Analiza numeryczna w pigułce

Definicje, wzory, twierdzenia
Podsumowanie informacji z semestru zimowego 2019/2020

Tomasz Woszczyński

Spis treści

1 Teoria błędu **3**
1.1 Błąd względny i bezwzględny 3
1.2 Reprezentacja liczb 3
1.3 Rodzaje zaokrągleń 3
1.4 Zbiór liczb zmiennopozycyjnych, nadmiar i niedomiar 3
1.5 Działania arytmetyczne w świecie liczb zmiennopozycyjnych . 4
1.6 Twierdzenie o kumulacji błędów 4
1.7 Zjawisko utraty cyfr znaczących 4

2 Uwarunkowanie zadania, numeryczna poprawność **5**
2.1 Zadanie źle uwarunkowane i dobrze uwarunkowane 5
2.2 Wskaźnik uwarunkowania zadania 5
2.3 Algorytm numerycznie poprawny 5

3 Rozwiązywanie równań nieliniowych **6**
3.1 Metoda bisekcji 6
3.2 Metoda Newtona (metoda stycznych) 7
3.3 Metoda siecznych 8
3.4 Wykładnik zbieżności ciągu (rząd metody) 9

4 Interpolacja wielomianowa **10**
4.1 Postaci wielomianów 10
4.1.1 Postać naturalna potęgowa 10
4.1.2 Postać Newtona 10
4.1.3 Postać Czebyszewa 10
4.2 Schemat Hornera 11
4.3 Uogólniony schemat Hornera 11
4.4 Algorytm Clenshawa 12
4.5 Interpolacja wielomianowa Lagrange'a 12
4.6 Obliczanie wartości L_n dla danego x 12
4.7 Ilorazy różnicowe 13
4.8 Wzór na błąd interpolacji 13
4.9 Naturalna interpolacyjna funkcja sklejana 3. stopnia 13
4.9.1 Definicja 13
4.9.2 I sposób konstrukcji 14
4.9.3 II sposób konstrukcji 14
4.10 Krzywe parametryczne na płaszczyźnie 15
4.11 Wielomian Bernsteina 15
4.12 Krzywa Beziera 16
4.13 Wyznaczanie punktu $P_n(t)$ - algorytm de Casteljau 16
4.14 Kombinacja barycentryczna punktów 17

5 Aproksymacja średniokwadratowa na zbiorze dyskretnym 18
5.1 Norma średniokwadratowa na zbiorze dyskretnym 18
5.2 Znajdowanie elementu optymalnego funkcji f 18
5.3 Wielomianowa aproksymacja średniokwadratowa na zbiorze dyskretnym 19
5.3.1 Dyskretny iloczyn skalarny na zbiorze $\mathcal{X}$ 19
5.3.2 Ortogonalność funkcji 19
5.3.3 Ortogonalizacja Grama-Schmidta 20
5.3.4 Ciąg wielomianów ortogonalnych 20
5.3.5 Rozwiązanie zadania 21

6 Kwadratury - całkowanie numeryczne 22
6.1 Funkcja podcałkowa, funkcja pierwotna 22
6.2 Metody obliczania całek 22
6.2.1 Całkowanie przez części 22
6.2.2 Całkowanie przez podstawianie 22
6.3 Kwadratura liniowa 22
6.3.1 Błąd kwadratury liniowej 23
6.3.2 Rząd kwadratury liniowej 23
6.3.3 Strategia konstrukcji kwadratur 23
6.4 Kwadratura interpolacyjna 23
6.5 Kwadratura Newtona-Cotesa 24
6.5.1 Współczynniki kwadratury Newtona-Cotesa 24
6.5.2 Błąd kwadratury Newtona-Cotesa 24
6.5.3 Rząd kwadratury Newtona-Cotesa 24
6.6 Wzór trapezów, złożony wzór trapezów 25
6.7 Wzór Simpsona, złożony wzór Simpsona 26
6.8 Metoda Romberga 27
6.8.1 Tablica Romberga 27
6.8.2 Efektywne obliczanie wartości w pierwszej kolumnie . 27
6.9 Kwadratura Gaussa 28

7 Algorytmy numeryczne algebry liniowej 29
7.1 Krótkie przypomnienie z algebry (macierze) 29
7.1.1 Podstawowe działania na macierzach 29
7.1.2 Wyznacznik macierzy kwadratowej 29
7.1.3 Odwracanie macierzy kwadratowej 30
7.1.4 Układ równań liniowych 30
7.1.5 Wzory Cramera 30
7.1.6 Przykład rozwiązania układu równań liniowych 31
7.2 Rozwiązywanie układów równań o macierzy trójkątnej dolnej 31
7.3 Metoda faktoryzacji rozwiązywania układów równań liniowych 31
7.4 Rozkład trójkątny macierzy ≡ rozkład LU 32

1 Teoria błędu

1.1 Błąd względny i bezwzględny

Niech x będzie dokładną wartością, a $\bar{x}$ wartością przybliżoną. Błąd bezwzględny ma wartość $|x - \bar{x}|$, a błąd względny $\left|\frac{x-\bar{x}}{x}\right|$.

1.2 Reprezentacja liczb

(a) Całkowite: $l \in \mathbb{Z} \Rightarrow l = \pm \sum_{i=0}^{n} e_i 2^i$, gdzie $e_i \in \{0, 1\}, e_n = 1$.

(b) Rzeczywiste: $x \in \mathbb{R} \Rightarrow x = s \cdot m \cdot 2^c$, gdzie $s \in \{1, -1\}$ (znak), $m \in \left[\frac{1}{2}, 1\right)$ (mantysa), $c \in \mathbb{Z}$ (cecha).

1.3 Rodzaje zaokrągleń

Rozwinięcie dwójkowe liczb rzeczywistych jest zwykle nieskończone, dlatego używamy dwóch typów zaokrągleń:

(a) Obcięcie: $m \approx m_t^c := \sum_{i=1}^{t} e_{-i} \cdot 2^{-i}$, tzn. ignorujemy cyfry od $t + 1$.

(b) Zaokrąglenie symetryczne: $m \approx m_t^r := \sum_{i=1}^{t} e_{-i} \cdot 2^{-i} + e_{-(t+1)} \cdot 2^{-t}$.

Reprezentację zmiennopozycyjną liczby $x = s \cdot m \cdot 2^c \in \mathbb{R}$ nazywamy liczbę:

- $chop(x) := s \cdot m_t^c \cdot 2^c$ (obcięcie - chopping)
- $rd(x) := s \cdot m_t^r \cdot 2^c$ (zaokrąglenie symetryczne - symmetric rounding)

Twierdzenie:

$$\left|\frac{x - chop(x)}{x}\right| \leqslant 2 \cdot 2^{-t}$$

$$\left|\frac{x - rd(x)}{x}\right| \leqslant 2^{-t}$$

1.4 Zbiór liczb zmiennopozycyjnych, nadmiar i niedomiar

Weźmy $D = 2^{c_{max}}$ i zdefiniujmy zbiór

$$\mathcal{X} := \left(-D, -\frac{1}{2D}\right] \cup \{0\} \cup \left[\frac{1}{2D}, D\right)$$

Nadmiarem są wtedy wszystkie liczby ze zbioru $(-\infty, -D] \cup [D, \infty)$, a niedomiarem liczby ze zbioru $\left(-\frac{1}{2D}, \frac{1}{2D}\right) \setminus \{0\}$ - nie da się ich przedstawić w reprezentacji zmiennopozycyjnej wcześniej przez nas zdefiniowanej. Wtedy zbiorem liczb zmiennopozycyjnych nazwiemy zbiór

$$\mathcal{X}_{fl} := rd\,(\mathcal{X})$$

1.5 Działania arytmetyczne w świecie liczb zmiennopozycyjnych

Weźmy $x, y \in \mathcal{X}_{fl}$ i określmy działanie $\circ \in \{+, -, \cdot, /\}$. Dla $x \circ y \in \mathcal{X}'$, czyli wyniku działania bez nadmiaru zachodzi własność

$$fl\,(x \circ y) := (x \circ y)\,(1 + \mathcal{E}_{x,y,\circ})$$

gdzie $\mathcal{E}_{x,y,\circ} \leqslant 2^{-t}$, a t to liczba bitów na zapamiętanie mantysy.
Błąd względny operacji $\circ$ wynosi co najwyżej 2^{-t}.

1.6 Twierdzenie o kumulacji błędów

Jeśli $|\delta_i| \leqslant 2^{-t}$, $(i = 1, 2, \ldots, n)$, to zachodzi równość

$$\prod_{i=1}^{n} (1 + \delta_i) = 1 + \sigma_n$$

dla $\sigma_n = \sum\limits_{i=1}^{n} \delta_i + O(2^{-2t})$. Jeśli $n \cdot 2^{-t} < 2$, to $|\sigma_n| \lesssim n \cdot 2^{-t}$.

1.7 Zjawisko utraty cyfr znaczących

Operacja + liczb tych samych znaków oraz odejmowanie liczb przeciwnych znaków oraz $\cdot$ i / są uznawane za "bezpieczne". Kłopot występuje wtedy, gdy odejmujemy od siebie liczby tych samych znaków. Następuje to w przypadku liczb bardzo bliskich sobie, ponieważ znormalizowanie liczby "obcina" początkowe zera w liczbie $x - y$, jednak wtedy ostatnie niezerowe cyfry są przesuwane na pierwsze pozycje i jeżeli $i < t$, to nie będzie wiadomo czym zapełnić pozostałe miejsca w mantysie.

