*r(j1)+B(3));
    a=g*(r(j1))^2;
    disk=sqrt(b^2-4*a*c);
    h(j1,:)=(-b+disk)/(2*a);
    p(j1,:)=(-b-disk)/(2*a);
    plot(h(j1,:), 'k')
    hold on
    plot(p(j1,:), 'k')
end
```


5.1. taulan Enright-en metodoen zenbait ezaugarri jaso dira. Hala nola ordena, $A(\alpha)$ -egonkortasunari dagokion α angelua eta metodoaren errore-konstantea C . 5.1. irudian zenbait Enright-en metodori dagozkien egonkortasun-eremuak irudikatu dira.

k	1	2	3	4	5	6	7
ordena	3	4	5	6	7	8	9
$A(\alpha)$	90°	90°	$87,8^\circ$	$82,03^\circ$	$73,10^\circ$	$59,95^\circ$	$37,61^\circ$
C	0,01389	0,00486	0,00236	0,00136	0,00086	0,00059	0,00042

5.1. taula. Enright-en metodoen zenbait ezaugarri.



5.1. irudia. Enright-en metodoen egonkortasun-eremuak (kurben kanpoko aldeak).

5.1.2. SDBDF metodoak

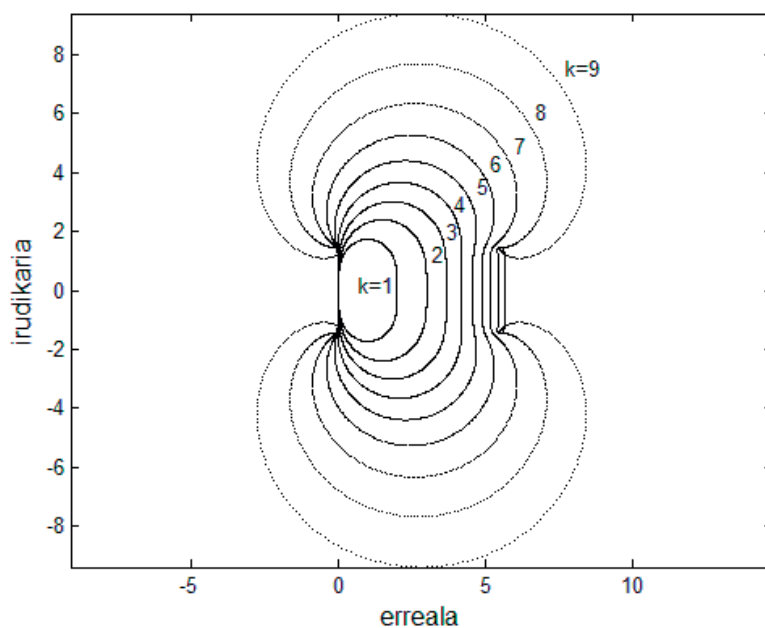
SDBDF metodoek (*Second Derivative BDF method* ingelesez) ere bigarren deribatua darabilte eta BDFak oinarritzat hartuz sortutako metodoak dira. Beraien adierazpena hauxe da:

$$\sum_{j=1}^k \left(\sum_{i=j}^k \frac{1}{i} \right) \frac{\nabla^j y_{n+k}}{j} = \left(\sum_{i=1}^k \frac{1}{i} \right) h f_{n+k} - \frac{h^2}{2} g_{n+k} \quad (5.10)$$

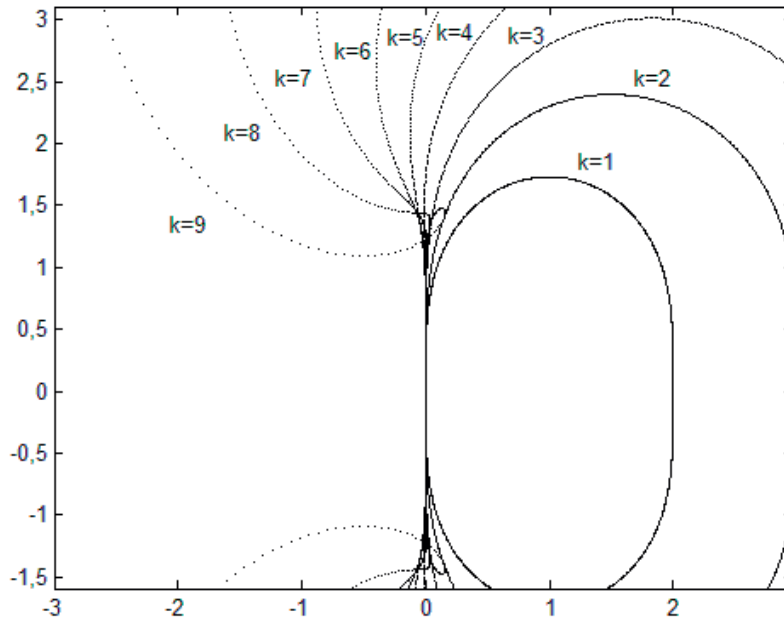
SDBDF metodoak laugarren ordenaraino A-egonkorak dira. 5.2. taulan jaso dira SDBDF metodoei dagozkien $A(\alpha)$ -egonkortasun ezaugarriak, ordena eta metodoaren errore-konstantea C. SDBDF metodoen egonkortasun-eremuak 5.2 eta 5.3. irudietan ikus daitezke.

k	1	2	3	4	5	6	7	8	9	10
ordena	2	3	4	5	6	7	8	9	10	11
$A(\alpha)$	90°	90°	90°	$89,36^\circ$	$86,35^\circ$	$80,82^\circ$	$72,53^\circ$	$60,71^\circ$	$43,39^\circ$	$12,34^\circ$
C	0,1667	0,0556	0,0273	0,0160	0,0104	0,0073	0,0054	0,0041	0,0032	0,0026

5.2. taula. SDBDF metodoei dagozkien zenbait ezaugarri.



5.2. irudia. SDBDF metodoen egonkortasun-eremuak (kurben kanpoko aldeak).



5.3. irudia. SDBDF metodoen egonkortasun-eremuak hurbilagotik (kurben kanpoko aldeak).

5.1.3. Bigarren deribatua darabilen pauso anitzeko metodo eraginkor berri bat

Ismail eta Ibrahim-en (1999) erreferentzian bigarren deribatuak darabiltzan pauso anitzeko metodo linealen familia berezi bat aurkezten da, *New Efficient Second Derivative Multistep Methods* izenekoa. Metodo honi dagokion adierazpena hauxe da:

$$\sum_{i=0}^k \alpha_i y_{n+i} = h\beta_k (f_{n+k} - \beta^* f_{n+k-1}) + h^2 \gamma_k (g_{n+k} - \gamma^* g_{n+k-1}) \quad (5.11)$$

$g(t, y) = y'' = f_t + f_y f$ izanik.

Metodo berri honek SDBDF familiarekiko duen diferentzia da, SDBDF metodoak bakarrik y_{n+k} puntuko bigarren deribatua darabilela eta metodo berri honetan y_{n+k} eta y_{n+k-1} puntuetako bigarren deribatuak kontsideratzen dira. 5.3. eta 5.4. tauletan ikus daitezke β^*, γ^* parametro bikote bati dagozkion $A(\alpha)$ -egonkortasuna eta ordena.

k	1	2	3	4	5	6	7	8	9
ordena	2	3	4	5	6	7	8	9	10
$A(\alpha)$	90°	90°	90°	89°	86°	81,7°	75°	63,5°	47,6°

5.3. taula. $\beta^* = -0,2$ eta $\gamma^* = 0,2$ balioei dagozkien *New Efficient Second Derivative Multistep Method* metodoen egonkortasun-ezaugarriak.

k	1	2	3	4	5	6	7	8	9
ordena	2	3	4	5	6	7	8	9	10
$A(\alpha)$	90°	90°	90°	89,9°	87,3°	84,2°	80°	71°	57,8°

5.4. taula. $\beta^* = -0,05$ eta $\gamma^* = 0,9$ balioei dagozkien *New Efficient Second Derivative Multistep Method* metodoen egonkortasun-ezaugarriak.

5.2. PAUSO ANITZEKO ZENBAKIZKO METODO HEDATUAK

5.2.1. EBDF metodoak

Egonkortasun-ezaugarri hobeak lortzeko asmoarekin, Cash-ek BDF metodoen hedapen bat proposatu zuen, zeinean super-etorkizuneko puntuak erabiltzea proposatzen baitzuen (Cash, 1980). Metodo berri honi EBDF deitu zitzaion (ingelesezko *Extended Backward Differentiation Formula*). Beraien adierazpena honela emana dago:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \beta_k f_{n+k} + h \beta_{k+1} \bar{f}_{n+k+1} \quad (5.12)$$

$\bar{f}_{n+k+1}$ terminoa t_{n+k+1} aldiunean BDFak erabilita kalkulatzen den puntuari dagokion deribatua da jarraian ikusiko dugun bezalaxe. (5.12) adierazpeneko metodoa $(k+1)$ ordenakoa izan dadin α_i, β_i koefizienteek ondorengo ekuazio-sistema bete behar dute:

$$\sum_{j=0}^k \alpha_j j^q = q \sum_{j=0}^k \beta_j j^{q-1}, \quad q = 0, 1, \dots, k+1 \quad (5.13)$$

$y_n, y_{n+1}, \dots, y_{n+k-1}$ balioak baditugula kontuan izanda, EBDF metodoa ondorengo prozesua aplikatzean datza:

1. BDFak erabilita, lehenengo iragarpena egiten da: $\bar{y}_{n+k}$ kalkulatu da k ordenako BDFaren emaitza bezala:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = h f_{n+k} \quad (y_{n+k} := \bar{y}_{n+k}) \quad (5.14)$$

2. BDFak erabilita, bigarren iragarpena egiten da: BDFak erabilita beste pauso bat ematen da, $\bar{y}_{n+k+1}$ kalkulatu:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j+1} = h f_{n+k+1} \quad (y_{n+k+1} := \bar{y}_{n+k+1}) \quad (5.15)$$

3. $\bar{f}_{n+k+1} = f(t_{n+k+1}, \bar{y}_{n+k+1})$ balioa kalkulatu da.
4. Zuzentzailea aplikatu da: $\bar{f}_{n+k+1}$ balioa (5.12) ekuazioan ordezkatu eta y_{n+k} kalkulatu da. Eta y_{n+k} izango da EBDF metodoaz lortzen den balioa. (5.12) adierazpenak prozesu honetan zuzentzaile gisa jokatzen duela esan daiteke.

Lemma: (5.12) formulak ematen digun zenbakizko balioa $(k+1)$ ordenakoa bada eta (5.14) eta (5.15) formuletan erabilitako metodoak k ordenakoak badira, aurreko (1)-(4) pausoek osatzen duten EBDF metodoaren ordena $(k+1)$ da. Lemma honen frogapen bat aurki daiteke Hairer eta Wanner-en (1991: 288-289) erreferentzian.

EBDF metodoen mozketak-errore lokala hau da:

$$LTE = h^{k+2} \left[C_3 y^{(k+2)} + \beta_{k+1} C_1 \left(1 - \frac{\hat{\alpha}_{k-1}}{\hat{\alpha}_k} \right) \frac{\partial f}{\partial y} \cdot y^{(k+1)} \right] (t_n) + O(h^{k+3}) \quad (5.16)$$

C_1 koefizientea BDF metodoei dagokien errore-konstantea izanik eta C_3 koefizientea EBDF metodoko (5.12) zuzentzaileari dagokiona.

Egonkortasun-azterketa egiteko metodo osoa test-ekuazioari aplikatu zaio eta diferentzian emandako ekuazio lineala lortzen da. Bertan $y_p = r^p$ ordezkatur eta r^n terminoaz zatituz, hau lortzen dugu:

$$A\hat{h}^3 + B\hat{h}^2 + C\hat{h} + D = 0 \quad (5.17)$$

$$\text{Non: } \begin{cases} A = \beta_k r^k \\ B = -2\hat{\alpha}_k \beta_k r^k - T + \beta_{k+1} S \\ C = \beta_k \hat{\alpha}_k^2 r^k + 2\hat{\alpha}_k T - \hat{\alpha}_k \beta_{k+1} S + \beta_{k+1} \hat{\alpha}_{k-1} R \\ D = -\hat{\alpha}_k^2 T \\ R = \sum_{j=0}^{k-1} \hat{\alpha}_j r^j, S = \sum_{j=0}^{k-2} \hat{\alpha}_j r^{j+1}, T = \sum_{j=0}^k \alpha_j r^j, \end{cases} \quad (5.18)$$

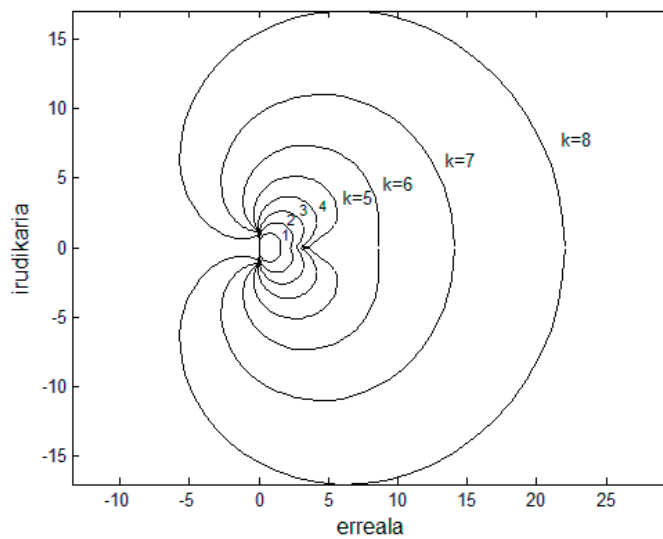
(5.18) ekuazio-multzoan $r = e^{i\theta}$ ordezkatzuz gero, metodoaren egonkortasun-eremuak lortzen dira. EBDF metodoari dagozkion egonkortasun-ezaugarriak 5.5. taulan jaso dira eta 5.4. eta 5.5. irudietan daude irudikatuta egonkortasun-eremuok.

k	1	2	3	4	5	6	7	8
ordena	2	3	4	5	6	7	8	9
$A(\alpha)$	90°	90°	90°	87,61°	80,21°	67,73°	48,82°	19,98°

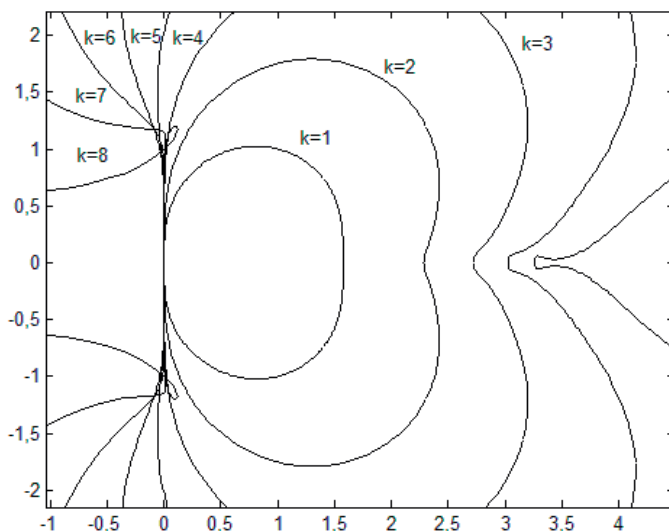
5.5. taula. EBDF metodoen zenbait ezaugarri.

**Oharra: EBDF metodoko konstanteak izendatzeko α_j, β_j izenak erabili dira eta BDF metodoko konstanteak izendatzeko $\hat{\alpha}_j$. BDFak era honetan kontsideratu

dira: $\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = h f_{n+k}$.



5.4. irudia. EBDF metodoen egonkortasun-eremuak (kurben kanpoko aldea).



5.5. irudia. EBDP metodoen egonkortasun-eremuak hurbilagotik (kurben kanpoko aldeak).

5.2.2. MEBDF metodoak

Cash-en (1983) lanean MEBDF metodoak aurkezten dira. EBDP-en eskema berari jarraitzen dioten metodoak dira, baina zuzentzailea aldatzen da berauetan. Kasu honetan, MEBDF metodoetako zuzentzailea honela dator emana:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \hat{\beta}_k f_{n+k} + h(\beta_k - \hat{\beta}_k) \bar{f}_{n+k} + h \beta_{k+1} \bar{f}_{n+k+1} \quad (5.19)$$

β_k koefizienteak EBDP metodokoak izanik eta $\hat{\beta}_k$ koefizienteak BDF metodokoak.

MEBDF metodoak ere, EBDP-en antzera, $(k+1)$ ordenakoak dira eta berauen mozketa-errore lokala hauxe da:

$$LTE = h^{k+2} \left[C_4 y^{(k+2)} + C_1 \left(\beta_{k+1} \left(1 - \frac{\hat{\alpha}_{k-1}}{\hat{\alpha}_k} \right) + (\beta_k - \hat{\beta}_k) \right) \frac{\partial f}{\partial y} \cdot y^{(k+1)} \right] (t_n) + O(h^{k+3}) \quad (5.20)$$

C_1 koefizientea BDF metodoei dagokien errore-konstantea izanik eta C_4 koefizientea MEBDF metodoko zuzentzaileari dagokiona.

k	1	2	3	4	5	6	7	8
MEBDFen $A(\alpha)$	90°	90°	90°	88,4°	83,1°	74,5°	62°	43°

5.6. taula. MEBDF metodoen egonkortasun-ezaugarriak.

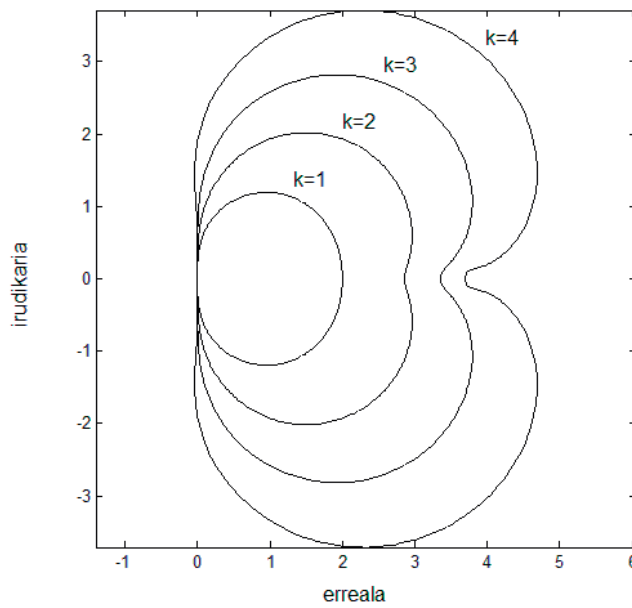
Test-ekuazioari MEBDF metodoa aplikatuta, 3. mailako polinomio batera iritsiko gara:

$$A\hat{h}^3 + B\hat{h}^2 + C\hat{h} + D = 0 \quad (5.21)$$

(5.21) ekuazioaren koefizienteak hauek izanik:

$$\begin{cases} A = \hat{\beta}_k r^k \\ B = -2\hat{\alpha}_k \hat{\beta}_k r^k - T + \beta_{k+1} S + (\beta_k - \hat{\beta}_k) R \\ C = \hat{\beta}_k \hat{\alpha}_k^2 r^{k+1} + 2\hat{\alpha}_k T - \hat{\alpha}_k \beta_{k+1} S + \beta_{k+1} \hat{\alpha}_{k-1} R - (\beta_k - \hat{\beta}_k) \hat{\alpha}_k R \\ D = -\hat{\alpha}_k^2 T \\ R = \sum_{j=0}^{k-1} \hat{\alpha}_j r^j, S = \sum_{j=0}^{k-2} \hat{\alpha}_j r^{j+1}, T = \sum_{j=0}^k \alpha_j r^j \end{cases} \quad (5.22)$$

5.6. taulan jaso dira MEBDF metodoei dagozkien $A(\alpha)$ -egonkortasunari dagozkion angeluak, eta angelu hori pauso kopuru bera duen EBDF metodoarena baino handiagoa da. 5.6. irudian MEBDFen zenbait egonkortasun-eremu irudikatu dira.



5.6. irudia. MEBDF metodoen egonkortasun-eremuak (kurben kanpoko aldeak).

5.3. PAUSO ANITZEKO METODO KONBINATUAK

Pauso anitzeko zenbakizko metodo konbinatuak bi ideia hauetan oinarrituta eraiki ziren:

- Alde batetik, badakigu Adams Moulton-en metodoak onak direla ekuazio diferentzial ez-zurrunak ebazteko:

$$-y_{n+k} + y_{n+k-1} + h \sum_{j=1}^k \beta_j^* f_{n+k+1-j} = 0 \quad (AMF^{(k)}) \quad (5.23)$$

- Badakigu, baita ere, BDF metodoak onak direla ekuazio diferentzial zurrunak askatzeko.

$$-\sum_{j=0}^k \hat{\alpha}_j y_{n+j} + h f_{n+k} = 0 \quad (BDF^{(k)}) \quad (5.24)$$

Problema ez-zurrunen ezaugarria da $-h \frac{\partial f}{\partial y}$ txikia dela, eta problema zurrunetan,

aldiz, $-h \frac{\partial f}{\partial y}$ balioa handia da. Konbinatutako pauso anitzeko zenbakizko metodoen helburua bi metodo horien arteko konbinazio bat egitea izan zen, bakoitzari pisu ezberdinak emanez, hau da:

$$\{AMF^{(k+1)} - \gamma^{(k)} \cdot h \cdot J \cdot BDF^{(k)}\} = 0 \quad (5.25)$$

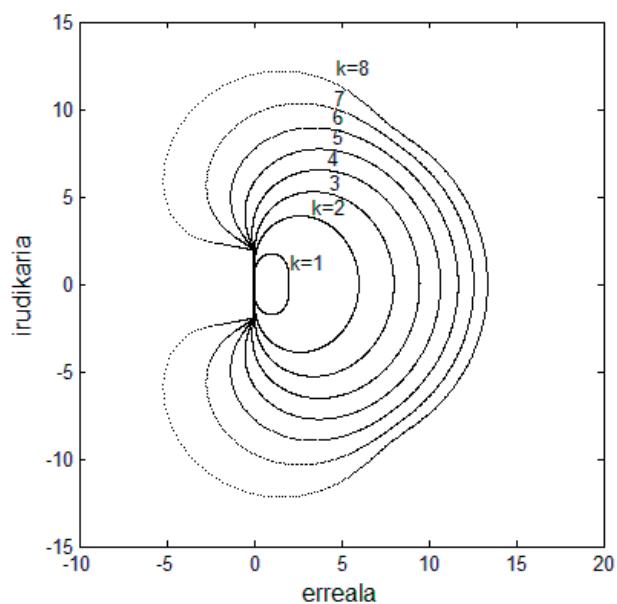
γ^k parametro aske bat izanik eta $J = \partial f / \partial y$ jacobitarra.

Era horretan lortzen den (5.25) konbinazioa $(k+1)$ ordenakoa da edozein $\gamma^{(k)}$ baliotarako. $-hJ$ faktorea handia izan edo txikia izan, pisua (5.25) formulako batugai egokian jartzean dago gakoa. 5.7. taulan metodo honi dagozkion egonkortasun-ezaugarriak jaso dira $\gamma^{(k)}$ parametroaren bi aukeraketa ezberdinetarako (Hairer eta Wanner, 1991: 287), Lehenengoa $\gamma^{(k)} = -k\gamma_k^*$ aukeraketari dagokio, γ_k^* Adams Moulton-en metodoko konstanteak izanik. Bigarren kasuan aukeratu den $\gamma^{(k)}$ parametroak ahalbidetzen du metodoak $A(\alpha)$ -egonkortasun angelu maximoa izatea. 5.7. taulan p letraz adierazi dugu metodoaren ordena. $A(\alpha)$ -egonkortasunari dagokion zutabeen zenbakirik ez dugu jarri, zeren pauso kopuru hori duen metodoa $A(0)$ -egonkorra ez baita. Hori gertatzen da adibidez, $k=9$, $k=10$ eta $k=11$

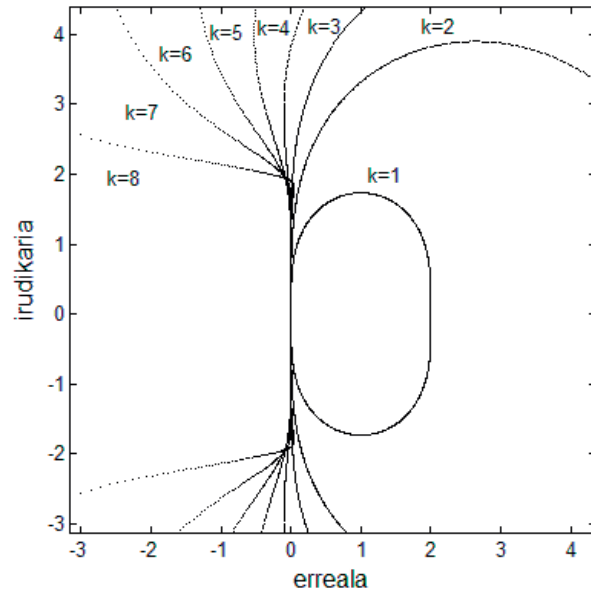
pauso kopurua duten metodoetan $\gamma^{(k)} = -k\gamma_k^*$ aukeraketa egin dugunean. 5.7. eta 5.8. irudietan, $\gamma^{(k)} = -k\gamma_k^*$ kasuari dagozkion egonkortasun-eremuak irudikatu dira.

k	p	$\gamma^{(k)} = -k\gamma_k^*$	$\gamma^{(k)} = -k\gamma_k^*$ deneko $A(\alpha)$	$\gamma^{(k)} = \gamma_{opt}^{(k)}$ deneko $A(\alpha)$
1	2	0.5	90°	$[0, +\infty)$ 90°
2	3	0,166667	90°	$[0.125, +\infty)$ 90°
3	4	0,125	90°	$[0.12189, 0.68379)$ 90°
4	5	0,1055556	87,88°	0,1284997 89,42°
5	6	0,09375	82,03°	0,1087264 86,97°
6	7	0,8561508	73,10°	0,0962596 82,94°
7	8	0,07957176	59,95°	0,08754864 77,43°
8	9	0,07485229	37,61°	0,08105624 70,22°
9	10	0,07103299	---	0,07599875 60,68°
10	11	0,06785850	---	0,07192937 47,63°
11	12	0,06516462	---	0,06857226 28,68°

5.7. taula. Metodo konbinatuen egonkortasun-ezaugarriak $\gamma^{(k)}$ parametroaren zenbait baliotarako.



5.7. irudia. Metodo konbinatuei dagozkien egonkortasun-eremuak (kurben kanpoko aldeak), $\gamma^{(k)} = -k\gamma_k^*$ izanik.



5.8. irudia. Metodo konbinatuen egonkortasun-eremuak hurbilagotik.

5.4. PAUSO ANITZEKO KOKAPENEN METODOAK

$c_1, c_2, \dots, c_s$ s zenbaki erreal izanik (normalean $(0,1)$ tartean kokatuta egoten direnak) eta $y_n, y_{n+1}, \dots, y_{n+k-1}$ aurreko k pausoetan lortutako ekuazio diferentzialaren soluzioak eskura izanik, puntu horietatik igarotzen den $(s+k-1)$ ordenako polinomioa, $u(t)$ deituko duguna, era honetan emana dator:

$$\begin{aligned} u(t_j) &= y_j, \quad j = n, \dots, n+k-1 \\ u'(t_n + c_i h) &= f(t_n + c_i h, u(t_n + c_i h)), \quad i = 1, 2, \dots, s \end{aligned} \quad (5.26)$$

Era horretan, t_{n+k} aldiuneko zenbakizko soluzioa $y_{n+k} = u(t_{n+k})$ da. $u(t)$ polinomioa idatzi ahal izateko, tt deituko dugun koordenatu adimentsionala definituko dugu.

$$tt = \frac{t - t_{n+k-1}}{h} \quad (5.27)$$

$tt_1 = -k+1, \dots, tt_{k-1} = -1, tt_k = 0$ nodoak kontsideratuko ditugu eta $i = 1, 2, \dots, k$ balioetarako $(s+k-1)$ ordenako $\varphi_i(tt)$ polinomioak definituko ditugu:

$$\begin{cases} \varphi_i(tt_j) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}, & j = 1, \dots, k \\ \varphi'_i(c_j) = 0, & j = 1, \dots, s \end{cases} \quad (5.28)$$

Eta $i = 1, \dots, s$ balioetarako $\psi_i(tt)$ polinomioak definituko ditugu:

$$\begin{cases} \psi_i(tt_j) = 0, & j = 1, \dots, k \\ \psi'_i(c_j) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}, & j = 1, \dots, s \end{cases} \quad (5.29)$$

(5.28) eta (5.29) ekuazioak erabilita, $u(t)$ polinomioa era honetan idatz daiteke:

$$u(t_{n+k-1} + tt \cdot h) = \sum_{j=1}^k \varphi_j(t) y_{n+j-1} + h \sum_{j=1}^s \psi_j(t) u'(t_{n+k-1} + c_j h) \quad (5.30)$$

Eta (5.30) ekuazioan $tt = c_i$ ordezkatzuz, $u(t_{n+k-1} + c_i h) = v_i$ idatziz eta kokapeneko baldintza (5.26) kontuan hartuz, ondorengo berdintzak lortzen dira:

$$\begin{cases} v_i = \sum_{j=1}^k a_{ij} y_{n+j-1} + h \sum_{j=1}^s b_{ij} f(t_{n+k-1} + c_j h, v_j), & i = 1, 2, \dots, s \\ y_{n+k} = \sum_{j=1}^k a_{k+1,j} y_{n+j-1} + h \sum_{j=1}^s b_{k+1,j} f(t_{n+k-1} + c_j h, v_j), & i = 1, 2, \dots, s \end{cases} \quad (5.31)$$

Non: $a_{ij} = \varphi_j(c_i)$, $b_{ij} = \psi_j(c_i)$, $a_{k+1,j} = \varphi_j(1)$ eta $b_{k+1,j} = \psi_j(1)$.

Era horretan lortzen diren metodoek pauso anitzeko kokapeneko metodo izena dute (*Multistep Collocation Method* ingelesez). Metodo hauek egonkortasun-ezaugarri onak dituzte. c_i konstanteen aukeraketa bat $c_s = 1$ eta gainerako koefizienteak metodoaren ordena $p = 2s + k - 2$ izateko eran aukeratzen direna da. Metodo konkretu hauek, Radau-ren metodo bezala ezagutzen direnak dira eta A-egonkorak dira seigarren ordenara arte. 5.8. taulan, $s = 3$ balioari dagozkion Radau-ren metodoen egonkortasun-ezaugarriak ikus daitezke (Hairer eta Wanner, 1991: 297), p metodoaren ordena izanik eta α angelua $A(\alpha)$ -egonkortasuneko angelua.

k	P	c_1	c_2	c_3	$A(\alpha)$
1	5	0,155051025721682	0,644948974278318	1	90°
2	6	0,177891722985607	0,673235257220651	1	90°
3	7	0,192169638937766	0,689317969824851	1	89,73°
4	8	0,202814874040288	0,700407719104611	1	89,13°
5	9	0,211395456069620	0,708798418188500	1	88,61°
6	10	0,218626151232186	0,715507419158199	1	88,14°
7	11	0,224897548200883	0,721072684914921	1	87,70°
8	12	0,230448266933707	0,725812172023161	1	87,28°
9	13	0,235435607740434	0,729928926504599	1	86,89°
10	14	0,239969169367303	0,733560240031675	1	86,51°
11	15	0,244128606044551	0,736803122952198	1	86,14°
12	16	0,247973766491964	0,739728565298052	1	85,79°
13	17	0,251550844436705	0,742390019356757	1	85,44°

5.8. taula. Radau $s=3$ metodoari dagozkion koefizienteak eta zenbait parametro.**Adibidea**

$s=3$ eta $k=1$ kasua kontsideratuko dugu. Kasu horretan φ_1 polinomio bakarra daukagu eta ψ_i motako hiru polinomio ditugu, $tt_1=0$ izanik. Polinomioak 3. mailakoak dira kasu honetan, $s+k-1=3$, eta ondorengo baldintzak bete behar dituzte:

$$\begin{cases} \varphi_1(tt_1)=1, \\ \varphi_1'(c_j)=0, \quad j=1,2,3 \end{cases} \quad \text{eta} \quad \begin{cases} \psi_i(tt_1)=0, \quad i=1,2,3 \\ \psi_i'(c_j)=\begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}, \quad i,j=1,2,3 \end{cases} \quad (5.32)$$

Hau da, $\varphi_1(t) = At^3 + Bt^2 + Ct + D$ eta $\psi_i(t) = M_it^3 + N_it^2 + P_it + R_i$ $i=1,2,3$ balioetarako. Polinomio horietako bakoitzeko 4 koefizienteak kalkulatu ahal izateko, (5.32) adierazpeneko baldintzak erabiltzen dira. Behin polinomio horietako koefizienteak kalkulatu ditugunean, beste konstante hauek kalkulatzeko dira: $a_{ij} = \varphi_j(c_i)$, $b_{ij} = \psi_j(c_i)$, $a_{k+1,j} = \varphi_j(1)$ eta $b_{k+1,j} = \psi_j(1)$. Gure kasuan, $k=1$ denez, hauek dira konstanteak: $a_{i1} = \varphi_1(c_i)$, $b_{ij} = \psi_j(c_i)$, $a_{2,1} = \varphi_1(1)$ eta $b_{2,j} = \psi_j(1)$, $i,j=1,2,3$ balioetarako.

Eta koefiziente guztiak ditugunean eta $s=3$, $k=1$ kasuan gaudela kontuan hartuz, koefizienteak (5.31) ekuazioetan ordezkatzeko ditugu eta zera geratzen zaigu:

$$\begin{cases} v_i = a_{i1}y_n + h \sum_{j=1}^3 b_{ij}f(t_n + c_j h, v_j), & i=1,2,3 \\ y_{n+1} = a_{2,1}y_n + h \sum_{j=1}^3 b_{2,j}f(t_n + c_j h, v_j), & i=1,2,3 \end{cases} \quad (5.33)$$

Eta (5.33) adierazpeneko bi ekuazioak era matritzialean honela idatz daitezke:

$$\begin{cases} V = Ay_n + hBF \\ y_{n+1} = \tilde{A}y_n + h\tilde{B}F \end{cases} \quad (5.34)$$

Non: $V = (v_1, v_2, v_3)^T$, $A = (a_{11}, a_{21}, a_{31})^T$, $B = (b_{ij})$, $F = (f(t_n + c_j h, v_j))^T$, $\tilde{B} = (b_{21}, b_{22}, b_{23})$, $\tilde{A} = a_{2,1}$ izanik eta beraien dimentsioak $V, A, F \in M_{3 \times 1}$, $B \in M_{3 \times 3}$, $\tilde{B} \in M_{1 \times 3}$ eta $\tilde{A} \in M_{1 \times 1}$ izanik.

Orokorrean, edozein s eta k -ren balioetarako, (5.34) adierazpenak forma hau hartzen du:

$$\begin{cases} V = AY_{n+k-1} + hBF \\ y_{n+k} = \tilde{A}Y_{n+k-1} + h\tilde{B}F \end{cases} \quad (5.35)$$

Non: $Y = (y_n, y_{n+1}, \dots, y_{n+k-1})^T \in M_{k \times 1}$ izanik eta $V \in M_{s \times 1}$,

$F = (f(t_{n+k-1} + c_j h, v_j)) \in M_{s \times 1}$, $B = (b_{ij}) \in M_{s \times s}$, $A = (a_{ij}) \in M_{s \times k}$,

$\tilde{A} = (a_{k+1,j}) \in M_{1 \times k}$, $\tilde{B} = (b_{k+1,j}) \in M_{1 \times s}$

Metodoaren egonkortasun azterketa egin ahal izateko (5.34) adierazpen matritzialekin egingo dugu lana. Metodoa test ekuazioari aplikatzen diogu eta F matrizea era honetan geratzen zaigu: $F = \lambda V$. Adierazpen hau (5.34) adierazpenetako bi ekuazioetan ordezkatzuz zera daukagu:

$$\begin{cases} V = Ay_n + \hat{h}BV \\ y_{n+1} = \tilde{A}y_n + \hat{h}\tilde{B}V \end{cases} \quad (5.36)$$

Beti bezala $\hat{h} = h\lambda$ izanik.

(5.36) adierazpeneko lehenengotik V -ren balioa aterako dugu:

$$V = Ay_n + \hat{h}BV \Rightarrow V = (I_3 - \hat{h}B)^{-1} Ay_n \quad (5.37)$$

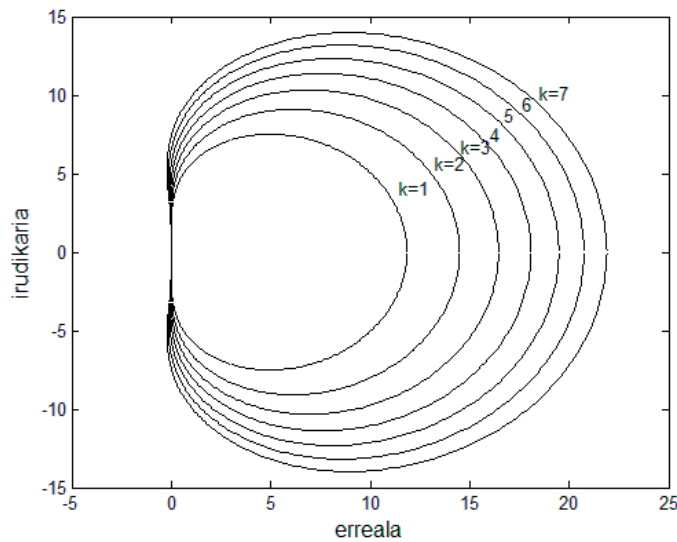
(5.37) adierazpeneko V -ren balioa (5.36) adierazpeneko bigarrenean ordezkatzuz, zera daukagu:

$$y_{n+1} = \tilde{A}y_n + \hat{h}\tilde{B}(I_3 - \hat{h}B)^{-1} Ay_n \Rightarrow y_{n+1} = \left(\tilde{A} + \hat{h}\tilde{B}(I_3 - \hat{h}B)^{-1} A \right) y_n \quad (5.38)$$

$s=3$, $k=1$ kasuari dagokion egonkortasun-funtzioa hauxe da, beraz:

$$R(\hat{h}) = \left(\tilde{A} + \hat{h}\tilde{B}(I_3 - h\lambda B)^{-1} A \right) \quad (5.39)$$

Edozein s eta k -ren baliori dagokion egonkortasun-funtzioa ere era bertsuan lortzen da. 5.9. irudian ikus daitezke $s=3$ denean ditugun kokapeneko zenbait metodoren egonkortasun-eremuak.



5.9. irudia. Radau $s=3$ metodoari dagozkion egonkortasun-eremuak (kurben kanpoko aldeak).

5.5. NDF METODOAK

BDFei egindako beste eraldaketa bat *Numerical Differentiation Formulae* izeneko metodoak dira. Konputazionalki garestia ez den aldaketa da eta $(k+1)$ ordenako atzerakako diferentzia k ordenako metodoan barneratzean datza. k ordena duen NDFaren adierazpena hau da:

$$\sum_{j=1}^k \frac{1}{j} \nabla^j y_{n+k} = h f_{n+k} + \kappa \gamma_k \nabla^{k+1} y_{n+k} \quad (5.40)$$

γ_k konstanteak BDF metodoen atalean ikusitakoak izanik, hau da:

$$\gamma_k = \hat{\alpha}_k = \sum_{j=1}^k \frac{1}{j} = \begin{cases} 1, & k=1 \\ 3/2, & k=2 \\ 11/6, & k=3 \\ 25/12, & k=4 \\ 137/60, & k=5 \end{cases}$$

κ koefizienteak Klopfenstein eta Shampine-k proposatu zituzten, $A(\alpha)$ -egonkortasunari dagokion angelu maximoa lortu eta mozketa-errore lokal txikiagoa lortzeko eran. 5.9. taulan jaso dira κ koefizienteak. Shampine-k $(k+1)$ ordenako atzerakako diferentzia hau bakarrik BDF metodoko $k=1,2,3,4$ ordenetan gehitzea proposatzen du, bosgarren ordenan ez baita oso eraginkorra. Esan beharra dago, NDF metodoak BDFak baino zehaztasun handiagokoak direla, hau da, ordena berean mozketa-errore lokal txikiagoa daukate. Beraien mozketa-errore lokala honela emana dator:

$$LTE = \left(-\frac{1}{k+1} - \kappa \gamma_k \right) h^{k+1} y^{(k+1)}(t_n) + O(h^{k+2}) \quad (5.41)$$

NDFetan sartzen den $(k+1)$ ordenako atzerakako diferentziak errorean positiboki eragiten badu ere, NDFen egonkortasun-eremua BDFena baino txikiagoa da. NDF metodoei dagozkien zenbait ezaugarri 5.9. taulan jaso dira, hala nola beraien $A(\alpha)$ -egonkortasuneko angelua. NDFen mozketa-errore lokala txikiagoa izatearen ondorioz, BDFetan zein NDFetan errore bera lortzeko NDF metodoek BDFek darabilten pauso-tamaina baino % 26 handiagoa erabili ahal izango lukete $k=1,2,3$ ordenetan eta % 12 handiagoa $k=4$ ordenan. NDFak, beraz, BDF metodoak baino efizienteagoak dira aipatu ditugun ordenetan.

k	κ	% pauso- tamaina	BDFen $A(\alpha)$	NDFen $A(\alpha)$
1	-0,1850	% 26	90°	90°
2	-1/9	% 26	90°	90°
3	-0,0823	% 26	86°	80°
4	-0,0415	% 12	73°	66°

5.9. taula. NDF eta BDF metodoei dagozkien zenbait ezaugarri.

5.6. A-BDF METODOAK

Fredebeul-ek BDF inplizituaren eta esplizituaren konbinazio lineal bat erabiltzen duen metodo berri bat proposatu zuen (Fredebeul, 1998). Metodoak a parametro aske bat erabiltzen du eta konbinazio lineala honela emana dator:

$$BDF^{(i)} - a \cdot BDF^{(e)} = 0 \quad (5.42)$$

(e) eta (i) goi-indizeak erabili dira, hurrenez hurren esplizitua eta inplizitua adierazteko. BDF esplizituaren eta inplizituaren adierazpenak hauek badira:

$$\blacksquare \quad BDF^{(e)}: \sum_{j=0}^k \bar{\alpha}_j y_{n+j} = h \bar{\beta}_k f_{n+k-1} \quad (5.43)$$

$$\blacksquare \quad BDF^{(i)}: \sum_{j=0}^k \alpha_j y_{n+j} = h \beta_k f_{n+k} \quad (5.44)$$

(5.42) adierazpena garatuz gero, A-BDF metodoen adierazpena iristen gara:

$$\sum_{j=0}^k (\alpha_j - a \bar{\alpha}_j) y_{n+j} = h \beta_k f_{n+k} - h a \bar{\beta}_k f_{n+k-1} \quad (5.45)$$

α_j, β_j konstanteak BDF inplizituari dagozkionak izanik eta $\bar{\alpha}_j, \bar{\beta}_j$ konstanteak BDF esplizituari dagozkionak. BDF esplizituko konstanteak ondorioztatzeko, k pausoko BDF esplizitua k ordenakoa izan dadin bete beharreko baldintza erabiliko dugu:

$$\sum_{i=0}^k \bar{\alpha}_i i^q = q \sum_{i=0}^k \bar{\beta}_i i^{q-1}, \quad 0 \leq q \leq k \quad (5.46)$$

(5.46) ordena-baldintza erabilita BDF^(e) metodoari dagozkion konstanteak kalkula daitezke, 5.10. taulan jaso ditugunak.

k	$\bar{\alpha}_5$	$\bar{\alpha}_4$	$\bar{\alpha}_3$	$\bar{\alpha}_2$	$\bar{\alpha}_1$	$\bar{\alpha}_0$	$\bar{\beta}_k$
1					1	-1	1
2				1	0	-1	2
3			1	$\frac{3}{2}$	-3	$\frac{1}{2}$	3
4		1	$\frac{10}{3}$	-6	2	$-\frac{1}{3}$	4
5	1	$\frac{65}{12}$	-10	5	$-\frac{5}{3}$	$\frac{1}{4}$	5

5.10. taula. BDF esplizituko koefizienteak.

A-BDF metodoen egonkortasun-azterketarako, test-ekuazioari metodoa aplikatzen zaio eta edozein k ordenatarako egonkortasun-funtziora iritsiko gara:

$$\hat{h} = \frac{\sum_{j=0}^k (\alpha_j - a\bar{\alpha}_j) r^j}{\beta_k r^k - a\bar{\beta}_k r^{k-1}} \quad (5.47)$$

5.11. taulan jaso dira A-BDF metodoen $A(0)$ -egonkortasun ezaugarriak. Horrela, $k=1$ kasurako $a \in \mathbb{R} - \{1\}$ balioetarako A-BDF metodoak $A(0)$ -egonkorak dira.

