

M. JANKOWSKI i H. WOŹNIAKOWSKI (Warszawa)

O złożoności obliczeniowej w analizie numerycznej

1. Wstęp. Celem tego artykułu jest przedstawienie polskiemu czytelnikowi problematyki nowego działu analizy numerycznej: tzw. złożoności obliczeniowej (computational complexity). Do niedawna główny nacisk w badaniach położony był na poszukiwanie i wszechstronne badanie nowych metod rozwiązujących dany problem. Miara efektywności nowej metody było na ogół jej porównanie z efektywnością znanych już metod. Dla wielu problemów możemy obserwować wydłużanie się ciągu znanych metod $\{M_n\}$, gdzie ostatnia metoda M_n jest efektywniejsza (w sensie przyjętego kryterium) od poprzednich. To wydłużanie się ciągu $\{M_n\}$ nie zawsze wiąże się z odkryciem metody najefektywniejszej ze znanych. Znacznie częściej nowo odkryte metody zajmują w ciągu miejsca wcześniejsze, powodując przenumarowanie części dawniej znanych lepszych metod.

Wady tego podejścia są oczywiste. Ten sam problem jest badany wielokrotnie i to, w gruncie rzeczy, za każdym razem nie w pełni. Znajomość n -tej metody nie przeczy na ogół istnieniu następnych, lepszych od niej metod. Stosowalność praktyczna M_n powinna (co rzadko ma miejsce) kończyć się z chwilą znalezienia M_{n+1} .

Od niedawna (wczesne lata sześćdziesiąte) można zaobserwować dążenie do poszukiwania i badania metod optymalnych dla danego problemu. Zamiast „wydłużać” ciąg znanych metod, staramy się znaleźć metodę, która dla danego problemu i przyjętego kryterium jest optymalna. Jeśli przyjmiemy za kryterium ilość działań arytmetycznych potrzebnych do rozwiązania problemu, to metodą optymalną będzie metoda minimalizująca ilość działań arytmetycznych (a więc potencjalnie najtańsza).

Problemami istnienia i własności metod optymalnych w sensie minimalizacji pracochłonności rozwiązania zajmuje się właśnie dział złożoności obliczeniowej.

Postawienie tego zagadnienia pozwoliło stwierdzić, że wiele klasycznych metod nie spełnia postulatu optymalności. Jako przykład podajmy, że metoda eliminacji Gaussa, rozwiązująca dowolny układ liniowy $n \times n$ kosztem $O(n^3)$ działań, nie jest na ogół optymalna, gdyż metoda Strassena wymaga $O(n^{\log_2 7})$ działań. Dalej, mnożenie dwóch wielomianów stopnia n można wykonać za pomocą szybkich transformacji Fouriera (FFT) kosztem $O(n \log_2 n)$ działań, zamiast $O(n^2)$ w przypadku klasycznym. Obliczenie wartości wielomianu stopnia n i wszystkich jego pochodnych w danym punkcie można wykonać algorytmem Shaw–Trauba kosztem $3n$ mnożeń zamiast $O(n^2)$ w przypadku algorytmu Hornera.

Należy podkreślić, że w tej chwili tylko dla niewielu problemów, i to stosunkowo prostych, znamy metody optymalne. Na przykład, nadal otwarte jest pytanie postawione w 1956 roku przez G. E. Forsythe’a, jaka jest minimalna ilość działań arytmetycznych potrzebnych

do rozwiązania dowolnego układu równań liniowych $n \times n$? Wiemy jedynie, że minimalna ilość działań $\varphi(n)$ spełnia warunek

$$O(n^2) \leq \varphi(n) \leq O(n^{\log_2 7}).$$

W końcu chcielibyśmy podkreślić, że na ogół przyjmowane kryterium kosztu, mierzonego ilością działań arytmetycznych, stanowi ważną, ale nie jedyną cechę oceny metod numerycznych. Musimy pamiętać, że z powodu nieuchronnych błędów zaokrągleń każdą metodę możemy w praktyce obliczeniowej realizować tylko w sposób przybliżony. Każda metoda przeznaczona do numerycznej realizacji musi więc posiadać pewną „odporność” na wpływ błędów zaokrągleń, co nazywamy numeryczną stabilnością (por. [11]). Dlatego też, dopiero stabilne metody optymalne mogą być rekomendowane do praktyki obliczeniowej.

W niniejszej pracy przedstawiamy kilka typowych zadań numerycznych z punktu widzenia złożoności obliczeniowej. Rozważania koncentrujemy na problemach związanych z wielomianami, przedstawiając stosunkowo nowy, bardzo wydajny i posiadający wiele zastosowań algorytm szybkich transformacji Fouriera (FFT) Cooley’a – Tukey’a. Następnie omawiamy problem rozwiązywania układu równań liniowych.

W rozdziale 6 przedstawiamy problem złożoności obliczeniowej dla zadań, dla których znamy jedynie algorytmy nieskończone (tzw. analityczna złożoność obliczeniowa), koncentrując się na iteracyjnym rozwiązywaniu równań nieliniowych.

2. Uwagi ogólne. Typowe zadanie numeryczne polega na ogół na obliczeniu wartości pewnej funkcji F ,

$$(2.1) \quad \vec{w} = F(\vec{a}),$$

gdzie $F: D \rightarrow R^m$, $D \subset R^n$. Współrzędne wektora $\vec{a} = [a_1, a_2, \dots, a_n]^T$ nazywamy danymi, zaś współrzędne wektora $\vec{w} = [w_1, \dots, w_m]^T$ – wynikami. Zastanówmy się, jaka jest minimalna ilość działań arytmetycznych $\varphi(F, D)$ potrzebna do obliczenia \vec{w} dla dowolnych $\vec{a} \in D$. Wielkości $\varphi(F, D)$ definiujemy jako

$$(2.2) \quad \varphi(F, D) = \sup_{\vec{a} \in D} \varphi(F, \vec{a}),$$

gdzie $\varphi(F, \vec{a})$ oznacza minimalną ilość działań dla obliczenia $F(\vec{a})$. Wartość $\varphi(F, D)$ nazywamy również *złożonością obliczeniową* zadania (2.1).

Rozwiązanie problemu (2.2) jest trudne i oczywiście w sposób istotny zależy od własności funkcji F , czyli od zadania obliczeniowego. W tym miejscu wyprowadzimy proste oszacowanie z dołu wartości $\varphi(F, D)$, przyjmując pewne dość naturalne ograniczenia dla rozważanych zadań i algorytmów.

Założmy, że dla zadania (2.1) znamy skończony algorytm W , w którym, po wykonaniu k działań arytmetycznych, otrzymujemy wynik. (Jeśli algorytm obejmuje również operacje logiczne, to ograniczamy dziedzinę D tak, aby wyniki operacji logicznych nie zależały od danych \vec{a} .)

Niech ciąg

$$(2.3) \quad d_1, d_2, \dots, d_k$$

zawiera wszystkie wyniki działań (zgodnie z ich kolejnością) w algorytmie W . Zatem możemy zapisać

$$(2.4) \quad d_i = \square_i(u_i, v_i)$$

dla pewnych u_i oraz v_i ze zbioru $\{a_1, a_2, \dots, a_n, d_1, d_2, \dots, d_{i-1}\}$, gdzie \square_i oznacza określone działanie arytmetyczne $(+, -, *, /)$. Zakładamy dalej, że wyniki w_1, w_2, \dots, w_m zadania są równe

$$w_1 = d_{i_1}, \quad w_2 = d_{i_2}, \quad \dots, \quad w_m = d_{i_m}$$

dla wskaźników $i_1, i_2, \dots, i_m \in [1, k]$, niezależnych od danych \vec{a} (stąd $m \leq k$).

Mówimy, że rozważane zadanie na zbiorze D ma n danych istotnych jeśli

$$(2.5) \quad \exists_{\vec{a} \in D} \forall_{1 \leq j \leq n} \exists [\vec{a} + \delta \vec{e}_j \in D \text{ oraz } F(\vec{a}) \neq F(\vec{a} + \delta \vec{e}_j)],$$

gdzie $\vec{e}_j = [0, \dots, \underset{j}{1}, \dots, 0]^T$.

TWIERDZENIE. Jeśli zadanie (2.1) ma n danych istotnych, to minimalna ilość działań $\varphi(F, D) \geq n/2$.

D o w ó d. Załóżmy przeciwnie, że $\varphi(F, D) < n/2$. Istnieje więc algorytm W , który dla wektora danych \vec{a} z (2.5) jest jednoznacznie określony przez (2.3) i (2.4), gdzie $k < n/2$. Wśród argumentów $u_1, v_1, \dots, u_k, v_k$ działań $\square_1, \dots, \square_k$ nie mogą występować dane a_1, \dots, a_n gdyż $2k < n$. Niech $a_j \notin \{u_1, v_1, \dots, u_k, v_k\}$. Oznacza to, że dla dowolnej wartości δ , takiej, że $\vec{a} + \delta \vec{e}_j \in D$, wyniki w_1, \dots, w_m otrzymane algorytmem W nie zależą od δ , co sprzeczne jest z (2.5) ($F(\vec{a}) \neq F(\vec{a} + \delta \vec{e}_j)$) i tym samym kończy dowód.

