

**Jerzy Krupka
Andrzej Miękina
Roman Z. Morawski
Leszek J. Opalski**

Wstęp do metod numerycznych

**dla studentów elektroniki
i technik informacyjnych**

Praca zbiorowa pod redakcją
Romana Z. Morawskiego

**OFICYNA WYDAWNICZA POLITECHNIKI WARSZAWSKIEJ
WARSZAWA 2009**

Opiniodawcy

Michał Mrozowski

Andrzej Napieralski

Do książki dołączono płytę CD (gratis) z programami opisanymi w podręczniku.

Jest ona przeznaczona wyłącznie do użytku indywidualnego

lub edukacyjnego i nie może być używana do celów komercyjnych.



Opracowanie redakcyjne

Teresa Woźniak

Projekt okładki

Danuta Czudek-Puchalska

Skład komputerowy

Hanna Jakubicka

Andrzej Kowalczyk

© Copyright by Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1999

ISBN 978-83-7207-802-5

Księgarnia internetowa Oficyny Wydawniczej PW www.wydawnictwopw.pl

tel.: 0-22 825-75-18, 0-22 234-75-03; fax 0-22 234-70-60; e-mail: oficyna@wpw.pw.edu.pl

Oficyna Wydawnicza PW, ul. Polna 50, 00-644 Warszawa. Wydanie II zm. Zamówienie nr 136/2008

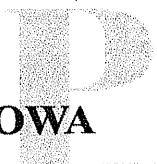
EN 11K / 024-04

SPIS TREŚCI



PRZEDMOWA	9
SYSTEM OZNACZEŃ	11
1. WPROWADZENIE	13
1.1. Komputer w rozwiązywaniu zadań inżynierskich	13
1.2. Modelowanie matematyczne jako podstawa obliczeń naukowo-technicznych	18
1.3. Modelowanie matematyczne w pracy inżyniera	22
1.3.1. Modelowanie matematyczne w projektowaniu	22
1.3.2. Modelowanie matematyczne w pomiarach	23
1.3.3. Zastosowanie komputera do modelowania matematycznego	25
1.4. Algorytm numeryczny i formy jego zapisu	26
1.5. Dokładność obliczeń inżynierskich	27
1.5.1. Identyfikacja problematyki	27
1.5.2. Reprezentacja liczb i zaokrąglanie wyników obliczeń w komputerze	28
1.5.3. Ogólny model przenoszenia błędów w algorytmie numerycznym	30
1.6. Złożoność obliczeń inżynierskich	30
2. PODSTAWOWE METODY ANALIZY DOKŁADNOŚCI ALGORYTMÓW NUMERYCZNYCH	33
2.1. Liniowy model przenoszenia błędów	33
2.2. Przenoszenie błędów danych	37
2.2.1. Współczynniki przenoszenia błędów danych	37
2.2.2. Uwarunkowanie numeryczne zadania	40
2.3. Przenoszenie błędów zaokrągleń	41
2.3.1. Współczynniki przenoszenia błędów zaokrągleń	41
2.3.2. Numeryczna poprawność algorytmu	43
2.4. Analiza dokładności algorytmów za pomocą komputera	44
2.5. Pozanumeryczne zastosowania metod analizy dokładności	47
3. ELEMENTY ANALIZY ALGORYTMÓW ITERACYJNYCH	50
3.1. Informacje wstępne	50
3.2. Algorytmy iteracyjne jednoargumentowe	52
3.3. Algorytmy iteracyjne wieloargumentowe	57
3.4. Analiza algorytmów iteracyjnych wspomagana komputerem	63
3.5. Pozanumeryczne zastosowania metod analizy algorytmów iteracyjnych	64

4. ROZWIĄZYwanie LINIOWYCH RÓWNAŃ ALGEBRAICZNYCH	67	7.3. Różniczkowanie funkcji jednej zmiennej	147
4.1. Pojęcia podstawowe	68	7.3.1. Formuły różnicowe	147
4.1.1. Macierze	68	7.3.2. Różniczkowanie formuł interpolacyjnych	151
4.1.2. Normy wektorów i macierzy	70	7.3.3. Zwiększanie dokładności różniczkowania metodą ekstrapolacji RichardsoNa	153
4.1.3. Uwarunkowanie zadania rozwiązywania układu liniowych równań algebraicznych	72	7.3.4. Różniczkowanie formuł aproksymacji wygładzającej	155
4.2. Metoda eliminacji Gaussa	75	8. CAŁKOWANIE FUNKCJI – METODA MONTE CARLO	157
4.3. Metody iteracyjne	78	8.1. Wprowadzenie do metody Monte Carlo	157
4.3.1. Metoda Jacobiego	79	8.2. Metody estymacji wartości oczekiwanej zmiennej losowej	160
4.3.2. Metoda RichardsoNa	80	8.3. Proste metody Monte Carlo	163
4.3.3. Metoda Gaussa-Seidla	81	8.3.1. Wariant podstawowy w wersji ogólnej	163
4.3.4. Metoda SOR	81	8.3.2. Wariant podstawowy w wersji „orzeł-reszka”	164
4.3.5. Porównanie zbieżności metod iteracyjnych	82	8.4. Złożone metody Monte Carlo	167
4.4. Liniowe zadanie najmniejszych kwadratów	84	8.4.1. Metoda losowania ważonego	167
4.5. Podsumowanie	88	8.4.2. Metoda zmiennej kontrolnej	168
5. ROZWIĄZYwanie NIELINIOWYCH RÓWNAŃ ALGEBRAICZNYCH	89	8.4.3. Metoda oparta na obniżaniu krotności całki	168
5.1. Rozwiązywanie skalarnych równań nieliniowych	89	8.4.4. Metoda losowania warstwowego	171
5.1.1. Metoda bisekcji	89	8.5. Metody generacji zmiennych losowych	171
5.1.2. Metoda stycznych (Newtona) i metoda siecznych	91	8.5.1. Generatory liczb pseudolosowych o rozkładzie równomiernym	172
5.1.3. Metody wyznaczania zer wielomianów	93	8.5.2. Generatory liczb pseudolosowych o zadanym rozkładzie prawdopodobieństwa	173
5.2. Rozwiązywanie układów równań nieliniowych	97	9. ROZWIĄZYwanie RÓWNAŃ RÓŻNICZKOWYCH ZWYCZAJNYCH	176
5.2.1. Wielowymiarowa metoda Newtona	97	9.1. Podstawowe własności metod rozwiązywania równań różniczkowych zwyczajnych	179
5.2.2. Wielowymiarowa metoda siecznych	98	9.2. Metody jednokrokowe typu Rungego-Kutty	180
5.2.3. Uwagi praktyczne	99	9.2.1. Konstrukcja i własności metod jednokrokowych	180
6. INTERPOLACJA I APROKSYMACJA FUNKCJI	102	9.2.2. Wybór kroku całkowania	188
6.1. Interpolacja przy użyciu wielomianów Newtona i Lagrange'a	102	9.3. Metody wielokrokowe	195
6.1.1. Zależności ogólne	102	9.3.1. Konstrukcja i własności metod wielokrokowych	195
6.1.2. Zależności dla węzłów równoodległych	104	9.3.2. Stabilność numeryczna metod wielokrokowych	198
6.2. Interpolacja przy użyciu wielomianowych funkcji sklejanych	105	9.3.3. Schemat predykt-or-korektor	208
6.3. Interpolacja trygonometryczna	107	9.3.4. Wybór kroku całkowania i rzędu metody	210
6.4. Aproksymacja średniokwadratowa	110	9.3.5. Rozwiązywanie układów równań różniczkowo-algebraicznych	212
6.4.1. Aproksymacja funkcji danej w postaci analitycznej	110	DODATEK: PROGRAMY W JĘZYKU MATLAB	216
6.4.2. Aproksymacja funkcji na podstawie ciągu jej dyskretnych wartości	115	Programy do rozdziału 4	217
6.5. Inne rodzaje aproksymacji	120	Programy do rozdziału 5	227
6.5.1. Aproksymacja jednostajna	120	Programy do rozdziału 6	236
6.5.2. Aproksymacja funkcjami nieliniowymi względem parametrów	121	Programy do rozdziału 7	250
7. CAŁKOWANIE I RÓŻNICZKOWANIE FUNKCJI – METODY KLASYCZNE	124	Programy do rozdziału 8	267
7.1. Całkowanie funkcji jednej zmiennej	124	Programy do rozdziału 9	273
7.1.1. Proste kwadratury interpolacyjne Newtona-Cotesa	125	LITERATURA	299
7.1.2. Złożone kwadratury interpolacyjne Newtona-Cotesa	129	SKOROWIDZ RZECZOWY	300
7.1.3. Kwadratury interpolacyjne Gaussa	135		
7.1.4. Przyspieszanie zbieżności kwadratur metodą ekstrapolacji RichardsoNa	140		
7.1.5. Obliczanie całek z osobliwościami i całek niewłaściwych	142		
7.2. Całkowanie funkcji wielu zmiennych	145		



PRZEDMOWA

Niniejszy podręcznik jest drugim – zmienionym – wydaniem podręcznika wydanego przez autorów w roku 1999. Zupełnie nowym elementem tego wydania są programy do przykładów i zadań, które wymagają użycia komputera. Powstały one w odpowiedzi na wielokrotnie wyrażane zapotrzebowanie studentów, którzy – jak pokazuje doświadczenie dydaktyczne – najszybciej uczą się sztuki programowania przez wzorowanie się na dobrych przykładach.

Pierwsze wydanie tego podręcznika opracowane zostało z myślą o studentach Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej – studiujących według, wprowadzonego w połowie lat dziewięćdziesiątych ubiegłego wieku, programu ośmiosemestralnych studiów pierwszego stopnia na makrokierunku *elektronika i techniki informacyjne*. Przez wiele lat stanowiło podstawowy materiał dydaktyczny do przedmiotu „wstęp do metod numerycznych”, prowadzonego dla studentów trzeciego semestru w wymiarze dwóch godzin wykładu i jednej godziny ćwiczeń laboratoryjnych tygodniowo. W związku ze skróceniem cyklu studiów do siedmiu semestrów, a także z obniżeniem się ogólnego poziomu przygotowania studentów w zakresie matematyki, program tego przedmiotu ulegał ostatnio istotnemu zubożeniu. Ewolucja podręcznika poszła jednak w przeciwnym kierunku: został on istotnie wzbogacony. Jego drugie wydanie zostało zredagowane w taki sposób, aby mogło być wyczerpującym materiałem dydaktycznym do przedmiotu „wstęp do metod numerycznych”, a jednocześnie służyć, jako materiał pomocniczy, słuchaczom bardziej zaawansowanych przedmiotów dotyczących metod numerycznych.

Głównym celem przedmiotu „wstęp do metod numerycznych” jest zapoznanie studentów z metodą systematycznego badania przydatności algorytmów numerycznych do rozwiązywania zadań inżynierskich oraz z wybranymi algorytmami numerycznymi. Do pełnego zrozumienia wykładu wymagana jest znajomość matematyki w zakresie pierwszego roku studiów technicznych oraz elementów teorii obwodów elektrycznych. Program zajęć laboratoryjnych zakłada natomiast umiejętność programowania w języku środowiska MATLAB. Podczas tych zajęć bowiem studenci opracowują, uruchamiają i badają proste programy komputerowe do rozwiązywania problemów numerycznych dotyczących elektroniki, metrologii, radioelektroniki i telekomunikacji.

Koncepcja programowa i metodyczna przedmiotu „wstęp do metod numerycznych” istotnie odbiega od tradycyjnego ujęcia problematyki metod numerycznych, reprezentowanego przez wiele dostępnych na naszym rynku księgarskim podręczników z tego zakresu, zarówno tych przeznaczonych dla matematyków, jak tych adresowanych do in-

żynierów i adeptów nauk przyrodniczych. Koncepcja ta opiera się na następujących pre-
słankach:

- Powszechnie użycie komputerów w projektowaniu inżynierskim, pomiarach i sterowaniu oznacza konieczność zapoznania z metodami numerycznymi każdego studenta elektroniki i technik informacyjnych.
- Powszechna dostępność bibliotek oprogramowania zawierających procedury numeryczne oznacza, że maleje znaczenie umiejętności tworzenia takich procedur przez inżyniera, a rośnie znaczenie umiejętności ich systematycznego badania, oceny i adaptacji do specyficznych warunków działania.
- Dane do obliczeń technicznych pochodzą z pomiarów lub obliczeń przybliżonych, a komputer nie tylko ze skończoną dokładnością reprezentuje dane, ale też zaokrąglą wyniki poszczególnych operacji na tych danych; wynika stąd konieczność ciągłej kontroli dokładności prowadzonych obliczeń.