k	1	2	3	4 eta 5	6
a	$\mathbb{R} - \{1\}$	$(-\infty, 1)$	$(-5, 1)$	$(-4, 1)$	$\left(-\frac{13}{3}, 1\right)$

5.11. taula. A-BDF metodoari dagozkion $A(0)$ -egonkortasun ezaugarriak.

5.7. BIGARREN DERIBATUA ETA SUPER-ETORKIZUNEN PUNTUAK DARABILTZATEN ZENBAIT METODO

5.7.1. Second Derivative Extended Methods (E2BD) familia

Cash-en (1981) erreferentzian pauso anitzeko metodo hedatuen familia bat aurkezten da bigarren deribatua eta super-etorkizuneko puntuak darabiltzana: *Second Derivative Extended Methods* (E2BD) izeneko. Metodoa ondorengo pausoak aplikatzean datza:

1. Lehenengo iragarpena egiten da: $\bar{y}_{n+k}$ kalkulatzeko da k pausoko eta $(k+2)$ ordenako ondorengo metodoa erabilia:

$$y_{n+k} - h\beta_k f_{n+k} - h^2 \gamma_k g_{n+k} = \sum_{j=0}^{k-1} (-\alpha_j y_{n+j} + h\beta_j f_{n+j} + h^2 \gamma_j g_{n+j}), \quad y_{n+k} := \bar{y}_{n+k} \quad (5.48)$$

(5.48) adierazpeneko koefizienteak $(k+2)$ ordena lortzeko eran kalkulatzeko dira.

2. Bigarren iragarpena egiten da: $\bar{y}_{n+k+1}$ kalkulatzeko da:

$$y_{n+k+1} - h\beta_k f_{n+k+1} - h^2 \gamma_k g_{n+k+1} = -\alpha_{k-1} \bar{y}_{n+k} + h\beta_{k-1} f(\bar{y}_{n+k}) + h^2 \gamma_{k-1} g(\bar{y}_{n+k}) \\ + \sum_{j=0}^{k-2} (-\alpha_j y_{n+j+1} + h\beta_j f_{n+j+1} + h^2 \gamma_j g_{n+j+1}), \quad y_{n+k+1} := \bar{y}_{n+k+1}$$

3. Deribatuak kalkulatzeko dira: $\bar{f}_{n+k+1} = f(\bar{y}_{n+k+1})$, $\bar{g}_{n+k+1} = g(\bar{y}_{n+k+1})$

4. Zuzentzailea aplikatzeko da: y_{n+k} kalkulatzeko da ondorengo formula zuzentzailea erabilia:

$$y_{n+k} - h\bar{\beta}_k f_{n+k} - h^2 \bar{\gamma}_k g_{n+k} = \sum_{j=0}^{k-1} (-\bar{\alpha}_j y_{n+j} + h\bar{\beta}_j f_{n+j} + h^2 \bar{\gamma}_j g_{n+j}) + h\bar{\beta}_{k+1} \bar{f}_{n+k+1} + h^2 \bar{\gamma}_{k+1} \bar{g}_{n+k+1} \quad (5.49)$$

Cash-en (1981) artikuluan, bi metodo aurkezten dira. Batak aurkeztu dugun formula zuzentzailea (5.49) darabil, eta besteak, aldiz, ez du kontsideratzen super-etorkizuneko puntuaren bigarren deribatua formula zuzentzailean. Hau da, ez du kontsideratzen (5.49) adierazpeneko azken batugaia. 1 klasea deitu diogu formula zuzentzailean super-etorkizuneko puntuaren bigarren deribatua darabilen metodoari, eta 2 klasea ez darabilenari. 5.12. taulan jaso dira bi klase horien egonkortasun-ezaugarriak eta 1 klaseak, formula zuzentzailean super-etorkizuneko bigarren deribatua darabilenak, ezaugarri hobeak dituela ikus daiteke.

k	1	2	3	4	5	6
ordena	4	5	6	7	8	9
$A(\alpha)$ 2 klasea	90°	90°	90°	89°	87°	83°
$A(\alpha)$ 1 klasea	90°	90°	90°	90°	90°	89°

5.12. taula. E2BD metodoen egonkortasun-ezaugarriak.

5.7.2. New SDMM metodoak

Hojjati, Rahimi Ardabili eta Hosseini-ren (2006) lanean New SDMM (*New Second Derivative Multistep Methods*) izeneko familia aurkezten da. Metodo honek ere bigarren deribatua eta super-etorkizuneko puntuak darabiltza eta prozesuak pauso hauek ditu:

1. Lehenengo iragarpena egiten da: $\bar{y}_{n+k}$ kalkulatzeko ondoko ekuazioaren emaitza bezala:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h\beta_k f_{n+k} + h^2 \gamma_k g_{n+k}, \quad y_{n+k} := \bar{y}_{n+k} \quad (5.50)$$

(5.50) adierazpenean $\alpha_k = 1$ da eta gainerako koefizienteak (5.50) adierazpena $(k+1)$ ordenakoa izateko eran aukeratzen dira.

2. Bigarren iragarpena egiten da: $\bar{y}_{n+k+1}$ kalkulatzeko (5.50) adierazpena bera erabiliz:

$$\sum_{j=0}^k \alpha_j y_{n+j+1} = h\beta_k f_{n+k+1} + h^2 \gamma_k g_{n+k+1}, \quad y_{n+k+1} := \bar{y}_{n+k+1}$$

3. Super-etorkizuneko puntuari dagokion bigarren deribatua kalkulatzeko da:

$$\bar{g}_{n+k+1} = g(\bar{y}_{n+k+1})$$

4. Zuzentzailea aplikatzeko da: azkenik, y_{n+k} kalkulatzeko ondoko ekuazioa erabiliz:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = h\hat{\beta}_k f_{n+k} + h^2 (\hat{\gamma}_k g_{n+k} - \hat{\gamma}_{k+1} \bar{g}_{n+k+1}) \quad (5.51)$$

Metodo honi dagozkion $\hat{\alpha}_j$, $\hat{\beta}_k$, $\hat{\gamma}_k$ eta $\hat{\gamma}_{k+1}$ koefizienteak aipatu dugun erreferentzian aurki daitezke eta metodoaren egonkortasun-ezaugarriak 5.13. taulan jaso dira. Metodo hau seigarren ordenaraino A-egonkorra dela ikus daiteke bertan.

k	1	2	3	4	5	6	7	8	9	10	11	12
ordena	3	4	5	6	7	8	9	10	11	12	13	14
$A(\alpha)$	90°	90°	90°	90°	89,8°	88,3°	85,3°	80,5°	73,5°	61,9°	50,3°	29,9°

5.13. taula. New SDMM metodoei dagozkien egonkortasun-ezaugarriak.

5.8. A-EBDF METODOAK

Hojjati, Rahimi Ardabili eta Hosseini-ren (2004) lanean EBDF metodoa aplikatzen zaio A-BDF metodoari. Hau da, A-BDFak erabiltzen dira iragarle gisa eta EBDF metodoko zuzentzailea mantendu egiten da. A-BDF metodoan agertzen den α balioa doituaz A-EBDF metodoa laugarren ordenaraino A-egonkorra izatea lortzen da, eta $A(\alpha)$ -egonkorra bederatzigarren ordenaraino. Gainera, A-EBDF metodoari dagokion $A(\alpha)$ -egonkortasun angelua BDF, A-BDF eta EBDF metodoetakoa baino handiagoa izatea lortzen da. Metodo berri honen egonkortasun-ezaugarriak 5.14. taulan jaso dira.

k	1	2	3	4	5	6	7	8
ordena	2	3	4	5	6	7	8	9
$A(\alpha)$	90°	90°	90°	88,85°	84,2°	75°	61°	30,50°

5.14. taula. A-EBDF metodoen egonkortasun-ezaugarriak.

5.9. EBDF-ETAN ZUZENTZAILEA ALDATUTA LORTZEN DIREN BESTE METODO BATZUK

EBDFetan erabiltzen den iragarlea BDFa da. Jakin nahi izan dugu ea zer gertatuko litzatekeen BDFen ordeztu NDFA erabiliko bagenu iragarle bezala. NDFak iragarle bezala erabiltzeko hiru aukera ditugu:

- Lehenengo iragarlea BDFa izatea eta bigarrena NDFA. Metodo berri honi EBNDF deitu diogu.
- Lehenengo iragarlea NDFA izatea eta bigarrena BDFa. Metodo berri honi ENBDF deitu diogu.
- Bi iragarleak NDFak izatea. Metodo berri honi ENDF deitu diogu.

5.9.1. ENDF metodoak

ENDF metodoan EBDF metodoko eskemari jarraitzen zaio, baina iragarle bietan NDFA erabiliz. Ondoren, EBDF metodoan egiten den antzera, $\bar{f}_{n+k+1} = f(t_{n+k+1}, \bar{y}_{n+k+1})$ kalkulatu da, eta, azkenik, EBDFetako zuzentzaile bera erabiltzen da. Horrela lortzen den metodoa $(k+1)$ ordenakoa da eta haren mozketaren errore lokala hau da:

$$LTE = h^{k+2} \left(\beta_{k+1} A_k \frac{\partial f}{\partial y} \cdot y^{(k+1)} + C_3 y^{(k+2)} \right) (t_n) + O(h^{k+3}) \quad (5.52)$$

Non: $A_k = C_2 \left(1 - \frac{\hat{\alpha}_{k-1}}{\hat{\alpha}_k} - \frac{\kappa \gamma_k (k+1)}{\hat{\alpha}_k} \right)$, C_2 koefizientea NDF metodoei dagokien errore-konstantea da eta C_3 koefizientea EBDF metodoko (5.12) zuzentzaileari dagokiona.

Lehenengo iragarlea aplikatzen denean, $\bar{y}_{n+k}$ kalkulatzeko da eta emaitza zehatzaren eta zenbakizko emaitzaren arteko diferentzia hauxe litzateke:

$$y(t_{n+k}) - \bar{y}_{n+k}^* = C_2 h^{k+1} y^{(k+1)}(t_n) + O(h^{k+2})$$

C_2 izanik NDF metodoari dagokion errore-konstantea. $\bar{y}_{n+k}^*$ izendatu dugu kokapen-bereganaketa eginez lortzen den balioa.

Bigarren iragarlea kalkulatu ostean, $\bar{y}_{n+k+1}$ balioa daukagu eta une honetan emaitza zehatzaren eta kalkulatuaren arteko diferentzia hauxe da:

$$y(t_{n+k+1}) - \bar{y}_{n+k+1}^* = C_2 \left(1 - \frac{\hat{\alpha}_{k-1}}{\hat{\alpha}_k} - \frac{\kappa \gamma_k (k+1)}{\hat{\alpha}_k} \right) h^{k+1} y^{(k+1)}(t_n) + O(h^{k+2})$$

Azkenik, zuzentzailea aplikatzen dugu eta aplikatu ostean lortuko dugu prozesu osoari dagokion mozketa-errore lokala, (5.52) adierazpenak emana dagoena.

Adibidea

$k=2$ kasurako bigarren iragarlearen ondorengo mozketa-errore lokalaren balioa ondorioztatu nahi izan dugu. Horretarako, mozketa-errore lokalaren definizioa erabiliko dugu eta $\bar{y}_{n+2}^*$ deituko diogu, aurreko balioak zehatzak direla suposatuta lehenengo iragarlea aplikatu ondoren lortzen dugun balioari. Horrela, lehenengo iragarle bezala NDF2 aplikatu ostean daukagun mozketa-errore lokala hauxe da:

$$LTE = y(t_{n+2}) - \bar{y}_{n+2}^* = C_2 h^3 y'''(t_n) + O(h^4) \quad (5.53)$$

C_2 konstantea NDF2 metodoari dagokion errore-konstantea izanik.

Bigarren iragarlea aplikatuko dugu berriro ere NDF2a erabiliz eta $\bar{y}_{n+3}^*$ kalkulatu dugu:

$$\hat{\alpha}_0 y(t_{n+1}) + \hat{\alpha}_1 \bar{y}_{n+2}^* + \hat{\alpha}_2 \bar{y}_{n+3}^* = h \bar{f}_{n+3}^* + \kappa \gamma_2 \nabla^3 \bar{y}_{n+3}^* \quad (5.54)$$

$\nabla^3 \bar{y}_{n+3}^*$ adierazpena garatzen badugu, (5.54) ekuaziotik $\bar{y}_{n+3}^*$ askatu ahal izango dugu:

$$\bar{y}_{n+3}^* = \frac{-\hat{\alpha}_0 y(t_{n+1}) - \hat{\alpha}_1 \bar{y}_{n+2}^* + h \bar{f}_{n+3}^* + \kappa \gamma_2 (\bar{y}_{n+3}^* - 3\bar{y}_{n+2}^* + 3y(t_{n+1}) - y(t_n))}{\hat{\alpha}_2} \quad (5.55)$$

Horrela, bada, mozketa-errore lokalaren balioa bigarren iragarlea aplikatu ostean haxe litzateke:

$$y(t_{n+3}) - \bar{y}_{n+3}^* = y(t_{n+3}) - \left(\frac{-\hat{\alpha}_0 y(t_{n+1}) - \hat{\alpha}_1 \bar{y}_{n+2}^* + h \bar{f}_{n+3}^* + \kappa \gamma_2 (\bar{y}_{n+3}^* - 3\bar{y}_{n+2}^* + 3y(t_{n+1}) - y(t_n))}{\hat{\alpha}_2} \right)$$

(5.53) adierazpenetik $\bar{y}_{n+2}^*$ -ren balioa aterako dugu:

$$\bar{y}_{n+2}^* = y(t_{n+2}) - C_2 h^3 y'''(t_n) + O(h^4) \quad (5.56)$$

Eta (5.56) adierazpena (5.55) adierazpenean ordezkatzuz bigarren iragarlearen ondorengo mozketa-errore lokala era honetan idatzita izango dugu:

$$y(t_{n+3}) - \left(\frac{-\hat{\alpha}_0 y(t_{n+1}) - \hat{\alpha}_1 (y(t_{n+2}) - C_2 h^3 y'''(t_n)) + h \bar{f}_{n+3}^*}{\hat{\alpha}_2} \right) - \frac{\kappa \gamma_2 (\bar{y}_{n+3}^* - 3(y(t_{n+2}) - C_2 h^3 y'''(t_n)) + 3y(t_{n+1}) - y(t_n))}{\hat{\alpha}_2} + O(h^4) \quad (5.57)$$

Aurreko adierazpenean batugaiak era egokian ordenatuz, haxe lortzen da:

$$y(t_{n+3}) - \left(\frac{-\hat{\alpha}_0 y(t_{n+1}) - \hat{\alpha}_1 y(t_{n+2}) + h \bar{f}_{n+3}^* + \kappa \gamma_2 (\bar{y}_{n+3}^* - 3y(t_{n+2}) + 3y(t_{n+1}) - y(t_n))}{\hat{\alpha}_2} \right) - \frac{\hat{\alpha}_1}{\hat{\alpha}_2} C_2 h^3 y'''(t_n) - 3\kappa \gamma_2 \frac{C_2}{\hat{\alpha}_2} h^3 y'''(t_n) + O(h^4) \quad (5.58)$$

(5.58) adierazpeneko lehenengo bi batugaiak NDFaren mozketa-errore lokala dira eta, beraz, zera daukagu:

$$y(t_{n+3}) - \left(\frac{-\hat{\alpha}_0 y(t_{n+1}) - \hat{\alpha}_1 y(t_{n+2}) + h \bar{f}_{n+3}^* + \kappa \gamma_2 (\bar{y}_{n+3}^* - 3y(t_{n+2}) + 3y(t_{n+1}) - y(t_n))}{\hat{\alpha}_2} \right) = C_2 h^3 y'''(t_n) + O(h^4) \quad (5.59)$$

Eta (5.59) adierazpena (5.58) adierazpenean ordezkaturaz, NDF2 iragarlea bi aldiz aplikatu osteko mozketaren errorea lokalak daukagu:

$$\begin{aligned} y(t_{n+3}) - \bar{y}_{n+3}^* &= C_2 h^3 y'''(t_n) - \frac{\hat{\alpha}_1}{\hat{\alpha}_2} C_2 h^3 y'''(t_n) - 3\kappa\gamma_2 \frac{C_2}{\hat{\alpha}_2} h^3 y'''(t_n) + O(h^4) \\ &= C_2 \left(1 - \frac{\hat{\alpha}_1}{\hat{\alpha}_2} - \frac{3\kappa\gamma_2}{\hat{\alpha}_2} \right) h^3 y'''(t_n) + O(h^4) \end{aligned}$$

ENDF metodoaren egonkortasun-azterketa egiteko metodoa test-ekuazioari aplikatu behar zaio eta 3. mailako ekuazio honetara iristen gara:

$$A\hat{h}^3 + B\hat{h}^2 + C\hat{h} + D = 0$$

$$\text{Non: } \begin{cases} A = -\beta_k r^{k+1} \\ B = 2(\hat{\alpha}_k - \kappa\gamma_k) \beta_k r^{k+1} + T - \beta_{k+1} S \\ C = -\beta_k (\hat{\alpha}_k - \kappa\gamma_k)^2 r^{k+1} - 2(\hat{\alpha}_k - \kappa\gamma_k) T + (\hat{\alpha}_k - \kappa\gamma_k) \beta_{k+1} S - \beta_{k+1} (-\hat{\alpha}_{k-1} - (k+1)\kappa\gamma_k) R \\ D = (\hat{\alpha}_k - \kappa\gamma_k)^2 T \end{cases}$$

Eta:

$$\begin{cases} R = (-1)^{k+1} \binom{k+1}{0} \kappa\gamma_k + \sum_{j=1}^k r^j \left(-\hat{\alpha}_{j-1} + (-1)^{k+1-j} \binom{k+1}{j} \kappa\gamma_k \right) \\ S = (-1)^k r \kappa\gamma_k - \sum_{j=1}^{k-1} r^{j+1} \left(-\hat{\alpha}_{j-1} + (-1)^{k+1-j} \binom{k+1}{j} \kappa\gamma_k \right) \\ T = \sum_{j=0}^k \alpha_j r^{j+1} \end{cases}$$

5.15. taulan jaso dira ENDFen egonkortasun-ezaugarriak.

5.9.2. ENBDF eta EBNDF metodoak

ENBDF metodoan NDF metodoa erabiltzen da lehenengo iragarle bezala eta BDFa bigarren iragarle bezala. EBNDFan alderantziz egiten da, lehenengoz BDFarekin iragarritik eta bigarrenez NDFarekin. Kasu guztietan, bai aurreko atalean ikusi dugun ENDF kasuan bai eta beste bi hauetan ere, mozketaren errorea lokalak honela emana dator:

$$h^{k+2} \left[\beta_{k+1} A_k \frac{\partial f}{\partial y} y^{(k+1)} + C_3 y^{(k+2)} \right] (t_n) + O(h^{k+3}) \quad (5.60)$$

C_3 koefizientea EBDF metodoko (5.12) zuzentzaileari dagokiona izanik eta (5.60) adierazpeneko A_k konstantea iragarleen arabera ondokoa izanik:

- EBDF kasuan (BDF-BDF iragarleak erabilita):

$$A_k = C_1 \left(-\frac{\hat{\alpha}_{k-1}}{\hat{\alpha}_k} + 1 \right)$$

- ENDF kasuan (NDF-NDF iragarleak erabilita):

$$A_k = C_2 \left(-\frac{\hat{\alpha}_{k-1}}{\hat{\alpha}_k} - \kappa(k+1) \gamma_k \frac{1}{\hat{\alpha}_k} + 1 \right)$$

- ENBDF kasuan (NDF-BDF iragarleak erabilita):

$$A_k = \left(-C_2 \frac{\hat{\alpha}_{k-1}}{\hat{\alpha}_k} + C_1 \right)$$

- EBNDF kasuan (BDF-NDF iragarleak erabilita):

$$A_k = \left(-C_1 \frac{\hat{\alpha}_{k-1}}{\hat{\alpha}_k} - C_1 \kappa(k+1) \gamma_k \frac{1}{\hat{\alpha}_k} + C_2 \right)$$

C_1 BDF metodoaren errore-konstantea izanik eta C_2 NDF metodoarena.

Egonkortasun-azterketa egiteko metodo bakoitza test-ekuazioari aplikatzen diogu, 3. mailako polinomio karakteristikoa lortuz:

$$A\hat{h}^3 + B\hat{h}^2 + C\hat{h} + D = 0$$

Kasuan kasu, hauek dira polinomio karakteristikoko koefizienteen balioak:

- EBNDFFen kasuan:

$$\begin{cases} A = -\beta_k r^k \\ B = (2\hat{\alpha}_k - \kappa\gamma_k)\beta_k r^k + T - \beta_{k+1}S \\ C = -\beta_k \hat{\alpha}_k (\hat{\alpha}_k - \kappa\gamma_k) r^k - (2\hat{\alpha}_k - \kappa\gamma_k)T + \hat{\alpha}_k \beta_{k+1}S + \beta_{k+1}(-\hat{\alpha}_{k-1} - (k+1)\kappa\gamma_k)R \\ D = \hat{\alpha}_k (\hat{\alpha}_k - \kappa\gamma_k)T \end{cases}$$

$$\text{Non: } \begin{cases} R = \sum_{j=0}^{k-1} \hat{\alpha}_j r^j, & T = \sum_{j=0}^k \alpha_j r^j, \\ S = (-1)^k \kappa\gamma_k - \sum_{j=1}^{k-1} r^j \left(-\hat{\alpha}_{j-1} + (-1)^{k+1-j} \binom{k+1}{j} \kappa\gamma_k \right) \end{cases}$$

- ENBDFen kasuan:

$$\begin{cases} A = -\beta_k r^{k+1} \\ B = (2\hat{\alpha}_k - \kappa\gamma_k)\beta_k r^{k+1} + T - \beta_{k+1}S \\ C = -\beta_k \hat{\alpha}_k (\hat{\alpha}_k - \kappa\gamma_k) r^{k+1} - (2\hat{\alpha}_k - \kappa\gamma_k)T + (\hat{\alpha}_k - \kappa\gamma_k)\beta_{k+1}S + \beta_{k+1}\hat{\alpha}_{k-1}R \\ D = \hat{\alpha}_k (\hat{\alpha}_k - \kappa\gamma_k)T \end{cases}$$

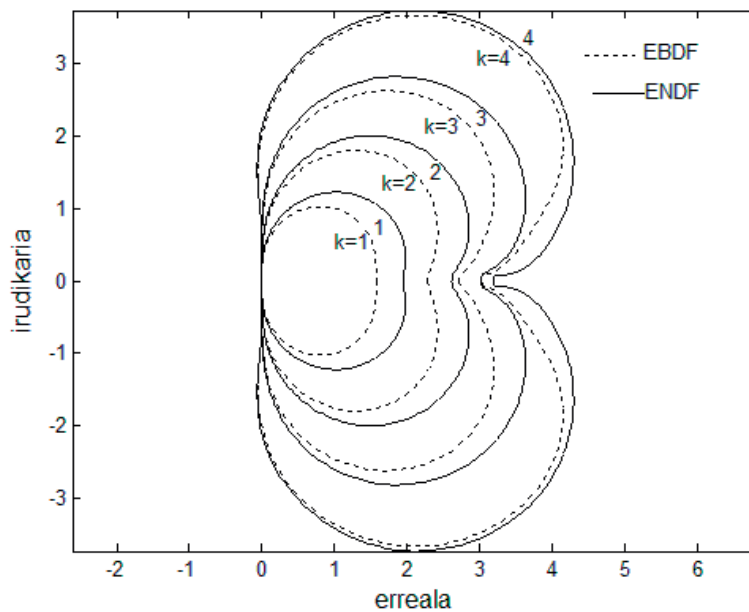
$$\text{Non: } \begin{cases} R = (-1)^{k+1} \binom{k+1}{0} \kappa\gamma_k + \sum_{j=1}^k r^j \left(-\hat{\alpha}_{j-1} + (-1)^{k+1-j} \binom{k+1}{j} \kappa\gamma_k \right) \\ S = \sum_{j=0}^{k-2} \hat{\alpha}_j r^{j+2}, & T = \sum_{j=0}^k \alpha_j r^{j+1} \end{cases}$$

5.15. taulan jaso dira iragarle bezala BDFak edo NDFak erabilia sortzen diren 4 metodoen egonkortasun-ezaugarriak. Lau metodoak dira laugarren ordenaraino A-egonkorak. Bosgarren ordenan ($k=4$ denean), bigarren iragarlea berdin mantenduta, lehenengo iragarle bezala BDFa darabilgunean lortzen ditugun metodoek (EBDF eta EBNDFF) egonkortasun-ezaugarri hobeak dituzte lehenengo iragarle bezala NDFak darabiltzatenek baino (ENDF eta ENBDF): EBDFak ENBDFak baino hobe eta EBNDFFak ENDFak baino hobe. Aldiz, lehenengo iragarlea berdin mantenduta, bigarren iragarle bezala NDFak darabiltzaten metodoek (EBNDFF eta ENDF) bigarren iragarle bezala BDFa darabiltzenek baino egonkortasun-ezaugarri hobeak dituzte: EBNDFFak EBDFak baino hobe eta ENDFak ENBDFak

baino hobe. Konparaketa biak kontuan izanda, lehenengo iragarle bezala BDF-a darabilena eta bigarren iragarle bezala NDFa darabilena da egonkortasun-ezaugarriarik onenak dituen metodoa, hau da, EBNDFa. 5.10. irudian EBDF eta ENDF metodoei dagozkien egonkortasun-eremuak irudikatu dira.

k	ordena	EBDF $A(\alpha)$	EBNDF $A(\alpha)$	ENDF $A(\alpha)$	ENBDF $A(\alpha)$
1	2	90°	90°	90°	90°
2	3	90°	90°	90°	90°
3	4	90°	90°	90°	90°
4	5	87,61°	87,68°	87,54°	87,49°

5.15. taula. EBDF, EBNDF, ENDF eta ENBDF metodoen egonkortasun-ezaugarriak.



5.10. irudia. EBDF eta ENDF metodoen egonkortasun-eremuak (kurben kanpoko aldeak).

6. MATLABeko zenbait funtzio EDak askatzeko

MATLABen hasierako baliodun EDA baten edo EDA sistema baten zenbakizko soluzioa bilatzen laguntzen duten zenbait funtzio daude inplementatuta. Funtzio horietako batzuk, besteak beste, ode23, ode45 eta ode15s funtzioak dira. Ode23 eta ode45 funtzioak kateatutako Runge-Kutta metodo bitan oinarritzen dira eta EDaren autobalioen parte erreala txikia den kasuetan eraginkorrak dira. Ode15s funtzioa, aldiz, BDF eta NDFetan oinarritua, autobalioen parte erreala handia den kasuetan da eraginkorrena. Atal honetan bereziki ode45 eta ode15s funtzioak aztertu ditugu, baina ode23 funtzioari ere egin diogu aipamena.

6.1. EDA-K ASKATZEKO ODE45 ETA ODE23 FUNTZIOAK

Ode45 funtzioa kateatutako DOPRI(5,4) metodoan oinarrituta dago (Dormand eta Prince, 1980). 3. kapituluan ikusi dugunez, DOPRI(5,4) metodoa 7 etapako Runge-Kutta metodo esplizitu bat da, nahiz eta praktikan 6 etapako metodoa den, $\hat{b}^T$ bektorearen azken koefizienteak zero baita. FSAL (*First Same As Last*) izenaz ezagutzen diren metodoetako bat da, zeinak esan nahi baitu gauden pausoko azken etapako deribatuaren ebaluazioak hurrengo pausoko lehen etaparako balio duela. Ode45 funtzioak bosgarren ordenako metodoaren emaitza ematen digu, zeina $\hat{y}_{n+1}$ deituko dugun eta $\hat{b}^T$ bektoreko koefizienteak erabilita kalkulatzen den.

Ode23 funtzioa ere kateatutako RK 3(2) Runge-Kutta esplizitu batean oinarrituta dago (Bogacki eta Shampine, 1989). Lau etapako metodoa da, baina kasu honetan ere $\hat{b}^T$ bektorearen azken koefizienteak nulua denez, hiru etapako metodoa da praktikan, FSAL motakoa hain zuzen. Metodo horri dagozkion koefizienteak 6.1. taulan jaso dira. Kasu honetan ere metodoko ordenarik altuenak ematen duen emaitza da integrazioan aurrera egiteko erabiltzen dena (hirugarren ordena kasu honetan), zeina $\hat{y}_{n+1}$ deituko dugun eta $\hat{b}^T$ bektoreko koefizienteak erabilita kalkulatzen den.

c_i	a_{i1}	a_{i2}	a_{i3}	a_{i4}
0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
b^T	$\frac{7}{24}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{8}$
$\hat{b}^T$	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	0
E^T	$-\frac{5}{72}$	$\frac{1}{12}$	$\frac{1}{9}$	$-\frac{1}{8}$

6.1. taula. RK3(2) metodoko koefizienteak.

6.1.1. Errorearen estimazioa ode45 eta ode23 funtzioetan

Errore globalaren kalkuluak daukan arazorik handiena, berau kalkulatzeko egin beharreko kalkuluen kopuru handiak eta konplexuak sortzen duen kostu konputazionala da. Arrazoi hori dela-eta, algoritmo gehienek errore lokalaren estimazio bat baino ez dute kalkulatu. Algoritmo batzuek erabiltzen duten errore lokalaren estimazioa elkarren segidako ordena duten metodoak erabilia lortu diren emaitzen arteko diferentzia da, hau da, ondoz ondoko ordenako emaitzen diferentzia. Diferentzia hori tolerantzia baten azpitik mantentzen saiatzen da EDA askatzen ari den funtzioa. Kateatutako Runge-Kutta metodoen abantaila bat $(p, p+1)$ ordenako metodoek ematen dizkiguten balioak eskura izatea da. Kateatutako Runge-Kutta bateko ondoz ondoko ordenako metodoek a_{ij} eta c_i koefizienteak partekatzen dituzte eta normalean baino kalkulu gutxiago egin behar dira bien diferentzia lortzeko. Horrek esan nahi du, ezen kateatutako Runge-Kutta metodoetan ondoz ondoko ordenako emaitza biren arteko diferentzia kalkulatzeko konputazionalki merkea dela. Abantaila hori aprobetxatzen dute ode45 eta ode23 funtzioek, zeinek ondoz ondoko ordenako bi metodok emandako emaitzen diferentzia erabiltzen duten errore lokalaren estimazio gisa. Hau da, $(p+1)$ ordenako metodoaz lortutako $\hat{y}_{n+1}$

emaitzaren eta p ordenan lortutako y_{n+1} emaitzaren arteko diferentzia darabilte errore lokalaren estimazio gisa:

$$est = \hat{y}_{n+1} - y_{n+1} = [y(t_n + h) - y_{n+1}] - [y(t_n + h) - \hat{y}_{n+1}] \quad (6.1)$$

(6.1) ekuazioko lehenengo kortxetearen barruko adierazpena p ordenako metodoaren mozketa-errore lokala da, eta bigarren kortxeteko adierazpena, berriz, $(p+1)$ ordenako metodoaren mozketa-errore lokala.

$$\begin{cases} LTE^{(p)} = y(t_n + h) - y_{n+1} = O(h^{p+1}) \\ LTE^{(p+1)} = y(t_n + h) - \hat{y}_{n+1} = O(h^{p+2}) \end{cases} \quad (6.2)$$

(6.2) adierazpena kontuan hartuz, honela geratzen zaigu (6.1) adierazpena:

$$\begin{aligned} est &= \hat{y}_{n+1} - y_{n+1} = [y(t_n + h) - y_{n+1}] - [y(t_n + h) - \hat{y}_{n+1}] = \\ &= LTE^{(p)} + O(h^{p+2}) \end{aligned} \quad (6.3)$$

Ode23 funtzioaren kasuan $p=2$ da eta ode45 funtzioaren kasuan $p=4$. Ohartu (6.2) eta (6.3) adierazpenetan (p) eta $(p+1)$ goi-indizeak erabili ditugula ordena bakoitzeko metodoei lotutako mozketa-errore lokalak izendatzeko. Runge-Kutta metodoetan y_{n+1} eta $\hat{y}_{n+1}$ balioek duten adierazpena gogoratzen dugu:

$$\begin{cases} \hat{y}_{n+1} = h \sum_{i=1}^s \hat{b}_i k_i \\ y_{n+1} = h \sum_{i=1}^s b_i k_i \end{cases} \quad (6.4)$$

Metodo esplizituetan $k_i = f(x_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j)$, $i=1,2,\dots,s$ izanik.

(6.4) adierazpenak (6.3) adierazpenean ordezkatzek baditugu, ode45 eta ode23 funtzioek darabilten errore lokalaren estimazioaren adierazpen baliokide honetara iritsiko gara:

$$est = \hat{y}_{n+1} - y_{n+1} = h \sum_{i=1}^s (\hat{b}_i - b_i) k_i \quad (6.5)$$

Pausoa ontzat hartzea nahi badugu, pauso bakoitzeko errorearen estimazioak definitutako tolerantziaren balio bat baino txikiagoa edo berdina izan beharko du. Gogora dezagun tolerantziaren adierazpena:

$$tol^i = Atol^i + |y^i| Rtol, \quad i = 1, 2, \dots, m \quad (6.6)$$

m askatzen ari garen $y' = f(t, y)$ ekuazio diferentzial arruntaren sistemako dimentsioa izanik. $Rtol$ eta $Atol$ deitu diegu hurrenez hurren tolerantzia erlatiboari eta absolutuari. $Rtol = 0$ den kasuan, (6.6) adierazpeneko tolerantzia absolutua izango da. $Atol = 0$ denean, aldiz, tolerantzia erlatiboa izango da. Tolerantzia erlatiboa eskalar bat izaten da, eta absolutua, aldiz, bektorea; soluzioko osagai bakoitzarentzat tolerantzia absolutu bat definitu dezakegu, alegia. Tolerantzia absolutua eta erlatiboa erlazionatzea posible da. Horretarako, nahikoa da soluzioaren osagai bakoitzeko y_{min}^i balio esanguratsu bat definitzea. Horrela, tolerantzia absolutua honela idatzi ahal izango dugu:

$$Atol = Rtol \cdot y_{min}^i \quad (6.7)$$

Eta (6.7) adierazpena kontuan izanik, posible da tolerantziaren adierazpena bakarrik tolerantzia erlatiboaren menpe jartzea:

$$tol^i = (y_{min}^i + |y^i|) Rtol, \quad i = 1, 2, \dots, m \quad (6.8)$$

Eta eman dugun pausoa ona izango da, ondoko baldintza betetzen bada:

$$\|est\| \leq \|tol\| \quad (6.9)$$

Besterik esan ezean, ode23, ode45 eta ondoren ikusiko dugun ode15s funtzioak darabilten norma osagairik handienean oinarritutakoa da, norma euklidearra ere erabil dezakete, ordea. Osagairik handieneko norma honela emana dago:

$$\|est\| = \|\hat{y}_{n+1} - y_{n+1}\| = \max_{i=1,2,\dots,m} |\hat{y}^i - y^i| \quad (6.10)$$

tol^i tolerantzia osagai bakoitzerako definituta dugunez, (6.8) estimazioak bete beharreko tolerantzia-baldintza, (6.9) adierazpenaren bidez emana datorrena, era honetan idatz daiteke:

$$\blacksquare \text{ Osagai maximoko normaren kasuan: } \max_{i=1,2,\dots,m} \left(\frac{|\hat{y}^i - y^i|}{y_{min}^i + |y^i|} \right) \leq Rtol \quad (6.11)$$

$$\blacksquare \text{ Norma euklidearraren kasuan: } \sqrt{\sum_{i=1}^m \left(\frac{|\hat{y}^i - y^i|}{y_{min}^i + |y^i|} \right)^2} \leq Rtol \quad (6.12)$$

Eta lortu ditugun (6.11) eta (6.12) adierazpenak aurretik izan dugun (6.9) adierazpenaren baliokideak dira. (6.11) eta (6.12) adierazpenetako normak *normalizatuta* hartu ditugu eta *Rtol* baino txikiagoak izan beharko dute pausoa balekotzat emateko. Bi adierazpen horietako norma normalizatua izendatzeko era ezberdin bat erabiltzea ekiditeko asmoz, aurretik erabili dugun $\|est\|$ izendegia erabiliko dugu berriz, eta, beraz, honela idatziko dugu pauso bakoitzean bete beharreko baldintza:

$$\|est\| \leq Rtol \quad (6.13)$$

Eta pauso bakoitzean (6.13) adierazpena bete beharko da, pausoa ontzat hartua izan dadin.

6.1.2. Pauso-tamainaren kontrola

Ode23 eta ode45 funtzioen oinarrian dauden p ordenako metodoen mozketaren errore lokalak honela idatz daitezke:

$$LTE^{(p)} = y(t_n + h) - y_{n+1} = h^{p+1} \phi(t_n) + O(h^{p+2}) \quad (6.14)$$