Zauważmy, że oszacowania $n/2$ nie można na ogół poprawić. Na przykład definiując F przez

$$w_i = a_{2i-1} + a_{2i} \quad i = 1, 2, \dots, m, \quad (n = 2m), \quad a_i \in R,$$

otrzymujemy zadanie, w którym ilość działań potrzebnych do obliczenia \vec{w} jest dokładnie równa $n/2$.

Przedstawione twierdzenie oznacza, że liczba istotnych danych określa oszacowanie z dołu złożoności obliczeniowej. Stąd też wynika, że obliczenia związane z wielomianami stopnia n muszą wymagać co najmniej $O(n)$ działań, a rozwiązywanie równań liniowych z dowolną macierzą $n \times n$, co najmniej $O(n^2)$ działań.

3. Szybka transformacja Fouriera. Rozpatrzmy następujące zadania:

(Z1) Dla danych liczb zespolonych $w(0), w(1), \dots, w(n-1)$ szukamy wielkości

$$(3.1) \quad c(k) = \frac{1}{n} \sum_{l=0}^{n-1} e^{-2\pi i k l / n} w(l), \quad k = 0, 1, \dots, n-1, \quad i = \sqrt{-1},$$

lub odwrotnie,

(Z2) Mając dane liczby zespolone $c(0), c(1), \dots, c(n-1)$ szukamy

$$(3.2) \quad w(l) = \sum_{k=0}^{n-1} e^{2\pi i k l / n} c(k), \quad l = 0, 1, \dots, n-1.$$

Klasyczna metoda rozwiązywania (Z1) lub (Z2) wymaga około n^2 zespolonych mnożeń i dodawań oraz wyznaczania $e^{-2\pi i k l / n}$. W 1965 roku Cooley i Tukey [5] zaproponowali algorytm tzw. szybkiej transformacji Fouriera (FFT), w którym ilość zespolonych mnożeń i dodawań została zmniejszona do około $n(r_1 + r_2 + \dots + r_p)$ jeśli $n = r_1 \cdot r_2 \cdot \dots \cdot r_p$. Ideę tego algorytmu przedstawiamy za Reinschem [21].

Wprowadźmy następujące oznaczenia:

$$\begin{aligned} P_0 &\stackrel{\text{df}}{=} 1, & Q_0 &\stackrel{\text{df}}{=} r_1 \cdot r_2 \cdot \dots \cdot r_p, \\ P_1 &\stackrel{\text{df}}{=} r_1, & Q_1 &\stackrel{\text{df}}{=} r_2 \cdot \dots \cdot r_p, \\ P_2 &\stackrel{\text{df}}{=} r_1 \cdot r_2, & & \dots \dots \dots \\ & \dots \dots \dots & Q_{p-1} &\stackrel{\text{df}}{=} r_p, \\ P_p &\stackrel{\text{df}}{=} r_1 \cdot r_2 \cdot \dots \cdot r_p (=n), & Q_p &\stackrel{\text{df}}{=} 1. \end{aligned}$$

Zauważmy, że dowolną liczbę całkowitą z przedziału $\{0, n-1\}$ możemy jednoznacznie przedstawić w postaci

$$k = k_p P_{p-1} + \dots + k_1 P_0, \quad \text{gdzie} \quad 0 \leq k_i \leq r_i - 1 \quad \text{dla } i = 1, 2, \dots, p$$

lub

$$l = l_1 Q_1 + \dots + l_p Q_p, \quad \text{gdzie} \quad 0 \leq l_i \leq r_i - 1 \quad \text{dla } i = 1, 2, \dots, p.$$

Łatwo także sprawdzić, że

$$\sum_{l=0}^{n-1} \alpha_l = \sum_{l_p=0}^{r_p-1} \dots \sum_{l_1=0}^{r_1-1} \alpha_{l_1 Q_1 + \dots + l_p Q_p}.$$

Możemy zatem przepisać (3.1) następująco

$$(3.3) \quad c(k) = c(k_p P_{p-1} + \dots + k_1 P_0) =$$

$$= \frac{1}{n} \sum_{l_p=0}^{r_p-1} \dots \sum_{l_1=0}^{r_1-1} e^{-2\pi i (k_p P_{p-1} + \dots + k_1 P_0)(l_1 Q_1 + \dots + l_p Q_p)/n} w(l_1 Q_1 + \dots + l_p Q_p)$$

ale

$$(l_1 Q_1 + \dots + l_p Q_p)/n = \frac{l_1}{P_1} + \dots + \frac{l_p}{P_p}.$$

Stąd i z (3.3)

$$\begin{aligned}
& n \cdot c(k_p P_{p-1} + \dots + k_1 P_0) = \\
& = \sum_{l_p=0}^{r_p-1} e^{-2\pi i(k_p P_{p-1} + \dots + k_1 P_0) \cdot \frac{l_p}{P_p} \times} \\
& \quad \times \sum_{l_{p-1}=0}^{r_{p-1}-1} e^{-2\pi i(k_p P_{p-1} + \dots + k_1 P_0) \cdot \frac{l_{p-1}}{P_{p-1}} \times} \\
& \quad \dots \dots \dots \\
& \quad \times \sum_{l_1=0}^{r_1-1} e^{-2\pi i(k_p P_{p-1} + \dots + k_1 P_0) \cdot \frac{l_1}{P_1} \times} \\
& \quad \times w(l_1 Q_1 + \dots + l_p Q_p) = \\
& = \left. \begin{aligned} & \sum_{l_p=0}^{r_p-1} e^{-2\pi i(k_p P_{p-1} + \dots + k_1 P_0) \cdot \frac{l_p}{P_p} \times} \\ & \times \sum_{l_{p-1}=0}^{r_{p-1}-1} e^{-2\pi i(k_{p-1} P_{p-2} + \dots + k_1 P_0) \cdot \frac{l_{p-1}}{P_{p-1}} \times} \\ & \dots \dots \dots \\ & \times \sum_{l_1=0}^{r_1-1} e^{-2\pi i k_1 P_0 \cdot \frac{l_1}{P_1} \times} \\ & \times w(l_1 Q_1 + \dots + l_p Q_p) \end{aligned} \right\}_0 \Bigg\}_1 \Bigg\}_p
\end{aligned}$$

Wyrażenie zakreślone klamrą $\}_0$ oznaczmy symbolem $S_0(l_1, \dots, l_p)$, klamrą $\}_1$ symbolem $S_1(k_1, l_2, \dots, l_p)$, klamrą $\}_p$ symbolem $S_p(k_1, \dots, k_p)$.

Algorytm Cooley'a i Tukey'a ma przy tych oznaczeniach następującą postać

start:

$$S_0(l_1, \dots, l_p) := w(l_1 Q_1 + \dots + l_p Q_p)$$

$$\text{dla } 0 \leq l_i \leq r_i - 1, \quad i = 1, 2, \dots, p.$$

$\alpha = 1(1)p$:

$$S_\alpha(k_1, \dots, k_\alpha, l_{\alpha+1}, \dots, l_p)$$

$$:= \sum_{l_\alpha=0}^{r_\alpha-1} e^{-2\pi i(k_\alpha P_{\alpha-1} + \dots + k_1 P_0) l_\alpha / P_\alpha} \cdot S_{\alpha-1}(k_1, \dots, k_{\alpha-1}, l_\alpha, \dots, l_p)$$

$$\text{dla } 0 \leq k_i \leq r_i - 1, \quad i = 1, \dots, \alpha, \quad 0 \leq l_j \leq r_j - 1, \quad j = \alpha + 1, \dots, p.$$

wynik:

$$c(k_p P_{p-1} + \dots + k_1 P_0) = \frac{1}{n} S_p(k_1, \dots, k_p)$$

$$\text{dla } 0 \leq k_i \leq r_i - 1, \quad i = 1, \dots, p.$$

Niech

$$a = l_{\alpha+1} Q_{\alpha+1} + \dots + l_p Q_p, \quad b = k_{\alpha-1} P_{\alpha-2} + \dots + k_1 P_0,$$

$$\bar{b} = k_1 Q_1 + \dots + k_{\alpha-1} Q_{\alpha-1}$$

oraz

$$u = e^{-2\pi i / P_\alpha}, \quad v = e^{-2\pi i / r_\alpha}.$$

A zatem w kroku α

$$S_\alpha(\bar{b} + k Q_\alpha + a) = \sum_{l_\alpha=0}^{r_\alpha-1} e^{-2\pi i \left(\frac{k_\alpha}{r_\alpha} + \frac{b}{P_\alpha} \right) l_\alpha} S_{\alpha-1}(\bar{b} + l_\alpha Q_\alpha + a)$$

dla

$$b = 0, \dots, P_{\alpha-1} - 1, \quad a = 0, \dots, Q_\alpha - 1, \quad k = 0, \dots, r_\alpha - 1.$$

Możemy więc dla $\alpha = 1(1)p$ realizować algorytm następująco

$$\text{dla } b = 0(1)P_{\alpha-1} - 1:$$

$$\text{dla } a = 0(1)Q_\alpha - 1:$$

skalowanie

$$\text{dla } l = 0(1)r_\alpha - 1 : S'_{\alpha-1}(a + \bar{b} + l Q_\alpha) = u^{bl} S_{\alpha-1}(a + \bar{b} + l Q_\alpha)$$

transformacja

$$\text{dla } k = 0(1)r_\alpha - 1 : S_\alpha(a + \bar{b} + k Q_\alpha) = \sum_{l=0}^{r_\alpha-1} v^{kl} S'_{\alpha-1}(a + \bar{b} + l Q_\alpha).$$

Skalowanie i transformacja równoważne są mnożeniu odpowiednio przez macierze D i T ,

$$D = \begin{bmatrix} 1 & & & & \\ & u^b & & & \\ & & u^{2b} & & \\ & & & \ddots & \\ & & & & u^{(r_\alpha-1)b} \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 1 & \dots & 1 \\ & 1 & v & \dots & v^{r_\alpha-1} \\ & & \dots & \dots & \dots \\ & & & & 1 & v^{r_\alpha-1} & \dots & v^{(r_\alpha-1)^2} \end{bmatrix}$$

i wymagają łącznie $r_\alpha^2 + r_\alpha - 4$ mnożeń oraz $r_\alpha(r_\alpha - 1)$ dodawań. Ogółem w kroku α musimy wykonać $O(P_{\alpha-1} \cdot Q_\alpha \cdot r_\alpha^2)$ działań, czyli, ze względu na równość $P_{\alpha-1} \cdot Q_\alpha = \frac{n}{r_\alpha}$, koszt kroku α jest $O(n \cdot r_\alpha)$.

