Zaawansowane metody numeryczne

Komputerowa analiza zagadnień różniczkowych

8. Metody wielokrokowe

Metody Verleta

P. F. Góra

http://th-www.if.uj.edu.pl/zfs/gora/

semestr letni 2006/07

Motywacja

Metody wielokrokowe są starsze i bardziej popularne od metod Rugego-Kutty. Są też prostsze w tym sensie, że nie wymagają (często żmudnych) obliczeń prawej strony (pochodnej) w „niepotrzebnych” punktach pośrednich — korzystają wyłącznie z wartości obliczonych w „potrzebnych” punktach, czyli w punktach, w których chcieliśmy uzyskać rozwiązanie. Jawne metody Adamsa wymagają tylko jednego obliczenia prawej strony na krok.

Wyróżnia się trzy podstawowe typy metod wielokrokowych:

1. Jawne metody Adamsa-Bashfortha,
2. Niejawne metody Adamsa-Moultona,
3. Niejawne metody BDF dla problemów sztywnych.

Metody Adamsa — ogólne sformułowanie

Rozważamy problem Cauchy'ego

$$\begin{cases} \dfrac{d\mathbf{y}}{dx} = \mathbf{f}(x, \mathbf{y}) \,, \\ \mathbf{y}(x) = \mathbf{y}_0 \,. \end{cases} \tag{1}$$

Rozwiązanie ma postać

$$\mathbf{y}(x_{n+1}) = \mathbf{y}_n + \int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\, dx \,. \tag{2}$$

Klasyczne metody Adamsa polegają na zastąpieniu $\mathbf{f}(x, \mathbf{y}(x))$ w (2) przez *wzór ekstrapolacyjny* (Bashforth) lub *interpolacyjny* (Moulton) z węzłami interpolacji odległymi o krok całkowania, h, i scałkowaniu tego wzoru. Celem takiego postępowania, podobnie jak w metodach Rungego-Kutty, jest uwzględnienie zmienności pochodnej w obrębie kroku całkowania. Zauważmy, że wynik całkowania wielomianu interpolacyjnego *nie zależy* od wartości funkcji — wartości $\mathbf{f}(x_l, \mathbf{y}_l)$ wchodzą jako „ustalone” wartości interpolowanej funkcji w węzłach.

k-krokowe metody Adamsa:

$$\text{Adams-Bashforth:} \quad \mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{j=1}^{k} \beta_j \mathbf{f}_{n+1-j} + O(h^{k+1}) \,, \tag{3}$$

$$\text{Adams-Moulton:} \quad \mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{j=0}^{k-1} \widetilde{\beta}_j \mathbf{f}_{n+1-j} + O(h^{k+1}) \,. \tag{4}$$

Przykład - wyprowadzenie trzykrokowej metody Adamsa-Bashfortha

Funkcję podcałkową w (2) przybliżam poprzez ekstrapolację wielomianową z trzech ostatnio obliczonych punktów, odległych od siebie o h:

$$\begin{aligned}
\mathbf{f}(x,\mathbf{y}(x)) \simeq \mathbf{f}_{\text{extr}}(x) &= \frac{(x-x_{n-1})(x-x_n)}{(x_{n-2}-x_{n-1})(x_{n-2}-x_n)}\mathbf{f}_{n-2} \\
&+ \frac{(x-x_{n-2})(x-x_n)}{(x_{n-1}-x_{n-2})(x_{n-1}-x_n)}\mathbf{f}_{n-1} + \frac{(x-x_{n-2})(x-x_{n-1})}{(x_n-x_{n-2})(x_n-x_{n-1})}\mathbf{f}_n \\
&= \frac{1}{2h^2}(x-x_n+h)(x-x_n)\mathbf{f}_{n-2} - \frac{1}{h^2}(x-x_n+2h)(x-x_n)\mathbf{f}_{n-1} \\
&\quad + \frac{1}{2h^2}(x-x_n+2h)(x-x_n+h)\mathbf{f}_n\,. \qquad (5)
\end{aligned}$$

Następnie

$$\int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\, dx \simeq \int_{x_n}^{x_{n+1}} \mathbf{f}_{\text{extr}}(x) dx = \frac{1}{2h^2}\mathbf{f}_{n-2}\int_0^h (z+h)z\, dz$$

$$- \frac{1}{h^2}\mathbf{f}_{n-1}\int_0^h (z+2h)z\, dz + \frac{1}{2h^2}\mathbf{f}_n \int_0^h (z+2h)(z+h)\, dz$$