Aurretik ere esan dugun arren, berriz gogorarazten dugu ode23 funtzioaren kasuan $p = 2$ dela, eta ode45 funtzioaren kasuan $p = 4$ dela.

Demagun h pauso-tamainako pausoa eman dugula eta hurrengo pausoa zer tamainarekin emango dugun kalkulatu nahi dugula. Hurrengo pauso hori, pauso on baten ondorengo pauso berri bat izan daiteke edo, aurreko pausoa txarra izan bada, haren errepikapena ere izan daiteke. Kasu bietan, hurrengo pauso-tamaina kalkulatzeko da azkeneko pauso-tamaina (h) konstante batez biderkatuz. Konstante horri σ deituko diogu eta, beraz, hurrengo pauso-tamaina σh izango da. σ konstantea hurrengo pausoa lortuko dugun errorearen estimazioa kontuan hartuta zehazten da. Horrela, hurrengo pausoa σh pauso-tamainarekin emango bagenu, izango genukeen errorearen estimazioa hauxe litzateke:

$$(\sigma h)^{p+1} \phi(t_{n+1}) + O((\sigma h)^{p+2}) = \sigma^{p+1} h^{p+1} \phi(t_n) + O(h^{p+2}) \quad (6.15)$$

(6.3) eta (6.14) adierazpenak kontuan izanda, honela geratzen zaigu hurrengo pausoa egingo dugun errorearen estimazioaren (6.15) adierazpena:

$$\begin{aligned} (\sigma h)^{p+1} \phi(t_{n+1}) + O((\sigma h)^{p+2}) &= \sigma^{p+1} h^{p+1} \phi(t_n) + O(h^{p+2}) \\ &= \sigma^{p+1} est + O(h^{p+2}) \end{aligned} \quad (6.16)$$

(6.16) adierazpenak σh tamainako pauso bat eman ostean egiten den errore lokalaren adierazpena ematen digu, eta adierazpen hori aurreko pausoko errorearen estimazioaren menpe dago idatzia (*est* baita aurreko pausoko errorearen estimazioa). Tolerantzia-baldintza pasatzea ahalbidetuko duen pauso-tamainarik handiena aukeratzea, σ konstantea baldintza hau betetzeko eran kalkulatzeko litzateke: $\|\sigma^{p+1} est\| \approx Rtol$. Hau da:

$$\sigma = \left(\frac{Rtol}{\|est\|} \right)^{1/(p+1)} \quad (6.17)$$

Eta pauso-tamaina optimoa hauxe litzateke:

$$h_{opt} = h \cdot \sigma = h \cdot \left(\frac{Rtol}{\|est\|} \right)^{1/(p+1)} \quad (6.18)$$

Pauso-tamaina optimoko (6.18) adierazpenean emango duen pausoak tolerantzia-baldintza ziur pasa dezan, 0,8 eta 0,9 bezalako konstanteak erabiltzen dira. Konstante horiek (6.18) adierazpenak ematen digun pausoa apur bat txikitu egiten dute, baina horrela pausoa ona izateko probabilitatea handiagoa da. Ode45 eta ode23 funtzioek 0,8 konstantea darabilte ziurtasuneko faktore gisa, eta, beraz, hauxe da pauso on zein txar baten ostean ematen duten pausoa:

$$h_{berria} = 0,8 \cdot h_{opt} = 0,8 \cdot h \cdot \left(\frac{Rtol}{\|est\|} \right)^{1/(p+1)} \quad (6.19)$$

6.1.3. Lehenengo pauso-tamaina

Aspalditik kodeei lehenengo pauso-tamaina emateko ohitura egon izan da, eta ode45 eta ode23 funtzioek ere aukera hau badaukate: posible da h_{try} erabilita lehenengo pausoa zer tamainarekin eman behar duen kodeari esatea. Edozelan ere, hasierako pauso horrek pauso txar bat eragingo balu, algoritmoak hasierako pauso-tamaina hori segituan erreparatuko luke. Kode bietan, lehenengo pauso-tamaina esaten ez diogun kasuetan *absh* izeneko aldagai bat definitzen da, ondoko eran:

$$absh = \min(h_{max}, htspan) \quad (6.20)$$

h_{max} aldagaiak kodeak eman dezakeen pauso-tamaina maximoa izanik, $h_{max} = 1$ delarik. Eta *htspan* EDAREN soluzioa aurkitu nahi dugun denbora-tartearen luzera izanik. Kode bietan pauso-tamaina minimoa h_{min} ere definitzen da, ondoko eran:

$$h_{\min} = 16 \cdot \text{eps}(t) = \begin{cases} 16 \cdot t \cdot \text{eps}, & t \neq 0 \\ 16 \cdot 4,9407 \cdot 10^{-324}, & t = 0 \end{cases} \quad (6.21)$$

$\text{eps} = 2,2204 \cdot 10^{-16}$ izanik. Kode bietan rh izeneko aldagai bat ere definitzen da:

$$rh = \frac{1,25 \|y_0'\|}{\|y_0\| \cdot Rtol^{1/(p+1)}} \quad (6.22)$$

Non $Rtol$ tolerantzia erlatiboa, y_0 ekuazio diferentzialaren hasierako balioa eta y_0' hasierako puntuan deribatuaren balioa diren. Behin $absh$ eta rh aldagaiak definituta ditugula, beraien arteko biderketaren emaitza kontrolatzen da eta honen arabera izango da lehenengo pauso-tamaina:

$$absh = \begin{cases} absh, & absh \cdot rh \leq 1 \\ \frac{1}{rh}, & absh \cdot rh > 1 \end{cases} \quad (6.23)$$

Azkenik, (6.23) erabilita lortu den $absh$ pauso-tamaina $[h_{\min}, h_{\max}]$ tartean dagoela ziurtatzen da, $absh = \max(absh, h_{\min})$ eginez. Hori egin ostean lortzen den $absh$ -ren balioa izango da kodeak lehenengo pausoa emateko erabiliko duen tamaina.

Kodeari lehen pauso-tamaina h_{try} esaten zaion kasuan, kodeak bakarrik ziurtatuko du esan diogun pauso-tamaina hori $[h_{\min}, h_{\max}]$ tartean kokatzen dela. Horretarako $absh = \min(h_{\max}, \max(h_{\min}, h_{try}))$ eginez.

6.1.4. Pauso on baten osteko hurrengo pauso-tamaina

Pausoa ona izateak $\|est\| \leq Rtol$ baldintza bete duen pauso bat eman dela esan nahi du. Pauso horren osteko pauso-tamaina kalkulatzeko (6.19) adierazpena darabilte ode45 eta ode23 funtzioek. (6.19) adierazpena $Rtol$ tolerantzia erlatiboaren, h momentu honetako pauso-tamainaren eta est errorearen estimazioaren mendekoa da. Ode45 funtzioak pauso on baten osteko pauso-tamaina kalkulatzeko darabilen adierazpena hauxe da:

$$h_{berria} = \begin{cases} 5 \cdot h, & \|est\| \leq Rtol \cdot 1,0486 \cdot 10^{-4} \\ 0,8 \cdot h \cdot \left(\frac{Rtol}{\|est\|} \right)^{1/5}, & Rtol \cdot 1,0486 \cdot 10^{-4} < \|est\| \leq Rtol \end{cases} \quad (6.24)$$

Ode23 funtzioak ere antzeko adierazpena darabil eta hauxe da zehazki darabilena:

$$h_{berria} = \begin{cases} 5 \cdot h, & \|est\| \leq Rtol \cdot 4,0960 \cdot 10^{-3} \\ 0,8 \cdot h \cdot \left(\frac{Rtol}{\|est\|} \right)^{1/3}, & Rtol \cdot 4,0960 \cdot 10^{-3} < \|est\| \leq Rtol \end{cases} \quad (6.25)$$

Beste era batean esanda, ode45 funtzioan errore lokalaren estimazioa $Rtol \cdot 1,0486 \cdot 10^{-4}$ baino txikiagoa edo berdina izan denean, hurrengo pauso-tamaina pauso horretako pauso-tamaina bider bost eginda kalkulatzen du, hurrengo pauso-tamaina oraingoa baino 5 aldiz handiagoa izango da beraz. Errorearen estimazioa $(Rtol \cdot 1,0486 \cdot 10^{-4}, Rtol]$ tartean dagoenean, pauso-tamaina berria (6.24) adierazpeneko 2. formula erabiltuta kalkulatzen du ode45 funtzioak. Konkretuki errorearen estimazioa tolerantzia erlatiboaren berdina denean ($\|est\| = Rtol$) izango da hurrengo pauso-tamaina minimoa, hau da: $h_{berria} = 0,8 \cdot h$ balioa izango du. Gauza bertsua gertatzen da ode23 funtzioaren kasuan ere, aldatzen direnak errore lokalaren estimazioaren limiteak baino ez dira. Beraz, kasu bietan, bai ode45 funtzioaren kasuan baita ode23arenean ere, pauso-tamaina jakin batentzat eta tolerantzia erlatiboaren balio jakin batentzat, zenbat eta txikiagoa izan pauso horretako errore lokalaren estimazioa, handiagoa izango da hurrengo pauso-tamaina.

Deskribatu dugun prozesua ez da agerian dagoen prozesu bat ode45 eta ode23 funtzioetan, ordea. Algoritmo bietan *temp* izeneko aldagai bat definitzen da:

$$temp = 1,25 \left(\frac{\|est\|}{Rtol} \right)^{1/(p+1)} \quad (6.26)$$

Hurrengo pauso-tamaina, h_{berria} , une horretako pauso-tamainaren eta *temp* aldagaiaren arabera definitzen da kode bietan:

$$h_{berria} = \begin{cases} 5 \cdot h, & temp \leq 0,2 \\ \frac{h}{temp}, & temp > 0,2 \end{cases} \quad (6.27)$$

(6.27) adierazpenak erakusten du hurrengo pauso-tamaina *temp* aldagaia txikiago edo berdin 0,2 edo handiago 0,2 den aintzat hartuta definitzen dela. $temp = 0,2$ betetzen den kasua kalkulatu dugu eta emaitzak (6.24) eta (6.25) adierazpenetan jarri ditugunak dira. Emandako pausoa ona izan denean, beste era batean esanda $\|est\| \in [0, Rtol]$ bete denean, *temp* aldagaia tarte honetan egongo da:

$temp \in [0,5 / 4]$. Eta (6.24) eta (6.25) adierazpenak kontuan izanda, hurrengo pauso-tamainak hauxe beteko du kode bietan:

$$0,8 \cdot h \leq h_{berria} \leq 5 \cdot h \quad (6.28)$$

(6.24) eta (6.25) adierazpenetan argi ikusten da pauso-tamaina berria tolerantzia erlatiboaren mendekoa ere badela. Horrela, errorearen estimazio bererako, zenbat eta handiagoa izan darabilgun tolerantzia erlatiboa, hainbat eta handiagoa izango da hurrengo pauso-tamaina.

6.1.5. Pautso txar baten osteko pauso-tamaina

Pausoa txarra izateak esan nahi du lortu den errorearen estimazioak errore-baldintza ez duela betetzen, hau da, $\|est\| > Rtol$ gertatu dela. Kasu horretan pausoa errepikatu beharra dago, jakina, aurrekoa baino pauso-tamaina txikiagoa erabilita. Kasu honetan ere, ode45 eta ode23 funtzioek erabiliko duten pauso-tamaina berria $Rtol$ tolerantzia erlatiboaren, h momentu honetako pauso-tamainaren eta errorearen estimazioaren mendekoa izango da. Ode45 funtzioak pauso txar baten ostean pausoa errepikatzeko darabilen pauso-tamaina hauxe da:

$$h_{berria} = \begin{cases} 0,8 \cdot h \cdot \left(\frac{Rtol}{\|est\|} \right)^{1/5}, & Rtol < \|est\| \leq Rtol \cdot 32.768 \\ 0,1 \cdot h, & \|est\| \geq Rtol \cdot 32.768 \end{cases} \quad (6.29)$$

Ode23 funtzioak ere antzeko adierazpena darabil pauso txar baten osteko pauso-tamaina kalkulatzeko:

$$h_{berria} = \begin{cases} 0,8 \cdot h \cdot \left(\frac{Rtol}{\|est\|} \right)^{1/3}, & Rtol < \|est\| \leq Rtol \cdot 4,0960 \\ 0,5 \cdot h, & \|est\| \geq Rtol \cdot 4,0960 \end{cases} \quad (6.30)$$

Pauso txar hori (6.29) edota (6.30) adierazpenen bidez lortuko genituzkeen pauso-tamainekin errepikatu eta berriro ere txarra suertatuko balitz, hurrengo pauso-tamaina kalkulatu litzateke aurretik eman duguna zati bi eginez. Eta horrela jarraitu behar da harik eta pauso on bat ematea lortzen den arte. Hori guztia egiteko, bi kodeek darabilten agindua hauxe da:

$$h_{berria} = \begin{cases} \max \left(h_{min}, \max \left(h \cdot k_{ode}, \frac{h}{temp} \right) \right), & \text{1. proba pauso txar baten ostean} \\ 0,5 \cdot h, & \text{2. probatik aurrera} \end{cases} \quad (6.31)$$

(6.31) adierazpeneko $temp$ aldagaia lehen (6.26) adierazpenean definitu duguna izanik, $h_{\min}$ kode bietan definituta datorren pauso-tamaina minimoa eta k_{ode} konstantea ode45 funtzioaren kasuan $k_{ode} = 0,1$ balio duena eta ode23 funtzioaren kasuan $k_{ode} = 0,5$. Ode45 funtzioaren kasuan $k_{ode} \cdot h = \frac{h}{temp}$ berdintza $\|est\| = Rtol \cdot 32.768$ denean betetzen da, eta $\|est\| = Rtol \cdot 4,0960$ denean ode23-renean. Hau guztia kontuan izanda, pauso txar baten osteko pauso-tamaina tarte honetan aurkitzen da kodeetako bakoitzean:

- Ode45 funtzioaren kasuan: $0,1 \cdot h \leq h_{berria} \leq 0,8 \cdot h$
- Ode23 funtzioaren kasuan: $0,5 \cdot h \leq h_{berria} \leq 0,8 \cdot h$

Laburbilduz, ode45 funtzioak h tamainako pauso baten ostean (pausoa ona zein txarra izan) emango duen pauso-tamaina adierazpen bakarrean honela bil daiteke:

$$h_{berria} = \begin{cases} 5 \cdot h, & \|est\| \leq Rtol \cdot 1,0486 \cdot 10^{-4} \\ \frac{h}{temp}, & Rtol \cdot 1,0486 \cdot 10^{-4} < \|est\| \leq Rtol \\ \frac{h}{temp}, & Rtol < \|est\| \leq Rtol \cdot 32.768 \\ 0,1 \cdot h, & \|est\| \geq Rtol \cdot 32.768 \end{cases} \quad (6.32)$$

(6.32) adierazpeneko lehenengo biak pauso on baten osteko tamainari dagozkionak lirateke, eta azken biak, ostera, pauso txar baten ostekoak. Azken biak $h/2$ bihurtzen dira berauekin proba egin eta gero ematen den pausoa berriz ere txarra suertatzen denean.

Ode23 funtzioaren kasuan, hauxe litzateke h pauso baten osteko hurrengo pauso-tamaina kalkulatzeko adierazpena:

$$h_{berria} = \begin{cases} 5 \cdot h, & \|est\| \leq Rtol \cdot 4,0960 \cdot 10^{-3} \\ \frac{h}{temp}, & Rtol \cdot 4,0960 \cdot 10^{-3} < \|est\| \leq Rtol \\ \frac{h}{temp}, & Rtol < \|est\| \leq Rtol \cdot 4,0960 \\ 0,5 \cdot h, & \|est\| \geq Rtol \cdot 4,0960 \end{cases} \quad (6.33)$$

Bi funtzio horien kasuan, 0,1; 0,2; 0,5; 0,8 eta 5 konstanteak dira hurrengo pauso-tamaina zehazteko inplementatuta dauden konstanteak. Horien ordeztu batzuk erabiliko balira, (6.32) eta (6.33) adierazpenen ordeztu beste adierazpen batzuk izango genituzke.

6.2. Ode15s FUNTZIOA

Ode15s funtzioa BDF metodoetan oinarrituta dago. Funtzio honetan posible da 1-5 ordenako BDFak erabiltzea. Horretaz gain, 5. kapituluaren ikusi ditugun NDF metodoak ere ezarrita ditu ode15s funtzioak, eta besterik adierazi ezean, NDFak erabiliz kalkulatu du EDAren soluzioa. Askatu behar dugun EDA BDFak erabiliz askatu nahi badugu, adierazi egin beharko diogu, horretarako ode15s funtzioak dituen aukerak erabilia. Ode15s funtzioak ez du ordena konstantean ebatzen. $k=1$ ordenan askatzen hasten da eta ordena gora eginez joaten da. Posible da, baita ere, ode15s funtzioari gehienez zer ordenatan askatu behar duen esatea, hau da, posible da ordena maximoa zein izan behar duen esatea, eta besterik esan ezik, $k=5$ ordenaraino iritsi ahal izango da EDA askatzeko prozesuan.

6.2.1. Zenbait ezaugarri konputazional

BDF eta NDF metodoen adierazpenak gogoratuko ditugu lehenik eta behin:

- BDF metodoak adierazpen honen bidez emanak daude:
$$\sum_{j=0}^k \nabla^j y_{n+k} = hf_{n+k}$$
- NDF metodoen adierazpena, berriz, hau da:
$$\sum_{j=1}^k \frac{1}{j} \nabla^j y_{n+k} = hf_{n+k} + \kappa \gamma_k \nabla^{k+1} y_{n+k}$$

Shampine eta Reichelt-en (1997) lanean BDF eta NDF metodoen adierazpeneko ezkerreko aldea (metodo bietarako berdina dena) idazteko beste era bat ematen da:

$$\sum_{j=1}^k \frac{1}{j} \nabla^j y_{n+k} = \gamma_k (y_{n+k} - y_{n+k}^{(0)}) + \sum_{j=1}^k \gamma_j \nabla^j y_{n+k-1} \quad (6.34)$$

$$\text{non: } \begin{cases} \gamma_j = \sum_{l=1}^j \frac{1}{l} \\ y_{n+k}^{(0)} = \sum_{j=0}^k \nabla^j y_{n+k-1} = \nabla^0 y_{n+k-1} + \nabla y_{n+k-1} + \dots + \nabla^k y_{n+k-1} \\ y_{n+k} - y_{n+k}^{(0)} = \nabla^{k+1} y_{n+k} \end{cases} \quad (6.35)$$

(6.34) adierazpena kontuan hartuz, BDF eta NDF metodoak era honetan idatz daitezke:

$$(1-\kappa)\gamma_k (y_{n+k} - y_{n+k}^{(0)}) + \sum_{j=1}^k \gamma_j \nabla^j y_{n+k-1} = hf(t_{n+k}, y_{n+k}) \quad (6.36)$$

κ parametroa NDF metodoetako parametroa izanik eta BDFen kasuan $\kappa = 0$ izanik. (6.36) adierazpena $(1-\kappa)\gamma_k$ terminoaz zatitzen badugu, BDF eta NDFentzat ondorengo adierazpen baliokidea lortzen da:

$$(y_{n+k} - y_{n+k}^{(0)}) + \frac{\sum_{j=1}^k \gamma_j \nabla^j y_{n+k-1}}{(1-\kappa)\gamma_k} = \frac{hf(t_{n+k}, y_{n+k})}{(1-\kappa)\gamma_k} \quad (6.37)$$

(6.37) adierazpena ekuazio ez-lineal bat da eta, adibidez, Newton-Raphson-en metodoa erabiliz aska daiteke. Horretarako, $d = y_{n+k} - y_{n+k}^{(0)}$ aldagaia definituko dugu, eta, beraz, $y_{n+k} = d + y_{n+k}^{(0)}$ izango da. Aurkeztu berri ditugun azken bi berdintzak (6.37) ekuazioan ordezkaturik, hauxe da lortzen dena:

$$d + \frac{\sum_{j=1}^k \gamma_j \nabla^j y_{n+k-1}}{(1-\kappa)\gamma_k} = \frac{hf(t_{n+k}, d + y_{n+k}^{(0)})}{(1-\kappa)\gamma_k} \quad (6.38)$$

(6.38) adierazpeneko notazioa apur bat sinplifikatuko dugu, aurreko pausoetako balioen menpe (zehazki aurreko pausoetako atzerakako diferentzien menpe) dagoen ψ terminoa definituz:

$$\psi = \frac{1}{(1-\kappa)\gamma_k} \sum_{j=1}^k \gamma_j \nabla^j y_{n+k-1} \quad (6.39)$$

Eta (6.39) adierazpena (6.38) adierazpenean ordezkaturaz, d -ren mendeko ekuazio ez-lineal hau lortzen da:

$$d + \psi - \frac{hf(t_{n+k}, d + y_{n+k}^{(0)})}{(1 - \kappa)\gamma_k} = 0 \quad (6.40)$$

Ode15s funtzioak (6.40) ekuazio ez-lineala Newton-Raphson-en metodoa aplikatuta askatzen du. Newton-en metodoa erabiliz, $\Delta^{(v)}$ -ren balioa lortzen da: $\Delta^{(v)} = J^{-1}R$, iterazioetako R eta J -ren balioak hauek izanik:

$$\begin{cases} R = d^{(v)} + \psi - \frac{hf(t_{n+k}, d^{(v)} + y_{n+k}^{(0)})}{(1 - \kappa)\gamma_k} \\ J = \frac{\partial R(d^{(v)})}{\partial d^{(v)}} = I - \frac{h}{(1 - \kappa)\gamma_k} \cdot \frac{\partial f(t_{n+k}, d^{(v)} + y_{n+k}^{(0)})}{\partial d^{(v)}} \end{cases} \quad (6.41)$$

Behin $\Delta^{(v)}$ daukagunean, $d^{(v)}$ eguneratzen da, $d^{(v+1)} = d^{(v)} - \Delta^{(v)}$ eginez. Pauso bakoitzeko azken iterazioan lortzen den azken $d^{(v+1)}$ -en balioa $(k+1)$ ordenako atzerakako diferentzia da. Balio horri $d^{(f+1)}$ deituko diogu, eta balio hori erabilia lortu ahal izango dugu pauso horretako y_{n+k} -ren balioa:

$$y_{n+k} = y_{n+k}^{(0)} + d^{(f+1)}$$

6.2.2. Diferentzien matrizea

Ode15s funtzioa atzerakako diferentziak oinarritzat hartuz diseinatuta dago. Hori dela-eta, ode15s funtzioak atzerakako diferentzia horiek gordeta dauden matrize bat erabiltzen du. Matrize horren errenkada kopurua askatzen ari garen EDAk duen dimentsioaren berdina da eta haren zutabe kopurua askatzen ari garen ordena maximoa baino 2 unitate handiagoa da. Ode15s funtzioari lan egingo duen ordena maximoa $k = 4$ izatea eskatzen badiogu, diferentzien matrizeak $4 + 2 = 6$ zutabe izango ditu. Oro har, ode15s funtzioak lan egingo duen ordena maximoa k dela suposatuz, atzerakako diferentziak gordeta dauden matrizeak $(k + 2)$ zutabe ditu. Hauxe izango da atzerakako diferentzien matrizea, oro har:

$$dif_{n+k-1}(h) = \begin{pmatrix} \nabla y_{n+k-1} & \nabla^2 y_{n+k-1} & \dots & \nabla^k y_{n+k-1} & \nabla^{k+1} y_{n+k-1} & \nabla^{k+2} y_{n+k-1} \end{pmatrix} \quad (6.42)$$

$$\nabla^j y_{n+k-1} = \nabla(\nabla^{j-1} y_{n+k-1}) \text{ izanik eta } \nabla y_{n+k-1} = y_{n+k-1} - y_{n+k-2}.$$

Atzerakako diferentziek propietate hau ere betetzen dute:

$$\nabla^{j+1}y_{n+k-1} = \nabla^j y_{n+k-1} - \nabla^j y_{n+k-2} \quad (6.43)$$

Pausoaren amaieran y_{n+k} kalkulatzeko da eta $(k+1)$ ordenako atzerakako diferentzia ere lortzen da: $\nabla^{k+1}y_{n+k}$. Pauso on bakoitza eman ostean atzerakako diferentzien matrizea (6.42) eguneratu beharra dago. Eguneraketa erlazio hauek kontuan izanda egiten da:

$$\begin{cases} \text{dif}_{n+k,k+2}(h) = \nabla^{k+1}y_{n+k} - \text{dif}_{n+k-1,k+1}(h) \\ \text{dif}_{n+k,k+1}(h) = \nabla^{k+1}y_{n+k} \\ \text{dif}_{n+k,j}(h) = \text{dif}_{n+k-1,j}(h) + \text{dif}_{n+k,j+1}(h), \quad j = k, (k-1), \dots, 1 \end{cases} \quad (6.44)$$

(6.44) adierazpenean $\text{dif}_{n+k}(h)$ matrizeko i zutabea $\text{dif}_{n+k,i}(h)$ notazioa erabilita adierazi dugu. (6.44) adierazpenetako lehenengo berdintzak esan nahi du ezen $\nabla^{k+1}y_{n+k}$ balioaren eta une honetara arte izan dugun diferentzien matrizean, $\text{dif}_{n+k-1}(h)$ matrizean, $(k+1)$ zutabearen zegoen balioaren arteko kenketa sartzen dugula diferentzien matrizeko $(k+2)$ zutabearen. Aurretik $(k+1)$ posizioan zegoen balioa $\text{dif}_{n+k-1,k+1}(h) = \nabla^{k+1}y_{n+k-1}$ denez eta (6.43) propietatea kontuan hartuz, zera daukagu:

$$\text{dif}_{n+k,k+2}(h) = \nabla^{k+1}y_{n+k} - \nabla^{k+1}y_{n+k-1} = \nabla^{k+2}y_{n+k} \quad (6.45)$$

Hau da, eguneratzen ari garen diferentzien matrizeko $(k+2)$ zutabearen y_{n+k} balioaren $(k+2)$ ordenako atzerakako diferentzia sartu dugu.

(6.44) adierazpeneko bigarren berdintzak esan nahi du $\nabla^{k+1}y_{n+k}$ balioa diferentzien matrizeko $(k+1)$ zutabearen sartuko dela. Horrela, bada, (6.44) adierazpeneko lehen eta bigarren formulek diotena egin ostean, diferentzien matrizearen $(k+1)$ eta $(k+2)$ zutabeak eguneratu dira eta haxe da momentuz daukaguna:

$$\text{dif}_{n+k}(h) = \begin{pmatrix} \nabla y_{n+k-1} & \nabla^2 y_{n+k-1} & \dots & \nabla^k y_{n+k-1} & \nabla^{k+1} y_{n+k} & \nabla^{k+2} y_{n+k} \end{pmatrix} \quad (6.46)$$

(6.44) adierazpeneko hirugarren berdintzak dioena egitea falta zaigu oraindik. Berdintza horretan esaten da diferentzien matrizeko $k, (k-1), \dots, 2, 1$ zutabeetako bakoitzean, adibidez j zutabearen, sartzen dugula aurreko diferentzien matrizean j zutabearen zegoenaren eta oraingoan $(j+1)$ zutabearen dagoenaren arteko batuketara.

Prozesu hau $j = k$ indizean hasten dugu eta $j = 1$ den artean errepikatzen da. Egiten den lehenengo eragiketa hau da:

$$dif_{n+k,k}(h) = \nabla^k y_{n+k-1} + \nabla^{k+1} y_{n+k} \quad (6.47)$$

Eta berriz (6.43) propietatea kontuan hartuta, (6.47) berdintza honela geratzen da:

$$\nabla^k y_{n+k-1} + \nabla^{k+1} y_{n+k} = \nabla^k y_{n+k}$$

Hori egin ostean, $j = k$ zutabean y_{n+k} balioaren k ordenako atzerakako diferentzia sartu dugu:

$$dif_{n+k}(h) = \left(\nabla y_{n+k-1} \quad \nabla^2 y_{n+k-1} \quad \dots \quad \nabla^k y_{n+k} \quad \nabla^{k+1} y_{n+k} \quad \nabla^{k+2} y_{n+k} \right) \quad (6.48)$$

Prozesu hau $j = (k-1), \dots, 1$ balioentzat ordena honetan errepikatuz, eguneratutako diferentzien matrizea lortzen da, zeina t_{n+k} aldiuneko atzerakako diferentzien menpe geratzen den:

$$dif_{n+k}(h) = \left(\nabla y_{n+k} \quad \nabla^2 y_{n+k} \quad \dots \quad \nabla^k y_{n+k} \quad \nabla^{k+1} y_{n+k} \quad \nabla^{k+2} y_{n+k} \right) \quad (6.49)$$

6.2.3. Errorearen estimazioa ode15s funtzioan

Ode15s funtzioak darabilen errorearen estimazioa mozketa-errore lokalaren hurbilpen bat da. Hori dela-eta, k ordenako BDF edo NDFen mozketa-errore lokala zein den gogoratuko dugu:

$$LTE = Ch^{k+1} y^{(k+1)}(t_n) + O(h^{k+2}) \quad (6.50)$$

C konstantea k ordenako BDF edo NDFaren errore-konstantea izanik, askatzen ari garen metodoaren arabera.

Ode15s funtzioko programazioa atzerakako diferentzietan oinarrituta dagoela ikusi dugu eta hauek dira, hain zuzen, errorearen estimaziorako ere erabiltzen direnak. Era honetan egiten da:

- $y(t)$ funtzioa hurbiltzen da, $\{(t_{n+i}, y_{n+i}) : i = -1, 0, 1, 2, \dots, k\}$ $(k+2)$ puntuetatik igarotzen den Newton-en polinomio interpolatzailea erabilita:

$$y(t) \approx Q(t) = y_{n+k} + \sum_{j=1}^{k+1} \nabla^j y_{n+k} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t - t_{n+k-m}) \quad (6.51)$$

- $Q(t)$ funtzioaren $(k+1)$. deribatua kalkulatu da. Horretarako konturatu behar gara ezen (6.51) polinomioaren maila $(k+1)$ dela eta maila hori (6.51) adierazpeneko batuketako $j = k+1$ indizeari dagokiona dela. Hauxe da, hain zuzen, (6.51) polinomioko mailarik handieneko terminoa:

$$\begin{aligned} \nabla^{k+1} y_{n+k} \frac{1}{(k+1)! h^{k+1}} \prod_{m=0}^k (t - t_{n+k-m}) &= \\ &= \nabla^{k+1} y_{n+k} \frac{1}{(k+1)! h^{k+1}} (t - t_{n+k})(t - t_{n+k-1}) \dots (t - t_n) \end{aligned} \quad (6.52)$$

$(k+1)$ aldiz (6.51) adierazpena deribatzen denean, (6.52) batugaiak izan ezik beste batugai guztiek zero ematen dute. (6.52) adierazpenaren $(k+1)$ ordenako deribatua, berriz, hauxe da:

$$Q^{(k+1)}(t) = \nabla^{k+1} y_{n+k} \frac{1}{(k+1)! h^{k+1}} (k+1)! = \nabla^{k+1} y_{n+k} \frac{1}{h^{k+1}} \quad (6.53)$$

Beraz, (6.53) kontuan hartuta, hauxe da daukaguna:

$$y^{(k+1)}(t) \approx Q^{(k+1)}(t) = \nabla^{k+1} y_{n+k} \frac{1}{h^{k+1}} \quad (6.54)$$

Eta BDF eta NDF metodoen mozketaren erroreak lokalaren (6.50) adierazpeneko $y^{(k+1)}(t_n)$ hurbiltzeko (6.54) erabiltzen badugu, honela geratuko zaigu mozketaren erroreak lokalaren adierazpena:

$$LTE = Ch^{k+1} y^{(k+1)}(t_n) + O(h^{k+2}) \approx Ch^{k+1} \nabla^{k+1} y_{n+k} \frac{1}{h^{k+1}} = C \nabla^{k+1} y_{n+k} \quad (6.55)$$

Eta hauxe da, hain zuzen, ode15s funtzioak darabilen erroreakaren estimazioa:

$$est = C \nabla^{k+1} y_{n+k} \approx LTE$$

Errorearen estimazioan ageri den $(k+1)$ ordenako atzerakako diferentzia, diferentzien matrizeko $(k+1)$ zutabea daukagu eskuragarri.

6.2.4. Lehenengo pauso-tamaina

Ode15s funtzioak ere ode23 eta ode45 funtzioek darabiltzaten antzeko irizpideak erabiliz ematen du lehenengo pausoa. h_{ny} erabilita posible da kodeari esatea zer pauso-tamainarekin egin behar duen proba. Probarako pauso-tamaina hau eman ezean, ode23 eta ode45 funtzioek ematen duten era berean ematen du lehenengo pausoa. Ode15s funtzioak ode23 eta ode45 funtzioekiko duen ezberdintasuna zera da, pauso-tamaina minimoa $h_{\min}$ apur bat ezberdin definitzen duela:

$$h_{\min} = 16 \cdot \text{eps} \cdot \text{abs}(t) = \begin{cases} 16 \cdot |t| \cdot \text{eps}, & t \neq 0 \\ 0, & t = 0 \end{cases} \quad (6.57)$$

Ode15s funtzioak pauso-tamaina minimoarentzat darabilen adierazpenak $t=0$ kasurako duen pauso-tamaina minimoa zero da, eta ode23 eta ode45 funtzioen kasuan, aldiz, zero izan ez arren, zerotik oso hurbil dagoen balio bat da. Gainerakoan, ode15s funtzioak ere ode23 eta ode45 funtzioek darabilten prozedura bera darabil lehenengo pauso-tamaina definitzeko.

6.2.5. Pauso on baten osteko hurrengo pauso-tamaina

Emandako pausoa ona izan den kasuan, hau da, $\|est\| \leq Rtol$ baldintza bete egin den kasuan, eta k ordenan lanean gabiltzala suposatuz, ode15s funtzioak k ordenan emango lukeen hurrengo pauso-tamaina, ondorengo adierazpenaren bidez kalkulatu da:

$$h_{opt} = \begin{cases} 10 \cdot h, & \|est\| \leq Rtol \cdot A_k \\ \frac{h}{1,2} \left(\frac{\|est\|}{Rtol} \right)^{1/(k+1)}, & \|est\| > Rtol \cdot A_k \end{cases} \quad (6.58)$$

$A_k = \left(\frac{0,1}{1,2} \right)^{(k+1)}$ izanik. Ordena ezberdinei dagozkien A_k konstanteen balioak 6.2. taulan jaso dira.

k	A_k	A_{k-1} (<i>errkm1</i>)	A_{k+1} (<i>errkp1</i>)
1	$6,944 \cdot 10^{-3}$	-----	$3,644 \cdot 10^{-4}$
2	$5,787 \cdot 10^{-4}$	$5,917 \cdot 10^{-3}$	$2,603 \cdot 10^{-5}$
3	$4,823 \cdot 10^{-5}$	$4,552 \cdot 10^{-4}$	$1,859 \cdot 10^{-6}$
4	$4,019 \cdot 10^{-6}$	$3,501 \cdot 10^{-5}$	$1,328 \cdot 10^{-7}$
5	$3,349 \cdot 10^{-7}$	$2,693 \cdot 10^{-6}$	-----

6.2. taula. Ordena ezberdinetako A_k , A_{k-1} eta A_{k+1} konstanteen balioak.

Ode15s funtzioak ez dauka (6.58) adierazpena esplizituki ezarrita. Berez, *temp* izeneko aldagai bat definitzen du era honetan:

$$temp = 1,2 \left(\frac{\|est\|}{Rtol} \right)^{1/(k+1)} \quad (6.59)$$

Eta *temp* aldagaiak hartzen duen balioaren arabera, honela definitzen da hurrengo pauso-tamaina ode15s funtzioan:

$$h_{opt} = \begin{cases} 10 \cdot h, & temp \leq 0,1 \\ \frac{h}{temp}, & temp > 0,1 \end{cases} \quad (6.60)$$

temp = 0,1 betetzen deneko errorearen estimazioaren balioa kalkulatu dugu eta haxe da:

$$\|est\| = Rtol \cdot \left(\frac{0,1}{1,2} \right)^{(k+1)} \quad (6.61)$$

Eta *temp* = 0,1 betetzen duen errorearen estimazioaren balioa erabilia eta (6.60) adierazpena erabilia iritsi gara hasieran eman dugun pauso-tamaina berriaren (6.58) adierazpenera.

Ode15s funtzioak k ordena bera eta h pauso-tamaina bera erabilia eman diren pauso kopurua zenbatzeko zenbatzaile bat dauka. Horrela, ode15s funtzioak pauso on baten ostean ez du planteatzen ordena-aldaketarik ezta pauso-tamaina aldaketarik ere, gutxienez $(k+2)$ pauso ez badira eman k ordena bera eta h pauso-tamaina bera erabilia. Behin baldintza hau betetzen duten $(k+2)$ pausoak eman

direnean bakarrik alda daitezke metodoaren ordena eta pauso-tamaina. Horrela, ordena berean lanean jarraituz gero hurrengo pauso-tamaina zein izango litzatekeen kalkulatzeko du azaldu dugun (6.58) adierazpena erabilita. Gainera, hurrengo pausoa $(k-1)$ ordenan emango bagenu zer pauso-tamainarekin eman beharko genukeen ere kalkulatzeko du. Jakina, $(k-1)$ ordenari dagokion pauso-tamaina kalkulatzeko du $k > 1$ denean. Horretarako $(k-1)$ ordenari dagokion errorearen estimazioa izan beharko du kontuan. Errorearen estimazio hori k ordenako atzerakako diferentziaren eta dagokion errore-konstantearen arteko biderkadura da, atzerakako diferentzia hori diferentzien matrizeko k zutabearen eskuragarri dagoelarik. Oraindik, ordena maximoa lortu ez badugu, hurrengo pausoa $(k+1)$ ordenan emango balitz zer pauso-tamainarekin egingo litzatekeen ere kalkulatzeko du. Kasu honetan ere, ordena horri dagokion errorearen estimazioa izan beharko da kontuan, zeina diferentzien matrizeko $(k+2)$ zutabearen dagoen $(k+2)$ ordenako atzerakako diferentziaren eta errore-konstantearen biderkadura izango den.

$(k-1)$ ordenari dagokion errorearen estimazioari *errkm1* deituko diogu, eta $(k+1)$ ordenari dagokionari *errkp1*. Ode15s funtzioan *temp_{k-1}* aldagai berri bat definitzen da, ondoko eran:

$$temp_{k-1} = 1,3 \cdot \left(\frac{\|errkm1\|}{Rtol} \right)^{1/k} \quad (6.62)$$

Eta *temp_{k-1}* aldagaia kontuan hartuta, $(k-1)$ ordenan lan egitera pasatuko bagina hurrengo pauso-tamainak zein izan beharko lukeen kalkulatzeko da. *hkm1* deituko diogu pauso-tamaina horri:

$$hkm1 = \begin{cases} 10 \cdot h, & est \leq Rtol \cdot A_{k-1} \\ \frac{h}{temp_{k-1}}, & est > Rtol \cdot A_{k-1} \end{cases} \quad (6.63)$$

$A_{k-1} = \left(\frac{0,1}{1,3} \right)^k$ izanik. $2 \leq k \leq 5$ ordenei dagozkien A_{k-1} konstanteen balioak ere 6.2. taulan jaso dira.

$(k+1)$ ordenarako errorearen estimazioa daukagunean, *errkp1* deitu duguna, *temp_{k+1}* aldagaia definitzen da:

$$temp_{k+1} = 1,4 \cdot \left(\frac{\|errkp1\|}{Rtol} \right)^{1/(k+2)} \quad (6.64)$$

Eta $temp_{k+1}$ aldagaia kontuan hartuta, $(k+1)$ ordenan lan egitera pasatuko bagina, hurrengo pauso-tamainak $hkp1$ izan beharko luke:

$$hkp1 = \begin{cases} 10 \cdot h, & est \leq Rtol \cdot A_{k+1} \\ \frac{h}{temp_{k+1}}, & est > Rtol \cdot A_{k+1} \end{cases} \quad (6.65)$$

$A_{k+1} = \left(\frac{0,1}{1,4} \right)^{(k+2)}$ izanik. 6.2. taulan jaso ditugu $1 \leq k \leq 4$ ordenei dagozkien A_{k+1} konstanteen balioak.

Behin ode15s funtzioak hurrengo pausoa k , $(k-1)$ edota $(k+1)$ ordenan emango bagenu beharko genituzkeen pauso-tamainak kalkulatu dituenean, hurrengo pauso-tamaina eta berau emango den ordena zehazteko jarraitzen duen prozesua hauxe da:

- i. $(k-1)$ eta k ordenei dagozkien pauso-tamainak konparatzen ditu. $hkm1 > h_{opt}$ betetzen bada, $hkm1$ balioa gordeko du h_{opt} aldagaian eta $hkm1$ pauso-tamainari dagokion ordena hartuko du, hau da: $k_{berria} = (k-1)$.
- ii. Ondoren, $(k+1)$ ordenari dagokion pauso-tamaina eta h_{opt} konparatzen ditu. $hkp1 > h_{opt}$ betetzen bada, $hkp1$ balioa gordeko du h_{opt} aldagaian eta prozesu honetako i. puntua aplikatu ostean lortu dugun ordenari unitate bat gehituko dio.
- iii. Azkenik i. eta ii. bete ostean, h_{opt} -n daukagun balioa eta eman dugun azken pausoko pauso-tamaina h konparatzen ditu. $h_{opt} > h$ bada, prozesu honetako i. eta ii. puntuen ostean iritsi gareneko pausoa-tamainaz eta ordenaz emango da hurrengo pausoa. $h_{opt} > h$ ez balitz beteko, azken pausoa emateko erabili diren ordena eta pauso-tamaina mantenduko lirateke hurrengo pausoa ere, hau da k eta h .

Aurreko prozesuko puntu bakoitzetik pasatu beharrak eragiten du pauso- eta ordena-aldaketak sarri ez gertatzea. Goiko prozesua da ode15s funtzioak ordena eta pauso-tamaina aldatu barik hainbat pauso ematearen arrazoia.

6.2.6. Pauso txar baten osteko hurrengo pauso-tamaina

Emandako pausoak tolerantzia-baldintza betetzen ez duenean, hau da, $\|est\| > Rtol$ gertatu bada, pausoa txarra da eta berau errepikatu beharra dago. Kasu

horretan, ordena berarekin lanean jarraituko bagenu eman beharko genukeen pauso-tamaina kalkulatzeko da:

$$h_{opt} = h \cdot \max \left(0, 1, 0,833 \cdot \left(\frac{Rtol}{\|est\|} \right)^{\frac{1}{(k+1)}} \right) \quad (6.66)$$

$k > 1$ kasuan baldin bagaude, hurrengo pauso-tamaina $(k-1)$ ordenan emango bagenu zer pauso-tamainarekin eman beharko genukeen kalkulatzeko da. Lehengo moduan *errkm1* deitu diogu $(k-1)$ ordenan izango genukeen errore-estimazioari eta *hkm1* ordena honetan emango genukeen pausoari:

$$hkm1 = \max \left(hmin, h \cdot \max \left(0, 1, 0,769 \cdot \left(\frac{Rtol}{\|errkm1\|} \right)^{1/k} \right) \right) \quad (6.67)$$

$hkm1 > h_{opt}$ gertatzen bada, hurrengo pauso-tamaina $(k-1)$ ordenan emango da eta pauso-tamaina hori azken pausoaren eta *hkm1*-en arteko minimoa izango da. Hau da, ode15s funtzioak ez du uzten pauso txar baten ostean ematen den pausoa aurreko pausoa baino handiagoa izaterik:

$$h_{berria} = \min(h, hkm1) \quad (6.68)$$

Pauso txar baten osteko saiakera hau ere txarra gertatuko balitz, gainerako saiakerak pauso-tamaina zati bi eginda egingo liriteke:

$$h_{berria} = 0,5 \cdot h, \quad \text{bigarren saiakerak eta hurrengoak} \quad (6.69)$$

6.2.7. Pauso-tamaina aldatu ostean diferentzien matrizearen eguneraketa

Pauso-tamaina aldatzeak atzerakako diferentziek osatzen duten matrizearen eguneraketa eskatzen du. Arrazoi horregatik, pauso-tamaina aldatzeak kostu gehigarri bat dauka ode15s funtzioan. Demagun t_{n+k} aldiunera iritsi garela eta azken pausoko tamaina h izan dela. t_{n+k} aldiunean gaudenez, eskuragarri ditugu $t_{n+k} - jh$, $j = 0, 1, \dots, k$ izanik, aldiuneetan kalkulatu ditugun y_{n+j} balioak. y_{n+j} balio horietatik igarotzen den polinomio interpolatzailearen adierazpena hau da:

$$P(t) = y_{n+k} + \sum_{j=1}^k \nabla^j y_{n+k} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t - t_{n+k-m}) \quad (6.70)$$

(6.70) polinomioan ageri diren $\nabla^j y_{n+k}$ balioak, $j = 1, 2, \dots, k$ izanik, atzerakako diferentziez osatutako matrizearen lehenengo k zutabeetan ditugu eskuragarri. D deituko diogu k ordena arteko atzerakako diferentziak dituen matrize horri:

$$D = \begin{pmatrix} \nabla y_{n+k} & \nabla^2 y_{n+k} & \nabla^3 y_{n+k} & \dots & \nabla^k y_{n+k} \end{pmatrix} \quad (6.71)$$

h pauso-tamainatik h^* pauso-tamainara aldatzeak, jakina $h^* \neq h$ izanik, $P(t)$ polinomioaren balioak $t_{n+k-j}^* = t_{n+k} - jh^*$ puntuetan kalkulatu behar ditugula esan nahi du, $j = 0, 1, \dots, k$ izanik. Era horretan, atzerakako diferentzien beste matrize hau osatuz:

$$D^* = \begin{pmatrix} \nabla y_{n+k}^* & \nabla^2 y_{n+k}^* & \nabla^3 y_{n+k}^* & \dots & \nabla^k y_{n+k}^* \end{pmatrix} \quad (6.72)$$

$P(t)$ polinomioa h edota h^* pauso-tamainak erabilita idazteak berdintza honetara garamatza:

$$P(t) = y_{n+k} + \sum_{j=1}^k \nabla^j y_{n+k} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t - t_{n+k-m}) = y_{n+k} + \sum_{j=1}^k \nabla^j y_{n+k}^* \frac{1}{j! (h^*)^j} \prod_{m=0}^{j-1} (t - t_{n+k-m}^*) \quad (6.73)$$

(6.73) adierazpenean y_{n+k} faktorea sinplifikatuz berdintza honetara iristen gara:

$$\sum_{j=1}^k \nabla^j y_{n+k} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t - t_{n+k-m}) = \sum_{j=1}^k \nabla^j y_{n+k}^* \frac{1}{j! (h^*)^j} \prod_{m=0}^{j-1} (t - t_{n+k-m}^*) \quad (6.74)$$

(6.74) berdintza $t = t_{n+k-r}^*$ balioetan, $r = 1, \dots, k$ izanik, ebaluatzen badugu, honelako k berdintza lortuko da:

$$\sum_{j=1}^k \nabla^j y_{n+k} \frac{1}{j! h^j} \prod_{m=0}^{j-1} (t_{n+k-r}^* - t_{n+k-m}) = \sum_{j=1}^k \nabla^j y_{n+k}^* \frac{1}{j! (h^*)^j} \prod_{m=0}^{j-1} (t_{n+k-r}^* - t_{n+k-m}^*) \quad (6.75)$$

Ondorengo adierazpenak kontuan hartuz, zera daukagu:

$$\begin{cases} t_{n+k-r}^* - t_{n+k-m} = (t_{n+k} - rh^*) - (t_{n+k} - mh) = mh - rh^* \\ t_{n+k-r}^* - t_{n+k-m}^* = (t_{n+k} - rh^*) - (t_{n+k} - mh^*) = mh^* - rh^* \end{cases} \quad (6.76)$$

Eta (6.76) adierazpenak (6.75) adierazpenean ordezkatzuz, hauxe daukagu:

$$\sum_{j=1}^k \nabla^j y_{n+k} R_{jr} = \sum_{j=1}^k \nabla^j y_{n+k}^* U_{jr} \quad (6.77)$$

U eta R matrizeak $(k \times k)$ ordenako matrizeak izanik, eta beraien elementuak honela emanak egonik:

$$R_{jr} = \frac{1}{j!} \prod_{m=0}^{j-1} \left(m - r \frac{h^*}{h} \right) \quad (6.78)$$

$$U_{jr} = \frac{1}{j!} \prod_{m=0}^{j-1} (m - r)$$

(6.77) adierazpena era matritzialean honela idatziko dugu:

$$\sum_{j=1}^k \nabla^j y_{n+k} R_{jr} = \sum_{j=1}^k \nabla^j y_{n+k}^* U_{jr} \Rightarrow D \cdot R = D^* \cdot U \Rightarrow D^* = D \cdot R \cdot U^{-1} \quad (6.79)$$

(6.78) adierazpenean U matrizeari dagozkion eragiketak egiten baditugu, erraz iritsiko gara $k = 5$ ordenako U matrizerara:

$$U = \begin{pmatrix} -1 & -2 & -3 & -4 & -5 \\ 0 & 1 & 3 & 6 & 10 \\ 0 & 0 & -1 & -4 & -10 \\ 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \quad (6.80)$$

Shampine eta Reichelt-ek (1997) $U^{-1} = U$ dela frogatzen dute. Beraz, (6.79) adierazpena honela geratzen da:

$$D^* = D \cdot R \cdot U \quad (6.81)$$

Eta azken hau da, hain zuzen, h pauso-tamainako atzerakako diferentziak jasota dauden D matritzetik h^* pauso-tamainak darabiltzan D^* matrizerara igarotzea ahalbidetzen duen adierazpena.

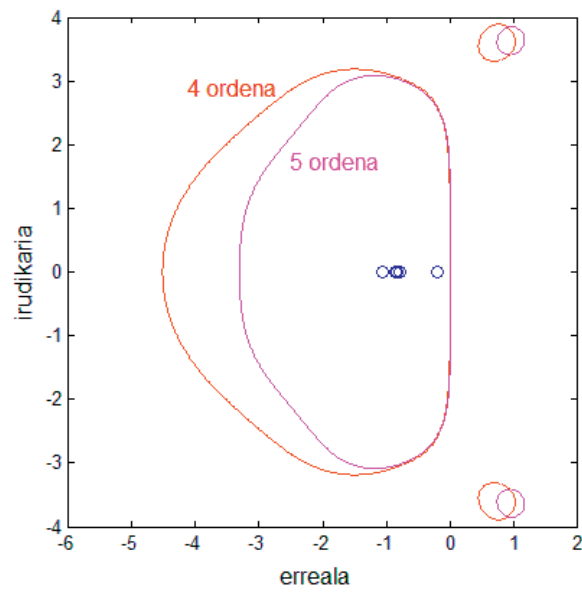
6.3. ODE45 ETA ODE15S APLIKATUZ ZENBAIT ADIBIDE

Honako hasierako baliodun EDA hau kontsideratuko dugu:

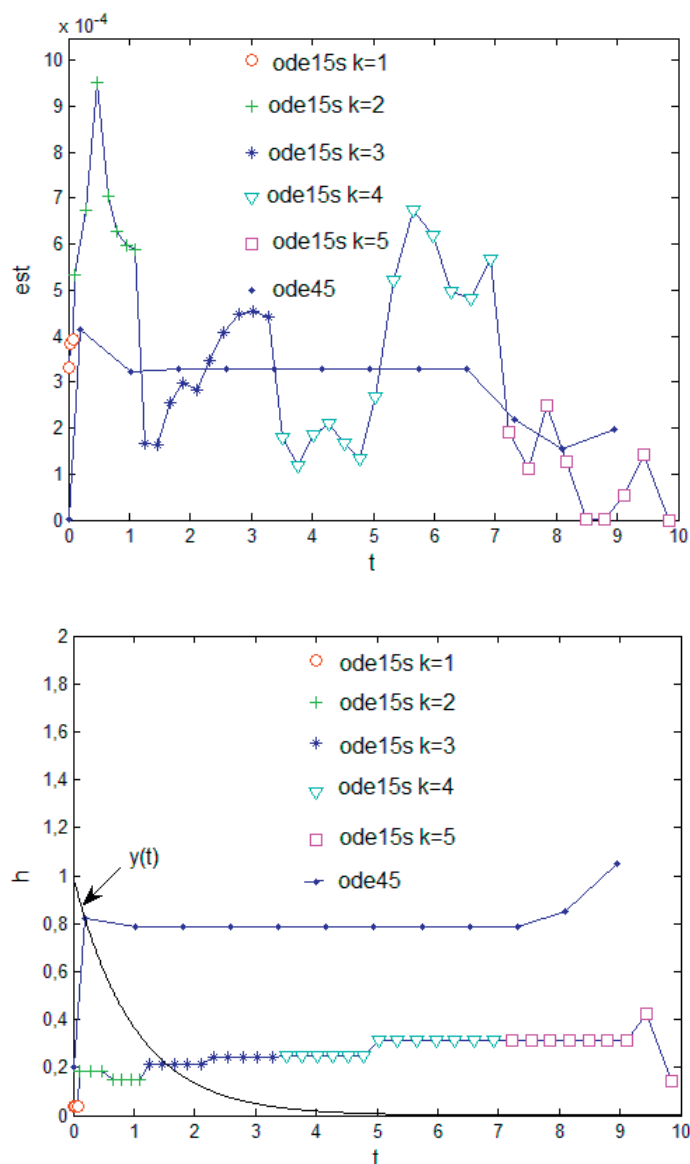
$$y' = \lambda \cdot y, \quad y(0) = 1, \quad \lambda < 0 \quad (6.82)$$

Problema honetako autobalioa λ da. Ode45 eta ode15s funtzioak erabilita, $T = [0, 10]$ denbora-tartean aurkitu dugu problema honen soluzioa. Ode15s funtzioaren kasuan, NDFak erabili dira. λ autobalioaren bi baliotarako aurkitu dugu problema honen soluzioa. Alde batetik, $\lambda = -1$ baliorako, zeinean ode45 funtzioak 13 pauso eman dituen eta ode15s funtzioak 42 pauso. Eta bestetik, $\lambda = -100$ baliorako, ode45 funtzioak 314 pauso behar izan ditu kasu honetan eta ode15s funtzioak 80 pauso. Autobalio bakoitzerako jakin nahi izan dugu funtzio bakoitzak eman duen pauso bakoitzaren tamaina zer izan den. Ode15s funtzioari dagokionez ordena aldakorra darabilenez, pauso bakoitza zein ordenatan eman duen ere jakin nahi izan dugu. Ode45 funtzioaren kasuan, autobalioaren eta pauso-tamainaren arteko biderkadura λh , egonkortasun-eremuk zer lekutan erori den ere jakin nahi izan dugu. Ode15s funtzioaren kasuan azken hori ez dugu irudikatu, BDF eta NDFen egonkortasun-eremuen parte baita $\mathbb{R}^-$ osoa.

$\lambda = -1$ kasuan, λh balio guztiak DOPRI(5,4) metodoko bosgarren ordenari dagokion egonkortasun-eremuan erortzen direla ikus daiteke 6.1. irudian (ode45 funtzioak 5. ordenan askatzen duela kontuan izan). 6.2. irudian ode45 eta ode15s funtzioek ematen duten pauso bakoitzean errorearen estimazioa zein den (goiko irudia) eta pauso guztietako tamainak (beheko irudia) irudikatu dira. Ode45 funtzioaren kasuan errorearen estimazioak eta, ondorioz, pauso-tamainak, oso erregularrak direla ikus daiteke. Gainera, errorearen estimazioa jaisten denean, hurrengo pauso-tamaina handitu egiten da eta alderantziz. Ode15s funtzioaren kasuan errorearen estimazioaren txikitzeak beti ez du pauso-tamaina handitzea eragiten, pauso bakoitzaren ostean ez baita pauso-tamaina aldatzen (kodea azaldu dugunean ere azaldu da hau). Era berean, kontuan izan behar da funtzio batean eta bestean ez direla berdinak pauso-tamainen definizioak. Adibidez, ode15s funtzioko pauso batzuetan errorearen estimazioa ode45 funtziokoa baino txikiagoa izan da, nahiz eta ode15s funtzioak ematen dituen pausoen tamainak ode45 funtzioak ematen dituenak baino txikiagoak izan.



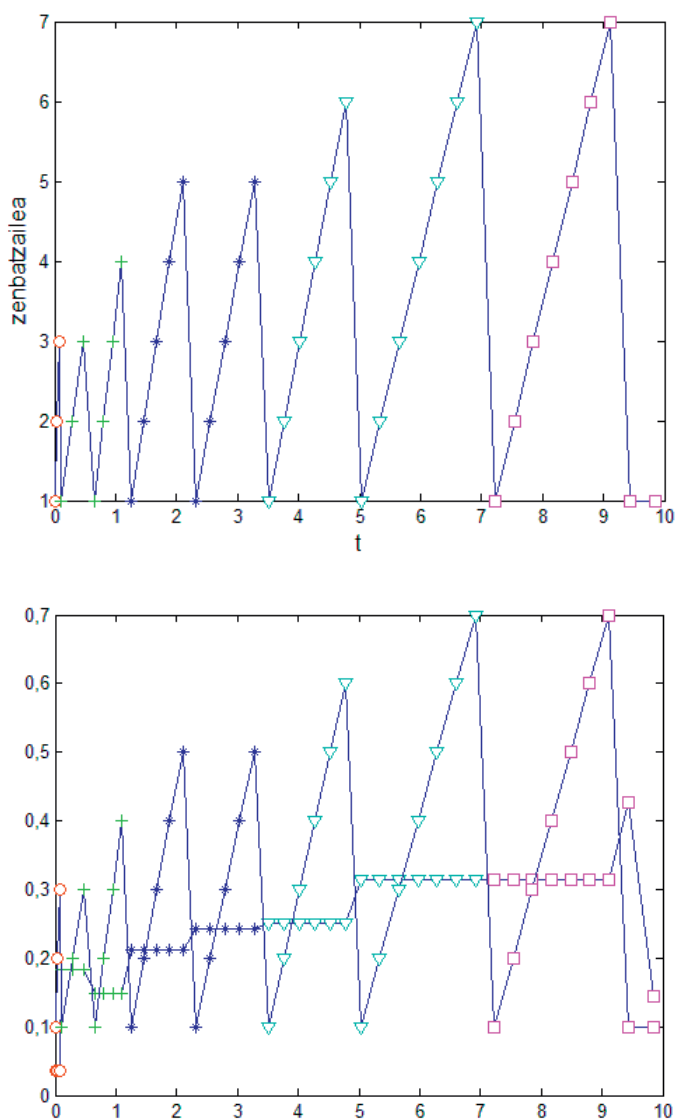
6.1. irudia. (6.82) problemako λh balioen kokapena DOPRI(5,4) metodoko bosgarren ordenako egonkortasun-eremuan, $\lambda = -1$ denean.



6.2. irudia. (6.82) problemen ode45 eta ode15s funtzioen errorearen estimazioak (goian) eta pauso-tamainak (behean), $\lambda = -1$ denean.

Pauso on bat eman ostean eta k ordenan ari garela lanean suposatuz, ode15s funtzioak pauso-tamaina edota ordena-aldaketa planteatzen dezan, ordena eta pauso-tamaina berean gutxienez $(k+2)$ pauso eman izana bete behar da. 6.3. irudiko goiko aldeko grafikoen pauso bakoitza bukatu ostean ordena eta pauso-tamaina berarekin emandako pauso kopurua kontatzen duen zenbatzaileak izan duen balioa

irudikatu dugu. 6.3. irudiko azpiko aldeko grafikoan, ode15s funtzioak eman dituen pauso-tamainak eta zenbatzailearen balioa 10^{-1} balioaz biderkatuta gainjarri ditugu. Horrela, grafiko berean jaso nahi izan ditugu gauza biak (pausoen tamaina eta *zenbatzailea*). Irudi honetan argi ikusten da zenbatzailea ($k+2$) baliora iristen ez den bitartean, ez dela ordenarik ezta pauso-tamainarik ere aldatzen.

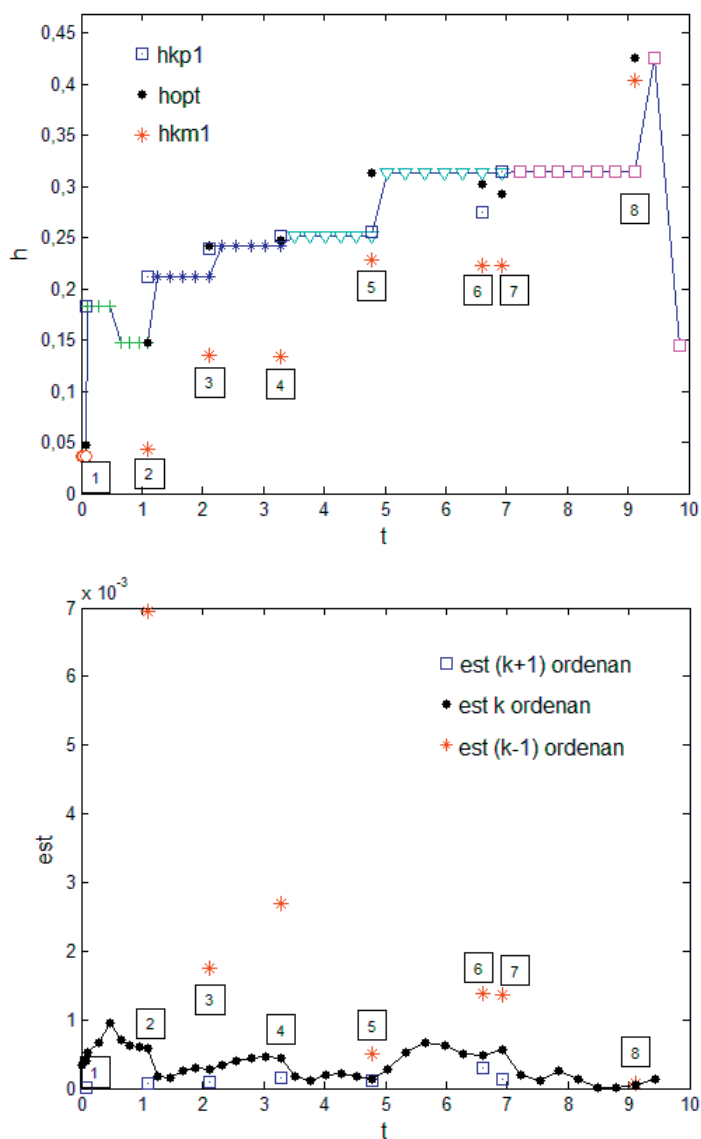


6.3. irudia. Ode15s funtzioko zenbatzailearen funtzionamendua.
(6.82) probleman, $\lambda = -1$ denean.

Behin zenbatzaileak ezarritako balio minimo hori gainditzen duenean, ode15s funtzioak ordena edota pauso-tamainaren aldaketa planteatzen du. Hori gertatzen da 6.4. irudian 1-8 zenbakiak erabilia markatu ditugun puntuetan. Ode15s funtzioak k ordenan eta h pauso-tamainaz lanean diharduela suposatuz, ordena berari legokiokeen hurrengo pauso-tamaina kalkulatzeko du: h_{opt} . Gainera, $k > 1$ bada, azken pauso hori $(k-1)$ ordenaz eman izan balitz izango genukeen errorearen estimazioa kalkulatzeko du eta hurrengo pausoa $(k-1)$ ordenan emango bagenu zer pauso-tamainarekin emango genukeen kalkulatzeko du: h_{kml} . Lanean ari garen k ordena, ode15s funtzioak lan egin dezakeen ordena maximoa baino txikiagoa denean, azken pausoa $(k+1)$ ordenaz eman izan balitz edukiko genukeen errorearen estimazioa eta hurrengo pauso-tamaina ere $(k+1)$ ordenan emango balitz izango lukeen balioa kalkulatzeko dira: h_{kpl} . Aipatu ditugun balioak eskura dituenean, ode15s funtzioa h_{kml} eta h_{opt} konparatzen hasten da. $h_{kml} > h_{opt}$ betez gero, h_{opt} aldagaiaren barruan h_{kml} balioa gordeko du, eta pauso-tamaina honi dagokion ordena hartuko du $(k-1)$. Ondoren, h_{kpl} eta h_{opt} konparatzen ditu eta $h_{kpl} > h_{opt}$ betez gero, h_{opt} aldagaian h_{kpl} gordeko du, eta gordeta duen ordenaren balioari unitate bat gehituko dio. Konparaketa bi hauetatik eratortzen den pauso-tamaina orain artekoa baino handiagoa gertatu behar da pauso-tamaina eta ordenaren aldaketa emateko, bestela, orain arteko ordena eta pauso-tamaina berarekin jarraituko du lanean funtzioak.

6.4. irudiko beheko grafikoan ode15s funtzioak pauso edota ordena-aldaketa planteatu duen bakoitzean, $(k-1)$, k eta $(k+1)$ balioei dagokien errorearen estimazioa irudikatu dugu. Normala den bezala, ordena altuagoei dagokien errorearen estimazioa txikiagoa da kasu guztietan. Hau da, $(k+1)$ ordenari dagozkion errorearen estimazioak dira hiruretatik txikienak. 6.4. irudiko goiko aldean pauso-tamainak irudikatu dira eta azpian adierazten dugu ode15s funtzioak 1-8 zenbakiekin markatu ditugun puntuetan nola egin duen pauso-aukeraketa:

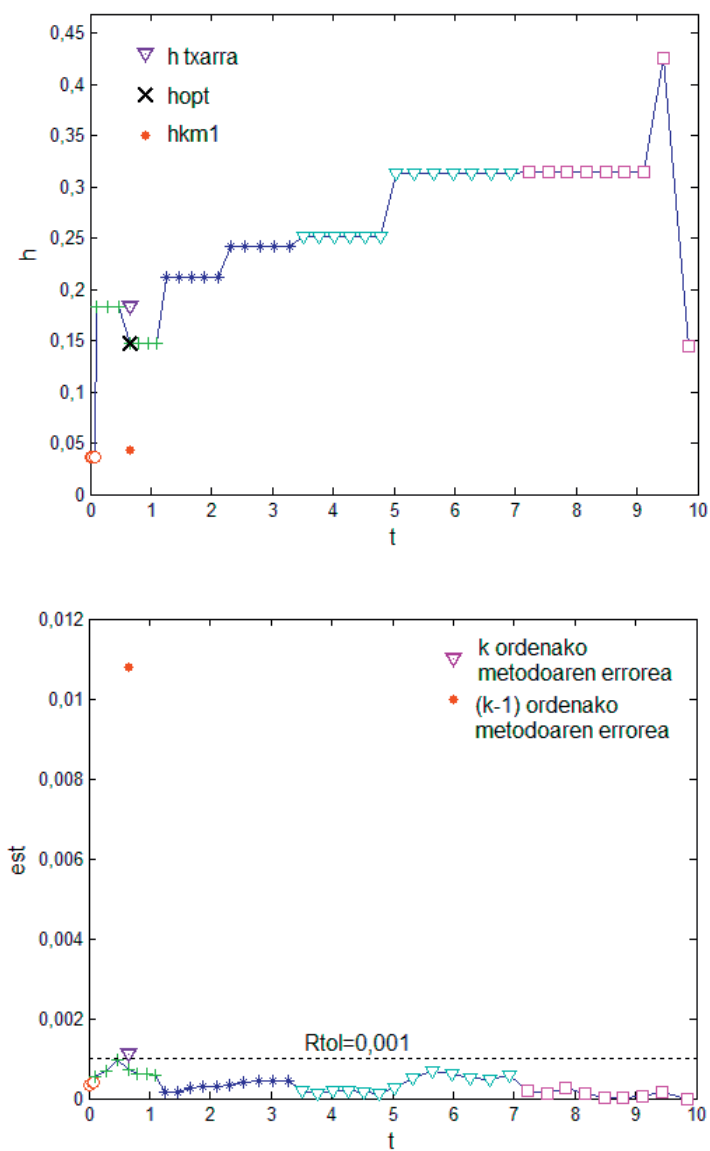
- 1 puntuan $k=1$ ordenan ari gara lanean. Bakarrik kalkulatu dira $(k+1)=2$ ordenari dagozkion errorearen estimazioa eta pauso-tamaina. Ordena honi dagokion pauso-tamaina k ordenari dagokiona baino handiagoa denez, hau da, $h_{kpl} > h_{opt}$ gertatzen denez, eta gainera, orain arteko pauso-tamaina baino handiagoa denez ere, hurrengo pauso-tamaina h_{kpl} izango da eta $k=2$ ordenaz emango da.
- 2, 4 eta 7 puntuetan $h_{kml} > h_{opt}$ ez da betetzen. Baina $h_{kpl} > h_{opt}$ bai betetzen da. Kasu honetan ere h_{kpl} orain arteko pauso-tamaina baino handiagoa denez, hurrengo pauso-tamaina h_{kpl} izango da eta $(k+1)$ ordenaz emango da. Hau da, ordenaren handitzea eta pauso-tamainaren handitzea gertatzen dira.
- 3 eta 5 puntuetan ez da betetzen $h_{kml} > h_{opt}$, ezta $h_{kpl} > h_{opt}$ ere. Aldiz, $h_{opt} > h_{kml}$ bai betetzen denez, pauso-tamaina berria h_{opt} izango da, orain artekoa baino handiagoa dena, baina orain arteko ordena berarekin jarraituko da lanean.



6.4. irudia. Ode15s funtzioan (6.82) problemarako pauso on baten osteko hurrengo pausoa (goian) eta errorearen estimazioa (behean), $\lambda = -1$ denean.

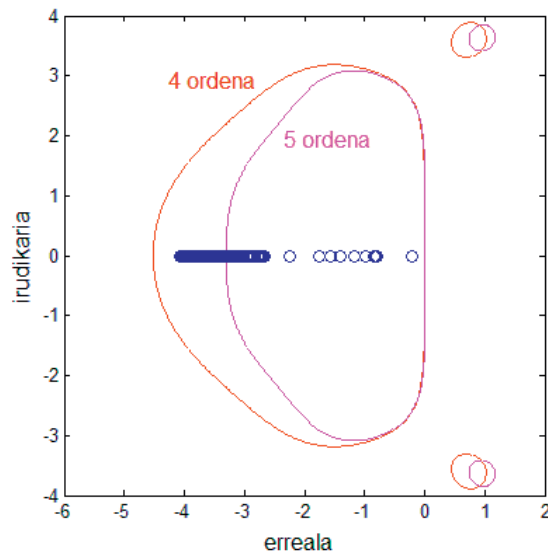
- 6 puntuan ez da betetzen $h_{kp1} > h_{opt}$, ezta $h_{opt} > h$ ere. Beraz, orain arteko ordenaz eta pauso-tamainaz jarraitzen da lanean.
- Problema hau askatzean, ode15s funtzioari ez diogu jarri ordena maximorako inolako mugarik. Kontrakorik esan ezik, ode15s funtzioak lan egin dezakeen ordenarik altuena $k=5$ da, eta 8 puntuan ordena maximo horretara iritsita dago. Hori dela-eta, ezin da h_{kp1} kalkulatu. Ez da betetzen $h_{km1} > h_{opt}$, baina bai betetzen da, ordea, $h_{opt} > h$. Beraz, kasu honetan pauso-tamaina berria h_{opt} izango da, baina orain arteko ordena berarekin jarraituko da lanean.

Ode15s funtzioak pauso txar baten ostean egin duen aukeraketa zein izan den ere aztertu nahi izan dugu. Pauso txar baten osteko pauso-tamainaren aukeraketa 6.5. irudian jaso dugu. (6.82) problema askatzeko prozesuan bakarrik pauso txar bat egon da. $Rtol=10^{-3}$ balioaz lanean egon gara eta errorearen estimazioak $Rtol=10^{-3}$ balioa gaitu du: $\|est\| > Rtol$ gertatu da. Pauso txar baten ostean ode15s funtzioak k eta $(k-1)$ ordenei dagozkien pauso-tamainak kalkulatu ditu, h_{opt} eta h_{km1} hurrenez hurren. Baldin $h_{km1} > h_{opt}$ gertatuko balitz, hurrengo pausoa $(k-1)$ ordenan emango litzateke eta oraingo pausoaren eta h_{km1} balioaren arteko minimoa erabilita emango litzateke. Gure kasuan, ez da betetzen $h_{km1} > h_{opt}$, beraz, h_{opt} pauso-tamainakoa izango da hurrengo pausoa, eta k ordenan jarraituko da lanean.

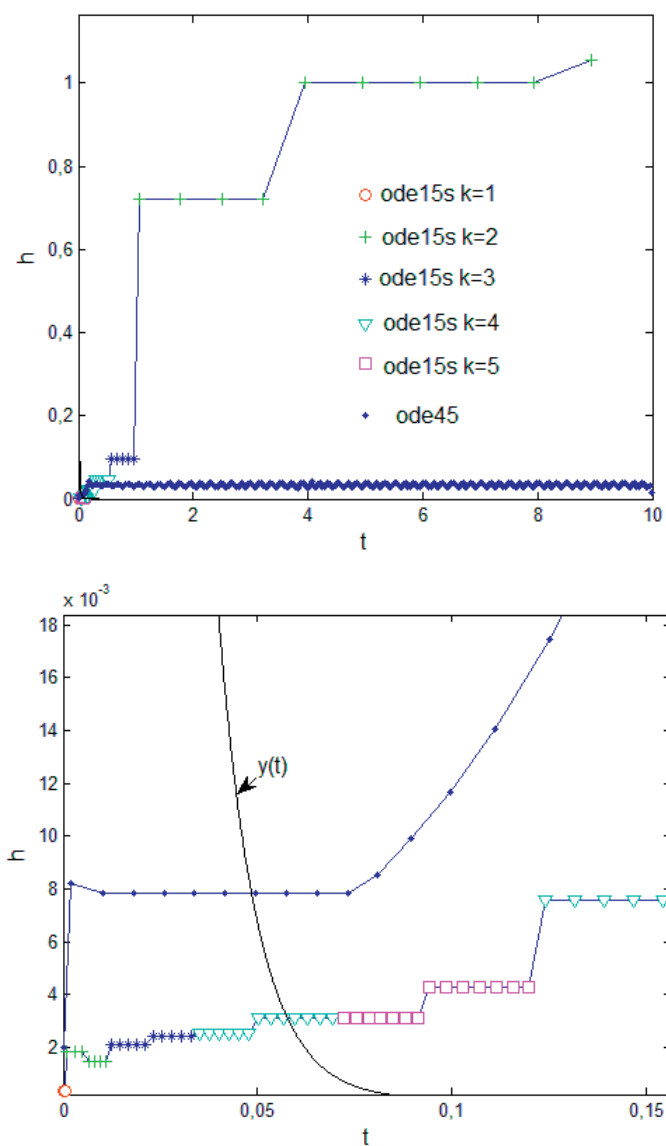


6.5. irudia. Ode15s funtzioan (6.82) problemarako pauso txar baten osteko hurrengo pausoa (goian) eta errorearen estimazioa (behean), $\lambda = -1$ denean.

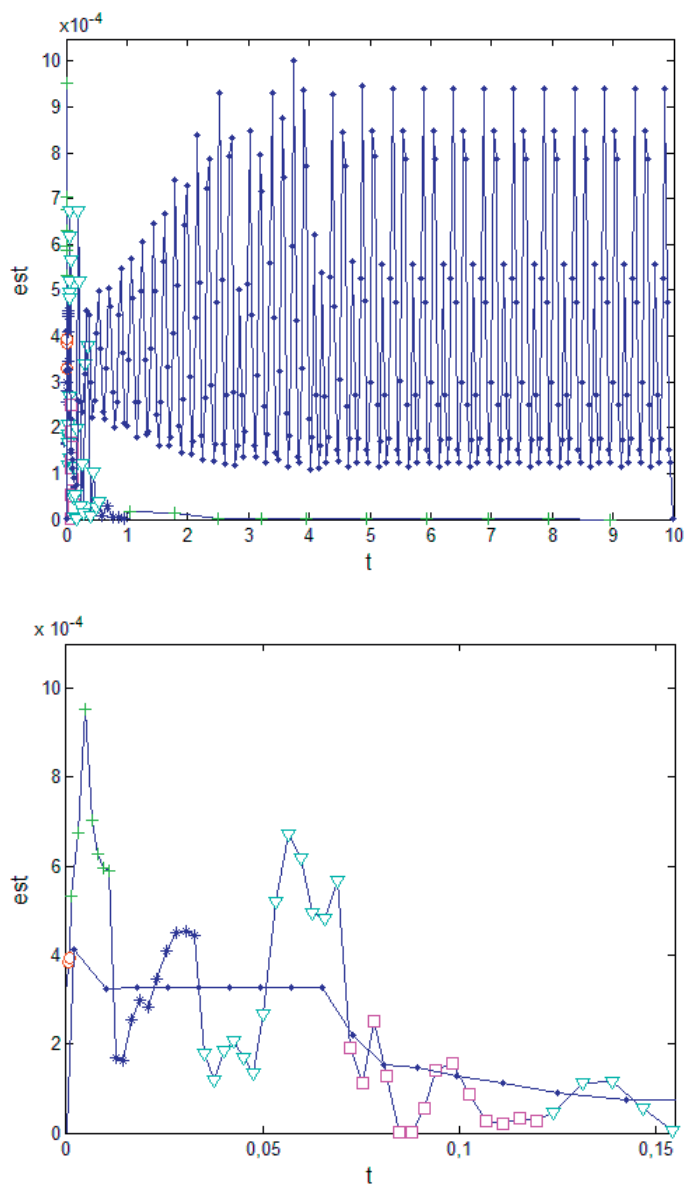
$\lambda = -100$ kasuan eta ode45 funtzioaren kasuan, λh balio batzuk DOPRI(5,4) metodoko bosgarren ordenako egonkortasun-eremuaren kanpoan erortzen dira, ikus 6.6. irudia. Hori gertatzea ez da arraroa, kode biek kontrolatzen dutena errorearen estimazio bat baino ez baita. Hainbat pausori dagozkion λh balioak egonkortasun-eremuan erori ez arren, emaitza ona ematen digu ode45 funtzioak. Pauso-tamainei erreparatzen badiegu, berriz ere ode45 funtzioak ematen dituen pauso-tamainak oso erregularrak dira eta errorearen estimazioarekin lotura zuzena daukate. Ode15s funtzioaren kasuan, denboran aurrera egin ahala errorearen estimazioa asko jaitsi da eta horrek pauso-tamainak handitzea eragin du. Funtzio biek darabiltzaten pauso-tamainak 6.7. irudian adierazi dira eta errorearen estimazioak 6.8. irudian.



6.6. irudia. (6.82) problemako λh balioen kokapena DOPRI(5,4) metodoko bosgarren ordenako egonkortasun-eremuan, $\lambda = -100$ denean.



6.7. irudia. (6.82) probleman ode45 eta ode15s funtzioen pauso-tamainak, $\lambda = -100$ denean (beheko irudian zoom-a aplikatu da).



6.8. irudia. (6.82) problemen ode45 eta ode15s funtzioen errorearen estimazioak, estimazioak, $\lambda = -100$ denetan (azpiko irudian zoom-a aplikatu da).

7. Zenbait metodoren programazioa MATLABen

MATLABek EDAk askatzeko zenbait funtzio eskaintzen ditu. Funtzio horietako batzuk dira 6. kapituluaren ikusi ditugun `ode23`, `ode45` eta `ode15s` funtzioak. Baina MATLABen EDAk askatzeko zenbakizko metodoak guk geuk ere programa ditzakegu eta, ondoren, `ode23`, `ode45`, `ode15s`, etab. erabiltzen ditugun bezala, geuk programatutako funtzioak erabilia ere ebatzi ahal izango ditugu EDAk.

Liburu honetan barrena askotariko metodoak ikusi ditugu: pauso bakarrekoak, pauso anitzekoak, pauso bakarrekoak izanik etapa anitz dituztenak, bakarrik lehen deribatua darabiltenak, bigarren deribatua darabiltenak, super-etorkizuneko puntuak darabiltzatenak, etab. Aztertu ditugun metodoak bi multzotan banatzeko esango baligute, bi multzo horiek hauek izan litezke: zenbakizko metodo esplizituak eta zenbakizko metodo implizituak. Esplizituetan, aurkitu nahi dugun balioa kalkulatzeko beharrezko informazio guztia daukagu eta eragiketak eginez emaitza aurki daiteke. Implizituetan, ezezaguna askatzeko darabilgun ekuazioan, ezezagunaz gain ezezagunaren deribatua ere agertuko da, eta kasu horretan, ezezaguna aurkitzeko metodo iteratiboak erabili beharko ditugu. Baina, zenbakizko metodo implizitu bat EDA lineal bat askatzeko darabilgunean, litekeena da metodo iteratiboen erabilera ekiditea, zeren askatu behar dugun ekuazioan eragiketak eginez posible baita ezezaguna askatzea.

Kapitulu honetan, zenbait zenbakizko metodo implizitu EDA linealetara egokitu ditugu, eta metodo horien programazioa MATLABen egin dugu. Ondoren, edozein eratako EDAk askatzeko zenbakizko metodoen programaziora pasatu gara. Atal hau zenbakizko metodo esplizituen programazioarekin hasi dugu eta, ondoren, implizituen programazioarekin bukatu. Implizituen programazioan Newton Raphson-en metodoa erabili dugu metodo iteratibo bezala.

7.1. EDA LINEALETARAKO ZENBAIT METODO INPLIZITUREN APLIKAZIOA ETA PROGRAMAZIOA

Ekuazio diferentzial arrunten sistema bat lineala da, ondorengo forma baldin badu:

$$y' = A(t)y + g(t) \quad (7.1)$$

Dimentsio bakarreko problema linealaren kasuan, A eskalar bat izango da, eta s dimentsiodun problemaren kasuan, aldiz, $s \times s$ ordenako matrizea.

(7.1) EDA homogeneoa da, $g(t) = 0$ betetzen bada; bestela, ez-homogeneoa izango da. A matrizeko koefizienteak denboraren mendekotasunik ez badute, (7.1) koefiziente konstanteko EDA lineala izango da eta hauexek dira kapitulu honetan askatuko ditugunak.

MATLABen (7.1) motako EDA nola definitu dezakegun ikusiko dugu. Bi kasutarako ikusiko dugu: bata, EDA dimentsio bakarrekoa den kasurako, eta bestea, EDAn bat dimentsioa baino handiagoa duen kasu baterako.

Adibideak

1. (7.1) forma duen dimentsio bakarreko problemaren adibide bat hauxe da:

$$y' = -y \quad (7.2)$$

(7.2) problemaren kasuan, (7.1) ekuazioko $A = -1$ da eta $g(t) = 0$. Kasu honetan EDA lineal eta homogeneoa daukagu, $g(t) = 0$ betetzen baita. MATLAB erabilita, honela definituko genuke problema honi dagokion funtzioa:

```
function [A,g]=p1lin(t)

%%Dimentsio bakarreko problemaren kasua:
A=-1;
g=0;
```

Goiko artxiboa 'p1lin' izenez eta .m luzapenez gordeko genuke.

2. (7.1) problemaren bi dimentsioko adibide bat hauxe litzateke:

$$y' = \begin{pmatrix} -4 & -3 \\ 2 & -1 \end{pmatrix} \cdot y + \begin{pmatrix} 5 \cos(t) - \frac{1}{2} \sin(t) \\ -3 \cos(t) + \frac{1}{3} \sin(t) \end{pmatrix} \quad (7.3)$$

Kasu honetan $A = \begin{pmatrix} -4 & -3 \\ 2 & -1 \end{pmatrix}$ eta $g(t) = \begin{pmatrix} 5 \cos(t) - \frac{1}{2} \sin(t) \\ -3 \cos(t) + \frac{1}{3} \sin(t) \end{pmatrix}$ dira. MATLABen honelako funtzio bat erabiliz definitu daiteke (7.3) EDA sistema.


```
function [A,g]=p2lin(t)

%%Bi dimentsioko problema baten kasua:
A=[-4 -3 ; 2 -1];
g(1)=5*cos(t)-(1/2)*sin(t);
g(2)=-3*cos(t)+(1/3)*sin(t);
g=g'; %g matrizea zutabe matrize bezala jartzen dugu.
```

Kasu honetan 'p2lin.m' izenaz gordeko genuke goiko funtzioa.

Ikusi ditugun adibide bietan, 'p1lin.m' eta 'p2lin.m' funtzioek sarrerako argumentu bakarra dute, t bezala adierazi duguna, eta irteerako argumentu bina dituzte: A matrizea eta g funtzioa. Ikusi duguna izan daiteke (7.1) motako EDA linealak MATLABen definitzeko era bat.

Behin askatu beharreko EDA lineala MATLABen definitu dugunez, EDA lineal bat askatzean zenbait metodo inplizituk izango luketen forma zein den ikusiko dugu bai eta metodo horiek MATLABen nola programatu daitezkeen ere. Euler inplizituaren metodotik hasiko gara.

7.1.1. EDA linealetarako Euler inplizitua

Hasierako baliodun $y' = f(t, y)$ EDA askatzeko Euler-en metodo inplizitua honela emana datorrela ikusi dugu:

$$y_{n+1} = y_n + hf_{n+1}$$

non $f_{n+1} = f(t_{n+1}, y_{n+1})$.

Askatzen ari garen EDA koefiziente konstantekoa eta lineala denean, Euler-en metodo inplizituak forma berezi hau hartzen du:

$$y_{n+1} = y_n + h(Ay_{n+1} + g(t_{n+1})) \quad (7.4)$$

Eta eragiketak eginez, posible da (7.4) ekuaziotik y_{n+1} terminoa askatzea, ondoko eran:

$$(I - hA)y_{n+1} = y_n + hg(t_{n+1}) \Rightarrow y_{n+1} = (I - hA)^{-1}(y_n + hg(t_{n+1})) \quad (7.5)$$

I matrizea problemaren dimentsio bera duen identitate matrizea izanik. EDAk forma lineala duenean, Euler-en metodo inplizituak (7.5) forma hartzen du.

MATLABen metodo honi dagokion programa hiru funtzio erabilia egin dugu: ‘eulerinplin.m’ izeneko funtzioa definitu dugu, ‘hasi.m’ eta ‘paseulerinplin.m’ izenekoak. Hiru horietatik lehenengoa izango da programaren egitura osoa jasotzen duen funtzioa, bigarrenak (‘hasi.m’k) hasierako parametro batzuen kalkulua ahalbidetuko du —lehenengo kapituluaren ere ikusi dugu funtzio hau, baina berriz errebisatuko dugu— eta hirugarrena, ‘paseulerinplin.m’ izenekoak, Euler-en metodo implizituko pausoa EDA linealetara egokitua izango da.

‘eulerinplin’ funtzioaren sarrerako argumentuak edo datuak hauek izango dira:

- Askatu nahi dugun ekuazio diferentzialaren 2. parte definitzen den funtzioa. Oro har ‘fytlin’ deituko duguna.
- Problema askatu nahi dugun denbora-tartea $[t_0 \ t_1]$, ‘tartea’ deituko duguna.
- Hasierako baldintza jasotzen duen y_0 errenkada matrizea $[y_1 \ y_2 \ \dots \ y_s]$, ‘y0’ deituko duguna, s EDAREN dimentsioa izanik.
- Problema askatzeko emango dugun pauso kopurua, ‘m’ deituko duguna. Beste era batean esanda, $[t_0 \ t_1]$ tartea zenbat zatitan zatituko dugun esaten digun zenbakia da pauso kopurua.

‘eulerinplin’ funtzioaren irteerako datuak bi izango dira:

- $[t_0 \ t_1]$ tartea diskretizatu dugun t puntuak jasotzen dituen eta $(m+1)$ errenkada dituen ‘tsol’ zutabe matrizea.
- ‘tsol’ aldiuneetan y aldagaiak hartzen dituen balioak jasota daudeneko $((m+1) \times s)$ dimentsioko ‘ysol’ matrizea. ‘ysol’ matrizearen errenkada bakoitzean $tsol(i)$ aldiuneko y soluzioaren s osagaiak egongo dira jasota, eta haren zutabe bakoitzean, soluzioaren j osagaiak $(m+1)$ aldiuneetan hartzen dituen balioak egongo dira jasota.

