Notatki do wykładu Metody Numeryczne 1

Iwona Wróbel, 15.05.2021r.

Na podstawie notatek dr. Adama Grabarskiego, dr. Pawła Kellera, własnych notatek oraz innych źródeł.

Ważne: Notatki te zawierają część zagadnień omawianych na wykładzie z Metod Numerycznych 1.
Są przygotowywane w związku z zawieszeniem stacjonarnych zajęć, w sytuacji, kiedy niestety nie ma dość czasu, aby wszystko dopracować z należytą starannością. Dlatego mogą zawierać błędy (choć, mam nadzieję, niezbyt wiele) i dlatego ich forma nie zawsze jest dostatecznie zadowalająca pod względem edytorskim i graficznym.

Zmiany w stosunku do wcześniejszej wersji:
- dodany podrozdział 1.8 (14.03.2021r.),
- dodane rozdziały 2 i 3 (20.03.2021r.),
- dodany podrozdział 3.5 (25.03.2021r.),
- dodane podrozdziały 3.5.1 i 3.6 oraz rozdział 4 (10.04.2021r.),
- dodany podrozdział 4.6 oraz rozdział 5 (17.04.2021r.),
- dodane podrozdziały 5.2.1 i 5.2.2 (25.04.2021r.),
- dodane podrozdziały 5.3 i 5.4 (2.05.2021r.),
- dodane podrozdziały 5.5-5.9 i rozdział 6 (15.05.2021r.).

Spis treści

1 Błędy w obliczeniach numerycznych

Błędy, ze względu na źródło ich powstawania, możemy podzielić na:

- błędy pomiarów (błędy danych zadania) – nimi się nie będziemy zajmować,
- błędy zaokrągleń,
- błędy metod.

1.1 Błąd bezwzględny i błąd względny

Wprowadźmy oznaczenia:
$x \in \mathbb{R}$ – wartość dana dokładnie,
$\tilde{x} \in \mathbb{R}$ – przybliżenie x.

błąd bezwzględny: $\Delta = \tilde{x} - x$,
błąd względny (dla $x \neq 0$): $\delta = \frac{\tilde{x}-x}{x} = \frac{\Delta}{x}$,

Z określenia tych błędów wynika, że

$$\tilde{x} = x + \Delta, \qquad \tilde{x} = x(1 + \delta).$$

Najczęściej nie znamy dokładnych wartości błędu bezwzględnego i względnego. W praktyce korzystamy z oszacowań tych błędów:

$$|\Delta| \leq \varepsilon, \qquad |\delta| \leq \varepsilon.$$

1.2 Uwarunkowanie zadania numerycznego

Zadanie numeryczne to problem obliczeniowy, w którym dane (argumenty wejściowe) i wyniki (argumenty wyjściowe) są liczbami. Ogólnie, zadanie numeryczne to funkcja $\varphi : D \to W$, gdzie
$D \in \mathbb{R}^n$ – zbiór możliwych danych dla zadania numerycznego φ,
$W \in \mathbb{R}^m$ – zbiór możliwych wyników dla zadania numerycznego φ.

1.2.1 Błąd wartości funkcji

Spróbujmy ocenić w przybliżeniu błąd bezwzględny i błąd względny, jaki popełniamy, obliczając wartość funkcji

$$y = f(x),$$

gdzie $f : \mathbb{R} \to \mathbb{R}$. Założymy przy tym, że przyjęty do obliczeń argument x jest dany niedokładnie, a oszacowanie błędu δ jest niewielkie.

Załóżmy, że funkcja f ma ciągłe pochodne do rzędu drugiego włącznie na pewnym otoczeniu punktu x oraz $\tilde{x} = x(1+\delta) = x + \delta x$ jest punktem z tego otoczenia. Wtedy, korzystając z rozwinięcia f w szereg Taylora, możemy zapisać

$$f(\tilde{x}) = f(x) + f'(x)(\tilde{x} - x) + \frac{1}{2}f''(\xi)(\tilde{x} - x)^2,$$

dla pewnego ξ, przy czym $\xi \in (x, \tilde{x})$, jeśli $x < \tilde{x}$ lub $\xi \in (\tilde{x}, x)$, jeśli $\tilde{x} < x$.
Stąd

$$f(\tilde{x}) - f(x) = f'(x)(\tilde{x} - x) + \frac{1}{2}f''(\xi)(\tilde{x} - x)^2.$$

Dla dostatecznie małego $(\tilde{x} - x)$ wartość $(\tilde{x} - x)^2$ jest dużo mniejsza niż $(\tilde{x} - x)$, można zatem przyjąć, że

$$f(\tilde{x}) - f(x) \approx f'(x)(\tilde{x} - x).$$

Stąd, jeśli $f(x) \neq 0$, mamy

$$\frac{|f(\tilde{x}) - f(x)|}{|f(x)|} \approx \frac{|f'(x)|}{|f(x)|}|\tilde{x} - x| = \frac{|f'(x)|}{|f(x)|}|\delta x| = \frac{|xf'(x)|}{|f(x)|}|\delta|.$$

Pomińmy środkowe fragmenty powyższego przekształcenia i zapiszmy ponownie zależność, którą otrzymaliśmy:

$$\frac{|f(\tilde{x}) - f(x)|}{|f(x)|} \approx \frac{|xf'(x)|}{|f(x)|}|\delta|.$$

Po lewej stronie mamy błąd względny wartości funkcji (czyli wyniku obliczeń), a po prawej błąd względny argumentu przemnożony przez wielkość $\frac{|xf'(x)|}{|f(x)|}$. Wielkość ta jest zatem mnożnikiem, z jakim błąd danych przenosi się na wynik. Oznaczana jest przez $c(x)$ i nazywana wskaźnikiem uwarunkowania obliczania wartości funkcji f. Mamy zatem:

$$c(x) = \frac{|xf'(x)|}{|f(x)|}. \tag{1.1}$$

Wskaźnik uwarunkowania zależy zarówno od funkcji (ogólniej: od zadania obliczeniowego), jak i argumentu x (czyli danych wejściowych zadania). Dla pewnych funkcji może nie zależeć od x, ale to są szczególne przypadki.
Wskaźnik uwarunkowania mierzy wpływ błędu względnego danych wejściowych na błąd wyniku.

Mówimy, że zadanie numeryczne jest dobrze uwarunkowane, jeśli niewielka zmiana danych wejściowych powoduje niewielką zmianę wyniku (niewielki błąd w danych nie zmieni wyniku w sposób znaczący). W przeciwnym wypadku zadanie jest źle uwarunkowane.

Uwaga 1.1. *Nie ma ścisłej granicy (konkretnej wartości wskaźnika uwarunkowania), przy której kończy się dobre uwarunkowanie, a zaczyna złe. Im wskaźnik uwarunkowania mniejszy, tym zadanie jest lepiej uwarunkowane.*

Ponieważ uwarunkowanie rozważa się w kontekście obliczeń numerycznych, gdzie bierze się pod uwagę błędy (danych oraz wyniku) wynikające z zaokrągleń, a większość systemów obliczeniowych korzysta z arytmetyki podwójnej precyzji, gdzie błąd pojedynczego zaokrąglenia jest rzędu $1e-16$ *(czyli* 10^{-16}*), najczęściej przyjmuje się, że jeśli wskaźnik uwarunkowania nie przekracza 100, a nawet 1000, to zadanie jest dobrze uwarunkowane.*

Przykład 1.1.

Zbadaj uwarunkowanie obliczania wartości funkcji $f(x) = x^2 - 1$, $x \in \mathbb{R}$?

Rozwiązanie:
Funkcja $f(x) = x^2 - 1$ jest różniczkowalna, zatem możemy skorzystać ze wzoru (1.1). Po podstawieniu otrzymujemy

$$C(x) = \left| \frac{2x^2}{x^2 - 1} \right|.$$

Wskaźnik uwarunkowania $C(x)$ jest duży dla x bliskiego 1 lub -1. Oznacza to, że dla $x \approx 1$ lub $x \approx -1$ małe względne zaburzenie danych wejściowych (czyli argumentu x) może powodować duży błąd względny wyniku (czyli obliczonej wartości $f(x)$). Zatem dla argumentów bliskich 1 lub -1 zadanie obliczania wartości funkcji f jest źle uwarunkowane. Dla x dalekich od zera wskaźnik uwarunkowania jest mały ($\lim_{x\to\pm\infty} C(x) = 2$), zatem dla argumentów o dużym module uwarunkowanie jest dobre.

Ważne! Uwarunkowanie jest własnością problemu obliczeniowego. Nie jest związane z algorytmem (sposobem), jakim ten problem rozwiązujemy. W przypadku obliczania wartości funkcji, sposób zapisania wzoru tej funkcji nie wpływa na uwarunkowanie. Na przykład, funkcję z powyższego przykładu można zapisać jako $f(x) = x^2 - 1$ lub $f(x) = (x - 1)(x + 1)$ (lub jeszcze inaczej). Uwarunkowanie pozostaje takie samo.

Przykład 1.2.

Zbadaj uwarunkowanie obliczania wartości funkcji $f(x) = x^2 + 1$, $x \in \mathbb{R}$?

Rozwiązanie:
Postępując podobnie, jak w Przykładzie 1, otrzymujemy

$$C(x) = \left| \frac{2x^2}{x^2 + 1} \right| = 2\left| \frac{x^2 + 1}{x^2 + 1} - \frac{1}{x^2 + 1} \right| = 2\left| 1 - \frac{1}{x^2 + 1} \right| < 2.$$

Wskaźnik uwarunkowania jest ograniczony i nie przekracza 2, niezależnie od wartości x. Zatem zadanie obliczania wartości funkcji f jest dobrze uwarunkowane dla każdego argumentu x.

Uwarunkowanie obliczania wartości funkcji wielu zmiennych bada się podobnie jak w przypadku funkcji jednej zmiennej, z tym, że rozważamy częściowe wskaźniki uwarunkowania, ze względu na każdą ze zmiennych.
Niech $f : \mathbb{R}^2 \to \mathbb{R}$, czyli $f = f(x_1, x_2)$. Mamy wtedy dwa częściowe wskaźniki uwarunkowania

$$c_1(x_1, x_2) = \frac{|x_1 \frac{\partial f}{\partial x_1}(x_1, x_2)|}{|f(x_1, x_2)|}, \qquad c_2(x_1, x_2) = \frac{|x_2 \frac{\partial f}{\partial x_2}(x_1, x_2)|}{|f(x_1, x_2)|}, \tag{1.2}$$

gdzie $\frac{\partial f}{\partial x_k}(x_1, x_2)$ oznacza pochodną cząstkową funkcji f względem zmiennej x_k, $k = 1, 2$. O uwarunkowaniu obliczania wartości funkcji f świadczą oba te wskaźniki, zatem łączny wskaźnik uwarunkowania można zdefiniować jako ich sumę

$$C(x_1, x_2) = c_1(x_1, x_2) + c_2(x_1, x_2).$$

Formalnie można powyższe wzory uzasadnić korzystając z rozwinięcia Taylora funkcji wielu zmiennych.

W przypadku funkcji większej liczby zmiennych wzory są analogiczne.

Przykład 1.3.

Zbadaj uwarunkowanie obliczania wartości funkcji

$$f(x_1, x_2) = x_1^2 + x_2, \quad (x_1, x_2) \in \mathbb{R}^2?$$

Rozwiązanie:
Obliczamy wskaźniki uwarunkowania ze względu na każdą ze zmiennych, korzystając ze wzorów (1.2). Otrzymujemy:

$$c_1(x_1, x_2) = \left|\frac{2x_1^2}{x_1^2 + x_2}\right|, \qquad c_2(x_1, x_2) = \left|\frac{x_2}{x_1^2 + x_2}\right|,$$

a następnie:

$$C(x_1, x_2) = \frac{2x_1^2 + |x_2|}{|x_1^2 + x_2|}.$$

Wskaźnik uwarunkowania jest duży, gdy $x_1^2 \approx -x_2$. Zatem dla takich argumentów zadanie obliczania wartości funkcji f jest źle uwarunkowane. Natomiast, gdy wartość $|x_1^2 + x_2|$ nie jest zbyt bliska 0, uwarunkowanie jest dobre. W szczególności, dla $x_2 > 0$,

$$C(x_1, x_2) = \frac{2x_1^2 + x_2}{x_1^2 + x_2} < \frac{2x_1^2 + 2x_2}{x_1^2 + x_2} = 2,$$

z czego wynika, że błąd względny wyniku (obliczonej wartości funkcji f) będzie co najwyżej 2 razy większy niż błąd względny argumentów.

W przypadku, gdy $f : \mathbb{R}^n \to \mathbb{R}$, gdzie $n > 2$, wskaźnik uwarunkowania oblicza się analogicznie jak w przypadku $f : \mathbb{R}^2 \to \mathbb{R}$. W przypadku innych zadań numerycznych $\varphi \colon \mathbb{R}^n \to \mathbb{R}^m$, gdzie $n, m \geq 1$, zbadanie uwarunkowania może być bardziej skomplikowane.

1.3 Błędy działań arytmetycznych

1.3.1 Dodawanie i odejmowanie

Dodawanie lub odejmowanie można wyrazić za pomocą funkcji $f(x_1, x_2) = x_1 + x_2$, gdzie $x_1, x_2 \in \mathbb{R}$. Będziemy rozważać błąd względny wartości tej funkcji, zatem załóżmy, że $x_1 + x_2 \neq 0$. Niech $\tilde{x}_i = x_i(1 + \delta_i)$, $|\delta_i| \leq \varepsilon$ dla $i = 1, 2$. Mamy

$$\begin{aligned}\left|\frac{f(x_1,x_2) - f(\tilde{x}_1,\tilde{x}_2)}{f(x_1,x_2)}\right| &= \left|\frac{x_1 + x_2 - \tilde{x}_1 - \tilde{x}_2}{x_1 + x_2}\right| = \left|\frac{x_1 + x_2 - x_1(1+\delta_1) - x_2(1+\delta_2)}{x_1 + x_2}\right| = \\ &= \left|\frac{x_1 + x_2 - x_1 - x_1\delta_1 - x_2 - x_2\delta_2}{x_1 + x_2}\right| = \left|\frac{x_1\delta_1 + x_2\delta_2}{x_1 + x_2}\right| = \\ &= \left|\delta_1\frac{x_1}{x_1 + x_2} + \delta_2\frac{x_2}{x_1 + x_2}\right| \leq |\delta_1|\left|\frac{x_1}{x_1 + x_2}\right| + |\delta_2|\left|\frac{x_2}{x_1 + x_2}\right| \leq \\ &\leq \frac{|x_1| + |x_2|}{|x_1 + x_2|}\,\varepsilon.\end{aligned}$$

Jeśli $|x_1 + x_2|$ ma małą wartość w stosunku do $|x_1|$ i $|x_2|$, to błąd względny wyniku może być bardzo duży, nawet jeśli błąd danych (czyli δ_1 i δ_2) jest niewielki. Zatem, gdy $x_1 \approx -x_2$, błąd może się bardzo wzmocnić.

Odejmowanie bliskich liczb jest operacją, która może być źródłem dużych błędów.

Jeśli x_1 i x_2 są tego samego znaku, to $\frac{|x_1|+|x_2|}{|x_1+x_2|} = 1$, i w konsekwencji

$$\left|\frac{f(x_1,x_2) - f(\tilde{x}_1,\tilde{x}_2)}{f(x_1,x_2)}\right| \leq \varepsilon,$$

co oznacza, że błąd względny wyniku jest na poziomie błędu względnego danych. (To bardzo dobrze – operacja nie powoduje wzmocnienia błędu.)

Do podobnych wniosków można dojść obliczając wskaźnik uwarunkowania z wykorzystaniem wzorów podanych podrozdziale 1.2.

1.3.2 Mnożenie

Mnożenie można wyrazić za pomocą funkcji $f(x_1, x_2) = x_1\, x_2$, gdzie $x_1, x_2 \in \mathbb{R}$. Założymy, analogicznie jak poprzednio, że $x_1 x_2 \neq 0$. Ponownie, niech $\tilde{x}_i = x_i(1 + \delta_i)$, $|\delta_i| \leq \varepsilon$ dla $i = 1, 2$. Mamy

$$\begin{aligned}\left|\frac{f(x_1,x_2) - f(\tilde{x}_1,\tilde{x}_2)}{f(x_1,x_2)}\right| &= \left|\frac{x_1x_2 - \tilde{x}_1\tilde{x}_2}{x_1x_2}\right| = \left|\frac{x_1x_2 - x_1(1+\delta_1)x_2(1+\delta_2)}{x_1x_2}\right| = \\ &= |1 - (1+\delta_1)(1+\delta_2)| = |\delta_1 + \delta_2 + \delta_1\delta_2| \approx |\delta_1 + \delta_2| \leq \\ &\leq\ 2\varepsilon.\end{aligned}$$

Błąd względny wyniku zależy od błędów argumentów i jest co najwyżej 2 razy większy od maksymalnego z tych błędów. Nie zależy natomiast od samych argumentów, co jest bardzo wygodną własnością.

1.3.3 Dzielenie

Do uzupełnienia samodzielnie.
Wskazówka: można skorzystać z tego, że $(1+\delta)(1-\delta) = 1-\delta^2 \approx 1$, a zatem $\dfrac{1}{1+\delta} \approx 1-\delta$. (To przybliżenie jest prawdziwe dla małych wartości $|\delta|$, a tutaj mamy właśnie taki przypadek).

1.4 Reprezentacja maszynowa liczb zmiennoprzecinkowych

Twierdzenie 1.1. *Każda liczba rzeczywista $x \neq 0$ posiada jednoznaczne przedstawienie*

$$x = sm2^c,$$

gdzie $s \in \{-1, 1\}$, $1 \leq m < 2$, a $c \in \mathbb{Z}$ (c jest liczbą całkowitą).

Mantysę m można zapisać jako

$$m = \sum_{i=0}^{\infty} d_i 2^{-i}, \tag{1.3}$$

gdzie $d_i \in \{0, 1\}$ dla wszystkich $i = 0, 1, \ldots$. Liczba bitów przeznaczona na zapisanie mantysy w pamięci komputera jest skończona, zatem zachodzi konieczność zaokrąglenia. Oznaczmy tę liczbę przez t, a przez m_t mantysę zaokrągloną do $t+1$ bitów. Zaokrągloną mantysę można zapisać jako

$$m_t = \sum_{i=0}^{t} d_i 2^{-i} + d_{t+1} 2^{-t}, \tag{1.4}$$

Zauważmy, że z warunku $1 \leq m < 2$ wynika, że $d_0 = 1$ w (1.3), co oznacza, że nie ma konieczności przechowywania wiodącego bitu m_t. Zatem do przechowania zaokrąglonej mantysy m_t wystarcza t bitów.

Liczba rzeczywista $x \neq 0$ przechowywana jest w pamięci komputera jako

$$\mathrm{rd}(x) = sm_t 2^c. \tag{1.5}$$

Można pokazać, że $|m_t - m| \leq 2^{-t-1}$ oraz, że

$$\mathrm{rd}(x) = x(1+\delta),$$

gdzie δ jest błędem względnym reprezentacji, przy czym $|\delta| \leq 2^{-t-1}$. Liczba $u = 2^{-t-1}$ jest zatem ograniczeniem górnym na błąd pojedynczego zaokrąglenia (ang. *unit round-off error*). Wartość $\varepsilon_M = 2^{-t}$ nazywana jest *precyzją obliczeń*. Liczba $1+\varepsilon_M$ jest najmniejszą liczbą maszynową większą niż 1. *Liczba maszynowa* to liczba, która posiada dokładną reprezentację (w danej arytmetyce), tj. x jest liczbą maszynową, jeśli $x = \mathrm{rd}(x)$.

Uwaga 1.2. *Wzór* (1.4) *jest wykorzystywany w rozważaniach teoretycznych. W praktyce w procesorach zaokrąglenia mogą się odbywać nieco inaczej:*

- *jeśli* $m = \sum_{i=0}^{t} d_i 2^{-i} + 2^{-t-1}$, *to oba zaokrąglenia (w górę i w dół),*

$$m_t = \sum_{i=0}^{t} d_i 2^{-i} + 2^{-t} \quad i \quad \widehat{m_t} = \sum_{i=0}^{t} d_i 2^{-i},$$

są równie dobre ($|m_t - m| = |\widehat{m_t} - m| = 2^{-t-1}$*). Według standardu IEEE 754 należy wybrać to z powyższych zaokrągleń, w którym najmniej znaczący bit jest równy* 0*.*

- *jeśli* $m = \sum_{i=0}^{t+1} 2^{-i} + \sum_{i=t+2}^{\infty} d_i 2^{-i}$*, to z* (1.4) *wynika, że* $m_t = 2$*. W komputerze* m_t *otrzymuje wtedy wartość* 1*, a wykładnik c w* (1.5) *jest zwiększany o* 1*.*

1.5 Własności podstawowych operacji arytmetycznych

Niech fl(*wyrażenie*) oznacza wartość *wyrażenia* obliczoną za pomocą komputera. Arytmetyka większości procesorów ma tę własność, że jeśli $x = \mathrm{rd}(x)$ oraz $y = \mathrm{rd}(y)$ (czyli x i y są dokładnie reprezentowane), to $\mathrm{fl}(x \bullet y) = \mathrm{rd}(x \bullet y)$, gdzie $\bullet$ oznacza jedną (dowolną) z czterech podstawowych operacji arytmetycznych: $+, -, *, /$. Podobnie, zachodzi też $\mathrm{fl}(\sqrt{x}) = \mathrm{rd}(\sqrt{x})$. Oznacza to, że błąd arytmetyki jest taki jak błąd reprezentacji wyniku (porównaj podrozdział 1.6).

1.6 Elementy rachunku błędów

W analizie algorytmów bada się, jaki wpływ na wynik mają błędy względne argumentów oraz błędy generowane przez działania arytmetyczne.
Notacja $\tilde{x} = x(1+\delta)$ jest bardzo wygodna w analizie błędów. Zbadajmy dla przykładu przenoszenie błędów przy obliczaniu iloczynu trzech liczb: $\mathrm{fl}(xyz)$. Przyjmijmy dla uproszczenia, że dane są reprezentowane dokładnie. Mamy

$$\begin{aligned} \mathrm{fl}(xyz) &= \big(xy(1+\delta_1)z\big)(1+\delta_2) = xyz(1+\delta_1)(1+\delta_2) \\ &= xyz(1+\delta_1+\delta_2+\delta_1\delta_2). \end{aligned}$$

Błąd δ_1 jest błędem mnożenia x przez y, a błąd δ_2 jest błędem mnożenia otrzymanego wyniku mnożenia x przez y przez z. Każde działanie jest źródłem błędu.
Jeśli $|\delta_1|, |\delta_2| \le u$, to

$$\mathrm{fl}(xyz) = xyz(1+\delta),$$

gdzie $|\delta| \le 2u + \mathcal{O}(u^2)$. Zauważmy, że u^2 jest dużo mniejsze niż u. (Na przykład, jaka jest wartość u, gdy $t = 52$, czyli przy podwójnej precyzji?) Zatem błąd otrzymanego wyniku jest zawsze bardzo mały.
Czasem, dla uproszczenia rachunku błędów tego typu, pomija się iloczyny dwóch lub więcej błędów. (Można tak będzie zrobić w jednym z zadań na liście C.)

Uwaga 1.3. *W zbiorze liczb rzeczywistych zachodzą prawa łączności, przemienności i rozdzielności. W zbiorze liczb maszynowych mamy wyłącznie przemienność dodawania i odejmowania (porównaj przykład 1.4). Wynika stąd, że algorytmy matematycznie równoważne mogą dawać istotnie różne wyniki.*

1.7 Problem utraty cyfr znaczących

Z własności operacji arytmetycznych opisanych w podrozdziale 1.5 nie wynika, że wszystkie obliczenia wykonywane na komputerze są dokładne. Załóżmy, że x i y nie są liczbami maszynowymi. Wtedy,

$$\begin{aligned}\mathrm{fl}(x+y) &= \mathrm{fl}(\mathrm{rd}(x)+\mathrm{rd}(y)) = \big(x(1+\delta_1)+y(1+\delta_2)\big)(1+\delta_3)\\ &= x+y+x(\delta_1+\delta_3)+y(\delta_2+\delta_3)+x\delta_1\delta_3+y\delta_2\delta_3,\end{aligned}$$

gdzie $|\delta_1|, |\delta_2|, |\delta_3| < u$. Jeśli $|\delta_1 - \delta_2| \approx 2u$, to (w pewnym sensie)

$$\lim_{x\to -y} \frac{|\mathrm{fl}(x+y)-(x+y)|}{|x+y|} \approx \lim_{x\to -y} \frac{|2xu|}{|x+y|} = \infty.$$

Problem utraty cyfr znaczących pojawia się przy dodawaniu lub odejmowaniu (dwóch lub więcej) składników, przy czym oczekiwany (dokładny) wynik tego działania jest dużo mniejszy niż składniki. Jeśli przez T oznaczymy największy (co do modułu) składnik, a przez R dokładny (prawdziwy) wynik, to – z grubsza – można oczekiwać, że błąd względny wyniku jest w przybliżeniu $|R|^{-1}|T|$ razy większy niż błąd względny argumentów.