Konsekwencją powyższych konstatacji jest:

- ograniczenie w niniejszym podręczniku formalizmu matematycznego i matematycznych uzasadnień algorytmów numerycznych do minimum wyznaczonego strukturą logiczną typowych bibliotek oprogramowania numerycznego;
- wprowadzenie metodyki analizy dokładności algorytmów numerycznych w rozdziałach 1–3 i szerokie jej wykorzystywanie w rozdziałach 4–9;
- demonstracja zjawisk numerycznych, takich jak wzmacnianie błędów danych i błędów zaokrągleń operacji zmiennopozycyjnych, na najprostszych (np. skalarnych) wersjach podstawowych zadań numerycznych, jakie pojawiają się w dziedzinie elektroniki, metrologii i telekomunikacji;
- dołączenie programów w języku środowiska MATLAB użytych do rozwiązania wszystkich przykładów opisanych w podręczniku i zadań do samodzielnego rozwiązania (w dodatku i na CD).

W ogólności chodzi bowiem o to, aby z jednej strony umotywować studenta do szerskiego, ale krytycznego korzystania z procedur numerycznych zawartych w dostępnych bibliotekach oprogramowania, z drugiej zaś – wytworzyć nawyk systematycznego sprawdzania technicznych skutków tego rodzaju działania poprzez pomiarowo-obliczeniową analizę dokładności procedur numerycznych zastosowanych do rzeczywistych danych technicznych.

Koncepcja programowa i metodyczna podręcznika wyrasta z wieloletnich doświadczeń dydaktycznych autorów w zakresie zastosowań metod i technik obliczeniowych w pracy inżyniera. Nie są to – jak się wydaje – doświadczenia specyficzne dla praktyki kształcenia studentów na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Podręcznik ten może być więc w całości wykorzystywany przez studentów innych uczelni kształcących w obszarze elektroniki i technik informacyjnych, a w znacznej części – przez studentów innych kierunków.

Autorzy

Zasady

a, b, c, \dots	zmienne skalarne, zdeterminowane
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$	zmienne wektorowe, zdeterminowane
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \dots$	zmienne macierzowe, zdeterminowane
$\underline{a}, \underline{b}, \underline{c}$	zmienne skalarne, losowe
$\underline{\mathbf{a}}, \underline{\mathbf{b}}, \underline{\mathbf{c}}, \dots$	zmienne wektorowe, losowe
$\underline{\mathbf{A}}, \underline{\mathbf{B}}, \underline{\mathbf{C}}, \dots$	zmienne macierzowe, losowe
$\mathbb{A}, \mathbb{B}, \mathbb{C}, \dots$	zbiory
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \dots$	operatorzy
\tilde{x}	zaburzona zmienna x
\hat{x}	estymata zmiennej x
$f'(x), f''(x), f'''(x)$	pierwsza, druga i trzecia pochodna funkcji $f(x)$
$f^{(k)}(x)$	k -ta pochodna funkcji $f(x)$

Ważniejsze symbole

$\mathbb{N}, \mathbb{R}, \mathbb{C}$	zbiory liczb: naturalnych, rzeczywistych i zespolonych
$J(\mathbf{p})$	kryterium optymalizacji
$\inf \{f(x) \mid x \in \mathbb{X}\}$	operator wyznaczania kresu dolnego funkcji $f(x)$, gdy $x \in \mathbb{X}$
$\sup \{f(x) \mid x \in \mathbb{X}\}$	operator wyznaczania kresu górnego funkcji $f(x)$, gdy $x \in \mathbb{X}$
$\Pr[\cdot]$	operator wyznaczania prawdopodobieństwa
$E[\cdot]$	operator wyznaczania wartości oczekiwanej
$\text{Var}[\cdot]$	operator wyznaczania wariancji
μ_x	wartość oczekiwana zmiennej losowej x
σ_x^2	wariancja zmiennej losowej x
$f_x(v)$	funkcja gęstości rozkładu prawdopodobieństwa zmiennej losowej x
$F_x(v)$	dystrybuanta zmiennej losowej x

Głównym celem niniejszego podręcznika jest zapoznanie Czytelnika z metodyką systematycznego badania przydatności algorytmów numerycznych do rozwiązywania zadań inżynierskich oraz z wybranymi algorytmami numerycznymi. Osiągnięcie tego celu byłoby niezwykle trudne bez odwołania się do kilku podstawowych pojęć z dziedziny informatyki stosowanej i modelowania matematycznego obiektów fizycznych. Ich wprowadzeniu i interpretacji poświęcony jest właśnie ten rozdział.

1.1. KOMPUTER W ROZWIĄZYWANIU ZADAŃ INŻYNIERSKICH

Komputery stały się urządzeniem wszechobecnym w naszym życiu. Posługujemy się nimi, w szczególności, każdy inżynier, niezależnie od specjalności. Najważniejszymi ich zastosowaniami w pracy inżyniera jest wspomaganie projektowania obiektów technicznych, czynności pomiarowo-diagnostycznych oraz procesów decyzyjnych. Realizacja wszystkich tych funkcji komputera wiąże się z wykorzystaniem metod numerycznych, tzn. metod przybliżonego rozwiązywania problemów matematycznych z dziedziny algebry, analizy, probabilistyki czy geometrii, za pomocą narzędzi obliczeniowych (dzisiaj głównie komputerów), umożliwiających wykonywanie jedynie operacji logicznych i algebraicznych. Ilustrują to następujące trzy przykłady – kolejno z dziedziny elektroniki, metrologii i telekomunikacji – o wzrastającym stopniu złożoności.

Pierwszy z tych przykładów pokazuje przydatność metod numerycznej aproksymacji i metod numerycznego całkowania do wyznaczania mocy wydzielanej w dwójniku elektrycznym; drugi – możliwość zastosowania metody numerycznego różniczkowania do odtwarzania przebiegu temperatury na podstawie ciągu zmierzonych wartości napięcia na wyjściu czujnika temperatury; trzeci natomiast – metodykę projektowania filtra analogowego, opartą na metodach optymalizacji z ograniczeniami.

Przykład 1.1. Wyznaczyć średnią moc wydzielającą się w nieliniowym dwójniku, którego charakterystyka statyczna dana jest ciągiem odpowiadających sobie wartości napięcia u i prądu i :

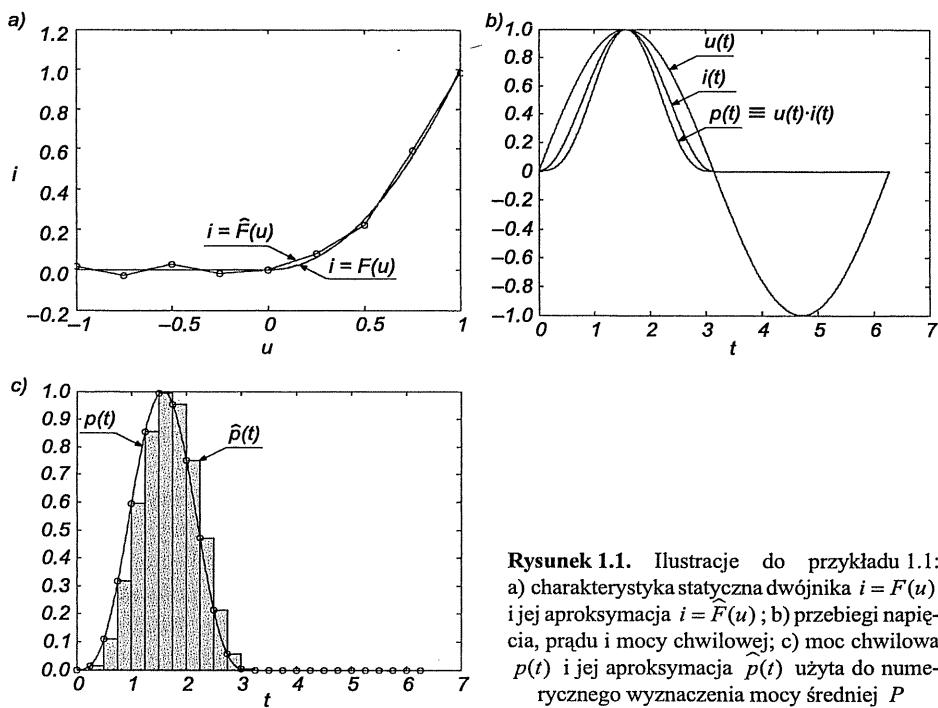
u_m	u_1	...	u_M
i_m	i_1	...	i_M

Założyć, że dwójnik pobudzany jest napięciem $u(t) = \sin(t)$, a wszelkie efekty dynamiczne są pomijalne.

Analiza tego problemu prowadzi do wyodrębnienia dwóch zadań numerycznych, które należy kolejno rozwiązać:

- aproksymacji charakterystyki statycznej $i = F(u)$ funkcją ciągłą $i = \hat{F}(u)$ na podstawie $\{u_m, i_m \mid m = 1, 2, \dots, M\}$;
- całkowania iloczynu napięcia $u(t)$ i prądu $i(t)$ w przedziale $[0, 2\pi]$, odpowiadającym okresowi napięcia i prądu, zgodnie z definicją mocy średniej:

$$P = \frac{1}{2\pi} \int_0^{2\pi} u(t) i(t) dt$$



Intuicyjnie można te dwa zadania rozwiązać, stosując:

- metodę aproksymacji charakterystyki statycznej łamana, złożoną z odcinków łączących punkty o współrzędnych (u_m, i_m) ;
- metodę prostokątów do obliczania całki definiującej moc średnią.

Metoda prostokątów wynika z zastąpienia całki w przedziale $[0, 2\pi]$ sumą całek w podprzedziałach o szerokości $2\pi/N$:

$$P \approx \hat{P} = \frac{1}{2\pi} \sum_{n=0}^{N-1} \int_{2\pi \frac{n}{N}}^{2\pi \frac{n+1}{N}} u(t) \hat{F}[u(t)] dt$$

a następnie podstawienia stałych wartości:

$$u\left(2\pi \frac{n}{N}\right) \hat{F}\left[u\left(2\pi \frac{n}{N}\right)\right] \quad \text{dla } n = 0, 1, \dots, N-1$$

zamiast odpowiednich funkcji podcałkowych:

$$P \approx \frac{1}{2\pi} \sum_{n=0}^{N-1} u\left(2\pi \frac{n}{N}\right) \hat{F}\left[u\left(2\pi \frac{n}{N}\right)\right] \frac{2\pi}{N} = \frac{1}{N} \sum_{n=0}^{N-1} \sin\left(2\pi \frac{n}{N}\right) \hat{F}\left[\sin\left(2\pi \frac{n}{N}\right)\right]$$

Szczególny przypadek opisanej procedury numerycznej przedstawiono na rys. 1.1. ♣

Przykład 1.2. Wyznaczyć przebieg temperatury $\vartheta(t)$, dysponując wynikami pomiaru napięcia $u(t)$ na wyjściu czujnika temperatury:

$$\tilde{u}_n \approx u(n \Delta t) \quad \text{dla } n = 0, 1, \dots$$

i wiedząc, że adekwatnym modelem zależności napięcia $u(t)$ od temperatury $\vartheta(t)$ jest równanie różniczkowe zwyczajne:

$$T \frac{du(t)}{dt} + u(t) = K_0 \vartheta(t)$$

gdzie t jest czasem, T – stałą czasową czujnika, zaś K_0 – współczynnikiem konwersji wielkości fizycznych.

Nasuwającym się sposobem numerycznego wyznaczania estymaty pochodnej:

$$\frac{du(t)}{dt} \quad \text{dla } t = n \Delta t, \quad n = 0, 1, \dots$$

jest użycie przybliżenia różnicowego:

$$\hat{u}_n^{(1)} \approx \frac{\tilde{u}_n - \tilde{u}_{n-1}}{\Delta t} \quad \text{dla } n = 1, 2, \dots$$

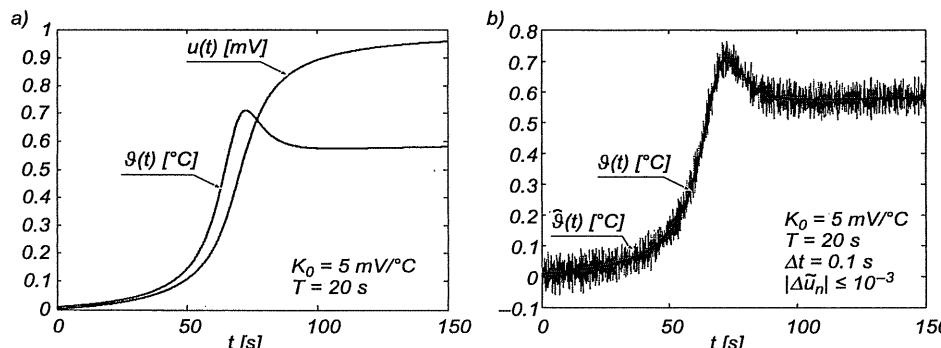
które dla wolnych od błędów danych pomiarowych $\{\tilde{u}_n\}$ zapewnia tym lepszą dokładność aproksymacji pochodnej, im mniejszy jest odstęp czasu między pomiarami Δt . W przypadku gdy dane obarczone są niepowiązanymi błędami pomiaru $\{\Delta \tilde{u}_n\}$, następuje przeniesienie tych błędów na estymatę pochodnej:

$$\Delta \tilde{u}_n^{(1)} = \frac{\Delta \tilde{u}_n - \Delta \tilde{u}_{n-1}}{\Delta t}$$

ze współczynnikiem $1 / \Delta t$. Całkowity błąd estymaty pochodnej składa się zatem z dwóch składników, z których jeden zanika do zera, drugi zaś rośnie nieograniczenie, gdy $\Delta t \rightarrow 0$. Oznacza to istnienie takiego odstępu czasu między pomiarami Δt_{opt} , dla którego błąd estymacji pochodnej osiąga minimum. Oszacowanie tej wartości w celu wyznaczenia estymatu temperatury:

$$\hat{\vartheta}(n\Delta t) = \frac{1}{K_0} \left(T \frac{\tilde{u}_n - \tilde{u}_{n-1}}{\Delta t_{\text{opt}}} + \tilde{u}_n \right)$$

z możliwie małym błędem jest problemem numerycznym, który należy rozwiązać dla konkretnych wartości T , K_0 i niepewności pomiaru napięcia na wyjściu czujnika temperatury. Szczególny przypadek opisanej procedury numerycznej przedstawiono na rys. 1.2.



