METODE NUMERICE PENTRU ECUAȚII DIFERENȚIALE ORDINARE

NOTE DE CURS - versiunea 1.0, august 2014

Alexandru I. Nicolin

Departamentul de Fizică Computațională și Tehnologii Informaționale
Institutul Național de Cercetare-Dezvoltare pentru Fizică și Inginerie Nucleară „Horia Hulubei”
Măgurele, 2014

Turning to ordinary differential equations, we feel that no one understands the real nature of a differential equation $dy/dx = f(x, y)$ until he has experimented with computational or graphical procedures for its solution. Such a simple process as Euler's method, while not accurate, is both useful and worthy of study. What normally passes for numerical methods in differential equations courses bears little resemblance to how such equations are solved in practice. Why not mix realistic practice right with the theory, where they can influence and strengthen each other?

The Role of Numerical Analysis in an Undergraduate Program, George E. Forsythe, The American Mathematical Monthly **66**, 651 (1959)

Cuprins

Listă de tabele

Cuvânt liminar

Cursurile tradiționale de analiză matematică discută în partea de început aspectele generale ce țin de teoria funcțiilor continue și de limita unei funcții într-un punct, pentru ca mai apoi sa fie introdusă derivarea și integrarea funcțiilor. Până și titlul generic *Calcul diferențial și integral* induce studentului ideea că derivarea vine, într-un fel, înaintea integrării, chiar daca dezvoltarea istorica a analizei matematice înregistreaza întâi contribuțiilor remarcabile ale lui Kepler cu privire la calculul volumului butoaielor de vin cuprinse în *Nova stereometria doliorum vinariorum* (1615) și abia apoi pe acelea ale lui Newton din *Annotations from Wallis* (1665) cu privire la derivate. De fapt, predarea curentă a analizei matematice urmează firul invers al dezvoltarii ei istorice, subiectele din finalul cursurilor de profil fiind, de obicei, cele care au fost investigate cel mai devreme. Așa se face că *metodele numerice* și, într-un cadru mai larg, *analiza numerică* ajung sa fie studiate la finalul cursurilor de analiză matematică, deseori de-o maniera superficială, chiar dacă o abordare diacronică a domeniului arată că dezvoltarea acestuia ar fi fost imposibilă fără utilizarea calculului numeric.

Prezentele note de curs prezintă principalele metode folosite în determinarea soluțiilor numerice ale ecuațiilor diferențiale ordinare, îmbinând aspectele istorice (extrem de utile înțelegerii intime a domeniului) cu cele aplicative și, cumpătat, cu cele de natură teoretică. Cursul nu se adreseaza, așadar, matematicienilor și, în general, acelora care doresc o tratare riguroasă a domeniului, ci fizicienilor de toate specializările care doresc să înțeleagă la nivel calitativ proprietățile metodelor numerice folosite în studiul ecuațiilor diferențiale și să poată utiliza cu ușurință informațiile dobândite pentru rezolvarea ecuațiilor ce descriu sisteme fizice reale. Capitolele care urmează prezintă în detaliu cele mai folosite metode multipas și unipas, acceptul fiind pus pe înțelegerea proprietăților generale ale acestora (în special pe acelea legate de stabilitate și calculul erorilor) și pe particularitățile ce privesc implementarea numerică a acestora.

În final autorului mulțumește colegilor din Departamentul de Fizică Computațională și Tehnologii Informaționale al Institutului Național pentru Fizică și Inginerie Nucleară „Horia Hulubei” pentru comentariile și sugestiile lor asupra textului și studenților Facultății de Fizică a Universității din București ale căror observații au sporit considerabil claritatea textului. Mulțumirile autorului se îndreaptă, de asemenea, către Prof.

Cuvânt liminar

Univ. Dr. Virgil BĂRAN, șeful departamantului de *Fizică teoretică, Matematici, Optică, Spectroscopie, Plasmă, Laseri*, pentru oportunitatea de a preda studenților înscriși în programul masteral *Fizică teoretică și computațională* prezentele capitole de metode numerice.

Orice comentariu cu privire la prezentele note de curs pot fi trimise autorului pe adresa *alexandru.nicolin@nipne.ro*.

Alexandru Nicolin,

Măgurele, august, 2014

1 Introducere

Ideea de bază a acestui curs este că pentru o ecuație diferențială de tipul

$$y' = f(x, y), \qquad y(x_0) = y_0 \tag{1.1}$$

există două tipuri distincte de metode numerice cu ajutorul cărora obținem aproximația numerică a soluției ecuației precedente. Pentru o discretizare a lui x de tipul x_0, $x_1 = x_0 + h$, $x_2 = x_1 + h = x_0 + 2h$, $x_3 = x_2 + h = x_1 + 2h = x_0 + 3h$, etc., ambele tipuri de metode numerice determină valorile lui y pentru precedentele valori ale lui x, anume $y(x_1) = y_1$, $y(x_2) = y_2$, $y(x_3) = y_3$, etc., însă *diferă calitativ prin tipul polinomului care interpolează funcția* y. Astfel, cunoscută fiind valoarea funcției y de la punctul inițial x_0 până la un punct oarecare x_n și fiind necesară aflarea funcției în punctul x_{n+1}, ecuația (1.1) se rescrie de obicei sub forma

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} dx f(x, y(x)), \tag{1.2}$$

determinarea lui y_{n+1} depinzând acum de calculul integralei din partea dreaptă a ecuației.

Primele metode folosite pentru calculul numeric al acestei integrale porneau de la construirea unui polinom de interpolare $F(x)$, ce coincide cu f atât în x_n, cât și într-un număr de puncte anterioare lui x_n, anume x_{n-1}, x_{n-2}, etc., valoarea aproximativă a lui y_{n+1} fiind dată de

$$\begin{aligned} y_{n+1} &\approx y_n + \int_{x_n}^{x_{n+1}} dx F(x; x_n, x_{n-1}, x_{n-2}...) \\ &= y_n + h(\beta_n f_n + \beta_{n-1} f_{n-1} + \beta_{n-2} f_{n-2}....), \end{aligned} \tag{1.3}$$

unde β_n, β_{n-1}, β_{n-2}, etc., sunt coeficienți numerici a căror valoare depinde de ordinul polinomului de interpolare. Deoarece polinomul de interpolare folosit în calcul integralei este determinat folosind informații cu privire la soluția ecuației în puncte precedente lui x_n aceste metode sunt denumite în mod curent *metode multi-pas* sau *metode cu memorie*. Aceste metode apar în a doua jumătate a secolului al XIX-lea, fiind introduse într-o carte celebră scrisă de Francis Bashforth și John Couch Adams asupra acțiunii capilare (Bashforth și Adams (1883)) și sunt apoi rafinate de Forest Ray Moulton, un important astronom american, în a cărui carte *New methods in exterior ballistics* (Moulton (1926)) sunt cuprinse o serie de noi metode numerice de determinare a traiectoriei proiectilelor, dezvoltate de autor în perioada Primului Război Mondial pentru Armata Statelor

Unite ale Americii[1]. Particularitatea acestor metode numerice este aceea că polinomul de interpolare F este determinat folosind *și* punctul x_{n+1}, nu doar punctul x_n și cele precedente, valoarea aproximativă a lui y_{n+1} fiind

$$y_{n+1} \approx y_n + h\left(\beta_{n+1}f_{n+1} + \beta_n f_n + \beta_{n-1}f_{n-1} + \beta_{n-2}f_{n-2}....\right). \tag{1.4}$$

Aceste metode sunt așadar implicite, valoarea numerice a lui y_{n+1} fiind obținută după rezolvarea unei ecuații al cărei grad de neliniaritate depinde de dependența funcțională a lui f de y. Capitolul 2 al acestui volum cuprinde o discuție detaliată asupra acestor metode, fiind prezentată pe larg o serie largă de formule de calcul (atât implicite cât și explicite).

Cea de-a doua mare categorie de metode folosite pentru determinarea numerică a soluțiilor ecuației (1.1) este reprezentată de așa-numitele *metode unipas,* în cadrul cărora polinomul de interpolare F care aproximează pe f este determinat fără a folosi puncte anterioare lui x_n. Aceste metode au fost introduse într-un articol devenit acum clasic al lui Carl David Tolmé Runge în care este propusă următoare schemă pentru determinarea lui y_{n+1}

$$y_{n+1} = y_n + hf\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}f\left(x_n, y_n\right)\right). \tag{1.5}$$

După cum se observă cu ușurință schema de calcul este una explicită, partea dreaptă a ecuației precedente fiind calculată exclusiv pe baza valorii funcției y în punctul x_n. Aceste metode explicite au fost dezvoltate apoi de către Karl Heun, Martin Wilhelm Kutta, Nystrom și Erwin Fehlberg, acesta din urmă propunând o serie de metode simple de calcul al erorii locale. Un rol aparte în dezvoltarea domeniului l-a avut John C. Butcher care a introdus conceptul metodelor implicite și a schematizat metodele prin așa-numitul tabel Butcher. Cea mai simplă asemenea metodă este metoda Euler implicită

$$y_{n+1} = y_n + hf\left(x_{n+1}, y_{n+1}\right) \tag{1.6}$$

ce necesită rezolvarea unei ecuații algebrice la fiecare pas de integrare. Capitolul 3 cuprinde o prezentare detaliata a metodelor unipas, atât cele explicite, cât și cele implicite.

1.1 Limbaje de programare și sisteme de calcul

Dezvoltarea metodelor numerice a avut loc în strânsă legătură cu dezvoltarea limbajelor de programare. Astfel, primul limbaj de programare care a avut librarii importante

[1]Implicarea matematicienilor americani în Al Doilea Război Mondial este magistral surprinsă într-o recenzie (nu lipsită de critici) la cartea lui Moulton: „Toward the close of the World War, and even during the period of hostilities, every person who made any pretense of being interested in contributions to mathematics longed to secure a book on ballistics from which he could compute a trajectory; not by the old antiquated methods which, he had heard, had been entirely discarded, but by the methods that had been devised by American mathematicians who had patriotically devoted their talents to the solution of these important problems" (a se vedea *Moulton on exterior ballistics*, J. E. Rowe, Bulletin of the American Mathematical Society **34**, 229 (1928)).

dedicate rezolvării numerice a ecuațiilor diferențiale ordinare a fost FORTRAN, al cărui nume înseamnă FORmula TRANslation (System). Limbajul a fost creat în 1957 de către John Warner Backus, iar succesul metodelor numerice implementate în FORTRAN este cel mai bine ilustrat de faptul că NAG Toolbox for MATLAB, unanim recunoscută ca cea mai mare și mai vastă librarie de rutine numerice pentru MATLAB, utilieaza încă (în 2014!) implementari Fortran. Prima varianta a FORTRAN cuprindea 32 de instrucțiuni și era destul de rudimentară după standardele actuale, funcționând doar pe bază de cartele perforate. Versiunile FORTRAN II și FORTRAN III apar ambele în 1958 însă nu aduc mari modificari. Prima versiune care schimba tiparul celor anterioare este FORTRAN IV, aparută în 1962, în care sunt eliminate funcțiile dependente de arhitectura mașinii de calcul. Aparut in 1966, FORTRAN 66 schimbă radical tradiția versiunilor precedente și devine standardul în industria de profil grație numeroaselor standardizări. Versiunea FORTRAN 77, aparută în 1978, repară o parte din minusurile versiunii precedente, prin completarea multora din instrucțiunile existente. Sunt introduse, de pilda, brațele *else* si *else if* pentru instrucțiunea *if*, inexistente până atunci, se flexibilizează buclele de tip *do*, se pot defini constante, etc. Versiunile urmatoare apar sub numele Fortran, cele mai importante fiind Fortran 95 și Fortran 2003.

