Metody numeryczne, III rok Informatyki, 2011/2012

1. Rozwiązywanie równań nieliniowych
2. Arytmetyka zmiennopozycyjna
3. Błędy w obliczeniach. Uwarunkowanie zadania. Numeryczna poprawność i stabilność algorytmu
4. Rozwiązywanie układów równań liniowych. Metody bezpośrednie i iteracyjne
5. Liniowe zadania najmniejszych kwadratów
6. Algebraiczne zagadnienie własne
7. Interpolacja wielomianowa
8. Interpolacja funkcjami sklejanymi
9. Interpolacja trygonometryczna. Algorytm FFT
10. Aproksymacja funkcji
11. Numeryczne obliczanie całek
12. Wybrane środowiska i biblioteki dla obliczeń numerycznych

Zasady zaliczania przedmiotu

Na zaliczenie przedmiotu składają się: zaliczenie ćwiczeń i zdanie egzaminu. Połowa ćwiczeń ma miejsce w laboratorium, pozostałe ćwiczenia są w sali przy tablicy. Na końcową ocenę składają się

- punkty, którymi prowadzący ćwiczenia ocenił prace domowe, tj. rozwiązania zadań na kartce,
- punkty za rozwiązania zadań programistycznych,
- punkty zdobyte na egzaminie pisemnym.

Po egzaminie pisemnym będą wystawione propozycje ocen, w których zadania domowe, zadania programistyczne i egzamin pisemny mają udziały odpowiednio 25%, 35% i 40%. Otrzymaną propozycję oceny uczestnik zajęć może przyjąć, lub wystawić na ryzyko zmiany na egzaminie ustnym.

Literatura

- Kincaid D., Cheney W.: *Analiza numeryczna*, WNT, Warszawa, 2006.
- Krzyżanowski P.: *Obliczenia inżynierskie i naukowe*, PWN, Warszawa, 2011.
- Jankowska J., Jankowski M., Dryja M: *Przegląd metod i algorytmów numerycznych* cz. 1 i 2, WNT, Warszawa, 1988.
- Dahlquist G., Björck Å: *Metody numeryczne*, PWN, Warszawa, 1983.

Egzamin z Metod Numerycznych, III rok Inf.

(*Ściśle tajne* przed godz. 14:30 28 stycznia 2012.)

Proszę bardzo uważnie przeczytać treść zadań. Na ocenę bardzo duży wpływ będzie miała czytelność rozwiązań i poprawność uzasadnienia każdej odpowiedzi.

1. Wykonaj dwie iteracje metody Newtona dla układu równań

$$\begin{cases} x^3 + x - xy - 2y^2 = -4, \\ y^3 - y = -6, \end{cases}$$

dla punktu startowego $(x_0, y_0) = (2, -2)$.

2. Wartość wyrażenia $w = a^3 - b^3$ została obliczona przy użyciu następującego algorytmu, zrealizowanego za pomocą arytmetyki zmiennopozycyjnej:

```
x1 = a*a+b*b;
x2 = a+b;
x3 = 0.5*(x1+x2*x2);
w = x3*(a-b);
```

Napisz wyrażenie, którego wartością jest błąd (bezwzględny) otrzymanego wyniku, jeśli w żadnym z działań nie wystąpił nadmiar ani niedomiar.

3. Wartości $f_1, \ldots, f_N$ pewnej funkcji rzeczywistej f są podane w punktach $x_1, \ldots, x_N$. Funkcja ta ma być przybliżona przez wielomian w stopnia co najwyżej $n < N$ tak, aby wyrażenie $\sum_{i=1}^{N} (f_i - w(x_i))^2$ było jak najmniejsze. Napisz układ równań liniowych, taki że rozwiązanie powyższego zadania aproksymacji można sprowadzić do liniowego zadania najmniejszych kwadratów dla tego układu. Podaj algorytm rozwiązywania tego zadania za pomocą odbić Householdera. Jaki jest koszt tego algorytmu w zależności od liczb n i N?

4. Skonstruuj odpowiednią bazę Newtona i rozwiąż przy użyciu algorytmu różnic dzielonych zadanie interpolacyjne Hermite'a dla danych przedstawionych w tabelce:

x_i	1	3
$f(x_i)$	-4	-8
$f'(x_i)$	-10	30
$f''(x_i)$	-8	

5. Rozważamy konstrukcję interpolacyjnej funkcji sklejanej drugiego stopnia, $s(x) = \sum_{i=0}^{N-3} d_i N_i^2(x)$, której węzły są liczbami naturalnymi, $u_i = i$ dla $i = 0, \ldots, N$, reprezentowanej za pomocą funkcji B-sklejanych N_i^2. Warunki interpolacyjne (tj. wartości funkcji, $s_k = s(v_k)$) są zadane w punktach $v_0 = 2$, $v_k = k + 1\frac{1}{2}$ dla $k = 1, \ldots, N-4$ i $v_{N-3} = N - 2$. Wiedząc, że $N_i^2(i + x) = N_0^2(x)$ dla każdego $x \in \mathbb{R}$ oraz $i \in \{0, \ldots, N-3\}$, a ponadto $N_0^2(x) = 0$ jeśli $x \leqslant 0$ lub $x \geqslant 3$, oraz $N_0^2(\frac{1}{2}) = N_0^2(2\frac{1}{2}) = \frac{1}{8}$, $N_0^2(1) = N_0^2(2) = \frac{1}{2}$ i $N_0^2(1\frac{1}{2}) = \frac{3}{4}$, napisz układ równań, którego rozwiązanie jest wektorem współczynników d_i poszukiwanej funkcji.

6. Które z podanych na wykładzie metod rozwiązywania układów równań liniowych mogą być użyte do rozwiązania układu równań liniowych:

   a) Z poprzedniego zadania.

   b) Układu równań normalnych dla regularnego liniowego zadania najmniejszych kwadratów z liczbą niewiadomych nie przekraczającą 100.

   c) Układu równań z wielką macierzą ($n \times n$, gdzie $n > 10^4$) symetryczną i dodatnio określoną, która ma w każdym wierszu mniej niż 20 niezerowych współczynników rozmieszczonych nieregularnie.

   W każdym przypadku napisz, z uzasadnieniem, która z tych metod wydaje się najbardziej odpowiednia.

7. Podaj najmniejsze n, takie że błąd aproksymacji jednostajnej funkcji $f(x) = \sin x$ w przedziale $[-4\pi, 4\pi]$ przez optymalnie dobrany wielomian stopnia n jest mniejszy niż 1. Odpowiedź uzasadnij, powołując się na stosowne twierdzenie.

8. Całkę

$$I(f) = \int_{-1}^{1} f(x)\,dx,$$

chcemy przybliżać kwadraturą o postaci

$$Q(f) = A_0\big(f(-1) + f(1)\big) + A_1\big(f(-a) + f(a)\big).$$

Dobierz liczbę a i współczynniki A_0, A_1 tak, aby otrzymać kwadraturę o największym rzędzie. Podaj oszacowanie błędu tej kwadratury, jeśli funkcja f ma w przedziale $[-1, 1]$ ciągłą pochodną czwartego rzędu i istnieje stała M_4, taka że dla każdego $x \in [-1, 1]$ zachodzi nierówność $|f^{(4)}(x)| \leqslant M_4$.

Rozwiązywanie równań nieliniowych

Rozważamy zadanie znalezienia liczby x, takiej że

$$f(x) = 0,$$

przy czym mamy do dyspozycji podprogram obliczający wartość funkcji f dla argumentu x podanego jako parametr. To dla programu. Natomiast aby taki program napisać, lub wybrać gotowy do rozwiązania konkretnego zadania, zawsze musimy wiedzieć coś więcej o funkcji f. Przede wszystkim trzeba wiedzieć, czy rozwiązanie istnieje. Czy istnieje więcej niż jedno? A może nieskończenie wiele? To oczywiście zależy od funkcji f. Dalej, jeśli rozwiązań jest więcej, to czy mamy znaleźć wszystkie, kilka, czy tylko jedno, obojętnie które, albo spełniające jakiś dodatkowy warunek?

Aby wybrać algorytm, musimy wiedzieć też w jakim zbiorze ta funkcja jest określona i czy jest ciągła, przyda się też wiedza np. czy ciągła jest jej pochodna rzędu 1, 2 i być może dalsze. W niektórych metodach oprócz podprogramu obliczającego $f(x)$ będzie też potrzebny podprogram obliczający $f'(x)$, a nawet dalsze pochodne.

Metoda Newtona

Niech A oznacza ograniczony przedział domknięty, w którym jest określona funkcja rzeczywista f klasy C^2. Chcemy znaleźć miejsce zerowe funkcji f w tym przedziale (założymy, że istnieje i jest tylko jedno, w każdym praktycznym zastosowaniu to założenie oczywiście trzeba sprawdzić). Napiszemy wzór Taylora:

$$f(x+h) = \frac{f(x)}{0!} + \frac{f'(x)}{1!}h + \frac{f''(\xi)}{2!}h^2.$$

Rozumiemy go tak: jeśli liczby x oraz $x+h$ należą do przedziału A, w którym funkcja f jest klasy C^2, to istnieje liczba ξ, leżąca pomiędzy x oraz $x+h$, takia że powyższa równość zachodzi.

Metoda Newtona (często w literaturze nazywana metodą stycznych lub metodą Newtona-Raphsona, wersja współczesna różni się od wersji podanych przez nich obu) jest następująca: wybieramy liczbę x_0, która jest przybliżeniem miejsca zerowego funkcji f, a następnie konstruujemy rekurencyjnie elementy ciągu $x_1, x_2, \ldots$, w taki sposób: mając x_k, określamy wielomian

$$w_k(h) = f(x_k) + f'(x_k)h.$$

Znajdujemy miejsce zerowe δ wielomianu w_k i przyjmujemy $x_{k+1} = x_k + \delta$. Mamy stąd formułę

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

Interpretacja geometryczna jest taka: wykres funkcji f jest gładką krzywą, przechodzącą przez punkt $(x_k, f(x_k))$. Konstruujemy prostą styczną do wykresu w tym punkcie i przyjmujemy za x_{k+1} punkt przecięcia stycznej z osią x.

Zbadamy, jakie warunki wystarczy spełnić, aby ciąg $(x_k)_{k\in\mathbb{N}}$ dla dowolnego $x_0 \in A$ zbiegał do rozwiązania, które oznaczymy literą α. Przede wszystkim zauważamy, że w żadnym punkcie tego ciągu pochodna funkcji f nie może być zerowa. Naturalne jest założenie, że w przedziale A pochodna znaku nie zmienia, co więcej, zachodzi nierówność $|f'(x)| \geqslant K$ dla pewnej stałej $K > 0$. Dalej, ponieważ f jest klasy $C^2(A)$, istnieje stała M, taka że $|f''(x)| \leqslant M$ dla każdego $x \in A$.

Oznaczmy $\varepsilon_k = x_k - \alpha$ — jest to (bezwzględny) błąd aproksymacji rozwiązania przez k-ty element ciągu. Na podstawie wzoru Taylora piszemy

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

Liczba ξ_k leży między α i x_k. Dzielimy strony przez $f'(x_k)$:

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

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

$$\varepsilon_{k+1} = \frac{f''(\xi_k)}{2f'(x_k)}\varepsilon_k^2. \qquad (*)$$

Możemy oszacować

$$|\varepsilon_{k+1}| \leqslant \frac{M}{2K}|\varepsilon_k|^2.$$

Aby zachodziła nierówność $|\varepsilon_{k+1}| < |\varepsilon_k|$, wystarczy, że $\frac{M}{2K}|\varepsilon_k| < 1$, czyli

$$|\varepsilon_k| < \frac{2K}{M}.$$

Jeśli błąd przybliżenia rozwiązania przez punkt x_0, z którego zaczynamy, spełnia tę nierówność, to każdy następny błąd ma mniejszą wartość bezwzględną niż poprzedni, co więcej, ciąg błędów zbiega do zera.

Mamy zatem warunek dostateczny zbieżności metody, ale zbadajmy jeszcze szybkość tej zbieżności. Wybierzmy dowolną podstawę logarytmu większą niż 1 i oznaczmy $a_k = \log|\varepsilon_k|$, $g(k) = \log\left|\frac{f''(\xi_{k-1})}{2f'(x_{k-1})}\right|$. Na podstawie równości (*) możemy napisać równanie różnicowe

$$a_k = 2a_{k-1} + g(k).$$

Niech $G = \log\frac{M}{2K}$. Jeśli rozważymy równanie uproszczone,

$$\tilde{a}_k = 2\tilde{a}_{k-1} + G,$$

dla którego przyjmiemy $\tilde{a}_0 = a_0 < -G$, to dla każdego k mamy

$$a_k \leqslant \tilde{a}_k = (a_0 + G)\cdot 2^k - G.$$

Ciąg $(\tilde{a}_k)_{k\in\mathbb{N}}$ dąży wykładniczo do $-\infty$, a ciąg $(a_k)_{k\in\mathbb{N}}$ dąży do $-\infty$ co najmniej tak samo szybko. To zaś oznacza, że jeśli x_k jest przybliżeniem rozwiązania, które ma n cyfr dokładnych, to x_{k+1} będzie mieć w przybliżeniu $2n$ cyfr dokładnych. Zatem zbieżność jest bardzo szybka. Na podstawie powyższej nierówności, znając oszacowanie $|\varepsilon_0|$ i G oraz tolerancję błędu, można oszacować liczbę iteracji wystarczającą do otrzymania rozwiązania z błędem w granicach tej tolerancji.

Uwaga. Można udowodnić zbieżność metody przy słabszych założeniach, np. że funkcja f niekoniecznie jest klasy C^2, ale jej pochodna spełnia warunek Lipschitza.

Podstawowe pojęcia

Równania nieliniowe (i ich układy) są rozwiązywane za pomocą metod iteracyjnych — przykładem jest przedstawiona wyżej metoda Newtona. Powód jest taki, że rozwiązania na ogół nie dają się wyrazić za pomocą czterech działań algebraicznych i to dotyczy nawet równań drugiego stopnia (oczywiście, mamy do dyspozycji pierwiastek kwadratowy, ale jego też oblicza się iteracyjnie, za pomocą odpowiedniego mikroprogramu procesora). Aby mieć ogólne spojrzenie na metody iteracyjne, wprowadzimy kilka pojęć.

Funkcja iteracyjna jest to funkcja, która elementowi x_k, będącemu przybliżeniem rozwiązania, przyporządkowuje kolejne przybliżenie, x_{k+1}. W metodzie Newtona funkcja iteracyjna jest określona wzorem

$$\varphi_N(x) = x - \frac{f(x)}{f'(x)}.$$

Jest jasne, że funkcja iteracyjna powinna być tak skonstruowana, aby rozwiązanie α było jej punktem stałym, tj. aby było $\varphi(\alpha) = \alpha$.

Funkcje iteracyjne dla pewnych metod są bardziej skomplikowane. Po pierwsze, argumentem funkcji iteracyjnej oprócz ostatniego przybliżenia może być także jedno lub więcej poprzednich (czasami takie metody nazywa się metodami z pamięcią). Na przykład w metodzie siecznych, o której będzie mowa dalej, potrzebne są dwa ostatnie przybliżenia, które nie mogą być jednakowe. Funkcja iteracyjna ma w tym przypadku postać

$$\varphi_S(x,y) = x - \frac{f(x)}{f[x,y]}, \quad \text{gdzie} \quad f[x,y] = \frac{f(x)-f(y)}{x-y},$$

zatem w kolejnych iteracjach obliczamy $x_{k+1} = \varphi_S(x_k, x_{k-1})$. Wreszcie, funkcja iteracyjna może w jawny sposób zależeć od numeru iteracji, k — w tym przypadku mówimy o metodzie niestacjonarnej.

Kula zbieżności rozwiązania α jest to kula o środku α (w przypadku równań z jedną niewiadomą jest to przedział symetryczny względem α), taka że jeśli wybierzemy dowolny punkt startowy x_0 wewnątrz tej kuli, to ciąg $(x_k)_{k\in\mathbb{N}}$ zbiega do α. Znalezienie kuli zbieżności jest na ogół bardzo trudne, więc tego nie robimy, ale na podstawie własności funkcji f i definicji metody możemy szacować jej promień r. Na przykład, dla metody Newtona znaleźliśmy oszacowanie $r \geqslant \frac{2K}{M}$. Jest oczywiste, że jeśli równanie ma kilka rozwiązań, to każde z nich ma własną kulę zbieżności i wszystkie te kule są rozłączne. Kule zbieżności pewnych rozwiązań mogą być zbiorem pustym — może się zdarzyć, że dana metoda nie jest w stanie takich rozwiązań znaleźć. Warto natomiast zwrócić uwagę, że jeśli punkt startowy nie należy do kuli zbieżności żadnego rozwiązania, to metoda może znaleźć rozwiązanie, jeśli otrzymany po pewnej liczbie iteracji punkt „wpadł” do kuli zbieżności. Tylko, że *nie należy* liczyć na taki przypadek.

Twierdzeniem o ogromnym znaczeniu praktycznym, a zwłaszcza w metodach numerycznych, jest twierdzenie Banacha o punkcie stałym: *jeśli zbiór* X *z metryką* ρ *jest zupełną przestrzenią metryczną, a funkcja* $\varphi\colon X \to X$ *jest przekształceniem zwężającym (tj. istnieje stała* $L < 1$*, taka że* $\forall_{a,b\in X}\,\rho(\varphi(a),\varphi(b)) \leqslant L\rho(a,b)$*), to funkcja* φ *ma jeden punkt stały w zbiorze* X. Wykazanie, że metoda działa, tj. wytwarza ciąg zbieżny do rozwiązania, często sprowadza się do znalezienia (wykazania istnienia lub oszacowania promienia) kuli X, w której funkcja iteracyjna φ jest przekształceniem zwężającym.

Wykładnik zbieżności metody opisuje asymptotyczną szybkość zbieżności ciągu $(x_k)_{k\in\mathbb{N}}$ do rozwiązania. Przeprowadzony rachunek dla metody Newtona dowiódł, że *jeśli funkcja* f *spełnia uczynione założenia*, to wykładnik zbieżności jest nie mniejszy niż 2. Formalna definicja jest taka: wykładnik zbieżności metody

jest to największa liczba p, taka że istnieją stałe K i $C < +\infty$, takie że dla każdego $k \geqslant K$ zachodzi nierówność

$$|\varepsilon_{k+1}| \leqslant C|\varepsilon_k|^p, \quad \text{czyli} \quad \log|\varepsilon_{k+1}| \leqslant \log C + p\log|\varepsilon_k|.$$

Wykładnik zbieżności powinien być większy lub równy 1, przy czym jeśli $p = 1$, to oczywiście musi być $C < 1$. Przykładem metody o wykładniku zbieżności 1 jest metoda bisekcji: w każdej iteracji otrzymujemy przybliżenie rozwiązania z oszacowaniem błędu mniejszym o połowę. Również metoda Newtona ma wykładnik zbieżności 1, jeśli nie jest spełnione założenie, że pochodna funkcji f w otoczeniu rozwiązania jest niezerowa. Dla $p > 1$, gdyby było $K = 0$, to by oznaczało, że istnieją stałe a i b, takie że

$$\log|\varepsilon_k| \leqslant a + (\log|\varepsilon_0| + b)p^k.$$

Ostatnie przedstawiane tu pojęcie podstawowe to maksymalna graniczna dokładność. Analiza metody Newtona była przeprowadzona przy założeniu, że nie ma błędów w obliczeniach, tj. zarówno wartości funkcji f i pochodnej w x_k są obliczane dokładnie, jak i w końcowych działaniach obliczenia wartości funkcji iteracyjnej nie ma błędów. Obliczenia wykonujemy jednak z błędami zaokrągleń, które ograniczają możliwą do uzyskania dokładność rozwiązania. Wartość funkcji f jest obliczana z pewnym błędem, dalsze działania w obliczaniu funkcji iteracyjnej też są niedokładne. Za rozwiązanie metoda może zatem przyjąć dowolny punkt przedziału, w którym błąd obliczonej wartości funkcji f jest większy lub równy 100%. Jeśli pochodna funkcji jest bliska 0, to ten przedział może być długi.

Metoda siecznych

Wadą metody Newtona jest konieczność obliczania wartości pochodnej funkcji f. Metoda siecznych jest modyfikacją metody Newtona, w której pochodna została zastąpiona przez różnicę dzieloną (albo iloraz różnicowy, jak kto woli), czyli pewne przybliżenie pochodnej. Mając *dwa różne* przybliżenia rozwiązania, x_k i x_{k-1}, prowadzimy prostą przez punkty $(x_k, f(x_k))$ i $(x_{k-1}, f(x_{k-1}))$. Prosta ta przecina (siecze) wykres funkcji f w tych punktach, i w tym sensie jest jego sieczną.

Skonstruowana sieczna jest wykresem wielomianu pierwszego stopnia. Punkt x_{k+1} jest miejscem zerowym tego wielomianu. W metodzie siecznych należy podać dwa początkowe przybliżenia rozwiązania, x_0 i x_1, a następnie w każdej iteracji obliczać

$$x_{k+1} = x_k - \frac{f(x_k)}{f[x_k, x_{k-1}]}, \quad \text{gdzie} \quad f[x_k, x_{k-1}] = \frac{f(x_k) - f(x_{k-1})}{x_k - 1_{k-1}}.$$

Aby dokonać analizy metody siecznych, użyjemy pewnego uogólnienia wzoru Taylora:

$$f(z) = f(x) + f[x, y](z - x) + \frac{f''(\xi)}{2!}(z - x)(z - y).$$

Wzór ten jest szczególnym przypadkiem wzoru opisującego *resztę interpolacyjną Hermite'a*, który podam (z dowodem) na jednym z dalszych wykładów. Podany wyżej wzór rozumiemy w ten sposób, że jeśli liczby x, y, z leżą w przedziale A, w którym funkcja f jest klasy C^2, to istnieje $\xi \in A$, takie że podana wyżej równość zachodzi (liczba ξ leży między najmniejszą i największą spośród tych trzech liczb).

Jak poprzednio, α oznacza poszukiwane rozwiązanie, zaś $\varepsilon_k = x_k - \alpha$. Liczymy

$$0 = f(\alpha) = f(x_k) + f[x_k, x_{k-1}](\alpha - x_k) + \frac{f''(\xi_k)}{2}(\alpha - x_k)(\alpha - x_{k-1})$$

i dzielimy stronami przez $f[x_k, x_{k-1}]$:

$$0 = \underline{\frac{f(x_k)}{f[x_k, x_{k-1}]}} + \alpha - x_{k+1} + \underline{x_{k+1} - x_k} + \frac{f''(\xi_k)}{2f[x_k, x_{k-1}]}(\alpha - x_k)(\alpha - x_{k-1}),$$

skąd, po skróceniu podkreślonych składników, otrzymujemy

$$0 = \alpha - x_{k+1} + \frac{f''(\xi_k)}{2f[x_k, x_{k-1}]}(\alpha - x_k)(\alpha - x_{k-1}),$$

a po uporządkowaniu i uwzględnieniu faktu, że istnieje liczba η_k położona między x_k i x_{k-1}, taka że $f[x_k, x_{k-1}] = f'(\eta_k)$, mamy stąd równość

$$\varepsilon_{k+1} = \frac{f''(\xi_k)}{2f'(\eta_k)}\varepsilon_k\varepsilon_{k-1}. \qquad (**)$$

Jeśli, jak poprzednio, możemy oszacować $|f'(x)| \geqslant K > 0$ i $|f''(x)| \leqslant M$ dla każdego $x \in A$, to mamy

$$|\varepsilon_{k+1}| \leqslant \frac{M}{2K}|\varepsilon_k||\varepsilon_{k-1}|.$$

Jeśli oba błędy, ε_k i ε_{k-1}, mają wartości bezwzględne mniejsze niż $\frac{2K}{M}$, to wartości bezwzględne kolejnych błędów będą coraz mniejsze — w ten sposób mamy oszacowany promień kuli zbieżności. Zbadajmy jeszcze rząd zbieżności. W tym celu oznaczmy $a_k = \log|\varepsilon_k|$ oraz $g(k) = \log\left|\frac{f''(\xi_{k-1})}{f'(\eta_{k-1})}\right|$ i $G = \log\left|\frac{M}{2K}\right|$.
Na podstawie (**) możemy napisać równanie różnicowe drugiego rzędu,

$$a_k = a_{k-1} + a_{k-2} + g(k),$$

i jego uproszczoną wersję

$$\tilde{a}_k = \tilde{a}_{k-1} + \tilde{a}_{k-2} + G.$$

Dla ustalonych wyrazów początkowych, $\tilde{a}_0 = a_0$ i $\tilde{a}_1 = a_1$, istnieją liczby a, b, c (mniejsza o dokładne wzory, ale zachęcam do ich znalezienia dla wprawy), takie że

$$\tilde{a}_k = a\lambda_1^k + b\lambda_2^k + c, \quad \text{gdzie} \quad \lambda_1 = \frac{1-\sqrt{5}}{2},\ \lambda_2 = \frac{1+\sqrt{5}}{2},$$

i jeśli liczby a_0 i a_1 są dostatecznie małe, to $b < 0$. Składnik $b\lambda_2^k$ dominuje i wtedy ciąg $(\tilde{a}_k)_{k\in\mathbb{N}}$ zbiega wykładniczo do $-\infty$. Zachodzi też nierówność $a_k \leqslant \tilde{a}_k$ dla każdego k, w związku z czym, dla dostatecznie dużych k, jeśli przybliżenie x_k rozwiązania α ma n cyfr dokładnych, to przybliżenie x_{k+1} będzie ich miało około $\lambda_2 n$. Wykładnik zbieżności metody siecznych, $\lambda_2 \approx 1.618$, jest ułamkiem.

Metoda siecznych ma mniejszy wykładnik zbieżności niż metoda Newtona, ale jedna jej iteracja jest tańsza — odpada obliczanie pochodnej. Okazuje się, że jeśli zadamy tolerancję ε dopuszczalnego błędu, to metoda siecznych może znaleźć dostatecznie dokładne rozwiązanie szybciej (w większej liczbie iteracji, z których każda zajmuje mniej czasu). Z tego punktu widzenia, jeśli koszt obliczania różnicy dzielonej uznamy za nieistotny, to metoda Newtona jest opłacalna, gdy koszt obliczania pochodnej nie przewyższa ok. 0.44 kosztu obliczania wartości funkcji f.

Metoda Newtona dla układu równań

Rozważamy teraz zadanie znalezienia wspólnego miejsca zerowego n rzeczywistych funkcji skalarnych, których argumentami jest n zmiennych rzeczywistych. Możemy zatem napisać układ w postaci rozwiniętej:

$$\begin{cases} f_1(x_1, \dots, x_n) = 0, \\ \qquad\qquad \vdots \\ f_n(x_1, \dots, x_n) = 0, \end{cases}$$

lub „zwiniętej”

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}.$$

Funkcja $\mathbf{f}$ jest określona w pewnym obszarze A przestrzeni $\mathbb{R}^n$ i ma wartości w $\mathbb{R}^n$.

Niech $\mathbf{h} = (h_1, \dots, h_n)$. Dla funkcji *skalarnej* f_i klasy $C^2(A)$, możemy napisać wzór Taylora:

$$f_i(\mathbf{x}+\mathbf{h}) = \frac{1}{0!}f_i(\mathbf{x}) + \frac{1}{1!}Df_i|_{\mathbf{x}}(\mathbf{h}) + \frac{1}{2!}D^2f_i|_{\xi_i}(\mathbf{h},\mathbf{h}).$$

Rozumiemy go tak: jeśli obszar A zawiera odcinek o końcach $\mathbf{x}$ i $\mathbf{x}+\mathbf{h}$, to istnieje punkt ξ_i na tym odcinku, taki że powyższa równość zachodzi. Symbol $Df_i|_{\mathbf{x}}$ oznacza <u>różniczkę</u> funkcji f_i w punkcie $\mathbf{x}$, czyli przekształcenie liniowe, które dowolnemu wektorowi $\mathbf{h}$ przyporządkowuje liczbę

$$Df_i|_{\mathbf{x}}(\mathbf{h}) = \frac{\partial f_i}{\partial x_1}\Big|_{\mathbf{x}} h_1 + \dots + \frac{\partial f_i}{\partial x_n}\Big|_{\mathbf{x}} h_n.$$

Wartością tego przekształcenia jest zatem iloczyn skalarny gradientu funkcji f_i w punkcie $\mathbf{x}$ i wektora $\mathbf{h}$. Symbol $D^2f_i|_{\xi_i}$ oznacza <u>różniczkę drugiego rzędu</u>, tj. przekształcenie *dwuliniowe*, którego wartością dla pary wektorów $(\mathbf{g},\mathbf{h})$ jest liczba

$$D^2f_i|_{\xi_i}(\mathbf{g},\mathbf{h}) = \sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^2 f_i}{\partial x_j \partial x_k}\Big|_{\xi_i} g_j h_k.$$

Drobny kłopot (o którym *nie należy* zapominać) jest taki, że punkt ξ_i dla każdego i może być różny, dlatego nie można tak prosto zapisać odpowiedniego wzoru dla funkcji wektorowej $\mathbf{f}$. Niemniej, ze wzoru Taylora wynika, że jeśli obszar A zawiera odcinek $\overline{\mathbf{x},\mathbf{x}+\mathbf{h}}$, to dla wektorowej funkcji $\mathbf{f}$ klasy $C^2(A)$ zachodzi równość

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) = \mathbf{f}(\mathbf{x}) + Df|_{\mathbf{x}}(\mathbf{h}) + \mathbf{r}, \qquad (**)$$

przy czym $Df|_{\mathbf{x}}$ jest różniczką przekształcenia $\mathbf{f}$ w punkcie $\mathbf{x}$, a ponadto istnieje macierz B (zależna od $\mathbf{x}$ i $\mathbf{h}$) o wymiarach $n \times n$ i współczynnikach *wektorowych* $\mathbf{b}_{jk} = \left[\frac{\partial^2 f_1}{\partial x_j \partial x_k}\Big|_{\xi_1}, \dots, \frac{\partial^2 f_n}{\partial x_j \partial x_k}\Big|_{\xi_n}\right]^T \in \mathbb{R}^n$, taka że reszta we wzorze $(**)$ jest równa

$$\mathbf{r} = \mathbf{h}^T B \mathbf{h} = \sum_{j=1}^{n}\sum_{k=1}^{n} \mathbf{b}_{jk} h_j h_k, \qquad (***)$$

i spełnia oszacowanie

$$\|\mathbf{r}\| \leqslant \frac{M}{2}\|\mathbf{h}\|^2$$

dla pewnej stałej M (stała ta jest określona przez pochodne drugiego rzędu funkcji f_i w obszarze A i przez używaną normę).

<u>Metoda Newtona</u> polega tym, że mając przybliżenie $\mathbf{x}_k$ rozwiązania $\boldsymbol{\alpha}$, konstruujemy przekształcenie afiniczne $\mathbb{R}^n \to \mathbb{R}^n$, określone przez pierwsze dwa składniki po prawej stronie wzoru $(**)$, a następnie przyjmujemy za $\mathbf{x}_{k+1}$ miejsce zerowe tego przekształcenia. Czyli

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (Df|_{\mathbf{x}_k})^{-1}\big(\mathbf{f}(\mathbf{x}_k)\big).$$

Aby obliczyć $\mathbf{x}_{k+1}$, należy obliczyć wektor $\mathbf{f}_k = \mathbf{f}(\mathbf{x}_k)$ oraz macierz pochodnych cząstkowych pierwszego rzędu

$$J_k = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}\big|_{\mathbf{x}_k} & \cdots & \frac{\partial f_1}{\partial x_n}\big|_{\mathbf{x}_k} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}\big|_{\mathbf{x}_k} & \cdots & \frac{\partial f_n}{\partial x_n}\big|_{\mathbf{x}_k} \end{bmatrix}$$

zwaną jakobianem, która reprezentuje różniczkę funkcji $\mathbf{f}$ w punkcie $\mathbf{x}_k$, a następnie rozwiązać układ równań liniowych

$$J_k\boldsymbol{\delta} = -\mathbf{f}_k$$

i obliczyć $\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}$.

Oczywiście, aby to obliczenie było wykonalne, macierz J_k musi być nieosobliwa. Przyjmiemy założenie, że istnieje taka stała K, że dla każdego punktu $\mathbf{x}$ w rozpatrywanym obszarze A różniczka przekształcenia $\mathbf{f}$ spełnia warunek $\|(D\mathbf{f}|_{\mathbf{x}})^{-1}\| \leqslant K^{-1}$. Zatem, dla $\mathbf{x}_k \in A$ jest $\|J_k^{-1}\| \leqslant K^{-1}$. Na podstawie wzorów $(*_*^*)$ i $(**)$, mamy

$$\mathbf{0} = \mathbf{f}(\boldsymbol{\alpha}) = \mathbf{f}(\mathbf{x}_k) + J_k(\boldsymbol{\alpha} - \mathbf{x}_k) + (\boldsymbol{\alpha} - \mathbf{x}_k)^T B_k(\boldsymbol{\alpha} - \mathbf{x}_k),$$

Dalej postępujemy identycznie, jak w przypadku skalarnym. Oznaczamy $\boldsymbol{\varepsilon}_k = \mathbf{x}_k - \boldsymbol{\alpha}$. Strony równości mnożymy przez J_k^{-1}, oraz odejmujemy i dodajemy $\mathbf{x}_{k+1}$:

$$\mathbf{0} = \underline{J_k^{-1}\mathbf{f}(\mathbf{x}_k)} + \boldsymbol{\alpha} - \mathbf{x}_{k+1} + \underline{\mathbf{x}_{k+1} - \mathbf{x}_k} + J_k^{-1}\boldsymbol{\varepsilon}_k^T B_k\boldsymbol{\varepsilon}_k = \boldsymbol{\alpha} - \mathbf{x}_{k+1} + J_k^{-1}\boldsymbol{\varepsilon}_k^T B_k\boldsymbol{\varepsilon}_k.$$

Stąd wielkość błędu kolejnego przybliżenia rozwiązania,

$$\boldsymbol{\varepsilon}_{k+1} = J_k^{-1}\boldsymbol{\varepsilon}_k^T B_k\boldsymbol{\varepsilon}_k,$$

możemy oszacować tak:

$$\|\boldsymbol{\varepsilon}_{k+1}\| \leqslant \frac{M}{2K}\|\boldsymbol{\varepsilon}_k\|^2.$$

Jeśli funkcja $\mathbf{f}$ spełnia przyjęte założenia, to wykładnik zbieżności metody Newtona jest równy 2 — końcowy rachunek (z rozwiązywaniem równania różnicowego) jest identyczny jak dla równania z jedną niewiadomą.

Modyfikacje

Metoda Newtona dla układu równań może być dość kosztowna: oprócz wartości funkcji $\mathbf{f}$, składającej się z n liczb, trzeba obliczyć macierz J_k, tj. w ogólności

n^2 liczb, a następnie rozwiązać układ równań, co może wymagać wykonania $\Theta(n^3)$ działań zmiennopozycyjnych. Ze wzrostem liczby równań i niewiadomych koszty te mogą stać się zaporowe. Dla bardzo dużych n często macierz J_k jest *rzadka*, tj. ma znacznie mniej niż n^2 współczynników niezerowych. W takim przypadku należy po pierwsze obliczać tylko współczynniki niezerowe (ich rozmieszczenie w macierzy należy wyznaczyć zawczasu), a ponadto użyć metody rozwiązywania układu równań liniowych dostosowanej do macierzy rzadkiej.

Często stosuje się rozmaite modyfikacje metody Newtona. Po pierwsze, zamiast obliczać współczynniki macierzy J_k na podstawie dokładnych wzorów, które mogą być znacznie bardziej skomplikowane (czyli droższe) niż wzory opisujące funkcje f_i, można obliczać różnice dzielone; w tym celu trzeba obliczyć wartości funkcji $\mathbf{f}$ w $n+1$ punktach.

Jeśli punkty $\mathbf{x}_{k-n}, \ldots, \mathbf{x}_k$ są w położeniu ogólnym, tj. wektory $\mathbf{x}_j - \mathbf{x}_k$ dla $j = k-n, \ldots, k-1$ są liniowo niezależne, to można obliczyć przybliżenie $\tilde{J}_k$ macierzy J_k na podstawie wartości funkcji $\mathbf{f}$ w tych punktach (to jednak jest dość kosztowne, w ogólności trzeba wykonać $\Theta(n^3)$ operacji, choć istnieją pewne sposoby zmniejszania tego kosztu). To jest wielowymiarowa metoda siecznych. Jej wadą jest bardzo mały wykładnik zbieżności (bliski 1) dla dużych n.

Kolejna modyfikacja polega na wykorzystaniu macierzy J_k w kilku kolejnych iteracjach. To również obniża wykładnik zbieżności, ale dodatkowe iteracje z tą samą macierzą są bardzo tanie: nie trzeba obliczać pochodnych i można skorzystać z „gotowych" czynników (np. trójkątnych) rozkładu macierzy. Koszt rzędu n^3 w rozwiązywaniu układów równań liniowych jest związany z rozkładaniem macierzy na te czynniki, mając je, można rozwiązać układ kosztem $\Theta(n^2)$ działań.

Istnieją modyfikacje metody Newtona, mające na celu „powiększenie" kuli zbieżności poszukiwanych rozwiązań. Dla nie dość dobrego punktu $\mathbf{x}_k$ często zdarza się, że przyrost $\boldsymbol{\delta}$, otrzymany przez rozwiązanie układu równań $J_k\boldsymbol{\delta} = -\mathbf{f}_k$ jest za duży. Wtedy można przyjąć $\mathbf{x}_{k+1} = \mathbf{x}_k + \beta\boldsymbol{\delta}$, dla odpowiednio wybranego parametru $\beta \in (0, 1)$. Metoda skuteczniejsza, choć bardziej kosztowna, polega na wyznaczeniu przyrostu przez rozwiązanie układu równań

$$(J_k + \lambda I)\boldsymbol{\delta} = -\mathbf{f}_k,$$

z odpowiednio wybranym parametrem λ. Metoda ta może być też skuteczna w pewnych przypadkach, gdy macierz J_k jest osobliwa. Parametr λ dobieramy tak, aby otrzymać jak najmniejsze residuum układu, tj. aby zminimalizować normę wektora $\mathbf{f}_{k+1}$. Po pewnej liczbie iteracji możemy otrzymać przybliżenie

rozwiązania należące do kuli zbieżności metody Newtona i od tej chwili przyjmować $\lambda = 0$.

Kryteria stopu

Ważną decyzją w obliczeniach jest, kiedy je przerwać. Na przykład wykonywanie kolejnych iteracji po osiągnięciu maksymalnej granicznej dokładności jest stratą czasu. Dlatego w pętli, realizującej iteracje, musi się pojawić jedna lub więcej instrukcji przerywających obliczenia po spełnieniu pewnego warunku.

Po pierwsze, można dać limit liczby iteracji, np. określony przez parametr procedury. W *wielu* typowych zastosowaniach, jeśli metoda Newtona nie znalazła rozwiązania (z graniczną dokładnością) po siedmiu[1] iteracjach, to już nie znajdzie (bo funkcja nie spełnia warunków koniecznych działania metody, zaczęliśmy od złego przybliżenia startowego, lub w ogóle nie ma rozwiązania).

Drugie kryterium stopu jest residualne. Residuum równania w punkcie x_k jest to liczba $f(x_k)$ (lub wektor $\mathbf{f}(\mathbf{x}_k)$). Jeśli residuum ma dostatecznie małą wartość bezwzględną (lub normę, dla układu równań), na przykład porównywalną z oszacowaniem błędu, z jakim obliczamy wartości funkcji f, to przerywamy obliczenia.

Wreszcie jest kryterium przyrostowe. Obliczenia przerywamy, gdy wartość bezwzględna (lub norma) przyrostu $\delta = x_{k+1} - x_k$ jest mniejsza niż pewna wielkość progowa. Dla wielu metod długość przyrostu w danym kroku jest górnym oszacowaniem błędu rozwiazania przybliżonego x_k (ale to zależy także od funkcji f).

[1] Proszę nie sugerować się, że zawsze taki limit jest dobry!

Zadania i problemy

1. Wykaż, że algorytm

   ```
   w = a[n];
   p = 0.0;
   for ( i = n-1; i >= 0; i-- ) {
     p = p*x + w;
     w = w*x + a[i];
   }
   ```

   dla $n > 0$ oblicza wartość wielomianu $w(x) = a_n x^n + \cdots + a_1 x + a_0$ i jego pochodnej w punkcie x.
2. Znajdź wzór opisujący funkcję iteracyjną dla obliczania pierwiastka stopnia n z liczby rzeczywistej a przy użyciu metody Newtona. Oblicz pierwsze trzy przybliżenia liczby $\sqrt{2}$, jeśli $x_0 = 1$.
3. Znajdź wzór opisujący funkcję iteracyjną dla metody Newtona zastosowanej do funkcji $f(x) = a - 1/x$. Czy można obliczyć iloraz $x = a/b$ nie wykonując operacji dzielenia?
4. Niech $f(x) = (x - a)^n$, dla pewnego $n > 1$. Jak działa metoda Newtona w tym przypadku (nie znamy a, startujemy z $x_0 \neq a$)? Jak działa metoda zmodyfikowana (zakładamy, że znamy n), określona wzorem

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

5. Dla funkcji f jak w poprzednim zadaniu określamy funkcję $g(x) = f(x)/f'(x)$. Jak działa metoda Newtona, zastosowana do funkcji g?

   Napisz funkcję iteracyjną dla tego przypadku wzorem, w którym występują tylko wartości i pochodne funkcji f.
6. Przypuśćmy, że pochodna funkcji f klasy C^3 jest niezerowa. Określamy funkcję g wzorem $g(x) = f(x)/\sqrt{f'(x)}$. Oblicz pochodną drugiego rzędu funkcji g w punkcie α, takim że $f(\alpha) = 0$. Co można stąd wywnioskować o zbieżności metody Newtona zastosowanej do funkcji g?

   Zastąpienie funkcji f przez określoną wyżej funkcję g daje metodę zwaną metodą Halleya (tak, tego, który odkrył kometę). Napisz wzór opisujący funkcję iteracyjną dla metody Halleya.
7. Zbadaj, jak zachowuje się metoda Newtona, jeśli funkcja f, której miejsce zerowe należy znaleźć, jest określona wzorem

   $$f(x) = \operatorname{sgn} x\sqrt{|x|}$$

   dla dowolnego punktu startowego $x_0 \neq 0$.

8. Przypuśćmy, że w metodzie siecznych początkowe przybliżenia rozwiązania są końcami przedziału, w którym funkcja f zmienia znak. Metodę modyfikujemy w ten sposób, że po wyznaczeniu kolejnego przybliżenia odrzucamy to, w którym funkcja f ma ten sam znak, co w nowym punkcie (ta metoda nazywa się regula falsi). Jaki jest wykładnik zbieżności tej metody, jeśli funkcja f jest ściśle wypukła albo ściśle wklęsła?
9. (zadanie z książki Kincaida i Cheneya) Wykonaj jeden lub dwa kroki metody Newtona dla układów równań

$$\begin{cases} xy - z^2 = 1 \\ xyz - x^2 + y^2 = 2 \\ e^x - e^y + z = 3 \end{cases}, \qquad \mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\begin{cases} 4x^2 - y^2 = 0 \\ 4xy - x = 1 \end{cases}, \qquad \mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$\begin{cases} xy^2 + x^2y + x^4 = 3 \\ x^3y^5 - 2x^5y - x^2 = -2 \end{cases}, \qquad \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Programy i projekty

1. Podany niżej program zawiera implementacje trzech metod rozwiązywania równań nieliniowych. Przetestuj działanie tych metod, wybierając różne funkcje (np. wielomiany) i różne początkowe przybliżenia rozwiązania. Typ float w makrodefinicji FLOAT zamień na double i powtórz eksperymenty.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define FLOAT float

FLOAT f1 ( void *usrptr, FLOAT x )
{
  return sin ( x ) - 0.5*x; /* funkcja */
} /*f1*/

void fd1 ( void *usrptr, FLOAT x, FLOAT *f, FLOAT *df )
{
  *f = sin ( x ) - 0.5*x; /* funkcja */
  *df = cos ( x ) - 0.5; /* pochodna */
  return;
} /*fd1*/

void Bisekcja ( FLOAT (*f)(void *usrptr, FLOAT x), void *usrptr,
                FLOAT a, FLOAT b, FLOAT eps,
                FLOAT *x, char *error )
{
  FLOAT c, fa, fb, fc;

  fa = f ( usrptr, a );
  fb = f ( usrptr, b );
  if ( fa*fb <= 0.0 && eps > 0.0 ) {
    while ( fabs(b-a) > eps ) {
      c = 0.5*(a+b);
      fc = f ( usrptr, c );
      if ( fa*fc <= 0.0 ) b = c;
                     else a = c;
    }
    *x = a;
    *error = 0;
```

```
  }
  else
    *error = 1;
} /*Bisekcja*/

void Newton ( void (*fd)(void *usrptr, FLOAT x,
                         FLOAT *f, FLOAT *fd), void *usrptr,
              FLOAT x0, FLOAT eps, FLOAT delta, int maxit,
              FLOAT *x, char *error )
{
  FLOAT f0, d0, dx;
  int   i;

  if ( eps > 0.0 || delta > 0.0 ) {
    for ( i = 0; i < maxit; i++ ) {
      fd ( usrptr, x0, &f0, &d0 );
      dx = f0/d0;
      if ( fabs (f0) < delta || fabs(dx) < eps ) {
        *x = x0;
        *error = 0;
        return;
      }
      x0 -= dx;
    }
  }
  *error = 1;
} /*Newton*/

void Secant ( FLOAT (*f)(void *usrptr, FLOAT x), void *usrptr,
              FLOAT a, FLOAT b, FLOAT eps, FLOAT delta, int maxit,
              FLOAT *x, char *error )
{
  FLOAT c, fa, fb, fc;
  int   i;

  if ( eps > 0.0 || delta > 0.0 ) {
    fa = f ( usrptr, a );
    fb = f ( usrptr, b );
    for ( i = 0; i < maxit; i++ ) {
      c = a - (b-a)/(fb-fa)*fa;
```



```
      fc = f ( usrptr, c );
      if ( fabs(fc) < delta || fabs(c-b) < eps ) {
        *x = c;
        *error = 0;
        return;
      }
      a = b;  fa = fb;
      b = c;  fb = fc;
    }
  }
  *error = 1;
} /*Secant*/

int main ( void )
{
  FLOAT x;
  char  error;

  Bisekcja ( f1, NULL, 1.0, 3.0, 1.0e-6, &x, &error );
  if ( error )
    printf ( "Blad\n" );
  else
    printf ( "rozwiazanie x = %f\n", x );
  Newton ( fd1, NULL, 1.0, 1.0e-6, 1.0e-6, 20, &x, &error );
  if ( error )
    printf ( "Blad\n" );
  else
    printf ( "rozwiazanie x = %f\n", x );
  Secant ( f1, NULL, 1.0, 3.0, 1.0e-6, 1.0e-6, 20, &x, &error );
  if ( error )
    printf ( "Blad\n" );
  else
    printf ( "rozwiazanie x = %f\n", x );
  exit ( 0 );
} /*main*/
```

2. Przeprowadź eksperymenty z metodą Newtona dla kilku różnych układów równań, za pomocą poniższego programu. Porównaj wyniki eksperymentów z użyciem liczb zmiennopozycyjnych pojedynczej i podwójnej precyzji (wystarczy odkomentować linię z makrodefinicją `__DOUBLE__`).

   Implementacja metody korzysta z procedury `sgesv_` lub `dgesv_` z biblioteki LAPACK, w celu rozwiązania układów równań liniowych tworzonych przez metodę Newtona. Makrodefinicja `IND` zapewnia dostęp do współczynników macierzy, przechowywanych w tablicy jednowymiarowej; tu jest stosowana konwencja z języka FORTRAN, w której współczynniki macierzy są przechowywane *kolumnami*, choć wiersze i kolumny indeksujemy od zera (jak w C), a nie od jedynki (jak w FORTRANie).

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>

#include "f2c.h"
#include "clapack.h"

/*#define __DOUBLE__*/
#ifndef __DOUBLE__
#define FLOAT float
#define GESV sgesv_
#else
#define FLOAT double
#define GESV dgesv_
#endif

#define IND(n,i,j) ((n)*(j)+(i))

#define EKSPERYMENT

char SysNewton ( long int n,
                 void (*fd)(long int n, void *usrptr,
                            FLOAT *x, FLOAT *f, FLOAT *Df),
                 void *usrptr, FLOAT eps, FLOAT delta, int maxit,
                 FLOAT *x )
{
  int  i, j;
```



```
  FLOAT *f, *Df, res, dx;
  long int  *permut, one = 1, info;

  f = malloc ( n*(n+1)*sizeof(FLOAT) + n*sizeof(long int) );
  if ( !f )
    return 0;
  Df = &f[n];
  permut = (long int*)&Df[n*n];
  for ( i = 0; i < maxit; i++ ) {
#ifdef EKSPERYMENT
    printf ( "%2d: ", i );
    for ( j = 0; j < n; j++ )
      printf ( "%12.9f,", x[j] );
/*   printf ( "\n" );*/
#endif
       /* oblicz wartosc funkcji i pochodna */
    fd ( n, usrptr, x, f, Df );
#ifdef EKSPERYMENT
    printf ( "  " );
    for ( j = 0; j < n; j++ )
      printf ( "%12.9f,", f[j] );
    printf ( "\n" );
#endif
       /* residualne kryterium stopu */
    for ( res = 0.0, j = 0; j < n; j++ )
      res += f[j]*f[j];
    if ( res <= delta*delta )
      goto sukces;
       /* rozwiaz uklad rownan Df*dx = -f i oblicz nowy punkt */
    info = 0;
    GESV ( &n, &one, Df, &n, permut, f, &n, &info );
    if ( info ) {
      printf ( "GESV info: %ld\n", info );
      goto klops;
    }
    for ( j = 0; j < n; j++ )
      x[j] -= f[j];
       /* przyrostowe kryterium stopu */
    for ( dx = 0.0, j = 0; j < n; j++ )
      dx += f[j]*f[j];
```

```
    if ( dx < eps*eps )
      goto sukces;
  }

klops:
  free ( f );
  return 0;

sukces:
  free ( f );
  return 1;
} /*SysNewton*/

#define NN 3
void TestFD ( long int n, void *usrptr,
              FLOAT *x, FLOAT *f, FLOAT *Df )
{
        /* n == NN == 3 */
  f[0] = x[0] + x[1] + x[2] - 3.0;
  f[1] = x[0]*x[0] + x[1]*x[1] + x[2]*x[2] - 17.0;
  f[2] = x[0]*x[1]*x[2] + 12.0;
  Df[IND(NN,0,0)] = 1.0;
  Df[IND(NN,0,1)] = 1.0;
  Df[IND(NN,0,2)] = 1.0;
  Df[IND(NN,1,0)] = 2.0*x[0];
  Df[IND(NN,1,1)] = 2.0*x[1];
  Df[IND(NN,1,2)] = 2.0*x[2];
  Df[IND(NN,2,0)] = x[1]*x[2];
  Df[IND(NN,2,1)] = x[0]*x[2];
  Df[IND(NN,2,2)] = x[0]*x[1];
} /*TestFD*/

void TestSysNewton ( void )
{
  FLOAT x[NN] = { 1.0, 2.0, -2.0 };
/* inny punkt startowy */
/*  FLOAT x[NN] = { 1.0, 2.0, -1.0 }; */
  int  i;

  printf ( "Test metody Newtona\n" );
```



```
  if ( SysNewton ( NN, TestFD, NULL, 1.0e-6, 1.0e-6, 20, x ) ) {
    for ( i = 0; i < NN; i++ )
      printf ( "%f,", x[i] );
    printf ( "\n" );
  }
  else
    printf ( "klops\n" );
} /*TestSysNewton*/
#undef NN

int main ( void )
{
  TestSysNewton ();
  exit ( 0 );
} /*main*/
```

3. Metoda Bairstowa

Metoda Bairstowa ma na celu rozłożenie wielomianu rzeczywistego w na iloczyn trójmianów kwadratowych (o współczynnikach rzeczywistych). Pierwiastki tego wielomianu są pierwiastkami tych trójmianów, przy czym ich obliczenie przez rozwiązywanie równań kwadratowych jest znacznie wygodniejsze niż rozwiązywanie bezpośrednio równania algebraicznego $w(x) = 0$.

