Metody numeryczne

Piotr Krzyżanowski

https://www.mimuw.edu.pl/~przykry/mn
2023/01/26, 14:58:41

Spis treści

Układy równań liniowych

Układy równań liniowych

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N & = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N & = b_2 \\ \vdots & \\ a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N & = b_N \end{cases}$$

Macierzowo możemy zapisać ten układ w zwartej postaci

$$Ax = b,$$

gdzie

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1N} \\ a_{21} & a_{22} & \ldots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & \ldots & a_{NN} \end{bmatrix}, \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}.$$

Tak, to GAL!

$$Ax = b$$

Układ ma dokładnie jedno rozwiązanie $\iff$ spełniony jest którykolwiek z warunków:

- $\det(A) \neq 0$
- jedynym rozwiązaniem $Ax = 0$ jest $x = 0$
- $\ker(A) = \{0\}$
- A nie ma zerowej wartości własnej
- A jest nieosobliwa

Praca domowa

Powtórzyć sobie podstawowe wiadomości z GALu:

- macierze i wektory
- mnożenie macierzy przez macierz i przez wektor
- macierz odwrotna
- wartości własne i wektory własne macierzy
- rząd macierzy, macierz pełnego rzędu
- ortogonalność wektorów

Łatwe układy równań: z macierzą diagonalną

$$\begin{cases} a_{11}x_1 & & & = b_1 \\ & a_{22}x_2 & & = b_2 \\ & & & \vdots \\ & & a_{NN}x_N & = b_N \end{cases}$$

Trywialne!...

$$x_i = b_i/a_{ii}, \qquad i = 1, \ldots N,$$

Koszt: $\mathcal{O}(N)$ flopów.

Koszt algorytmów numerycznych

- obliczeniowy
- pamięciowy

Koszt obliczeniowy algorytmów numerycznych

Tradycyjnie, jest to liczba wykonanych operacji arytmetycznych:

Czym jest *flop*?

FLOP = FLoating point OPeration

Tradycyjnie, przyjmujemy, że wszystkie operacje ($+$, $-$, $\times$, $\div$) kosztują tyle samo: 1 flop.
Tradycyjnie, nie uwzględniamy kosztu obsługi pętli i innych instrukcji sterujących, ani kosztu dostępu do pamięci.
Zwykle przyjmujemy, że $\sqrt{}$ też kosztuje 1 flop.

Dzisiaj żadne z tych założeń nie jest prawdziwe na typowym komputerze!

Niemniej w zdecydowanej większości przypadków ten model daje dobry wgląd w *jakościowe* zachowanie się algorytmu w rzeczywistości.

Łatwe układy równań: z macierzą trójkątną

$$\begin{cases} a_{11}x_1 & = b_1 \\ a_{21}x_1 + a_{22}x_2 & = b_2 \\ a_{21}x_1 + a_{22}x_2 + a_{33}x_3 & = b_3 \\ & \vdots \\ a_{N1}x_1 + a_{N2}x_2 + a_{N3}x_3 + \ldots a_{NN}x_N & = b_N \end{cases}$$

Macierz układu jest *dolna trójkątna*

$$A = \begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{N1} & a_{N2} & \ldots & a_{NN} \end{bmatrix}$$

Łatwe układy równań: z macierzą trójkątną

Rozwiązanie metodą podstawienia w przód:

$$\begin{aligned} a_{11}x_1 &= b_1 \quad \Longrightarrow \\ x_1 &= \frac{1}{a_{11}} b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \quad \Longrightarrow \\ x_2 &= \frac{1}{a_{22}}(b_2 - a_{12}x_1) \\ a_{21}x_1 + a_{22}x_2 + a_{33}x_3 &= b_3 \quad \Longrightarrow \\ x_3 &= \frac{1}{a_{33}}(b_3 - a_{13}x_1 - a_{23}x_2) \\ &\vdots \\ a_{k1}x_1 + a_{k2}x_2 + a_{k3}x_3 + \ldots + a_{kk}x_k &= b_k \quad \Longrightarrow \\ x_k &= \frac{1}{a_{kk}}(b_k - a_{1k}x_1 - a_{2k}x_2 - \ldots - a_{k-1,k}x_{k-1}) \\ &\vdots \end{aligned}$$

Łatwe układy równań: z macierzą trójkątną

Algorytm 1 Podstawienie w przód. Rozwiązywanie układu $Ax = b$ z macierzą dolną trójkątną

$x_1 = b_1/a_{11}$
for $i = 2 : N$
  $x_i = \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j)$
end
return $(x_1, \ldots, x_N)$

Koszt tego algorytmu jest rzędu $\mathcal{O}(N^2)$ flopów.

Łatwe układy równań: z macierzą trójkątną

Analogicznie, kosztem $\mathcal{O}(N^2)$ flopów rozwiązujemy układ z macierzą górną trójkątną:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ & a_{22} & \dots & a_{2N} \\ & & \ddots & \vdots \\ & & & a_{NN} \end{bmatrix},$$

metodą podstawienia w tył.

Łatwe układy równań: z macierzą ortogonalną

Definicja (Macierz ortogonalna)

Macierz $Q \in \mathbb{R}^{N \times N}$ nazywamy ortogonalną, jeśli

$$Q^T Q = I.$$

Inaczej: kolumny macierzy Q są wzajemnie prostopadłe i ich norma euklidesowa jest równa 1.

Wniosek

$$Q^{-1} = Q^T,$$

zatem rozwiązaniem układu

$$Qx = b$$

jest

$$x = Q^{-1} b.$$

Koszt: $\mathcal{O}(N^2)$ flopów.

A co zrobić w przypadku ogólnym?

Jak rozwiązać

$$Ax = b,$$

gdy A — macierz nieosobliwa.

Zły pomysł

Wzory Cramera ???

Koszt zaporowy: obliczenie $\det(A)$ z rozwinięcia Laplace'a kosztuje przynajmniej $N!$ flopów.

Są też inne powody, by tego nie robić.

A co zrobić w przypadku ogólnym?

Jak rozwiązać

$$Ax = b,$$

gdy A — macierz nieosobliwa.

Kolejny zły pomysł

Obliczyć A^{-1}, a następnie $x = A^{-1}b$???

Pierwsze przykazanie numeryka

Nie będziesz odwracać macierzy swojej nadaremno!

A co zrobić w przypadku ogólnym?

Jak rozwiązać

$$Ax = b,$$

gdy A — macierz nieosobliwa.

Gdyby

$$A = X \cdot Y \cdot Z,$$

gdzie X, Y, Z — „łatwe" do rozwiązania, na przykład:

- diagonalne (koszt: $\mathcal{O}(N)$)
- trójkątne (koszt: $\mathcal{O}(N^2)$)
- ortogonalne (koszt: $\mathcal{O}(N^2)$)

to

$$A \cdot x = b \iff X \cdot Y \cdot Z \cdot x = b.$$

A co zrobić w przypadku ogólnym?

$$A \cdot x = b \iff X \cdot Y \cdot Z \cdot x = b.$$

$$Ax = b \iff X\,Y\,\underbrace{\underbrace{Zx}_{=:y}}_{=:z} = b.$$

Algorytm:

1. Znaleźć rozkład $A = XYZ$
2. Rozwiązać układ $Xz = b$ (max koszt $\mathcal{O}(N^2)$)
3. Rozwiązać układ $Yy = z$ (max koszt $\mathcal{O}(N^2)$)
4. Rozwiązać układ $Zx = y$ (max koszt $\mathcal{O}(N^2)$)

Łączny koszt: koszt rozkładu $+\mathcal{O}(N^2)$.

Ważne rozkłady macierzy

- rozkład LU: iloczyn macierzy trójkątnych L i U:

  $$A = LU \qquad \text{lub} \qquad PA = LU$$

  (P — macierz permutacji)
- rozkład QR: iloczyn macierzy ortogonalnej Q i trójkątnej R:

  $$A = QR$$

Jak je wyznaczać? Ile to kosztuje?

Rozkład LU (bez wyboru)

Na początek rozważamy algorytm bez wyboru elementu głównego (bez osiowania).

$$A = LU$$

Podzielmy macierze na bloki:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ & U_{22} \end{bmatrix}$$

gdzie bloki $(1,1)$ są rozmiaru $k \times k$.

- U_{ii} — górne trójkątne
- L_{ii} — dolne trójkątne (z jedynkami na diagonali $\implies$ jednoznaczność)

Rozkład LU bez wyboru — wyprowadzenie

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ & U_{22} \end{bmatrix}$$

Po wymnożeniu

$$\begin{aligned} A_{11} &= L_{11}U_{11} & A_{12} &= L_{11}U_{12} \\ A_{21} &= L_{21}U_{11} & A_{22} &= L_{21}U_{12} + L_{22}U_{22} \end{aligned}$$

skąd dostajemy algorytm rekurencyjny:

1. Wyznacz (rekurencyjnie) rozkład $A_{11} = L_{11}U_{11}$
2. $U_{12} = L_{11}^{-1}A_{12}$ (tzn. rozwiąż układ z macierzą trójkątną)
3. $L_{21} = A_{21}U_{11}^{-1}$ (tzn. rozwiąż układ z macierzą trójkątną)
4. Aktualizuj $\tilde{A}_{22} = A_{22} - L_{21}U_{12}$
5. Wyznacz (rekurencyjnie) rozkład $\tilde{A}_{22} = L_{22}U_{22}$

Dygresja: blokowy rozkład LU

Jeśli A_{11} — nieosobliwa, to

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} I & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} A_{11} & \\ & S \end{bmatrix} \begin{bmatrix} I & U_{12} \\ & I \end{bmatrix}$$

gdzie

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12}$$

— uzupełnienie Schura oraz

$$L_{21} = A_{21}A_{11}^{-1}, \qquad U_{12} = A_{11}^{-1}A_{12}.$$

Koszt rozkładu LU

Dwa interesujące przypadki:

- $k = 1$, tzn. macierz A_{11} jest skalarem
- $k = \lceil N/2 \rceil$, tzn. dzielimy A na ćwiartki

Inny interesujący przypadek:

- Dzielimy A na $B \times B$ bloków „równego" rozmiaru $\approx N/B$ (wcześniejszy przypadek to $B = 2$)
- Nie będziemy go tu rozważać.

Klasyczny algorytm rozkładu LU: $k = 1$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ & U_{22} \end{bmatrix}$$

1. Wyznacz (rekurencyjnie) rozkład $A_{11} = L_{11}U_{11}$
2. $U_{12} = L_{11}^{-1}A_{12}$ (tzn. rozwiąż układ z macierzą trójkątną)
3. $L_{21} = A_{21}U_{11}^{-1}$ (tzn. rozwiąż układ z macierzą trójkątną)
4. Aktualizuj $\tilde{A}_{22} = A_{22} - L_{21}U_{12}$
5. Wyznacz (rekurencyjnie) rozkład $\tilde{A}_{22} = L_{22}U_{22}$

$k = 1 \implies A_{11} = a_{11}$ (liczba) $\implies l_{11} = 1 \implies$ pierwsze wywołanie rekurencji jest zbędne!

Klasyczny algorytm rozkładu LU (bez wyboru)

$$\underbrace{\left[\begin{array}{c|c} a_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right]}_{A} = \underbrace{\left[\begin{array}{c|c} 1 & 0^T \\ \hline l_{21} & L_{22} \end{array}\right]}_{L} \cdot \underbrace{\left[\begin{array}{c|c} u_{11} & u_{12}^T \\ \hline 0 & U_{22} \end{array}\right]}_{U},$$

1. $u_{11} = a_{11}$
2. $u_{12} = a_{12}$
3. $l_{21} = a_{21} u_{11}^{-1}$ (mnożenie wektora a_{21} przez liczbę u_{11}^{-1})
4. Aktualizuj $A_{22} = A_{22} - l_{21} u_{12}^T$ (tzn. o macierz rzędu 1)
5. Wyznacz rozkład $A_{22} = L_{22} U_{22}$ — to już nie musi być rekurencja: wystarczy pętla!

Rozkład możemy wykonać *w miejscu*, zapisując U w górnym trójkącie A, a L (bez jedynek) — w dolnym trójkącie A.

Klasyczny algorytm rozkładu LU (bez wyboru): implementacja

Rozkład wykonujemy w miejscu, nadpisując A czynnikami L i U:

for $k = 1 : N-1$
  for $i = k+1 : N$
    $a_{ik} = a_{ik}/a_{kk}$ (wyznaczenie k-tej kolumny L)
  end
  for $i = k+1 : N$
    for $j = k+1 : N$
      $a_{ij} = a_{ij} - a_{ik} a_{kj}$ (aktualizacja $\tilde{A}_{22}$)
    end
  end
end

To nic innego jak poczciwa eliminacja Gaussa! Stąd inna nazwa: GE.

Klasyczny algorytm rozkładu LU (bez wyboru): ograniczenia

Algorytm GE nie zawsze zadziała, np.:

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

W pierwszym kroku dzielimy przez $a_{11} = 0$...

Twierdzenie

Jeśli wszystkie minory główne macierzy A są niezerowe, to GE jest wykonalna, tzn. na diagonali macierzy U nie pojawiają się zera.

Jak uniknąć dzielenia przez zero? Przez wybór elementu głównego (osiowanie).

Klasyczny algorytm rozkładu LU (z częściowym wyborem)

Rozkład wykonujemy w miejscu, nadpisując A czynnikami L i U:

for $k = 1 : N-1$
  Znajdź wiersz $p \geqslant k$ t.że $|a_{pk}| = \max\{|a_{ik}| : k \leqslant i \leqslant N\}$
  Zamień wiersze p i k
  for $i = k+1 : N$
    $a_{ik} = a_{ik}/a_{kk}$ (wyznaczenie k-tej kolumny L)
  end
  for $i = k+1 : N$
    for $j = k+1 : N$
      $a_{ij} = a_{ij} - a_{ik} a_{kj}$ (aktualizacja $\tilde{A}_{22}$)
    end
  end
end

GEPP — GE with Partial Pivoting

Klasyczny algorytm rozkładu LU (z częściowym wyborem)

- Zawsze wykonalny (w arytmetyce dokładnej), gdy A jest nieosobliwa
- Daje się zinterpretować jako

$$PA = LU,$$

  gdzie P — macierz permutacji
- Istnieją inne strategie
  - częściowy wybór w wierszu: $AP = LU$
  - wybór pełny (w całej $\tilde{A}_{22}^{(k)}$): $P_1 A P_2 = LU$

Klasyczny algorytm rozkładu LU: koszt

$$\begin{bmatrix} a_{11} & a_{12}^T \\ a_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} 1 & \\ l_{21} & L_{22} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12}^T \\ & U_{22} \end{bmatrix}$$

1. $l_{21} = a_{21} u_{11}^{-1}$ (mnożenie wektora a_{21} przez liczbę u_{11}^{-1}) Koszt: $\mathcal{O}(N)$
2. Aktualizuj $A_{22} = A_{22} - l_{21} u_{12}^T$ (tzn. o macierz rzędu 1) Koszt: $\mathcal{O}(N^2)$
3. Wyznacz rozkład $A_{22} = L_{22} U_{22}$

Zatem koszt w zależności od N:

$$T(N) = \mathcal{O}(N) + \mathcal{O}(N^2) + T(N-1) = \ldots = \mathcal{O}(N^3).$$

Podsumowanie: rozwiązanie układu $Ax = b$ metodą GEPP

$$Ax = b$$

1. GEPP — kosztem $\mathcal{O}(N^3)$ — daje nam rozkład:

   $$PA = LU$$

   Zatem

   $$PAx = LUx = Pb$$

   i stąd dalszy ciąg algorytmu:
2. $\tilde{b} = Pb$ (permutacja wektora b)
3. Rozwiąż $Ly = \tilde{b}$ (koszt $\mathcal{O}(N^2)$)
4. Rozwiąż $Ux = y$ (koszt $\mathcal{O}(N^2)$)

Koszt całkowity: $\mathcal{O}(N^3)$ (zdominowany przez rozkład $PA = LU$)

Kącik porad

Jak czytać wzory?

Napisy typu $A^{-1}Y$ należy czytać jako „rozwiąż równanie $AX = Y$"

Jak rozwiązywać problemy numeryczne?

Warto je sprowadzać sekwencji prostszych zadań, które już umiemy (tanio) rozwiązać!

Arytmetyka zmiennopozycyjna

Zapis inżynierski liczb rzeczywistych

$$6.63 \cdot 10^{-34}$$

6.63 $\longrightarrow$ „mantysa" (odpowiada za precyzję)

-34 $\longrightarrow$ cecha (odpowiada za wielkość)

10 $\longrightarrow$ podstawa (jaki system liczbowy wybieramy)

Zapis liczby rzeczywistej

$$x = (-1)^s \cdot m \cdot \beta^e$$

m $\longrightarrow$ mantysa (odpowiada za precyzję), $0 \leqslant m < \beta$

e $\longrightarrow$ cecha (odpowiada za wielkość), $e \in \mathbb{Z}$

β $\longrightarrow$ podstawa (jaki system liczbowy wybieramy)

s $\longrightarrow$ znak, $s \in \{0, 1\}$

Liczby maszynowe

Standardowo $\beta = 2$.

Możemy przeznaczyć jedynie skończoną liczbę bitów na zapis m i e.

$$x = (-1)^s \cdot m \cdot 2^e$$

$m = (f_0.f_1 f_2 \ldots f_{p-1})_2$, $f_i \in \{0, 1\}$ cyfry binarne (razem p bitów)

e integer t. że
$$(1 - e_{\max}) \leqslant e \leqslant e_{\max}$$

$\beta = 2$

s (1 bit)

Liczby maszynowe (znormalizowane) w pamięci komputera

Normalizacja: $f_0 = 1$, czyli $1 \leqslant m < 2$.

$$x = (-1)^s \cdot (1.f_1 f_2 \ldots f_{p-1})_2 \cdot 2^e$$

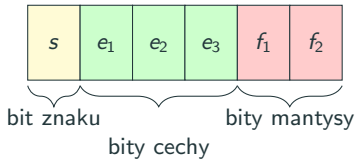
przechowujemy jako

s	$e + e_{\max}$	f_1	f_2	$\ldots$	f_{p-1}

Dzięki temu m.in. $e + e_{\max} \geqslant 0$.

Przykład: 6–bitowa arytmetyka zmiennopozycyjna i

$\beta = 2, \qquad p = 3, \qquad e_{\max} = 3.$



Liczby znormalizowane:

$$x = (-1)^s \cdot (1.f_1 f_2)_2 \cdot 2^e, \qquad e \in \{-2, \ldots, 3\}.$$

Możliwe wartości mantysy to

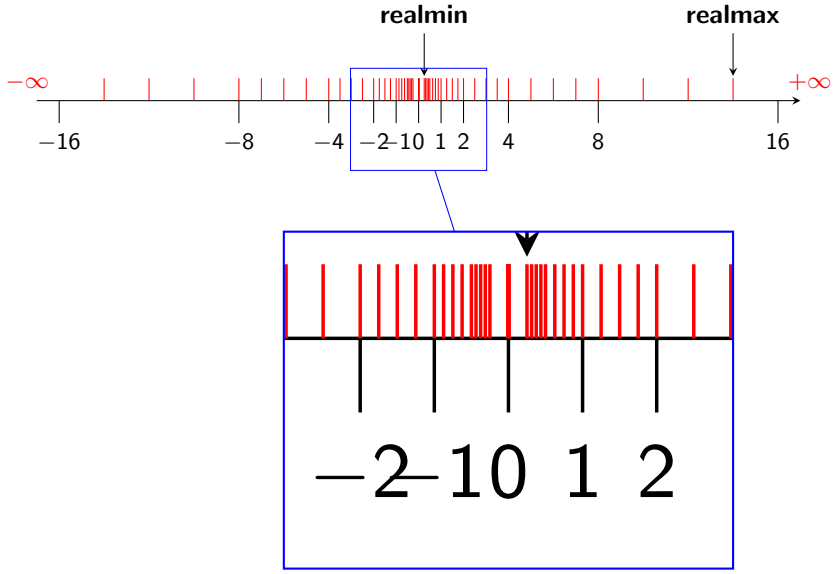
$(1.00)_2 = 1, \qquad (1.01)_2 = 1.25, \qquad (1.10)_2 = 1.5, \qquad (1.11)_2 = 1.75.$

Przykład: 6–bitowa arytmetyka zmiennopozycyjna ii

Oto wszystkie (dodatnie, znormalizowane) liczby maszynowe:

0.2500	0.3125	0.3750	0.4375
0.5000	0.6250	0.7500	0.8750
1.0000	1.2500	1.5000	1.7500
2.0000	2.5000	3.0000	3.5000
4.0000	5.0000	6.0000	7.0000
8.0000	10.0000	12.0000	14.0000

Przykład: 6–bitowa arytmetyka zmiennopozycyjna iii



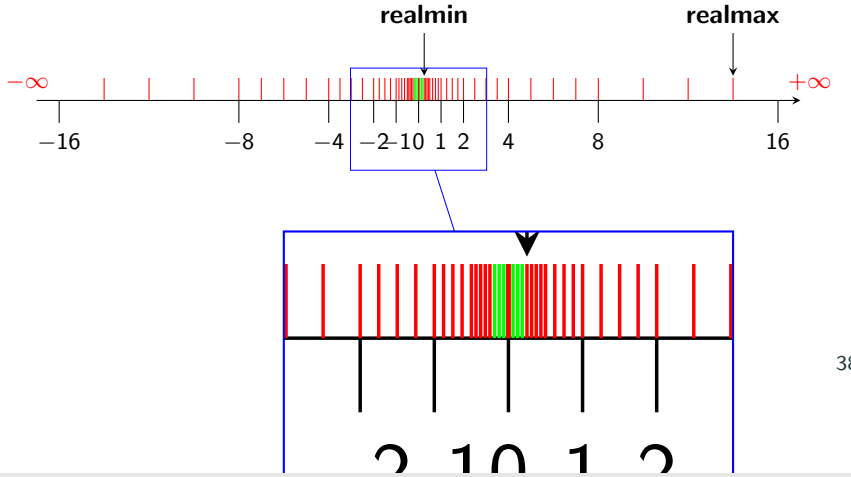
Przykład: 6–bitowa arytmetyka zmiennopozycyjna iv

Liczby subnormalne:

$$x = (-1)^s \cdot (0.f_1 f_2)_2 \cdot 2^{-2}.$$

Oto wszystkie (dodatnie) subnormalne liczby maszynowe:

0.0625, 0.125, 0.1875.