Apariția FORTRAN a fost urmată de apariția BASIC (Begginers All purpose Symbolic Instruction Code) în 1964, limbajul fiind dezvoltat în 1964 de către John G. Kemeny și Thomas E. Kurtz pentru studenții de la Darmouth College, SUA. Țelul acestora era acela de a oferi tuturor studenților, nu doar celor care studiau fizica și matematica, un limbaj prietenos, ușor de utilizat.

Dezvoltat în perioada 1969-1973 la AT&T Bell Labs de către Dennis Ritchie, limbajul de programare C nu poate fi desprins de dezvoltarea sistemului de operare UNIX, al carui kernel (nucleu, motor) a fost scris (după câteva încercări nereușite) în acest limbaj. Codurile numerice dezvoltate în C sunt foarte numeroase iar varietatea algoritmilor disponibili e cel mai bine ilustrată de o carte devenită acum clasică: *Numerical recipes in C. The art of scientific computing*, W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Cambridge University Press, prima ediție 1988. Bjarne Stroustrup a modificat C prin introducerea noțiunii de clase, de funcții virtuale, prin posibilitatea de suprascriere a operatorilor, de mostenire multiplă, șamd, dând astfel naștere C++. Apărut pentru prima oară în 1983 limbajul a fost folosit pentru crearea suitei Microsoft Office, a Acrobat Reader și a browserului Firefox.

Python este un limbaj de programare creat în 1989 de Guido van Rossum folosit atat pentru programarea aplicațiilor web (de către companii ca Google sau Yahoo!), cât și pentru aplicații științifice. Python suportă mai multe paradigme de programare, permițând programare imperativă, funcțională, procedurală sau *object-oriented*.

Menționăm în cele din urma Java, un limbaj de programare orientat pe obiecte, puternic tipizat, conceput de către James Gosling (cu contribuții ulterioare din partea lui Mike Sheridan și Patrick Naughton) la începutul anilor '90. Aplicațiile Java sunt compilate astfel încât să poată rula pe orice mașină virtuala Java, independent de arhitectura hardware a mașinii fizice.

Istoria metodelor numerice este, în bună măsură, și istoria limbajelor de programare, iar cine trece în revistă literatura de specialitate va observa cum majoritatea volumelor

dedicate calculului numeric în anii 80 ofereau implementari FORTRAN (și conțineau adesea un mic breviar de funcții FORTRAN alături de multe seturi de note explicative), în timp ce volumele scrise în anii 90 arată deja o preferință pentru C (și, oarecum mai rar, C++). Volumele ultimului deceniu arată deja creșterea interesului către noile limbaje, cum sunt Python și Java, pentru care există deja librării extrem de performante. Tomuri masive cum sunt *A numerical library in Java for scientists and engineers*, H. T. Lau, Chapman & Hall/CRC (2004), *A primer on scientific programming with Python*, H. P. Langtangen, Springer (2010), *Numerical methods, algorithms and tools in C#*, W. Dos Passos, CRC Press (2010), și multe, multe altele reflectă extrem clar tendința de a muta calculul științific către noile limbaje de programare.

Pe lângă limbajele de programare prezentate anterior, un rol aparte l-au avut așa-numitele API-uri, Application Programming Interfaces, care permit împărțirea memoriei pe mai multe procesoare, deschizând astfel calea spre calcul paralel și distribuit. Cel mai cunoscut API este OpenMP (Open Multi-Processing) care lucrează cu C, C++, și Fortran, și rulează pe marea majoritate a arhitecturilor de procesoare și sisteme de operare. Pe un palier complementar se situează MPI, un protocol de comunicare extrem de util în programarea paralelă și calcul de înaltă performanță.

Paragrafele precedente surprind doar o parte din dinamica calculului științific, cel mai important aspect încă nediscutat fiind evoluția sistemelor de calcul. O discuție detaliată depășește cu mult cadrul acestor note de curs[2], dar putem observa, de pildă, impactul fenomenal pe care dezvoltarea așa-numitelor *Graphical Processing Units - GPUs* l-a avut asupra calcului științific, oferind unei categorii extrem de largi de utilizatori posibilitatea de a experimenta calculul științific puternic paralelizat. Unitățile de procesare grafică sunt, de fapt, cele care au transformat calculul științific de înaltă performanță dintr-un domeniu de interes mai degraba teoretic/academic, într-un subiect de maxim interest pentru comunitatea științifică. O serie întreagă de metode extrem de cronofage (legate de tranformările integrale, algebră liniară, procesarea de imagini, șamd) au cunoscut dezvoltări uimitoare, fiind acum implementate în mod curent pe platforme de calcul științific gen MATLAB și MATHEMATICA.

1.2 Bibliografie

1. Bashforth, F., Adams, J. C., *An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluids, with an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops*, Cambridge at the University Press (1883)

2. Moulton, F. R., *New methods in exterior ballistics*, Chicago University Press (1926)

3. Goldstine, H. H, *A history of numerical analysis from the 16th through the 19th century*, Springer-Verlag (1977)

[2]Recomandăm cititorilor interesați remarcabilul volum scris de D. R. Hartree asupra *Calculating instruments and machines* în care descrie începuturile mașinilor digitale (calculatoarelor) și implementările hardware ale câtorva metode numerice (astăzi) elementare.

4. Bergin, T. J., și Gibson, R. G., *History of programming languages*, 3 volume, Addison-Wesley (1996)

5. Brezinski, C., și Wuytack, L., editori, *Numerical analysis: Historical developments in the 20th century*, Elsevier (2001)

2 Metode multipas

THE NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS
177. Theory of the Method The best method of integrating differential equations numerically is one devised by J. C. Adams; it is applicable to equations of any order, but for simplicity we shall describe its application to equations of the first order. [...]
178. Bibliographical Note Of the other methods which have been proposed for integrating differential equations, the best known is that of Runge, *Math. Ann.* **46** (1895), p. 167, improved and extended by Kutta, *Zeits. f. Math. u. Phys.* **46** (1901), p. 435.

(The calculus of observations. A treatise on numerical mathematics, E. T. Whittaker și G. Robinson, 50 Old Bailey, London (1924))

Considerate multă vreme a fi începutul și sfârșitul metodelor numerice pentru ecuații diferențiale ordinare (așa cum foarte bine reiese din citatul ales ca moto al capitolului), metodele multipas oferă o imagine extrem de intuitivă asupra modului în care putem construi soluții aproximative pentru o ecuație diferențială dată. Să presupunem, de pildă, că pentru ecuația (1.1) am putut construi o soluție aproximativă până la un punct x_n și că am pastrat, de asemenea, toate informațiile aferente punctului x_{n-1}. Știind, așadar, punctele (x_n, f_n) și (x_{n-1}, f_{n-1}) putem construi fără probleme un polinom de interpolare pentru f care ia forma

$$F(x) = \frac{x - x_n}{x_{n-1} - x_n} f_{n-1} + \frac{x - x_{n-1}}{x_n - x_{n-1}} f_n. \tag{2.1}$$

Cu el putem determina imediat soluția aproximativă a ecuației diferențiale în punctul x_{n+1}, calculând o integrală extrem de simplă, *i.e.*,

$$\begin{aligned} y_{n+1} &\approx y_n + \int_{x_n}^{x_{n+1}} dx F(x), \\ &= y_n + \int_{x_n}^{x_{n+1}} dx \left(\frac{x - x_n}{x_{n-1} - x_n} f_{n-1} + \frac{x - x_{n-1}}{x_n - x_{n-1}} f_n \right), \\ &= y_n + h \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right). \end{aligned} \tag{2.2}$$

Pentru a spori acuratețea rezultatului anterior, putem crește ordinul polinomului de interpolare (în calculul precedent, doar o dreaptă), prin adăugarea punctului (x_{n-2}, f_{n-2}). Noul polinom de interpolare (acum de ordinul doi) este dat de

$$F(x) = \frac{x-x_n}{x_{n-2}-x_n}\frac{x-x_{n-1}}{x_{n-2}-x_{n-1}}f_{n-2} + \frac{x-x_n}{x_{n-1}-x_n}\frac{x-x_{n-2}}{x_{n-1}-x_{n-2}}f_{n-1} + \frac{x-x_{n-2}}{x_n-x_{n-2}}\frac{x-x_{n-1}}{x_n-x_{n-1}}f_n, \tag{2.3}$$

iar noua soluție aproximativă este dată de

$$\begin{aligned} y_{n+1} &\approx y_n + \int_{x_n}^{x_{n+1}} dx F(x), \\ &= y_n + \frac{1}{h^2}\int_{x_n}^{x_{n+1}} dx \left[\frac{1}{2}(x-x_n)(x-x_{n-1}) - (x-x_n)(x-x_{n-2}) \right. \\ &\quad \left. + \frac{1}{2}(x-x_{n-2})(x-x_{n-1})\right], \\ &= y_n + h\left(\frac{23}{12}f_n - \frac{16}{12}f_{n-1} + \frac{5}{12}f_{n-2}\right) \end{aligned} \tag{2.4}$$

Soluția aproximativă obținută mai sus poate fi determinată mai simplu, fără a aproxima $f(x)$, scriind y_{n+1} ca o combinație liniară de f_n, f_{n-1} și f_{n-2}, și dezvoltând în serie în jurul lui x_n, anume

$$y_{n+1} \approx y_n + h\beta_1 f_n + h\beta_2 f_{n-1} + h\beta_3 f_{n-2}, \tag{2.5}$$

$$y_{n+1} \approx y_n + h\beta_1 y'_n + h\beta_2 y'_{n-1} + h\beta_3 y'_{n-2}, \tag{2.6}$$

$$\begin{aligned} y_n + hy'_n + \frac{h^2}{2}y''_n + \frac{h^3}{6}y'''_n + \mathcal{O}\left(h^4\right) &\approx y_n + h\beta_1 y'_n + h\beta_2 y'_n - h^2\beta_2 y''_n \\ &\quad + \frac{h^3}{2}\beta_2 y'''_n + h\beta_3 y'_n - 2h^2\beta_3 y''_n \\ &\quad + 2h^3\beta_3 y'''_n + \mathcal{O}\left(h^4\right). \end{aligned} \tag{2.7}$$

Egalând derivatele de ordinul 1, 2 și 3 ale lui y_n obținem sistemul

$$\beta_1 + \beta_2 + \beta_3 = 1, \tag{2.8}$$

$$-\beta_2 - 2\beta_3 = \frac{1}{2}, \tag{2.9}$$

$$\frac{\beta_2}{2} + 2\beta_3 = \frac{1}{6}, \tag{2.10}$$

a cărui soluție este dată de

$$\beta_1 = \frac{23}{12}, \quad \beta_2 = -\frac{16}{12}, \quad \beta_3 = \frac{5}{12}, \tag{2.11}$$

așadar tocmai formula (2.4). Metode similare de determinare a coeficienților sunt folosite și pentru metodele unipas de tip Runge-Kutta de ordin scăzut. Metoda nu funcționează

însă pentru metode (fie ele uni- sau multipas) de ordin înalt căci ecuațiile care rezultă necesită condiții suplimentare pentru determinarea unică a coeficienților β.