Rysunek 1.2. Ilustracje do przykładu 1.2: a) odpowiadające sobie dokładne przebiegi napięcia $u(t)$ i temperatury $\theta(t)$; b) wyznaczony numerycznie przebieg temperatury $\hat{\theta}(t)$ na tle przebiegu dokładnego $\theta(t)$

Przykład 1.3. Zaprojektować analogowy filtr liniowy, eliminujący możliwie skutecznie zakłócenia w paśmie częstotliwości $f > f_2$ i przenoszący w sposób jak najmniej zniekształcony sygnał użyteczny w paśmie $f < f_1 < f_2$.

Transmitancja idealnego filtru, spełniającego warunki zadania, ma postać:

$$K_0(f) = \begin{cases} 1 & \text{dla } f < f_1 \\ \phi(f) & \text{dla } f \in [f_1, f_2] \\ 0 & \text{dla } f > f_2 \end{cases}$$

gdzie $\phi(f)$ jest dowolną ciągłą i nierosnącą funkcją taką, że $\phi(f_1) = 1$ oraz $\phi(f_2) = 0$. Transmitancja filtra rzeczywistego ma natomiast postać:

$$K(f; \mathbf{p}) = \frac{p_0 + p_1(jf) + p_2(jf)^2 + \dots + p_{LN}(jf)^{LN}}{1 + p_{-1}(jf) + p_{-2}(jf)^2 + \dots + p_{-LD}(jf)^{LD}}$$

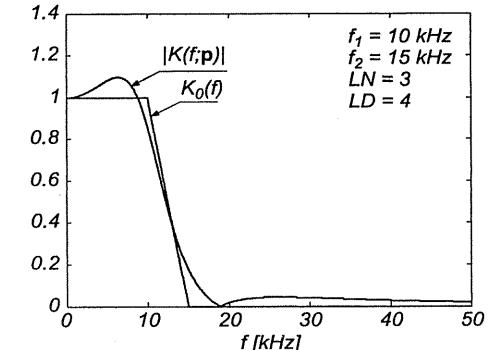
Projektowanie takiego filtra polega na wyznaczaniu wektora jego parametrów:

$$\mathbf{p} = [p_{-LD} \dots p_{-1} \ p_0 \ p_1 \dots p_{LN}]^T$$

w taki sposób, aby transmitancja $K(f; \mathbf{p})$ była w określonym sensie bliska transmitancji $K_0(f)$. Prowadzi to do zadania numerycznej optymalizacji, np. minimalizacji funkcjonału:

$$J(\mathbf{p}) = \int_0^\infty |K(f; \mathbf{p}) - K_0(f)|^2 df$$

Wynik takiej minimalizacji w szczególnym przypadku przedstawiono na rys. 1.3.



Rysunek 1.3. Ilustracja do przykładu 1.3 – moduł transmitancji filtra $K(f; \mathbf{p})$ wyznaczonej numerycznie na tle modułu charakterystyki idealnej $K_0(f)$

Ponieważ zaprojektowany filtr powinien być stabilny, więc będzie to optymalizacja z ograniczeniami: poszukiwanie rozwiązania zostanie ograniczone do zbioru wektorów parametrów \mathbf{p} , dla których wszystkie bieguna transmitancji $K\left(\frac{s}{j2\pi}; \mathbf{p}\right)$ leżą w lewej półpłaszczyźnie zmiennej zespolonej s . Jest to złożone zadanie numeryczne, które wymaga użycia metod wykraczających poza ramy niniejszego podręcznika; trudno jednak wyobrazić sobie skuteczne użycie tych metod bez znajomości metod elementarnych, o których ten podręcznik traktuje.

1.2. MODELOWANIE MATEMATYCZNE JAKO PODSTAWA OBLCZEŃ NAUKOWO-TECHNICZNYCH

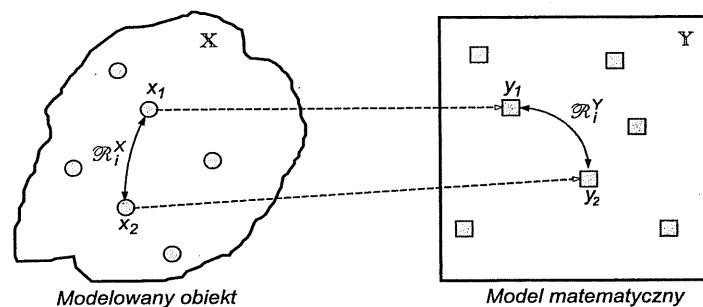
Kluczem do skutecznego rozwiązywania zadań inżynierskich przy użyciu komputera jest matematyczne modelowanie obiektów fizycznych. Takie modelowanie umożliwia opisanie projektowanego obiektu za pomocą formuł matematycznych, które następnie – w wyniku algebraizacji – mogą stać się podstawą symulacji zachowania tego obiektu za pomocą komputera. To właśnie zasady modelowania matematycznego gwarantują możliwość przeniesienia wniosków wynikających z tej symulacji na projekt obiektu fizycznego i proces jego wytwarzania.

Pojecie *modelu* coraz częściej traktowane jest w metodologii nauk jako pojęcie pierwotne: niedefiniowalne bądź definiowalne jedynie poprzez konteksty, w których bywa stosowane. Analizując jego treść, musimy więc odwoływać się zarówno do doświadczeń języka potocznego, jak do doświadczeń języka nauk indukcyjnych. Z jednej strony, mówimy: „model gospodarki”, „model krawiecki”, „model samochodu”; z drugiej zaś: „model zjawisk elektromagnetycznych” czy „teoria modeli”. Interpretacja pojęcia modelu powinna obejmować możliwie wiele kontekstów, w których użycie tego słowa uznajemy za poprawne. Jak się wydaje, jedynym niekwestionowanym w literaturze elementem takiej interpretacji jest zasada podobieństwa, na której opiera się metoda tworzenia modelu. Matematycznym uścišleniem idei podobieństwa obiektów i zjawisk fizycznych jest pojęcie *homomorfizmu*, tj. odwzorowania systemów relacyjnych zachowującego relacje. Przez system relacyjny rozumie się przy tym zbiór elementów dowolnej natury wraz z relacjami zachodzącymi między tymi elementami. Odwzorowanie systemu relacyjnego $\{X; R_1^X, R_2^X, \dots\}$ w inny system relacyjny $\{Y; R_1^Y, R_2^Y, \dots\}$ jest homomorfizmem, jeśli:

- relacje R_i^Y odpowiadają relacjom R_i^X w tym sensie, iż mają (parami) taką samą liczbę argumentów;
- z zachodzenia relacji R_i^X między pewnymi elementami zbioru X wynika zachodzenie relacji R_i^Y między ich obrazami w zbiorze Y .

Zgodnie z tzw. systemowym paradygmatem nauk przyrodniczych i technicznych poznanie dowolnego obiektu można sprowadzić do wyróżnienia jego istotnych elementów oraz relacji zachodzących między tymi elementami. Wynika stąd, przedstawiona schematycznie na rys. 1.4, interpretacja *modelu* jako homomorficznego obrazu oryginału, tj. modelowanego obiektu. W odróżnieniu od wielu innych interpretacji tego pojęcia spotykanych w literaturze, które kładą nacisk na naturę modelowanych obiektów albo na przeznaczenie modelu, ta eksponuje metodę jego tworzenia, co jest jej istotną zaletą z punktu widzenia zastosowań technicznych. Homomorficzna interpretacja modelu może być uznana za definicję modelu jednak tylko wtedy, gdy chodzi o modelowanie obiektów abstrakcyjnych

przy użyciu obiektów abstrakcyjnych, a więc na gruncie matematyki lub logiki. W technice chodzi jednak częściej o modelowanie matematyczne obiektów fizycznych, które jako takie nie mogą być utożsamione z żadnym abstrakcyjnym systemem relacyjnym.



Rysunek 1.4. Istota modelowania matematycznego – interpretacja homomorficzna

Modelowanie matematyczne obiektu lub zjawiska fizycznego składa się z dwóch jakościowo różnych etapów. Pierwszy polega na sformułowaniu słownego opisu modelowanego obiektu lub zjawiska, obejmującego specyfikację jego cech (atrybutów, właściwości) uznanych za istotne oraz relacji zachodzących między nimi; drugi zaś – na homomorficznym odwzorowaniu tych cech i relacji przy użyciu wybranego systemu relacyjnego. Wynikiem pierwszego etapu jest lingwistyczny model obiektu lub zjawiska fizycznego, którego związek z oryginałem jest przedmiotem filozoficznych dociekań teorii poznania a nie matematyki. Z tego względu homomorfizm nie może być zadowalającą logicznie podstawą definicji matematycznego modelowania obiektów i zjawisk fizycznych.

Najprostszym modelem matematycznym obiektu fizycznego jest liczba (stała). Jest to w pewnym sensie model trywialny, jako że zbiór relacji charakteryzujących dziedzinę modelu jest w tym przypadku pusty. Nie oznacza to jednak, że jest to model niepotrzebny. W istocie bez tego modelu nie sposób skonstruować poprawnie żadnego z bardziej złożonych modeli matematycznych, wykorzystywanych w praktyce badawczej czy inżynierskiej. Podstawowym uogólnieniem tego modelu są zmienne skalarne modelujące wielkości fizyczne. W procesie ustanawiania homomorfizmu między wielkością a zmienną:

- zmienną skalarną utożsamia się z symbolem, np. y , oraz zbiorem wartości, które może ona przyjmować, np. Y ;
- poszczególne wartości zmiennej y przyporządkowuje się przejawom modelowanej wielkości w taki sposób, aby relacje zachodzące między tymi wartościami ($<$, $>$, $=$) odpowiadały pewnym relacjom empirycznym ustanowionym na przejawach;

- w codziennej praktyce językowej symbol zmiennej y utożsamia się bardzo często z modelowaną wielkością.

Naturalnym uogólnieniem podanego sposobu rozumowania jest zastosowanie zmiennych wektorowych do modelowania uporządkowanych zbiorów wielkości skalarnych oraz zastosowanie funkcji do modelowania wielkości zależnych od czasu i położenia w przestrzeni. Następnym krokiem jest uwzględnienie relacji empirycznych zachodzących między wielkościami. Prowadzi to do modeli matematycznych w postaci równań algebraicznych, różniczkowych lub całkowych. Wychodząc z paradygmatu przyczynowości zjawisk zachodzących w przyrodzie, można modelem tym nadać ogólną postać:

$$\mathcal{G}(y_p) = y_s \quad \text{dla } y_p \in \mathbb{Y}_p \quad (1.1)$$

gdzie y_p jest elementem pewnego zbioru \mathbb{Y}_p modelującego przyczyny badanego zjawiska; y_s – elementem innego zbioru \mathbb{Y}_s , modelującego jego skutki; zaś:

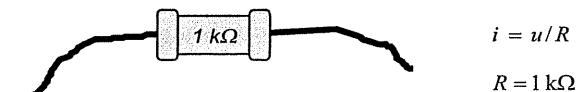
$$\mathcal{G}: \mathbb{Y}_p \rightarrow \mathbb{Y}_s \quad (1.2)$$

operatorem modelującym przyczynową zależność y_s od y_p . W tym miejscu warto zwrócić uwagę na fakt, że model określony jest nie tylko przez operator \mathcal{G} , ale także przez zbiory \mathbb{Y}_p i \mathbb{Y}_s , które w przypadkach szczególnych są zbiorami liczb, ciągów liczb, funkcji lub ciągów funkcji. Właściwy wybór tych zbiorów oraz postaci operatora \mathcal{G} jest pierwszym etapem tworzenia modelu matematycznego. Etap ten, zwany zwykle *identyfikacją strukturalną* modelu, jest o tyle trudny, że jego realizacja w większym stopniu opiera się na intuicji, doświadczeniu i innych umiejętnościach typu heurystycznego niż na systematycznej metodzie naukowej. W wyniku identyfikacji strukturalnej dokonuje się wyboru struktur matematycznych, które zostaną użyte do modelowania wielkości (zbiory \mathbb{Y}_p i \mathbb{Y}_s) oraz relacji zachodzących między nimi (operator \mathcal{G}). Rzadko jednak jest to wybór jednoznaczny; częściej zachodzi potrzeba dalszej konkretyzacji modelu przez dobór jego parametrów. Ten etap modelowania matematycznego, zwany *identyfikacją parametryczną*, ma na ogół charakter algorytmiczny. Następujący prosty przykład ilustruje logikę tworzenia modelu matematycznego jako homomorficznego obrazu oryginału fizycznego.