Zatem w całym algorytmie mamy $O(n(r_1 + r_2 + \dots + r_p))$ zespolonych mnożeń i dodawań.

Szybkie transformacje Fouriera dla $2n$ danych rzeczywistych. W przypadku rzeczywistym odpowiednikami (Z1) i (Z2) są zadania:

(R1) Mając dane liczby rzeczywiste y_l ($l = 0, \dots, 2n-1$) szukamy

$$a_k = \frac{1}{n} \sum_{l=0}^{2n-1} y_l \cos \frac{\pi k l}{n} \quad \text{dla} \quad k = 0, \dots, n$$

oraz

$$b_k = \frac{1}{n} \sum_{l=0}^{2n-1} y_l \sin \frac{\pi k l}{n} \quad \text{dla} \quad k = 1, \dots, n-1;$$

lub odwrotnie

(R2) Przy danych współczynnikach rzeczywistych $a_k, k = 0, \dots, n$, oraz $b_k, k = 1, \dots, n-1$, chcemy wyznaczyć wartości

$$y_l = \frac{a_0}{2} + \sum_{k=1}^{n-1} \left\{ a_k \cos \frac{\pi k l}{n} + b_k \sin \frac{\pi k l}{n} \right\} + \frac{a_n}{2} (-1)^l$$

dla $l = 0, \dots, 2n-1$.

Rozwiązanie (R1) sprowadza się do wykonania transformacji zespolonej (por. (Z1))

$$c(k) = \frac{1}{n} \sum_{l=0}^{n-1} e^{-2\pi i k l / n} w(l), \quad k = 0, \dots, n-1,$$

gdzie

$$w(l) = y_{2l} + i \cdot y_{2l+1}, \quad l = 0, \dots, n-1.$$

Następnie podstawiamy $c(n) = c(0)$ i obliczamy wielkości

$$f_k = c(k) + \bar{c}(n-k), \quad g_k = c(k) - \bar{c}(n-k), \quad h_k = g_k e^{-i\pi k/n},$$

dla $k = 0, \dots, [n/2]$.

Dalej łatwo sprawdzić, że

$$\begin{aligned} a_k &= \frac{1}{2} (\operatorname{Re} f_k + \operatorname{Im} h_k), & b_k &= \frac{1}{2} (-\operatorname{Im} f_k + \operatorname{Re} h_k), \\ a_{n-k} &= \frac{1}{2} (\operatorname{Re} f_k - \operatorname{Im} h_k), & b_{n-k} &= \frac{1}{2} (\operatorname{Im} f_k + \operatorname{Re} h_k), \end{aligned}$$

dla $k = 0, \dots, [n/2]$.

Podobnie zadanie (R2) można rozwiązać następująco: Najpierw wyznaczamy wielkości pomocnicze

$$\begin{aligned} f_k &= a_k + a_{n-k} + i(-b_k + b_{n-k}), & h_k &= b_k + b_{n-k} + i(a_k - a_{n-k}), \\ g_k &= e^{i\pi k/n} \cdot h_k, \end{aligned}$$

dla $k = 0, \dots, [n/2]$.

Następnie obliczamy

$$c(k) = \frac{1}{2} (f_k + g_k), \quad c(n-k) = \frac{1}{2} (\bar{f}_k - \bar{g}_k)$$

dla $k = 0, \dots, [n/2]$.

Jeśli wykonamy teraz zespoloną transformację (por. (Z2))

$$w(l) = \sum_{k=0}^{n-1} e^{2\pi i k l / n} c(k), \quad l = 0, \dots, n-1,$$

to szukane wartości y_l są równe

$$y_{2l} = \operatorname{Re} w(l), \quad y_{2l+1} = \operatorname{Im} w(l) \quad \text{dla} \quad l = 0, \dots, n-1.$$

Zastosowania FFT. Aby uzmysłować sobie, jakie korzyści może dać FFT, rozpatrzmy następujący przykład.

Niech $n = 2^k$. Zwykły algorytm rozwiązywania np. (Z1) wymagałby około n^2 zespolonych działań, natomiast FFT tylko około $2n \log_2 n$ działań. Załóżmy, że realizujemy obliczenia na maszynie wykonującej 10^6 zespolonych operacji na sekundę. Oto tabela przybliżonych czasów rozwiązywania (Z1):

k	Zwykły algorytm	FFT
10	1 sek	2_{10} – 2 sek
15	17 min	1 sek
20	ponad 250 godz	40 sek

Szybka transformacja Fouriera jest interesująca sama w sobie ze względu na szerokie zastosowanie dyskretnej analizy Fouriera. Okazuje się jednak, że często jest to dobre „narzędzie pomocnicze” przy konstruowaniu algorytmów. Ważne są również zastosowania teoretyczne w problematyce złożoności obliczeniowej. Koszt algorytmów wykorzystujących FFT stanowi często znacznie lepsze oszacowanie z góry wielkości $\varphi(F, D)$ niż koszt algorytmów „klasycznych”. Jako przykład zastosowań FFT rozważymy teraz algorytm szybkiego mnożenia wielomianów.

Niech $P_1(x), P_2(x)$ będą wielomianami w przybliżeniu tego samego stopnia o współczynnikach zespolonych takimi, że stopień $P_1 \cdot P_2(x)$ równy jest $n - 1$.

„Zwykły” algorytm mnożenia takich wielomianów wymaga co najwyżej $O(n^2)$ działań. Jeśli $n = r_1 \cdot r_2 \cdot \dots \cdot r_p$ i $\sum r_i \ll n$, to możemy tę ilość działań istotnie zmniejszyć wykorzystując szybką transformację Fouriera (por. [10]). Algorytmem FFT wyznaczamy kosztem co najwyżej $O(n(r_1 + \dots + r_p))$ działań wartości $P_1(x_l)$ oraz $P_2(x_l)$, gdzie $x_l = e^{2\pi i l/n}$, a następnie obliczamy $w(x_l) = P_1(x_l) \cdot P_2(x_l)$ dla $l = 0, 1, \dots, n - 1$.

W ten sposób zadanie wyznaczania współczynników iloczynu $P_1(x) \cdot P_2(x)$ sprowadziliśmy do rozwiązania (Z1), co wymaga znowu $O(n(r_1 + \dots + r_p))$ działań.

Można pokazać, że stosując przedstawiony algorytm szybkiego mnożenia wielomianów zadanie obliczania wartości wielomianu stopnia n w n punktach lub znalezienie wielomianu interpolacyjnego, opartego na $n + 1$ węzłach, wymaga tylko $O(n \log_2^2 n)$ działań ($n = 2^k$) w porównaniu z $O(n^2)$ w „klasycznych” algorytmach [14]. Zadanie interpolacyjne jest zresztą jednym z niewielu, dla których znamy nieliniowe ze względu na ilość danych oszacowanie z dołu minimalnej ilości działań $\varphi(n)$.

Strassen [26] pokazał na gruncie geometrii algebraicznej, że $O(n \log_2 n) \leq \varphi(n)$.

4. Jako następne zadanie rozpatrzmy problem obliczania wartości wielomianu i jego znormalizowanych pochodnych w danym punkcie. Niech więc

$$(4.1) \quad P(y) = \sum_{k=0}^n a_{n-k} y^k$$

będzie wielomianem stopnia n . Chcemy obliczyć

$$(4.2) \quad \frac{P(x)}{0!}, \quad \frac{P'(x)}{1!}, \quad \dots, \quad \frac{1}{m!} P^{(m)}(x)$$

dla danego $x \neq 0$ oraz $m \leq n$.