Calculul polinomului de interpolare al lui f este relativ ușor atunci când sunt două sau trei puncte, însă devine din ce în ce mai complicat odată cu creșterea numărului de puncte. O soluție simplă și ingenioasă de calcul se bazează pe diferențele divizate din tabelul de mai jos.

$$
\begin{array}{llllll}
x_n & f_n & & & & \\
x_{n-1} & f_{n-1} & \dfrac{f_n - f_{n-1}}{h} & & & \\
x_{n-2} & f_{n-2} & \dfrac{f_{n-1} - f_{n-2}}{h} & \dfrac{f_n - 2f_{n-1} + f_{n-2}}{2h^2} & & \\
x_{n-3} & f_{n-3} & \dfrac{f_{n-2} - f_{n-3}}{h} & \dfrac{f_{n-1} - 2f_{n-2} + f_{n-3}}{2h^2} & \dfrac{f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3}}{6h^3}
\end{array}
$$

Cu ajutorul lor putem scrie polinomul de interpolare Newton

$$
\begin{aligned}
F_0(x) &= f_n, && (2.12)\\
F_1(x) &= f_n + \frac{f_n - f_{n-1}}{h}(x - x_n), && (2.13)\\
F_2(x) &= f_n + \frac{f_n - f_{n-1}}{h}(x - x_n) \\
&\quad + \frac{f_n - 2f_{n-1} + f_{n-2}}{2h^2}(x - x_n)(x - x_{n-1}), && (2.14)\\
F_3(x) &= f_n + \frac{f_n - f_{n-1}}{h}(x - x_n) + \frac{f_n - 2f_{n-1} + f_{n-2}}{2h^2}(x - x_n)(x - x_{n-1}) \\
&\quad + \frac{f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3}}{6h^3}(x - x_n)(x - x_{n-1})(x - x_{n-2}). && (2.15)
\end{aligned}
$$

Să observăm că polinoamele de interpolare folosite anterior (a se vedea ecuațiile (2.1) și (2.3)) erau de tip Lagrange, adăugarea unui nou punct necesitând calculul de la zero al polinomului de interpolare și nu doar simpla adăugare a unui noi termen, așa cum se întâmplă în cazul polinoamelor de interpolare de tip Newton. Integralele necesare calculului soluțiilor aproximative se pot scrie sub forma

$$\int_{x_n}^{x_{n+1}} dx F_0(x) = h f_n, \tag{2.16}$$

$$\begin{aligned}\int_{x_n}^{x_{n+1}} dx F_1(x) &= h f_n + \frac{1}{2}\left(f_n - f_{n-1}\right), \\ &= h\left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1}\right), \end{aligned} \tag{2.17}$$

$$\begin{aligned}\int_{x_n}^{x_{n+1}} dx F_2(x) &= h\left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1}\right) + \frac{5}{12}\left(f_n - 2f_{n-1} + f_{n-2}\right), \\ &= h\left(\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\right), \end{aligned} \tag{2.18}$$

$$\begin{aligned}\int_{x_n}^{x_{n+1}} dx F_3(x) &= h\left(\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\right) + \frac{3}{8}\left(f_n - 3f_{n-1} + 3f_{n-2} - f_{n-3}\right), \\ &= h\left(\frac{55}{24} f_n - \frac{59}{24} f_{n-1} + \frac{37}{24} f_{n-2} - \frac{9}{24} f_{n-3}\right). \end{aligned} \tag{2.19}$$

Am subliniat cu roșu ultimul termen al polinomului de interpolare și al soluției aproximative pentru a sublinia ca am obținut o metodă *iterativă* cu ajutorul căreia putem construi o soluție de ordin $n+1$ pornind de la o soluție (cunoscută) de ordin n. Mai clar, modul acesta de a construi metodele multipas subliniază ideea ca o metode de ordin $n+1$ corecteaza (în sensul ameliorării preciziei) o metodă de ordin n, iar termenul de corecție este cel marcat cu roșu. Practic, dacă știm, de exemplu, aproximarea integralei

$$\int_{x_n}^{x_{n+1}} dx f(x)$$

pentru un polinom de grad doi, adica $f(x) \approx F_1(x)$, obținem aproximarea integralei pentru un polinom de gradul trei calculând doar corecția, anume

$$\begin{aligned}\int_{x_n}^{x_{n+1}} dx F_2(x) &= \int_{x_n}^{x_{n+1}} dx F_1(x) \\ &\quad + \int_{x_n}^{x_{n+1}} dx \frac{f_n - 2f_{n-1} + f_{n-2}}{2h^2}\left(x - x_n\right)\left(x - x_{n-1}\right), \end{aligned} \tag{2.20}$$

$$= h\left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1}\right) + \frac{5}{12}\left(f_n - 2f_{n-1} + f_{n-2}\right), \tag{2.21}$$

$$= h\left(\frac{23}{12} f_n - \frac{16}{12} f_{n-1} + \frac{5}{12} f_{n-2}\right). \tag{2.22}$$

La modul general putem scrie relația de recurență

$$\int_{x_n}^{x_{n+1}} dx F_s(x) \quad = \quad \int_{x_n}^{x_{n+1}} dx F_{s-1}(x) + \gamma_s \sum_{j=0}^{s} (-1)^j f_{n-j} \begin{pmatrix} s \\ j \end{pmatrix}, \qquad (2.23)$$

unde

s	1	2	3	4	5	6	7	8
γ_s	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$	$\frac{19087}{60480}$	$\frac{5257}{17280}$	$\frac{1070017}{3628800}$

Pentru o discuție detaliată asupra formulei (2.23) recomandăm Ref. [3], capitolul III.1 *Classical linear multistep formulas.*

2.1 Metode Adams-Bashforth

> After some trials a satisfactory micrometrical instrument was constructed for the measurement of the forms of drops of fluid, but my attempts to calculate their forms as surfaces of double curvature failed entirely, and my undertaking must have ended here, if I had depended upon my own resources. But at this point Professor J. C. Adams furnished me with a perfectly satisfactory method of calculating by quadratures the exact theoretical forms of drops of fluids from the Differential Equation of Laplace, an account of which he has now had the kindness to prepare for publication.
>
> *(An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluids, with an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops, F. Bashforth și J. C. Adams, Cambridge at the University Press (1883))*

Metodele derivate anterior sunt cunoscute în literatura de specialitate drep metode Adams-Bashforth, de la numele autorilor primului articol in care ele sunt introduse (a se vedea Ref. [1]). Istoria articolului (din care am extras moto-ul acestui capitol) ne arată că metoda a fost dezvoltată din nevoia practică de a descrie geometria unei picături aflate pe o suprafață solidă. Citatul ales arată că paternitatea metodei îi aparține doar lui Adams, Bashforth fiind citat mai mult datorită conjuncturii.

Folosind formula iterativa (2.23) am calculat mai jos formule Adams-Bashforth de la ordinu l, formula care se identifica cu metoda Euler, pana la ordinul 8.

$$y_{n+1} = y_n + hf_n \tag{2.24}$$

$$y_{n+1} = y_n + h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right) \tag{2.25}$$

$$y_{n+1} = y_n + h\left(\frac{23}{12}f_n - \frac{16}{12}f_{n-1} + \frac{5}{12}f_{n-2}\right) \tag{2.26}$$

$$y_{n+1} = y_n + h\left(\frac{55}{24}f_n - \frac{59}{24}f_{n-1} + \frac{37}{24}f_{n-2} - \frac{9}{24}f_{n-3}\right) \tag{2.27}$$

$$y_{n+1} = y_n + h\left(\frac{1901}{720}f_n - \frac{1387}{360}f_{n-1} + \frac{109}{30}f_{n-2} - \frac{637}{360}f_{n-3} + \frac{251}{720}f_{n-4}\right) \tag{2.28}$$

$$y_{n+1} = y_n + h\left(\frac{4277}{1440}f_n - \frac{2641}{480}f_{n-1} + \frac{4991}{720}f_{n-2} - \frac{3649}{720}f_{n-3} + \frac{959}{480}f_{n-4} - \frac{95}{288}f_{n-5}\right) \tag{2.29}$$

$$y_{n+1} = y_n + h\left(\frac{198721}{60480}f_n - \frac{18637}{2520}f_{n-1} + \frac{235183}{20160}f_{n-2} - \frac{10754}{945}f_{n-3} + \frac{135713}{20160}f_{n-4} - \frac{5603}{2520}f_{n-5} + \frac{19087}{60480}f_{n-6}\right) \tag{2.30}$$

$$y_{n+1} = y_n + h\left(\frac{16083}{4480}f_n - \frac{1152169}{120960}f_{n-1} + \frac{242653}{13440}f_{n-2} - \frac{296053}{13440}f_{n-3} + \frac{2102243}{120960}f_{n-4} - \frac{115747}{13440}f_{n-5} + \frac{32863}{13440}f_{n-6} - \frac{5257}{17280}f_{n-7}\right) \tag{2.31}$$

Determinarea metodelor Adams-Bashforth prezentată aici este una riguroasă, lipsită de artificii, dar există și derivări mai simple, lipsite însă de transparență (a se vedea în acest sens Exercițiul 1). Din punct de vedere practic pentru implementarea numerică a metodei este nevoie doar de coeficienții β. Subliniem, însă, o particularitate a acestor metode: ele necesită de obicei condiții inițiale mult mai numeroase decât este necesar pentru a avea o problemă numerică corect definită din punct de vedere matematic. Mai clar, la momentul $t = 0$ cunoaștem doar f_0 ceea ce ne împiedică să folosim metodele (2.25)-(2.31) din lipsa de condiții inițiale, singura opțiune fiind metoda de ordin 1 definită de (2.24). La pasul următor vom cunoaște atât f_0 cât și f_1, ceea ce ne permite să folosim o metodă de ordin doi, iar la pasul urmator putem folosi o metodă de ordin trei, căci cunoaștem f_2, șamd. Inițializarea metodei este principala problemă a metodelor multipas și din această cauză o bună parte din implementările existente (inclusiv cea din MATLAB) au la baza așa-numitele metode de ordin variabil ce permit schimbarea ordinului metodei de la un pas la altul.