Liczby maszynowe w/g standardu IEEE 754

Nazwa	binary16	binary32	binary64	binary128	
„Precyzja"	poł.	poj.	podw.	poczw.	rozsz.
Język C		**float**	**double**		**long double**
Rozmiar	16	32	64	128	80
p	11	24	53	113	64
$e_{\max}$	15	127	1023	16383	16383
$\nu = 2^{-p}$	$4.9 \cdot 10^{-4}$	$6.0 \cdot 10^{-8}$	$1.1 \cdot 10^{-16}$	$9.6 \cdot 10^{-35}$	$5.4 \cdot 10^{-20}$
realmax	$6.6 \cdot 10^{4}$	$3.4 \cdot 10^{38}$	$1.8 \cdot 10^{308}$	$1.2 \cdot 10^{4932}$	
realmin	$6.1 \cdot 10^{-5}$	$1.2 \cdot 10^{-38}$	$2.2 \cdot 10^{-308}$	$3.4 \cdot 10^{-4932}$	
submin	$6.0 \cdot 10^{-8}$	$1.4 \cdot 10^{-45}$	$4.9 \cdot 10^{-324}$	$6.7 \cdot 10^{-4966}$	

Liczby maszynowe w/g standardu IEEE 754

Liczby „zwyczajne", znormalizowane
$x = (-1)^s \cdot (1.f_1 f_2 \ldots f_{p-1})_2 \cdot 2^e$:

s	$e + e_{\max}$	f_1	f_2	$\ldots$	f_{p-1}

Liczby o specjalnym znaczeniu:

- (plus i minus) zero $x = (-1)^s \cdot 0$

s	0	0	$\ldots$	0	0	0	$\ldots$	0

- (plus i minus) nieskończoność $x = (-1)^s \cdot \infty$

s	1	1	$\ldots$	1	0	0	$\ldots$	0

- not a number $x =$ **NaN**

s	1	1	$\ldots$	1	coś niezerowego

- liczby subnormalne $x = (-1)^s \cdot \underbrace{(0.f_1 f_2 \ldots f_{p-1})}_{\neq 0} \cdot 2^{1-e_{\max}}$

s	0	0	$\ldots$	0	f_1	f_2	$\ldots$	f_{p-1}

Reprezentacja liczb rzeczywistych

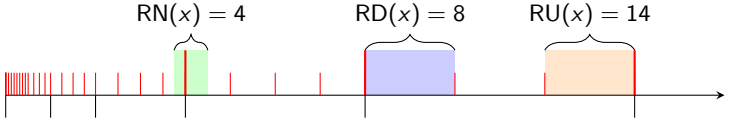
Niech $x \in \mathbb{R}$. Jeśli x nie jest liczbą maszynową, to jak ją reprezentować?

$fl(x)$ — reprezentacja x w arytmetyce zmiennopozycyjnej

IEEE 754 gwarantuje 4 możliwe tryby zaokrąglania x do liczby maszynowej:

RN do najbliższej (domyślnie)
RD w dół, tzn. w stronę $-\infty$
RU w górę, tzn. w stronę $+\infty$
RZ w stronę zera, tzn.

$$RZ(x) = \begin{cases} RD(x) & \text{gdy } x \geqslant 0 \\ RU(x) & \text{gdy } x \leqslant 0 \end{cases}$$



Precyzja arytmetyki

Jeśli $|x| \in [\textbf{realmin}, \textbf{realmax}]$, to

$$\frac{|x - RN(x)|}{|x|} \leqslant \frac{1}{2^p} =: \nu \quad \longleftarrow \text{precyzja arytmetyki}$$

Czyli wtedy błąd względny reprezentacji x przez $RN(x)$ nigdy nie przekracza ν.

Dalej zawsze przyjmujemy $fl(x) = RN(x)$.

$$fl(x) = x \cdot (1 + \epsilon), \qquad |\epsilon| \leqslant \nu.$$

Działania arytmetyczne: $\square \in \{+, -, \times, \div, \sqrt{}, FMA\}$

Jeśli a, b są liczbami maszynowymi, to $a \square b$

- może przekroczyć zakres (*overflow*):

$$fl(a \square b) = \textbf{Inf}$$

- może być zbyt bliski zera (*underflow*):

$$fl(a \square b) = 0 \qquad \textit{...lub l.subnormalna}$$

- może nie mieć sensu:

$$fl(a \square b) = \textbf{NaN}$$

- w przeciwnym przypadku,

$$fl(a \square b) = fl(a \square b)$$

Zatem poza skrajnościami,

$$fl(a \square b) = (a \square b) \cdot (1 + \epsilon), \qquad |\epsilon| \leqslant \nu.$$

Przykłady

Zakładamy działania w podwójnej precyzji.

- overflow: $10^{308} \times 100 = \textbf{Inf} = 1/0$
- underflow: $10^{-308} \div 10^{28} = 0$
- gradual underflow: $10^{-308} \div 100 \approx 10^{-310}$ (l. subnormalna)
- $0/0 =$**NaN**
- **Inf** $-$ **Inf** $=$ **NaN**
- przy okazji: (**NaN** $==$ **NaN**) $=$ False
- $0^0 =$?

Dziwaczności

- działania nie są łączne (w przeciwieństwie do dokładnej arytmetyki)
- nie musi być 1+x == 1 $\implies$ x==0

Epsilon maszynowy

Odległość ϵ_{mach} liczby 1 od następnej liczby maszynowej. Zatem $fl(1 + \epsilon_{\text{mach}}) > 1$, ale już $fl(1 + \epsilon_{\text{mach}}/2) = 1$.

- wynikiem $x - x + 1$ może być 1 ale też może być 0
- ryzykowne jest sprawdzanie x == y
- uwaga na optymalizujące kompilatory i obliczenia równoległe!

Przesąd

Fałsz!

Najważniejszym problemem w obliczeniach zmiennopozycyjnych jest kumulacja błędów zaokrągleń.

Prawda:

Kumulacja błędów ma znaczenie.
Ale większym problemem może być drastyczne wzmocnienie wcześniejszego (nawet minimalnego) błędu.

Redukcja cyfr przy odejmowaniu i

Wynik nawet jednego działania może być obarczony katastrofalnym błędem:

Dane: $x, y \in \mathbb{R}$. Obliczyć: $s = x + y$.

Zamiast x, y dysponujemy ich reprezentacjami:

$$\tilde{x} = fl(x), \qquad \tilde{y} = fl(y).$$

Zatem obliczona wartość wyniku, $\tilde{s}$, spełnia $(|\delta|, |\epsilon_x|, |\epsilon_y| \leqslant \nu)$

$$\begin{aligned}\tilde{s} = fl(\tilde{x} + \tilde{y}) &= (\tilde{x} + \tilde{y}) \cdot (1 + \delta) \\ &= (x \cdot (1 + \epsilon_x) + y \cdot (1 + \epsilon_y)) \cdot (1 + \delta) \\ &\approx \underbrace{x + y}_{=s} + (\epsilon_x + \delta) \cdot x + (\epsilon_y + \delta) \cdot y\end{aligned}$$

Redukcja cyfr przy odejmowaniu ii

Stąd błąd względny wyniku

$$\begin{aligned}\frac{|s - \tilde{s}|}{|s|} &= \frac{|(\epsilon_x + \delta) \cdot x + (\epsilon_y + \delta) \cdot y|}{|s|} \\ &\leqslant \frac{(|\epsilon_x| + |\delta|) \cdot |x| + (|\epsilon_y| + |\delta|) \cdot |y|}{|s|} \leqslant 2\frac{|x| + |y|}{|x + y|}\nu.\end{aligned}$$

Zatem

- gdy x,y są tego samego znaku, to

$$\frac{|s - \tilde{s}|}{|s|} \leqslant 2\nu \qquad \text{(wysoka precyzja wyniku!)}$$

Redukcja cyfr przy odejmowaniu iii

- gdy $x \approx -y$, to

$$\frac{|s - \tilde{s}|}{|s|} \approx \infty$$

(*żadna* cyfra znacząca wyniku może nie być dokładna!)

Wsparcie standardu IEEE 754 w językach programowania

- Bywa różnie.
- Wiele zależy od producenta konkretnego kompilatora.
- Nawet standardy nie są w pełni zgodne, np. C11 lub C++14 nie w pełni uwzględniają IEEE 754-2008.
- Niektóre opcje optymalizacji kodu wyłączają zgodność z IEEE 754. To może być jak zabawa z brzytwą.

Poza standardem IEEE 754

bfloat16:

- Używany w dedykowanych procesorach dla uczenia maszynowego, m.in. Google TPU, Intel AI (w planach).
- Coś pośredniego między binary16 a binary32, ale tylko na 16 bitach.

Nazwa	binary16	binary32	bfloat16
Rozmiar	16	32	16
p	11	24	8
$e_{\max}$	15	127	127
$\nu = 2^{-p}$	$4.9 \cdot 10^{-4}$	$6.0 \cdot 10^{-8}$	$3.9 \cdot 10^{-3}$
realmax	$6.6 \cdot 10^{4}$	$3.4 \cdot 10^{38}$	$3.4 \cdot 10^{38}$
realmin	$6.1 \cdot 10^{-5}$	$1.2 \cdot 10^{-38}$	$1.2 \cdot 10^{-38}$
submin	$6.0 \cdot 10^{-8}$	$1.4 \cdot 10^{-45}$	?

Parę słów o działaniach arytmetycznych na x86-64

- instrukcje **SSE/AVX**: nowoczesne, zalecane, wektorowe (128/256/512 bitów), binary32/binary64
- instrukcje **x87**: przestarzałe, ale jest też 80-bit i np. funkcje trygonometryczne
- unikać innych wyników niż znormalizowane (np. NaN, Inf, subnormal) — spowolnienie
- są instrukcje zwracające mniej dokładne wyniki $\div$, $\sqrt{}$ itp., ale znacznie szybsze

  (Skylake+AVX: $1/\sqrt{x}$ prawie 8× szybciej, gdy dokładność tylko 11 bitów zamiast 24)

Parę słów o działaniach arytmetycznych na x86-64 ii

Działanie	Instrukcja	Latency	Throughput	Latency	Throughput
$x + y$	ADD	4	0.5	4	0.5
$x \cdot y$	MUL	4	0.5	4	0.5
$x \cdot y + z$	FMA	–	–	4	0.5
x/y	DIV	11/14	3/4	11/14	5/8
$1/x$	RCP	4/?	1/?	4/?	1/?
$\sqrt{x}$	SQRT	13/18	3/6	12/18	6/12
$1/\sqrt{x}$	RSQRT	4/?	1/?	4/?	1/?

Tabela 1: Intel Skylake, 2019. Liczba cykli zegara dla wykonania (jednej — *Latency*, kolejnej — *Throughput*) instrukcji na liczbach single/double. Lewa strona: instrukcje 128-bit (SSE,SSE2) (single: xxxPS, double: xxxPD). Prawa strona: instrukcje AVX 256-bit (single: VxxxPS, double: VxxxPD).

Maksymalnie: $\underbrace{2}_{\text{potoki}} \times \underbrace{2}_{\text{wyniki/cykl}} \times \underbrace{2}_{\text{flopy}} \times \underbrace{4}_{\text{w rej. AVX}} = 32$ (double) flopów/cykl

Zegar 3.0 GHz $\rightarrow 32 \cdot 3 \cdot 10^9 \approx 100$ Gflop/s (/rdzeń).

Teoretycznie.

Parę słów o działaniach arytmetycznych na x86-64 iii

Więcej informacji:

- Intel® 64 and IA-32 Architectures Software Developer's Manual Vol 1: Basic Architecture
- Intel® 64 and IA-32 Architectures Optimization Reference Manual

Uwarunkowanie zadania. Numeryczna poprawność algorytmu

Zadanie obliczeniowe

Problem (zadanie obliczeniowe)

Dane jest $P : D \to W$ oraz x. Wyznaczyć $y = P(x)$.

Często (choć nie zawsze) D i W są podzbiorami przestrzeni skończonego wymiaru.

Przykłady:

- Oblicz $y = \cos(x^2)$
- Dla danego $x \in \mathbb{R}^N$, oblicz $y = A \cdot x$
- Dla danych $A \in \mathbb{R}^{N \times N}$ oraz $b \in \mathbb{R}^N$, wyznacz $y \in \mathbb{R}^N$ t.że $A \cdot y = b$. Zauważmy, że A może być poza dziedziną P
- Dla danej funkcji ciągłej f znajdź wielomian p stopnia co najwyżej n, który najlepiej ją przybliża w sensie $\| \cdot \|_{[a,b]}$.

Wrażliwość

Chcemy obliczyć

$$y = P(x).$$

Z oczywistych powodów, zamiast x dysponujemy $\tilde{x}$.

Co można powiedzieć o

$$P(\tilde{x}) - P(x) \quad ?$$

$C =?$

Błąd bezwzględny i względny

$\|\tilde{x} - x\|$ ← błąd bezwzględny

$\dfrac{\|\tilde{x} - x\|}{\|x\|}$ ← błąd względny $(x \neq 0)$

Uwarunkowanie

Wskaźnik uwarunkowania (bezwzględny): współczynnik wzmocnienia błędu (bezwzględnego) na poziomie ϵ:

$$\mathrm{cond}_{\mathrm{abs}}(P, x, \epsilon) := \sup_{\|\Delta\| \leqslant \epsilon} \frac{\|P(x+\Delta) - P(x)\|}{\|\Delta\|}$$

Faktycznie, wtedy

$$\|P(\tilde{x}) - P(x)\| \leqslant \mathrm{cond}_{\mathrm{abs}}(P, x, \epsilon) \cdot \|\tilde{x} - x\| \qquad \text{dla } \|\tilde{x} - x\| \leqslant \epsilon.$$

Idealizacja: *punktowy* wskaźnik uwarunkowania

$$\mathrm{cond}_{\mathrm{abs}}(P, x) := \lim_{\epsilon \to 0^+} \mathrm{cond}_{\mathrm{abs}}(P, x, \epsilon) = \lim_{\epsilon \to 0^+} \sup_{\|\Delta\| \leqslant \epsilon} \frac{\|P(x+\Delta) - P(x)\|}{\|\Delta\|}$$

Uwarunkowanie

Gdy P różniczkowalna w x, to

$$\mathrm{cond}_{\mathrm{abs}}(P, x) = \|P'(x)\|.$$

Analogicznie możemy pytać o wrażliwość dla błędu względnego ($x \neq 0$, $P(x) \neq 0$):

$$\mathrm{cond}_{\mathrm{rel}}(P, x, \epsilon) := \sup_{\frac{\|\Delta\|}{\|x\|} \leqslant \epsilon} \frac{\frac{\|P(x+\Delta) - P(x)\|}{\|P(x)\|}}{\frac{\|\Delta\|}{\|x\|}} = \mathrm{cond}_{\mathrm{abs}}(P, x, \epsilon\|x\|) \frac{\|x\|}{\|P(x)\|},$$

również punktowo

$$\mathrm{cond}_{\mathrm{rel}}(P, x) := \lim_{\epsilon \to 0^+} \mathrm{cond}_{\mathrm{rel}}(P, x, \epsilon)$$

Gdy P różniczkowalna w x, to

$$\mathrm{cond}_{\mathrm{rel}}(P, x) = \|P'(x)\| \frac{\|x\|}{\|P(x)\|}$$

Uwarunkowanie

Zatem dla $\tilde{x} \approx x$, mamy w przybliżeniu

- $\|P(\tilde{x}) - P(x)\| \lesssim \mathrm{cond}_{\mathrm{abs}}(P, x)\, \|\tilde{x} - x\|$
- $\dfrac{\|P(\tilde{x}) - P(x)\|}{\|P(x)\|} \lesssim \mathrm{cond}_{\mathrm{rel}}(P, x)\, \dfrac{\|\tilde{x} - x\|}{\|x\|}$

Terminologia:

- Zadanie dobrze uwarunkowane: $\mathrm{cond}(P, x)$ jest nieduże
- Zadanie źle uwarunkowane: $\mathrm{cond}(P, x)$ jest bardzo duże

Zła wiadomość: zdarzają się zadania, których uwarunkowanie może być *patologicznie* duże, np. $\mathrm{cond}(P, x) = 10^{23}$.

Uwarunkowanie układu równań liniowych

Rozwiązać układ równań:

$$Ay = b.$$

Jak wygląda zadanie obliczeniowe?

$$P : \underbrace{S \times \mathbb{R}^N}_{=D} \to \mathbb{R}^N, \qquad P(A, b) = A^{-1}b$$

gdzie $S = \{A \in \mathbb{R}^{N\times N} : A \text{ jest nieosobliwa}\}$.

Zaburzenie danych:

$$\frac{\|\tilde{A} - A\|}{\|A\|} \leqslant \epsilon, \qquad \frac{\|\tilde{b} - b\|}{\|b\|} \leqslant \epsilon.$$

Pytanie:

$$\frac{\|\tilde{y} - y\|}{\|y\|} \leqslant C \cdot \epsilon, \qquad C = ?$$

Normy wektorowe, $x \in \mathbb{R}^N$

$$\|x\|_1 = \sum_{i=1}^{N} |x_i|,$$

$$\|x\|_2 = \sqrt{\sum_{i=1}^{N} |x_i|^2},$$

$$\|x\|_p = \left(\sum_{i=1}^{N} |x_i|^p\right)^{1/p},$$

$$\|x\|_\infty = \max_{i=1,\ldots,N} |x_i|,$$

$$\|x\|_\infty \leqslant \|x\|_1 \leqslant N\|x\|_\infty,$$
$$\|x\|_\infty \leqslant \|x\|_2 \leqslant \sqrt{N}\|x\|_\infty,$$
$$\|x\|_2 \leqslant \|x\|_1 \leqslant \sqrt{N}\|x\|_2.$$

$\{x : \|x\|_2 \leqslant 1\}$

$\{x : \|x\|_\infty \leqslant 1\}$

$\{x : \|x\|_1 \leqslant 1\}$

Normy macierzowe, $A \in \mathbb{R}^{N\times M}$

Definicja (Norma macierzy indukowana normą wektorową)

$$\|A\| := \max_{\|x\|\neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\| = \max_{\|x\|\leqslant 1} \|Ax\|$$

Ozn.: $\|A\|_p := \max_{\|x\|_p=1} \|Ax\|_p$.

Twierdzenie

- $\|Ax\| \leqslant \|A\| \cdot \|x\|$
- $\|AB\| \leqslant \|A\| \cdot \|B\|$
- $\|I\| = 1$
- $\|A\|_1 = \max_j \sum_i |a_{ij}|$ *(stąd nazwa: norma kolumnowa)*
- $\|A\|_\infty = \max_i \sum_j |a_{ij}|$ *(stąd nazwa: norma wierszowa)*
- $\|A\|_2 = \max\{\sqrt{\mu} : \mu \text{ jest w.wł. } A^T A\}$ *(stąd nazwa: norma spektralna)*

Uwarunkowanie układu równań liniowych

Oznaczmy

$$\text{cond}(A) := \|A\| \cdot \|A^{-1}\|.$$

Twierdzenie

Jeśli $\epsilon\,\text{cond}(A) \leqslant 1/2$, *to*

$$\frac{\|\tilde{y} - y\|}{\|y\|} \leqslant 4\,\text{cond}(A) \cdot \epsilon.$$

Zła wiadomość: macierze *mogą* być patologicznie źle uwarunkowane, $\text{cond}(A) \gg 1$.

Uwarunkowanie układu równań liniowych — dowód

$$Ay = b \qquad (A+\Delta)\tilde{y} = b + \delta$$

stąd

$$(A+\Delta)(\tilde{y} - y) = \delta - \Delta y,$$

więc

$$\tilde{y} - y = (A+\Delta)^{-1}(\delta - \Delta y).$$

Zatem

$$\|\tilde{y} - y\| = \|(A+\Delta)^{-1}(\delta - \Delta y)\| \leqslant \|(A+\Delta)^{-1}\|\|\delta - \Delta y\|.$$

Uwarunkowanie układu równań liniowych — dowód

Lemat

Jeśli $\|\Delta\| < 1$, *to* $I + \Delta$ *nieosobliwa oraz*

$$\frac{1}{1+\|\Delta\|} \leqslant \|(I+\Delta)^{-1}\| \leqslant \frac{1}{1-\|\Delta\|}$$

Na mocy lematu, jeśli A odwracalna i $\|A^{-1}\|\|\Delta\| < 1$, to $(A+\Delta)^{-1}$ istnieje oraz

$$\|(A+\Delta)^{-1}\| \leqslant \frac{\|A^{-1}\|}{1-\|A^{-1}\|\|\Delta\|}$$

Uwarunkowanie układu równań liniowych — dowód

$$\begin{aligned}
\|\tilde{y} - y\| &\leqslant \|(A+\Delta)^{-1}\|\|\delta - \Delta y\| \\
&\leqslant \|(A+\Delta)^{-1}\|(\|\delta\| + \|\Delta\|\|y\|) \\
&\leqslant \frac{\|A^{-1}\|}{1-\|A^{-1}\|\|\Delta\|}(\|\delta\| + \|\Delta\|\|y\|) \\
&\leqslant \frac{\|A^{-1}\|}{1-\epsilon\|A\|\|A^{-1}\|}(\epsilon \cdot \underbrace{\|b\|}_{=\underbrace{\|Ay\|}_{\leqslant\|A\|\|y\|}} + \epsilon \cdot \|A\|\|y\|) \\
&\leqslant \frac{2\|A\|\|A^{-1}\|}{1-\frac{1}{2}}\epsilon \cdot \|y\| = 4\,\mathrm{cond}(A)\epsilon\|y\|.
\end{aligned}$$

Algorytm

Problem (zadanie obliczeniowe)

Dane jest $P : D \to W$ oraz x. Wyznaczyć $y = P(x)$.