Niech $w(x) = \sum_{i=0}^{n} a_i x^i$ i niech $t(x) = x^2 + cx + d$. Dzielenie z resztą wielomianu w przez trójmian t można wykonać następująco. Mamy

$$w(x) = z(x)t(x) + r(x),$$

gdzie $z(x) = z_0 + z_1 x + \cdots + z_{n-2}x^{n-2}$, $r(x) = px + q$, przy czym, jeśli przyjmiemy $z_n = z_{n-1} = 0$, to dla $k = 2, \ldots, n$ zachodzą równości

$$a_k = z_k d + z_{k-1} c + z_{k-2},$$

a ponadto mamy

$$a_1 = z_1 d + z_0 c + p,$$
$$a_0 = z_0 d + q.$$

Zatem, dla ustalonych liczb c, d, możemy obliczyć kolejno $z_{n-2}, \ldots, z_0$, a następnie, z ostatnich dwóch równań p i q. Określamy funkcję $\mathbf{f}\colon \mathbb{R}^2 \to \mathbb{R}^2$, której argumentami są liczby c i d, a wektorowe wartości mają współrzędne p i q. Znalezienie miejsca zerowego funkcji $\mathbf{f}$ jest równoznaczne ze znalezieniem trójmianu kwadratowego t, przez który wielomian w dzieli się bez reszty. Metoda Bairstowa polega na zastosowaniu metody Newtona do funkcji $\mathbf{f}$.

Deflacja polega na znalezieniu kolejnego trójmianu dzielącego wielomian w przez zastosowanie metody Bairstowa do wielomianu z. Kolejne trójmiany są znajdowane mniejszym kosztem. Obliczenia kończymy po otrzymaniu wielomianu stopnia mniejszego niż 3.

Projekt. Znajdź odpowiednie wzory, napisz algorytm obliczania wartości funkcji $\mathbf{f}$ i jej pochodnej (tj. macierzy pochodnych cząstkowych) i zastosuj ten algorytm w implementacji metody Bairstowa. Możesz do tego celu wykorzystać procedury podane wcześniej.

Arytmetyka zmiennopozycyjna

Liczb rzeczywistych jest nieskończenie (a nawet nieprzeliczalnie) wiele, a pamięć choćby największego komputera jest skończona. Dlatego w obliczeniach numerycznych musimy się zadowolić poruszaniem się w pewnym skończonym zbiorze, którego elementy tylko przybliżają wszelkie liczby rzeczywiste, jakie mogłyby się pojawić w tych obliczeniach.

Rzędy wielkości tych liczb mogą być różne i błędy ich przybliżenia też mogą być różne. Zwykle im większa liczba, tym większy jej błąd nam nie przeszkadza. Na przykład, jeśli dowiemy się, że jakiś obiekt ma długość 147 km, to informację, że w rzeczywistości ma o 1 mm mniej, jesteśmy skłonni zignorować. Co innego, jeśli obiekt ma tylko 0.5 mm długości — błąd rzędu 1 mm jesteśmy wtedy skłonni potraktować z całą powagą i stanowczością.

Naturalne jest zatem używanie takiego sposobu reprezentowania liczb, który umożliwia przybliżanie tych liczb rzeczywistych, które nie mają dokładnej reprezentacji, z małym błędem względnym. Na przykład, jeśli długość 147 km jest podana z błędem nie większym niż 0.1%, to wiemy, że błąd bezwzględny jest mniejszy niż 150 m, i taka dokładność nieraz nam wystarczy.

Reprezentacja zmiennopozycyjna

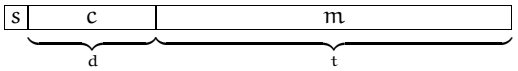
Powszechnie używana reprezentacja zmiennopozycyjna liczb rzeczywistych jest kompromisem między dokładnością i złożonością czasową i pamięciową. Jej głównym celem jest masowe przetwarzanie liczb, czemu służy stosunkowo mała ilość miejsca zajmowanego przez tę reprezentację i możliwość szybkiego wykonywania działań przez specjalnie opracowane w tym celu podukłady procesorów. Błędy tej reprezentacji są dostatecznie małe na potrzeby znakomitej większości zastosowań. Istnieją inne reprezentacje, umożliwiające prowadzenie obliczeń ze znacznie większą dokładnością, ale znacznie wolniej i w większej pamięci. Te inne reprezentacje są poza zakresem tego wykładu. Jeszcze jedno: reprezentacje zmiennopozycyjne mają powszechnie przyjęty standard, który ułatwia m.in. wymianę danych. Reprezentacje niestandardowe tak fajnie nie mają.

Idea reprezentacji zmiennopozycyjnej wiąże się z tzw. półlogarytmicznym zapisem liczby. Każdą dodatnią liczbę rzeczywistą możemy przedstawić za pomocą liczby z przedziału $[1,10)$ i całkowitej potęgi liczby 10, na przykład

$$27182818 = 2.7182818 \cdot 10^7.$$

W komputerach zamiast liczby 10 i dziesięciu różnych cyfr, wygodniej jest używać liczby 2 i bitów.

Podstawowa reprezentacja określona przez standard IEEE-754 (opracowany w 1985 r.) składa się z bitu znaku, s, po którym następuje cecha c i mantysa m:



Mantysa jest liczbą rzeczywistą; jeśli reprezentuje ją ciąg bitów $b_{t-1}b_{t-2}\ldots b_1b_0$, to $m = \sum_{k=0}^{t-1} b_k 2^{k-t}$, a zatem zawsze $0 \leqslant m < 1$. Cecha jest liczbą całkowitą (bez znaku), reprezentowaną za pomocą d bitów, która wpływa na sposób interpretacji całego ciągu bitów. Liczba reprezentowana przez taki ciąg, w zależności od cechy, jest równa

$$\begin{aligned}
x &= (-1)^s 2^{c-b}(1+m) && \text{dla } 0 < c < 2^d - 1,\\
x &= (-1)^s 2^{1-b} m && \text{dla } c = 0,\\
x &= (-1)^s \infty && \text{dla } c = 2^d - 1,\ m = 0,\\
x &= \mathrm{NaN}\ (\text{„nie-liczba”}) && \text{dla } c = 2^d - 1,\ m \neq 0.
\end{aligned}$$

Liczby d, t i b są ustalone dla konkretnej reprezentacji. Cechą charakterystyczną reprezentacji z użyciem pierwszego wzoru jest tzw. normalizacja. Mając dowolną liczbę rzeczywistą $x \neq 0$, przedstawioną w układzie dwójkowym, dobieramy cechę c (czyli równoważnie czynnik 2^{c-b}) tak, że czynnik $(1+m)$ w wyrażeniu opisującym x jest liczbą z przedziału $[1,2)$. Jeśli otrzymana w ten sposób cecha jest za duża (większa lub równa $2^d - 1$), to mamy nadmiar zmiennopozycyjny (ang. *floating point overflow*), czyli niewykonalne zadanie reprezentowania liczby o za dużej wartości bezwzględnej, zwykle będące powodem do przerwania obliczeń. Jeśli nie ma nadmiaru, to pierwszy wzór opisuje liczbę w ten sposób, że najbardziej znacząca jedynka w rozwinięciu dwójkowym nie jest jawnie pamiętana — właśnie to jest normalizacja. Dzięki niej każdy ciąg bitów reprezentuje inną liczbę, co m.in. umożliwia optymalne wykorzystanie bitów do zmniejszenia błędów.

Niech x oznacza dowolną liczbę rzeczywistą. Jej reprezentację, tj. położoną najbliżej niej liczbę zmiennopozycyjną, oznaczymy symbolem $\mathrm{rd}(x)$ (z ang. *rounding*). Jeśli liczbę x możemy przedstawić w postaci

$$x = (-1)^s 2^{c-b}(1+f),$$

dobierając cechę c tak, aby mieć $f \in [0,1)$ oraz $0 < c < 2^{d-1}$, to (z jednym rzadkim wyjątkiem, gdy f trzeba zaokrąglić w górę do jedynki) będziemy mieli

$$\mathrm{rd}(x) = (-1)^s 2^{c-b}(1+m),$$

przy czym $|f - m| \leqslant 2^{-t-1}$. Błąd względny reprezentacji spełnia nierówność

$$\frac{|x - \mathrm{rd}(x)|}{|x|} = \frac{|(-1)^s 2^{c-b}(1+f) - (-1)^s 2^{c-b}(1+m)|}{|(-1)^s 2^{c-b}(1+f)|} \leqslant |f - m| \leqslant 2^{-t-1}.$$

Co ciekawe, nierówność ta jest spełniona też w specjalnym przypadku wspomnianym wcześniej (bo w mianowniku $1 + f \approx 2$). Zatem, maksymalny błąd względny reprezentacji zmiennopozycyjnej, jeśli nie ma niedomiaru ani nadmiaru, jest na poziomie 2^{-t-1}, gdzie t jest liczbą bitów mantysy. Jeśli kierunek zaokrąglania wybieramy mniej starannie (np. zawsze obcinamy w kierunku zera), to błąd względny może być dwa razy większy, czyli rzędu $\nu = 2^{-t}$.

Bardziej skomplikowana sytuacja zdarza się w przypadku, gdy cecha jest za mała (tj. gdy w pierwszym wzorze należałoby przyjąć $c \leqslant 0$). Wtedy korzystamy z drugiego wzoru, w którym występuje czynnik m (przypominam, że $m \in [0, 1)$). Jeśli $c = m = 0$, to mamy reprezentację zera; liczba 0 jako jedyna ma dwie reprezentacje, różniące się bitem znaku. Jeśli $c = 0$ i $m \neq 0$, to mamy do czynienia z niedomiarem zmiennopozycyjnym, czyli reprezentowaniem liczby x za pomocą mantysy o mniejszej liczbie bitów istotnych (jeśli w użyciu jest pierwszy wzór, to istotne są wszystkie bity mantysy, jeśli drugi, to tylko bity od pozycji najmniej znaczącej, do najbardziej znaczącej pozycji, na której jest jedynka). Najdokładniejszą reprezentacją liczb o bardzo małej wartości bezwzględnej (mniejszej niż 2^{-b-t}) jest 0. Niedomiar wiąże się zatem ze (stopniową) utratą dokładności reprezentacji. Dla $x \to 0$ błąd względny reprezentacji dąży do 100%, a błąd bezwzględny jest ograniczony. W analizie błędów najczęściej nie bierzemy tego przypadku pod uwagę.

Reprezentacja umożliwia używanie nieskończoności, także w rachunkach (np. wynik dzielenia dowolnej liczby przez nieskończoność jest równy 0).

Nie-liczby są wykorzystywane do sygnalizowania błędów, np. próby obliczenia pierwiastka kwadratowego z liczby ujemnej. Można je też wykorzystać do odpluskwiania programu, np. nadając zmiennym takie wartości początkowe, a następnie śledząc, czy nie ma do nich odwołań przed przypisaniem właściwej wartości liczbowej.

W standardzie IEEE-754 są zdefiniowane formaty liczb pojedynczej i podwójnej precyzji, a także liczb pojedynczej i podwójnej rozszerzonej precyzji. Liczby pojedynczej rozszerzonej precyzji jakoś się nie przyjęły, procesory w komputerach PC ich nie obsługują. Dane na temat standardowych formatów są w tabelce:

	B	d	t	b	M	S	ν	μ
pojedyncza, `float`	32	8	23	127	10^{38}	10^{-38}	10^{-7}	10^{-45}
pojed. rozszerzona —	44	11	31	1023	10^{308}	10^{-308}	10^{-10}	10^{-317}
podwójna `double`	64	11	52	1023	10^{308}	10^{-308}	10^{-15}	10^{-323}
podw. rozszerzona `long double`	80	15	63	16383	10^{4932}	10^{-4932}	10^{-19}	10^{-4951}

Oznaczenia: B — całkowita liczba bitów, d — liczba bitów cechy, t — liczba bitów mantysy, b — stała odejmowana od cechy w celu otrzymania wykładnika. Stała b jest równa $2^{d-1} - 1$, dzięki czemu jeśli liczba x ma reprezentację znormalizowaną, to $1/x$ na ogół też. Liczby $M = 2^{2^d - b - 2}(2 - 2^{-t})$ — największa liczba zmiennopozycyjna, $S = 2^{1-b}$ — najmniejsza dodatnia liczba reprezentowana w postaci znormalizowanej, $\nu = 2^{-t}$ — oszacowanie względnych błędów zaokrągleń, oraz $\mu = 2^{1-b-t}$ — najmniejsza zmiennopozycyjna liczba dodatnia, są podane w przybliżeniu (tylko rząd wielkości).

Reprezentacje rozszerzonej precyzji nie wymuszają normalizacji (mantysa ma $t+1$ bitów i jest liczbą z przedziału $[0, 2)$, jej najbardziej znaczący bit ma wartość 1), ale wyniki działań, jeśli nie ma niedomiaru, są normalizowane przez procesor. Jeszcze jedno: w 32-bitowych systemach operacyjnych liczba rozszerzonej podwójnej precyzji zajmuje 96 bitów, z których 16 jest nieużywanych. Jeśli nie ma istotnego powodu, to najlepiej nie używać tej reprezentacji liczb.

Oprócz standardu IEEE-754 istnieje też standard IEEE-854, który definiuje reprezentacje liczb zmiennopozycyjnych z podstawami 2 i 10. Standard ten służy do wymiany danych między komputerami, natomiast określone przezeń reprezentacje nie są przetwarzane bezpośrednio przez jednostki zmiennopozycyjne procesorów (w każdym razie znanych mi). Jeśli nie ma ważnych powodów do używania reprezentacji określonych w tym standardzie, to można się nim nie przejmować.

Do celów specjalnych bywają używane reprezentacje niestandardowe; istnieje np. dość rzadko spotykany format poczwórnej precyzji, w którym reprezentacja liczby zajmuje 128 bitów (cecha ma w nim 15 bitów, mantysa 112). Nie słyszałem o procesorach z rejestrami zmiennopozycyjnymi o takiej długości, zatem działania na takich liczbach muszą być wykonywane przez odpowiednie podprogramy. Z drugiej strony, reprezentacje 16- 11- i 10-bitowe (bit znaku może być nieobecny,

cecha ma 5 bitów, a mantysa 10, 6 albo 5) są używane przez niektóre karty graficzne podczas wykonywania obrazów, gdy dokładność ma małe znaczenie, zaś najważniejsza jest szybkość obliczeń i oszczędność miejsca. Wspomniane karty graficzne mają specjalizowane podukłady do wykonywania działań na takich liczbach.

Arytmetyka i błędy zaokrągleń

Na potrzeby analizy błędów działanie procesora podczas wykonywania operacji arytmetycznych można sobie wyobrazić tak: dokładny wynik działania jest poddawany normalizacji (tj. dobierana jest cecha), a następnie zaokrągleniu — nieskończony ciąg bitów mantysy jest obcinany i ewentualnie zaokrąglany w górę. Nie wyznacza się oczywiście nieskończonego ciągu bitów mantysy, zamiast tego wykorzystuje się trzy bity dodatkowe („wystające" poza format), z których pierwsze dwa są zwykłe, a trzeci „lepki" — bit ten otrzymuje wartość 1, jeśli dowolny dalszy bit nieskończenie długiej mantysy jest niezerowy. Te trzy bity zawsze wystarczą do poprawnego zaokrąglenia liczby. Wyboru kierunku zaokrąglania można dokonać, ustawiając odpowiednie bity w rejestrze sterującym procesora (zwykle zostawiamy domyślne zaokrąglanie do najbliższej liczby zmiennopozycyjnej).

Istotne jest, że oprócz reprezentacji liczb, standard IEEE-754 określa własności działań, w tym wymagania dotyczące dokładności wyników — dotyczy to czterech działań arytmetycznych, pierwiastka kwadratowego, oraz konwersji reprezentacji całkowitej i zmiennopozycyjnej. Istnieją procesory, które wprawdzie przetwarzają liczby w standardowym formacie, ale realizowane przez nie działania nie spełnają wszystkich warunków określonych w standardzie. Najbardziej rozpowszechnionym sprzętem tego rodzaju są karty graficzne, które mogą m.in. nie obsługiwać liczb nieznormalizowanych (tj. zapisanych przy użyciu drugiego wzoru podanego w opisie formatu; w razie niedomiaru wynikiem działania jest zero) lub zaokrąglać wyniki działań w arbitralnie określony sposób (standard nakazuje umożliwiać dokonanie wyboru). Powinien o tym pamiętać każdy, kto zajmuje się tzw. GPGPU (*general programming on graphics processing unit*).

Jeśli x jest liczbą rzeczywistą, a $\mathrm{rd}(x)$ jest jej znormalizowanym zmiennopozycyjnym przybliżeniem (bez nadmiaru i niedomiaru), to mamy $|x - \mathrm{rd}(x)| \leqslant |x| 2^{-1-t}$, skąd wynika, że istnieje liczba ε, taka że

$$\mathrm{rd}(x) = x(1+\varepsilon) \quad \text{oraz} \quad |\varepsilon| \leqslant 2^{-1-t}.$$

Sposób zaokrąglania (do najbliższej liczby zmiennopozycyjnej, zawsze w stronę zera, zawsze w przeciwną stronę, zawsze w górę albo zawsze w dół) może być ustawiony różnie, przez co błąd względny może być dwa razy większy. Jeśli zatem $\diamond$ oznacza dowolne z czterech działań arytmetycznych, to zamiast wyniku $x = a \diamond b$, po zaokrągleniu, otrzymamy liczbę

$$\tilde{x} = \mathrm{fl}(a \diamond b) = (a \diamond b)(1+\varepsilon),$$

dla pewnego $\varepsilon \in (-\nu, \nu)$ (piszemy $\mathrm{fl}(a \diamond b)$ zamiast $\mathrm{rd}(a \diamond b)$, bo ten ostatni symbol oznacza u nas wynik zaokrąglenia do najbliższej liczby zmiennopozycyjnej).

Wyniki działań są najczęściej argumentami dalszych działań, zatem podczas obliczeń numerycznych ma miejsce zjawisko zwane kumulacją błędów. W szczególnych przypadkach może ono doprowadzić do otrzymania bardzo niedokładnych wyników końcowych, mimo że poszczególne błędy zaokrągleń są małe. Ponadto wskutek zaokrągleń zbiór liczb zmiennopozycyjnych z działaniami dodawania i mnożenia *nie jest* ciałem (z punktu widzenia algebry). Przede wszystkim, nie jest zamknięty ze względu na działania (bo może wystąpić nadmiar) i są w nim dzielniki zera (np. jeśli liczba $|x|$ jest dostatecznie mała, to $\mathrm{fl}(x * x) = 0$). Po drugie, dodawanie i mnożenie nie są działaniami łącznymi i dodawanie nie jest rozdzielne względem mnożenia. W konsekwencji, algorytmy oparte na różnych wzorach algebraicznie równoważnych (w ciele $\mathbb{R}$), mogą produkować różne wyniki (czasem bardzo od siebie odległe). Analiza algorytmów ma na celu między innymi badanie, na jaką dokładność wyników obliczeń wykonywanych z błędami zaokrągleń można liczyć (i może się przydać do wybrania najlepszego algorytmu, albo przynajmniej do odrzucenia najgorszego).

Arytmetyka zmiennopozycyjna zespolona

W różnych zadaniach występują liczby zespolone. W obliczeniach ich części rzeczywiste i urojone są reprezentowane w postaci zmiennopozycyjnej. Jeśli zatem zamiast liczby $z = (a, b) \neq 0$ mamy liczbę $\tilde{z} = (\tilde{a}, \tilde{b}) = (a(1+\varepsilon_a), b(1+\varepsilon_b))$, gdzie $|\varepsilon_a|, |\varepsilon_b| < \nu$, to liczbę z reprezentujemy z błędem względnym

$$\frac{|z - \tilde{z}|}{|z|} = \frac{\sqrt{a^2\varepsilon_a^2 + b^2\varepsilon_b^2}}{\sqrt{a^2+b^2}} < \frac{\sqrt{a^2\nu^2 + b^2\nu^2}}{\sqrt{a^2+b^2}} = \nu.$$

Zatem reprezentacja zmiennopozycyjna liczby zespolonej zapewnia równie mały błąd, jak reprezentacja liczby rzeczywistej. Dodawanie i odejmowanie liczb zespolonych wykonujemy na podstawie wzorów będących definicją tych działań, w związku z czym, jeśli nie ma nadmiaru ani niedomiaru, otrzymamy

$$\mathrm{fl}(z_1 \pm z_2) = (z_1 \pm z_2)(1+\varepsilon), \quad \text{gdzie} \quad |\varepsilon| < \nu.$$

Mnożenie też wykonuje się na podstawie definicji tego działania:

$$(a_1, b_1) \cdot (a_2, b_2) = (a_1a_2 - b_1b_2, a_1b_2 + a_2b_1).$$

Zamiast dokładnego wyniku otrzymamy

$$\begin{aligned}\mathrm{fl}((a_1, b_1) \cdot (a_2, b_2)) = ((a_1a_2(1+\varepsilon_1) - b_1b_2(1+\varepsilon_2))(1+\varepsilon_3),\\ (a_1b_2(1+\varepsilon_4) + a_2b_1(1+\varepsilon_5))(1+\varepsilon_6)),\end{aligned}$$

przy czym, jeśli w żadnym działaniu nie wystąpił nadmiar ani niedomiar, to wszystkie epsilony mają wartości bezwzględne mniejsze niż ν. Można udowodnić (za pomocą dosyć żmudnego rachunku), że wtedy otrzymany wynik jest równy

$$(a_1, b_1) \cdot (a_2, b_2) \cdot (1 + \xi),$$

gdzie ξ jest pewną liczbą zespoloną, taką że $|\xi| < (1 + \sqrt{2})\nu$.

Dzielenie zespolone jest bardziej kłopotliwe, bo algorytm musi unikać nadmiaru i niedomiaru (zwróćmy uwagę, że nawet w przypadku mnożenia, wynik działania może mieć reprezentację, zaś wyniki pośrednie mogą jej nie mieć z powodu nadmiaru — w dzieleniu ten problem też występuje). Dzielenie powinno się wykonywać za pomocą algorytmu

```
if ( a2 >= b2 ) {
  p = b2/a2;
  q = a2+b2*p;
  wynik = ((a1+b1*p)/q, (b1-a1*p)/q);
}
else {
  p = a2/b2;
  q = a2*p+b2;
  wynik = ((a1*p+b1)/q, (b1*p-a1)/q);
}
```

Jeśli nie ma nadmiaru ani niedomiaru, to względny błąd zaokrąglenia wyniku nie jest większy niż $(4 + \sqrt{2})\nu$.

Zadania i problemy

1. Oblicz oznaczone na wykładzie literami M, S i ν parametry charakteryzujące niestandardowe arytmetyki zmiennopozycyjne: 128-, 16-, 11- i 10-bitową. Przyjmij $b = 2^{d-1} - 1$.
2. Niech x oznacza liczbę zmiennopozycyjną pojedynczej lub podwójnej precyzji, a $i(x)$ liczbę całkowitą bez znaku (odpowiednio 32- lub 64-bitową) reprezentowaną przez ten sam ciąg bitów. Sprawdź, że jeśli $x_2 > x_1 \geqslant 0$, $x_1 > 0 > x_2$ lub $0 \geqslant x_1 > x_2$, to $i(x_2) > i(x_1)$. Wyjaśnij, jak można to wykorzystać do sortowania ciągu liczb zmiennopozycyjnych. Co przeszkadza w sortowaniu w analogiczny sposób liczb rozszerzonej podwójnej precyzji?

Błędy w obliczeniach

W obliczeniach numerycznych występują błędy pięciu rodzajów.

Błędy modelu. Model matematyczny dowolnego zjawiska (przyrodniczego, ekonomicznego i w ogóle każdego) jest tego zjawiska uproszczeniem. Na przebieg zjawiska ma wpływ wiele różnych czynników, z których jedne są ignorowane (bo ich wpływ został uznany za pomijalny), a inne nie są znane dostatecznie dokładnie, aby można było napisać całkowicie poprawny wzór. Jeśli model znacznie odbiega od zjawiska, to i wyniki obliczeń mogą bardzo się różnić od tego, co można zaobserwować w rzeczywistości.

Błędy danych wejściowych. Dane wejściowe trzeba zapisać w postaci liczb zmiennopozycyjnych, co powoduje ich zaburzenie. Jeśli wynik od danych zależy (a zwykle tak jest), to nawet gdyby nie było innych błędów, wynik obliczeń może się różnić od wyniku doświadczenia. Ponadto, na ogół dane otrzymujemy z pomiarów, których niedokładności mogą być znacznie większe niż błąd reprezentacji zmiennopozycyjnej. Najdokładniejsze pomiary w fizyce dają kilkanaście cyfr dokładnych, często znamy dane z dokładnością rzędu 1%, a czasami błędy są na poziomie kilkudziesięciu procent. Sygnały lub obrazy mogą być zniekształcone z powodu szumu i bardzo niewyraźne. To wszystko ma bardzo duży wpływ na wynik (albo jego brak, jeśli algorytm nie poradzi sobie z niedokładnymi danymi).

Błędy aproksymacji. W obliczeniach numerycznych stosuje się przybliżenia funkcji, których dokładne obliczenie jest niewykonalne lub zbyt kosztowne. Na przykład, zamiast granicy nieskończonego ciągu zbieżnego, bierze się pewien element tego ciągu. Zamiast sumy szeregu nieskończonego oblicza się sumę iluś początkowych składników. Zamiast całki oblicza się kwadraturę. Równania różniczkowe często zastępuje się równaniami różnicowymi; można podać wiele dalszych przykładów.

Błędy zaokrągleń. Wynik każdego działania wykonanego przez komputer podlega zaokrągleniu. Skutki badzo często są małe w porównaniu ze skutkami innych błędów, ale czasem mogą zupełnie zmienić wynik.

Błędy grube. To są skutki wszelkich pomyłek, awarii, oraz błędów popełnionych w procesie pozyskiwania danych lub w implementacji algorytmu. Z innych przyczyn można tu też wymienić sabotaż (np. uprawiany przez producentów wirusów komputerowych i przez nierzetelnych autorów oprogramowania).

Uwarunkowanie zadania

Większość zadań numerycznych polega na obliczeniu wartości pewnej funkcji f, której dziedziną jest pewien obszar $D \subset \mathbb{R}^n$. Wynik obliczenia jest wektorem w $\mathbb{R}^m$, przy czym m może być określone przez konkretny argument $\boldsymbol{x} \in D$ — na przykład, gdy trzeba znaleźć wszystkie rzeczywiste miejsca zerowe wielomianu, którego współczynniki są współrzędnymi wektora $\boldsymbol{x}$. Załóżmy jednak, że m jest ustalone (i znane) dla wszystkich $\boldsymbol{x} \in D$, a funkcja f jest ciągła. Zanim zaczniemy rozpatrywać jakiekolwiek algorytmy obliczania wyniku, zajmiemy się wpływem, jaki zaburzenia danych (które mogą pochodzić z niedokładnych pomiarów i które trzeba zastąpić liczbami zmiennopozycyjnymi) mają na wynik.

Pojęcie numerycznego uwarunkowania zadania określa wrażliwość wyniku na zaburzenia danych; dla zadania dobrze uwarunkowanego niewielkie zaburzenie danych powoduje niewielką zmianę wyniku. Zadanie jest źle uwarunkowane, jeśli po małej zmianie danych otrzymujemy zupełnie inny wynik. W związku ze sposobem reprezentowania liczb (który zapewnia mały błąd względny), bierzemy pod uwagę *względne* zaburzenia danych i spowodowane przez nie zmiany wyniku.

Liczbowa miara uwarunkowania nazywa się wskaźnikiem uwarunkowania zadania. Określa się go wzorem

$$\operatorname{cond}_{f(\boldsymbol{x})} \boldsymbol{x} = \sup_{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|<\varepsilon} \left(\frac{\|f(\tilde{\boldsymbol{x}}) - f(\boldsymbol{x})\|}{\|f(\boldsymbol{x})\|} \Big/ \frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \right).$$

Symbol cond pochodzi od angielskiego *condition number*; napis po lewej stronie czytamy: „wskaźnik uwarunkowania zadania obliczenia $f(\boldsymbol{x})$ dla danych $\boldsymbol{x}$”.
W określeniu wskaźnika uwarunkowania używamy jakichś norm (zależnie od zadania) i określamy największą dopuszczalną zmianę (zaburzenie) ε danych $\boldsymbol{x}$. Następnie badamy iloraz względnego zaburzenia wyniku i powodującej to zaburzenie względnej zmiany danych.

Co daje znajomość wskaźnika uwarunkowania? Jeśli dane znamy z błędem względnym nie większym niż ε, to błąd względny wyniku (uwaga: *dokładnego* wyniku dla danych $\tilde{\boldsymbol{x}}$, jakimi dysponujemy, w porównaniu z wynikiem dla nieznanych nam danych dokładnych $\boldsymbol{x}$) nie jest większy niż $\varepsilon \operatorname{cond}_{f(\boldsymbol{x})} x$. Na przykład, jeśli wskaźnik uwarunkowania jest równy 100 (to jeszcze nie jest dużo), a dane reprezentujemy w formacie pojedynczej precyzji, tj. z błędem nie większym niż $\nu \approx 10^{-7}$ (i poza zaokrągleniem nie ma innych błędów), to wiemy, że jesteśmy w stanie otrzymać wynik z pięcioma cyframi dokładnymi. Jeśli jednak pomiar danych ma błąd rzędu 1%, to otrzymany wynik może mieć błąd 100%; na ogół taki wynik jest bezwartościowy. Albo należy wtedy zdobyć dokładniejsze

dane, albo zająć się innym zadaniem (być może można jakoś przeformułować problem). Pamiętajmy przy tym, że założyliśmy brak błędów w algorytmie, który może dodatkowo zepsuć wynik.

Często przyjmuje się, że zaburzenia danych są bardzo małe (bo względne błędy reprezentacji zmiennopozycyjnej są bardzo małe), więc dla uproszczenia oblicza się wartość graniczną wskaźnika uwarunkowania, dla $\varepsilon \to 0$ (co ma sens, jeśli wskaźnik jest ciągły w otoczeniu $\mathbf{x}$). Jeśli zadanie polega na obliczeniu wartości skalarnej funkcji f, która ma skalarny argument x, przy czym funkcja f ma pochodną, to mamy wtedy

$$\operatorname{cond}_{f(x)} x = \left| \frac{x}{f(x)} f'(x) \right|.$$

Błędy reprezentacji wektorów

Niech $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ i niech $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots, \tilde{x}_n]^T \in \mathbb{R}^n$, przy czym $\tilde{x}_i = x_i(1 + \varepsilon_i)$ dla każdego i. Zamiast rozpatrywać osobno błędy poszczególnych składowych wektora, co mogłoby zbyt być pracochłonne, często błąd opisuje się jedną liczbą, za pomocą jakiejś normy. Najczęściej wykorzystywane są tzw. normy Höldera, określone wzorem

$$\|\mathbf{x}\|_p = \left(|x_1|^p + \cdots + |x_n|^p\right)^{1/p},$$

dla pewnego $p \geqslant 1$. Często stosuje się normę — przypadek graniczny, dla $p \to \infty$:

$$\|\mathbf{x}\|_\infty = \max_{i \in \{1,\ldots,n\}} |x_i|.$$

Za miarę błędu bezwzględnego możemy przyjąć liczbę $\|\mathbf{x} - \tilde{\mathbf{x}}\|_p$. Jeśli $\mathbf{x} \neq 0$ i dla każdego i jest $|\varepsilon_i| \leqslant \nu$, to miara błędu względnego spełnia nierówność

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|_p} = \frac{\left(|x_1\varepsilon_1|^p + \cdots + |x_n\varepsilon_n|^p\right)^{1/p}}{\|\mathbf{x}\|_p} \leqslant \frac{\left(|x_1\nu|^p + \cdots + |x_n\nu|^p\right)^{1/p}}{\|\mathbf{x}\|_p} = \frac{\|\mathbf{x}\|\nu}{\|\mathbf{x}\|} = \nu.$$

Zatem, błąd względny reprezentacji wektora, którego współrzędne zostały zokrąglone do najbliższych liczb zmiennopozycyjnych, mierzony za pomocą dodolnej normy Höldera (także $\|\cdot\|_\infty$), jest na poziomie błędu reprezentacji pojedynczej liczby.

Uwaga: Należy pamiętać, że z nierówności $\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|_p}{\|\mathbf{x}\|_p} \leqslant \varepsilon > 0$ *nie wynika*, że błędy poszczególnych składowych są małe. Jeśli pewna składowa jest równa 0, to dowolne niezerowe jej zaburzenie daje nieograniczony błąd względny. Tak więc, wykonując odpowiednie rachunki, nie należy wyciągać pochopnych wniosków.

Numeryczna poprawność algorytmu

Skutki błędów zaokrągleń w obliczeniach czasem można zinterpretować jako skutki takiego zaburzenia danych, że otrzymany wynik jest dla tych zaburzonych danych dokładny. Jeśli takie hipotetyczne zaburzenie danych jest małe, to mówimy, że algorytm jest numerycznie poprawny. Pewne algorytmy są numerycznie poprawne, inne nie są. W zasadzie numeryczna poprawność „to jest to" — w praktyce niczego lepszego po algorytmach numerycznych spodziewać się nie można.

Tak, jak uwarunkowanie zadania, numeryczną poprawność można mierzyć, badając tzw. stałe kumulacji algorytmu. Algorytm jest tym lepszy, im te stałe są mniejsze. Aby je zdefiniować, wprowadzimy potrzebne oznaczenia. Niech A oznacza algorytm. Zatem, niech $A(\mathbf{x})$ oznacza wynik obliczenia, który powinien być jak najbliższy „prawdziwemu" rozwiązaniu zadania, $f(\mathbf{x})$. Obliczony wynik składa się z liczb zmiennopozycyjnych, zatem możemy dopuścić do rozważań jego błąd reprezentacji. Przypuśćmy zatem, że istnieją liczby K_d i K_w, takie że dla każdego $\mathbf{x} \in D$ istnieją dane zaburzone $\tilde{\mathbf{x}}$, dla których spełnione są nierówności

$$\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} \leqslant K_d\nu, \quad \text{oraz} \quad \frac{\|f(\tilde{\mathbf{x}}) - A(\mathbf{x})\|}{\|f(\tilde{\mathbf{x}})\|} \leqslant K_w\nu.$$

Mówimy wtedy, że algorytm A jest numerycznie poprawnym algorytmem obliczania wartości funkcji f w dziedzinie (klasie zadań) D, ze stałymi kumulacji (danych) K_d i (wyniku) K_w.

Trzeba podkreślić, że w analizie algorytmu często występuje swoboda wybierania danych lub wyniku, do których „doczepiamy" błędy; z jednej strony to utrudnia analizę, a z drugiej stwarza możliwości pewnej „gimnastyki", wskutek czego pewne oszacowania mogą być poprawione — nieraz jest tak, że algorytm w praktyce działa bardzo dobrze, tj. wytwarza bardzo dokładne wyniki, zaś analiza tego nie potwierdza, bo na przykład daje bardzo grube oszacowania stałych kumulacji. Wspomniana „gimnastyka" czasem pomaga. W analizie błędu zwykle zakłada się, że błędy w poszczególnych działaniach są niezależne (i nieskorelowane), a ich wartości bezwzględne sumują się, tymczasem poszczególne błędy względne mogą być mniejsze niż ν, mogą się też znosić. Czasem analiza błędu pozwala wykryć newralgiczne miejsca i pomaga przeprojektować wzory.

Jeśli stała K_d jest równa 0, to znaczy, że niezależnie od uwarunkowania zadania otrzymany wynik jest bardzo dokładny, tj. otrzymany z dokładnością na poziomie błędu reprezentacji (tj. błąd wyniku jest co najwyżej K_w razy większy). Taka sytuacja występuje w praktyce nadzwyczaj rzadko. Częściej „winę" za

niedokładność wyniku można „zwalić” na dane. Takie postępowanie, tj. znalezienie i oszacowanie zaburzenia danych, które prowadzi do otrzymanego wyniku, nazywa się analizą wstecz; jej twórcą był Wilkinson. Jeśli zadanie jest dobrze uwarunkowane i stałe kumulacji są nieduże, to stąd wynika, że obliczony wynik jest dobrym przybliżeniem wyniku poszukiwanego.

Numeryczna stabilność algorytmu

Często się nie udaje udowodnienie numerycznej poprawności algorytmu, tj. znalezienie stałych kumulacji niezależnych od danych w ustalonej dziedzinie D. Wówczas można spróbować zbadać, czy jest on numerycznie stabilny — ta własność jest pewnego rodzaju „minimum przyzwoitości” algorytmu. Aby ją zdefiniować, zbadajmy, jak duży byłby błąd wyniku, gdyby dane zostały zaburzone na poziomie błędu reprezentacji (co musi mieć miejsce — dane do obliczeń są liczbami zmiennopozycyjnymi) i wynik też należałoby zaokrąglić (bo też go reprezentujemy w ten sposób), ale poza zaokrągleniem końcowego wyniku wszystkie obliczenia byłyby wykonywane dokładnie.

Błąd (bezwzględny) wyniku spełniający wymienione warunki można oszacować przez liczbę

$$\|f(\mathbf{x})\|(\mathrm{cond}_w\,\mathbf{d}+1)\nu.$$

Względny błąd danych, na poziomie ν, przenosi się na wynik z czynnikiem $\mathrm{cond}_w\,\mathbf{d}$; do tego wynik trzeba jeszcze zaokrąglić, stąd do wskaźnika uwarunkowania została dodana jedynka. Liczba będąca wartością podanego wyrażenia nazywa się optymalnym poziomem błędu.

Jeśli teraz algorytm zaokrągla wyniki wykonywanych działań, to może wynik zepsuć dodatkowo. Mówimy, że algorytm A jest numerycznie stabilnym algorytmem obliczania funkcji f, jeśli istnieje liczba K (stała kumulacji), taka że dla dowolnych danych $\mathbf{d}\in D$ spełniona jest nierówność

$$\|f(\mathbf{x})-A(\mathbf{x})\|\leqslant K\|f(\mathbf{x})\|(\mathrm{cond}_w\,\mathbf{d}+1)\nu.$$

Ważne jest też, aby stała kumulacji nie była bardzo duża.

W tym ujęciu analizy błędów nie zajmujemy się tym, czy istnieją takie dane, bliskie danych $\mathbf{x}$, dla których otrzymujemy (ewentualnie zaburzony na poziomie błędu reprezentacji) wynik. Dane takie mogą więc nie istnieć — możemy na przykład otrzymać sinus pewnego kąta rzeczywistego większy niż 1. Istotne jest to, że mając algorytm numerycznie stabilny, możemy dowolnie zmniejszyć skutki

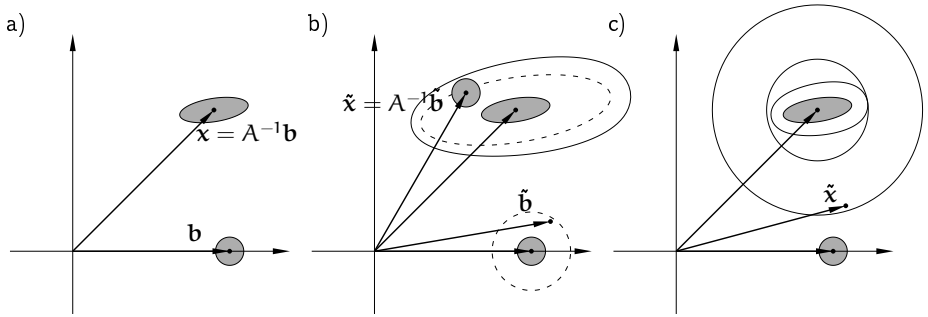
błędów zaokrągleń, wykorzystując w obliczeniach dostatecznie dokładną arytmetykę (czyli taką o dostatecznie długiej mantysie: przypominam, że $\nu=2^{-t}$). Oczywiście, dla zadań źle uwarunkowanych arytmetyki standardowe mogą nie wystarczyć, ale wtedy czy na pewno znamy dane aż tak dokładnie?

Jeśli funkcja f, której wartość należy obliczyć, spełnia warunek Lipschitza, tj. istnieje stała L, taka że

$$\forall_{\mathbf{x},\mathbf{y}\in D}\ \|f(\mathbf{x})-f(\mathbf{y})\|\leqslant L\|\mathbf{x}-\mathbf{y}\|,$$

to każdy algorytm numerycznie poprawny jest też numerycznie stabilny, ale numeryczna stabilność nie gwarantuje numerycznej poprawności.

Ilustracją opisanych pojęć może być rysunek, będący ilustracją zadania rozwiązywania układu dwóch równań liniowych $A\mathbf{x}=\mathbf{b}$, z nieosobliwą macierzą A. Rozwiązaniem zadania jest wektor $\mathbf{x}=A^{-1}\mathbf{b}$, przy czym ten wzór jest pożyteczny w teoretycznej analizie zadania i algorytmów jego rozwiązywania, ale nie jest dobrym algorytmem numerycznym (i proszę go *nie używać* w tym charakterze). Danymi są współczynniki macierzy A i wektora prawej strony $\mathbf{b}$. Dla ilustracji pojęć rozpatrujemy tylko zaburzenia wektora prawej strony. Wielkość tych zaburzeń jest taka, jak gdyby mantysa miała mniej więcej trzy bity.



Na rysunku a) mamy ilustrację uwarunkowania zadania. Zaznaczona kula (tj. koło) o środku $\mathbf{b}$ ma promień $\nu\|\mathbf{b}\|$. Zaburzenie danych polega na zastąpieniu wektora $\mathbf{b}$ przez jakiś element tej kuli. Obrazem tej kuli jest elipsoida (elipsa) o środku $\mathbf{x}$. Wskaźnik uwarunkowania zadania (ze względu na zaburzanie wektora $\mathbf{b}$, ale także macierzy A, co będziemy badać na jednym z dalszych wykładów) jest ilorazem długości najdłuższej i najkrótszej osi elipsoidy.

Numeryczna poprawność jest zilustrowana na rysunku b). Algorytm wyprodukował pewien wektor $\tilde{\mathbf{x}}$. Niech $\tilde{\mathbf{b}}=A\mathbf{x}$. Przypuśćmy, że stała kumulacji

$K_w = 0$. Wtedy

$$\frac{\|\tilde{\mathbf{b}} - \mathbf{b}\|}{\|\mathbf{b}\|\nu} \leqslant K_d,$$

i mamy gwarancję, że dla otrzymanego wyniku $\tilde{\mathbf{x}}$, który leży w obrębie narysowanej linią przerywaną elipsy, istnieją dane $\tilde{\mathbf{b}}$, które leżą w narysowanym linią przerywaną kole (promień tego koła jest K_d razy większy niż $\|\mathbf{b}\|\nu$). Jeśli zaś weźmiemy $K_w > 0$, to dopuszczamy dodatkowe zaburzenie wyniku; leży on w nieco większym obszarze ograniczonym przez krzywą zobrazowaną przez linię ciągłą (ta krzywa nie jest elipsą). Dla takiego wyniku istnieje bliski punkt leżący w obszarze ograniczonym elipsą, który jest dokładnym wynikiem dla pewnych danych położonych w większym kole o środku $\mathbf{b}$.

Numeryczna stabilność jest przedstawiona na rysunku c). Rozważamy zaburzenia danych $\mathbf{b}$ na poziomie błędu reprezentacji. Dla tak zaburzonych danych wynik leży w obszarze zacienionym, ograniczonym przez elipsę. Ten obszar rozszerzamy, aby uwzględnić błąd reprezentacji wyniku, a następnie opisujemy koło. Promień tego koła jest optymalnym poziomem błędu. Wynik jest punktem koła o promieniu K razy większym. Dla pewnych punktów tego koła, położonych daleko od elipsy, nie istnieją dane $\tilde{\mathbf{b}}$, leżące blisko danych $\mathbf{b}$ i takie, że mamy dokładny wynik dla danych $\tilde{\mathbf{b}}$.

Zadania i problemy

1. Zadanie polega na obliczeniu sumy n liczb: $s = x_1 + \cdots + x_n$. Dla algorytmu sumowania „po kolei" napisz wyrażenie, którego wartość jest sumą obliczoną z błędami zaokrągleń (przy założeniu, że nie wystąpił nadmiar ani niedomiar zmiennopozycyjny). Skonstruuj wektor $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots, \tilde{x}_n]$, taki że wynik obliczenia jest sumą jego współrzędnych i znajdź stałą kumulacji, dowodząc w ten sposób numerycznej poprawności.
2. Zrób to samo dla algorytmu sumowania parami, w którym oblicza się sumy kolejnych par danych liczb, sumy par tych sum, itd., aż do otrzymania sumy wszystkich składników. Porównaj stałe kumulacji tych dwóch algorytmów.
3. Algorytm Kahana sumowania liczb: para zmiennych zmiennopozycyjnych, `(s, c)`, symuluje liczbę zmiennopozycyjną o dwukrotnie dłuższej mantysie, dzięki czemu błędy względne zaokrągleń są na poziomie 2^{-2t}:

```
s = a[0];
c = 0.0;
for ( i = 1; i < n; i++ ) {
  y = a[i]-c;
  t = s+y;
  c = (t-s)-y;
  s = t;
}
```

4. Zadanie polega na obliczeniu wartości wielomianu $w(x) = a_n x^n + \cdots + a_1 x + a_0$ dla ustalonego x. Skonstruuj współczynniki w bazie potęgowej wielomianu $\tilde{w}(x)$, którego wartość w punkcie x została obliczona i znajdź stałą kumulacji.
5. Zbadaj błędy zaokrągleń wytworzone przez algorytm obliczania wartości wyrażenia $a^2 + ab + b^2$, korzystający ze wzoru

$$w = \frac{1}{2}\left((a+b)^2 + (a^2 + b^2)\right).$$

Mnożenie przez całkowite potęgi liczby 2 (tu przez $\frac{1}{2}$), jeśli nie ma nadmiaru ani niedomiaru, jest wolne od błędów zaokrągleń.

Przypomnienia

1. Przypomnienie pojęcia normy (potrzebne w dalszych wykładach): norma jest to *dowolny* funkcjonał $\|\cdot\|$ określony w przestrzeni liniowej V nad ciałem liczbowym $\mathbb{K}$ (np. $\mathbb{R}$ lub $\mathbb{C}$; w tym wykładzie rozpatrujemy tylko przestrzenie rzeczywiste) i przyjmujący wartości rzeczywiste, który spełnia warunki:

- dodatniość: $\forall_{x\in V}\ \|x\| \geqslant 0$ oraz $\|x\| = 0 \Rightarrow x = 0$,
- półliniowość: $\forall_{x\in V,\, a\in\mathbb{K}}\ \|ax\| = |a|\|x\|$,
- nierówność trójkąta: $\forall_{x,y\in V}\ \|x+y\| \leqslant \|x\| + \|y\|$.

W przestrzeni $\mathbb{R}^n$ (a także $\mathbb{C}^n$) bardzo często rozważa się normy Höldera: dla dowolnego $p \geqslant 1$ funkcja określona wzorem

$$\|x\|_p = (|x_1|^p + \cdots + |x_n|^p)^{1/p}$$

spełnia podane warunki, jest więc normą. Również w granicy dla $p \to \infty$ otrzymujemy normę

$$\|x\|_\infty = \max_{i\in\{1,\ldots,n\}} |x_i|.$$

Szczególnie często rozważa się normy Höldera dla $p = 1$, $p = \infty$ i $p = 2$.

2. Norma Höldera dla $p = 2$, czyli tak zwana norma euklidesowa, mierzy „zwykłą długość" wektora. Norma ta ma związek z iloczynem skalarnym w $\mathbb{R}^n$, określonym wzorem

$$\langle x, y\rangle = y^T x,$$

mianowicie dla każdego $x \in \mathbb{R}^n$ zachodzi równość

$$\|x\|_2 = \sqrt{\langle x, x\rangle}.$$

3. Niech V_a i V_b będą przestrzeniami określonymi nad tym samym ciałem liczbowym $\mathbb{K}$ i niech będą określone w nich jakieś (dowolne) normy, odpowiednio $\|\cdot\|_a$ i $\|\cdot\|_b$. Zbiór $L(V_a; V_b)$ wszystkich przekształceń liniowych $V_a \to V_b$ jest przestrzenią liniową nad $\mathbb{K}$. Funkcjonał określony w przestrzeni $L(V_a; V_b)$ wzorem

$$\|f\| = \sup_{x\neq 0} \frac{\|f(x)\|_b}{\|x\|_a} = \sup_{\|x\|\leqslant 1} \|f(x)\|_b$$

jest normą. Mówimy, że jest to norma indukowana przez normy $\|\cdot\|_a$ i $\|\cdot\|_b$.

Przestrzeń przekształceń liniowych $\mathbb{R}^n \to \mathbb{R}^m$ możemy utożsamić z przestrzenią $\mathbb{R}^{m,n}$ macierzy $m \times n$. Jeśli w przestrzeniach $\mathbb{R}^n$ i $\mathbb{R}^m$ przyjmiemy normy p-te Höldera, to normę indukowaną w $\mathbb{R}^{m,n}$ będziemy również oznaczać symbolem $\|\cdot\|_p$. W przestrzeni $\mathbb{R}^{m,1} = \mathbb{R}^m$ p-ta norma indukowana jest tożsama z p-tą normą Höldera (zatem symbol $\|\cdot\|_p$ nie prowadzi do nieporozumień).

Normy indukowane pierwsza i nieskończona są łatwe do obliczenia:

$$\|A\|_1 = \max_{j\in\{1,\ldots,n\}} \sum_{i=1}^{m} |a_{ij}|,$$

$$\|A\|_\infty = \max_{i\in\{1,\ldots,m\}} \sum_{j=1}^{n} |a_{ij}|.$$

Na ćwiczeniach, jeśli będzie czas, warto udowodnić te wzory.