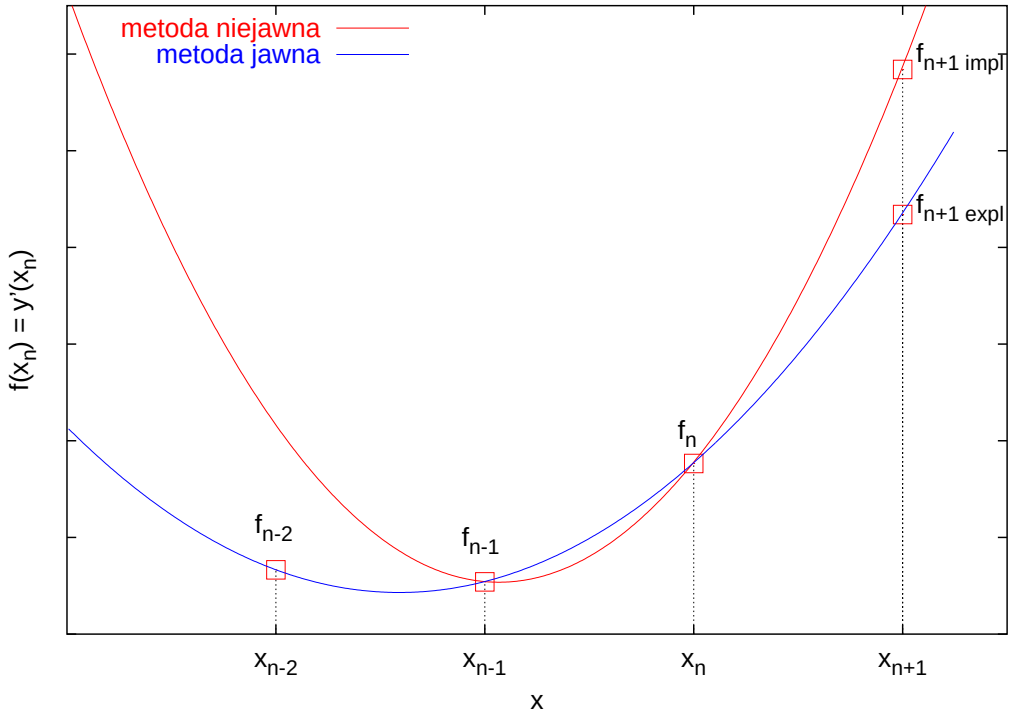
$$= \frac{1}{2h^2}\cdot\frac{5}{6}h^3\mathbf{f}_{n-2} - \frac{1}{h^2}\cdot\frac{4}{3}h^3\mathbf{f}_{n-1} + \frac{1}{2h^2}\cdot\frac{23}{6}h^3\mathbf{f}_n$$

$$= \frac{h}{12}\left(23\mathbf{f}_n - 16\mathbf{f}_{n-1} + 5\mathbf{f}_{n-2}\right). \tag{6}$$

Proszę to porównać z wyrażeniem (7c) poniżej.

Idea konstrukcji metod Adamsa:

rysunek nie jest wierny, jako że pochodnej w punkcie x_{n+1} *nie* oblicza się za pomocą prostej interpolacji/ekstrapolacji



Metody Adamsa-Bashfortha (jawne)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}_n + O(h^2) \tag{7a}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}\left(3\mathbf{f}_n - \mathbf{f}_{n-1}\right) + O(h^3) \tag{7b}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{12}\left(23\mathbf{f}_n - 16\mathbf{f}_{n-1} + 5\mathbf{f}_{n-2}\right) + O(h^4) \tag{7c}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{24}\left(55\mathbf{f}_n - 59\mathbf{f}_{n-1} + 37\mathbf{f}_{n-2} - 9\mathbf{f}_{n-3}\right) + O(h^5) \tag{7d}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{720}\left(1901\mathbf{f}_n - 2774\mathbf{f}_{n-1} + 2616\mathbf{f}_{n-2} - 1274\mathbf{f}_{n-3} + 251\mathbf{f}_{n-4}\right) + O(h^6) \tag{7e}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{1440}\left(4277\mathbf{f}_n - 7923\mathbf{f}_{n-1} + 2616\mathbf{f}_{n-2} - 7298\mathbf{f}_{n-3} + 2877\mathbf{f}_{n-4} - 475\mathbf{f}_{n-5}\right) + O(h^7) \tag{7f}$$

Metody Adamsa-Moultona (niejawne)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}_{n+1} + O(h^2) \tag{8a}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}\left(\mathbf{f}_{n+1} + \mathbf{f}_n\right) + O(h^3) \tag{8b}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{12}\left(5\mathbf{f}_{n+1} + 8\mathbf{f}_n - \mathbf{f}_{n-1}\right) + O(h^4) \tag{8c}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{24}\left(9\mathbf{f}_{n+1} + 19\mathbf{f}_n - 5\mathbf{f}_{n-1} + \mathbf{f}_{n-2}\right) + O(h^5) \tag{8d}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{720}\left(251\mathbf{f}_{n+1} + 646\mathbf{f}_n - 264\mathbf{f}_{n-1} + 106\mathbf{f}_{n-2} - 19\mathbf{f}_{n-3}\right) + O(h^6) \tag{8e}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{1440}\left(475\mathbf{f}_{n+1} + 1427\mathbf{f}_n - 798\mathbf{f}_{n-1} + 482\mathbf{f}_{n-2} - 173\mathbf{f}_{n-3} + 27\mathbf{f}_{n-4}\right) + O(h^7) \tag{8f}$$

Metody BDF, *Backward Differentiation Formula* (niejawne)

Dla układów sztywnych, pochodna obliczana tylko w prawym krańcu przedziału.

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}_{n+1} + O(h^2) \tag{9a}$$

$$\mathbf{y}_{n+1} = \frac{1}{3}\left(4\mathbf{y}_n - \mathbf{y}_{n-1}\right) + \frac{2}{3}h\mathbf{f}_{n+1} + O(h^3) \tag{9b}$$

$$\mathbf{y}_{n+1} = \frac{1}{11}\left(18\mathbf{y}_n - 9\mathbf{y}_{n-1} + 2\mathbf{y}_{n-2}\right) + \frac{6}{11}h\mathbf{f}_{n+1} + O(h^4) \tag{9c}$$

$$\mathbf{y}_{n+1} = \frac{1}{25}\left(48\mathbf{y}_n - 36\mathbf{y}_{n-1} + 16\mathbf{y}_{n-2} - 3\mathbf{y}_{n-3}\right) + \frac{12}{25}h\mathbf{f}_{n+1} + O(h^5) \tag{9d}$$