Do niedawna w tym celu stosowano iterowany algorytm Hornera, który wymaga $(m - 1) \times (n - m/2)$ dodawań i mnożeń. Czy jednak nie istnieje algorytm tańszy lub ogólniej, jaka jest minimalna ilość działań potrzebna do obliczenia wielkości (4.2)? Odpowiedź na to pytanie w pewnym sensie zawiera praca Shaw–Trauba [24]. Autorzy zdefiniowali jednoparametrową rodzinę algorytmów obliczania (4.2). Mianowicie, niech

$$n + 1 = p \cdot q$$

dla naturalnych p oraz q i niech

$$s(j) \equiv (n - j) \bmod q,$$

$$r(j) = \begin{cases} 0 & \text{dla } j \bmod q \neq 0, \\ q & \text{dla } j \bmod q \equiv 0. \end{cases}$$

Algorytm

$$T_i^{-1} = a_{i+1} x^{s(i+1)}, \quad i = 0, 1, \dots, n-1,$$

$$T_j^j = a_0 x^{s(0)}, \quad j = 0, 1, \dots, m,$$

$$T_i^j = T_{i-1}^{j-1} + T_{i-1}^j x^{r(i-j)}, \quad j = 0, 1, \dots, m; i = j+1, \dots, n.$$

Po obliczeniu T_i^j interesujące nas wielkości są równe

$$\frac{P^{(j)}(x)}{j!} = \frac{T_n^j}{x^{j \bmod q}}.$$

Dla $q = 1$ otrzymujemy iterowany algorytm Hornera. Natomiast dla $q = n + 1$ otrzymujemy algorytm wymagający wykonania $(m-1)(n-m/2)$ dodawań i tylko $2n+m$ mnożeń i dzieleni.

Co więcej Shaw–Traub udowodnili, że obliczenie (4.2) dla $m = n$ wymaga co najmniej $2n-1$ mnożeń, a więc algorytm dla $q = n+1$ jest ewentualnie gorszy od optymalnego co najwyżej o $n+1$ mnożeń. Algorytm Shaw–Trauba dla $q = n+1$ jest najtańszym znanym algorytmem obliczania wartości wielomianu i jego wszystkich pochodnych. Warto także wspomnieć, że dla $m = 0$ Borodin [2] udowodnił, że jedynym algorytmem, który minimalizuje ilość dodawań i mnożeń jest algorytm Hornera.

Numeryczna stabilność algorytmów Shaw–Trauba została pokazana w pracy [37], gdzie udowodniono, że każda obliczona wartość T_n^j w arytmetyce zmiennopozycyjnej jest dokładną wielkością dla wielomianu P o nieco zaburzonych współczynnikach. Algorytm Shaw–Trauba dla $q = n+1$ minimalizuje ilość mnożeń, natomiast ilość dodawań dla $m = n$ jest w dalszym ciągu rzędu n^2 . Zastosowanie szybkich transformacji Fouriera i w tym przypadku pozwala zmniejszyć ogólną ilość działań z n^2 na $O(n \log_2^2 n)$ mnożeń i dodawań (por. [14]). A więc teraz ilość mnożeń jest $\log_2^2 n$ razy większa, niż w algorytmie Shaw–Trauba, zaś ilość dodawań jest $n/\log_2^2 n$ razy mniejsza. W każdym razie widzimy, że postawienie problemu złożoności obliczeniowej dla zadania (4.1)–(4.2) doprowadziło do nowych, tańszych i w pełni efektywnych algorytmów.

5. Rozwiązywanie układów równań liniowych. Z rozdziału 2 wiemy, że minimalna liczba działań $\varphi(n)$, potrzebnych do rozwiązania układu równań liniowych

$$(5.1) \quad A \vec{x} = \vec{b},$$

o dowolnej nieosobliwej macierzy $A \ n \times n$, szacuje się z dołu liniowo ze względu na liczbę danych, tzn.

$$O(n^2) \leq \varphi(n).$$

Do roku 1969 wszystkie znane algorytmy wymagały co najmniej $O(n^3)$ działań. Ponadto Klujew i Kokowkin–Szczerbak [12] udowodnili, że algorytm eliminacji Gaussa jest optymal-

ny ($n^3/3$ mnożeń!) w klasie metod korzystających tylko z przekształceń elementarnych macierzy. W roku 1969 V. Strassen [25] pokazał, wykorzystując algorytm mnożenia macierzy 2×2 wymagający tylko siedmiu mnożeń, że

$$\varphi(n) \leq O(n^{\log_2 7}) \approx O(n^{2.81}).$$

Za Strassenem [25] zdefiniujemy najpierw algorytmy $\alpha_{m,k}$ (indukcyjnie ze względu na k) mnożenia macierzy kwadratowych stopnia $n = m \cdot 2^k$. Niech $\alpha_{m,0}$ będzie zwykłym algorytmem mnożenia macierzy $m \times m$ (wymagającym m^3 mnożeń i $m^2(m-1)$ dodawań). Zakładając, że znamy $\alpha_{m,k}$, definiujemy $\alpha_{m,k+1}$ następująco:

1^o macierze A i B stopnia $m \cdot 2^{k+1}$, które chcemy pomnożyć, oraz ich iloczyn zapisujemy w postaci blokowej

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad AB = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

gdzie A_{ik}, B_{ik} oraz C_{ik} są macierzami stopnia $n/2 = m \cdot 2^k$,

2^o obliczamy

$$M_1 = (A_{11} + A_{22}) * (B_{11} + B_{22}),$$

$$M_2 = (A_{21} + A_{22}) * B_{11},$$

$$M_3 = A_{11} * (B_{12} - B_{22}),$$

$$M_4 = A_{22} * (-B_{11} + B_{21}),$$

$$M_5 = (A_{11} + A_{12}) * B_{22},$$

$$M_6 = (-A_{11} + A_{21}) * (B_{11} + B_{12}),$$

$$M_7 = (A_{12} - A_{22}) * (B_{21} + B_{22}),$$

gdzie symbol $*$ oznacza mnożenie macierzy algorytmem $\alpha_{m,k}$ a $+$ oraz $-$ zwykłe dodawanie i odejmowanie macierzy.

Stąd

$$C_{11} = M_1 + M_4 - M_5 + M_7,$$

$$C_{21} = M_2 + M_4,$$

$$C_{12} = M_3 + M_5,$$

$$C_{22} = M_1 + M_3 - M_2 + M_6.$$

Można udowodnić, że algorytm $\alpha_{m,k}$ mnożenia dwóch macierzy stopnia $m \cdot 2^k$ wymaga $m^3 7^k$ mnożeń oraz $(5 + m) \cdot m^2 7^k - 6(m \cdot 2^k)^2$ dodawań (odejmowań).

Natomiast dla macierzy dowolnego stopnia n okazuje się, że ich iloczyn można obliczyć, wykorzystując mniej niż $4.7 n^{\log_2 7}$ operacji arytmetycznych.

Przejdźmy teraz do omówienia algorytmu odwracania macierzy. Zakładamy nie tylko, że macierz jest odwracalna, ale również, że algorytm nie załamie się, tzn. będą odwracalne odpowiednie macierze potrzebne dla jego realizacji.

Podobnie jak przy mnożeniu definiujemy algorytmy $\beta_{m,k}$ odwracania macierzy stopnia $n = m \cdot 2^k$ indukcyjnie ze względu na k .

$\beta_{m,0}$ niech będzie zwykłym algorytmem Gaussa.

Znając $\beta_{m,k}$ definiujemy $\beta_{m,k+1}$ następująco:

1^o odwracaną macierz A stopnia $m \cdot 2^{k+1}$ i jej odwrotność A^{-1} zapisujemy w postaci blokowej

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad A^{-1} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

gdzie A_{ik}, C_{ik} są macierzami stopnia $m \cdot 2^k$,

2^o obliczamy

$$M_1 = A_{11}^{-1},$$

$$M_2 = A_{21} * M_1,$$

$$M_3 = M_1 * A_{12},$$

$$M_4 = A_{21} * M_3,$$

$$M_5 = M_4 - A_{22},$$

$$M_6 = M_5^{-1},$$

$$C_{12} = M_3 * M_6,$$

$$C_{21} = M_6 * M_2,$$

$$M_7 = M_3 * C_{21},$$

$$C_{11} = M_1 - M_7,$$

$$C_{22} = -M_6,$$

gdzie symbol $*$ oznacza mnożenie algorytmem $\alpha_{m,k}$, a macierze M_1 i M_6 obliczamy algorytmem $\beta_{m,k}$. Pokazuje się, że algorytm $\beta_{m,k}$ odwracania macierzy stopnia $m \cdot 2^k$ wymaga $m \cdot 2^k$ dzielen, nie więcej niż $\frac{6}{5} m^3 7^k - m \cdot 2^k$ mnożeń i co najwyżej $\frac{6}{5} (5 + m) m^2 7^k - 7 (m \cdot 2^k)^2$ dodawań (odejmowań). Wnioskiem stąd jest oszacowanie z góry minimalnej liczby działań arytmetycznych potrzebnych do odwrócenia macierzy dowolnego stopnia n przez $5.64 n^{\log_2 7}$.

Analogicznie rozumując można pokazać, że jeśli c mnożeń wystarcza do obliczenia iloczynu dwóch dowolnych macierzy $p \times p$ oraz $n = m \cdot p^k$, to minimalna liczba działań $\varphi(n)$ potrzebnych do rozwiązania układu (5.1) spełnia nierówność

$$\varphi(n) \leq O(n^{\log p^c}).$$

A więc, aby otrzymać wynik lepszy od Strassena, należałoby podać np. dla $p = 3$ algorytm mnożenia dwóch dowolnych macierzy 3×3 , wymagający nie więcej niż 21 mnożeń, dla $p = 4$ co najwyżej 48 mnożeń itd. Problem oszacowania $\varphi(n)$ pozostaje nadal otwarty, tylko dla $p = 2$ znany jest cytowany już algorytm, wymagający mniej niż p^3 mnożeń dla obliczenia iloczynu macierzy $p \times p$. Jak dotychczas nie wiadomo także, czy algorytm Strassena jest numerycznie stabilny. Co więcej, jest on lepszy (tzn. szybszy) od algorytmu eliminacji Gaussa tylko dla odpowiednio dużych n . Przy założeniu, że oba algorytmy są programowane w języku Gier-algol oraz, że każde mnożenie i dodawanie związane jest z pobraniem jednego elementu tablicy dwuwskaźnikowej, okazuje się, że koszt odwrócenia macierzy $n \times n$ algorytmem Strassena jest mniejszy od kosztu algorytmu Gaussa dopiero dla $n > 1000$.