4. Normę w $\mathbb{R}^{m,n}$ indukowaną przez normy drugie w $\mathbb{R}^n$ i $\mathbb{R}^m$ można obliczyć na podstawie wzoru

$$\|A\|_2 = \sqrt{\rho(A^T A)},$$

w którym $\rho(A^T A)$ oznacza promień spektralny macierzy $A^T A$, tj. największą wartość bezwzględną wartości własnej macierzy $A^T A$ (w tym przypadku największa wartość własna jest liczbą rzeczywistą nieujemną). Jeśli macierz A jest kwadratowa i symetryczna, to zachodzi też równość $\|A\|_2 = \rho(A)$. Jeśli nie znamy wartości własnych, to możemy skorzystać z oszacowania $\|A\|_2 \leqslant \sqrt{\|A^T A\|_\infty}$.

5. Udowodnij, że norma Frobeniusa, określona w przestrzeni $\mathbb{R}^{m,n}$ wzorem

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} |a_{ij}|^2},$$

jeśli $n > 1$ i $m > 1$, *nie jest* normą indukowaną przez żadną p-tą normę Höldera.

Rozwiązywanie układów równań liniowych

Zajmujemy się rozwiązywaniem układu równań liniowych

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

w którym dane są: nieosobliwa macierz A o wymiarach $n \times n$ i wektor $\boldsymbol{b} \in \mathbb{R}^n$. Układ ten ma jednoznaczne rozwiązanie, $\boldsymbol{x} = A^{-1}\boldsymbol{b}$, ale ten wzór, poza bardzo szczególnymi przypadkami, nie nadaje się do numerycznego rozwiązywania naszego zadania (ale w rachunkach symbolicznych nie zawahamy się go użyć, do czego mamy pełne prawo).

Uwarunkowanie układu równań liniowych

Zbadamy, jak zmieni się rozwiązanie układu, jeśli dane, tj. macierz A lub wektor $\boldsymbol{b}$ zaburzymy. Dla układu równań

$$A\boldsymbol{x}' = \boldsymbol{b} + \delta\boldsymbol{b}$$

otrzymujemy rozwiązanie

$$\boldsymbol{x}' = A^{-1}\boldsymbol{b} + A^{-1}\delta\boldsymbol{b} = \boldsymbol{x} + A^{-1}\delta\boldsymbol{b},$$

skąd wynika, że

$$\|\boldsymbol{x}' - \boldsymbol{x}\| \leqslant \|A^{-1}\|\|\delta\boldsymbol{b}\| = \|A^{-1}\|\|\boldsymbol{b}\|\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} = \|A^{-1}\|\|A\boldsymbol{x}\|\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leqslant \|A^{-1}\|\|A\|\|\boldsymbol{x}\|\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|},$$

i ostatecznie

$$\frac{\|\boldsymbol{x}' - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \|A\|\|A^{-1}\|\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}.$$

Zaburzymy teraz macierz A, tj. będziemy rozwiązywać układ $(A + \delta A)\boldsymbol{x}'' = \boldsymbol{b}$. Mamy

$$A(I + A^{-1}\delta A)\boldsymbol{x}'' = \boldsymbol{b}.$$

Musimy założyć, że zaburzenie macierzy A jest na tyle małe, że macierz $(I + A^{-1}\delta A)$ jest nieosobliwa, dzięki czemu możemy ją odwrócić i użyć wzoru przybliżonego

$$(I + A^{-1}\delta A)^{-1} \approx I - A^{-1}\delta A.$$

Dostaniemy wtedy

$$\boldsymbol{x}'' \approx (I - A^{-1}\delta A)A^{-1}\boldsymbol{b} = A^{-1}\boldsymbol{b} - A^{-1}\delta A A^{-1}\boldsymbol{b} = \boldsymbol{x} - A^{-1}\delta A\boldsymbol{x},$$

skąd wynika przybliżona nierówność

$$\frac{\|\boldsymbol{x}'' - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|}.$$

Zatem, oba zaburzenia względne danych, tj. wektora $\boldsymbol{b}$ i macierzy A, mogą przenieść się na wynik z czynnikiem co najwyżej $\|A\|\|A^{-1}\|$. Ten czynnik jest wskaźnikiem uwarunkowania zadania rozwiązywania układu równań $A\boldsymbol{x} = \boldsymbol{b}$ i bywa też nazywany wskaźnikiem uwarunkowania macierzy A. Jeśli przyjmiemy normę p-tą indukowaną, to mamy wskaźnik uwarunkowania macierzy A w normie p-tej, który oznaczamy symbolem $\mathrm{cond}_p A$ ($\mathrm{cond}_p A = \|A\|_p\|A^{-1}\|_p$).

Normy indukowane $\|\cdot\|_1$ i $\|\cdot\|_\infty$ macierzy A są łatwe do znalezienia. Ponieważ na ogół nie znamy (i nie tracimy czasu na znajdowanie) macierzy A^{-1}, jej normę możemy zwykle tylko oszacować. Jeśli dysponujemy dodatkową informacją o zadaniu, z którego wziął się nasz układ równań, to warto z takiej informacji skorzystać w tym celu. Szacowanie normy macierzy A^{-1} jest też w zasadzie możliwe na podstawie czynników rozkładu znalezionych podczas rozwiązywania układu jedną z metod bezpośrednich.

Metody bezpośrednie

Metody bezpośrednie możemy stosować wtedy, gdy liczba równań i niewiadomych jest mała (co najwyżej rzędu 10^3) lub gdy macierz układu jest „szczególnie łatwa", np. trójdiagonalna. Metody te, gdyby nie było błędów zaokrągleń, dawałyby dokładny wynik po wykonaniu skończenie wielu działań. Błędy zaokrągleń oczywiście to psują.

Metoda eliminacji Gaussa

Metoda eliminacji Gaussa jest najprostszym i chyba najczęściej używanym algorytmem rozwiązywania układów równań liniowych. Składa się on z dwóch etapów. W pierwszym układ jest przekształcany tak, aby powstał równoważny danemu układ równań liniowych z macierzą trójkątną górną. W etapie drugim, na podstawie kolejnych równań (od końca) obliczamy kolejne niewiadome (też od końca) — w każdym równaniu występuje tylko jedna niewiadoma, której wartość nie została obliczona wcześniej.

W pierwszym etapie (który jest właściwą eliminacją Gaussa), konstruujemy ciąg macierzy $A^{(0)} = A, A^{(1)}, \ldots, A^{(n-1)} = U$, takich że macierz $A^{(k)}$ ma w kolumnach $1, \ldots, k$ współczynniki poniżej diagonali równe 0. Mając macierz $A^{(k-1)} = [a_{ij}^{(k-1)}]_{i,j}$, obliczamy współczynniki macierzy $A^{(k)}$:

$$\left.\begin{array}{rcl} l_{ik} & = & a_{ik}^{(k-1)}/a_{kk}^{(k-1)}, \\ a_{ij}^{(k)} & = & a_{ij}^{(k-1)} - l_{ik}a_{kj}^{(k-1)}, \quad \text{dla } j = k+1, \ldots, n \end{array}\right\} \quad \text{dla } i = k+1, \ldots, n$$

Ponadto $a_{ij}^{(k)} = a_{ij}^{(k-1)}$ dla $i \leqslant k$, oraz $a_{ik}^{(k)} = 0$ dla $i > k$.

Przekształcanie wektora prawej strony polega na skonstruowaniu ciągu wektorów $\mathbf{b}^{(0)} = \mathbf{b}, \mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(n-1)} = \mathbf{y}$. W k-tym kroku eliminacji obliczamy współrzędne wektora $\mathbf{b}^{(k)}$:

$$b_i^{(k)} = b_i^{(k-1)} - l_{ik}b_k^{(k-1)}, \quad \text{dla } i = k+1, \ldots, n,$$

zaś dla $i = 1, \ldots, k$ mamy $b_i^{(k)} = b_i^{(k-1)}$. W wyniku eliminacji otrzymujemy macierz trójkątną

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \ldots & u_{1n} \\ 0 & u_{22} & u_{23} & \ldots & u_{2n} \\ 0 & 0 & u_{33} & \ldots & u_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & u_{nn} \end{bmatrix},$$

i wektor $\mathbf{y}$, takie że układ $U\mathbf{x} = \mathbf{y}$ jest równoważny układowi danemu.

Niech

$$L = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ l_{21} & 1 & 0 & \ldots & 0 \\ l_{31} & l_{32} & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ l_{n1} & l_{n2} & \ldots & l_{n,n-1} & 1 \end{bmatrix},$$

Okazuje się, że zachodzi równość $A = LU$. Przekształcanie wektora prawej strony jest równoważne rozwiązywaniu układu równań $L\mathbf{y} = \mathbf{b}$. Zatem możemy najpierw wyznaczyć tylko macierze L i U, a przetwarzanie wektora prawej strony przenieść do drugiego etapu, w którym trzeba rozwiązać kolejno układy równań z macierzami trójkątnymi, $L\mathbf{y} = \mathbf{b}$ i $U\mathbf{x} = \mathbf{y}$.

Eliminację można wykonać *w miejscu* (po łacinie *in situ*). Po obliczeniu współczynnika l_{ik}, można go zapamiętać na miejscu współczynnika $a_{ik}^{(k-1)}$ (czyli na

miejscu zajmowanym początkowo przez a_{ik}). Obliczone współczynniki $a_{ij}^{(k)}$ dla $i \leqslant j$ wpisujemy w miejsce $a_{ij}^{(k-1)}$. W ten sposób otrzymamy tablicę z liczbami

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & \ldots & u_{1n} \\ l_{21} & u_{22} & u_{23} & \ldots & u_{2n} \\ l_{31} & l_{32} & u_{33} & \ldots & u_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ l_{n1} & l_{n2} & \ldots & l_{n,n-1} & u_{nn} \end{bmatrix},$$

do której może sięgać podprogram rozwiązujący układy trójkatne. Podprogram eliminacji *in situ* „psuje" początkową zawartość tablicy. Jeśli oryginalna macierz A jest potrzebna (często jest), należy ją skopiować i „zepsuć" kopię.

Opisany wyżej algorytm jest zawodny; nieosobliwość macierzy A nie gwarantuje wykonalności dzielenia przez współczynnik $a_{kk}^{(k)}$, który może być zerem. Co więcej, jeśli współczynnik ten ma małą wartość bezwzględną, to skutki błędów zaokrągleń mogą prowadzić do otrzymania bardzo niedokładnych wyników. Dlatego stosuje się wybór elementu głównego (ang. *pivoting*). Najczęściej stosowany wybór częściowy w kolumnie polega na wyszukaniu w zbiorze $\{a_{kk}^{(k-1)}, \ldots, a_{nk}^{(k-1)}\}$ współczynnika $a_{lk}^{(k-1)}$ o największej wartości bezwzględnej, a następnie (jeśli $l \neq k$) przestawieniu równań l i k. Jeśli macierz A jest nieosobliwa, to po takim przestawieniu dzielenie jest wykonalne i współczynniki l_{ik} mają wartości bezwzględne nie większe niż 1.

Można dowieść, że skutki takiego przestawiania są takie same, jak gdyby równania zostały poprzestawiane *przed* przystąpieniem do eliminacji. Zatem, po zastosowaniu częściowego wyboru elementu głównego, otrzymamy macierze L i U, takie że $LU = PA$, gdzie P oznacza macierz dokonanej permutacji równań.

Macierz P trzeba jakoś reprezentować, aby można było odpowiednio poprzestawiać współrzędne wektora prawej strony, ponieważ trzeba będzie rozwiązać układy równań $L\mathbf{y} = P\mathbf{b}$ i $U\mathbf{x} = \mathbf{y}$. Najprostszy sposób polega na użyciu tablicy liczb całkowitych o długości n; pozycji k-tej przypisujemy indeks l wiersza, który został przestawiony z k-tym (albo k, jeśli nie było przestawienia). Przestawianie liczb zmiennopozycyjnych w tablicy jest operacją wolną od błędów zaokrągleń.

Istnieje też wybór pełny elementu głównego; przestawiamy w nim wiersze i kolumny tak, aby współczynnik $a_{kk}^{(k-1)}$, przez który będziemy dzielić, miał największą wartość bezwzględną w prawej dolnej podmacierzy $n+1-k \times n+1-k$. W ten sposób otrzymujemy rozkład macierzy $LU = PAQ^T$. Do rozwiązania mamy układy $L\mathbf{y} = P\mathbf{b}$ i $U\mathbf{z} = \mathbf{y}$, a następnie trzeba obliczyć $\mathbf{x} = Q^T\mathbf{z}$, czyli odpowiednio poprzestawiać współrzędne rozwiązania. Macierz

permutacji Q można reprezentować w taki sam sposób jak P. Pełny wybór elementu głównego jest dosyć kosztowny i *bardzo rzadko* zdarza się sytuacja, gdy dokładność wyniku otrzymanego z wyborem częściowym jest za mała, a wybór pełny daje dostatecznie mały błąd.

Zwróćmy uwagę, że aby rozwiązać zadanie, rozwiazujemy numerycznie dwa podzadania, tj. układy z macierzami trójkątnymi. Dla każdego p iloczyn wskaźników uwarunkowania tych podzadań, $\operatorname{cond}_p L$ i $\operatorname{cond}_p U$, jest *zawsze* większy lub równy wskaźnikowi uwarunkowania całego zadania, $\operatorname{cond}_p A$. Ponadto dla dowolnych permutacji reprezentowanych przez macierze P i Q mamy $\operatorname{cond}_p A = \operatorname{cond}_p PAQ^T$. Wybór elementu głównego można interpretować jak dążenie do tego, aby iloczyn wskaźników uwarunkowania czynników rozkładu macierzy PA (lub PAQ^T) był możliwie mały.

Metoda eliminacji Gaussa z wyborem elementu głównego jest algorytmem numerycznie poprawnym, tj. istnieje macierz $\tilde{A}$, taka że zachodzi równość $P\tilde{A}Q^T = LU$ dla obliczonych macierzy trójkątnych L i U, oraz

$$\frac{\|\tilde{A} - A\|}{\|A\|} \leqslant F_n(A)\nu,$$

przy czym w tym wzorze jest użyta norma indukowana $\|\cdot\|_1$ lub $\|\cdot\|_\infty$. Liczba $F_n(A)$ jest stałą kumulacji; jeśli jest stosowany wybór częściowy, to ma ona duże oszacowanie ($F_n(A) \leqslant 3 \cdot 2^n - 5$), ale w praktyce stała ta jest prawie zawsze znacznie mniejsza; jej dokładniejsze oszacowanie zależy od wartości bezwzględnych obliczonych współczynników $a_{ij}^{(k)}$, a wybór elementu głównego jest pewną metodą przeciwdziałania wystąpieniu w obliczeniach wielkich liczb. Obliczenie wektora x przez rozwiązanie układów równań z macierzami L i U jest również numerycznie poprawnym algorytmem rozwiązywania układu równań liniowych. Rachunki w dowodzie tego twierdzenia są dosyć długie i żmudne.

Jeśli macierz A jest pełna, to wyznaczanie czynników trójkątnych ma koszt $(n^3 - n)/3$ operacji zmiennopozycyjnych (za jedną operację uznamy dzielenie lub mnożenie z dodawaniem), natomiast rozwiązywanie układów trójkątnych kosztuje n^2 takich operacji. W wielu zastosowaniach mamy do czynienia z macierzami rzadkimi, tj. mającymi dużo zerowych współczynników. W takich przypadkach *czynem karalnym*[2] jest użycie ogólnego algorytmu, odpowiedniego dla macierzy pełnych. Jeśli np. macierz jest wstęgowa, tj. istnieje $k \ll n$, takie że $a_{ij} = 0$ dla $|i - j| > k$, to odpowiedni wariant metody eliminacji Gaussa może znaleźć czynniki trójkątne kosztem rzędu $k^2 n$ operacji, a koszt rozwiązywania układów równań z tymi czynnikami jest rzędu kn. I tego wariantu należy użyć.

[2]Na podstawie §186 Kodeksu Etyki Zawodowej, Kompetencji i Przyzwoitości należy się za to od 1 do 3 lat Kompromitacji, a za notoryczną recydywę grozi Kompromitacja Dożywotnia.

Metoda odbić Householdera

Inną metodą doprowadzenia układu równań liniowych do układu równoważnego z macierzą trójkątną jest metoda odbić Householdera. Przypomnijmy własności odbić symetrycznych w $\mathbb{R}^n$. Niech v oznacza dowolny wektor jednostkowy (w sensie normy drugiej, tj. taki że $\|v\|_2 = 1$). Odbicie symetryczne względem hiperpłaszczyzny prostopadłej do wektora v jest określone wzorem

$$Hx = x - 2vv^T x.$$

Macierz odbicia jest więc równa

$$H = I - 2vv^T.$$

Odbicie jest inwolucją, tj. swoją własną odwrotnością:

$$H^2 = (I - 2vv^T)(I - 2vv^T) = I - 4vv^T + 4v\underbrace{v^T v}_{=1}v^T = I.$$

Macierz odbicia jest ponadto symetryczna, a zatem $H^T H = H^2 = I$, czyli macierz H jest ortogonalna. Tak więc odbicie jest izometrią; dla dowolnych wektorów $x, y \in \mathbb{R}^n$ zachodzi równość

$$\langle Hx, Hy\rangle = y^T H^T H x = y^T x = \langle x, y\rangle,$$

skąd dalej wynika, że dla dowolnego wektora x jest $\|Hx\|_2 = \|x\|_2$. Jeśli wektor v jest niezerowy, ale niekoniecznie jednostkowy, to macierz odbicia symetrycznego względem hiperpłaszczyzny prostopadłej do niego jest określona wzorem

$$H = I - v\frac{2}{v^T v}v^T.$$

W algorytmach numerycznych, jeśli to nie jest wynikiem, który koniecznie musimy otrzymać, nigdy nie wyznaczamy jawnie macierzy H. Mając wektor v, możemy obliczyć liczbę $\gamma = \frac{2}{v^T v}$, a następnie, chcąc obliczyć obraz y dowolnego wektora x w odbiciu, obliczamy kolejno

$$\begin{aligned} s &= v^T x, \\ t &= \gamma s, \\ y &= x - tv. \end{aligned}$$

W tym obliczeniu należy wykonać $2n + 1$ operacji (mnożeń z dodawaniami), podczas gdy mnożenie wektora x przez macierz H ma koszt n^2 operacji; ponadto metoda z macierzą H reprezentowaną jawnie jest znacznie gorsza ze względu na skutki błędów zaokrągleń. Zauważmy jeszcze jedną własność macierzy odbicia:

jeśli k-ta współrzędna wektora $\boldsymbol{v}$ jest równa 0, to k-ty wiersz i k-ta kolumna macierzy H są takie, jak w macierzy jednostkowej. Wtedy k-ta współrzędna wektora $H\boldsymbol{x}$ jest identyczna, jak k-ta współrzędna wektora $\boldsymbol{x}$. Zatem każda zerowa współrzędna wektora $\boldsymbol{v}$ (jeśli wiemy, które to są) umożliwia zaoszczędzenie dwóch operacji w powyższym obliczeniu.

Niech $\boldsymbol{a}$ oznacza pewien wektor w $\mathbb{R}^n$. Niech $\boldsymbol{e}$ oznacza pewien ustalony wektor jednostkowy (tj. $\|\boldsymbol{e}\|_2 = 1$). Odbicie Householdera jest to odbicie H skonstruowane w taki sposób, aby wektor $H\boldsymbol{a}$ miał kierunek wektora $\boldsymbol{e}$. Musi być zatem $H\boldsymbol{a} = \pm\|\boldsymbol{a}\|_2\boldsymbol{e}$. To zaś oznacza, że wektor normalny hiperpłaszczyzny odbicia, $\boldsymbol{v}$, musi mieć kierunek wektora $\boldsymbol{a} - \|\boldsymbol{a}\|_2\boldsymbol{e}$, albo $\boldsymbol{a} + \|\boldsymbol{a}\|_2\boldsymbol{e}$. Chcąc zmniejszyć skutki błędów zaokrągleń, należy zawsze wybierać dłuższy z tych dwóch wektorów.

Zastosujmy teraz odbicia do przekształcania układu równań liniowych $A\boldsymbol{x} = \boldsymbol{b}$. W pierwszym kroku odbijemy kolumny $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ macierzy A i wektor prawej strony $\boldsymbol{b}$ tak, aby obraz $H_1\boldsymbol{a}_1$ pierwszej kolumny miał kierunek wektora $\boldsymbol{e}_1$. Powstanie układ $H_1A\boldsymbol{x} = H_1\boldsymbol{b}$, którego macierz ma zera w pierwszej kolumnie pod diagonalą (tj. wszystkie współczynniki oprócz pierwszego są zerowe). Jeśli teraz odrzucimy pierwsze równanie, to otrzymamy podukład, w którym nie występuje niewiadoma x_1. Podukład ten możemy dalej przekształcać w podobny sposób.

Na początku k-tego kroku mamy równoważny wyjściowemu układ równań $A^{(k-1)}\boldsymbol{x} = \boldsymbol{b}^{(k-1)}$, którego macierz $A^{(k-1)}$ ma zerowe współczynniki poniżej diagonali w pierwszych $k-1$ kolumnach. Przekształcamy macierz $\tilde{A}^{(k-1)}$, która jest prawym dolnym blokiem macierzy $A^{(k-1)}$ o wymiarach $n+1-k \times n+1-k$, oraz blok $\tilde{\boldsymbol{b}}^{(k-1)}$ wektora $\boldsymbol{b}^{(k-1)}$ złożony z jego ostatnich $n+1-k$ współczynników. W tym celu konstruujemy wektor $\tilde{\boldsymbol{v}}^{(k)} \in \mathbb{R}^{n+1-k}$, dany wzorem

$$\tilde{\boldsymbol{v}}^{(k)} = \tilde{\boldsymbol{a}}_1^{(k-1)} \mp \|\tilde{\boldsymbol{a}}_1^{(k-1)}\|_2\boldsymbol{e}_1,$$

w którym $\tilde{\boldsymbol{a}}_1^{(k-1)}$ oznacza pierwszą kolumnę macierzy $\tilde{A}^{(k-1)}$ (czyli „dolną część" k-tej kolumny macierzy $A^{(k-1)}$). Pierwsza współrzędna wektora $\boldsymbol{e}_1$ jest jedynką, pozostałe $n-k$ to zera. Aby wektor $\tilde{\boldsymbol{v}}^{(k)}$ był jak najdłuższy, wybieramy znak „+" jeśli pierwsza współrzędna wektora $\tilde{\boldsymbol{a}}_1^{(k-1)}$ jest dodatnia, a „−" w przeciwnym razie. Następnie obliczamy liczbę $\gamma_k = \frac{2}{\tilde{\boldsymbol{v}}^{(k)T}\tilde{\boldsymbol{v}}^{(k)}}$ i poddajemy kolumny macierzy $\tilde{A}^{(k-1)}$ i wektor $\tilde{\boldsymbol{b}}^{(k-1)}$ odbiciu. Nie ma przy tym potrzeby stosowania ogólnego wzoru do odbijania wektora $\tilde{\boldsymbol{a}}_1^{(k-1)}$, bo skądinąd wiemy, co z tego wyjdzie.

Przekształcenie kolumn bloku $\tilde{A}^{(k-1)}$ jest równoważne odbiciu kolumn macierzy $A^{(k-1)}$ względem hiperpłaszczyzny, której wektor normalny $\boldsymbol{v}^{(k)}$ składa się z $k-1$ zer i z wektora $\tilde{\boldsymbol{v}}^{(k)}$. Oznaczmy macierz tego odbicia literą $H^{(k)}$. Po

wykonaniu $n-1$ odbić mamy układ równań liniowych

$$R\boldsymbol{x} = Q^T\boldsymbol{b},$$

którego macierz

$$R = H^{(n-1)} \ldots H^{(1)}A = Q^TA$$

jest trójkątna górna. Mamy równość

$$A = QR, \quad \text{gdzie} \quad Q = H^{(1)} \ldots H^{(n-1)},$$

przy czym macierz Q, jako iloczyn macierzy ortogonalnych, jest ortogonalna. W ten sposób, za pomocą odbić symetrycznych, znaleźliśmy rozkład ortogonalno-trójkątny macierzy A.

Podobnie, jak w eliminacji Gaussa, przekształcanie prawej strony możemy wykonać później, ale w tym celu trzeba zapamiętać wektory $\tilde{\boldsymbol{v}}^{(k)}$ (i, aby nie obliczać ich ponownie, co kosztuje, liczby $\gamma^{(k)}$). W tym celu możemy użyć miejsc w tablicy początkowo zawierającej współczynniki macierzy A, ale potrzebujemy dla każdego wektora odbicia dwóch dodatkowych miejsc. Jeden z możliwych sposobów przechowywania wyników obliczeń jest taki:

$$\left[\begin{array}{ccccc}
r_{11} & r_{12} & r_{13} & \ldots & r_{1n} \\
\hline
v_2^{(1)} & r_{22} & r_{23} & \ldots & r_{2n} \\
\hline
v_3^{(1)} & v_3^{(2)} & r_{33} & \ldots & r_{3n} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
\hline
v_n^{(1)} & v_n^{(2)} & \ldots & v_n^{(n-1)} & r_{nn} \\
\hline
v_1^{(1)} & v_2^{(2)} & \ldots & v_{n-1}^{(n-1)} & \bullet \\
\hline
\gamma^{(1)} & \gamma^{(2)} & \ldots & \gamma^{(n-1)} & \bullet
\end{array}\right]$$

Symbole r_{ij} oznaczają tu współczynniki macierzy R, zaś $v_i^{(k)}$ oznaczają współrzędne wektora $\boldsymbol{v}^{(k)}$. Jest też możliwe zmieszczenie wyników obliczenia z wykorzystaniem tylko jednej dodatkowej zmiennej dla każdej kolumny, po przeskalowaniu wektorów $\boldsymbol{v}^{(k)}$. Jak poprzednio, wykonanie obliczeń *in situ* oznacza „zepsucie" tablicy współczynników macierzy A, zatem najlepiej, aby takiemu „zepsuciu" poddać kopię.

Złożoność obliczeniowa wyznaczania rozkładu QR macierzy $n \times n$ jest równa $\frac{2}{3}n^3 + O(n^2)$, jest zatem w przybliżeniu dwukrotnie większa niż eliminacja Gaussa. Z drugiej strony, odpadają koszty wybierania elementu głównego, zresztą decydujący wpływ na czas obliczeń ma efektywność wykorzystania pamięci podręcznej (*cache'a*) procesora przez implementację algorytmu, dlatego nie można

powiedzieć z góry, że eliminacja Gaussa działa dwukrotnie szybciej. Natomiast użycie izometrii (tj. przekształceń reprezentowanych przez macierze ortogonalne) daje bardzo dobre własności numeryczne algorytmu. Zauważmy, że oryginalne zadanie zastępujemy dwoma podzadaniami — układami równań $Q\mathbf{y} = \mathbf{b}$ i $R\mathbf{x} = \mathbf{y}$. Wskaźnik uwarunkowania w normie drugiej macierzy ortogonalnej Q jest równy 1 (bo $\|Q\|_2 = \|Q^{-1}\|_2 = 1$), zaś $\operatorname{cond}_2 R = \operatorname{cond}_2 A$.

Metoda Choleskiego

W wielu zastosowaniach należy rozwiązać układ $A\mathbf{x} = \mathbf{b}$, którego macierz A jest symetryczna i dodatnio określona. Dla takich macierzy można stosować eliminację Gaussa, przy czym okazuje się, że wybór elementu głównego jest niepotrzebny. Jednak symetria macierzy to okazja do zmniejszenia kosztu obliczeń o połowę. Takich okazji nie wypada marnować.

Metoda Choleskiego polega na rozłożeniu macierzy A na czynniki trójkątne, z których każdy jest transpozycją drugiego: $A = LL^T$, gdzie macierz L jest trójkątna dolna. Po znalezieniu takiego rozkładu można rozwiązać kolejno układy równań $L\mathbf{y} = \mathbf{b}$ i $L^T\mathbf{x} = \mathbf{y}$.

Współczynniki macierzy L można obliczyć, traktując równość $LL^T = A$ jak układ równań, którego rozwiązanie można obliczyć „po kolei"; ponieważ macierz A jest symetryczna, mamy $\frac{1}{2}n(n+1)$ danych niezależnych współczynników. Tyle samo współczynników ma na diagonali i pod nią poszukiwana macierz L. Zatem, dla i, j takich że $1 \leqslant j \leqslant i \leqslant n$ mamy równania

$$a_{ij} = \sum_{k=1}^{n} l_{ik}l_{jk} = \sum_{k=1}^{j} l_{ik}l_{jk} = \sum_{k=1}^{j-1} l_{ik}l_{jk} + l_{ij}l_{jj}.$$

Wyodrębniony składnik sumy powyżej umożliwia obliczenie współczynnika l_{ij}, jeśli znamy wszystkie pozostałe współczynniki macierzy L występujące w sumowanych iloczynach; można uporządkować równania w takiej kolejności, że to jest możliwe. Mianowicie, możemy obliczać kolejno

$$\left.\begin{aligned} l_{ij} &= \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}}{l_{jj}} \quad \text{dla } j = 1, \ldots, i-1, \\ l_{ii} &= \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, \end{aligned}\right\} \quad \text{dla } i = 1, \ldots, n.$$

Obliczenie można wykonać *in situ*, wpisując liczby l_{ij} natychmiast po obliczeniu w miejsce a_{ij} (a zatem, „psując" daną macierz A lub jej kopię). Trzeba podkreślić, że macierz A musi być nie tylko symetryczna, ale także dodatnio określona, aby powyższy algorytm był wykonalny (tj. aby istniała nieosobliwa macierz trójkątna dolna L, taka że $A = LL^T$). To jest warunek konieczny i dostateczny, aby wyrażenia, z których należy obliczać pierwiastki kwadratowe, miały dodatnie wartości.

Zauważmy, że jeśli początkowych k współczynników w i-tym wierszu (dla $k < i$) to zera, to również macierz L ma na początku i-tego wiersza k zerowych współczynników. Można to wykorzystać do efektywnego wykorzystania miejsca w pamięci, a także do zmniejszenia kosztu znajdowania rozkładu (np. jeśli macierz A jest wstęgowa). Jeśli macierz A jest pełna, to też można przechowywać tylko dolny jej trójkąt w tablicy o długości $\frac{1}{2}n(n+1)$. Dla macierzy pełnej znalezienie rozkładu wymaga wykonania ok. $\frac{1}{6}n^3$ operacji mnożenia z dodawaniem lub dzielenia i obliczenia n pierwiastków kwadratowych.

Układy i algorytmy blokowe

Wiele układów równań liniowych rozwiązywanych w praktycznych zastosowaniach ma macierze o specjalnych własnościach, które można wykorzystać do zmniejszenia kosztu rozwiązywania. Bardzo często macierz w naturalny sposób dzieli się na wyróżniające się jakoś bloki. W najprostszej sytuacji, niech

$$A = \begin{bmatrix} B & C \\ D & E \end{bmatrix}.$$

Przypuśćmy, że macierz $A \in \mathbb{R}^{n,n}$ jest nieosobliwa i blok $B \in \mathbb{R}^{k,k}$ dla pewnego $k \in \{1, \ldots, n-1\}$ też jest nieosobliwy. Podzielmy również prawą stronę i wektor niewiadomy na bloki:

$$\mathbf{b} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix}.$$

Układ $A\mathbf{x} = \mathbf{b}$ podzieliliśmy w ten sposób na dwa podukłady:

$$\begin{cases} B\mathbf{y} + C\mathbf{z} = \mathbf{p}, \\ D\mathbf{y} + E\mathbf{z} = \mathbf{q}. \end{cases}$$

Znając $\mathbf{z}$, moglibyśmy rozwiązać pierwszy podukład; jego rozwiązanie wyraża się wzorem

$$\mathbf{y} = B^{-1}(\mathbf{p} - C\mathbf{z}).$$

Wstawiamy to wyrażenie do drugiego podukładu; mamy

$$DB^{-1}(\mathbf{p} - C\mathbf{z}) + E\mathbf{z} = \mathbf{q},$$

czyli

$$(E - DB^{-1}C)z = q - DB^{-1}p.$$

Macierz $S = E - DB^{-1}C$ nazywa się macierzą Schura; jeśli macierze A i B są nieosobliwe, to również macierz S jest nieosobliwa. Możemy znaleźć rozwiązanie z układu z tą macierzą, a następnie blok y. Jeśli o macierzy B wiemy tylko tyle, że jest nieosobliwa, to należałoby ją rozłożyć na czynniki trójkątne (lub ortogonalny i trójkątny). Powiedzmy, że mamy macierz permutacji P i macierze trójkątne L i U, takie że $PB = LU$. Wtedy należy wykonać następujący algorytm:

1. Korzystając z macierzy P, L, U, rozwiąż (macierzowy) układ równań $BF = C$, a następnie oblicz macierz $S = E - DF$,
2. Rozwiąż układ równań $Bv = p$, a następnie oblicz wektor $w = q - Dv$,
3. Rozwiąż układ równań $Sz = w$; w tym celu należy wyznaczyć i wykorzystać jakiś użyteczny rozkład macierzy S,
4. Oblicz wektor $u = p - Cz$, a następnie, korzystając z macierzy P, L, U, rozwiąż układ równań $By = u$.
   Alternatywnie, oblicz wektor $t = Cz$, a następnie rozwiąż układ $Bs = t$ i oblicz $y = v - s$.

Z wyjątkiem ograniczenia możliwości wyboru elementu głównego, nie mamy tu żadnych zmian (w szczególności kosztu) w porównaniu ze zwykłą eliminacją Gaussa (choć pierwsze dwa kroki można wykonać równolegle). Jeśli jednak blok B jest macierzą symetryczną dodatnio określoną, to zamiast eliminacji Gaussa możemy użyć dwukrotnie tańszej metody Choleskiego. Jeśli zaś blok B jest na przykład macierzą diagonalną lub ortogonalną, to niezależnie od tego, jakie są pozostałe bloki macierzy A, układy równań z macierzą B możemy rozwiązywać znacznie mniejszym kosztem. Ponadto, gdyby blok B był macierzą odbicia symetrycznego, reprezentowaną przez wektor normalny hiperpłaszczyzny odbicia (lub iloczynem takich macierzy, reprezentowanych przez odpowiednie wektory), to jawne wyznaczanie współczynników macierzy B, po to by następnie rozwiązać układ równań z tą macierzą, byłoby przejawem *skrajnego niedołęstwa*. Dlatego powtarzam stały apel: *najpierw* należy się dowiedzieć jak najwięcej o zadaniu, a *potem* dobierać algorytm jego rozwiązywania.

Metody iteracyjne

Jeśli liczba n równań i niewiadomych jest wielka, to koszt metod bezpośrednich rozwiązywania takich układów jest zbyt duży. Często w zastosowaniach pojawiają

się układy z n rzędu tysięcy lub milionów. Bardzo często macierze układów w takich zastosowaniach są rzadkie, np. mają tylko $O(n)$ niezerowych współczynników, ale rozmieszczenie tych współczynników uniemożliwia stosowanie metod bezpośrednich (np. znalezienie metodą różnic skończonych rozwiązania przybliżonego równania różniczkowego Poissona w kwadracie sprowadza się do rozwiązania układu równań liniowych, którego macierz jest wstęgowa, przy czym wstęga ma szerokość rzędu $\sqrt{n}$, a w każdym wierszu jest co najwyżej 5 współczynników niezerowych).

Do rozwiązywania wielkich układów równań liniowych stosuje się metody iteracyjne. Mając początkowe przybliżenie x_0 rozwiązania α, metoda konstruuje elementy ciągu, $x_1, x_2, \ldots$, zbieżnego do α. Obliczenia przerywa się na podstawie kryteriów stopu podobnych, jak dla równań nieliniowych, tj. ustalonego limitu liczby iteracji, kryterium residualnego (residuum jest w tym przypadku wektor $b - Ax$) lub kryterium przyrostowego (opartego na badaniu wartości wyrażenia $\|x_{k+1} - x_k\|$). Podstawową operacją, wykonywaną w każdej iteracji, jest mnożenie pewnego wektora przez macierz układu A, lub inną macierz, skonstruowaną na podstawie A. Dzięki takiemu ograniczeniu można korzystać z bardzo oszczędnych reprezentacji macierzy, które są w istocie wykazami (tablicami lub listami) miejsc, w których są niezerowe współczynniki. Koszt mnożenia wektora przez macierz jest wtedy proporcjonalny do liczby tych współczynników.

Metody iteracji prostej

Metody iteracji prostej polegają na tym, że na podstawie macierzy A i wektora prawej strony b konstruuje się pewną macierz B i wektor t, przyjmuje początkowe przybliżenie rozwiązania, x_0, i w kolejnych iteracjach oblicza

$$x_{k+1} = Bx_k + t.$$

Macierz B i wektor t muszą być tak dobrane, aby zachodziła równość $\alpha = B\alpha + t$, tj. aby rozwiązanie było punktem stałym funkcji $\varphi(x) = Bx + t$. Ponadto, aby ciąg wektorów x_k był zbieżny do α dla dowolnego punktu startowego x_0, funkcja φ musi być przekształceniem zwężającym, a zatem pewna (dowolna) norma indukowana macierzy B musi być mniejsza od 1. Okazuje się, że warunkiem koniecznym i dostatecznym istnienia takiej normy, która dla macierzy B przyjmuje wartość mniejszą od 1, jest nierówność $\rho(B) < 1$, gdzie $\rho(B)$ jest to promień spektralny, tj. największa wartość bezwzględna wartości własnej macierzy B. Aby zbieżność była szybka, $\rho(B)$ musi być jak najmniejsze, ale możliwość skonstruowania macierzy B o małym promieniu spektralnym zależy od macierzy A (i w szczególności od jej wskaźnika uwarunkowania).

Metoda Jacobiego: macierz A przedstawiamy w postaci sumy $A = L + D + U$, gdzie macierz L powstaje z A przez zastąpienie zerami współczynników na i nad diagonalą, macierz D jest diagonalna, a macierz U ma zerowe współczynniki na i pod diagonalą. Układ $A\boldsymbol{x} = \boldsymbol{b}$ przepisujemy w postaci

$$(L + D + U)\boldsymbol{x} = \boldsymbol{b}, \quad \text{czyli} \quad D\boldsymbol{x} = \boldsymbol{b} - (L + U)\boldsymbol{x},$$

skąd otrzymujemy metodę:

$$\boldsymbol{x}_{k+1} = D^{-1}\big(\boldsymbol{b} - (L + U)\boldsymbol{x}_k\big).$$

W metodzie Jacobiego mamy zatem $B_J = -D^{-1}(L + U)$ oraz $\boldsymbol{t}_J = D^{-1}\boldsymbol{b}$.

W obliczeniach nie wyznaczamy macierzy D^{-1}, choć to jest łatwe; zamiast tego rozwiązujemy układ równań z macierzą diagonalną D (podobnie postępujemy w innych metodach). Warunek $\rho(B_J) < 1$ jest spełniony dla wielu macierzy A; łatwym do sprawdzenia przypadkiem jest macierz diagonalnie dominująca, tj. taka że $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ dla każdego i.

Metoda Gaussa-Seidela: jak poprzednio, bierzemy $A = L + D + U$, po czym piszemy układ

$$(L + D)\boldsymbol{x} = \boldsymbol{b} - U\boldsymbol{x},$$

skąd otrzymujemy układ równań do rozwiązania w każdej iteracji:

$$(L + D)\boldsymbol{x}_{k+1} = \boldsymbol{b} - U\boldsymbol{x}_k.$$

W metodzie Gaussa-Seidela mamy zatem $B_{GS} = -(L + D)^{-1}U$ i $\boldsymbol{t}_{GS} = (L + D)^{-1}\boldsymbol{b}$. Podobnie jak w metodzie Jacobiego, jeśli macierz A jest diagonalnie dominująca, to ciąg $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$ jest zbieżny do $\boldsymbol{\alpha}$ dla każdego $\boldsymbol{x}_0 \in \mathbb{R}^n$, przy czym zbieżność metody Gaussa-Seidela jest zwykle szybsza.

Metoda Richardsona: Wprowadzamy parametr τ i piszemy układ równoważny układowi $A\boldsymbol{x} = \boldsymbol{b}$:

$$x + \tau A\boldsymbol{x} = \boldsymbol{x} + \tau\boldsymbol{b},$$

skąd mamy

$$\boldsymbol{x} = (I - \tau A)\boldsymbol{x} + \tau\boldsymbol{b} = \boldsymbol{x} + \tau(\boldsymbol{b} - A\boldsymbol{x}).$$

Metoda Richardsona polega na obliczaniu wektorów

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \tau(\boldsymbol{b} - A\boldsymbol{x}_k).$$

W tej metodzie mamy $B_R = I - \tau A$, $\boldsymbol{t}_R = \tau\boldsymbol{b}$. Podstawowym problemem jest dobranie parametru τ tak, aby ciąg $(\boldsymbol{x}_k)_k$ był zbieżny i aby ta zbieżność była jak najszybsza. Jeszcze do tego wrócimy.

Metoda sprzężonych gradientów

Metoda sprzężonych gradientów (ang. *conjugate gradient method*, w skrócie *CG*) służy do rozwiązywania układu n równań liniowych $A\boldsymbol{x} = \boldsymbol{b}$ z macierzą A symetryczną i dodatnio określoną. Co ciekawe, z punktu widzenia algebry metoda ta jest metodą bezpośrednią, mianowicie wytwarza skończony ciąg $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ dla pewnego $m \leqslant n$; gdyby nie było błędów zaokrągleń (które oczywiście są), to ostatni element tego ciągu byłby rozwiązaniem $\boldsymbol{\alpha}$. W praktyce najczęściej metodę tę wykorzystuje się jak metodę iteracyjną, która wytwarza dostatecznie dokładne przybliżenie $\boldsymbol{x}_k$ rozwiązania dla pewnego k znacznie mniejszego niż n.

Podstawowym krokiem metody CG jest minimalizacja wielomianu kwadratowego wzdłuż pewnej prostej. Funkcja określona wzorem

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}$$

jest wielomianem drugiego stopnia zmiennych $x_1, \ldots, x_n$. Ponieważ macierz A jest dodatnio określona, funkcja f ma minimum; jej gradient jest równy

$$\nabla f(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b},$$

a więc wektor residuum, $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x} = -\nabla f(\boldsymbol{x})$, określa kierunek najszybszego spadku funkcji f w punkcie $\boldsymbol{x}$. Poszukiwane rozwiązanie $\boldsymbol{\alpha}$ jest właśnie tym punktem przestrzeni $\mathbb{R}^n$, w którym funkcja f przyjmuje wartość minimalną. Niech $\boldsymbol{x}_k$ oznacza bieżące przybliżenie rozwiązania, zaś $\boldsymbol{v}_k$ oznacza pewien niezerowy wektor. Zbiór $L = \{\boldsymbol{x} = \boldsymbol{x}_k + t\boldsymbol{v}_k \colon t \in \mathbb{R}\}$ jest prostą w $\mathbb{R}^n$. Podstawiając jako argument funkcji f wyrażenie $\boldsymbol{x}_k + t\boldsymbol{v}_k$, otrzymujemy wielomian jednej zmiennej,

$$g(t) = f(\boldsymbol{x}_k + t\boldsymbol{v}_k) = \frac{1}{2}\big(\boldsymbol{v}_k^T A\boldsymbol{v}_k t^2 - 2\boldsymbol{v}_k^T(\boldsymbol{b} - A\boldsymbol{x}_k)t + \boldsymbol{x}_k^T(A\boldsymbol{x}_k - 2\boldsymbol{b})\big).$$

Trójmian kwadratowy $g(t) = at^2 - 2bt + c$ ze współczynnikiem $a > 0$ przyjmuje wartość minimalną dla $t = b/a$. Zatem, obliczając

$$\begin{aligned}
\boldsymbol{r}_k &= \boldsymbol{b} - A\boldsymbol{x}_k, \\
t_k &= \frac{\boldsymbol{v}_k^T\boldsymbol{r}_k}{\boldsymbol{v}_k^T A\boldsymbol{v}_k}, \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + t_k\boldsymbol{v}_k,
\end{aligned}$$

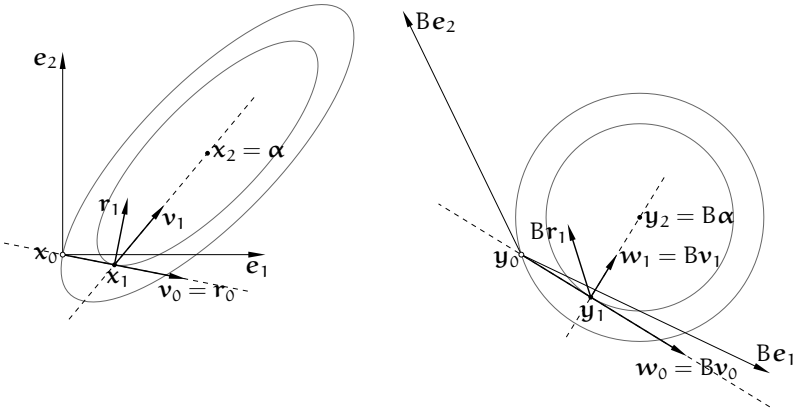
otrzymamy punkt $\boldsymbol{x}_{k+1}$ prostej L, w którym wartość funkcji f jest najmniejsza. Możemy zauważyć (lub przynajmniej sprawdzić), że następny wektor residuum, $\boldsymbol{r}_{k+1} = \boldsymbol{b} - A\boldsymbol{x}_{k+1} = \boldsymbol{r}_k - t_k A\boldsymbol{v}_k$, spełnia warunek $\boldsymbol{v}_k^T\boldsymbol{r}_{k+1} = 0$.

Istnieje rodzina metod iteracyjnych, w których powyższe obliczenie łączy się z pewną strategią wyboru wektorów $\boldsymbol{v}_k$, wyznaczających kierunki odpowiednich

prostych w kolejnych iteracjach; oczywiście, istotne są tylko *kierunki* tych wektorów. W metodzie CG, która należy do tej rodziny, wektory $\boldsymbol{v}_k$ mają kierunki sprzężone względem macierzy A, przez co rozumiemy, że spełniają równości $\boldsymbol{v}_i^T A \boldsymbol{v}_k = 0$ dla $i \neq k$. Przyjrzyjmy się tej własności i temu, co z niej wynika. Dla dowolnej symetrycznej i dodatnio określonej macierzy A istnieje (jednoznacznie określona) macierz B symetryczna i dodatnio określona[3], taka że $A = B^2$. Jeśli oznaczymy $\boldsymbol{w}_i = B\boldsymbol{v}_i$ dla każdego i, to dla $i \neq k$ mamy $\langle \boldsymbol{w}_k, \boldsymbol{w}_i \rangle = \boldsymbol{w}_i^T \boldsymbol{w}_k = 0$. Po podstawieniu $\boldsymbol{x} = B^{-1}\boldsymbol{y}$ jako argumentu funkcji f, otrzymujemy wielomian

$$h(\boldsymbol{y}) = f(B^{-1}\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^T B^{-T} A B^{-1} \boldsymbol{y} - \boldsymbol{y}^T B^{-T} \boldsymbol{b} = \frac{1}{2}\boldsymbol{y}^T \boldsymbol{y} - \boldsymbol{y}^T B^{-T} \boldsymbol{b},$$

którego część kwadratowa jest równa $\frac{1}{2}(y_1^2 + \cdots + y_n^2)$. Poszukiwanie minimum funkcji $f(\boldsymbol{x})$ wzdłuż prostych o kierunkach $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_k$, zaczynając od punktu $\boldsymbol{x}_0$, jest równoważne minimalizacji funkcji $h(\boldsymbol{y})$ wzdłuż prostych o wzajemnie prostopadłych kierunkach $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_k$, zaczynając od punktu $\boldsymbol{y}_0 = B\boldsymbol{x}_0$.



Obliczenie dla pewnego układu dwóch równań przedstawia rysunek. Wektor $\boldsymbol{v}_0 = \boldsymbol{r}_0$ jest prostopadły do przechodzącej przez $\boldsymbol{x}_0$ warstwicy funkcji f, która jest elipsą o środku $\boldsymbol{\alpha}$. Punkt $\boldsymbol{x}_1$ leży na kolejnej warstwicy; residuum w tym punkcie, $\boldsymbol{r}_1$, ma kierunek najszybszego spadku funkcji f, zaś wektor $\boldsymbol{v}_1$ wyznacza kierunek prostej łączącej punkt $\boldsymbol{x}_1$ i rozwiązanie $\boldsymbol{\alpha}$. Obrazy $\boldsymbol{w}_0$ i $\boldsymbol{w}_1$ wektorów $\boldsymbol{v}_0$ i $\boldsymbol{v}_1$ w przekształceniu liniowym określonym przez macierz B są prostopadłe do siebie; obrazami warstwic funkcji f, tj. warstwicami funkcji h, są okręgi. W ogólności do znalezienia w $\mathbb{R}^n$ minimum funkcji kwadratowej h, której warstwice są sferami,

[3] Istnieje macierz ortogonalna X i macierz diagonalna Λ z dodatnimi współczynnikami $\lambda_1, \ldots, \lambda_n$ na diagonali, takie że $A = X\Lambda X^T$; liczby $\lambda_1, \ldots, \lambda_n$ są wartościami własnymi macierzy A, a jej wektorami własnymi są kolumny macierzy X. Macierz B jest równa XMX^T, gdzie M jest macierzą diagonalną, z liczbami $\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}$ na diagonali.

wystarczy wykonać co najwyżej n kroków minimalizacji wzdłuż prostych wzajemnie prostopadłych.

W metodzie CG przyjmujemy $\boldsymbol{v}_0 = \boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ oraz

$$\boldsymbol{v}_{k+1} = \boldsymbol{r}_{k+1} + s_k \boldsymbol{v}_k, \text{ gdzie } s_k = -\frac{\boldsymbol{v}_k^T A \boldsymbol{r}_{k+1}}{\boldsymbol{v}_k^T A \boldsymbol{v}_k}, \text{ dla } k \geqslant 0, \text{ takiego że } \boldsymbol{v}_k \neq \boldsymbol{0}.$$

Otrzymane w ten sposób wektory $\boldsymbol{v}_k$ mają kierunki sprzężone względem macierzy A i metoda znajduje rozwiązanie po co najwyżej n iteracjach.

Szkic dowodu: Zauważamy, że dla każdego $j \leqslant k+1$ wektor $\boldsymbol{v}_j$ jest kombinacją liniową wektorów $\boldsymbol{r}_0, \ldots, \boldsymbol{r}_j$, a wektor $\boldsymbol{r}_j$ jest kombinacją liniową wektorów $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_j$. Niech K_j oznacza podprzestrzeń przestrzeni $\mathbb{R}^n$ rozpiętą przez te wektory[4]. Przed obliczeniem wektora $\boldsymbol{v}_{k+1}$ mamy wektory $\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{k+1}$ (oraz $\boldsymbol{v}_0, \ldots \boldsymbol{v}_k$), a zatem są określone przestrzenie $K_0 \subset \cdots \subset K_{k+1}$. Przyjmujemy założenie indukcyjne, że dla każdego wektora $\boldsymbol{u} \in K_i$, gdzie $0 \leqslant i < j \leqslant k$, jest $\boldsymbol{u}^T A \boldsymbol{v}_j = \boldsymbol{u}^T \boldsymbol{r}_j = 0$. Równoważnie, dla $0 \leqslant i < j \leqslant k$ są spełnione równości $\boldsymbol{v}_i^T A \boldsymbol{v}_j = \boldsymbol{r}_i^T A \boldsymbol{v}_j = \boldsymbol{v}_i^T \boldsymbol{r}_j = \boldsymbol{r}_i^T \boldsymbol{r}_j = 0$.