$$\mathbf{y}_{n+1} = \frac{1}{137}\left(300\mathbf{y}_n - 300\mathbf{y}_{n-1} + 200\mathbf{y}_{n-2} - 75\mathbf{y}_{n-3} + 12\mathbf{y}_{n-4}\right) + \frac{60}{137}h\mathbf{f}_{n+1} + O(h^6) \tag{9e}$$

$$\mathbf{y}_{n+1} = \frac{1}{147}\left(360\mathbf{y}_n - 450\mathbf{y}_{n-1} + 400\mathbf{y}_{n-2} - 225\mathbf{y}_{n-3} + 72\mathbf{y}_{n-4} - 10\mathbf{y}_{n-5}\right) + \frac{60}{147}h\mathbf{f}_{n+1} + O(h^7) \tag{9f}$$

Podane wyżej metody są *wszystkimi* metodami wielokrokowymi o stałym kroku, opartymi o interpolację/ekstrapolację wielomianową.

Zgodność metod Adamsa jest oczywista.

Zgodność metod BDF też, po prostych rachunkach, okazuje się być oczywista. Dla przykładu, dla drugiej z metod BDF otrzymujemy

$$3\frac{\mathbf{y}_{n+1}-\mathbf{y}_n}{h}-\frac{\mathbf{y}_n-\mathbf{y}_{n-1}}{h}=2\mathbf{f}_{n+1}\,. \tag{10}$$

Stabilność metod wielokrokowych

Stabilność metod wielokrokowych bada się nieco inaczej niż stabilność metod jednokrokowych typu metod Rungego-Kutty. Dla przykładu pokażemy jak badać stabilność k-krokowej jawnej metody Adamsa-Bashfortha; stabilność niejawnych metod Adamsa-Moultona i BDF bada się zupełnie podobnie.

Rozważmy równanie (3), w którym wszystkie $\mathbf{y}_l$ zostały zaburzone: $\mathbf{y}_l \to \mathbf{y}_l + \boldsymbol{\varepsilon}_l$. Mamy

$$\begin{aligned} \mathbf{y}_{n+1} + \boldsymbol{\varepsilon}_{n+1} &= \mathbf{y}_n + \boldsymbol{\varepsilon}_n + h \sum_{j=1}^{k} \beta_j \mathbf{f}\left(x_{n+1-j}, \mathbf{y}_{n+1-j} + \boldsymbol{\varepsilon}_{n+1-j}\right) \\ &\simeq \mathbf{y}_n + \boldsymbol{\varepsilon}_n + h \sum_{j=1}^{k} \beta_j \left(\mathbf{f}_{n+1-j} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{n+1-j} \boldsymbol{\varepsilon}_{n+1-j} \right). \quad (11) \end{aligned}$$

Wszystkie k jakobiany w równaniu (11) oblicza się w *różnych* punktach. Jeśli jednak krok h jest mały, możemy przyjąć, że w przybliżeniu są one sobie równe, podobnie jak to robiliśmy przy analizie stabilności metod Rungego-Kutty. (Założenie to nie jest spełnione w miejscach, w których funkcja zmienia się bardzo gwałtownie.) Wobec czego przechodzimy do reprezentacji, w której jakobian jest diagonalny — zastępujemy jakobian $\partial \mathbf{f}/\partial \mathbf{y}$ jego wartością własną λ, natomiast błędy ε_l zastępujemy odpowiednią składową ε_l. Formalnie rzecz biorąc, analizę taką trzeba powtórzyć dla wszystkich λ i odpowiadających im składowych błędów, okaże się jednak, że nie jest to konieczne.

Z równania (11) otrzymujemy zatem

$$\varepsilon_{n+1} = \varepsilon_n + h\lambda\beta_1\varepsilon_n + h\lambda\beta_2\varepsilon_{n-1} + \cdots + h\lambda\beta_k\varepsilon_{n+1-k}\,. \qquad (12)$$

Macierz wzmocnienia

Oznaczmy $h\lambda = z$. Równanie (12) możemy przepisać w postaci macierzowej

$$\begin{bmatrix} \varepsilon_{n+1} \\ \varepsilon_n \\ \varepsilon_{n-1} \\ \vdots \\ \varepsilon_{n+2-k} \end{bmatrix} = \begin{bmatrix} 1+\beta_1 z & \beta_2 z & \beta_3 z & \dots & \beta_{k-1} z & \beta_k z \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon_n \\ \varepsilon_{n-1} \\ \varepsilon_{n-2} \\ \vdots \\ \varepsilon_{n+1-k} \end{bmatrix} \tag{13a}$$

czyli

$$\vec{\varepsilon}_{n+1} = \mathbf{G}\vec{\varepsilon}_n \tag{13b}$$

Zwracam uwagę, że ε_n z równania (11) i $\vec{\varepsilon}_n$ z równania (13b) to są *zupełnie różne wektory*. Badanie stabilności metody (3) sprowadza się teraz do znalezienia wartości własnych macierzy wzmocnienia $\mathbf{G}$.