Przykład 1.4. *Przykład ten pochodzi z pozycji [9].*
Rozważmy liczby, które można zapisać w systemie dziesiętnym w postaci $x = sm_5 10^{c_1}$, tj. na zapisanie mantysy mamy 5 bitów, a na zapisanie cechy 1 bit. Znak liczby jest zapisywany na dodatkowym bicie.
W przykładzie tym podstawą jest 10, a nie 2, dla większej czytelności wyników.
Rozważmy działanie

$$0.98765 + 0.012424 - 0.0065432,$$

którego dokładnym wynikiem jest $0.\underline{9935}308$. *Ponieważ do zapisania mantysy mamy tylko 5 bitów, zapamiętane mogą być tylko podkreślone cyfry. Zaokrąglenie odbywa się w standardowy sposób.*
Sprawdźmy, jaki wpływ ma kolejność działań na wynik. Operacje wykonywane w obecności błędów zaokrągleń są wzięte w kółko. Każda z tych operacji obarczona jest błędem (wynik każdej operacji jest zaokrąglany, zanim zostanie przekazany do dalszych obliczeń). Wykonując powyższe działanie na trzech liczbach możemy postawić nawiasy na dwa sposoby.

Mamy:

$$0.98765 \oplus (0.012424 \ominus 0.0065432) = \\ 0.98765 \oplus 0.00588(08) = 0.99353(08),$$

$$(0.98765 \oplus 0.012424) \ominus 0.0065432 = \\ 1.000074\,(\approx 1.0001) \ominus 0.0065432 = 0.99356.$$

Pierwszy wynik jest dokładny (na tyle, na ile to możliwe przy 5-bitowej mantysie). Drugi wynik już nie. Błąd nie jest bardzo duży (różnica występuje na ostatniej cyfrze), ale widać, że kolejność wykonywania działań ma znaczenie.
Zbadajmy teraz rozdzielność mnożenia względem odejmowania. Rozważymy działanie

$$(4.2832 - 4.2821) * 5.7632,$$

którego dokładnym wynikiem jest $0.00\underline{63395}2$. *Wykonując obliczenia na komputerze (i zaokrąglając pośrednie wyniki), otrzymamy:*

$$(4.2832 \ominus 4.2821) \otimes 5.7632 = \\ 0.0011 \otimes 5.7632 = 0.00\underline{63395}(2),$$

$$(4.2832 \otimes 5.7632) \ominus (4.2821 \otimes 5.7632) = \\ 24.684(93824)\,(\approx 24.685) \ominus 24.678(59872)\,(\approx 24.679) = 0.006\underline{0000}.$$

Przy obliczeniach za pomocą drugiego algorytmu można zaobserwować dużą utratę cyfr znaczących. Uzyskany wynik zawiera tylko jedną poprawną cyfrę. W ostatnim działaniu odejmowane są bliskie liczby, wynik zależy więc od ostatnich bitów. A te zostały utracone przy wcześniejszych obliczeniach.

W najczęściej stosowanych arytmetykach jest więcej bitów na zapisanie cechy i mantysy, niemniej schemat (i wynikające z niego problemy) jest podobny.

Przykład 1.5. *Ciekawe wyniki można uzyskać wykonując w arytmetyce podwójnej precyzji obliczenia z zadania K8 (rozdział 2.2) z pozycji [1]. Warto samodzielnie zaimplementować oba sposoby obliczenia wartości podanej tam funkcji, np. w środowisku Matlab, Octave, C, R, itp.*

1.8 Numeryczna poprawność algorytmów

Oznaczenia:
$D \in \mathbb{R}^n$ – zbiór możliwych danych dla zadania numerycznego φ,
$W \in \mathbb{R}^m$ – zbiór możliwych wyników dla zadania numerycznego φ,
$\varphi : D \to W$ – zadanie numeryczne.

Definicja 1.1. *Algorytm A dla zadania numerycznego φ nazywamy numerycznie poprawnym, jeśli istnieje stała K, niezależna od wskaźnika uwarunkowania ($C(d)$) i arytmetyki (t) taka, że dla dowolnej danej $d \in D$ istnieje dana $\tilde{d} \in D$ taka, że*

$$\|\tilde{d} - d\| \leq K 2^{-t} \|d\| \tag{1.6}$$

oraz

$$fl(A(d)) = \varphi(\tilde{d}). \tag{1.7}$$

Innymi słowy, wynik algorytmu A dla danej d (dokładnej) uzyskany w arytmetyce fl jest dokładnym wynikiem zadania φ dla "nieco" zaburzonej danej $\tilde{d}$.

1.8.1 Oszacowanie błędu algorytmu numerycznie poprawnego

Załóżmy, że algorytm A dla zadania φ jest numerycznie poprawny. Rozpatrując normę różnicy między wynikiem otrzymanym algorytmem A w arytmetyce fl a wynikiem dokładnym otrzymamy:

$$\begin{aligned} \|fl(A(d)) - \varphi(d)\| = \|\varphi(\tilde{d}) - \varphi(d)\| &\leq C(d) \frac{\|\tilde{d} - d\|}{\|d\|} \|\varphi(d)\| \\ &\leq K\, 2^{-t} C(d) \|\varphi(d)\|. \end{aligned}$$

(Pierwsza z powyższych nierówności wynika z definicji wskaźnika uwarunkowania.)
Zatem

$$\frac{\|fl(A(d)) - \varphi(d)\|}{\|\varphi(d)\|} \leq K\, 2^{-t} C(d). \tag{1.8}$$

(Zakładamy, że $\|\varphi(d)\| \neq 0$ i $\|d\| \neq 0$; w przeciwnym razie nie ma sensu rozpatrywać błędów względnych.)

Uwaga 1.4. *Oszacowanie (1.8) zależy od wskaźnika uwarunkowania, zatem, jeśli zadanie jest źle uwarunkowane, wynik może być niedokładny, nawet jeśli stosujemy algorytm numerycznie poprawny.*

Uwaga 1.5. *Przy $t \to \infty$ błąd dąży do zera; ze wzrostem dokładności arytmetyki błąd jest coraz mniejszy.*

Przykład 1.6. *Rozpatrzmy zadanie obliczenia wartości funkcji*

$$f(x, y) = x^2 + y^2 = (x - y)(x + y).$$

Wartość tej funkcji można obliczyć korzystając z któregoś z poniższych algorytmów.

Algorytm 1 ($A1$)
$r = x^2$
$s = y^2$
$f = r - s$

Algorytm 2 ($A2$)
$p = x - y$
$q = x + y$
$f = p\,q$

Na wykładzie zbadamy przenoszenie błędów w przypadku każdego z tych algorytmów przy ich realizacji w arytmetyce fl oraz sprawdzimy czy któryś z tych algorytmów jest numerycznie poprawny.

2 Obliczanie wartości wielomianu i jego pochodnych

Niech

$$f(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + a_n x^n, \tag{2.1}$$

gdzie $a_k \in \mathbb{R}$, dla $k = 1, \ldots, n$.

Naszym celem będzie obliczenie wartości $f(\hat{x})$, gdzie $\hat{x} \in \mathbb{R}$ jest dany.
Można to zrobić na wiele sposobów.
Najprościej jest po prostu podstawić $\hat{x}$ do wzoru (2.1) i wykonać potrzebne obliczenia.
Koszt takiego podejścia, liczony liczbą operacji arytmetycznych, to około $2n$ mnożeń i n dodawań/odejmowań – jeśli skorzystamy z faktu, że $\hat{x}^k = \hat{x}^{k-1}\,\hat{x}$. Jeśli natomiast użyjemy potęgowania (np. funkcji `pow()` w C lub zapisu typu `x^k` w Matlabie), to koszt będzie rzędu n^2.

Zapisując wielomian f w nieco inny niż (2.1) sposób, można uzyskać algorytm o koszcie mniejszym niż (wymienione wyżej) $3n$ operacji arytmetycznych.

2.1 Schemat Hornera

Zapiszmy f jako

$$f(x) = a_0 + x(a_1 + x(\ldots + x(a_{n-2} + x(a_{n-1} + x a_n))\ldots)). \tag{2.2}$$

Obliczenia będziemy wykonywać począwszy od wielomianu znajdującego się w najbardziej zagłębionym nawiasie.
Oznaczmy $w_n = a_n$, i obliczmy $w_{n-1} = a_{n-1} + \hat{x} w_n$. Następnie obliczymy $w_{n-2} = a_{n-2} + \hat{x} w_{n-1}$, itd. Ostatnią obliczoną wartością będzie $w_0 = f(\hat{x})$.

Algorytm 1 (Schemat Hornera)

$w_n = a_n$
for $k = n-1, n-2, \ldots, 0$
    $w_k = a_k + \hat{x}\, w_{k+1}$
end
$f(\hat{x}) = w_0$

Koszt tego algorytmu: n mnożeń i n dodawań (lub odejmowań).
Zauważmy jeszcze, że nie ma konieczności pamiętania wszystkich pośrednich wartości w_k, $k = n, \ldots, 0$. W praktycznej implementacji wystarczy jedna zmienna (liczbowa). Algorytm 1a poniżej to zoptymalizowany w ten sposób algorytm Hornera.

Algorytm 1a (Schemat Hornera)

$w = a_n$
for $k = n-1, n-2, \ldots, 0$
    $w = a_k + \hat{x}\, w$
end
$f(\hat{x}) = w$

2.2 Obliczanie wartości pochodnej wielomianu

Niech $f(x)$ oraz $\hat{x}$ będą zdefiniowane jak w poprzednim podrozdziale.

Zastanowimy się teraz, jak obliczyć wartość $f'(\hat{x})$.
Pierwsza możliwość, jaka się większości osób nasuwa, to zróżniczkować f, zapisany w postaci naturalnej, czyli wzorem (2.1) (otrzymamy wtedy f' w postaci naturalnej), a następnie zastosować schemat Hornera do f'. Jaki będzie koszt takiego algorytmu?

Innym podejściem (może nieco dziwnym na pierwszy rzut oka) będzie zróżniczkowanie wielomianu zapisanego w postaci (2.2). Niemniej zobaczmy, co z tego wyniknie. Korzystając z oznaczeń z poprzedniego podrozdziału, zapiszmy f jako

$$f(x) = a_0 + xw_1,$$

przy czym $w_1 = w_1(x)$, czyli w_1 jest funkcją zmiennej x.

Różniczkując f, korzystając przy tym z wzoru na pochodną iloczyny funkcji, otrzymujemy

$$f'(x) = w_1 + xw_1'.$$

Następnie obliczamy w_1' i dostajemy

$$f'(x) = w_1 + xw_1' = w_1 + x(w_2 + xw_2').$$

Dalej obliczamy w_2', w_3', itd., aż dochodzimy do postaci:

$$f'(x) = w_1 + x(w_2 + x(\ldots + x(w_{n-2} + x(w_{n-1} + xw_n))\ldots)). \tag{2.3}$$

Aby obliczyć $f'(\hat{x})$, postępujemy, jak poprzednio, rozpoczynając obliczenia od wyrażenia w najbardziej wewnętrznym nawiasie.

Algorytm 2 (Algorytm Hornera obliczania $f(\hat{x})$ i $f'(\hat{x})$)

$w_n = a_n$
$p_n = w_n$
for $k = n-1, n-2, \ldots, 1$
    $w_k = a_k + \hat{x}\, w_{k+1}$
    $p_k = w_k + \hat{x}\, p_{k+1}$
end
$w_0 = a_0 + \hat{x}\, w_1$
$f(\hat{x}) = w_0$
$f'(\hat{x}) = p_1$

Koszt: $2n-1$ mnożeń, $2n-1$ dodawań (lub odejmowań).

Podobnie jak poprzednio (tj. w przypadku Algorytmu 1 i 1a), nie ma potrzeby pamiętania wszystkich pośrednich wartości w_k $(k = n, \ldots, 0)$ i p_k $(k = n, \ldots, 1)$.
Algorytm 2a poniżej to zoptymalizowany algorytm Hornera obliczania wartości wielomianu i jego pochodnej.

Algorytm 2a (Algorytm Hornera obliczania $f(\hat{x})$ i $f'(\hat{x})$)

$w = a_n$
$p = w$
for $k = n-1, n-2, \ldots, 1$
    $w = a_k + \hat{x}\, w$
    $p = w + \hat{x}\, p$
end
$w = a_0 + \hat{x}\, w$
$f(\hat{x}) = w$
$f'(\hat{x}) = p$

2.3 Obliczanie wartości drugiej pochodnej wielomianu

Różniczkując f' zapisaną w postaci (2.3) otrzymamy algorytm obliczania wartości $f''(\hat{x})$, $f'(\hat{x})$ i $f(\hat{x})$.

Uwaga: należy pamiętać o tym, że $w_k = w_k(x)$, dla $k = 1, 2, \ldots n$, czyli są to funkcje x (przy różniczkowaniu zajdzie potrzeba skorzystania ze wzoru na pochodną funkcji złożonej).

Algorytm 3a (Algorytm Hornera obliczania $f(\hat{x})$, $f'(\hat{x})$ i $f''(\hat{x})$)

$w = a_n$
$p = w$
$r = p$
for $k = n-1, n-3, \ldots, 2$
    $w := a_k + \hat{x}\, w$
    $p := w + \hat{x}\, p$
    $r := p + \hat{x}\, r$
end
$w := a_1 + \hat{x}\, w$
$p := w + \hat{x}\, p$
$w := a_0 + \hat{x}\, w$
$f(\hat{x}) := w$
$f'(\hat{x}) := p$
$f''(\hat{x}) := 2r$

Jaki jest koszt tego algorytmu?

3 Interpolacja

Sformułowanie problemu:

Mając dane

- $x_i \in \mathbb{R}$, dla $i = 0, 1, \ldots, n$ (węzły interpolacji)
- $f_i = f(x_i)$, gdzie f jest funkcją interpolowaną (nieznaną funkcją, której wartości są znane tylko w skończonej liczbie punktów)

znaleźć funkcję interpolującą p taką, że

$$p(x_i) = f(x_i), \quad \text{dla} \quad i = 0, 1, \ldots, n. \tag{3.1}$$

Funkcja p może mieć dowolną postać, w szczególności może być wielomianem lub przedziałami wielomianem.
W dalszej części zajmiemy się interpolacją wielomianową, czyli taką, gdzie funkcja p jest wielomianem.

3.1 Interpolacja wielomianowa

Zakładamy, że funkcja interpolująca p jest wielomianem.

Niech Π_n będzie przestrzenią wielomianów stopnia nie wyższego niż n, czyli wielomianów postaci $a_0 + a_1 x + \ldots + a_n x^n$, gdzie $a_i \in \mathbb{R}$ dla $i = 0, 1, \ldots, n$.

Twierdzenie 3.1. *Jeśli węzły $x_0, x_1, \ldots, x_n$ są parami różne ($x_i \neq x_j$ dla $i \neq j$, $i, j = 0, \ldots, n$), to dla dowolnych wartości $f_i = f(x_i)$ istnieje dokładnie jeden wielomian $p_n \in \Pi_n$, taki, że*

$$p_n(x_i) = f(x_i), \quad \textit{dla} \quad i = 0, 1, \ldots, n. \qquad \square$$

3.2 Postać Lagrange'a wielomianu interpolacyjnego

Wielomian interpolacyjny jest wyznaczony jednoznacznie, ale można go przedstawić na różne sposoby (w różnych bazach) i wyznaczać różnymi algorytmami. Jednym z możliwych sposobów zapisania wielomianu interpolacyjnego jest postać Lagrange'a:

$$p_n(x) = \sum_{i=0}^{n} f_i l_i(x),$$

gdzie

$$l_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Wielomian interpolacyjny p_n przedstawiony jest w bazie wielomianów $l_0, l_1, \ldots, l_n$. Nazywa się je wielomianami Lagrange'a. Każdy z tych wielomianów jest stopnia n.

Sprawdźmy, czy tak zdefiniowany wielomian p_n spełnia warunki interpolacji, czyli czy $p_n(x_k) = f(x_k)$ dla $k = 0, \ldots, n$. Mamy

$$l_i(x_k) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x_k - x_j}{x_i - x_j} = \begin{cases} 0 & \text{gdy} \quad k \neq i, \\ 1 & \text{gdy} \quad k = i, \end{cases}$$

a stąd

$$p_n(x_k) = \sum_{i=0}^{n} f_i l_i(x_k) = f_k,$$

czyli warunki interpolacji są spełnione.

Przykład 3.1. *Niech będą dane węzły $x_i = i - 1$, $i = 0, 1, 2$, i wartości funkcji $f_0 = 1$, $f_1 = 0$, i $f_2 = 1$. Znaleźć wielomian interpolacyjny w postaci Lagrange'a dla funkcji f.*

Rozwiązanie:
Węzłami są punkty $x_0 = -1$, $x_1 = 0$ i $x_2 = 1$.
Na trzech różnych punktach można skonstruować wielomian interpolacyjny drugiego stopnia. Jego postać Lagrange'a jest następująca:

$$p_2(x) = \sum_{i=0}^{2} f_i l_i(x), \tag{3.2}$$

gdzie

$$l_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{2} \frac{x - x_j}{x_i - x_j}, \qquad i = 0, 1, 2.$$

Wyznaczmy wielomiany bazowe l_i. Dla $i = 0$ mamy

$$l_0(x) = \prod_{j=1}^{2} \frac{x - x_j}{x_0 - x_j} = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{x(x-1)}{-1 \cdot (-2)} = \frac{1}{2}x(x-1).$$

Dla $i = 1$:

$$l_1(x) = \prod_{\substack{j=0 \\ j\neq 1}}^{2} \frac{x - x_j}{x_1 - x_j} = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x+1)(x-1)}{-1} = -(x+1)(x-1),$$

i dla $i = 2$:

$$l_2(x) = \prod_{j=0}^{1} \frac{x - x_j}{x_2 - x_j} = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{x(x+1)}{2} = \frac{1}{2}x(x+1).$$

Wstawiamy wyznaczone wielomiany bazowe l_i $(i = 0, 1, 2)$ do wzoru (3.2) i otrzymujemy

$$p_2(x) = 1 \cdot \frac{1}{2}x(x-1) + 1 \cdot \frac{1}{2}x(x+1) = x^2.$$

Uwagi. Ponieważ $f_1 = 0$, nie było potrzeby wyznaczania l_1. Nie ma także potrzeby sprowadzania wielomianu z postaci Lagrange'a do naturalnej – w przykładzie to zrobiono, aby lepiej było widać, jaką funkcję wyznaczyliśmy.

3.3 Postać Newtona wielomianu interpolacyjnego, schemat ilorazów różnicowych

Postać Newtona wielomianu interpolacyjnego:

$$p_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots + c_n(x - x_0)\ldots(x - x_{n-1}). \tag{3.3}$$

Wielomian p_n jest przedstawiony w bazie $1, x-x_0, (x-x_0)(x-x_1), \ldots, (x-x_0)\ldots(x-x_{n-1})$. Zwróćmy uwagę, że w jego wzorze nie występuje w sposób widoczny węzeł x_n. Węzeł ten jest wykorzystywany przy wyznaczaniu współczynnika c_n.

Zauważmy, że $p_n(x_0) = c_0$. Z drugiej strony, z warunków interpolacji (3.1) mamy $p_n(x_0) = f(x_0)$. Zatem $c_0 = f(x_0)$.
Pozostałe współczynniki $c_1, \ldots, c_n$ są równe
$c_1 = f_{0,1}$,
$c_2 = f_{0,1,2}$,
$\ldots$,
$c_n = f_{0,1,\ldots,n}$,
gdzie $f_{0,1,\ldots,k}$ dla $k = 1, \ldots, n$ są *ilorazami różnicowymi*. Definiuje się je następująco.

Ilorazy różnicowe pierwszego rzędu konstruuje się w oparciu o dwa węzły, zgodnie ze wzorem:

$$f_{j,j+1} = \frac{f(x_{j+1}) - f(x_j)}{x_{j+1} - x_j} \quad \text{dla} \quad j = 0, \ldots, n-1.$$

Ilorazy różnicowe drugiego rzędu definiuje się wykorzystując ilorazy pierwszego rzędu:

$$f_{j,j+1,j+2} = \frac{f_{j+1,j+2} - f_{j,j+1}}{x_{j+2} - x_j} \quad \text{dla} \quad j = 0, \ldots, n-2.$$

Ogólnie, ilorazy różnicowe k-tego rzędu:

$$f_{j,j+1,\ldots,j+k} = \frac{f_{j+1,\ldots,j+k} - f_{j,\ldots,j+k-1}}{x_{j+k} - x_j}$$

dla $j = 0, \ldots, n-k$.

Uwaga 3.1. *Ilorazy różnicowe można różnie oznaczać. Najczęściej stosuje się oznaczenia* f_{0123}, $f_{0,1,2,3}$ *lub* $f[x_0, x_1, x_2, x_3]$. *Zwykle będziemy używać pierwszego lub drugiego z nich (ponieważ są krótsze), ale czasem będziemy też korzystać z trzeciego, tam, gdzie będzie tego wymagała czytelność zapisu.*

Najwygodniej jest obliczać ilorazy różnicowe wykorzystując schemat ilorazów różnicowych, wpisując kolejne obliczone ilorazy różnicowe do tabeli:

$$\begin{array}{llllll}
x_0 & \underline{f_0} & & & & \\
x_1 & f_1 & \underline{f_{01}} = \frac{f_1 - f_0}{x_1 - x_0} & & & \\
x_2 & f_2 & f_{12} = \frac{f_2 - f_1}{x_2 - x_1} & \underline{f_{012}} = \frac{f_{12} - f_{01}}{x_2 - x_0} & & \\
x_3 & f_3 & f_{23} = \frac{f_3 - f_2}{x_3 - x_2} & f_{123} = \frac{f_{23} - f_{12}}{x_3 - x_1} & \underline{f_{0123}} = \frac{f_{123} - f_{012}}{x_3 - x_0} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
x_n & f_n & f_{n-1,n} = \frac{f_n - f_{n-1}}{x_n - x_{n-1}} & f_{n-2,n-1,n} = \frac{f_{n-1,n} - f_{n-2,n-1}}{x_n - x_{n-2}} & f_{n-3,n-2,n-1,n} & \cdots \quad \underline{f_{01\ldots n}}
\end{array}$$

Podkreślone elementy to współczynniki $c_0, \ldots, c_n$ ($c_k = f_{0\ldots k}$, $k = 0, 1, \ldots, n$) wielomianu interpolacyjnego w postaci Newtona (3.3).

Przykład 3.2. *Dla funkcji* $f(x) = x^4$ *znaleźć wielomian interpolacyjny* $p_3(x)$ *w postaci Newtona taki, że* $p_3(-2) = f(-2)$, $p_3(-1) = f(-1)$, $p_3(0) = f(0)$ *i* $p_3(2) = f(2)$.

Rozwiązanie:
Węzłami interpolacji są punkty: $x_0 = -2$, $x_1 = -1$, $x_2 = 0$ i $x_3 = 2$.
Wielomian interpolacyjny w postaci Newtona skonstruowany na czterech węzłach jest stopnia co najwyżej 3 i jest następujący:

$$p_3(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)(x - x_2), \qquad (3.4)$$

gdzie

$$\begin{aligned}
c_0 &= f(x_0), \\
c_1 &= f_{01}, \\
c_2 &= f_{012}, \\
c_3 &= f_{0123}.
\end{aligned}$$

Dla czterech węzłów schemat ilorazów różnicowych ma następującą postać:

$$\begin{array}{lllll}
x_0 & f_0 & \searrow & & \\
x_1 & f_1 & \rightarrow f_{01} & \searrow & \\
x_2 & f_2 & \rightarrow f_{12} & \rightarrow f_{012} & \searrow \\
x_3 & f_3 & \rightarrow f_{23} & \rightarrow f_{123} & \rightarrow f_{0123}
\end{array}$$

Po podstawieniu wartości liczbowych dostajemy:

$$\begin{array}{llllll}
-2 & 16 & & & & \\
 & & \searrow & & & \\
-1 & 1 & \to \frac{1-16}{-1-(-2)} = -15 & \searrow & & \\
 & & \searrow & & \searrow & \\
0 & 0 & \to \frac{0-1}{0-(-1)} = -1 & \to \frac{-1-(-15)}{0-(-2)} = 7 & & \\
 & & \searrow & \searrow & & \searrow \\
2 & 16 & \to \frac{16-0}{2-0} = 8 & \to \frac{8-(-1)}{2-(-1)} = 3 & \to \frac{3-7}{2-(-2)} = -1 &
\end{array}$$

Współczynniki c_i $(i = 0, 1, 2, 3)$ odczytujemy z tabeli: $c_0 = 16$, $c_1 = -15$, $c_2 = 7$, $c_3 = -1$.

Wielomian interpolacyjny w postaci Newtona jest zatem następujący:

$$p_3(x) = 16 - 15(x + 2) + 7(x + 2)(x + 1) - 1(x + 2)(x + 1)(x - 0).$$

Nie trzeba upraszczać tego wielomianu (tj. zmieniać bazy z 1, $x - x_0$, $(x - x_0)(x - x_1)$, $(x - x_0)(x - x_1)(x - x_2)$ na naturalną $1, x, x^2, x^3$).