Wiemy, że residuum w punkcie $\boldsymbol{x}_{k+1}$ spełnia warunek $\boldsymbol{v}_k^T \boldsymbol{r}_{k+1} = 0$. Mamy też

$$\boldsymbol{v}_k^T A \boldsymbol{v}_{k+1} = \boldsymbol{v}_k^T A(\boldsymbol{r}_{k+1} + s_k \boldsymbol{v}_k) = \boldsymbol{v}_k^T A \boldsymbol{r}_{k+1} - \frac{\boldsymbol{v}_k^T A \boldsymbol{r}_{k+1}}{\boldsymbol{v}_k^T A \boldsymbol{v}_k} \boldsymbol{v}_k^T A \boldsymbol{v}_k = 0.$$

Niech $i < k$. Jeśli $\boldsymbol{u} \in K_i$, to

$$\boldsymbol{u}^T \boldsymbol{r}_{k+1} = \boldsymbol{u}^T(\boldsymbol{r}_k - t_k A \boldsymbol{v}_k) = \boldsymbol{u}^T \boldsymbol{r}_k - t_k \boldsymbol{u}^T A \boldsymbol{v}_k = 0.$$

Ponieważ $\boldsymbol{v}_{k+1} = \boldsymbol{r}_{k+1} + s_k \boldsymbol{v}_k$ oraz $t_i A \boldsymbol{v}_i = \boldsymbol{r}_i - \boldsymbol{r}_{i+1} \in K_{i+1} \subset K_k$, mamy również

$$\boldsymbol{v}_i^T A \boldsymbol{v}_{k+1} = \boldsymbol{v}_i^T A(\boldsymbol{r}_{k+1} + s_k \boldsymbol{v}_k) = \boldsymbol{v}_i^T A \boldsymbol{r}_{k+1} + s_k \underbrace{\boldsymbol{v}_i^T A \boldsymbol{v}_k}_{=0} = \frac{1}{t_i}(\boldsymbol{r}_i - \boldsymbol{r}_{i+1})^T \boldsymbol{r}_{k+1} = 0.$$

Z założenia indukcyjnego i wykazanych wyżej równości wynika, że $\boldsymbol{u}^T A \boldsymbol{v}_{k+1} = \boldsymbol{u}^T \boldsymbol{r}_{k+1} = 0$ dla każdego wektora $\boldsymbol{u} \in K_k$.

Z przeprowadzonego wyżej rachunku wynika, że wektory $\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots$ są do siebie nawzajem prostopadłe. Ale ciąg ortogonalny w $\mathbb{R}^n$ może składać się z co najwyżej n niezerowych wektorów, zatem residuum w punkcie $\boldsymbol{x}_k$ otrzymanym po co najwyżej n iteracjach metody CG musi być zerowe. □

Wyrażenie s_k można przekształcić, aby zmniejszyć koszt jego obliczania:

$$s_k = -\frac{\boldsymbol{v}_k^T A \boldsymbol{r}_{k+1}}{\boldsymbol{v}_k^T A \boldsymbol{v}_k} = -\frac{1}{t_k} \frac{(\boldsymbol{r}_k - \boldsymbol{r}_{k+1})^T \boldsymbol{r}_{k+1}}{\boldsymbol{v}_k^T A \boldsymbol{v}_k} = \frac{\boldsymbol{v}_k^T A \boldsymbol{v}_k}{\boldsymbol{v}_k^T \boldsymbol{r}_k} \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{v}_k^T A \boldsymbol{v}_k} = \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k}.$$

[4] Przestrzenie $K_j = \lin\{\boldsymbol{r}_0, \ldots, \boldsymbol{r}_j\} = \lin\{\boldsymbol{v}_0, \ldots, \boldsymbol{v}_j\}$ są nazywane przestrzeniami Kryłowa.

Teoretycznie (a dokładniej, na podstawie takiej teorii, która nie przewiduje błędów zaokrągleń) obliczenia należy zakończyć po otrzymaniu $\mathbf{r}_k = 0$.
W implementacji korzystającej z arytmetyki zmiennopozycyjnej obliczenia są przerywane, gdy wektor $\boldsymbol{\nu}_k$ lub $\mathbf{r}_{k+1}$ ma dostatecznie małą normę drugą.
Metodę CG realizuje następujący podprogram:

```
r = b − Ax;  /* r = r_0 */
ν = r;       /* ν = ν_0 */
c = r^T r;
for ( k = 0; k < n; k++ ) {
  if ( ν^T ν < δ^2 ) return;
  z = Aν;
  t = c/(ν^T z);  /* t = t_k */
  x = x + tν;  /* x = x_{k+1} */
  r = r − tz;  /* r = r_{k+1} */
  d = r^T r;
  if ( d < ε^2 ) return;
  ν = r + (d/c)ν;  /* ν = ν_{k+1} */
  c = d;
}
```

Tablica $\mathbf{x}$ początkowo zawiera współrzędne wektora $\mathbf{x}_0$. Obliczone przybliżenie rozwiązania jest końcową zawartością tej tablicy; oprócz niej podprogram używa jeszcze trzech tablic o długości n. Mnożenie wektora przez macierz A może być realizowane przez podprogram podany jako parametr i będący „czarną skrzynką" dla implementacji metody. Parametry δ i ε określają kryteria stopu.

Poprawianie uwarunkowania

Rozważmy metodę Richardsona. Jeśli macierz A jest symetryczna i dodatnio określona, to jej wartości własne są rzeczywiste i dodatnie, zawarte w przedziale $[\lambda_{\min}, \lambda_{\max}]$. Macierz B_R o najmniejszym promieniu spektralnym otrzymamy, przyjmując

$$\tau = \tau_{\text{opt}} = \frac{2}{\lambda_{\min} + \lambda_{\max}}.$$

Często mamy pewne informacje na temat wartości własnych macierzy A, co umożliwia dobranie optymalnego lub prawie optymalnego parametru.
Przyjrzyjmy się jednak szybkości zbieżności. Promień spektralny macierzy $B_R = I - \tau_{\text{opt}} A$ (który dla macierzy symetrycznej jest równy jej normie drugiej indukowanej) jest równy

$$\rho(B_R) = 1 - \tau_{\text{opt}}\lambda_{\min} = 1 - \frac{2\lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}.$$

Dla macierzy symetrycznej i dodatnio określonej (założyliśmy, że taka jest macierz A) jest $\operatorname{cond}_2 A = \lambda_{\max}/\lambda_{\min}$. Korzystając z tej formuły, otrzymujemy

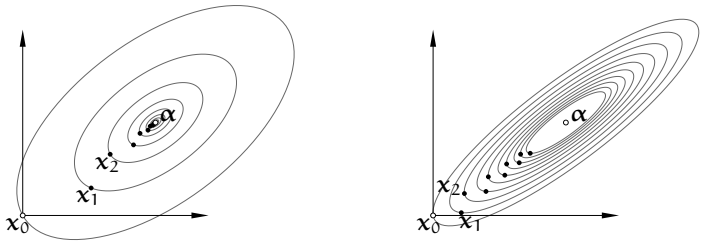
$$\rho(B_R) = \frac{\operatorname{cond}_2 A - 1}{\operatorname{cond}_2 A + 1}.$$

Tak więc, nawet jeśli wybierzemy optymalnie parametr τ, jeśli macierz A ma wielki wskaźnik uwarunkowania, zbieżność konstruowanego przez metodę Richardsona ciągu $(\mathbf{x}_k)_{k\in\mathbb{N}}$ do rozwiązania jest bardzo wolna. Podobne spostrzeżenie dotyczy także innych metod iteracyjnych. W szczególności dla metody CG błąd $\boldsymbol{\varepsilon}_k = \mathbf{x}_k - \boldsymbol{\alpha}$ ma oszacowanie (z macierzą $B = B^T$, taką że $B^2 = A$)

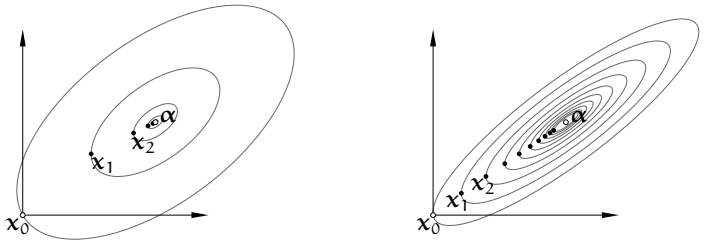
$$\|B\boldsymbol{\varepsilon}_k\|_2 \leqslant 2\left(\frac{\sqrt{\operatorname{cond}_2 A} - 1}{\sqrt{\operatorname{cond}_2 A} + 1}\right)^k \|B\boldsymbol{\varepsilon}_0\|_2.$$

---

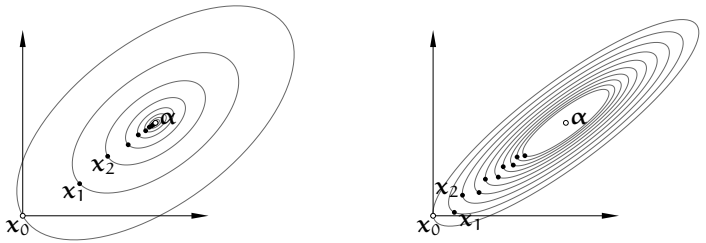
metoda Jacobiego:



metoda Gaussa-Seidela:



metoda Richardsona:



Rysunek przedstawia eksperyment, w którym były rozwiązywane dwa układy równań o tym samym rozwiązaniu, których macierze symetryczne i dodatnio określone mają wskaźniki uwarunkowania odpowiednio 4 i 16. Pokazane są ciągi przybliżeń rozwiązania wygenerowane przez opisane wcześniej metody (dla metody Richardsona przyjęty został optymalny parametr τ) i warstwice wielomianu kwadratowego f rozważanego w opisie metody CG, przechodzące przez kolejne punkty $\mathbf{x}_k$.

Zbieżność metod iteracyjnych można przyspieszyć, zastępując układ dany układem równoważnym, którego macierz ma mniejszy wskaźnik uwarunkowania. Cel ten można osiągnąć za pomocą macierzy C o następujących własnościach: układy równań z macierzą C są łatwe do rozwiązania, i zachodzi nierówność $\operatorname{cond}(C^{-1}A) \ll \operatorname{cond} A$; aby tak było, macierz C musi w jakiś sposób przybliżać macierz A. Metodę iteracyjną stosujemy do układu $C^{-1}A\mathbf{x} = C^{-1}\mathbf{b}$, przy czym nigdy nie wyznaczamy macierzy $C^{-1}A$; zamiast tego, mając obliczyć wektor $\mathbf{u} = C^{-1}A\mathbf{v}$, obliczamy $\mathbf{w} = A\mathbf{v}$ i rozwiązujemy układ $C\mathbf{u} = \mathbf{w}$.

Nawet jeśli macierze A i C są symetryczne, macierz $C^{-1}A$ na ogół nie jest taka. Niektóre metody, na przykład CG, wymagają, aby macierz układu była symetryczna. Aby zachować symetrię, wybieramy taką macierz C, aby układy równań z nią były łatwe do rozwiązania i aby było $\operatorname{cond}(C^{-1}AC^{-T}) \ll \operatorname{cond} A$. Układ dany zastępujemy układem równań

$$C^{-1}AC^{-T}\mathbf{y} = C^{-1}\mathbf{b}, \qquad (*)$$

po rozwiązaniu którego można obliczyć $\mathbf{x} = C^{-T}\mathbf{y}$; implementację metody CG można zmodyfikować tak, aby to ostatnie obliczenie było niepotrzebne — residua układu (*) możemy obliczać na podstawie wzoru[5]

$$\mathbf{r}_{k+1} = C^{-1}\mathbf{b} - C^{-1}AC^{-T}\mathbf{y}_{k+1} = C^{-1}(\mathbf{b} - A\mathbf{x}_{k+1}) = \mathbf{r}_k - t_k C^{-1}A\mathbf{v}_k.$$

Zastępowanie układu równań układem o macierzy lepiej uwarunkowanej ma angielską nazwę *preconditioning*, a używana do tego macierz C to tzw. *preconditioner*; terminy te nie mają powszechnie przyjętych polskich odpowiedników. Metody znajdowania odpowiednich macierzy są silnie związane

[5] Jak zawsze, mając jawną reprezentację macierzy C, *nie mnożymy* wektora przez C^{-1}, tylko rozwiązujemy odpowiedni układ równań. Dalsze przekształcenia prowadzą do otrzymania wzorów, w których poprawianie uwarunkowania układu rozwiazywanego za pomocą metody CG jest realizowane za pomocą (symetrycznej i dodatnio określonej) macierzy $S = CC^T$, która powinna przybliżać macierz A, ale być od niej „łatwiejsza"; w opartej na tych wzorach implementacji metody CG macierz C nie jest potrzebna, natomiast w kolejnych iteracjach trzeba rozwiązywać układy równań z macierzą S.

ze specyfiką zadania i daleko wykraczają poza ten wykład. Niemniej, warto wiedzieć, że metody iteracyjne z poprawianiem uwarunkowania są w zasadzie jedynymi skutecznymi metodami rozwiązywania *naprawdę wielkich* układów równań liniowych (które w praktyce często są *naprawdę źle* uwarunkowane).

Zadania i problemy

1. Znajdź wyrażenie opisujące wskaźnik uwarunkowania zadania obliczania iloczynu danej macierzy nieosobliwej A i wektora $\mathbf{x}$.
2. Udowodnij, że jeśli macierz kwadratowa nieosobliwa A jest iloczynem macierzy kwadratowych B i C, to (dla wskaźników uwarunkowania określonych za pomocą dowolnej normy indukowanej) zachodzi nierówność $\operatorname{cond} A \leqslant \operatorname{cond} B \operatorname{cond} C$.
3. Rozwiąż układy równań liniowych

$$\begin{bmatrix} 1 & 1 \\ 1 & 0.98 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 1 \\ 1 & 0.99 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

   Oblicz wskaźnik uwarunkowania macierzy pierwszego układu w normie pierwszej.
4. Udowodnij, że jeśli macierz A $n \times n$ jest symetryczna i współczynnik $a_{11} \neq 0$, to po wykonaniu pierwszego kroku eliminacji Gaussa blok $n-1 \times n-1$ w prawym dolnym rogu przekształconej macierzy jest symetryczny.
5. Napisz procedury eliminacji Gaussa bez wyboru elementu głównego dla macierzy trójdiagonalnej i cyklicznej trójdiagonalnej.
6. Macierz $n \times n$

$$A = \begin{bmatrix} H & \mathbf{d} \\ \mathbf{d}^T & c \end{bmatrix}$$

   ma blok $H = I - 2\boldsymbol{w}\boldsymbol{w}^T$, gdzie $\boldsymbol{w} \in \mathbb{R}^{n-1}$ jest wektorem jednostkowym. Jaki warunek muszą spełniać wektory $\boldsymbol{w}$ i $\mathbf{d}$ oraz liczba c, aby macierz A była nieosobliwa? Zaproponuj algorytm rozwiązywania układów równań liniowych z taką macierzą, przy założeniu, że dane są wektory $\boldsymbol{w}$ i $\mathbf{d}$, liczba c i wektor prawej strony.
7. Macierz o strukturze blokowej

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix},$$

   jeśli blok A jest symetryczny i dodatnio określony, a kolumny macierzy B są liniowo niezależne, jest nieosobliwa. Zaproponuj korzystający z tych informacji algorytm rozwiązywania układu równań liniowych z taką macierzą.

   Wskazówka: Po obliczeniu macierzy $C = A^{-1}B$, użyj odbić Householdera do znalezienia macierzy ortogonalnej Q i trójkątnej górnej R, takich że $C = QR$ oraz $C^TC = R^TR$.
8. Zmodyfikuj procedurę CG z wykładu tak, aby obliczenia były wykonywane dla układu z poprawionym uwarunkowaniem.

9. Metodą Choleskiego rozłóż na czynniki trójkątne macierz

$$A = \begin{bmatrix} 4 & 2 & 0 & 2 \\ 2 & 2 & 0 & 1 \\ 0 & 0 & 4 & -6 \\ 2 & 1 & -6 & 12 \end{bmatrix}$$

   Korzystając z tego rozkładu rozwiąż układ równań $A\boldsymbol{x} = \mathbf{b}$ z wektorem prawej strony $\mathbf{b} = [4, 1, -6, 13]^T$.

Programy i projekty

1. Eliminacja Gaussa. Podane niżej procedury są implementacją metody eliminacji Gaussa z częściowym wyborem elementu głównego. Są one tu zamieszczone po to, by pokazać, jak to wygląda. W praktyce znacznie lepiej jest używać bardzo dobrze zaimplementowanych (w szczególności zoptymalizowanych ze względu na szybkość) i gruntownie przetestowanych procedur z biblioteki LAPACK (tak, jak we wcześniej zrealizowanym eksperymencie z metodą Newtona dla układu równań nieliniowych).

```
#define FLOAT float

#define IND(n,i,j) ((n)*(i)+(j))

char LUDecomp ( int n, FLOAT *a, int *permut )
{
  int  i, j, k;
  FLOAT m, mmax;

       /* rozklad metoda eliminacji Gaussa */
  for ( j = 0; j < n-1; j++ ) {
         /* wybor elementu glownego */
    mmax = fabs(a[IND(n,j,j)]);
    k = j;
    for ( i = j+1; i < n; i++ ) {
      m = fabs(a[IND(n,i,j)]);
      if ( m > mmax ) { mmax = m; k = i; }
    }
         /* przestawianie wierszy */
    if ( (permut[j] = k) != j )
      for ( i = 0; i < n; i++ )
        { m = a[IND(n,j,i)]; a[IND(n,j,i)] = a[IND(n,k,i)];
          a[IND(n,k,i)] = m; }
         /* wlasciwa eliminacja Gaussa */
    if ( a[IND(n,j,j)] == 0.0 )
      return 0;
    for ( i = j+1; i < n; i++ ) {
      m = a[IND(n,i,j)] = a[IND(n,i,j)]/a[IND(n,j,j)];
      for ( k = j+1; k < n; k++ )
        a[IND(n,i,k)] -= a[IND(n,j,k)]*m;
    }
  }
  return 1;
} /*LUDecomp*/

void LUSolve ( int n, FLOAT *a, int *permut, FLOAT *b )
{
  int  i, j;
  FLOAT m;

       /* przestawianie wspolczynnikow prawej strony */
  for ( i = 0; i < n-1; i++ )
    if ( (j = permut[i]) != i )
      { m = b[i]; b[i] = b[j]; b[j] = m; }
       /* rozwiazywanie ukladu L*y = b */
  for ( i = 1; i < n; i++ )
    for ( j = 0; j < i; j++ )
      b[i] -= a[IND(n,i,j)]*b[j];
       /* rozwiazywanie ukladu U*x = y */
  for ( i = n-1; i >= 0; i-- ) {
    for ( j = i+1; j < n; j++ )
      b[i] -= a[IND(n,i,j)]*b[j];
    b[i] /= a[IND(n,i,i)];
  }
} /*LUSolve*/
```

Procedura LUDecomp dokonuje rozkładu macierzy PA na czynniki trójkątne L i U, *in situ*, tj. wpisując współczynniki macierzy L i U do tablicy, w której początkowo są dane współczynniki macierzy A. Reprezentacja macierzy P jest zapisywana w tablicy permut.

Procedure LUSolve rozwiązuje układy z macierzami trójkątnymi, zastępując w tablicy b wektor prawej strony obliczonym rozwiązaniem.

2. Numeryczne obliczanie wyznacznika. Wyznacznika macierzy kwadratowej, zwłaszcza dużej, *nie należy obliczać w ogóle*, chyba że naprawdę nie można tego uniknąć. Jeśli naprawdę nie można, to należy posłużyć się rozkładem trójkątno-trójkątnym lub ortogonalno-trójkątnym. Wyznacznik macierzy trójkątnej jest iloczynem jej współczynników na diagonali. Wyznacznik macierzy permutacji jest równy ± 1, zależnie od parzystości liczby transpozycji, których złożeniem jest ta permutacja. Wyznacznik macierzy ortogonalnej $Q = H_1 \ldots H_k$, będącej iloczynem macierzy odbić, jest równy ± 1, zależnie od parzystości liczby odbić. Zamieszczone niżej procedury realizują dwa algorytmy obliczania wyznacznika: opartą na rozkładzie znalezionym za pomocą eliminacji Gaussa (z wykorzystaniem procedury LUDecomp) i naiwną implementację rozwinięcia Laplace'a. Przeprowadź eksperymenty z tymi procedurami i porównaj wyniki oraz czasy ich obliczania.

```
#define FLOAT float

#define IND(n,i,j) ((n)*(i)+(j))

char LUDecomp ( int n, FLOAT *a, int *permut );

FLOAT detLU ( int n, FLOAT *a, char *error )
{
  int  *permut, i;
  FLOAT det;
  char  chs;

  *error = 1;
  permut = (int*)malloc ( n*sizeof(int) );
  if ( permut ) {
    if ( LUDecomp ( n, a, permut ) ) {
      for ( det = a[IND(n,0,0)], i = 1; i < n; i++ )
        det *= a[IND(n,i,i)];
      for ( chs = 0, i = 0; i < n; i++ )
        if ( permut[i] != i )
          chs = !chs;
      if ( chs )
        det = -det;
      *error = 0;
    }
    else
      det = 0.0;
    free ( permut );
    return det;
  }
  else return 0.0;
} /*detLU*/

FLOAT r_detLaplace ( int n, FLOAT **a, char *error )
{
  FLOAT **s, det, dd;
  int  i;

  if ( n == 1 ) {
    *error = 0;
    return a[0][0];
  }
  else {
    s = (FLOAT**)malloc ( (n-1)*sizeof(FLOAT*) );
    if ( s ) {
      for ( i = 1; i < n; i++ )
        s[i-1] = &a[i][1];
      det = 0.0;
      for ( i = 0; i < n; i++ ) {
        dd = r_detLaplace ( n-1, s, error );
        if ( *error )
          goto way_out;
        if ( i & 0x01 )
          det -= a[i][0]*dd;
        else
          det += a[i][0]*dd;
        if ( i < n-1 )
          s[i] = &a[i][1];
      }
way_out:
      free ( s );
      return det;
    }
    else {
      *error = 1;
      return 0.0;
    }
```

```
  }
} /*r_detLaplace*/

FLOAT detLaplace ( int n, FLOAT *a, char *error )
{
  FLOAT **r, det;
  int   i;

  r = (FLOAT**)malloc ( n*sizeof(FLOAT*) );
  if ( r ) {
    for ( i = 0; i < n; i++ ) /* UWAGA, trik! */
      r[i] = &a[n*i];
    det = r_detLaplace ( n, r, error );
    free ( r );
    return det;
  }
  else {
    *error = 1;
    return 0.0;
  }
} /*detLaplace*/
```

3. Reprezentacje macierzy rzadkich

Jeśli większość współczynników macierzy to zera, to nie należy zajmować nimi pamięci operacyjnej. Jeśli macierz ma jakąś wyraźną strukturę, to można użyć specjalnie dobranej struktury danych do jej reprezentowania. W ogólnym przypadku można stosować reprezentację, w której dla każdego współczynnika niezerowego pamięta się trzy liczby: indeksy wiersza i kolumny oraz współczynnik. Takie trójki mogą być w osobnych tablicach lub w jednej tablicy struktur. Szczególnie łatwe jest naapisanie procedury mnożenia wektora przez macierz w takiej postaci.

Inna reprezentacja to format spakowanych wierszy (albo kolumn). W każdym wierszu znajdujemy pozycje niezerowych współczynników. Indeksy tych pozycji (tj. kolumn) zapisujemy w tablicy, dla kolejnych wierszy jeden za drugim. Do drugiej tablicy, indeksowanej numerem wiersza, wpisujemy indeks (do pierwszej tablicy) pierwszego numeru kolumny dla danego wiersza. Liczba niezerowych współczynników w wierszu jest różnicą kolejnych elementów drugiej tablicy.

Podany niżej program zawiera implementacje trzech algorytmów rozwiązywania układu równań liniowych metodą iteracji prostej. Macierz jest reprezentowana za pomocą tablicy struktur, zawierających indeksy i wartości niezerowych współczynników. Tablica ta w zasadzie może być nieuporządkowana, ale na potrzeby implementacji metody Gaussa-Seidela jest sortowana w kolejności rosnących indeksów wierszy, a w wierszu w kolejności rosnących indeksów kolumn. Macierz układu równań w rozpatrywanym zastosowaniu (rozwiązywaniu równania Poissona metodą różnic skończonych) jest symetryczna i diagonalnie dominująca.

.....

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>

#define FLOAT double

typedef struct {
    int  i, j;
    FLOAT aij;
  } spmelem;

/* mnozenie macierzy spakowanej przez wektor */
void MultSPMatV ( int n, int nnze, spmelem *mat, FLOAT *x, FLOAT *y )
{
  int k;

  memset ( y, 0, n*sizeof(FLOAT) );
  for ( k = 0; k < nnze; k++ )
    y[mat[k].i] += mat[k].aij*x[mat[k].j];
} /*MultSPMatV*/

FLOAT NormaResiduum ( int n, int nnze, spmelem *mat,
                      FLOAT *b, FLOAT *x )
{
  FLOAT *y, res;
  int   i;

  y = malloc ( n*sizeof(FLOAT) );
  if ( !y )
```

```
    exit ( 1 );  /* !!! */
  MultSPMatV ( n, nnze, mat, x, y );
  for ( res = 0.0, i = 0; i < n; i++ ) {
    y[i] -= b[i];
    res += y[i]*y[i];
  }
  free ( y );
  res = sqrt ( res );
printf ( "%f\n", res );
  return res;
} /*NormaResiduum*/

void Jacobi ( int n, int nnze, spmelem *mat,
              FLOAT *b, FLOAT *x, FLOAT eps )
{
  FLOAT *y, r0;
  int  k, i, j, iter;

  y = malloc ( n*sizeof(FLOAT) );
  if ( !y )
    return;
  r0 = NormaResiduum ( n, nnze, mat, b, x );
iter = 0;
  do {
iter ++;
    memcpy ( y, b, n*sizeof(FLOAT) );
    for ( k = 0; k < nnze; k++ ) {
      i = mat[k].i;
      j = mat[k].j;
      if ( i != j )
        y[i] -= mat[k].aij*x[j];
    }
    for ( k = 0; k < nnze; k++ ) {
      i = mat[k].i;
      if ( i == mat[k].j )
        x[i] = y[i]/mat[k].aij;
    }
  } while ( NormaResiduum ( n, nnze, mat, b, x ) > eps*r0 );
  free ( y );
printf ( "wykonanych %d iteracji\n", iter );
```



```
} /*Jacobi*/

void GaussSeidel ( int n, int nnze, spmelem *mat,
                   FLOAT *b, FLOAT *x, FLOAT eps )
{
} /*GaussSeidel*/

void Richardson ( int n, int nnze, spmelem *mat, FLOAT tau,
                  FLOAT *b, FLOAT *x, FLOAT eps )
{
} /*Richardson*/

void QuickSort ( int n, void *usrptr,
                 char (*less)(int,int,void*),
                 void (*swap)(int,int,void*) );

char my_less ( int i, int j, void *usrptr )
{
  spmelem *a;

  a = (spmelem*)usrptr;
  if ( a[i].i < a[j].i )
    return 1;
  else if ( a[i].i == a[j].i && a[i].j < a[j].j )
    return 1;
  else
    return 0;
} /*my_less*/

void my_swap ( int i, int j, void *usrptr )
{
  spmelem *a, b;

  a = (spmelem*)usrptr;
  b = a[i]; a[i] = a[j]; a[j] = b;
} /*my_swap*/

void SetupLapMat ( int M, int N, int *n, int *nnze,
                   spmelem **mat, FLOAT *hh )
{
```

```
  int    _n, _nnze, i, j, k;
  spmelem *_mat;
  FLOAT  hh1, hh2;

  *n = _n = N*M;
  *nnze = _nnze = 5*M*N - 2*(M+N);
  *mat = _mat = malloc ( _nnze*sizeof(spmelem) );
  if ( !_mat )
    return;
  memset ( _mat, 0, _nnze*sizeof(spmelem) );
      /* okreslanie niezerowych elementow */
  hh1 = (FLOAT)(N+1); hh1 *= hh1;
  hh2 = (FLOAT)(M+1); hh2 *= hh2;
  *hh = hh1*hh2;
      /* na diagonali */
  for ( k = 0; k < M*N; k++ ) {
    _mat[k].i = _mat[k].j = k;
    _mat[k].aij = 2.0*(hh1+hh2);
  }
      /* na kodiagonalach */
  for ( i = 0; i < N; i++ )
    for ( j = 0; j < M-1; j++, k += 2 ) {
      _mat[k].i = _mat[k+1].j = i*M+j+1;
      _mat[k].j = _mat[k+1].i = i*M+j;
      _mat[k].aij = _mat[k+1].aij = -hh1;
    }
      /* na diagonalach blokow kodiagonalnych */
  for ( i = 0; i < N-1; i++ )
    for ( j = 0; j < M; j++, k += 2 ) {
      _mat[k].i = _mat[k+1].j = i*M+j;
      _mat[k].j = _mat[k+1].i = (i+1)*M+j;
      _mat[k].aij = _mat[k+1].aij = -hh2;
    }
      /* sortuj */
  QuickSort ( _nnze, _mat, my_less, my_swap );
} /*SetupLapMat*/

void Test1 ( int M, int N )
{
#define EPS 1.0e-4
```



```
  int    i, n, nnze;
  spmelem *mat;
  FLOAT  hh, *x, *b;

  SetupLapMat ( M, N, &n, &nnze, &mat, &hh );
  if ( !mat )
    exit ( 1 );
  x = malloc ( 2*n*sizeof(FLOAT) );
  if ( !x )
    exit ( 1 );
  b = &x[n];
  for ( i = 0; i < n; i++ )
    b[i] = hh;
      /* teraz rozne metody iteracyjne rozwiazywania */
      /* ukladow rownan liniowych */
        /* startujemy od zera */
  printf ( "Metoda Jacobiego:\n" );
  memset ( x, 0, n*sizeof(FLOAT) );
  Jacobi ( n, nnze, mat, b, x, EPS );

  printf ( "Metoda Gaussa-Seidela:\n" );
  memset ( x, 0, n*sizeof(FLOAT) );
  GaussSeidel ( n, nnze, mat, b, x, EPS );

  printf ( "Metoda Richardsona:\n" );
  memset ( x, 0, n*sizeof(FLOAT) );
  Richardson ( n, nnze, mat, 0.01, b, x, EPS );