Wartości własne macierzy wzmocnienia

Oznaczmy $W_k = \det(\mathbf{G} - g\mathbb{I})$. Rozwijając względem ostatniej kolumny (patrz (13a)) otrzymujemy

$$W_k = -gW_{k-1} + (-1)^{k+1}\beta_k z \det \begin{bmatrix} 1 & -g & 0 & \dots & 0 \\ 0 & 1 & -g & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \tag{14}$$

Wyznacznik macierzy wypisanej jawnie w równaniu (14) wynosi 1. Postępowanie to łatwo iterować. Ostatecznie otrzymujemy następujące równanie charakterystyczne dla macierzy wzmocnienia (uwaga: zmienną jest g):

$$g^k - (1 + z\beta_1)g^{k-1} - z\beta_2 g^{k-2} - \cdots - z\beta_{k-1}g - z\beta_k = 0\,. \qquad (15)$$

Metoda Adamsa-Bashfortha rzędu k jest stabilna jeśli wszystkie pierwiastki równania (15) *leżą wewnątrz okręgu jednostkowego*. Ogół takich z, dla których pierwiastki (15) leżą wewnątrz okręgu jednostkowego, nazywam *obszarem stabilności metody*. Zauważmy, że analizując obszar stabilności, uwalniamy się niejako od równania różniczkowego, obecnego tu formalnie tylko poprzez wartości własne jakobianu, a skupiamy się na samej metodzie, podobnie jak to było dla metod Rungego-Kutty.

Wyprowadzenie równań opisujących stabilność niejawnych metod Adamsa-Moultona i BDF przebiega w sposób analogiczny.

- Obszar stabilności metod Adamsa-Bashfortha maleje wraz ze wzrostem rzędu metody.
- Dwie pierwsze metody Adamsa-Moultona są A-stabilne, obszary stabilności pozostałych maleją wraz ze wzrostem rzędu metody.
- Metody BDF są stabilne *poza* pewnym obszarem ograniczonym, który rośnie wraz ze wzrostem rzędu metody.
- Żadna metoda wielokrokowa oparta o interpolację/ekstrapolację wielomianową z $k > 6$ nie jest stabilna!
- Można konstruować stabilne metody wielokrokowe wyższych rzędów, ale dla nich nie obowiązuje już paradygmat ekstrapolowania pochodnej na podstawie zachowania w poprzednich węzłach — nie są to więc metody Adamsa.

Zaledy metod wielokrokowych

- ☺ Koncepcyjna prostota, łatwość zaimplementowania.
- ☺ Szybkość (zwłaszcza dla metod jawnych).
- ☺ Uzyskanie wysokiego rzędu jest obliczeniowo tanie.
- ☺ Popularność, bardzo duża ilość gotowych kodów.

Wady metod wielokrokowych

- ☹ Wymagają inicjalizacji.
- ☹ Mają kiepskie własności stabilności.
- ☹ Kłopotliwa zmiana kroku.

Metody predyktor-korektor (*predictor-corrector*)

Jeśli problem *nie jest sztywny*, bardzo często pewną jawną metodę Adamsa-Bashfortha (7) stosuje się jednocześnie z niejawną metodą Adamsa-Moultona (8) o tym samym rzędzie. Za pomocą metody jawnej *przewiduje się* rozwiązanie, które potem *poprawia się* za pomocą metody niejawnej. Można to dalej powtarzać, poprawiając poprawione*. Uzyskana metoda jest oczywiście jawna — unika się rozwiązywania układu równań algebraicznych, na ogół nieliniowych.

Metody predyktor-korektor są bardzo popularne† — w praktycznych zastosowaniach metody Adamsa występują prawie wyłącznie w tym zestawieniu.

*Vide *grabit' nagrabljennoje*.

†Ale ja ich nie lubię ☹

Przykład:

$$\text{predyktor:} \quad \widetilde{\mathbf{y}}_{n+1} = \mathbf{y}_n + \tfrac{1}{2}h\left(3\mathbf{f}_n - \mathbf{f}_{n-1}\right), \tag{16a}$$

$$\text{korektor:} \quad \mathbf{y}_{n+1} = \mathbf{y}_n + \tfrac{1}{2}h\left(\mathbf{f}(x_{n+1}, \widetilde{\mathbf{y}}_{n+1}) + \mathbf{f}_n\right), \tag{16b}$$

$$\mathbf{f}_{n+1} = \mathbf{f}(x_{n+1}, \mathbf{y}_{n+1}). \tag{16c}$$

Niekiedy krok korektora powtarza się (iteruje) w celu uzyskania samouzgodnionego (*self-consistent*) rozwiązania nieliniowego równania algebraicznego. Na ogół jednak dokonuje się tylko jednego lub dwóch kroków korektora.

Metody Verleta — motywacja

Ponieważ klasyczne równania ruchu (równania Newtona) mają postać równań różniczkowych drugiego rzędu, w praktyce bardzo ważna jest umiejętność numerycznego rozwiązywania równań postaci $m_i\ddot{\mathbf{x}}_i = \mathbf{F}_i$, $i = 1, 2, \ldots, N$, czyli

$$\frac{d^2\mathbf{x}}{dt^2} = \mathbf{a}(t, \mathbf{x}(t))\,, \tag{17}$$

gdzie $\mathbf{x}, \mathbf{a} \in \mathbb{R}^{3N}$, N jest ilością cząstek[‡]. Równanie tej postaci można łatwo przekształcić do układu równań pierwszego rzędu, ale szczególnie efektywne powinny być metody „od razu” dostosowane do równań rzędu drugiego.

[‡]Niekiedy ruch jest z przyczyn fizycznych ograniczony do przypadku dwu- lub jednowymiarowego. Wówczas zamiast $3N$ w (17) mamy odpowiednio $2N$ lub N.