2 Uwarunkowanie zadania, numeryczna poprawność

2.1 Zadanie źle uwarunkowane i dobrze uwarunkowane

Zadania, w których mała względna zmiana danych powoduje dużą względną zmianę wyniku nazywamy zadaniami źle uwarunkowanymi. Wielkości mówiące o tym w jaki sposób względna zmiana danych wpływa na względną wartość wyniku nazywamy wskaźnikami uwarunkowania. Zadania, które nie są źle uwarunkowane nazywamy zadaniami dobrze uwarunkowanymi.

2.2 Wskaźnik uwarunkowania zadania

Wzorem ogólnym jest

$$Cond(x) := \frac{\text{względna zmiana wyniku}}{\text{względna zmiana danych}}$$

Wzór ten można łatwo sprawdzić na przykładzie, wyprowadzając wskaźnik uwarunkowania zadania obliczania wartości funkcji f w punkcie x. Względna zmiana danych wyrażana jest przez $\left|\frac{(x+\delta)-x}{x}\right| = \left|\frac{\delta}{x}\right|$, a względna zmiana wyniku przez $\left|\frac{f(x+\delta)-f(x)}{f(x)}\right|$. Po podstawieniu tych wartości do powyższego wzoru (i zauważenie, że po rozpisaniu jeden wyraz zamienia się na pochodną funkcji f w punkcie x) otrzymujemy wzór na wskaźnik uwarunkowania zadania:

$$Cond(x) = \left|\frac{x \cdot f'(x)}{f(x)}\right|$$

2.3 Algorytm numerycznie poprawny

Algorytm nazywamy numerycznie poprawnym, jeśli wynik jego działania w świecie liczb zmiennopozycyjnych może być zinterpretowany jako mało zaburzony wynik dokładny dla mało zaburzonych danych. Chcąc wyznaczyć wartość $A(a)$ w świecie liczb zmiennopozycyjnych otrzymamy

$$fl(A(a)) = (A(a \cdot (1+\beta)) \cdot (1+\alpha)$$

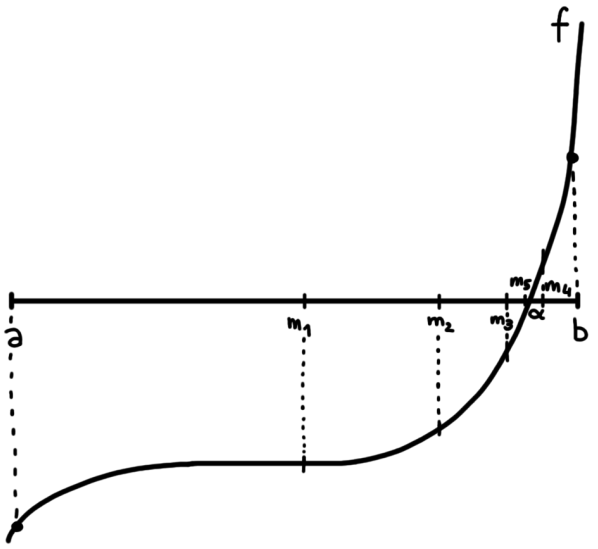
gdzie $(1+\beta)$ oznacza małe zaburzenie danych, a $(1+\alpha)$ małe zaburzenie wyniku.

3 Rozwiązywanie równań nieliniowych

Naszym zadaniem jest znalezienie miejsca zerowego funkcji $f \in C[a,b]$ spełniającej warunki $f(a)f(b)<0$ oraz $!\exists \alpha \in [a,b] \quad f(\alpha) = 0$.

3.1 Metoda bisekcji

Metoda ta polega na rekurencyjnym wyznaczaniu środka przedziału, a następnie sprawdzaniu czy miejsce zerowe znajduje się po lewej czy prawej stronie i przejściu na odpowiednią z nich. Ten rekurencyjny krok powtarzamy do momentu osiągnięcia odpowiedniego przybliżenia lub dokładnie miejsca zerowego.



Rysunek 1: Przykład działania metody bisekcji

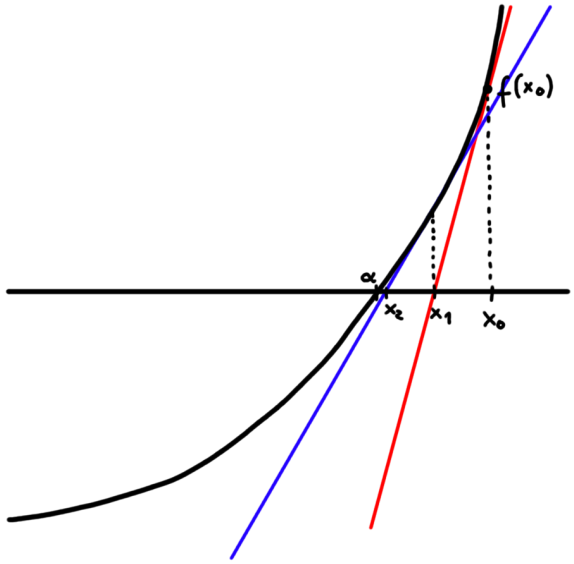
Niech $I_k := [a_k, b_k]$ dla $k = 0, 1, \ldots$ takie, że $I_0 > I_1 > \ldots > I_k > \ldots$ i $\alpha \in I_k$. Wyznaczmy środek k-tego przedziału jako $m_{k+1} = \frac{a_k + b_k}{2}$. Jeśli $f(m_{k+1}) = 0$, to $\alpha = m_{k+1}$, w przeciwnym wypadku naszym nowym przedziałem staje się $[a_k, m_{k+1}]$, gdy $f(m_{k+1})>0$ lub $[m_{k+1}, b_k]$, gdy $f(m_{k+1})<0$.
Metodę tę cechują:

- długość k-tego przedziału wynosi $|I_k| = \frac{b_0 - a_0}{2^k}$ dla $k \in \mathbb{N}$.,
- $|\alpha - m_{k+1}| \leqslant \frac{b_0 - a_0}{2^{k+1}}$
- wyznaczenie przybliżenia m_{k+1} z błędem bezwzględnym na poziomie danego $\mathcal{E}$ przebiega w $\left\lceil \log_2 \left(\frac{b_0 - a_0}{2\mathcal{E}} \right) \right\rceil$ krokach

3.2 Metoda Newtona (metoda stycznych)

Metoda Newtona polega na wybraniu punktu x_0, a następnie rekurencyjne wyznaczanie kolejnych punktów kontrolnych

$$x_{n+1} := x_n - \frac{f(x_n)}{f'(x_n)} \text{ dla } (n = 0, 1, 2, \ldots)$$



Rysunek 2: Przykład działania metody Newtona (stycznych)

Metodę tę cechują:

- dobry wybór x_0 skutkuje szybką zbieżnością do miejsca zerowego α
- konieczność znajomości pochodnej f', co w przypadku złożonych funkcji jest problematyczne
- czułość na dobór x_0 (dla $f(x) = \arctan(x-1) - \frac{1}{x^2+1}$ i $x_0 = 2.3604$ po 13-tej iteracji osiągamy 18-cyfrową dokładność, lecz dla $x_0 = 2.3605$ ciąg x_n dąży do nieskończoności)
- możliwość zapętlenia się (gdy $x_n = x_{n+1}$)
- warunkiem stopu jest osiągnięcie $|f(x_n)| < \delta$ oraz

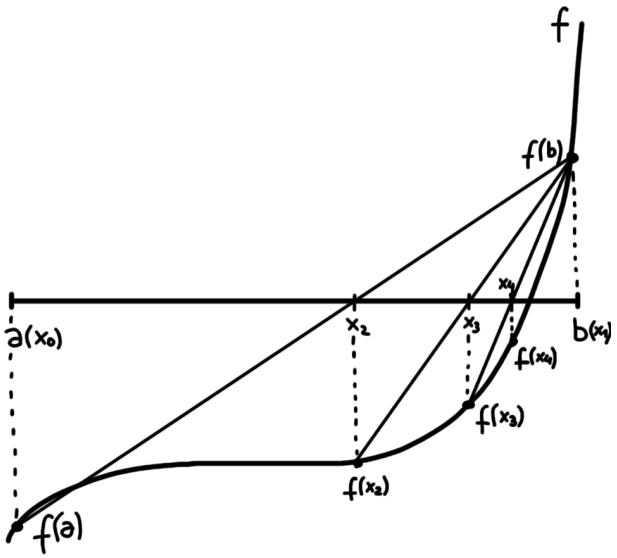
  $$\left|\frac{x_{n+1} - x_n}{x_n}\right|, \left|\frac{x_{n+2} - x_{n+1}}{x_{n+1}}\right|, \ldots, \left|\frac{x_{n+k} - x_{n+k-1}}{x_{n+k-1}}\right| < \mathcal{E} \text{ dla } k = 2, 3, \ldots$$

  lub $n \geqslant N_{max}$ (przekroczenie limitu wywołań)

3.3 Metoda siecznych

Metoda siecznych polega na wybraniu punktów początkowych x_0 oraz x_1, a następnie przeprowadzaniu między nimi siecznej. Następnie dla powstałego na przecięciu punktu powtarzamy operację, zmieniając punkty na których pracujemy. Wzór rekurencyjny metody jest następujący:

$$x_{n+1} := x_n - \frac{f(x_n)}{\frac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}}$$



Rysunek 3: Przykład działania metody siecznych

Metodę tę cechują:

- brak konieczności obliczania pochodnej
- wolniejszy czas działania niż przy metodzie Newtona
- punkty startowe x_0 oraz x_1 muszą zostać intuicyjnie wybrane
- możliwość zapętlenia się
- duża wrażliwość na dobór x_0 oraz x_1
- warunek stopu podobny jak w metodzie Newtona

3.4 Wykładnik zbieżności ciągu (rząd metody)

Niech dany będzie ciąg (x_n) taki, że $\lim_{n\to\infty} x_n = g$. Jeśli istnieje stałe p oraz $C \in \mathbb{R}_+$, dla których zachodzi własność

$$\lim_{n\to\infty} \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^p} = C$$

to p nazywamy wykładnikiem zbieżności ciągu, a C stałą asymptotyczną. Jeżeli $p = 1$, $C \in (0, 1)$, to mówimy o zbieżności liniowej, jeśli $p = 2$ to o zbieżności kwadratowej, a dla $p = 3$ o zbieżności sześciennej. Im większa wartość p, tym szybciej ciąg zbiega do granicy.