Definicja (algorytm dla zadania obliczeniowego)

Niech $X \subset D$. Przekształcenie $A : X \to W$, którego wartość można określić jako sekwencję operacji elementarnych, nazywamy *algorytmem* rozwiązywania zadania P *w klasie* X.

- Typowo algorytm będzie *skończoną* sekwencją operacji.
- Nieuniknione:
  - błąd reprezentacji w fl: informacji i wyniku
  - realizacja wszystkich działań w fl

Algorytm numerycznie poprawny

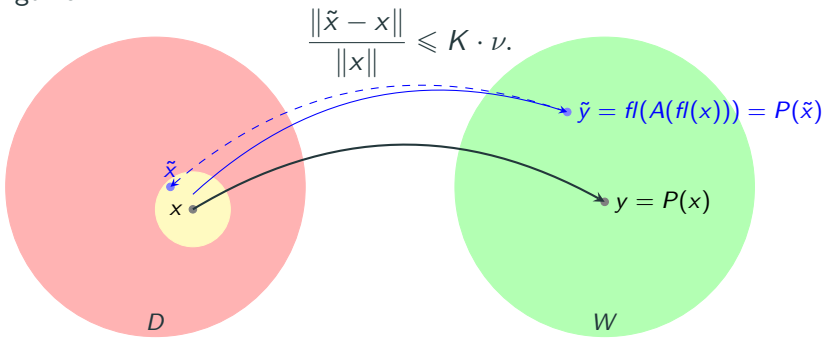
Chcielibyśmy, aby to co obliczyliśmy w fl za pomocą algorytmu:

$$fl(A(fl(x)))$$

miało *sensowny związek* z tym, co chcemy obliczyć: $P(x)$.

$$y = P(x), \qquad \tilde{y} = fl(A(fl(x))) = P(\tilde{x}),$$

gdzie



Algorytm numerycznie poprawny i

$$y = P(x), \qquad \tilde{y} = fl(A(fl(x))) = P(\tilde{x}).$$

Definicja (algorytm numerycznie poprawny)

Dla każdego $x \in X$ wynik algorytmu A zrealizowanego w fl $fl(A(fl(x))$ jest dokładnym rozwiązaniem zadania dla danych x zaburzonych na poziomie błędu reprezentacji.

Algorytm numerycznie poprawny ii

Wniosek

Algorytm NP daje wynik, którego błąd można oszacować na podstawie własności zadania obliczeniowego:

$$\frac{\|\tilde{y}-y\|}{\|y\|} = \frac{\|P(\tilde{x})-P(x)\|}{\|P(x)\|} \lesssim \mathrm{cond}_{\mathrm{rel}}(P,x)\frac{\|\tilde{x}-x\|}{\|x\|} \leqslant K \cdot \mathrm{cond}_{\mathrm{rel}}(P,x) \cdot \nu.$$

Błąd algorytmu NP

- jest na poziomie nieuniknionego błędu związanego z reprezentacją danych.
- będzie mały dla zadań dobrze uwarunkowanych
- może być duży dla zadań źle uwarunkowanych

Przykład: suma dwóch liczb

Zadanie obliczeniowe: $P(x_1, x_2) = x_1 + x_2$, gdzie $x_1, x_2 \in \mathbb{R}$.

Algorytm: $A(x_1, x_2) = x_1 + x_2$.

Czy jest NP?

$$fl(A(fl(x)) = fl\left(x_1(1+\epsilon_1) + x_2(1+\epsilon_2)\right) =$$

...gdzieś to już widzieliśmy...

$$= x_1 \underbrace{(1+\epsilon_1)(1+\delta)}_{=1+E_1} + x_2 \underbrace{(1+\epsilon_2)(1+\delta)}_{=1+E_2}$$

$$= x_1(1+E_1) + x_2(1+E_2) =: \tilde{x_1} + \tilde{x_2}$$

Przy czym

$$|E_i| \leqslant |\epsilon_i| + |\delta| \leqslant 2 \cdot \nu.$$

Przykład: suma dwóch liczb

Mamy więc

$$fl(A(fl(x)) = P(\tilde{x})$$

oraz

$$\|\tilde{x} - x\|_1 = \sum |\tilde{x}_i - x_i| = \sum |E_i x_i| \leqslant 2\nu \sum |x_i| = 2\nu \|x\|_1.$$

A więc jest NP! Dlaczego więc występuje redukcja cyfr?

Przykład: suma dwóch liczb

Zbadajmy uwarunkowanie zadania $P(x_1, x_2) = x_1 + x_2$.

$$P'(x_1, x_2) = [1, 1] \qquad \Longrightarrow \qquad \|P'(x)\|_1 = 1.$$

$$\mathrm{cond}_{\mathrm{rel}}(P, x) = \|P'(x)\|_1 \frac{\|x\|_1}{\|P(x)\|_1} = \frac{|x_1| + |x_2|}{|x_1 + x_2|}.$$

Zadanie sumy jest

- dobrze uwarunkowanie, gdy x_1, x_2 tego samego znaku,
- źle uwarunkowane, gdy $x_1 \approx -x_2$.

To złe uwarunkowanie jest przyczyną zjawiska redukcji cyfr przy odejmowaniu!

Numeryczna poprawność niektórych algorytmów

Zakładamy, że wszystkie dane, wyniki pośrednie i końcowe są liczbami maszynowymi znormalizowanymi.

- $+, \times, \div, \sqrt{}$, *FMA* — jeśli spełniają IEEE 754 — to są NP
- std algorytm rozw. układu z macierzą trójkątną $Ux = b$ jest NP: obliczony wynik $\tilde{x}$ spełnia $(U + \Delta)\tilde{x} = b$, gdzie $|\Delta_{ij}|/|u_{ij}| \leqslant N \cdot \nu$.

Numeryczna „poprawność" (?) GEPP

Twierdzenie

Wynik $\tilde{x}$ obliczony GEPP dla układu $Ax = b$ spełnia $(A + \Delta)\tilde{x} = b$,

$$\|\Delta\|_\infty / \|A\|_\infty \leqslant KN^3 \rho_N \nu,$$

gdzie $\rho_N = \max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}|$.

Dowód jest dość żmudny — pomijamy.

Niestety,

$$\rho_N \leqslant 2^{N-1} \qquad \text{(osiągane)}.$$

Dla wielu macierzy jest *znacznie lepiej*, dlatego mówimy, że GEPP jest *„praktycznie"* NP.

Numeryczne kryterium „numerycznej poprawności”

Rozważmy układ równań $Ax = b$. Niech $\tilde{x}$ — wynik obliczony w fl *jakimś* algorytmem. Czy jest prawdą, że

$$(A + \Delta)\tilde{x} = b + \delta, \qquad \text{przy czym } \frac{\|\Delta\|}{\|A\|}, \frac{\|\delta\|}{\|b\|} \leqslant \epsilon?$$

Twierdzenie

Jeśli

$$\frac{\|b - A\tilde{x}\|}{\|A\|\|\tilde{x}\| + \|b\|} \leqslant \epsilon,$$

to istnieją Δ, δ *t. że*

$$(A + \Delta)\tilde{x} = b + \delta \quad \textit{oraz} \quad \frac{\|\Delta\|}{\|A\|}, \frac{\|\delta\|}{\|b\|} \leqslant \epsilon.$$

Czy złożenie dwóch algorytmów NP jest NP?

Zadanie: $x \mapsto P(x) = y \mapsto Q(y) = z$, tzn. $z = Q \circ P(x)$.

Załóżmy, że

- zadanie P rozwiązujemy algorytmem A, który jest NP w całej dziedzinie
- zadanie Q rozwiązujemy algorytmem B, który jest NP w całej dziedzinie
- istnieje P^{-1}

Czy algorytm $B \circ A$ też jest NP? Niekoniecznie.

Kiedy złożenie dwóch algorytmów NP jest NP?

$x \mapsto P(x) = y \mapsto Q(y) = z$ (zadanie)
$x \mapsto A(x) = \tilde{y} \mapsto B(\tilde{y}) = \tilde{z}$ (algorytm, już w fl)

Przy czym

- $\tilde{y} = P(\tilde{x})$, $\|\tilde{x} - x\| \leqslant K\nu\|x\|$, bo A jest NP
- $\tilde{z} = Q(\hat{y})$, $\|\hat{y} - \tilde{y}\| \leqslant K\nu\|\tilde{y}\|$, bo B jest NP

Definiując $\hat{x} = P^{-1}(\hat{y})$ mamy

$$\tilde{z} = Q(\hat{y}) = Q(P(\hat{x})) = Q \circ P(\hat{x}).$$

Wystarczy sprawdzić, czy

$$\frac{\|\hat{x} - x\|}{\|x\|} \leqslant \hat{K}\nu?$$

Kiedy złożenie dwóch algorytmów NP jest NP?

$x \mapsto P(x) = y \mapsto Q(y) = z$
$x \mapsto A(x) = \tilde{y} \mapsto B(\tilde{y}) = \tilde{z} = Q(P(\hat{x}))$

Nietrudno policzyć, że

$$\|\hat{x} - x\| \lesssim K(1 + \mathrm{cond}_{\mathrm{rel}}(P^{-1}, \tilde{y})) \cdot \nu \cdot \|x\|.$$

Zatem $B \circ A$ jest NP, o ile P^{-1} jest dobrze uwarunkowane!

Liniowe zadanie najmniejszych kwadratów

Przekształcenie Householdera

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

Inaczej: $H = I - \gamma vv^T$, gdzie $\gamma = 2/\|v\|_2^2$. (Umowa: $v = 0 \implies H = I$)

Własności:

$$H = H^T \qquad \text{(symetria)}$$

Rzeczywiście:

$$H^T = (I - \gamma vv^T)^T = I^T - \gamma \underbrace{(vv^T)^T}_{(v^T)^T v^T} = I - \gamma vv^T = H.$$

$$H^T H = I \qquad \text{(ortogonalność)}$$

Rzeczywiście:

$$H^T H = HH = (I - \gamma vv^T)(I - \gamma vv^T) = I - 2\gamma vv^T + \gamma^2 v \underbrace{v^T v}_{=2/\gamma} v^T = I.$$

Przekształcenie Householdera

$$\|Hx\|_2 = \|x\|_2 \quad \forall x \qquad \text{(izometria)}$$

Rzeczywiście: $\|Hx\|_2^2 = (Hx)^T(Hx) = x^T \underbrace{H^T H}_{I} x = x^T x = \|x\|_2^2$.

- Reprezentacja H: wystarczy pamiętać wektor v (i kwadrat normy, $\|v\|_2^2$). Jeśli zrobisz to źle, pamięć rośnie do N^2 !
- Koszt wyznaczania $y = Hx$:

$$Hx = (I - 2\frac{vv^T}{v^T v})x = x - \frac{2}{\|v\|_2^2}(v^T x) \cdot v.$$

Czyli $\mathcal{O}(4N)$. Jeśli zrobisz to źle, koszt rośnie do $\mathcal{O}(N^2)$!

Przekształcenie Householdera

Dla zadanego $b \in \mathbb{R}^N$, szukamy $v \in \mathbb{R}^N$ takiego, by

$$Hb = \alpha \cdot e_1 = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Skoro $\|Hb\|_2 = \|b\|_2$, to $\alpha = \pm\|b\|_2$.

$$Hb = b - 2\frac{vv^T}{v^T v}b = b - \underbrace{2\frac{v^T b}{v^T v}}_{\beta} v,$$

czyli

$$\beta v = b - \alpha e_1.$$

Stąd (dlaczego?) $v = b - \alpha e_1 = b + \text{sign}(b_1)\|b\|_2 e_1$.

Koszt: $\mathcal{O}(N)$.

Rozkład QR macierzy

Zakładamy, że $A \in \mathbb{R}^{M\times N}$, $M \geqslant N$.

Rozkład wąski:

$$A = \hat{Q}\hat{R}$$

gdzie $\hat{Q} \in \mathbb{R}^{M\times N}$ ortogonalna $\hat{Q}^T\hat{Q} = I$ oraz $\hat{R} \in \mathbb{R}^{N\times N}$ — górna trójkątna.

Rozkład pełny:

$$A = \begin{bmatrix}\hat{Q} & \tilde{Q}\end{bmatrix}\begin{bmatrix}\hat{R}\\ 0\end{bmatrix} = QR$$

oraz $Q \in \mathbb{R}^{M\times M}$, $Q^TQ = I$, natomiast $R \in \mathbb{R}^{M\times N}$ — górna „trójkątna".

Jeśli $\operatorname{rank}(A) = N$ i ma rozkład QR jak powyżej, to macierz $\hat{R}$ musi być nieosobliwa.

Rozkład QR macierzy: istnienie i algorytm

$$A = \begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}, \qquad A_{11} \in \mathbb{R}^{p\times p}.$$

Niech

$$Q_1R_1 = Q_1\begin{bmatrix}\hat{R}_{11}\\ 0\end{bmatrix} = \begin{bmatrix}A_{11}\\ A_{21}\end{bmatrix}$$

Zatem

$$Q_1^TA = \begin{bmatrix}\hat{R}_{11} & R_{12}\\ 0 & \tilde{A}_{22}\end{bmatrix}$$

Niech dalej

$$Q_{22}R_{22} = \tilde{A}_{22},$$

Wtedy ostatecznie

$$A = \underbrace{Q_1Q_2}_{Q}\underbrace{\begin{bmatrix}\hat{R}_{11} & R_{12}\\ 0 & R_{22}\end{bmatrix}}_{R} \qquad \text{gdzie } Q_2 = \begin{bmatrix}I & \\ & Q_{22}\end{bmatrix}.$$

Rozkład QR macierzy: algorytm rekurencyjny ($p \approx N/2$)

$$QR = A = \begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}, \qquad A_{11} \in \mathbb{R}^{p\times p}, \qquad p \approx N/2.$$

Algorytm 2 $A = QR$ metodą Householdera (wersja rekurencyjna)

if $p = 1$ **then**

return $Q :=$ H m. Householdera t.że $H\cdot\begin{bmatrix}a_{11}\\ a_{21}\end{bmatrix} = \begin{bmatrix}r_{11}\\ 0\end{bmatrix} =: R$

end if

Rozłóż $Q_1R_1 = \begin{bmatrix}A_{11}\\ A_{21}\end{bmatrix}$ (rekurencja)

$\begin{bmatrix}R_{12}\\ A_{22}\end{bmatrix} := Q_1^T\begin{bmatrix}A_{12}\\ A_{22}\end{bmatrix}$

Rozłóż $Q_{22}R_{22} = A_{22}$ (rekurencja)

return $Q = Q_1\begin{bmatrix}I & \\ & Q_{22}\end{bmatrix}$, $R = \begin{bmatrix}R_{11} & R_{12}\\ 0 & R_{22}\end{bmatrix}$

Uwaga: czynników Q_i nie wymnażamy!

QR iteracyjnie metodą Householdera ($p = 1$)

Algorytm 3 $A = QR$ metodą Householdera (wersja z iteracją)

for $k = 1 : N$

Podziel $A = \begin{bmatrix}a_{11} & a_{12}\\ a_{21} & A_{22}\end{bmatrix}$, $\quad a_{11} \in \mathbb{R}$.

$Q_k :=$ m. Householdera t.że $H\cdot\begin{bmatrix}a_{11}\\ a_{21}\end{bmatrix} = \begin{bmatrix}r_{kk}\\ 0\end{bmatrix}$;

$\begin{bmatrix}r_{12}\\ A_{22}\end{bmatrix} := Q_k^T\begin{bmatrix}a_{12}\\ A_{22}\end{bmatrix}$

$A := A_{22}$

end

return $Q = Q_1\begin{bmatrix}1 & \\ & Q_2\end{bmatrix}\cdots\begin{bmatrix}I_{k-1} & \\ & Q_k\end{bmatrix}\cdots$, $R = \begin{bmatrix}r_{11} & r_{12}\\ 0 & \ddots\end{bmatrix}$

Uwaga: czynników Q_i nie wymnażamy!

Koszt rozkładu QR (iteracyjnie metodą Householdera)

$$T(M,N) = \underbrace{K_1 \cdot M}_{\text{wyzn. } H} + \underbrace{4 \cdot M \cdot (N-1)}_{\text{obl. } H \cdot \begin{bmatrix} a_{12} \\ A_{22} \end{bmatrix}} + \underbrace{T(M-1, N-1)}_{\text{pozost. kroki}}$$

Zatem sumarycznie

$$T(M,N) = 4\sum_{k=1}^{N}(N-k)(M-k+1) + K_1 \sum_{k=1}^{N}(M-k+1) \approx 2MN^2 - \frac{2}{3}N^3$$

(z pominięciem wyrazów rzędu niższego niż 3.)

Numeryczna poprawność algorytmu rozkładu QR (iteracyjnie metodą Householdera)

Twierdzenie

Niech $\tilde{R}$ będzie macierzą trójkątną wyznaczoną w fl algorytmem rozkładu QR (iteracyjnie, z metodą Householdera). Istnieje ortogonalna $Q = H_1 \cdots H_N$ taka, że

$$A + \Delta = Q \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix}, \qquad \text{oraz } \|\Delta_j\|_2 \leqslant K \cdot MN \|a_j\|_2 \cdot \nu,$$

przy czym H_k — macierz Householdera wyznaczona dokładnie dla podmacierzy obliczonej w fl na k–tym kroku algorytmu.

Dowód pomijamy.

Dygresja: jak używać wyniku

W wyniku dostajemy nie $A = QR$, ale

$$A = \underbrace{Q_1 \cdot Q_2 \cdots Q_N}_{=Q} \cdot R, \qquad Q_i \in \mathbb{R}^{M \times M}.$$

Jak obliczać wyrażenia postaci

$$X = Q \cdot B?$$

Źle:

$Q = Q_1 \cdot Q_2 \cdots Q_N$???
$X = Q \cdot B$

Dobrze:

$X = B$
for $i = N : -1 : 1$
  $X = Q_i \cdot X$
end

...dodatkowo wykorzystując postać Q_i!

Liniowe zadanie najmniejszych kwadratów

Problem (LZNK)

Niech $A \in \mathbb{R}^{M \times N}$, $M \geqslant N$ oraz $\operatorname{rank}(A) = N$ i niech $b \in \mathbb{R}^M$.

Znaleźć $x \in \mathbb{R}^N$ taki, że

$$\|b - Ax\|_2 \to \min!$$

tzn.

$$\|b - Ax\|_2 \leqslant \|b - Ay\|_2 \qquad \forall y$$

Rozwiązywanie LZNK przez rozkład QR

Jeśli znamy rozkład QR macierzy A pełnego rzędu,

$$A = QR = \begin{bmatrix}\hat{Q} & \tilde{Q}\end{bmatrix}\begin{bmatrix}\hat{R}\\ 0\end{bmatrix}$$

zadanie staje się banalne:

$$\|b - Ax\|_2^2 = \|b - QRx\|_2^2 = \|Q(Q^T b - Rx)\|_2^2$$
$$= \|Q^T b - Rx\|_2^2 = \| \begin{bmatrix}\hat{Q}^T\\ \tilde{Q}^T\end{bmatrix} b - \begin{bmatrix}\hat{R}\\ 0\end{bmatrix} x\|_2^2$$
$$\| \begin{bmatrix}\hat{Q}^T b - \hat{R}x\\ \tilde{Q}^T b\end{bmatrix} \|_2^2 = \underbrace{\|\hat{Q}^T b - \hat{R}x\|_2^2}_{\text{najmniejsze, gdy } \hat{R}x = \hat{Q}^T b} + \|\tilde{Q}^T b\|_2^2$$

Algorytm 4 Rozwiązywanie LZNK

Wyznacz rozkład QR macierzy A	▷ koszt: $\mathcal{O}(MN^2)$
Oblicz $\hat{b} := \hat{Q}^T b$	▷ koszt: $\mathcal{O}(MN)$
Rozwiąż $\hat{R}x = \hat{b}$	▷ koszt: $\mathcal{O}(N^2)$
return x	

Dygresja: układ równań liniowych przez rozkład QR

Jeśli A — kwadratowa nieosobliwa,

$$A = QR,$$

- Q — ortogonalna, $Q^T Q = I$
- R — górna trójkątna

Układ równań:

$$Ax = b \qquad \Longrightarrow \qquad Q\underbrace{Rx}_{=:y} = b,$$

skąd algorytm:

1. $y = Q^T b$ (koszt: $\mathcal{O}(N^2)$)
2. Rozwiąż układ z macierzą trójkątną $Rx = y$. (koszt: $\mathcal{O}(N^2)$)

Ale rozkład QR wymaga ok. 2 razy więcej flopów niż rozkład LU.

Liniowe zadanie najmniejszych kwadratów — układ normalny

Skoro

$$x = \hat{R}^{-1}\hat{Q}^T b,$$

to mnożąc przez (nieosobliwą) $\hat{R}^T\hat{R}$ mamy

$$\hat{R}^T\hat{R}x = \hat{R}^T\hat{Q}^T b = \begin{bmatrix}\hat{R}^T & 0\end{bmatrix}\begin{bmatrix}\hat{Q}^T\\ \tilde{Q}^T\end{bmatrix} b = R^T Q^T b = A^T b.$$

Z drugiej strony, $\hat{R}^T\hat{R} = \hat{R}^T\underbrace{\hat{Q}^T\hat{Q}}_{I}\hat{R} = A^T A$, więc ostatecznie x jest rozwiązaniem **układu równań normalnych**

$$A^T Ax = A^T b.$$

Liniowe zadanie najmniejszych kwadratów — układ normalny

$$A^T Ax = A^T b.$$

Własności $A^T A$:

- mała ($N \times N$)
- symetryczna
- dodatnio określona (bo A jest pełnego rzędu),

Stąd alternatywny algorytm rozwiązania LZNK:

Oblicz $B = A^T A$ (koszt MN^2)
Wyznacz rozkład Cholesky'ego $B = LL^T$ (koszt $\mathcal{O}(N^3)$)
Rozwiąż układ $LL^T x = A^T b$ (koszt $\mathcal{O}(MN)$)

Jest ok. dwa razy taniej, ale ten sposób może mieć gorsze własności numeryczne (od metody wykorzystującej rozkł. QR macierzy A)

Rozkład SVD

Twierdzenie (rozkład w/g wartości szczególnych)

Niech $A \in \mathbb{R}^{M\times N}$, $M \geqslant N$. Istnieje rozkład

$$A = U\Sigma V^T :$$

- $U \in \mathbb{R}^{M\times N}$ *ortogonalna,* $U^TU = I$
- $V \in \mathbb{R}^{N\times N}$ *ortogonalna,* $V^TV = I$
- $\Sigma \in \mathbb{R}^{N\times N}$ *diagonalna,* $\Sigma = diag(\sigma_1, \ldots, \sigma_N)$, $\sigma_1 \geqslant \cdots \geqslant \sigma_N \geqslant 0$.