Menționăm, de asemenea, că formulele de interpolare (2.12)-(2.15) și metodele definite de ecuațiile (2.24)-(2.31) pornesc de la presupunerea că punctele x_j sunt distribuite echidistant, *i.e.*, $x_2 - x_1 = x_3 - x_2 = x_4 - x_3 = ...$ Practica integrării numerice arată însă că această distribuție a punctelor este deseori ineficientă, căci sunt regiuni în care dinamica sistemului e minimală (*e.g.*, oscilații de frecvență foarte mică) și se poate folosi un pas de integrare relativ mare, în timp ce în alte regiuni dinamica este una complicată (*e.g.*, oscilații de frecvență foarte mare) și este necesar un pas de integrare relativ mic.

2.2 Metode Adams-Moulton

> The statement has often been made that I, being familiar with the mathematical methods used by astronomers, naturally introduced this method of computing trajectories. There seems to be a widespread belief that astronomers use it in determining the orbits of comets, in the lunar theory, and in perturbation theories. Nothing could be more remote from the facts. Astronomers as a class are as innocent of any acquaintanceship with this method of solving differential equations as most mathematicians were before the war. Probably not one out of twenty of them ever heard that such a method exists; and though the underlying principles of it have been used by a few mathematicians in computing special perturbations, the general procedure they employed has been so different in form from that used in ballistics that one would not easily detect the connection. So far as I am concerned, I did not learn the method from astronomy, but from a general survey of the various processes that are known for solving differential equations.
>
> *(New methods in exterior ballistics, F. R. Moulton, The American Mathematical Monthly* **35***, 246 (1928))*

În determinarea metodelor Adams-Bashforth prezentate anterior am folosit doar aproximări explicite ale integralelor (2.16)-(2.19) care au avantul că generează, la rândul lor, metode explicite, *i.e.*, y_{n+1} poate fi determinat imediat pe baza valorii f_n și a unei serii f_{n-1}, f_{n-2}, etc. O alta abordare des întâlnită în literatura de specialitate pornește de la aproximări implicite, *i.e.*, integralele (2.16)-(2.19) sunt aproximate folosind inclusiv f_{n+1}. Calculele fiind similare cu cele prezentate anterior nu le mai repetăm aici ci prezentăm mai jos varianta implicită a ecuațiilor (2.24)-(2.31). Aceste metode sunt cunoscute în literatură (a se vedea Ref. [2-4]) sub titulatura de metode Adams-Moulton si reprezintă o îmbunătățire semnificativă. Diferența dintre metodele Adams-Bashforth și metodele Adams-Moulton nu ține, așa cum ne-am putea imagina, de un plus de acurateță, ci de un plus de stabilitate. În cuvinte simple, stabilitatea se traduce prin nevoia unui pas de integrare foarte mic pentru a face schema numerică convergentă, iar metodele introduse F. R. Moulton[1] au avantajul de a face schemele numerice convergente pentru pași de integrare relativi mari, pași pentru care metodele Adams-Bashforth sunt

[1] Impactul contribuției lui F. R. Moulton la studiul dinamicii proiectilelor reiese cel mai bine din recenziile vremii, așa cum se poate vedea din urmatorul paragraf extras din *Moulton on exterior ballistics*, J. E. Rowe, Bulletin of the American Mathematical Society **34**, 229 (1928): „[...] The fundamental differential equations of motion of a projectile in flight have been known since the days of Euler. Up to the World War the progress in ballistics had consisted in devising approximate algebraic expressions whereby to wrench these equations into soluble form. [...] These approximations had been improved and improved, but still could not keep pace with the development of modern artillery. So finally Professor F. R. Moulton of the University of Chicago, while serving as a Major in the U. S. Army during the World War, cut the Gordian knot by going back to the Eulerian equations and solving them in their original and exact form by numerical integration. Thus he laid the cornerstone for an entirely new science of ballistics."

puternic divergente. Ca o observație de ordin general metodele implicite sunt mult mai stabile numeric decât cele explicite în implementarea lor necesită un efort computațional sporit.

Înainte de a prezenta formulele propriu-zise, subliniem că acestea nu au legătură cu dinamica corpurilor cerești, acesta fiind și sensul moto-ului acestui capitol. Parcursul biografic al lui Moulton este în mod vădit unul exotic, însă formulele care urmează se datorează aproape exclusiv studiilor balistice făcute în perioada primului război mondia (a se vedea în acest sens A. Gluchoff, *Artillerymen and mathematicians: Forest Ray Moulton and changes in American exterior ballistics, 1885–1934*, Historia Mathematica **38**, 506 (2011)).

$$y_{n+1} = y_n + h f_{n+1} \tag{2.32}$$

$$y_{n+1} = y_n + h\left(\frac{1}{2}f_{n+1} + \frac{1}{2}f_n\right) \tag{2.33}$$

$$y_{n+1} = y_n + h\left(\frac{5}{12}f_{n+1} + \frac{8}{12}f_n - \frac{1}{12}f_{n-1}\right) \tag{2.34}$$

$$y_{n+1} = y_n + h\left(\frac{9}{24}f_{n+1} + \frac{19}{24}f_n - \frac{5}{24}f_{n-1} + \frac{1}{24}f_{n-2}\right) \tag{2.35}$$

$$y_{n+1} = y_n + h\left(\frac{251}{720}f_{n+1} + \frac{323}{360}f_n - \frac{11}{30}f_{n-1} + \frac{53}{360}f_{n-2} - \frac{19}{720}f_{n-3}\right) \tag{2.36}$$

$$y_{n+1} = y_n + h\left(\frac{95}{288}f_{n+1} + \frac{1427}{1440}f_n - \frac{133}{240}f_{n-1} + \frac{241}{720}f_{n-2} - \frac{173}{1440}f_{n-3} + \frac{3}{160}f_{n-4}\right) \tag{2.37}$$

$$y_{n+1} = y_n + h\left(\frac{19087}{60480}f_{n+1} + \frac{2713}{2520}f_n - \frac{15487}{20160}f_{n-1} + \frac{586}{945}f_{n-2} - \frac{6737}{20160}f_{n-3} + \frac{263}{2520}f_{n-4} - \frac{863}{60480}f_{n-5}\right) \tag{2.38}$$

$$y_{n+1} = y_n + h\left(\frac{5257}{17280}f_{n+1} + \frac{139849}{120960}f_n - \frac{4511}{4480}f_{n-1} + \frac{123133}{120960}f_{n-2} - \frac{88547}{120960}f_{n-3} + \frac{1537}{4480}f_{n-4} - \frac{11351}{120960}f_{n-5} + \frac{275}{24192}f_{n-6}\right) \tag{2.39}$$

Pentru a înțelege de ce metodele implicite necesită un efort computațional sporit vom scrie explicit metoda Adams-Moulton de ordin 2 definită în ecuația (2.33)

$$y_{n+1} = y_n + h\left(\frac{1}{2}f\left(x_{n+1}, y_{n+1}\right) + \frac{1}{2}f\left(x_n, y_n\right)\right)$$

care definește acum o ecuație algebrică implicită în y_{n+1} ce trebuie rezolvată pentru fiecare x_n! Daca dependența lui f de y este, de pildă, una polinomială, soluția numerică a ecuației precedente nu este foarte dificilă, însă dacă dependența este una mai complicată, sa zicem de tip $\sin(y_n)$, ecuația care rezultă este una transcendentă iar soluția numerică

poate fi puternic netrivială. Ca o paranteză tematică menționăm următoarele: soluția numerică a unei ecuații diferențiale nu este, în mod standard, un subiect ce ține de calculul de înaltă performanță, în engleză *High-Performance Computing (HPC)*, pentru ca nu lasă loc paralelizării. Metodele implicite însă sunt potrivite pentru calculul de înaltă performanță deoarece atunci când vor fi folosite pentru sisteme mari de ecuații diferențiale ordinare puternic neliniare, să zicem, de ordinul miilor și zecilor de mii, soluția ecuației algebrice ce trebuie să fie rezolvată la fiecare pas e o problemă în sine, una care se pretează la calcul paralel atât pe sisteme cu mai multe core-uri cât și pe sisteme de tip *Graphical Processing Unit (GPU)*.

Mai menționăm în final un aspect deseori neglijat, dar extrem de important: prin creșterea exagerată a ordinului de marime al polinomului de interpolare apare așa-numitul fenomen Runge, *i.e.*, oscilații puternice (complet nefizice) la capetele domeniului de interpolare atunci când sunt folosite interpolări polinomiale de ordin ridicat. Fenomenul este puternic înrudit cu fenomenul Gibbs care se manifestă prin apariția unor oscilații puternice în spectrul Fourier al unei unde neliniare periodice formată din funcții treaptă. În mod deloc surprinzător metodele implicite atenuează oscilațiile puternice înrudite cu fenomenul Gibbs, dar concluzia generală rămâne validă: metodele de ordin înalt nu sunt în mod obligatoriu mai precise numeric.

2.3 Implementări software disponibile

Integratorii numerici de tip Adams-Bashforth și Adams-Moulton sunt implementați în MATLAB, dar nu și în MATHEMATICA. În MATLAB integratorul de tip Adams este *ode113*, care are la bază o metoda Adams-Bashforth-Moulton de ordin variabil. Integratorul acceptă un set standardizat de parametri care includ eroarea relativă (*RelTol*) și eroarea absolută (*AbsTol*), pasul inițial de integrare (*InitialStep*), numărul maxim de pași de integrare (*MaxStep*) și, foarte important, matricea Jacobi a sistemului de ecuații diferențiale (*Jacobian* și *JPattern*). Această din urmă opțiune este extrem de utilă în cazul ecuațiilor rigide și sporește considerabil eficiența integratorilor numerici.

În afara implementării din MATLAB cele mai cunoscute coduri dedicate metodelor de tip Adams sunt acelea dezvoltate de Numerical Algorithms Group - NAG. Cea mai importantă rutină a NAG este D02GFF care implementează o metodă Adams de ordin variabil, de pas variabil, dedicată cu precăderelor sistemelor nerigide. Metoda este disponibila în Fortran, varianta în C purtând eticheta D02QFC.

2.4 Exerciții

Exercițiul 1: (Adaptare după *An approach to the Adams-Bashforth formula*, M. Ritter, The Mathematical Gazette **63**, 48 (1979)) Considerând că $F(x)$ este un polinom de gradul n iar $F(x+h)$ se poate scrie ca

$$F(x+h) - F(x) \quad = \quad h\left(B_0 F'(x) + B_1 F'(x-h) + ... + B_{N-1} F'(x-(n-1)h)\right), \qquad \text{(E.2.1)}$$

să se arate că dezvoltarea în serie Taylor a funției F și a derivatelor sale și conduce la urmatorul sistem de ecuații

$$\begin{cases} B_0 + B_1 + ... + B_n = 1 \\ \begin{bmatrix} 1 & 2 & 3 & ... & ... & n-1 \\ 1^2 & 2^2 & 3^2 & ... & ... & (n-1)^2 \\ 1^3 & 2^3 & 3^3 & ... & ... & (n-1)^3 \\ ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... \\ 1^{n-1} & 2^{n-1} & 3^{n-1} & ... & ... & (n-1)^{n-1} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ ... \\ ... \\ B_{n-1} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{3} \\ -\frac{1}{4} \\ ... \\ ... \\ -\frac{(-1)^{n-1}}{n} \end{bmatrix} \end{cases} \tag{E.2.2}$$

Să se determine soluția numerică a ecuațiilor din sistemul (E.2.2) pentru $n = 1, 2, 3, 4$ și 5, și să se arate că ecuația (E.2.1) descrie metodele Adams-Bashforth de ordinul 1, 2, 3, 4 și 5.