4 Interpolacja wielomianowa

Mówiąc o wielomianach będziemy używać oznaczenia Π_n na zbiór wielomianów stopnia równego lub mniejszego n (dla $n \in \mathbb{N}$). Stąd wynika, że $\Pi_n \setminus \Pi_{n-1}$ oznacza zbiór wielomianów stopnia dokładnie n. Aby uniknąć nieporozumień przyjmijmy, że $\Pi_{-1} = \emptyset$.

4.1 Postaci wielomianów

4.1.1 Postać naturalna potęgowa

$$w \in \Pi_n : \ w(x) = \sum_{k=0}^{n} a_k x^k$$

gdzie a_k to współczynniki postaci potęgowej wielomianu w.

4.1.2 Postać Newtona

$$w \in \Pi_n : \ w(x) = \sum_{k=0}^{n} b_k p_k(x)$$

gdzie $x_0, x_1, \ldots$ są danymi, a $p_k(x) = \prod_{j=0}^{k-1}(x - x_j)$, $p_0(x) \equiv 1$, b_k to współczynniki postaci Newtona wielomianu w.

4.1.3 Postać Czebyszewa

Wprowadźmy ciąg wielomianów $T_0, T_1, \ldots$ zdefiniowany rekurencyjnie w następujący sposób:

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

Powyższe wielomiany spełniają następujące własności:

- $T_k \in \Pi_k \setminus \Pi_{k-1}$
- T_{2n} jest funkcją parzystą, a T_{2n+1} nieparzystą
- każdy wielomian można zapisać jako kombinację liniową wielomianów Czebyszewa (baza Π_n)
- T_k ma dokładnie k miejsc zerowych należących do przedziału $(-1, 1)$
- dla $x \in [-1, 1]$ wartość $T_k(x) = \cos(k \cdot \arccos x)$

4.2 Schemat Hornera

Aby obliczyć wartość $w(x)$ wyrażonego w postaci potęgowej należy skorzystać ze schematu Hornera w celu przyspieszenia naszych działań.

$$\begin{aligned} w(x) &= a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 \\ &= x(a_n x^{n-1} + a_{n-1}x^{n-2} + \ldots + a_1) + a_0 = \ldots = \\ &= x(x(\ldots x(xa_n + a_{n-1}) + a_{n-2}) + a_{n-3}) + \ldots + a_1) + a_0 \end{aligned}$$

Dzięki temu możemy w łatwy sposób zdefiniować algorytm Hornera:

- $w_n := a_n$
- $w_k := w_{k+1}x + a_k$ dla $k = n-1, n-2, \ldots, 0$

Wtedy obliczoną wartością wielomianu w w punkcie x będzie w_0.
Algorytm ten działa w czasie $O(n)$ i jest algorytmem numerycznie poprawnym.

4.3 Uogólniony schemat Hornera

Aby obliczyć wartość $w(x)$ wyrażonego w postaci Newtona należy skorzystać z ulepszonego schematu Hornera w celu przyspieszenia naszych działań. Jest on bardzo podobny do zwyczajnego schematu Hornera, co można zobaczyć poniżej.

$$\begin{aligned} w(x) &= b_n p_n(x) + b_{n-1}p_{n-1}(x) + \ldots + b_1 p_1(x) + p_0(x) \\ &= b_n(x-x_0)(x-x_1)\ldots(x-x_{n-2})(x-x_{n-1}) + \\ &+ b_{n-1}(x-x_0)(x-x_1)\ldots(x-x_{n-2}) + \ldots + \\ &+ b_1(x-x_0) + \\ &+ b_0 \cdot 1 \\ &= (\ldots((b_n(x-x_{n-1}) + b_{n-1})(x-x_{n-2}) + b_{n-2})(x-x_{n-3}) + \ldots + b_1) \\ &\quad (x-x_0) + b_0 \end{aligned}$$

Algorytm uogólnionego schematu Hornera wykorzystuje dane $x, x_0, \ldots, x_{n-1}$, $b_0, \ldots, b_n$, więc zużywa więcej pamięci. Zdefiniowany jest w następujący sposób:

- $w_n := b_n$
- $w_k := w_{k+1}(x - x_k) + b_k$ dla $k = n-1, n-2, \ldots, 0$

Obliczony wynik będzie miał wartość w_0.
Uogólniony schemat Hornera jest również algorytmem numerycznie poprawnym i działa w czasie $O(n)$.

4.4 Algorytm Clenshawa

Wielomian w w postaci Czebyszewa można obliczyć za pomocą wzoru

$$w \in \Pi_n : w(x) = \frac{1}{2}c_0T_0(x) + c_1T_1(x) + \ldots + c_nT_n(x) = \sum_{k=0}^{n}{}' c_kT_k(x)$$

gdzie c_k to współczynniki postaci Czebyszewa wielomianu w, a danymi są $x, c_0, \ldots, c_n$. Ze względów numerycznych wartość wielomianu w tej postaci zaleca się obliczać przy pomocy następującego algorytmu:

- $B_{n+2} := 0,\ B_{n+1} := 0$
- $B_k := 2xB_{k+1} - B_{k+2} + c_k$ dla $k = n, n-1, \ldots, 0$

Wartością $w(x)$ będzie wtedy $\frac{B_0 - B_2}{2}$. Algorytm ten działa w czasie $O(n)$.

4.5 Interpolacja wielomianowa Lagrange'a

Dla danych parami różnych $x_0, \ldots, x_n \in \mathbb{R}$ i odpowiadających im wartościom $y_0, \ldots, y_n$ chcemy znaleźć taki wielomian $L_n \in \Pi_n$, że $L_n(x_k) = y_k$ dla $k = 0, 1, \ldots, n$. Możemy go wyrazić jako

$$L_n(x) = \sum_{k=0}^{n} y_k\lambda_k(x), \text{ gdzie } \lambda_k(x) := \prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{x - x_i}{x_k - x_i}$$

Twierdzenie: Zadanie interpolacyjne Lagrange'a ma zawsze jednoznaczne rozwiązanie.

4.6 Obliczanie wartości L_n dla danego x

Aby obliczyć wartość L_n wprost ze wzoru Lagrange'a musielibyśmy użyć algorytmu o złożoności czasowej $O(n^2)$, a dodanie kolejnej obserwacji sprawiłoby, że wszystkie obliczenia musielibyśmy liczyć od nowa. Możemy jednak zapisać taki wielomian w postaci Newtona

$$L_n(x) = \sum_{k=0}^{n} b_kp_k(x)$$

gdzie $p_0(x) \equiv 1$, $p_k(x) = \prod_{j=0}^{k-1}(x - x_j)$ dla $k \geqslant 1$ oraz $b_k = \sum_{i=0}^{k} \frac{f(x_i)}{\prod_{\substack{j=0 \\ j\neq i}}^{k}(x_i - x_j)}$.

Zakładając, że znamy współczynniki $b_0, \ldots, b_n$ (obliczenie w czasie $O(n^2)$) wartość $L_n(x)$ możemy obliczyć w czasie $O(n)$ za pomocą uogólnionego schematu Hornera. Dodanie kolejnej obserwacji zajmuje jedynie $O(n)$, gdyż

$$L_{n+1} = L_n + b_{n+1}(x - x_0)(x - x_1)\ldots(x - x_n)$$

4.7 Ilorazy różnicowe

Dla danych parami różnych $x_k, x_{k+1}, \ldots, x_l$ oraz funkcji f określonej w tych punktach wprowadzamy iloraz różnicowy $f[x_k, x_{k+1}, \ldots, x_l]$ w następujący sposób rekurencyjny:

- $f[x_k] := f(x_k)$
- $f[x_k, x_{k+1}, \ldots, x_{l-1}, x_l] := \frac{f[x_{k+1},\ldots,x_{l-1},x_l]-f[x_k,x_{k+1},\ldots,x_{l-1}]}{x_l-x_k}$