Przykład 1.4. Proces modelowania rezystora przedstawionego na rys. 1.5 można rozłożyć na następujące etapy:

- stwierdzenie, iż jedynymi istotnymi wielkościami fizycznymi charakteryzującymi stan rezystora są: prąd i napięcie;
- przyporządkowanie tym wielkościom ich modeli matematycznych w postaci zmiennych skalarnych o wartościach rzeczywistych: i oraz u ;
- stwierdzenie, że jedyną istotną relacją empiryczną charakteryzującą rezistor jest relacja między prądem i napięciem;

- założenie, że relacja ta z wystarczającą dokładnością może być modelowana liniowym równaniem algebraicznym postaci: $i = u/R$, z jednym parametrem R ;
- wyznaczenie wartości parametru R na podstawie pomiarów odpowiadających sobie wartości prądu i napięcia.



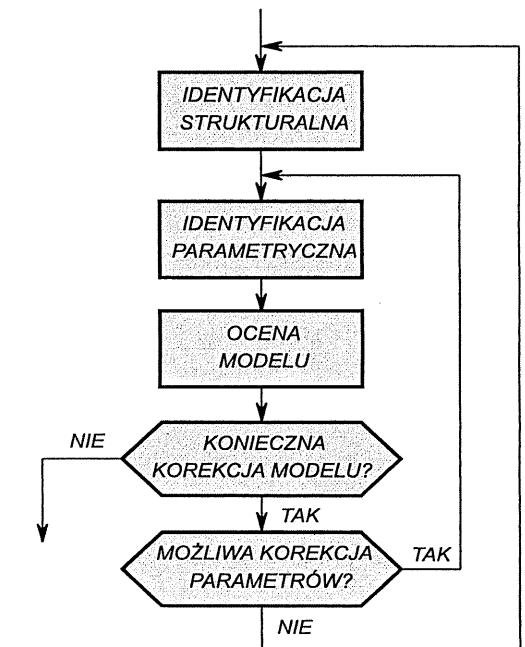
$$i = u/R$$

$$R = 1 \text{ k}\Omega$$

Rysunek 1.5. Rezystor jako obiekt modelowania matematycznego

Nie należy, oczywiście, domniemywać, iż jest to rozumowanie, które doprowadziło G. Ohma do ustanowienia w 1826 r. prawa fizyki zwanego jego imieniem. ♣

Pełny schemat procedury identyfikacji przedstawiono na rys. 1.6. Na uwagę zasługuje jej podwójnie iteracyjny charakter. Po pierwsze, korekcja parametrów modelu jest kontynuowana dopóki nie zostanie uzyskany model odpowiadający potrzebom albo dalsza korekcja nie będzie możliwa. Po drugie, zachodzi konieczność modyfikacji (na ogół rozszerzania) struktury modelu, ilekroć jego dostosowanie do potrzeb za pomocą korekcji parametrów staje się niemożliwe. Wewnątrz obydwiu pętli iteracyjnych musi się znajdować operacja oceny modelu ze względu na jego adekwatność do celów, którym ma on służyć. Podstawą tej oceny jest pomiar; jakość pomiaru decyduje więc o jakości identyfikowanego modelu.



Rysunek 1.6. Schemat tworzenia modelu matematycznego

Przykład 1.5. Złożoność matematycznego modelu rezystora zależy od przeznaczenia tego modelu, a w szczególności od zakresu częstotliwości, w którym ma on adekwatnie odwzorowywać zachowanie się rezystora:

- W zakresie częstotliwości akustycznych wystarczającym jego modelem jest liniowe równanie algebraiczne wynikające z prawa Ohma.
 - W zakresie częstotliwości radiowych model rezystora powinien uwzględnić efekt gromadzenia w nim ładunku oraz indukcyjność doprowadzeń, co prowadzi do modelu w postaci układu 2 lub 3 liniowych równań różniczkowych zwykłych (o stałych współczynnikach, gdy można zaniedbać efekt starzenia).
 - W zakresie częstotliwości mikrofalowych adekwatnym modelem matematycznym rezystora staje się układ równań różniczkowych cząstkowych.
- ♣

Przykład 1.6. Modelowanie czujnika przeznaczonego do statycznych pomiarów temperatury polega na aproksymacji jego charakterystyki statycznej $u = F(\vartheta)$ w pewnym zakresie temperatury $\vartheta \in [\vartheta_{\min}, \vartheta_{\max}]$, na podstawie zbioru par odpowiadających sobie wartości temperatury ϑ_n i napięcia wyjściowego czujnika u_n , zmierzonych za pomocą odpowiednio dokładnych przyrządów pomiarowych ($n = 1, 2, \dots$). Można, w tym celu, posłużyć się wielomianem trzeciego stopnia i w wyniku identyfikacji parametrycznej uzyskać wartości czterech jego współczynników. Jeżeli jednak nie da to rozwiązania zapewniającego zadowalającą dokładność pomiaru temperatury za pomocą czujnika, to trzeba będzie wielomian trzeciego stopnia zastąpić wielomianem stopnia wyższego, np. piątego; a jeśli i to nie da zadowalających rezultatów, to trzeba będzie podzielić przedział zmienności temperatury na dwa podprzedziały i powtórzyć iteracyjną procedurę dopasowywania wielomianu do danych dla każdego z tych podprzedziałów.

♣

Realizując w praktyce ogólny schemat modelowania matematycznego, należy pamiętać o trzech znamiennych jego cechach:

- wybór modelu opisowego – a w konsekwencji także struktury matematycznej użytej do modelowania – jest w znacznym stopniu arbitralny;
- struktura matematyczna użyta do modelowania powinna być wyczerpująco opisana za pomocą skończonej liczby parametrów;
- kryteria oceny modelu są ściśle związane z jego przeznaczeniem – model uznany za adekwatny w jednym zastosowaniu może się okazać nieadekwatny w innym.

1.3. MODELOWANIE MATEMATYCZNE W PRACY INŻYNIERA

1.3.1. MODELOWANIE MATEMATYCZNE W PROJEKTOWANIU

Celem projektowania obiektu technicznego jest stworzenie jego modelu matematycznego, umożliwiającego jego bezpośrednią realizację. Wynik projektowania

może mieć tradycyjną formę rysunku albo – coraz częściej – kodu cyfrowego umożliwiającego sterowanie urządzeniami wytwórczymi. Już samo sformułowanie zadania projektowania ma postać lingwistyczno-matematycznego modelu projektowanego obiektu, obejmującego specyfikację wymagań funkcjonalnych, stawianych temu obiekowi, oraz ograniczeń realizacyjnych. Proces projektowania prowadzi do odpowiedniego przetworzenia owego modelu, polegającego nie tylko na odpowiedniej jego formalizacji, ale także na wzbogaceniu o pierwiastek twórczy pochodzący od projektanta oraz o informację specjalistyczną pochodzączą z źródeł zewnętrznych: z katalogów technicznych, z baz danych, z fachowej literatury itp.

Rozwój informatyki zrewolucjonizował metodykę projektowania inżynierskiego. Stworzył bowiem – z jednej strony – narzędzia przetwarzania ogromnych strumieni informacji, z drugiej zaś – umożliwił pełne wykorzystanie modeli matematycznych zjawisk leżących u podstaw funkcjonowania projektowanych obiektów technicznych. Dzięki infrastrukturze informatycznej modele te z obszaru teorii zostały przeniesione do praktyki projektowania. Służą do symulacji projektowanych obiektów technicznych; tym samym – do przewidywania ich zachowania i właściwości, a w konsekwencji do ich iteracyjnej syntezy. W wielu praktycznie ważnych przypadkach umożliwiają nawet syntezę bezpośrednią (przykładem może być projektowanie pewnych klas filtrów cyfrowych).

1.3.2. MODELOWANIE MATEMATYCZNE W POMIARACH

Na gruncie tak zwanej *reprezentacyjnej teorii pomiaru* rozróżnia się pojęcie pomiaru w szerszym sensie oraz pojęcie pomiaru w węższym sensie. To pierwsze obejmuje idealizację obiektu pomiaru, czyli stworzenie jego modelu lingwistycznego w drodze abstrakcji, oraz ustanowienie homomorfizmu między tym modelem a liczbowym systemem relacyjnym. Pomiar w węższym sensie – to każdy akt wykorzystania owego homomorfizmu do przyporządkowania konkretnej wartości konkretnemu przejawowi wielkości mierzonej. Nietrudno w tym rozróżnieniu dostrzec się analogii z dokonanym na rys. 1.3 rozróżnieniem między strukturalną a parametryczną identyfikacją modelu matematycznego. W istocie pomiar w szerszym sensie jest szczególnym przypadkiem identyfikacji strukturalnej, a pomiar w węższym sensie – szczególnym przypadkiem identyfikacji parametrycznej. Względy wygody językowej przemawiają za traktowaniem pomiaru i modelowania jako pary pojęć przechodnich – w taki sam sposób, jak używa się pary pojęć *system-element*. Wygodnie jest bowiem mówić o:

- identyfikacji rezystancji i pojemności na podstawie wyników pomiaru napięć i prądów;
- identyfikacji transmitancji obwodu na podstawie wyników pomiaru rezystancji i pojemności;

- identyfikacji modelu układu na podstawie wyników pomiaru transmitancji jego obwodów itd.

Sformułowania te dobrze oddają logiczną i praktyczną zależność modelowania matematycznego od pomiaru i pomiaru od modelowania matematycznego. O ile ta pierwsza wydaje się oczywista, o tyle druga może budzić pewne wątpliwości i dlatego zostanie zilustrowana dwoma przykładami.

Przykład 1.7. Pomiar impedancji wejściowej wzmacniacza ma sens, o ile adekwatnym modelem zależności między napięciem a prądem wejściowym tego wzmacniacza jest liniowe równanie całkowe typu splotu. Osiagalna dokładność aproksymacji tej zależności przy użyciu takiego właśnie równania wyznacza granice dokładności pomiaru impedancji wejściowej wzmacniacza, których nie sposób przekroczyć, nawet za pomocą najdokładniejszych przyrządów.

Przykład 1.8. Pomiar średnicy wałka ma sens, o ile z dokładnością zadowalającą dla praktyki jego obwód może być aproksymowany okręgiem. W przeciwnym przypadku jako podstawę pomiaru trzeba przyjąć model wałka w postaci funkcji opisującej krzywą zamkniętą na płaszczyźnie jego przekroju. Jeżeli ma miejsce niedopuszczalna niejednorodność wałka wzdłuż jego osi, to trzeba sięgnąć po model w postaci funkcji opisującej powierzchnię wałka w przestrzeni trójwymiarowej.

Opisaną zależność pomiaru od modelowania matematycznego można skonkretyzować, zauważając, że każdy pomiar opiera się na inwersji pewnego modelu matematycznego, wiążącego wielkości objęte procedurą pomiarową. Stwierdzenie to odnosi się zarówno do klasycznych pomiarów analogowych, jak do współczesnych skomputeryzowanych systemów pomiarowych. Następujący przykład pokazuje inwersję modelu w przypadku najprostszych pomiarów napięcia.

Przykład 1.9. Voltomierz magnetoelektryczny połączony z ogniwem galwanicznym, którego siła elektromotoryczna e ma być zmierzona, może być reprezentowany modelem obejmującym dwa przetworniki elementarne: napięcie-kąt i kąt-odczyt. Kąt wychylenia wskazówki voltmierza względem położenia spoczynkowego α pozostaje w zależności funkcyjnej od e :

$$\alpha = \mathcal{J}(e) \approx \text{const}(e) \cdot e$$

bliskiej zależności liniowej. Układ *wskazówka + podziałka + eksperymentator* realizuje inwersję tej zależności; w idealnym bowiem przypadku:

$$\hat{e} = \mathcal{J}^{-1}(\alpha) \approx \frac{\alpha}{\text{const}(e)}$$

Rola eksperymentatora w tym układzie jest prawidłowa interpretacja wzajemnego położenia wskazówki i podziałki, prowadząca do analogowo-cyfrowego przetwarzania ostatecznego wyniku pomiaru.