2.5 Bibliografie

1. Bashforth, F., și Adams, J. C., *An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluids, with an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops*, Cambridge at the University Press (1883)
2. Butcher, J. C., *Numerical methods for ordinary differential equations*, John Wiley & Sons (2008)
3. Hairer, E., Nørsett, S.P., Wanner, G., *Solving ordinary differential equations I. Nonstiff Problems*, Springer (2008)
4. Dormand, J.R., *Numerical methods for differential equations. A computational approach*, CRC Press (1996)
5. Shampine, L.F., Gladwell, I., și Thompson, S., *Solving ODEs with Matlab*, Cambridge University Press (2003)

3 Metode unipas

Metodele prezentate în capitolul precedent se bazează pe determinarea soluției numerice folosind un polinom de interpolare global cu ajutorul căruia determinăm soluția unei ecuații diferențiale la x_n utilizând informațiile de la x_{n-1}, x_{n-2}, x_{n-3}, etc. Deoarece interpolarea este globală, ar putea părea firesc ca aceste metode să fie prezentate după metodele de Runge-Kutta care determină soluția la x_n folosind exclusiv informația la x_{n-1}. Numeroasele volume care preferă această abordare au în vedere faptul că metodele unipas sunt metode fără memorie, în timp ce metodele de tip Adams sunt metode cu memorie care necesită, de obicei, atenție sporită la implementarea numerică. Prezentând metodele multipas înaintea celor unipas vrem să subliniem diferențele calitative dintre polinoamele de interpolare. Astfel, chiar dacă metodele multipas se bazează pe o interpolare globală, aceasta este una foarte simplă, polinomială, în timp ce interpolarea locala a metodelor unipas este, cum vom vedea, mult mai complicată, aspect care motivează prezentarea acestora după metodele multipas.

În cele ce urmează vom prezenta metoda Euler și diferitele clase de metode Euler. Discuția asupra metodelor Runge-Kutta pornește de la metodele Runge-Kutta explicite, folosite în general pentru majoritatea problemelor care nu au probleme deosebite de stabilitate, și continua apoi cu metodele implicite. Acestea sunt folosite cu precădere pentru așa-numitele probleme rigide, *i.e.*, pentru determinarea numerică a soluțiilor ecuațiilor diferențiale ale căror soluții prezintă atât oscilații de frecvență foarte mică, cât și oscilații de frecvență foarte mare. Capitolul se încheie cu o discuție detaliată asupra metodelor Runge-Kutta împerecheate cu ajutorul cărora obținem controlul optim al erorii.

Lăsând la o parte aspectele tehnice mesajul principal al acestui capitol este că metodele de tip Runge-Kutta cunosc o diversitate mult mai mare decât metodele de tip Adams, fiind utilă pentru o clasă mult mai largă de probleme, numele unui articol classic al lui J. B. Rosser din 1967 fiind chiar *A Runge-Kutta for all seasons* (SIAM Review **9**, 417 (1967)). Acest aspect este reflectat și în literatura de profil, extrem de amplă în ceea ce privește metodele. Opțiunile bibliografice privitoare la metodele Runge-Kutta sunt extrem de variate, unii autori optând pentru vaste articole de sinteză, în timp ce alții optează pentru cărți de referință. În ceea ce ne privește am optat pentru autori clasici, cu contribuții recunoscute internațional la dezvoltarea domeniului, ale căror cărți sunt unanim recunoscute. Numele cel mai des întâlnit în paginele ce urmează este al Prof. John C. Butcher, Profesor de Matematică la University of Auckland, care introduce în anii '60 metodele Runge-Kutta implicite și tabelul care îi poartă numele cu ajutorul căruia se pot transmite într-o formă concisă toate informațiile privitoare la o anumită metodă. Istoria metodelor Runge-Kutta scrisă de Prof. Butcher, Ref. [6] a acestui capitol, a devenit un text de referință, în timp ce tratatul *Numerical methods for ordinary*

differential equations (a se vedea Ref. [7] a acestui capitol) a devenit o carte clasică. Munca Prof. Butcher a fost continuată de E. Hairer, S. P. Nørsett și G. Wanner ale căror două volume dedicate *Solving ordinary differential equations* (a se vedea Ref. [8] și Ref. [9]) sunt, pentru foarte multe probleme, referința ultimă în domeniu.

3.1 Metoda Euler

Cea mai simple dintre metodele folosite pentru rezolvarea numerică a unei ecuații diferențiale ordinare este metoda Euler, pe care am întâlnit-o deja în capitolul precedent (a se vedea ecuația (2.24)). Ideea din spatele metodei este extrem de simpla: soluția ecuației diferențiale este aproximată de o linie polinomială formată din tangentele la soluția locală a ecuației. Metoda a fost introdusă de Leonhard Euler în volumul *Institutionum calculi integralis* (publicat în perioada 1768–1770). Astfel, dată fiind ecuația diferențială

$$y' = f(x, y) \qquad y(x_0) = y_0 \tag{3.1}$$

soluția este construită ca

$$\begin{aligned}
y(x_0 + h) &= y(x_0) + hf(x_0, y_0), && (3.2)\\
y(x_0 + 2h) &= y(x_0 + h) + hf(x_0 + h, y_0(x + h)), && (3.3)\\
y(x_0 + 3h) &= y(x_0 + 2h) + hf(x_0 + 2h, y_0(x + 2h)), && (3.4)\\
&\ldots && (3.5)\\
y(x_0 + Nh) &= y(x_0 + (N-1)h) + hf(x_0 + (N-1)h, y_0(x + (N-1)h)). && (3.6)
\end{aligned}$$

Efortul computațional al metodei Euler este minim, dar precizia este de asemenea minima.

3.2 Metode Runge-Kutta

> Runge-Kutta used to be what you used when (i) you didn't know any better, or (ii) you had an intransigent problem where Bulirsch-Stoer was failing, or (iii) you had a trivial problem where computational efficiency was of no concern. However, advances in Runge-Kutta methods, particularly the development of higher-order methods, have made Runge-Kutta competitive with the other methods in many cases. Runge-Kutta succeeds virtually always.
>
> *(Numerical recipes The art of scientific computing, W. H. Press, S. A. Teukolsky, W. T. Vetterling și B. P. Flannery, Cambridge University Press (2007))*

Prima extensie a metodei Euler este așa-numită metodă a punctului de mijloc, pentru care, data fiind ecuația diferențială *autonoma*

$$y' = f(x), \qquad y(x_0) = y_0, \tag{3.7}$$

cu soluția formală

$$y(x) = y_0 + \int_{x_0}^{x} f(x), \tag{3.8}$$

aproximăm soluția numerică pe baza seriei

$$\begin{aligned} y(x_0 + h) &\approx y_0 + hf\left(x + \frac{h}{2}\right), && (3.9)\\ y(x_1 + h) &\approx y_1 + hf\left(x_1 + \frac{h}{2}\right), && (3.10)\\ &... && (3.11)\\ y(x) &\approx y_{N-1} + hf\left(x_{N-1} + \frac{h}{2}\right). && (3.12) \end{aligned}$$

Pornind de la această aproximare a soluției pentru sisteme dinamice *autonome,* vom considera mai jos ecuația *neautonoma*

$$\dot{y} = f(x, y), \qquad y(x_0) = y_0, \tag{3.13}$$

a carei soluții numerice este construită după schema

$$y(x_0 + h) \approx y_0 + hf\left(x_0 + \frac{h}{2}, y\left(x_0 + \frac{h}{2}\right)\right).$$

Această schemă se poate scrie sub forma

$$\begin{aligned} k_1 &= f(x_0, y_0), && (3.14)\\ k_2 &= f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2}k_1\right), && (3.15)\\ y_1 &= y_0 + hk_2, && (3.16) \end{aligned}$$

care este, după vom vedea în paginile următoare, forma standard de scriere a unei metode Runge-Kutta. Pentru a ne convinge că această metodă este de ordinul 2, vom scrie întâi serie Taylor a lui y_1 ca funcție de h

$$\begin{aligned} y_1 &= y_0 + hf\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2}f(x_0, y_0)\right), && (3.17)\\ &= y_0 + hf(x_0, y_0) + \frac{h^2}{2}(f_x + f_y f)(x_0, y_0)\\ &\quad + \frac{h^3}{8}\left(f_{xx} + 2f_{xy}f + f_{yy}f^2\right)(x_0, y_0) + \mathcal{O}\left(h^4\right), && (3.18) \end{aligned}$$

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \dots & b_{s-1} & b_s
\end{array}
$$

Tabel 3.1: Metode Runge-Kutta explicite. Tabel Butcher general.

iar apoi serie Taylor a lui y ca funcție de x, anume

$$
\begin{aligned}
y(x_0+h) &= y_0 + hf(x_0,y_0) + \frac{h^2}{2!}(f_x + f_y f)(x_0,y_0) \\
&\quad + \frac{h^3}{3!}\left(f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y f_x + f_y^2 f\right) + \mathcal{O}\left(h^4\right). \qquad (3.19)
\end{aligned}
$$

Din diferența $y(x_0+h) - y_1 = \mathcal{O}\left(h^3\right)$ reiese clar că metoda precedentă este precisă până la termeni de ordinul lui h^3, fiind așadar o metodă de ordinul 2.

Forma generală sub care este scrisă o metode Runge-Kutta este

$$
\begin{aligned}
k_1 &= f(x_0, y_0), && (3.20a)\\
k_2 &= f(x_0 + c_2 h, y_0 + h a_{21} k_1), && (3.20b)\\
k_3 &= f(x_0 + c_3 h, y_0 + h(a_{31}k_1 + a_{32}k_2)), && (3.20c)\\
&\dots && (3.20d)\\
k_s &= f(x_0 + c_s h, y_0 + h(a_{s1}k_1 + a_{s2}k_2 + \dots + a_{s,s-1}k_{s-1})), && (3.20e)\\
y_1 &= y_0 + h(b_1 k_1 + b_2 k_2 + \dots + b_s k_s), && (3.20f)
\end{aligned}
$$

ecuațiile precedente putând fi comprimate într-un tabel de forma de mai jos, numit în general tabel Butcher.

3.2.1 Metode Runge-Kutta explicite

Forma generală a metodelor Runge-Kutta de ordin 2, *i.e.*, cu erori locale de trunchiere de $\mathcal{O}\left(h^3\right)$, și cu doi pași, *i.e.*, cu două valori k, este prezentată în tabelul Butcher (3.2). În general vorbim de „metoda de ordin 2", dar așa cum se observă în tabel avem o

0		
c_2	c_2	
	$1 - \frac{1}{2c_2}$	$\frac{1}{2c_2}$

Tabel 3.2: Metoda Runge-Kutta explicită (generală) cu $s = p = 2$ și c_2 parametru liber.