Gdy $M < N$, rozkład SVD definiujemy dla macierzy A^T.
Kolumny U, V: lewe/prawe wektory szczególne.
σ_i: wartości szczególne.

Rozkład SVD

- $A = U\Sigma V^T$ — rozkład wąski
- $A = \underbrace{\begin{bmatrix} U & \tilde{U} \end{bmatrix}}_{\text{ortog. } M\times M} \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T$ — rozkład pełny

Rozkład SVD jest kosztowny.

- Jest trudniejszy od zadania własnego, które jest trudniejsze od układu równań liniowych
- Wymaga procesu iteracyjnego, który dopiero w granicy jest zbieżny do rozwiązania
- W praktyce, ze względu na ograniczenie precyzji wyniku w fl, wykonuje się skończoną liczbę iteracji.
  Koszt $\approx \mathcal{O}(MN^2) + \mathcal{O}(N^3)$ (stałe kilkakrotnie większe niż dla rozkładu QR)

Rozkład SVD — własności

Twierdzenie

1. *Wartości własne A^TA są równe $\sigma_1^2, \ldots, \sigma_N^2$.*
2. *Jeśli $\sigma_1 \geqslant \cdots \geqslant \sigma_r > \sigma_{r+1} = \ldots = \sigma_N = 0$, to rząd A jest równy r.*
3. *$A = \sum_{i=1}^N \sigma_i u_i v_i^T$, gdzie u_i, v_i — kolumny U, V.*
4. ***Tw. Eckarta–Younga***[1] *Macierz rzędu $\leqslant k \leqslant N$ najbliższa A w normie $\|\cdot\|_2$ lub $\|\cdot\|_F$ to $A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$.*

Dowód: Na tablicy. □

[1] A naprawdę Schmidta–Mirsky'ego.

Rozkład SVD — zastosowania

- Wyznaczenie rzędu macierzy
- PCA, używane w uczeniu maszynowym
- Rozwiązanie LZNK, gdy A jest pełnego rzędu[2]: Skoro x spełnia

  $$A^TAx = A^Tb$$

  to $A^TA = (V\Sigma \underbrace{U^T)(U}_{I}\Sigma V^T) = V\Sigma^2V^T$, więc

  $$A^TAx = A^Tb \iff$$
  $$V\Sigma^2V^Tx = V\Sigma U^Tb \iff V^Tx = \Sigma^{-1}U^Tb \iff$$
  $$x = V\Sigma^{-1}U^Tb$$

  (Σ nieosobliwa, bo $\operatorname{rank}(A) = N$.)
- Rozwiązanie LZNK, gdy A *nie jest* pełnego rzędu: o tym za chwilę.

[2] Ale taniej użyć rozkładu QR!

Uwarunkowanie LZNK

Twierdzenie

Niech $A \in \mathbb{R}^{M\times N}$, $M \geqslant N$ *oraz* $\operatorname{rank}(A) = N$ *i niech* $b \in \mathbb{R}^M$.

$$\underbrace{\|b - Ax\|_2 \to \min!}_{\text{zad. dokładne}} \qquad \underbrace{\|(b+\delta) - (A+\Delta)x\|_2 \to \min!}_{\text{zad. zaburzone}}$$

Niech $\epsilon = \max\{\frac{\|\Delta\|_2}{\|A\|_2}, \frac{\|\delta\|_2}{\|b\|_2}\}$, $\operatorname{cond}_2(A)\epsilon < 1$. *Wtedy zadanie zaburzone ma dokładnie jedno rozwiązanie* $\tilde{x}$ *oraz*

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} \leqslant \epsilon \underbrace{\left(2\frac{\operatorname{cond}_2(A)}{\cos\theta} + \operatorname{tg}\theta \cdot \operatorname{cond}_2^2(A)\right)}_{=:\operatorname{cond}_{LZNK}(A,b)} + \mathcal{O}(\epsilon^2),$$

gdzie $\sin\theta = \frac{\|b - Ax\|_2}{\|b\|_2}$, $\operatorname{cond}_2(A) := \frac{\max\sigma_i}{\min\sigma_i}$.

Uwarunkowanie LZNK

$$\operatorname{cond}_{LZNK}(A,b) = 2\frac{\operatorname{cond}_2(A)}{\cos\theta} + \operatorname{tg}\theta \cdot \operatorname{cond}_2^2(A)$$

gdzie $\sin\theta = \frac{\|b - Ax\|_2}{\|b\|_2}$, $\operatorname{cond}_2(A) := \frac{\max\sigma_i}{\min\sigma_i}$.

- **przypadek typowy**: $\sin\theta \approx 0$ (mała reszta)
  $\implies \operatorname{cond}_{LZNK}(A,b) \approx \operatorname{cond}_2(A)$
- $\sin\theta \approx 1$ (duża reszta i $x \approx 0$) $\implies \operatorname{cond}_{LZNK}(A,b) \gg 1$
- pozostałe: $\implies \operatorname{cond}_{LZNK}(A,b) \approx \operatorname{cond}_2^2(A)$

Zauważmy, że $\operatorname{cond}_2(A^TA) = \operatorname{cond}_2^2(A)$ — jeszcze jeden powód by r-ń normalnych używać z rozwagą.

Nieregularne liniowe zadanie najmniejszych kwadratów

Problem (nieregularne LZNK)

$A \in \mathbb{R}^{M\times N}$, $M \geqslant N$, ale $\operatorname{rank}(A) < N$.

Szukamy $x \in \mathbb{R}^N$ t.że

$$\|b - Ax\|_2 \to \min!$$

Skoro $\ker(A) \neq 0$, to rozwiązanie nieregularnego LZNK nie jest jednoznaczne:

x minimalizuje $\|b - Ax\|_2 \implies x + z$, gdzie $z \in \ker(A)$, też minimalizuje...

Nieregularne LZNK — rozwiązanie przez SVD

Niech $A = U\begin{bmatrix}\Sigma\\0\end{bmatrix}V^T$, przy czym $U = \begin{bmatrix}u_1 & \dots & u_M\end{bmatrix}$, $V = \begin{bmatrix}v_1 & \dots & v_N\end{bmatrix}$. Mamy

$$\|b - Ax\|_2^2 = \|b - U\Sigma V^Tx\|_2^2 = \|U(U^Tb - \Sigma V^Tx)\|_2^2$$

$$= \|U^Tb - \Sigma\underbrace{V^Tx}_{=:y}\|_2^2 = \left\|\begin{bmatrix}u_1^Tb\\\vdots\\u_r^Tb\\\vdots\\u_M^Tb\end{bmatrix} - \begin{bmatrix}\sigma_1y_1\\\vdots\\\sigma_ry_r\\0\\\vdots\end{bmatrix}\right\|_2^2$$

$$= \sum_{i=1}^{r}(\sigma_iy_i - u_i^Tb)^2 + \sum_{i=r+1}^{M}(u_i^Tb)^2$$

Nieregularne LZNK — rozwiązanie przez SVD

$$\|b - Ax\|_2^2 = \sum_{i=1}^{r}(\sigma_i y_i - u_i^T b)^2 + \sum_{i=r+1}^{M}(u_i^T b)^2$$

Czyli $x = Vy$ minimalizuje resztę wtw gdy

$$y_i = \frac{u_i^T b}{\sigma_i}, \quad i = 1, \ldots, r. \qquad \text{(pozostałe } y_i \text{ — dowolne)}$$

Wniosek

Rozwiązanie x^* o najmniejszej normie, nieregularnego LZNK $\|b - Ax\|_2 \to \min$! (gdzie $\operatorname{rank}(A) = r$) jest jednoznacznie określone i dane wzorem $x^* = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i$.

Dwd: Rzeczywiście, $\|x^*\|_2 = \|Vy\|_2 = \|y\|_2$. A przecież $\|y\|_2$ jest minimalna, bo wybieramy $y_{r+1} = \ldots = y_M = 0$.

Metody oparte na rozszczepieniu macierzy

Niech $A = M - Z$, przy czym M — nieosobliwa.

$$Ax^* = b \iff (M-Z)x^* = b \iff Mx^* = b+Zx^* \iff x^* = M^{-1}(b+Zx^*).$$

Rozwiązanie x^* spełnia więc

$$x^* = \underbrace{B}_{M^{-1}Z} \cdot x^* + \underbrace{c}_{M^{-1}b}$$

Możemy więc spróbować użyć iteracji prostej:

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

Jak dobrać M, by

- $x_k \to x^*$ niezależnie od x_0
- krok iteracji był niedrogi?

Metody iteracyjne dla układów równań liniowych

Zbieżność metod rozszczepienia

Niech $x^* = Bx^* + c$.

Twierdzenie (o zbieżności metody rozszczepienia)

Ciąg

$$x_{k+1} = Bx_k + c$$

jest zbieżny do x^* dla każdego x_0 $\iff$ $\rho(B) < 1$,

gdzie

$$\rho(B) = \max\{|\lambda| : \lambda \text{ jest w.wł } B\}$$

to promień spektralny macierzy B.

Zbieżność metod rozszczepienia

Twierdzenie (o liniowej zbieżności metody rozszczepienia)

Jeśli $\|B\| < 1$, to ciąg $x_{k+1} = Bx_k + c$ jest zbieżny do x^* dla każdego x_0 oraz

$$\|x_{k+1} - x^*\| \leqslant \|B\| \cdot \|x_k - x^*\|.$$

Dowód:

$$x_{k+1} - x^* = (Bx_k + c) - (Bx^* + c) = B(x_k - x^*),$$

skąd

$$\|x_{k+1} - x^*\| = \|B(x_k - x^*)\| \leqslant \|B\|\|x_k - x^*\|.$$

Przez indukcję,

$$\|x_k - x^*\| \leqslant \|B\|^k \|x_0 - x^*\| \to 0.$$

□

Zbieżność metod rozszczepienia

Wniosek (metody oparte na rozszczepieniu)

Niech $A = M - Z$ oraz A, M — nieosobliwe. $Ax^* = b$.

- *Jeśli $\rho(M^{-1}Z) < 1$, to metoda iteracyjna*

$$Mx_{k+1} = Zx_k + b$$

  jest zbieżna do x^* z każdego x_0.
- *Jeśli dodatkowo $\gamma := \|M^{-1}Z\| < 1$, to dodatkowo*

$$\|x_{k+1} - x^*\| \leqslant \gamma \cdot \|x_k - x^*\|.$$

Przykłady metod opartych na rozszczepieniu

Niech $A = L + D + U$, $D = \mathrm{diag}(A)$,
L (odp. U) — dolna (odp. górna) trójkątna z zerową diagonalą.

$$x_{k+1} = x_k + M^{-1}(b - Ax_k)$$

- Metoda **Jacobiego**:
$$M = D,$$
- Metoda **Gaussa–Seidela**:
$$M = D + L,$$
- Metoda **SOR**:
$$M = \frac{1}{\omega}D + L.$$
W niektórych przypadkach znamy optymalne ω.

Są to metody **algebraiczne**: wymagają dostępu do elementów macierzy.

Metoda Jacobiego

$$x_{k+1} = x_k + D^{-1}(b - Ax_k)$$

Twierdzenie

Jeśli A jest diagonalnie dominująca,

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \qquad \forall i = 1, \ldots, N,$$

to m. Jacobiego jest zbieżna do x^* dla każdego x_0.

Dowód: Wyraz (i, j) macierzy $M^{-1}Z$ wynosi 0 dla $i = j$ oraz $-a_{ij}/a_{ii}$ dla $i \neq j$, więc

$$\|M^{-1}Z\|_\infty = \max_i \sum_{j \neq i} |a_{ij}|/|a_{ii}| \; < \; 1.$$

□

Metody projekcji

W naszych metodach,

$$x_{k+1} = x_k + \delta_k.$$

Jak wybierać δ_k? Idealna poprawka, δ_k^*, spełniałaby:

$$A\delta_k^* = r_k \implies x_{k+1} = x^*$$

Wyznaczmy *przybliżoną* poprawkę idealną: $\delta_k = V_k a_k$, gdzie $V_k, U_k \in \mathbb{R}^{N\times r}$ max. rzędu t. że $a_k \in \mathbb{R}^r$ spełnia

$$\underbrace{U_k^T A V_k}_{r\times r} a_k = U_k^T r_k.$$

Metody projekcji — przykłady

Metoda najszybszego spadku dla $A = A^T > 0$:

$$U_k = V_k = r_k \implies a_k = \frac{r_k^T r_k}{r_k^T A r_k}$$

$$x_{k+1} = x_k + a_k r_k$$

Twierdzenie (o zbieżności metody najszybszego spadku)

Jeśli $A = A^T > 0$, *to*

$$\|x^* - x_{k+1}\|_A \leqslant \frac{\kappa - 1}{\kappa + 1}\|x^* - x_k\|_A,$$

gdzie $\kappa = \mathrm{cond}_2(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$, $\|x\|_A^2 := x^T A x$.

Metody projekcji — przykłady

Metoda najmniejszego residuum

$$V_k = r_k, \qquad U_k = AV_k = Ar_k.$$

$$a_k = \frac{r_k^T A^T r_k}{r_k^T A^T A r_k},$$
$$x_{k+1} = x_k + a_k r_k.$$

Twierdzenie

Załóżmy, że macierz $A_{\mathrm{sym}} = (A + A^T)/2$ *jest dodatnio określona i oznaczmy* $\mu = \lambda_{\min}(A_{\mathrm{sym}}) > 0$. *Wtedy*

$$\|r_{k+1}\|_2 \leqslant \left(1 - \frac{\mu^2}{\|A\|_2^2}\right)^{1/2} \|r_k\|_2.$$

Metody przestrzeni Kryłowa

Niech $r_k = b - Ax_k$.

Zdefiniujmy **przestrzeń Kryłowa** $K_k = \mathrm{lin}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$. Oczywiście,

$$K_0 \subseteq K_1 \subseteq \ldots \subseteq K_k \subseteq K_{k+1} \subseteq \ldots \subseteq \mathbb{R}^N.$$

Definiujemy x_k jako rozwiązanie zadania minimalizacji

$$\|x - x^*\| = \min! \qquad \text{albo} \qquad \|b - Ax\| = \min!$$

dla $x \in x_0 + K_k$.

Wybór normy i minimalizowanej wielkości determinuje konkretną metodę iteracyjną (są też inne m. Kryłowa).

Będą to metody **operatorowe**: wystarczy dostęp do procedury obliczania $y := A \cdot x$.

Metoda CG (gradientów sprzężonych)

Dla $A = A^T > 0$.
Iteracja x_k zdefiniowana przez $\|x - x^*\|_A = \min!$, tzn.

$$\|x_k - x^*\|_A \leqslant \|x - x^*\|_A \quad \forall x \in x_0 + K_k.$$

Jest to LZNK: Gdy kolumny V_k stanowią bazę K_k, to $x_k = x_0 + V_k a$ i

$$\|x_k - x^*\|_A^2 = \|A^{1/2}(x_k - x^*)\|_2^2 = \|A^{1/2}(V_k a + x_0 - x^*)\|_2^2,$$

Układ normalny to $V_k^T A V_k a = V_k^T A(x^* - x_0) = V_k^T r_0$. Można go bardzo tanio rozwiązywać. (Jeśli zrobisz to źle, koszt k-tej iteracji wyniesie $\mathcal{O}(N \cdot k^2)$ flopów plus $N \times k$ pamięci.)

Metoda CG — implementacja

Wyprowadzenie tego algorytmu jest nietrywialne.

$r = b - Ax$;
$\rho_0 = \|r\|_2^2$, $\beta = 0$, $k = 1$
while not stop **do**
  $p = r + \beta p$
  $w = Ap$
  $\alpha = \rho_{k-1} / p^T w$
  $x = x + \alpha p$
  $r = r - \alpha w$
  $\rho_k = \|r\|_2^2$; $\beta = \rho_k / \rho_{k-1}$
  $k = k + 1$
end while

Kosz jednej iteracji: $\underbrace{\text{SPMV}(A, x)}_{=2 \cdot \text{nnz}(A)} + \mathcal{O}(N)$. Pamięć: $\mathcal{O}(N)$.

Metoda CG — zbieżność

Twierdzenie (o zbieżności CG jako metody iteracyjnej)

Po k iteracjach metody CG,

$$||x_k - x^*||_A \leqslant 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k ||x_0 - x^*||_A,$$

gdzie $\kappa = \text{cond}_2(A) = \lambda_{\max}(A) / \lambda_{\min}(A)$.

Metody przestrzeni Kryłowa

Zostały uznane za jedno z 10 najważniejszych osiągnięć algorytmicznych XX wieku.

Metoda GMRES

Iteracja x_k zdefiniowana przez $\|r\|_2 = \min!$, tzn.

$$\|r_k\|_2 = \|b - Ax_k\|_2 \leqslant \|b - Ax\|_2 \quad \forall x \in x_0 + K_k.$$

Inteligentnie zaimplementowana m. GMRES ma koszt k-tej iteracji

- $\mathcal{O}(N \cdot k)$ flopów (trzeba rozwiązywać LZNK, aczkolwiek dość tanie)
- $\mathcal{O}(N \cdot k)$ pamięci (trzeba pamiętać bazę p-ni Kryłowa)

Ściskanie macierzy

$$A \cdot x = b$$

Przykry fakt

Szybkość zbieżności m. iteracyjnych zależy od własności spektralnych A, np. CG zależy od $\mathrm{cond}_2(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$.

Pomysł: Zadanie równoważne (M nieosobliwa):

$$\underbrace{MA}_{\text{nowa macierz}} \cdot x = Mb$$

nowa macierz może mieć *znacznie lepsze* własności!

Ściskanie macierzy

Wymagania:

- MA ma **korzystniejsze własności spektralne** z punktu używanej metody iteracyjnej,
- M jest **łatwa w „konstrukcji”**,
- M jest **tania w mnożeniu przez wektor**.

Kilka nietrafionych wyborów:

- $M = A^{-1}$??
- $M = I$??

Lepsze wybory („$M \approx A^{-1}$, ale tania”):

- jeden (kilka?) kroków prostej metody iteracyjnej
- niepełny rozkład LU, itp.
- najlepiej: dobrać specjalnie do konkretnego zadania!

Sposoby reprezentacji macierzy rzadkich

Najbardziej popularne formaty:

- współrzędnych
- CSR/CSC

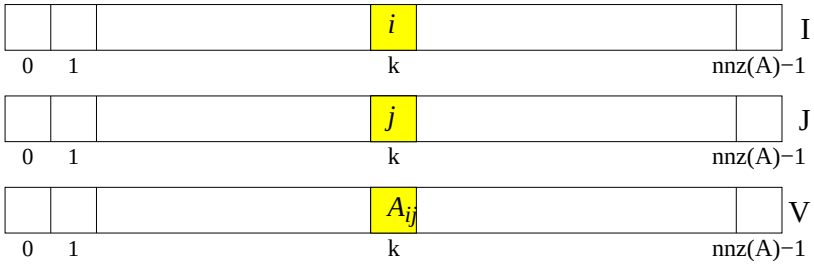
Format współrzędnych

Macierz A rozmiaru $N \times M$, licząca $\mathrm{nnz}(A)$ niezerowych elementów.

- V — tablica typu **double**,
- I, J — tablice typu **int**;

Wszystkie o długości $\mathrm{nnz}(A)$, przy czym zachodzi

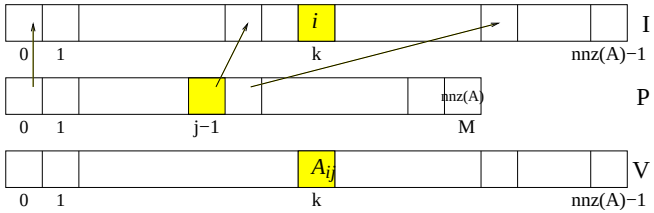
$$A_{I[k],J[k]} = V[k], \qquad k = 0, \ldots, \mathrm{nnz}(A) - 1,$$



Formaty CSR/CSC

Zajmiemy się **CSC = *Compressed Sparse Column***.
Analogicznie działa CSR = *Compressed Sparse Row*.

- V — tablica typu **double**, kolejne el-ty A kolumnami
- I — tablica typu **int**, długości nnz(A)
- P — tablica typu **int**, długości $M+1$



P[0]=0, P[M]= nnz(A).

$A_{I[k],\,j} = V[k], \qquad k = P[j-1], \ldots, P[j]-1.$

Zagadnienie własne

Zagadnienie własne — zadania obliczeniowe

Problem

Znaleźć $v \in \mathbb{C}^N$ oraz $\lambda \in \mathbb{C}$ t. że

$$Av = \lambda v, \qquad \|v\| = 1.$$

W praktyce często spotyka się następujące wersje tego zadania:

- znaleźć **największą/najmniejszą** (co do modułu) wartość własną (i ew. odp. wektor własny)
- znaleźć wartość własną **bliską zadanej liczbie** (i ew. odp. wektor własny)
- znaleźć **wszystkie** wartości własne (i ew. odp. wektory własne)

Przypomnienie z GAL-u

- Każda macierz $A \in \mathbb{R}^{N\times N}$, **symetryczna** ma rozkład

$$A = Q\Lambda Q^T,$$

gdzie $Q \in \mathbb{R}^{N\times N}$ — macierz ortogonalna, $Q^TQ = I$,

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}$$

$\lambda_i \in \mathbb{R}$ — wartości własne A.
Kolumny Q — wektory własne A.