1.3.3. ZASTOSOWANIE KOMPUTERA DO MODELOWANIA MATEMATYCZNEGO

Metodyka zastosowania komputera do modelowania matematycznego obejmuje cztery powtarzalne elementy:

- konkretyzację modelu matematycznego, opisującego klasę obiektów fizycznych, tj. dostosowanie tego modelu do specyfiki konkretnego modelowanego obiektu;
- upraszczanie modelu do postaci zapewniającej minimalną wystarczającą dokładność modelowania;
- dyskretyzację modelu umożliwiającą jego reprezentację w komputerze za pomocą skończonej liczby procedur, zmiennych i parametrów;
- formalny opis modelu w języku programowania komputera.

Przykład 1.10. Adekwatnym modelem matematycznym szerokiej klasy obiektów sterowania jest układ równań różniczkowych zwyczajnych pierwszego rzędu. Zastosowanie tego modelu do symulacji ramienia robota przemysłowego wymaga określenia liczby równań, wartości ich parametrów oraz warunków początkowych. Istotne ułatwienie zadania symulacji może wynikać z uproszczenia tego modelu, polegającego na linearyzacji równań. Sama symulacja opiera się na numerycznym rozwiązaniu równań różniczkowych przez zastąpienie występujących w nich pochodnych ilorazami różnicowymi i rozwiązanie wynikającego stąd układu liniowych równań algebraicznych. Algorytm symulacji może być zapisany w jednym z języków programowania wyższego poziomu (np. C++) albo – jeśli tego wymagają względy techniczne – w języku asemblera.

Istotnym elementem metodyki użycia komputerów do rozwiązywania zadań inżynierskich jest umiejętność sprowadzania tych zadań do standardowych problemów numerycznych, takich jak:

- aproksymacja i interpolacja funkcji;
- rozwiązywanie układów liniowych równań algebraicznych;
- rozwiązywanie układów nieliniowych równań algebraicznych;
- różniczkowanie funkcji jednej i wielu zmiennych;
- całkowanie układów równań różniczkowych zwyczajnych.

1.4. ALGORYTM NUMERYCZNY I FORMY JEGO ZAPISU

Zadaniem numerycznym nazywa się proces przetwarzania pewnego elementu zbioru danych \mathbb{D} w taki element zbioru wyników \mathbb{W} , który spełnia zadane wymagania $\mathcal{R}_1, \mathcal{R}_2, \dots$. Układ:

$$\{\mathbb{D}, \mathbb{W}, \mathcal{R}_1, \mathcal{R}_2, \dots\} \quad (1.3)$$

nazywany jest klasą zadań numerycznych. W zadaniach numerycznych, będących przedmiotem niniejszego podręcznika, zbiory danych \mathbb{D} i wyników \mathbb{W} są zbiorami wektorów.

Przykład 1.11. Zadanie wyznaczania wartości wielomianu:

$$y = P_N(x; \mathbf{a}) = \sum_{n=0}^N a_n x^n$$

zdefiniowane jest wektorem danych $\mathbf{d} = [\hat{x} \ a_0 \ a_1 \ \dots \ a_N]^T$, jednoelementowym wektorem wyników $\mathbf{w} = [\hat{y}]$ oraz wymaganiem $\mathcal{R}_1: \hat{y} = P_N(\hat{x}; \mathbf{a})$. Zbiór $(N+1)$ -wymiarowych wektorów liczb rzeczywistych \mathbb{D} , zbiór liczb rzeczywistych \mathbb{W} oraz wymaganie \mathcal{R}_1 definiują klasę zadań numerycznych. Jej podklaśc tworzą np. zadania wyznaczania wartości wielomianów $P_N(x; \mathbf{a})$ o współczynnikach całkowitych.

Algorytm numeryczny – to opis jednoznacznie uporządkowanego ciągu operacji, które przekształcają zbiór danych \mathbb{D} w zbiór wyników \mathbb{W} dla pewnej klasy zadań numerycznych. W praktyce rozwiązywania zadań numerycznych stosowane są następujące równoważne formy opisu algorytmów:

- tradycyjny zapis matematyczny;
- zapis w języku programowania komputerów;
- sieć działań;
- zapis sekwencyjny.

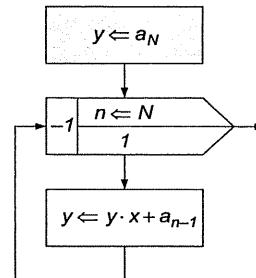
Przykład 1.12. Algorytm Hornera wyznaczania wartości wielomianu $P(x; \mathbf{a})$ ma postać:

$$\begin{aligned} y_N &= a_N \\ y_{n-1} &= y_n x + a_{n-1} \quad \text{dla } n = N, N-1, \dots, 1 \\ y &= y_0 \end{aligned}$$

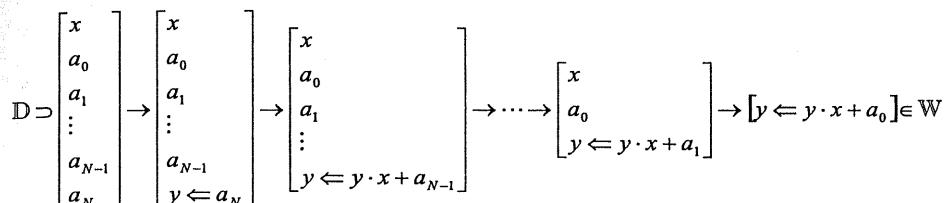
Zapis tego algorytmu w języku C++ ma postać:

```
y = a[N]; for (n = N; n > 0; n--) {y = y * x + a[n-1];}
```

gdzie $a[n] \equiv a_n$. Sieć działań algorytmu Hornera jest przedstawiona na rys. 1.7, natomiast równoważny jej zapis sekwencyjny – na rys. 1.8.



Rysunek 1.7. Sieć działań algorytmu Hornera



Rysunek 1.8. Sekwencyjny zapis algorytmu Hornera

1.5. DOKŁADNOŚĆ OBLCZEŃ INŻYNIERSKICH

1.5.1. IDENTYFIKACJA PROBLEMATYKI

Zacznijmy od znanej anegdotki o tokarzu i lufie. Tokarzowi zlecono wytoczenie lufy o średnicy wewnętrznej 10 mm i średnicy zewnętrznej 11 mm. Kiedy ten z profesjonalnego obowiązku zapytał o pożądaną dokładność, uzyskał niefrasobliwą odpowiedź: 5% wystarczy. Tak się nieszczerliwie złożyło, że błąd toczenia powierzchni wewnętrznej wyniósł dokładnie +5% i błąd toczenia powierzchni zewnętrznej – dokładnie -5%. I lufy nie było... Anegdotka jest ilustracją ogólniejszego spostrzeżenia, iż względny błąd wyniku obliczeń może być istotnie większy niż błąd danych użytych do tych obliczeń.

Przykład 1.13. Rozważmy odejmowanie liczb bliskich sobie z dokładnością do 3 cyfr po przecinku:

+369.711 (± 0.0005)	\Rightarrow błąd względny ok. $1.3 \cdot 10^{-10}$
-369.702 (± 0.0005)	\Rightarrow błąd względny ok. $1.3 \cdot 10^{-10}$
0.009 (± 0.0010)	\Rightarrow błąd względny ok. 11%

Mamy tutaj do czynienia ze wzrostem błędu o 5 rzędów wielkości!

Przykład 1.14. Wyznaczanie zer wielomianu:

$$y = a_{20}x^{20} + a_{19}x^{19} + \dots + a_1x + a_0 \equiv \prod_{n=1}^{20} (x - n)$$

jest zadaniem szczególnie wrażliwym na błędy współczynników a_n . Na przykład, zaburzenie a_{19} na poziomie $6 \cdot 10^{-6}\%$ powoduje pojawienie się pierwiastków zespłonnych, między innymi $13.99 \pm j2.51$, podczas gdy dla współczynników dokładnych wszystkie zera są liczbami naturalnymi, od 1 do 20.

1.5.2. REPREZENTACJA LICZB I ZAOKRĄGLANIE WYNIKÓW OBLCZEŃ W KOMPUTERZE

Komputer jest urządzeniem cyfrowym, w którym każda liczba musi być reprezentowana za pomocą skończonego ciągu cyfr (binarnych, dziesiętnych, szesnastkowych, ...). Spośród wielu sposobów przedstawiania liczb w komputerze, które pojawiły się w historycznym jego rozwoju, przetrwały dwa: *reprezentacja stałopozycyjna* (zwana dawniej stałoprzecinkową) oraz *reprezentacja zmiennopozycyjna* (dawniej zwana zmienoprzecinkową). Ta pierwsza zdefiniowana jest jednoznacznie liczbą cyfr przed przecinkiem LP oraz liczbą cyfr po przecinku LM ; reprezentowane liczby mają więc postać:

$$x = \pm d_{LP-1}d_{LP-2} \cdots d_1d_0.d_{-1}d_{-2} \cdots d_{-LM}, \quad d_i \in \{0, \dots, P-1\} \quad (1.4)$$

gdzie P jest podstawą liczenia; najczęściej $P = 2, 10$ lub 16 . We współczesnych komputerach podstawowym sposobem przedstawiania liczb jest reprezentacja zmiennopozycyjna, umożliwiająca oszczędniejszą niż stałopozycyjna gospodarkę pamięcią komputera. Zostanie ona przedstawiona nieco dokładniej, jako że w dalszej części podręcznika to ona stanowi podstawę analizy dokładności algorytmów numerycznych. Liczba dokładna:

$$x = \pm m P^c \quad (1.5)$$

gdzie c jest liczbą całkowitą zwaną cechą, a m jest liczbą rzeczywistą postaci:

$$m = 0.m_1m_2 \dots m_{L-1}m_Lm_{L+1} \dots, \quad m_i \in \{0, \dots, P-1\}, \quad m_1 \neq 0 \quad (1.6)$$

zwaną znormalizowaną mantysą, jest reprezentowana w komputerze jako liczba przybliżona:

$$\tilde{x} = \pm \tilde{m} P^c \quad (1.7)$$

gdzie \tilde{m} jest mantysą zaokrągloną do L cyfr znaczących:

$$\tilde{m} = \begin{cases} 0.m_1m_2 \dots m_L, & \text{gdy } 0 \leq m_{L+1} < P/2 \\ 0.m_1m_2 \dots m_L + P^{-L}, & \text{gdy } P/2 \leq m_{L+1} < P \end{cases} \quad (1.8)$$

Graniczny błąd względny reprezentacji zmiennopozycyjnej wynika z następującego oszacowania:

$$|\delta[\tilde{x}]| = \left| \frac{\tilde{m} P^c - m P^c}{m P^c} \right| \leq \frac{\sup |\tilde{m} - m|}{\inf |m|} = \frac{0.5P \cdot P^{-(L+1)}}{P^{-1}} = 0.5P \cdot P^{-L} \equiv \epsilon ps \quad (1.9)$$

Przykład 1.15. Zalecana przez normę IEEE754 liczba cyfr znaczących mantisy w binarnej ($P = 2$) reprezentacji zmiennopozycyjnej wynosi:

$$L = \begin{cases} 24 & \text{dla liczb pojedynczej precyzji} \\ 53 & \text{dla liczb podwójnej precyzji} \end{cases}$$

Zatem, zgodnie ze wzorem (1.9):

$$\epsilon ps = \begin{cases} 2^{-24} \approx 5.90 \cdot 10^{-8} & \text{dla liczb pojedynczej precyzji} \\ 2^{-53} \approx 1.11 \cdot 10^{-16} & \text{dla liczb podwójnej precyzji} \end{cases}$$

Standardowym założeniem przyjmowanym przez projektantów oprogramowania przeznaczonego do realizacji algorytmów numerycznych jest zaokrąglanie wyników operacji podstawowych do takiego samego formatu zmiennopozycyjnego jak format reprezentacji liczb oraz gwarantowanie błędu względnego tych wyników nieprzekraczającego wartości ϵps . Założenie to nie wyklucza, oczywiście, możliwości użycia innych formatów wewnętrz procedur realizujących zmiennopozycyjne operacje podstawowe.

Przykład 1.16. Operacjami podstawowymi w języku C++ są nie tylko cztery działania arytmetyczne, ale także funkcje standardowe, np. $\sin(x)$, $\ln(x)$ czy $\exp(x)$.

Błędy reprezentacji danych oraz błędy zaokrągleń wyników operacji zmiennopozycyjnych pojawiają się w każdym procesie obliczeniowym, na wszystkich etapach realizacji algorytmu numerycznego; podlegają transformacjom wynikającym ze struktury tego algorytmu i składają się na błąd wyniku obliczeń.