Za ich pomocą możemy wyrazić kolejne współczynniki $b_k \equiv f[x_0, x_1, \ldots, x_k]$ potrzebne do zdefiniowania wielomianu L_n w postaci Newtona. Tabela ilorazów różnicowych przedstawiona jest na poniższym schemacie:

$$
\begin{array}{lllll}
f[x_0] & & & & \\
f[x_1] & f[x_0,x_1] & & & \\
f[x_2] & f[x_1,x_2] & f[x_0,x_1,x_2] & & \\
f[x_3] & f[x_2,x_3] & f[x_1,x_2,x_3] & f[x_0,x_1,x_2,x_3] & \\
f[x_4] & f[x_3,x_4] & f[x_2,x_3,x_4] & f[x_1,x_2,x_3,x_4] & \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
f[x_n] & f[x_{n-1},x_n] & f[x_{n-2},x_{n-1},x_n] & \ldots & \ldots \quad f[x_0,\ldots,x_n]
\end{array}
$$

4.8 Wzór na błąd interpolacji

Niech $L_n \in \Pi_n$ spełnia warunki interpolacji Lagrange'a, wtedy wzorem na błąd interpolacji jest

$$f(x) - L_n(x) = \frac{f^{(n+1)}(\eta_x)}{(n+1)!}(x-x_0)(x-x_1)\ldots(x-x_n)$$

gdzie $\eta_x \in (a,b) \ni \{x_0, x_1, \ldots, x_n\}$ oraz $f \in C^{n+1}[a,b]$. Możemy stąd wywnioskować, że aby zminimalizować błąd musimy odpowiednio dobrać węzły. Rozwiązaniem jest (bez straty ogólności) wybranie węzłów Czebyszewa na przedziale $[-1,1]$, tzn. $x_k = \cos\left(\frac{2k+1}{2n+2}\pi\right)$ dla $k = 0, 1, \ldots, n$. W takim przypadku węzły rozmieszczone są gęsto w pobliżu krańców przedziału, a im bliżej środka, tym jest ich mniej.

4.9 Naturalna interpolacyjna funkcja sklejana 3. stopnia

4.9.1 Definicja

Konstrukcja NIFS3 opiera się na znajdowaniu prostych funkcji wielomianowych trzeciego stopnia interpolujących dane (x_k, y_k), $(0 \leqslant k \leqslant n)$. Dla danych $n \in \mathbb{N}$, wartości $x_0 < x_1 < \ldots < x_n$ oraz $y_0, y_1, \ldots, y_n \in \mathbb{R}$ wprowadzamy funkcję $s : [x_0, x_n] \to \mathbb{R}$ nazywaną NIFS3 spełniającą warunki:

- $s(x_k) = y_k$ dla $k = 0, 1, \ldots, n$

- funkcje $s,\ s',\ s''$ są ciągłe w $[x_0, x_n]$
- s w każdym przedziale $[x_{k-1}, x_k]$ jest wielomianem należącym do Π_3
- funkcja jest "naturalna", czyli $s''(x_0) = s''(x_n) = 0$

Twierdzenie: NIFS3 zawsze istnieje i jest wyznaczona jednoznacznie.

4.9.2 I sposób konstrukcji

Najbardziej naiwnym sposobem konstrukcji NIFS3 jest stworzenie układu równań dla kolejnych podprzedziałów. Niestety dla $n+1$ węzłów należy wtedy rozwiązać układ $4n$ równań liniowych, który w ogólnej sytuacji rozwiązujemy w czasie $O(n^3)$, co jest nieopłacalne.

4.9.3 II sposób konstrukcji

Dla $x \in [x_{k-1}, x_k]$, gdzie $k = 1, 2, \ldots, n$ definiujemy wzór na k-ty segment NIFS3 poprzez

$$s_k(x) = h_k^{-1}\left[\frac{1}{6}M_{k-1}\left(x_k - x\right)^3 + \frac{1}{6}M_k\left(x - x_{k-1}\right)^3 + \left(y_{k-1} - \frac{1}{6}M_{k-1}h_k^2\right)\left(x_k - x\right) + \left(y_k - \frac{1}{6}M_k h_k^2\right)\left(x - x_{k-1}\right)\right]$$

gdzie $h_k := x_k - x_{k-1}$ oraz $M_k := s''(x_k)$, $M_0 = M_n = 0$.
Momenty M_k $(k = 1, 2, \ldots, n-1)$ spełniają układ równań liniowych postaci

$$\lambda_k M_{k-1} + 2M_k + (1-\lambda_k)M_{k+1} = 6f[x_{k-1}, x_k, x_{k+1}] \text{ dla } \lambda_k = \frac{h_k}{h_k + h_{k+1}}.$$

Postać macierzowa tego układu zwana jest układem trójprzekątniowym:

$$\begin{bmatrix} 2 & 1-\lambda_1 & 0 & 0 & \ldots & 0 \\ \lambda_2 & 2 & 1-\lambda_2 & 0 & \ldots & 0 \\ \vdots & \lambda_3 & 2 & 1-\lambda_3 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & \ldots & \lambda_{n-2} & 2 & 1-\lambda_{n-2} \\ 0 & 0 & \ldots & 0 & \lambda_{n-1} & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{bmatrix}$$

Wyrazy d_k są zdefiniowane przez $6f[x_{k-1}, x_k, x_{k+1}]$. Układ ten możemy rozwiązać za pomocą poniższego algorytmu:

1. $q_0 := 0$

2. $u_0 := 0$
3. $p_k := \lambda_k q_{k-1} + 2$
4. $q_k := \frac{(\lambda_k - 1)}{p_k}$
5. $u_k := \frac{(d_k - \lambda_k u_{k-1})}{p_k}$

Kroki 3-5 powtarzamy w pętli dla $k = 1, 2, \ldots, n-1$.
Wtedy $M_{n-1} = u_{n-1}$, $M_k = u_k + q_k M_{k+1}$ dla $k = n-2, n-3, \ldots, 1$, wszystkie momenty znajdujemy w czasie $O(n)$.

4.10 Krzywe parametryczne na płaszczyźnie

Niech x, y będą ustalonymi funkcjami zmiennej $t \in [a, b]$. Wtedy krzywą parametryczną nazywamy

$$\gamma(t) := \{((x(t), y(t)) : \ a \leqslant t \leqslant b\}.$$

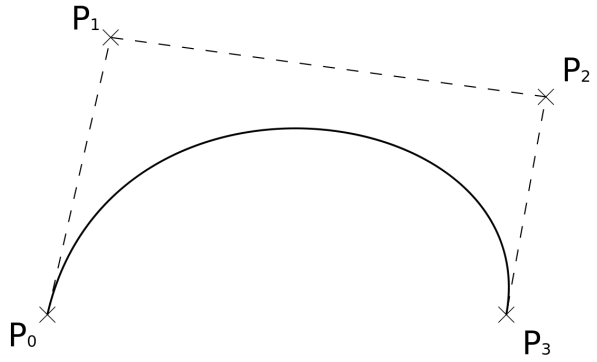
4.11 Wielomian Bernsteina

k-ty wielomian Bernsteina stopnia n wyraża się przez

$$B_k^n(t) := \binom{n}{k} t^k (1-t)^{n-k}$$

Jego podstawowymi własnościami są:

- $t \in [0,1] \Rightarrow B_k^n(t) \geqslant 0$
- B_k^n ma dokładnie jedno ekstremum w przedziale $[0,1]$ dla $t = \frac{k}{n}$
- rekurencyjny wzór: $B_k^n(t) = (1-t)B_k^{n-1}(t) + tB_{k-1}^{n-1}(t)$ dla $0 \leqslant k \leqslant n$
- $B_q^p(t) \equiv 0$ dla $q>p$ lub $q<0$
- $(B_k^n(t))' = n\left(B_{k-1}^{n-1}(t) - B_k^{n-1}(t)\right)$
- $\forall t \in \mathbb{R} : \ \sum_{k=0}^{n} B_k^n(t) \equiv 1$
- wielomiany $B_0^n, B_1^n, \ldots, B_n^n$ tworzą bazę Π_n



Rysunek 4: Otoczka wypukła punktów P_0, P_1, P_2, P_3 (linie przerywane)

4.12 Krzywa Beziera

Dla danych punktów kontrolnych $W_0, W_1, \ldots, W_n$ na płaszczyźnie definiujemy krzywą Beziera stopnia n jako

$$P_n(t) := \sum_{k=0}^{n} W_k B_k^n(t) \text{ dla } t \in [0,1]$$

Dla danego $t \in [0,1]$, $P_n(t)$ jest punktem na płaszczyźnie, mamy więc do czynienia z kombinacją barymetryczną punktów, co więcej punkt $P_n(t)$ należy do otoczki wypukłej $conv\{W_0, W_1, \ldots, W_n\}$. Oprócz tych własności mamy następujące:

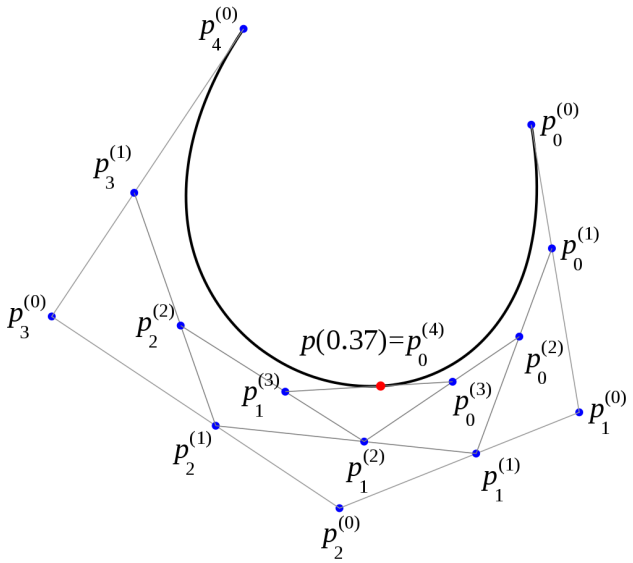
- $P_n(0) = W_0, \ P_n(1) = W_n$
- $P_n^{'}(0) = n(W_1 - W_0)$
- $P_n^{'}(1) = n(W_n - W_{n-1})$

4.13 Wyznaczanie punktu $P_n(t)$ - algorytm de Casteljau

Dla danego $t \in [0,1]$ wyznaczamy punkt $P_n(t)$ na krzywej Beziera za pomocą algorytmu de Casteljau:

- $W_k^{(0)} := W_0 \quad (0 \leqslant k \leqslant n)$
- $W_k^{(i)} := (1-t)W_k^{(i-1)} + tW_{k+1}^{(i-1)} \quad (i = 0, 1, \ldots, n; \ k = 0, 1, \ldots, n-1)$

Wtedy punktem $P_n(t)$ jest $W_0^{(n)}$. Algorytm ten działa w czasie $O(n^2)$.