W przypadku szczególnych postaci macierzy A znamy algorytmy wykorzystujące efektywnie tę postać i to często w sposób optymalny. Na przykład, dla macierzy wielodiagonalnych w wielu znanych algorytmach liczby danych oraz działań są proporcjonalne do n . Dla układów trójkątnych lub prawie trójkątnych liczba danych i działań w najprostszych znanych algorytmach jest rzędu n^2 . Ciekawą klasę równań liniowych stanowią układy pochodzące z dyskretnej aproksymacji zagadnień brzegowych dla równań różniczkowych. Macierze tych układów są silnie rozrzedzone, co pozwala często (ale nie zawsze) zastosować w sposób efektywny iteracyjne metody rozwiązywania. Typowym zadaniem w tej dziedzinie jest problem rozwiązania układu pochodzącego z pięciopunktowej aproksymacji równania Poissona na prostokącie. Macierz tego układu ma postać

$$M = \begin{bmatrix} T & -I & & & \\ -I & T & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -I \\ & & & & -I & T \end{bmatrix},$$

gdzie T jest macierzą trójdziagonalną $n \times n$, a I macierzą jednostkową $n \times n$ (liczba danych = $O(n^2)$).

Jedną z najbardziej efektywnych metod iteracyjnych, metoda ADI, rozwiązuje ten układ kosztem $O(n^2 \log_2 n \log_2 \frac{1}{\varepsilon})$, gdzie $\varepsilon > 0$ jest zadaną dokładnością rozwiązania. Zastosowanie szybkich transformacji Fouriera pozwala to samo zadanie rozwiązać kosztem $O(n^2 \log_2 n)$ działań, a więc $\log_2 \frac{1}{\varepsilon}$ razy taniej, niż metodą ADI. Co więcej Bank, Birkhoff i Rose [1] podali algorytm rozwiązywania układów tej postaci, wymagający $O(n^2)$ działań, a zatem optymalny. Niestety autorom tego sprawozdania cytowany algorytm nie jest znany i możemy jedynie za Traubem [29] podać, że nie jest on numerycznie stabilny.

6.1. Dotychczas omawialiśmy złożoność obliczeniową problemów, dla których istnieją skończone algorytmy. Teraz przejdziemy do problemów, dla których znamy jedynie algorytmy nieskończone. Problematykę oceny kosztów rozwiązania tych zadań nazywamy za J. F. Traubem (por. [29]) *analityczną złożonością obliczeniową*.

Nasze rozważania skoncentrujemy na iteracyjnym rozwiązywaniu równań nieliniowych. Założmy zatem, że poszukujemy rozwiązania równania

$$(6.1) \quad F(x) = 0$$

dla $F : D \subset \mathbb{C}^N \rightarrow \mathbb{C}^N$, gdzie \mathbb{C}^N oznacza N -wymiarową przestrzeń zespoloną.

Iteracyjne rozwiązywanie (6.1) polega na konstrukcji ciągu $\{x_i\} \subset \mathbb{C}^N$, zbieżnego przy dodatkowych założeniach do rozwiązania α , $F(\alpha) = 0$. Założmy, że znamy $n + 1$ kolejnych przybliżeń $x_i, x_{i-1}, \dots, x_{i-n}$ rozwiązania α . Aby skonstruować następne przybliżenie x_{i+1} korzystamy z pewnej informacji o funkcji F w obliczonych już punktach x_i, \dots, x_{i-n} . Informacja ta może być dana np. poprzez wartości jej pochodnych

$$(6.2) \quad F^{(k)}(x_{i-j}) \quad \text{dla} \quad k = 0, 1, \dots, s; \quad j = 0, 1, \dots, n.$$

Ogólnie możemy założyć, że informacja o funkcji F jest dana przez pewną funkcję

$$(6.3) \quad \mathcal{U} = \mathcal{U}(x_i, \dots, x_{i-n}; F),$$

gdzie $\mathcal{U} : D_{\mathcal{U}} \rightarrow V$, $D_{\mathcal{U}} \subset (\mathbb{C}^N)^{n+1} \times \mathcal{F}$ dla zadanej klasy zadań \mathcal{F} , $F \in \mathcal{F}$, oraz zadanej przestrzeni V (por. [35]). Na ogół elementy V są zbiorami, zawierającymi dyskretne wartości F , oraz jej pochodnych.

Funkcję \mathcal{U} nazywamy *informacją*. Informację daną związkiem (6.3) nazywamy *informacją standardową*.

Przy zadanej funkcji \mathcal{U} niech φ będzie metodą iteracyjną, konstruującą następne przybliżenie, x_{i+1} , w myśl reguły

$$(6.4) \quad x_{i+1} = \varphi(x_i, \dots, x_{i-n}; \mathcal{U}(x_i, \dots, x_{i-n}; F)).$$

(por. [35]).

PRZYKŁAD

1. Metoda *Newtona* korzysta z informacji standardowej, $n = 0, s = 1$,

$$\mathcal{U}(x_i; F) = \{F(x_i), F'(x_i)\}$$

i konstruuje ciąg $\{x_i\}$ zgodnie z wzorem

$$x_{i+1} = x_i - [F'(x_i)]^{-1} F(x_i)$$

(por. np. [20]).

2. Wielowymiarowa metoda *siecznych* korzysta z informacji standardowej $n = N, s = 0$,

$$\mathcal{U}(x_i, \dots, x_{i-N}; F) = \{F(x_i), \dots, F(x_{i-N})\}.$$

Ciąg $\{x_i\}$ jest dany poprzez

$$x_{i+1} = x_i - X_i F_i^{-1} F(x_i),$$

gdzie

$$X_i = [x_i - x_{i-1}, \dots, x_{i-N+1} - x_{i-N}],$$

$$F_i = [F(x_i) - F(x_{i-1}), \dots, F(x_{i-N+1}) - F(x_{i-N})]$$

(por. np. [20], [7]).

Przystąpmy teraz do oceny efektywności metody iteracyjnej φ . Z praktycznego punktu widzenia chcielibyśmy, aby konstruowany ciąg $\{x_i\}$ z (6.4) możliwie szybko dążył do rozwiązania α . Z drugiej strony musimy pamiętać, że wykonanie każdego kroku iteracyjnego wiąże się z określonym kosztem. Przez „koszt” na ogół rozumiemy liczbę działań arytmetycznych potrzebnych do obliczenia żądanej wielkości. Czasami ograniczamy się tylko do liczby mnożeń i dzieleni, ignorując dodawanie i odejmowanie jako działania tańsze. Wielkością charakteryzującą „miarę dobroci” metody φ jest tzw. wskaźnik efektywności $e(\varphi; F)$, definiowany następująco

$$(6.5) \quad e(\varphi; F) = \frac{\log_2 p(\varphi)}{c(\varphi; F)},$$

gdzie $p(\varphi)$ oznacza wykładnik zbieżności φ , a $c(\varphi; F)$ jest miarą kosztu wykonania jednego kroku iteracyjnego (por. np. [3], [16], [29], [30]).

6.2. Omówmy pokrótce przedstawione wielkości $p(\varphi)$ i $c(\varphi; F)$. Wykładnik zbieżności $p(\varphi)$ określa szybkość zbieżności ciągu $\{x_i\}$ do rozwiązania. Zdefiniowanie $p(\varphi)$ w przypadku ogólnym jest sprawą nieco skomplikowaną (por. [4], [33], [34] i [35]). Tu przytoczymy tylko jedną z najprostszych definicji (niezbyt ogólną). *Wykładnikiem zbieżności* $p \doteq p(\varphi)$ nazywamy największą liczbę ≥ 1 , dla której

$$(6.6) \quad \limsup_{i \rightarrow \infty} \frac{\|x_{i+1} - \alpha\|}{\|x_i - \alpha\|^p} < +\infty$$

dla wszystkich funkcji F odpowiednio regularnych, o prostym zerze α (tzn. istnieje $[F'(\alpha)]^{-1}$).

PRZYKŁADY:

1. Dla metody Newtona $p = 2$.

2. Przypadek wielowymiarowej metody siecznych jest dużo bardziej skomplikowany. Na ogół nie możemy zagwarantować zbieżności z wykładnikiem zbieżności większym od jedności (wyjątek stanowi przypadek jednowymiarowy, $N = 1$, gdzie $p = (1 + \sqrt{5})/2$). Kluczową sprawą w tej metodzie jest położenie punktów $x_i, x_{i-1}, \dots, x_{i-N}$ w przestrzeni C^N . Jeśli założymy, że punkty te są odpowiednio położone, to wówczas p jest jedynym dodatnim zerem wielomianu

$$t^{N+1} - t^N - 1$$

(por. [6], [20]).