1.5.3. OGÓLNY MODEL PRZENOSZENIA BŁĘDÓW W ALGORYTMIE NUMERYCZNYM

Wygodnym punktem wyjścia do budowy modelu przenoszenia błędów w algorytmie numerycznym jest sekwencyjny jego opis w postaci:

$$\mathbf{d} \equiv \mathbf{v}^{(0)} \xrightarrow{\Phi^{(1)}} \mathbf{v}^{(1)} \xrightarrow{\Phi^{(2)}} \dots \xrightarrow{\Phi^{(K-1)}} \mathbf{v}^{(K)} \xrightarrow{\Phi^{(K)}} \mathbf{w} \quad (1.10)$$

gdzie $\mathbf{d} \in \mathbb{D}$ jest wektorem danych, $\mathbf{w} \in \mathbb{W}$ jest wektorem wyników, zaś $\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(K-1)}$ są wektorami wyników pośrednich, natomiast $\Phi^{(1)}, \dots, \Phi^{(K)}$ są operatorami elementarnymi definiującymi algorytm rozwiązywania zadania numerycznego: $\Phi: \mathbb{D} \rightarrow \mathbb{W}$. Zachodzi więc dokładna lub przybliżona równość:

$$\Phi = \Phi^{(K)} \circ \Phi^{(K-1)} \circ \dots \circ \Phi^{(2)} \circ \Phi^{(1)}$$

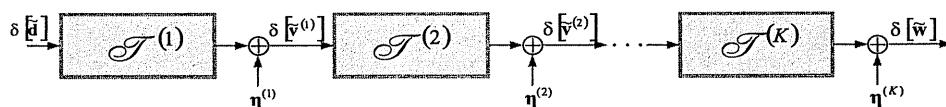
gdzie symbolem \circ oznaczono operację składania operatorów.

Wynikający z sekwencyjnego zapisu algorytmu model przenoszenia błędu przedstawiono na rys. 1.9, przy czym zastosowano tam następujące oznaczenia:

- $\eta^{(k)}$ – wektor względnych błędów zaokrągleń wyniku operacji $\Phi^{(k)}$;
- $\delta[\tilde{\mathbf{v}}^{(k)}]$ – wektor błędów względnych $\tilde{\mathbf{v}}^{(k)}$, $k = 0, 1, \dots, K$;
- $\mathcal{S}^{(k)}$ – zależna od $\mathbf{v}^{(k)}$ funkcja przenoszenia błędu przez operator $\Phi^{(k)}$.

Pojedynczy blok tego modelu opisany jest następującym równaniem algebraicznym (na ogół nieliniowym):

$$\delta[\tilde{\mathbf{v}}^{(k)}] = \mathcal{S}^{(k)}\{\delta[\tilde{\mathbf{v}}^{(k-1)}]\} + \eta^{(k)} \quad (1.11)$$



Rysunek 1.9. Ogólny model przenoszenia błędu w algorytmie numerycznym

1.6. ZŁOŻONOŚĆ OBLCZEŃ INŻYNIERSKICH

W niniejszym podręczniku ograniczamy się do przedstawienia zaledwie kilku podstawowych pojęć dotyczących złożoności obliczeniowej oraz zarysu intuicyjnych metod oceny tej złożoności, na ogół wystarczających w praktyce inżynierskiej. Czytelnika zainteresowanego pełniejszym zapoznaniem się z tą problematyką odsyłamy do literatury specjalistycznej, np. [B, T].

Złożoność obliczeniowa algorytmu numerycznego zależy zwykle od rozmiaru rozwiązywanego zadania numerycznego, np. od stopnia wielomianu w przypadku

zadania wyznaczania zer wielomianu czy od liczby równań w przypadku rozwiązywania układu liniowych równań algebraicznych. Charakterystyki złożoności są więc funkcjami rzeczywistymi zmiennej całkowitej, typu $f: \mathbb{N} \rightarrow \mathbb{R}_+$, gdzie \mathbb{N} jest zbiorem liczb naturalnych, a \mathbb{R}_+ – zbiorem liczb rzeczywistych dodatnich; do ich klasyfikacji służy pojęcie *rzędu funkcji*. Funkcja $f: \mathbb{N} \rightarrow \mathbb{R}_+$ jest rzędu $g: \mathbb{N} \rightarrow \mathbb{R}_+$, co zapisujemy $f \in O(g)$, jeżeli:

$$\exists C \in \mathbb{R}_+, N \in \mathbb{N} \quad \forall n > N: |f(n)| \leq C|g(n)| \quad \text{albo} \quad \lim_{n \rightarrow \infty} \left| \frac{f(n)}{g(n)} \right| < \infty \quad (1.12)$$

Przykład 1.17. Funkcja $n \ln(n)$ jest rzędu n^2 , co zapisuje się $n \ln(n) \in O(n^2)$, ponieważ:

$$\lim_{n \rightarrow \infty} \left| \frac{f(n)}{g(n)} \right| = \lim_{n \rightarrow \infty} \left| \frac{\ln(n)}{n} \right| = \lim_{n \rightarrow \infty} \frac{n^{-1}}{1} = 0$$

przy czym druga równość wynika z zastosowania reguły de l'Hôpitala. ♣

Funkcją złożoności obliczeniowej algorytmu \mathcal{A} rozwiązywania klasy zadań numerycznych $\{\mathbb{D}_n, \mathbb{W}, \mathcal{R}_1, \mathcal{R}_2, \dots\}$ nazywa się następującą funkcję wymiaru zadania n :

$$f_{\mathcal{A}}(n) = \sup \{ \text{LKE}(\mathbf{x}) \mid \mathbf{x} \in \mathbb{D}_n \} \quad (1.13)$$

gdzie LKE(\mathbf{x}) jest liczbą kroków elementarnych, niezbędnych do rozwiązania zadania za pomocą algorytmu \mathcal{A} dla danych \mathbf{x} .

Ze względu na złożoność obliczeniową algorytmy numeryczne dzieli się na *algorytmy wielomianowe* (efektywne), dla których:

$$f_{\mathcal{A}} \in O\left(\sum_{k=0}^K \alpha_k n^k\right) \quad (1.14)$$

oraz *algorytmy wykładnicze* (nieefektywne), dla których:

$$f_{\mathcal{A}} \notin O\left(\sum_{k=0}^K \alpha_k n^k\right) \quad (1.15)$$

Głęboki sens nazywania algorytmów wielomianowych efektywnymi oraz wykładniczymi nieefektywnymi wyjaśniają tablice 1.1 i 1.2.

W praktyce inżynierskiej stosowane są dwie intuicyjne miary złożoności obliczeniowej:

- czas wykonania algorytmu na komputerze odniesienia $T_{\mathcal{A}}(n)$;
- ilość pamięci tego komputera, niezbędnej do realizacji algorytmu $M_{\mathcal{A}}(n)$.



PODSTAWOWE METODY ANALIZY DOKŁADNOŚCI ALGORYTMÓW NUMERYCZNYCH

Tablica 1.1

Porównanie złożoności obliczeniowej algorytmów wielomianowych i wykładniczych (wg [B])

$f_{\infty}(n)$	CZAS WYKONANIA ALGORYTMU	
	$n = 10$	$n = 60$
ALGORYTMY WIELOMIANOWE		
$O(n)$	0.0001 sekundy	0.0006 sekundy
$O(n^3)$	0.001 sekundy	0.216 sekundy
$O(n^5)$	0.1 sekundy	13.0 minut
ALGORYTMY WYKŁADNICZE		
$O(2^n)$	0.001 sekundy	3366 stuleci
$O(3^n)$	0.059 sekundy	$1.3 \cdot 10^{13}$ stuleci

Tablica 1.2

Wpływ wzrostu szybkości obliczeń na czas wykonywania algorytmów wielomianowych i wykładniczych (wg [B])

$f_{\infty}(n)$	ROZMIAR NAJWIĘKSZEGO PROBLEMU ROZWIĄZYWANEGO W CIĄGU GODZINY PRZEZ		
	KOMPUTER ODNIESIENIA	KOMPUTER 100 RAZY SZYBSZY	KOMPUTER 1000 RAZY SZYBSZY
ALGORYTMY WIELOMIANOWE			
$O(n)$	W_1	$100 \cdot W_1$	$1000 \cdot W_1$
$O(n^3)$	W_3	$4.64 \cdot W_3$	$10 \cdot W_3$
$O(n^5)$	W_5	$2.5 \cdot W_5$	$3.98 \cdot W_5$
ALGORYTMY WYKŁADNICZE			
$O(2^n)$	P_2	$P_2 + 6.64$	$P_2 + 9.97$
$O(3^n)$	P_3	$P_3 + 4.19$	$P_3 + 6.29$

Tę pierwszą charakterystykę oblicza się na podstawie wyników pomiaru czasu $T_i(n) \in [0, \infty)$ wykonania algorytmu dla danych $\mathbf{x}_i \in \mathbb{D}_n$ ($i = 1, 2, \dots$). Najpierw wyznacza się ciąg:

$$T_{\infty}(n) = \sup \{ T_i(n) \mid i = 1, 2, \dots \} \quad (1.16)$$

a następnie funkcję $\widehat{T}_{\infty}(n)$ aproksymującą ten ciąg, np. w sensie najmniejszych kwadratów. Często bardziej czytelny wynik uzyskuje się, aproksymując ciąg $\{\log[T_{\infty}(n)]\}$.

W analogiczny sposób wyznacza się charakterystykę zajętości pamięci. Najpierw, na podstawie wyników oszacowania rozmiaru pamięci $M_i(n) \in [0, \infty)$, niezbędnego do wykonania algorytmu dla danych $\mathbf{x}_i \in \mathbb{D}_n$ ($i = 1, 2, \dots$), wyznacza się ciąg:

$$M_{\infty}(n) = \sup \{ M_i(n) \mid i = 1, 2, \dots \} \quad (1.17)$$

a następnie funkcję $\widehat{M}_{\infty}(n)$ aproksymującą ten ciąg. Często bardziej czytelny wynik uzyskuje się, aproksymując ciąg $\{\log[M_{\infty}(n)]\}$.

2.1. LINIOWY MODEL PRZENOSZENIA BŁĘDÓW

Praktyczna przydatność wyników obliczeń dla inżyniera zależy od tego, czy możliwa jest ocena ich dokładności i czy dokładność ta odpowiada wymaganiom wynikającym z celu obliczeń, np. z wymaganej precyzyji działania projektowanego urządzenia. Z tego względu praktyczne znaczenie ma wyłącznie analiza dokładności obliczeń, gdy błędy obliczeń są *odpowiednio małe*. Przy założeniu, że względne błędy danych i wszystkich wyników pośrednich są istotnie mniejsze od jedności, można dokonać linearizacji przedstawionego w podrozdziale 1.5.3 modelu przenoszenia błędów:

$$\delta[\tilde{\mathbf{v}}^{(k)}] = \mathbf{T}^{(k)} \cdot \delta[\tilde{\mathbf{v}}^{(k-1)}] + \boldsymbol{\eta}^{(k)} \quad (2.1)$$

gdzie $\mathbf{T}^{(k)}$ jest macierzą współczynników przenoszenia błędów względnych. Sposoby wyznaczania elementów tej macierzy są głównym tematem niniejszego rozdziału.