întreagă clasă de metode, fiecare metodă fiind definită de valoarea lui c_2. De exemplu, metoda definita de ecuațiile (3.14)-(3.16) se poate obține din tabelul Butcher (3.2) pentru $c_2 = 0.5$.

Parte din arta dezvoltării metodelor Runge-Kutta ține de calculul tabelelor Butcher. Pentru metodele de ordin scăzut se poate aplica un raționament similar celui din jurul ecuațiilor (3.17)-(3.19). Ideea generală este aceea de a folosi seriile Taylor ale lui y ca funcție de x și h pentru a identifica parametrii unei metode de ordin dat. Astfel, pentru o metodă în trei pași de ordinul trei, *i.e.*, $s = p = 3$, obținem (din egalarea seriilor Taylor) urmatoarele patru ecuații algebrice

$$b_1 + b_2 + b_3 = 1, \tag{3.21}$$
$$b_2 c_2 + b_3 c_3 = \frac{1}{2}, \tag{3.22}$$
$$b_2 c_2^2 + b_3 c_3^2 = \frac{1}{3}, \tag{3.23}$$
$$b_3 a_{32} c_2 = \frac{1}{6}. \tag{3.24}$$

Rezolvarea analitică a acestor ecuații definește familia metodelor Runge-Kutta de ordin și număr de pași cunoscuți. Astfel, pentru sistemul definit de ecuațiile (3.21)-(3.24) avem trei mari clase de metode. Prima clasă de metode cuprinde soluțiile pentru care $c_2 \neq c_3$, $c_2 \neq \frac{2}{3}$, $c_2 \neq c_3 \neq 0$. Cea de-a doua clasă de metode cuprinde metodele pentru care $c_2 = c_3 = \frac{2}{3}$, $b_3 \neq 0$, și e definită de tabelul Butcher , în timp ce ultima clasă de soluții cuprinde metodele pentru care $c_2 = \frac{2}{3}$, $c_3 = 0$, $b_3 \neq 0$.

Problema determinării metodelor Runge-Kutta de ordin superior este că, spre deosebire de metodele Adams-Bashforth si Adams-Moulton, acestea nu se pot obține inductiv, folosind informațiile de la o metodă de ordin inferior, fiind necesară determinarea lor de la zero. Astfel, calculele facute pentru clasa de metode Runge-Kutta cu $s = p = 3$ sunt irelevante pentru metodele cu $s = p = 4$. Acestea din urmă sunt definite de sistemul de ecuații algebrice

0			
c_2	c_2		
c_3	$\frac{c_3\left(3c_2-3c_2^2-c_3\right)}{c_2\left(2-3c_2\right)}$	$\frac{c_3\left(c_3-c_2\right)}{c_2\left(2-3c_2\right)}$	
	$\frac{-3c_3+6c_2c_3+2-3c_2}{6c_2c_3}$	$\frac{3c_3-2}{6c_2\left(c_3-c_2\right)}$	$\frac{2-3c_2}{6c_3\left(c_3-c_2\right)}$

Tabel 3.3: Metoda Runge-Kutta explicită (generală) cu $s = p = 3$. Clasa I de soluții corespunde metodelor pentru care $c_2 \neq 0 \neq c_3 \neq c_2 \neq \frac{2}{3}$.

0			
$\frac{2}{3}$	$\frac{2}{3}$		
$\frac{2}{3}$	$\frac{2}{3}-\frac{1}{4b_3}$	$\frac{1}{4b_3}$	
	$\frac{1}{4}$	$\frac{3}{4}-b_3$	b_3

Tabel 3.4: Metoda Runge-Kutta explicită (generală) cu $s = p = 3$. Clasa II de soluții corespunde metodelor pentru care $c_2 = c_3 = \frac{2}{3}$, $b_3 \neq 0$.

0			
$\frac{2}{3}$	$\frac{2}{3}$		
0	$-\frac{1}{4b_3}$	$\frac{1}{4b_3}$	
	$\frac{1}{4}-b_3$	$\frac{3}{4}$	b_3

Tabel 3.5: Metoda Runge-Kutta explicită (generală) cu $s = p = 3$. Clasa III de soluții corespunde metodelor pentru care $c_2 = \frac{2}{3}$, $c_3 = 0$, $b_3 \neq 0$.

$$b_1 + b_2 + b_3 + b_4 = 1 \tag{3.25}$$
$$b_1 c_1 + b_2 c_2 + b_3 c_3 + b_4 c_4 = \frac{1}{2} \tag{3.26}$$
$$b_1 c_1^2 + b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \frac{1}{3} \tag{3.27}$$
$$b_3 a_{32} c_2 + b_4 \left(a_{42} c_2 + a_{43} c_3\right) = \frac{1}{6} \tag{3.28}$$
$$b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \frac{1}{4} \tag{3.29}$$
$$b_3 c_3 a_{32} c_2 + b_4 c_4 \left(a_{42} c_2 + a_{43} c_3\right) = \frac{1}{8} \tag{3.30}$$
$$b_3 a_{32} c_2^2 + b_4 \left(a_{42} c_2^2 + a_{43} c_3^2\right) = \frac{1}{12} \tag{3.31}$$
$$b_4 a_{43} a_{32} c_2 = \frac{1}{24} \tag{3.32}$$

ale cărui soluții sunt mult mai dificile decât cele ale ecuațiilor (3.21)-(3.24). Soluțiile ecuațiilor (3.25)-(3.32) au fost determinate pentru prima oară de M. W. Kutta care le-a împărțit în cinci clase distincte, fiecare având mai mulți parametrii liberi. Doua dintre metodele determinate de Kutta au devenit extrem de populare în literatura de specialitate, anume *metoda Runge-Kutta* (cunoscută în engleză ca *the Runge-Kutta method*, prezentată în tabelul Butcher (3.6)) și „regula 3/8" (prezentată în tabelul Butcher (3.7)). Pentru o discuție detaliată asupra metodelor Runge-Kutta de ordin 4 recomandăm Ref. [7], secțiunea 3.2 *Low order explicit methods*, subsecțiunea 3.2.2 *Methods of order 4*. În alegerea unei metode contează povara computațională a evaluărilor funcției f. Din acest motiv metoda din tabelul Butcher (3.6) a devenit *metoda Runge-Kutta*: având trei elemente nule în tabelul Butcher necesită cu 30% mai puține evaluări decât metoda (3.7) la aceeași acuratețe numerică.

În articolele de pionierat asupra metodele Runge-Kutta de la începutul secolului XX ecuațiile algebrice care descriu o anumită metodă erau determinate manual, prin calcule laborioase care duceau, cum vom vedea, la unele erori. Astăzi însă determinarea ecuațiilor se face automat, grație așa-numiților *arbori Runge-Kutta* (a se vedea Ref. [7], secțiunea 3.1 *Order conditions*). Deoarece ecuațiile algebrice care definesc metodele de ordin înalt sunt extrem de complicate prezentăm mai jos doar tabelele Butcher pentru două din metodele de ordin 5 introduse de Kutta, (3.8) și (3.9). A doua din aceste metode este însă greșită, ecuațiile algebrice asociate fiind însă rezolvate incorect, varianta corectă a tabelului Butcher fiind publicată de Nyström în 1925, tabel pe care noi îl reproducem în (3.10).

3.2.2 Metode Runge-Kutta implicite

Cea mai simplă definiție a metodelor Runge-Kutta implicite este că acestea sunt acele metode Runge-Kutta care au tabele Butcher cu elemente ne-nule deasupra diagonalei.

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

Tabel 3.6: *Metoda Runge-Kutta* de ordin 4.

0				
$\frac{1}{3}$	$\frac{1}{3}$			
$\frac{2}{3}$	$-\frac{1}{3}$	1		
1	1	-1	1	
	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

Tabel 3.7: Metoda Runge-Kutta de ordin 4, cunoscută drept „regula 3/8”.

0						
$\frac{1}{5}$	$\frac{1}{5}$					
$\frac{2}{5}$	0	$\frac{2}{5}$				
1	$\frac{9}{4}$	-5	$\frac{15}{4}$			
$\frac{3}{5}$	$-\frac{63}{100}$	$\frac{9}{5}$	$-\frac{13}{20}$	$\frac{2}{25}$		
$\frac{4}{5}$	$-\frac{6}{25}$	$\frac{4}{5}$	$\frac{2}{15}$	$\frac{8}{75}$	0	
	$\frac{17}{144}$	0	$\frac{25}{36}$	$\frac{1}{72}$	$-\frac{25}{72}$	$\frac{25}{48}$

Tabel 3.8: Metodă Runge-Kutta de ordin 5 determinată de Kutta.

0						
$\frac{1}{3}$	$\frac{1}{3}$					
$\frac{2}{5}$	$\frac{4}{25}$	$\frac{6}{25}$				
1	$\frac{1}{4}$	-3	$\frac{15}{4}$			
$\frac{2}{3}$	$\frac{6}{81}$	$\frac{90}{81}$	$-\frac{50}{81}$	$\frac{8}{81}$		
$\frac{4}{5}$	$\frac{7}{30}$	$\frac{18}{30}$	$-\frac{5}{30}$	$\frac{4}{30}$	0	
	$\frac{48}{192}$	0	$\frac{125}{192}$	0	$-\frac{81}{192}$	$\frac{100}{192}$

Tabel 3.9: Metodă Runge-Kutta de ordin 5 determinată de Kutta. Metoda este incorectă, variantă corectă a coeficienților fiind determintă de Nyström în 1925 (a se vedea tabelul Butcher (3.10)).

0						
$\frac{1}{3}$	$\frac{1}{3}$					
$\frac{2}{5}$	$\frac{4}{25}$	$\frac{6}{25}$				
1	$\frac{1}{4}$	-3	$\frac{15}{4}$			
$\frac{2}{3}$	$\frac{2}{27}$	$\frac{10}{9}$	$-\frac{50}{81}$	$\frac{8}{81}$		
$\frac{4}{5}$	$\frac{2}{25}$	$\frac{12}{25}$	$\frac{2}{15}$	$\frac{8}{75}$	0	
	$\frac{23}{192}$	0	$\frac{125}{192}$	0	$-\frac{27}{64}$	$\frac{125}{192}$

Tabel 3.10: Metodă Runge-Kutta de ordin 5 corectată de Nyström.

Evident, definiția este prea puțin utilă așa că vom încerca mai jos o abordare diferită. Ideea fundamentală este următoarea: sistemele algebrice care descriu metodele Runge-Kutta atunci când niciun element a_{ij} nu este în mod obligatoriu zero sunt imposibil de rezolvat fără ipoteze simplificatoare. Altfel spus, fără a veni cu ipoteze sau presupuneri cu privire la valorile numerice ale unor elemente din tabelul Butcher, nu putem obține soluții de genul celor sistematizate în tabelele Butcher (3.3)-(3.5).

În paginile care urmează vom prezenta cea mai importantă clasă de metode Runge-Kutta implicite ce are la baza cuadraturile gaussiene.