Rysunek 5: Interpretacja geometryczna algorytmu de Casteljau dla $t = 0.37$

4.14 Kombinacja barycentryczna punktów

Niech $W_0, W_1, \ldots, W_n$ będą punktami na płaszczyźnie, a $\alpha_0, \alpha_1, \ldots, \alpha_n$ liczbami rzeczywistymi takimi, że $\alpha_0 + \alpha_1 + \ldots + \alpha_n = 1$. Wtedy wyrażenie

$$\alpha_0 W_0 + \alpha_1 W_1 + \ldots + \alpha_n W_n$$

nazywamy kombinacją barycentryczną punktów (jest ona zawsze określona jednoznacznie) i utożsamiamy z punktem postaci

$$\underbrace{W_0}_{\text{punkt}} + \underbrace{\alpha_1(W_1 - W_0) + \alpha_2(W_2 - W_0) + \ldots + \alpha_n(W_n - W_0)}_{\text{wektor}}$$

5 Aproksymacja średniokwadratowa na zbiorze dyskretnym

Ideą aproksymacji jest jak najlepsze dopasowanie funkcji do chmury punktów, jednak rezygnujemy z interpolacji, aby zaoszczędzić pamięć.

5.1 Norma średniokwadratowa na zbiorze dyskretnym

Niech dane będą parami różne punkty $\mathcal{X} := \{x_0, x_1, \ldots, x_N\}$ oraz funkcja f określona na $\mathcal{X}$. Normę średniokwadratową funkcji f na zbiorze $\mathcal{X}$ oznaczamy symbolem $||f||_2$ i definiujemy wzorem

$$||f||_2 := \sqrt{\sum_{k=0}^{N} (f(x_k))^2}$$

Norma średniokwadratowa to funkcjonał $||\cdot||_2 : \mathcal{F} \to \mathbb{R}$, gdzie $\mathcal{F}$ jest zbiorem funkcji. Własnościami normy są (dla funkcji f, g):

- $||f||_2 = 0 \Leftrightarrow f(x_k)$ dla $k = 0, 1, \ldots, N$
- $||\alpha \cdot f||_2 = |\alpha| \cdot ||f||_2$ dla $\alpha \in \mathbb{R}$
- $||f + g||_2 \leqslant ||f||_2 + ||g||_2$ (nierówność trójkąta)

Aby sprawdzić czy dwie funkcje są do siebie podobne, należy obliczyć wartość wyrażenia

$$||f - g||_2 := \sqrt{\sum_{k=0}^{N} (f(x_k) - g(x_k))^2}$$

i na jej podstawie ocenić "bliskość" - jeśli otrzymany wynik jest mały, to f, g są bliskie siebie.

5.2 Znajdowanie elementu optymalnego funkcji f

Dla danego zbioru $\mathcal{X}$ oraz funkcji f określonej na $\mathcal{X}$ element $w^* \in \mathcal{F}$ nazywamy elementem optymalnym w sensie aproksymacji średniokwadratowej, że

$$||f - w^*||_2 = \min_{w \in \mathcal{F}} ||f - w||_2 = \min_{w \in \mathcal{F}} \sqrt{\sum_{k=0}^{N} \left(\underbrace{f(x_k)}_{y_k} - w(x_k) \right)^2}$$

Jeżeli aproksymacja średniokwadratowa napotka "odstające" obserwacje, tzn. niepasujące do reszty punktów z chmury, zostaną one praktycznie pominięte - współczynnik zwykle zmienia się bardzo minimalnie.
W sytuacji ogólnej wybieramy pewne funkcje podstawowe $g_0(x), \ldots, g_m(x)$ (np. $g_i(x) = x^i$, wtedy $lin\{g_0, \ldots, g_m\} \equiv \Pi_m$) i za model przyjmujemy

$\mathcal{F} := \{a_0 g_0(x) + a_1 g_1(x) + \ldots + a_m g_m(x) : \ a_0, \ldots, a_m \in \mathbb{R}\}$. Dla pomiarów $(x_k, y_k),\ (0 \leqslant k \leqslant N)$ szukamy elementu optymalnego $w^* \in \mathcal{F}$ o własności:

$$||f - w^*||_2 = \min_{w \in \mathcal{F}} ||f - w||_2 = \min_{a_0,\ldots,a_m \in \mathbb{R}} \sqrt{E(a_0, \ldots, a_m)}$$

gdzie E jest funkcją błędu zadaną przez

$$E(a_0, \ldots, a_m) = \sum_{k=0}^{N} \left(y_k - \underbrace{\sum_{i=0}^{m} a_i g_i(x_k)}_{w(x_k)} \right)^2$$

Następnie szukamy minimum funkcji błędu E za pomocą pochodnych cząstkowych (przyrównujemy je do zera), definiuje ono element optymalny w^* w sensie aproksymacji średniokwadratowej na zbiorze dyskretnym.

5.3 Wielomianowa aproksymacja średniokwadratowa na zbiorze dyskretnym

Dla danego zbioru $\mathcal{X}$, funkcji f na nim określonej i liczby naturalnej m chcemy znaleźć wielomian $w_m^* \in \Pi_m$, dla którego

$$||f - w_m^*||_2 = \min_{w_m \in \Pi_m} ||f - w_m||_2 \equiv \min_{w_m \in \Pi_m} \sqrt{\sum_{k=0}^{N} \left(y_k - w_m(x_n)\right)^2}$$

Efektywna metoda (o złożoności $O(mN)$) konstrukcji wielomianu optymalnego $w_m \in \Pi_m$ wymaga znajomości niżej opisanych narzędzi.

5.3.1 Dyskretny iloczyn skalarny na zbiorze $\mathcal{X}$

Dla funkcji f, g iloczyn skalarny wyraża się wzorem

$$(f, g)_N = \sum_{k=0}^{N} f(x_k) g(x_k)$$

5.3.2 Ortogonalność funkcji

Funkcje f oraz g są ortogonalne gdy dla ustalonego iloczynu skalarnego zachodzi

$$(f, g)_N = \sum_{k=0}^{N} f(x_k) g(x_k) = 0$$

Układ funkcji $f_0, f_1, \ldots, f_m$ dla $m \leqslant N$ jest ortogonalny względem iloczynu skalarnego $(\cdot, \cdot)_N$ wtedy i tylko wtedy, gdy:

- $(f_i, f_j)_N = 0$ dla $i \neq j,\ 0 \leqslant i, j \leqslant m$
- $(f_i, f_i)_N > 0$

5.3.3 Ortogonalizacja Grama-Schmidta

Dla danego układu liniowo niezależnych funkcji $g_0, g_1, \ldots, g_m$ oraz ustalonego wcześniej iloczynu skalarnego, układ funkcji $f_0, f_1, \ldots, f_m$ zadanych w następujący sposób rekurencyjny:

- $f_0 := g_0$
- $f_k := g_k - \sum_{j=0}^{k-1} \left(\frac{(g_k, f_j)_N}{(f_j, f_j)_N} \cdot f_j \right)$ dla $k = 1, 2, \ldots, m$

jest układem ortogonalnym względem wybranego iloczynu skalarnego.
Aby znaleźć układ ortogonalny wielomianów generujących przestrzeń Π_m możemy użyć ortogonalizacji Grama-Schmidta dla układu $1, x, x^2, \ldots, x^m$. Skonstruujemy go używając przedstawionej wyżej zależności rekurencyjnej dla $g_k(x) := x^k$ oraz $P_k \equiv f_k$ o własnościach:

- $P_k \in \Pi_k \setminus \Pi_{k-1}$ dla $k = 0, 1, \ldots, m;\ \Pi_{-1} = \emptyset$
- $(P_k, P_l)_N = 0$ dla $k \neq l,\ 0 \leqslant k, l \leqslant m$
- $(P_k, P_k)N{>}0$

Ta metoda jednak jest zbyt droga, jej złożoność to $O(m^2 N)$, a sama numeryczna realizacja ortogonalizacji Grama-Schmidta sprawia problemy numeryczne.