W przypadku skalarnej funkcji zmiennej skalarnej φ , $y = \varphi(x)$, definiującej zadanie numeryczne, zachodzi zależność:

$$y + \Delta y \equiv \tilde{y} = \varphi(\tilde{x}) \equiv \varphi(x + \Delta x) \quad (2.2)$$

w której Δy i Δx to odpowiadające sobie bezwzględne błędy wyniku y i danej x . Przybliżona zależność między tymi błędami wynika z rozwinięcia funkcji φ w szereg Taylora:

$$\tilde{y} = \varphi(x) + \varphi'(x)\Delta x + \frac{1}{2}\varphi''(x)(\Delta x)^2 + \dots = y + x\varphi'(x)\frac{\Delta x}{x} + \frac{1}{2}x^2\varphi''(x)\left(\frac{\Delta x}{x}\right)^2 + \dots \quad (2.3)$$

$$\Delta y = \tilde{y} - y = x\varphi'(x)\varepsilon + \frac{1}{2}x^2\varphi''(x)\varepsilon^2 + \dots \quad (2.4)$$

gdzie $\varepsilon = \Delta x/x$ jest błędem względnym danej x . Ma ona postać:

$$\delta[y] \equiv \frac{\Delta y}{y} = \frac{x}{y}\varphi'(x)\varepsilon + \frac{1}{2}\frac{x^2}{y}\varphi''(x)\varepsilon^2 + \dots \quad (2.5)$$

a przy założeniu, że $\left| \frac{x \phi''(x)}{2 \phi'(x)} \right| \cdot |\varepsilon| \ll 1$, może być uproszczona do zależności liniowej:

$$\delta[\tilde{y}] \approx T\varepsilon \quad (2.6)$$

gdzie T jest współczynnikiem przenoszenia błędu względnego przez funkcję ϕ , wyrażającym się wzorem, który warto zapamiętać:

$$T = \frac{x}{y} \frac{dy}{dx} = \frac{y}{\frac{dx}{d \ln(x)}} = \frac{d \ln(y)}{d \ln(x)} \quad (2.7)$$

Przeprowadzając analogiczne rozumowanie dla przypadku skalarnej funkcji N zmiennych ϕ , $y = \phi(x_1, x_2, \dots, x_N)$, otrzymuje się następującą zależność między błędami względnymi:

$$\delta[\tilde{y}] \approx \sum_{n=1}^N T_n \varepsilon_n \quad (2.8)$$

gdzie:

$$\varepsilon_n = \delta[\tilde{x}_n], \quad T_n = \frac{x_n}{y} \frac{\partial y}{\partial x_n} = \frac{\partial \ln(y)}{\partial \ln(\tilde{x}_n)}$$

Wynikające stąd liniowe formuły przenoszenia błędów względnych przez najbardziej typowe funkcje mają postać:

$$\delta[\tilde{x}_1 \pm \tilde{x}_2] \approx \frac{x_1 \varepsilon_1 \pm x_2 \varepsilon_2}{x_1 \pm x_2} \quad (2.9a)$$

$$\delta[\tilde{x}_1^a \tilde{x}_2^b] \approx a \varepsilon_1 + b \varepsilon_2 \quad (2.9b)$$

$$\delta[\tilde{x}_1^{\tilde{x}_2}] \approx x_2 (\varepsilon_1 + \varepsilon_2 \ln(x_1)) \quad (2.9c)$$

$$\delta[\ln(\tilde{x})] \approx \frac{1}{\ln(x)} \varepsilon \quad (2.9d)$$

Przykład 2.1. Odejmowanie dwóch liczb o zbliżonych wartościach jest operacją „ryzykowną” w tym sensie, że jej wynik może być obarczony znacznie większym błędem względnym niż dane. Wynika to wprost z formuły (2.9a):

$$\delta[\tilde{x}_1 - \tilde{x}_2] = \frac{x_1 \varepsilon_1 - x_2 \varepsilon_2}{x_1 - x_2} \xrightarrow[x_1 \rightarrow x_2; \varepsilon_1 \neq \varepsilon_2]{} \infty$$

Zadanie 2.1. Wyznaczyć zależność $\delta[\sin(\tilde{x})]$ oraz $\delta[\arctg(\tilde{x})]$ od ε . ♠

Przy wyznaczaniu współczynników przenoszenia błędów przez bardziej złożone funkcje – stanowiące superpozycje funkcji, dla których znane są te współczynniki – użyteczne są przybliżone zależności zwane „regułami arytmetyki epsilonów”:

$$(1 + \varepsilon_1)(1 + \varepsilon_2) \approx 1 + \varepsilon_1 + \varepsilon_2 \quad (2.10a)$$

$$(1 + \varepsilon)^a \approx 1 + a\varepsilon \quad \text{dla } |a| \ll \text{eps}^{-1} \quad (2.10b)$$

$$\ln(1 + \varepsilon) \approx \varepsilon \quad (2.10c)$$

$$e^{1+\varepsilon} \approx (1 + \varepsilon)e \quad (2.10d)$$

Reguły te wynikają z rozwinięcia lewej strony w szereg Taylora i pominięcia składników zawierających względne błędy danych w potęgach wyższych niż pierwsza.

Przykład 2.2. Różnica kwadratów dwóch liczb x_1 i x_2 :

$$y = x_1^2 - x_2^2 \quad \text{dla } \left| \frac{x_1}{x_2} \right| \leq \frac{1}{2}$$

może być obliczona na dwa sposoby wynikające z tożsamości: $x_1^2 - x_2^2 = (x_1 - x_2)(x_1 + x_2)$.

Pierwszy sposób zapisać można w postaci algorytmu:

$$\mathcal{A}_1: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 = x_1^2 \\ v_2 = x_2^2 \end{bmatrix} \rightarrow [y = v_1 - v_2]$$

Wynik uzyskany za pomocą tego algorytmu związany jest z błędami danych i błędami zaokrągleń operacji zmienopozycyjnych w następujący sposób:

$$\tilde{y}_1 = [x_1^2(1 + \varepsilon_1)^2(1 + \eta_1) - x_2^2(1 + \varepsilon_2)^2(1 + \eta_2)](1 + \eta_3)$$

gdzie η_1 jest błędem zaokrąglenia wyniku podnoszenia x_1 do kwadratu, η_2 – błędem zaokrąglenia wyniku podnoszenia x_2 do kwadratu, η_3 – błędem zaokrąglenia wyniku odejmowania $x_1^2 - x_2^2$. Stosując wzór (2.10b) do wyrażeń $(1 + \varepsilon_1)^2$ i $(1 + \varepsilon_2)^2$, otrzymuje się:

$$\tilde{y}_1 \approx [x_1^2(1 + 2\varepsilon_1)(1 + \eta_1) - x_2^2(1 + 2\varepsilon_2)(1 + \eta_2)](1 + \eta_3)$$

Stosując następnie wzór (2.10a), upraszcza się iloczyny wyrażeń zawierających błędy danych i błędy zaokrągleń wyników potęgowania:

$$\begin{aligned}\tilde{y}_1 &= [x_1^2(1+2\epsilon_1+\eta_1) - x_2^2(1+2\epsilon_2+\eta_2)](1+\eta_3) = \\ &= \{(x_1^2 - x_2^2) + [x_1^2(2\epsilon_1+\eta_1) - x_2^2(2\epsilon_2+\eta_2)]\}(1+\eta_3)\end{aligned}$$

Wyłączając zaś przed nawias dokładną wartość wyrażenia $y = x_1^2 - x_2^2$:

$$\tilde{y}_1 = y \left\{ 1 + \left[\frac{x_1^2}{y} (2\epsilon_1 + \eta_1) - \frac{x_2^2}{y} (2\epsilon_2 + \eta_2) \right] \right\} (1 + \eta_3)$$

otrzymuje się liniową kombinację błędów aproksymującą względny błąd wyniku:

$$\delta[\tilde{y}_1] \approx T_1 \epsilon_1 + T_2 \epsilon_2 + K_1 \eta_1 + K_2 \eta_2 + K_3 \eta_3$$

gdzie:

$$T_1 = 2 \frac{x_1^2}{y} \in \left[-\frac{2}{3}, 0 \right], \quad T_2 = -2 \frac{x_2^2}{y} \in \left[2, \frac{8}{3} \right]$$

$$K_1 = \frac{x_1^2}{y} \in \left[-\frac{1}{3}, 0 \right], \quad K_2 = -\frac{x_2^2}{y} \in \left[1, \frac{4}{3} \right], \quad K_3 = 1$$

Błąd ten podlega więc następującemu oszacowaniu:

$$|\delta[\tilde{y}_1]| \leq |T_1| \text{eps} + |T_2| \text{eps} + |K_1| \text{eps} + |K_2| \text{eps} + |K_3| \text{eps} \equiv \delta_{\text{gr1}}$$

W analogiczny sposób przeprowadza się analizę alternatywnego algorytmu wyznaczania różnicicy kwadratów:

$$\mathcal{A}_2 : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 = x_1 + x_2 \\ v_2 = x_1 - x_2 \end{bmatrix} \rightarrow [y = v_1 v_2]$$

Wyrażenie wiążące wynik uzyskany za pomocą tego algorytmu z błędami danych i błędami zaokrągleń operacji zmiennopozycyjnych ma postać:

$$\tilde{y}_2 = \{[x_1(1+\epsilon_1) + x_2(1+\epsilon_2)](1+\eta_s)[x_1(1+\epsilon_1) - x_2(1+\epsilon_2)](1+\eta_o)\}(1+\eta_m)$$

gdzie η_s jest błędem zaokrąglenia wyniku sumowania $x_1 + x_2$, η_o – błędem zaokrąglenia wyniku odejmowania $x_1 - x_2$, zaś η_m – błędem zaokrąglenia wyniku mnożenia sumy przez różnicę. Błąd wyniku wyraża się wzorem:

$$\delta[\tilde{y}_2] \approx T_1 \epsilon_1 + T_2 \epsilon_2 + K_s \eta_s + K_o \eta_o + K_m \eta_m$$

gdzie T_1 i T_2 zdefiniowane są tak samo jak w poprzednim algorytmie, zaś $K_s = K_o = K_m = 1$. Oszacowanie tego błędu ma postać:

$$|\delta[\tilde{y}_2]| \leq |T_1| \text{eps} + |T_2| \text{eps} + |K_s| \text{eps} + |K_o| \text{eps} + |K_m| \text{eps} \equiv \delta_{\text{gr2}}$$

Porównując błędy graniczne δ_{gr1} i δ_{gr2} , można zauważać, że różnią się one jedynie składową pochodzącą od błędów zaokrągleń, odpowiednio:

$$\left(\frac{x_1^2 + x_2^2}{|y|} + 1 \right) \text{eps} \quad \text{oraz} \quad 3 \text{eps}$$

Algorytm \mathcal{A}_1 można uznać za lepszy, gdy:

$$0 < \left| \frac{x_1}{x_2} \right| < \frac{1}{\sqrt{3}}$$

ponieważ wtedy $\delta_{\text{gr1}} < \delta_{\text{gr2}}$, zaś algorytm \mathcal{A}_2 , gdy:

$$\frac{1}{\sqrt{3}} < \left| \frac{x_1}{x_2} \right| < \frac{1}{2}$$

ponieważ wtedy $\delta_{\text{gr2}} < \delta_{\text{gr1}}$. Wynika to z następującego wnioskowania:

$$\left(\frac{x_1^2 + x_2^2}{|y|} + 1 \right) \text{eps} \geq 3 \text{eps} \Rightarrow \frac{x_1^2 + x_2^2}{|x_1^2 - x_2^2|} \geq 2 \Rightarrow \frac{1}{\sqrt{3}} \leq \left| \frac{x_1}{x_2} \right| \leq \sqrt{3}$$



2.2. PRZENOSZENIE BŁĘDÓW DANYCH

2.2.1. WSPÓŁCZYNNIKI PRZENOSZENIA BŁĘDÓW DANYCH

Nieprzypadkowo składowa błędu wyniku obliczeń, pochodząca od danych, jest identyczna dla obydwu algorytmów rozpatrywanych w przykładzie 2.2. Prawdziwe jest bowiem ogólne stwierdzenie, iż błąd wyniku spowodowany przeniesieniem błędów danych:

$$\delta^o[\tilde{y}] = \mathbf{T}^{(K)} \cdot \mathbf{T}^{(K-1)} \cdot \dots \cdot \mathbf{T}^{(1)} \cdot \boldsymbol{\epsilon} \quad (2.11)$$

nie zależy od algorytmu użytego do obliczeń.

Przykład 2.3. Współczynnik przenoszenia błędu danej x na wynik sumowania zbieżnego szeregu geometrycznego:

$$y = \sum_{n=0}^{\infty} x^n$$

może być wyznaczony na podstawie wzoru na sumę takiego szeregu:

$$y = \lim_{N \rightarrow \infty} \sum_{n=0}^N x^n = \lim_{N \rightarrow \infty} \frac{1 - x^{N+1}}{1 - x} = \frac{1}{1 - x} \quad \text{dla } |x| < 1$$

Stosując do tego wzoru reguły rachunku epsilonów, zdefiniowane wzorem (2.10), otrzymuje się:

$$\tilde{y} = \frac{1}{1-x(1+\varepsilon)} = \frac{1}{(1-x)-x\varepsilon} = \frac{1}{(1-x)\left[1-\frac{x}{1-x}\varepsilon\right]} \approx y \left[1 + \frac{x}{1-x}\varepsilon\right]$$

dla odpowiednio małych $|\varepsilon|$. Błąd wyniku ma więc postać:

$$\delta^o[\tilde{y}] = \frac{x}{1-x}\varepsilon$$

Wynika stąd, że:

$$T(x) \equiv \frac{\delta[\tilde{y}]}{\delta[\tilde{x}]} = \frac{x}{1-x}$$

Zadanie 2.2. Wyznaczyć współczynnik przenoszenia błędu (względnego) danej x dla szeregu:

$$y = \sum_{n=1}^{\infty} nx^n$$

przy założeniu, że $|x| < 1$.

$$\text{Odp.: } T(x) = \frac{1+x}{1-x}.$$

Zadanie 2.3. Naszkicować wykres współczynnika przenoszenia błędu (względnego) danej x przez funkcję

$$y = \ln[\sin(x)] \quad \text{dla } x \in \left(0, \frac{\pi}{2}\right)$$

$$\text{Odp.: } T(x) = \frac{x \operatorname{ctg}(x)}{\ln[\sin(x)]}.$$

Przykład 2.4. Wyznaczone numerycznie zera \tilde{y}_1 , \tilde{y}_2 i \tilde{y}_3 wielomianu

$$w(y) = y^3 - 30y^2 + 299y - 990$$

obarczone są nieznanymi błędami względnymi odpowiednio, η_1 , η_2 i η_3 , i dla tego $w(\tilde{y}_i) = \rho_i \neq 0$ dla $i = 1, 2, 3$. Oszacować te błędy.