Pewien bardzo zły algorytm

Równanie (17) rozwiązujemy z pewnym stałym i niewielkim krokiem h. *Naiwne* rozumowanie może doprowadzić do wniosku, że dla odpowiednio małych h, siła — a więc i przyspieszenie, czyli prawa strona równania (17) — nie zmienia się, a więc układ w ciągu jednego kroku zachowuje się tak, jakby był układem jednostajnie przyspieszonym. Wobec tego

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n h + \tfrac{1}{2}\mathbf{a}_n h^2 \,, \tag{18a}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n h \,, \tag{18b}$$

gdzie $\mathbf{v}_n$, $\mathbf{a}_n$ są, odpowiednio, prędkością i siłą obliczaną w chwili t_n. Równanie (18a) jest, w gruncie rzeczy, rozwinięciem Taylora $\mathbf{x}(t_n + h)$, nie jest zatem dziwne, że algorytm (18) jest zgodny z równaniem (17). Można to jednak także

pokazać w nieco inny sposób: Odejmijmy stronami równanie (18a) i to samo równanie cofnięte o jeden krok w czasie. Dostaniemy

$$\mathbf{x}_{n+1} - \mathbf{x}_n = \mathbf{x}_n - \mathbf{x}_{n-1} + (\mathbf{v}_n - \mathbf{v}_{n-1})h + \tfrac{1}{2}(\mathbf{a}_n - \mathbf{a}_{n-1})h^2 \,. \tag{19}$$

Z równania (18b) dostajemy

$$\mathbf{v}_n - \mathbf{v}_{n-1} = \mathbf{a}_{n-1} h \,. \tag{20}$$

Po podstawieniu do (19) i po uporządkowaniu dostajemy

$$\frac{\frac{\mathbf{x}_{n+1}-\mathbf{x}_n}{h} - \frac{\mathbf{x}_n-\mathbf{x}_{n-1}}{h}}{h} = \tfrac{1}{2}(\mathbf{a}_n + \mathbf{a}_{n-1}) \,. \tag{21}$$

W granicy $h \longrightarrow 0^+$ odtwarzamy równanie (17).

Jeśli przyspieszenie w (17) nie zależy od prędkości, macierz wzmocnienia dla metody (18) ma postać

$$\mathbf{G} = \begin{bmatrix} \mathbb{I} + \frac{1}{2}\mathbf{J}h^2 & h \\ \mathbf{J}h & \mathbb{I} \end{bmatrix}, \tag{22}$$

gdzie $\mathbf{J}_{ij} = \partial \mathbf{a}_i / \partial \mathbf{x}_j \big|_{t_n}$ jest Jakobianem przyspieszeń po położeniach. Widać, iż metoda (18) może mieć *bardzo poważne problemy ze stabilnością*.

Klasyczny algorytm Verleta

Spróbujmy wyprowadzić metodę całkowania równania (17) w sposób bardziej systematyczny. Zakładamy, że przyspieszenia nie zależą od prędkości. Dokonujemy następujących rozwinięć w szereg Taylora:

$$\mathbf{x}(t_n+h) = \mathbf{x}(t_n) + \left.\frac{d\mathbf{x}}{dt}\right|_{t_n} h + \frac{1}{2}\left.\frac{d^2\mathbf{x}}{dt^2}\right|_{t_n} h^2 + \frac{1}{6}\left.\frac{d^3\mathbf{x}}{dt^3}\right|_{t_n} h^3 + O(h^4) \quad (23a)$$

$$\mathbf{x}(t_n-h) = \mathbf{x}(t_n) - \left.\frac{d\mathbf{x}}{dt}\right|_{t_n} h + \frac{1}{2}\left.\frac{d^2\mathbf{x}}{dt^2}\right|_{t_n} h^2 - \frac{1}{6}\left.\frac{d^3\mathbf{x}}{dt^3}\right|_{t_n} h^3 + O(h^4) \quad (23b)$$

Po dodaniu równań (23) stronami i uporządkowaniu wyrazów otrzymujemy

$$\mathbf{x}_{n+1} = 2\mathbf{x}_n - \mathbf{x}_{n-1} + \mathbf{a}_n h^2 + O(h^4)\,. \quad (24)$$

Algorytm (24) zwany jest *metodą (algorytmem) Verleta* (Verlet 1967).

Zauważmy, iż metoda Verleta jest *metodą wielokrokową* — do obliczenia przyszłej wartości $\mathbf{x}$ potrzebna jest znajomość obecnej *oraz* poprzedniej wartości $\mathbf{x}$.

W celu pokazania, iż algorytm Verleta jest zgodny z równaniem (17), przekształcamy równanie (24) do postaci

$$\frac{\frac{\mathbf{x}_{n+1}-\mathbf{x}_n}{h} - \frac{\mathbf{x}_n-\mathbf{x}_{n-1}}{h}}{h} = \mathbf{a}_n \,. \tag{25}$$

I znów, w granicy $h \longrightarrow 0^+$ odtwarzamy równanie (17).

Z uwagi na to, że jest to metoda wielokrokowa, stabilność algorytmu Verleta bada się nieco inaczej niż stabilność dotąd poznanych algorytmów. Dla uproszczenia notacji przyjmijmy na chwilę, iż interesuje nas wyłącznie przypadek jednowymiarowy. Propagacja błędu opisana jest wówczas równaniem

$$\varepsilon_{n+1} = (2+\lambda)\varepsilon_n - \varepsilon_{n-1}\,, \tag{26}$$

gdzie $\lambda = da/dx|_{t_n} h^2$. Równanie (26) możemy przedstawić w postaci[§]

$$\begin{bmatrix} \varepsilon_{n+1} \\ \varepsilon_n \end{bmatrix} = \mathbf{G} \begin{bmatrix} \varepsilon_n \\ \varepsilon_{n-1} \end{bmatrix} = \begin{bmatrix} 2+\lambda & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon_n \\ \varepsilon_{n-1} \end{bmatrix}. \tag{27}$$

[§]Wielokrokowa propagacja błędu jest jednokrokowa w przestrzeni o odpowiednio większej ilości wymiarów!