5.3.4 Ciąg wielomianów ortogonalnych

Z powodów przedstawionych w poprzednim paragrafie, skonstruujemy bazę przestrzeni Π_m wykorzystując wielomiany $P_0, P_1, \ldots, P_m,\ (m \leqslant N)$ spełniające zależność rekurencyjną:

- $P_0(x) \equiv 1$
- $P_1(x) = x - c_1$
- $P_k(x) = (x - c_k)P_{k-1}(x) - d_k P_{k-2}(x) \quad (k = 2, 3, \ldots, m;\ m \leqslant N)$

gdzie

- $c_k := \frac{(xP_{k-1}, P_{k-1})_N}{(P_{k-1}, P_{k-1})_N}$ dla $(1 \leqslant k \leqslant m)$
- $d_k := \frac{(P_{k-1}, P_{k-1})_N}{(P_{k-2}, P_{k-2})_N}$ dla $(2 \leqslant k \leqslant m)$

Zapis $(xP_{k-1}, P_{k-1})_N$ z licznika c_k oznacza $\sum_{k=0}^{N} x_k P_{k-1}(x_k) \cdot P_{k-1}(x_k)$.
Dzięki temu uzyskujemy koszt $O(mN)$, a numerycznie metoda działa bardzo dobrze.

5.3.5 Rozwiązanie zadania

Wielomian optymalny $w_m^* \in \Pi_m$ wyraża się wzorem

$$w_m^*(x) = \sum_{k=0}^{m} a_k P_k(x), \quad a_k := \frac{(f, P_k)_N}{(P_k, P_k)_N}$$

gdzie $P_0, P_1, \ldots, P_m$ to ciąg wielomianów ortogonalnych względem dyskretnego iloczynu skalarnego dla zbioru $\mathcal{X} = \{x_0, x_1, \ldots, x_N\}$.
Błąd wielomianowej aproksymacji średniokwadratowej wyraża się wzorem

$$||f - w_m^*||_2 = \sqrt{||f||_2 - \sum_{k=0}^{m} \frac{(f, P_k)_N^2}{(P_k, P_k)_N}}$$

Możemy zaobserować, że $w_{m+1}^*(x) = w_m^*(x) + a_{m+1} P_{m+1}(x)$, czyli w łatwy sposób możemy budować ciąg wielomianów optymalnych, przy czym dla każdego kolejnego m kontrolujemy wartość błędu. Wielomian $w_N(x)$ jest wielomianem interpolacyjnym dla danych (x_k, y_k).
Aproksymacji tej warto używać dla $m \ll N$, jako że wtedy bardzo mocno kompresujemy dane.

6 Kwadratury - całkowanie numeryczne

Idea kwadratur czyli całkowania numerycznego polega na zastąpieniu trudnej funkcji $f(x)$ dla $x \in [a, b]$ poprzez inną, łatwiejszą - np. wielomian $w(x)$, jako że potrafimy je łatwo całkować, co pozwoli nam na przybliżenie $f(x)$.

$$w(x) \approx f(x) \text{ dla } x \in [a,b] \Rightarrow \underbrace{\int_a^b f(x)dx}_{\text{trudna całka}} \approx \underbrace{\int_a^b w(x)dx}_{\text{łatwa całka}}$$

6.1 Funkcja podcałkowa, funkcja pierwotna

Niech $f : \mathbb{R} \to \mathbb{R}$ będzie dowolną funkcją podcałkową dla $x \in [a, b]$. Funkcją pierwotną F będzie wtedy funkcja spełniająca równanie

$$F'(x) = f(x)$$

Wtedy możemy również obliczyć całkę oznaczoną korzystając z własności

$$\int_a^b f(x)dx = F(b) - F(a)$$

6.2 Metody obliczania całek

6.2.1 Całkowanie przez części

Jeżeli funkcje f, g są ciągłe oraz mają ciągłe pochodne, to

$$\int f(x)g'(x)dx = f(x) \cdot g(x) - \int f'(x) \cdot g(x)dx$$

6.2.2 Całkowanie przez podstawianie

Jeżeli funkcję f można zapisać w postaci $f(x) = g(h(x)) \cdot h'(x)$, gdzie funkcja $h(x)$ ma ciągłą pochodną, to możemy skorzystać z całkowania przez podstawianie:

$$\int f(x)dx = \int g(h(x)) \cdot h'(x)dx = \underbrace{\left| \begin{smallmatrix} t=h(x) \\ dt=h'(x)dx \end{smallmatrix} \right|}_{\text{podstawienie}} = \int g(t)dt$$

6.3 Kwadratura liniowa

Dla danych parami różnych węzłów kwadratury $x_0^{(n)}, x_1^{(n)}, \ldots, x_n^{(n)}$ oraz liczb $A_0^{(n)}, A_1^{(n)}, \ldots, A_n^{(n)}$, kwadraturą liniową funkcji f nazywamy wyrażenie postaci

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

Naszym celem jest dobranie takich współczynników $A_k^{(n)}$ oraz węzłów $x_k^{(n)}$ kwadratury Q_n w taki sposób, aby

$$Q_n(f) \approx \int_a^b f(x)dx = I(f)$$

dla f należącej do "dużej" rodziny funkcji, tzn. współczynniki i węzły mają być uniwersalne (dobre dla wielu funkcji).

6.3.1 Błąd kwadratury liniowej

Błędem kwadratury liniowej Q_n nazywamy

$$R_n(f) := I(f) - Q_n(f)$$

Jeśli $R_n(f) = 0$, to $I(f) = Q_n(f)$, a więc dokładnie całkujemy funkcję f.

6.3.2 Rząd kwadratury liniowej

Rząd kwadratury liniowej Q_n wynosi r, jeśli:

- $\forall w \in \Pi_{r-1}.\ I(w) = Q_n(w)$ (błąd kwadratury $R_n(w) = 0$)
- $\exists w \in \Pi_r \setminus \Pi_{r-1}.\ I(w) \neq Q_n(w)$ (błąd kwadratury $R_n(w) \neq 0$)

Twierdzenie: Rząd kwadratury liniowej Q_n jest nie przekracza $2n + 2$.

6.3.3 Strategia konstrukcji kwadratur

Węzły $x_k^{(n)}$ oraz współczynniki $A_k^{(n)}$ dobieramy tak, aby rząd kwadratury liniowej Q_n był jak największy. Oznacza to, że chcemy dokładnie całkować wielomiany tak wysokiego stopnia, jak to tylko możliwe.

6.4 Kwadratura interpolacyjna

Ideą kwadratury interpolacyjnej jest całkowanie wielomianu $L_n(x)$ zamiast funkcji $f(x)$ w węzłach $x_0^{(n)}, x_1^{(n)}, \ldots, x_n^{(n)}$. Wielomian $L_n \in \Pi_n$ musi spełniać warunek $L_n(x_k^{(n)}) = f(x_k^{(n)})$ dla $k = 0, 1, \ldots, n$ i jest on postaci

$$L_n(x) = \sum_{k=0}^{n} \lambda_k(x) f\left(x_k^{(n)}\right), \text{ gdzie } \lambda_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i} \in \Pi_n$$

Scałkujmy postać Lagrange'a wielomianu interpolacyjnego:

$$\int_a^b L_n(x)dx = \int_a^b \left[\sum_{k=0}^{n} \lambda_k(x) f\left(x_k^{(n)}\right)\right] dx = \sum_{k=0}^{n} \left[\int_a^b \lambda_k(x)dx \cdot f\left(x_k^{(n)}\right)\right] = Q_n(f)$$

W takiej postaci kwadratura interpolacyjna jest przedstawiona takim wzorem jak wcześniej. Jej współczynniki mają postać

$$A_k^{(n)} := \int_a^b (\lambda_k(x)) = \int_a^b \left(\prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i^{(n)}}{x_k^{(n)} - x_i^{(n)}} \right)$$

Twierdzenie: Kwadratura liniowa Q_n ma rząd większy lub równy $n+1$ wtedy i tylko wtedy, gdy jest ona kwadraturą interpolacyjną.

6.5 Kwadratura Newtona-Cotesa

Ideą tej kwadratury jest stworzenie kwadratury interpolacyjnych dla węzłów równoodległych, a więc przyjmujemy, że są nimi

$$x_k^{(n)} = a + \underbrace{\frac{b-a}{n}}_{h} \cdot k$$

Kwadratura Newtona-Cotesa będzie wtedy zdefiniowana przez

$$Q_n^{NC}(f) := \sum_{k=0}^{n} A_k^{(n)} f(a + hk)$$

6.5.1 Współczynniki kwadratury Newtona-Cotesa

Jawnym wzorem na współczynniki kwadratury Newtona-Cotesa jest

$$A_k^{(n)} = \frac{(-1)^{n-k} h}{k!(n-k)!} \int_0^n \left[\prod_{\substack{j=0 \\ j \neq k}}^{n} (t-j) \right] dt$$

gdzie t jest taką zmienną, że $x = a + th$.

6.5.2 Błąd kwadratury Newtona-Cotesa

$$R_n(f) = \begin{cases} \frac{f^{(n+1)}(\eta)}{(n+1)!} \int_a^b p_{n+1}(x)dx & (n = 1, 3, 5, \ldots) \\ \frac{f^{(n+2)}(\alpha)}{(n+2)!} \int_a^b x p_{n+1}(x)dx & (n = 2, 4, 6, \ldots) \end{cases}$$

dla $p_{n+1}(x) = \left(x - x_0^{(n)}\right)\left(x - x_1^{(n)}\right) \ldots \left(x - x_n^{(n)}\right)$.