Uwaga 3.2. *W praktyce nie zaleca się zmiany bazy Newtona na naturalną. Baza Newtona ma lepsze własności numeryczne, a ponadto łatwo jest zmodyfikować algorytm Hornera tak, aby można nim było obliczać wartości wielomianu interpolacyjnego w postaci Newtona.*

3.4 Uogólniony schemat Hornera

Schemat Hornera obliczania wartości wielomianu w postaci Newtona w punkcie $\hat{x} \in \mathbb{R}$. Zapiszmy $p_n(x)$ jako

$$\begin{aligned}
&p_n(x) = \\
&c_0 + (x - x_0)\big(c_1 + (x - x_1)\big(\ldots + (x - x_{n-3})\big(c_{n-2} + (x - x_{n-2})\big(c_{n-1} + (x - x_{n-1})c_n\big)\big)\ldots\big)\big).
\end{aligned}$$

Algorytm (uogólniony schemat Hornera)

$w = c_n$
for $k = n - 1, n - 2, \ldots, 0$
    $w = c_k + (\hat{x} - x_k)\, w$
end
$p_n(\hat{x}) = w$

Koszt: n mnożeń, $2n$ dodawań/odejmowań.

3.5 Błąd interpolacji, zbieżność procesów interpolacyjnych, dobór węzłów

Dokonując interpolacji, czyli przybliżając pewną (najczęściej nieznaną) funkcję, inną funkcją, warto zastanowić się, jaki jest błąd tego przybliżenia.
Podamy teraz podstawowe twierdzenie dotyczące błędu interpolacji wielomianowej.

Twierdzenie 3.2. *Jeśli $f \in C^{n+1}([a,b])$, a wielomian $p_n \in \Pi_n$ jest wielomianem interpolacyjnym dla f opartym na parami różnych węzłach $x_0, \ldots, x_n \in [a,b]$, to dla każdego $x \in [a,b]$ istnieje takie $\xi \in (a,b)$, że*

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\ldots(x-x_n). \qquad \square$$

Najczęściej dokładna wartość ξ nie jest znana. Wiemy tylko tyle, że $\xi \in (a,b)$. Dlatego w praktyce korzysta się z oszacowania błędu.

Wniosek 3.1. *(z Twierdzenia 3.2) Jeśli $f \in C^{n+1}([a,b])$ oraz $p_n \in \Pi_n$ jest wielomianem interpolacyjnym dla f opartym na parami różnych węzłach $x_0, \ldots, x_n \in [a,b]$, to dla każdego $x \in [a,b]$ zachodzi oszacowanie*

$$|f(x) - p_n(x)| \leq \frac{M_{n+1}}{(n+1)!}|(x-x_0)(x-x_1)\ldots(x-x_n)|,$$

gdzie

$$M_{n+1} = \max_{x\in[a,b]} |f^{(n+1)}(x)|. \qquad \square$$

Patrząc na powyższe twierdzenie (oraz wniosek) widzimy, że (przy ustalonym n) błąd interpolacji zależy od funkcji, którą interpolujemy (a konkretniej, od jej n+1-ej pochodnej) oraz od węzłów. To raczej nie są bardzo zaskakujące obserwacje, niemniej możemy zadać sobie pytanie: co zrobić (jak dobrać parametry problemu), aby błąd był jak najmniejszy. W przypadku, gdy funkcja f jest nieznana (np. gdy dokonujemy interpolacji na danych z pomiaru, sporządzonego w skończonej liczbie punktów), nie znamy ani $f^{(n+1)}$, ani może się nie dać dobrze oszacować M_{n+1}. Poza tym, nawet, jeśli znamy $f^{(n+1)}$, to nie mamy na nią wpływu. Możemy mieć natomiast wpływ na dobór węzłów. Można się starać tak dobrać węzły $x_0, \ldots, x_n$, aby zminimalizować wielkość

$$r_n = \max_{x\in[a,b]} |(x-x_0)(x-x_1)\ldots(x-x_n)|. \tag{3.5}$$

Węzły równoodległe (tj. $x_k = a + kh$, gdzie $h = \frac{b-a}{n}$, $k = 0, 1, \ldots, n$) nie są dobrym wyborem. Zauważmy, że wielomian $(x-x_0)(x-x_1)\ldots(x-x_n)$ (zbudowany na węzłach

równoodległych) ma największe wartości blisko brzegów przedziału $[a, b]$, a mniejsze bliżej jego środka (warto zrobić wykres tego wielomianu – w Matlabie lub innym programie – np. dla $[a, b] = [-1, 1]$, $n = 20$). Najlepsze byłyby takie węzły, przy których wartości analogicznego wielomianu są zbliżone w całym przedziale $[a, b]$. Dzieje się tak, gdy węzłami są pierwiastki wielomianów Czebyszewa. Wielomian Czebyszewa n-tego stopnia (dla $[a, b] = [-1, 1]$) zdefiniowany jest wzorem:

$$T_n(x) = \cos(n \arccos x), \quad \text{dla} \quad x \in [-1, 1], \quad n = 0, 1, \ldots. \tag{3.6}$$

Wielomiany Czebyszewa spełniają związek rekurencyjny

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k = 2, 3, \ldots.$$

Wielomian Czebyszewa stopnia $n + 1$ (dla $[a, b] = [-1, 1]$) ma następujące pierwiastki

$$x_i = \cos\Big(\frac{2i + 1}{2n + 2}\pi\Big) \quad i = 0, 1, \ldots, n. \tag{3.7}$$

Ze wzoru trygonometrycznego (3.6) wynika, że $\max_{x\in[-1,1]} |T_n(x)| = 1$, dla każdego $n = 1, 2, \ldots$. Ponadto, prawdziwa jest równość

$$(x - x_0)(x - x_1) \ldots (x - x_n) = 2^{-n}T_{n+1}.$$

Zatem dla węzłów Czebyszewa otrzymujemy następujące oszacowanie błędu interpolacji

$$|f(x) - p_n(x)| \leq \frac{M_{n+1}}{2^n(n + 1)!}.$$

Co więcej, jeśli chodzi o dobór węzłów, jest to najlepsze oszacowanie. W 1857r. P. Czebyszew udowodnił, że dla $[a, b] = [-1, 1]$ wartość r_n (określona wzorem (3.5)) osiąga minimum, gdy węzłami są punkty (3.7). Dlatego teraz te punkty nazywane są węzłami Czebyszewa. Wynik uzyskany przez Czebyszewa można sformułować w następujący sposób.

Twierdzenie 3.3. *Spośród wszystkich wielomianów postaci $p(x) = x^{n+1} + w(x)$, gdzie $w \in \Pi_n$, najmniejszą wartość $\max_{x\in[-1,1]} |p(x)|$ ma wielomian $2^{-n}T_{n+1}$.* □

Uwaga 3.3. *Gdy chcemy użyć węzłów Czebyszewa na dowolnym przedziale $[a, b]$, możemy je liniowo przeskalować. Otrzymujemy:*

$$x_i = \frac{a + b}{2} - \frac{b - a}{2} \cos\Big(\frac{2i + 1}{2n + 2}\pi\Big) \quad i = 0, 1, \ldots, n.$$

3.5.1 Zbieżność procesów interpolacyjnych. Zjawisko Rungego

Rozwiązując zadania interpolacyjne można się zastanawiać, czy dla każdego zadania da się tak dobrać liczbę węzłów oraz rodzaj (równoodległe, Czebyszewa, itp.), aby błąd interpolacji był dowolnie mały. W pewnym stopniu odpowiadają na to następujące trzy twierdzenia.

Twierdzenie 3.4. *(Weierstrassa) Jeśli funkcja f jest ciągła w przedziale $[a, b]$, to dla każdego $\varepsilon > 0$ istnieje taki wielomian p, że $\max_{x\in[a,b]} |f(x) - p(x)| \leq \varepsilon$.*

Można powiedzieć, że twierdzenie to jest optymistyczne. Dzięki niemu wiemy, że dowolną funkcję ciągłą można dowolnie blisko przybliżyć wielomianem. Ciągłość nie musi być ograniczeniem. Większość zjawisk w przyrodzie ma charakter ciągły. Na przykład, temperatura, wilgotność, itp. zmieniają się w sposób ciągły. Jeśli chcemy stworzyć model ruchu jakiegoś obiektu w czasie (np. jadącego samochodu czy rzuconej piłki), to ruch ten odbywa się po krzywej ciągłej.
Z twierdzeniem Weierstrassa jest jednak taki problem, że jest ono egzystencjalne. Mówi nam, że odpowiedni wielomian istnieje, ale nie mówi, jak go określić.
Można tworzyć różne układy węzłów (np. Czebyszewa, równoodległe, jeszcze inne), ale nie ma uniwersalnego układu węzłów, który gwarantowałby zbieżność ciągu wielomianów interpolacyjnych na nich skonstruowanych do każdej funkcji. Mówi o tym następujące twierdzenie.

Twierdzenie 3.5. *(Fabera) Dla dowolnego ciągu układów węzłów $a \leq x_0^{(n)} < x_1^{(n)} < \ldots < x_n^{(n)} \leq b$, gdzie $n \geq 0$, istnieje taka funkcja ciągła w $[a, b]$, że ciąg wielomianów interpolacyjnych zbudowanych na tych węzłach nie jest do niej zbieżny.*

Zakończymy ten wątek trzecim twierdzeniem, znów optymistycznym.

Twierdzenie 3.6. *Jeśli f jest funkcją ciągłą w $[a, b]$, to istnieje taki ciąg układów węzłów $a \leq x_0^{(n)} < x_1^{(n)} < \ldots < x_n^{(n)} \leq b$, gdzie $n \geq 0$, że zbudowane na nich wielomiany interpolacyjne tworzą ciąg zbieżny do f.*

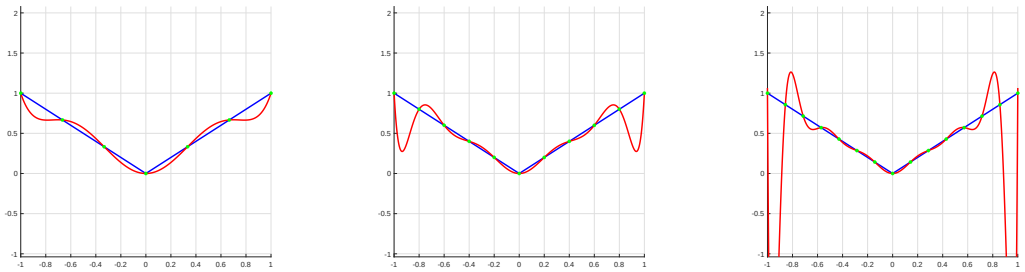
Problem znalezienia takiego ciągu pozostaje jednak nierozwiązany.

Zastanówmy się teraz, jak liczba oraz rozkład węzłów może wpływać na błąd interpolacji. Większa liczba węzłów oznacza, że mamy więcej informacji o funkcji interpolowanej. Poza tym, we wzorze określającym błąd interpolacji (Twierdzenie 3.2) czy oszacowaniu błędu (Wniosek 3.1) w mianowniku mamy $(n+1)!$, a więc wartość szybko rosnącą wraz z liczbą węzłów. To może sugerować, że zwiększając liczbę węzłów dostajemy coraz lepsze przybliżenie funkcji f. Nie zawsze jednak tak się dzieje. Nie zapominajmy, że błąd interpolacji zależy też od wartości wyrażenia $(x - x_0)(x - x_1)\ldots(x - x_n)$ oraz od wartości $f^{(n+1)}$, a te mogą być bardzo duże.
Poza tym, co z funkcjami, które nie są różniczkowalne? O nich Twierdzenie 3.2 nic nie mówi.

Przykład 3.3.

Zobaczmy, jak w praktyce może zachowywać się błąd interpolacji gdy zwiększamy liczbę węzłów.

Niech $f_1(x) = |x|$. Rysunek 1 przedstawia wykresy funkcji f_1 i jej wielomianów interpolacyjnych skonstruowanych na dla 7, 11 i 15 równoodległych węzłach z przedziału $[-1, 1]$.



Rysunek 1: Wykresy wielomianów interpolacyjnych dla funkcji $f_1(x) = |x|$, skonstruowanych na węzłach równoodległych z przedziału $[-1, 1]$. Liczba węzłów: 7 (lewy wykres), 11 (środkowy wykres) i 15 (prawy wykres).

Widać, że w tym przypadku ciąg wielomianów interpolacyjnych rosnących stopni nie jest zbieżny do funkcji interpolowanej. Największe wartości błędu są dla argumentów leżących blisko krańców przedziału interpolacji (porównaj wyjaśnienia znajdujące się bezpośrednio po wzorze (3.5)).

Wykonajmy teraz analogiczny test dla funkcji $f_2(x) = \dfrac{1}{1+15x^2}$. Rysunek 2 przedstawia wykresy funkcji f_2 oraz jej wielomianów interpolacyjnych zbudowanych na 6, 12, 16 i 20 węzłach równoodległych z przedziału $[-1, 1]$.

Zjawisko, które tu obserwujemy jest dość częste przy interpolacji wielomianami wysokich stopni opartymi na węzłach równoodległych. Nosi ono nazwę *zjawiska Rungego*, a funkcja postaci $f(x) = \dfrac{1}{1+ax^2}$, gdzie $a \geq 1$, nazywana jest funkcją Rungego.

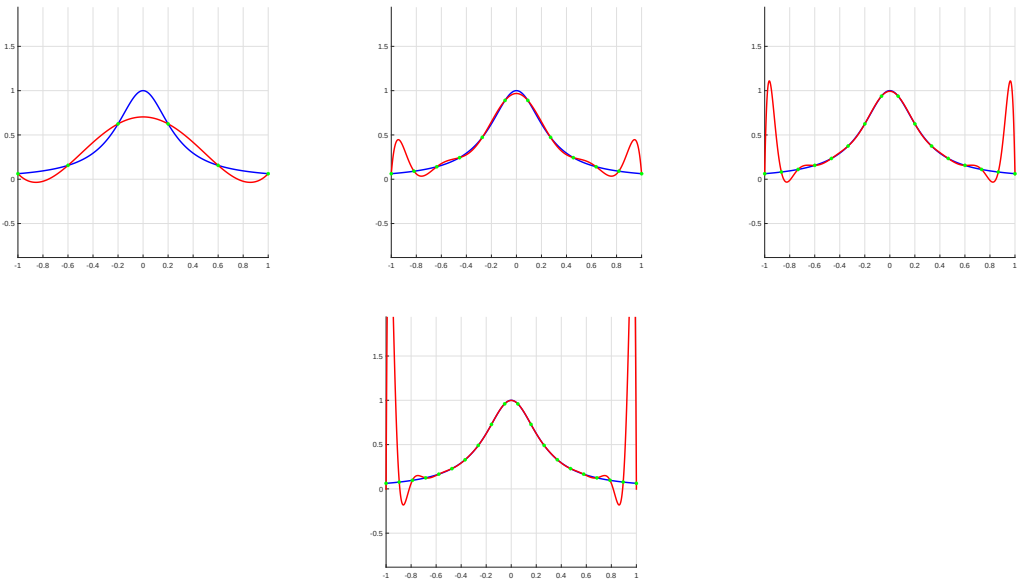
Sprawdźmy teraz, jakie wyniki uzyskamy, gdy zastosujemy węzły Czebyszewa.

Rysunki przedstawiają wykresy funkcji f_1 i f_2 oraz ich wielomianów interpolacyjnych skonstruowanych na węzłach Czebyszewa. Liczba węzłów jest taka sama jak w teście dla węzłów równoodległych.

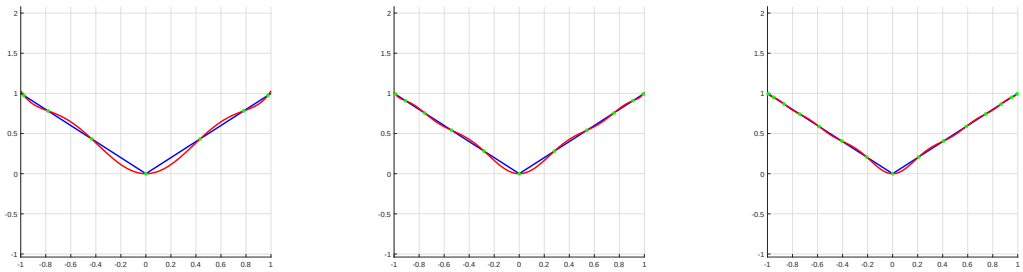
3.6 Interpolacja Hermite'a

Interpolacja Hermite'a to interpolacja z węzłami wielokrotnymi (w węzłach mamy dane nie tylko wartości funkcji interpolowanej, ale też jej pochodnych).

Niech $x_i \in \mathbb{R}$, dla $i = 0, 1, \ldots, m$, takie, że $x_i \neq x_j$ dla $i \neq j$, $i, j = 0, 1, \ldots, m$ będą węzłami interpolacji, a f niech będzie funkcją interpolowaną, dostatecznie wiele razy



Rysunek 2: Wykresy wielomianów interpolacyjnych dla funkcji $f_2(x) = \frac{1}{1+15x^2}$, skonstruowanych na węzłach równoodległych z przedziału $[-1, 1]$. Liczba węzłów: 6, 12 i 16 (wykresy górne) oraz 20 (wykres dolny).

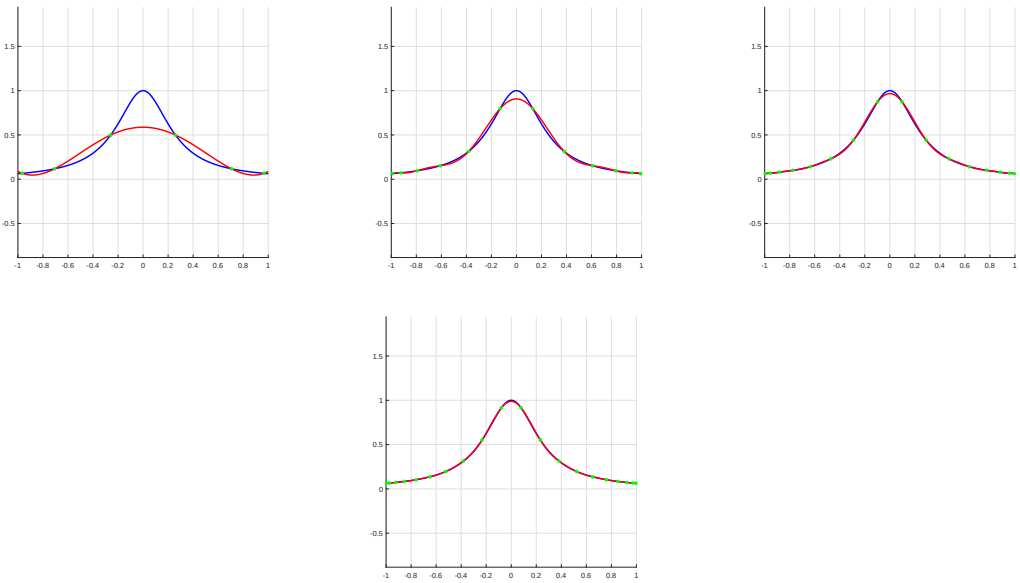


Rysunek 3: Wykresy wielomianów interpolacyjnych dla funkcji $f_1(x) = |x|$, skonstruowanych na węzłach Czebyszewa z przedziału $[-1, 1]$. Liczba węzłów: 7 (lewy wykres), 11 (środkowy wykres) i 15 (prawy wykres).

różniczkowalną (zakładamy, że istnieją wartości pochodnych f, które wykorzystujemy w interpolacji).
Zadanie interpolacji Hermite'a polega na poszukiwaniu wielomianu (interpolacyjnego) p takiego, że

$$p^{(j)}(x_i) = f^{(j)}(x_i), \qquad \text{dla} \quad j = 0, 1, \ldots, k_i - 1, \quad i = 0, 1, \ldots, m, \tag{3.8}$$

gdzie wszystkie wartości $f^{(j)}(x_i)$ są dane. Stałe k_i są dane, definiują krotność węzłów. Często spotykanym przypadkiem interpolacji Hermite'a jest, gdy $k_i = 2$, dla $i = 0, 1, \ldots, m$, czyli wszystkie węzły są dwukrotne (w każdym jest zadana wartość funkcji f i jej pierwszej pochodnej).
Oznaczmy łączną liczbę warunków przez $n + 1$. Zatem $n + 1 = k_0 + k_1 + \ldots + k_m$.



Rysunek 4: Wykresy wielomianów interpolacyjnych dla funkcji $f_2(x) = \frac{1}{1+15x^2}$, skonstruowanych na węzłach Czebyszewa z przedziału $[-1, 1]$. Liczba węzłów: 6, 12 i 16 (wykresy górne) oraz 20 (wykres dolny).

Twierdzenie 3.7. *W klasie* Π_n *istnieje dokładnie jeden wielomian* p *spełniający warunki* (3.8).

Wielomian interpolacyjny Hermite'a jest zatem określony jednoznacznie. Dzięki następnemu twierdzeniu oraz uwadze będziemy mogli go łatwo wyznaczyć.

Twierdzenie 3.8. *Jeśli* $f \in C([a, b])$ *oraz* $x_i \in [a, b]$, $i = 0, 1, \ldots, m$, *przy czym* $x_i \neq x_j$ *dla* $i \neq j$, $i, j = 0, 1, \ldots, m$, *to istnieje takie* $\xi \in (a, b)$, *że*

$$f[x_0, x_1, \ldots, x_m] = \frac{1}{m!} f^{(m)}(\xi).$$

Dowód. Będzie uzupełniony. □

Uwaga 3.4. *Gdy długość przedziału* $[a, b]$ *w Twierdzeniu 3.8 dąży do zera, otrzymujemy w granicy równość*

$$f[x_0, x_0, \ldots, x_0] = \frac{1}{m!} f^{(m)}(x_0).$$

W przypadku interpolacji Hermite'a wielomian interpolacyjny (w postaci Newtona) można budować korzystając ze schematu ilorazów różnicowych, przyjmując

$$f_{ii\ldots i} = f[x_i, x_i, \ldots, x_i] = \frac{1}{k!} f^{(k)}(x_i). \tag{3.9}$$

W powyższym wzorze, w każdym z oznaczeń $f_{ii\ldots i}$ i $f[x_i, x_i, \ldots, x_i]$ znak i pojawia się $k + 1$ razy.

Przykład 3.4. *Dla funkcji $f(x) = -x^4$ znaleźć wielomian interpolacyjny Hermite'a $p_3(x)$ w postaci Newtona taki, że $p_3(-1) = f(-1)$, $p_3(1) = f(1)$, $p_3'(1) = f'(1)$ i $p_3''(1) = f''(1)$.*

Rozwiązanie:
Węzłami interpolacji są punkty: $x_0 = -1$ (węzeł jednokrotny) i $x_1 = 1$ (węzeł trzykrotny). Wielomian interpolacyjny w postaci Newtona skonstruowany na tych węzłach jest następujący:

$$p_3(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)^2. \tag{3.10}$$

Współczynniki tego wielomianu odczytamy z tablicy ilorazów różnicowych.

W przypadku interpolacji Hermite'a tabelę ilorazów różnicowych konstruujemy podobnie jak przy interpolacji z węzłami jednokrotnymi, przy czym w pierwszej kolumnie umieszczamy każdy węzeł tyle razy, ile mamy z nim związanych warunków, przez co liczba elementów w każdej z dwóch pierwszych kolumn będzie równa łącznej liczbie warunków. W drugiej kolumnie umieszczamy wartości funkcji interpolowanej, a w kolejnych kolumnach ilorazy różnicowe (tam, gdzie jest to możliwe, tj. tam, gdzie próba policzenia ilorazu różnicowego nie prowadzi do dzielenia 0 przez 0) lub wartości wynikające ze wzoru (3.9). W tym przykładzie schemat ten będzie wyglądał następująco:

$$\begin{array}{llll}
x_0 & f_0 & & \\
x_1 & f_1 \to f_{01} & & \\
x_1 & f_1 \to \boldsymbol{f_{11}} \to f_{011} & & \\
x_1 & f_1 \to \boldsymbol{f_{11}} \to \boldsymbol{f_{111}} \to f_{0111} &
\end{array}$$

Wartości pogrubionych elementów obliczymy korzystając ze wzoru (3.9).
Po podstawieniu wartości liczbowych dostajemy:

$$\begin{array}{llll}
-1 & -1 & & \\
1 & -1 \to \frac{-1-(-1)}{1-(-1)} = 0 & & \\
1 & -1 \to f_{11} = f'(x_1) = -4 & \to \frac{-4-0}{1-(-1)} = -2 & \\
1 & -1 \to f_{11} = f'(x_1) = -4 & \to f_{111} = \frac{1}{2}f''(x_1) = -6 & \to \frac{-6-(-2)}{1-(-1)} = -2
\end{array}$$

Współczynniki c_i $(i = 0, 1, 2, 3)$ odczytujemy z tabeli: $c_0 = -1$, $c_1 = 0$, $c_2 = -2$, $c_3 = -2$.

Wielomian interpolacyjny Hermite'a (w postaci Newtona) jest zatem następujący:

$$p_3(x) = -1 - 2(x + 1)(x - 1) - 2(x + 1)(x - 1)^2.$$

Można go także zapisać przyjmując odwrotną kolejność węzłów:

$$p_3(x) = -1 - 4(x - 1) - 6(x - 1)^2 - 2(x - 1)^3.$$

Dzięki Twierdzeniu 3.7 wiemy, że jest to ten sam wielomian, ale jeśli ktoś ma ochotę, to zawsze można to sprawdzić (sprowadzając do tej samej bazy).

Dodatkowa lektura: podrozdział 6.0, 6.1, 6.2 i 6.3 z pozycji [1].