```
function [tsol,ysol]=eulerinplin(fytlin,tartea,y0,m)
%"s" ekuazio diferentzial arrunteko y'=Ay+g(t) sistema askatzen
%duen funtzioa da, Euler implizitua EDA linealetara egokituta
%erabiliz.
%Sarrerako datuak:
%>fytlin: Ekuazio diferentzialeko 2. parte definitzen den
%funtzioaren izena jarri behar da hemen. Funtzio hau haxe izan
%daiteke:
```

```

function [A,g]=p1lin(t)
%Bi dimentsiodun problema:
%      A=[-4 -3 ; 2 -1];
%      g(1)=5*cos(t)-(1/2)*sin(t);
%      g(2)=-3*cos(t)+(1/3)*sin(t);
%      g=g';
%%Bat dimentsiodun problema:
% A=-1;
% g=0;
%>tartea: Sistema askatuko dugun denbora tartea da: [t0 t1].
%>y0: Hasierako balioaren s osagaiak jasotzen diren errenkada
%matrizea da: [y1 y2 ... ys].
%>m: Metodoa erabilita emango dugun pauso kopurua da. Beste era
%batean esanda: [t0 t1] tartea zenbat zatitan zatituko dugun
%esaten digun zenbakia da.

%Irteerako datuak:
%>tsol: (m+1) dimentsioko zutabe matrizea da. Bertan, [t0 t1]
%tartea diskretizatu dugun (t) puntuak daude jasota.
%>ysol: ((m+1) x s) dimentsioko matrizea da, "s" sistemako
%ekuzazio kopurua izanik. Errenkada bakoitzean (i=1:m+1): tsol(i)
%aldiuneko soluzioaren "s" osagaiak daude jasota.
%Zutabe bakoitzean (j=1:s): Soluzioaren (j) osagaiak (m+1)
%aldiuneetan hartzen dituen balioak daude jasota.
%-----
s=length(y0);
I=eye(s);      %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);
t=tsol(1);
y=ysol(1,:);

%A matrizearen balioa lortzeko egiten dugu ondorengo ebaluaketa.
%Momentuz, ez dugu "g"-ren balioa erabiliko.

[A,g]=feval(ftylin,t);

%%Pausoa emateko behar dugun matrizea:
C1=(I-A*h)\I;    %Alderantzizkoa kalkulatzeko dugu.
%% Euler inplizituko pausoak EDA lineala askatzerakoan
for j1=2:m+1

    [t,y]=pasoeulerinplin(ftylin,t,y,h,C1);

    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin ysol eguneratzen dugu.

end

```

‘eulerinplin.m’ funtzioak ‘hasi.m’ funtzioari deitzen dio une batean. Funtzio horrek, ‘hasi.m’ funtzioak, programan beharko ditugun zenbait parametro kalkulatzeko ditu. Bere sarrerako datuak dira: denbora-tartea —‘tarte’ deitu duguna—, hasierako balioa —‘y0’ deitu duguna— eta pauso kopurua, ‘m’. Hasierako datu horiek erabilita, pauso-tamaina kalkulatu du —‘h’-z adieraziko duguna—, eta hasierako ‘y0’ bektorearen dimentsioa kalkulatu duenez, ‘tsol’ eta ‘ysol’ matrizeei hasiera emango die berauek zeroekin betetz eta, ondoren, hasierako aldiuneko denbora eta y -ren balioak matrize horietako bakoitzaren lehenengo errenkadetan barneratuz. Horrela, ‘tsol’ eta ‘ysol’ matrizeak prest izango ditugu beste pausoak eman ahala betetzen joateko.

```
function [h,tsol,ysol]=hasi(tarte,y0,m)
%
% Ekuazio diferentzial arruntaren sistema bat integratzeko hasierako
% funtzioa:

t0=tarte(1);
t1=tarte(2);
h=(t1-t0)/m;      % Pausoaren tamaina.
s=length(y0);     % y0 bektorearen dimentsioa.
% tsol zutabe matrizea zeroekin betetzen da hasieran.
tsol=zeros(m+1,1);
% ysol izeneko soluzio matrizea zeroekin betetzen da hasieran.
ysol=zeros(m+1,s);

tsol(1)=t0;
% ysol-eko 1. errenkadan sartzen da hasierako y0 baldintza.
ysol(1,:)=y0';
```

‘eulerinplin.m’ funtzioak ‘paseulerinplin.m’ funtzioari ere deitzen dio. Haren sarrerako datuak dira: ebatziko dugun problema definituta dagoen 2. parte —‘fytlin’ izendatu duguna—, aurreko pausoko aldiunea t_n —programazioan ‘tn’ izendatu dugu—, aurreko pausoko y aldagaiak hartu duen balioa y_n —programazioan ‘yn’ deitu diogu—, pauso-tamaina h eta $C_1 = (I - hA)^{-1}$ matrizea, ‘C1’ izendatu duguna. Irteerako datuak izango dira: y aldagaiaren balioa kalkulatu nahi dugun aldiunea t_{n+1} —programazioan ‘tn1’ izendatu duguna— eta aldiune horretako y aldagaiaren balioa y_{n+1} —programazioan ‘yn1’ izendatu duguna—. Funtzio honetan honela egin ditugu (7.5) adierazpeneko eragiketak:

- $g(t_{n+1})$ balioa kalkulatu dugu.

- Termino independentea TI aldagaian gorde dugu, ondorengo: $TI = y_n$.
- $g(t_{n+1})$ dagoen kasuan, aurreko termino independenteari $hg(t_{n+1})$ terminoa gehitu diogu, eta batuketa berri hori berriz ere TI aldagaian gorde dugu: $TI = y_n + hg(t_{n+1})$.
- Bukatzeko, $C_1 = (I - hA)^{-1}$ matrizeaz biderkatu dugu aurreko termino independentea, eta (7.5) adierazpeneko $y_{n+1} = C_1 \cdot TI$ balioa lortu.

```
function [tn1,yn1]=pasoEulerinpln(ftylin,tn,yn,h,C1);
%
%"s" ekuazio diferentzial arrunteko y'=Ay+g(t) sistema
%askatzeko Euler inplizituaren forma linealeko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:

tn1 = tn+h;

%A eta g-ren ebaluazioa tn1 aldiunean:

[A,gn1]=feval(ftylin,tn1);

%Euler inplizituko pausoa:

TI=yn;
if ~isempty(gn)
    TI=TI+gn1*h;
end
yn1=C1*TI;
```

Adibidea

(7.3) adierazpena duen EDA lineala kontsideratuko dugu. EDA lineal hau 'p2lin.m' bezala gorde dugu lehen. Adibidez, EDA lineal honetan $(y(0), y'(0)) = (1, 1)$ hasierako balioa kontsideratuko dugu:

$$y' = \begin{pmatrix} -4 & -3 \\ 2 & -1 \end{pmatrix} \cdot y + \begin{pmatrix} 5 \cos(t) - \frac{1}{2} \sin(t) \\ -3 \cos(t) + \frac{1}{3} \sin(t) \end{pmatrix}, \quad (y(0), y'(0)) = (1, 1) \quad (7.6)$$

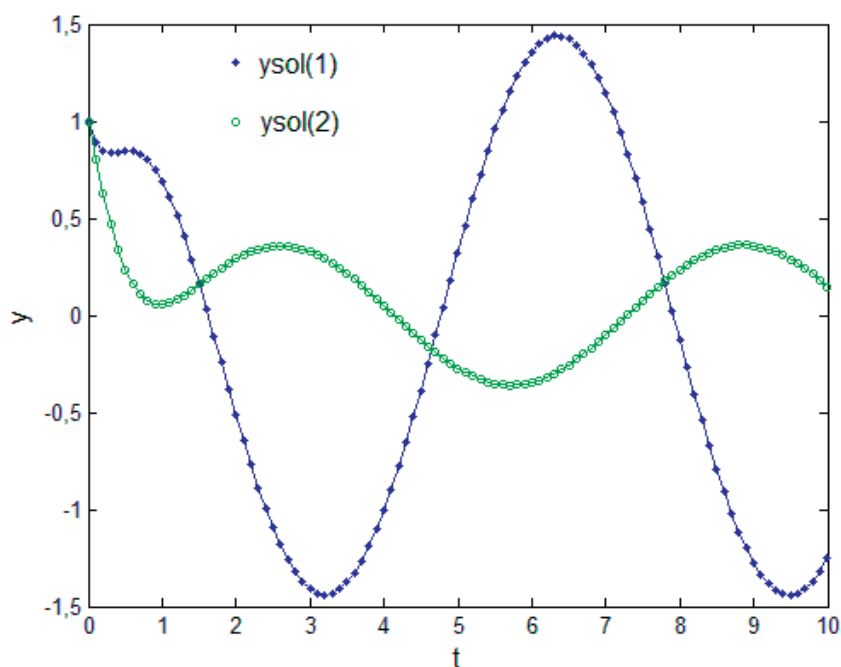
Euler-en metodo inplizitua erabilia EDA lineal honen soluzioa $T = [0, 10]$ denbora-tartean kalkulatu nahi badugu eta problema askatzeko $m = 100$ pauso eman nahi baditugu, MATLABen agindu hau idatzi behar dugu:

```
[tsol,ysol]=eulerinplin('p2lin',[0 10],[1 1],100)
```

Kalkuluak egin ondoren, soluzioa, 'ysol', eta soluzio horiek kalkulatu dituen aldiunea, 'tsol', izango ditugu. Soluzioen grafiko bat egin ahal dugu agindu hau erabilita:

```
plot(tsol,ysol)
```

7.1. irudian marraztu ditugu problema honetarako Euler-en metodo inplizituaz lortu dugun emaitzako osagai biak.



7.1. irudia. EDaren emaitza Euler inplizitua erabilita.

7.1.2. EDA linealetarako metodo trapezoidala

Hasierako baliodun $y' = f(t, y)$ EDA askatzeko metodo trapezoidalari dagokion pausoa hau da:

$$y_{n+1} = y_n + \frac{h}{2}(f_n + f_{n+1}) \quad (7.7)$$

Non $f_n = f(t_n, y_n)$ eta $f_{n+1} = f(t_{n+1}, y_{n+1})$.

Askatzen ari garen EDA koefiziente konstantekoa eta lineala denean, metodo trapezoidalak forma konkretu hau hartzen du:

$$y_{n+1} = y_n + \frac{h}{2} (Ay_n + g(t_n) + Ay_{n+1} + g(t_{n+1})) \quad (7.8)$$

Eta eragiketak eginez, posible da (7.8) ekuaziotik y_{n+1} terminoa askatzea:

$$y_{n+1} = \left(I - \frac{h}{2} A \right)^{-1} \left(\left(I + \frac{h}{2} A \right) y_n + \frac{h}{2} g(t_n) + \frac{h}{2} g(t_{n+1}) \right) \quad (7.9)$$

I matrizea problemaren dimentsio bera duen identitate matrizea izanik.

MATLABen metodo hau hiru funtzio erabilita programatu dugu: 'traplin.m', 'hasi.m' eta 'pasotraplin.m' funtzioak erabilita, hain zuzen. 'traplin.m' funtzioak 'eulerinplin.m' funtzioak izan dituen sarrerako eta irteerako argumentu berak izango ditu eta haxe izango da funtzio nagusia. 'hasi.m' funtzioa aurretik Euler inplizituan erabili dugun berbera da eta 'pasotraplin.m' funtzioa metodo trapezoidaleko pausoa EDA linealetara egokitua da.

```
function [tsol,ysol]=traplin(ftylin,tartea,y0,m)
% "s" ekuazio diferentzial arrunteko y'=Ay+g(t) sistema
%askatzen duen funtzioa da, metodo trapezoidala EDA linealetara
%egokituta erabiliz.
%-----
s=length(y0);
I=eye(s);          %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);
t=tsol(1);
y=ysol(1,:)' ;

%A matrizearen balioa lortzeko egiten dugu ondorengo ebaluaketa.
%Momentuz, ez dugu "g"-ren balioa erabiliko.
[A,g]=feval(ftylin,t);

%% Metodo trapezoidaleko pausoak EDA lineala askatzerakoan
for j1=2:m+1

    [t,y]=pasotraplin(ftylin,t,y,h);
    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.

end;
```

‘pasotraplin.m’ funtzioan ere aurretik Euler-en metodo inplizituan erabili dugun antzeko teknika erabili dugu, eta (7.9) adierazpeneko eragiketak honela egin ditugu:

- $g(t_n)$ eta $g(t_{n+1})$ balioak kalkulatu ditugu.
- $\left(I + \frac{h}{2}A\right)y_n$ eragiketa egin dugu eta termino independente izena eman diogu: $TI = \left(I + \frac{h}{2}A\right)y_n$.
- $g(t_n)$ dagoenean, aurreko termino independenteari $\left(\frac{h}{2}g(t_n) + \frac{h}{2}g(t_{n+1})\right)$ terminoa gehitu diogu eta batuketa berri hori berriz ere TI aldagaian gorde dugu: $TI = \left(I + \frac{h}{2}A\right)y_n + \frac{h}{2}g(t_n) + \frac{h}{2}g(t_{n+1})$.
- Bukatzeko, $C_1 = \left(I - \frac{h}{2}A\right)^{-1}$ matrizeaz biderkatu dugu aurreko termino independentea, eta horrela (7.9) adierazpeneko $y_{n+1} = C_1 \cdot TI$ balioa lortu dugu.

```
function [tn1,yn1] = pasotraplin(ftylin,tn,yn,h)
% "s" ekuazio diferentzial arrunteko y'=Ay+g(t) sistema askatzeko
%metodo trapezoidalaren forma linealeko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:
tn1 = tn+h;

%A eta g-ren ebaluazioa tn eta tn1 aldiuneetan:
[A,gn]=feval(ftylin,tn);
[A,gn1]=feval(ftylin,tn1);

%Metodo trapezoidal linealeko pausoa:
s=length(yn);
I=eye(s);
TI=(I+A*h/2)*yn;

%%Pausoa emateko behar dugun matrizea:
C1=(I-A*h/2)\I; %Alderantzizkoa kalkulatzen dugu.
if ~isempty(gn)
    TI=TI+(gn+gn1)*(h/2);
end
yn1=C1*TI;
```


7.1.3. EDA linealetarako Adams Moulton-en metodoa

Hasierako baliodun $y' = f(t, y)$ EDA askatzeko Adams Moulton-en metodo inplizitua honela emana datorrela ikusi dugu:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=1}^k \beta_j^* f_{n+k+1-j} \quad (7.10)$$

non $f_{n+k+1-j} = f(t_{n+k+1-j}, y_{n+k+1-j})$.

Askatzen ari garen EDA koefiziente konstantekoa eta lineala denean, Adams Moulton-en metodoak forma hau hartzen du:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=1}^k \beta_j^* (A y_{n+k+1-j} + g(t_{n+k+1-j})) \quad (7.11)$$

(7.11) ekuaziotik y_{n+k} terminoa askatuko dugu:

$$y_{n+k} = (I - h\beta_1^* A)^{-1} \left((I + h\beta_2^* A) y_{n+k-1} + h \sum_{j=3}^k \beta_j^* A y_{n+k+1-j} + h \sum_{j=1}^k \beta_j^* g(t_{n+k+1-j}) \right) \quad (7.12)$$

Programaziorako (7.12) adierazpena era honetan orokortuko dugu:

$$y_{n+k} = C_1 (C_2 y_{n+k-1} + C_3 Y_{n+k-2} + C_4 G_{n+k}) \quad (7.13)$$

(7.13) adierazpeneko matrizeak hauek izanik:

$$\left\{ \begin{array}{l} C_1 = (I - h\beta_1^* A)^{-1}, C_2 = (I + h\beta_2^* A), C_3 = (h\beta_3^* A \quad h\beta_4^* A \quad \dots \quad h\beta_{k-1}^* A \quad h\beta_k^* A), \\ C_4 = (h\beta_1^* \quad h\beta_2^* \quad \dots \quad h\beta_{k-1}^* \quad h\beta_k^*), Y_{n+k-2} = \begin{pmatrix} y_{n+k-2} \\ y_{n+k-3} \\ \vdots \\ y_{n+2} \\ y_{n+1} \end{pmatrix}, G_{n+k} = \begin{pmatrix} g(t_{n+k}) \\ g(t_{n+k-1}) \\ \vdots \\ g(t_{n+2}) \\ g(t_{n+1}) \end{pmatrix} \end{array} \right.$$

Ohartu $k=1$ eta $k=2$ kasuetarako C_3 eta Y_{n+k-2} matrizeak ez direla existitzen. $k=3$ denean, (7.13) adierazpena honela geratzen da:

$$y_{n+3} = C_1 \left(C_2 y_{n+2} + h\beta_3^* A y_{n+1} + \begin{pmatrix} h\beta_1^* & h\beta_2^* & h\beta_3^* \end{pmatrix} \cdot \begin{pmatrix} g(t_{n+3}) \\ g(t_{n+2}) \\ g(t_{n+1}) \end{pmatrix} \right) \quad (7.14)$$

Eta (7.14) adierazpenak agerian uzten du, $k=3$ kasurako aurreko pausoko balioaz gain, hau da, y_{n+2} balioaz gain, arinagoko pauso bateko y_{n+1} balioa behar dugula Adams Moulton-ekin pausoa emateko. $k=4$ denean, aurreko pausoko y_{n+3} balioaz gain, arinagoko bi pausoko y_{n+2} eta y_{n+1} balioak beharko ditugu, etab. Oro har, $k \geq 3$ ordenatik aurrera, aurreko pausoko y_{n+k-1} balioaz aparte, arinagoko $(k-2)$ pausoetako balioak behar dira Adams Moulton-ekin pausoa ematen hasteko. Arinagoko $(k-2)$ pausoetako balioak dira, hain zuzen, Y_{n+k-2} matrizean sartu ditugunak, eta programazioan matrize horri 'ykm1' deitu diogu. Y_{n+k-2} matrizea osatu arteko pausoak metodo trapezoidal linealeko pausoa emanda osatu ditugu. Eta behin Y_{n+k-2} matrizea osatuta dugunean hasi gara Adams Moulton-en metodoko pausoak ematen.

MATLAB-en metodo honi dagokion programa lau funtzio erabilia egin dugu: 'amoultonlin.m', 'hasi.m', 'pasoamlin.m' eta 'pasotraplin.m'. 'amoultonlin.m' funtzioak 'eulerinpln.m' eta 'traplin.m' funtzioek izan dituzten sarrerako argumentuez gain beste bat izango du, ordena adieraziko duena eta 'orden' deitu dioguna. Horrela, $k=1$ ordenatik $k=5$ ordenarainoko Adams Moulton-en zenbakizko metodoak funtzio bakarrean izango ditugu erabilgarri. 'hasi.m' funtzioa aurretik erabilitakoa izango da eta 'pasoamlin.m' funtzioa Adams Moulton-en metodoko pausoa EDA linealetara egokitua izango da. 'pasotraplin.m' funtzioa erabili dugu $k \geq 3$ ordenetan hasieran eman beharreko pauso osagarriak lortzeko.

```

function [tsol,ysol]=amoultonlin(ftylin,tartea,y0,m,orden)
%"s" ekuazio diferentzial arrunteko  $y'=Ay+g(t)$  sistema
%askatzen duen funtzioa da, Adams Moulton-en metodoa EDA
%linealetara egokituta erabiliz.
%-----

s=length(y0);
I=eye(s);           %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);
t=tsol(1);
y=ysol(1,:);

%A matrizearen balioa lortzeko egiten dugu ondorengo ebaluaketa.
%Momentuz, ez dugu "g"-ren balioa erabiliko.
[A,g]=feval(ftylin,t);

%%Adams Moulton metodoari dagozkion C1, C2 eta C3 matrizeak
%(ordenaren arabera ezberdinak direnak):
if orden==1
    C1=(I-A*h)\I;    %Alderantzizkoa kalkulatzen dugu.
    C2=I;%y_{n+k-1} elementuari bidertzen doan matrizea.
    C3=[];
    ykm1=[];

elseif orden==2
    C1=(I-A*h/2)\I;  %Alderantzizko matrizea.
    C2=I+A*h/2;
    C3=[];
    ykm1=[];

elseif orden==3
    C1=(I-A*h*5/12)\I;  %Alderantzizko matrizea.
    C2=I+A*h*8/12;
    C3=-1/12*h*A;
    ykm1=y;

elseif orden==4
    C1=(I-A*h*9/24)\I;  %Alderantzizko matrizea.
    C2=I+A*h*19/24;
    C3=[-5/24*h*A   h/24*A];
    ykm1=y;

elseif orden==5
    C1=(I-A*h*251/720)\I;  %Alderantzizko matrizea.
    C2=I+A*h*323/360;
    C3=[-11/30*h*A  53/360*h*A  -19/720*h*A];
    ykm1=y;

else 'Bakarrik k=5-era arte egiten dut'
end

```

```

%Adams Moulton-en 1. eta 2. ordenako metodoetan zuzenean has
%gaitezke Adams Moulton-en metodoko pausoa aplikatzen. Baina
%3. ordenatik aurrera, Adams Moulton-eko pausoa emateko  $Y_{n+k-2}$ 
%matrizea osatuta izan behar dugu. 3. ordenan pauso bat eman
%beharko dugu beste metodoren batekin, 4. ordenan 2 pauso, e.a.
%Hasierako pauso hauek metodo trapezoidal lineala erabilita
%emango ditugu.

for j1=2:orden-1
    [t,y]=pasotraplin(ftylin,t,y,h);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.

    if (j1+1)<orden
        ykm1=[y; ykm1];
    end
end

%%Gainontzeko pausoak Adams Moulton linealei dagokien pausoarekin
%ematen ditugu:
if orden==1
    p=2;
elseif orden==2
    p=2;
elseif orden>2
    p=orden;
end

for j1=p:m+1
    [t,y,ykm1]=pasoamlin(ftylin,t,y,h,orden,C1,C2,C3,ykm1);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.
end;

```

Adams Moulton-en metodoko pausoa EDA linealetara egokitua:

```
function [tn1,yn1,ykm1]=pasoamlin(ftylin,tn,yn,h,orden,...
                                C1,C2,C3,ykm1)
%"s" ekuazio diferentzial arrunteko y'=Ay+g(t) sistema
%askatzeko Adams Moulton metodoaren forma linealeko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:
tn1 = tn+h;
%A eta g-ren ebaluazioa tn eta tn1 aldiuneetan:
s=size(yn,1);
gk=zeros(s,orden);
for j1=1:orden
    [A,gk(:,j1)]=feval(ftylin,tn1-(j1-1)*h);
end
%Adams Moulton linealetako pausoa:
if orden==1
    TI=C2*yn;
    if ~isempty(gk)
        TI=TI+gk*h;
    end
    yn1=C1*TI ;
elseif orden==2
    TI=C2*yn;
    if ~isempty(gk)
        TI=TI+h/2*gk*[1;1];
    end
    yn1=C1*TI;
elseif orden==3
    TI=C2*yn+C3*ykm1;
    if ~isempty(gk)
        TI=TI+(h/12)*gk*[5;8;-1];
    end
    yn1=C1*TI ;
elseif orden==4
    TI=C2*yn+C3*ykm1;
    if ~isempty(gk)
        TI=TI+(h/24)*gk*[9;19;-5;1];
    end
    yn1=C1*TI ;
elseif orden==5
    TI=C2*yn+C3*ykm1;
    if ~isempty(gk)
        TI=TI+(h/720)*gk*[251;646;-264;106;-19];
    end
    yn1=C1*TI ;
end
if orden>2
    ykm1=[yn; ykm1(1:s*(orden-3),:)];%ykm1 matrizearen eguneraketa
end
```

7.1.4. EDA linealaterako BDF metodoak

BDF metodoen adierazpena $\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = hf_{n+k}$ dela ere ikusi dugu 4.

kapituluan. Ebatzi nahi dugun EDA koefiziente konstantekoa eta lineala denean, BDFek forma hau hartzen dute:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = h(Ay_{n+k} + g(t_{n+k})) \quad (7.15)$$

(7.15) adierazpeneko eragiketak eginez hauxe daukagu:

$$(\hat{\alpha}_k I - hA)y_{n+k} = -\sum_{j=0}^{k-1} \alpha_j y_{n+j} + hg(t_{n+k}) \quad (7.16)$$

(7.16) berdintzan azken pausoko balioa, hau da, t_{n+k-1} aldiunekoa eta gainerakoak, $\{y_{n+j} : j = 0, 1, \dots, k-2\}$, bereizita idatziko ditugu:

$$(\hat{\alpha}_k I - hA)y_{n+k} = -\hat{\alpha}_{k-1} y_{n+k-1} - \sum_{j=0}^{k-2} \hat{\alpha}_j y_{n+j} + hg(t_{n+k}) \quad (7.17)$$

(7.17) adierazpenetik y_{n+k} askatuko dugu, ondokoa lortuz:

$$y_{n+k} = (\hat{\alpha}_k I - hA)^{-1} \left(-\hat{\alpha}_{k-1} y_{n+k-1} - \sum_{j=0}^{k-2} \hat{\alpha}_j y_{n+j} + hg(t_{n+k}) \right) \quad (7.18)$$

Eta (7.18) adierazpena ordena guztietarako era honetan orokortuko dugu:

$$y_{n+k} = C_1 (C_2 y_{n+k-1} + C_3 y_{n+k-2} + hg(t_{n+k})) \quad (7.19)$$

$$\text{Non: } \begin{cases} C_1 = (\hat{\alpha}_k I - hA)^{-1}, C_2 = -\alpha_{k-1} I, \\ C_3 = (-\hat{\alpha}_{k-2} I \quad -\hat{\alpha}_{k-3} I \quad \dots \quad -\hat{\alpha}_1 I \quad -\hat{\alpha}_0 I), Y_{n+k-2} = \begin{pmatrix} y_{n+k-2} \\ y_{n+k-3} \\ \vdots \\ y_{n+1} \\ y_n \end{pmatrix} \end{cases}$$

$k=1$ kasuan ez genduke C_3 eta Y_{n+k-2} matrizerik izango. $k=2$ kasuan, $C_3 = -\hat{\alpha}_0 I$ eta $Y_n = y_n$ izango lirateke. Horrek esan nahi du ezen $k=2$ kasuan aurreko pausoko y_{n+1} balioaz gain, arinagoko pauso bateko y_n balioa behar dugula BDF metodoko pausoa emateko. Oro har, $k \geq 2$ ordenatik aurrera, aurreko pausoko y_{n+k-1} balioaz gain, arinagoko $(k-1)$ pausoetako balioak behar dira BDFekin pausoa emateko. Arinagoko $(k-1)$ pausoetako balioak dira hain zuzen Y_{n+k-2} matrizean sartu ditugunak, eta programazioan matrize horri 'ykm1' deitu diogu. Y_{n+k-2} matrizea osatu arteko pausoak metodo trapezoidal linealeko pausoa emanda osatu ditugu. Eta behin Y_{n+k-2} matrizea osatuta dugunean hasi gara BDF metodoko pausoak ematen.

Metodo honi dagokion programarako lau funtzio behar izan ditugu: 'BDFlin.m', 'hasi.m', 'pasoBDFlin.m' eta 'pasotraplin.m'. 'BDFlin.m' funtzioa da funtzio nagusia eta 'amoultonlin.m' funtzioaren kasuan bezalaxe, zer ordenatuko BDFa erabiltzea nahi dugun adierazi behar dugu. $k=1$ ordenatik $k=4$ ordenarainoko BDF zenbakizko metodoak funtzio bakarrean programatu ditugu. 'hasi.m' funtzioa aurretik erabilitakoa da eta 'pasoBDFlin.m' funtzioa BDF metodoko pausoa EDA linealetara egokitua izango da. $k \geq 2$ ordenetan behar diren balioak lortzeko 'pasotraplin.m' funtzioa erabili dugu hasieran.