Należy tu podkreślić, że dla zadań wielowymiarowych, $N \geq 2$, korzystających z informacji $\mathcal{N} = \mathcal{N}(x_i, \dots, x_{i-n}; F)$ dla $n \geq 1$, położenie poprzednich przybliżeń x_i, \dots, x_{i-n} w sposób istotny wpływa na charakter zbieżności. Aby uniezależnić własności badanej metody iteracyjnej od położenia punktów, definiuje się tak zwany zbiór dopuszczalnych przybliżeń \mathcal{M} w przestrzeni C^n i bada wpływ zbioru \mathcal{M} na wykładnik zbieżności metody φ (por. [6], [7] i [35]).

Przejdźmy do omówienia wielkości $c(\varphi; F)$, która oznacza koszt wykonania jednego kroku iteracyjnego. Wielkość ta składa się z dwóch niezależnych od siebie składników

$$(6.7) \quad c(\varphi; F) = c(\mathcal{U}; F) + c(\varphi),$$

gdzie $c(\mathcal{U}; F)$ oznacza koszt otrzymania informacji o funkcji F w punktach x_i, \dots, x_{i-n} , a więc koszt obliczenia $\mathcal{U}(x_i, \dots, x_{i-n}; F)$, natomiast $c(\varphi)$ jest kosztem obliczenia funkcji $\varphi(x_i, \dots, x_{i-n}; \mathcal{U}(x_i, \dots, x_{i-n}; F))$.

PRZYKŁAD.

1. Dla metody Newtona:

$c(\mathcal{U}; F) = < \text{koszt obliczenia } F(x_i) \text{ oraz } F'(x_i) >$,

$c(\varphi) = < \text{koszt rozwiązania układu równań liniowych } F'(x_i) \cdot \Delta x_i = -F(x_i) \text{ oraz do-}$
dania poprawki $x_{i+1} = x_i + \Delta x_i >$.

Ponieważ do rozwiązania układu równań liniowych stosujemy na ogół metodę eliminacji Gaussa z pewnym wariantem wyboru elementu głównego, więc liczba działań arytmetycznych jest rzędu N^3 , a stąd również

$$c(\varphi) = O(N^3).$$

2. Dla wielowymiarowej metody siecznych:

$c(\mathcal{U}; F) = < \text{koszt obliczenia } F(x_i) >$, gdyż wielkości $F(x_{i-1}), \dots, F(x_{i-N})$ były już poprzednio obliczone i mogły być zapamiętane.

$c(\varphi) = < \text{koszt rozwiązania układu równań liniowych}$

$$(6.8) \quad x_i F_i^{-1} \Delta x_i = -F(x_i)$$

oraz obliczenia $x_{i+1} = x_i + \Delta x_i >$.

Z uwagi na specyfikę układu (6.8), jeśli punkty x_i, \dots, x_{i-n} są odpowiednio położone w C^N , to rozwiązanie Δx_i można otrzymać kosztem proporcjonalnym do N^2 działań arytmetycznych, czyli

$$c(\varphi) = O(N^2)$$

(por. [20], [7]).

6.3. Powróćmy do wskaźnika efektywności $e(\varphi; F)$. Najistotniejszą cechą wskaźnika efektywności jest:

Wskaźnik efektywności jest odwrotnie proporcjonalny do ogólnego kosztu otrzymania dostatecznie dobrego przybliżenia rozwiązania.

Oznacza to, że rozwiązując iteracyjnie układ równań (6.1), powinniśmy poszukiwać metod iteracyjnych φ , które mają możliwie duży wskaźnik efektywności.

Badanie analitycznej złożoności obliczeniowej współcześnie koncentruje się na badaniu własności wskaźników efektywności i na doskonaleniu tych metod iteracyjnych, które prowadzą do najlepszych wskaźników efektywności.

U w a g a. Jeśli dla danego zadania (6.1) oraz dla danej informacji \mathcal{U} istnieją tylko metody zbieżne liniowo ($p = 1$ oraz $e(\varphi; F) = 0$), tzn. takie, że

$$\|x_{i+1} - \alpha\| \leq \rho^{i+1} \|x_0 - \alpha\| \quad \text{dla stałej } \rho = \rho(\varphi) < 1,$$

to wówczas definiujemy L -wskaźnik efektywności

$$e_L(\varphi; F) = \frac{-\log_2 \rho(\varphi)}{c(\varphi; F)},$$

i poszukujemy metod φ o możliwie dużym L -wskaźniku efektywności.

6.4. Omówimy teraz pokrótce otrzymane rezultaty dotyczące analitycznej złożoności obliczeniowej. Zauważmy przede wszystkim, że dla wielu zadań koszt wykonania jednego kroku iteracyjnego $c(\varphi; F)$ jest w przybliżeniu równy kosztowi uzyskania informacji $c(\mathcal{I}; F)$, tzn. z (6.7):

$$(6.9) \quad c(\varphi) \ll c(\mathcal{I}; F).$$

Zależność (6.9) jest prawdziwa dla funkcji F , dla których obliczenie informacji \mathcal{I} jest stosunkowo drogie (na przykład, gdy obliczenie $F(x_i)$ wymaga rozwiązania oddzielnego problemu, lub $F(x_i)$ zadane jest skomplikowaną formułą).

PRZYKŁAD. 1. Rozważmy metodę siecznych w przypadku skalarnym, $N = 1$, dla funkcji

$$F(x) = \det(A - xI), \quad A \quad (k \times k).$$

Wówczas $c(\mathcal{I}; F) = O(k^3)$ (ewentualnie wykorzystując algorytm Strassena $c(\mathcal{I}; F) = O(k^{\log_2 7})$), a $c(\varphi) = 4$.

Przyjmując założenie (6.9) otrzymujemy

$$(6.10) \quad e(\varphi; F) \cong \frac{\log_2 p(\varphi)}{c(\mathcal{I}; F)}.$$

Stąd też wynika, że dla ustalonej informacji \mathcal{I} , a więc dla ustalonego kosztu $c(\mathcal{I}; F)$, największy wskaźnik efektywności ma metoda φ o możliwie dużym wykładniku zbieżności $p(\varphi)$. Metody, które przy ustalonej informacji \mathcal{I} mają maksymalnie duży wykładnik zbieżności, nazywamy metodami *maksymalnymi*. Problem poszukiwania metod maksymalnych został po raz pierwszy postawiony przez Trauba ([27], [31]). Początkowe prace dotyczyły informacji standardowej

$$(6.11) \quad \mathcal{I}_{n,s} = \mathcal{I}(x_i, \dots, x_{i-n}; F) = \{F^{(k)}(x_{i-j}); k = 0, 1, \dots, s; j = 0, 1, \dots, n\}.$$

I tak w przypadku skalarnym, $N = 1$, definiuje się interpolacyjną metodę $I_{n,s}$ jak następuje. Niech $w_{r,i}$ będzie wielomianem interpolacyjnym Hermite'a stopnia co najwyżej $r = (n+1)(s+1) - 1$ danym warunkami

$$w_{r,i}^{(k)}(x_{i-j}) = F^{(k)}(x_{i-j}), \quad k = 0, 1, \dots, s; j = 0, 1, \dots, n.$$

Następne przybliżenie x_{i+1} w metodzie $I_{n,s}$ definiuje się jako zero wielomianu $w_{r,i}$, $w_{r,i}(x_{i+1}) = 0$, z pewnym kryterium jego wyboru (np. zero leżące najbliżej x_i), por. [27] i [33]. Dla $n = 0$, Traub [27] oraz Kung i Traub [17] udowodnili, że maksymalny wykładnik zbieżności równa się $s+1$ i jest osiągany dla metody interpolacyjnej $I_{0,s}$ (por. [17], [37]).

Dla $n = 1$ oraz $s = 0$, Rissanen [23] udowodnił maksymalność metody siecznych I_{10}

o wykładniku $p = (1 + \sqrt{5})/2$. Następnie, dla dowolnych wartości n, s , w pracach [33] i [34] udowodniono maksymalność metod interpolacyjnych $I_{n,s}$ o wykładniku $p_{n,s}$ równym jednemu dodatniemu zeru wielomianu

$$t^{n+1} - (s+1) \sum_{j=0}^n t^j, \quad s+1 \leq p_{n,s} < s+2, \quad \lim_n p_{n,s} = s+2.$$

Ponadto Brent, Winograd i Wolfe [4] udowodnili, że w przypadku tak zwanych niestacjonarnych metod iteracyjnych dla dowolnej wartości n wykładnik zbieżności jest $\leq s+2$. W przypadku wielowymiarowym ($N \geq 2$) w pracach [33] i [34] udowodniono, że dla dowolnej wartości n maksymalny wykładnik zbieżności jest równy $s+1$ i jest osiągany dla metody interpolacyjnej $I_{0,s}^N$. Rezultat ten oznacza, że informacja standardowa zawarta w $F^{(k)}(x_{i-j})$ dla $k = 0, 1, \dots, s; j = 1, \dots, n$ nie może zwiększyć wykładnika zbieżności. Warunkiem koniecznym na właściwe wykorzystanie tej informacji jest dodatkowe założenie o położeniu punktów $x_i, x_{i-1}, \dots, x_{i-n}$ w przestrzeni C^N , co prowadzi do wspomnianego już zbioru dopuszczalnych przybliżeń (por. [35]).