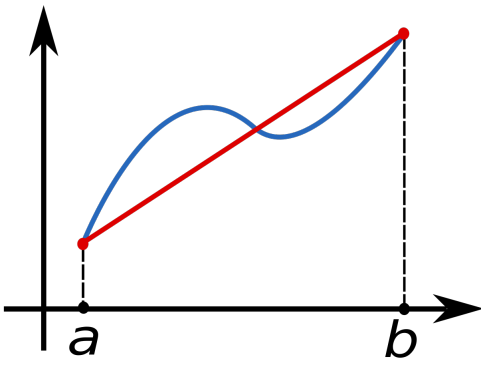
6.5.3 Rząd kwadratury Newtona-Cotesa

Rząd kwadratury Newtona-Cotesa wynosi $n+1$ dla n nieparzystych oraz $n+2$ dla n parzystych. Warto jednak pamiętać, że pod względem rzędu, kwadratury Newtona-Cotesa są słabym narzędziem, bo jesteśmy "na granicy" danego oszacowania dla rzędu kwadratur interpolacyjnych.

6.6 Wzór trapezów, złożony wzór trapezów

Wzór trapezów jest jednym ze wzorów do przybliżonego obliczania całek oznaczonych. Dla danej funkcji f na przedziale $x \in [a, b]$ możemy oszacować wartość całki $\int_a^b f(x)dx$ za pomocą kwadratury

$$Q_1(f) := \frac{b-a}{2}\left(f(a) + f(b)\right)$$



Rysunek 6: Zasada działania wzoru trapezów

Rząd tej metody wynosi 2. Jednokrotne wykorzystanie wzoru trapezów może dokładnie obliczyć całkę (w każdym przypadku) tylko z funkcji liniowej i stałej, dlatego powinniśmy używać **złożonego wzoru trapezów**, tzn. przedział $[a, b]$ dzielimy na równe części i na każdej z nich stosujemy wzór trapezów. Otrzymujemy wtedy dla węzłów $t_k := a + h_n k$, $(k = 0, 1, \ldots, n)$, $h_n := \frac{b-a}{n}$ wzór

$$\int_a^b f(x)dx = T_n(f) + R_n^T$$

gdzie $T_n(f)$ jest złożonym wzorem trapezów, a R_n^T błędem złożonego wzoru trapezów. Złożony wzór trapezów wyraża się wzorem

$$T_n(f) := h_n \sum_{k=0}^{n}{}'' f(t_k) = h_n(\frac{1}{2}f(t_0) + f(t_1) + \ldots + f(t_{n-1}) + \frac{1}{2}f(t_n)$$

Oznaczenie: $\sum''$ x - pierwszy i ostatni wyraz mnożony jest przez $\frac{1}{2}$.
Twierdzenie: Jeśli funkcja $f \in C^2[a, b]$, to

$$R_n^T(f) := \frac{a-b}{12}h_n^2 f''(\eta_n)$$

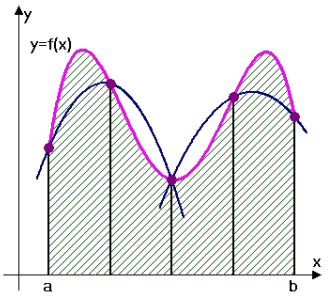
dla $\eta_n \in (a, b)$, a h_n^2 jest rzędu $O(n^{-2})$.

Twierdzenie: Jeśli $f \in C[a, b]$, to $\lim_{n\to\infty} T_n(f) = \int_a^b f(x)dx$.

6.7 Wzór Simpsona, złożony wzór Simpsona

Wzór Simpsona jest kolejnym sposobem przybliżonego obliczania całek oznaczonych. Polega on na interpolacji kwadratowej w trzech punktach: a, $\frac{a+b}{2}$, b. Kwadratura ta wyznaczona jest wzorem

$$Q_2(f) := \frac{b-a}{2}\left(\frac{1}{3}f(a) + \frac{4}{3}f\left(\frac{a+b}{2}\right) + \frac{1}{3}f(b)\right)$$



Rysunek 7: Zasada działania złożonego wzoru Simpsona dla 4 części

Rząd tej metody wynosi 4, a więc możemy dokładnie obliczyć całki nawet trzeciego stopnia. Jednak tutaj podobnie jak w przypadku wzoru trapezów, warto byłoby skorzystać z podziału przedziału na parzystą liczbę równych podprzedziałów i dla każdych trzech kolejnych punktów stosować wzór Simpsona, aby dokładniej wyliczać całki oznaczone. Otrzymujemy wtedy dla węzłów $t_k := a + h_n k$, $(k = 0, 1, \ldots, n)$, $h_n := \frac{b-a}{n}$ wzór

$$\int_a^b f(x)dx = S_n(f) + R_n^S$$

gdzie $S_n(f)$ jest złożonym wzorem Simpsona, a R_n^S błędem złożonego wzoru Simpsona. Złożony wzór Simpsona wyraża się wzorem

$$S_n(f) := \frac{h_n}{3}\left(2\sum_{k=0}^{m}{}'' f(t_{2k}) + 4\sum_{k=1}^{m} f(t_{2k-1})\right)$$

Twierdzenie: Jeśli $f \in C^4[a,b]$, to

$$R_n^S(f) := \frac{a-b}{180}h_n^4 f^{(4)}(\alpha_n)$$

dla $\alpha_n \in (a,b)$, a h_n^4 jest rzędu $O(n^{-4})$.

Twierdzenie: Jeśli $f \in C[a,b]$, to $\lim_{n\to\infty} S_n(f) = \int_a^b f(x)dx$.

Obserwacja: $\underset{(n=2m)}{S_n(f)} = \frac{4T_n(f)-T_m(f)}{4-1}$ - ekstrapolacja

6.8 Metoda Romberga

Niech $n = 2^k$ dla $k \in \mathbb{N}$, $h_k := \frac{b-a}{2^k}$, $x_i^{(k)} := a + ih_k$ dla $i = 0, 1, \ldots, 2^k$. Zdefiniujmy $T_{0k} := T_{2^k}(f) = h_k \sum_{i=0}^{2^k}{}'' f(x_i^{(k)})$ (złożone wzory trapezów). Kolejne elementy T_{mk} dla $k = 0, 1, \ldots$ oraz $m = 1, 2, \ldots$ definiujemy rekurencyjnie za pomocą wzoru

$$T_{m,k} := \frac{4^m T_{m-1,k+1} - T_{m-1,k}}{4^m - 1}$$

6.8.1 Tablica Romberga

Tworzenie tablicy Romberga przebiega w sposób pokazany na przykładzie poniżej (podobnie jak w ilorazach różnicowych).

$$\begin{array}{cccccc}
T_{0,0} & & & & & \\
T_{0,1} & T_{1,0} & & & & \\
T_{0,2} & T_{1,1} & T_{2,0} & & & \\
T_{0,3} & T_{1,2} & T_{2,1} & T_{3,0} & & \\
T_{0,4} & T_{1,3} & T_{2,2} & T_{3,1} & T_{4,0} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
T_{0,m} & T_{1,m-1} & T_{2,m-2} & \ldots & \ldots & \ldots \quad T_{m,0}
\end{array}$$

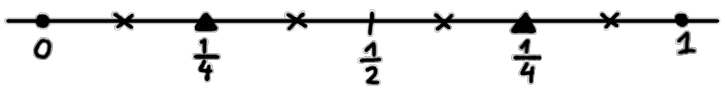
Przeważnie każda kolejna kolumna tablicy Romberga zbiega szybciej do wartości całki, a najszybsza zbieżność występuje na przekątnej tablicy.

Twierdzenie: $\lim_{k\to\infty} T_{mk} = \int_a^b f(x)dx$ dla ustalonego m i funkcji ciągłej f.

Twierdzenie: $\lim_{m\to\infty} T_{mk} = \int_a^b f(x)dx$ dla ustalonego k i funkcji ciągłej f.

6.8.2 Efektywne obliczanie wartości w pierwszej kolumnie

Wartość funkcji f możemy obliczyć tylko raz w każdym punkcie i ją zapamiętać, jako że dla każdego kolejnego k (przy $n = 2^k$) będziemy dzielić przedział na coraz mniejsze równe części. Dzięki temu w pierwszym kroku obliczymy $f(0)$ oraz $f(1)$i, w kolejnym $f(\frac{1}{2})$, w kolejnym $f(\frac{1}{4})$ oraz $f(\frac{3}{4})$ itd.



Rysunek 8: Przykład efektywnego obliczania $T_{0,n}$ na przedziale $[0, 1]$

6.9 Kwadratura Gaussa

Kwadraturami Gaussa nazywamy kwadratury Q_n rzędu $2n+2$. Są one zawsze interpolujące, jako że ich rząd jest większy od $n+1$. Współczynniki wyrażone są wzorem

$$A_k^{(n)} = \int_a^b \left[\prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i^{(n)}}{x_k^{(n)} - x_i^{(n)}} \right] dx$$

Przyjmijmy $a = -1,\ b = 1$, tzn. weźmy całki $\int_{-1}^{1} f(x)dx$ - wtedy będziemy mówić o **kwadraturach Gaussa-Legandre'a**. Należy jeszcze znaleźć węzły, które można obliczyć poprzez znalezienie miejsc zerowych w wielomianach Legandre'a wyrażone rekurencyjnym wzorem:

- $P_0(x) \equiv 1$
- $P_1(x) \equiv x$
- $P_k(x) = \frac{2k-1}{k} x P_{k-1}(x) - \frac{k-1}{2} P_{k-2}(x)$ dla $k = 2, 3, \ldots$

Miejsca zerowe wielomianu P_{n+1} są węzłami kwadratury Gaussa-Legandre'a mającej rząd równy $2n+2$. Nie ma jawnego wzoru na obliczanie ich, jednak ich wartości można przybliżyć numerycznie (chociażby używając metody bisekcji, stycznych czy siecznych). Współczynniki i węzły są od dawna dokładnie stablicowane, więc możemy ich używać w łatwy sposób.