```

function [tsol,ysol]=BDFlin(ftylin,tartea,y0,m,orden)
%"s" ekuazio diferentzial arrunteko  $y'=Ay+g(t)$  sistema askatzen
%duen funtzioa da, BDF metodoa EDA linealetara egokituta erabiliz.
%-----

s=length(y0);
I=eye(s);           %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:)' ;

%A matrizearen balioa lortzeko egiten dugu ondorengo ebaluaketa.
%Momentuz, ez dugu "g"-ren balioa erabiliko.
[A,g]=feval(ftylin,t);

%%BDF metodoari dagozkion zenbait matrize (ordenaren arabera
%ezberdinak direnak):
if orden==1
    C1=(I-A*h)\I;    %Alderantzizkoa kalkulatzen dugu.
    C2=I;
    C3=[];
    ykm1=[];

elseif orden==2
    C1=(3/2*I-A*h)\I;    %Alderantzizko matrizea.
    C2=2*I;
    C3=-1/2*I;
elseif orden==3
    C1=(11/6*I-A*h)\I;    %Alderantzizko matrizea.
    C2=3*I;
    C3=[-3/2*I    1/3*I];
elseif orden==4
    C1=(25/12*I-A*h)\I;    %Alderantzizko matrizea.
    C2=4*I;
    C3=[-3*I    4/3*I    -1/4*I];

else 'Bakarrik k=4-ra arte egiten dut'

end

%Y_{n+k-2} matrizea osatuta izateko behar ditugun
%pausoak emango ditugu metodo trapezoidal lineala erabilita.

```



```
% "k" ordenarako "k" puntu izan behar ditugu hurrengo pausoa eman
%ahal izateko. Balioetako bat ematen diguten y0 da, beraz,
%(k-1) puntu behar ditugu. Honek esan nahi du k ordenan (k-1)
%pauso eman behar ditugula beste metodoren bat erabilita. Bat
%ordenan azpiko for buklea salto egingo luke.

if orden >1
    ykm1=y0;
end
for j1=2:orden
    [t,y]=pasotraplin(ftylin,t,y,h);

    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin ysol eguneratzen dugu.

    if j1<orden
        ykm1=[y' ykm1];
    end

end
ykm1=ykm1';

%%Gainontzeko pausoak BDF linealei dagokien pausoarekin ematen
%ditugu:
for j1=orden+1:m+1

    [t,y,ykm1]=pasoBDFlin(ftylin,t,y,h,orden,C1,C2,C3,ykm1);

    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin ysol eguneratzen dugu.

end;
```

BDF metodoko pausoa EDA linealetara egokitua:

```
function [tn1,yn1,ykm1]=pasoBDFlin(ftylin,tn,yn,h,orden,...
                                   C1,C2,C3,ykm1)
%"s" ekuazio diferentzial arrunteko y'=Ay+g(t) sistema
%askatzeko BDF metodoaren forma linealeko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:

tn1 = tn+h;

%A eta g-ren ebaluazioa tn eta tn1 aldiuneetan:

[A,gn]=feval(ftylin,tn);
[A,gn1]=feval(ftylin,tn1);

%BDF linealetako pausoa:

s=size(yn,1);
if orden==1
    TI=C2*yn;
    if ~isempty(gn)
        TI=TI+gn1*h;
    end
    yn1=C1*TI ;
else
    TI=C2*yn+C3*ykm1;
    if ~isempty(gn)
        TI=TI+gn1*h;
    end
    yn1=C1*TI ;
%%ykm1 matrizearen eguneraketa:
ykm1=[yn; ykm1(1:s*(orden-2),:)] ;
end
```

7.1.5. EDA linealaterako NDF metodoak

Aurreko kasuetan egin dugun antzera, NDF metodoaren adierazpenetik abiatuko gara:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = h f_{n+k} + \kappa \gamma_k \nabla^{k+1} y_{n+k} \quad (7.20)$$

(7.20) adierazpena koefiziente konstanteko EDA linealen sistema bati aplikatzean haxe daukagu:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = h(Ay_{n+k} + g(t_{n+k})) + \kappa \gamma_k \nabla^{k+1} y_{n+k} \quad (7.21)$$

$$\text{non: } \nabla^{k+1} y_{n+k} = \sum_{j=0}^{k+1} (-1)^j \binom{k+1}{j} y_{n+k-j}$$

(7.21) adierazpenean eragiketak eginez hona iristen gara:

$$(\hat{\alpha}_k I - hA - \kappa \gamma_k I) y_{n+k} = - \sum_{j=0}^{k-1} \hat{\alpha}_j y_{n+j} + \kappa \gamma_k \sum_{j=1}^{k+1} (-1)^j \binom{k+1}{j} y_{n+k-j} + hg(t_{n+k}) \quad (7.22)$$

Lehen BDFen adierazpenean egin dugun bezalaxe, (7.22) adierazpenean ere bereizita idatziko ditugu azken pausoko y_{n+k-1} balioa eta aurreko pausoetako

$\{y_{n+j} : j = -1, 0, 1, \dots, k-2\}$ balioak:

$$\begin{aligned} (\hat{\alpha}_k I - hA - \kappa \gamma_k I) y_{n+k} = & -\hat{\alpha}_{k-1} y_{n+k-1} - \sum_{j=0}^{k-2} \hat{\alpha}_j y_{n+j} - \kappa \gamma_k \binom{k+1}{1} y_{n+k-1} \\ & + \kappa \gamma_k \sum_{j=2}^{k+1} (-1)^j \binom{k+1}{j} y_{n+k-j} + hg(t_{n+k}) \end{aligned} \quad (7.23)$$

Azkenik, (7.23) adierazpenetik y_{n+k} askatzen dugu:

$$y_{n+k} = C_1 \left[(-\hat{\alpha}_{k-1} I - \kappa \gamma_k I(k+1)) y_{n+k-1} - \sum_{j=0}^{k-2} \hat{\alpha}_j y_{n+j} + \kappa \gamma_k \sum_{j=2}^{k+1} (-1)^j \binom{k+1}{j} y_{n+k-j} + hg(t_{n+k}) \right] \quad (7.24)$$

$C_1 = (\hat{\alpha}_k I - hA - \kappa \gamma_k I)^{-1}$ izanik.

(7.24) adierazpena NDFen ordena guztietarako erabilgarria izan dadin honela orokortu dugu:

$$y_{n+k} = C_1 [C_2 y_{n+k-1} + C_3 Y_{n+k-2} + hg(t_{n+k})] \quad (7.25)$$

$$\text{Non: } \begin{cases} C_1 = (\hat{\alpha}_k I - hA - \kappa \gamma_k I)^{-1}, C_2 = (-\hat{\alpha}_{k-1} - \kappa \gamma_k (k+1))I, \\ C_3 = \left(-\hat{\alpha}_{k-2} I + \kappa \gamma_k I (-1)^2 \binom{k+1}{2}, \dots, -\hat{\alpha}_0 I + \kappa \gamma_k I (-1)^k \binom{k+1}{k}, \kappa \gamma_k I (-1)^{k+1} \right), \\ Y_{n+k-2} = \begin{pmatrix} y_{n+k-2} \\ y_{n+k-3} \\ \vdots \\ y_{n+1} \\ y_n \\ y_{n-1} \end{pmatrix} \end{cases}$$

Kasu honetan ohartu, NDF metodoetako C_3 matrizeak BDF metodokoak baino zutabe bat gehiago duela, eta Y_{n+k-2} matrizeak ere NDF metodoetan BDFetan baino errenkada bat gehiago duela. $k=1$ kasuan $C_3 = \kappa \gamma_k I$ eta $Y_{n-1} = (y_{n-1})$ izango lirateke. $k=2$ kasuan, $C_3 = \begin{pmatrix} -\frac{1}{2}I + 3\kappa \gamma_k I & -\kappa \gamma_k I \end{pmatrix}$ eta $Y_n = \begin{pmatrix} y_n \\ y_{n-1} \end{pmatrix}$ izango genituzke. Oro har, $k \geq 1$ ordenatik aurrera, aurreko pausoko y_{n+k-1} balioaz gain, arinagoko k pausoetako balioak behar dira NDFekin pausoa emateko. Arinagoko k pausoetako balioek Y_{n+k-2} matrizea osatzen dute eta programazioan 'ykm1' deitu diogu matrize horri. Y_{n+k-2} matrizea osatu arteko pausoak metodo trapezoidaleko pausoa emanda osatu ditugu.

NDFak programatzeko ere lau funtzio behar izan ditugu: 'NDFlin.m', 'hasi.m', 'pasoNDFlin.m' eta 'pasotraplin.m'. Jarraian jarri ditugu 'NDFlin.m' eta 'pasoNDFlin.m' funtzioak.

```

function [tsol,ysol]=NDFlin(ftylin,tartea,y0,m,orden)
%"s" ekuazio diferentzial arrunteko  $y'=Ay+g(t)$  sistema
%askatzen duen funtzioa da, NDF metodoa EDA linealetara egokituta
%erabiliz.
%-----

s=length(y0);
I=eye(s);           %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:)' ;

%A matrizearen balioa lortzeko egiten dugu ondorengo ebaluaketa.
%Momentuz, ez dugu "g"-ren balioa erabiliko.

[A,g]=feval(ftylin,t);

%%NDF metodoari dagozkion zenbait matrize (ordenaren arabera
%ezberdinak direnak):
%kappa = [-37/200; -1/9; -0.0823; -0.0415; 0];
%gamma = [1; 3/2; 11/6; 25/12; 137/60];
if orden==1
    kappa=-37/200;
    gamma=1;
    kg=kappa*gamma;
    C1=(I-A*h-kg*I)\I;
    C2=I-2*kg*I;
    C3=kg*I;

elseif orden==2
    kappa=-1/9;
    gamma=3/2;
    kg=kappa*gamma;
    C1=(3/2*I-A*h-kg*I)\I;
    C2=2*I-3*kg*I;
    C3=[-1/2*I+3*kg*I -kg*I];

elseif orden==3
    kappa=-0.0823;
    gamma=11/6;
    kg=kappa*gamma;
    C1=(11/6*I-A*h-kg*I)\I;
    C2=3*I-4*kg*I;
    C3=[-3/2*I+6*kg*I 1/3*I-4*kg*I kg*I];

```

```

elseif orden==4
    kappa=-0.0415;
    gamma=25/12;
    kg=kappa*gamma;
    C1=(25/12*I-A*h-kg*I)\I;
    C2=4*I-5*kg*I;
    C3=[-3*I+10*kg*I   4/3*I-10*kg*I   -1/4*I+5*kg*I   -kg*I];

else 'Bakarrik k=4-ra arte egiten dut'

end

%Y_{n+k-2} matrizea osatuta izateko behar
%ditugun pausoak emango ditugu metodo trapezoidal lineala erabilita.

%"k" ordenarako "k+1" puntu izan behar ditugu hurrengo pausoa eman
%ahal izateko. Balioetako bat ematen diguten y0 da, beraz,
%k puntu behar ditugu. Beraz, k pauso eman behar ditugu beste
%metodoren bat erabilita.

ykm1=y0;
%BDF-ek baino pauso bat gehiago ematen dute hasieran.
for j1=2:orden+1
    [t,y]=pasotraplin(ftylin,t,y,h);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.

    if j1<orden+1
        ykm1=[y' ykm1];
    end

end
ykm1=ykm1';

%Gainontzeko pausoak NDF linealei dagokien pausoarekin ematen
%ditugu:
for j1=orden+2:m+1

    [t,y,ykm1]=pasoNDFlin(ftylin,t,y,h,orden,C1,C2,C3,ykm1);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.

end;

```

NDF metodoko pausoa EDA linealetara egokitua:

```
function [tn1,yn1,ykm1]=pasoNDFlin(ftylin,tn,yn,h,orden,...
                                   C1,C2,C3,ykm1)
%"s" ekuazio diferentzial arrunteko y'=Ay+g(t) sistema
%askatzeko NDF metodoaren forma linealeko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:

tn1 = tn+h;

%A eta g-ren ebaluazioa tn eta tn1 aldiuneetan:

[A,gn]=feval(ftylin,tn);
[A,gn1]=feval(ftylin,tn1);

%NDF linealetako pausoa:

s=size(yn,1);
TI=C2*yn+C3*ykm1;
if ~isempty(gn)
    TI=TI+gn1*h;
end
yn1=C1*TI ;

%%ykm1 matrizearen eguneraketa:
ykm1=[yn; ykm1(1:s*(orden-1),:)];
```

7.1.6. EDA linealaterako EBDF metodoak

EBDF metodoan bi aldiz aplikatzen dira BDFak iragarle bezala, eta, azkenean, zuzentzaile bat aplikatzen da, zuzentzailearen formula hauxe izanik:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \beta_k f_{n+k} + h \beta_{k+1} \bar{f}_{n+k+1} \quad (7.26)$$

$f_{n+k} = f(t_{n+k}, y_{n+k})$, $\bar{f}_{n+k+1} = f(t_{n+k+1}, \bar{y}_{n+k+1})$ eta $\bar{y}_{n+k+1}$ BDFarekin lortutako iragarlea izanik.

Iragarleetan aurretik egin ditugun BDF linealetako pausoak erabiliko ditugu. Beraz, bakarrik formula zuzentzailea EDA linealetara egokitzea geratzen zaigu. Horretarako, (7.26) formula zuzentzailea koefiziente konstanteko EDA linealari aplikatuko diogu:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h\beta_k (Ay_{n+k} + g(t_{n+k})) + h(\beta_{k+1} A\bar{y}_{n+k+1} + g(t_{n+k+1})) \quad (7.27)$$

(7.27) adierazpenean eragiketak egin ondoren, haxe lortzen da:

$$(\alpha_k I - h\beta_k A)y_{n+k} = -\sum_{j=0}^{k-1} \alpha_j y_{n+j} + h\beta_k g(t_{n+k}) + h\beta_{k+1} (A\bar{y}_{n+k+1} + g(t_{n+k+1})) \quad (7.28)$$

(7.28) adierazpenean bereizita idatziko ditugu y_{n+k-1} balioa eta aurreko pausoetako $\{y_{n+j} : j=0,1,\dots,k-2\}$ balioak:

$$(\alpha_k I - h\beta_k A)y_{n+k} = -\alpha_{k-1}y_{n+k-1} - \sum_{j=0}^{k-2} \alpha_j y_{n+j} + h\beta_k g(t_{n+k}) + h\beta_{k+1} (A\bar{y}_{n+k+1} + g(t_{n+k+1})) \quad (7.29)$$

Eta (7.29) adierazpenetik y_{n+k} askatuz, haxe daukagu:

$$y_{n+k} = C_1 (C_2 y_{n+k-1} + C_3 Y_{n+k-2} + C_4 \bar{y}_{n+k+1} + h\beta_k g(t_{n+k}) + h\beta_{k+1} g(t_{n+k+1}))$$

$$\text{Non: } \begin{cases} C_1 = (\alpha_k I - h\beta_k A)^{-1}, & C_2 = -\alpha_{k-1} I, \\ C_3 = (-\alpha_{k-2} I & -\alpha_{k-3} I & \dots & -\alpha_1 I & -\alpha_0 I), & C_4 = h\beta_{k+1} A, \\ Y_{n+k-2} = \begin{pmatrix} y_{n+k-2} \\ y_{n+k-3} \\ \vdots \\ y_{n+1} \\ y_n \end{pmatrix} \end{cases}$$

BDFetan gertatzen zaigun bezalaxe, $k=1$ kasuan ez genuke C_3 eta Y_{n+k-2} matrizerik izango. $k=2$ kasuan, $C_3 = -\alpha_0 I$ eta $Y_n = y_n$ izango lirateke. Oro har, eta BDFetan gertatzen den bezalaxe, $k \geq 2$ ordenatik aurrera, aurreko pausoko y_{n+k-1} balioaz gain, arinagoko $(k-1)$ pausoetako balioak behar dira EBDFein pausoa emateko. Arinagoko $(k-1)$ pausoetako balioak Y_{n+k-2} matrizean daude eta programazioan matrize horri 'ykm1' deitu diogu. Y_{n+k-2} matrizea osatu arteko pausoak metodo trapezoidaleko pausoa emanda osatu ditugu.

Metodo honi dagokion programarako lau funtzio behar izan ditugu: 'EBDFlin.m', 'hasi.m', 'pasoEBDFlin.m' eta 'pasotraplin.m'. 'EBDFlin.m' funtzioa da funtzio nagusia eta zer ordenatan ebatzi nahi dugun adieraziko dugu. $k=1$ ordenatik $k=4$ ordenarainoko EBDf zenbakizko metodoak funtzio bakarrean programatu ditugu. 'pasoEBDFlin.m' funtzioa EBDf metodoko pausoa EDA linealetara egokitua izango da. $k \geq 2$ ordenetan behar diren balioak lortzeko 'pasotraplin.m' funtzioa erabili dugu hasieran.

```

function [tsol,ysol]=EBDFlin(ftylin,tartea,y0,m,orden)
% "s" ekuazio diferentzial arrunteko  $y'=Ay+g(t)$  sistema
%askatzen duen funtzioa da, EBDF metodoa EDA linealetara
%egokituta erabiliz.
%-----

s=length(y0);
I=eye(s);           %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:)' ;

%A matrizearen balioa lortzeko egiten dugu ondorengo ebaluaketa.
%Momentuz, ez dugu "g"-ren balioa erabiliko.

[A,g]=feval(ftylin,t);

%%EBDF metodoko alfa eta beta koefizienteak.
%alfa koefizienteak AA izeneko matrizean daude eta beta
%koefizienteak BB izeneko matrizean:
%AA=[alf0 alf1 alf2..... ] Ordena honetan idatzita daude.
%Errenkada bakoitzean ordena horretako alfa_i koefizienteak daude.
AA=[-1 1 0 0 0 0 0 0 0;...
     5/23 -28/23 1 0 0 0 0 0 0;...
    -17/197 99/197 -279/197 1 0 0 0 0 0;...
    111/2501 -728/2501 2124/2501 -4008/2501 1 0 0 0 0];
%BB=[beta(k+1) beta(k)]; %Ordena honetan idatzita daude.
BB=[-1/2 3/2;...
     -4/23 22/23;...
     -18/197 150/197 ;...
     -144/2501 1644/2501 ];

C1=(AA(orden,orden+1)*I-h*BB(orden,2)*A)\I;
%y_{n+k-1} terminoa bidertzen duen matrizea da C2.
C2=-AA(orden,orden)*I;
%yb_{n+k+1} terminoa bidertzen duen matrizea da C4.
C4=h*BB(orden,1)*A;
%Y_{n+k-2} bidertzen duen matrizea da C3.
if orden==1
    C3=[];
else
    C3=-AA(orden,orden-1)*I;
    for j1=3:orden
        C3=[C3 -AA(orden,orden+1-j1)*I];
    end
end
end

```

```

%BDF metodoetako koefizienteak. Iragarpeneko bi pausoak BDF-ak
%erabilita emango baititugu.
if orden==1
    CC1=(I-A*h)\I;
    CC2=I;
    CC3=[];
    ykm1=[];

elseif orden==2
    CC1=(3/2*I-A*h)\I;
    CC2=2*I;
    CC3=-1/2*I;
elseif orden==3
    CC1=(11/6*I-A*h)\I;
    CC2=3*I;
    CC3=[-3/2*I  1/3*I];
elseif orden==4
    CC1=(25/12*I-A*h)\I;
    CC2=4*I;
    CC3=[-3*I  4/3*I  -1/4*I];

else 'Bakarrik k=4-ra arte egiten dut'
end
%BDF lineala hasten dugun era berdinean hasten dugu EBDF lineala
%ere:
if orden >1
    ykm1=y0;
end
for j1=2:orden
    [t,y]=pasotraplin(ftylin,t,y,h);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.
    if j1<orden
        ykm1=[y' ykm1];
    end
end

end
ykm1=ykm1';

%Gainontzeko pausoak EBDF linealei dagokien pausoarekin ematen
%ditugu:
for j1=orden+1:m+1

    [t,y,ykm1]=pasoEBDFlin(ftylin,t,y,h,orden,BB,C1,C2,C3, ...
                           C4,CC1,CC2,CC3,ykm1);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.
end;

```

EBDF metodoko pausoa EDA linealetara egokitua:

```
function [tn1,yn1,ykm1]=pasoEBDFlin(ftylin,tn,yn,h,orden,BB,...
                                   C1,C2,C3,C4,CC1,CC2,CC3,ykm1)
%"s" ekuazio diferentzial arrunteko  $y'=Ay+g(t)$  sistema askatzeko
%EBDF metodoaren forma linealeko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:

tn1 = tn+h;

%A eta g-ren ebaluazioa tn eta tn1 aldiuneetan:

[A,gn]=feval(ftylin,tn);
[A,gn1]=feval(ftylin,tn1);

%Iragarpeneko bi pausoak ematen ditugu BDF-ak erabilita:

s=size(yn,1);
[t1,ynb1,ykm2]=pasoBDFlin(ftylin,tn,yn,h,orden,CC1,CC2,CC3,ykm1);
[t2,ynb2,ykm3]=pasoBDFlin(ftylin,t1,ynb1,h,orden,CC1,CC2,...
                           CC3,ykm2);

[Ak2,gn2]=feval(ftylin,tn1+h);

%EBDF metodoko zuzentzaileari dagokion pausoa ematen dugu:
if orden ==1
    TI=C2*yn+C4*ynb2;
else
    TI=C2*yn+C3*ykm1+C4*ynb2;
end

if ~isempty(gn)
    TI=TI+(h*BB(orden,2)*gn1+h*BB(orden,1)*gn2);
end

yn1=C1*TI ;

%%ykm1 matrizearen eguneraketa:
if orden>1
    ykm1=[yn; ykm1(1:s*(orden-2),:)] ;
end
```

7.2. EDOZEIN MOTATAKO EDA-K ASKATZEKO ZENBAIT METODO ESPLIZITUREN PROGRAMAZIOA

1. kapituluan hasierako baliodun lehen ordenako $y'(t) = f(t, y(t))$, $y(t_0) = y_0$ EDAk askatzeko Euler-en metodo esplizitua programatu dugu. Hurrena lan honetan barrena ikusi ditugun Adams Bashforth-en metodoak eta zenbait Runge-Kutta metodo programatuko ditugu. Metodo horiek ere esplizituak dira. Adams Bashforth-en metodoek, pauso anitzeko metodoak izanik, beste metodoren baten laguntza beharko dute programazioari hasiera emateko. Runge-Kutta metodoak, aldiz, pauso bakarreko eta etapa anitzeko metodoak izanik, gai dira edozein ordenatan beren kabuz hasiera emateko.

7.2.1. Edozein motatako EDAtarako Adams Bashforth-en metodoak

Adams Bashforth-en metodoak honako adierazpen honen bidez emanda datoz:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=1}^k \tilde{\beta}_j f_{n+k-j}$$

Metodo hauek pauso anitzekoak dira. Horrela, k pausoko metodoak $t_n, t_{n+1}, \dots, t_{n+k-1}$ aldiuneetako $f_n, f_{n+1}, \dots, f_{n+k-1}$ balioak eskura izan behar ditu t_{n+k} aldiuneko y_{n+k} balioa kalkulatu ahal izateko. Balio horiek lortzeko behar diren hasierako pausoak Euler-en metodo esplizitua erabiliz kalkulatu ditugu, baina beste metodoren bat ere erabili ahal izango genuke Adams Bashforth-en metodoari hasiera emateko. Hasieran, k pausoko metodoak behar dituen puntuak lortu ahala beraien f -ak kalkulatu ditugu, eta balio horiekin ‘maldak’ izeneko matrizea sortu dugu. ‘maldak’ matrizeak k zutabe eta EDAREN dimentsioak adina errenkada ditu. Horrela, k pausoko metodorako ‘maldak’ bektorea hauxe litzateke:

$$maldak = (f_{n+k-1} \ f_{n+k-2} \ \dots \ f_n) \quad (7.30)$$

Pausoa eman eta t_{n+k} aldiuneko y_{n+k} kalkulatu ostean, aldiune horretako deribatua f_{n+k} kalkulatzeko da eta ‘maldak’ izeneko matrizea eguneratzen da. ‘maldak’ matrizeko lehenengo $(k-1)$ zutabeak, $f_{n+k-1}, f_{n+k-2}, \dots, f_{n+1}$ balioak dituzten zutabeak hain zuzen, posizio bat eskuinera eramaten dira eta matrize horretako lehenengo zutabea f_{n+k} balioa sartzen da. Horrek esan nahi du ezen ‘maldak’ matrizean aurreko pausoan izan den azken zutabea, f_n , galdu egiten dela, hurrengo pausoa emateko ez baitugu behar balio hori. Eguneraketaren ondoren, ‘maldak’ matrizea prest daukagu t_{n+k+1} aldiuneko y_{n+k+1} balioa kalkulatzeko:

$$maldak = (f_{n+k} \ f_{n+k-1} \ \dots \ f_{n+1}) \quad (7.31)$$

MATLABen ‘abashforth.m’, ‘pasoab.m’ eta ‘hasi.m’ funtzioak programatu ditugu Adams Bashforth-en metodoa programatu ahal izateko. ‘hasi.m’ funtzioa orain arte erabili dugun funtzio bera da, ‘abshforth.m’ funtzio nagusia da eta ‘pasoab.m’ Adams Bashforth-en metodoko pausoa.

```
function [tsol,ysol]=abashforth(fty,tartea,y0,m,orden)
%
%"s" ekuazio diferentzial arrunteko y'=f(t,y) EDA askatzen
%duen Adams Bashforth-en metodoa.
%-----
[h,tsol,ysol]=hasi(tartea,y0,m);

%Adams Bashforth-eko hasiera Euler-en pausoa erabilita egiten
%dugu:
tn = tsol(1);
yn = ysol(1,:)' ;
maldak = feval(fty,tn,yn);
for j1=2:orden
    [tn,yn] = pasoeu(fty,tn,yn,h);
    malda = feval(fty,tn,yn);
    tsol(j1)=tn;
    ysol(j1,:)=yn';
    maldak = [malda maldak];
end

%Adams Bashforth-en metodoko pausoak ematen ditugu:

t=tsol(orden);
y=ysol(orden,:)' ;

for j1=orden+1:m+1

    [t,y,maldak]=pasoab(fty,t,y,h,maldak,orden);

    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin ysol eguneratzen dugu.
end
```

Adams Bashforth-en metodoko pausoa:

```
function [tn1,yn1,maldak] = pasoab(fty,tn,yn,h,maldak,orden)
%
%"s" ekuazio diferentzial arrunteko y'=f(t,y) sistema askatzeko
%Adams Bashforth-en metodoko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Adams Bashforth metodoko formulak:

    if orden==1
        yn1 = yn + h*maldak;
    elseif orden==2
        yn1 = yn + (h/2)*maldak*[3;-1];
    elseif orden==3
        yn1 = yn + (h/12)*maldak*[23;-16;5];
    elseif orden==4
        yn1 = yn + (h/24)*maldak*[55;-59;37;-9];
    elseif orden==5
        yn1 = yn + (h/720)*maldak*[1901;-2774;2616;-1274;251];
    else 'Bakarrik k=5-era arte egiten dut'
    end

%Denbora eguneratzen dugu:

tn1 = tn+h;

%(tn+1,yn+1) puntuko maldaren kalkulua:
malda = feval(fty,tn1,yn1);

%Deribatuen zerrendaren eguneraketa:
maldak=[malda maldak(:,1:orden-1)];
```

Adams Bashforth-en metodoek egonkortasun-eremu txikiak dituzte, bereziki ordenan gora egiten dugunean, eta pauso asko eman beharko dituzte autobalio handidun EDAk ondo askatzea nahi badugu.

7.2.2. Edozein motatako EDAtarako Runge-Kutta metodoak

Hasierako baliodun EDA bat emanik, s etapatako Runge-Kutta metodo esplizituak adierazpen hauxe dauka:

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i k_i \quad (7.32)$$

$$k_i = f(t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j), \quad i=1,2,\dots,s \text{ izanik.}$$

MATLAB-en etapa kopurua eta ordena berdinak diren Runge-Kutta esplizituak (eta hau gertatzen da 1, 2, 3 eta 4 ordenetan) eta DOPRI(5,4) metodoko bosgarren ordenako Runge-Kutta metodoa programatu ditugu. Pauso bakarreko metodoak dira eta, beraz, ez dute behar izan hasiera berezirik.

```
function [tsol,ysol]=rk(fty,tartea,y0,m,orden)
%
%"s" ekuazio diferentzial arrunteko  $y'=f(t,y)$  EDA askatzen
%duten zenbait Runge-Kutta metodo esplizitu.
%-----

%Hasiera:
[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:)' ;

for j1=2:m+1

    [t,y]=pasork(fty,t,y,h,orden);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.
end
```

Zenbait Runge-Kutta metodotako pausoa:

```
function [tn1,yn1] = pasork(fty,tn,yn,h,orden)
%
%"s" ekuazio diferentzial arrunteko  $y'=f(t,y)$  sistema askatzeko
%zenbait Runge-Kutta metodotako pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%(tn,yn) puntuko maldaren kalkulua:

malda = feval(fty,tn,yn);          % malda = f(tn,yn).
```



```

%Runge-Kutta metodoetako formulak:

if orden==1
    k1 = h*malda;
    yn1 = yn + k1;

elseif orden==2
    k1 = h*malda;
    k2 = h*feval(fty,tn+h,yn+k1);
    yn1 = yn + (k1 + k2)/2;

elseif orden==3
    k1 = h*malda;
    k2 = h*feval(fty,tn+(h/2),yn+(k1/2));
    k3 = h*feval(fty,tn+h,yn-k1+2*k2);
    yn1 = yn + (k1 + 4*k2 + k3)/6;

elseif orden==4
    k1 = h*malda;
    k2 = h*feval(fty,tn+(h/2),yn+(k1/2));
    k3 = h*feval(fty,tn+(h/2),yn+(k2/2));
    k4 = h*feval(fty,tn+h,yn+k3);
    yn1 = yn + (k1 + 2*k2 + 2*k3 + k4)/6;

elseif orden==5 %DOPRI(5,4) metodoko 5. ordenako metodoa
    k1 = h*malda;
    k2 = h*feval(fty,tn+(h/5),yn+(k1/5));
    k3 = h*feval(fty,tn+(3*h/10),yn+(3/40)*k1+(9/40)*k2);
    k4 = h*feval(fty,tn+(4*h/5),yn+(44/45)*k1 ...
        -(56/15)*k2+(32/9)*k3);
    k5 = h*feval(fty,tn+(8*h/9),yn+(19372/6561)*k1 ...
        -(25360/2187)*k2 + (64448/6561)*k3 - (212/729)*k4);
    k6 = h*feval(fty,tn+h,yn+(9017/3168)*k1 - (355/33)*k2 ...
        +(46732/5247)*k3 + (49/176)*k4 - (5103/18656)*k5);
    k7 = h*feval(fty,tn+h,yn+(35/384)*k1 + 0*k2 +(500/1113)*k3 ...
        +(125/192)*k4 - (2187/6784)*k5+(11/84)*k6);
    yn1 = yn + (35/384)*k1 + (500/1113)*k3 ...
        +(125/192)*k4 - (2187/6784)*k5 + (11/84)*k6;
end

%Denbora eguneratzen dugu:

tn1 = tn+h;

```

7.3. EDOZEIN MOTATAKO EDAK ASKATZEKO ZENBAIT METODO INPLIZITUREN PROGRAMAZIOA

1. kapituluaren hasierako baliodun lehen ordenako $y'(t) = f(t, y(t))$, $y(t_0) = y_0$ EDAk askatzeko Euler-en metodo inplizitua ere programatu dugu. Metodoa inplizitua izatean, Newton Raphson-en metodo iteratiboa erabili dugu emaitza kalkulatzeko. Newton Raphson-en metodoak funtzio baten erroak edo zeroak kalkulatzeko balio du. $R(x) = 0$ motako ekuazio baten erroak kalkulatzeko $x^{(0)}$ hasierako balio batetik abiatuko gara eta $(\nu + 1)$ iterazioan izango dugun erroaren balioa hauxe izango da:

$$x^{(\nu+1)} = x^{(\nu)} - \Delta^{(\nu)}, \quad \nu = 0, 1, 2, \dots \quad (7.33)$$

R funtzioa eskalarra den kasuan $\Delta^{(\nu)} = \frac{R(x^{(\nu)})}{R'(x^{(\nu)})}$ izanik, eta R bektoriala den kasuan, $\Delta^{(\nu)} = \left(\frac{\partial R}{\partial x}(x^{(\nu)}) \right)^{-1} R(x^{(\nu)})$ izanik. Kasu bietarako, funtzio eskalar zein bektorialetarako erabiliko dugun nomenklatura $\Delta^{(\nu)} = \left(J(x^{(\nu)}) \right)^{-1} R(x^{(\nu)})$ izango da:

- Funtzio eskalarren kasuan, $J = R'(x)$ deribatua izanik.
- Funtzio bektorialen kasuan, $J = \frac{\partial R}{\partial x}$ jacobitarra izanik.

Newton Raphson-en iterazio-prozesua bukatutzat jotzen da zehaztu dugun iterazio kopurua bete denean edo $|\Delta^{(\nu)}|$ balioa ezarri dugun tolerantzia bat baino txikiagoa denean: $|\Delta^{(\nu)}| \leq tol$. Horrek esan nahi baitu, momentu honetatik aurrera $\Delta^{(\nu)}$ balioaren aportazioa (7.33) adierazpenean ez dela izango handia.

Atal honetan, Euler-en metodo inplizituko ereduari jarraituz beste metodo inplizitu batzuk programatu ditugu, hala nola metodo trapezoidala, Adams Moulton-en metodoak, BDF metodoak eta super-etorkizuneko puntuak darabiltzaten EBDF metodoak. Metodo horietako bakoitzean argi izan behar dugu, zer funtzioaren erroak bilatzen ari garen, R funtzioa zein den, alegia. Eta behin R zein den badakigunean, haren deribatua edo jacobitarra zein den ere zehaztu egin beharko dugu. Newton Raphson-en metodoan iterazio kopuru maximo bezala 100 finkatu dugu eta $tol = 0,001$ ezarri dugu.

7.3.1. Metodo trapezoidalaren forma orokorra

Metodo trapezoidalaren pausoaren adierazpena hau da:

$$y_{n+1} = y_n + \frac{h}{2}(f_n + f_{n+1}) \quad (7.34)$$

(7.34) ekuazioa era honetan idatziko dugu:

$$(y_{n+1} - y_n) - \frac{h}{2}(f_n + f_{n+1}) = 0 \quad (7.35)$$

Eta gure helburua (7.35) ekuazioaren erroak aurkitzea izango da. Ekuazioa inplizitua izanik, pauso bakoitzean, Newton Raphson-en metodoa aplikatuko dugu:

$$y_{n+1}^{(\nu+1)} = y_n^{(\nu)} - \left[J(y_{n+1}^{(\nu)}) \right]^{-1} R(y_{n+1}^{(\nu)}), \quad \nu = 0, 1, 2, \dots \quad (7.36)$$

Metodo trapezoidaleko R eta J hauek izanik:

$$\begin{cases} R(y_{n+1}^{(\nu)}) = (y_{n+1}^{(\nu)} - y_n) - \frac{h}{2}(f_n + f_{n+1}^{(\nu)}) \\ J(y_{n+1}^{(\nu)}) = \frac{\partial R}{\partial y}(y_{n+1}^{(\nu)}) = I - \frac{h}{2} \cdot \frac{\partial f}{\partial y}(y_{n+1}^{(\nu)}) \end{cases} \quad (7.37)$$

Hasierako iragarle bezala, aurreko pausoko balioa erabiliko dugu. Hau da, $y_{n+1}^{(0)} = y_n$ izango da 1. iterazioan erabiliko den balioa.

MATLABen hiru funtzio erabilita programatu dugu metodo hau: 'trap.m', 'hasi.m' eta 'pasotrap.m' erabilita. 'trap.m' funtzioa izango da funtzio nagusia eta bere sarrerako argumentuak izango dira: ekuazio diferentzialeko 2. parte definitzen deneko funtzioa $f(t, y)$, programazioan 'fty' deituko duguna; 2. parte horren y aldagaiarekiko deribatua $\partial f / \partial y$, programazioan 'dfy' deitu duguna; integrazio-tartea, 'tarte' deitu duguna; EDaren hasierako balioa, 'y0' deitu duguna, eta pauso kopurua 'm'. Azpian jartzen ditugu 'trap.m' eta 'pasotrap.m' funtzioak.