Przedstawione powyżej rezultaty uzyskano, narzucając na rozważane metody iteracyjne pewne na ogół mało krępujące warunki regularności. W pracy [35] rozważano problem metod maksymalnych korzystających z dowolnej informacji \mathcal{I} przy zadanym zbiorze dopuszczalnych przybliżeń \mathcal{M} . Zdefiniowano pojęcie wykładnika informacji $p(\mathcal{I}; \mathcal{M})$ i udowodniono, że maksymalny wykładnik zbieżności jest równy wykładnikowi informacji i jest osiągany dla uogólnionej metody interpolacyjnej $I_{\mathcal{I}, \mathcal{M}}$.

Powyższy rezultat, dzięki zmodyfikowanej definicji wykładnika zbieżności, nie wymaga narzucenia warunków na rozważaną klasę metod iteracyjnych. Przykładem pracy, gdzie rozważa się informację całkową

$$\mathcal{I}(x_i; F) = \left\{ F(x_i), \dots, F^{(s)}(x_i), \int_0^1 F(x_i + t(y_i - x_i)) dt \right\},$$

dla odpowiednio dobranych y_i , są prace [8], [9], gdzie udowodniono, że wykładnik informacji (a tym samym i maksymalny wykładnik zbieżności) jest równy

$$p = \begin{cases} s+3 & \text{dla } N=1 \text{ lub } (N \geq 2 \text{ i } s \geq 2), \\ 3 & \text{dla } N \geq 2 \text{ i } s=1. \end{cases}$$

Pamiętając, że maksymalny wykładnik zbieżności dla informacji $F(x_i), \dots, F^{(s)}(x_i)$ wynosi $s+1$, widzimy, że powiększenie informacji standardowej o wartość całki $\int_0^1 F(x_i + t(y_i - x_i)) dt$, zwiększa maksymalny wykładnik zbieżności o jeden dla $N \geq 2$ i $s=1$ a w pozostałych przypadkach o dwa. Wydaje się, że szczególnie dla układów równań ($N \geq 2$) otrzymany wynik jest ciekawy, gdyż całka jest reprezentowana poprzez N liczb a ewentualne wykorzystywanie kolejnych pochodnych $F^{(k)}(x)$ wiąże się ze wzrostem liczby danych proporcjonalnym do N^k .

Wpływ zbioru dopuszczalnych przybliżeń \mathcal{M} na charakter zbieżności w przypadku wie-

lowymiarowej metody siecznych jest przedstawiony w pracy [7]. Pokazano tam, że maksymalny wykładnik zbieżności przy różnych zbiorach \mathcal{M} zmienia się w przedziale $[1, p_N]$, gdzie p_N jest jedynym dodatnim zerem wielomianu $t^{N+1} - t^N - 1$. Wykorzystanie informacji standardowej $\mathcal{M}_{n,s}$ (6.11) dla zadań wielowymiarowych, $N \geq 2$, przy założonych zbiorach dopuszczalnych przybliżeń \mathcal{M} , prowadzi do maksymalnego wykładnika zbieżności $p_{n,s}$ równego dodatniemu zeru wielomianu

$$t^{\gamma_r} - \sum_{k=0}^r t^{\gamma_r - \gamma_k},$$

gdzie r dane jest przez warunek

$$\binom{N+r}{r} \leq (n+1) \binom{N+s}{s} < \binom{N+r+1}{r+1},$$

a γ_k jest najmniejszą liczbą naturalną $\geq \binom{N+k}{k} / \binom{N+s}{s}$ (por. [3] dla $s = 0$ i [32] dla dowolnego s).

Omówimy teraz tak zwane wielopunktowe metody iteracyjne bez pamięci ($n = 0$, por. [27]), w których informacja jest postaci

$$\mathcal{M}(x_1; F) = \{F^{(j_1^1)}(x_1), \dots, F^{(j_k^1)}(x_1), \dots, F^{(j_1^r)}(x_r), \dots, F^{(j_k^r)}(x_r)\},$$

gdzie $j_\nu^\mu, \mu = 1, 2, \dots, r, \nu = 1, 2, \dots, k_\mu$, oznaczają nieujemne liczby całkowite, przy czym

$$k_1 + k_2 + \dots + k_r = k,$$

dla zadanej z góry liczby k . Ponadto punkt $x_{\nu+1}$ jest funkcją poprzednich punktów i informacji, tzn.

$$x_{\nu+1} = x_{\nu+1}(x_1, x_2, \dots, x_\nu, F^{(j_1^1)}(x_1), \dots, F^{(j_{k_\nu}^1)}(x_1), \dots, F^{(j_1^\nu)}(x_\nu), \dots, F^{(j_{k_\nu}^\nu)}(x_\nu))$$

dla $\nu = 1, 2, \dots, r-1$ (por. [17]).

Pytanie postawione przez Kunga i Trauba jest następujące: Jaki jest maksymalny wykładnik zbieżności przy ustalonym k ? Tzn., jak należy określić r, x_2, x_3, \dots, x_r oraz j_ν^μ , aby otrzymać informację o możliwie dużym wykładniku informacji $p(\mathcal{M})$. Dla przypadku skalarnego ($N = 1$), Kung i Traub postawili hipotezę:

$$p(\mathcal{M}) \leq 2^{k-1}.$$

W tej chwili wiadomo, że wykładnik ten jest osiągalny dla informacji postaci

$$(i) \quad F(x_1), \dots, F(x_k)$$

lub

$$(ii) \quad F(x_1), F'(x_1), F(x_2), \dots, F(x_{k-1})$$

dla odpowiednio dobranych x_2, \dots, x_k .

Ponadto, Kung i Traub w [18] udowodnili prawdziwość tej hipotezy dla $k = 1$ i 2 w pewnej klasie metod iteracyjnych. Opierając się na pojęciu wykładnika informacji można udowodnić tę hipotezę dla $k = 3$ oraz dla dowolnego k w przypadku $j_v^\mu = v - 1$.

6.5. Powróćmy raz jeszcze do wskaźnika efektywności $e(\varphi; F)$ danego przez (6.5). Przyjmijmy teraz, że nie możemy w związku (6.7) pominąć kosztu $c(\varphi)$. Określając koszt $c(\varphi; F)$ jako liczbę mnożeń potrzebnych do wykonania jednego kroku iteracyjnego, Kung [13] udowodnił, że zawsze

$$e(\varphi; F) \leq 1,$$

a dla szczególnych funkcji F powyższe oszacowanie jest osiągane dla metody Newtona w przypadku skalarnym. Oznacza to, że wykonując $k = c(\varphi; F)$ mnożeń, można osiągnąć wykładnik zbieżności co najwyżej równy 2^k . Ponadto dla równań skalarnych Kung i Traub [16] udowodnili ciekawe oszacowania na wskaźnik efektywności. Zacytujmy dla przykładu, że dla informacji standardowej przy $n = 0$, mamy

$$\frac{\log_2(s+1)}{c(\mathcal{N}, F) + \rho(s+1)^2 \log_2(s+1)} \leq e(\varphi; F) \leq \frac{\log_2(s+1)}{c(\mathcal{N}, F) + s}$$

dla pewnej stałej $\rho > 0$.

6.6. Rozdział ten kończymy pewnymi uwagami ogólnymi. Wskaźnik efektywności stanowi asymptotyczną własność metody iteracyjnej. Nie charakteryzuje on wielkości stałej asymptotycznej, ani obszaru zbieżności metody, ani osiągalnej w rachunku numerycznym dokładności. Dlatego przedstawione rezultaty należy rozumieć tylko jako wskazówki dla wyboru optymalnej taktyki obliczeniowej i to nie dla określonego zadania, lecz raczej dla pewnej klasy zadań. W przypadku konkretnej funkcji F możemy niekiedy wykorzystać specyficzne własności tej funkcji i w rezultacie zaproponować efektywniejszą metodę, niż metoda optymalna w szerokiej klasie zadań.

Dla wielu zadań w praktyce obliczeniowej kluczową sprawą jest zapewnienie zbieżności. Korzystając z informacji standardowej na ogół musimy dodatkowo założyć, że przybliżenia początkowe są dostatecznie bliskie rozwiązaniu. Ma to miejsce np. dla często używanych metod Newtona i siecznych. Podanie przybliżeń początkowych bliskich rozwiązaniu jest na ogół trudne, a stosowanie losowych przybliżeń początkowych często oznacza, że początkowe wyrazy konstruowanego ciągu $\{x_i\}$ wcale nie muszą zbliżać się do rozwiązania. Dopiero z chwilą gdy jeden z wyrazów x_i znajdzie się w obszarze zbieżności metody, uzyskujemy zbieżność procesu, asymptotycznie tym szybszą, im większy jest wykładnik zbieżności. Poszukiwanie metod iteracyjnych o zapewnionej zbieżności jest sprawą bardzo istotną. Warto tu wspomnieć o nowej pracy Micchelliego i Mirankera [19], gdzie zdefiniowano dla rzeczywistych równań skalarnych maksymalne metody iteracyjne o wykładnikach zbieżności większych od jedności, które w określonej klasie zadań konstruują ciągi na pewno zbieżne do rozwiązania. Metody te wykorzystują informację postaci

$$\mathcal{N}(x_i, \dots, x_{i-n}; F) = \{F(x_i), \dots, F(x_{i-n}), m_i, M_i\},$$

gdzie $x_i \leq x_{i-1} \leq \dots \leq x_{i-n}$, natomiast m_i, M_i spełniają związek $m_i \leq F^{(k)}(x) \leq M_i$ dla $x \in [x_i, x_{i-n}]$ i dla pewnej liczby $k \leq n$.