4 Całkowanie numeryczne

Problemem, którym się teraz zajmiemy, będzie obliczanie (najczęściej w sposób przybliżony) wartości całki oznaczonej

$$I(f) = \int_a^b f(x)dx, \tag{4.1}$$

gdzie $f : [a, b] \to \mathbb{R}$ jest funkcją całkowalną.

Dlaczego potrzebne są metody numerycznego całkowania?

- Funkcja pierwotna nie zawsze istnieje, zatem nie zawsze jesteśmy w stanie obliczyć całkę analitycznie. Z drugiej strony, całki, które pojawiają się w praktycznych zastosowaniach, często wyrażają jakąś wielkość fizyczną (np. masa, rozkład ładunku elektrycznego) albo wynikają z równań różniczkowych które modelują jakieś zjawiska fizyczne. Zatem, nawet jeśli funkcja pierwotna nie istnieje, wielkość, którą całka opisuje, jak najbardziej może istnieć i możemy próbować ją obliczyć w sposób przybliżony.
- Funkcja f może być dana w postaci dyskretnej, jako zbiór punktów i wartości w tych punktach (przykładem takiej funkcji są dane pomiarowe) zatem nie da się jej scałkować analitycznie.
- Są także takie przypadki, gdzie funkcja pierwotna istnieje, ale ma taką postać, że numeryczne obliczenie całki $I(f)$ jest szybsze i dokładniejsze niż analityczne.

Z zagadnieniem przybliżonego obliczania całek wiąże się pojęcie kwadratury.

4.1 Kwadratury

Kwadratury (czyli wzory przybliżonego całkowania) pozwalają obliczyć przybliżoną wartość całki $I(f)$. Będziemy rozważać kwadratury postaci

$$S(f) = \sum_{k=0}^{n} A_k f(x_k), \tag{4.2}$$

gdzie

- punkty $x_k \in [a, b]$, dla $k = 0, \ldots, n$, oraz $x_k \neq x_j$ dla $k \neq j$ $(k, j = 0, \ldots, n)$ nazywane są węzłami kwadratury,
- A_k, dla $k = 0, \ldots, n$, są współczynnikami kwadratury; są to stałe niezależne od f.

Zauważmy, że aby obliczyć przybliżoną wartość całki za pomocą wzoru (4.2), musimy znać wartości funkcji f *tylko w węzłach* tej kwadratury.

Wzór (4.2) to ogólna postać kwadratury. Aby uzyskać konkretną metodę (kwadraturę), trzeba określić liczbę węzłów (czyli związana z nią wartość n) oraz wartości węzłów i współczynników.
Powstaje pytanie: jak (przy ustalonym n) dobrać współczynniki A_k oraz węzły x_k, dla $k = 0, \ldots, n$, aby wzór (4.2) jak najlepiej przybliżał wartość całki (4.1)? Innymi słowy, jak dobrać parametry kwadratury, aby jej błąd, czyli

$$E(f) = I(f) - S(f),$$

był mały (tj. jak najbliższy zeru)?

Kwadratura $S(f)$ przy ustalonym n ma $2n + 2$ parametry, tj. $n + 1$ węzłów i $n + 1$ współczynników. Często wyznacza się je zakładając, że $I(f) = S(f)$ dla wielomianów określonego stopnia.
Z tym wiąże się też pojęcie rzędu kwadratury.

Definicja 4.1. *Kwadratura $S(f)$ jest rzędu r, jeśli*

1) *jest dokładna (czyli $E(f) = 0$) dla wszystkich wielomianów stopnia mniejszego niż r, oraz*

2) *istnieje wielomian stopnia r, dla którego $S(f)$ nie jest dokładna.*

4.2 Kwadratury interpolacyjne

Bardzo ważną grupę kwadratur (i jedyną omawianą na tym wykładzie) stanowią kwadratury interpolacyjne.
Idea ich konstrukcji jest taka: funkcję podcałkową f zastępujemy funkcją p, która jest funkcją interpolującą dla f i jednocześnie jest łatwa do całkowania; następnie przyjmujemy, że $S(f) = I(p)$, czyli przybliżeniem całki $I(f)$ jest całka z funkcji interpolującej. Najczęściej funkcją interpolującą jest po prostu wielomian interpolacyjny Lagrange'a (czyli oparty na węzłach jednokrotnych). Węzłami kwadratury są węzły interpolacji.

Skonstruujemy teraz pewne kwadratury interpolacyjne. Zastosujemy wielomian interpolacyjny Lagrange'a, zapisany w postaci Lagrange'a, czyli:

$$p_n(x) = \sum_{k=0}^{n} f(x_k) l_k(x), \quad \text{gdzie} \quad l_k(x) = \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}.$$

Po zastąpieniu f wielomianem p_n w całce (4.1) otrzymujemy:

$$I(f) = \int_a^b f(x)dx \approx \int_a^b p_n(x)dx$$
$$= \int_a^b \sum_{k=0}^{n} f(x_k)l_k(x)dx = \sum_{k=0}^{n} f(x_k) \int_a^b l_k(x)dx.$$

Skorzystaliśmy z faktu, że całka sumy funkcji jest równa sumie całek oraz z tego, że $f(x_k) \in \mathbb{R}$, zatem można tę wartość wyłączyć przed całkę.
Porównując teraz ostatnie wyrażenie w powyższym wzorze z ogólnym wzorem kwadratury (4.2), widzimy, że jeśli przyjmiemy

$$A_k = \int_a^b l_k(x)dx \quad \text{dla} \quad k = 0, \ldots, n, \tag{4.3}$$

to otrzymaliśmy kwadraturę. Węzłami x_k są węzły interpolacji, a współczynniki A_k możemy obliczyć z powyższego wzoru, całkując wielomiany l_k.

Pojawia się teraz jeszcze pytanie: jak wybrać węzły? Można to zrobić w dowolny sposób, pamiętając tylko o tym, że węzły mają być (parami) różne oraz, że mają należeć do przedziału $[a, b]$.
W praktyce, jeśli z góry narzuca się wartości węzłów, to są to najczęściej węzły równoodległe (oparte na nich kwadratury noszą nazwę kwadratur Newtona-Cotesa). Można też nie narzucać konkretnych wartości, tylko starać się je tak dobrać (obliczyć), aby zmaksymalizować rząd kwadratury (co podniesie jej dokładność). Takie kwadratury to kwadratury Gaussa.

4.3 Kwadratury Newtona-Cotesa (proste)

W przypadku, gdy węzły są równoodległe oraz $x_0 = a$ i $x_n = b$ (czyli $x_k = a + hk$, gdzie $h = (b - a)/n$), wzór

$$S(f) = \sum_{k=0}^{n} f(x_k)A_k$$

przy A_k zdefiniowanych wzorem (4.3), nosi nazwę kwadratury Newtona-Cotesa.

Rozpatrzmy teraz kilka przypadków szczególnych kwadratur tego typu. Będą to kwadratury *proste*, nazywane tak w przeciwieństwie do kwadratur złożonych, które będą opisane w podrozdziale 4.4.

4.3.1 Kwadratura (wzór) trapezów (prosta)

Dla $n = 1$ otrzymujemy kwadraturę dwupunktową (czyli opartą na dwóch węzłach), której węzłami są końce przedziału całkowania: a i b. Nosi ona nazwę kwadratury (wzoru) trapezów i ma postać:

$$S(f) = \frac{b-a}{2}\big(f(a) + f(b)\big).$$

Funkcja f jest przybliżana wielomianem stopnia 1. Jeśli $f(a) > 0$ oraz $f(b) > 0$, to $S(f)$ jest polem trapezu, utworzonego przez odcinek $[a, b]$, wykres wielomianu interpolacyjnego przybliżającego f na odcinku $[a, b]$ oraz odcinków łączących punkty $(a, 0)$ i $(a, f(a))$ oraz $(b, 0)$ i $(b, f(b))$.

Rysunek 5: Graficzna interpretacja metody trapezów: niebieska linia to wykres funkcji podcałkowej, czerwona linia to wykres przybliżającego ją wielomianu interpolacyjnego opartego na węzłach a i b.

Uwaga 4.1. *Jeśli funkcja podcałkowa nie ma dodatnich wartości w punktach a lub b, wartość kwadratury nie może być interpretowana jako pole trapezu (wartość kwadratury nie musi być dodatnia, a odpowiednia figura nie musi być trapezem).*

Błąd prostej kwadratury trapezów wynosi:

$$E(f) = -h^3 \frac{1}{12} f''(\xi_1),$$

gdzie $h = b - a$, a ξ_1 jest pewnym punktem leżącym w przedziale (a, b).

Zauważmy, że błąd $E(f)$ jest zerowy, gdy $f''(x) = 0$ dla wszystkich $x \in (a, b)$, co ma miejsce w przypadku wielomianów stopnia 1 lub 0. Natomiast, gdy f jest wielomianem stopnia 2, to $f''(x) \neq 0$.
Dlatego rząd kwadratury trapezów jest równy 2.
Dokładna wartość błędu może być trudna do określenia, ponieważ nie znamy wartości ξ_1. Szacując f'' z góry i z dołu można otrzymać oszacowania błędu.

4.3.2 Kwadratura (wzór) Simpsona (prosta)

Gdy $n = 2$, mamy 3 węzły: a, b i $\frac{a+b}{2}$. Kwadratura na nich skonstruowana to kwadratura (wzór) Simpsona, zwana również wzorem parabol.

Funkcja f jest przybliżana wielomianem stopnia 2. W tym przypadku kwadratura ma postać

$$S(f) = \frac{b-a}{6}\left(f(a) + 4f\Big(\frac{a+b}{2}\Big) + f(b)\right).$$

Błąd prostej kwadratury Simpsona wynosi:

$$E(f) = -h^5 \frac{1}{90} f^{(4)}(\xi_2),$$

gdzie $h = \frac{b-a}{2}$, a ξ_2 jest pewnym punktem leżącym w przedziale (a, b).
W tym przypadku błąd $E(f)$ jest równy zero, gdy $f^{(4)}(x) = 0$, dla wszystkich $x \in (a, b)$, co ma miejsce w przypadku wielomianów stopnia nie przekraczającego 3. Dlatego rząd kwadratury Simpsona wynosi 4.
Zauważmy, że wielomian interpolacyjny skonstruowany na trzech węzłach jest stopnia co najwyżej 2, zatem błąd interpolacji dla wielomianu stopnia 3 jest niezerowy. Kwadratura Simpsona, natomiast, jest dokładna dla wszystkich wielomianów stopnia 3 (por. Twierdzenie 4.1 oraz Uwaga 4.3).

4.3.3 Kwadratura (wzór) Newtona 3/8 (prosta)

Dla $n = 3$ kwadratura jest oparta na 4 węzłach: a, b, $\frac{2a+b}{3}$ i $\frac{a+2b}{3}$.
Funkcja f jest przybliżana wielomianem stopnia 3. Kwadratura Newtona „$\frac{3}{8}$" („trzech ósmych") ma postać:

$$S(f) = \frac{b-a}{8}\left(f(a) + 3f\Big(\frac{2a+b}{3}\Big) + 3f\Big(\frac{a+2b}{3}\Big) + f(b)\right),$$

gdzie $h = \frac{b-a}{3}$.
Błąd tej kwadratury wynosi

$$E(f) = -h^5 \frac{3}{80} f^{(4)}(\xi_3),$$

dla pewnego $\xi_3 \in (a, b)$.
Rząd kwadratury Newtona $\frac{3}{8}$ wynosi 4 (jest TAKI SAM jak kwadratury Simpsona) – porównaj Twierdzenie 4.1 oraz Uwaga 4.3.

4.3.4 Kwadratura prostokątów (prosta)

Dla $n = 0$ warunek $x_0 = a$, $x_n = b$ nie może być spełniony, zatem kwadratura jednopunktowa (czyli oparta na jednym weźle) często (tj. w wielu źródłach) nie jest zaliczana do kwadratur Newtona-Cotesa.
Kwadratura oparta na jednym węźle $x_0 \in [a, b]$ nosi nazwę kwadratury (wzoru) prostokątów. Jest następująca:

$$S(f) = (b-a)f(x_0).$$

Funkcja podcałkowa f jest w tym przypadku przybliżana wielomianem stopnia 0 opartym na węźle x_0.
Węzłem x_0 może być dowolny punkt należący do przedziału $[a, b]$, ale najczęściej stosuje się $x_0 = \frac{a+b}{2}$, czyli środek przedziału. Można udowodnić, że w tym przypadku błąd kwadratury prostokątów wyraża się wzorem

$$E(f) = h^3 \frac{1}{24} f''(\xi_4),$$

gdzie $h = b - a$, a ξ_4 jest pewnym punktem leżącym w przedziale (a, b).
Rząd kwadratury prostokątów (z węzłem środkowym) wynosi więc 2 (*jest taki sam, jak w przypadku kwadratury trapezów*).

Uwaga 4.2. *Jeśli węzłem nie jest środek przedziału, to rząd kwadratury prostokątów jest równy 1.*

Ogólnie, błąd kwadratur Newtona-Cotesa określony jest przez następujące twierdzenie.

Twierdzenie 4.1. *Błąd kwadratury Newtona-Cotesa wyraża się wzorem*

$$E(f) = I(f) - S(f) = \int_a^b (f(x) - p_n(x))dx =$$

$$= \begin{cases} h^{n+3}\kappa_n f^{(n+2)}(\xi), & \textit{dla parzystych } n, \\ h^{n+2}\hat{\kappa}_n f^{(n+1)}(\xi), & \textit{dla nieparzystych } n, \end{cases}$$

dla pewnego $\xi \in (a, b)$*, gdzie*

$$\kappa_n = \frac{1}{(n+2)!} \int_0^n t^2(t-1)\dots(t-n)dt,$$

$$\hat{\kappa}_n = \kappa_{n-1} + \frac{1}{(n+1)!} \int_{n-1}^n t(t-1)\dots(t-n)dt.$$

Uwaga 4.3. *Będzie na wykładzie.*

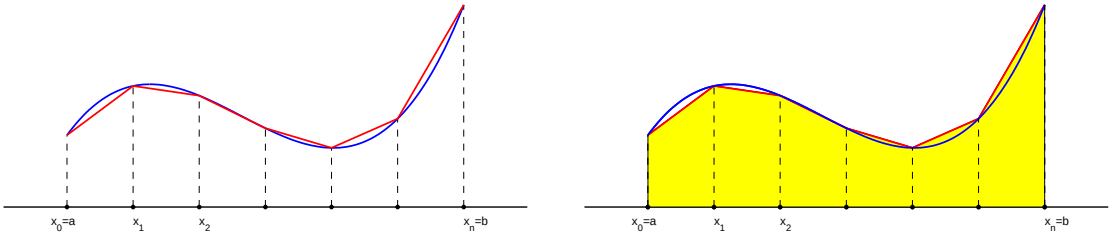
4.4 Kwadratury złożone

Uwaga 4.4. *Gdy $n \geq 7$ (z wyjątkiem 9) niektóre współczynniki kwadratur Newtona-Cotesa są ujemne, co może prowadzić do zwiększenia wpływu błędów zaokrągleń na wynik.*

Z tego względu w celu zwiększenia dokładności nie stosuje się kwadratur opartych na interpolacji wielomianem wysokiego stopnia, tylko kwadratury złożone.
Idea kwadratur złożonych: Przedział $[a, b]$ dzielimy na N podprzedziałów równej długości, na każdym stosujemy kwadraturę prostą i sumujemy uzyskane wartości.

4.4.1 Złożona kwadratura trapezów

Przedział $[a, b]$ dzielimy na podprzedziały $[x_{k-1}, x_k]$ $(k = 1, \ldots, N)$, o długości $H = \frac{b-a}{N}$, przy czym $x_k = a+kH$ dla $k = 0, \ldots, N$. Na każdym podprzedziale stosujemy kwadraturę prostą.



Rysunek 6: Graficzna interpretacja złożonej metody trapezów: niebieska linia to wykres funkcji podcałkowej, czerwona linia to wykres przybliżających ją wielomianów interpolacyjnych stopnia co najwyżej 1 opartych na węzłach x_{k-1} i x_k, gdzie $k = 1, 2, \ldots, N$.

Złożony wzór trapezów uzyskamy ze wzoru prostego, stosując go dla każdego z podprzedziałów $[x_{k-1}, x_k]$, gdzie $k = 1, 2, \ldots, N$ i sumując uzyskane przybliżenia

$$S(f) = \sum_{k=1}^{N} \frac{x_k - x_{k-1}}{2}\big(f(x_k) + f(x_{k-1})\big).$$

Po przekształceniu:

$$S(f) = \frac{H}{2}\left(f(a) + f(b) + 2\sum_{k=1}^{N-1} f(a + kH)\right).$$

W implementacji lepiej jest stosować ten drugi wzór, ponieważ unikamy dwukrotnego obliczania wartości funkcji w tym samym punkcie. W pewnych językach (np. w Matlabie) można zaimplementować kwadratury tego typu nie obliczając sumy (w sposób jawny). Jak?

Błąd złożonej kwadratury trapezów jest równy

$$E(f) = -\frac{1}{12}H^2(b-a)f''(\mu_1),$$

dla pewnego $\mu_1 \in (a, b)$.

4.4.2 Złożona kwadratura Simpsona

Złożony wzór Simpsona ma postać:

$$S(f) = \sum_{k=1}^{N} \frac{H}{6}\left(f(x_{k-1}) + 4f\big(x_{k-1} + \frac{H}{2}\big) + f(x_k)\right).$$

Po przekształceniu:

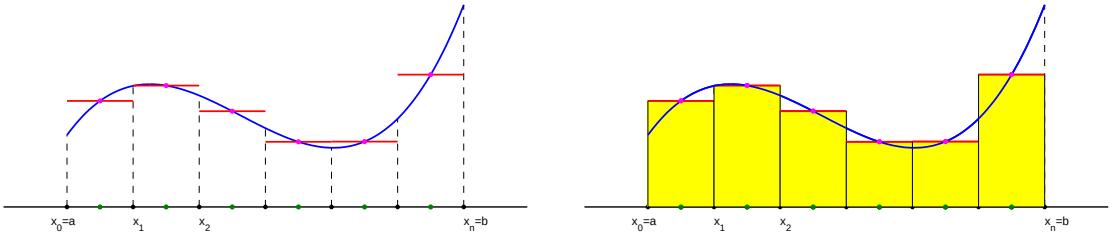
$$S(f) = \frac{H}{6}\Big(f(a) + f(b) + 2\sum_{k=1}^{N-1} f(a+kH) + 4\sum_{k=0}^{N-1} f\big(a+kH+\frac{H}{2}\big)\Big).$$

Błąd złożonej kwadratury Simpsona jest równy

$$E(f) = -\frac{1}{180 \cdot 2^4} H^4 (b-a) f^{(4)}(\mu_2),$$

dla pewnego $\mu_2 \in (a, b)$.

4.4.3 Złożona kwadratura prostokątów



Rysunek 7: Graficzna interpretacja złożonej metody prostokątów (z węzłem środkowym): niebieska linia to wykres funkcji podcałkowej, czerwona linia to wykres przybliżających ją wielomianów interpolacyjnych stopnia 0 opartych na węzłach $x_{k-1} + \frac{H}{2}$, gdzie $k = 1, 2, \ldots, N$.

Złożony wzór prostokątów ma postać:

$$S(f) = \sum_{k=1}^{N} (x_k - x_{k-1}) f\big(\frac{x_k + x_{k-1}}{2}\big).$$

Po przekształceniu:

$$S(f) = H \sum_{k=1}^{N} f\big(\frac{x_k + x_{k-1}}{2}\big).$$

Błąd złożonej kwadratury prostokątów (z węzłem środkowym) jest równy

$$E(f) = \frac{1}{24} H^2 (b-a) f''(\mu_3),$$

dla pewnego $\mu_3 \in (a, b)$.

Uwaga 4.5. *Oszacowanie błędu kwadratury prostokątów z węzłem środkowym jest LEPSZE niż kwadratury trapezów.*

4.4.4 Złożona kwadratura Newtona

Do wyprowadzenia samodzielnie.

4.5 Zmiana przedziału całkowania

W literaturze często można znaleźć kwadratury określone na przedziale $[-1, 1]$ lub $[0, 1]$. Co zrobić, aby wykorzystać taką kwadraturę na dowolnym przedziale $[a, b]$? W przypadku kwadratur Newtona-Cotesa problem ten raczej nie występuje (zwykle są one od razu definiowane na dowolnym przedziale, tak, jak w poprzednim podrozdziale), ale dotyczy innych często używanych kwadratur, na przykład kwadratur Gaussa.
Aby sobie w takiej sytuacji poradzić, można dokonać zamiany zmiennych w całce. Zauważmy, że dla

$$s(x) = \frac{(b-a)x + ad - bc}{d - c}$$

mamy

$$\int_a^b f(x)\,dx \;=\; \frac{b-a}{d-c}\int_c^d f\big(s(x)\big)\,dx.$$

Oznacza to, że mając kwadraturę,

$$\sum_{k=0}^{n} A_k f(x_k) \approx \int_c^d f(x)\,dx,$$

przybliżającą całkę na przedziale $[c, d]$, możemy łatwo zmodyfikować tę kwadraturę, tak, aby przybliżała całkę na innym przedziale:

$$\int_a^b f(x)\,dx \;\approx\; \frac{b-a}{d-c}\sum_{k=0}^{n} A_k\, f\Big(\frac{(b-a)x_k + ad - bc}{d-c}\Big).$$

W szczególności, gdy $[c, d] = [\mathbf{0}, \mathbf{1}]$, to

$$\int_a^b f(x)\,dx \;\approx\; (b-a)\sum_{k=0}^{n} A_k\, f\big((b-a)x_k + a)\big),$$

a gdy $[c, d] = [-\mathbf{1}, \mathbf{1}]$, to

$$\int_a^b f(x)\,dx \;\approx\; \frac{b-a}{2}\sum_{k=0}^{n} A_k\, f\Big(\frac{b-a}{2}x_k + \frac{a+b}{2}\Big).$$

4.6 Metoda Romberga

Niech T_N oznacza przybliżoną wartość całki (4.1) obliczoną za pomocą złożonej kwadratury trapezów z podziałem przedziału całkowania na N podprzedziałów, a E_{T_N} niech oznacza błąd tego przybliżenia.
Błąd ten można wyrazić jako:

$$E_{T_N}(f) = I(f) - T_N(f) = \sum_{k=1}^{\infty} \frac{B_{2k}}{(2k)!}\, H^{2k}\big(f^{(2k-1)}(a) - f^{(2k-1)}(b)\big), \tag{4.4}$$

gdzie B_k $(k = 2, 4, \dots)$ są tak zwanymi *liczbami Bernoulliego*. Uzasadnienie powyższego wzoru wykracza poza zakres tego przedmiotu. Na potrzeby naszego wykładu wystarczy

wiedzieć, że B_k są pewnymi stałymi. Istotne dla nas w tym momencie w tym wzorze jest to, że zawiera wyłącznie parzyste potęgi H. Na tym opiera się metoda Romberga. Jej istotą jest zastosowanie tzw. ekstrapolacji Richardsona.

Zdefiniujmy

$$T_{i,0}(f) = T_{2^i}(f)\,, \qquad i = \mathbf{0}, 1, \ldots . \tag{4.5}$$

Na podstawie (4.4), możemy zapisać

$$I(f) - T_{i,0}(f) = \sum_{k=1}^{\infty} c_{2k} H^{2k} = \; c_2 H^2 + c_4 H^4 + c_6 H^6 + \ldots\,, \tag{4.6}$$

gdzie $H = 2^{-i}(b-a)$, a $c_2, c_4, \ldots$ są pewnymi stałymi. Zauważmy, że jeśli $i > 0$, to

$$I(f) - T_{i-1,0}(f) = \sum_{k=1}^{\infty} c_{2k} (2H)^{2k} = \; 4c_2 H^2 + 16 c_4 H^4 + 64 c_6 H^6 + \ldots\,. \tag{4.7}$$

Zdefiniujmy

$$T_{i,1}(f) = \frac{4T_{i,0}(f) - T_{i-1,0}(f)}{3}\,, \qquad i = \mathbf{1}, 2, \ldots . \tag{4.8}$$

Z zależności (4.6), (4.7) i (4.8) otrzymujemy

$$I(f) - T_{i,1}(f) \;=\; d_4 H^4 + d_6 H^6 + \ldots$$

gdzie $d_4, d_6, \ldots$ są pewnymi stałymi. Ogólnie, jeśli przyjmiemy

$$T_{i,k}(f) = \frac{4^k T_{i,k-1}(f) - T_{i-1,k-1}(f)}{4^k - 1}\,, \quad k > 0, \quad i = k, k+1, \ldots\,,$$

i założymy, że funkcja f posiada w $[a, b]$ pochodne dostatecznie wysokiego rzędu, otrzymamy tablicę (zwaną *tablicą Romberga*) zawierającą przybliżone wartości całki (4.1):

$$\begin{array}{llll} T_{0,0} & & & \\ T_{1,0} & T_{1,1} & & \\ T_{2,0} & T_{2,1} & T_{2,2} & \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

Przybliżenia te mają następujące własności:

- $T_{i,0}(f) = \frac{1}{2} T_{i-1,0}(f) + \frac{1}{2} P_{2^{i-1}}(f)$ dla $i > 0$, gdzie $P_N(f)$ oznacza złożoną kwadraturę prostokątów (z węzłem środkowym) przy podziale przedziału całkowania na N podprzedziałów (zależność ta przydaje się przy tworzeniu pierwszej kolumny tablicy Romberga),
- $T_{i,1}(f) = S_{2^{i-1}}(f)$ dla $i > 0$, gdzie $S_N(f)$ oznacza złożoną kwadraturę Simpsona przy podziale przedziału całkowania na N podprzedziałów (to tylko taka ciekawostka),
- $\big|I(f) - T_{i,k}(f)\big| = \mathcal{O}(h^{2k+2}) = \mathcal{O}(4^{-i(k+1)})$ **przy założeniu, że** $f \in C^{2k+2}[a, b]$,

- $I(p) = T_{i,k}(p)$ dla każdego wielomianu $p \in \Pi_{2k+1}$.