7 Algorytmy numeryczne algebry liniowej

7.1 Krótkie przypomnienie z algebry (macierze)

Macierzą A nazywamy tablicę elementów rozmiaru $n \times m$, której elementami a_{ij} są liczby rzeczywiste. Standardowymi zapisami macierzy są

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \in \mathbb{R}^{n \times m} \text{ oraz } A = [a_{ij}] \in \mathbb{R}^{n \times m}$$

gdzie n jest liczbą wierszy, a m liczbą kolumn. Dla $n = m$ mówimy o macierzy kwadratowej o stopniu n.

7.1.1 Podstawowe działania na macierzach

Niech $A = [a_{ij}] \in \mathbb{R}^{n \times m}$, $B = [b_{ij}] \in \mathbb{R}^{n \times m}$, $C = [c_{ij}] \in \mathbb{R}^{m \times k}$. Podstawowymi działaniami na macierzach są:

- dodawanie/odejmowanie: $A \pm B =: D = [d_{ij}] \in \mathbb{R}^{n \times m}$, $d_{ij} = a_{ij} \pm b_{ij}$
- mnożenie przez skalar: $A \cdot \alpha =: E = [e_{ij}] \in \mathbb{R}^{n \times m}$, $e_{ij} = a_{ij} \cdot \alpha$ $(\alpha \in \mathbb{R})$
- mnożenie macierzy: $A \cdot C =: F = [f_{ij}] \in \mathbb{R}^{n \times k}$, $f_{ij} := \sum_{h=1}^{m} a_{ih} \cdot c_{hj}$
- transpozycja ^{T}: $A^T = [a_{ji}] \in \mathbb{R}^{m \times n}$

7.1.2 Wyznacznik macierzy kwadratowej

Wyznacznikiem macierzy nazywamy jednoznacznie określoną funkcję $det : \mathbb{R}^{n \times n} \to \mathbb{R}$. Przyporządkowuje ona macierzy pewną liczbę rzeczywistą. Istnieje jawny wzór na obliczanie wyznacznika, jednak jego złożoność czasowa to $O(n!)$. Bardziej opłacalną metodą jest zastosowanie przekształceń elementarnych do zamiany macierzy na macierz trójkątną:

$$A = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

Wtedy wyznacznik można łatwo obliczyć ze wzoru:

$$det(A) = a_{11} \cdot a_{22} \cdot \ldots \cdot a_{nn}$$

Podstawowymi własnościam wyznaczników są:

- $det(A) = det(A^T)$
- $det(A \cdot B) = det(A) \cdot det(B)$

7.1.3 Odwracanie macierzy kwadratowej

Zdefiniujmy macierz jednostkową $I_n \in \mathbb{R}^{n\times n}$ w następujący sposób:

$$I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}$$

Macierzą odwrotną do $A \in \mathbb{R}^{n\times n}$ nazywamy taką macierz $A^{-1} \in \mathbb{R}^{n\times n}$, że spełniona jest zależność

$$A^{-1} \cdot A = A \cdot A^{-1} = I_n$$

Macierz odwrotna do A istnieje wtedy i tylko wtedy, gdy $det(A) \neq 0$.

7.1.4 Układ równań liniowych

Weźmy układ równań liniowych następującej postaci:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases} \equiv A \cdot x = b$$

gdzie $A = [a_{ij}] \in \mathbb{R}^{n\times n}$ oraz $b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$ są danymi, a $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ szukanymi niewiadomymi. Taki układ ma jednoznaczne rozwiązanie wtedy i tylko wtedy, gdy $det(A) \neq 0$. Rozwiązaniem takiego układu równań liniowych jest:

$$\begin{aligned} Ax &= b / \cdot_L A^{-1} \\ A^{-1}(Ax) &= A^{-1}b \\ \underbrace{A^{-1}A}_{I_n} x &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

7.1.5 Wzory Cramera

Mając macierz A, wektor b oraz x możemy utworzyć macierz A_k poprzez zamianę k-tej kolumnty macierzy A na elementy wektora b. Wtedy kolejnymi wartościami wektora x będą wyrazy

$$x_k = \frac{det(A_k)}{det(A)}$$

7.1.6 Przykład rozwiązania układu równań liniowych

Niech $A = \begin{bmatrix} 1 & 0.99 \\ 0.99 & 0.98 \end{bmatrix}$, $b = \begin{bmatrix} 1.99 \\ 1.97 \end{bmatrix}$, $\widetilde{b} = b + \begin{bmatrix} -0.000097 \\ +0.000106 \end{bmatrix} = \begin{bmatrix} 1.989903 \\ 1.970106 \end{bmatrix}$, a więc $b \approx \widetilde{b}$. Rozwiązując układ $Ax = b$ otrzymamy $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, jednak dla układu $Ax = \widetilde{b}$ otrzymamy $x = \begin{bmatrix} 3.0000 \\ -1.0203 \end{bmatrix}$. Możemy stąd wywnioskować, że zadanie jest źle uwarunkowane i należy być ostrożnym nawet dla prostych danych.

7.2 Rozwiązywanie układów równań o macierzy trójkątnej dolnej

$$\begin{cases} l_{11}x_1 & = b_1 \\ l_{21}x_1 + l_{22}x_2 & = b_2 \\ \quad \vdots & \quad \vdots \\ l_{n1}x_1 + l_{n2}x_2 + \cdots + l_{nn}x_n & = b_n \end{cases} \equiv Lx = b$$

gdzie L jest macierzą trójkątną dolną (tzn. wszystkie pozostałe wartości są 0), b wektorem znanych wartości, a x wektorem niewiadomych. Taki układ możemy rozwiązać w czasie $O(n^2)$ za pomocą poniższej metody:

$$x_k = \frac{b_k - \sum_{j=1}^{k-1} l_{kj}x_j}{l_{kk}}$$

dla $k = 1, \ldots, n$, $l_{kk} \neq 0$. W podobny sposób możemy rozwiązać układ trójkątny górny, jednak w tym przypadku wartości x liczymy od x_n do x_1.

7.3 Metoda faktoryzacji rozwiązywania układów równań liniowych

Załóżmy, że układ $Ax = b$ ma jednoznaczne rozwiązanie ($A \in \mathbb{R}^{n \times n}$ oraz $x, b \in \mathbb{R}^n$) i że znamy taką macierz trójkątną dolną $L \in \mathbb{R}^{n \times n}$ oraz macierz trójkątną górną $U \in \mathbb{R}^{n \times n}$, że $A = L \cdot U$. Możemy wtedy pokazać, że rozwiązaniem wyjściowego układu jest rozwiązanie dwóch układów trójkątnych.

$$\begin{aligned} Ax &= b \\ LUx &= b \\ L\underbrace{(Ux)}_{y} &= b \end{aligned}$$

Przyjmijmy, że $y := Ux$, wtedy $Ly = b$ (układ trójkątny). Stąd w łatwy sposób możemy wyznaczyć najpierw y, a następnie rozwiązać drugi układ

równań postaci $Ux = y$, którego rozwiązaniem jest szukany x, będący rozwiązaniem wyjściowego układu

$$Ax = b \Leftrightarrow \begin{cases} Ly = b, \text{ gdzie znamy } L \text{ oraz } b \\ Ux = y, \text{ gdzie znamy } U \text{ oraz } y \text{ z powyższego równania} \end{cases}$$

Rozwiązanie takiego układu zajmuje czas $O(n^2)$, jednak musimy znać rozkład LU macierzy, który nie zawsze istnieje.

7.4 Rozkład trójkątny macierzy ≡ rozkład LU

Weźmy macierz $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ taką, że wyznacznik wszystkich jej minorów głównych jest różny od zera, tzn.

$$det \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{bmatrix} \neq 0 \text{ dla } k = 1, 2, \dots, n$$

Wówczas istnieje dokładnie jedna para macierzy $L, U \in \mathbb{R}^{n \times n}$, gdzie L jest macierzą trójkątną dolną **z jedynkami na przekątnej**, a U jest macierzą trójkątną górną, spełniającą równość $A = L \cdot U$. Ponadto wyznacznikiem macierzy A jest $det(A) = u_{11} \cdot u_{22} \cdot \ldots \cdot u_{nn}$. Jawnym wzorem na rozkład LU o złożoności czasowej $O(n^3)$ jest

$$\begin{cases} u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \text{ dla } (i \leqslant j) \\ l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}}{u_{jj}} \text{ dla } (i > j) \end{cases}$$

Dzięki rozkładowi LU możemy między innymi łatwo obliczyć wyznacznik ze wzoru:

$$det(A) = det(L \cdot U) = det(L) \cdot det(U) = u_{11} \cdot u_{22} \cdot \ldots \cdot u_{nn}$$

czy odwrócić macierz ($det(A) \neq 0$):

$$A = LU \Leftrightarrow A^{-1} = (LU)^{-1} = U^{-1} \cdot L^{-1}$$

Odwrotnością macierzy trójkątnej jest zawsze macierz trójkątna tego samego typu.