Widać zatem, iż algorytm Verleta jest stabilny, jeżeli

$$|g_{\pm}| = \tfrac{1}{2}\left|2 + \lambda \pm \sqrt{4\lambda + \lambda^2}\right| < 1\,. \tag{28}$$

Rozważmy trzy przypadki:

1. $-4 \leqslant \lambda \leqslant 0$. Wówczas $4\lambda + \lambda^2 \leqslant 0$.

$$\begin{aligned} g_{\pm} &= 1 + \tfrac{1}{2}\lambda \pm \tfrac{i}{2}\sqrt{\left|4\lambda + \lambda^2\right|}\,, \\ |g_{\pm}|^2 &= = 1 + \lambda + \tfrac{1}{4}\lambda^2 + \tfrac{1}{4}\left|4\lambda + \lambda^2\right| \\ &= 1 + \lambda + \tfrac{1}{4}\lambda^2 - \tfrac{1}{4}(4\lambda + \lambda^2) \equiv 1\,. \end{aligned}$$

   Algorytm jest zatem *neutralnie (marginalnie) stabilny*, co oznacza, że błędy nie są co prawda tłumione, ale też nie narastają i nie prowadzą do rozbieżności.

2. $\lambda > 0$. Wówczas $g_+ > 1$.

3. $\lambda < -4$. Wówczas $g_- < -1$.

Rekapitulując, dla ruchu jednowymiarowego *klasyczny algorytm Verleta jest* marginalnie stabilny *dla* $da/dx|_{t_n} h^2 \in [-4, 0]$ i niestabilny poza tym przedziałem.

Wady algorytmu Verleta

- Dziwne traktowanie prędkości — prędkość w kroku n znana jest dopiero po obliczeniu położenia w kroku $n+1$. W symulacjach znajomość prędkości nie jest potrzebna do obliczania sił, ale jest potrzebna do obliczania energii kinetycznej, temperatury, ciśnienia itp. Prędkość obliczana jest ze wzoru

$$\mathbf{v}_n = \frac{\mathbf{x}_{n+1} - \mathbf{x}_{n-1}}{2h}\,, \tag{29}$$

- We wzorze (24) *różnica* dwu „dużych” wyrażeń rzędu $O(h^0)$ jest dodawana do „małego” wyrażenia rzędu $O(h^2)$, co może prowadzić do znacznej utraty dokładności numerycznej.

Zauważmy, że wyrażenie (29) jest symetryczne w czasie; próba obliczania $\mathbf{v}_n = (\mathbf{x}_n - \mathbf{x}_{n-1})/h$ łamałaby symetrię odwrócenia w czasie, a przecież równania mechaniki (bez sił zależnych od prędkości) są mikroskopowo odwracalne!

Algorytm *leap–frog* („żabiego skoku”)

Spsobem na obejście (niektórych) trudności z algorytmem Verleta jest *algorytm leap–frog* (Hockney 1970), w którym prędkości obliczane są w czasach pośrednich pomiędzy położeniami¶:

$$\mathbf{v}\left(t_n + \tfrac{1}{2}h\right) = \mathbf{v}\left(t_n - \tfrac{1}{2}h\right) + h\mathbf{a}_n\,, \tag{30a}$$

$$\mathbf{x}_{n+1} = \mathbf{x}(t_n + h) = \mathbf{x}_n + h\mathbf{v}\left(t_n + \tfrac{1}{2}h\right). \tag{30b}$$

Prędkości obliczane są według wzoru

$$\mathbf{v}_n = \tfrac{1}{2}\left[\mathbf{v}\left(t_n - \tfrac{1}{2}h\right) + \mathbf{v}\left(t_n + \tfrac{1}{2}h\right)\right]. \tag{31}$$

¶Żaby wcale tak nie skaczą.

Zgodność algorytmu leap–frog

Aby sprawdzić czy algorytm leap–frog jest zgodny, należy z (30) wyeliminować prędkości. Po podstawieniu pierwszego z równań (30) do drugiego dostajemy

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{v}\left(t_n - \tfrac{1}{2}h\right) + h^2\mathbf{a}_n\,. \tag{32}$$

Drugie z równań (30) przesunięte w tył w czasie daje

$$h\mathbf{v}\left(t_n - \tfrac{1}{2}h\right) = \mathbf{x}_n - \mathbf{x}_{n-1}\,. \tag{33}$$

Ostatecznie

$$\mathbf{x}_{n+1} = 2\mathbf{x}_n - \mathbf{x}_{n-1} + h^2\mathbf{a}_n\,. \tag{34}$$

Równanie (34) ma taką samą postać co równanie (24), widzimy zatem, iż *algorytm leap–frog jest algebraicznie równoważny algorytmowi Verleta*, a więc jest zgodny z równaniem (17). Nie oznacza to jednak, iż *numeryczne* rezultaty zastosowania obu tych algorytmów są takie same!