```

function [tsol,ysol]=trap(fty,dfty,tartea,y0,m)
%
%"s" ekuazio diferentzial arrunteko  $y'=f(t,y)$  EDA askatzeko
%metodo trapezoidala.
%-----

[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:)' ;

for j1=2:m+1

    [t,y]=pasotrap(fty,dfty,t,y,h);

    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin ysol eguneratzen dugu.

end

```

Metodo trapezoidaleko pausoa:

```

function [tn1,yn1] = pasotrap(fty,dfty,tn,yn,h)
%
%"s" ekuazio diferentzial arrunteko  $y'=f(t,y)$  sistema askatzeko
%metodo trapezoidaleko pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:

tn1 = tn+h;

s=length(yn);
I=eye(s);
%Newton Raphson metodorako tolerantzia:
tol=0.001;

yn1=yn;
%tol baino handiagoa den balio batekin hasiera ematen diogu.
dyn1=1;
maxiter=0;
while (abs(dyn1)>tol & maxiter<100)
    R=(yn1-yn) - h/2*(feval(fty,tn+h,yn1)+feval(fty,tn,yn));
    J=eye(s) - h/2*feval(dfty,tn+h,yn1);
    [L,U] = lu ( J );
    dyn1 = -U\ (L\R);
    yn1 = yn1+dyn1;
    maxiter=maxiter+1;
end

```

7.3.2. Adams Moulton-en metodoaren forma orokorra

Adams Moulton-en metodoak ere inplizituak dira eta adierazpen honen bidez emanda datoz:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=1}^k \beta_j^* f_{n+k+1-j} \quad (7.38)$$

(7.38) ekuazioa era honetan ere idatz daiteke:

$$y_{n+k} - y_{n+k-1} - h \sum_{j=1}^k \beta_j^* f_{n+k+1-j} = 0 \quad (7.39)$$

Guk (7.39) ekuazioaren erroak aurkitu beharko ditugu. Horretarako, Newton Raphson-en metodoa aplikatu dugu eta lehen bezala, hasierako iragarle bezala $y_{n+1}^{(0)} = y_n$ balioa erabiliko dugu:

$$y_{n+1}^{(\nu+1)} = y_n^{(\nu)} - \left[J(y_{n+1}^{(\nu)}) \right]^{-1} R(y_{n+1}^{(\nu)}), \quad \nu = 0, 1, 2, \dots \quad (7.40)$$

Adams Moulton-en metodoko R eta J hauek dira:

$$\begin{cases} R(y_{n+k}^{(\nu)}) = (y_{n+k}^{(\nu)} - y_{n+k-1}) - h \sum_{j=2}^k \beta_j^* f_{n+k+1-j} - h \beta_k^* f_{n+k}^{(\nu)} \\ J(y_{n+k}^{(\nu)}) = \frac{\partial R}{\partial y}(y_{n+k}^{(\nu)}) = I - h \beta_k^* \cdot \frac{\partial f}{\partial y}(y_{n+k}^{(\nu)}) \end{cases} \quad (7.41)$$

MATLABen ‘amoulton.m’ eta ‘pasoam.m’ funtzioak sortu ditugu Adams Moulton-en metodoa programatzeko. Adams Bashforth-en metodoan gertatzen den antzera, k pausoko Adams Moulton-en metodoak ere deribatuaren k balio darabiltza y_{n+k} -ren balioa lortzeko, baina kasu honetan, $t_{n+1}, t_{n+2}, \dots, t_{n+k}$ aldiuneetako $f_{n+1}, f_{n+2}, \dots, f_{n+k}$ balioak dira darabiltzanak. Balio horiek lortzeko behar diren hasierako pausoak Euler-en metodo esplizitua erabiliz kalkulatu ditugu. Hasieran, k pausoko metodoak behar dituen puntuak lortu ahala, beraien f -ak kalkulatu ditugu, eta balio horiekin ‘maldak’ izeneko matrizea osatuz joan gara. Hasiera batean, k pausoko metodorako $(k-1)$ zutabedun matrizea da ‘maldak’ izenekoa:

$$maldak = (f_{n+k-1} f_{n+k-2} \cdots f_{n+1}) \quad (7.42)$$

Eta (7.42) adierazpena duen ‘maldak’ matrizea da ‘pasoam.m’ funtzioan sartuko dena. Pausoa eman eta t_{n+k} aldiuneko y_{n+k} kalkulatzeko f_{n+k} balioa ere behar dugu, ordea. Iterazio bakoitzean iterazioa emateko erabiliko dugun $y_{n+k}^{(v)}$ puntuaren deribatua $f_{n+k}^{(v)} = f(t_{n+k}, y_{n+k}^{(v)})$ sartuko dugu ‘maldak’ izeneko matrizeko lehenengo zutabea. ‘pasoam.m’ funtzioaren barruan joango gara lehenengo zutabe hori aldatzen, eta y_{n+k} kalkulatzeko erabiliko dugun ‘maldak’ matrize osoa haxe izango da:

$$maldak = (f_{n+k}^{(v)} f_{n+k-1} f_{n+k-2} \cdots f_{n+1}) \quad (7.43)$$

```
function [tsol,ysol]=amoulton(fty,dfty,tartea,y0,m,orden)
%
% "s" ekuazio diferentzial arrunteko y'=f(t,y) EDA askatzeko
% Adams Moulton-en metodoa.
%-----

[h,tsol,ysol]=hasi(tartea,y0,m);

% Adams Moulton-eko hasiera Euler-en pausoa erabilita egiten dugu:
tn = tsol(1);
yn = ysol(1,:)' ;
% maldak = feval(fty,tn,yn);
maldak=[];
for j1=2:orden
    [tn,yn] = pasoeu(fty,tn,yn,h);
    malda = feval(fty,tn,yn);
    tsol(j1)=tn;
    ysol(j1,:)=yn';
    maldak = [malda maldak];
end

% Algoritmoa
t=tsol(orden);
y=ysol(orden,:)' ;

for j1=orden+1:m+1

    [t,y,maldak]=pasoam(fty,dfty,t,y,h,orden,maldak);

    tsol(j1)=t;      % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y';  % y'-ren balio berriarekin ysol eguneratzen dugu.
end
```

Adams Moulton-en metodoko pausoa:

```
function [tn1,yn1,maldak] = pasoam(fty,dfty,tn,yn,h,orden,maldak)

%"s" ekuazio diferentzial arrunteko y'=f(t,y) sistema askatzeko
%Adams Moulton-en pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:

tn1 = tn+h;

s=length(yn);
I=eye(s);

%Newton Raphson metodorako tolerantzia:
tol=0.001;
maldak=zeros(s,1) maldak;

yn1=yn;
dyn1=1;%tol baino handiagoa den balio batekin hasiera ematen diogu.
maxiter=0;
while (abs(dyn1)>tol & maxiter<100)
    fn1=feval(fty,tn+h,yn1);
    maldak(:,1)=fn1;
    j=feval(dfty,tn+h,yn1);
    if orden==1
        R=yn1-yn-h*maldak;
    elseif orden==2
        R=yn1-yn-(h/2)*maldak*[1;1];
    elseif orden==3
        R=yn1-yn-(h/12)*maldak*[5;8;-1];

    elseif orden==4
        R=yn1-yn-(h/24)*maldak*[9;19;-5;1];
    elseif orden==5
        R=yn1-yn-(h/720)*maldak*[251;646;-264;106;-19];
    else 'Bakarrik k=5-era arte egiten dut'

    end
```

```

%%Newton Raphson-en metodoko jacobitarra:
if orden==1
    J=I-h*j;
elseif orden==2
    J=I-h/2*j;
elseif orden==3
    J=I-5*h/12*j;
elseif orden==4
    J=I-9*h/24*j;
elseif orden==5
    J=I-251*h/720*j;

    else 'Bakarrik k=5-era arte egiten dut'

end
[L,U] = lu ( J );
dyn1 = -U\ (L\R);
yn1 = yn1+dyn1;
maxiter=maxiter+1;
end

%(tn1,yn1) puntuko maldaren kalkulua:
malda = feval(f ty,tn1,yn1);

%Deribatuen zerrendaren eguneraketa:
maldak=[maldak(:,1:orden-1)];

```

7.3.3. BDFen forma orokorra

$y' = f(t, y)$ formako EDA sistema batentzat, BDF metodoen adierazpidea hauxe da:

$$\sum_{j=0}^k \hat{\alpha}_j y_{n+j} = h f_{n+k} \quad (7.44)$$

(7.44) berdintzako batugaian t_{n+k-1} eta t_{n+k} aldiuneetako balioak eta gainerako $\{y_{n+j} : j = 0, 1, \dots, k-2\}$ balioak bereizita idatziko ditugu:

$$\hat{\alpha}_k y_{n+k} + \hat{\alpha}_{k-1} y_{n+k-1} + \sum_{j=0}^{k-2} \hat{\alpha}_j y_{n+j} - h f_{n+k} = 0 \quad (7.45)$$

Eta ordena ezberdinetako BDFetarako programazioa unifikatzeko asmoz (7.45) adierazpena era honetan idatziko dugu:

$$\hat{\alpha}_k y_{n+k} + \hat{\alpha}_{k-1} y_{n+k-1} + C \cdot Y_{n+k-2} - h f_{n+k} = 0 \quad (7.46)$$

$$\text{non: } \left\{ C = \begin{pmatrix} \hat{\alpha}_{k-2} I & \hat{\alpha}_{k-3} I & \dots & \hat{\alpha}_1 I & \hat{\alpha}_0 I \end{pmatrix}, Y_{n+k-2} = \begin{pmatrix} y_{n+k-2} \\ y_{n+k-3} \\ \vdots \\ y_{n+1} \\ y_n \end{pmatrix} \right.$$

Helburua (7.46) adierazpenaren erroak aurkitzea izango da, eta, guk hori, Newton Raphson-en metodoa erabilia egingo dugu. Kasu honetan, BDF metodoari dagozkion Newton Raphson-en metodoko R eta J hauek izango dira:

$$\left\{ \begin{aligned} R(y_{n+k}^{(\nu)}) &= \hat{\alpha}_k y_{n+k}^{(\nu)} + \hat{\alpha}_{k-1} y_{n+k-1} + C \cdot Y_{n+k-2} - h f_{n+k}^{(\nu)} \\ J(y_{n+k}^{(\nu)}) &= \frac{\partial R}{\partial y}(y_{n+k}^{(\nu)}) = \hat{\alpha}_k I - h \cdot \frac{\partial f}{\partial y}(y_{n+k}^{(\nu)}) \end{aligned} \right. \quad (7.47)$$

Lehen bezala, hasierako iragarle bezala $y_{n+1}^{(0)} = y_n$ balioa erabiliko dugu.

MATLABen ‘BDF.m’ eta ‘pasoBDF.m’ funtzioak erabilia programatu ditugu BDF metodoak. $k \geq 2$ ordenatik aurrerako BDFek behar dituzten hasierako balioak metodo trapezoidala erabilia kalkulatu ditugu. Ondoren jartzen ditugu ‘BDF.m’ eta ‘pasoBDF.m’ funtzioak.

```

function [tsol,ysol]=BDF(fty,dfty,tartea,y0,m,orden)
%
%"s" ekuazio diferentzial arrunteko  $y'=f(t,y)$  EDA askatzeko
%BDF metodoak.
%-----
s=length(y0);
I=eye(s);           %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:)' ;

%%BDF metodoari dagozkion zenbait koefiziente eta matrize
%(ordenaren arabera ezberdinak direnak):
if orden==1
    ak=1;
    ak1=-1;
    C=[];
    ykm1=[];

elseif orden==2
    ak=3/2;
    ak1=-2;
    C=1/2*I;

elseif orden==3
    ak=11/6;
    ak1=-3;
    C=[3/2*I  -1/3*I];
elseif orden==4
    ak=25/12;
    ak1=-4;
    C=[3*I  -4/3*I  1/4*I];

else 'Bakarrik k=4-ra arte egiten dut'

end

```

```
%Y{n+k-2} matrizea izateko behar ditugun pausoak emango ditugu.
%"k" ordenarako "k" puntu izan behar ditugu hurrengo pausoa eman
%ahal izateko. Balioetako bat ematen diguten y0 da, beraz, (k-1)
%puntu behar ditugu. Beraz, (k-1) pauso eman behar ditugu beste
%metodoren bat erabilita. Metodo trapezoidala aukeratu dugu
%lehenengo pauso hauek emateko. Bat ordenan azpiko for buklea
%salto egingo luke.

if orden>1
    ykm1=y0;
end

for j1=2:orden
    [t,y]=pasotrap(fty,dfty,t,y,h);

    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin tsol eguneratzen dugu.

    if j1<orden
        ykm1=[y' ykm1];
    end

end
ykm1=ykm1';

%%Gainontzeko pausoak BDF-ei dagokien pausoarekin ematen %ditugu:
for j1=orden+1:m+1
    [t,y,ykm1]=pasoBDF(fty,dfty,t,y,h,orden,ak,akl,C,ykm1);

    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin ysol eguneratzen dugu.
end
```

BDF metodoko pausoa:

```

function [tn1,yn1,ykml]= pasoBDF(fty,dfty,tn,yn,h,orden,ak,...
                                ak1,C,ykml)

%"s" ekuazio diferentzial arrunteko y'=f(t,y) sistema askatzeko
%BDF-en pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:

tn1 = tn+h;

s=length(yn);
I=eye(s);
%Newton Raphson metodorako tolerantzia:
tol=0.001;

yn1=yn;
dyn1=1;
maxiter=0;
while (abs(dyn1)>tol & maxiter<100)
    fn1=feval(fty,tn+h,yn1);
    j=feval(dfty,tn+h,yn1);
    if orden==1
        TI=ak1*yn;
    else
        TI=ak1*yn+C*ykml;
    end
    R=ak*yn1+TI-h*fn1;
    %%Newton Raphson-en metodoko jacobitarra:
    J=ak*I-h*j;
    [L,U] = lu ( J );
    dyn1 = -U\ (L\R);
    yn1 = yn1+dyn1;
    maxiter=maxiter+1;
end

%%ykml matrizearen eguneraketa:
if orden >1
    ykml=[yn; ykml(1:s*(orden-2),:)];
end

```

7.3.4. EBDFen forma orokorra

EBDF metodoek BDFak erabiliz bi iragarpen egiten dituzte, eta ondorengo zuzentzailea erabilita zuzentzen dute:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h\beta_k f_{n+k} + h\beta_{k+1} \bar{f}_{n+k+1} \quad (7.48)$$

$f_{n+k} = f(t_{n+k}, y_{n+k})$, $\bar{f}_{n+k+1} = f(t_{n+k+1}, \bar{y}_{n+k+1})$ eta $\bar{y}_{n+k+1}$ BDFarekin lortutako iragarlea izanik.

EBDF metodoko iragarleak diren BDF metodoak programatuta ditugu eta aski da zuzentzailea programatzea guztia programatuta izateko. BDF metodoaren kasuan egin dugun bezalaxe, (7.48) berdintzako batugaian t_{n+k-1} eta t_{n+k} aldiuneetako balioak eta gainerako $\{y_{n+j} : j = 0, 1, \dots, k-2\}$ balioak bereizita idatziko ditugu:

$$\alpha_k y_{n+k} + \alpha_{k-1} y_{n+k-1} + \sum_{j=0}^{k-2} \alpha_j y_{n+j} - h\beta_k f_{n+k} - h\beta_{k+1} \bar{f}_{n+k+1} = 0 \quad (7.49)$$

Eta programazioa ordena ezberdinetara orokortzeko, (7.49) adierazpena era honetan idatzita erabiliko dugu:

$$\alpha_k y_{n+k} + \alpha_{k-1} y_{n+k-1} + D \cdot Y_{n+k-2} - h\beta_k f_{n+k} - h\beta_{k+1} \bar{f}_{n+k+1} = 0 \quad (7.50)$$

$$\text{Non: } D = (\alpha_{k-2} I \quad \alpha_{k-3} I \quad \dots \quad \alpha_1 I \quad \alpha_0 I), \quad Y_{n+k-2} = \begin{pmatrix} y_{n+k-2} \\ y_{n+k-3} \\ \vdots \\ y_{n+1} \\ y_n \end{pmatrix}$$

Eta orain arte bezala, (7.50) adierazpenaren erroak Newton Raphson-en metodoa erabilita aurkituko ditugu, R eta J hauek izanik:

$$\begin{cases} R(y_{n+k}^{(v)}) = \alpha_k y_{n+k}^{(v)} + \alpha_{k-1} y_{n+k-1} + D \cdot Y_{n+k-2} - h\beta_k f_{n+k}^{(v)} - h\beta_{k+1} \bar{f}_{n+k+1} \\ J(y_{n+k}^{(v)}) = \frac{\partial R}{\partial y}(y_{n+k}^{(v)}) = \alpha_k I - h\beta_k \frac{\partial f}{\partial y}(y_{n+k}^{(v)}) \end{cases} \quad (7.51)$$

MATLABen 'EBDF.m' eta 'pasoEBDF.m' funtzioak erabilia programatu ditugu EBDF metodoak. $k \geq 2$ ordenatik aurrerako EBDFe behar dituzten hasierako balioak metodo trapezoidala erabilia kalkulatu ditugu, nahiz eta ordena altuei hasiera emateko metodorik egokiena trapezoidala ez izan.

```
function [tsol,ysol]=EBDF(fdy,dft,tartea,y0,m,orden)
%
%"s" ekuazio diferentzial arrunteko y'=f(t,y) EDA askatzeko
%EBDF metodoak.
%-----
s=length(y0);
I=eye(s);           %s dimentsioko identitate matrizea.

[h,tsol,ysol]=hasi(tartea,y0,m);

t=tsol(1);
y=ysol(1,:);

%%EBDF metodoko alfa eta beta koefizienteak.
%alfa koefizienteak AA izeneko matrizean daude eta beta
%koefizienteak BB izeneko matrizean: AA=[alf0 alf1 alf2..... ].
%Ordena honetan idatzita daude. Errenkada bakoitzean ordena
%horretako alfa_i koefizienteak daude.
AA=[-1 1 0 0 0 0 0 0;...
     5/23 -28/23 1 0 0 0 0 0;...
    -17/197 99/197 -279/197 1 0 0 0 0;...
    111/2501 -728/2501 2124/2501 -4008/2501 1 0 0 0 0];
%BB=[beta(k+1) beta(k)]; %Ordena honetan idatzita daude.
BB=[-1/2 3/2;...
     -4/23 22/23;...
     -18/197 150/197 ;...
     -144/2501 1644/2501 ];
%Y_{n+k-2} matrizea bidertzen duen D matrizea definitzen dugu:
if orden==1
    D=[];
    ykm1=[];
else
    D=AA(orden,orden-1)*I;
    for j1=3:orden
        D=[D AA(orden,orden+1-j1)*I];
    end
    ykm1=y0;
end
```

```

%%BDF metodoetako koefizienteak. Iragarpeneko bi pausoak BDF-ak
%erabilita emango baititugu.
if orden==1
    ak=1;
    ak1=-1;
    C=[];
    ykm1=[];
elseif orden==2
    ak=3/2;
    ak1=-2;
    C=1/2*I;
elseif orden==3
    ak=11/6;
    ak1=-3;
    C=[3/2*I   -1/3*I];
elseif orden==4
    ak=25/12;
    ak1=-4;
    C=[3*I   -4/3*I   1/4*I];

else 'Bakarrik k=4-ra arte egiten dut'

end

%%BDF ez lineala hasten dugun era berdinean hasten dugu:
for j1=2:orden
    %Trapezoidalarekin hasten dugu:
    [t,y]=pasotrap(fty,dfty,t,y,h);

    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin ysol eguneratzen dugu.

    if j1<orden
        ykm1=[y' ykm1];
    end
end

ykm1=ykm1';
%%Gainontzeko pausoak EBDF-ei dagokien pausoarekin ematen ditugu,
%zehaztu den ordenean:
for j1=orden+1:m+1
    [t1,y1,ykm2]=pasoBDF(fty,dfty,t,y,h,orden,ak,ak1,C,ykm1);
    [t2,y2,ykm3]=pasoBDF(fty,dfty,t1,y1,h,orden,ak,ak1,C,ykm2);
    fbk2=feval(fty,t2,y2);
    [t,y,ykm1]=pasoEBDF(fty,dfty,t,y,h,orden,D,AA,BB,ykm1,fbk2);
    tsol(j1)=t;    % t-ren balio berriarekin tsol eguneratzen dugu.
    ysol(j1,:)=y'; % y'-ren balio berriarekin ysol eguneratzen dugu.
end

```

EBDF metodoko pausoa:

```

function [tn1,yn1,ykml]= pasoEBDF(fty,dfty,tn,yn,h,orden,D,...
                                AA,BB,ykml,fbk2)
%"s" ekuazio diferentzial arrunteko y'=f(t,y) sistema askatzeko
%EBDF-en pausoa.
%tn aldiunetik tn1=tn+h aldiunerako pausoa eman nahi da.

%Denbora eguneratzen dugu:

tn1 = tn+h;

s=length(yn);
I=eye(s);
%Newton Raphson metodorako tolerantzia:
tol=0.001;

yn1=yn;
dyn1=1;
maxiter=0;
while (abs(dyn1)>tol & maxiter<100)
    fn1=feval(fty,tn+h,yn1);
    j=feval(dfty,tn+h,yn1);
    if orden==1
        TI=AA(orden,orden)*yn;
    else
        TI=AA(orden,orden)*yn+D*ykml;
    end

    R=AA(orden,orden+1)*yn1+TI-h*BB(orden,2)*fn1-h*BB(orden,1)*fbk2;
    J=AA(orden,orden+1)*I-h*BB(orden,2)*j;
    [L,U] = lu ( J );
    dyn1 = -U\(L\R);
    yn1 = yn1+dyn1;
    maxiter=maxiter+1;
end

%%ykml matrizearen eguneraketa:
if orden>1
    ykml=[yn; ykml(1:s*(orden-2),:)];
end

```


Bibliografia

- Abelman, S. eta Patidar, K. C. (2008): "Comparison of some recent numerical methods for initial-value problems for stiff ordinary differential equations", *Comput. Math. Appl.*, **55** (4), 733-744.
- Aguirregabiria, J. M. (2000): *Fisika ikasleentzako Ekuazio Diferentzial Arruntak*, Euskal Herriko Unibertsitatea, Leioa.
- Alberdi, E. (2009): "Ekuazio diferentzialak egonkortasun bila", *Elhuyar zientzia eta teknologia aldizkaria*, **258**, 49-52.
- Ascher, U. M. eta Petzold, L. R. (1998): *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia.
- Bashforth, F. eta Adams, J. C. (1883): *An attempt to test the theories of Capillary action by comparing the theoretical and measured forms of drops of fluid*, Cambridge University Press, Cambridge.
- Bogacki, P. eta Shampine, L. F. (1989): "A 3(2) pair of Runge-Kutta formulas", *Appl. Math. Lett.*, **2**, 1-9.
- Brown, P. N. eta Hindmarsh, A. C. (1986): "Matrix-free methods for stiff systems of ODEs", *SIAM J. Numer. Anal.*, **23** (3), 610-638.
- Burden, R. eta Faires, J. D. (1998): *Numerical Methods*. Brooks/Cole publishing Thomson Learning, California.
- Butcher, J. C. (2000): "Numerical methods for Ordinary Differential Equations in the 20th century", *J. Comput. Appl. Math.*, **125**, 1-29.
- , (2008): *Numerical Methods for Ordinary Differential Equations*, John Wiley & Sons, Chichester.
- Campbell, S. L. (1990): *An introduction to differential equations and their applications*, Wadsworth Pub. Co., California.
- Cash, J. R. (1980): "On the integration of stiff systems of ODEs using extended backward differentiation formula", *Numer. Math.*, **34** (3), 235-246.
- , (1981): "Second Derivative Extended Backward Differentiation formulas for the numerical integration of stiff systems", *SIAM J. Numer. Anal.*, **18** (1), 21-36.
- , (1983): "The integration of stiff initial value problems in ODEs using modified extended backward differentiation formula", *Comput. Math. Appl.*, **9** (5), 645-657.
- , (2000): "Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs", *J. Comput. Appl. Math.*, **125**, 117-130.

- Cash, J. R. eta Considine, S. (1992): "An MEBDF code for initial value problems", *ACM Trans. Math. Softw.*, **18** (2), 142-155.
- Dahlquist, G. G. (1963): "A special stability problem for linear multistep methods", *BIT*, **3** (1), 27-43.
- Dormand, J. R. eta Prince, P. J. (1980): "A family of embedded Runge-Kutta formulae", *J. Comput. Appl. Math.*, **6** (1), 19-26.
- Enright, W.H. (1974): "Second derivative multistep methods for stiff ordinary differential equations", *SIAM J. Numer. Anal.*, **11** (2), 321-331.
- Fehlberg, E. (1968): "Classical fifth, sixth, seventh and eighth order Runge-Kutta formulas with stepsize control", *NASA TR R*, **287**.
- , (1969): "Low order classical Runge-Kutta formulas with step-size control and their application to some heat transfer problems", *NASA TR R*, **315**.
- Fredebeul, C. (1998): "A-BDF: A generalization of the backward differentiation formulae", *SIAM J. Numer. Anal.*, **35** (5), 1.917-1.938.
- Gear, C. W. (1971): *Numerical initial value problems in Ordinary Differential Equations*, Prentice Hall, New Jersey.
- Golubitsky, M. eta Dellnitz, M. (2001): *Álgebra lineal y ecuaciones diferenciales con uso de Matlab*, Internacional Thomson Editores, Mexiko.
- Griffiths, D. F. eta Highman D. J. (2010): *Numerical methods for ordinary differential equations*, Springer, Londres.
- Hairer, E. eta Wanner, G. (1983): "On the instability of the BDF Formulas", *SIAM J. Numer. Anal.*, **20** (6), 1.206-1.209.
- , (1991): *Solving ordinary differential equations, II, Stiff and Differential-Algebraic Problems*, Springer, Berlin.
- Hairer, E.; Nørsett, S. P. eta Wanner, G. (1993): *Solving ordinary differential equations, I, Nonstiff problems*, Springer, Berlin.
- Heath, M. T. (1997): *Scientific Computing. An introductory survey*, Mc Graw Hill, New York.
- Hoffman, J. D. (2001): *Numerical methods for engineers and scientists*, Marcel-Dekker, New York.
- Hojjati, G.; Rahimi Ardabili, M.Y. eta Hosseini, S.M. (2004): "A-EBDF: an adaptative method for numerical solution of stiff systems of ODEs", *Math. Comput. Simul.*, **66**, 33-41.
- , (2006): "New second derivative multistep methods for stiff systems", *Appl. Math. Model.*, **30** (5), 466-476.
- Hosseini, S.M. eta Hojjati, G. (1999): "Matrix-free MEBDF method for numerical solution of systems of ODEs", *Math. Comput. Modell.*, **29**, 67-77.
- Hubbard, J. H. eta West, B. H. (1991): *Differential equations. A dynamical systems approach*, Springer-Verlag, New York.
- Ismail, G. A. F. eta Ibrahim, I. H. (1998): "A new higher effective P-C methods for stiff systems", *Math. Comput. Simul.*, **47** (6), 541-552.
- , (1999): "New efficient second derivative multistep methods for stiff systems", *Appl. Math. Model.*, **23** (4), 279-288.

- Kostelich, E. J. eta Armbruster, D. (1996): *Introductory differential equations: From linearity to chaos*, Addison Wesley, Massachusetts.
- Lambert, J.D. (1973): *Computational Methods in Ordinary Differential Equations*, John Wiley, Chichester.
- , (1991): *Numerical methods for ordinary differential systems: the initial value problem*, John Wiley & Sons, New York.
- Moulton, F. R. (1926): *New methods in exterior Ballistics*, University of Chicago Press, Chicago.
- Peterson, A. C. eta Kelley, W. G. (2001): *Difference equations. An introduction with applications*, Academic Press, New York.
- Psihoyios, G. (2006): “A block implicit advanced-step point (BIAS) algorithm for Stiff Differential Systems”, *Computing Letters*, **2**, 51-58.
- Shampine, L. F. eta Corless, R. M. (2000): “Initial value problems for ODEs in problem solving environments”, *J. Comput. Appl. Math.*, **125**, 31-40.
- Shampine, L. F.; Gladwell, I. eta Thompson, S. (2003): *Solving Odes with MATLAB*, Cambridge University Press, New York.
- Shampine, L. F. eta Reichelt, M. W. (1997): “The MATLAB ODE Suite”, *SIAM J. Sci. Comput.*, **18**, 1-22.
- Skeel, R. D. (1986): “Thirteen ways to estimate global error”, *Numer. Math.*, **48 (1)**, 1-20.
- Skeel, R. D. eta Kong. A. K. (1977): “Blended linear multistep methods”, *ACM Trans. Math. Softw.*, **3 (4)**, 326-345.
- The Mathworks Inc. (2000): *MATLAB. The language of technical computing*, Prentice-Hall Inc., New Jersey.

Sailean argitaratu diren beste liburu batzuk

Bioestatistika

Arantza Urkaregi
1991n argitaratua
ISBN: 84-86967-37-6

Kalkulu diferentziala eta integrala I

Joserra Aizpurua eta Patxi Angulo (koord.)
1992an argitaratua
ISBN: 84-86967-47-3

Kalkulu diferentziala eta integrala II

Joserra Aizpurua eta Patxi Angulo (koord.)
1992an argitaratua
ISBN: 84-86967-48-1

Ekuzio diferentzialak. Aplikazioak eta ariketak

Elena Agirre (euskaratzailea)
1994an argitaratua
ISBN: 84-86967-63-5

Optimizazioa. Programazio lineala

Victoria Fernandez eta Ana Zelaia
1992an argitaratua
ISBN: 84-86967-65-1

Estatistikarako Sarrera

Karmele Fernandez
1996an argitaratua
ISBN: 84-86967-70-8

Estatistikarako Sarrera. Ariketak

Karmele Fernandez, Josu Orbe, Marian Zubia
1997an argitaratua
ISBN: 84-86967-80-5

Estatistika I eta Estatistika II. Ariketak

Karmele Fernandez, Josu Orbe, Marian Zubia
1996an argitaratua
ISBN: 84-86967-79-1

R Hasiberrientzat

Gorka Azkune eta Yosu Yurramendi (itzul.)
2005ean PDFn argitaratua
ISBN: 84-86967-077-7

Estatistikaren oinarriak. Ariketak

Elena Agirre Basurko
2006an argitaratua
ISBN: 84-86967-079-3

Erregresio lineala, bariantza-analisiak eta hipotesi-testak. Datu-analisirako lanabesak

Xabier Isasi Balanzategi
2010ean argitaratua
ISBN: 978-84-8438-290-4

Estatistikaren Oinarriak. Ariketak (2. argitalpena)

Elena Agirre Basurko
2010ean argitaratua
ISBN: 978-84-8438-288-1

Kalkulu diferentziala eta integrala (2. argitalpena)

Patxi Angulo
2009an argitaratua
ISBN: 84-86967-079-3