Dodatkowa lektura: podrozdziały 7.2 i 7.5 z pozycji [1].

5 Metody rozwiązywania układów równań liniowych

Zajmiemy się teraz metodami rozwiązywania układów równań liniowych postaci

$$Ax = b,$$

gdzie $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$. Załóżmy, że macierz tego układu jest nieosobliwa, tj. $\det(A) \neq 0$. Zatem rozwiązanie tego układu istnieje i jest jednoznaczne.

Metody numeryczne rozwiązywania układów równań liniowych można podzielić na dwie główne grupy:

- metody bezpośrednie (liczba operacji arytmetycznych potrzebnych do znalezienia rozwiązania jest skończona, znana i niezmienna dla danej metody – zależy od wymiaru macierzy układu, czyli od rozmiaru zadania, a nie od elementów tej macierzy).
- metody iteracyjne (w metodach tych tworzony jest ciąg wektorów, które są kolejnymi przybliżeniami wektora x; jeśli metoda jest zbieżna, ciąg ten dąży do x; liczba iteracji, czyli liczba elementów tego ciągu, które trzeba wyznaczyć, konieczna go uzyskania przybliżenia z danym błędem, zależy od macierzy układu równań).

Zaczniemy od omówienia podstawowych metod bezpośrednich rozwiązywania układów równań liniowych.

5.1 Układ łatwy do rozwiązania

Jednym z prostszych do rozwiązania układów równań liniowych jest układ z macierzą trójkątną (górną lub dolną). Przyjmijmy, że A jest macierzą trójkątną górną, tj.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix}. \tag{5.1}$$

W przypadku macierzy trójkątnych dolnych rozważania są analogiczne.
Układ równań z macierzą (5.1) można zapisać tak:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ &\ldots \\ a_{nn}x_n &= b_n. \end{aligned}$$

Jak rozwiązać taki układ, wszyscy się na pewno domyślają. Z ostatniego równania obliczamy x_n, wstawiamy do przedostatniego, obliczamy x_{n-1}, następnie obie wartości (x_n i x_{n-1}) wstawiamy do poprzedniego równania, itd.

Algorytm:

$$x_n := \frac{b_n}{a_{nn}}$$
for $k = n-1, n-2, \ldots, 1$
$$x_k := \frac{b_k - \sum_{j=k+1}^{n} a_{kj} x_j}{a_{kk}}$$
end

Założyliśmy na początku, że $\det(A) \neq 0$. W przypadku macierzy trójkątnej, $\det(A) = a_{11} \cdot \ldots \cdot a_{nn}$ (co można pokazać korzystając z rozwinięcia Laplace'a), zatem powyższy algorytm jest wykonalny (w tym sensie, że nie nastąpi dzielenie przez zero).
Koszt tego algorytmu (liczony jako liczba mnożeń i dzieleń, dla uproszczenia):

$$T(n) = \sum_{k=1}^{n} (n-k+1) = 1 + 2 + \ldots + n = \frac{1+n}{2} n \approx \frac{n^2}{2}.$$

Ostatnie przybliżenie jest prawdziwe dla dużych wartości n.

5.2 Metoda eliminacji Gaussa i jej warianty

Załóżmy teraz, że macierz A nie ma określonej struktury (typu, trójkątna, diagonalna, itd.). W ogólności może to być macierz *pełna* (tj. nie ma w niej elementów zerowych).

Ideą metody eliminacji Gaussa jest sprowadzenie wyjściowego układu równań do układu równoważnego (tj. takiego, który ma takie samo rozwiązanie jak układ wyjściowy), ale o prostszej strukturze. Sprowadzenie to odbywa się za pomocą operacji elementarnych.
Istnieje wiele wariantów eliminacji Gaussa. Czasem eliminacją Gaussa określa się każde uproszczenie macierzy za pomocą operacji elementarnych prowadzące do powstania w niej elementów zerowych.

Tutaj skupimy się na obliczeniowej i implementacyjnej stronie problemu, przedstawimy kilka algorytmów i zastanowimy się nad ich zaletami, wadami i własnościami numerycznymi.

Metody, które omówimy to:

- metoda eliminacji Gaussa (w skrócie GE, z ang. Gaussian Elimination),
- metoda eliminacji Gaussa z częściowym wyborem elementów głównych (w skrócie GEPP, z ang. Gaussian Elimination with Partial Pivoting),
- metoda eliminacji Gaussa z pełnym wyborem elementów głównych (w skrócie GECP, z ang. Gaussian Elimination with Complete Pivoting).

Każdy z tych trzech wariantów zawiera dwa etapy:

- etap eliminacji – sprowadzenie macierzy układu do postaci trójkątnej (najczęściej górnej),
- rozwiązanie układu z macierzą trójkątną.

5.2.1 Metoda eliminacji Gaussa (GE)

W tym, najbardziej podstawowym, wariancie metody Gaussa, w etapie eliminacji wykorzystujemy tylko operacje typu $r_i := r_i - lr_j$, gdzie $i \neq j$, $l \in \mathbb{R}$, tj. odjęcie od i-tego wiersza, wiersza j-tego przemnożonego przez pewną stałą.
Eliminację wykonujemy zaczynając od pierwszej kolumny – zerujemy w niej wszystkie elementy z wyjątkiem pierwszego, odejmując od wierszy 2-go, 3-go, ..., n-go odpowiednie wielokrotności pierwszego wiersza. Następnie, w drugiej kolumnie zerujemy wszystkie elementy z wyjątkiem dwóch pierwszych, odejmując od wierszy 3-go, 4-go, ..., n-go odpowiednie wielokrotności drugiego wiersza, itd.
Wszystkie operacje wykonujemy na macierzy rozszerzonej układu.

Zapiszmy układ wyjściowy $Ax = b$ jako:

$$\begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1n}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & \dots & a_{2n}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & \dots & a_{nn}^{(0)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(0)} \\ b_2^{(0)} \\ \vdots \\ b_n^{(0)} \end{pmatrix}.$$

Macierz rozszerzona tego układu (przy oznaczeniu $A^{(0)} = A$ i $b^{(0)} = b$) to:

$$(A^{(0)}|b^{(0)}) = \left(\begin{array}{cccc|c} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1n}^{(0)} & b_1^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & \dots & a_{2n}^{(0)} & b_2^{(0)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & \dots & a_{nn}^{(0)} & b_n^{(0)} \end{array}\right).$$

Górny indeks przy elementach macierzy oznacza ile kroków eliminacji już wykonaliśmy (przez jeden krok rozumiemy wyzerowanie elementów leżących w jednej kolumnie, pod główną przekątną).
W pierwszym kroku eliminujemy elementy $a_{21}^{(0)}, \dots, a_{n1}^{(0)}$ za pomocą operacji

$$r_i := r_i - l_{i1}r_1, \quad \text{gdzie} \quad l_{i1} = \frac{a_{i1}^{(0)}}{a_{11}^{(0)}}, \quad i = 2, 3, \dots, n.$$

Zwróćmy uwagę, że od kolejnych wierszy odejmujemy wielokrotność pierwszego wiersza. Element $a_{11}^{(0)}$, przez który dzielimy, nazywamy *elementem głównym*. Jeśli $a_{11}^{(0)} = 0$, metoda nie jest wykonalna. Tą kwestią zajmiemy się później. Na razie przyjmijmy, że udało się zredukować pierwszą kolumnę.

Po pierwszym kroku (czyli po zredukowaniu pierwszej kolumny) otrzymujemy:

$$(A^{(1)}|b^{(1)}) = \left(\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & a_{32}^{(1)} & \dots & a_{3n}^{(1)} & b_3^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right).$$

W drugim kroku eliminujemy $a_{32}^{(1)}, \dots, a_{n2}^{(1)}$, itd.

Ogólnie, w k-tym kroku modyfikujemy elementy macierzy $(A^{(k-1)}|b^{(k-1)})$ zerując elementy k-tej kolumny znajdujące się pod główną przekątną, wg. wzorów

$$a_{ij}^{(k)} = \begin{cases} a_{ij}^{(k-1)} & & i \le k \\ a_{ij}^{(k-1)} - l_{ik} a_{kj}^{(k-1)} & \text{gdzie } l_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} & k < i \le n,\ k < j \le n \\ 0 & & k < i \le n,\ 1 < j \le k \end{cases}$$

oraz

$$b_i^{(k)} = \begin{cases} b_i^{(k-1)} & i \le k \\ b_i^{(k-1)} - l_{ik} b_k^{(k-1)} & k < i \le n. \end{cases}$$

Elementem głównym w k-tym kroku jest $a_{kk}^{(k-1)}$.

Uwaga. Powyższe wzory nie są sugestią implementacji. Efektywność takiego czy innego podejścia zależy od środowiska programistycznego.
W Matlabie można wykorzystać możliwość odwołań do podmacierzy (nie wykonywać obliczeń element po elemencie). Można to zrobić na różne sposoby.

Całkowity koszt eliminacji Gaussa: $\frac{n^3}{3} + \frac{n^2}{2}$ mnożeń/dzieleń.

Przykład 5.1.

Niech

$$A = \begin{pmatrix} 2 & 1 & -1 \\ -4 & -1 & 3 \\ 6 & 1 & -3 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}.$$

Rozwiąż układ równań liniowych $Ax = b$ metodą eliminacji Gaussa (GE). Oblicz wyznacznik macierzy A.

Rozwiązanie:
Pierwszym etapem jest sprowadzenie macierzy A do postaci trójkątnej górnej za pomocą

odpowiedniego ciągu operacji elementarnych. Operacje te wykonujemy na macierzy rozszerzonej układu. W tym przykładzie operacje będą następujące:

$$\left(\begin{array}{ccc|c} 2 & 1 & -1 & 1 \\ -4 & -1 & 3 & 1 \\ 6 & 1 & -3 & -1 \end{array}\right) \xrightarrow[r_3-3r_1]{r_2+2r_1} \left(\begin{array}{ccc|c} 2 & 1 & -1 & 1 \\ 0 & 1 & 1 & 3 \\ 0 & -2 & 0 & -4 \end{array}\right) \xrightarrow{r_3+2r_2} \left(\begin{array}{ccc|c} 2 & 1 & -1 & 1 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & 2 & 2 \end{array}\right).$$

Operacje elementarne, które wykorzystaliśmy, nie zmieniają wyznacznika macierzy, zatem

$$\det(A) = \det \begin{pmatrix} 2 & 1 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix} = 2 \cdot 1 \cdot 2 = 4.$$

Pozostaje rozwiązać układ równań z macierzą trójkątną. Przyjmując $x = (x_1, x_2, x_3)^T$ możemy zapisać ten układ jako

$$\begin{aligned} 2x_1 + x_2 - x_3 &= 1, \\ x_2 + x_3 &= 3, \\ 2x_3 &= 2. \end{aligned}$$

Korzystając z algorytmu opisanego na początku tego rozdziału, obliczymy najpierw x_3, potem x_2, a na końcu x_1, otrzymując

$$x = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}.$$

Łatwo zauważyć, że nie dla każdej macierzy metoda eliminacji Gaussa (GE) jest wykonalna. Opisany wyżej algorytm nie zadziała, gdy $a_{kk}^{(k-1)} = 0$ dla pewnego $k = 1, \ldots, n$. Powstaje pytanie: czy da się to przewidzieć, zanim zaczniemy obliczenia? Zauważmy, że jeśli $a_{11}^{(0)} = 0$, to na pewno metody GE nie da się zastosować. Ale jeśli $a_{ii}^{(0)} = 0$ dla pewnego $i = 2, \ldots, n$, to nie musi to oznaczać, że $a_{kk}^{(k-1)} = 0$ dla pewnego $k = 2, \ldots, n$. Z drugiej strony, jeśli $a_{ii}^{(0)} \neq 0$ dla wszystkich $i = 1, \ldots, n$, to nie daje to gwarancji, że $a_{kk}^{(k-1)} \neq 0$ dla wszystkich $k = 2, \ldots, n$.
Są jednak pewne klasy macierzy, dla których można pokazać, że metoda eliminacji Gaussa (GE) jest wykonalna. Tak jest w przypadku macierzy silnie diagonalnie dominujących (wierszowo lub kolumnowo) oraz macierzy dodatnio określonych.

Definicja 5.1. *Macierz $A \in \mathbb{R}^{n \times n}$ jest (silnie) diagonalnie dominująca wierszowo, jeśli dla każdego $i = 1, \ldots n$*

$$|a_{ii}| > \sum_{j=1, j \neq i}^{n} |a_{ij}|.$$

Czasem o macierzach silnie diagonalnie dominujących mówi się krócej: diagonalnie dominujące. Dlatego w powyższej definicji słowo „silnie" jest w nawiasie.

Jeśli którakolwiek z powyższych nierówności jest słaba, mówimy, że macierz jest *słabo* diagonalnie dominująca wierszowo.

Definicja 5.2. *Macierz $A \in \mathbb{R}^{n\times n}$ jest (silnie) diagonalnie dominująca kolumnowo, jeśli dla każdego $i = 1, \dots n$*

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ji}|.$$

Definicja 5.3. *Macierz $A \in \mathbb{R}^{n\times n}$ jest dodatnio określona, jeśli $x^T Ax > 0$ dla każdego niezerowego wektora $x \in \mathbb{R}^n$.*

Twierdzenie 5.1. *Jeśli $A \in \mathbb{R}^{n\times n}$ jest silnie diagonalnie dominująca (wierszowo lub kolumnowo), $b \in \mathbb{R}^n$, to metoda eliminacji Gaussa (GE) zastosowana do układu $Ax = b$ jest wykonalna.*

Twierdzenie 5.2. *Jeśli $A \in \mathbb{R}^{n\times n}$ jest dodatnio określona, $b \in \mathbb{R}^n$, to metoda eliminacji Gaussa (GE) zastosowana do układu $Ax = b$ jest wykonalna.*

5.2.2 Metoda eliminacji Gaussa z częściowym wyborem elementu głównego (GEPP)

W tym wariancie metody eliminacji Gaussa, k-ty krok eliminacji poprzedzamy wyborem *elementu głównego*. Szukamy takiego indeksu p, dla którego

$$|a_{pk}^{(k-1)}| = \max_{i=k,\dots,n} |a_{ik}^{(k-1)}|,$$

i zamieniamy miejscami wiersze p-ty i k-ty w macierzy $(A^{(k-1)}|b^{(k-1)})$ (jest to równoważne zamianie miejscami równań p-tego i k-tego, zatem rozwiązanie układu się nie zmieni).
W szczególności, przed pierwszym krokiem eliminacji szukamy w całej pierwszej kolumnie elementu o największym module i zamieniamy wiersze tak, aby ten element znalazł się w pierwszym wierszu. Przed drugim krokiem eliminacji elementu o największym module szukamy w drugiej kolumnie, wśród elementów leżących na głównej przekątnej i pod nią. Przed k-tym krokiem eliminacji przeszukujemy k-tą kolumnę – element leżący na diagonali i elementy pod nią.

Uwaga 5.1. *Zamiana w macierzy dwóch wierszy miejscami zmienia znak jej wyznacznika. Zatem, wykorzystując metodę GEPP do obliczenia wyznacznika, musimy uwzględnić liczbę zamian wierszy.*

Przykład 5.2.

Niech

$$A = \begin{pmatrix} -2 & 2 & 1 & 0 \\ 4 & -2 & 0 & 0 \\ 0 & 1 & 2 & 5 \\ 0 & 0 & 5 & 20 \end{pmatrix}, \quad b = \begin{pmatrix} -1 \\ 4 \\ 0 \\ -5 \end{pmatrix}.$$

Rozwiązać układ równań liniowych $Ax = b$ metodą eliminacji Gaussa z częściowym wyborem elementu głównego (GEPP). Obliczyć wyznacznik macierzy A.

Rozwiązanie:
Tak, jak w poprzednim przykładzie, będziemy działać na macierzy rozszerzonej układu. Wykonujemy następujące operacje:

$$\left(\begin{array}{cccc|c} -2 & 2 & 1 & 0 & -1 \\ 4 & -2 & 0 & 0 & 4 \\ 0 & 1 & 2 & 5 & 0 \\ 0 & 0 & 5 & 20 & 5 \end{array}\right) \xrightarrow{r_1 \leftrightarrow r_2} \left(\begin{array}{cccc|c} 4 & -2 & 0 & 0 & 4 \\ -2 & 2 & 1 & 0 & -1 \\ 0 & 1 & 2 & 5 & 0 \\ 0 & 0 & 5 & 20 & -5 \end{array}\right) \longrightarrow$$

$$\xrightarrow{r_2 + \frac{1}{2} r_1} \left(\begin{array}{cccc|c} 4 & -2 & 0 & 0 & 4 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 2 & 5 & 0 \\ 0 & 0 & 5 & 20 & -5 \end{array}\right) \xrightarrow{r_3 - r_2} \left(\begin{array}{cccc|c} 4 & -2 & 0 & 0 & 4 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 5 & -1 \\ 0 & 0 & 5 & 20 & -5 \end{array}\right) \longrightarrow$$

$$\xrightarrow{r_3 \leftrightarrow r_4} \left(\begin{array}{cccc|c} 4 & -2 & 0 & 0 & 4 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 5 & 20 & -5 \\ 0 & 0 & 1 & 5 & -1 \end{array}\right) \xrightarrow{r_4 - \frac{1}{5} r_3} \left(\begin{array}{cccc|c} 4 & -2 & 0 & 0 & 4 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 5 & 20 & -5 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right).$$

Rozwiązujemy układ równań z macierzą trójkątną i otrzymujemy

$$x = \begin{pmatrix} 2 \\ 2 \\ -1 \\ 0 \end{pmatrix}.$$

Pozostaje obliczyć wyznacznik. Mamy:

$$\det(A) = (-1)^2 \cdot 4 \cdot 1 \cdot 5 \cdot 1 = 20.$$

Zapis $(-1)^2$ jest po to, aby pokazać, że uwzględniamy dwie zamiany wierszy.

Zauważmy, że po zamianie wierszy znaleziony element o największym module znajduje się na głównej przekątnej macierzy, dzięki czemu wszystkie mnożniki l_{ik} są niewielkie, a

konkretnie $|l_{ik}| \leq 1$, dla $k < i \leq n$, $k = 1, \ldots, n$. Jakie to ma znaczenie, o tym powiemy dalszej części.
Tymczasem zauważmy jeszcze, że dzięki takiemu wyborowi elementu głównego, jeśli $\det A \neq 0$, to na diagonali nie pojawi się zero. (Dlaczego?) Oznacza to, że *metoda eliminacji Gaussa z częściowym wyborem elementu głównego jest wykonalna dla każdej nieosobliwej macierzy.* Jest to jedna z bardzo niewielu metod, dla których jedynym ograniczeniem jest istnienie jednoznacznego rozwiązania. Dlatego jest to prawdopodobnie najczęściej stosowana metoda rozwiązywania układów równań liniowych. Prawdopodobnie każde środowisko obliczeniowe ją zawiera. W Matlabie polecenie `x=A\b` czy funkcja `linsolve` korzystają z metody GEPP (nieco szczegółów można znaleźć w dokumentacji Matlaba).
Na koniec tego podrozdziału mała uwaga na marginesie odnośnie polecenia `x=A\b`. Składnia sugeruje dzielenie (lewostronne przez macierz `A`), zatem można by sądzić, że to polecenie jest równoważne działaniu `A^{-1}*b` czy `inv(A)*b`. Tak nie jest! Generalnie, rozwiązywanie układów równań liniowych z wykorzystaniem macierzy odwrotnej jest, ze względów numerycznych (głównie chodzi wpływ błędów zaokrągleń na wynik), praktyką bardzo niewłaściwą. W dalszej części notatek (po omówieniu poszczególnych metod) do tego wrócimy.

5.2.3 Metoda eliminacji Gaussa z pełnym wyborem elementów głównych (GECP)

Podobnie, jak w poprzednim wariancie metody eliminacji Gaussa, w tym także k-ty krok eliminacji poprzedzamy wyborem *elementu głównego.* W metodzie eliminacji Gaussa z pełnym wyborem elementu głównego szukamy takich indeksów p i q, dla których

$$|a_{pq}^{(k-1)}| = \max_{i=k,\ldots,n, j=k,\ldots,n} |a_{ij}^{(k-1)}|,$$

a następnie zamieniamy miejscami wiersze p-ty i k-ty oraz kolumny k-tą i q-tą w macierzy $(A^{(k-1)}|b^{(k-1)})$.
W szczególności, przed pierwszym krokiem eliminacji, elementu o największym module szukamy w całej macierzy układu, a następnie zamieniamy wiersze i kolumny tak, aby ten element znalazł się w pierwszym wierszu na głównej przekątnej. Przed drugim krokiem eliminacji elementu o największym module szukamy w podmacierzy $A^{(1)}(2\!:\!n, 2\!:\!n)$ (oznaczenie w konwencji matlabowej – podmacierz $A^{(1)}$ składająca się z wierszy od 2-go do n-go i kolumn od 2-ej do n-tej). Przed k-tym krokiem eliminacji przeszukujemy podmacierz $A^{(k-1)}(k\!:\!n, k\!:\!n)$.

Uwaga 5.2. *Zmiana kolejności kolumn w macierzy układu równań liniowych zmienia kolejność niewiadomych. Należy to uwzględnić przy wyznaczaniu rozwiązania.*

Implementując w Matlabie metodę eliminacji Gaussa z pełnym wyborem elementu głównego można utworzyć pomocniczy wektor indeksów `p = 1:n`, na którym będzie zapamię-

tywana kolejność niewiadomych (przy przestawianiu kolumn macierzy należy przestawiać odpowiednie elementy w wektorze `p`), a po obliczeniu rozwiązania `x` można odzyskać właściwą kolejność zmiennych instrukcją `x(p) = x`.

Uwaga 5.3. *Zamiana w macierzy dwóch wierszy lub dwóch kolumn miejscami zmienia znak jej wyznacznika. Zatem, wykorzystując metodę GECP do obliczenia wyznacznika, musimy uwzględnić łączną liczbę zamian wierszy i kolumn.*

5.3 Rozkład Cholesky'ego-Banachiewicza

Ponieważ rozkład Cholesky'ego-Banachiewicza jest ściśle związany z pojęciem dodatniej określoności macierzy, zacznijmy od przypomnienia twierdzenia Sylvestera, które można stosować do sprawdzenia, czy dana macierz jest dodatnio określona. Sprawdzenie na podstawie definicji, czy macierz posiada tę własność, może być w wielu przypadkach bardzo trudne.

Twierdzenie 5.3. *(Kryterium Sylvestera) Niech $A \in \mathbb{R}^{n\times n}$ będzie macierzą symetryczną. Macierz A jest dodatnio określona wtedy i tylko wtedy, gdy wszystkie jej wiodące minory główne są dodatnie, tj.*

$$\omega_k = \det \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \dots & a_{kk} \end{pmatrix} > 0 \quad dla \quad k = 1, \dots, n. \qquad \square$$

Rozkład Cholesky'ego-Banachiewicza jest jednym z wielu stosowanych w praktyce rozkładów macierzy na czynniki. We wszystkich rozkładach macierzy chodzi o to, żeby daną macierz zapisać jako iloczyn (dwóch lub więcej) macierzy o określonej strukturze i własnościach. W przypadku rozkładu Cholesky'ego-Banachiewicza macierz wyjściowa jest zapisywana jako iloczyn macierzy trójkątnej (dolnej) i jej transpozycji. Warunki istnienia takiego rozkładu precyzuje następujące twierdzenie.

Twierdzenie 5.4. *Jeśli $A \in \mathbb{R}^{n\times n}$ jest macierzą symetryczną i dodatnio określoną, to istnieje dokładnie jedna macierz trójkątna dolna L z dodatnimi elementami na głównej przekątnej, taka, że $A = LL^T$.* $\square$

Rozkład $A = LL^T$ nazywamy rozkładem Cholesky'ego-Banachiewicza macierzy A. Inne spotykane nazwy: rozkład Cholesky'ego (w literaturze anglojęzycznej), rozkład Banachiewicza-Cholesky'ego.

Uwaga 5.4. *Z Twierdzenia 5.4 wynika, że jeśli rozkład LL^T nie istnieje, to A nie może być macierzą symetryczną dodatnio określoną.*

Algorytm wyznaczania rozkładu Cholesky'ego-Banachiewicza otrzymujemy na podstawie równania $A = LL^T$, tj.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{12} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & \dots & l_{n1} \\ 0 & l_{22} & \dots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & l_{nn} \end{pmatrix}.$$

Po rozpisaniu iloczynu znajdującego się po prawej stronie powyższej równości możemy porównać odpowiadające sobie elementy macierzy A i LL^T. Otrzymamy zależności, z których wyznaczymy elementy macierzy L.

W przypadku gdy $A \in \mathbb{R}^{3\times 3}$, otrzymamy

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{pmatrix}.$$

Stąd po kolei wyznaczamy l_{11}, l_{21}, itd.