Chcielibyśmy poruszyć jeszcze jedną bardzo istotną sprawę. Do tej pory charakteryzowaliśmy rozważane metody poprzez pewne ich własności teoretyczne. Warunkiem koniecznym stosowalności tych metod w praktyce obliczeniowej jest podanie algorytmu numerycznie stabilnego. Musimy przyznać, że problemy analizy numerycznej procesów iteracyjnych nastrożają spore trudności i na razie nie potrafimy odpowiedzieć na wiele pytań związanych z możliwością dobrej numerycznej realizacji tych metod. Analizę numeryczną wielowymiarowej metody siecznych oraz metod interpolacyjnych $I_{n,s}$ w przypadku skalarnym i $I_{0,s}$ w przypadku wielowymiarowym zawierają prace [7] i [36].

6.7. W rozdziale 6 omówiliśmy analityczną złożoność obliczeniową na przykładzie iteracyjnego rozwiązywania równań nieliniowych. Oczywiście tematyka ta dotyczy dowolnych zadań, dla których stosujemy algorytmy nieskończone. Wystarczy tu wymienić takie problemy, jak iteracyjne rozwiązywanie wielkich układów równań liniowych, znajdowanie minimum funkcji jednej i wielu zmiennych, numeryczne rozwiązywanie równań różniczkowych, szczególnie problemy brzegowe dla równań eliptycznych czy też problemy aproksymacji funkcji. Nie omawiamy dokładnie tych spraw odsyłając zainteresowanych do przytoczonej literatury (por. np. książka [38] pod redakcją Trauba i praca [22]).

7. Uwagi końcowe. Do niedawna metody numeryczne przypominały ogromną „książkę kucharską”, w której dla jednego problemu można było na ogół znaleźć wiele metod jego rozwiązywania. Proponowane metody różniły się często tylko drugorzędnymi szczegółami i nieraz tylko „wtajemniczeni” wiedzieli, jak porównywać te metody i w rezultacie, jak wybierać metody najlepsze. Wydaje się, że dalszy rozwój metod numerycznych będzie polegał nie tyle na powiększaniu już i tak bogatego zbioru metod, lecz raczej na badaniu złożoności obliczeniowej problemów i poszukiwaniu metod optymalnych. Będziemy więc zapewne obserwowali w najbliższych latach kompletowanie zbioru „prawdziwych” metod numerycznych, tzn. metod numerycznie stabilnych i optymalnych w sensie przyjętego kryterium. W zbiorze tym prawdopodobnie nie znajdzie się wiele z obecnie stosowanych metod. Wydaje się nam, że ocena każdej metody obliczeniowej dla danego problemu powinna zmierzać do uzyskania odpowiedzi na następujące pytania:

1. Co wiemy o złożoności obliczeniowej badanego problemu?
2. Jaki jest koszt i charakterystyka pozostałych własności istniejących najlepszych metod rozwiązujących dany problem?
3. Czy rozpatrywana metoda jest optymalna, tzn. czy miara jej kosztu jest równa złożoności obliczeniowej? Jeśli nie, to czy jest to najtańsza ze znanych metod?
4. Czy oceniana metoda jest numerycznie stabilna?

Jeśli odpowiedzi na dwa pytania ostatnie są pozytywne, to na pewno mamy do czynienia z metodą zalecaną do stosowania w praktyce. Jeśli nie potrafimy odpowiedzieć chociażby na jedno z tych pytań, to nie posiadamy pełnej wiedzy o rozważanym problemie. W tej chwili tylko dla nielicznych problemów znamy optymalne metody obliczeniowe. Mamy jednak nadzieję, że dalsze badania pozwolą poznać złożoność obliczeniową wielu dalszych problemów i być może uda się skonstruować „prawdziwe” metody numeryczne dla tych zadań.

Pragniemy złożyć wyrazy podziękowania doc. dr hab. Andrzejowi Kiełbasińskiemu za cenne rady i pomoc w przygotowaniu tej pracy.

Literatura

- [1] R. Bank, G. Birkhoff and D. J. Rose, *An $O(N^2)$ Method for Solving Constant Coefficient Boundary Value Problems in Two Dimensions*, Report, Harvard University, 1973.
- [2] A. Borodin, *Horner's Rule is Uniquely Optimal*, Theory of Machines and Computations, red. Kahavi i Paz, 1971, str. 45–58.
- [3] R. P. Brent, *The computational complexity of iterative methods for systems of nonlinear equations*, Proc. Complexity Symposium, Yorktown Heights, March 1972.
- [4] R. P. Brent, S. Winograd and P. Wolfe, *Optimal iterative process for root-finding*, Num. Math. 20 (1973), str. 327–341.
- [5] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comput. 19 (1965), str. 297–301.
- [6] J. Jankowska, *O wielowymiarowej metodzie siecznych*, referat na naradzie w Solinie, 1973.
- [7] – *Multivariate secant method*, w przygotowaniu Uniwersytet Warszawski, 1974.
- [8] B. Kaciewicz, *Iteracyjna metoda całkowa rozwiązywania równań skalarnych*, referat na naradzie w Solinie, 1973.
- [9] – *Iteracyjna metoda całkowa dla układów równań nieliniowych*, praca magisterska, Uniwersytet Warszawski, 1974.
- [10] R. M. Karp, *Wykłady w Carnegie-Mellon* (nieopublikowane).
- [11] A. Kiełbasiński, *Podstawowe pojęcia analizy błędów w metodach numerycznych algebry liniowej*, Matematyka Stosowana 4 (1975), str. 5–27.
- [12] К л ю е в и К о к о в к и н - Щ е р б а к, *О минимализации числа арифметических операций при решении линейных алгебраических систем уравнений*, Ж. В. М. М. Ф. 5, № 1, (1965), str. 21–33.
- [13] H. T. Kung, *A bound on the multiplication efficiency of iteration*, JCSS 7 (4) (1973), str. 334–342.
- [14] – *Fast evaluation and interpolation*, Report of Computer Science Department of Carnegie-Mellon University, January 1973.
- [15] – *The computational complexity of algebraic numbers*, ukaże się w SIAM Journal on Numerical Analysis.
- [16] H. T. Kung and J. F. Traub, *Computational complexity of one-point and multipoint iteration*, Report, Carnegie-Mellon University, 1973. Ukaże się w Complexity of Real Computation pod red. R. Karpa, American Mathematical Society.
- [17] – *Optimal order of one-point and multipoint iteration*, Journal ACM, Vol. 21, No. 4, October 1974.
- [18] – *Optimal order and efficiency for iterations with two evaluations*, Report, Carnegie-Mellon University, 1973.
- [19] C. A. Micchelli and W. L. Miranker, *High order search methods for finding roots*. Ukaże się w Journal of the ACM.
- [20] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, 1970.
- [21] C. H. Reinsch, wykład w Technische Hochschule, München (nieopublikowane).
- [22] J. R. Rice, *On the computational complexity of approximation operations*, Report, Purdue University, 1973.
- [23] J. Rissanen, *On optimum root-finding algorithms*, Journ. Math. Anal. Appl. 36 (1971), str. 220–225.
- [24] M. Shaw and J. F. Traub, *On the number of multiplications for the evaluation of a polynomial and some of its derivatives*, Journal ACM, 1974.
- [25] V. Strassen, *Gaussian elimination is not optimal*, Num. Math. 13 (1969), str. 354–356.
- [26] – *Die Berechnungskomplexität von elementar symmetrischen Funktionen und von Interpolationskoeffizienten*, Num. Math. 20 (1973), str. 238–251.
- [27] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice Hall, 1964.
- [28] – *Computational complexity of iterative processes*, SIAM J. Comput. 1 (1972), str. 167–179.
- [29] – *An introduction to some current research in numerical computational complexity*, Report, Carnegie-Mellon University, 1973.
- [30] – *Theory of optimal algorithms*, Report, Carnegie-Mellon University, 1973.

- [31] J. F. Traub, *On functional iteration and the calculation of roots*, Proceedings 16th National ACM Conference, 1961,
 - [32] H. Woźniakowski, *O nieliniowych procesach iteracyjnych w metodach numerycznych*, praca doktorska, Uniwersytet Warszawski, 1972.
 - [33] – *Iteracyjne metody maksymalne rozwiązywania równań nieliniowych*, sprawozdania IMM UW i ZON UW, zeszyty 38 i 44 (1973 i 1974).
 - [34] – *Maximal stationary iterative methods for the solution of operator equations*, SIAM J. Numer. Anal. Vol. 11, No. 5, October 1974.
 - [35] – *Generalized information and maximal order of iteration for operation equations*, SIAM J. Numer. Anal. Vol. 12, No. 1, March 1975.
 - [36] – *Analiza numeryczna interpolacyjnych metod iteracyjnych rozwiązywania układów równań nieliniowych* (w przygotowaniu).
 - [37] – *Analiza numeryczna algorytmów obliczania wartości wielomianów i ich pochodnych*, Matematyka Stosowana 3 (1974), str. 79–92. (por. także angielską wersję SIAM J. Numer. Anal. Vol. 11, No. 4, September 1974).
 - [38] *Complexity of Sequential and Parallel Numerical Algorithms* (pod red. J. F. Trauba), Academic Press, 1973.
-