Stabilność algorytmu leap–frog

Wyrażenia na propagację błędu w algorytmie leap–frog mają postać

$$\boldsymbol{\eta}_{n+1/2} = \boldsymbol{\eta}_{n-1/2} + h\,\mathbf{J}\boldsymbol{\varepsilon}_n\,, \tag{35a}$$

$$\boldsymbol{\varepsilon}_{n+1} = \boldsymbol{\varepsilon}_n + h\,\boldsymbol{\eta}_{n+1/2}\,, \tag{35b}$$

gdzie $\mathbf{J}_{ij} = \partial \mathbf{a}_i/\partial \mathbf{x}_j\big|_{t_n}$. Pod względem propagacji błędu *metoda leap–frog zachowuje się jak metoda pół-niejawna!* Ostatecznie macierz wzomocnienia ma postać

$$\mathbf{G} = \begin{bmatrix} \mathbb{I} & h\,\mathbf{J} \\ h\mathbb{I} & \mathbb{I} + h^2\,\mathbf{J} \end{bmatrix}. \tag{36}$$

Dla przypadku jednowymiarowego otrzymujemy stąd

$$g_{\pm} = \frac{2 + \lambda \pm \sqrt{4\lambda + \lambda^2}}{2}, \tag{37}$$

gdzie, jak poprzednio, $\lambda = da/dx|_{t_n} h^2$. Jest to identyczne z analogicznym wyrażeniem dla metody Verleta[‖], a zatem własności stabilności metody leap-frog i metody Verleta są — co najmniej w przypadku jedowymiarowym — identyczne.

[‖]Co nie dziwi wobec *algebraicznej* równoważności tych metod.

Cechy algorytmu leap–frog:

- ☺ Różnica dwu dużych wielkości nie jest porównywana z małą.
- ☺ Prędkość w chwili t_n jest znana w chwili t_n, nie dopiero w chwili t_{n+1}.
- ☹ Algorytm wymaga inicjalizacji (na przykład metodą Eulera z krokiem połówkowym).
- ☹ Przyspieszenia (siły) nie mogą zależeć od prędkości.
- ☹ Prędkości nadal obsługiwane są dość dziwacznie.

Algorytm *velocity Verlet*

Problemy z dziwacznym tarktowaniem prędkości rozwiązuje kolejna modyfikacja algorytmu Verleta, zwana *velocity Verlet* (Swope et. al. 1982):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\,\mathbf{v}_n + \tfrac{1}{2}h^2\,\mathbf{a}_n\,, \tag{38a}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \tfrac{1}{2}h\left(\mathbf{a}_n + \mathbf{a}_{n+1}\right). \tag{38b}$$

Algorytm ten nie nadaje się do sił (przyspieszeń) zależnych od prędkości.

Uogólniony algorytm *velocity Verlet*

Okazuje się, że algorytm *velocity Verlet* można łatwo uogólnić (PFG):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\,\mathbf{v}_n + \alpha h^2\,\mathbf{a}_n\,, \tag{39a}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\left(\alpha\,\mathbf{a}_n + (1-\alpha)\mathbf{a}_{n+1}\right), \tag{39b}$$

gdzie $\alpha \in [0, 1]$.

Zgodność algorytmu (39) z równaniem (17) pokażemy eleiminując z (39) prędkości. Odejmując od pierwszego z równań (39) to samo równanie cofnięte o krok w czasie, dostajemy

$$\mathbf{x}_{n+1} = 2\mathbf{x}_n - \mathbf{x}_{n-1} + h\left(\mathbf{v}_n - \mathbf{v}_{n-1}\right) + \alpha h^2\left(\mathbf{a}_n - \mathbf{a}_{n-1}\right). \qquad (40)$$

Z kolei na mocy drugiego z równań (39),

$$\mathbf{v}_n - \mathbf{v}_{n-1} = h\left(\alpha\,\mathbf{a}_{n-1} - (1-\alpha)\mathbf{a}_n\right). \qquad (41)$$

Ostatecznie

$$\begin{aligned} \mathbf{x}_{n+1} &= 2\mathbf{x}_n - \mathbf{x}_{n-1} + h^2 \left(\alpha\, \mathbf{a}_n - \alpha\, \mathbf{a}_{n-1} + \alpha\, \mathbf{a}_{n-1} + (1-\alpha)\mathbf{a}_n \right) \\ &= 2\mathbf{x}_n - \mathbf{x}_{n-1} + h^2 \mathbf{a}_n \,, \end{aligned} \tag{42}$$

a więc znów dostajemy algebraiczną równoważność (uogólnionego) algorytmu velocity Verlet i klasycznego algorytmu Verleta.

Propagacja błędu dla sił niezależnych od prędkości ma postać

$$\begin{aligned} \boldsymbol{\varepsilon}_{n+1} &= \boldsymbol{\varepsilon}_n + h\boldsymbol{\eta}_n + \alpha h^2\,\mathbf{J}\boldsymbol{\varepsilon}_n\,, && \text{(43a)} \\ \boldsymbol{\eta}_{n+1} &= \boldsymbol{\eta}_n + h\alpha\,\mathbf{J}\boldsymbol{\varepsilon}_n + h(1-\alpha)\widetilde{\mathbf{J}}\boldsymbol{\varepsilon}_{n+1}\,, && \text{(43b)} \end{aligned}$$

$\mathbf{J}$ jest jakobianem w kroku n, $\widetilde{\mathbf{J}}$ jest jakobianem w kroku $n+1$. Jako macierz wzmocnienia dostajemy

$$\mathbf{G} = \begin{bmatrix} \mathbb{I} + \alpha h^2\,\mathbf{J} & h\mathbb{I} \\ \alpha h\,\mathbf{J} + (1-\alpha)h\widetilde{\mathbf{J}}(\mathbb{I} + \alpha h^2\,\mathbf{J}) & \mathbb{I} + (1-\alpha)h^2\,\widetilde{\mathbf{J}} \end{bmatrix}. \qquad \text{(44)}$$

W przypadku jedowymiarowym, niezależnie od wartości parametru α, dostajemy takie same współczynniki wzmocnienia, jak w metodach Verleta i leap–frog, a zatem własności stabilności uogólnionej metody velocity Verlet są takie same, jak i innych metod Verleta.