Algorytm (rozkład Cholesky'ego-Banachiewicza)

for $k = 1, 2, \dots, n$

$l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$

for $i = k+1, k+2, \dots, n$

$l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj}) / l_{kk}$

end

end

Jaki jest koszt (liczony liczbą operacji arytmetycznych) tego algorytmu?

Algorytm ten został wyprowadzony przy założeniu, że $A = A^T$. Zauważmy, że jeśli tak nie jest, to algorytm ten może nawet zwróci jakąś macierz L, ale nie będzie spełnione $A = LL^T$.

Wykorzystując rozkład Cholesky'ego-Banachiewicza możemy obliczyć wiodące minory główne macierzy A. Można udowodnić następujący fakt.

Fakt 5.1.

$$l_{11} = \sqrt{\omega_1}, \quad \textit{oraz} \quad l_{kk} = \sqrt{\frac{\omega_k}{\omega_{k-1}}} \quad \textit{dla} \quad k = 2, \dots, n.$$

Zauważmy, że algorytm Cholesky'ego-Banachiewicza nie będzie wykonalny, gdy w pewnym kroku pod pierwiastkiem pojawi się liczba niedodatnia (zakładamy, że obliczenia są prowadzone w dziedzinie rzeczywistej, a poza tym rozkład LL^T w omawianej tu, podstawowej, postaci jest definiowany z uwzględnieniem wyłącznie macierzy o elementach rzeczywistych). Jeśli zatem A jest symetryczna, a algorytm nie jest wykonalny, to to oznacza, że $\omega_k \leq 0$ dla pewnego $k = 1, \ldots, n$. A to, na podstawie kryterium Sylvestera, oznacza, że macierz nie jest dodatnio określona.
Prawdziwa jest też implikacja w drugą stronę. Jeśli rozkład uda się wyznaczyć, (tj. jeśli algorytm się nie załamie, a macierz jest symetryczna), to znaczy, że macierz jest dodatnio określona. To wynika z następującego faktu.

Fakt 5.2. *Jeśli $B \in \mathbb{R}^{n\times n}$ jest macierzą nieosobliwą, to macierz BB^T jest symetryczna i dodatnio określona.*

5.3.1 Zastosowania rozkładu LL^T

1. Sprawdzanie, czy dana symetryczna macierz jest dodatnio określona.
2. Obliczanie $\det(A)$:
$$\det(A) = \det(LL^T) = \det(L)\det(L^T) = \det(L)^2 = \prod_{i=1}^{n} l_{ii}^2.$$
3. Rozwiązywanie układów równań liniowych $Ax = b$.

   Podstawiając $A = LL^T$ otrzymujemy:
$$LL^T x = b.$$
   Rozwiązanie tego układu znajdujemy rozwiązując 2 układy z macierzami trójkątnymi:
$$Ly = b \qquad \text{oraz} \quad L^T x = y.$$
   Ważne: jest to przydatne zwłaszcza w zastosowaniach, gdy mamy do rozwiązania wiele układów równań z tą samą macierzą, a różnymi wektorami b.
4. Obliczanie A^{-1}:
$$A^{-1} = (LL^T)^{-1} = (L^T)^{-1}L^{-1} = (L^{-1})^T L^{-1}.$$

5.4 Rozkład LU

Rozkład LU to kolejny rozkład macierzy na czynniki, jaki tu omówimy. W tym rozkładzie macierz A zapisuje się jako iloczyn macierzy trójkątnej dolnej L oraz trójkątnej górnej U, przy czym na głównej przekątnej jednej z nich znajdują się wyłącznie jedynki. Jeśli L ma jedynki na głównej przekątnej, to mamy do czynienia z rozkładem wynikającym z metody GE lub rozkładem Doolittle'a, a jeśli U, to mówimy o rozkładzie Crouta.

5.4.1 Metoda GE a rozkład LU

Krok k-ty eliminacji Gaussa (zerowanie elementów w k-tej kolumnie pod główną przekątną) jest równoważny przemnożeniu macierzy $A^{(k-1)}$ z lewej strony przez macierz

$$L_k = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & -l_{k+1,k} & 1 & & \\ & & & -l_{k+2,k} & & 1 & \\ & & & \vdots & & & \ddots \\ & & & -l_{n,k} & & & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}, \tag{5.2}$$

gdzie $l_{k+1,k}, l_{k+2,k}, \ldots, l_{n,k}$ są współczynnikami eliminacji, a pozostałe elementy leżące poza główną przekątną są równe 0. Macierz tej postaci nazywamy macierzą eliminacji.

Fakt 5.3. *Jeśli L_k jest macierzą eliminacji postaci* (5.2)*, to*

$$L_k^{-1} = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & l_{k+1,k} & 1 & & \\ & & & l_{k+2,k} & & 1 & \\ & & & \vdots & & & \ddots \\ & & & l_{n,k} & & & 1 \end{pmatrix}.$$

Dowód tego faktu jest prosty – warto zrobić samemu.

Pierwszy krok eliminacji można zapisać jako:

$$L_1 A^{(0)} = A^{(1)},$$

gdzie $A^{(0)} = A$. Następne kroki:

$$\begin{aligned} L_2 A^{(1)} &= A^{(2)}, \\ L_3 A^{(2)} &= A^{(3)}, \\ &\ldots \\ L_{n-1} A^{(n-2)} &= A^{(n-1)} = U. \end{aligned}$$

Macierz $A^{(n-1)}$ jest trójkątna górna, zatem jest ona macierzą U z rozkładu. Aby znaleźć L, połączmy wszystkie powyższe zależności i zapiszmy wszystkie kroki razem:

$$L_{n-1} L_{n-2} \ldots L_1 A = U.$$

Stąd

$$L = (L_{n-1}L_{n-2}\dots L_1)^{-1} = L_1^{-1}\dots L_{n-2}^{-1}L_{n-1}^{-1}.$$

Jeśli rozpisalibyśmy iloczyn $L_1^{-1}\dots L_{n-2}^{-1}L_{n-1}^{-1}$, zauważylibyśmy, że

$$L = \begin{pmatrix} 1 & & & & & \\ l_{21} & 1 & & & & \\ l_{31} & l_{32} & 1 & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ \vdots & \vdots & & \ddots & 1 & \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{n,n-1} & 1 \end{pmatrix}. \tag{5.3}$$

5.4.2 Zastosowania rozkładu LU

1. Obliczanie $\det(A)$.

   W przypadku, gdy rozkład LU jest typu Doolittle'a, mamy:

   $$\det(A) = \det(LU) = \det(L)\det(U) = 1 \cdot \det(U) = \prod_{i=1}^{n} u_{ii}.$$

2. Rozwiązywanie układów równań liniowych $Ax = b$.

   Podstawiając $A = LU$ otrzymujemy:

   $$LUx = b.$$

   Rozwiązanie tego układu znajdujemy rozwiązując 2 układy z macierzami trójkątnymi:

   $$Ly = b \qquad \text{oraz} \quad Ux = y.$$

   Ważne: jest to przydatne zwłaszcza w zastosowaniach, gdy mamy do rozwiązania wiele układów równań z tą samą macierzą, a różnymi wektorami b.

3. Obliczanie A^{-1}:

   $$A^{-1} = (LU)^{-1} = U^{-1}L^{-1}.$$

Przykład 5.3.

Wyznaczyć rozkład LU macierzy

$$A = \begin{pmatrix} 2 & 1 & -1 \\ -4 & -1 & 3 \\ 6 & 1 & -3 \end{pmatrix}.$$

Następnie wykorzystując ten rozkład rozwiązać równanie macierzowe $AX = B$, gdzie

$$B = \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 3 & -1 \end{pmatrix}.$$

Szkic rozwiązania:
Do znalezienia rozkładu wykorzystamy macierze eliminacji. W pierwszym kroku metody eliminacji Gaussa wykonalibyśmy dwie operacje $r_2 + 2r_1$ i $r_3 - 3r_1$, co jest równoważne przemnożeniu macierzy A przez macierz

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}.$$

Mamy zatem

$$L_1 A = \begin{pmatrix} 2 & 1 & -1 \\ 0 & 1 & 1 \\ 0 & -2 & 0 \end{pmatrix} = A^{(1)}.$$

W drugim kroku eliminacji zerujemy element poddiagonalny w drugiej kolumnie

$$L_2 A^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & -1 \\ 0 & 1 & 1 \\ 0 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix} = U.$$

Macierz L możemy (szybko) zapisać korzystając ze wzoru (5.6), albo (nieco wolniej) ją wyprowadzić. Zrobimy to drugie. Połączymy zależności

$$L_1 A = A^{(1)} \quad \text{oraz} \quad L_2 A^{(1)} = U,$$

otrzymując

$$L_2 L_1 A = U.$$

Mamy zatem

$$L = (L_2 L_1)^{-1} = L_1^{-1} L_2^{-1},$$

skąd, korzystając z Faktu 5.3 i podstawiając wartości liczbowe, otrzymujemy

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix}.$$

Pozostaje rozwiązanie równania $AX = B$. Oznaczmy kolumny macierzy X przez z_1 i z_2, czyli $X = (z_1, z_2)$, a kolumny B przez v_1 i v_2, tj. $B = (v_1, v_2)$. Równanie $AX = B$ można zatem zapisać jako

$$A(z_1, z_2) = (v_1, v_2),$$

i dalej, równoważnie

$$(Az_1, Az_2) = (v_1, v_2).$$

Dwie macierze są równe, jeśli wszystkie odpowiadające sobie elementy są równe, można zatem powyższą równość rozpisać kolumnami

$$Az_1 = v_1,$$
$$Az_2 = v_2.$$

Otrzymaliśmy w ten sposób dwa układy równań liniowych.
Każdy z nich rozwiążemy korzystając z rozkładu LU. Będziemy mieli zatem w sumie do rozwiązania 4 układy równań liniowych z macierzami trójkątnymi. Szczegóły obliczeń pomijamy.

5.4.3 Rozkład Doolittle'a

W rozkładzie Doolittle'a macierze L i U są identyczne jak te otrzymane za pomocą macierzy eliminacji (patrz poprzedni podrozdział).
Algorytm wyznaczania rozkładu Doolittle'a otrzymujemy na podstawie równania $A = LU$, tj.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{pmatrix}.$$

Po rozpisaniu iloczynu znajdującego się po prawej stronie powyższej równości możemy porównać odpowiadające sobie elementy macierzy A i LU. Otrzymamy zależności, z których wyznaczymy elementy macierzy L i U.

Algorytm można znaleźć na przykład w [3], [1].

Jaki jest koszt (liczony liczbą operacji arytmetycznych) tego algorytmu dla macierzy $n \times n$?

5.4.4 Rozkład Crouta

W rozkładzie Crouta macierz L jest trójkątna dolna, a U trójkątna górna z jedynkami na głównej przekątnej.
Algorytm wyznaczania tego rozkładu otrzymujemy podobnie jak algorytm Doolittle'a. Po zapisaniu

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & \dots & u_{1n} \\ 0 & 1 & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix},$$

rozpisaniu iloczynu znajdującego się po prawej stronie oraz porównaniu odpowiadających sobie elementów macierzy A i LU otrzymamy zależności, z których wyznaczymy elementy macierzy L i U.

Algorytm można znaleźć na przykład w [3], [1].

Uwaga 5.5. *Nie dla każdej macierzy rozkład LU istnieje. Niech* $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$. *Co otrzymamy próbując znaleźć rozkład LU przez rozpisanie* $A = \begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix}$*?*

5.5 Rozkład $PA = LU$

W rozkładzie $PA = LU$ wykorzystywane są macierze transpozycji. Macierz transpozycji $T_{ij} \in \mathbb{R}^{n\times n}$ to macierz jednostkowa, w której dwa wiersze (i-ty i j-ty) zostały zamienione miejscami. Przemnożenie danej macierzy $A \in \mathbb{R}^{n\times n}$ przez T_{ij} z lewej strony (czyli działanie $T_{ij}A$) skutkuje zamianą wierszy i-tego i j-tego w macierzy A, a przemnożenie przez T_{ij} z prawej strony (czyli działanie AT_{ij}) – zamianą kolumn.

Fakt 5.4. *Jeśli* T_{ij} *jest macierzą transpozycji, to* $\det(T_{ij}) = -1$.

Fakt 5.5. *Jeśli* T_{ij} *jest macierzą transpozycji, to* $T_{ij}^{-1} = T_{ij}$.

Rozkład $PA = LU$ jest związany z metodą eliminacji Gaussa z częściowym wyborem elementów głównych (GEPP). Pierwszy krok można zapisać jako:

$$L_1 T_1 A^{(0)} = A^{(1)},$$

gdzie $A^{(0)} = A$, a T_1 jest macierzą transpozycji wykorzystaną w pierwszym kroku. Dla uproszczenia zapisu macierz transpozycji ma tutaj tylko jeden indeks dolny, oznaczający numer kroku. W szczególności T_1 może być macierzą jednostkową (jeśli mamy do czynienia z sytuacją, gdzie zamiana wierszy nie jest potrzebna).
Następne kroki:

$$\begin{aligned} L_2 T_2 A^{(1)} &= A^{(2)}, \\ L_3 T_3 A^{(2)} &= A^{(3)}, \\ &\dots \\ L_{n-1} T_{n-1} A^{(n-2)} &= A^{(n-1)} = U. \end{aligned}$$

Macierz $A^{(n-1)}$ jest trójkątna górna, zatem jest ona macierzą U z rozkładu. Połączmy wszystkie powyższe zależności i zapiszmy wszystkie kroki razem:

$$L_{n-1} T_{n-1} L_{n-2} T_{n-2} \dots L_1 T_1 A = U.$$

Dążymy do uzyskania rozkładu $PA = LU$, zatem

$$L^{-1}P = L_{n-1}T_{n-1}L_{n-2}T_{n-2}\dots L_1T_1.$$

Czy da się jakoś rozdzielić ten iloczyn, aby uzyskać macierze L i P? Okazuje się, że

$$P = T_{n-1}T_{n-2}\dots T_1, \tag{5.4}$$

natomiast L, podobnie jak w przypadku rozkładu LU, zawiera w k-tej kolumnie mnożniki z k-tego kroku eliminacji, przy czym być może w zmienionej kolejności. Wyznacznik macierzy P jest równy 1 lub -1, w zależności od tego, ile faktycznych transpozycji zawiera iloczyn $T_{n-1}T_{n-2}\dots T_1$.
Rozpiszmy to bardziej szczegółowo w przypadku, gdy $A \in \mathbb{R}^{4\times 4}$ (przypadek ogólny jest analogiczny). Mamy:

$$L_3\,T_3\,L_2\,T_2\,L_1\,T_1\,A = U.$$

Ponieważ $T_iT_i = I$, gdzie T_i jest dowolną macierzą transpozycji, możemy takie iloczyny wstawić w dowolnych (ale strategicznie wybranych) miejscach i otrzymać:

$$L_3\,T_3L_2(T_3T_3)T_2L_1(T_2T_2)T_1A = U.$$

Wstawmy jeszcze jeden iloczyn:

$$L_3\,T_3L_2T_3\,T_3T_2L_1T_2(T_3T_3)T_2T_1A = U,$$

pogrupujmy macierze po lewej stronie znaku równości

$$L_3(T_3L_2T_3)(T_3T_2L_1T_2T_3)T_3T_2T_1A = U,$$

i wprowadźmy oznaczenia:

$$\tilde{L}_3 = L_3, \quad \tilde{L}_2 = T_3L_2T_3, \quad \tilde{L}_1 = T_3T_2L_1T_2T_3,$$

co pozwoli nam zapisać powyższą zależność nieco krócej:

$$\tilde{L}_3\tilde{L}_2\tilde{L}_1T_3T_2T_1A = U, \tag{5.5}$$

Z zależności (5.5) wynika, że (porównaj wzór (5.4))

$$P = T_3T_2T_1$$

oraz

$$L = (\tilde{L}_3\tilde{L}_2\tilde{L}_1)^{-1} = \tilde{L}_1^{-1}\tilde{L}_2^{-1}\tilde{L}_3^{-1}. \tag{5.6}$$

Przyjrzyjmy się teraz, jaką postać mają macierze $\tilde{L}_2$ i $\tilde{L}_1$. Macierz L_2 jest macierzą eliminacji Gaussa:

$$L_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -l_{32} & 1 & 0 \\ 0 & -l_{42} & 0 & 1 \end{pmatrix},$$

natomiast

$$\tilde{L}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -l_{42} & 1 & 0 \\ 0 & -l_{32} & 0 & 1 \end{pmatrix} \quad \text{lub} \quad \tilde{L}_2 = L_2,$$

przy czym druga ewentualność ma miejsce gdy $T_3 = I$. Widać zatem, że podziałanie na macierz L_2 macierzą T_3 z każdej strony nie zmieniło jej struktury ($\tilde{L}_2$ jest także macierzą eliminacji Gaussa, z być może zmienioną kolejnością współczynników eliminacji w drugiej kolumnie).
Podobną własność ma macierz $\tilde{L}_1$ (warto sobie rozpisać).
Wynika stąd, że macierz L określona wzorem (5.6) jest trójkątna dolna z jedynkami na głównej przekątnej, zatem faktycznie otrzymaliśmy rozkład $PA = LU$, gdzie każda z macierzy ma taką postać, jak zdefiniowaliśmy na początku.

5.5.1 Zastosowania rozkładu $PA = LU$

1. Obliczanie $\det(A)$.

   Mamy:

   $$\det(P)\det(A) = \det(L)\det(U) = \det(U),$$

   przy czym $\det(P) = \pm 1$ (w zależności od liczby faktycznych zamian wierszy), skąd łatwo już obliczymy $\det(A)$.

2. Rozwiązywanie układów równań liniowych $Ax = b$.

   Mnożąc (z lewej strony) układ $Ax = b$ stronami przez P (co jest równoważne zmianie kolejności równań) i podstawiając $PA = LU$ otrzymujemy:

   $$LUx = Pb.$$

   Rozwiązanie tego układu znajdujemy rozwiązując 2 układy z macierzami trójkątnymi:

   $$Ly = Pb \qquad \text{oraz} \quad Ux = y.$$

   Ważne: jak w przypadku wszystkich rozkładów, jest to przydatne zwłaszcza w zastosowaniach, gdy mamy do rozwiązania wiele układów równań z tą samą macierzą, a różnymi wektorami b.

3. Obliczanie A^{-1}.

5.6 Rozkład $PAQ = LU$

Rozkład $PAQ = LU$ jest powiązany z metodą eliminacji Gaussa z pełnym wyborem elementów głównych (GECP). Macierz P odpowiada za zmianę kolejności wierszy, macierz Q – kolumn, macierz L jest trójkątna dolna z jedynkami na głównej przekątnej, a macierz U jest trójkątna górna.
Rozkład ten ma podobne zastosowania jak pozostałe omówione tu rozkłady.

5.7 Układy równań liniowych – uwarunkowanie zadania

W tym podrozdziale zastanowimy się nad uwarunkowaniem zadania rozwiązywania układów równań liniowych $Ax = b$, gdzie $A \in \mathbb{R}^{n\times n}$ jest nieosobliwa a $x, b \in \mathbb{R}^n$. Jeśli układ $Ax = b$ jest źle uwarunkowany, to rozwiązanie obliczone w arytmetyce zmiennopozycyjnej zwykle jest niedokładne. „Of course, we may always be lucky, but we cannot be guaranteed it" (James R. Bunch).

Będziemy chcieli zbadać, jaki wpływ mają zaburzenia danych zadania (czyli zaburzenia A i b) na zaburzenia wyniku (czyli x).

Najpierw sprawdzimy od czego zależy uwarunkowanie układu równań liniowych w sytuacji, gdy zaburzenie występuje tylko w wektorze b. Zaburzony układ ma postać

$$A\tilde{x} = \tilde{b},$$

gdzie

$$\tilde{b} = b + f, \qquad \|f\| \leq \varepsilon\|b\|, \tag{5.7}$$

przy czym ε jest małą liczbą (zwykle przyjmuje się, że jest to precyzja obliczeń).

Zastanówmy się teraz, jak bardzo $\tilde{x}$ może różnić się od x. Jaki jest wpływ zaburzenia (błędu) b na rozwiązanie x? Aby to określić, najpierw oszacujemy błąd bezwzględny rozwiązania, czyli $\|x - \tilde{x}\|$. Wektor x jest rozwiązaniem układu $Ax = b$, a $\tilde{x}$ rozwiązaniem układu zaburzonego, zatem prawdą jest, że $x = A^{-1}b$ oraz $\tilde{x} = A^{-1}\tilde{b}$. Mamy zatem:

$$\|x - \tilde{x}\| = \|A^{-1}b - A^{-1}\tilde{b}\| = \|A^{-1}(b - \tilde{b})\| \leq \|A^{-1}\|\,\|b - \tilde{b}\|,$$

przy czym norma macierzowa, która się tu pojawiła, jest dowolną normą zgodną z użytą normą wektorową.
Oszacujemy teraz błąd względny rozwiązania przez błąd względny danych. Przyjmijmy, że $||x|| \neq 0$ i $||b|| \neq 0$. Skoro $Ax = b$, to $||Ax|| = ||b||$, zatem możemy zapisać

$$\|x - \tilde{x}\| \leq \|A^{-1}\|\,\|b - \tilde{b}\| = \|A^{-1}\|\,\|Ax\|\frac{\|b - \tilde{b}\|}{\|b\|} \leq \|A^{-1}\|\,\|A\|\,\|x\|\frac{\|b - \tilde{b}\|}{\|b\|}.$$

Dzieląc stronami przez $\|x\|$ otrzymujemy:

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \|A^{-1}\|\,\|A\|\frac{\|b - \tilde{b}\|}{\|b\|}.$$

Korzystając z tego, że $\|b - \tilde{b}\| \leq \varepsilon \|b\|$ (założyliśmy to na początku – w (5.7)), dostajemy

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \varepsilon \|A^{-1}\| \, \|A\| \leq \varepsilon \, \mathrm{cond}(A),$$

gdzie

$$\mathrm{cond}(A) = \|A^{-1}\| \, \|A\|$$

jest wskaźnikiem uwarunkowania macierzy A.

Uwaga 5.6. *Obliczając wskaźnik uwarunkowania można stosować dowolną normę zgodną z normą wektorową. Oczywiście, w zależności od użytej normy wskaźnik uwarunkowania może mieć różną wartość liczbową.*

Jeśli chcemy zaznaczyć, w której normie wskaźnik uwarunkowania jest liczony, możemy wykorzystać oznaczenia:

$$\begin{aligned} \mathrm{cond}_2(A) &= \|A^{-1}\|_2 \, \|A\|_2, \\ \mathrm{cond}_F(A) &= \|A^{-1}\|_F \, \|A\|_F, \\ \mathrm{cond}_1(A) &= \|A^{-1}\|_1 \, \|A\|_1, \quad \text{itd.} \end{aligned}$$

Wskaźnik uwarunkowania macierzy można w Matlabie obliczyć funkcją wbudowaną `cond`. Użycie: `cond(A)` (obliczony zostanie $\mathrm{cond}_2(A)$) lub np. `cond(A,1)` (drugi parametr określa normę, czyli w tym przypadku obliczony zostanie $\mathrm{cond}_1(A)$). Więcej informacji: `help cond`.

Przykład 5.4. *Kilka układów źle uwarunkowanych*

1. Rozważmy układ równań

$$\begin{aligned} 10x_1 + 7x_2 &= 17, \\ 0.9999x_1 + 0.7x_2 &= 1.6999, \end{aligned}$$

   którego dokładnym rozwiązaniem jest:

$$x_1 = x_2 = 1.$$

   Macierz tego układu jest źle uwarunkowana. Wskaźnik uwarunkowania wynosi:

$$\mathrm{cond}_2(A) \approx 2.1499e{+}5,$$

   zatem możemy oczekiwać, że mała zmiana danych (macierzy układu lub wektora prawej strony) spowoduje dużą zmianę rozwiązania.

   Rozważmy teraz następujący zaburzony układ:

$$\begin{aligned} 10x_1 + 7x_2 &= 17, \\ 0.9999x_1 + 0.7x_2 &= 1.7. \end{aligned}$$

Rozwiązanie:

$$x_1 = 0,$$
$$x_2 = \frac{17}{7} \approx 2.4286.$$

Uwaga na marginesie: warto rozwiązać oba układy ręcznie, a potem w Matlabie (funkcją wbudowaną) i porównać uzyskane wyniki.

Uwaga. Macierz tego układu jest bliska macierzy osobliwej (jej wyznacznik jest mały a wiersze są „prawie liniowo zależne"), zatem można się było spodziewać niedokładności w rozwiązaniu. Istnieje jednak wiele macierzy źle uwarunkowanych o wyznacznikach znacząco różnych od zera.

2. Macierz Hilberta

   $H = \left\{h_{ij}\right\}_{i,j=1}^{n}$ to macierz o elementach

   $$h_{ij} = \frac{1}{i+j-1}.$$

   Jest ona bardzo źle uwarunkowana, na przykład:

   - dla $n = 3$, $\mathrm{cond}_2(H) \approx 524$,
   - dla $n = 5$, $\mathrm{cond}_2(H) \approx 4.7661e{+}05$,
   - dla $n = 10$, $\mathrm{cond}_2(H) \approx 1.6025e{+}13$.

   W Matlabie można tę macierz wygenerować poleceniem `hilb(n)`, gdzie `n` definiuje rozmiar.