Cuadraturi gaussiene Această metodă are la bază polinoamele Legendre definite pe intervalul $[0, 1]$ definite ca

$$P_0^* = 1, \tag{3.33}$$
$$P_1^* = 2x - 1, \tag{3.34}$$
$$P_2^* = 6x^2 - 6x + 1, \tag{3.35}$$
$$P_3^* = 20x^3 - 30x^2 + 12x - 1. \tag{3.36}$$

Cu ajutorul lor putem construi o metode Runge-Kutta de ordin $p = 2$ folosind doar un singur pas, *i.e.*, $s = 1$, folosind pentru c rădăcina ecuației $P_1^* = 0$, *i.e.*, $c = \frac{1}{2}$. Metoda este guvernată de tabelul Butcher

Observăm imediat că metoda este foarte similară metodei implicite Adams-Moulton de ordin 2 (2.33) și că nu necesita algorimti de calcul dedicați. Lucrurile se schimbă radical atunci când considerăm o metodă Runge-Kutta de ordin $p = 4$ folosind doar doi pași, *i.e.*, $s = 2$. Să observăm mai întâi că un asemenea nivel de precizie într-un număr așa de

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

Tabel 3.11: Metodă Runge-Kutta bazată pe cuadraturi gaussiene cu $s = 1$ și $p = 2$.

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

Tabel 3.12: Metodă Runge-Kutta bazată pe cuadraturi gaussiene cu $s = 2$ și $p = 4$.

mic de pași nu poate fi atins cu metode explicite. Pentru cele două valori numerice ale lui c vom utiliza radăcinile ecuației $P_2^* = 0$, i.e., $c_1 = \frac{1}{2} - \frac{\sqrt{3}}{6}$ și $c_2 = \frac{1}{2} + \frac{\sqrt{3}}{6}$. Calcule detaliate arată ca tabelul Butcher al metodei este cel din (3.12).
Menționăm în cele din urmă că folosind pentru c rădăcinile ecuației $P_3^* = 0$, i.e., $c_1 = \frac{1}{2} - \frac{\sqrt{15}}{10}$, $c_2 = \frac{1}{2}$ și $c_3 = \frac{1}{2} + \frac{\sqrt{15}}{10}$, putem obține metoda Runge-Kutta implicită cu $s = 3$ și $p = 4$ definită în tabelul Butcher (3.13).

Se impun câteva precizări privitoare la metodele Runge-Kutta implicite:

1. Faptul că aceste metode oferă un ordin mare cu un număr relativ mic de pași nu implică și un plus de eficiență a metodelor, căci costul rezolvării numerice, la fiecare pas, a ecuațiilor (3.20a)-(3.20f), atenție (!), acum implicite, poate fi extrem de mare pentru sistemele dinamice în care funcția f este puternic neliniară.

2. Aceste metode devin eficiente pentru așa-numitele sisteme rigide, pentru care se

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

Tabel 3.13: Metodă Runge-Kutta bazată pe cuadraturi gaussiene cu $s = 3$ și $p = 4$.

observă suprapunerea a două tipuri de oscilații: unele pe o frecvență foarte joasă și unele pe o frecvență foarte ridicată. Altfel spus, metodele sunt utile pentru sistemele de ecuații diferențiale ale caror matrice Jacobi au valori proprii care acoperă mai multe ordine de marime. Cu cât valoarea proprie cea mai mică e mai de depărtată de valoarea proprie cea mai mare, cu atât metodele devin mai utile. Să remarcă că pentru acest tip de probleme metodele Runge-Kutta explicite nu converg decât pentru valori extremi de mici ale pașilor de integrare, ceea ce le face ineficiente.

3. Tipul de probleme pentru care metodele Runge-Kutta implicite sunt eficiente este excelent descris intr-un articol clasic scris de L. F. Shampine și C. W. Gear, *A user's view of solving stiff ordinary differential equations*, SIAM Review **21**, 1 (1979): „The problems called "stiff" are too important to ignore, and are too expensive to overpower. They are too important to ignore because they occur in many physically important situations. They are too expensive to overpower because of their size and the inherent difficulty they present to classical methods, no matter how great an improvement in computer capacity becomes available. Even if one can bear the expense, classical methods of solution require so many steps that roundoff errors may invalidate the solution. It is all the more frustrating that the solutions of stiff problems look like they should be particularly easy to compute. [...] By a stiff problem we mean one for which no solution component is unstable (no eigenvalue has a real part which is at all large and positive) and at least some component is very stable (at least one eigenvalue has a real part which is large and negative). Further, we will not call a problem stiff unless its solution is slowly varying with respect to the most negative real part of the eigenvalues. (Roughly, we mean that the derivatives of the solution are small compared to the corresponding derivatives [...]) Consequently, a problem may be stiff for some intervals of the independent variable and not for others. [...] The essence of the matter is that for most problems the accuracy requirement dictates the choice of step size, but for some, the stiff problems, the stability requirement does. [...] One worry should be dispelled at once. When implemented properly, the instability on encountering stiffness of classical methods such as Euler's is automatically detected and handled by reducing the step size. Computer programs suitable for nonstiff problems do not "blow up" in the presence of stiffness, they just become inefficient."

3.2.3 Metode Runge-Kutta împerecheate. Controlul erorii

Problema metodelor Runge-Kutta împerecheate este una veche și a pornit din nevoia practică de a putea estima *în mod eficient* eroarea locală de trunchiere. Am subliniat în mod eficient pentru că evaluarea în sine a erorii locale de trunchiere nu ridică, de fapt, nicio problemă. Știind, de pildă, că avem o metodă de ordin p care e corectă, preț de un pas, până la termeni $\mathcal{O}\left(h^{p+1}\right)$, putem estima eroarea împărțind pasul în 10 și calculând diferența dintre cele două rezultate. Pentru o estimare mai precisă a erorii putem împărți pasul în 100 sau 1000 de pași mai mici, însă această abordare este extrem

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	0
	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{8}$

Tabel 3.14: Metode Runge-Kutta împerecheate. Prima metodă, cea superioară, este de ordin $p = 2$ în timp ce a doua, cea inferioară, este de ordin $p = 3$.

de ineficientă sub raportul timpului de calcul.

Un mod de a rezolva problema are la baza așa-numitele metode Runge-Kutta împerecheate. Filozofia acestei abordări este următoarea: dată fiind o metodă Runge-Kutta de ordin p și o metodă Runge-Kutta de ordin $p + 1$ care folosește aproape toate evaluările funcției f ale metodei de ordin p se poate calcula eroarea locala de trunchiere calculând diferența dintre cele doua estimări. În mod evident, marea problemă este determinarea acestor două metode Runge-Kutta cuplate care folosesc aceleași estimări ale funcției f. Una dintre cele mai populare perechi împerecheate este cea determinată de Bogacki și Shampine, prezentată în tabelul Butcher (3.14). Metoda a fost introdusă de P. Bogacki și L. F. Shampine în *A 3(2) pair of Runge-Kutta formulas*, Applied Mathematics Letters **2**, 321 (1989), și a devenit folosită pentru ecuații diferențiale care nu necesitau un control al erorii foarte precis și care nu sunt rigide. Această metodă este folosită de către funția *ode23* din MATLAB.

O alta metode extrem de populară este metoda Merson care se bazează pe împerecherea unei metode de ordin 3 cu una de ordin 4. Metoda este ilustrată în tabelul Butcher (3.15). Metoda este extrem de populară în cadrul comunității utilizatorilor de FORTRAN fiind implementată în două din rutinele standard ale Numerical Algorithms Group - NAG, *i.e.*, D02BGF și D02BHF. Metoda nu este implementată în MATLAB și MATHEMATICA. Metoda Merson are particularitatea că, atunci când e aplicată unui set de ecuații liniare, metodă inferioară, care este în mod curent o metodă de ordin $p = 4$ devine o metodă de ordin $p = 5$.

Menționăm, în final, metoda Zonneveld ca un exemplu de împerechere simplă a două metode Runge-Kutta de nivel înalt. Tabelul Butcher al metodei este prezentat în tabelul (3.16). Asemeni metodei lui Merson, și această metodă a aparut într-o lucrare obscură, recte, J.A. Zonneveld, *Automatic integration of ordinary differential equations*, Report R743, Mathematisch Centrum, Postbus 4079, 1009AB Amsterdam, 1963, cunoscând

0					
$\frac{1}{3}$	$\frac{1}{3}$				
$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$			
$\frac{1}{2}$	$\frac{1}{8}$	0	$\frac{3}{8}$		
1	$\frac{1}{2}$	0	$-\frac{3}{2}$	2	
	$\frac{1}{6}$	0	0	$\frac{2}{3}$	$\frac{1}{6}$
	$\frac{1}{10}$	0	$\frac{3}{10}$	$\frac{2}{5}$	$\frac{1}{5}$

Tabel 3.15: Metode Runge-Kutta-Merson împerecheate. Prima metodă, cea superioară, este de ordin $p = 3$ în timp ce a doua, cea inferioară, este de ordin $p = 4$.

apoi o larga utilizare.

Lista metodelor simplu împerecheate este una lungă, iar foarte scurta noastră trecere în revistă a omis contribuții remarcabile din partea lui Ceschino, Fehlberg[1], Dormand și Prince, etc. Pentru o tratare amănunțită a subiectului recomandă cititorilor interesați Ref. [7], Ref. [8], Ref. [9] și Ref. [11].

În final dorim să facem următoarele comentarii: metodele Runge-Kutta explicite de ordin 3 prezentate în tabelele Butcher (3.3), (3.4) și (3.5) pot fi transformate în tabele Butcher pentru metode *triplu* împerecheate. Astfel, tabelul (3.3) se transformă cu ușurință în (3.17), tabelul (3.4) în (3.18), iar tabelul (3.5) în (3.19).

[1]Erwin Fehlberg este considerat a fi unul dintre părinții metodelor Runge-Kutta împerecheate, lucrările sale de pionierat din perioada când lucra la NASA, toate cu un singur autor, fiind puncte de referință în domeniu. Deoarece o parte din ele au fost avut o bună perioadă un caracter secret, abia recent fiind disponibile pe serverul de documente al NASA, menționăm aici pe cele mai importante dintre ele: Technical Note D-1834, *On the error propagation of some interpolation formulas for second-order differential equations*, July 1963; Nasa Technical Report - 248, *New one-step integration methods of high-order accuracy applied to some problems in celestial mechanics*, October 1966; Nasa Technical Report - 287, *Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control*, October 1968; Nasa Technical Report - 315, *Low order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems*, July 1969; Nasa Technical Report - 352, *Some experimental results concerning the error propagation in Runge-Kutta type integration formulas*, October 1970; Nasa Technical Report - 381, *Classical eightth- and lower-order Runge-Kutta-Nystrom formulas with stepsize control for special second-order differential equations*, March 1972; Nasa Technical Report - 410, *Classical eightth- and lower-order Runge-Kutta-Nyström formulas with a new stepsize control procedure for special second-order differential equations*, June 1973; Nasa Technical Report - 432, *Classical seventh-, sixth-, and fifty-order Runge-Kutta-Nyström formulas with stepsize control for general second-order differential equations*, October 1974.