Metoda potęgowa

Problem

Zakładamy, że $|\lambda_1| > |\lambda_2| \geqslant \ldots \geqslant |\lambda_N|$.
Wyznaczyć parę własną (λ_1, v_1).

Wybieramy (losowy?) x_0.
while not stop **do**
$x_{k+1} = Ax_k$
$x_{k+1} = x_{k+1}/\|x_{k+1}\|$
$k = k + 1$
end while

Ta metoda legła u podstaw PageRank. To, że $\lambda = 1$ jest jedyną *dominującą* w.wł., wynika z tw. Perrona.

Metoda potęgowa

Pomijając normowanie, mamy:

$$x_k = A \cdot x_{k-1} = A^2 \cdot x_{k-2} = \ldots = A^k \cdot x_0.$$

Twierdzenie

Jeśli $x_0^T v_1 \neq 0$, to x_k dąży do kierunku v_1, gdy $k \to \infty$.

Dowód: Tylko dla przypadku, gdy istnieje baza ortogonalna złożona z wektorów własnych A. Zatem $x_0 = \sum_i \alpha_i v_i$, $\alpha_1 \neq 0$ oraz
$x_k = A^k x_0 = \sum_i \lambda_i^k \alpha_i v_i = \lambda_1^k (\alpha_1 v_1 + \sum_{i>1} \underbrace{\left(\frac{\lambda_i}{\lambda_1}\right)^k}_{\downarrow 0} \alpha_i v_i)$.

Gdy $|\lambda_2|/|\lambda_1| \approx 1$, zbieżność może być wolna. □

Błąd zaokrągleń pomaga: nawet jeśli $x_0^T v_1 = 0$, to wskutek fl składowa v_1 ma szansę się pojawić w trakcie obliczeń.

Iloraz Rayleigh

Problem

Mając przybliżony wektor własny v, jak dobrać dla niego λ?

Zminimalizować resztę,

$$\|Av - \lambda v\|_2 \to \min!$$

To jest LZNK, a rozwiązaniem jest

$$\lambda = \frac{x^H A x}{x^H x} \qquad \text{(iloraz Rayleigh)}$$

Dowód: Na tablicy. □

Transformacje spektrum

Jeśli pary własne A to (λ_i, v_i), to wtedy

Macierz	wart. wł.	wekt. wł.	zastrz.
A	λ_i	v_i	
$A - \mu I$	$\lambda_i - \mu$	v_i	
A^{-1}	$\frac{1}{\lambda_i}$	v_i	A nieosobl.
$(A - \mu I)^{-1}$	$\frac{1}{\lambda_i - \mu}$	v_i	$A - \mu I$ nieosobl.

Dowód: Na tablicy. □

Odwrotna metoda potęgowa

Problem

Zakładamy, że $|\lambda_1| \geqslant |\lambda_2| \geqslant \ldots \geqslant |\lambda_{N-1}| > |\lambda_N| > 0$.
Wyznaczyć parę własną (λ_N, v_N).

Wartości własne A^{-1}:

$$\frac{1}{|\lambda_N|} > \frac{1}{|\lambda_{N-1}|} \geqslant \ldots \geqslant \frac{1}{|\lambda_1|}.$$

Metoda potęgowa dla A^{-1}, czyli **odwrotna metoda potęgowa**:

Wyznacz rozkład, np. $PA = LU$
while not stop **do**
  Rozwiąż $LUx_{k+1} = Px_k$
  $x_{k+1} = x_{k+1}/\|x_{k+1}\|$
  $k = k + 1$
end while

Jeśli zrobisz to źle, koszt jednej iteracji rośnie N-krotnie.

Wyznaczanie wartości własnej najbliższej zadanej liczbie

Problem

Dla zadanej liczby μ szukamy w. własnej λ^* takiej, że $|\lambda^* - \mu| \leqslant |\lambda_i - \mu|$ dla każdego i.

Wartości własne $(A - \mu)^{-1}$: $\dfrac{1}{|\lambda_i - \mu|}$.

Metoda potęgowa dla $(A - \mu I)^{-1}$, czyli **odwrotna metoda potęgowa z przesunięciem**:

Wyznacz rozkład, np. $P(A - \mu I) = LU$
while not stop **do**
  Rozwiąż $LUx_{k+1} = Px_k$
  $x_{k+1} = x_{k+1}/\|x_{k+1}\|$
  $k = k + 1$
end while
$\lambda^* = x_k^H A x_k$ ▷ $x_k^H x_k = 1$

Jeśli zrobisz to źle, koszt jednej iteracji rośnie N-krotnie.

RQI

Zbieżność odwrotnej metody potęgowej z przesunięciem można przyspieszyć.

Zamiast:

Odwrotna metoda potęgowa (ze stałym przesunięciem)
Wybieramy x_0.
while not stop **do**
  Rozwiąż $(A - \mu I)x_{k+1} = x_k$
  $x_{k+1} = x_{k+1}/\|x_{k+1}\|$
  $k = k + 1$
end while
$\lambda^* = x_k^H A x_k$

„Poprawiamy" μ:

Metoda RQI
Wybieramy x_0. $\lambda_0 = \mu$
while not stop **do**
  Rozw. $(A - \lambda_k I)x_{k+1} = x_k$
  $x_{k+1} = x_{k+1}/\|x_{k+1}\|$
  $\lambda_{k+1} = x_{k+1}^H A x_{k+1}$
  $k = k + 1$
end while

RQI = Rayleigh Quotient Iteration

Kłopot: nie mamy gwarancji, że zbiegniemy do najbliższej μ.

Pełne zagadnienie własne

Problem

Znaleźć wszystkie wartości własne (i ew. wektory własne) A.

Co do zasady, **nie należy** tego robić przez wyznaczenie, a potem znalezienie miejsc zerowych *wielomianu charakterystycznego* Warto wcześniej sprowadzić A do prostszej postaci.

- Można skonstruować sekwencję przekształceń ortog. Q_i, które kosztem $\mathcal{O}(N^3)$ sprowadzą A do postaci **Hessenberga**:

$$Q_{N-1} \cdots Q_1 A Q_1^T \cdots Q_{N-1}^T = \begin{bmatrix} * & * & * & * & \cdots & * \\ * & * & * & * & \cdots & * \\ & * & * & * & \cdots & * \\ & & \ddots & \ddots & \cdots & \vdots \\ & & & \ddots & \ddots & * \\ & & & & * & * \end{bmatrix}$$

- Gdy $A = A^T$, postać Hessenberga jest trójdiagonalna (i symetryczna).

Metoda dziel i rządź

Pełne zadanie własne dla macierzy **symetrycznej** $A = Q\Lambda Q^T$

Zakładamy, że A już została sprowadzona do postaci trójdiagonalnej.

$$A = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{N-1} \\ & & b_{N-1} & a_N \end{bmatrix} = \begin{bmatrix} T_1 & \\ & T_2 \end{bmatrix} + b_m u u^T,$$

gdzie T_1, T_2 trójdiagonalne i symetryczne, $m \approx N/2$.

$u = e_m + e_{m+1} = [0, \cdots, 0, 1, 1, 0, \cdots]^T$

$b_m u u^T$ ma tylko cztery niezerowe elementy, każdy równy b_m.

Metoda dziel i rządź (idea)

$$A = \begin{bmatrix} T_1 & \\ & T_2 \end{bmatrix} + b_m u u^T$$

Dziel: załóżmy, że już rozwiązaliśmy zadanie dla T_i: $Q_i^T T_i Q_i = D_i$, $i = 1, 2$. **Rządź:** Wtedy

$$\begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}^T \left(\begin{bmatrix} T_1 & \\ & T_2 \end{bmatrix} + b_m u u^T \right) \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} = \underbrace{\begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix}}_{=D,\ \text{diagonalna}} + b_m v v^T.$$

Jeśli $\lambda \notin \sigma(D)$, to wartości własne λ macierzy $D + b_m v v^T$ spełniają równanie

$$f(\lambda) \equiv 1 + b_m \sum_{j=1}^{N} \frac{v_j^2}{d_j - \lambda} = 0.$$

Ostateczny koszt: $O(N^3)$ z małą stałą.

Deflacja

Stwierdzenie

Jeśli

$$A = \begin{bmatrix} T_{11} & T_{12} \\ & T_{22} \end{bmatrix}$$

i T_{11}, T_{22} są kwadratowe, to

$$\sigma(A) = \sigma(T_{11}) \cup \sigma(T_{22}).$$

Często T_{22} jest rozmiaru 1×1.

Dowód: $\det(A - \lambda I) = \det(T_{11} - \lambda I) \det(T_{22} - \lambda I)$. $\square$

Wniosek

Jeśli znamy wartości własne T_{22}, wystarczy znaleźć wartości własne T_{11}.

Metoda QR — algorytm bazowy

Pełne zadanie własne dla macierzy **niesymetrycznej**

$A_1 = A$;
for $k = 1 : stop$
  wykonaj rozkład $A_k = Q_k R_k$;
  $A_{k+1} = R_k \cdot Q_k$;
end

Zachodzi $\sigma(A_k) = \sigma(A)$, bo $A_{k+1} = \underbrace{R_k}_{=Q_k^T A_k} \cdot Q_k = Q_k^T A_k Q_k$.

Metoda QR

Twierdzenie (O zbieżności bazowej metody QR)

Niech wartości własne $A \in R^{N\times N}$ spełniają $|\lambda_1| > \ldots > |\lambda_N| > 0$ oraz macierz $T = [v_1, \ldots, v_N]$ o kolumnach v_i złożonych z kolejnych wektorów własnych A ma taką własność, że T^{-1} ma rozkład LU, $T^{-1} = LU$.

Wtedy w metodzie QR ciąg macierzy Q_k jest zbieżny do macierzy diagonalnej, a ciąg A_k ma podciąg zbieżny do macierzy trójkątnej, której elementy diagonalne u_{ii} są równe λ_i dla $i = 1, \ldots, N$.

Metoda QR z przesunięciem (idea)

$A_1 = A$;
for $k = 1 : stop$
  wybierz *sprytne* przesunięcie σ_k
  wykonaj rozkład $A_k - \sigma_k I = Q_k R_k$;
  $A_{k+1} = R_k \cdot Q_k + \sigma_k I$;
end

Najprostsza strategia[3]: $\sigma_k = a_{nn}$, a potem deflacja.

Koszt wyznaczenia wszystkich wektorów i wartości własnych jest rzędu $O(N^3)$ ze stałą równą około 30.

AFAIK nie jest znana „doskonała strategia", gwarantująca zbieżność dla każdej macierzy.

[3] Nie zawsze skuteczna; są lepsze.

Wrażliwość wartości własnych

Twierdzenie (Bauera–Fikego)

Jeśli A jest diagonalizowalna, $X^{-1}AX = D = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ i μ jest wartością własną $A + \Delta$, to

$$\min_i |\mu - \lambda_i| \leqslant \mathrm{cond}_2(X)\|\Delta\|_2.$$

Dowód: Na tablicy. □

Wniosek

Jeśli $A = A^T$, to w sytuacji powyżej $\min_i |\mu - \lambda_i| \leqslant \|\Delta\|_2$.

Twierdzenie (Weyla)

Jeśli $A = A^T$ i $\Delta = \Delta^T$ oraz
$\lambda_1 \geqslant \cdots \geqslant \lambda_N$ – w.wł A, $\quad \tilde{\lambda}_1 \geqslant \cdots \geqslant \tilde{\lambda}_N$ – w.wł $A + \Delta$, to

$$|\lambda_i - \tilde{\lambda}_i| \leqslant \|\Delta\|_2.$$

Lokalizacja wartości własnych

Stwierdzenie

Jeśli λ — w. własna macierzy A, to

$$|\lambda| \leqslant ||A||,$$

gdzie $||A||$ — norma indukowana przez normę wektorową.

Twierdzenie (Gerszgorina, o kołach)

Jeśli λ — w. własna macierzy A, to

$$\lambda \in \bigcup_{i=1}^{N} K_i,$$

gdzie

$$K_i = \{z \in \mathbb{C} : |z - a_{ii}| \leqslant \sum_{j\neq i} |a_{ij}|\}.$$

Interpolacja wielomianowa

Bazy wielomianowe

Wielomian stopnia co najwyżej N:

$$p(x) = \sum_{i=0}^{N} a_i x^i.$$

- wszystkie wielomiany stopnia $\leqslant N$ tworzy przestrzeń liniową wymiaru $N+1$, oznaczmy ją P_N.
- jeśli $\phi_0, \ldots, \phi_N$ są bazą p-ni P_N, to każdy $p \in P_N$ daje się zapisać

$$p(x) = \sum_{i=0}^{N} \alpha_i \phi_i(x).$$

Bazy wielomianowe — przykłady

$p(x) = \sum_{i=0}^{N} \alpha_i \phi_i(x)$.

- „naturalna": $\phi_i(x) = x^i$;
- Newtona:

$$\phi_0(x) = 1, \qquad \phi_i(x) = \prod_{j<i}(x - x_j),$$

  gdzie $x_0, \ldots, x_{N-1}$ — zadane punkty;
- Lagrange'a:

$$l_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

  gdzie $x_0, \ldots, x_N$ — zadane i różne punkty;
- Są też inne **użyteczne** bazy, o nich — innym razem

Odpowiednią dać rzeczy bazę!...

Obliczanie wartości wielomianu

Algorytm zależy od wyboru bazy!

W bazie Newtona (i naturalnej też):

Algorytm 5 (Hornera) Obliczanie wartości wielomianu w bazie Newtona $w = p(x) = a_0 + a_1(x - x_0) + \ldots + a_N(x - x_0)\cdots(x - x_{N-1})$

```
w = a_N;
for k = N − 1 downto : 0
    w = w · (x − x_k) + a_k;
end
return w
```

Koszt: $\mathcal{O}(N)$

.

Obliczanie wartości wielomianu

W bazie Lagrange'a:

$$p(x) = \sum_{i=0}^{N} a_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

Jeśli robisz to źle, koszt będzie kwadratowy...

Postać barycentryczna:

$$p(x) = \sum_{i=0}^{N} a_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} = \begin{cases} a_i & \text{gdy } x = x_i, \\ \phi_{N+1}(x) \cdot \sum_{i=0}^{N} \frac{a_i}{x - x_i} \beta_i, \end{cases}$$

gdzie

$$\phi_{N+1}(x) = \prod_{i=0}^{N} (x - x_i), \qquad \beta_i = \prod_{j \neq i} \frac{1}{x_i - x_j}$$

Współczynniki β_i obliczamy w fazie *precomputingu*.

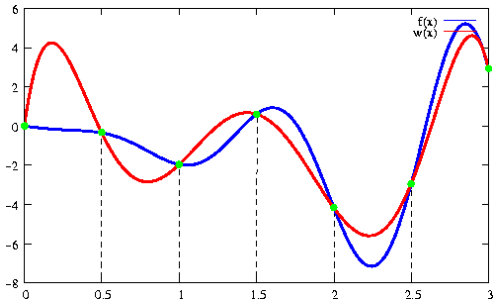
Bonus: dla niektórych ważnych typów węzłów można to zrobić całkiem tanio.

Interpolacja wielomianowa

Problem (interpolacja Lagrange'a)

Dla zadanych (parami różnych) węzłów interpolacji $x_0, \ldots, x_N$ i wartości $f(x_0), \ldots, f(x_N)$ znaleźć wielomian $p \in P_N$ taki, że

$$p(x_i) = f(x_i), \qquad i = 0, \ldots, N$$



Interpolacja Lagrange'a

Twierdzenie

Zadanie interpolacji Lagrange'a ma jednoznaczne rozwiązanie.

Dowód: Niech $p(\cdot) = \sum_{j=0}^{N} c_j \varphi_j(\cdot)$, gdzie

$$P_N = \operatorname{lin}\{\varphi_0, \varphi_1, \ldots, \varphi_n\}.$$

Z warunków interpolacji,

$$p(x_i) = \sum_{j=0}^{N} c_j \varphi_j(x_i) = f(x_i), \qquad 0 \leqslant i \leqslant N.$$

Jest to układ $N+1$ równań liniowych z $N+1$ niewiadomymi c_j.

Ma on jednoznaczne rozwiązanie wtw wektor zerowy jest jedynym rozwiązaniem układu jednorodnego.

Ale jedyny wielomian st $\leqslant N$, który zeruje się w $N+1$ punktach, musi być zerowy. $\square$

Wyznaczenie WIL

W bazie naturalnej, tzn. $w(x) = \sum_i a_i \cdot x^i$???

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^N \\ 1 & x_1 & x_1^2 & \cdots & x_1^N \\ \vdots & \vdots & & \vdots & \\ 1 & x_N & x_N^2 & \cdots & x_N^N \end{bmatrix}}_{\text{macierz Vandermonde'a}} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}$$

- gęsta, niesymetryczna, $N \times N$
- koszt GEPP: $\mathcal{O}(N^3)$
- patologicznie źle uwarunkowana

W innej bazie będzie łatwiej!

Wyznaczenie WIL

W bazie Newtona, tzn.

$$p(x) = \sum_i b_i \cdot (x - x_0) \cdots (x - x_{i-1})$$

Definicja (Różnice dzielone)

$$f(t_0, t_1, \ldots, t_s) = \frac{f(t_1, t_2, \ldots, t_s) - f(t_0, t_1, \ldots, t_{s-1})}{t_s - t_0}.$$

Twierdzenie (O różnicach dzielonych)

Jeśli p jest WIL w bazie Newtona j.w., to

$$b_i = f(x_0, x_1, \ldots, x_i), \qquad 0 \leqslant i \leqslant N.$$

Algorytm różnic dzielonych wyznaczania WIL

$$\begin{array}{llllll}
x_0 & f(x_0) & & & & \\
x_1 & f(x_1) & f(x_0, x_1) & & & \\
x_2 & f(x_2) & f(x_1, x_2) & f(x_0, x_1, x_2) & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \\
x_N & f(x_N) & f(x_{N-1}, x_N) & f(x_{N-2}, x_{N-1}, x_N) & \cdots & f(x_0, x_1, \ldots, x_N)
\end{array}$$

Tabelkę wypełniamy kolejnymi kolumnami.
Odpowiadający algorytm można wykonać *in situ*, kosztem $\mathcal{O}(N^2)$ flopów.

Wyznaczenie WIL w bazie Lagrange'a

$$p(x) = \sum_{i=0}^{N} a_i \underbrace{\prod_{j \neq i} \frac{x - x_j}{x_i - x_j}}_{= l_i(x)}$$

Rozwiązaniem ZIL jest

$$p(x) = \sum_{i=0}^{N} f(x_i) \cdot l_i(x).$$

Koszt „zerowy", ale pamiętajmy o koszcie obliczenia wartości p: alg. barycentr. wymaga precomputingu o koszcie $\mathcal{O}(N^2)$.

Twierdzenie (Highama)

Niech $x_i, f(x_i), x$ będą l. maszynowymi. Obliczona alg. barycentr. w fl wartość $\tilde{p}(x)$ jest dokładną wartością WIL dla $\tilde{f}(x_i) = f(x_i) \cdot (1 + \delta_i)$, gdzie $|\delta_i| \lesssim 5(N+1)\nu$.

Błąd interpolacji Lagrange'a

Twierdzenie (Postać błędu interpolacji)

Niech p będzie WIL dla f w punktach $x_0, \ldots, x_N$.

- *Dla dowolnego $\bar{x} \in \mathbb{R}$*

$$f(\bar{x}) - p(\bar{x}) = f(x_0, x_1, \ldots, x_N, \bar{x}) \cdot \phi_{N+1}(\bar{x}). \qquad (1)$$

- *Ponadto, jeśli $f \in C^{(N+1)}[a, b]$, gdzie $[a, b]$ zawiera $x_0, \ldots, x_N, \bar{x}$, to istnieje $\xi = \xi(\bar{x}) \in (a, b)$ t. że*

$$f(\bar{x}) - p(\bar{x}) = \frac{f^{(N+1)}(\xi)}{(N+1)!} \cdot \phi_{N+1}(\bar{x}).$$

$\phi_{N+1}(x) = (x - x_0) \cdots (x - x_N)$.

Uwarunkowanie ZIL

Jak odporne jest ZIL na zaburzenie wartości w węzłach? Niech

- p_f — WIL dla f,
- $p_{\tilde{f}}$ — WIL dla $\tilde{f}$,

oba oparte na tych samych węzłach $x_0, \ldots, x_N$.

Twierdzenie (uwarunkowanie ZIL)

Jeśli $|f(x_i) - \tilde{f}(x_i)| \leqslant \epsilon |f(x_i)|$ *dla* $i = 0, \ldots, N$, *to dla dowolnego* x

$$\frac{|p_f(x) - p_{\tilde{f}}(x)|}{|p_f(x)|} \leqslant \mathrm{cond}_{rel}(x, f) \cdot \epsilon,$$

gdzie

$$\mathrm{cond}_{rel}(x, f) = \frac{\sum_{j=0}^{N} |l_j(x)| \cdot |f(x_j)|}{|p_f(x)|} \geqslant 1.$$

$\max_{x \in [a,b]} \sum_{j=0}^{N} |l_j(x)| \leftarrow$ stała Lebesgue'a dla $x_0, \ldots, x_N$ na $[a, b]$.

Interpolacja wielomianowa: uogólnienie

Problem (interpolacja Hermite'a)

Dla zadanych (parami różnych) węzłów interpolacji $x_0, \ldots, x_n$ i odp. wartości pochodnych f, znaleźć wielomian $p \in P_N$ taki, że

$$p^{(k)}(x_i) = f^{(k)}(x_i) \qquad \text{dla} \quad k = 0 \ldots m_i - 1, \quad i = 0 \ldots n,$$

przy czym $N + 1 = m_0 + \ldots + m_n$.

„Węzeł x_i ma krotność m_i"

Twierdzenie

ZIH ma jednoznaczne rozwiązanie.