3. Macierz Pascala

   $A = \left\{a_{ij}\right\}_{i,j=1}^{n}$ to macierz o elementach

   $$\begin{aligned} a_{1,j} &= 1 \quad \text{dla} \quad j = 1, 2, \ldots, n, \\ a_{i,1} &= 1 \quad \text{dla} \quad i = 1, 2, \ldots, n, \\ a_{ij} &= a_{i,j-1} + a_{i-1,j} \quad \text{dla} \quad i, j = 2, 3, \ldots, n. \end{aligned}$$

   Wyznacznik macierzy Pascala dowolnego wymiaru jest równy 1. Wskaźnik uwarunkowania rośnie dość szybko razem z n (choć nie aż tak szybko, jak w przypadku macierzy Hilberta). Warto to samemu sprawdzić w Matlabie! Do generowania macierzy Pascala służy funkcja `pascal`.

Wskaźnik uwarunkowania zadania rozwiązywania układów równań liniowych wyprowadziliśmy w przypadku, gdy zaburzenie występuje tylko w wektorze b. Jeśli zaburzenie jest tylko w macierzy układu lub we wszystkich danych (tj. zarówno w macierzy układu, jak i w wektorze b), wskaźnik uwarunkowania wyraża się tym samym wzorem.

5.8 Lokalizacja wartości własnych macierzy

Poniższe twierdzenie będzie szerzej omówione na wykładzie.

5.8.1 Twierdzenie Gerszgorina

Twierdzenie 5.5. *(S. Gerszgorin, 1931) Niech $A \in \mathbb{C}^{n\times n}$. Dla każdej wartości własnej λ macierzy A istnieje taki indeks $i \in \{1, 2, \ldots, n\}$, że*

$$\lambda \in K_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{\substack{j=1,\ldots,n \\ j\neq i}} |a_{ij}|\}.$$

Ponadto,

$$\sigma(A) \subset \bigcup_{i=1}^{n} K_i = G(A),$$

gdzie $\sigma(A)$ oznacza zbiór wszystkich wartości własnych A. □

Koła K_i nazywane są kołami Gerszgorina, a zbiór $G(A)$ nazywany jest zbiorem Gerszgorina macierzy A.

Twierdzenie 5.6. *Niech $A \in \mathbb{C}^{n\times n}$. Jeśli k ($k \geq 1$) kół Gerszgorina tworzy zbiór S, spójny i rozłączny z pozostałymi kołami, to S zawiera dokładnie k wartości własnych macierzy A.* □

5.9 Metody iteracyjne rozwiązywania układów równań liniowych

Niech, jak poprzednio, $Ax = b$ będzie układem równań liniowych, który chcemy rozwiązać, przy czym $A \in \mathbb{R}^{n\times n}$ jest nieosobliwa a $x, b \in \mathbb{R}^n$.
Idea metod iteracyjnych jest następująca: startując z danego przybliżenia początkowego

$$x^{(0)} = \begin{pmatrix} x_1^{(0)} \\ \vdots \\ x_n^{(0)} \end{pmatrix} \in \mathbb{R}^n$$

tworzymy ciąg kolejnych przybliżeń $\{x^{(k)}\}$, taki, że $x^{(k)} \to x$ przy $k \to \infty$.
Każdy element tego ciągu jest wektorem (tj. $x^{(k)} \in \mathbb{R}^n$ dla $k = 0, 1, \ldots$). Zbieżność jest rozumiana w sensie normy, tj. $||x^{(k)} - x|| \to 0$.

Metody iteracyjne mają duże znaczenie zwłaszcza przy rozwiązywaniu wielkich układów równań. Mogą być szybsze i mogą mieć mniejsze wymagania pamięciowe niż metody bezpośrednie. W praktyce nierzadko spotyka się układy, które mają kilkaset tysięcy czy kilka milionów niewiadomych. Często też zdarza się, że układ ma określoną strukturę, np.

elementy niezerowe znajdują się tylko na określonych miejscach w macierzy, najczęściej w pobliżu diagonali. Można (a często trzeba, jeśli chcemy uzyskać rozwiązanie w rozsądnym czasie) to odpowiednio wykorzystać w implementacji.

Metody, które tu omówimy, należą do grupy tzw. *metod iteracji prostej*. Zasada ich konstrukcji jest taka, że układ wyjściowy $Ax = b$ zapisujemy w postaci równoważnej $x = Bx + c$, a następnie kolejne przybliżenia obliczamy według wzoru

$$x^{(k+1)} = Bx^{(k)} + c, \quad \text{dla} \quad k = 0, 1, \ldots. \tag{5.8}$$

Macierz B nazywamy macierzą iteracji.
Jeśli nie znamy przybliżonego rozwiązania, to jako przybliżenie początkowe $x^{(0)}$ można wziąć wektor o elementach losowych albo wektor zerowy.

5.9.1 Zbieżność metod iteracyjnych

Najczęściej ciężko jest stwierdzić czy dany wektor $x^{(0)}$ jest dobrym przybliżeniem początkowym (tj. gwarantującym zbieżność metody). Dlatego szuka się warunków zapewniających zbieżność dla każdego $x^{(0)} \in \mathbb{R}^n$.
Następujące twierdzenie jest podstawowym warunkiem dotyczącym zbieżności metod iteracyjnych postaci (5.8).

Twierdzenie 5.7. *Metoda iteracyjna postaci* (5.8) *jest zbieżna globalnie (tj. dla każdego przybliżenia początkowego $x^{(0)}$) wtedy i tylko wtedy, gdy $\rho(B) < 1$, gdzie $\rho(B)$ jest promieniem spektralnym macierzy B.*

Dla zainteresowanych: dowód tego twierdzenia można znaleźć na przykład w [1] oraz [8] (rozdział 4.2.1). Druga książka dostępna jest na stronie domowej Autora.

Uwaga 5.7. *Im wartość $\rho(B)$ jest mniejsza, tym zbieżność jest szybsza (tj. tym mniej iteracji jest potrzebnych do uzyskania rozwiązania z zadaną dokładnością).*

Skorzystanie z twierdzenia 5.7 wymaga znajomości $\rho(B)$. W przypadku dowolnej macierzy może on być trudny do wyznaczenia. Metody numeryczne znajdowania wartości własnych macierzy mogą być bardziej pracochłonne (w sensie liczby potrzebnych operacji arytmetycznych) niż metody rozwiązywania układów równań liniowych. (Zagadnienie to będzie omawiane na Metodach Numerycznych 2.) Dlatego łatwiej może być stwierdzić zbieżność korzystając z oszacowań wartości własnych. Wykorzystując oszacowanie z zadania H2 otrzymamy warunek dostateczny zbieżności.

Twierdzenie 5.8. *(Wniosek z twierdzenia 5.7) Jeśli $\|B\| < 1$ (gdzie $\|\cdot\|$ jest normą zgodną z pewną normą wektorową), to metoda iteracyjna postaci* (5.8) *jest zbieżna globalnie.*

Dowód. Z nierówności $|\lambda| \leq ||B||$, gdzie λ jest dowolną wartością własną B, wynika, że $\rho(B) \leq ||B||$. Jeśli zatem $\|B\| < 1$, to $\rho(B) < 1$. Na mocy twierdzenia 5.7 metoda jest więc zbieżna globalnie. □

Uwaga 5.8. *Twierdzenie 5.8 jest słabsze niż twierdzenie 5.7, jest warunkiem tylko dostatecznym.*
W praktyce oznacza to, że jeśli obliczymy wartości kilku norm macierzy B i żadna nie będzie mniejsza niż 1, to nadal nic nie wiemy o zbieżności.

5.9.2 Możliwe warunki zakończenia obliczeń

Niech d będzie parametrem definiującym dokładność (niewielką liczbą rzeczywistą dodatnią). Jako kryterium zakończenia obliczeń można użyć jednego z następujących warunków:

1) $||Ax^{(k)} - b|| < d$ (warunek bardziej kosztowny niż następne – wymaga obliczenia iloczynu $Ax^{(k)}$),

2) $\|x^{(k+1)} - x^{(k)}\| < d$ („błąd" bezwzględny),

3) $\|x^{(k+1)} - x^{(k)}\| < d\|x^{(k)}\|$ („błąd" względny),

4) (warunek Gilla) $\|x^{(k+1)} - x^{(k)}\| < d_1\|x^{(k)}\| + d_2$,

gdzie d_1 i d_2 to parametry określające dokładność (liczby dodatnie, bliskie zeru, przy czym $d_1 > d_2$), przykładowo $d_1 = 10^{-10}$ i $d_2 = 10^{-20}$ lub $d_1 = 10^{-13}$ i $d_2 = 10^{-25}$ a $\|\cdot\|$ jest dowolną normą wektorową.
Cudzysłów przy słowie „błąd" jest dlatego, że nie jest to tak naprawdę błąd, ponieważ w tych warunkach nie pojawia się dokładne rozwiązanie.
Warunek Gilla to połączenie dwóch poprzednich warunków. Jeśli wartość $\|x^{(k)}\|$ jest odpowiednio bliska zeru, sprawdzana będzie dokładność bezwzględna ($d_1\|x^{(k)}\|$ jest dużo mniejsze niż d_2), a jeśli wartość $\|x^{(k)}\|$ jest wystarczająco duża, to sprawdzana będzie dokładność względna ($d_1\|x^{(k)}\|$ będzie większe niż d_2 i warunek Gilla sprowadzi się do warunku 3)).

5.9.3 Metoda Jacobiego

Załóżmy, że elementy znajdujące się na głównej przekątnej macierzy A sa niezerowe, tj. $a_{ii} \neq 0$ dla $i = 1, \ldots, n$.
Metodę Jacobiego można wyprowadzić na dwa sposoby. Pokażemy oba.

Pierwszy sposób.

Zapiszmy układ równań $Ax = b$ w postaci

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n. \end{aligned}$$

Z pierwszego równania wyznaczymy x_1, z drugiego x_2, itd.:

$$\begin{aligned} x_1 &= (b_1 - \sum_{j=2}^{n} a_{1j}x_j)/a_{11}, \\ x_2 &= (b_2 - \sum_{j=1, j\neq 2}^{n} a_{2j}x_j)/a_{22}, \\ &\ldots \\ x_n &= (b_n - \sum_{j=1}^{n-1} a_{nj}x_j)/a_{nn}. \end{aligned}$$

Wzory, które otrzymaliśmy, będą podstawą iteracji. Zaczynając z danego przybliżenia początkowego $x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})^T \in \mathbb{R}^n$ obliczamy kolejne przybliżenia $x^{(k+1)}$ $(k = 0, 1, \ldots)$ według wzorów:

$$\begin{aligned} x_1^{(k+1)} &= (b_1 - \sum_{j=2}^{n} a_{1j}x_j^{(k)})/a_{11}, \\ x_2^{(k+1)} &= (b_2 - \sum_{j=1, j\neq 2}^{n} a_{2j}x_j^{(k)})/a_{22}, \\ &\ldots \\ x_n^{(k+1)} &= (b_n - \sum_{j=1}^{n-1} a_{nj}x_j^{(k)})/a_{nn}. \end{aligned} \tag{5.9}$$

Algorytm (metoda Jacobiego)

$x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})^T$ – przybliżenie początkowe

for $k = 0, 1, \ldots,$ (dopóki nie będzie spełniony wybrany warunek stopu)
  for $i = 1, 2, \ldots, n$
    $x_i^{(k+1)} = (b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)})/a_{ii}$
  end
end

Uwaga. Przy implementacji metod iteracyjnych oprócz któregoś z kryteriów zbieżności wymienionych w podrozdziale 5.9.2 dobrze jest ustalić też pewną maksymalną liczbę iteracji (np. 1000), po wykonaniu których program zakończy działanie, niezależnie czy będzie spełniony wybrany warunek z podrozdziału 5.9.2, czy nie. Zapewni to zakończenie obliczeń w przypadku gdy metoda nie jest zbieżna.

Uwaga 5.9. *Nigdy nie należy stosować ustalonej liczby iteracji jako jedynego kryterium zakończenia obliczeń (tj. np. program zawsze kończy działanie po wykonaniu dokładnie 100 kroków). Liczba kroków potrzebna do uzyskania dobrego przybliżenia rozwiązania zależy od danych wejściowych. Dotyczy to wszystkich numerycznych metod iteracyjnych (nie tylko tych, które służą do rozwiązywania układów równań liniowych).*

Wzory (5.9) można zapisać w postaci macierzowej

$$x^{(k+1)} = B_J x^{(k)} + c_J,$$

gdzie

$$x^{(k)} = \begin{pmatrix} x_1^{(k)} \\ \vdots \\ x_n^{(k)} \end{pmatrix},$$

$$B_J = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \dots & -\frac{a_{2n}}{a_{22}} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \dots & 0 \end{pmatrix}, \tag{5.10}$$

$$c_J = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{pmatrix}.$$

Macierz B_J jest *macierzą iteracji* w metodzie Jacobiego.

Uwaga 5.10. *Macierz iteracji zależy wyłącznie od macierzy układu równań (nie zależy od wektora b). W związku z tym zbieżność metody także zależy wyłącznie od macierzy układu równań. Jeśli metoda Jacobiego jest zbieżna, to jest zbieżna dla wszystkich $b \in \mathbb{R}^n$.*

Drugi sposób wyprowadzenia metody Jacobiego.

Zapiszmy macierz A w postaci sumy trzech macierzy, L (macierz trójkątna dolna zawierająca elementy leżące *pod* diagonalą A), D (macierz diagonalna, zawierająca elementy

diagonali A) i U (macierz trójkątna górna zawierająca elementy leżące *ponad* diagonalą A):

$$A = L + D + U =$$

$$= \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ a_{21} & 0 & \dots & 0 & 0 \\ a_{31} & a_{32} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{n,n-1} & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & 0 & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{n-1,n} \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}.$$

Po podstawieniu tak zapisanej macierzy A do układu $Ax = b$ otrzymujemy

$$(L + D + U)x = b,$$

co można zapisać jako

$$Dx + (L + U)x = b,$$

$$Dx = -(L + U)x + b,$$

$$x = -D^{-1}(L + U)x + D^{-1}b.$$

Pozostaje jeszcze tylko dopisać numer iteracji:

$$x^{(k+1)} = -D^{-1}(L + U)x^{(k)} + D^{-1}b, \quad k = 0, 1, \dots$$

Otrzymaliśmy metodę Jacobiego zapisaną w sposób macierzowy:

$$x^{(k+1)} = B_J x^{(k)} + c_J, \quad k = 0, 1, \dots,$$

gdzie

$$B_J = -D^{-1}(L + U) \quad \text{a} \quad c_J = D^{-1}b.$$

Przykład 5.5.

Zbadamy zbieżność metody Jacobiego zastosowanej do układu $Ax = b$, gdzie

$$A = \begin{pmatrix} 3 & -2 & 0 \\ 4 & 3 & 0 \\ -1 & 1 & -1 \end{pmatrix}$$

a $b \in \mathbb{R}^3$.

Rozwiązanie (i pewne wyjaśnienia):
Aby zbadać zbieżność, musimy znać macierz iteracji. Łatwo ją znajdziemy korzystając z (5.10). Mamy:

$$B_J = \begin{pmatrix} 0 & \frac{2}{3} & 0 \\ -\frac{4}{3} & 0 & 0 \\ -1 & 1 & 0 \end{pmatrix}.$$

Warunkiem koniecznym i dostatecznym zbieżności jest $\rho(B_J) < 1$. Jednak, ponieważ obliczanie wartości własnych może być dość pracochłonne, spróbujemy najpierw skorzystać z twierdzenia 5.8.
Mamy:

$$\|B_J\|_F = \sqrt{\frac{4}{9} + \frac{16}{9} + 1 + 1} = \frac{\sqrt{38}}{3} > 1,$$

$$\|B_J\|_1 = \max\left\{\frac{7}{3}, \frac{5}{3}, 0\right\} = \frac{7}{3} > 1,$$

$$\|B_J\|_\infty = \max\left\{\frac{2}{3}, \frac{4}{3}, 2\right\} = \frac{4}{3} > 1,$$

zatem, jak na razie, ciągle nic nie wiemy o zbieżności. Możemy obliczyć też wartości innych norm, jednak nie ma gwarancji, że znajdziemy taką normę, która jest mniejsza niż 1. Możliwe jest też oczywiście, że metoda nie jest zbieżna.
Nie będziemy już obliczać norm, tylko wyznaczymy $\rho(B_J)$. Wyznaczając wielomian charakterystyczny otrzymujemy:

$$\det(B_J - \lambda I) = \begin{vmatrix} -\lambda & \frac{2}{3} & 0 \\ -\frac{4}{3} & -\lambda & 0 \\ -1 & 1 & -\lambda \end{vmatrix} = -\lambda\left(\lambda^2 + \frac{8}{9}\right) = 0,$$

skąd wyznaczymy spektrum (zbiór wartości własnych) macierzy B_J:

$$\sigma(B_J) = \left\{0, \frac{2\sqrt{2}}{3}i, -\frac{2\sqrt{2}}{3}i\right\}.$$

Zatem

$$\rho(B_J) = \frac{2\sqrt{2}}{3} < 1,$$

co oznacza, że metoda Jacobiego jest zbieżna globalnie.

5.9.4 Metoda Gaussa-Seidla

Metodę Gaussa-Seidla, podobnie jak metodę Jacobiego, można zapisać dwojako.
Spójrzmy najpierw na iterację (5.9). Obliczając $x_i^{(k+1)}$ $(i = 2, 3, \ldots, n)$ mamy już obliczone przybliżenia $x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$. Możemy zatem użyć tych „najświeższych" przybliżeń przy obliczaniu $x_i^{(k+1)}$. I tak właśnie jest skonstruowana metoda Gaussa-Seidla.

Algorytm (metoda Gaussa-Seidla)

$x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})^T$ – przybliżenie początkowe

for $k = 0, 1, \ldots,$ (dopóki nie będzie spełniony wybrany warunek stopu)
  for $i = 1, 2, \ldots, n$
    $x_i^{(k+1)} = (b_i - \sum_{j=1, j<i}^{n} a_{ij} x_j^{(k+1)} - \sum_{j=1, j>i}^{n} a_{ij} x_j^{(k)})/a_{ii}$
  end
end

Macierzy iteracji dla tej metody nie da się tak prosto zapisać jak dla metody Jacobiego (wzór analogiczny do (5.10) byłby dużo bardziej złożony), ponieważ wymagałoby to zapisania odwrotności macierzy trójkątnej. Dlaczego tak jest, zobaczymy zapisując metodę Gaussa-Seidla w postaci macierzowej.
Zaczniemy identycznie, jak w przypadku metody Jacobiego – zapiszmy macierz A jako $A = L + D + U$, gdzie L jest macierzą trójkątną dolną zawierającą elementy leżące *pod* diagonalą A), D jest macierzą diagonalną, zawierająca elementy diagonali A, a U jest macierzą trójkątną górną zawierającą elementy leżące *ponad* diagonalą A.
Po podstawieniu tak zapisanej macierzy A do układu $Ax = b$ otrzymujemy

$$(L + D + U)x = b.$$

Aby otrzymać iterację Gaussa-Seidla, musimy pogrupować składniki po lewej stronie następująco

$$(L + D)x + Ux = b,$$

skąd

$$(L + D)x = -Ux + b,$$

a następnie

$$x = -(L + D)^{-1}Ux + (L + D)^{-1}b.$$

Pozostaje jeszcze tylko dopisać numer iteracji:

$$x^{(k+1)} = -(L + D)^{-1}Ux^{(k)} + (L + D)^{-1}b, \quad k = 0, 1, \ldots.$$

Otrzymaliśmy metodę Gaussa-Seidla zapisaną w sposób macierzowy:

$$x^{(k+1)} = B_{GS}x^{(k)} + c_{GS}, \quad k = 0, 1, \ldots,$$

gdzie

$$B_{GS} = -(L + D)^{-1}U \quad \text{a} \quad c_{GS} = (L + D)^{-1}b.$$

Uwaga 5.11. *Podobnie jak w metodzie Jacobiego, w przypadku metody Gaussa-Seidla macierz iteracji także zależy wyłącznie od macierzy układu równań (nie zależy od wektora b). Tak samo jest też w przypadku innych metod postaci* (5.8)*, tj. np. metody SOR (patrz niżej), Richardsona (ta nie będzie omówiona, ale zainteresowani mogą zajrzeć do [1], [3] i innych źródeł).*

Przykład 5.6.

Zbadamy zbieżność metody Gaussa-Seidla zastosowanej do układu $Ax = b$, gdzie

$$A = \begin{pmatrix} 1 & 0 & 0 & -3 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -2 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 3 \\ -1 \\ -2 \\ 1 \end{pmatrix}.$$

a $b \in \mathbb{R}^4$.

Rozwiązanie:
Można by wyznaczyć macierz iteracji korzystając wprost ze wzoru macierzowego, tj. $B_{GS} = -(L + D)^{-1}U$. Byłoby to poprawne, ale ponieważ odwracanie i mnożenie macierzy wszyscy już trenowali na zajęciach z algebry liniowej, w tym przykładzie wyznaczymy B_{GS} inaczej.
Macierz układu zawiera dużo zer, co uprości obliczenia. Po zapisaniu układu $Ax = b$ w formie równań

$$\begin{aligned} x_1 - 3x_4 &= 3 \\ x_1 - x_2 &= -1 \\ x_3 + x_4 &= -2 \\ -2x_3 + 2x_4 &= 1, \end{aligned}$$

możemy utworzyć wzory iteracyjne Gaussa-Seidla:

$$\begin{aligned} x_1^{(k+1)} &= 3x_4^{(k)} + 3 \\ x_2^{(k+1)} &= x_1^{(k+1)} + 1 \\ x_3^{(k+1)} &= -x_4^{(k)} - 2 \\ x_4^{(k+1)} &= x_3^{(k+1)} + \frac{1}{2} \end{aligned}$$

dla $k = 1, 2, \ldots$. Teraz musimy je doprowadzić do postaci, w której po prawej stronie równości mamy wyłącznie k-te przybliżenia. Wykonując odpowiednie podstawienia

otrzymamy:

$$\begin{aligned} x_1^{(k+1)} &= 3x_4^{(k)} + 3, \\ x_2^{(k+1)} &= 3x_4^{(k)} + 4, \\ x_3^{(k+1)} &= -x_4^{(k)} - 2, \\ x_4^{(k+1)} &= -x_4^{(k)} - \frac{3}{2}, \end{aligned}$$

skąd można już zapisać macierz iteracji

$$B_{GS} = \begin{pmatrix} 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

Jest to macierz trójkątna górna, zatem na jej spektrum składają się elementy diagonali. Stąd $\rho(B_{GS}) = 1$, co oznacza, że metoda Gaussa-Seidla nie jest zbieżna.

5.9.5 Metoda SOR

Nazwa SOR (ang. **S**uccessive **O**ver**R**elaxation), nazywana też metodą nadrelaksacji, jest uogólnieniem metody Gaussa-Seidla.

W metodzie tej występuje parametr $\omega \in \mathbb{R}$, zwany parametrem relaksacji.

Algorytm (metoda SOR)

$x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})^T$ – przybliżenie początkowe

for $k = 0, 1, \ldots,$ (dopóki nie będzie spełniony wybrany warunek stopu)
  for $i = 1, 2, \ldots, n$
    $x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega(b_i - \sum\limits_{j=1, j<i}^{n} a_{ij}x_j^{(k+1)} - \sum\limits_{j=1, j>i}^{n} a_{ij}x_j^{(k)})/a_{ii}$
  end
end

Zauważmy, że gdy $\omega = 1$, metoda SOR staje się metodą Gaussa-Seidla.
Metodę SOR można wyprowadzić zapisując macierz A jako

$$A = L + D + U$$

podobnie, jak miało to miejsce przy metodach Jacobiego i Gaussa-Seidla (oznaczenia są wyjaśnione w notatkach do Metod Numerycznych 1). Mnożąc powyższe równanie stronami przez $\omega \neq 0$, a następnie dodając stronami macierz D i odpowiednio grupując wyrazy, otrzymamy:

$$\omega A = (D + \omega L) - \big((1-\omega)D - \omega U\big).$$

Podstawiając powyższą zależność do równania $\omega Ax = \omega b$ (które jest równoważne równaniu $Ax = b$), otrzymamy:

$$(D + \omega L)x - \big((1-\omega)D - \omega U\big)x = \omega b,$$

a dalej

$$x = (D + \omega L)^{-1}\big((1-\omega)D - \omega U\big)x + \omega(D + \omega L)^{-1}b.$$

Wzór iteracyjny ma zatem postać:

$$x^{(k+1)} = B_{SOR}x^{(k)} + c_{SOR},$$

gdzie

$$B_{SOR} = (D + \omega L)^{-1}\big((1-\omega)D - \omega U\big)$$

oraz

$$c_{SOR} = \omega(D + \omega L)^{-1}b.$$

Twierdzenie 5.9. *(Kahan) Promień spektralny $\rho(B_{SOR})$ macierzy iteracji spełnia zależność: $\rho(B_{SOR}) \geq |\omega - 1|$.*

Wniosek 5.1. *Jeśli $\omega \notin (0, 2)$, to metoda SOR nie jest zbieżna.*

Twierdzenie 5.10. *Jeśli $A \in \mathbb{R}^{n\times n}$ jest macierzą symetryczną i dodatnio określoną, a $b \in \mathbb{R}^n$, to metoda SOR jest zbieżna globalnie wtedy i tylko wtedy, gdy $0 < \omega < 2$.*

6 Wyznaczanie miejsc zerowych funkcji jednej zmiennej

Załóżmy, że f jest funkcją zmiennej rzeczywistej, oraz, na razie ogólnie, że $f : \mathbb{R} \to \mathbb{R}$.