Dla $\alpha = 1$ algorytm (39) można stosować gdy przyspieszenia zależą od prędkości!

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\,\mathbf{v}_n + h^2\,\mathbf{a}_n\,, \tag{45a}$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\,\mathbf{a}_n\,, \tag{45b}$$

co można też zapisać jako

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\,\mathbf{a}_n\,, \tag{46a}$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\,\mathbf{v}_{n+1}\,. \tag{46b}$$

W tym wypadku macierz wzmocnienia ma postać

$$\mathbf{G} = \begin{bmatrix} \mathbb{I} + h^2\,\mathbf{J}_\mathbf{x} & h\,\mathbb{I} + h^2\,\mathbf{J}_\mathbf{v} \\ h\,\mathbf{J}_\mathbf{x} & \mathbb{I} + h\,\mathbf{J}_\mathbf{v} \end{bmatrix}, \tag{47}$$

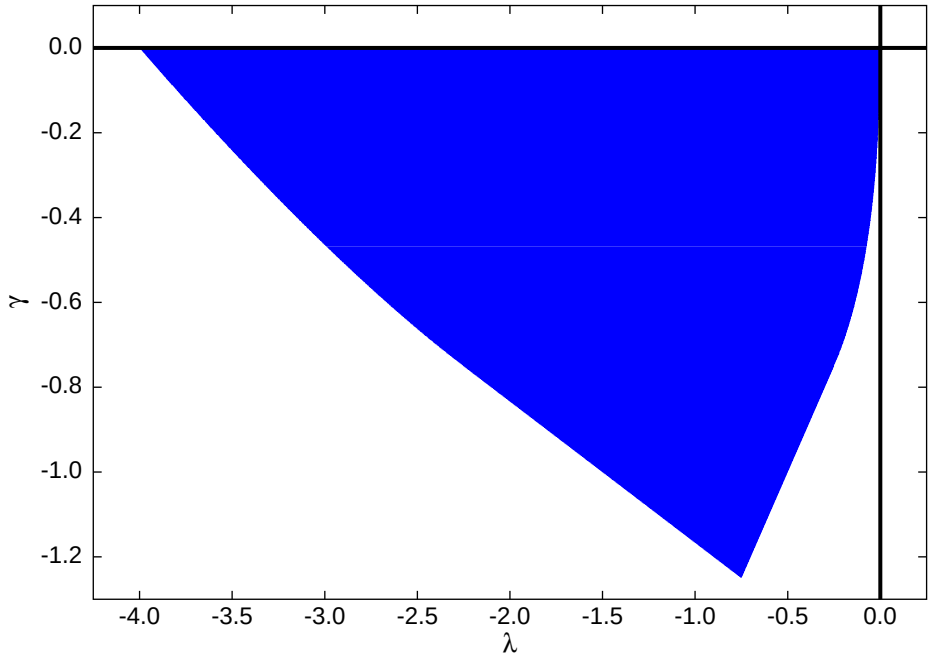
gdzie $\mathbf{J_x}$, $\mathbf{J_v}$ oznaczają, odpowiednio, jakobiany po położeniach i prędkościach.

W przypadku jednowymiarowym dostajemy

$$g_{\pm} = \frac{1}{2}\left(2 + \lambda + \gamma \pm \sqrt{4\lambda + (\lambda+\gamma)^2}\right), \tag{48}$$

gdzie $\lambda = \partial a/\partial x|_{t_n} h^2$, $\gamma = \partial a/\partial v|_{t_n} h$. Dla $\gamma = 0$ (przyspieszenia nie zależą od prędkości) daje to oczywiście to samo, co inne metody Verleta, czyli marginalną stabilność dla $\lambda \in [-4, 0], \gamma = 0$. Dla $\gamma < 0$ i $\lambda < 0$ otrzymujemy ograniczony obszar absolutnej stabilności.

Obszar stabilności dla uogólnionej metody *velocity Verlet* w przypadku jednowymiarowym



Inna modyfikacja algorytmu *velocity Verlet*

Opublikowano kilka modyfikacji algorytmu Verleta przeznaczonych do rozwiązywania równań z przyspieszeniami (siłami) zależnymi od prędkości. Najczęściej używana pochodzi z pracy R. D. Groot, P. B. Warren, *Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulations*, J. Chem. Phys. **107**, 4423 (1997). Jest to algorytm typu *predyktor-korektor*:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{v}_n + \tfrac{1}{2}h^2\mathbf{a}_n\,, \tag{49a}$$

predyktor:

$$\widetilde{\mathbf{v}} = \mathbf{v}_n + \beta h\mathbf{a}_n\,, \tag{49b}$$

$$\widetilde{\mathbf{a}} = \mathbf{a}\left(\mathbf{x}_{n+1}, \widetilde{\mathbf{v}}\right), \tag{49c}$$

korektor:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \tfrac{1}{2}h\left(\mathbf{a}_n + \widetilde{\mathbf{a}}\right), \tag{49d}$$

gdzie $\beta \in [0, 1]$. W przeciwieństwie do algorytmu (45), algorytm (49) wymaga dwu obliczeń przyspieszenia na krok czasowy.