0					
$\frac{1}{2}$	$\frac{1}{2}$				
$\frac{1}{2}$	0	$\frac{1}{2}$			
1	0	0	1		
$\frac{3}{4}$	$\frac{5}{32}$	$\frac{7}{32}$	$\frac{13}{32}$	$-\frac{1}{32}$	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$	0
	$-\frac{1}{2}$	$\frac{7}{3}$	$\frac{7}{3}$	$\frac{13}{6}$	$-\frac{16}{3}$

Tabel 3.16: Metode Runge-Kutta-Zonneveld împerecheate. Prima metodă, cea superioară, este de ordin $p = 4$ în timp ce a doua, cea inferioară, este de ordin $p = 5$.

0				
c_2	c_2			
c_3	$\frac{c_3\left(3c_2\left(1-c_2\right)-c_3\right)}{c_2\left(2-3c_2\right)}$	$\frac{c_3\left(c_3-c_2\right)}{c_2\left(2-3c_2\right)}$		
	$\frac{2-3\left(c_2+c_3\right)+6c_2c_3}{6c_2c_3}$	$\frac{2-3c_3}{6c_2\left(c_2-c_3\right)}$	$\frac{2-3c_2}{6c_3\left(c_3-c_2\right)}$	ordin 3
	$1-\frac{1}{2c_2}$	$\frac{1}{2c_2}$		ordin 2
	1			ordin 1

Tabel 3.17: Metode Runge-Kutta triplu împerecheate făcute pe baza tabelului Butcher (3.3).

0				
$\frac{2}{3}$	$\frac{2}{3}$			
0	$-\frac{1}{4b_3}$	$\frac{1}{4b_3}$		
	$\frac{1}{4} - b_3$	$\frac{3}{4}$	b_3	ordin 3
	$\frac{1}{4}$	$\frac{3}{4}$		ordin 2
	1			ordin 1

Tabel 3.18: Metode Runge-Kutta triplu împerecheate făcute pe baza tabelului Butcher (3.4).

0				
$\frac{2}{3}$	$\frac{2}{3}$			
$\frac{2}{3}$	$\frac{2}{3} - \frac{1}{4b_3}$	$\frac{1}{4b_3}$		
	$\frac{1}{4}$	$\frac{3}{4} - b_3$	b_3	ordin 3
	$\frac{1}{4}$	$\frac{3}{4}$		ordin 2
	1			ordin 1

Tabel 3.19: Metode Runge-Kutta triplu împerecheate făcute pe baza tabelului Butcher (3.5).

3.3 Implementări software disponibile

Cea mai bună implementarea a metodelor Runge-Kutta este cea oferită de MATHEMATICA, care permite rezolvarea ecuațiilor diferențiale ordinare folosind atît metode explicite, cât și metode implicite. Pentru metodele explicite MATHEMATICA are o serie de metode Runge-Kutta împerecheate (*ExplicitRungeKutta*), cea mai puțin eficientă fiind de ordin 2(1), iar cea mai eficientă fiind de ordin 9(8). Pentru metodele Runge-Kutta implicite MATHEMATICA acoperă (prin opțiunea *ImplicitRungeKutta*) clasa metodelor bazate pe cuadraturi gaussiene (prezentată de noi mai sus), alături de numeroase alte metode implicite cum sunt Runge-Kutta-Lobatto-IIIA, Runge-Kutta-Lobatto-IIIB, Runge-Kutta-Lobatto-IIIC, Runge-Kutta-Radau-IA, Runge-Kutta-Radau-IIA. Mai mult, MATHEMATICA are un pachet dedicat integratorilor simplectici (*SymplecticPartitionedRungeKutta*), care descriu precis sistemele hamiltoniene și sunt extrem de utili în modelarea numerică a sistemelor fizice. Principala funcție folosită pentru rezolvarea numerică a ecuațiilor diferențiale ordinare este *NDSolve* care acceptă ca parametri tipul metodei (*Method*), ordinul metodei (*DifferenceOrder*), tabelul de coeficienți (*Coefficients*, în cazul în care metoda este una specială), eroarea (*AccuracyGoal* și *PrecisionGoal*), marimea pasului inițial folosit pentru inițializarea integrării (*StartingStepSize*), marimea maximă a pasului (*MaxStepSize*), numărul maxim de pași de integrare (*MaxSteps*), etc. Integratorul numeric poate verifica numeric dacă ecuația diferențială este rigidă sau nu, și, mai important, poate schimba tipul și ordinul metodei pentru a obține un anumit nivel de eroare.

Filosofia din spatele integratorului numeric folosit de MATHEMATICA a fost aceea de a strange cele mai eficiente metode existente și a le oferi spre utilizare într-un mod unificat. La polul opus, MATLAB are o selecție destul de săracă de integratori numerici, care reflectă în bună măsură temele de cercetare ale creatorilor suitei, L. F. Shampine și M. W. Reicheit (a se vedea în acest sens articolul *The MATLAB ODE suite*, SIAM Journal of Scientific Computing 18, 1 (1997)). Suita MATLAB nu cuprinde integratori Runge-Kutta impliciți de ordin înalt (ci doar trei integratori de ordin 3(2) care descriu sisteme nu foarte rigide, *i.e.*, *ode23s*, *ode23t* și *ode23tb*), iar la capitolul metode Runge-Kutta explicite sunt implementate două metode împerecheată, recte aceea dezvoltată de Bogacki și Shampine, prezentată de noi în tabelul Butcher (3.14), și o metodă de ordin 5(4) dezvoltată de Dormand și Prince (neprezentată de noi anterior datorită coeficienților destul de complicați, dar care poate fi găsită în Ref. [11]). Toți integratorii numerici din MATLAB acceptă un set standardizat de parametri care includ eroarea relativă (*RelTol*) și eroarea absolută (*AbsTol*), pasul inițial de integrare (*InitialStep*), numărul maxim de pași de integrare (*MaxStep*) și, foarte important, matricea Jacobi a sistemului de ecuații diferențiale (*Jacobian* și *JPattern*). Această din urmă opțiune este extrem de utilă în cazul ecuațiilor rigide și sporește considerabil eficiența integratorilor numerici.

Chiar daca codurile MATLAB dedicate ecuațiilor diferențiale ordinare sunt neimpresionante, există o librărie - GniCodes - dezvoltată de E. Hairer și M. Hairer în care sunt implementate metode Runge-Kutta implicite dezvoltate anume pentru ecuații diferențiale rigide și cu invarianți (*i.e.*, ecuații de tip Hamilton). Librăria este descrisă pe larg în *GniCodes - Matlab progras for geometric numerical integration*, E. Hai-

rer și M. Hairer, Frontiers in Numerical Analysis, Universitext, Springer (2003), codurile propriu-zise fiind prezentate în detaliu pe site-ul personal al lui Ernst Hairer, http://www.unige.ch/~hairer/software.html. O altă librărie populară în rândul utilizatorilor este DiffMan, introduă în *DiffMan - an object oriented MATLAB toolbox for solving differential equations on manifolds*, K. Engo, A. Marthinsen, și H. Munthe-Kaas, raport al Departamentului de Informatică al Universității din Bergen, Norvegia, disponibil la adresa http://www.ii.uib.no/publikasjoner/texrap/pdf/1999-164.pdf.

3.4 Exerciții și probleme

Exercițiul 1: Arătați că metoda Runge-Kutta descrisă de tabelul Butcher

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array},$$

unde $\gamma = \left(3 \pm \sqrt{3}\right)/6$, este de ordinul 3. Comentați influența elementului nul din tabelul Butcher asupra schemei numerice.

Exercițiul 2: Să se arate că tabelul Butcher

$$\begin{array}{c|ccc} 0 & & & \\ \frac{2}{3} & \frac{2}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} & \\ \hline & \frac{1}{4} & \frac{3}{8} & \frac{3}{8} \end{array}$$

descrie o metoda Runge-Kutta cuplată de ordinul 2(3).

Problema 1: Să se rezolve numeric sistemul Lorentz

$$\frac{dx}{dt} = \sigma\,(y-x) \tag{P.3.1a}$$

$$\frac{dy}{dt} = x\,(\rho-z)-y \tag{P.3.1b}$$

$$\frac{dz}{dt} = xy-\beta z \tag{P.3.1c}$$

pentru $\rho = 28$, $\sigma = 10$, $\beta = 8/3$ și condițiile inițiale $x(0) =$, $y(0) =$, $z(0) =$ și să se determine traiectoriile din spațiul fazelor folosind metode Runge-Kutta implicite și explicite (de exemplu, Zonneveld și Gauss-Legendre cu $s = 2$, $p = 4$). Se justifică utilizarea metodelor Runge-Kutta implicite? Să se determine traiectoriile din spațiul fazelor pentru $\rho = 99.96$, păstrând ceilalți parametri constanți. Ce diferențe se observă?

Problema 2: Să se rezolve numeric sistemul Rössler

$$\frac{dx}{dt} = -y - z \tag{P.3.2a}$$
$$\frac{dy}{dt} = x + ay \tag{P.3.2b}$$
$$\frac{dz}{dt} = b + z\,(x - c) \tag{P.3.2c}$$

și să se determine traiectoriile soluțiilor în planul $x - y$ pentru $a = b = 1$ și $c = 4$, 6, 8.5, 8.7, 9, 12, 13 și 18. Se vor folosi condițiile inițiale $x(0) =$, $y(0) =$, $z(0) =$ Să se determine perioada traiectoriilor determinate numeric.

3.5 Bibliografie

1. Gear, C. W., *Numerical initial value problems in ordinary differential equations*, Prentice-Hall (1971)
2. Iserles, A., *A first course in the numerical analysis of differential equations*, Cambridge University Press (2009)
3. Ascher, U. M., *Numerical methods for evolutionary differential equations*, Society for Industrial and Applied Mathematics (2008)
4. Griffiths, D. F., și Higham, D. J., *Numerical methods for ordinary differential equations. Initial value problems*, Springer (2010)
5. Atkinson, K. E., Han, W., și Stewart, D. E., *Numerical solution of ordinary differential equations*, John Wiley & Sons (2009)
6. Butcher, J. C., *A history of Runge-Kutta methods*, Applied Numerical Mathematics **20**, 247 (1996).
7. Butcher, J. C., *Numerical methods for ordinary differential equations*, John Wiley & Sons (2008)
8. Hairer, E., Nørsett, S.P., și Wanner, G., *Solving ordinary differential equations I. Nonstiff Problems*, Springer (2008)
9. Hairer, E., și Wanner, G., *Solving ordinary differential equations II. Stiff and differential-algebraic problems*, Springer (2010)
10. Hairer, E., Lubich, C., și Wanner, G., *Geometric numerical integration. Structure-preserving algorithms for ordinary differential equations*, Springer (2000)
11. Dormand, J. R., *Numerical methods for differential equations. A computational approach*, CRC Press (1996)
12. Leimkuhler, B., și Reich, S., *Simulating Hamiltonian dynamics*, Cambridge University Press (2004)
13. Shampine, L. F., Gladwell, I., și Thompson, S., *Solving ODEs with Matlab*, Cambridge University Press (2003)
14. Verner, J. H., *Families of imbedded Runge-Kutta methods*, SIAM Journal of Numerical Analysis **16**, 857 (1979)