Zadaniem, którym się zajmiemy, będzie znalezienie takiej wartości $\alpha \in \mathbb{R}$, że

$$f(\alpha) = 0.$$

W praktyce rzadko mamy do czynienia z funkcjami, których miejsca zerowe da się znaleźć analitycznie używając skończonej liczby operacji arytmetycznych. Na przykład, wyznaczenie pierwiastków wielomianu drugiego stopnia jest łatwe, czego nie da się powiedzieć o wielomianach wyższych stopni. Co więcej, dla wielomianów stopnia 5 i większego, nie istnieją ogólne analityczne wzory określające ich pierwiastki na podstawie współczynników. Oczywiście, pierwiastki pewnych konkretnych wielomianów stopnia wyższego niż 4 da się znaleźć analitycznie (tak jest, na przykład w przypadku $f(x) = x^5 - x$), ale w ogólnym przypadku jest to niemożliwe w skończonej liczbie operacji arytmetycznych.

Można wyznaczyć pierwiastki w sposób przybliżony, tj. z pewną dokładnością, stosując *metody iteracyjne*. Polegają one na tym, że startując z danego przybliżenia początkowego $x_0 \in \mathbb{R}$ tworzymy ciąg kolejnych przybliżeń $\{x_k\}$, taki, że $x_k \to \alpha$ przy $k \to \infty$.

6.1 Metoda Newtona (stycznych)

Niech $f : \mathbb{R} \to \mathbb{R}$ będzie funkcją klasy C^2 oraz niech $x_0 \in \mathbb{R}$ będzie danym przybliżeniem początkowym.

Zapiszmy rozwinięcie funkcji f w szereg Taylora w otoczeniu x_k:

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(\xi)(x - x_k)^2,$$

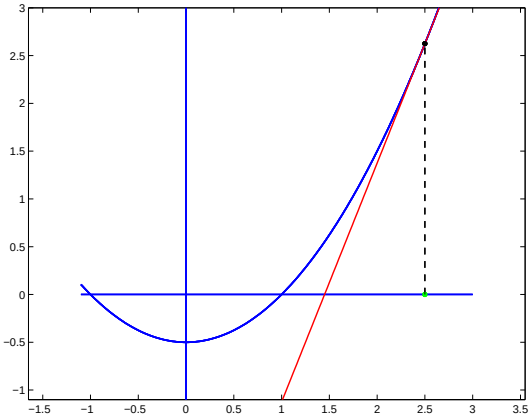
przy czym $\xi \in (x, x_k)$, jeśli $x < x_k$, lub $\xi \in (x_k, x)$, jeśli $x_k < x$.
Jeśli pominiemy resztę tego szeregu (czyli ostatni wyraz w powyższym wzorze), otrzymamy przybliżenie funkcji f wielomianem stopnia 1. Oznaczmy go przez p. Mamy:

$$p(x) = f(x_k) + f'(x_k)(x - x_k).$$

Jest to wielomian interpolacyjny Hermite'a skonstruowany na jednym węźle x_k i jednocześnie jest to styczna do wykresu funkcji f poprowadzona w punkcie $(x_k, f(x_k))$ (czerwona linia na Rysunku (8)).

W metodzie Newtona, kolejne przybliżenie, czyli x_{k+1}, jest miejscem zerowym stycznej p. Wyznaczmy je. Skoro x_{k+1} jest miejscem zerowym p, to $p(x_{k+1}) = 0$. Mamy zatem:

$$0 = p(x_{k+1}) = f(x_k) + f'(x_k)(x_{k+1} - x_k).$$



Rysunek 8: Graficzna interpretacja metody Newtona: niebieska zakrzywiona linia to wykres funkcji f, zielony punkt to x_k, czarny punkt to $(x_k, f(x_k))$ (punkt, w którym prowadzona jest styczna), czerwona linia to wykres stycznej.

Po przeniesieniu $f(x_k)$ na lewą stronę i podzieleniu stronami przez $f'(x_k)$ (zakładamy chwilowo, że $f'(x_k) \neq 0$), otrzymujemy:

$$-\frac{f(x_k)}{f'(x_k)} = x_{k+1} - x_k,$$

skąd wyznaczymy

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{6.1}$$

Korzystając z tego wzoru dla $k = 0, 1, \ldots$ utworzymy ciąg punktów, który ma szansę być zbieżnym do miejsca zerowego funkcji f.
Metoda Newtona nie zawsze jest zbieżna – to zależy zarówno od funkcji f, jak i przybliżenia początkowego. Może się zdarzyć, że dla pewnego k, $f'(x_k) = 0$. Dzieje się tak, gdy styczna w k-tym kroku jest równoległa do osi OX. Może się także zdarzyć, że $f'(x_k) \neq 0$ dla wszystkich k (we wzorze (6.1) nie wystąpi dzielenie przez zero), ale wygenerowany ciąg albo nie ma granicy albo jest rozbieżny do nieskończoności.

Twierdzenie 6.1. *Jeżeli istnieje przedział $[a, b] \subset \mathbb{R}$, taki, że*

1) f jest funkcją klasy $C^2([a, b])$,

2) $f(a)\, f(b) < 0$,

3) f' i f'' nie zmieniają znaku w $[a, b]$,

4) $x_0 \in [a, b]$ jest przybliżeniem początkowym takim, że $f(x_0)\, f''(x_0) > 0$,

to metoda Newtona jest zbieżna do miejsca zerowego funkcji f.

Uwaga 6.1. *Twierdzenie 6.1 to warunek dostateczny, implikacja zachodzi tylko w jedną stronę (tj. jeśli pewne warunki sa spełnione, to mamy zbieżność). Z twierdzenia tego nie wynika, co się dzieje, jeśli któryś z tych warunków nie jest spełniony! W szczególności, nieprawdą jest, że jeśli któryś warunek nie jest spełniony, to nie ma zbieżności.*

Warunek 2) twierdzenia 6.1 zapewnia, że w przedziale $[a, b]$ znajduje się co najmniej jedno miejsce zerowe funkcji f.

Możliwe warunki kończenia obliczeń (warunki stopu):

- $|x_{k+1} - x_k| < d_1$,
- $|x_{k+1} - x_k| < d_1|x_k|$,
- (warunek Gilla) $|x_{k+1} - x_k| < d_1|x_k| + d_2$,
- $|f(x_k)| < d_1$ (ten warunek, mimo, że wydaje się naturalny, należy stosować ostrożnie; jeśli wartość funkcji w jakimś punkcie jest bliska zeru, nie musi to oznaczać, że funkcja ta ma miejsce zerowe blisko tego punktu),

gdzie d_1 i d_2 to parametry określające dokładność (liczby dodatnie, bliskie zeru), przykładowo w warunku Gilla można przyjąć $d_1 = 10^{-10}$ i $d_2 = 10^{-20}$ lub $d_1 = 10^{-13}$ i $d_2 = 10^{-25}$.

Przykład 6.1. *Zastosowanie metody Newtona*

Jak obliczyć $\sqrt{a}$, gdzie $a > 0$
(szybko, z dużą dokładnością i bez korzystania z pierwiastków)?

Zastosujemy metodę Newtona dla funkcji, której pierwiastkiem jest $\sqrt{a}$, a mianowicie:

$$f(x) = x^2 - a.$$

Mamy

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - a}{2x_k} = \frac{x_k^2 + a}{2x_k}.$$

Po uproszczeniu otrzymujemy wzór iteracyjny:

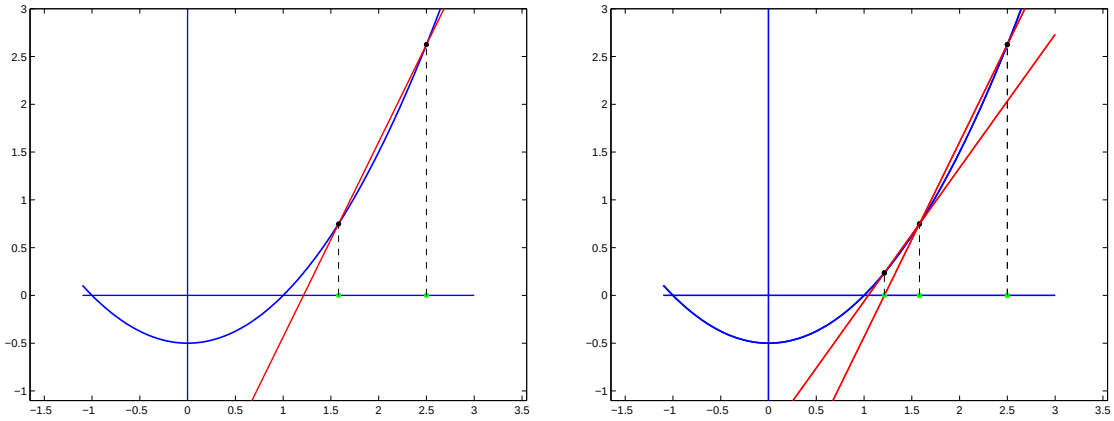
$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{a}{x_k}\right) \quad \text{dla} \quad k = 0, 1, \ldots.$$

Metoda nie zadziała, gdy przybliżeniem początkowym jest $x_0 = 0$. Dla innych przybliżeń początkowych, wygenerowany ciąg kolejnych przybliżeń zbiega do $\sqrt{a}$ lub $-\sqrt{a}$.
Dla jakich x_0 metoda jest zbieżna do $\sqrt{a}$?

6.2 Metoda siecznych

Niech $f : \mathbb{R} \to \mathbb{R}$ oraz niech $x_0, x_1 \in \mathbb{R}$ będą danymi przybliżeniami początkowymi, takimi, że $f(x_0) \neq f(x_1)$.

Idea metody siecznych (k-ty krok, $k = 1, 2, \ldots$): przez punkty $(x_{k-1}, f(x_{k-1}))$ oraz $(x_k, f(x_k))$ prowadzimy sieczną. Kolejne przybliżenie x_{k+1} jest miejscem zerowym tej siecznej.
Sieczna ta jest wielomianem interpolacyjnym (Lagrange'a) opartym na węzłach x_{k-1} i x_k.



Rysunek 9: Graficzne przedstawienie metody siecznych. Po lewej: pierwszy krok – niebieska zakrzywiona linia to wykres funkcji f, zielone punkty to x_{k-1} i x_k, czarne punkty to $(x_{k-1}, f(x_{k-1}))$ i $(x_k, f(x_k))$ (punkty, w których prowadzona jest sieczna), czerwona linia to wykres siecznej. Po prawej: pierwszy i drugi krok.

Wzór iteracyjny metody siecznych można wyprowadzić na kilka sposobów, na przykład, można we wzorze metody Newtona zastąpić $f'(x_k)$ ilorazem różnicowym $\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$. Dostajemy

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \quad \text{dla} \quad k = 1, 2, \ldots.$$

Brak konieczności obliczania pochodnych zmniejsza koszt obliczeniowy każdego kroku (liczbę operacji arytmetycznych wymaganych w każdym kroku). Zwiększa się natomiast liczba iteracji wymaganych do osiągnięcia określonej dokładności. W tym sensie, w ogólnym przypadku metoda siecznych jest wolniejsza niż metoda Newtona – wymaga większej liczby kroków do osiągnięcia tej samej dokładności. Nie musi to oznaczać, że czas wykonania metody Newtona jest mniejszy.

6.3 Metoda Halleya

Niech $f : \mathbb{R} \to \mathbb{R}$ będzie klasy $C^2(\mathbb{R})$ oraz niech $x_0 \in \mathbb{R}$ będzie danym przybliżeniem początkowym.

Wykorzystując rozwinięcie funkcji $f(x)$ w szereg Taylora w otoczeniu punktu x_k otrzymujemy przybliżenie p funkcji f:

$$p(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2.$$

Następnie przyjmujemy:

$$0 = f(x_k) + f'(x_k)(x_{k+1} - x_k) + \frac{1}{2}f''(x_k)(x_{k+1} - x_k)^2.$$

Po przekształceniu:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k) + \frac{1}{2}f''(x_k)(x_{k+1} - x_k)}.$$

Wartość x_{k+1} po prawej stronie przybliżamy metodą Newtona (tj. podstawiamy $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$) otrzymując wzór określający metodę Halleya:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k) - f''(x_k)\frac{f(x_k)}{2f'(x_k)}} \quad \text{dla} \quad k = 0, 1, \ldots.$$

Idea metody Halleya: kolejne przybliżenie x_{k+1} jest miejscem zerowym hiperboli przybliżającej funkcję $f(x)$ w punkcie x_k.

6.4 Metoda parabol

Niech, podobnie, jak w przypadku metody Halleya, $f : \mathbb{R} \to \mathbb{R}$ będzie klasy $C^2(\mathbb{R})$ oraz niech $x_0 \in \mathbb{R}$ będzie danym przybliżeniem początkowym.

Podobnie, jak przy wyprowadzeniu metody Halleya, zapiszmy wielomian stopnia drugiego p interpolujący f, oparty na punkcie x_k:

$$p(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2.$$

W metodzie parabol kolejne przybliżenie x_{k+1} jest miejscem zerowym wielomianu p. Zatem x_{k+1} spełnia równanie

$$0 = f(x_k) + f'(x_k)(x_{k+1} - x_k) + \frac{1}{2}f''(x_k)(x_{k+1} - x_k)^2. \tag{6.2}$$

Jest to równanie kwadratowe, zatem posiada 2 rozwiązania (rzeczywiste lub zespolone oraz licząc z krotnościami). Powstaje pytanie: jeśli rozwiązania są różne, to które z nich wybrać?
Zauważmy, że jeśli metoda jest zbieżna, różnica $x_{k+1} - x_k$ staje się coraz mniejsza (dla dużych wartości k). Zatem wybieramy to rozwiązanie równania (6.2), które jest bliższe x_k.

W praktyce, wartości x_{k+1} nie obliczamy wprost z równania (6.2), tylko wprowadzamy pomocniczą zmienną $y = x_{k+1} - x_k$ (wyjaśnienie w uwadze poniżej).

Algorytm:

- rozwiązujemy równanie

$$0 = f(x_k) + f'(x_k)y + \frac{1}{2}f''(x_k)y^2.$$

względem y (gdzie $y = x_{k+1} - x_k$), otrzymując wartości $y^{(1)}$ oraz $y^{(2)}$ (może się zdarzyć, że $y^{(1)} = y^{(2)}$, ale to w niczym nie przeszkadza);

- jako y przyjmujemy tę z wartości $y^{(1)}$ i $y^{(2)}$, która położona jest bliżej x_k, czyli $y := \text{argmin}\{|y^{(1)}|, |y^{(2)}|\}$;

- obliczamy x_{k+1}:

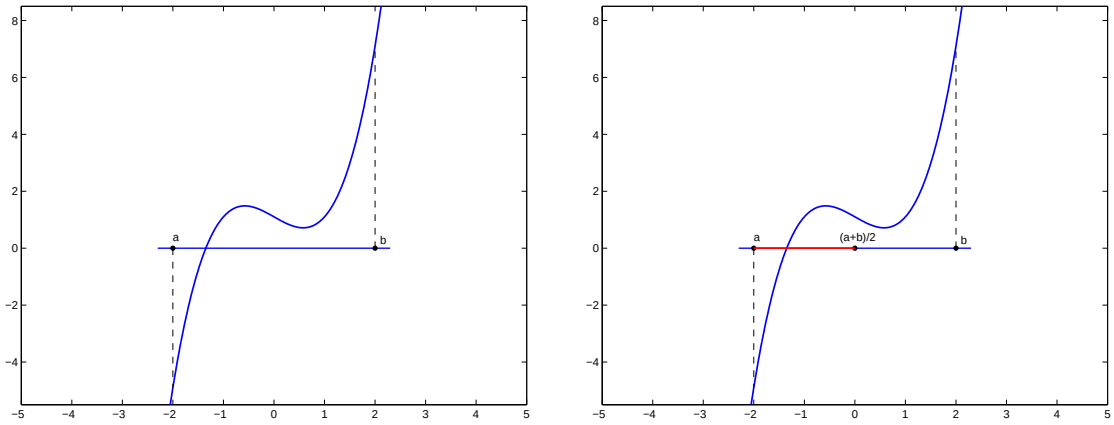
$$x_{k+1} = x_k + y.$$

Uwaga. Powyższe podejście (tj. wykorzystujące podstawienie $y = x_{k+1} - x_k$) ma lepsze własności numeryczne niż podejście polegające na obliczeniu x_{k+1} wprost z równania (6.2). Jest tak, ponieważ $y \to 0$ przy $k \to 0$, a x_{k+1} jest obliczane jako poprawka x_k (tj. $x_{k+1} = x_k + y$), a nie obliczane od początku. W sytuacji gdy $y \ll x_k$, może to mieć istotne znaczenie.

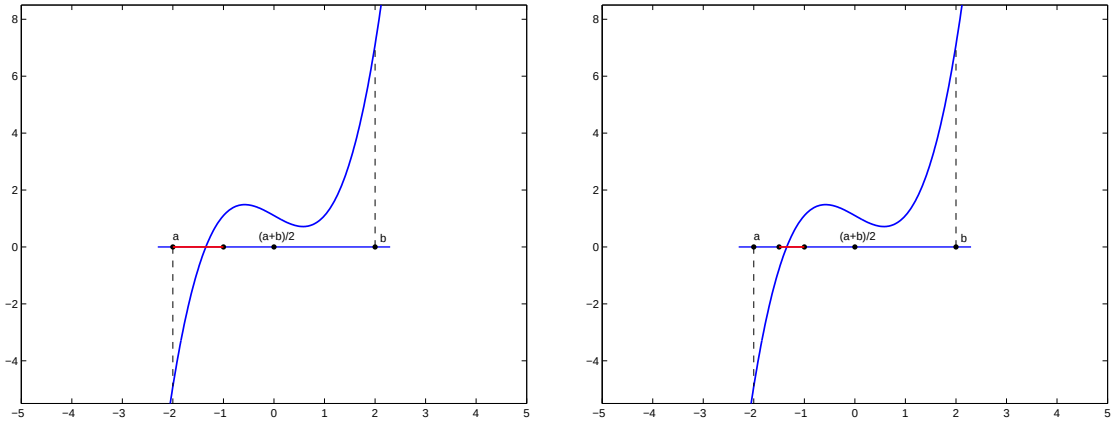
6.5 Metoda bisekcji

Twierdzenie 6.2. *(Darboux) Niech $[a, b] \subset \mathbb{R}$ oraz niech $f : [a, b] \to \mathbb{R}$ będzie funkcją ciągłą taką, że $f(a)f(b) < 0$. Wtedy istnieje co najmniej jedno α, $(a < \alpha < b)$ takie, że $f(\alpha) = 0$.*

W metodzie bisekcji wykorzystywane jest powyższe twierdzenie. Aby można było zastosować tę metodę, funkcja f, której miejsc zerowych szukamy, musi być ciągła na pewnym przedziale $[a, b] \subset \mathbb{R}$, oraz przedział ten musi być taki, że $f(a)f(b) < 0$. To gwarantuje, że w tym przedziale znajduje się co najmniej jedno miejsce zerowe funkcji f. W kolejnych krokach metody bisekcji przedział zawierający miejsce zerowe jest zawężany, aż do osiągnięcia żądanej dokładności.
Rysunek 10 (po lewej) pokazuje przykładową funkcję i przedział.
W pierwszym kroku metody dzielimy przedział $[a, b]$ na dwa podprzedziały równej długości $[a, \frac{a+b}{2}]$ i $[\frac{a+b}{2}, b]$, sprawdzamy, który z nich zawiera miejsce zerowe funkcji f. Robimy to, sprawdzając, w którym z tych podprzedziałów funkcja f zmienia znak. Następnie zastępujemy przedział $[a, b]$ tym z podprzedziałów, w którym nastąpiła zmiana znaku (Rysunek 10, po prawej).
W kolejnych krokach powtarzamy to postępowanie. Aktualny przedział jest dzielony na dwa równe podprzedziały, i ustala się, który z nich zawiera miejsce zerowe funkcji f. Rysunek 11 przedstawia 2 kolejne kroki metody.



Rysunek 10: Graficzne przedstawienie pierwszego kroku metody bisekcji. Po lewej: funkcja f oraz wyjściowy przedział, po prawej: zakres, w którym znajduje się miejsce zerowe, zostaje zawężony – przedział zaznaczony na czerwono będzie przekazany do następnego kroku.



Rysunek 11: Graficzne przedstawienie drugiego i trzeciego kroku metody bisekcji.

Może się zdarzyć, że w pewnym kroku obliczony punkt środkowy jest miejscem zerowym (wartość funkcji jest w tym punkcie równa 0). Wtedy obliczenia można skończyć. Jest to jednak raczej rzadka sytuacja. Zwykle obliczenia są kończone, gdy długość przedziału jest już dostatecznie mała. Wtedy przyjmuje się, że jego środek jest szukanym miejscem zerowym. Błąd takiego przybliżenia jest nie większy niż połowa długości ostatniego wyznaczonego przedziału.
Ponieważ długość przedziału jest w każdym kroku zmniejszana dwukrotnie, można w przybliżeniu przyjąć, że

$$e_{k+1} \approx \frac{1}{2}\, e_k,$$

gdzie $e_k = x_k - \alpha$ jest błędem w k-tym kroku, $k = 1, 2, \ldots$ (α jest szukanym miejscem zerowym, a x_k jest, w przypadku metody bisekcji, środkiem przedziału w k-tym kroku). Zatem (porównaj Uwaga 6.6), wykładnik zbieżności tej metody jest równy 1. Zbieżność jest więc wolna.

6.6 Szybkość zbieżności metod iteracyjnych

Definicja 6.1. *Liczbę α nazywamy m-krotnym pierwiastkiem równania $f(x) = 0$, gdy*

$$f(x) = (x - \alpha)^m g(x),$$

przy czym $g(\alpha) \neq 0$.

Jeżeli $m = 1$, to mówimy, że α jest pierwiastkiem pojedynczym. Jeżeli $m > 1$, to α jest pierwiastkiem wielokrotnym.

Niech x_k będzie ciągiem kolejnych przybliżeń wygenerowanym pewną metodą iteracyjną (Newtona lub inną). Załóżmy, że metoda ta jest zbieżna, czyli, że ciąg $x_k \to \alpha$ przy $k \to \infty$.

Powstaje pytanie: jak szybko ten ciąg dąży do α? Jak zbadać szybkość zbieżności?

Miarą szybkości zbieżności metod iteracyjnych jest wielkość zwana wykładnikiem zbieżności.

Definicja 6.2. *Przyjmijmy, że $e_k = x_k - \alpha$ jest błędem w k-tym kroku. Jeśli istnieją takie liczby p i C ($C > 0$), że*

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|^p} = C$$

to p nazywa się wykładnikiem zbieżności metody iteracyjnej.

Uwaga 6.2. *Im większa wartość p, tym szybciej metoda jest zbieżna.*

Dlaczego?

Uwaga 6.3. *Dla dostatecznie dużych wartości k, można przyjąć, że $|e_{k+1}| \approx C|e_k|^p$.*

6.6.1 Wartości wykładnika zbieżności

- Metoda Newtona: $p = 2$ (zbieżność kwadratowa)
- Metoda siecznych: $p = \frac{1+\sqrt{5}}{2} \approx 1.61$ (zbieżność ponadliniowa)
- Metoda parabol: $p = 3$ (zbieżność sześcienna)
- Metoda Halleya: $p = 3$ (zbieżność sześcienna)

  Uwaga!!! Powyższe wartości są prawdziwe jedynie dla zer jednokrotnych! W przypadku miejsc zerowych *wielokrotnych* wyżej wymienione metody są *wolniej* zbieżne.

- Metoda bisekcji: $p = 1$ (zbieżność liniowa)

Uwaga. Wykładnik zbieżności metody bisekcji nie zależy od krotności zer. Dlaczego?

Dodatkowa lektura: podrozdział 3.2 i 3.3 z pozycji [1].

Literatura

[1] D.Kincaid, W.Cheney: Analiza numeryczna, WNT 2005;

[2] J. i M. Jankowscy (M.Dryja): Przegląd metod i algorytmów numerycznych cz. 1 i 2, WNT, Warszawa 1988;

[3] Z.Fortuna, B.Macukow, J.Wąsowski: Metody numeryczne, WNT, Warszawa 2006;

[4] A.Kiełbasiński, H.Schwetlick: Numeryczna algebra liniowa, WNT, Warszawa 1994;

[5] G.Dahlquist, A.Björck: Metody numeryczne, PWN, Warszawa 1987;

[6] J.Stoer, R.Bulirsch: Wstęp do analizy numerycznej, PWN, Warszawa 1987;

[7] Praca zbiorowa pod red. J.Wąsowskiego: Ćwiczenia laboratoryjne z metod numerycznych, OWPW, Warszawa 2002,

[8] Y.Saad: Iterative Methods for Sparse Linear Systems, SIAM, 1996.

[9] M.Bollhöfer, V.Mehrmann: Numerische Mathematik, Eine projektorientierte Einführung für Ingenieure, Mathematiker und Naturwissenschaftler, Vieweg Verlag, Brunszwik, 2004.