  free ( mat );
  free ( x );
#undef EPS
} /*Test1*/

int main ( void )
{
  Test1 ( 20, 20 );
  exit ( 0 );
} /*main*/
```

Liniowe zadania najmniejszych kwadratów

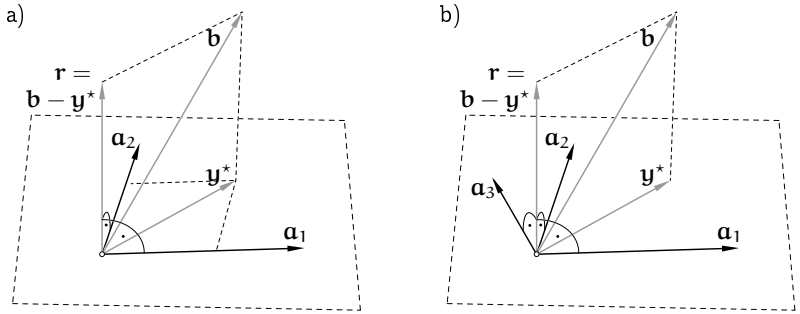
Rozważamy układ równań liniowych $A\mathbf{x} = \mathbf{b}$ z macierzą $A \in \mathbb{R}^{m,n}$ i wektorem $\mathbf{b} \in \mathbb{R}^m$. Układ ten może (choć nie musi) być sprzeczny. Liniowe zadanie najmniejszych kwadratów (LZNK) polega na znalezieniu wektora $\mathbf{x}^\star$, takiego że norma druga wektora residuum, $\mathbf{b} - A\mathbf{x}^\star$, jest najmniejsza. Jeśli układ jest niesprzeczny, to rozwiązanie LZNK jest zwykłym rozwiązaniem tego układu.

LZNK ma rozwiązanie; jest nim taki wektor $\mathbf{x}^\star$, że wektor residuum jest prostopadły (w sensie iloczynu skalarnego $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{v}^T\mathbf{u}$) do przestrzeni liniowej (podprzestrzeni $\mathbb{R}^m$) rozpiętej przez kolumny $\mathbf{a}_1, \ldots, \mathbf{a}_n$ macierzy A. Dla dowodu rozważmy wektor $\mathbf{y}^\star$, który jest rzutem prostopadłym wektora $\mathbf{b}$ na tę podprzestrzeń. Zatem istnieje wektor $\mathbf{x}^\star$, taki że $\mathbf{y}^\star = A\mathbf{x}^\star$ i wektor $\mathbf{b} - \mathbf{y}^\star = \mathbf{b} - A\mathbf{x}^\star$ (czyli residuum) jest prostopadły do tej podprzestrzeni. Jeśli weźmiemy dowolny wektor $\mathbf{x} \in \mathbb{R}^n$ i obliczymy $\mathbf{y} = A\mathbf{x}$, to wektor $\mathbf{y} - \mathbf{y}^\star = A(\mathbf{x} - \mathbf{x}^\star)$ jest prostopadły do wektora $\mathbf{b} - \mathbf{y}^\star$. Ale wtedy, na podstawie twierdzenia Pitagorasa, mamy

$$\|\mathbf{b} - A\mathbf{x}\|_2^2 = \|\mathbf{b} - \mathbf{y}\|_2^2 = \|\mathbf{b} - \mathbf{y}^\star\|_2^2 + \|\mathbf{y}^\star - \mathbf{y}\|_2^2 \geqslant \|\mathbf{b} - \mathbf{y}^\star\|_2^2 = \|\mathbf{b} - A\mathbf{x}^\star\|_2^2.$$

Dla $\mathbf{y} \neq \mathbf{y}^\star$ powyższa nierówność jest ostra. □

LZNK ma rozwiązanie jednoznaczne wtedy i tylko wtedy, gdy kolumny macierzy A są liniowo niezależne (co jest możliwe tylko dla $m \geqslant n$). Zadania z takimi macierzami to tzw. regularne liniowe zadania najmniejszych kwadratów (RLZNK). Jeśli macierz ma kolumny liniowo zależne, to zadanie (nieregularne, NLZNK) ma wiele rozwiązań, ich zbiór jest warstwą przestrzeni $\mathbb{R}^n$ o wymiarze $n - r$ (gdzie r oznacza rząd macierzy A).



Ilustracje liniowych zadań najmniejszych kwadratów mamy na rysunku. Rysunek a) przedstawia zadanie regularne dla układu z macierzą 3×2. Kolumny

macierzy $A = [\mathbf{a}_1, \mathbf{a}_2]$ rozpinają dwuwymiarową podprzestrzeń przestrzeni $\mathbb{R}^3$, nie zawierającą wektora prawej strony $\mathbf{b}$. Ponieważ kolumny te są liniowo niezależne, rzut prostopadły $\mathbf{y}^\star$ wektora $\mathbf{b}$ na tę podprzestrzeń jest ich kombinacją liniową o jednoznacznie określonych współczynnikach — współrzędnych wektora $\mathbf{x}^\star$, który jest jedynym rozwiązaniem tego zadania.

Na rysunku b) jest pokazane zadanie nieregularne, z macierzą 3×3 o liniowo zależnych kolumnach. Kolumny te rozpinają przestrzeń dwuwymiarową, której elementem wektor $\mathbf{b}$ nie jest. Jego rzut prostopadły $\mathbf{y}^\star$ na tę podprzestrzeń jest jednoznacznie określony, ale można go wyrazić jako kombinację liniową kolumn $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ na nieskończenie wiele sposobów i właśnie tyle rozwiązań ma zadanie.

Regularne LZNK

Prostopadłość dowolnego wektora w $\mathbb{R}^m$ do podprzestrzeni jest równoważna prostopadłości tego wektora do wszystkich elementów dowolnej bazy tej podprzestrzeni. Zatem, mając układ równań $A\mathbf{x} = \mathbf{b}$, możemy pomnożyć skalarnie residuum przez kolumny macierzy A i przyrównać do zera:

$$\langle \mathbf{b} - A\mathbf{x}, \mathbf{a}_i \rangle = 0, \quad i = 1, \ldots, n.$$

Można to zapisać w postaci macierzowej, po prostych przekształceniach otrzymując tzw. układ równań normalnych:

$$A^TA\mathbf{x} = A^T\mathbf{b}.$$

Jeśli kolumny $\mathbf{a}_1, \ldots, \mathbf{a}_n$ są liniowo niezależne, to ich zbiór jest bazą odpowiedniej podprzestrzeni; wtedy macierz symetryczna $M = A^TA$ jest dodatnio określona (skąd wynika, że nieosobliwa) i układ ma jednoznaczne rozwiązanie — rozwiązanie RLZNK (jeśli kolumny są liniowo zależne, to układ równań normalnych jest niesprzeczny, ale ma nieskończenie wiele rozwiązań, którymi są wszystkie rozwiązania NLZNK).

Algorytm równań normalnych jest najprostszą i najtańszą metodą numeryczną rozwiązywania RLZNK. Polega on na obliczeniu macierzy $M = A^TA$ i wektora $\mathbf{d} = A^T\mathbf{b}$, a następnie rozwiązaniu układu równań $M\mathbf{x} = \mathbf{d}$. Ponieważ macierz M jest symetryczna, obliczenie jej współczynników może być wykonane kosztem $mn(n+1)/2$ działań (mnożeń i dodawań zmiennopozycyjnych). Układ równań z macierzą M może być rozwiązany metodą Choleskiego.

Większą dokładność rozwiązania można osiągnąć, korzystając z rozkładu ortogonalno-trójkątnego macierzy A. Dla ustalonej macierzy $A \in \mathbb{R}^{m,n}$ istnieje

macierz ortogonalna $Q \in \mathbb{R}^{m,m}$ i macierz $R \in \mathbb{R}^{m,n}$, której współczynniki poniżej diagonali są zerowe, przy czym jeśli macierz A ma liniowo niezależne kolumny, to macierz R również (zatem ma niezerowe współczynniki diagonalne). Dla $m \geqslant n$ pierwsze n kolumn macierzy Q i wierszy macierzy R są określone jednoznacznie z dokładnością do zwrotów.

Macierze Q i R podzielimy na bloki, odpowiednio

$$Q = [Q_1, Q_2], \qquad R = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix},$$

takie że $Q_1 \in \mathbb{R}^{m,n}$ i $R_1 \in \mathbb{R}^{n,n}$. Ponieważ blok R_2 jest zerowy, mamy $A = Q_1 R_1$. Podstawmy to do układu równań normalnych:

$$R_1^T Q_1^T Q_1 R_1 x = R_1^T Q_1^T b.$$

Macierz $Q_1^T Q_1$ jest macierzą jednostkową $n \times n$, a ponieważ macierz R_1 jest nieosobliwa, mamy układ równoważny układowi równań normalnych:

$$R_1 x = Q_1^T b,$$

z nieosobliwą macierzą trójkątną górną R_1.

Jeśli dany układ równań, $Ax = b$, dla którego stawiamy LZNK, pomnożymy stronami przez Q^T, to możemy w nim wyróżnić dwa podukłady:

$$\begin{cases} R_1 x = Q_1^T b, \\ 0x = Q_2^T b. \end{cases}$$

Układ dany jest niesprzeczny wtedy i tylko wtedy, gdy wektor $Q_2^T b = 0$. Co więcej, ponieważ pierwszy podukład ma rozwiązanie jednoznaczne (jest nim rozwiązanie LZNK), a macierz drugiego podukładu jest zerowa, długość wektora $Q_2^T b$ jest najmniejszą osiągalną normą residuum, $b - Ax$.

Istnieje wiele metod rozkładania macierzy A na czynniki Q i R albo Q_1 i R_1. Jedną z nich jest zastosowanie odbić Householdera. Za pomocą n odbić, konstruowanych tak samo, jak w zastosowaniu do układu równań liniowych z nieosobliwą macierzą $n \times n$, macierz A przekształcamy na macierz R. Macierz ortogonalną Q reprezentujemy za pomocą wektorów normalnych hiperpłaszczyzn kolejnych odbić (które możemy przechowywać w tablicy początkowo zawierającej współczynniki macierzy A); mamy

$$Q^T = H_n H_{n-1} \dots H_1, \quad Q = H_1 \dots H_{n-1} H_n,$$

gdzie $H_i = I - \gamma_i v v^T$. Macierzy Q *nie wyznaczamy* w postaci jawnej.

Algorytm rozwiązywania RLZNK za pomocą odbić składa się z następujących kroków:

1. Znadź rozkład macierzy A, tj. reprezentację macierzy Q w postaci wektorów odbić i macierz R.
2. Oblicz wektor $y = Q^T b = H_n \dots H_1 b$.
3. Wybierz pierwsze n wierszy macierzy R i wektora y, tj. macierz $R_1 = Q_1^T A$ i wektor $y_1 = Q_1^T b$, i rozwiąż układ $R_1 x = y_1$.

Rozkładu macierzy A na czynniki Q_1 i R_1 możemy dokonać za pomocą ortonormalizacji Grama-Schmidta; są różne numeryczne implementacje tej procedury. W tak zwanym algorytmie modyfikowanym (MGS), dającym (w implementacji używającej arytmetyki zmiennopozycyjnej) dokładniejszy wynik niż algorytm klasyczny z podręcznika algebry liniowej, konstruujemy macierze $A^{(0)} = A, \dots, A^{(n)} = Q_1$. Jeśli kolumny macierzy $A^{(k)}$ oznaczymy $a_1^{(k)}, \dots, a_n^{(k)}$, to obliczamy

for ($k = 1$; $k \leqslant n$; k++) {
  $r_{kk} = \sqrt{a_k^{(k-1)T} a_k^{(k-1)}}$;
  $a_k^{(k)} = \frac{1}{r_{kk}} a_k^{(k-1)}$;
  for ($i = k+1$; $i \leqslant n$; i++) {
    $r_{ki} = a_k^{(k)T} a_i^{(k-1)}$;
    $a_i^{(k)} = a_i^{(k-1)} - r_{ki} a_k^{(k)}$;
  }
}

Wynikiem obliczenia są kolumny $q_i = a_i^{(n)}$ macierzy Q_1 i współczynniki r_{ki} na i powyżej diagonali macierzy R_1.

Do rozwiązania RLZNK za pomocą ortonormalizacji służy następujący algorytm:

1. Za pomocą ortonormalizacji Grama-Schmidta znajdź macierze Q_1 i R_1.
2. Oblicz wektor $y_1 = Q_1^T b$.
3. Rozwiąż układ równań $R_1 x = y_1$.

Przyczyna, dla której algorytmy korzystające z rozkładu ortogonalno-trójkątnego dają dokładniejsze wyniki niż algorytm równań normalnych jest taka, że wyjściowe zadanie jest zwykle znacznie lepiej uwarunkowane niż układ równań normalnych. Dlatego błędy zaokrągleń popełnione podczas obliczania macierzy M i jej rozkładania na czynniki trójkątne przenoszą się na wynik ze znacznie większym czynnikiem. Tymczasem uwarunkowanie układu równań $R_1 x = Q_1^T b$ (w normie drugiej) jest takie samo, jak uwarunkowanie zadania wyjściowego.

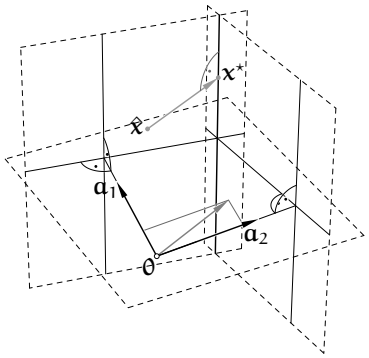
Dualne LZNK

Inny rodzaj liniowego zadania najmniejszych kwadratów możemy postawić, gdy dany układ współrzędnych, $A\mathbf{x} = \mathbf{b}$, jest niesprzeczny i nieokreślony.
W dualnym liniowym zadaniu najmniejszych kwadratów (DLZNK) celem jest wybranie jednego elementu ze zbioru rozwiązań układu $A\mathbf{x} = \mathbf{b}$. Należy wybrać rozwiązanie $\mathbf{x}^\star$ najkrótsze (o najmniejszej normie drugiej), lub takie, aby dla ustalonego wektora $\hat{\mathbf{x}} \in \mathbb{R}^n$ wektor $\mathbf{x}^\star - \hat{\mathbf{x}}$ był najkrótszy; pierwsza sytuacja jest szczególnym przypadkiem drugiej.

DLZNK ma rozwiązanie. Niech $A^T = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$, tj. niech wektory $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{R}^n$ będą transponowanymi wierszami macierzy A. Rozwiązaniem DLZNK jest taki wektor $\mathbf{x}^\star$, że $A\mathbf{x}^\star = \mathbf{b}$ i różnica $\mathbf{x}^\star - \hat{\mathbf{x}}$ jest kombinacją liniową wektorów $\mathbf{a}_1, \ldots, \mathbf{a}_m$. Dowiedźmy tego. Zbiór rozwiązań równania liniowego $\mathbf{a}_i^T\mathbf{x} = b_i$ jest warstwą równoległą do podprzestrzeni o wymiarze $n-1$ prostopadłej do wektora $\mathbf{a}_i$. Zbiór rozwiązań całego układu równań $A\mathbf{x} = \mathbf{b}$ jest przecięciem tych warstw, i jest to warstwa przestrzeni $\mathbb{R}^n$ równoległa do podprzestrzeni, do której należą wszystkie wektory prostopadłe do wektorów $\mathbf{a}_1, \ldots, \mathbf{a}_m$. Jeśli zatem wektor $\mathbf{x}$ jest dowolnym rozwiązaniem układu $A\mathbf{x} = \mathbf{b}$, to wektor $\mathbf{x} - \mathbf{x}^\star$ jest prostopadły do wektorów $\mathbf{a}_1, \ldots, \mathbf{a}_m$, a więc także do ich kombinacji liniowej $\mathbf{x}^\star - \hat{\mathbf{x}}$, i z twierdzenia Pitagorasa mamy

$$\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 = \|\mathbf{x} - \mathbf{x}^\star\|_2^2 + \|\mathbf{x}^\star - \hat{\mathbf{x}}\|_2^2 \geqslant \|\mathbf{x}^\star - \hat{\mathbf{x}}\|_2^2.$$

Jeśli $\mathbf{x} \neq \mathbf{x}^\star$, to nierówność jest ostra. □



Ilustrację DLZNK dla układu dwóch równań z trzema niewiadomymi mamy na rysunku. Zbiór rozwiązań układu jest prostą prostopadłą do płaszczyzny rozpiętej przez wektory $\mathbf{a}_1$, $\mathbf{a}_2$, tj. przecięciem dwóch płaszczyzn prostopadłych do tych wektorów. Wektor $\mathbf{x}^\star - \hat{\mathbf{x}}$ jest prostopadły do tej prostej.

Jeśli więc wektor $\mathbf{x}$ jest rozwiązaniem DLZNK, to wektor $\mathbf{x} - \hat{\mathbf{x}}$ musi być kombinacją liniową transponowanych wierszy macierzy A, a zatem istnieje wektor $\mathbf{y} \in \mathbb{R}^m$, taki że $A^T\mathbf{y} = \mathbf{x} - \hat{\mathbf{x}}$. Jeśli tę równość pomnożymy przez macierz A, to otrzymamy

$$AA^T\mathbf{y} = A\mathbf{x} - A\hat{\mathbf{x}}.$$

Po podstawieniu $A\mathbf{x} = \mathbf{b}$ mamy stąd układ równań z niewiadomym wektorem $\mathbf{y}$

$$AA^T\mathbf{y} = \mathbf{b} - A\hat{\mathbf{x}},$$

zwany dualnym układem równań normalnych. Macierz AA^T jest symetryczna i jeśli wiersze macierzy A są liniowo niezależne, to jest dodatnio określona. Aby tak było, musi być $n \geqslant m$. Jeśli wiersze macierzy A są liniowo zależne, to nie mamy gwarancji, że układ równań $A\mathbf{x} = \mathbf{b}$ jest niesprzeczny, i mamy do czynienia z zadaniem nieregularnym.

Algorytm dualnych równań normalnych polega na obliczaniu macierzy $M = AA^T$ i wektora $\mathbf{d} = \mathbf{b} - A\hat{\mathbf{x}}$, a następnie rozwiązaniu układu $M\mathbf{y} = \mathbf{d}$ (do czego można użyć metody Choleskiego) i obliczeniu rozwiązania $\mathbf{x} = \hat{\mathbf{x}} + A^T\mathbf{y}$. Jeśli ma być znalezione rozwiązanie o najmniejszej normie drugiej, to $\hat{\mathbf{x}} = \mathbf{0}$; można wtedy pominąć niektóre obliczenia.

Większą dokładność można uzyskać, korzystając z rozkładu trójkątno-ortogonalnego macierzy A. Istnieje macierz $L \in \mathbb{R}^{m,n}$, która ma zera za współczynnikiem diagonalnym w każdym wierszu, i macierz ortogonalna $Q \in \mathbb{R}^{n,n}$, takie że $A = LQ^T$; macierze te można otrzymać, stosując do macierzy A^T (kolumnowo regularnej) te same algorytmy rozkładu ortogonalno-trójkątnego, których użycie do rozwiązania RLZNK było opisane wcześniej. Otrzymujemy macierze $L = [L_1, L_2]$ i $Q = [Q_1, Q_2]$, w których blok $L_1 \in \mathbb{R}^{m,m}$ jest nieosobliwą macierzą trójkątną dolną, blok L_2 jest zerowy, i macierze L_1 i Q_1 są dane jednoznacznie z dokładnością do zwrotów kolumn. Zachodzi równość $A = L_1Q_1^T$.

Po podstawieniu czynników rozkładu do dualnego układu równań normalnych mamy

$$L_1Q_1^TQ_1L_1^T\mathbf{y} = \mathbf{b} - L_1Q_1^T\hat{\mathbf{x}},$$

a ponieważ $Q_1^TQ_1 = I$ i macierz L_1 jest nieosobliwa, mamy układ równoważny

$$L_1^T\mathbf{y} = L_1^{-1}\mathbf{b} - Q_1^T\hat{\mathbf{x}}.$$

Rozwiązując powyższy układ równań, można by obliczyć wektor $\mathbf{y}$, a następnie obliczyć $\mathbf{x} = \hat{\mathbf{x}} + A^T\mathbf{y}$, ale ponieważ poza tym wektor $\mathbf{y}$ *nie jest do niczego potrzebny*, lepszym rozwiązaniem po znalezieniu czynników rozkładu macierzy A

jest użycie *tylko* tych czynników. Oznaczając $\boldsymbol{w} = L_1^{-1}\mathbf{b} - Q_1^T\hat{\boldsymbol{x}}$ i podstawiając $\boldsymbol{y} = L_1^{-T}\boldsymbol{w}$, otrzymamy $A^T\boldsymbol{y} = Q_1L_1^TL_1^{-T}\boldsymbol{w} = Q_1\boldsymbol{w}$. Stąd otrzymujemy algorytm rozwiązywania DLZNK:

1. Za pomocą ortonormalizacji Grama-Schmidta znajdź macierze trójkątną dolną L_1 i kolumnowo-ortogonalną Q_1, takie że $A = L_1Q_1^T$.
2. Rozwiąż układ równań liniowych $L_1\boldsymbol{z} = \mathbf{b}$ i oblicz wektor $\boldsymbol{w} = \boldsymbol{z} - Q_1^T\hat{\boldsymbol{x}}$.
3. Oblicz $\boldsymbol{x} = \hat{\boldsymbol{x}} + Q_1\boldsymbol{w}$.

Powyższy algorytm można zrealizować również za pomocą odbić Householdera, bez jawnego wyznaczania macierzy Q_1. Inny algorytm rozwiązywania DLZNK korzystający z odbić możemy otrzymać w taki sposób: Niech $\mathbf{s} = Q^T\boldsymbol{x}$ i $\hat{\mathbf{s}} = Q^T\hat{\boldsymbol{x}}$. Podstawiając nowe wyrażenie do układu $LQ^T\boldsymbol{x} = \mathbf{b}$, otrzymujemy układ równań $L\mathbf{s} = \mathbf{b}$, który możemy przedstawić w postaci $L_1\mathbf{s}_1 + L_2\mathbf{s}_2 = \mathbf{b}$. Ponieważ blok L_2 jest zerowy, wektor $\mathbf{s}_1$ musi być rozwiązaniem układu równań $L_1\mathbf{s}_1 = \mathbf{b}$, zaś wektor $\mathbf{s}_2$ trzeba zatem wybrać tak, aby wektor $\boldsymbol{x} - \hat{\boldsymbol{x}} = Q(\mathbf{s} - \hat{\mathbf{s}})$ miał najmniejszą normę drugą. Ale jest ona równa normie drugiej wektora $\mathbf{s} - \hat{\mathbf{s}}$. Zatem, jeśli wektor $\hat{\mathbf{s}}$ podzielimy (w tym samym miejscu co $\mathbf{s}$) na bloki $\hat{\mathbf{s}}_1 = Q_1^T\hat{\boldsymbol{x}}$ i $\hat{\mathbf{s}}_2 = Q_2^T\hat{\boldsymbol{x}}$, to aby zminimalizować normę drugą wektora $\mathbf{s} - \hat{\mathbf{s}}$, musimy przyjąć $\mathbf{s}_2 = \hat{\mathbf{s}}_2$. Mamy stąd taki algorytm:

1. Znajdź macierz trójkątną dolną L i wektory odbić reprezentujące macierz Q, takie że $A = LQ^T$. Wybierz blok L_1 macierzy L.
2. Oblicz $\hat{\mathbf{s}} = Q^T\hat{\boldsymbol{x}}$, stosując odpowiednie odbicia.
3. Rozwiąż układ $L_1\mathbf{s}_1 = \mathbf{b}$ i złóż wektor $\mathbf{s}$ z bloków $\mathbf{s}_1$ i $\mathbf{s}_2 = \hat{\mathbf{s}}_2$.
4. Oblicz $\boldsymbol{x} = Q\mathbf{s}$, stosując odpowiednie odbicia.

Nieregularne LZNK

Jeśli rząd r macierzy A jest mniejszy zarówno od liczby kolumn n, jak i od liczby wierszy m, to liniowe zadanie najmniejszych kwadratów dla układu $A\boldsymbol{x} = \mathbf{b}$ jest nieregularne. Zbiór rozwiązań takiego zadania jest nieskończony; jest on warstwą $n - r$-wymiarową (przestrzeni $\mathbb{R}^n$), której elementami są takie wektory $\boldsymbol{x}$, że wektor $\boldsymbol{y}^\star = A\boldsymbol{x}$ jest rzutem prostopadłym wektora $\mathbf{b}$ na podprzestrzeń rozpiętą przez kolumny macierzy A (tj. residuum, $\mathbf{b} - \boldsymbol{y}^\star$, jest wektorem prostopadłym do tej podprzestrzeni). Dokładnie jeden element tej warstwy ma najmniejszą normę drugą; co więcej, dla dowolnego wektora $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ istnieje dokładnie jeden element $\boldsymbol{x}^\star$ tej warstwy, taki że norma druga różnicy $\boldsymbol{x}^\star - \hat{\boldsymbol{x}}$ jest najmniejsza. Rozwiązanie NLZNK zwykle polega na znalezieniu tego wektora $\boldsymbol{x}^\star$. Rozumowanie podobne do przeprowadzonego wcześniej dla DLZNK uzasadnia stwierdzenie, że wektor $\boldsymbol{x}^\star - \hat{\boldsymbol{x}}$ jest kombinacją liniową transponowanych wierszy macierzy A.

NLZNK *są trudne* do numerycznego rozwiązania. Jest tak dlatego, że rozwiązanie zadania zależy od danych w sposób *paskudnie nieciągły*. NLZNK jest szczególnie trudne, jeśli nie znamy rzędu macierzy A i dopiero mamy go na podstawie obliczeń numerycznych ustalić.

Najodporniejsze numeryczne algorytmy rozwiązywania NLZNK korzystają z rozkładu względem wartości szczególnych (ang. *singular value decomposition*, *SVD*) macierzy A. Dowodzi się, że dla dowolnej macierzy $A \in \mathbb{R}^{m,n}$ istnieją macierze ortogonalne $U \in \mathbb{R}^{m,m}$ i $V \in \mathbb{R}^{n,n}$ oraz macierz diagonalna $\Sigma \in \mathbb{R}^{m,n}$, takie że $A = U\Sigma V^T$. Współczynniki diagonalne macierzy Σ, $\sigma_1, \ldots, \sigma_l$, gdzie $l = \min\{m, n\}$, nazywają się wartościami szczególnymi macierzy A i są nieujemne. Rozkład w ogólności *nie jest* jednoznaczny, ale same wartości szczególne i liczby ich wystąpień (krotności) są określone przez macierz A jednoznacznie. Zwykle rozkładu dokonuje się w taki sposób, aby wartości szczególne były uporządkowane nierosnąco na diagonali macierzy Σ: $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r > \sigma_{r+1} = \cdots = \sigma_l = 0$. Liczba niezerowych wartości szczególnych jest rzędem macierzy A.

Wyznaczanie rozkładu SVD wiąże się z rozwiązywaniem algebraicznego zagadnienia własnego i dla macierzy o więcej niż czterech wierszach i kolumnach może być dokonane tylko jakąś metodą iteracyjną. Opis algorytmu Goluba, który dokonuje rozkładu (otrzymując reprezentacje macierzy U i V w postaci ciągu wektorów odbić Householdera i tzw. obrotów Givensa), pominiemy (kto będzie potrzebował, ten go znajdzie). Natomiast przyjrzyjmy się zastosowaniu tego rozkładu do rozwiązania zadania.

NLZNK dla układu równań $A\boldsymbol{x} = \mathbf{b}$ i wektora $\hat{\boldsymbol{x}}$ można zastąpić zadaniem równoważnym dla układu równań $\Sigma\boldsymbol{y} = \mathbf{d}$, gdzie $\mathbf{d} = U^T\mathbf{b}$, i wektora $\hat{\boldsymbol{y}} = V^T\hat{\boldsymbol{x}}$ (po rozwiązaniu tego zadania możemy obliczyć $\boldsymbol{x} = V\boldsymbol{y}$). Załóżmy dla uproszczenia, że $\hat{\boldsymbol{x}} = 0$, czyli $\hat{\boldsymbol{y}} = 0$. Wtedy rozwiązaniem NLZNK dla układu $\Sigma\boldsymbol{y} = \mathbf{d}$ jest wektor o współrzędnych

$$y_i = \begin{cases} d_i/\sigma_i & \text{dla } i \leqslant r, \\ 0 & \text{dla } i > r. \end{cases}$$

Mamy stąd wyjaśnienie trudności zadania: niewielkie zaburzenie macierzy A może spowodować pewne niegroźne zmiany macierzy U i V, oraz zaburzenie macierzy Σ: jeśli dowolna zerowa wartość szczególna zmieni się na niezerową (czyli skutkiem zaburzenia będzie zwiększenie rzędu macierzy A) i $d_i \neq 0$, to trzeba będzie przyjąć $y_i = d_i/\sigma_i$, zamiast zera, dla pewnego $i > r$. Tak więc, *im mniej* zaburzymy macierz A (w sposób zmieniający σ_i), *tym większa* będzie zmiana wyniku.

Jeśli znamy rząd r macierzy A, to po znalezieniu rozkładu SVD możemy zamienić na zera obliczone numerycznie wartości szczególne σ_i dla $i > r$ — obliczone wartości niezerowe są skutkiem błędów zaokrągleń i aproksymacji popełnionych podczas rozkładania. Jeśli rzędu nie znamy, to możemy przyjąć pewien próg (zależny od oszacowania błędów) i zamienić na zera znalezione wartości szczególne mniejsze od tego progu; to postępowanie nazywa się regularyzacją dyskretną. Inne podejście to tzw. regularyzacja ciągła — do *wszystkich* wartości szczególnych dodajemy pewną liczbę $s > 0$, otrzymując zadanie z macierzą pełnego rzędu, tj. RLZNK, jeśli $m > n$, układ równań z macierzą kwadratową nieosobliwą, jeśli $n = m$, albo DLZNK, jeśli $m < n$. Wybór metody regularyzacji zależy od zastosowania.

Zadania i problemy

1. Niech
$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}.$$
Jaki będzie skutek użycia arytmetyki pojedynczej precyzji do rozwiązania LZNK dla układu $A\boldsymbol{x} = \mathbf{b}$ za pomocą algorytmu równań normalnych, jeśli $|\varepsilon| < 10^{-4}$?
2. Tabela zawiera wyniki pomiarów (z błędami) wartości pewnej funkcji:

x	-2	-1	1	2
$f(x)$	-1	5	2	14

Przy założeniu, że funkcja ma postać $f(x) = a_0 + a_1(x^2 - 4)$, znajdź liczby a_0 i a_1 najlepiej pasujące do wyników tych pomiarów. Postaw i rozwiąż w tym celu LZNK, używając algorytmu równań normalnych oraz odbić Householdera.
3. Pseudoodwrotność macierzy $A \in \mathbb{R}^{m,n}$ jest to macierz $A^+ \in \mathbb{R}^{n,m}$, taka że macierz AA^+ jest macierzą rzutu ortogonalnego przestrzeni $\mathbb{R}^m$ na podprzestrzeń rozpiętą przez kolumny macierzy A, zaś macierz A^+A jest macierzą rzutu ortogonalnego na podprzestrzeń rozpiętą przez wiersze macierzy A.
Udowodnij, że

a) jeśli macierz A jest kwadratowa nieosobliwa, to $A^+ = A^{-1}$,
b) jeśli macierz A jest kolumnowo regularna, to $A^+ = (A^TA)^{-1}A^T$,
c) jeśli macierz A jest wierszowo regularna, to $A^+ = A^T(AA^T)^{-1}$.

Zbadaj związek pseudoodwrotności z liniowymi zadaniami najmniejszych kwadratów.
Wskazówka: znajdź wzory opisujące rozwiązania odpowiednich układów równań normalnych.
4. LZNK z więzami. Dany układ równań liniowych jest podzielony na dwa podukłady, z macierzami $B \in \mathbb{R}^{m,n}$ i $C \in \mathbb{R}^{k,n}$ (takimi, że $k < n < m + k$):
$$\begin{cases} B\boldsymbol{x} = \mathbf{d}, \\ C\boldsymbol{x} = \boldsymbol{e}, \end{cases}$$
przy czym macierz $A = \left[\begin{smallmatrix} B \\ C \end{smallmatrix}\right]$ jest kolumnowo regularna, a macierz C jest wierszowo regularna. Zadanie polega na znalezieniu wektora $\boldsymbol{x}^\star$, takiego że $C\boldsymbol{x}^\star = \boldsymbol{e}$ oraz norma druga wektora residuum pierwszego podukładu jest najmniejsza.

Algorytm zamiany zmiennych: Znajdujemy rozkład macierzy C na czynniki L i Q^T, odpowiednio trójkątny dolny i ortogonalny. Dokonujemy zamiany zmiennych, wprowadzając nowy wektor niewiadomy, $\mathbf{y} = Q^T\mathbf{x}$. Macierz L dzielimy na bloki, $L = [L_1, L_2]$, z których pierwszy jest nieosobliwą macierzą trójkątną dolną, a drugi jest macierzą zerową. Macierz Q dzielimy na bloki Q_1 i Q_2. W ten sposób drugi podukład zastępujemy układem równoważnym $L_1\mathbf{y}_1 = \boldsymbol{e}$, z którego wyznaczymy $\mathbf{y}_1$. Po zamianie zmiennych w pierwszym podukładzie otrzymujemy układ równań liniowych

$$BQ\mathbf{y} = \mathbf{d}, \quad \text{czyli} \quad BQ_1\mathbf{y}_1 + BQ_2\mathbf{y}_2 = \mathbf{d}, \quad \text{czyli} \quad BQ_2\mathbf{y}_2 = \mathbf{d} - BQ_1\mathbf{y}_1.$$

w ogólności sprzeczny. Dla tego układu rozwiązujemy regularne liniowe zadanie najmniejszych kwadratów, po czym na podstawie otrzymanych wektorów $\mathbf{y}_1$ i $\mathbf{y}_2$ możemy obliczyć $\mathbf{x} = Q\mathbf{y}$.

Algorytm z mnożnikami Lagrange'a: Jeśli macierz B jest kolumnowo regularna, to symetryczna macierz $M = B^TB$ jest dodatnio określona. Wtedy kwadrat normy drugiej residuum pierwszego podukładu jest wielomianem drugiego stopnia współrzędnych wektora $\mathbf{x}$, który w każdej warstwie przestrzeni $\mathbb{R}^n$ ma jednoznacznie określone minimum:

$$f(\mathbf{x}) = \|\mathbf{d} - B\mathbf{x}\|_2^2 = \mathbf{x}^TM\mathbf{x} - 2\mathbf{x}^TB^T\mathbf{d} + \mathbf{d}^T\mathbf{d}.$$

Aby znaleźć minimum funkcji f w zbiorze rozwiązań drugiego podukładu, wystarczy rozwiązać układ równań

$$\begin{bmatrix} M & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} B^T\mathbf{d} \\ \boldsymbol{e} \end{bmatrix}.$$

Po rozwiązaniu tego układu blok $\mathbf{y}$ odrzucamy; jego współrzędne są tzw. mnożnikami Lagrange'a dla postawionego tu zadania minimalizacji z więzami. Podany wyżej blokowy układ równań możemy rozwiązać w podobny sposób, jak układ w zadaniu 7 na s. 4.21, ale zamiast obliczenia macierzy M i zastosowania do niej metody Choleskiego, lepiej jest dokonać rozkładu ortogonalno-trójkątnego macierzy B.

Opracuj szczegóły obu algorytmów (w szczególności określ wymiary i sposoby reprezentowania wszystkich macierzy otrzymanych w obliczeniach).

Uzasadnij stwierdzenie, że macierz BQ_2 przetwarzana w pierwszym algorytmie jest kolumnowo-regularna.

Udowodnij, że drugi algorytm daje rozwiązanie zadania.

5. Opisz, jak zrealizować pierwszy podany na wykładzie algorytm rozwiązywania DLZNK korzystający z rozkładu trójkątno-ortogonalnego, za pomocą odbić Householdera (bez jawnego wyznaczania macierzy Q_1).

6. Metoda Newtona z pseudoodwrotnością służy do numerycznego rozwiązywania układów równań nieliniowych, w których liczba równań, m, jest mniejsza niż liczba niewiadomych, n. Układ równań liniowych $J_k\boldsymbol{\delta} = -\mathbf{f}_k$ jest rozwiązywany jako DLZNK, w celu wyznaczenia najkrótszego spełniającego ten układ wektora $\boldsymbol{\delta}$, po czym przyjmuje się $\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}$. W ten sposób, jeśli funkcja $\mathbf{f}$ spełnia odpowiednie warunki, powstaje ciąg $(\mathbf{x}_k)_{k\in\mathbb{N}}$ zbieżny do pewnego rozwiązania $\boldsymbol{\alpha}$ (z nieskończonego zbioru rozwiązań), położonego w pobliżu punktu startowego $\mathbf{x}_0$.

   Wykonaj dwa kroki metody Newtona z pseudoodwrotnością dla układu równań

   $$\begin{cases} x^2 + y^2 - z^2 = 3 \\ x + y + z = 1 \end{cases}$$

   przyjmując $\mathbf{x}_0 = [0, 0, 0]^T$. W rachunkach „ręcznych" układaj i rozwiązuj dualny układ równań normalnych.

7. Sprawdź, że w metodzie sprzężonych gradientów obliczenie

   $$\mathbf{r}_{k+1} = \mathbf{r}_k - t_kA\boldsymbol{\nu}_k$$

   jest krokiem ortogonalizacji Grama-Schmidta w sensie iloczynu skalarnego $\langle\mathbf{u}, \boldsymbol{\nu}\rangle = \boldsymbol{\nu}^T\mathbf{u}$, zaś konstrukcja

   $$\boldsymbol{\nu}_{k+1} = \mathbf{r}_{k+1} + s_k\boldsymbol{\nu}_k$$

   jest krokiem ortogonalizacji Grama-Schmidta w sensie iloczynu skalarnego $\langle\mathbf{u}, \boldsymbol{\nu}\rangle_A = \boldsymbol{\nu}^TA\mathbf{u}$.

Projekt

Napisz procedurę rozwiązywania LZNK z więzami (zad. 4 na s. 5.10–11), realizującą jedną (dowolną) z metod rozważanych w zadaniu. Wbuduj tę procedurę w program umożliwiający przeczytanie danych z pliku. Wskazane jest maksymalne wykorzystanie w implementacji algorytmu procedur z pakietu LAPACK (ten projekt jest ćwiczeniem z umiejętności opracowania implementacji na podstawie dokumentacji pakietu). Możesz skorzystać ze stron pod adresami

`http://www.netlib.org/lapack/`
`http://www.netlib.org.lapack/lug`

i ze stron manuala. Nazwy procedur piszemy małymi literami, np. `man dgesv`, tylko litery, choć w programie w C nazwy te mają dodane na końcu podkreślenie, np. `dgesv_`, zaś w Fortranie pisze się je wielkimi literami (bez podkreślenia). Procedury, które mogą się przydać w tym projekcie, to m.in. `dgetrf`, `dgetrs`, `dgelqf`, `dgeqrf`, `dgels`, `dtbtrs`, a także procedury BLAS: `daxpy`, `ddot`, `dgemm`, `dgemv`, `dspmv`, `dsymm`, `dtrmm`, `dtrmv`, `dtrsm`, `dtrsv`.

Algebraiczne zagadnienie własne

Niech $A \in \mathbb{R}^{n,n}$. Jeśli wektor $\mathbf{x} \neq \mathbf{0}$ spełnia równanie $A\mathbf{x} = \lambda\mathbf{x}$ dla pewnej liczby λ, to mówimy, że jest to wektor własny macierzy A, zaś liczba λ jest to wartość własna tej macierzy; parę $(\mathbf{x}, \lambda)$ nazywamy parą własną macierzy A.

Algebraiczne zagadnienie własne polega na znalezieniu, dla danej macierzy A, jej (wszystkich, kilku lub jednej) wartości własnych albo par własnych. Algebraiczne zagadnienia własne występują w różnych zastosowaniach, np. w mechanice, mają też związek z innymi zadaniami numerycznej algebry liniowej, np. rozwiązywaniem układów równań lub liniowych zadań najmniejszych kwadratów.

Równanie $A\mathbf{x} = \lambda\mathbf{x}$ można przepisać w postaci $(A - \lambda I)\mathbf{x} = \mathbf{0}$. Z tej postaci natychmiast wynika, że para $(\mathbf{x}, \lambda)$, w której wektor $\mathbf{x} \neq \mathbf{0}$, może spełniać to równanie (czyli jest parą własną) wtedy i tylko wtedy, gdy macierz $A - \lambda I$ jest osobliwa. To oznacza, że jej wyznacznik jest zerowy. Wyrażenie $\det(A - \lambda I)$ jest wielomianem stopnia n zmiennej λ. Na podstawie zasadniczego twierdzenia algebry (Gauss, 1799 r.), równanie charakterystyczne $\det(A - \lambda I) = 0$ ma rozwiązanie, które jest liczbą rzeczywistą albo zespoloną. Tak więc każda macierz ma jakąś wartość własną. Zbiór (w ogólności zespolonych) wartości własnych dowolnej macierzy A nazywa się widmem tej macierzy; oznaczamy je symbolem $\operatorname{spect} A$.

Dla ustalonego λ układ równań $(A - \lambda I)\mathbf{x} = \mathbf{0}$ jest jednorodny; jeśli zatem $\lambda \in \mathbb{R}$ jest wartością własną macierzy A, to zbiór rozwiązań jest podprzestrzenią liniową przestrzeni $\mathbb{R}^n$. Jest to tzw. podprzestrzeń własna macierzy A przynależna do wartości własnej λ. Wymiar tej podprzestrzeni jest nazywany krotnością geometryczną wartości własnej λ. Z kolei, wielomian charakterystyczny można przedstawić w postaci

$$\det(A - \lambda I) = (\lambda_1 - \lambda) \cdot \ldots \cdot (\lambda_n - \lambda).$$

Liczby $\lambda_1, \ldots, \lambda_n$ to wartości własne, które mogą się powtarzać. Liczba wystąpień wartości własnej λ_i w tym rozkładzie jest zwana jej krotnością algebraiczną. Krotność algebraiczna dowolnej wartości własnej jest większa lub równa krotności geometrycznej tej wartości własnej.

O macierzach A i B, dla których istnieje nieosobliwa macierz C, taka że $B = C^{-1}AC$ mówimy, że to są macierze podobne. Podobieństwo macierzy jest oczywiście relacją równoważności. Można udowodnić, że jeśli macierze są podobne, to mają identyczne wartości własne, o identycznych krotnościach algebraicznych i geometrycznych.

Wektory własne przynależne do różnych wartości własnych dowolnej danej macierzy są liniowo niezależne. Jeśli krotność algebraiczna każdej wartości własnej macierzy A jest równa krotności geometrycznej, to suma baz wszystkich podprzestrzeni własnych składa się z n niezależnych liniowo wektorów własnych macierzy A. Ustawmy te wektory w macierz $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$; macierz ta jest nieosobliwa. Wtedy

$$AX = [A\mathbf{x}_1, \ldots, A\mathbf{x}_n] = [\lambda_1\mathbf{x}_1, \ldots, \lambda_n\mathbf{x}_n] = X\Lambda,$$

gdzie macierz Λ jest diagonalna; jej współczynniki diagonalne są wartościami własnymi macierzy A. Możemy napisać równości

$$X^{-1}AX = \Lambda \quad \text{i} \quad X\Lambda X^{-1} = A.$$

Taka macierz A jest zatem podobna do macierzy diagonalnej, mówimy też, że jest diagonalizowalna. Macierz nie jest diagonalizowalna, jeśli co najmniej jedna jej wartość własna ma krotność algebraiczną różną (większą) od geometrycznej.

Przykłady: Macierz

$$\begin{bmatrix} 3 & 4 \\ 4 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 7 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$$

jest diagonalizowalna. Macierz

$$\begin{bmatrix} 3 & -4 \\ 4 & 3 \end{bmatrix}$$

też jest diagonalizowalna, ale jej wartości własne są liczbami zespolonymi, $\lambda_1 = (3, -4)$, $\lambda_2 = (3, 4)$, zatem wektory własne — kolumny odpowiedniej macierzy X — mają co najmniej jedną współrzędną zespoloną. Natomiast macierz

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

nie jest diagonalizowalna; krotność algebraiczna wartości własnej 1 jest równa 2, a krotność geometryczna jest równa 1.

Niech $w(x) = a_k x^k + \cdots + a_1 x + a_0$ będzie dowolnym wielomianem. Możemy użyć macierzy A jako argumentu, tj. napisać

$$w(A) = a_k A^k + \cdots + a_1 A + a_0 I.$$

Łatwo jest sprawdzić, że jeśli liczba λ jest wartością własną macierzy A, to liczba $w(\lambda)$ jest wartością własną macierzy $w(A)$. To samo stwierdzenie dotyczy

funkcji wymiernych, tj. ilorazów wielomianów. Dzielenie licznika przez mianownik (podczas obliczania skalarnej wartości funkcji dla danego x) zastępujemy przez mnożenie macierzy — licznika — przez odwrotność macierzy — mianownika.

Jeśli macierz $A \in \mathbb{R}^{n,n}$ jest symetryczna, to jest diagonalizowalna, co więcej, jej wszystkie wartości własne są liczbami rzeczywistymi i istnieje baza ortonormalna przestrzeni $\mathbb{R}^n$ złożona z wektorów własnych tej macierzy. Zatem, istnieje macierz ortogonalna X, taka że $X^{-1}AX = X^TAX = \Lambda$ jest macierzą diagonalną. W wielu zastosowaniach pojawia się potrzeba rozwiązania algebraicznego zagadnienia własnego z macierzą symetryczną — jest to przypadek prostszy do numerycznego rozwiązywania niż przypadek ogólny i głównie na nim się dalej skupimy.

Kilka uwag na temat uwarunkowania zadania: jeśli macierz A jest diagonalizowalna, to na podstawie twierdzenia Bauera-Fikego mamy następujące oszacowanie: niech δA oznacza zaburzenie macierzy A. Niech μ oznacza (dowolną) wartość własną macierzy $A + \delta A$ i niech λ_i oznacza wartość własną macierzy A, taką że różnica $\mu - \lambda_i$ ma najmniejszą wartość bezwzględną. Zachodzi nierówność

$$|\mu - \lambda_i| \leqslant \operatorname{cond} X \cdot \|\delta A\|,$$

gdzie X oznacza macierz, której kolumny są wektorami własnymi macierzy A (norma indukowana może tu być dowolna). Jeśli macierz A jest symetryczna, to istnieje odpowiednia macierz ortogonalna X, i wtedy $\operatorname{cond}_2 X = 1$. Dla macierzy diagonalizowalnej niesymetrycznej żadna macierz X zbudowana z wektorów własnych nie jest ortogonalna i dlatego $\operatorname{cond}_2 X > 1$. Jeśli macierz A nie jest diagonalizowalna, to zmiany wartości własnych zależą od powodujących je zaburzeń macierzy A w sposób ciągły, ale nie lipschitzowski. Numeryczne obliczanie wartości własnych takich macierzy jest kłopotliwe.

Jeśli macierze A i $A + \delta A$ są symetryczne i symbolami $\boldsymbol{\lambda}$ i $\boldsymbol{\mu}$ oznaczymy wektory zbudowane odpowiednio z tak samo (np. malejąco) uporządkowanych wartości własnych tych macierzy, to na podstawie twierdzenia Wielandta-Hoffmana zachodzi nierówność

$$\|\boldsymbol{\mu} - \boldsymbol{\lambda}\|_2 \leqslant \|\delta A\|_F.$$

Zadanie wyznaczania wektora $\boldsymbol{\lambda}$ jest zatem bardzo dobrze uwarunkowane, choć jeśli pewne wartości własne mają bardzo małe wartości bezwzględne, to ich zaburzenia względne spowodowane dodaniem małego zaburzenia δA mogą być duże.

Uwarunkowanie zadania wyznaczania wektorów własnych zależy od odległości między wartościami własnymi, i jest tym gorsze, im mniej odpowiednie wartości

własne się różnią. Zauważmy, że jeśli pewna wartość własna ma krotność geometryczną $k > 1$, to istnieje nieskończenie wiele baz odpowiedniej podprzestrzeni własnej, złożonych z wektorów jednostkowych. Macierz zaburzona może mieć zamiast tej wartości własnej k różnych wartości własnych (jednokrotnych) i dlatego w tym przypadku rozwiązanie zależy od zaburzenia w sposób nieciągły (jest to możliwe nawet, jeśli macierz A jest symetryczna).

Uwaga: Nie jest dobrym pomysłem obliczanie współczynników wielomianu charakterystycznego $\det(A - \lambda I)$, np. w bazie potęgowej, a następnie znajdowanie jego miejsc zerowych. Nawet jeśli zadanie wyjściowe jest dobrze uwarunkowane, zadanie znalezienia miejsc zerowych wielomianu na podstawie jego współczynników jest zwykle *bardzo źle* uwarunkowane.

Metoda potęgowa

Przypuśćmy, że jedna z wartości własnych macierzy A dominuje, tj. jej wartość bezwzględna jest większa niż wartości bezwzględne wszystkich pozostałych wartości własnych, i przypuśćmy, że mamy wyznaczyć parę własną z właśnie tą wartością własną. Założymy, że dominująca wartość własna jest liczbą rzeczywistą (możemy, jeśli macierz A jest symetryczna) i chwilowo przyjmiemy, że jej krotność jest równa 1. Niech będzie to wartość własna λ_1.

Wybieramy niezerowy wektor $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, a następnie dla $k > 0$ określamy wektory $\boldsymbol{x}^{(k)}$, wzorem $\boldsymbol{x}^{(k)} = A\boldsymbol{x}^{(k-1)}$ (czyli $\boldsymbol{x}^{(k)} = A^k\boldsymbol{x}^{(0)}$). Jeśli macierz A jest diagonalizowalna, to istnieją liczby $c_1, \ldots, c_n$, takie że

$$\boldsymbol{x}^{(0)} = \sum_{i=1}^{n} c_i \boldsymbol{x}_i,$$

gdzie $\boldsymbol{x}_i$ to wektory własne macierzy A. Wtedy mamy

$$\boldsymbol{x}^{(k)} = \sum_{i=1}^{n} c_i \lambda_i^k \boldsymbol{x}_i = \lambda_1^k \sum_{i=1}^{n} c_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \boldsymbol{x}_i.$$

Jeśli $|\lambda_i| < |\lambda_1|$, to dla $k \to \infty$ ciąg liczb $(\lambda_i/\lambda_1)^k$ dąży do zera. To oznacza, że jeśli $c_1 \neq 0$, to ciąg kierunków wektorów $\boldsymbol{x}^{(k)}$ dąży do kierunku wektora własnego $\boldsymbol{x}_1$, przynależnego do dominującej wartości własnej. Po wykonaniu dostatecznie wielu iteracji możemy w ten sposób znaleźć wektor bliski wektora własnego $\boldsymbol{x}_1$.

Podane rozumowanie jest podstawą metody potęgowej rozwiązywania algebraicznego zagadnienia własnego, a dokładniej wyznaczania pary własnej $(\boldsymbol{x}_1, \lambda_1)$ z dominującą wartością własną. Jeśli krotność geometryczna tej wartości

własnej jest większa niż 1, to kierunki otrzymanego ciągu wektorów zbiegają do kierunku *pewnego* wektora własnego związanego z dominującą wartością własną. Opisane postępowanie jest jednak niepraktyczne, ponieważ jeśli $|\lambda_1| \neq 1$, to długości wektorów $\mathbf{x}^{(k)}$ maleją do zera lub rosną nieograniczenie. Dlatego należy stosować normalizację, tj. dzielić kolejne otrzymane wektory przez ich długości — pamiętamy, że istotne są tylko kierunki tych wektorów. Mamy stąd algorytm:

1. Przyjmij $z^{(0)} \neq 0$,
2. Dla $k = 1, 2, \ldots$ obliczaj

$$y^{(k)} = Az^{(k-1)}, \qquad z^{(k)} = \frac{1}{\|y^{(k)}\|_2} y^{(k)}.$$

Jeśli pewien wektor z jest wektorem własnym macierzy A, to spełnia równanie $Az = \lambda z$. Możemy je potraktować jak układ n równań z jedną niewiadomą, którą jest wartość własna λ; macierz tego układu jest kolumnowa, jest nią wektor z. Dla takiego układu stawiamy RLZNK. Układ równań normalnych ma postać

$$z^T z\lambda = z^T Az,$$

aby go rozwiązać, obliczamy tzw. iloraz Rayleigha

$$\lambda = \frac{z^T Az}{z^T z}.$$

Jeśli wektor z *nie jest* wektorem własnym, to oczywiście układ $Az = \lambda z$ jest sprzeczny, ale jeśli wektor z jest przybliżeniem wektora własnego x_i, to iloraz Rayleigha jest przybliżeniem wartości własnej λ_i. Ale jeśli $\|z\|_2 = 1$, to mianownik ilorazu Rayleigha jest równy 1. Zatem, po obliczeniu wektora $z^{(k)}$ obliczamy liczbę $\rho_{k-1} = z^{(k)T} y^{(k-1)}$. Podczas gdy ciąg wektorów jednostkowych $z^{(k)}$ zbiega do wektora własnego, ciąg liczb ρ_k zbiega do dominującej wartości własnej λ_1.

Jeśli macierz A jest symetryczna, λ_2 jest drugą co do wartości bezwzględnej wartością własną, i symbolem t_k oznaczymy tangens najmniejszego kąta między wektorem $z^{(k)}$ i wektorem x_1 należącym do podprzestrzeni własnej przynależnej do wartości własnej λ_1, to można udowodnić, że

$$|t_k| \leqslant \left|\frac{\lambda_2}{\lambda_1}\right|^k |t_0|, \quad \text{oraz} \quad |\rho_k - \lambda_1| \leqslant 2\|A\||t_k|^2 = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right).$$

Szybkość zbieżności zależy więc od tego, „jak bardzo dominuje" wartość własna λ_1. Zbieżność nie ma miejsca, jeśli dwie wartości własne dominują, tj. $\lambda_2 = -\lambda_1$. W takim przypadku „prosta" metoda potęgowa nie wystarczy do rozwiązania zadania.

Jeśli liczba c_1 dla przyjętego wektora $z^{(0)}$ jest zerem, to teoretycznie ciąg $(z^{(k)})_{k\in\mathbb{N}}$ zbiega do wektora własnego związanego z którąś z pozostałych wartości własnych. Ale w obliczeniach numerycznych występują błędy zaokrągleń, których skutki w tym przypadku *mogą być dobroczynne*: zaburzenie spowodowane zaokrągleniem zwykle doprowadza do pojawienia się odpowiedniej składowej o kierunku wektora własnego związanego z wartością własną λ_1, po czym kolejne iteracje „wzmacniają" tę składową, jednocześnie „wygaszając" pozostałe.

Odwrotna metoda potęgowa

Jeśli liczba λ jest wartością własną macierzy A, to dla dowolnego $a \notin \operatorname{spect} A$ liczba $1/(\lambda - a)$ jest wartością własną macierzy $(A - aI)^{-1}$. Zauważmy, że jeśli liczba a jest najbliżej wartości własnej λ_i macierzy A (tj. $|\lambda_i - a| < |\lambda_j - a|$ dla każdego $j \neq i$), to wartość własna $1/(\lambda_i - a)$ macierzy $(A - aI)^{-1}$ dominuje; co więcej, im lepsze przybliżenie a wartości własnej λ_i wybierzemy, tym szybsza jest zbieżność metody potęgowej zastosowanej do macierzy $(A - aI)^{-1}$.

Otrzymana na podstawie powyższego spostrzeżenia odwrotna metoda potęgowa, zwana też metodą Wielandta, umożliwia obliczenie dowolnej wartości własnej macierzy A (a nie tylko dominującej), a poza tym umożliwia otrzymanie szybkiej zbieżności. Algorytm jest taki:

1. Oblicz macierz $B = A - aI$ i rozłóż ją (np. na czynniki trójkątne, za pomocą eliminacji Gaussa).
2. Przyjmij $z^{(0)} \neq 0$,
3. Dla $k = 1, 2, \ldots$ obliczaj

$$y^{(k)} = B^{-1} z^{(k-1)}, \quad \text{rozwiązując układ równań} \quad By^{(k)} = z^{(k-1)},$$
$$z^{(k)} = \frac{1}{\|y^{(k)}\|_2} y^{(k)}.$$

Ciąg wektorów $(z^{(k)})_{k\in\mathbb{N}}$ dąży do wektora własnego macierzy B^{-1}, który jest także wektorem własnym macierzy A. Na podstawie ilorazu Rayleigha $\rho_k = z^{(k)T} y^{(k-1)}$ można obliczyć przybliżenie $\lambda_i \approx 1/\rho_k + a$.

Jeśli macierz A jest pełna, to koszt jej rozłożenia w kroku pierwszym jest rzędu n^3, zaś koszt rozwiązywania układu równań w każdej iteracji jest rzędu n^2, czyli taki sam jak koszt jednej iteracji zwykłej metody potęgowej. Koszt jednej iteracji można zmniejszyć, dokonując wstępnego przekształcenia macierzy, co będzie opisane dalej.

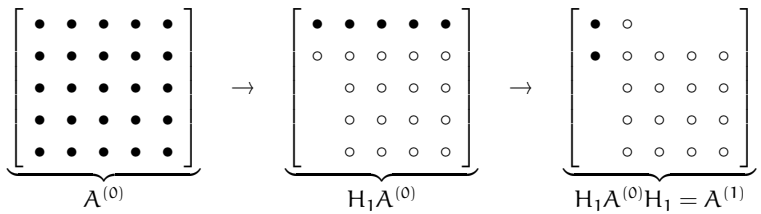
Przybliżenie wartości własnej otrzymane na podstawie ilorazu Rayleigha po wykonaniu pewnej liczby iteracji umożliwia (znaczne) przyspieszenie zbieżności, kosztem ponownego rozkładania na czynniki macierzy $A - \alpha' I$. Macierz ta jest źle uwarunkowana (tym gorzej, im lepszym przybliżeniem wartości własnej macierzy A jest liczba α'), ale ponieważ prawa strona rozwiązywanego układu równań jest przybliżeniem wektora własnego przynależnego do dominującej wartości własnej macierzy $(A - \alpha' I)^{-1}$, okazuje się, że skutki błędów zaokrągleń nie są groźne dla dokładności obliczeń.

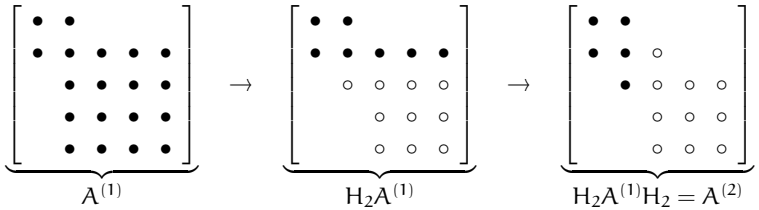
Sprowadzanie macierzy symetrycznej do postaci trójdiagonalnej

Wprawdzie (dla macierzy $n \times n$, gdzie $n > 4$) na ogół *nie można* w skończenie wielu krokach skonstruować macierzy X, takiej że macierz $\Lambda = X^{-1}AX$ jest diagonalna, ale dla macierzy symetrycznej *można* skonstruować macierz ortogonalną U, taką że macierz $T = U^{-1}AU$ jest trójdiagonalna. Koszt tego obliczenia jest (dla macierzy pełnej) rzędu n^3, ale można je wykonać jednorazowo, a następnie rozwiązać zagadnienie własne dla macierzy T; ma ona te same wartości własne, co macierz A, jeśli zaś wektor $\mathbf{y}$ jest wektorem własnym macierzy T, to wektor $\mathbf{x} = U\mathbf{y}$ jest wektorem własnym macierzy A. Zarówno koszt obliczania iloczynu $\mathbf{y}^{(k)} = T\mathbf{z}^{(k-1)}$, jak i koszt rozwiązywania układu równań $(T - \alpha I)\mathbf{y}^{(k)} = \mathbf{z}^{(k-1)}$, jest rzędu n. Wstępne przekształcenie macierzy do postaci trójdiagonalnej jest też wstępnym krokiem wielu innych algorytmów rozwiązywania algebraicznego zagadnienia własnego.

Opiszemy algorytm Ortegi-Householdera. Otrzymana w nim macierz U jest iloczynem macierzy $n-2$ odbić Householdera; jak zwykle, nie wyznaczamy jej w postaci jawnej, tylko zapamiętujemy odpowiedni ciąg wektorów normalnych hiperpłaszczyzn odbić. Obliczenie polega na skonstruowaniu ciągu macierzy symetrycznych, $A^{(0)} = A, A^{(1)}, \ldots, A^{(n-2)} = T$. Współczynniki macierzy $A^{(k)}$ spełniają warunek $a^{(k)}_{ij} = a^{(k)}_{ji} = 0$ dla $j \leqslant k$ oraz $i > j+1$. Ponadto, jeśli $i < k$ lub $j < k$, to $a^{(k)}_{ij} = a^{(k-1)}_{ij}$.

Idea przekształcenia jest przedstawiona na schemacie:





W podanych wyżej schematach symbol „•” oznacza oryginalny lub niezmieniony współczynnik macierzy, zaś „○” oznacza współczynnik, który wskutek odbicia uległ zmianie. Puste miejsca oznaczają (wytworzone lub zachowane) zera.

Pierwsza współrzędna wektora $\boldsymbol{\nu}_1$, określającego odbicie reprezentowane przez macierz $H_1 = I - \gamma_1 \boldsymbol{\nu}_1 \boldsymbol{\nu}_1^T$, jest równa 0. Dla takiego odbicia macierze $A^{(0)}$ i $H_1A^{(0)}$ mają taki sam pierwszy wiersz. Odbicie konstruujemy w taki sposób, aby w pierwszej kolumnie macierzy $H_1A^{(0)}$ w wierszach $3, \ldots, n$ otrzymać zera. Mnożenie przez macierz odbicia z prawej strony zachowuje pierwszą kolumnę macierzy $H_1A^{(0)}$, w tym jej zerowe współczynniki. Wykonane przekształcenie $A^{(0)} \to A^{(1)}$ jest podobieństwem macierzy, ponieważ macierz H_1 jest symetryczna i ortogonalna. Ponadto przekształcenie to zachowuje symetrię, a zatem w pierwszym wierszu macierzy $A^{(1)}$, w kolumnach $3, \ldots, n$ też mamy zera.

Wektor $\boldsymbol{\nu}_2$ ma dwie pierwsze współrzędne równe zero, czego konsekwencją jest zachowanie pierwszego wiersza i pierwszej kolumny macierzy $A^{(1)}$.

Teraz implementacja. W k-tym kroku mamy obliczyć macierz

$$\begin{aligned} A^{(k)} &= H_k A^{(k-1)} H_k = (I - \gamma_k \boldsymbol{\nu}_k \boldsymbol{\nu}_k^T) A^{(k-1)} (I - \gamma_k \boldsymbol{\nu}_k \boldsymbol{\nu}_k^T) \\ &= A^{(k)} - \gamma_k \boldsymbol{\nu}_k \boldsymbol{\nu}_k^T A^{(k-1)} - \gamma_k A^{(k-1)} \boldsymbol{\nu}_k \boldsymbol{\nu}_k^T + \gamma_k^2 \boldsymbol{\nu}_k \boldsymbol{\nu}_k^T A^{(k-1)} \boldsymbol{\nu}_k \boldsymbol{\nu}_k^T. \end{aligned}$$

Oznaczmy $\boldsymbol{w} = \gamma_k A^{(k-1)} \boldsymbol{\nu}_k$. Wtedy

$$A^{(k)} = A^{(k-1)} - \boldsymbol{\nu}_k \boldsymbol{w}^T - \boldsymbol{w} \boldsymbol{\nu}_k^T + \boldsymbol{\nu}_k (\gamma_k \boldsymbol{\nu}_k^T \boldsymbol{w}) \boldsymbol{\nu}_k^T.$$

Niech $\mathbf{p} = \boldsymbol{w} - \boldsymbol{\nu}_k (\boldsymbol{\nu}_k^T \boldsymbol{w}) \gamma_k / 2$. Możemy sprawdzić, że

$$A^{(k)} = A^{(k-1)} - (\boldsymbol{\nu}_k \mathbf{p}^T + \mathbf{p} \boldsymbol{\nu}_k^T).$$

Właśnie tego wzoru używamy w obliczeniach. Zauważmy, że wektory $\boldsymbol{w}$ i $\mathbf{p}$ obliczone w k-tym kroku mają $k-1$ początkowych współrzędnych równych 0. Dzięki symetrii można obliczać tylko współczynniki na i pod (albo na i nad) diagonalą, dla zmniejszenia kosztu.

Algorytm QR

Niech A będzie nieosobliwą macierzą symetryczną i niech Z_{k-1} będzie dowolną macierzą nieosobliwą $n \times n$. Kolumny macierzy $Y_k = AZ_{k-1}$, zgodnie ze spostrzeżeniami, na których opiera się metoda potęgowa, mają „kierunki bliższe" kierunku wektora własnego $\boldsymbol{x}_1$, przynależnego do dominującej wartości własnej, λ_1. Ale gdybyśmy układ wektorów $\boldsymbol{y}_1^{(k)}, \ldots, \boldsymbol{y}_n^{(k)}$, tj. kolumn macierzy Y_k poddali ortonormalizacji Grama-Schmidta, to otrzymalibyśmy układ wektorów $\boldsymbol{z}_1^{(k)}, \ldots, \boldsymbol{z}_n^{(k)}$, z których każdy ma „kierunek bliższy" kierunku wektora przynależnego do kolejnej wartości własnej (zakładamy, że wartości własne są ponumerowane w taki sposób, że $|\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_n|$). Jest tak dlatego, bo ortonormalizacja „likwiduje" składowe wektora $\boldsymbol{y}_i^{(k)}$ w kierunkach wektorów $\boldsymbol{z}_1^{(k)}, \ldots, \boldsymbol{z}_{i-1}^{(k)}$, które są przybliżeniami wektorów własnych $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}$ macierzy A. Stąd wynika przypuszczenie, że dla każdego $i \in \{1, \ldots, n\}$ ciąg wektorów $(\boldsymbol{z}_i^{(k)})_{k\in\mathbb{N}}$ dąży do wektora własnego $\boldsymbol{x}_i$ przynależnego do wartości własnej λ_i.

Macierz $Z_k = [\boldsymbol{z}_1^{(k)}, \ldots, \boldsymbol{z}_n^{(k)}]$ jest ortogonalna, a ponadto istnieje macierz trójkątna górna R_k, taka że $Y_k = Z_k R_k$. Przyjmijmy $Z_0 = I$ i oznaczmy

$$A_k \stackrel{\text{def}}{=} Z_k^T A Z_k$$

(czyli w szczególności $A_0 = A$, ponadto wszystkie macierze A_k są podobne do A i symetryczne). Wtedy

$$A_k = Z_k^T Y_{k+1} = Z_k^T Z_{k+1} R_{k+1}.$$

Niech $Q_{k+1} = Z_k^T Z_{k+1}$. Stąd $Z_{k+1} = Z_k Q_{k+1}$, a przez indukcję mamy stąd

$$Z_k = Z_0 Q_1 \ldots Q_k = Q_1 \ldots Q_k.$$

Na tej podstawie

$$A_k = Q_k^T \ldots Q_1^T A Q_1 \ldots Q_k = Q_k^T A_{k-1} Q_k = R_k Q_k.$$

Podany wyżej rachunek jest podstawą dla następującego algorytmu:

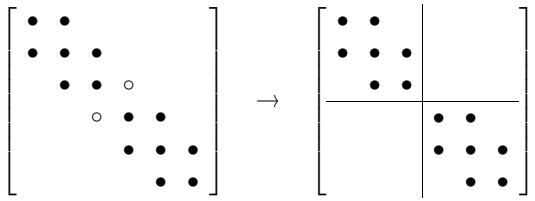
1. Przyjmij $A_0 = A$,
2. Dla $k = 1, 2, \ldots$

   znajdź macierze ortogonalną Q_k i trójkątną górną R_k,
   takie że $A_{k-1} = Q_k R_k$,
   oblicz $A_k = R_k Q_k$.

Jeśli ciąg macierzy $(Z_k)_{k\in\mathbb{N}}$ zbiega do macierzy X, której kolumny są wektorami własnymi macierzy A, to ciąg macierzy $(A_k)_{k\in\mathbb{N}}$ zbiega do macierzy diagonalnej Λ, której znalezienie jest równoznaczne z obliczeniem wszystkich wartości własnych. Zbieżność może jednak nie mieć miejsca. Aby ją po pierwsze osiągnąć, a po drugie sprawić, by była jak najszybsza, w kolejnych iteracjach dobiera się parametr a_k (tzw. przesunięcie) i znajduje czynniki rozkładu macierzy $A_{k-1} - a_k I = Q_k R_k$, a następnie oblicza się macierz $A_k = R_k Q_k + a_k I$. Dla dowolnego a_k otrzymana w ten sposób macierz A_k też jest podobna do A_{k-1}. Istnieją różne sposoby dobierania przesunięcia; jego wartość powinna przybliżać jedną z wartości własnych macierzy A. Najprostszy (i skuteczny) wybór to $a_k = a_{nn}^{(k-1)}$, bardziej wyrafinowane sposoby pominiemy.

Okazuje się, że jeśli macierz A_{k-1} jest trójdiagonalna, to macierz A_k też jest. Dlatego pierwszym etapem obliczeń powinno być przekształcenie macierzy do postaci trójdiagonalnej (przy użyciu algorytmu Ortegi-Householdera), co kosztuje $O(n^3)$ operacji. Przekształcenia w kolejnych iteracjach zachowują tę postać. Rozkładanie macierzy $A_{k-1} - a_k I$ na czynniki Q_k i R_k może być wykonane dowolną metodą, niekoniecznie przez ortonormalizację Grama-Schmidta. Zwykle rozkładu macierzy dokonuje się za pomocą tzw. obrotów Givensa. Koszt jednej iteracji dla macierzy trójdiagonalnej jest rzędu n.

Współczynniki diagonalne kolejnych macierzy A_k dążą do wartości własnych, zaś współczynniki kodiagonalne (tj. sąsiadujące z diagonalą) dążą do zera. Jeśli wartość bezwzględna pewnego współczynnika na kodiagonali jest dostatecznie mała, tj. na poziomie błędów zaokrągleń, to współczynnik ten zastępuje się zerem, ale wtedy powstaje macierz blokowo-diagonalna z trójdiagonalnymi blokami:



i obliczenia można kontynuować dla tych bloków niezależnie, dobierając niezależnie przesunięcia. Przejście od zadania postawionego dla całej macierzy do zadań w mniejszych blokach nazywa się deflacją. Algorytm QR ze wstępnym przekształceniem do postaci trójdiagonalnej, przesunięciami i rekurencyjną deflacją jest najefektywniejszym znanym algorytmem znajdowania wszystkich wartości własnych macierzy symetrycznej.

Zadania i problemy

1. Niech $A = [a_{ij}]_{i,j} \in \mathbb{C}^{n,n}$. Kołem Gerszgorina nazywa się zbiór liczb zespolonych z spełniających nierówność $|z - a_{ii}| \leqslant \sum_{j \neq i} |a_{ij}|$.

   Twierdzenie Gerszgorina: *Każda wartość własna macierzy A leży w pewnym kole Greszgorina.*

   Wskazówka do dowodu: Wybierz wektor własny $\mathbf{x}$, spełniający warunek $\|\mathbf{x}\|_\infty = 1$, i zbadaj koło Gerszgorina o środku a_{ii}, takie że $|x_i| = 1$.

   Wnioski: 1. Ponieważ macierz A^T ma to samo widmo co A, wartości własne leżą w przecięciu sumy kół Gerszgorina obu tych macierzy.
   2. Macierz diagonalnie dominująca jest nieosobliwa.

2. Zbadaj, jak zmieni się widmo macierzy dwudiagonalnej $n \times n$

$$\begin{bmatrix} a & 1 & & \\ & a & \ddots & \\ & & \ddots & 1 \\ & & & a \end{bmatrix}$$

   po zastąpieniu współczynnika $a_{n1} = 0$ przez $\varepsilon \neq 0$.

3. Udowodnij, że jeśli wartości własne λ_i, λ_j macierzy $A = A^T \in \mathbb{R}^{n,n}$ są różne, to przynależne do nich wektory własne są prostopadłe.

4. Niech $\rho(A)$ oznacza promień spektralny macierzy A, tj. największą wartość bezwzględną wartości własnej tej macierzy. Udowodnij, że norma druga indukowana macierzy A wyraża się wzorem

$$\|A\|_2 = \sqrt{\rho(A^T A)}.$$

   Jeśli macierz A jest symetryczna, to jest też $\|A\|_2 = \rho(a)$. Korzystając z tego wzoru i podanych wiadomości o zagadnieniu własnym, wykaż, że dla macierzy symetrycznej $\operatorname{cond}_2 A = |\lambda_{max}|/|\lambda_{min}|$ (λ_{max} i λ_{min} to odpowiednio wartości własne o największej i najmniejszej wartości bezwzględnej).

5. Udowodnij, że jeśli macierz T, symetryczna i trójdiagonalna, ma współczynniki kodiagonalne (tj. sąsiadujące z diagonalnymi) niezerowe, to każda jej wartość własna ma krotność 1.

6. Niech $c, s \in \mathbb{R}$ i $c^2 + s^2 = 1$. Dla $i \neq j$ macierz G_{ij}, która ma współczynniki $g_{ii} = g_{jj} = c$, $g_{ji} = -g_{ij} = s$, a pozostałe współczynniki takie, jak macierz jednostkowa, jest macierzą obrotu w płaszczyźnie $\operatorname{lin}\{\boldsymbol{e}_i, \boldsymbol{e}_j\}$; istnieje liczba ϕ, taka że $c = \cos\phi$ oraz $s = \sin\phi$ — jest to kąt obrotu. Macierz G_{ij} jest ortogonalna. Iloczyn $G_{ij}A$ macierzy obrotu i dowolnej macierzy A ma takie same wiersze co A, poza i-tym i j-tym. Iloczyn AG_{ij} ma takie same kolumny poza i-tą i j-tą.

Niech $A \in \mathbb{R}^{n,n}$ będzie macierzą daną i niech $B = G_{ij}A$. Kąt obrotu można dobrać tak, aby współczynnik b_{ij} macierzy B był równy 0; tak skonstruowane przekształcenie nazywa się obrotem Givensa.

Przypuśćmy, że $a_{ij} \neq 0$; w przeciwnym razie wystarczy wziąć $\phi = 0$, tj. $G_{ij} = I$. Jeśli $a_{ij} \neq 0$, to przyjmujemy $\phi = \operatorname{arctg} a_{ij}/a_{jj}$, skąd mamy

$$c = \frac{a_{jj}}{\sqrt{a_{jj}^2 + a_{ij}^2}}, \qquad s = -\frac{a_{ij}}{\sqrt{a_{jj}^2 + a_{ij}^2}}.$$

Nie ma zatem potrzeby obliczania wartości funkcji trygonometrycznych ani cyklometrycznych, jest tylko pierwiastek kwadratowy. Obrót możemy reprezentować za pomocą jednej liczby ξ, określonej według algorytmu Stewarta:

```
if ( |a_jj| > |a_ij| ) {
  if ( a_jj ≠ 0 ) { d = a_ij/a_jj;  r = sqrt(1 + d^2);  ξ = d/r;  }
  else ξ = 0;
}
else {
  if ( a_ij ≠ 0 ) { d = a_jj/a_ii;  r = sqrt(1 + d^2);  ξ = r/d;  }
  else ξ = 1;
}
```

Dokonując rozkładu macierzy na czynniki ortogonalny i trójkątny, można liczbę ξ zapamiętać na miejscu współczynnika a_{ij}. Na podstawie ξ możemy obliczyć c i s:

```
if ( |ξ| < 1 ) { c = sqrt(1 − ξ^2);  s = −ξ;  }
else if ( |ξ|==1 ) { c = 0;  s = 1;  }
else { c = 1/ξ;  s = −sqrt(1 − c^2);  }
```

7. Udowodnij, że jeśli macierz T jest symetryczna i trójdiagonalna i $T = QR$, gdzie Q jest macierzą ortogonalną, a R trójkątną górną, to macierz $RQ = Q^T TQ$ też jest symetryczna i trójdiagonalna.

   Wskazówka. Przedstaw macierz Q w postaci iloczynu $G_{12}G_{23} \cdot \ldots \cdot G_{n-1,n}$ macierzy obrotów Givensa i sprawdź, że iloczyn RQ ma pod dolną kodiagonalą zerowe współczynniki. Następnie powołaj się na symetrię.

Interpolacja wielomianowa

Zadania interpolacyjne Lagrange'a i Hermite'a

Niech $x_0, \ldots, x_n$ będą danymi liczbami, z których każde dwie są różne i niech $y_0, \ldots, y_n$ będą liczbami dowolnymi. Zadanie interpolacyjne Lagrange'a polega na skonstruowaniu wielomianu $h(x)$ stopnia co najwyżej n, takiego że $h(x_i) = y_i$ dla $i = 0, \ldots, n$.

Wymaganie, aby liczby x_i, zwane węzłami interpolacyjnymi, były parami różne, jest oczywiste; nie można zadawać dwóch różnych wartości funkcji w tym samym punkcie. Ale możemy dopuścić, aby węzły powtarzały się, jeśli dla każdego dodatkowego „egzemplarza" węzła określimy inny warunek interpolacyjny. Jeśli warunek ten polega na podaniu wartości pochodnej kolejnego rzędu, to mamy ogólniejsze zadanie interpolacyjne Hermite'a: dla każdego węzła określamy jego krotność — jest to liczba jego wystąpień w danym ciągu węzłów. Dla węzła x_i o krotności $r > 1$ zadajemy wartość funkcji, $h(x_i)$, pochodnej, $h'(x_i)$, i pochodnych do rzędu $r-1$ włącznie.

Zadanie interpolacyjne Hermite'a i jego przypadek szczególny — zadanie interpolacyjne Lagrange'a — ma jednoznaczne rozwiązanie. Jeśli poszukiwany wielomian przedstawimy jako kombinację liniową elementów dowolnej bazy przestrzeni $\mathbb{R}[x]_n$ (przestrzeni wielomianów stopnia co najwyżej n), to możemy warunki interpolacyjne zapisać w postaci układu równań liniowych, z niewiadomymi współczynnikami w wybranej bazie. Wymiar przestrzeni, czyli liczba niewiadomych, jest równy $n+1$, tj. taki sam jak liczba równań. Przypuśćmy, że warunki interpolacyjne są jednorodne, tj. wszystkie zadane wartości funkcji i pochodnych są równe 0. Wtedy rozwiązaniem układu jest wektor zerowy, który reprezentuje wielomian zerowy. Gdyby istniał niezerowy wielomian $h(x)$ stopnia co najwyżej n spełniający te same warunki interpolacyjne, to musiałby być podzielny przez wielomian $p_{n+1}(x) = (x-x_0)\cdot\ldots\cdot(x-x_n)$, ale to oznacza, że stopień wielomianu h musiałby być co najmniej $n+1$. Jednoznaczność rozwiązania układu równań opisującego jednorodne warunki interpolacyjne oznacza, że macierz tego układu jest nieosobliwa, a więc dla dowolnej prawej strony (tj. dowolnych zadanych wartości funkcji i pochodnych) zadanie ma jednoznaczne rozwiązanie.

Rozwiązanie zadania interpolacyjnego Lagrange'a można przedstawić wzorem

$$h(x) = \sum_{i=0}^{n} y_i \Phi_i(x), \quad \text{gdzie} \quad \Phi_i(x) = \prod_{j \in \{0,\ldots,n\}\setminus\{i\}} \frac{x - x_j}{x_i - x_j},$$

ale wzór ten nie jest praktyczny w obliczeniach numerycznych (należy go raczej traktować jako dowód istnienia rozwiązania zadania, czasem przydaje się też w rachunkach symbolicznych i w rozważaniach teoretycznych).

Bazy Newtona

Niech $x_0, \ldots, x_n$ będą liczbami danymi. Możemy określić wielomiany

$$\begin{aligned} p_0(x) &= 1, \\ p_1(x) &= x - x_0, \\ p_2(x) &= (x-x_0)(x-x_1), \\ &\vdots \\ p_n(x) &= (x-x_0)\cdot\ldots\cdot(x-x_{n-1}), \\ p_{n+1}(x) &= (x-x_0)\cdot\ldots\cdot(x-x_{n-1})(x-x_n). \end{aligned}$$

Zbiór wielomianów $\{p_0, \ldots, p_k\}$ jest bazą przestrzeni $\mathbb{R}[x]_k$, której elementami są wszystkie wielomiany stopnia co najwyżej k. Ta tzw. baza Newtona jest wygodniejsza od bazy potęgowej w zastosowaniu do zadań interpolacji wielomianowej[6]. W szczególności, mając współczynniki $b_0, \ldots, b_n$ wielomianu stopnia co najwyżej n, możemy obliczyć wartość wielomianu $w(x) = \sum_{i=0}^{n} b_i p_i(x)$ za pomocą odpowiednio uogólnionego schematu Hornera:

```
w = b_n;
for ( i = n-1; i >= 0; i-- )
  w = w*(x - x_i) + b_i;
```

Aby rozwiązać zadanie interpolacyjne Lagrange'a, możemy dla wybranej bazy $\{f_0, \ldots, f_n\}$ przestrzeni $\mathbb{R}[x]_n$ utworzyć macierz $A \in \mathbb{R}^{n+1,n+1}$, taką że jej współczynnik $a_{ij} = f_i(x_j)$ (numerujemy tu wiersze i kolumny od 0 do n). Rozwiązanie zadania sprowadza się do rozwiązania układu równań z tą macierzą. Dla bazy potęgowej mamy układ równań z macierzą pełną

$$\begin{bmatrix} 1 & x_0 & \ldots & x_0^n \\ 1 & x_1 & \ldots & x_1^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \ldots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix},$$

którego rozwiązaniem jest wektor współczynników wielomianu $h(x) = \sum_{k=0}^{n} a_k x^k$, zaś dla bazy Newtona określonej za pomocą węzłów interpolacyjnych mamy układ

[6] Baza potęgowa jest szczególnym przypadkiem bazy Newtona, dla $x_0 = \cdots = x_{k-1} = 0$.

z macierzą trójkątną dolną:

$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & p_1(x_1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 1 & p_1(x_n) & \dots & p_n(x_n) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Możemy obliczyć współczynniki tej macierzy i rozwiązać układ kosztem tylko $\Theta(n^2)$ operacji (dalej poznamy inny algorytm obliczania współczynników wielomianu interpolacyjnego w bazie Newtona). W razie potrzeby, możemy następnie kosztem $\Theta(n^2)$ operacji przejść do bazy potęgowej, ale jeśli nie jest to konieczne, to nie warto tego robić.

Różnice dzielone

Niech f oznacza pewną funkcję $A \subset \mathbb{R} \to \mathbb{R}$. Dla ustalonych liczb $x_i \in A$ (węzłów interpolacyjnych) określamy różnice dzielone rzędu 0:

$$f[x_i] \stackrel{\text{def}}{=} f(x_i).$$

Zakładajac, że węzły są jednokrotne (czyli parami różne), możemy następnie określić dla $k > 0$ różnice dzielone rzędu k wzorem

$$f[x_i, \dots, x_{i+k}] \stackrel{\text{def}}{=} \frac{f[x_i, \dots, x_{i+k-1}] - f[x_{i+1}, \dots, x_{i+k}]}{x_i - x_{i+k}}. \qquad (*)$$

Różnicę dzieloną można postrzegać na dwa sposoby:

1. Dla ustalonych węzłów $x_i, \dots, x_{i+k}$ jest to kombinacja liniowa wartości funkcji f w tych węzłach, a zatem jest to funkcjonał liniowy w przestrzeni funkcji o ustalonej dziedzinie A, do której należą te węzły,
2. Dla ustalonej funkcji f jest to funkcja $k+1$ zmiennych. Łatwo jest dowieść, że jest to funkcja symetryczna, tj. dowolne przestawienie jej argumentów (węzłów) nie zmienia jej wartości.

Jak wiemy, jeśli wyrażenie $\lim_{h\to 0} f[x, x+h]$ ma określoną (i skończoną) wartość, to jest to pochodna funkcji f w punkcie x. Możemy zatem rozszerzyć definicję różnicy dzielonej na przypadek, gdy pewne (lub nawet wszystkie) węzły mają krotności większe niż 1, wykorzystując przejście do granicy. Okazuje się (co uzasadnimy później), że jeśli funkcja f jest klasy C^k, to różnica dzielona, widziana jako funkcja, której argumentami są węzły, jest ciągła i zachodzi równość

$$\lim_{x_{i+1},\dots,x_{i+k}\to x_i} f[x_i, \dots, x_{i+k}] = \frac{f^{(k)}(x_i)}{k!}.$$

Na tej podstawie możemy zdefiniować różnicę dzieloną rzędu k w przypadku, gdy $x_i = \dots = x_{i+k}$, wzorem

$$f[\underbrace{x_i, \dots, x_i}_{k+1}] \stackrel{\text{def}}{=} \frac{f^{(k)}(x_i)}{k!}, \qquad (**)$$

natomiast w przypadku, gdy pewne węzły mają krotność większą niż 1, ale nie wszystkie węzły są jednakowe, możemy (dzięki symetrii) uporządkować je tak, aby było $x_i \neq x_{i+k}$, i użyć wzoru (*).

W przypadku ogólnym różnica dzielona rzędu k, $f[x_i, \dots, x_{i+k}]$, jest kombinacją liniową wartości funkcji f i jej pochodnych w węzłach, przy czym jeśli pewien węzeł ma krotność r, to kombinacja obejmuje pochodne funkcji f w tym węźle do rzędu $r-1$.

Algorytm różnic dzielonych

Przypuśćmy, że węzły $x_0, \dots, x_n$ są parami różne. Obliczmy różnicę dzieloną wielomianu $p_k(x)$ należącego do bazy Newtona określonej dla tych węzłów:

$$p_k[x, x_0] = \frac{(x - x_0) \cdot \ldots \cdot (x - x_{k-1}) - (x_0 - x_0) \cdot \ldots \cdot (x_0 - x_{k-1})}{x - x_0} = (x - x_1) \cdot \ldots \cdot (x - x_{k-1}).$$

Otrzymaliśmy wielomian stopnia $k-1$. Obliczając różnice dzielone coraz wyższych rzędów, dostaniemy wielomiany coraz niższych stopni:

$$\begin{aligned} p_k[x, x_0, x_1] &= (x - x_2) \cdot \ldots \cdot (x - x_{k-1}), \\ &\vdots \\ p_k[x, x_0, \dots, x_{k-2}] &= (x - x_{k-1}), \\ p_k[x, x_0, \dots, x_{k-2}, x_{k-1}] &= 1. \end{aligned}$$

Po ostatnim kroku możemy oczywiście podstawić $x = x_k$, co nie zmieni wartości otrzymanego wielomianu stopnia 0. Różnice dzielone rzędów wyższych niż k są równe 0. Biorąc pod uwagę zbiór miejsc zerowych wielomianu p_k, mamy

$$p_k[x_0, \dots, x_i] = \begin{cases} 0 & \text{dla } i \neq k, \\ 1 & \text{dla } i = k. \end{cases}$$

Podany wyżej rachunek „przechodzi” też na przypadek węzłów powtarzających się — wystarczy użyć indukcji i w kroku indukcyjnym dokonać odpowiedniego przejścia do granicy.

Niech $h(x)$ będzie rozwiązaniem zadania interpolacyjnego Lagrange'a dla węzłów $x_0, \ldots, x_n$. Wielomian h możemy przedstawić jako kombinację liniową elementów bazy Newtona: $h(x) = \sum_{k=0}^{n} b_k p_k(x)$. Z tego, że dla ustalonych węzłów różnice dzielone są funkcjonałami liniowymi, i z przeprowadzonego wyżej rachunku wynika, że zachodzą równości

$$h[x_0, \ldots, x_i] = \sum_{k=0}^{n} b_k p_k[x_0, \ldots, x_i] = b_i.$$

Znamy wartości wielomianu h w węzłach interpolacyjnych, są nimi liczby $y_0, \ldots, y_n$, a zatem możemy obliczyć współczynniki $b_0, \ldots, b_n$ wielomianu h w bazie Newtona. Wygodnie jest przedstawić ich obliczenie za pomocą schematu

$$\begin{array}{c|ccccc}
x_0 & y_0 = b_0 & & & & \\
 & \searrow & & & & \\
x_1 & y_1 & \to & h[x_0, x_1] = b_1 & & \\
 & \searrow & & \searrow & & \\
x_2 & y_2 & \to & h[x_1, x_2] & \to & h[x_0, x_1, x_2] = b_2 \\
\vdots & \vdots & & \vdots & & \ddots \\
 & \searrow & & \searrow & & \searrow \\
x_n & y_n & \to & h[x_{n-1}, x_n] & \to & h[x_{n-2}, x_{n-1}, x_n] \ldots \to h[x_0, \ldots, x_n] = b_n
\end{array}$$

Podprogram realizujący to obliczenie zastępuje w tablicy y dane wartości funkcji przez współczynniki $b_0, \ldots, b_n$:

```
for ( j = 1; j ⩽ n; j++ )
  for ( i = n; i ⩾ j; i-- )
    y[i] = (y[i] − y[i − 1])/(x[i] − x[i − j]);
```

Aby rozwiązać zadanie interpolacyjne Hermite'a, należy zmodyfikować ten algorytm. Istotne jest uporządkowanie danych; wymagamy, aby w tablicy x wszystkie „egzemplarze" węzła krotnego występowały obok siebie. W tablicy y zadaną wartość funkcji podajemy w miejscu odpowiadającym pierwszemu wystąpieniu odpowiedniego węzła, a na kolejnych miejscach podajemy wartości kolejnych pochodnych. Algorytm można zrealizować w taki sposób:

```
k[0] = 0;
for ( i = 1; i ⩽ n; i++ )
  k[i] = x[i] == x[i − 1] ? k[i − 1] + 1 : 0;
for ( j = 1; j ⩽ n; j++ )
  for ( i = n; i ⩾ j; i-- )
    if ( k[i] == 0 )
      y[i] = (y[i] − y[i − 1 − k[i − 1]])/(x[i] − x[i − j]);
    else { y[i] /= j; k[i]--; }
```


W pierwszej pętli w miejscu i-tym pomocniczej tablicy k zapisujemy informację, którego rzędu pochodnej wartością jest dana liczba y[i]. W drugiej pętli używamy tej informacji do wybrania odpowiedniej instrukcji: obliczenia różnicy dzielonej za pomocą wzoru (*) lub podzielenia y[i] przez odpowiednią liczbę całkowitą, co prowadzi do otrzymania silni w mianowniku wzoru (**).

Dowodzi się, że jeśli węzły są monotonicznie uporządkowane, to algorytm różnic dzielonych jest numerycznie poprawny, tj. obliczone współczynniki w bazie Newtona reprezentują wielomian interpolacyjny dla odpowiednio mało zaburzonych danych wartości funkcji (i jej pochodnych).

Reszta interpolacyjna

Z uwagi na liczne zastosowania zadań interpolacyjnych Lagrange'a i Hermite'a w aproksymacji funkcji i w konstrukcji różnych metod numerycznych (np. rozwiązywania równań nieliniowych i obliczania całek), duże znaczenie ma wzór opisujący resztę interpolacyjną.

Twierdzenie. *Jeśli funkcja f jest klasy C^{n+1} w przedziale $A \subset \mathbb{R}$ i $h(x)$ oznacza wielomian interpolacyjny Hermite'a funkcji f dla węzłów $x_0, \ldots, x_n \in A$, to dla każdego $x \in A$ istnieje liczba $\xi \in A$, taka że*

$$f(x) - h(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} p_{n+1}(x).$$

Dowód. Możemy ponumerować węzły tak, aby tworzyły ciąg niemalejący. Jeśli $x = x_i$ dla pewnego $i \in \{0, \ldots, n\}$, to $p_{n+1}(x) = 0$ i dowodzona równość jest oczywista (z dowolnym ξ). Dla ustalonego $x \in A \setminus \{x_0, \ldots, x_n\}$ określamy funkcję

$$g_x(s) \stackrel{\text{def}}{=} f(s) - h(s) - z p_{n+1}(s),$$

z parametrem $z = z(x)$, który dobierzemy za chwilę. Korzystając z tego, że $p_{n+1}(x) \neq 0$, bierzemy

$$z = \frac{f(x) - h(x)}{p_{n+1}(x)},$$

i w ten sposób dostajemy $g_x(x) = 0$. Tak określona funkcja ma co najmniej $n+2$ miejsca zerowe tj. wszystkie węzły interpolacyjne[7] i punkt x. Funkcja g_x jest klasy $C^{n+1}(A)$. Jej pochodna rzędu $k \leqslant n+1$ ma co najmniej $n+2-k$ miejsca zerowe. Istotnie, jeśli $x_i < x_{i+1}$, to (z twierdzenia Rolle'a) funkcja g_x, która na

[7] Jeśli pewien węzeł ma krotność $r > 1$, to liczymy go r razy; funkcja g_x przyjmuje w tym punkcie wartość 0 razem z pochodnymi rzędu $1, \ldots, r-1$.

końcach przedziału $[x_i, x_{i+1}]$ przyjmuje tę samą wartość 0, osiąga w tym przedziale maksimum lub minimum, w punkcie będącym miejscem zerowym funkcji g'_x. Jeśli zaś funkcja g_x ma miejsce zerowe o krotności $r > 1$ (w węźle x_i), to jej pochodna ma w tym punkcie miejsce zerowe o krotności $r - 1$. Korzystając z indukcji, stosujemy to rozumowanie do kolejnych pochodnych. Wynika z niego, że pochodna rzędu $n+1$ funkcji g_x ma w przedziale A co najmniej jedno miejsce zerowe, ξ. Podstawiając $s = \xi$, dostajemy

$$0 = g_x^{(n+1)}(\xi) = f^{(n+1)}(\xi) - h^{(n+1)}(\xi) - z p_{n+1}^{(n+1)}(\xi).$$

Pochodna rzędu $n+1$ wielomianu $h(s)$ (stopnia n) jest równa 0, zaś pochodna wielomianu $p_{n+1}(s)$ (stopnia $n+1$), którego współczynnik (w bazie potęgowej) przy s^{n+1} jest równy 1, jest dla każdego s równa $(n+1)!$. Zatem

$$z = \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Dowód zakończymy, wstawiając to do definicji funkcji g_x i biorąc $s = x$. □

Ze wzoru na resztę interpolacyjną łatwo wynika[8] podany wcześniej bez dowodu fakt, że jeśli funkcja jest klasy C^k w otoczeniu punktu x_i, to

$$\lim_{x_{i+1},\ldots,x_{i+k} \to x_i} f[x_i, \ldots, x_{i+k}] = \frac{f^{(k)}(x_i)}{k!},$$

co jest podstawą podanej definicji różnic dzielonych także dla węzłów o krotnościach większych niż 1. Dowiedziony wzór w szczególnym przypadku, gdy $x_0 = \cdots = x_n$, jest wzorem Taylora (z resztą w postaci Lagrange'a). Inny przypadek szczególny, dla dwóch węzłów jednokrotnych, wykorzystaliśmy już w analizie metody siecznych. Dalsze zastosowania nastąpią.

[8] Za przedział A możemy przyjąć najkrótszy przedział zawierający wszystkie węzły — podczas przejścia do granicy długość tego przedziału zbiega do zera.

Zadania i problemy

1. Rozwiąż zadanie interpolacyjne Lagrange'a (tj. skonstruuj bazę Newtona i oblicz współczynniki wielomianu interpolacyjnego w tej bazie) dla węzłów i wartości funkcji podanych w tabelce:

x_i	-2	-1	0	2	3
$f(x_i)$	17	1	1	1	37

2. Rozwiąż zadanie interpolacyjne Hermite'a dla węzłów i wartości funkcji i pochodnych podanych w tabelce:

x_i	0	1	3
$f(x_i)$	-1	-2	80
$f'(x_i)$		-2	
$f''(x_i)$		12	

3. Przejście między bazą Newtona i potęgową może być dokonane za pomocą schematu Hornera; zobaczmy, w jaki sposób. Rozważmy dzielenie z resztą wielomianu $w(x) = a_n x^n + \cdots + a_1 x + a_0$ przez dwumian $(x - x_0)$:

$$w(x) = \sum_{i=0}^{n} a_i x^i = \left(\sum_{i=0}^{n-1} c_{i+1} x^i \right)(x - x_0) + c_0 = c_n x^n + \sum_{i=0}^{n-1} (c_i - c_{i+1} x_0) x^i.$$

Stąd $c_n = a_n$ oraz $c_i = c_{i+1} x_0 + a_i$ dla $i < n$. Zauważamy, że liczby c_i są kolejno nadawanymi wartościami zmiennej w podczas obliczania za pomocą (zwykłego) schematu Hornera wartości $w(x_0)$. Otrzymujemy resztę z dzielenia, $b_0 = c_0 = w(x_0)$, i współczynniki w bazie potęgowej ilorazu $c(x)$. Jeśli określimy bazę Newtona, której elementami są wielomiany

$$q_0(x) = 1,$$
$$q_i(x) = \prod_{k=1}^{i-1} (x - x_k) \quad \text{dla } i = 1, \ldots, n-1,$$

to współczynniki wielomianu $c(x)$ w tej bazie są identyczne ze współczynnikami $b_1, \ldots, b_n$ wielomianu w w bazie Newtona $\{p_0, \ldots, p_n\}$. Aby otrzymać b_1, można następnie podzielić wielomian $c(x)$ przez $(x - x_1)$, i tak dalej, rekurencyjnie. Mamy zatem algorytm przejścia od bazy potęgowej do bazy Newtona:

```
for ( j = 0; j < n; j++ )
  for ( i = n - 1; i >= j; i-- )
    a[i] += a[i + 1] * x_j;
```

(algorytm ten zastępuje w tablicy a współczynniki wielomianu w bazie potęgowej współczynnikami w bazie Newtona). Algorytm przejścia w drugą stronę wykonuje operacje przeciwne w odwrotnej kolejności:

```
for ( j = n − 1; j ⩾ 0; j-- )
  for ( i = j; i < n; i++ )
    a[i] -= a[i + 1] ∗ x_j;
```

4. Algorytm Aitkena. Wartość w dowolnym punkcie x wielomianu interpolacyjnego Lagrange'a określonego przez węzły $x_0, \ldots, x_n$ i wartości w tych węzłach $y_0, \ldots, y_n$, może być obliczona (kosztem $O(n^2)$ operacji) bezpośrednio na podstawie tych danych. Przypuśćmy, że dwa wielomiany stopnia co najwyżej $k-1$, $p_i^{(k-1)}(x)$ i $p_{i+1}^{(k-1)}(x)$, są rozwiązaniami zadań interpolacyjnych Lagrange'a odpowiednio dla węzłów $x_i, \ldots, x_{i+k-1}$ oraz $x_{i+1}, \ldots, x_{i+k}$. Wtedy wielomian

$$p_i^{(k)}(x) = \frac{x_{i+k} - x}{x_{i+k} - x_i} p_i^{(k-1)}(x) + \frac{x - x_i}{x_{i+k} - x_i} p_{i+1}^{(k-1)}(x)$$

ma stopień co najwyżej k i jest rozwiązaniem zadania dla węzłów $x_i, \ldots, x_{i+k}$. Istotnie, dla $x = x_i$ pierwszy z ułamków w powyższym wzorze ma wartość 1, a drugi 0 (zatem $p_i^{(k)}(x_i) = p_i^{(k-1)}(x_i) = y_i$, dla $x = x_{i+k}$ ułamki mają wartości 0 i 1 (skąd wynika $p_i^{(k)}(x_{i+k}) = p_{i+1}^{(k-1)}(x_{i+k}) = y_{i+k}$), zaś dla każdego $x \in \mathbb{R}$ (w tym dla $x \in \{x_{i+1}, \ldots, x_{i+k-1}\}$) suma ułamków jest równa 1, a więc $p_i^{(k)}(x_j) = y_j$ dla $j = i+1, \ldots, i+k-1$.

Wartości wielomianów stopnia 0 w punkcie x, $p_i^{(0)}(x) = y_i$, mamy za darmo. Możemy zatem wpisać liczby $y_0, \ldots, y_n$ do tablicy p, i wykonać algorytm

```
for ( k = 1; k ⩽ n; k++ )
  for ( i = 0; i ⩽ n − k; i++ )
    p[i] = ((x_{i+k} − x) ∗ p[i] + (x − x_i) ∗ p[i + 1])/(x_{i+k} − x_i);
```

otrzymując w ten sposób wartość wielomianu interpolacyjnego w zmiennej $p[0]$.

5. Zachodzi wzór Leibniza: jeśli funkcje g i h są odpowiednio gładkie i $f(x) = g(x)h(x)$, to

$$f[x_0, \ldots, x_k] = \sum_{l=0}^{k} g[x_0, \ldots, x_l] h[x_l, \ldots, x_k].$$

Aby go udowodnić, określamy bazy Newtona:

$$p_l(x) = \prod_{j=0}^{l-1} (x - x_j), \qquad q_m(x) = \prod_{j=k-m+1}^{k} (x - x_j).$$

Niech $F(x)$ oznacza iloczyn wielomianów interpolacyjnych Hermite'a funkcji g i h opartych na węzłach $x_0, \ldots, x_k$:

$$F(x) = \sum_{l=0}^{k} g[x_0, \ldots, x_l] p_l(x) \sum_{m=0}^{k} h[x_m, \ldots, x_k] q_{k-m}(x).$$

Dla dowolnego węzła x_i o krotności r funkcje F i f mają w tym węźle takie same wartości i pochodne do rzędu $r-1$ włącznie. Możemy napisać

$$F(x) = \sum_{0 \leqslant l \leqslant m \leqslant k} \big(g[x_0, \ldots, x_l] h[x_m, \ldots, x_k] p_l(x) q_{k-m}(x) \big) + \sum_{0 \leqslant m < l \leqslant k} (\ldots)$$

(składniki obu sum są opisane tym samym wzorem). Dla $m < l$ iloczyn wielomianów $p_l(x)q_{k-m}(x)$ ma w węźle x_i o krotności r miejsce zerowe o krotności co najmniej r, a zatem pierwsza suma (której wszystkie składniki mają stopień nie większy niż k) reprezentuje wielomian interpolacyjny Hermite'a funkcji f, oparty na węzłach $x_0, \ldots, x_k$. Współczynnik tego wielomianu odpowiadający wielomianowi p_k w bazie Newtona i jednocześnie x^k w bazie potęgowej jest równy $f[x_0, \ldots, x_k]$. Składniki stopnia k w pierwszej sumie mają $l = m$, a pozostałe mają stopień niższy. Pozostaje zauważyć, że współczynnik przy x^k (w bazie potęgowej) pierwszej sumy jest równy wyrażeniu po prawej stronie wzoru Leibniza. □

Interpolacja funkcjami sklejanymi

Motywacja dla stosowania funkcji sklejanych

Funkcje sklejane są to funkcje określone w ten sposób, że pewien przedział $[a, b] \subset \mathbb{R}$ (albo, jeśli jest taka potrzeba, cały zbiór liczb rzeczywistych) dzielimy na podprzedziały, wybierając węzły. W każdym przedziale, którego końcami są węzły, funkcja sklejana stopnia n jest wielomianem stopnia co najwyżej n.

W wielu zastosowaniach funkcje sklejane są wygodniejsze niż funkcje wielomianowe. W szczególności, kształt wykresu funkcji sklejanej nawet niskiego stopnia może być dowolnie skomplikowany, łatwo jest więc aproksymować różne funkcje z dobrą dokładnością. Zastosowanie funkcji sklejanych w interpolacji ma również przewagę nad wielomianami. Występujący we wzorze opisującym rozwiązanie zadania interpolacji Lagrange'a wielomian

$$\Phi_i(x) = \prod_{j \in \{0,\dots,n\} \setminus \{i\}} \frac{x - x_j}{x_i - x_j},$$

który przyjmuje wartość 1 dla $x = x_i$ oraz 0 dla $x = x_j \neq x_i$, między swoimi miejscami zerowymi oscyluje i (zależnie od n i od liczb $x_0, \dots, x_n$) może przyjmować wartości wychodzące daleko poza przedział $[0, 1]$. Funkcje sklejane nie mają tej wady, dzięki czemu np. wykres funkcji sklejanej przechodzącej przez zadane punkty wydaje się zwykle zgodny z oczekiwaniami.

Obcięte funkcje potęgowe

Węzły funkcji sklejanej oznaczymy symbolami u_i; przyjmiemy, że tworzą one ciąg rosnący (na razie pominiemy przypadek węzłów krotnych, tj. tworzących ciąg niemalejący, ale niekoniecznie różnowartościowy). Założymy, że wielomiany p_{i-1} i p_i stopnia co najwyżej n, opisujące funkcję sklejaną s odpowiednio w przedziałach (u_{i-1}, u_i) oraz (u_i, u_{i+1}), przyjmują w punkcie u_i tę samą wartość, a ponadto mają takie same pochodne rzędu $1, \dots, n-1$ (uwaga: założenie, że również pochodna rzędu n obu wielomianów jest taka sama oznacza, że to jest ten sam wielomian).

Różnica wielomianów p_i i p_{i-1} musi być zatem wielomianem stopnia co najwyżej n, który w punkcie u_i ma miejsce zerowe o krotności n, ale każdy taki wielomian ma postać $c(x - u_i)^n$ dla pewnej stałej c. Jednym z wielu sposobów określania funkcji sklejanych jest użycie tzw. obciętych funkcji potęgowych,

określonych wzorem

$$(x - u_i)_+^n \stackrel{\text{def}}{=} \begin{cases} (x - u_i)^n & \text{dla } x \geqslant u_i, \\ 0 & \text{dla } x < u_i. \end{cases}$$

Mając węzły np. $u_0, \dots, u_N$, możemy wybrać dowolny wielomian p_{-1} (stopnia co najwyżej n) oraz liczby $c_0, \dots, c_N$, i określić funkcję sklejaną stopnia n wzorem

$$s(x) = p_{-1}(x) + \sum_{i=0}^{N} c_i (x - u_i)_+^n.$$

Takie przedstawienie funkcji sklejanej daje pewne intuicje (np. „od razu widać", że funkcja s jest klasy $C^{n-1}(\mathbb{R})$), ale obcięte funkcje potęgowe są niewygodne w zastosowaniach i oparta na nich reprezentacja jest podatna na błędy zaokrągleń. Dlatego do reprezentowania funkcji sklejanych i przetwarzania ich w obliczeniach numerycznych stosuje się inne sposoby. Chyba *najmniej wygodnym* z tych sposobów jest stosowanie bazy potęgowej.

Uwaga. W tym wykładzie posługuję się pojęciem stopnia funkcji sklejanej, tj. największego dopuszczalnego przez reprezentację stopnia wielomianu opisującego tę funkcję w pewnym przedziale, ale w wielu publikacjach i bibliotekach podprogramów jest w użyciu tzw. rząd funkcji sklejanej, tj. liczba o 1 większa od stopnia. Zatem, np. funkcje sklejane pierwszego rzędu to funkcje stopnia 0, czyli kawałkami stałe, wykresem funkcji sklejanej rzędu 2, czyli stopnia 1, jest łamana.

Z wielu różnych względów w zastosowaniach dominują funkcje sklejane trzeciego stopnia (tzw. kubiczne) i teraz na nich skupimy uwagę.

Reprezentacja Hermite'a funkcji sklejanych trzeciego stopnia

Określamy cztery wielomiany:

$$H_{00}(t) = 2t^3 - 3t^2 + 1, \qquad H_{10}(t) = -2t^3 + 3t^2,$$
$$H_{01}(t) = t^3 - 2t^2 + t, \qquad H_{11}(t) = t^3 - t^2.$$

Wielomiany te są rozwiązaniami zadań interpolacyjnych Hermite'a dla dwóch dwukrotnych węzłów, 0 i 1. Mianowicie, zachodzą równości

$$H_{00}(0) = 1, \quad H_{00}(1) = H'_{00}(0) = H'_{00}(1) = 0,$$
$$H_{10}(1) = 1, \quad H_{10}(0) = H'_{10}(0) = H'_{10}(1) = 0,$$
$$H'_{01}(0) = 1, \quad H_{01}(0) = H_{01}(1) = H'_{01}(1) = 0,$$
$$H'_{11}(1) = 1, \quad H_{11}(0) = H_{11}(1) = H'_{11}(0) = 0.$$

Dzięki temu rozwiązanie zadania interpolacyjnego Hermite'a z tymi węzłami dla dowolnej funkcji f możemy zapisać wzorem

$$h(t) = f(0)H_{00}(t) + f'(0)H_{01}(t) + f(1)H_{10}(t) + f'(1)H_{11}(t).$$

Co więcej, przez zamianę zmiennej możemy znaleźć rozwiązanie zadania interpolacyjnego dla dowolnych dwóch węzłów interpolacyjnych o krotności 2. Jeśli węzłami tymi są liczby u_i i u_{i+1}, i oznaczymy $h_i = u_{i+1} - u_i$ (zakładamy, że $h_i > 0$), to mamy stąd wzór

$$h(x) = f(u_i)H_{i,00}(x) + f'(u_i)H_{i,01}(x) + f(u_{i+1})H_{i,10}(x) + f'(u_{i+1})H_{i,11}(x),$$

w którym użyliśmy funkcji

$$H_{i,00}(x) = H_{00}(t), \qquad H_{i,01}(x) = h_i H_{01}(t),$$
$$H_{i,10}(x) = H_{10}(t), \qquad H_{i,11}(x) = h_i H_{11}(t),$$

przy czym $t = (x - u_i)/h_i$.

Dla ustalonych węzłów $u_0, \ldots, u_N$ (tworzących ciąg rosnący), mając dowolne liczby $a_0, \ldots, a_N$ oraz $b_0, \ldots, b_N$, możemy określić funkcję kawałkami wielomianową w przedziale $[u_0, u_N]$ wzorem

$$s(x) = p_i(x) = a_i H_{i,00}(x) + b_i H_{i,01}(x) + a_{i+1} H_{i,10}(x) + b_{i+1} H_{i,11}(x) \qquad \text{dla } x \in [u_i, u_{i+1}].$$

Jest oczywiste, że funkcja ta jest klasy $C^1[u_0, u_N]$, i w węźle u_i ma wartość a_i, a jej pochodna w u_i jest równa b_i, dla $i = 0, \ldots, N$.

Funkcja s jest funkcją sklejaną, ale w istocie opartą na ciągu węzłów, z których każdy należy liczyć dwukrotnie. Funkcja ta jest (sklejanym) rozwiązaniem zadania interpolacyjnego Hermite'a, w którym dla każdego węzła zadajemy *dwa* warunki interpolacyjne: wartość funkcji i pochodnej. Funkcja ta jest skonstruowana za pomocą wielomianów spełniających warunki interpolacyjne Hermite'a dla dwóch węzłów o krotności 2 (końców każdego przedziału (u_i, u_{i+1})) i dlatego jej reprezentacja w tej postaci jest nazywana reprezentacją Hermite'a[9]

Kubiczne interpolacyjne krzywe sklejane

Aby określić krzywą sklejaną trzeciego stopnia, która jest rozwiązaniem zadania interpolacyjnego Lagrange'a (tj. dla każdego węzła u_i chcemy podać tylko jedną

[9] W czasach, gdy żył Charles Hermite (1822–1901 r.), funkcje sklejane nie były jeszcze znane; odkrył je Schoenberg w 1946 r.

liczbę, a_i, będącą wartością funkcji), skorzystamy z warunku ciągłości pochodnej drugiego rzędu. Zatem, pochodne drugiego rzędu wielomianów $p_{i-1}(x)$ i $p_i(x)$, opisujących poszukiwaną funkcję s w przedziałach (u_{i-1}, u_i) oraz (u_i, u_{i+1}), mają przyjmować w punkcie u_i tę samą wartość (reprezentacja Hermite'a gwarantuje, że będą miały w tym punkcie identyczne wartości i pochodne pierwszego rzędu). Na tej podstawie możemy obliczyć liczby b_i, tj. wartości pochodnej pierwszego rzędu funkcji s w węzłach interpolacyjnych.

Aby wyprowadzić odpowiednie równania, należy obliczyć pochodne wielomianów, za pomocą których przedstawiamy rozwiązanie:

$$H''_{i-1,00}(u_i) = \frac{6}{h_{i-1}^2}, \quad H''_{i,00}(u_i) = \frac{-6}{h_i^2}, \quad H''_{i-1,10}(u_i) = \frac{-6}{h_{i-1}^2}, \quad H''_{i,10}(u_i) = \frac{6}{h_i^2},$$
$$H''_{i-1,01}(u_i) = \frac{2}{h_{i-1}}, \quad H''_{i,01}(u_i) = \frac{-4}{h_i}, \quad H''_{i-1,11}(u_i) = \frac{4}{h_{i-1}}, \quad H''_{i,11}(u_i) = \frac{-2}{h_i}.$$

Zatem, warunek ciągłości drugiej pochodnej w punkcie u_i, $p''_{i-1}(u_i) = p''_i(u_i)$, ma postać

$$\frac{6}{h_{i-1}^2}a_{i-1} - \frac{6}{h_{i-1}^2}a_i + \frac{2}{h_{i-1}}b_{i-1} + \frac{4}{h_{i-1}}b_i = -\frac{6}{h_i^2}a_i + \frac{6}{h_i^2}a_{i+1} - \frac{4}{h_i}b_i - \frac{2}{h_i}b_{i+1}.$$

Po pomnożeniu stron przez $h_{i-1}h_i/2$ i uporządkowaniu, dostajemy równanie

$$h_i b_{i-1} + 2(h_{i-1} + h_i)b_i + h_{i-1}b_{i+1} = 3\left(\frac{h_i}{h_{i-1}}(a_i - a_{i-1}) + \frac{h_{i-1}}{h_i}(a_{i+1} - a_i)\right). \qquad (*)$$

W ten sposób otrzymaliśmy równania ciągłości pochodnych drugiego rzędu funkcji s w „wewnętrznych" węzłach $u_1, \ldots, u_{N-1}$; dla ustalonych liczb $a_0, \ldots, a_N$ musimy znaleźć liczby $y_0, \ldots, y_N$ spełniające te równania. Zauważamy, że liczba niewiadomych jest o 2 większa niż liczba równań. Aby mieć rozwiązanie jednoznaczne, trzeba dołożyć dodatkowe dwa równania.

Dodatkowe równania w konstrukcji sklejanych krzywych interpolacyjnych opisują zwykle tzw. warunki brzegowe, tj. pewne warunki narzucone na pochodne funkcji s w skrajnych węzłach u_0 i u_N. Najprostszy sposób, to arbitralne określenie wartości pochodnych pierwszego rzędu, tj. liczb b_0 i b_N. To jednak może być kłopotliwe dla użytkownika programu. Często stosowanym rozwiązaniem jest żądanie, aby pochodna drugiego rzędu funkcji s była w punktach u_0 i u_N równa 0. Powstaje wtedy tzw. naturalna krzywa sklejana. Na podstawie wypisanych wcześniej wartości funkcji $H_{i,jk}$, równania $p''_0(u_0) = 0$ i $p''_{N-1}(u_N) = 0$ możemy przedstawić w postaci

$$2h_0 b_0 + h_0 b_1 = 3(a_1 - a_0),$$
$$h_{N-1}b_{N-1} + 2h_{N-1}b_N = 3(a_N - a_{N-1}).$$

Po dołączeniu tych równań otrzymujemy układ równań liniowych z macierzą trójdiagonalną. Możemy zauważyć, że dla dowolnego rozmieszczenia węzłów, tj. dla dowolnych dodatnich liczb $h_0, \ldots, h_{N-1}$, macierz ta jest diagonalnie dominująca. Zatem, mamy układ o jednoznacznym rozwiązaniu, które możemy znaleźć za pomocą eliminacji Gaussa kosztem $O(N)$ działań arytmetycznych.

Istnieje wiele innych sposobów określania warunków brzegowych, np. można zażądać, aby pochodna trzeciego rzędu wielomianów p_0 i p_{N-1} była równa 0 (zatem, aby były to wielomiany drugiego stopnia), lub też, aby wielomian p_0 był identyczny z p_1, a wielomian p_{N-1} z p_{N-2} (a więc, aby węzły interpolacyjne u_1 i u_{N-1} *nie były* węzłami funkcji sklejanej — po angielsku nazywa się to warunek *not a knot*).

Jeszcze inna możliwość, to przyjęcie $b_N = b_0$ i wymaganie, aby pochodna drugiego rzędu funkcji s w węzłach u_0 i u_N była taka sama. Wtedy, jeśli $a_0 = a_N$, tj. funkcja s ma tę samą wartość w punktach u_0 i u_N, skonstruujemy tzw. okresową funkcję sklejaną — nakładając warunek, że dla każdego $x \in \mathbb{R}$ $s(x+T) = s(x)$, gdzie $T = u_N - u_0$, otrzymujemy okresową funkcję klasy $C^2(\mathbb{R})$. W układzie równań (*) zastępujemy niewiadomą b_N przez b_0 i dołączamy równanie

$$h_0 b_{N-1} + 2(h_{N-1} + h_0)b_0 + h_{N-1}b_1 = 3\left(\frac{h_0}{h_{N-1}}(a_0 - a_{N-1}) + \frac{h_{N-1}}{h_0}(a_1 - a_0)\right),$$

otrzymując w ten sposób układ równań z macierzą cykliczną trójdiagonalną, diagonalnie dominującą.

Twierdzenie Holladaya

Dla dużej liczby węzłów wielomiany interpolacyjne Lagrange'a „źle się zachowują", tj. ich wartości między sąsiednimi węzłami mogą wystawać daleko poza przedział, którego końcami są wartości wielomianu w tych węzłach. Udowodnimy twierdzenie, które można zinterpretować w ten sposób, że pod tym względem najlepiej, jak tylko się da, zachowuje się naturalna kubiczna interpolacyjna funkcja sklejana. W tym celu określimy funkcjonał, który nazwiemy energią, i który przyjmiemy za miarę „powyginania" wykresów funkcji klasy $C^2[u_0, u_N]$:

$$E(f) \stackrel{\text{def}}{=} \int_{u_0}^{u_N} \big(f''(x)\big)^2 \,dx.$$

Twierdzenie Holladaya. *W zbiorze funkcji klasy $C^2[u_0, u_N]$ spełniających warunki interpolacyjne Lagrange'a określone w węzłach $u_0, \ldots, u_N$ najmniejszą energię ma naturalna kubiczna funkcja sklejana.*

Dowód. Niech s oznacza naturalną kubiczną funkcję sklejaną spełniającą zadane warunki interpolacyjne, i niech f oznacza dowolną inną funkcję klasy C^2, przyjmującą w węzłach te same wartości. Mamy $f = (f - s) + s$, zatem

$$E(f) = \int_{u_0}^{u_N} \big(f''(x) - s''(x)\big)^2 \,dx + 2\int_{u_0}^{u_N} \big(f''(x) - s''(x)\big)s''(x)\,dx + \int_{u_0}^{u_N} \big(s''(x)\big)^2 \,dx.$$

Obliczymy drugą całkę w powyższym wzorze, całkując przez części:

$$\int_{u_0}^{u_N} \big(f''(x) - s''(x)\big)s''(x)\,dx = \big(f'(x) - s'(x)\big)s''(x)\Big|_{u_0}^{u_N} - \int_{u_0}^{u_N} \big(f'(x) - s'(x)\big)s'''(x)\,dx.$$

Dla naturalnej funkcji sklejanej s jest $s''(u_0) = s''(u_N) = 0$, ponadto w każdym przedziale (u_i, u_{i+1}) pochodna trzeciego rzędu funkcji s jest stała; oznaczmy ją symbolem s_i. Rozpatrywana całka jest więc równa

$$-\sum_{i=0}^{N-1} \int_{u_i}^{u_{i+1}} \big(f'(x) - s'(x)\big)s_i\,dx = -\sum_{i=0}^{N-1} s_i\big(f(x) - s(x)\big)\Big|_{u_i}^{u_{i+1}} = 0,$$

bo dla każdego i jest $f(u_i) = s(u_i)$. Mamy stąd

$$E(f) = \int_{u_0}^{u_N} \big(f''(x) - s''(x)\big)^2 \,dx + \int_{u_0}^{u_N} \big(s''(x)\big)^2 \,dx.$$

Jeśli funkcje f i s mają taką samą pochodną drugiego rzędu w przedziale $[u_0, u_N]$ i są różne, to ich różnica jest wielomianem stopnia 0 lub 1, ale wtedy nie mogą przyjmować tych samych wartości we wszystkich węzłach. Zatem, jeśli przyjmują i są różne, to zachodzi nierówność $E(f) > E(s)$, co pragnęliśmy udowodnić. □

Funkcje B-sklejane

Funkcje sklejane trzeciego stopnia są odpowiednie w większości zastosowań praktycznych, ale w pewnych przypadkach potrzebne są też funkcje sklejane innych stopni. Reprezentacja Hermite'a, opisana wcześniej, jest w miarę wygodna dla funkcji stopnia nieparzystego, ale jeszcze wygodniejsza w zastosowaniach, dla dowolnego stopnia, jest reprezentacja B-sklejana. Funkcję sklejaną stopnia n z węzłami $u_0, \ldots, u_N$ (ogólnie w literaturze jest wiele sposobów numerowania i oznaczania węzłów i funkcji B-sklejanych) reprezentuje się w postaci

$$s(x) = \sum_{i=0}^{N-n-1} d_i N_i^n(x),$$

tj. jako kombinację liniową tzw. funkcji B-sklejanych N_i^n, określonych wzorem Mansfielda-de Boora-Coxa:

$$N_i^0(x) = \begin{cases} 1 & \text{dla } x \in [u_i, u_{i+1}) \\ 0 & \text{w przeciwnym razie} \end{cases}$$

$$N_i^n(x) = \frac{x - u_i}{u_{i+n} - u_i} N_i^{n-1}(x) + \frac{u_{i+n+1} - x}{u_{i+n+1} - u_{i+1}} N_{i+1}^{n-1}(x).$$

Określenie funkcji B-sklejanych dopuszcza węzły o krotności większej niż 1, przy czym dla każdego i powinna zachodzić nierówność $u_i < u_{i+n+1}$, bo w przeciwnym razie funkcja N_i^n byłaby funkcją zerową. Bez dowodu podam najważniejsze własności tych funkcji. Funkcje N_i^n są nieujemne. Funkcja N_i^n jest różna od zera tylko w przedziale $[u_i, u_{i+n+1})$ i między kolejnymi węzłami w tym przedziale jest opisana za pomocą wielomianów stopnia n. Ponadto funkcja ta ma tylko jedno maksimum (stąd nazwa B-sklejane, ang. *B-spline*, od kształtu wykresu, który trochę przypomina przekrój dzwonu). Suma funkcji stopnia n określonych za pomocą węzłów $u_0, \ldots, u_N$ w przedziale $[u_n, u_{N-n})$ jest równa 1. Zwykle w zastosowaniach za dziedzinę funkcji s przyjmuje się ten przedział (ewentualnie domknięty, tj. $[u_n, u_{N-n}]$).

Pochodna funkcji B-sklejanej stopnia n wyraża się wzorem

$$\frac{d}{dx} N_i^n(x) = \frac{n}{u_{i+n} - u_i} N_i^{n-1}(x) - \frac{n}{u_{i+n+1} - u_{i+1}} N_{i+1}^{n-1}(x).$$

Okazuje się (jest kilka dowodów, wszystkie są dosyć żmudne), że w otoczeniu dowolnego węzła o krotności r funkcje B-sklejane (a zatem i każda funkcja s, która jest ich kombinacją liniową) są ciągłe razem z pochodnymi rzędu $1, \ldots, n - r$. Zależnie od potrzeb, możemy więc dobrać stopień i węzły tak, aby otrzymać funkcje sklejane odpowiedniej do zastosowania klasy. Prawie zawsze wystarczy $n < 10$, najczęściej bierze się $n = 3$. W pewnych zastosowaniach przydaje się wzór

$$\int_{\mathbb{R}} N_i^n(x)\,dx = \int_{u_i}^{u_{i+n+1}} N_i^n(x)\,dx = \frac{1}{n}(u_{i+n+1} - u_i).$$

W przedziale $[u_k, u_{k+1}) \subset [u_n, u_{N-n})$ niezerowe wartości przyjmuje $n + 1$ funkcji B-sklejanych stopnia n, mianowicie funkcje $N_{k-n}^n, \ldots, N_k^n$. Wielomiany opisujące te funkcje w przedziale $[u_k, u_{k+1})$ są liniowo niezależne. Wartości tych wielomianów dla ustalonego x można obliczyć za pomocą algorytmu de Boora, który realizuje obliczenie na podstawie wzoru Mansfielda-de Boora-Coxa. Przed wykonaniem podanej niżej procedury należy znaleźć przedział $[u_k, u_{k+1})$, do

którego należy liczba x; można w tym celu zastosować wyszukiwanie binarne, lub, jeśli węzły $u_n, \ldots, u_{N-n}$ są równoodległe, posłużyć się dzieleniem (tj. obliczyć $k = n + \lfloor (x - u_n)/(u_{n+1} - u_n) \rfloor$).