Interpolacja Hermite'a w bazie Newtona

Baza Newtona uwzględniająca krotność węzłów:

$$\begin{aligned}
\phi_0(x) &= 1 \\
\phi_1(x) &= (x - x_0),\ \phi_2(x) = (x - x_0)^2,\ \ldots,\ \phi_{m_0}(x) = (x - x_0)^{m_0}, \\
\phi_{m_0+1}(x) &= (x - x_0)^{m_0}(x - x_1),\ \ldots,\ \phi_{m_0+m_1}(x) = (x - x_0)^{m_0}(x - x_1)^{m_1}, \\
&\vdots \\
\phi_{N+1}(x) &= (x - x_0)^{m_0}(x - x_1)^{m_1} \cdots (x - x_n)^{m_n}
\end{aligned}$$

Twierdzenie

WIH jest postaci $p(x) = \sum_{k=0}^{N} f(x_0, \ldots, x_k)\phi_k(x)$, *gdzie*

$$f(t_0, \ldots, t_k) = \begin{cases} \dfrac{f^{(k)}(x_0)}{k!} & \text{gdy } t_0 = \ldots = t_k, \\ \dfrac{f(t_1, \ldots, t_k) - f(t_0, \ldots, t_{k-1})}{t_k - t_0} & \text{w p.p.} \end{cases}$$

Błąd interpolacji Hermite'a

Twierdzenie (Postać błędu interpolacji)

Niech p *będzie WIH dla* f *w punktach* $x_0, \ldots, x_n$ *o łącznej krotności* $N + 1$. *Jeśli* $f \in C^{(N+1)}[a, b]$, *gdzie* $[a, b]$ *zawiera* $x_0, \ldots, x_n, \bar{x}$, *to istnieje* $\xi = \xi(\bar{x}) \in (a, b)$ *t. że*

$$f(\bar{x}) - p(\bar{x}) = \frac{f^{(N+1)}(\xi)}{(N+1)!} \cdot \phi_{N+1}(\bar{x}).$$

$\phi_{N+1}(x) = (x - x_0)^{m_0} \cdots (x - x_N)^{m_n}$.

Splajny

Sposoby reprezentacji splajnów

- W postaci PP (*piecewise polynomial*): nieco nadmiarowa, ale prosta koncepcyjnie
- W bazie tej przestrzeni (więc minimalna)
  - w bazie obciętych wielomianów (przydatna głównie w teorii)
  - w B-bazie (wykorzystywana w praktyce)

Splajn, czyli funkcja sklejana

Niech $a = x_0 < x_1 < \cdots < x_n = b$.

Definicja

$s : [a, b] \to \mathbb{R}$ jest splajnem stopnia k opartym na węzłach $\{x_j\}$, jeśli spełnione są następujące dwa warunki:

(i) s jest wielomianem stopnia co najwyżej k na każdym z przedziałów $[x_{j-1}, x_j]$,

(ii) $s \in C^{k-1}[a, b]$.

$C^k[a, b] := \{f : [a, b] \to \mathbb{R} \mid f$ ma przynajmniej k pochodnych ciągłych na $[a, b]\}$.

Stwierdzenie

Niech $\mathcal{S}_k$ — przestrzeń liniowa splajnów stopnia k opartych na (ustalonych) węzłach x_j.

$$\dim \mathcal{S}_k = n + k.$$

Reprezentacja PP

- Na każdym odcinku $[x_i, x_{i+1})$ splajn $s \in \mathcal{S}_k$ to po prostu wielomian, więc...
- ...zadajemy go przez współczynniki $a_{i0}, \ldots, a_{ik}$ rozwinięcia w (jakiejś z góry ustalonej) bazie wielomianowej:
  - Standardowo, w bazie Newtona z wielokrotnym węzłem x_i, czyli $1, (x - x_i), \ldots, (x - x_i)^k$
  - Możliwe są inne opcje: np. splajn *kubiczny* w bazie wielomianów bazowych Hermite'a

$$\begin{aligned}
&p_{00}(x_i) = \mathbf{1}, \quad p'_{00}(x_i) = 0, \quad p_{00}(x_{i+1}) = 0, \quad p_{00}(x_{i+1}) = 0,\\
&p_{10}(x_i) = 0, \quad p'_{10}(x_i) = \mathbf{1}, \quad p_{10}(x_{i+1}) = 0, \quad p'_{10}(x_{i+1}) = 0,\\
&p_{01}(x_i) = 0, \quad p'_{01}(x_i) = 0, \quad p_{01}(x_{i+1}) = \mathbf{1}, \quad p'_{01}(x_{i+1}) = 0,\\
&p_{11}(x_i) = 0, \quad p'_{11}(x_i) = 0, \quad p_{11}(x_{i+1}) = 0, \quad p'_{11}(x_{i+1}) = \mathbf{1}.
\end{aligned}$$

Interpolacja splajnowa

Dla uproszczenia przyjmujemy, że węzłami interpolacji są węzły $\{x_i\}$, na których opiera się splajn. Ograniczamy się do k nieparzystego, $k = 2m + 1$.

Problem

Znaleźć $s \in \mathcal{S}_k$ taki, że

$$s(x_i) = f(x_i), \qquad i = 0, \ldots, n.$$

Interpolacja splajnowa — warunki brzegowe

Uzupełnienie warunków interpolacji o dowolny z poniższych zestawów $2m$ warunków prowadzi do *jednoznacznego* zagadnienia:

- Hermitowskie warunki brzegowe:

$$s^{(r)}(a) = f^{(r)}(a), \quad s^{(r)}(b) = f^{(r)}(b), \qquad r = 1, \ldots, m.$$

- Naturalne warunki brzegowe:

$$s^{(r)}(a) = 0 = s^{(r)}(b), \qquad r = m + 1, \ldots, 2m.$$

- Periodyczne warunki brzegowe[4]:

$$s^{(r)}(a) = s^{(r)}(b), \qquad r = 1, \ldots, 2m.$$

[4] Zakładamy $f^{(r)}(a) = f^{(r)}(b)$ dla $r = 0, \ldots, m$.

Konstrukcja naturalnego interpolacyjnego splajnu kubicznego w reprezentacji PP

Węzły: $a = x_0 < \ldots < x_n = b$. Dla uproszczenia zapisu oznaczmy $h_i = x_{i+1} - x_i$.

Splajn $s \in \mathcal{S}_3$ na odcinku $[x_i, x_{i+1}]$ jest postaci:

$$s_i(x) := s(x)_{|_{[x_i, x_{i+1}]}} = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \qquad i = 0, \ldots, n-1$$

Warunki interpolacji:

$$s(x_i) = y_i, \qquad i = 0, \ldots, n.$$

Naturalne warunki brzegowe:

$$s''(a) = 0 = s''(b).$$

Konstrukcja naturalnego interpolacyjnego splajnu kubicznego w reprezentacji PP

$$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \qquad i = 0, \ldots, n-1$$
$$s_i'(x) = b_i + 2c_i\,(x - x_i) + 3d_i\,(x - x_i)^2$$
$$s_i''(x) = 2c_i + 6d_i\,(x - x_i)$$

Krok 1 Z warunków interpolacji $s_i(x_i) = y_i$ wynika natychmiast, że

$$a_i = y_i \qquad i = 0, \ldots n-1.$$

Krok 2 Z ciągłości drugiej pochodnej w wewnętrznych węzłach

$$s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}) \implies 2c_i + 6d_i\,h_i = 2c_{i+1}, \qquad i = 0, \ldots, n-2,$$

i z warunku brzegowego $2c_i + 6d_i\,h_i = 0$ dla $i = n-1$ zatem

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \qquad i = 0, \ldots, n-1 \quad \text{(przyjmując } c_n := 0\text{)}.$$

Konstrukcja naturalnego interpolacyjnego splajnu kubicznego w reprezentacji PP

$$s(x)_{|_{[x_i,x_{i+1}]}} = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Krok 3 Z warunków ciągłości samej funkcji s dostajemy

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1}) \implies y_i + b_i\, h_i + c_i\, h_i^2 + d_i\, h_i^3 = y_{i+1}$$

skąd, uwzględniając wyznaczone przed chwilą d_i,

$$b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i) \qquad i = 0, \ldots, n-2.$$

Konstrukcja naturalnego interpolacyjnego splajnu kubicznego w reprezentacji PP

$$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$
$$s_i'(x) = b_i + 2c_i\,(x - x_i) + 3d_i\,(x - x_i)^2$$
$$s_i''(x) = 2c_i + 6d_i\,(x - x_i)$$

Krok 4 W końcu, z warunków ciągłości pochodnej

$$s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}) \implies b_i + 2c_i\, h_i + 3d_i\, h_i^2 = b_{i+1}$$

skąd, po przekształceniach,

$$\frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i) + 2c_i\, h_i + 3\frac{c_{i+1} - c_i}{3h_i} h_i^2 = \frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{h_{i+1}}{3}(c_{i+2} + 2c_{i+1}$$

i ostatecznie układ równań na współczynniki $c_0, \ldots, c_{n-1}, c_n$:

$$\frac{h_i}{h_i + h_{i+1}}\, c_i + 2\, c_{i+1} + \frac{h_{i+1}}{h_i + h_{i+1}}\, c_{i+2} = 3f[x_i, x_{i+1}, x_{i+2}] \qquad i = 0, \ldots, n-2.$$

Konstrukcja naturalnego interpolacyjnego splajnu kubicznego w reprezentacji PP

$$\frac{h_i}{h_i + h_{i+1}}\, c_i + 2\, c_{i+1} + \frac{h_{i+1}}{h_i + h_{i+1}}\, c_{i+2} = 3f[x_i, x_{i+1}, x_{i+2}] \qquad i = 0, \ldots, n-2.$$

Mamy więc $n - 1$ równań na współczynniki $c_0, \ldots, c_n$, ale

$$s''(x_0) = 0 = s''(x_n) \implies c_0 = c_n = 0.$$

Krok 6 Wykorzystując ostatni warunek interpolacji

$$s(x_n) = y_n$$

otrzymujemy równanie (na b_{n-1}):

$$y_{n-1} + b_{n-1}\, h_{n-1} + c_{n-1}\, h_{n-1}^2 + d_{n-1}\, h_{n-1}^3 = y_n.$$

Konstrukcja naturalnego interpolacyjnego splajnu kubicznego w reprezentacji PP

$$\frac{h_i}{h_i + h_{i+1}}\, c_i + 2\, c_{i+1} + \frac{h_{i+1}}{h_i + h_{i+1}}\, c_{i+2} = 3f[x_i, x_{i+1}, x_{i+2}] \qquad i = 0, \ldots, n-2.$$

Ostatecznie zadanie sprowadza się do *zamkniętego* układu równań na współczynniki $c_1, \ldots, c_{n-1}$, z macierzą

$$T = \begin{bmatrix} 2 & \omega_0 & & \\ \gamma_1 & \ddots & \ddots & \\ & \ddots & \ddots & \omega_{n-2} \\ & & \gamma_{n-1} & 2 \end{bmatrix}, \qquad \gamma_i := \frac{h_i}{h_i + h_{i+1}}, \quad \omega_i := \frac{h_{i+1}}{h_i + h_{i+1}}. \tag{2}$$

Jest to macierz diagonalnie dominująca, więc nieosobliwa.

Trójdiagonalna $\implies$ możemy ją rozwiązać kosztem $\mathcal{O}(n)$ eliminacją Gaussa bez osiowania.

Własność ekstremalna splajnów

Twierdzenie (Holladay)

Niech będą dane wartości $f(x_i)$ w węzłach $a = x_0 < \ldots < x_n = b$. Naturalny, interpolacyjny splajn kubiczny minimalizuje wartość

$$\int_a^b (v''(x))^2 \, dx$$

wśród wszystkich funkcji $v \in C^2[a,b]$ interpolujących f w tych węzłach.

Dowód: Na ćwiczeniach. □

Reprezentacja splajnu w B-bazie

Określamy następujące funkcje B_i^k:

$$B_i^0(x) = \begin{cases} 1 & \text{dla } x \in [x_i, x_{i+1}) \\ 0 & \text{w p.p.} \end{cases}$$

$$B_i^k(x) = V_i^k(x) \cdot B_i^{k-1}(x) + (1 - V_{i+1}^k(x)) \cdot B_{i+1}^{k-1}(x)$$

gdzie

$$V_i^k(x) = \frac{x - x_i}{x_{i+k} - x_i}.$$

Okazuje się, że

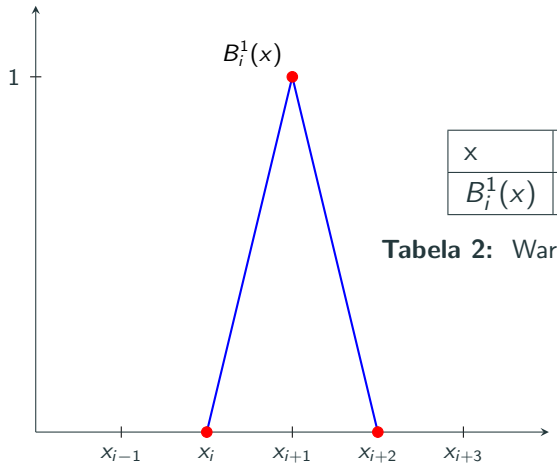
$$\mathcal{S}_k = \text{lin}\{\underbrace{B_{-k}^k, B_{-k+1}^k, \ldots,}_{k \text{ funkcji}} \underbrace{B_0^k, \ldots, B_{n-1}^k}_{n \text{ funkcji}}\}$$

zatem $s \in \mathcal{S}_k$ jest jednoznacznie zadany przez współczynniki c_i:

$$s(x) = \sum_{i=-k}^{n-1} c_i^k B_i^k(x).$$

B-baza splajnów liniowych

Dla S_1 opartych na węzłach $x_0 < \ldots < x_n$, B-baza to $B_{-1}^1, \ldots, B_{n-1}^1$.
Tych funkcji jest $n + 1$.

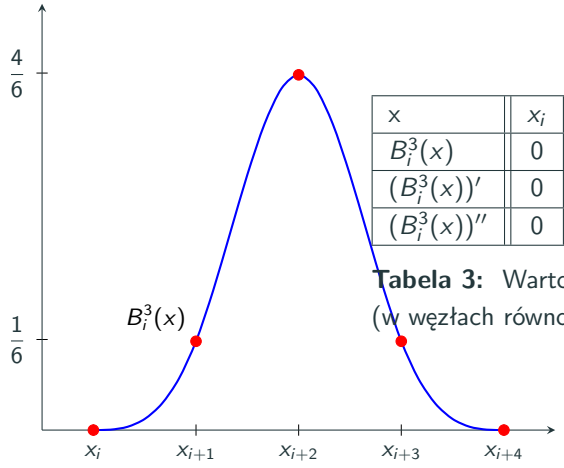


x	x_i	x_{i+1}	x_{i+2}
$B_i^1(x)$	0	1	0

Tabela 2: Wartości splajnu bazowego B_i^1

B-baza splajnów kubicznych

Dla S_3 opartych na węzłach $x_0 < \ldots < x_n$, B-baza to $B_{-3}^3, \ldots, B_{n-1}^3$. Tych funkcji jest $n + 3$.



x	x_i	x_{i+1}	x_{i+2}	x_{i+3}	x_{i+4}
$B_i^3(x)$	0	$1/6$	$4/6$	$1/6$	0
$(B_i^3(x))'$	0	$1/2h$	0	$-1/2h$	0
$(B_i^3(x))''$	0	$1/h^2$	$-2/h^2$	$1/h^2$	0

Tabela 3: Wartości splajnu bazowego B_i^3 (w węzłach równoodległych o h)

Własności B-bazy

- **Ograniczony nośnik:** $B_i^k(x)$ zeruje[5] się poza (x_i, x_{i+k+1}).
- **Dodatniość:** $B_i^k(x) > 0$ w (x_i, x_{i+k+1}).
- **Rozkład jedności:** $\sum_{i=-\infty}^{+\infty} B_i^k(x) = 1$.

[5] Dla $k = 0$ jest nieco inaczej.

Algorytm de Boora obliczania wartości splajnu zadanego w B-bazie

Problem

Splajn $s \in \mathcal{S}_k$ oparty na węzłach $a = x_0 < \ldots < x_N = b$ jest zadany przez współczynniki rozwinięcia w B-bazie:

$$s(x) = \sum_{i=-k}^{N-1} c_i B_i^k(x).$$

Obliczyć $s(x)$ w zadanym $x \in [a, b]$.

Niech $x \in [x_j, x_{j+1}]$. Obserwacja:

- W punkcie x niezerowe są tylko funkcje $B_{j-k}^k, \ldots, B_j^k$, więc

$$s(x) = \sum_{i=j-k}^{j} c_i^k B_i^k(x).$$

- Wartości funkcji $B_i^k(x)$ są zadane rekurencyjnie.

Algorytm de Boora obliczania wartości splajnu zadanego w B-bazie

$$s(x) = \sum_{i=j-k}^{j} c_i^k B_i^k(x)$$

oraz

$$B_i^k(x) = V_i^k(x) \cdot B_i^{k-1}(x) + (1 - V_{i+1}^k(x)) \cdot B_{i+1}^{k-1}(x).$$

Zatem

$$s(x) = \sum_{i=j-k}^{j} c_i^k B_i^k = \sum_{i=j-k}^{j} c_i^k \left(V_i^k \cdot B_i^{k-1} + (1 - V_{i+1}^k) \cdot B_{i+1}^{k-1} \right)$$

$$= \sum_{i=j-k}^{j} c_i^k V_i^k \cdot B_i^{k-1} + \sum_{i=j-k}^{j} c_i^k (1 - V_{i+1}^k) \cdot B_{i+1}^{k-1}.$$

Algorytm de Boora obliczania wartości splajnu zadanego w B-bazie

$$s(x) = \sum_{i=j-k}^{j} c_i^k V_i^k \cdot B_i^{k-1} + \sum_{i=j-k}^{j} c_i^k (1 - V_{i+1}^k) \cdot B_{i+1}^{k-1}.$$

Zmieniając indeksowanie w ostatniej sumie

$$s(x) = \sum_{i=j-k}^{j} c_i^k V_i^k \cdot B_i^{k-1} + \sum_{i=j-k+1}^{j+1} c_{i-1}^k (1 - V_i^k) \cdot B_i^{k-1}$$

$$= \underbrace{c_{j-k}^k V_{j-k}^k \cdot B_{j-k}^{k-1}}_{=0} + \sum_{i=j-(k-1)}^{j} \underbrace{\left(c_i^k V_i + c_{i-1}^k (1 - V_{i-1}^k) \right)}_{=: c_i^{k-1}} \cdot B_i^{k-1} + \underbrace{c_j^k (1 - V_{j+1}^k) \cdot}_{=0}$$

gdyż wśród funkcji bazowych stopnia $k - 1$, jedyne niezerowe nośniki w $[x_j, x_{j+1}]$ mają $B_j^{k-1}, B_{j-1}^{k-1}, \ldots, B_{j-(k-1)}^{k-1}$.

Algorytm de Boora obliczania wartości splajnu zadanego w B-bazie

Dostajemy więc zależność rekurencyjną

$$s(x) = \sum_{i=j-k}^{j} c_i^k B_i^k = \sum_{i=j-k+1}^{j} c_i^{k-1} B_i^{k-1},$$

gdzie

$$c_i^{k-1} = c_i^k V_i + c_{i-1}^k (1 - V_{i-1}^k), \qquad i = j-k+1, \ldots, j.$$

Iterując ten pomysł, należy wyznaczyć kolejne kolumny tabelki

$$\begin{array}{c|cccc} c_j^k & c_j^{k-1} & \cdots & \cdots & c_j^0 \\ c_{j-1}^k & c_{j-1}^{k-1} & & \iddots & \\ \vdots & \vdots & c_{j-k+2}^{k-2} & & \\ \vdots & c_{j-k+1}^{k-1} & & & \\ c_{j-k}^k & & & & \end{array}$$

Wartość splajnu to $s(x) = c_j^0$. Koszt: $\mathcal{O}(k^2)$.

Wykorzystanie B-bazy

- Wyznaczenie splajnu interpolacyjnego w B-bazie: macierz pasmowa
- Pochodna, całka ze splajnu w B-bazie: uogólnienie algorytmu de Boore'a.

Błąd interpolacji naturalnym splajnem kubicznym

Węzły równoodległe: $x_i = a + i \cdot h$, gdzie $h = (b-a)/n$.

Twierdzenie

Niech s będzie naturalnym kubicznym splajnem interpolacyjnym dla $f \in C^4[a,b]$ Istnieje stała c niezależna od h i n taka, że

$$\|f - s\|_\infty \leqslant c \cdot h^2 \cdot \|f^{(4)}\|_\infty.$$

$\|g\|_\infty := \sup_{x \in [a,b]} |g(x)|$.

Błąd interpolacji hermitowskim splajnem kubicznym

Węzły równoodległe: $x_i = a + i \cdot h$, gdzie $h = (b-a)/n$.

Twierdzenie

Niech s będzie kubicznym splajnem interpolacyjnym dla $f \in C^4[a,b]$, z hermitowskimi warunkami brzegowymi:

$$s'(a) = f'(a), \qquad s'(b) = f'(b).$$

Istnieją stałe c_r niezależne od h i n takie, że

$$\|f^{(r)} - s^{(r)}\|_\infty \leqslant c_r \cdot h^{4-r} \cdot \|f^{(4)}\|_\infty, \qquad r = 0, 1, 2, 3.$$

W szczególności,

$$\|f - s\|_\infty \leqslant \frac{5}{384} \cdot h^4 \cdot \|f^{(4)}\|_\infty.$$

$\|g\|_\infty := \sup_{x \in [a,b]} |g(x)|$.

Aproksymacja

Zadanie najlepszej aproksymacji

Problem

Niech F bedzie przestrzenią liniową i niech $V \subset F$ — podprzestrzeń skończonego wymiaru. Dla danego $f \in F$ znaleźć $v^* \in V$ taki, że

$$\|f - v^*\| \leqslant \|f - v\| \qquad \forall v \in V.$$

Będziemy mówić, że v^* jest ENA dla f.