```
b[k] = 1;
for ( j = 1; j <= n; j++ ) {
  β = (u_{k+1} - x)/(u_{k+1} - u_{k-j+1});
  b[k-j] = β * b[k-j+1];
  for ( i = k-j+1; i < k; i++ ) {
    α = 1 - β;
    β = (u_{i+j+1} - x)/(u_{i+j+1} - u_{i+1});
    b[i] = α * b[i] + β * b[i+1];
  }
  b[k] *= (1 - β);
}
```

Liczby $N_{k-n}^n(x), \ldots, N_k^n(x)$ są końcowymi wartościami zmiennych $b[k-n], \ldots, b[k]$. Koszt tego obliczenia jest rzędu n^2. Można wykazać, że algorytm ten ma dobre własności numeryczne (tj. dobrze działa w implementacji z użyciem arytmetyki zmiennopozycyjnej), co jest konsekwencją faktu, że liczby α i β należą do przedziału $[0, 1]$.

Konstrukcja B-sklejanej reprezentacji kubicznej funkcji interpolacyjnej, tj. obliczenie współczynników d_i, również polega na rozwiązaniu układu równań liniowych. W tym przypadku przyjmujemy, że wartości a_i funkcji są zadane w węzłach $u_n, \ldots, u_{N-n}$, tworzących ciąg rosnący. Ponieważ w każdym z tych węzłów tylko trzy funkcje B-sklejane są niezerowe, mamy równania

$$N_{i-3}^3(u_i)d_{i-3} + N_{i-2}^3(u_i)d_{i-2} + N_{i-1}^3(u_i)d_{i-1} = a_i.$$

Wartości funkcji B-sklejanych w węzłach możemy obliczyć za pomocą algorytmu de Boora. Do tych równań (dla $i = 3, \ldots, N-3$) należy dołączyć jeszcze dwa równania opisujące warunki brzegowe (np. prowadzące do otrzymania naturalnej funkcji sklejanej s, tj. spełniającej warunek $s''(u_n) = s''(u_{N-n}) = 0$). Dla dowolnych sensownych warunków brzegowych równania można przekształcić tak, aby otrzymać układ równoważny z macierzą trójdiagonalną.

Twierdzenie Schoenberga-Whitney

Przypomnijmy, że słowo „węzły” było już używane w dwóch znaczeniach. Po pierwsze, w znaczeniu węzły interpolacyjne, czyli punkty, w których zadajemy wartości funkcji. Drugie znaczenie to węzły funkcji sklejanej, czyli punkty

rozgraniczające przedziały, w których funkcja jest (a dokładniej, może być) opisana za pomocą różnych wielomianów. Do tej pory wybieraliśmy węzły interpolacyjne pokrywające się z węzłami funkcji sklejanej, ale możemy dopuścić inny ich wybór. Trzeba jednak wiedzieć, jaki wybór jest dopuszczalny, aby rozwiązanie zadania interpolacyjnego Lagrange'a istniało.

Oznaczmy zatem symbolami $u_0, \dots, u_N$ węzły funkcji sklejanej, a konkretniej niemalejący ciąg węzłów, których użyjemy do określenia funkcji B-sklejanych stopnia n. Liczba tych funkcji to $N - n$. Węzły interpolacyjne oznaczymy symbolami $\nu_0, \dots, \nu_{N-n-1}$. Zatem, liczba warunków interpolacyjnych, które nakładamy, jest równa wymiarowi przestrzeni funkcji sklejanych rozpiętej przez nasze funkcje B-sklejane, dzięki czemu warunki brzegowe są zbędne. Załóżymy, że węzły interpolacyjne są ponumerowane tak, aby tworzyły ciąg rosnący (jest jasne, że węzły interpolacyjne muszą być parami różne).

Twierdzenie Schoenberga-Whitney. *Funkcja sklejana stopnia n, oparta na ciągu węzłów $u_0, \dots, u_N$ i przyjmująca zadane wartości $a_0, \dots, a_{N-n-1}$ odpowiednio w punktach $\nu_0, \dots, \nu_{N-n-1}$, które tworzą ciąg rosnący, istnieje i jest jednoznaczna wtedy i tylko wtedy, gdy $N_i^n(\nu_i) \neq 0$ dla $i = 0, \dots, N-n-1$.*

Dowód tego twierdzenia jest żmudny i polega na wykazaniu że odpowiednia macierz Vandermonde'a (tj. macierz V o współczynnikach $a_{ij} = N_j^n(\nu_i)$) jest nieosobliwa wtedy i tylko wtedy, gdy podany w twierdzeniu warunek jest spełniony. Twierdzenie rozstrzyga problem z punktu widzenia algebry, ale nie gwarantuje, że macierz V jest dobrze uwarunkowana. Aby tak było, dla każdego $i \in \{0, \dots, N-n-1\}$ węzeł interpolacyjny ν_i powinien leżeć w pobliżu punktu, w którym odpowiadająca mu funkcja B-sklejana N_i^n osiąga wartość maksymalną.

Przypuśćmy, że dane są węzły interpolacyjne, $\nu_0, \dots, \nu_{N-n-1}$, ustawione w ciąg rosnący. Jeśli n jest nieparzyste, to dobrym (ale *nie jedynym dobrym*) wyborem jest przyjęcie węzłów funkcji sklejanej $u_0 = \dots = u_n = \nu_0$, oraz $u_i = \nu_{i-(n+1)/2}$ dla $i = n+1, \dots, N-n-1$, i $u_{N-n} = \dots = u_N = \nu_{N-n-1}$.

Dla parzystego n można wybrać $u_0 = \dots = u_n = \nu_0$, i $u_i = (\nu_{i-n/2-1} + \nu_{i-n/2})/2$ dla $i = n+1, \dots, N-n-1$, oraz $u_{N-n} = \dots = u_N = \nu_{N-n-1}$.

Wspomniana wyżej macierz V jest wstęgowa, a dokładniej, ma w każdym wierszu co najwyżej $n+1$ niezerowych współczynników. Dzięki temu znalezienie sklejanej funkcji interpolacyjnej może być bardzo mało kosztowne (koszt eliminacji Gaussa w tym przypadku to $O(Nn^2)$ operacji).

Zadania i problemy

1. Napisz układ równań, którego rozwiązaniem są liczby $b_0, \dots, b_N$, będące wartościami pochodnej naturalnej kubicznej sklejanej funkcji interpolacyjnej w węzłach $0, \dots, N$. Rozwiąż ten układ dla $N = 4$ i danych wartości funkcji $a_0 = \dots = a_3 = 0$, $a_4 = 1$.

   Podaj oszacowanie wskaźnika uwarunkowania macierzy tego układu w normie drugiej, na podstawie twierdzenia Gerszgorina.
2. Napisz układ równań dla okresowej kubicznej funkcji interpolacyjnej (warunki interpolacyjne, tj. wartości funkcji, zadane są w węzłach funkcji sklejanej), z węzłami równoodległymi $u_i = i$ dla $i = 0, \dots, 5$, i rozwiąż ten układ.
3. Napisz równanie opisujące warunek *not a knot* dla kubicznej interpolacyjnej funkcji sklejanej klasy C^2; węzeł interpolacyjny u_1 ma nie być węzłem funkcji sklejanej.
4. Wyprowadź alternatywną postać równania umożliwiającego skonstruowanie kubicznej sklejanej funkcji interpolacyjnej klasy C^2, z niewiadomymi wartościami c_{i-1}, c_i, c_{i+1} pochodnej drugiego rzędu odpowiednio w węzłach u_{i-1}, u_i i u_{i+1}.
   Wskazówka: Pochodne drugiego rzędu wielomianów p_{i-1} i p_i, opisujących funkcję sklejaną w przedziałach $[u_{i-1}, u_i]$ oraz $[u_i, u_{i+1}]$, są wielomianami pierwszego stopnia, interpolującymi odpowiednio liczby c_{i-1}, c_i oraz c_i, c_{i+1}. Znajdź te wielomiany i scałkuj dwukrotnie, dobierając stałe całkowania tak, aby otrzymane wyrażenia opisywały wielomiany p_{i-1} i p_i i w szczególności przyjmowały w węźle u_i (daną) wartość a_i, a ich pochodne pierwszego rzędu przyjmowały (niewiadomą) wartość b_i. Następnie oblicz $a_{i-1} = p_{i-1}(u_{i-1})$ oraz $a_{i+1} = p_i(u_{i+1})$ i stąd otrzymaj dwa wyrażenia na b_i. Postaw znak równości między tymi wyrażeniami i uporządkuj otrzymane równanie.
5. Znajdź wielomiany stopnia 5, będące rozwiązaniami sześciu zadań interpolacyjnych Hermite'a, dla dwóch węzłów trzykrotnych 0 i 1, analogicznych do zadań, których rozwiązaniami są wielomiany trzeciego stopnia z wykładu. Podaj sposób określenia interpolacyjnej funkcji sklejanej stopnia 5 dla węzłów $u_0, \dots, u_N$ przy użyciu tych wielomianów.
   Wskazówka: Wystarczy znaleźć trzy wielomiany przez rozwiązanie zadań interpolacyjnych. Pozostałe trzy są do nich symetryczne, tj. $H_{10}(x) = H_{00}(1-x)$, $H_{11}(x) = -H_{01}(1-x)$, $H_{12}(x) = H_{02}(1-x)$.
6. Znajdź wielomiany opisujące funkcje B-sklejane N_i^2 oraz N_i^3, określone dla węzłów będących kolejnymi liczbami naturalnymi.
   Wskazówka: Dla każdego $i \in \mathbb{Z}$ oraz $x \in \mathbb{R}$ jest $N_i^n(x) = N_0^n(x - i)$, wystarczy zatem znaleźć wielomiany opisujące tylko jedną funkcję B-sklejaną ustalonego stopnia.

7. Napisz układ równań, którego rozwiązaniem jest wektor współczynników $d_0, \ldots, d_{N-4}$ naturalnej kubicznej sklejanej funkcji interpolacyjnej w bazie $\{N_0^3, \ldots, N_{N-4}^3\}$, przy czym funkcje B-sklejane, z których składa się ta baza, są określone za pomocą ciągu węzłów równoodległych: $u_i = i$ dla $i = 0, \ldots, N$ (warunki interpolacyjne, tj. wartości funkcji, są określone w węzłach $3, \ldots, N-3$).
8. Napisz układ równań, którego rozwiązaniem jest wektor współczynników (w bazie B-sklejanej) funkcji sklejanej stopnia 2, przy czym węzłami funkcji sklejanej są liczby naturalne $0, 1, \ldots, N$, a węzłami interpolacyjnymi są liczby $2, 2\frac{1}{2}, 3\frac{1}{2}, \ldots, N-2\frac{1}{2}, N-2$ (z wyjątkiem węzłów skrajnych odległości między kolejnymi węzłami są równe 1).
9. Na podstawie wzoru Mansfielda-de Boora-Coxa wyprowadź wzór, umożliwiający obliczenie wartości wielomianu $p_k(x)$, opisującego funkcję sklejaną $s(x) = \sum_{i=0}^{N-n-1} d_i N_i^n(x)$ w przedziale $[u_k, u_{k+1})$ (dla $x \in [u_k, u_{k+1})$ jest $p_k(x) = \sum_{i=k-n}^{k} d_i N_i^n(x)$), za pomocą funkcji B-sklejanych stopnia $n-1$. Stosując ten wzór rekurencyjnie, można obliczyć wartość funkcji $s(x)$ na postawie danych liczb x i $d_{k-n}, \ldots, d_k$. Odpowiedni algorytm ma koszt $O(n^2)$ i jest numerycznie poprawny. Zapisz ten algorytm w postaci kodu w C.
10. Na podstawie wzoru opisującego pochodną funkcji B-sklejanej wyprowadź wzór opisujący pochodną funkcji sklejanej $s(x) = \sum_{i=0}^{N-n-1} d_i N_i^n(x)$.

Interpolacja trygonometryczna

Def. Wielomian trygonometryczny stopnia n jest to funkcja o postaci

$$w(t) = a_0 + \sum_{k=1}^{n} a_k \cos kt + b_k \sin kt. \qquad (*)$$

Wielomiany trygonometryczne występują w różnych zastosowaniach, zwłaszcza takich, w których pojawiają się funkcje okresowe. Często powstają z obcięcia szeregów Fouriera, tj. szeregów opisanych wzorem podobnym do wzoru (*), w którym zamiast n jest ∞.

Wzór (*) opisuje funkcję o okresie 2π. Aby otrzymać funkcję o dowolnym okresie T, można dokonać zamiany zmiennych. Będziemy mieli wtedy

$$f(x) = a_0 + \sum_{k=1}^{n} a_k \cos kt + b_k \sin kt,$$

gdzie $t = 2\pi(x - x_0)/T$, dla dowolnie wybranego x_0.

Trygonometryczne zadanie interpolacyjne Lagrange'a polega na znalezieniu wielomianu trygonometrycznego stopnia n, którego wartości są określone w $2n+1$ węzłach interpolacyjnych, $x_j \in \mathbb{R}$, gdzie $j = 0, \dots, 2n$. Ponieważ wielomiany trygonometryczne są funkcjami okresowymi, jest oczywiste, że warunkiem koniecznym, aby zadanie miało rozwiązanie dla dowolnych zadanych wartości funkcji, jest $(x_j - x_k)/T \notin \mathbb{Z}$ dla $j \neq k$. Jest to też warunek dostateczny.

Dowód (a właściwie szkic). Wystarczy to udowodnić dla przypadku szczególnego $T = 2\pi$. Dla węzłów interpolacyjnych x_j i wartości funkcji f_j podanych dla tych węzłów, określamy liczby $z_j = e^{ix_j}$ oraz $h_j = z_j^n f_j$. Jeśli węzły $x_0, \dots, x_{2n}$ spełniają rozważany warunek, to liczby z_j są parami różne. Jak wiemy, zadanie interpolacyjne Lagrange'a, tj. wyznaczenie wielomianu $h(z)$ stopnia co najwyżej $2n$, takiego że $h(z_j) = h_j$ dla $j = 0, \dots, 2n$, ma rozwiązanie[10]. Możemy je przedstawić w bazie potęgowej w taki sposób: $h(z) = \sum_{k=0}^{2n} c_{k-n} z^k$.

Zespolona funkcja wymierna

$$g(z) \stackrel{\text{def}}{=} \sum_{k=-n}^{n} c_k z^k$$

w węzłach z_j przyjmuje wartości f_j, i jest tylko jedna taka funkcja o tej postaci (bo liczby $c_{-n}, \dots, c_n$ są określone jednoznacznie przez warunki interpolacyjne nałożone na wielomian h).

[10] Pod tym względem zadanie interpolacji wielomianowej dla zespolonych węzłów i wartości funkcji nie różni się od przypadku rzeczywistego.

Niech $\hat{g}(z) \stackrel{\text{def}}{=} \overline{g(z)}$. Dla każdego $z \in \mathbb{C}$ takiego że $|z| = 1$, w tym dla każdego z_j, jest $\overline{z} = \frac{1}{z}$ i stąd

$$\hat{g}(z) = \overline{g(z)} = \sum_{k=-n}^{n} \overline{c}_k \overline{z}^k = \sum_{k=-n}^{n} \overline{c}_k z^{-k} = \sum_{k=-n}^{n} \overline{c}_{-k} z^k.$$

Ponieważ liczby f_j są rzeczywiste, zachodzą równości $\hat{g}(z_j) = f_j = g(z_j)$ dla każdego j. Spełniająca te warunki interpolacyjne funkcja $\hat{g}(z)$ też jest tylko jedna (tj. liczby $\overline{c}_{-n}, \dots, \overline{c}_n$ są jednoznacznie określone przez te warunki), ale to oznacza, że w zbiorze $\{ z \in \mathbb{C} \colon |z| = 1 \}$ funkcje $g(z)$ i $\hat{g}(z)$ są identyczne, a stąd wynika, że $c_{-k} = \overline{c}_k$ dla $k = -n, \dots, n$.

Zatem, funkcja $w(t) = g(e^{it}) = \overline{g(e^{it})}$ ma dla każdego $t \in \mathbb{R}$ wartość rzeczywistą i spełnia warunki $w(x_j) = f_j$ dla $j = 0, \dots, 2n$. Bezpośrednim rachunkiem możemy sprawdzić, że jest to wielomian trygonometryczny (*), o współczynnikach $a_0 = \operatorname{Re} c_0$ (jest $\operatorname{Im} c_0 = 0$), oraz $a_k = 2 \operatorname{Re} c_k$ i $b_k = -2 \operatorname{Im} c_k$ dla $k > 0$. □

W praktycznych zastosowaniach najczęściej wybiera się węzły $x_0, \dots, x_{2n}$, które dzielą przedział $[x_0, x_0 + T)$ (o długości okresu T interpolowanej funkcji) na części o jednakowych długościach. Zamiast bezpośrednio rozwiązywać zadania interpolacji trygonometrycznej, zwykle sprowadza się problem do konstrukcji zespolonego wielomianu algebraicznego, w sposób podobny do użytego w powyższym dowodzie.

Dyskretna transformata Fouriera

Def. Dyskretną transformatą Fouriera ciągu zespolonego $(a_k)_{k \in \mathbb{Z}}$ o okresie n (tj. spełniającego warunek $a_{k+n} = a_k$ dla każdego $k \in \mathbb{Z}$) jest ciąg zespolony $(b_j)_{j \in \mathbb{Z}}$ określony wzorem

$$b_j = \sum_{k=0}^{n-1} a_k e^{-2\pi i jk/n}.$$

Odwrotną dyskretną transformatą Fouriera ciągu $(a_k)_{k \in \mathbb{Z}}$ nazywamy ciąg $(c_j)_{j \in \mathbb{Z}}$ określony wzorem

$$c_j = \frac{1}{n} \sum_{k=0}^{n-1} a_k e^{2\pi i jk/n}.$$

Ciągi $(b_j)_{j \in \mathbb{Z}}$ i $(c_j)_{j \in \mathbb{Z}}$ są okresowe o okresie n. Oba przekształcenia zdefiniowane wyżej są liniowe i każde z nich jest jest odwrotnością tego drugiego, co uzasadnia nazwę. Mamy bowiem

$$d_l = \frac{1}{n} \sum_{j=0}^{n-1} \Big(\sum_{k=0}^{n-1} a_k e^{-2\pi i jk/n} \Big) e^{2\pi i lj/n} = \frac{1}{n} \sum_{k=0}^{n-1} a_k \sum_{j=0}^{n-1} e^{2\pi i (l-k) j/n}.$$

Jeśli $k = l$, to $e^{2\pi i(l-k)j/n} = e^0 = 1$, zaś jeśli $k \neq l$, to liczby $e^{2\pi i(l-k)j/n}$ są pierwiastkami zespolonymi z 1; ich suma dla $j = \{0, \ldots, n-1\}$ jest równa 0. Stąd wynika, że $d_l = a_l$.

Interpolacja trygonometryczna i dyskretna transformata Fouriera występuje w wielu problemach związanych z analizą, transmisją i przetwarzaniem sygnałów (np. akustycznych lub obrazów), a także w rozwiązywaniu równań różniczkowych.

Algorytm FFT

Możemy zauważyć, że ciąg $(b_j)_{j\in\mathbb{Z}}$, który jest transformatą ciągu $(a_k)_{k\in\mathbb{Z}}$, składa się z wartości wielomianu stopnia $n-1$ o współczynnikach $a_0, \ldots, a_{n-1}$ w punktach $e^{-2\pi ij/n}$, $j = 0, \ldots, n-1$. Dyskretną transformatę Fouriera można wyznaczyć za pomocą schematu Hornera; wyznaczenie pełnej transformaty kosztowałoby wtedy $n^2 - n$ mnożeń i dodawań zespolonych. Okazuje się, że można to zadanie rozwiązać kosztem $\Theta(n(p_1 + \cdots + p_r))$ działań, gdzie $p_1, \ldots, p_r$ są liczbami pierwszymi, takimi że $n = p_1 \cdot \ldots \cdot p_r$. Odkrycia tego dokonali w 1952 r. Cooley i Tukey.

Zauważmy, że ciąg okresowy $(a_k)_{k\in\mathbb{Z}}$ o okresie 1 jest ciągiem stałym i jest on identyczny ze swoją dyskretną transformatą Fouriera. Dalej, przypuśćmy, że liczba n jest podzielna przez $p > 1$. Oznaczmy $w_j = e^{-2\pi ij/n}$. Wtedy wzór definiujący dyskretną transformatę Fouriera można przedstawić w postaci

$$b_j = \sum_{k=0}^{n/p-1} a_{pk} w_j^{pk} + w_j \sum_{k=0}^{n/p-1} a_{pk+1} w_j^{pk} + \cdots + w_j^{p-1} \sum_{k=0}^{n/p-1} a_{pk+p-1} w_j^{pk}.$$

Podzieliliśmy tu ciąg $p_0, \ldots, p_{n-1}$ na podciągi n/p-elementowe, wybierając do każdego z nich co p-ty element. Możemy dalej zauważyć, że sumy mnożone przez kolejne potęgi liczby w_j są wyrażeniami opisującymi transformaty tych podciągów, a dokładniej ich obustronnie nieskończonych rozszerzeń o okresie n/p. Obliczenie dyskretnej transformaty Fouriera dla ciągu o okresie n może być zatem wykonane przez następujący algorytm rekurencyjny:

- Jeśli $n = 1$, to przyjmij $b_0 = a_0$ (dla $n = 1$ przekształceniu poddajemy ciąg stały, którego obrazem jest ten sam ciąg).
- Jeśli n jest liczbą pierwszą, to zastosuj wzór podany jako definicja dyskretnej transformaty Fouriera i użyj schematu Hornera.
- Jeśli $n > 1$ jest podzielne przez liczbę pierwszą $p < n$, to podziel ciąg na p podciągów (zgodnie z opisem wyżej), oblicz transformaty tych podciągów i „scal" je, stosując wzór podany wyżej i schemat Hornera.

Wzór opisujący transformatę odwrotną może być przekształcony podobnie; zamiast $w_j = (\cos\frac{2\pi j}{n}, -\sin\frac{2\pi j}{n})$ występuje w nim liczba $\overline{w_j} = (\cos\frac{2\pi j}{n}, \sin\frac{2\pi j}{n})$. Możemy zatem użyć takiego samego algorytmu, zostawiając mnożenie wyniku działania procedury rekurencyjnej przez czynnik $\frac{1}{n}$ na sam koniec. Koszt algorytmu w istotny sposób zależy od możliwości rozłożenia liczby n na czynniki.

Algorytm jest najbardziej efektywny, jeśli liczba n jest potęgą liczby 2 i często określenie FFT (od angielskiego *Fast Fourier Transform*) dotyczy takiego wariantu algorytmu. Zbadamy go dokładniej. Dla parzystej liczby n transformatę otrzymamy przez „scalenie" transformat dwóch podciągów, złożonych odpowiednio z elementów parzystych i nieparzystych ciągu danego. Transformaty te oznaczymy symbolami $(p_j)_{j\in\mathbb{Z}}$ i $(q_j)_{j\in\mathbb{Z}}$. Przypomnijmy, że transformaty te są ciągami obustronnie nieskończonymi, o okresie $n/2$, reprezentowanymi przez podciągi $p_0, \ldots, p_{n/2-1}$ i $q_0, \ldots, q_{n/2-1}$. Możemy napisać

$$b_j = \sum_{k=0}^{n/2-1} a_{2k} w_j^{2k} + w_j \sum_{k=0}^{n/2-1} a_{2k+1} w_j^{2k} = p_j + w_j q_j.$$

Podstawiając $j + n/2$ w miejsce j, i biorąc pod uwagę, że $w_{j+n/2} = e^{-2\pi i(j+n/2)/n} = e^{-2\pi ij/n} e^{-2\pi in/(2n)} = -w_j$ oraz $w_{j+n/2}^{2k} = w_j^{2k}$, dostajemy

$$b_{j+n/2} = \sum_{k=0}^{n/2-1} a_{2k} w_{j+n/2}^{2k} + w_{j+n/2} \sum_{k=0}^{n/2-1} a_{2k+1} w_{j+n/2}^{2k} = p_j - w_j q_j.$$

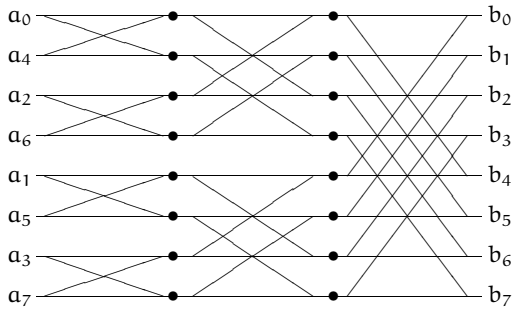
Implementacja algorytmu FFT w postaci procedury rekurencyjnej jest następująca:

```
void rFFT ( int n, complex a[] )
{
  complex *p, *q, u, w, t; int j;

  if ( n > 1 ) {
    p = malloc ( n/2*sizeof(complex) );
    q = &p[n/2];
    for ( j = 0; j < n/2; j++ ) {
      p[j] = a[2*j];
      q[j] = a[2*j+1];
    }
    rFFT ( n/2, p );
    rFFT ( n/2, q );
    u = 1;
    w = e^{-2πi/n};
```

```
    for ( j = 0; j < n/2; j++ ) {
      t = u*q[j];
      a[j] = p[j] + t;
      a[j+n/2] = p[j] - t;
      u = u*w;
    }
    free ( p );
  }
} /*rFFT*/
```

Oglądając tę implementację, widzimy, że choć procedura umieszcza transformatę w tej samej tablicy, w której są początkowo dane liczby $a_0, \ldots, a_{n-1}$, potrzebuje ona sporo pamięci dodatkowej (w rzeczywistości potrzeba dodatkowych tablic o sumarycznej długości $2n$). Można jednak zaprojektować taką implementację, która wszystkie obliczenia wykonuje „w miejscu", tj. która oprócz tablicy z danym ciągiem, który należy zastąpić przez jego transformatę, potrzebuje tylko niewielkiej ustalonej liczby zmiennych prostych. Aby otrzymać taką procedurę, nierekurencyjną i dodatkowo oszczędzającą pewne działania, przyjrzymy się „przepływowi danych", to znaczy zbadamy, od których współczynników zależą transformaty obliczane „po drodze". Dla $n = 8$ „przepływ danych" jest przedstawiony na rysunku (najlepiej go oglądać od prawej do lewej strony).



Krawędzie łączą dane z wynikami, tj. każda liczba (z wyjątkiem danych) jest obliczana na podstawie liczb znajdujących się w kolumnie na lewo od niej, połączonych z nią kreskami. Widzimy, że w każdym przypadku obliczenie polega na zastąpieniu pary liczb przez inną parę, obliczoną tylko na jej podstawie (i dana para liczb nie jest do niczego innego potrzebna). Jeśli zatem ustawimy dane wejściowe w odpowiedniej kolejności, to można całe obliczenie wykonać bez potrzeby rezerwowania dodatkowej tablicy.

Ostatnia transformata powstaje z transformat podciągów „parzystego" i „nieparzystego". Każda z tych dwóch transformat jest obliczana na podstawie transformat „parzystego" i „nieparzystego" podciągu odpowiedniego podciągu itd.; zatem ogólna reguła porządkowania danych wejściowych polega na ustawieniu ich *w kolejności odwróconych bitów*. Jeśli indeks j danego współczynnika a_j przedstawimy w układzie dwójkowym, przy użyciu $l + 1 = (\log_2 n) + 1$ cyfr dwójkowych (bitów), to indeks miejsca w tablicy, na którym ma się on znaleźć, otrzymamy wypisując te bity w odwrotnej kolejności. Procedura FFT, która realizuje to obliczenie, ma postać

```
void FFT ( int n, complex a[] )
{
  complex t, u, w;
  int i, j, k, l, m, p;

  l = log2n; m = n/2;
       /* przestawianie danych w tablicy */
  for ( i = 1, j = m; i < n-1; i++ ) {
    if ( i < j ) przestaw ( &a[i], &a[j] );
    k = m;
    while ( k <= j ) { j -= k; k /= 2; }
    j += k;
  }
       /* obliczanie transformaty */
  for ( k = 1; k <= l; k++ ) {
    m = 2^k; p = m/2;
    u = 1; w = e^(-πi/p);
    for ( j = 0; j < p; j++ ) {
      i = j;
      do {
        t = a[i+p]*u;
        a[i+p] = a[i]-t; a[i] = a[i]+t;
        i += m;
      } while ( i <= n );
      u *= w;
    }
  }
} /*FFT*/
```

Algorytm ten opublikowali w 1965 r. Cooley, Lewis i Welch. Pierwsza pętla, `for ( i = ... ) ...`, dokonuje przestawienia elementów w tablicy zgodnie

z kolejnością odwróconych bitów. Kolejne przebiegi drugiej pętli, `for ( k = ... ) ...`, mają na celu obliczenie $n/2$ transformat podciągów o okresie 2, $n/4$ transformat podciągów o okresie 4, itd. Liczba $e^{-2\pi i/n}$ (wartość zmiennej `w`) i jej potęgi, czyli liczby w_j (kolejne wartości zmiennej `u`) są obliczane tylko raz dla wszystkich transformat podciągów o tym samym okresie. W każdym przebiegu pętli `for ( j = ... ) ...` obliczane są pary współczynników o numerach j oraz $j+p$ we wszystkich transformatach podciągów o okresie $m = 2p$, ponieważ pętla najbardziej wewnętrzna (`do ... while`) przebiega przez wszystkie te transformaty.

Można udowodnić, że algorytm FFT, także w wersji ogólnej (dla dowolnego n), jest numerycznie stabilny, tj. istnieje stała K (zależna od n), taka że współczynniki $\tilde{b}_j$ obliczone przy użyciu arytmetyki zmiennopozycyjnej przybliżają dokładne współczynniki b_j dyskretnej transformaty Fouriera z błędem spełniającym nierówność

$$\max_j |\tilde{b}_j - b_j| \leqslant K\nu \max_j |b_j|,$$

gdzie $\nu = 2^{-t}$. Dla liczby n będącej potęgą 2 można przyjąć

$$K = \left(\sqrt{2}\log_2 n + (\log_2 n - 1)(3 + 2\varepsilon)\right)\sqrt{n},$$

gdzie ε jest oszacowaniem błędu bezwzględnego obliczonych kosinusów i sinusów, tj. części rzeczywistych i urojonych liczb w_j.

Szybkie mnożenie wielomianów

Zajmiemy się następującym zadaniem: dane są współczynniki $a_0, \ldots, a_n$ i $b_0, \ldots, b_m$ wielomianów $a(x) = \sum_{k=0}^{n} a_k x^k$ i $b(x) = \sum_{k=0}^{m} b_k x^k$. Należy obliczyć współczynniki $c_0, \ldots, c_{n+m}$ wielomianu $c(x) = \sum_{k=0}^{n+m} c_k x^k = a(x)b(x)$. „Zwykły” algorytm mnożenia wielomianów można zrealizować za pomocą podprogramu

```
for ( k = 0; k <= n+m; k++ ) c[k] = 0;
for ( i = 0; i <= n; i++ )
  for ( j = 0; j <= m; j++ ) c[i+j] += a[i]*b[j];
```

Operacją dominującą w tym algorytmie jest mnożenie współczynników; operacji tych należy wykonać $(n+1)(m+1)$; jeśli $m \approx n$, to złożoność obliczeniowa ma rząd $\Theta(n^2)$, choć zarówno danych, jak i wyników jest $\Theta(n)$.

Alternatywny sposób rozwiązywania tego zadania polega na wybraniu liczb $x_0, \ldots, x_{n+m}$, obliczeniu wartości wielomianów a i b, obliczeniu wartości $c(x_j) = a(x_j)b(x_j)$ wielomianu c i znalezieniu jego współczynników w bazie

potęgowej, przez rozwiązanie zadania interpolacyjnego Lagrange'a. Mnożenie wielomianów — w postaci mnożenia ich wartości w wybranych punktach — wymaga wykonania tylko $n + m + 1$ mnożeń. Trzeba tylko umieć szybko obliczyć wartości wielomianów a i b i szybko rozwiązać zadanie interpolacyjne.

Do tego celu możemy użyć algorytmu FFT; jeśli przyjmiemy, że $x_j = e^{-2\pi ij/N}$, gdzie liczba N jest najmniejszą całkowitą potęgą liczby 2 większą niż $n + m$, to ciąg wartości wielomianu a w tych punktach jest dyskretną transformatą Fouriera ciągu współczynników $a_0, \ldots, a_n, 0, \ldots, 0$ o długości (a raczej okresie) N. Mając wartości wielomianu c w punktach x_j, możemy obliczyć jego współczynniki w bazie potęgowej, wyznaczając odwrotną dyskretną transformatę Fouriera. Całe to obliczenie jest wykonalne za pomocą $\Theta(N \log N) = \Theta((n+m)\log(n+m))$ działań zmiennopozycyjnych.

Zadania i problemy

1. Udowodnij, wykonując odpowiedni rachunek, że jeśli $c_{-k} = \overline{c}_k$ dla $k = -n, \ldots, n$, to dla każdego $t \in \mathbb{R}$

$$\sum_{k=-n}^{n} c_k e^{it} = a_0 + \sum_{k=1}^{n} a_k \cos kt + b_k \sin kt,$$

   gdzie $c_0 = (a_0, 0)$ oraz $c_k = \frac{1}{2}(a_k, -b_k)$ dla $k = 1, \ldots, n$.

2. Znajdź złożoność obliczeniową algorytmu FFT dla ciągu o długości $n = p^k$, gdzie p jest liczbą pierwszą, a k jest liczbą naturalną.

   Czy istnieje monotoniczna funkcja $f(n)$, taka że złożoność algorytmu FFT jest równa $\Theta(f(n))$? Odpowiedź uzasadnij.

3. Dwuwymiarowa dyskretna transformata Fouriera układu liczb $a_{jk} \in \mathbb{C}$, gdzie $j, k \in \mathbb{Z}$, takiego że istnieją liczby naturalne n, m, takie że dla każdego $j, k \in \mathbb{Z}$ jest $a_{jk} = a_{j+n,k} = a_{j,k+m}$, jest układem liczb

$$b_{pq} = \sum_{j=0}^{n} \sum_{k=0}^{m} a_{jk} e^{-2\pi ijp/n - 2\pi ikq/m}.$$

   Znajdź wzór opisujący dwuwymiarową dyskretną transformatę odwrotną.

   Podaj algorytm obliczania dwuwymiarowej dyskretnej transformaty Fouriera za pomocą algorytmu FFT (dla transformaty jednowymiarowej) i podaj złożoność tego algorytmu w zależności od liczb n i m.

Aproksymacja funkcji

Niech f oznacza funkcję określoną w przedziale $[a, b]$. Definicja tej funkcji może nie być wygodnym algorytmem obliczania wartości tej funkcji (np. funkcja f może być granicą nieskończonego ciągu), ewentualnie możemy mieć tylko „czarną skrzynkę" w postaci podprogramu obliczającego wartość funkcji f, przy czym koszt „sięgnięcia do tej skrzynki" może być bardzo duży, jeśli na przykład obliczenie wartości funkcji f w punkcie x polega na przeprowadzeniu eksperymentu fizycznego z parametrem x i dokonaniu pomiaru.

Zadanie aproksymacji polega na znalezieniu w ustalonej przestrzeni liniowej V, której elementy są funkcjami określonymi na przedziale $[a, b]$, funkcji g przybliżającej funkcję f (która w ogólności *nie jest* elementem przestrzeni V). Oczywiście, przestrzeń V wybieramy tak, aby koszt obliczania wartości należących do niej funkcji był mały, bo w zamierzeniu będziemy wielokrotnie obliczać wartości funkcji g, której chcemy używać zamiast f w jakimś celu.

Aby funkcja g mogła skutecznie „udawać" funkcję f, *musi* być skonstruowana w oparciu o dodatkową wiedzę na temat własności funkcji f i na temat *zamierzonej jakości aproksymacji*. Jeśli na przykład funkcja f ma ciągłą pochodną, to możemy chcieć, aby nie tylko wartości funkcji g były bliskie wartości funkcji f, ale także aby pochodna funkcji g przybliżała pochodną funkcji f (samo przybliżanie wartości funkcji f tego *nie zapewnia*). Przyjmiemy, że funkcja f jest elementem pewnej przestrzeni liniowej U, na przykład przestrzeni funkcji klasy $C^k[a, b]$ dla pewnego k. Będziemy rozpatrywać algorytmy dobierania funkcji z przestrzeni $V \subset U$. Algorytm będzie skuteczny, jeśli konkretna funkcja f, której przybliżenie chcemy znaleźć, jest istotnie elementem przestrzeni U (uwaga: to jest nietrywialne, jeśli funkcję f znamy tylko na podstawie pomiarów).

Błąd aproksymacji będziemy mierzyć za pomocą pewnej normy określonej w przestrzeni U. Zwykle normy określa się za pomocą całek i bardzo często bierze się normy Höldera; wtedy miarą błędu przybliżenia funkcji f przez g jest wyrażenie (dla ustalonego $p \geqslant 1$)

$$\|f - g\|_p = \left(\int_a^b |f(x) - g(x)|^p \, dx \right)^{1/p}.$$

Dwa szczególnie ważne przypadki to $p = 2$ (mówimy wtedy o aproksymacji średniokwadratowej) oraz przypadek graniczny dla $p \to \infty$, gdy błąd jest określony wzorem

$$\|f - g\|_\infty = \max_{x \in [a,b]} |f(x) - g(x)|.$$

Ten przypadek nazywa się aproksymacją jednostajną.

Gdybyśmy byli zainteresowani także przybliżaniem pochodnej funkcji f, to mierzylibyśmy błąd innymi sposobami, np. obliczając wyrażenie

$$\max\{\|f - g\|_\infty, c\|f' - g'\|_\infty\},$$

z jakoś wybraną zawczasu stałą $c > 0$.

Aproksymacja jednostajna

Z analizy znamy twierdzenie aproksymacyjne Weierstrassa: *Jeśli funkcja f jest ciągła na przedziale $[a, b]$, to dla każdego $\varepsilon > 0$ istnieje wielomian p_n pewnego stopnia n, taki że $\|f - p_n\|_\infty \leqslant \varepsilon$.*

Twierdzenie Weierstrassa ma konstruktywny dowód (Bernstein, 1912 r.), ale konstrukcja użyta w tym dowodzie nie nadaje się do praktycznego stosowania, bo nawet dla „łatwych" funkcji i niezbyt małego ε wynikające z dowodu oszacowanie liczby n może być rzędu wielu tysięcy, podczas gdy wystarczy n mniejsze niż 10. Jedną z przyczyn tak słabych wyników konstrukcji jest to, że poza ciągłością o funkcji f niczego się nie zakłada.

Jeśli funkcja f jest klasy $C^{n+1}[a, b]$, to zadanie aproksymacji możemy rozwiązać przez skonstruowanie wielomianu interpolacyjnego Lagrange'a lub Hermite'a. W tym celu wybieramy węzły interpolacyjne $x_i \in [a, b]$ dla $i = 0, \ldots, n$, obliczamy wartości funkcji f (i ewentualnie pochodnych, jeśli są krotne węzły) i stosujemy algorytm różnic dzielonych. Dla tak skonstruowanego wielomianu $h_n(x)$, na podstawie wzoru opisującego resztę, mamy

$$\|f - h_n\|_\infty = \max_{x \in [a,b]} \frac{|f^{(n+1)}(\xi(x))|}{(n+1)!} |p_{n+1}(x)|,$$

gdzie $p_{n+1}(x) = (x - x_0) \cdot \ldots \cdot (x - x_n)$. Mamy zatem problem, jak dobrać węzły, aby opisany powyższym wzorem błąd aproksymacji był jak najmniejszy.

Wielomiany i węzły Czebyszewa

Możemy ustalić $\varepsilon > 0$, a następnie starać się dobrać węzły interpolacyjne w przedziale $[a, b]$ w dowolny sposób zapewniający, że błąd aproksymacji jest mniejszy niż ε. Jeśli się to uda, to nie przejmujemy się tym, że inny wybór mógłby dać jeszcze mniejszy błąd. Jeśli $\max_{x \in [a,b]} |f^{(n+1)}(x)| \leqslant M_{n+1}$, to

$$\|f - h_n\|_\infty \leqslant \frac{M_{n+1}}{(n+1)!} \|p_{n+1}\|_\infty. \qquad (*)$$

Możemy wybierać węzły tak, aby zminimalizować czynnik $\|p_{n+1}\|_\infty$. Aby to zrobić, zbadamy tzw. wielomiany Czebyszewa, zdefiniowane za pomocą wzorów

$$T_0(u) = 1,$$
$$T_1(u) = u,$$
$$T_k(u) = 2uT_{k-1}(u) - T_{k-2}(u) \quad \text{dla } k > 1.$$

Jest jasne, że funkcja $T_k(u)$ jest wielomianem stopnia k. Wzór rekurencyjny dla $k > 1$ to tak zwana formuła trójczłonowa, która umożliwia m.in. numeryczne obliczanie wartości tych wielomianów i ich kombinacji liniowych dla ustalonego u. Wielomiany Czebyszewa można określić także innymi sposobami, z których nam się przyda taki:

$$T_k(u) = \cos(k \arccos u) \quad \text{dla } u \in [-1, 1].$$

Sprawdźmy, że to jest równoważna definicja: oznaczmy $u = \cos t$. Wtedy $T_0(u) = \cos 0 = 1$ oraz $T_1(u) = \cos t = u$, zaś dla $k > 1$, podstawiając $\alpha = kt$ i $\beta = (k-2)t$ do tożsamości trygonometrycznej

$$\cos\alpha + \cos\beta = 2\cos\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2},$$

otrzymujemy równość

$$\cos kt = 2\cos(k-1)t\cos t - \cos(k-2)t,$$

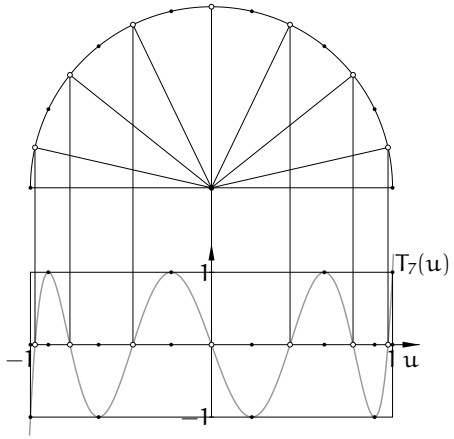
czyli formułę trójczłonową.

Na podstawie trygonometrycznego wzoru określającego wielomiany Czebyszewa możemy stwierdzić, że k miejsc zerowych wielomianu T_k (czyli wszystkie) znajduje się w przedziale $[-1, 1]$, mianowicie są nimi liczby

$$z_j = \cos\frac{2j+1}{2k}\pi \quad \text{dla } j = 0, \ldots, k-1,$$

a ponadto wielomian T_k w przedziale $[-1, 1]$ przyjmuje wartości ekstremalne, na przemian $+1$ i -1, w punktach

$$y_j = \cos\frac{j}{k}\pi \quad \text{dla } j = 0, \ldots, k.$$



Mając ustalony przedział $[a, b]$ oraz liczbę $k > 0$, możemy określić wielomian

$$q_k(x) = \frac{(b-a)^k}{2^{2k-1}}T_k(u),$$

gdzie $u = 2(x-a)/(b-a) - 1$, czyli $x = \frac{b+a}{2} + \frac{b-a}{2}u$. Z formuły trójczłonowej łatwo można wywnioskować, że wielomian $T_k(u)$ jest sumą wyrażenia $2^{k-1}u^k$ i pewnego wielomianu stopnia mniejszego niż k. Zatem współczynnik w bazie potęgowej przy x^k, czyli współczynnik wiodący wielomianu $q_k(x)$ jest równy 1. Wielomian q_k ma k miejsc zerowych w przedziale $[a, b]$ i w $k+1$ punktach tego przedziału, w tym w obu jego końcach, przyjmuje wartości ekstremalne, równe $\pm(b-a)^k/2^{2k-1}$.

Udowodnimy, że żaden wielomian stopnia k ze współczynnikiem wiodącym równym 1 nie może mieć mniejszych co do modułu wartości w całym przedziale $[a, b]$. Istotnie, gdyby taki wielomian, $w(x)$, istniał, to wielomian $r(x) = q_k(x) - w(x)$ miałby stopień mniejszy niż k, ale musiałby mieć co najmniej k miejsc zerowych w przedziale $[a, b]$, bo wykres wielomianu w przecinałby wykres wielomianu q_k co najmniej raz między każdymi jego sąsiednimi punktami ekstremalnymi (sąsiednie ekstrema mają tę samą wartość bezwzględną i przeciwne znaki, a wielomian w ma mieć w przedziale $[a, b]$ mniejsze wartości bezwzględne). Zatem, taki wielomian w nie istnieje. □

Dla dowolnych węzłów interpolacyjnych wielomian p_{n+1} występujący w oszacowaniu błędu ma współczynnik wiodący równy 1. Mamy zatem narzędzie do rozwiązywania zadania aproksymacji: aby przybliżyć funkcję klasy C^{n+1} w przedziale $[a, b]$, wybieramy tzw. węzły Czebyszewa, określone wzorem

$$x_j = \frac{b+a}{2} + \frac{b-a}{2}\cos\frac{2j+1}{2n+2}\pi \quad \text{dla } j = 0, \ldots, n,$$

i konstruujemy wielomian interpolacyjny Lagrange'a h_n stopnia n z tymi węzłami. Wtedy otrzymamy $p_{n+1} = q_{n+1}$ i

$$\|f - h_n\|_\infty \leqslant \frac{M_{n+1}}{(n+1)!}\frac{(b-a)^{n+1}}{2^{2n+1}}.$$

Wyrażenie po prawej stronie tej nierówności możemy porównać z przyjętym progiem ε, aby sprawdzić, czy błąd jest dostatecznie mały. Jeśli nie, ale funkcja f ma ciągłe pochodne wyższych rzędów (i umiemy znaleźć ich oszacowania), to możemy spróbować szczęścia z wielomianem interpolacyjnym wyższego stopnia.

Alternans i algorytm Remeza

Teraz zajmiemy się następującym problemem: dla ustalonej funkcji rzeczywistej f należy dobrać taki wielomian g^* stopnia co najwyżej n, aby błąd aproksymacji w normie maksimum w przedziale $[a, b]$ był najmniejszy. Nieco uogólniając zadanie, rozważymy problem aproksymacji przez określone w przedziale $[a, b]$ funkcje, które są elementami ustalonej przestrzeni V o wymiarze k; zatem, mając

taką przestrzeń, chcemy w niej znaleźć element najlepiej przybliżający daną funkcję f, o której założymy, że jest ciągła.

Def. Przestrzeń liniowa V o wymiarze k, której elementami są rzeczywiste funkcje ciągłe określone w przedziale $[a,b]$, spełnia warunek Haara (albo: ma własność Haara), jeśli z faktu, że funkcja $g \in V$ ma k różnych miejsc zerowych w przedziale $[a,b]$ wynika, że jest to funkcja zerowa.

Własność Haara, dla dowolnie wybranego przedziału $[a,b]$, ma zatem przestrzeń liniowa $\mathbb{R}[x]_n$, której elementy są wielomianami stopnia co najwyżej n, ale nie tylko: weźmy przestrzeń wielomianów trygonometrycznych stopnia co najwyżej n i ustalmy dowolny przedział $[a,b]$ krótszy niż 2π (tj. krótszy niż okres wszystkich tych funkcji). Przestrzeń ta ma wymiar $2n+1$, i jak wiemy, zadanie interpolacji Lagrange'a dla $2n+1$ dowolnie wybranych w przedziale $[a,b]$ (parami różnych) węzłów ma w tej przestrzeni jednoznaczne rozwiązanie (zobacz rozdział 9 tych notatek). Jeśli więc pewien wielomian trygonometryczny stopnia n ma $2n+1$ miejsc zerowych w przedziale $[a,b]$, to jest on funkcją zerową. Natomiast *nie mają* własności Haara przestrzenie, których elementami są funkcje sklejane: istnieją niezerowe funkcje sklejane, które mają nieskończenie wiele miejsc zerowych.

Twierdzenie Czebyszewa o alternansie: *Jeśli przestrzeń V o wymiarze k spełnia warunek Haara, to dla dowolnej funkcji ciągłej f zadanie aproksymacji jednostajnej ma w przestrzeni V jednoznaczne rozwiązanie, $g^\star$. Funkcja $f-g^\star$, opisująca błąd aproksymacji, ma w przedziale $[a,b]$ co najmniej $k+1$ punktów, w których przyjmuje maksymalną wartość bezwzględną, przy czym znaki wartości funkcji $f-g^\star$ w kolejnych punktach z tego zbioru są przeciwne.*

Dowód twierdzenia Czebyszewa, który pominiemy, jest podobny do przeprowadzonego wcześniej dowodu stwierdzenia, że wielomian q_k ma najmniejszą normę $\|\cdot\|_\infty$ dla przedziału $[a,b]$ wśród wszystkich wielomianów stopnia k o współczynniku wiodącym 1 (i jest to jedyny taki wielomian).

Zbiór punktów, w których funkcja $f-g^\star$ przyjmuje na przemian minimalną i maksymalną wartość (wszystkie o tej samej wartości bezwzględnej $\|f-g^\star\|_\infty$) nazywany jest alternansem. Rozwiązanie zadania aproksymacji polega na znalezieniu takiego wielomianu $g^\star$ stopnia co najwyżej n, aby funkcja $f-g^\star$ przyjmowała w $n+2$ punktach przedziału $[a,b]$ wartości ekstremalne o zmieniających się znakach. Jeśli funkcja f jest wypukła albo wklęsła i poszukujemy optymalnego wielomianu stopnia 1, to alternans składa się z trzech punktów, z których dwa są końcami przedziału $[a,b]$, dzięki czemu zadanie jest łatwe.

Jeśli poszukujemy optymalnego wielomianu wyższego stopnia, to możemy użyć opisanego niżej algorytmu Remeza, w którym konstruuje się pewien ciąg wielomianów $(g^{(j)})_{j\in\mathbb{N}}$ stopnia n, zbieżny do poszukiwanego wielomianu $g^\star$.

Za $g^{(0)}$ można przyjąć wielomian interpolacyjny Lagrange'a z $n+1$ węzłami Czebyszewa w przedziale $[a,b]$. Istotne jest, aby funkcja $f-g^{(0)}$ miała w przedziale $[a,b]$ co najmniej $n+2$ lokalne minima i maksima, rozmieszczone na przemian (wartości bezwzględne tych ekstremów mogą być różne), i taki wybór funkcji $g^{(0)}$ to zapewnia: funkcja $g^{(0)}$ ma minimum lub maksimum między każdymi dwoma węzłami interpolacyjnymi, a także przed pierwszym i za ostatnim węzłem, a jeśli znaki kolejnych ekstremów są takie same (o co jest bardzo trudno), to zamiast jednego z nich można przyjąć węzeł.

Na podstawie wielomianu $g^{(j-1)}$ należy skonstruować $g^{(j)}$. W tym celu trzeba znaleźć *wszystkie* ekstrema funkcji $f-g^{(j-1)}$ w przedziale $[a,b]$. To może być bardzo trudnym zadaniem obliczeniowym. Mając pewne informacje o funkcji f, możemy ustalić gęstość, z jaką wystarczy stablicować tę funkcję i wielomian $g^{(j-1)}$ w przedziale $[a,b]$, aby nie „zgubić" żadnego ekstremum (to może być np. 100, 1000, lub nawet więcej punktów), potem trzeba zastosować jakąś metodę numeryczną znajdowania punktów ekstremalnych z dużą dokładnością[11]. Następnie tworzymy j-te przybliżenie alternansu: wybieramy $n+2$ punkty w przedziale $[a,b]$, w których funkcja $f-g^{(j-1)}$ przyjmuje wartości ekstremalne, przy czym jeśli lokalnych ekstremów jest więcej niż $n+2$, to trzeba wybrać punkty, w których ekstrema mają największe wartości bezwzględne, z zachowaniem warunku zmieniających się znaków. Oznaczmy wybrane punkty symbolami $y_0^{(j)},\dots,y_{n+1}^{(j)}$ (lub lepiej w skrócie $y_0,\dots,y_{n+1}$). Założymy, że są one uporządkowane monotonicznie.

Wielomian $g^{(j)}$ ma spełniać następujący warunek: dla $i=0,\dots,n+1$ ma być $f(y_i)-g^{(j)}(y_i)=(-1)^i r_j$, gdzie r_j jest niewiadomą liczbą. Zatem, zachodzi równość $f(x)-g^{(j)}(x)=r_j h^{(j)}(x)$ dla pewnej funkcji $h^{(j)}$, takiej że $h^{(j)}(y_i)=(-1)^i$ dla $i=0,\dots,n+1$. Obliczając różnicę dzieloną rzędu $n+1$, otrzymamy

$$f[y_0,\dots,y_{n+1}] = r_j h^{(j)}[y_0,\dots,y_{n+1}],$$

bo różnica dzielona rzędu $n+1$ wielomianu $g^{(j)}$ stopnia n jest zerem. Ale stąd możemy obliczyć

$$r_j = \frac{f[y_0,\dots,y_{n+1}]}{h^{(j)}[y_0,\dots,y_{n+1}]},$$

a następnie użyć r_j do obliczenia wartości wielomianu $g^{(j)}$ w punktach y_i i znaleźć

[11] Metody znajdowania ekstremów, nieobecne w tym wykładzie, mają związek z metodami rozwiązywania równań nieliniowych.

ten wielomian przez rozwiązanie zadania interpolacyjnego Lagrange'a (mamy tu o 1 węzeł i warunek interpolacyjny za dużo, ale to nie szkodzi).

Ciąg wielomianów $g^{(j)}$ zwykle dość szybko zbiega do wielomianu $g^\star$, który przybliża funkcję f z najmniejszym błędem w przestrzeni $\mathbb{R}[x]_n$, przy czym ciąg liczb $|r_j|$ zbiega do normy błędu, tj. maksymalnej wartości bezwzględnej różnicy $f(x) - g^\star(x)$ w przedziale $[a,b]$. Jak widać z opisu (który jest dosyć uproszczony), to jest kosztowny algorytm, którego stosowanie opłaca się, jeśli wartości wielomianu $g^\star$ mają być obliczane *bardzo wiele razy*. Najczęściej poprzestaje się na wielomianach nieoptymalnych, np. wielomianach interpolacyjnych z węzłami Czebyszewa, które zwykle dają niewiele większy błąd aproksymacji, a są nieporównanie łatwiejsze do znalezienia. Jeśli błąd aproksymacji przez taki wielomian jest zbyt duży, rozwiązaniem problemu może być przyjęcie większego stopnia. Inna możliwość, to konstrukcja aproksymacyjnej funkcji sklejanej.

Aproksymacja jednostajna przez funkcje sklejane

Zauważmy, że gdyby funkcja f miała być przybliżana w przedziale dwukrotnie krótszym (np. w połowie przedziału $[a,b]$), w którym przyjęlibyśmy węzły interpolacyjne rozmieszczone w dwukrotnie mniejszych odstępach, to czynnik $\|p_{n+1}\|_\infty$ we wzorze (*) dla tego krótszego przedziału byłby 2^{n+1} razy mniejszy. Zatem skrócenie przedziału $[a,b]$ jest radykalnym sposobem zmniejszenia błędu aproksymacji jednostajnej i czasami jest to jedyny skuteczny sposób. Mając długi przedział, możemy podzielić go na krótsze podprzedziały i aproksymować funkcję f w każdym z nich innym wielomianem niskiego stopnia. Odpowiednio dobrana reprezentacja aproksymacyjnej funkcji sklejanej może zapewnić dobre własności analityczne w danym zastosowaniu, tj. ciągłość pochodnych dostatecznie wysokiego rzędu.

Znanych jest wiele twierdzeń na temat aproksymacji funkcjami sklejanymi różnych stopni. Bez dowodu, który jest dosyć żmudny (choć pouczający), podam jedno z tych twierdzeń.

Twierdzenie. *Niech* $f \in C^2[a,b]$ *i niech* s *oznacza kubiczną funkcję sklejaną klasy* $C^2[a,b]$ *z węzłami* $u_0 = a < u_1 < \cdots < u_N = b$, *taką że* $s(u_i) = f(u_i)$ *dla* $i = 0,\ldots,n$. *Niech* M_2 *oznacza stałą taką że* $|f''(x)| \leqslant M_2$ *dla każdego* $x \in [a,b]$, *oraz* $|s''(a)| \leqslant 3M_2$ *i* $|s''(b)| \leqslant 3M_2$. *Wtedy dla każdego* $x \in [a,b]$ *zachodzą nierówności*

$$|f(x) - s(x)| \leqslant \frac{1}{2}M_2h^2, \qquad |f'(x) - s'(x)| \leqslant 2M_2h,$$

gdzie $h = \max_i(u_{i+1} - u_i)$.

Twierdzenie to stosuje się m.in. do naturalnych kubicznych funkcji sklejanych (dla których $s''(a) = s''(b) = 0$), ale nie tylko. Wynika z niego, że do osiągnięcia dowolnie małego błędu wystarczy wybranie dostatecznie gęstego ciągu węzłów w przedziale $[a,b]$, a ponadto można w ten sposób również dowolnie zmniejszyć błąd aproksymacji pochodnej funkcji f.

Aproksymacja średniokwadratowa

Niech ρ oznacza funkcję określoną w przedziale A (który może być otwarty lub domknięty, ograniczony lub nieograniczony). Zakładamy, że funkcja ρ jest nieujemna, zbiór jej miejsc zerowych w A jest miary zero (np. pusty) i całka z funkcji ρ w zbiorze A jest skończona. Funkcję ρ nazywamy funkcją wagową albo wagą. Dla takiej funkcji wzór

$$\langle f,g\rangle_\rho = \int_A f(x)g(x)\rho(x)\,dx$$

określa iloczyn skalarny, a funkcjonał

$$\|f\|_\rho = \sqrt{\langle f,f\rangle_\rho} = \sqrt{\int_A f(x)^2\rho(x)\,dx}$$

jest normą. Zadanie aproksymacji średniokwadratowej często jest uogólniane w ten sposób, że dla danej funkcji f należy znaleźć w ustalonej przestrzeni V (której wymiar jest skończony) funkcję g, taką że wyrażenie $\|f-g\|_\rho$ jest najmniejsze.

Rozwiązaniem zadania jest wektor (funkcja) $g^\star$, która jest rzutem prostopadłym wektora (funkcji) f na przestrzeń V; zadanie aproksymacji średniokwadratowej jest w istocie uogólnieniem liniowego zadania najmniejszych kwadratów. Mając bazę $p_0,\ldots,p_n$ przestrzeni V, wystarczy znaleźć współczynniki $x_0,\ldots,x_n$ wektora $g^\star = \sum_{j=0}^n x_jp_j$ w tej bazie. Wektor $f - g^\star$ jest prostopadły do wszystkich elementów bazy przestrzeni V. Na podstawie tego warunku możemy wyprowadzić układ równań normalnych

$$\sum_{j=0}^n \langle p_i,p_j\rangle_\rho x_j = \langle p_i,f\rangle_\rho, \quad \text{dla } i = 0,\ldots,n.$$

Macierz $A = [\langle p_i,p_j\rangle_\rho]_{i,j}$ tego układu równań jest symetryczna i dodatnio określona.

Wielomiany ortogonalne

Zadanie aproksymacji średniokwadratowej jest znacznie łatwiejsze do rozwiązania, jeśli dysponujemy bazą ortogonalną przestrzeni V, tj. układem wektorów

(funkcji) $p_0, \ldots, p_n$, takich że $\mathrm{lin}\{p_0, \ldots, p_n\} = V$ oraz $\langle p_i, p_j\rangle_\rho = 0$ dla $i \neq j$. Dla takiej bazy macierz układu równań normalnych jest diagonalna. Mając dowolną bazę przestrzeni V, możemy znaleźć bazę ortogonalną za pomocą ortogonalizacji Grama-Schmidta. Jeśli $V = \mathbb{R}[x]_n$, tj. elementami przestrzeni V są wszystkie wielomiany stopnia co najwyżej n, to za pomocą ortogonalizacji bazy potęgowej możemy znaleźć bazę ortogonalną $p_0, \ldots, p_n$, w której dla każdego k stopień wielomianu p_k jest równy k. Bazę taką możemy również znaleźć za pomocą odpowiedniej formuły trójczłonowej; wcześniej otrzymaliśmy tym sposobem wielomiany Czebyszewa.

Twierdzenie. *Dla ustalonego przedziału $A \subset \mathbb{R}$ i funkcji wagowej ρ wielomiany p_k, tworzące układ ortogonalny dla $k = 0, 1, \ldots$ i takie, że dla każdego k stopień wielomianu p_k jest równy k, wyrażają się wzorem*

$$p_k(x) = (\alpha_k x + \beta_k)p_{k-1}(x) + \gamma_k p_{k-2}(x) \quad \textit{dla } k = 1, 2, \ldots,$$

gdzie $\alpha_k \neq 0$ (można wybrać dowolnie), oraz

$$\beta_k = -\frac{\alpha_k\langle xp_{k-1}, p_{k-1}\rangle_\rho}{\|p_{k-1}\|_\rho^2}, \qquad \gamma_k = -\frac{\alpha_k\langle xp_{k-1}, p_{k-2}\rangle_\rho}{\|p_{k-2}\|_\rho^2},$$

przy czym $p_{-1}(x) \stackrel{\mathrm{def}}{=} 0$, $p_0(x) \stackrel{\mathrm{def}}{=} a_0 \neq 0$.

Dowód. Dla każdego k stopień wielomianu p_k jest równy k, zatem jego współczynnik wiodący $a_k \neq 0$. Niech $\alpha_k = a_k/a_{k-1}$. Niech $w_k = p_k - \alpha_k x p_{k-1}$. Wielomian w_k jest stopnia mniejszego niż k. Dla iloczynu skalarnego określonego za pomocą całki z wagą zachodzi równość $\langle xf, g\rangle_\rho = \langle f, xg\rangle_\rho$, zatem dla $j < k-2$

$$\langle w_k, p_j\rangle_\rho = \langle p_k - \alpha_k x p_{k-1}, p_j\rangle_\rho = \langle p_k, p_j\rangle_\rho - \alpha_k\langle p_{k-1}, xp_j\rangle_\rho = 0.$$

Wyrażając wielomian w_k w bazie $p_0, \ldots, p_{k-1}$, otrzymamy

$$\langle w_k, p_j\rangle_\rho = \langle \sum_{i=0}^{k-1} b_{ki}p_i, p_j\rangle_\rho = \sum_{i=0}^{k-1} b_{ki}\langle p_i, p_j\rangle_\rho = b_{kj}\langle p_j, p_j\rangle_\rho.$$

Stąd $b_{kj} = 0$ dla $j < k-2$, a zatem mamy

$$p_k = \alpha_k x p_{k-1} + \beta_k p_{k-1} + \gamma_k p_{k-2},$$

dla $\beta_k = b_{k,k-1}$, $\gamma_k = b_{k,k-2}$. Możemy obliczyć

$$0 = \langle p_k, p_{k-1}\rangle_\rho = \alpha_k\langle xp_{k-1}, p_{k-1}\rangle_\rho + \beta_k\langle p_{k-1}, p_{k-1}\rangle_\rho + \gamma_k\underbrace{\langle p_{k-2}, p_{k-1}\rangle_\rho}_{=0},$$

$$0 = \langle p_k, p_{k-2}\rangle_\rho = \alpha_k\langle xp_{k-1}, p_{k-2}\rangle_\rho + \beta_k\underbrace{\langle p_{k-1}, p_{k-2}\rangle_\rho}_{=0} + \gamma_k\langle p_{k-2}, p_{k-2}\rangle_\rho$$

skąd otrzymujemy podane wyrażenia na β_k i γ_k. □

Oczywiście, można wybrać liczby a_0 i α_k tak, aby dostać bazę ortonormalną, ale nie zawsze się tak robi. Ważną własnością wielomianów ortogonalnych (dowód jest prostym ćwiczeniem) jest to, że wszystkie ich miejsca zerowe są rzeczywiste, jednokrotne i położone wewnątrz przedziału A.

Wielomiany ortogonalne są znane dla wielu różnych przedziałów i wag; największe znaczenie praktyczne mają

- wielomiany Legendre'a: $\rho(x) = 1$, $A = (-1, 1)$,
- wielomiany Czebyszewa: $\rho(x) = (1-x^2)^{-1/2}$, $A = (-1, 1)$,
- wielomiany Hermite'a[12]: $\rho(x) = e^{-x^2}$, $A = \mathbb{R}$,
- wielomiany Laguerre'a: $\rho(x) = e^{-x}$, $A = (0, +\infty)$.

Tak więc jeśli mamy bazę przestrzeni $\mathbb{R}[x]_n$ ortogonalną w sensie iloczynu skalarnego $\langle\cdot,\cdot\rangle_\rho$, to zadanie znalezienia wielomianu $g_n^\star$ stopnia co najwyżej n, najlepiej przybliżającego funkcję f, sprowadza się do obliczenia współczynników wielomianu $g_n^\star$ w tej bazie:

$$x_i = \frac{\langle f, p_i\rangle_\rho}{\|p_i\|_\rho^2}.$$

Mamy przy tym, na podstawie twierdzenia Pitagorasa, wyrażenie opisujące błąd:

$$\|f - g_n^\star\|_\rho^2 = \|f\|_\rho^2 - \sum_{i=0}^{n} x_i^2\|p_i\|_\rho^2,$$

dzięki czemu, jeśli błąd jest za duży, możemy zwiększać n, obliczając tylko kolejne współczynniki x_i (ale uwaga: są takie funkcje f, dla których błąd nie maleje do zera, gdy $n \to \infty$ — trzeba uważać). Podstawą rozwiązywania zadań aproksymacji średniokwadratowej jest obliczanie całek, co można robić analitycznie (jeśli umiemy, a nie zawsze tak jest) lub numerycznie. Numerycznym całkowaniem zajmiemy się wkrótce.

[12] Nie należy mieszać wielomianów ortogonalnych Hermite'a z zadaniem interpolacyjnym Hermite'a.

Zadania i problemy

1. Algorytm Clenshawa. Niech $h(x) = \sum_{i=0}^{n} a_i T_i(x)$. Dla $n > 1$ możemy obliczyć

$$\begin{aligned} h(x) = {} & a_n T_n(x) + a_{n-1} T_{n-1}(x) + a_{n-2} T_{n-1}(x) + \cdots = \\ & a_n(2x T_{n-1}(x) - T_{n-2}(x)) + a_{n-1} T_{n-1}(x) + a_{n-2} T_{n-1}(x) + \cdots = \\ & (2x a_n + a_{n-1}) T_{n-1}(x) + (a_{n-2} - a_n) T_{n-2}(x) + \cdots = \\ & \tilde{a}_{n-1} T_{n-1}(x) + \tilde{a}_{n-2} T_{n-2}(x) + \cdots . \end{aligned}$$

   Wykonując ten rachunek rekurencyjnie, możemy otrzymać liczby $\hat{a}_1$ i $\hat{a}_0$, takie że $h(x) = \hat{a}_1 T_1(x) + \hat{a}_0 T_0(x) = \hat{a}_1 x + \hat{a}_0$. Napisz procedurę obliczającą $h(x)$, opartą na tym pomyśle.

2. Znajdź najmniejsze n, takie że wielomian interpolacyjny h_n stopnia n z węzłami Czebyszewa w przedziale $[0, \pi/2]$ przybliża jednostajnie funkcję $f(x) = \sin x$ w tym przedziale z błędem nie większym niż 10^{-4}.

   Co można powiedzieć o błędzie względnym aproksymacji w tym przypadku?

   Wskazówka: w oszacowaniach wygodnie jest korzystać z nierówności $\pi^2 < 10$.

3. Znajdź wielomian stopnia co najwyżej 1, który przybliża jednostajnie funkcję $f(x) = e^x$ w przedziale $[0, 1]$ z najmniejszym możliwym błędem.

4. Znajdź wielomiany stopnia 0, 1 i 2, będące rozwiązaniami aproksymacji średniokwadratowej dla funkcji $f(x) = e^x$ w przedziale $[0, 1]$ z wagą $\rho(x) = 1$. W rachunkach użyj bazy potęgowej.

5. Udowodnij, że wielomiany Czebyszewa stanowią układ ortogonalny dla iloczynu skalarnego określonego za pomocą całki w przedziale $(-1, 1)$ z wagą $(1 - x^2)^{-1/2}$.

6. Znajdź wielomiany ortogonalne Legendre'a stopnia 0, 1, 2 i 3, za pomocą ortogonalizacji bazy potęgowej.

7. Udowodnij, że jeśli p_k jest wielomianem k-tego stopnia, należącym do rodziny ortogonalnej w sensie iloczynu skalarnego określonego za pomocą całki w spójnym przedziale A z funkcją wagową ρ, to wszystkie jego miejsca zerowe są rzeczywiste, jednokrotne i leżą wewnątrz przedziału A.

   Wskazówka: przy założeniu, że liczba j punktów w przedziale A, w otoczeniu których wielomian p_k zmienia znak jest mniejsza niż k, określ wielomian stopnia j, którego to są miejsca zerowe, i zbadaj iloczyn skalarny tego wielomianu i p_k.

Numeryczne obliczanie całek

Def. Niech f oznacza pewną funkcję określoną w przedziale $[a, b]$. Kwadratura jest to kombinacja liniowa wartości funkcji f w pewnych punktach $x_i \in [a, b]$, zwanych węzłami kwadratury:

$$Q(f) = \sum_{i=0}^{n-1} A_i f(x_i).$$

Liczby A_i są nazywane współczynnikami kwadratury.
Ogólniejsza definicja określa kwadraturę jako kombinację liniową wartości funkcji f i jej pochodnych w węzłach kwadratury.

Kwadratura jest zatem funkcjonałem liniowym na przestrzeni funkcji określonych w przedziale $[a, b]$, podobnie jak całka oznaczona[13]:

$$I(f) = \int_a^b f(x)\rho(x)\,dx.$$

W odróżnieniu od całki, mogąc obliczać wartości funkcji f w dowolnych punktach przedziału $[a, b]$, można obliczyć wartość kwadratury za pomocą skończenie wielu działań arytmetycznych. Numeryczne obliczanie całek polega na obliczaniu kwadratur. Ważne jest zapewnienie dostatecznej dokładności, tj. dostatecznie małego błędu aproksymacji całki przez kwadraturę. Temu celowi służy wybór węzłów i współczynników kwadratury. Jak zwykle, skuteczność wyboru zależy od własności funkcji, które mamy zamiar całkować.

Kwadratury interpolacyjne

Kwadratura interpolacyjna jest całką z wielomianu interpolacyjnego Lagrange'a lub Hermite'a funkcji f z węzłami w przedziale $[a, b]$. Jeśli jest to wielomian interpolacyjny Lagrange'a (tj. węzły są jednokrotne, obliczamy w nich tylko wartości funkcji f), to kwadratura ma współczynniki

$$A_i = \int_a^b \prod_{j\in\{0,\ldots,n-1\}\setminus\{i\}} \frac{x - x_j}{x_i - x_j}\rho(x)\,dx.$$

Wśród kwadratur interpolacyjnych wyróżniamy kwadratury Newtona-Cotesa, których węzły dzielą przedział $[a, b]$ na części o równych długościach (kwadratury te określa się z wagą $\rho(x) = 1$), kwadratury Gaussa, których węzły są miejscami

[13] Obliczając całkę, w pewnych przypadkach wygodnie jest wyróżnić pewien czynnik w funkcji podcałkowej, $\rho(x)$, i traktować go jako wagę, choć zwykle przyjmuje się wagę $\rho(x) = 1$. Funkcje wagowe, które nie są stałe, przydają się zwłaszcza podczas obliczania całek w przedziale nieograniczonym lub całek z funkcji nieograniczonych. Zakładamy, że funkcja ρ spełnia warunki podane w wykładzie o aproksymacji, i w szczególności jest nieujemna.

zerowymi wielomianów ortogonalnych, a także inne kwadratury, dobierane specjalnie do zastosowań.

Błąd kwadratury jest to oczywiście różnica $I(f) - Q(f)$, która zależy od funkcji f. Błąd kwadratury interpolacyjnej opartej na n węzłach można oszacować, obliczając całkę z wyrażenia opisującego resztę interpolacji:

$$|I(f) - Q(f)| \leqslant \frac{M_n}{n!}\int_a^b |p_n(x)|\rho(x)\,dx,$$

ale to oszacowanie jest poprawne, jeśli funkcja f jest klasy $C^n[a, b]$, i możemy go użyć bezpośrednio, jeśli umiemy znaleźć stałą M_n, taką że $\|f^{(n)}\|_\infty \leqslant M_n$.

Def. Rząd kwadratury jest to liczba r, taka że kwadratura ma tę samą wartość co całka dla każdego wielomianu stopnia mniejszego niż r oraz inną wartość niż całka dla pewnego wielomianu stopnia r.

Z definicji kwadratury interpolacyjnej natychmiast wynika, że jej rząd jest nie mniejszy niż liczba węzłów. Rząd żadnej kwadratury opartej na n węzłach nie może być większy niż $2n$, ponieważ jeśli p_n jest wielomianem stopnia n, którego miejscami zerowymi są wszystkie węzły, to mamy $Q(p_n^2) = 0$ oraz $I(p_n^2) > 0$.

Możemy wybrać pewien ciąg kwadratur $Q_1, Q_2, \ldots$, np. kwadratur Newtona-Cotesa coraz wyższych rzędów, i zbadać zbieżność ciągu liczb $Q_1(f), Q_2(f), \ldots$ dla funkcji f spełniającej określone warunki (np. funkcji ciągłej). Chciałoby się, aby ten ciąg miał granicę, równą $I(f)$; jeśli ją ma, to istotna jest szybkość zbieżności do tej granicy. Korzystając z twierdzenia Weierstrassa, można udowodnić

Twierdzenie. *Ciąg $Q_1(f), Q_2(f), \ldots$ jest zbieżny do granicy $I(f)$ dla dowolnej funkcji ciągłej f wtedy i tylko wtedy, gdy jest zbieżny dla każdego wielomianu i istnieje stała K, taka że suma wartości bezwzględnych współczynników każdej kwadratury w rozpatrywanym ciągu jest mniejsza niż K.*

Pierwszy warunek podany w twierdzeniu jest spełniony przez każdy ciąg kwadratur interpolacyjnych coraz wyższych rzędów, natomiast aby spełnić drugi warunek, wystarczy zapewnić, że współczynniki każdej kwadratury są nieujemne. Niestety, ciąg kwadratur Newtona-Cotesa tego warunku nie spełnia, co więcej, sumy wartości bezwzględnych współczynników tych kwadratur rosną nieograniczenie. Praktycznie użyteczne kwadratury Newtona-Cotesa mają tylko kilka (mniej niż 8) węzłów. Zbadamy dwie najprostsze z nich.

Kwadratura trapezów oparta jest na dwóch węzłach, będących końcami przedziału $[a,b]$:

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

Łatwo jest sprawdzić, że rząd tej kwadratury jest równy 2. Jeśli funkcja f jest klasy $C^2[a,b]$, to $p_2(x) = (x-a)(x-b)$ i mamy oszacowanie błędu

$$|I(f) - T(f)| \leqslant \frac{M_2}{2}\int_a^b |p_2(x)|\,dx = \frac{M_2}{12}(b-a)^3,$$

ze stałą M_2, taką że $|f''(x)| \leqslant M_2$ dla każdego $x \in [a,b]$.

Kwadratura Simpsona oparta jest na trzech węzłach: końcach i środku przedziału $[a,b]$; oznaczymy $c = (a+b)/2$:

$$S(f) = \frac{b-a}{6}\big(f(a)+4f(c)+f(b)\big).$$

Okazuje się, że rząd kwadratury Simpsona jest równy 4. Jest tak dlatego, że to jest kwadratura interpolacyjna, której środkowy węzeł jest dwukrotny, ale współczynnik, przez który należałoby pomnożyć $f'(c)$, jest równy 0. Błąd kwadratury Simpsona możemy oszacować na dwa sposoby:

$$|I(f) - S(f)| \leqslant \frac{M_3}{6}\int_a^b |(x-a)(x-c)(x-b)|\,dx = \frac{M_3}{192}(b-a)^4,$$
$$|I(f) - S(f)| \leqslant \frac{M_4}{24}\int_a^b |(x-a)(x-c)^2(x-b)|\,dx = \frac{M_4}{2880}(b-a)^5,$$

gdzie M_3 i M_4 to oszacowania wartości bezwzględnych pochodnych trzeciego i czwartego rzędu funkcji f w przedziale $[a,b]$. Oczywiście, każdego z tych oszacowań możemy używać pod warunkiem, że odpowiednia pochodna funkcji f jest ciągła.

Zamiana zmiennych

Jeśli $f(u) = g(x)$ dla $x = su+t$, gdzie $s > 0$ i t są ustalonymi liczbami, oraz $c = sa+t$, $d = sb+t$, to

$$\int_a^b f(u)\rho(u)\,du = \frac{1}{s}\int_c^d g(x)\rho((x-t)/s)\,dx, \quad \text{oraz}$$
$$Q_1(f) = \sum_{i=0}^{n-1} A_i f(u_i) = \frac{1}{s}\sum_{i=0}^{n-1} sA_i g(su_i+t) = \frac{1}{s}Q_2(g).$$

W ten sposób, mając dowolną kwadraturę Q_1:

$$Q_1(f) = \sum_{i=0}^{n-1} A_i f(u_i) \approx \int_a^b f(u)\rho(u)\,du,$$

możemy otrzymać nową kwadraturę Q_2:

$$Q_2(g) = \sum_{i=0}^{n-1} B_i g(x_i) \approx \int_c^d g(x)\rho((x-t)/s)\,dx,$$

z węzłami $x_i = su_i + t$ i współczynnikami $B_i = sA_i$.

Kwadratury Q_1 i Q_2 mają ten sam rząd. Ponadto, mając oszacowanie błędu kwadratury Q_1, podobne do podanych wcześniej oszacowań dla kwadratur trapezów i Simpsona, można podać oszacowanie błędu kwadratury Q_2. Mianowicie, jeśli funkcje f i g są klasy C^k w swoich przedziałach całkowania[14] i błąd kwadratury Q_1 ma górne oszacowanie o postaci

$$C(b-a)^{k+1}\max_{u\in[a,b]}|f^{(k)}(u)|,$$

to błąd kwadratury Q_2 jest nie większy niż

$$C(c-d)^{k+1}\max_{x\in[c,d]}|g^{(k)}(x)|,$$

z tą samą stałą C.

Kwadratury Gaussa

Niech $A \subset \mathbb{R}$ oznacza (ograniczony lub nieograniczony) przedział całkowania, niech ρ oznacza funkcję wagową i niech $p_0, p_1, \ldots$ będzie ciągiem wielomianów ortogonalnych w sensie iloczynu skalarnego

$$\langle f, g\rangle_\rho \stackrel{\text{def}}{=} \int_A f(x)g(x)\rho(x)\,dx.$$

Ustalmy liczbę n i określmy kwadraturę interpolacyjną Q z węzłami, które są miejscami zerowymi $x_0, \ldots, x_{n-1}$ wielomianu p_n; możemy to zrobić, bo miejsca zerowe tego wielomianu są jednokrotne i znajdują się w przedziale A. Dowolny wielomian w stopnia mniejszego niż $2n$ możemy przedstawić w postaci

$$w(x) = p_n(x)a(x) + r(x),$$

[14] Tu rozważamy k w ogólności inne niż liczba węzłów n. Dla kwadratury interpolacyjnej rzędu $r \geqslant n$ można wskazać takie oszacowania dla $k = \{n, \ldots, r\}$, przykładem jest kwadratura Simpsona i kwadratury Gaussa, opisane dalej.

gdzie a i r to iloraz i reszta z dzielenia wielomianu w przez p_n; stopnie wielomianów a i r są mniejsze niż n. Dzięki temu zachodzą równości

$$I(w) = \int_A w(x)\rho(x)\,dx = \int_A \big(p_n(x)a(x) + r(x)\big)\rho(x)\,dx = \underbrace{\langle p_n, a\rangle_\rho}_{=0} + \int_A r(x)\rho(x)\,dx = Q(r) = Q(w),$$

ponieważ wartości wielomianów w i r we wszystkich węzłach kwadratury są jednakowe. Skonstruowana w ten sposób kwadratura jest zatem rzędu $2n$.

Kwadratury interpolacyjne, których węzły są miejscami zerowymi wielomianów ortogonalnych (odpowiadających danemu przedziałowi i funkcji wagowej) są nazywane kwadraturami Gaussa; nazwisko rodziny, do której odpowiedni wielomian należy, jest dołączane do nazwiska Gauss, i w ten sposób mówi się np. o kwadraturach Gaussa-Legendre'a, kwadraturach Gaussa-Czebyszewa, kwadraturach Gaussa-Hermite'a i kwadraturach Gaussa-Laguerre'a.

Konstruując kwadraturę Gaussa, na ogół trzeba jej węzły znaleźć, rozwiązując numerycznie równanie $p_n(x) = 0$. Współczynniki kwadratury Gaussa można obliczyć tak, jak współczynniki dowolnej kwadratury interpolacyjnej, lub na podstawie wzoru

$$A_i = \frac{1}{\sum_{k=0}^{n-1} \tilde{p}_k(x_i)^2},$$

w którym występują wielomiany ortonormalne $\tilde{p}_k(x) = p_k(x)/\|p_k\|_\rho$. Wygodnie jest użyć w tym obliczeniu formuły trójczłonowej. Zatem współczynniki każdej kwadratury Gaussa są dodatnie i z podanego wcześniej twierdzenia wynika, że dla dowolnej funkcji ciągłej ciąg kwadratur Gaussa coraz wyższych rzędów zbiega do całki z tej funkcji (z odpowiednią wagą).

Największe znaczenie praktyczne mają kwadratury Gaussa-Legendre'a, ponieważ najczęściej oblicza się całki w skończonym przedziale, z wagą $\rho(x) = 1$. Najprostsza kwadratura Gaussa-Legendre'a jest iloczynem długości przedziału całkowania i wartości funkcji w środku tego przedziału. Jest to więc kwadratura rzędu 2, oparta na jedym węźle.

Niech p_n oznacza wielomian ortogonalny Legendre'a stopnia n, wyskalowany tak, aby jego współczynnik wiodący był równy 1. Błąd aproksymacji jednostajnej funkcji f klasy $C^{2n}[-1,1]$ przez wielomian interpolacyjny Hermite'a h_{2n-1} stopnia $2n-1$, oparty na węzłach kwadratury Gaussa-Legendre'a (czyli miejscach

zerowych wielomianu p_n), z których każdy liczymy dwukrotnie[15], ma oszacowanie

$$\max_{x\in[-1,1]} |f(x) - h_{2n-1}(x)| \leqslant \frac{M_{2n}}{(2n)!} p_n(x)^2,$$

gdzie $M_{2n} = \max_{x\in[-1,1]} |f^{(2n)}(x)|$. Niech

$$C_n = \int_{-1}^{1} p_n(x)^2\,dx.$$

Po dokonaniu zamiany zmiennych, możemy oszacować błąd kwadratury Gaussa-Legendre'a rzędu $2n$ dla przedziału $[a,b]$:

$$|I(f) - Q(f)| \leqslant C_n \frac{M_{2n}}{(2n)!}\left(\frac{b-a}{2}\right)^{2n+1},$$

przy czym teraz M_{2n} oznacza oszacowanie pochodnej rzędu $2n$ funkcji f w przedziale $[a,b]$.

Kwadratury złożone

Tak jak w aproksymacji jednostajnej funkcji, skutecznym sposobem zmniejszenia błędu aproksymacji całki przez kwadraturę jest podzielenie przedziału całkowania na krótsze podprzedziały i obliczenie sumy kwadratur interpolacyjnych dla tych podprzedziałów. W ten sposób otrzymuje się kwadratury złożone. Błąd takiej kwadratury jest sumą błędów kwadratur dla podprzedziałów, przy czym błędy te mogą mieć różne znaki, a zatem mogą się znosić. Oszacowania błędów kwadratur złożonych zwykle są sumami oszacowań błędów w podprzedziałach, przez co często bywają pesymistyczne.

Dodatkową korzyścią z zastosowania kwadratury złożonej jest możliwość podziału przedziału całkowania w punktach nieciągłości funkcji podcałkowej lub jej pochodnych (jeśli punktów tych jest skończenie wiele i je znamy). Wtedy w każdym podprzedziale funkcja podcałkowa ma wyższą klasę ciągłości, co umożliwia stosowanie kwadratur odpowiednio wyższego rzędu. Ponadto, po dokonaniu podziału można stosować w podprzedziałach różne kwadratury, dostosowane do zachowania funkcji podcałkowej w tych podprzedziałach. Kolejna możliwość to adaptacja — dla konkretnej funkcji można znaleźć oszacowania błędów w poszczególnych podprzedziałach, i na tej podstawie podejmować decyzję o dalszym (rekurencyjnym) podziale niektórych z nich.

[15] Kwadratura Gaussa jest w istocie całką z wielomianu interpolacyjnego Hermite'a, którego wszystkie węzły mają krotność 2. Oprócz wartości funkcji podcałkowej należałoby więc uwzględnić wartości jej pochodnej w węzłach, ale wybór węzłów zapewnia, że wszystkie współczynniki, przez które trzeba pomnożyć wartości pochodnej, są równe 0 — zobacz też opis kwadratury Simpsona.

Kwadratury w podprzedziałach konstruujemy za pomocą opisanej wcześniej zamiany zmiennych. Zobaczmy przykłady kwadratur z podziałem przedziału $[a, b]$ na N części o tej samej długości $h = (b - a)/N$.

Złożona kwadratura trapezów powstaje w ten sposób, że w każdym z podprzedziałów przedziału $[a, b]$ stosujemy kwadraturę trapezów. W ten sposób otrzymamy liczbę

$$T_h(f) = h\Big(\frac{1}{2}f(x_0) + \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2}f(x_N)\Big),$$

gdzie $x_i = a + ih$. Jeśli funkcja f jest klasy $C^2[a, b]$ i $|f''(x)| \leqslant M_2$ dla każdego $x \in [a, b]$, to wartość bezwzględna lokalnego błędu kwadratury trapezów w przedziale $[x_i, x_{i+1}]$ nie przekracza $\frac{M_2}{12}h^3$, a zatem suma tych błędów ma oszacowanie

$$|I(f) - T_h(f)| \leqslant \frac{M_2}{12}(b - a)h^2.$$

Złożoną kwadraturę Simpsona otrzymujemy analogicznie. Oznaczmy $x_i = a + ih/2$ dla $i = 0, \ldots, 2N$. Suma kwadratur Simpsona w N podprzedziałach o długości h jest równa

$$S_h(f) = \frac{h}{6}\Big(f(x_0) + 4f(x_1) + \sum_{i=1}^{N-1}\big(2f(x_{2i}) + 4f(x_{2i+1})\big) + f(x_{2N})\Big),$$

zaś dla funkcji f odpowiednio klasy $C^3[a, b]$ i $C^4[a, b]$ błąd ma oszacowania

$$|I(f) - S_h(f)| \leqslant \frac{M_3}{192}(b - a)h^3,$$
$$|I(f) - S_h(f)| \leqslant \frac{M_4}{2880}(b - a)h^4.$$

Konstruowanie złożonych kwadratur Gaussa jest utrudnione, jeśli funkcja wagowa nie jest stała, dlatego powyższe podejście stosuje się tylko do kwadratur Gaussa-Legendre'a. Jeśli funkcja f jest klasy $C^{2n}[a, b]$, to możemy w każdym przedziale o długości h użyć kwadratury Gaussa-Legendre'a opartej na n węzłach i wtedy dostaniemy oszacowanie błędu o postaci

$$|I(f) - Q_h(f)| \leqslant C_n M_{2n}(b - a)h^{2n},$$

w którym stała C_n zależy tylko od rzędu kwadratury. Jak widać, dla $h \to 0$ błąd bardzo szybko dąży do zera. Jeśli funkcja f nie ma ciągłych pochodnych aż tak wysokiego rzędu, to błąd nadal dąży do zera, choć wolniej.

Ekstrapolacja Richardsona i metoda Romberga

Niech f oznacza funkcję klasy $C^{2n+2}[a, b]$. Dowodzi się, że błąd złożonej kwadratury trapezów, z przedziałem $[a, b]$ podzielonym na podprzedziały o jednakowej długości h, można wyrazić wzorem

$$I(f) - T_h(f) = c_1h^2 + c_2h^4 + \cdots + c_nh^{2n} + O(h^{2n+2}),$$

zwanym wzorem sumacyjnym Eulera-Maclaurina. Współczynniki $c_1, \ldots, c_n$ zależą od pochodnych funkcji f w przedziale $[a, b]$, ale nie zależą od długości podprzedziałów.

Możemy ten wzór przepisać dla złożonej kwadratury trapezów z dwukrotnie drobniejszym podziałem przedziału całkowania:

$$I(f) - T_{h/2}(f) = \frac{c_1}{4}h^2 + \frac{c_2}{16}h^4 + \cdots + \frac{c_n}{4^n}h^{2n} + O(h^{2n+2}),$$

Jeśli strony powyższego wzoru pomnożymy przez $4/3$ i odejmiemy od nich strony wzoru dla kwadratury z podprzedziałami o długości h pomnożone przez $1/3$, to otrzymamy równość

$$I(f) - \Big(\frac{4}{3}T_{h/2}(f) - \frac{1}{3}T_h(f)\Big) = d_2h^4 + \cdots + d_nh^{2n} + O(h^{2n+2}).$$

Kombinacja liniowa[16] $T_h^{(1)}(f) = 4/3T_{h/2}(f) - 1/3T_h(f)$ jest kwadraturą, której dominujący składnik błędu jest rzędu h^4, zatem znacznie szybciej maleje podczas zmniejszania h. Opisany sposób wyeliminowania dominującego składnika błędu (który można stosować także w innych przypadkach, gdy błąd jest opisany za pomocą szeregu potęgowego) jest nazywany ekstrapolacją Richardsona.

Ekstrapolację Richardsona możemy iterować. Mając kwadratury $T_h^{(j)}$ i $T_{h/2}^{(j)}$, których dominujące składniki błędów są proporcjonalne do h^{2j+2}, określamy kwadraturę

$$T_h^{(j+1)}(f) = \frac{2^{2j+2}}{2^{2j+2} - 1}T_{h/2}^{(j)}(f) - \frac{1}{2^{2j+2} - 1}T_h^{(j)}(f),$$

której błąd ma dominujący składnik błędu h^{2j+4}. Oparta na tym pomyśle metoda numerycznego całkowania jest nazywana metodą Romberga. Podprogram obliczający całkę, dla ustalonego h, oblicza kwadratury $T_h(f)$ i $T_{h/2}(f)$ i oblicza kwadraturę $T_h^{(1)}(f)$. Wyrażenie $|T_h(f) - T_{h/2}(f)|$ może być przyjęte za oszacowanie

[16] Co ciekawe, to jest złożona kwadratura Simpsona, S_h. Natomiast dalej opisane kwadratury $T_h^{(2)}, T_h^{(3)}, \ldots$ *nie są* złożonymi kwadraturami interpolacyjnymi.

błędu, co jest analogią do przyrostowego kryterium stopu w metodach numerycznych rozwiązywania równań nieliniowych. Jeśli to oszacowanie jest zbyt duże, to obliczana jest kwadratura $T_{h/4}(f)$, a następnie $T^{(1)}_{h/2}(f)$ i $T^{(2)}_{h}(f)$ itd. Obliczenie przebiega zgodnie ze schematem

$$
\begin{array}{ccccccc}
T_h(f) & & & & & & \\
 & \searrow & & & & & \\
T_{h/2}(f) & \to & T^{(1)}_h(f) & & & & \\
 & \searrow & & \searrow & & & \\
T_{h/4}(f) & \to & T^{(1)}_{h/2}(f) & \to & T^{(2)}_h(f) & & \\
\vdots & & \vdots & & \vdots & \ddots & \\
 & \searrow & & \searrow & & \searrow & \\
T_{h/2^k}(f) & \to & T^{(1)}_{h/2^{k-1}}(f) & \to & \dots & \to & T^{(k)}_h(f)
\end{array}
$$

Za oszacowanie błędu każdej kwadratury otrzymanej przez ekstrapolację możemy przyjąć różnicę kwadratur, na podstawie których została ona obliczona. Zauważmy, że po każdym zmniejszeniu długości podprzedziałów dla kwadratury trapezów wartości funkcji podcałkowej wystarczy tylko obliczyć tylko w nowych węzłach i nie ma potrzeby przechowywania wartości funkcji f w tablicy.

Całkowanie funkcji wielu zmiennych

Znane z analizy twierdzenie Fubiniego umożliwia sprowadzenie zadania obliczenia całki z funkcji f określonej w wielowymiarowym obszarze A do obliczenia całek jednowymiarowych. Analogicznie można postępować z kwadraturami. Jest to szczególnie proste, gdy obszar A jest kostką. Powiedzmy, że jest to prostokąt: $A = [a, b] \times [c, d]$. Mając kwadratury przybliżające całki w przedziałach $[a, b]$ i $[c, d]$, odpowiednio z węzłami $x_0, \dots, x_{n-1}$ i $y_0, \dots, y_{m-1}$ oraz współczynnikami $A_0, \dots, A_{n-1}$ i $B_0, \dots, B_{m-1}$, możemy obliczyć

$$Q(f) = \sum_{i=0}^{n-1} A_i \sum_{j=0}^{m-1} B_j f(x_i, y_j) \approx \int_a^b \left(\int_c^d f(x, y)\, dy \right) dx.$$

Znane są również kwadratury odpowiednie dla obszarów o innym kształcie. Jeśli np. obszar całkowania A jest trójkątem, to kwadratury określone wzorami

$$Q_1(f) = \frac{T}{3}\big(f(\mathbf{a}) + f(\mathbf{b}) + f(\mathbf{c})\big),$$
$$Q_2(f) = \frac{T}{3}\big(f(\mathbf{p}) + f(\mathbf{q}) + f(\mathbf{r})\big),$$
$$Q_3(f) = \frac{T}{60}\Big(3\big(f(\mathbf{a}) + f(\mathbf{b}) + f(\mathbf{c})\big) + 8\big(f(\mathbf{p}) + f(\mathbf{q}) + f(\mathbf{r})\big) + 27 f(\mathbf{s})\Big),$$

w których T oznacza pole trójkąta A o wierzchołkach $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, środkach boków $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ i środku ciężkości $\mathbf{s}$, są dokładne, jeśli funkcja f jest odpowiednio wielomianem stopnia 1, 2 i 3. Dowolny obszar wielokątny możemy podzielić na trójkąty i stosować kwadratury złożone.

Numeryczne całkowanie jest kłopotliwe, jeśli wymiar obszaru A jest duży. Istnieją zadania praktyczne (biorące się m.in. z ekonomii), w których wymiar d obszaru całkowania jest rzędu kilkuset. Obliczenie całki w takiej kostce d-wymiarowej za pomocą kwadratury otrzymanej analogicznie, jak dla prostokąta, jest niewykonalne. Nawet gdyby w przedziale zmienności każdej zmiennej wybrać tylko dwa węzły, liczba punktów, w których trzeba by obliczyć wartości funkcji podcałkowej, byłaby równa 2^d. Zjawisko wykładniczego wzrostu złożoności obliczeniowej zadania ze wzrostem wymiaru dziedziny funkcji nosi nazwę przekleństwa wymiaru (ang. *dimensionality curse*).

Znanych jest kilka sposobów obliczania przybliżonych wartości całek wielowymiarowych za pomocą wartości funkcji obliczonych w znacznie mniejszej liczbie punktów. Sposób najprostszy i jednocześnie skuteczny dla najszerszej klasy takich zadań wynalazł Ulam w 1946 r. Sposób ten jest znany pod nazwą metody Monte Carlo. Obszar A uznajemy za przestrzeń zadrzeń elementarnych i określamy w nim jednostajny rozkład prawdopodobieństwa. Wtedy funkcja f jest zmienną losową. Iloczyn wartości oczekiwanej tej zmiennej losowej i miary $|A|$ obszaru A jest poszukiwaną całką, $\int_A f$. Dla n niezależnych losowań punktów $\mathbf{x}_i \in A$ możemy określić nową zmienną losową wzorem

$$Q(f) = |A| \frac{1}{n} \sum_{i=0}^{n-1} f(\mathbf{x}_i).$$

Jest to właśnie kwadratura Monte Carlo; jej wartość oczekiwana też jest równa poszukiwanej całce. Jeśli zmienna losowa f ma wariancję σ^2, to wariancja σ_n^2 kwadratury Monte Carlo jest równa $|A|\sigma^2/n$. Zatem odchylenie standardowe σ_n zmiennej losowej $Q(f)$ jest proporcjonalne do $n^{-1/2}$ i w szczególności nie zależy od wymiaru d obszaru A. Dla dostatecznie dużego n możemy oczekiwać, że błąd jest bardzo mały — z dużym prawdopodobieństwem, ale nie z całkowitą pewnością.

Zadania i problemy

1. Dla kwadratury interpolacyjnej opisanej wzorem

$$Q(f) = A_0 f(a) + A_1 f(c),$$

przybliżającej całkę $\int_a^b f(x)\,dx$, znajdź węzeł c i współczynniki A_0, A_1 tak, aby rząd tej kwadratury był jak największy. Podaj rząd i oszacowanie błędu tej kwadratury.

Wskazówka. Zacznij od przypadku $a = 0$, $b = 1$, następnie uogólnij otrzymane wyniki na przypadek dowolnego przedziału $[a, b]$.

2. Znajdź współczynniki kwadratury

$$Q(f) = A_0 f(a) + B_0 f'(a) + A_1 f(b) + B_1 f'(b),$$

przybliżającej całkę $\int_a^b f(x)\,dx$.
Jaki jest rząd tej kwadratury? Podaj oszacowanie jej błędu.

Wskazówka. Rozwiąż zadanie dla przypadku $a = 0$, $b = 1$, korzystając z wielomianów H_{00}, H_{01}, H_{10} i H_{11}, użytych do konstrukcji interpolacyjnych kubicznych funkcji sklejanych. Następnie uogólnij wynik.

3. Oblicz współczynniki kwadratury Gaussa-Czebyszewa rzędu $2n$.

Wskazówka. Dokonaj zamiany zmiennych, podstawiając $x = \cos t$.

4. Jaka kwadratura wydaje się najodpowiedniejsza do obliczania tzw. zupełnych całek eliptycznych pierwszego i drugiego rodzaju, tj. funkcji określonych wzorami

$$K(k) = \int_0^1 \frac{1}{\sqrt{(1-k^2x^2)(1-x^2)}}\,dx \qquad \text{i} \qquad E(k) = \int_0^1 \sqrt{\frac{1-k^2x^2}{1-x^2}}\,dx$$

dla $k \in [0, 1)$, i jak jej użyć?

5. Przypuśćmy, że należy obliczyć całkę z funkcji f w dwuwymiarowym obszarze A, który leży między wykresami dwóch funkcji ciągłych, c i d, określonych w przedziale $[a, b]$ i takich że $c(x) < d(x)$ dla każdego $x \in [a, b]$. Zaprojektuj algorytm obliczania tej całki za pomocą pewnej kwadratury Q, której węzły i współczynniki są podane w tablicach. Kwadratura Q przybliża całkę z funkcji jednej zmiennej w przedziale $[0, 1]$.

Wybrane środowiska i biblioteki dla obliczeń numerycznych