Zadanie najlepszej aproksymacji

Twierdzenie (o istnieniu ENA)

Niech F bedzie przestrzenią liniową i niech $V \subset F$ — podprzestrzeń skończonego wymiaru. Dla każdego $f \in F$ istnieje $v^* \in V$ taki, że

$$\|f - v^*\| \leqslant \|f - v\| \qquad \forall v \in V.$$

Aproksymacja średniokwadratowa

Przypomnienie: zadanie najlepszej aproksymacji

Problem

Niech F bedzie przestrzenią liniową i niech $V \subset F$ — podprzestrzeń skończonego wymiaru. Dla danego $f \in F$ znaleźć $v^* \in V$ taki, że

$$\|f - v^*\| \leqslant \|f - v\| \qquad \forall v \in V.$$

Zadanie najlepszej aproksymacji średniokwadratowej wielomianami

Problem

Niech F bedzie przestrzenią liniową i niech $V \subset F$ — podprzestrzeń skończonego wymiaru. Dla danego $f \in F$ znaleźć $v^* \in V$ taki, że

$$\|f - v^*\| \leqslant \|f - v\| \qquad \forall v \in V.$$

- $F = C[a, b]$ — przestrzeń funkcji ciągłych,
- $V = P_N$ — przestrzeń wielomianów stopnia co najwyżej N,
- $\|f\|_{2,\rho} = \left(\int_a^b |f(x)|^2 \, \rho(x) \, dx\right)^{1/2}$,
  ρ — nieujemna funkcja wagowa, zerująca się co najwyżej w skończonej liczbie punktów.

Zadanie najlepszej aproksymacji średniokwadratowej wielomianami

Problem

Dla zadanej funkcji $f \in C[a, b]$ znaleźć wielomian $p^* \in P_N$ taki, że

$$\|f - p^*\|_2 \to \min!$$

(to znaczy: $\|f - p^*\|_2 \leqslant \|f - p\|_2$ dla każdego $p \in P_N$).

Jak wiemy z ogólnego twierdzenia o istnieniu ENA, taki p^* **istnieje**.

Zadanie najlepszej aproksymacji w przestrzeni z iloczynem skalarnym

Twierdzenie (charakteryzacja ENA w p-ni unitarnej)

Niech F będzie przestrzenią liniową z iloczynem skalarnym $(\cdot, \cdot)$, z normą indukowaną

$$\|f\| := (f, f)^{1/2}.$$

Niech $V \subset F$ będzie sk.wym. podp-nią liniową w F. Wtedy $v^* \in V$ jest ENA dla $f \in F$ wtedy i tylko wtedy, gdy

$$(f - v^*, v) = 0 \qquad \forall v \in V.$$

Na przykład,

$$(f, g) := \int_a^b f(x) \, g(x) \, \rho(x) \, dx$$

jest iloczynem skalarnym w $F = C[a, b]$.

Zadanie najlepszej aproksymacji w przestrzeni z iloczynem skalarnym

Wniosek (algorytm wyznaczenia ENA w p-ni unitarnej)

Niech F — unitarna oraz $V = \mathrm{lin}\{\phi_1, \ldots, \phi_N\} \subset F$.
$v^* = \sum_i a_i \phi_i$ jest ENA dla $f \in F$ wtedy i tylko wtedy, gdy współczynniki a_i są rozwiązaniem układu równań

$$\underbrace{\begin{bmatrix} (\phi_1,\phi_1) & \cdots & (\phi_1,\phi_N) \\ \vdots & & \vdots \\ (\phi_N,\phi_1) & \cdots & (\phi_N,\phi_N) \end{bmatrix}}_{\text{macierz Grama}} \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} \begin{bmatrix} (f,\phi_1) \\ \vdots \\ (f,\phi_N) \end{bmatrix}.$$

Zadanie najlepszej aproksymacji w przestrzeni z iloczynem skalarnym

Wniosek (algorytm wyznaczenia ENA w bazie ortogonalnej)

Jeśli baza $V = \mathrm{lin}\{\phi_1, \ldots, \phi_N\}$ jest ortogonalna,

$$(\phi_i, \phi_j) = 0 \qquad \text{dla } i \neq j,$$

to współczynniki a_i są rozwiązaniem diagonalnego układu równań

$$\begin{bmatrix} (\phi_1,\phi_1) & & \\ & \ddots & \\ & & (\phi_N,\phi_N) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} \begin{bmatrix} (f,\phi_1) \\ \vdots \\ (f,\phi_N) \end{bmatrix},$$

zatem wówczas

$$v^* = \sum_i \frac{(f,\phi_i)}{(\phi_i,\phi_i)}\,\phi_i$$

jest ENA dla $f \in F$.

Warto więc, jeśli to możliwe, korzystać z bazy ortogonalnej!

Wielomiany ortogonalne

Definicja (wielomiany ortogonalne)

Wielomiany $p_0, p_1, \ldots$ tworzą układ wielomianów ortogonalnych, jeśli

- p_k jest stopnia k dla $k = 0, 1, \ldots$
- $(p_i, p_j) = 0$ dla $i \neq j$.

Wielomiany ortogonalne: formuła trójczłonowa

Twierdzenie

Wielomiany ortogonalne postaci x^k + ...niższe potęgi... spełniają zależności:

$$p_0(x) = 1, \quad p_{-1}(x) = 0$$
$$p_k(x) = (x - \alpha_k)p_{k-1}(x) - \beta_k p_{k-2}(x), \qquad k = 1, 2, \ldots.$$

gdzie

$$\alpha_k = \frac{(xp_{k-1}, p_{k-1})}{\|p_{k-1}\|^2}, \qquad \beta_k = \frac{(xp_{k-1}, p_{k-2})}{\|p_{k-2}\|^2}.$$

Wniosek

Algorytm Clenshaw'a obliczania kosztem $\mathcal{O}(N)$ wartości wielomianu zadanego w bazie ortogonalnej: na ćwiczeniach.

Wielomiany ortogonalne: przykłady

- Wielomiany Legendre'a: $(f,g) := \int_{-1}^{+1} f(x)\,g(x)\,dx$:

  $$L_k(x) = \frac{2k-1}{k}\,x\,L_{k-1}(x) - \frac{k-1}{k}\,L_{k-2}(x), \qquad \|L_k\|^2 = \frac{2}{2k+1}, \qquad k = 1,2,\ldots$$

- Wielomiany Czebyszewa: $(f,g) := \int_{-1}^{+1} f(x)\,g(x)\,\rho(x)\,dx$, $\rho(x) = 1/\sqrt{1-x^2}$:

  $$T_k(x) = 2\,x\,T_{k-1}(x) - T_{k-2}(x), \qquad \|T_k\|^2 = \frac{\pi}{2}, \qquad k = 1,2,\ldots$$

- Wielomiany Hermite'a: $(f,g) := \int_{-\infty}^{+\infty} f(x)\,g(x)\,\rho(x)\,dx$, $\rho(x) = e^{-x^2}$:

  $$H_k(x) = 2\,x\,H_{k-1}(x) - 2(k-1)\,H_{k-2}(x), \qquad \|H_k\|^2 = \sqrt{\pi}\,2^k\,k!, \qquad k = 1,2,\ldots$$

Aproksymacja jednostajna

Zadanie najlepszej aproksymacji jednostajnej wielomianami

- $F = C[a,b]$ — przestrzeń funkcji ciągłych,
- $V = P_N$ — przestrzeń wielomianów stopnia co najwyżej N,
- $\|f\|_\infty = \sup_{x\in[a,b]} |f(x)|$.

Problem

Dla zadanej funkcji $f \in C[a,b]$ znaleźć wielomian $p^* \in P_N$ taki, że

$$\|f - p^*\|_\infty \to \min!$$

(to znaczy: $\|f - p^*\|_\infty \leqslant \|f - p\|_\infty$ dla każdego $p \in P_N$).

Jak wiemy z ogólnego twierdzenia o istnieniu ENA, taki p^* **istnieje**.

Alternans

Twierdzenie (o alternansie)

Dla zadanej $f \in C[a,b]$ istnieje dokładnie jeden $p^* \in P_N$ będący ENA dla f w sensie normy supremum. Co więcej,

$$p^* \in P_N \text{ jest ENA dla } f \iff \text{dla } f \text{ i } p^* \text{ istnieje alternans.}$$

Definicja (alternans)

Niech $p^* \in P_N$. Zbiór $N+2$ punktów
$a \leqslant \xi_0 < \xi_1 \ldots < \xi_{N+1} \leqslant b$ takich, że $e = f - p^*$ spełnia

- $|e(\xi_i)| = \|e\|_\infty$ dla $i = 0,\ldots,N+1$,
- $e(\xi_i) = -e(\xi_{i+1})$ dla $i = 0,\ldots,N$,

nazywamy **alternansem** dla f i p^*.

Zbieżność najlepszej aproksymacji jednostajnej

Twierdzenie (Jacksona)

Niech r będzie ustalone takie, że $f \in C^r[-1,1]$ oraz

$$\exists M_r > 0 \qquad |f^{(r)}(x) - f^{(r)}(y)| \leqslant M_r|x-y| \qquad \forall x, y \in [-1,1].$$

Jeśli $p^* \in P_N$ jest ENA dla f i $N > r$, to

$$\|f - p^*\|_\infty \leqslant c_r \frac{M_r}{N^{r+1}},$$

gdzie $c_r > 0$ nie zależy od f.

Wyznaczenie ENA jednostajnej

- W ogólnym przypadku jest to zadanie trudne (nieliniowe)
- Algorytm Remeza wyznaczania ENA: iteracyjne poprawianie alternansu. *W granicy* dostajemy alternans i wielomian optymalny.
  (Tak, to znaczy, że liczba iteracji $\to \infty$)

Pomysł: zastąpić ENA czymś niewiele gorszym, ale za to znacznie tańszym do wyznaczenia.

Aproksymacja jednostajna a interpolacja wielomianowa

Przypomnienie:

Twierdzenie (Postać błędu interpolacji)

Niech p będzie WIL dla $f \in C^{(N+1)}[a,b]$ w punktach $x_0, \ldots, x_N$ w $[a,b]$. Dla $x \in [a,b]$ istnieje $\xi \in (a,b)$ t. że

$$f(x) - p(x) = \frac{f^{(N+1)}(\xi)}{(N+1)!} \cdot \underbrace{(x-x_0)\cdots(x-x_N)}_{=:\phi_{N+1}(x)},$$

Wniosek

$$\|f - p\|_\infty \leqslant \frac{\|f^{(N+1)}\|_\infty}{(N+1)!} \cdot \|\phi_{N+1}\|_\infty.$$

Pytanie: jak dobrać węzły interpolacji, by zminimalizować w/w oszacowanie?

Węzły Czebyszewa

Niech $[a,b] = [-1,1]$. Szukamy $x_0, \ldots x_N$ t. że

$$\|\phi_{N+1}\|_\infty \to \min!$$

czyli

$$\max_{x\in[-1,1]} |\phi_{N+1}(x)| \to \min!$$

czyli

$$\max_{x\in[-1,1]} |(x-x_0)\cdots(x-x_N)| \to \min!$$

Okazuje się, że rozwiązaniem tego zadania są miejsca zerowe wielomianu Czebyszewa T_{N+1},

$$x_i = \cos\left(\frac{2i+1}{2N+2}\pi\right), \qquad i = 0, \ldots N.$$

Wielomiany Czebyszewa

Formuła trójczłonowa

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

$$\begin{aligned}
T_0(x) &= 1\\
T_1(x) &= x\\
T_2(x) &= 2x^2 - 1\\
T_3(x) &= 4x^3 - 3x\\
T_4(x) &= 8x^4 - 8x^2 + 1\\
&\vdots\\
T_n(x) &= 2^{n-1}x^n + \ldots \text{niższe potęgi} \ldots
\end{aligned}$$

Wielomiany Czebyszewa — własności

- Dla $x \in [-1,1]$, $T_n(x) = \cos n\theta$, gdzie $x = \cos\theta$.
- (Ortogonalność) Jeśli $i \neq j$, to
$$\int_{-1}^{1} T_i(x) \cdot T_j(x) \frac{1}{\sqrt{1-x^2}}\,dx = 0.$$
- Wszystkie miejsca zerowe T_n są jednokrotne i w $[-1,1]$:
$$T_n(x_i) = 0 \iff x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \qquad i = 1, \ldots, n.$$
- $|T_n(x)| \leqslant 1$ dla $x \in [-1,1]$ oraz
$$T_n(y_j) = (-1)^j \iff y_j = \cos\left(\frac{j}{n}\pi\right), \qquad j = 0, \ldots, n.$$
- (własność minimaksu) Wśród wszystkich wielomianów postaci
$$x^n + \ldots \text{niższe potęgi} \ldots,$$
najmniejszą normę $\|\cdot\|_\infty$ na $[-1,1]$ ma wielomian $\frac{T_n}{2^{n-1}}$.

Węzły Czebyszewa

Wniosek

$$\max_{x\in[-1,1]} |(x - x_0)\cdots(x - x_N)| \to \min!$$

wtedy i tylko wtedy, gdy $\{x_i\}$ — miejsca zerowe T_{N+1}. Są to węzły Czebyszewa.

Twierdzenie

Jeśli p jest WIL dla $f \in C^{N+1}[-1,1]$ opartym na węzłach Czebyszewa, to

$$\|f - p\|_\infty \leqslant \frac{\|f^{(N+1)}\|_\infty}{2^N(N+1)!}.$$

- Węzły Czebyszewa: miejsca zerowe wielomianu Czebyszewa T_{N+1}
- Węzły Czebyszewa–Lobatto: ekstrema wielomianu Czebyszewa T_N na $[-1,1]$

Aproksymacja jednostajna a interpolacja wielomianowa

Twierdzenie

Niech $f \in C[a,b]$ oraz

- *$p^* \in P_N$ będzie ENA dla f w normie supremum na $[a,b]$*
- *$p \in P_N$ będzie WIL dla f, opartym na $a \leqslant x_0 < \cdots < x_N \leqslant b$.*

Wtedy

$$\|f - p^*\|_\infty \leqslant \|f - p\|_\infty \leqslant (1 + \Lambda_N) \cdot \|f - p^*\|_\infty,$$

gdzie $\Lambda_N = \|\lambda_N\|_\infty$ (stała Lebesgue'a), gdzie

$$\lambda_N(x) := \sum_{i=0}^{N} |l_i(x)| \qquad \leftarrow l_i : \textit{funkcje bazy Lagrange'a.}$$

Stała Lebesgue'a

$\Lambda_N = \|\lambda_N\|_\infty$ (stała Lebesgue'a).

Twierdzenie

- *Dla dowolnego układu węzłów w* $[-1,1]$,

$$\Lambda_N \geqslant \frac{2}{\pi}\ln(N+1) + 0.5215\ldots.$$

- *Dla węzłów równoodległych w* $[-1,1]$,

$$\Lambda_N > \frac{2^{N-2}}{N^2}.$$

- *Dla węzłów Czebyszewa–Lobatto w* $[-1,1]$,

$$\Lambda_N \leqslant \frac{2}{\pi}\ln(N+1) + 1.$$

Interpolacja w węzłach Czebyszewa jest prawie optymalna

Wniosek

Jeśli

- $p^* \in P_N$ *jest ENA dla* f *w normie supremum na* $[a,b]$
- $p \in P_N$ *jest WIL dla* f*, opartym na (przeskalowanych) węzłach Czebyszewa–Lobatto w* $[a,b]$,

to

$$\|f-p\|_\infty \leqslant C_N \cdot \|f-p^*\|_\infty,$$

gdzie $C_N < 3$ *dla* $N \leqslant 20$*;* $C_N < 10$ *dla* $N \leqslant 10^6$.

Równania nieliniowe

Równania nieliniowe

Problem (równanie skalarne)

Dana jest $f : \mathbb{R} \supset (a,b) \to \mathbb{R}$. Znaleźć $x \in (a,b)$ taki, że

$$f(x) = 0.$$

Problem (układ równań nieliniowych)

Dana jest $f : \mathbb{R}^N \supset D \to \mathbb{R}^N$, gdzie D jest otwarty i niepusty. Znaleźć $x \in D$ taki, że

$$f(x) = 0.$$

Źródła kłopotów:

- rozwiązanie może nie istnieć
- rozwiązań może być wiele (czy chcemy znaleźć wszystkie?)
- f może być zadana przez czarną skrzynkę (*oracle*)
- N może być duże, f może być kosztowna

Uwarunkowanie zadania wyznaczania miejsca zerowego

Problem

Mamy

$$f(x^*) = 0, \qquad \tilde{f}(\widetilde{x^*}) = 0,$$

przy czym $\|f - \tilde{f}\|_\infty \leqslant \epsilon$. Jak można oszacować $\|x^* - \widetilde{x^*}\|$?

Przypomnienie (Zadanie obliczenia $x = P(y)$)
Gdy P różniczkowalna w y, to

$$\mathrm{cond}_{abs}(P, y) = \|P'(y)\| \text{ oraz } \|x - \tilde{x}\| \lesssim \mathrm{cond}_{abs}(P, y) \cdot \|y - \tilde{y}\|.$$

U nas $x^* = f^{-1}(0) =: P(0)$. Jeśli f jest klasy C^1 w otoczeniu x^* oraz $f'(x^*)$ jest nieosobliwa, to $P'(0) = [f'(x^*)]^{-1}$, zatem

$$\|x^* - \widetilde{x^*}\| \lesssim \underbrace{\|[f'(x^*)]^{-1}\|}_{\mathrm{cond}(f^{-1},0)} \cdot \epsilon.$$

Metoda bisekcji (dla równania skalarnego)

Niech $f \in C[a, b]$ taka, że $f(a) \cdot f(b) < 0$. (Więc na pewno ma miejsce zerowe w (a, b))

$x = (a + b)/2$
while $|b - a| > \epsilon$ **do**
  if $f(x) == 0$ **then return** x ▷ trafiliśmy!
  end if
  if $\operatorname{sign} f(x) \neq \operatorname{sign} f(a)$ **then**
    $b = x$ ▷ rozwiązanie w lewej połówce
  else
    $a = x$ ▷ rozwiązanie w prawej połówce
  end if
  $x = (a + b)/2$
end while
return x

Metoda bisekcji — szybkość zbieżności

Twierdzenie

$f \in C[a, b]$ taka, że $f(a) \cdot f(b) < 0$.

Oznaczmy $[a_n, b_n]$ — przedział lokalizujący zero w n-tej iteracji. Wtedy

$$|b_n - a_n| = \frac{1}{2}|b_{n-1} - a_{n-1}| = \frac{1}{2^n}|b - a|.$$

Ponadto $x_n = \frac{1}{2}(a_n + b_n)$ spełnia

$$|x_n - x^*| \leqslant \frac{1}{2}|b_n - a_n|.$$

Wniosek

Aby $|x_n - x^*| \leqslant \epsilon$, wystarczy, że $n + 1 \geqslant \log_2\left(\frac{|b - a|}{\epsilon}\right)$.

Szybkość zbieżności metody iteracyjnej

Ciąg $x_n \to x^* \in \mathbb{R}^N$:

- jest zbieżny z rzędem (co najmniej) $p > 1$, jeśli

  $$\exists C \geqslant 0 \quad \|x_{n+1} - x^*\| \leqslant C\|x_n - x^*\|^p \qquad \forall n = 0, 1, \ldots$$

  ($p = 2$: zbieżność *kwadratowa*, $p = 3$: zbieżność *sześcienna/kubiczna*)
- jest zbieżny (co najmniej) liniowo, jeśli

  $$\exists \gamma \in [0, 1) \quad \|x_{n+1} - x^*\| \leqslant \gamma\|x_n - x^*\| \qquad \forall n = 0, 1, \ldots$$

- jest r-zbieżny z rzędem p (odp. liniowo), jeśli

  $$\|x_n - x^*\| \leqslant r_n$$

  dla pewnego ciągu r_n zbieżnego z rzędem p (odp. liniowo) do zera.

Szybkość zbieżności metody iteracyjnej

Wniosek

Metoda bisekcji jest zbieżna r-liniowo z ilorazem $\frac{1}{2}$.

Przykład (Jak wyglądają różne rodzaje zbieżności)
Startujemy z $\|x_0 - x^*\| = 0.5$ i...

	Iteracja (n)				
Szybkość	1	2	3	4	5
liniowa $\gamma = 1/2$	0.25	0.13	0.06	0.03	0.02
liniowa $\gamma = 0.9$	0.45	0.41	0.36	0.33	0.30
kwadratowa[6] ($p = 2$)	0.25	$6 \cdot 10^{-2}$	$4 \cdot 10^{-3}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-10}$
kubiczna[6] ($p = 3$)	0.13	$2 \cdot 10^{-3}$	$7 \cdot 10^{-9}$	$4 \cdot 10^{-25}$	$7 \cdot 10^{-74}$

[6] Przyjmując, że $\|x_{n+1} - x^*\| = \|x_n - x^*\|^p$

Zadanie punktu stałego

Problem

Niech $\Phi : \mathbb{R}^N \supset D \to \mathbb{R}^N$. Znaleźć $x^* \in D$ taki, że

$$x^* = \Phi(x^*).$$

x^* to **punkt stały** Φ.

To zadanie zawsze jest równoważne jakiemuś zadaniu wyznaczania zera funkcji, np. dla

$$f(x) = x - \Phi(x).$$

Metoda Banacha (iteracja prosta)

Twierdzenie (Banacha, o kontrakcji)

Niech $\Phi : D \to D$ *będzie* ***kontrakcją****:*

$$\exists L \in [0,1) \quad \|\Phi(x) - \Phi(y)\| \leqslant L\,\|x - y\| \qquad \forall x, y \in D,$$

gdzie $D \subset \mathbb{R}^N$ *— domknięty, niepusty. Wtedy*

- *istnieje dokładnie jeden punkt stały* Φ, $x^* = \Phi(x^*)$;
- *iteracja Banacha*

$$x_{n+1} = \Phi(x_n)$$

jest zbieżna liniowo do x^* *z dowolnego* $x_0 \in D$*:*

$$\|x_{n+1} - x^*\| \leqslant L\|x_n - x^*\|.$$

Szybkość zbieżności metody Banacha

Twierdzenie (maszynka do rzędu)

Załóżmy, że iteracja $x_{n+1} = \Phi(x_n)$ *jest zbieżna do* $x^* = \Phi(x^*) \in \operatorname{int} D$*, przy czym* $\Phi \in C^p(D)$, $p \geqslant 1$, *oraz*

$$\Phi'(x^*) = \ldots = \Phi^{(p-1)}(x^*) = 0.$$

Jeśli x_0 *jest dostatecznie blisko* x^* *to*

$$\exists C \quad \|x_{n+1} - x^*\| \leqslant C\|x_n - x^*\|^p.$$

Szybkość zbieżności metody Banacha — dowód

Dowód: $\underbrace{x_{n+1}-x^*}_{=e_{n+1}} = \Phi(x_n) - x^* = \underbrace{\Phi(x_n)}_{=\Phi(x^*+e_n)} - \Phi(x^*)$. Na mocy wzoru Taylora (o ile e_n jest dostatecznie małe)

$$\Phi(x^*+e_n) = \Phi(x^*) + \sum_{k=1}^{p-1} \frac{1}{k!} \underbrace{\Phi^{(k)}(x^*)}_{=0} e_n^k + \frac{1}{p!}\Phi^{(p)}(x^*)e_n^p + R(x^*, e_n),$$

gdzie $\dfrac{\|R(x^*, e_n)\|}{\|e_n\|^p} \to 0$ dla $e_n \to 0$. Zatem jeśli $\|e_0\|$ jest dostatecznie mała, to

$$\|e_{n+1}\| \leqslant \frac{1}{p!}\|\Phi^{(p)}(x^*)\|\|e_n\|^p + C\|e_n\|^p.$$

Co więcej, dla $n \to \infty$,

$$\frac{\|e_{n+1}\|}{\|e_n\|^p} \to \frac{1}{p!}\|\Phi^{(p)}(x^*)\|.$$

□

Metoda Newtona

$$x_{n+1} = x_n - [f'(x_n)]^{-1} f(x_n)$$

Twierdzenie (ogólne, o lokalnej zbieżności m. Newtona)

Niech $f : D \to \mathbb{R}^N$ oraz $D \subset \mathbb{R}^N$ otwarty i niepusty. Niech $x^* \in D$ będzie[7] miejscem zerowym, $f(x^*) = 0$. Jeśli

- *f jest różniczkowalna w D oraz*

$$\exists L \quad \|f'(x) - f'(y)\| \leqslant L\|x - y\| \qquad \forall x, y \in D,$$

- *$f'(x^*)$ jest nieosobliwa,*

to dla x_0 dost. blisko x^* metoda Newtona jest zbieżna oraz

$$\exists C \quad \|x_{n+1} - x^*\| \leqslant C\|x_n - x^*\|^2 \qquad n = 0, 1, \ldots.$$

[7] Zakładamy więc, że takie x^* istnieje.

Metoda Newtona — wersja skalarna

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

Twierdzenie (o lokalnej zbieżności m. Newtona — wersja uproszczona[8])

Niech $f : (a, b) \to \mathbb{R}$ i niech $x^* \in (a, b)$ będzie jej miejscem zerowym: $f(x^*) = 0$. Jeśli

- *$f \in C^2(a, b)$*
- *$f'(x^*) \neq 0$,*

to dla x_0 dost. blisko x^* metoda Newtona jest zbieżna oraz

$$\exists C \quad |x_{n+1} - x^*| \leqslant C|x_n - x^*|^2 \qquad n = 0, 1, \ldots.$$

[8] Ma mocniejsze założenia, ale łatwiej się dowodzi.

Modyfikacje m. Newtona dla r-nia skalarnego: bez pochodnych

Idea: zamiast metody stycznych (Newtona): $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ użyć metody przybliżonej

$$x_{n+1} = x_n - \frac{f(x_n)}{g_n}, \qquad \text{gdzie } g_n \approx f'(x_n).$$

- Metoda siecznych (nie wymaga dodatk. obl. f, r-zbieżność rzędu $(1+\sqrt{5})/2 \approx 1.618$)

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

- Metoda Steffensena (wymaga 1 dodatk. obl. f), zb. rzędu 2)

$$g_n = \frac{f(x_n + f(x_n)) - f(x_n)}{f(x_n)}$$

- Metoda z ilorazem różnicowym (1 dod. obl. f, zb. liniowa)

$$g_n = \frac{f(x_n + h) - f(x_n)}{h}.$$

Implementacja wielowymiarowej metody Newtona

Metoda Newtona:

$$x_{n+1} = x_n - s,$$

wymaga obliczenia poprawki,

$$s = [f'(x_n)]^{-1} f(x_n).$$

Aby wyznaczyć poprawkę należy więc **rozwiązać układ równań** z macierzą $N \times N$,

$$f'(x_n) \cdot s = f(x_n).$$

Uproszczona metoda Newtona

Rozwiązywanie układu równań w każdym kroku kosztuje.

$$x_{n+1} = x_n - [f'(x_n)]^{-1} f(x_n)$$

Uproszczenie:

$$x_{n+1} = x_n - [f'(x_0)]^{-1} f(x_n)$$

Teraz wystarczy tylko raz wyznaczyć rozkład macierzy.

Metoda będzie jednak (lokalnie) zbieżna tylko liniowo.

Przybliżona metoda Newtona

Rozwiązywanie układu równań w każdym kroku kosztuje.

$$x_{n+1} = x_n - [f'(x_n)]^{-1} f(x_n)$$

Dlaczego by nie wykorzystać metody iteracyjnej do rozwiązywania układu $f'(x_n)s = f(x_n)$?

$$x_{n+1} = x_n - s,$$

gdzie s spełnia residualne kryterium stopu

$$\|f'(x_n)s - f(x_n)\| \leqslant \eta_n \|f(x_n)\|.$$

η_n proporcjonalne do $\|f(x_n)\|$ pozwala zachować kwadratowy rząd (lokalnej) zbieżności.

Metoda z przybliżoną pochodną

Uogólnienie metody z ilorazem różnicowym dla r-nia skalarnego.

$$x_{n+1} = x_n - [G_n]^{-1} f(x_n),$$

gdzie $G_n = [g_1, \ldots, g_N]$ i każdą kolumnę przybliżamy ilorazem różnicowym:

$$g_i = \frac{1}{h_n}(f(x_n + h_n e_i) - f(x_n))$$

e_i — i-ty wektor jednostkowy.

h_n proporcjonalne do $\|f(x_n)\|$ pozwala zachować kwadratowy rząd (lokalnej) zbieżności.

Uwaga na dobór h_n:

- za duże — kiepskie przybliżenie pochodnej
- za małe — problemy z redukcją cyfr przy odejmowaniu

Skalarne równania wielomianowe

Problem

Znaleźć $x \in \mathbb{C}$ takie, że

$$p(x) := a_N x^N + \ldots a_1 x + a_0 = 0,$$

gdzie $a_i \in \mathbb{R}$ (lub $\mathbb{C}$) są zadane.

Dla wszystkich pierwiastków jedną z opcji jest wyznaczenie wszystkich wartości własnych macierzy stowarzyszonej:

$$C = \begin{bmatrix} -\frac{a_{N-1}}{a_N} & -\frac{a_{N-2}}{a_N} & \cdots & -\frac{a_0}{a_N} \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix}$$

Zachodzi bowiem: $\{\lambda \in \mathbb{C} : \lambda \text{ jest w. wł } C\} = \{x \in \mathbb{C} : p(x) = 0\}$.

Całkowanie

Kwadratury

Problem

Dla $f : \mathbb{R}^d \supset D \to \mathbb{R}$ przybliżyć wartość całki

$$I(f) = \int_D f(x)\,\rho(x)\,dx \qquad (\rho \text{ to zadana waga})$$

za pomocą kwadratury $Q(f)$,

$$Q(f) = \sum_{i=0}^{n} A_i f(x_i).$$

Chodzi nam o wyznaczenie współczynników kwadratury A_i oraz węzłów kwadratury $x_i \in D$ takich, by $Q(f) \approx I(f)$ możliwie dobrze w pewnej klasie funkcji, $f \in F$.

Kwadratury jednowymiarowe

Na wykładzie będziemy zajmować się wyłącznie całkami na odcinku:

Problem

Dla $f : [a, b] \to \mathbb{R}$ przybliżyć wartość

$$I(f) = \int_a^b f(x)\,\rho(x)\,dx$$

za pomocą kwadratury $Q(f)$,

$$Q(f) = \sum_{i=0}^{n} A_i f(x_i).$$

Kwadratury — zamiana zmiennych

Całkę $I(f) = \int_a^b f(x)\,dx$ można sprowadzić do całki na zadanym odcinku. Np. dla $[0,1]$:

$$[a,b] \ni x = a + t \cdot (b-a) \text{ dla } t \in [0,1]$$

i wtedy

$$I(f) = \int_0^1 f(a + t\cdot(b-a)) \cdot (b-a)\,dt = (b-a)\int_0^1 g(t)\,dt$$

gdzie $g(t) = f(a + t\cdot(b-a))$.

Kwadratura $\sum_i A_i g(t_i)$ dla $\int_0^1 g(t)\,dt$ rzędu r daje kwadraturę tego samego rzędu dla $\int_a^b f(x)\,dx$ z wagami $(b-a)\cdot A_i$ i węzłami $x_i = a + t_i \cdot (b-a)$.

Kwadratury interpolacyjne

Idea: funkcję f przybliżyć wielomianem interpolacyjnym p, a całkę z f — całką z p.

Niech $x_0, \ldots, x_n$ węzły interpolacji Lagrange'a oraz $p(x) = \sum_{i=0}^n f(x_i) \underbrace{l_i(x)}_{\text{baza Lagrange'a}}$. Wtedy

$$I(f) \approx Q(f) := \int_a^b p(x) = \int_a^b \sum_{i=0}^n f(x_i)\, l_i(x) = \sum_{i=0}^n \left(\underbrace{\int_a^b l_i(x)\,dx}_{=:A_i} \right) \cdot f(x_i)$$

A_i — wyznaczamy raz na zawsze (nie zależą od f) w precomputingu: umiemy obliczać całki z wielomianów.

Nie wszystkie kwadratury są interpolacyjne.

Kwadratury Newtona-Cotesa

To kwadratury interpolacyjne oparte na węzłach równoodległych, np.:

- prostokątów

$$Q^P(f) = (b-a)\cdot f\left(\frac{a+b}{2}\right)$$

- trapezów

$$Q^T(f) = \frac{b-a}{2}\cdot(f(a)+f(b))$$

- parabol (Simpsona)

$$Q^S(f) = \frac{b-a}{6}\cdot\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

Błąd kwadratur interpolacyjnych

Twierdzenie

Jeśli $f \in C^{n+1}[a,b]$, a Q jest kwadraturą interpolacyjną opartą na węzłach $x_0, \ldots, x_n \in [a,b]$, to

$$|I(f) - Q(f)| \leqslant \frac{\|f^{(n+1)}\|}{(n+1)!}\,|b-a|^{n+2}.$$

Błąd prostych kwadratur

Twierdzenie

- *(dla kwadratury prostokątów) jeśli $f \in C^2[a,b]$, to*

$$I(f) - Q^P(f) = \frac{(b-a)^3}{24} f''(\xi), \qquad \text{dla pewnego } \xi \in [a,b],$$

- *(dla kwadratury trapezów) jeśli $f \in C^2[a,b]$, to*

$$I(f) - Q^T(f) = -\frac{(b-a)^3}{12} f''(\xi), \qquad \text{dla pewnego } \xi \in [a,b],$$

- *(dla kwadratury Simpsona) jeśli $f \in C^4[a,b]$, to*

$$I(f) - Q^S(f) = -\frac{(b-a)^5}{2880} f^{(4)}(\xi), \qquad \text{dla pewnego } \xi \in [a,b].$$

Kwadratury maksymalnego rzędu

Definicja

Kwadratura Q dla całki

$$I(f) = \int_a^b f(x) \cdot \rho(x)\, dx \qquad (\rho \text{ to funkcja wagowa})$$

jest rzędu (co najmniej) r, jeśli

$$I(p) = Q(p) \quad \forall p \in P_{r-1}.$$

Jest rzędu r, gdy jest co najmniej rzędu r, ale już nie $r+1$.

Wniosek

Każda kwadratura interpolacyjna oparta na n węzłach jest co najmniej rzędu n.

Problem

Jaki jest maksymalny rząd kwadratury? Jaka to kwadratura?

Kwadratury Gaussa

Twierdzenie

Rząd kwadratury interpolacyjnej opartej na n węzłach nie przekracza $2n$. Realizuje go kwadratura Gaussa, oparta na zerach n-tego wielomianu ortogonalnego w $L^2_\rho(a,b)$.

Twierdzenie

Jeśli $f \in C^{2n}[a,b]$, a Q jest kwadraturą Gaussa opartą na n węzłach, to

$$\int_a^b f(x) \cdot \rho(x)\, dx - Q(f) = \|\phi_n\|^2_\rho \cdot \frac{f^{(2n)}(\xi)}{(2n)!}$$

gdzie $\phi_n(x)$ — wielomian bazowy Newtona oparty na węzłach kwadratury, $\xi \in [a,b]$.

Kwadratury złożone

Obserwacja:

$$\int_a^b f(x)\, dx = \sum_{j=0}^{N-1} \int_{x_i}^{x_{i+1}} f(x)\, dx$$

gdzie $a = x_0 < x_1 < \cdots < x_N = b$, zatem wystarczy aproksymować tylko całki na (krótszych) odcinkach $[x_i, x_{i+1}]$:

$$\int_{x_i}^{x_{i+1}} f(x)\, dx \approx Q_{[x_i,x_{i+1}]}(f) \quad \Longrightarrow \quad \int_a^b f(x)\, dx \approx \sum_{j=0}^{N-1} Q_{[x_i,x_{i+1}]}(f).$$

Kwadratury złożone: przykłady

Zakładamy jednorodny podział $[a, b]$ węzłami $x_i = a + i \cdot h$, gdzie $h = (b - a)/N$.

- prostokątów

$$\int_a^b f(x)\,dx \approx Q_N^P(f) = h \sum_{i=0}^{N-1} f\left(\frac{x_i + x_{i+1}}{2}\right)$$

- trapezów

$$\int_a^b f(x)\,dx \approx Q_N^T(f) = \frac{h}{2}\left(f(a) + 2\sum_{i=1}^{N-1} f(x_i) + f(b)\right)$$

- Simpsona

$$\int_a^b f(x)\,dx \approx Q_N^S(f) = \frac{h}{6}\Bigg(f(a) + 2\sum_{i=1}^{N-1} f(x_i) + f(b) + 4\sum_{i=0}^{N-1} f\left(\frac{x_i + x_{i+1}}{2}\right)\Bigg)$$

Błąd kwadratur złożonych

Szacuje się przez sumę błędów lokalnych:

$$|I(f) - Q_N(f)| \leqslant \sum_i |I_{[x_i, x_{i+1}]}(f) - Q_{[x_i, x_{i+1}]}(f)|.$$

Jeśli więc błędy lokalne są rzędu h^{r+1}, to błąd kwadratury będzie rzędu h^r. Ponadto na przykład:

Twierdzenie

Jeśli $f \in C^2[a, b]$, to dla złożonej kwadratury trapezów

$$I(f) - Q_N^T(f) = -\frac{h^2}{12}\,|b - a|\,f''(\xi) \qquad \text{dla pewnego } \xi \in [a, b].$$

Podwyższanie rzędu aproksymacji. Kwadratury Romberga

Oznaczmy $R_0(f, h) := Q_N^T(f)$. Okazuje się, że dla $f \in C^{2m}[a, b]$ istnieją stałe α_i t. że

$$I(f) - R_0(f, h) = \alpha_2 h^2 + \alpha_4 h^4 + \ldots + \alpha_{2m-2} h^{2m-2} + \mathcal{O}(h^{2m}).$$

Zatem też dla $q \in (0, 1)$

$$I(f) - R_0(f, qh) = \alpha_2 q^2 h^2 + \alpha_4 q^4 h^4 + \ldots + \alpha_{2m-2} q^{2m-2} h^{2m-2} + \mathcal{O}(h^{2m}).$$

Stąd

$$R_1(f) = \frac{1}{1 - q^2}\left(R_0(f, qh) - q^2 R_0(f, h)\right)$$

ma wyższy rząd błędu:

$$I(f) - R_1(f, h) = \tilde{\alpha}_4 h^4 + \ldots + \tilde{\alpha}_{2m-2} h^{2m-2} + \mathcal{O}(h^{2m}).$$

Podwyższanie rzędu aproksymacji. Kwadratury Romberga

Iterując ten pomysł definiujemy, oznaczając $R_j^k = R_j\left(f, \frac{h}{2^k}\right)$,

$$R_j^k = \frac{1}{4^j - 1}\left(4^j R_{j-1}^{k+1} - R_{j-1}^k\right), \qquad \leftarrow \text{ kwadratury Romberga.}$$

$$\begin{bmatrix} R_0^0 & & & & \\ R_0^1 & R_1^0 & & & \\ R_0^2 & R_1^1 & R_2^0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ R_0^m & R_1^{m-1} & & & R_m^0 \end{bmatrix}$$

R_0^k — kw. trapezów, R_1^k — kw. Simpsona

Twierdzenie

Jeśli $f \in C^{2m}[a, b]$, to dla $j \leqslant m$ błąd kwadratury R_j^0 spełnia

$$I(f) - R_j^0 = \mathcal{O}(h^{2j+2}).$$

Szybka transformacja Fouriera (FFT)

Dyskretna transformacja Fouriera (DFT)

Oznaczając $\omega_N = e^{-2i\pi/N}$

$$c_k = \sum_{j=0}^{N-1} f_j \cdot e^{-ikj\cdot 2\pi/N} = \sum_{j=0}^{N-1} f_j \cdot {\omega_N}^{kj}, \qquad k = 0, \ldots, N-1.$$

Macierzowo:

$$c = F_N \cdot f,$$

gdzie

$$F_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)} \\ & & & \cdots & \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)^2} \end{bmatrix}, \quad f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{bmatrix}, \quad c = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{bmatrix}$$

Nawet jeślibyśmy wyznaczyli F_N w preprocessingu, koszt zwykłego mnożenia przez F_N byłby $\mathcal{O}(N^2)$.

Szybka transformacja Fouriera (FFT)

$\omega_N = e^{-2i\pi/N}$

$$F_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)} \\ & & & \cdots & \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)^2} \end{bmatrix}$$

- Koszt mnożenia przez F_N obniżymy z $\mathcal{O}(N^2)$ do $\mathcal{O}(N \log_2 N)$
- Koszt rozwiązywania układu z F_N też obniżymy do $\mathcal{O}(N \log_2 N)$

przez wykorzystanie metody *dziel i rządź*.

Stwierdzenie

$\omega_N^N = 1$.

Algorytm FFT

$\omega_N = e^{-2i\pi/N}$. Mamy obliczyć $c_j = \sum_{k=0}^{N-1} f_k \cdot \omega_N^{kj}$ dla $j = 0, \ldots, N-1$.

Dla $\alpha = 0, \ldots, m-1$,

$$c_\alpha = \Phi_\alpha + \omega_N^\alpha \Psi_\alpha,$$
$$c_{\alpha+m} = \Phi_\alpha - \omega_N^\alpha \Psi_\alpha.$$

gdzie

$$\Phi_\alpha = \sum_{k=0}^{m-1} f_{2k} \omega_m^{k\alpha}, \qquad \Psi_\alpha = \sum_{k=0}^{m-1} f_{2k+1} \omega_m^{k\alpha}$$

Czyli $c = F_N f$ obliczamy rekurencyjnie!

Koszt:

$$K(N) = 2K(N/2) + \mathcal{O}(N),$$

zatem na mocy TRU

$$K(N) = \mathcal{O}(N \log_2 N).$$

DFT — własności

$$F_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)} \\ & & & \cdots & \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)^2} \end{bmatrix}, \quad T_N = \begin{bmatrix} 1 & & & & \\ & & & & 1 \\ & & & \iddots & \\ & & \iddots & & \\ & 1 & & & \end{bmatrix}$$

Twierdzenie

- $F_N = F_N^T$
- $\frac{1}{N} F_N^H F_N = I;\ F_N^{-1} = \frac{1}{N}\bar{F}_N;\ \frac{1}{\sqrt{N}} F_N$ *jest unitarna*
- $\bar{F}_N = T_N F_N = F_N T_N$
- $F_N^{-1} y = \frac{1}{N}\bar{F}_N y = \frac{1}{N}\overline{F_N \bar{y}}$

Metody numeryczne!

FFT — zastosowania

- obliczanie DFT, DST i DCT różnej maści
- obliczanie cyklicznego splotu dwóch wektorów:

$$z_k = \sum_{j=0}^{N-1} u_j v_{k-j}, \qquad k = 0, \ldots N-1,$$

- mnożenie wielomianów wysokiego stopnia
- ....

Specyfika metod numerycznych

Przedmiot interdyscyplinarny, na pograniczu

- matematyki teoretycznej (analiza algorytmów dla zadań ciągłych)
- informatyki teoretycznej (analiza algorytmów dla zadań dyskretnych)
- matematyki stosowanej (konstrukcja metod dla konkretnego modelu)
- informatyki praktycznej (implementacja algorytmów, dopasowanie do architektury)
- ma też swoją własną specyfikę (np. arytmetyka fl)

Ciekawe wykłady z numeryki na MIM UW

Wykłady obieralne dostępne dla Informatyki w stałej ofercie:

- Analiza numeryczna
- Aproksymacja i złożoność
- Grafika komputerowa
- Obliczenia naukowe
- Numeryczne równania różniczkowe
- Metody obliczeniowe w finansach

Do tego wykłady monograficzne i seminaria monograficzne z numeryki.

Ciekawe prace magisterskie z numeryki na MIM UW

- **Seminarium magisterskie dostępne dla informatyki** i matematyki
- Prace magisterskie o charakterze
  - implementacyjnym
  - teoretycznym (analitycznym)

Informacje:

https://www.mimuw.edu.pl/~przykry/mnmgr