Rozwiążanie tego problemu opiera się na przekształcaniu zależności

$$w(\tilde{y}_i) = \tilde{y}_i^3 - 30\tilde{y}_i^2 + 299\tilde{y}_i - 990 = \rho_i$$

gdzie $\tilde{y}_i = y_i(1+\eta_i)$, przy użyciu reguł rachunku epsilonów:

$$y_i^3(1+\eta_i)^3 - 30y_i^2(1+\eta_i)^2 + 299y_i(1+\eta_i) - 990 = \rho_i$$

$$y_i^3(1+3\eta_i) - 30y_i^2(1+2\eta_i) + 299y_i(1+\eta_i) - 990 \approx \rho_i$$

$$(y_i^3 - 30y_i^2 + 299y_i - 990) + (3y_i^3 - 60y_i^2 + 299y_i)\eta_i \approx \rho_i$$

Ponieważ $y_i^3 - 30y_i^2 + 299y_i - 990 = 0$, więc

$$\eta_i \approx \frac{\rho_i}{3y_i^3 - 60y_i^2 + 299y_i} \approx \frac{\rho_i}{3\tilde{y}_i^3 - 60\tilde{y}_i^2 + 299\tilde{y}_i}$$

Przykład 2.5. Wyznaczyć względny błąd obliczenia $y = \cos^{10}(1.25\pi)$ spowodowany zaokrągleniem liczby π do dwóch cyfr po przecinku.

Rozwiążając ten problem należy uwzględniać to, iż błąd bezwzględny reprezentacji danej jest w tym przypadku znany dokładnie i wynosi $3.14 - \pi$. Korzystając ze wzoru (2.7), otrzymuje się:

$$\delta[\tilde{y}] = \frac{\pi}{y} \frac{dy}{d\pi} \delta\tilde{\pi} = \frac{1}{y} \frac{dy}{d\pi} \Delta[\tilde{\pi}] = -12.5 \operatorname{tg}(1.25\pi) \cdot (3.14 - \pi)$$

$$\delta[\tilde{y}] \approx 1.99 \cdot 10^{-2}$$

Zadanie 2.4. Wyznaczyć minimalną liczbę cyfr znaczących dziesiętnej reprezentacji liczby π , wystarczającą do zapewnienia względnej dokładności obliczania wartości wyrażenia

$$y = \frac{1}{2 + \pi x} \quad \text{dla } x \in \left[0, \frac{1}{\pi}\right]$$

nie gorszej niż 10^{-6} .

Odp.: 5.

Zadanie 2.5. Określić minimalną liczbę cyfr znaczących mantisy L , umożliwiającą wyznaczenie wartości wyrażenia

$$y = \sqrt{\frac{\sin(\pi x)}{x}} \quad \text{dla } x \in \left(0, \frac{1}{2}\right)$$

z dokładnością względną nie gorszą niż 10^{-3} . Wskazówka: do oszacowania $|\delta[\tilde{y}]|$ wykorzystać zależność: $v \operatorname{ctg}(v) \in (0, 1)$ dla $v \in (0, \pi/2)$.

Odp.: 5.

2.2.2. UWARUNKOWANIE NUMERYCZNE ZADANIA

Jeżeli małe względne zmiany danych zadania powodują duże względne zmiany jego rozwiązań, to zadanie takie nazywamy *źle uwarunkowanym numerycznie*. Uwarunkowanie zadania charakteryzuje macierz:

$$\mathbf{T} = \mathbf{T}^{(K)} \cdot \mathbf{T}^{(K-1)} \cdots \cdot \mathbf{T}^{(1)} \quad (2.12)$$

Do zadań źle uwarunkowanych numerycznie należy zarówno odejmowanie liczb bliskich sobie, jak wyznaczanie zer wielomianów wyższych stopni czy rozwiązywanie układów liniowych równań algebraicznych $\mathbf{A} \cdot \mathbf{y} = \mathbf{b}$ o macierzy prawie osobliwej, tzn. takiej, że $\det(\mathbf{A}) \approx 0$.

Przykład 2.6. Zbadać uwarunkowanie zadania rozwiązywania równania kwadratowego

$$y^2 - 2x_1 y + x_2 = 0 \quad \text{dla } x_1 > 0, x_2 < 0$$

Ponieważ pierwiastki równania kwadratowego wyrażają się wzorami:

$$y_1 = x_1 + \sqrt{x_1^2 - x_2}, \quad y_2 = x_1 - \sqrt{x_1^2 - x_2}$$

wartości współczynników przenoszenia błędów danych podlegają następującym ograniczeniom:

$$T_{1,1} \equiv \frac{x_1}{y_1} \cdot \frac{\partial y_1}{\partial x_1} = \frac{x_1}{y_1} \cdot \frac{y_1}{y_1 - x_1} = \frac{x_1}{\sqrt{x_1^2 - x_2}} \in (0, 1)$$

$$T_{1,2} \equiv \frac{x_2}{y_1} \cdot \frac{\partial y_1}{\partial x_2} = \frac{x_2}{y_1} \cdot \frac{1}{2(x_1 - y_1)} = \frac{1}{2} \left(1 - \frac{x_1}{\sqrt{x_1^2 - x_2}} \right) \in \left(0, \frac{1}{2} \right)$$

$$T_{2,1} \equiv \frac{x_1}{y_2} \cdot \frac{\partial y_2}{\partial x_1} = \frac{x_1}{y_2} \cdot \frac{y_2}{y_2 - x_1} = \frac{-x_1}{\sqrt{x_1^2 - x_2}} \in (-1, 0)$$

$$T_{2,2} \equiv \frac{x_2}{y_2} \cdot \frac{\partial y_2}{\partial x_2} = \frac{x_2}{y_2} \cdot \frac{1}{2(x_1 - y_2)} = \frac{1}{2} \left(1 + \frac{x_1}{\sqrt{x_1^2 - x_2}} \right) \in \left(\frac{1}{2}, 1 \right)$$

Oznacza to, że $|T_{i,j}| \leq 1$ dla $i, j = 1, 2$. Zadanie rozwiązywania równania kwadratowego jest więc bardzo dobrze uwarunkowane numerycznie.

2.3. PRZENOSZENIE BŁĘDÓW ZAOKRĄGLEŃ

2.3.1. WSPÓŁCZYNNIKI PRZENOSZENIA BŁĘDÓW ZAOKRĄGLEŃ

Błąd wniesiony przez błędy zaokrągleń wyników operacji zmiennopozycyjnych zależy od algorytmu i wyraża się wzorem:

$$\delta^G[\tilde{\mathbf{y}}] = \sum_{k=1}^K \mathbf{K}^{(k)} \quad (2.13)$$

gdzie $\mathbf{K}^{(k)} = \mathbf{T}^{(K)} \cdot \mathbf{T}^{(K-1)} \cdots \cdot \mathbf{T}^{(k+1)}$ dla $k = 1, \dots, K-1$, $\mathbf{K}^{(K)} = \mathbf{I}$ (por. rys. 1.9).

Przykład 2.7. Dwa algorytmy obliczania wartości wyrażenia

$$y = \frac{1}{x} - \frac{1}{1+x} \quad \text{dla } x > 0$$

wynikające z tożsamości:

$$\frac{1}{x} - \frac{1}{1+x} = \frac{1}{x(1+x)}$$

różnią się dokładnością. Analiza pierwszego z nich:

$$\mathcal{O}_1: [x] \rightarrow \begin{bmatrix} v_1 = 1/x \\ v_2 = 1/(1+x) \end{bmatrix} \rightarrow [y = v_1 - v_2]$$

składa się z następujących kroków, polegających na wielokrotnym wykorzystaniu wzorów (2.9) i (2.10):

$$\tilde{y}_1 = \left[\frac{1}{x} (1 + \eta_1) - \frac{1}{(1+x)(1+\eta_s)} (1 + \eta_2) \right] (1 + \eta_o)$$

$$\tilde{y}_1 = \left[\frac{1}{x} (1 + \eta_1) - \frac{1}{1+x} (1 + \eta_2 - \eta_s) \right] (1 + \eta_o)$$

$$\tilde{y}_1 = \left\{ \left(\frac{1}{x} - \frac{1}{1+x} \right) + \left[\frac{1}{x} \eta_1 - \frac{1}{1+x} (\eta_2 - \eta_s) \right] \right\} (1 + \eta_o)$$

$$\tilde{y}_1 = y \left\{ 1 + \frac{1}{xy} \eta_1 - \frac{1}{(1+x)y} (\eta_2 - \eta_s) \right\} (1 + \eta_o)$$

gdzie η_1 jest błędem zaokrąglenia wyniku dzielenia $1/x$, η_s – błędem zaokrąglenia wyniku sumowania $1+x$, η_2 – błędem zaokrąglenia wyniku dzielenia $1/(1+x)$, zaś η_o – błędem zaokrąglenia wyniku odejmowania $v_1 - v_2$. Składowa

błędu wyniku otrzymanego za pomocą algorytmu \mathcal{A}_1 , wniesiona przez błędy zaokrągleń operacji zmiennopozycyjnych, wyraża się więc wzorem:

$$\delta^G[\tilde{y}_1] = (1+x)\eta_1 - x\eta_2 + x\eta_s + \eta_o$$

i podlega oszacowaniu:

$$|\delta^G[\tilde{y}_1]| \leq |1-x|eps + |x|eps + |x|eps + eps = (3x+2)eps$$

Alternatywny algorytm ma postać:

$$\mathcal{A}_2: [x] \rightarrow \begin{bmatrix} v_1 = x \\ v_2 = 1+x \end{bmatrix} \rightarrow \left[y = \frac{1}{v_1 v_2} \right]$$

Jego analiza daje się streszczyć w jednym wierszu:

$$\tilde{y}_2 = \frac{1+\eta_d}{x(1+x)(1+\eta_s)(1+\eta_m)} \approx y(1+\eta_d - \eta_s - \eta_m) \Rightarrow |\delta^G[\tilde{y}_2]| \leq 3eps$$

gdzie η_m jest błędem zaokrąglenia wyniku mnożenia $x(1+x) \equiv v$, zaś η_d – błędem zaokrąglenia wyniku dzielenia $y = 1/v$. Porównanie oszacowań błędów wnoszonych przez obydwa algorytmy pokazuje, że dla $x < 1/3$ algorytm \mathcal{A}_1 jest nieco lepszy niż algorytm \mathcal{A}_2 , ponieważ:

$$\sup |\delta^G[\tilde{y}_1]| < \sup |\delta^G[\tilde{y}_2]|.$$

Zadanie 2.6. Oszacować spowodowany zaokrągleniami błąd obliczania:

$$y = (x+x^{-1})^{\frac{1}{2}} - (x-x^{-1})^{\frac{1}{2}} \text{ dla } x \gg 1$$

za pomocą następujących algorytmów:

$$\mathcal{A}_1: [x] \rightarrow \begin{bmatrix} x \\ x^{-1} \end{bmatrix} \rightarrow \begin{bmatrix} v_1 = (x+x^{-1})^{\frac{1}{2}} \\ v_2 = (x-x^{-1})^{\frac{1}{2}} \end{bmatrix} \rightarrow [y = v_1 - v_2]$$

$$\mathcal{A}_2: [x] \rightarrow \begin{bmatrix} x \\ x^{-1} \end{bmatrix} \rightarrow \begin{bmatrix} v_1 = (x+x^{-1})^{\frac{1}{2}} \\ v_2 = (x-x^{-1})^{\frac{1}{2}} \end{bmatrix} \rightarrow \left[y = \frac{2}{x(v_1 + v_2)} \right]$$

$$\text{Odp.: } |\delta^G[\tilde{y}_1]| \leq (3x^2 + 1)eps \text{ oraz } |\delta^G[\tilde{y}_2]| \leq \frac{9}{2}eps.$$

2.3.2. NUMERYCZNA POPRAWNOŚĆ ALGORYTMU

Za numerycznie poprawne uważa się algorytmy, które spełniają pewne realistyczne wymagania co do wartości wnoszonych przez nie błędów. Nie chodzi na ogół o to, aby utrzymać te błędy na poziomie eps czy jakiejś niewielkiej krotności eps , ale o to, aby te błędy pozostawały w praktycznie akceptowalnym stosunku do błędów wnoszonych przez dane. Dla użytkownika wyników obliczeń na ogół obojętne jest bowiem źródło ich błędów, liczy się tylko ich sumaryczna wartość. Za realistyczne należy zatem uznać wymaganie, aby błędy wnoszone przez algorytm były małe dla tych danych, dla których małe są błędy wnoszone przez dane. Wymaganie to leży u podstaw następującego określenia numerycznej poprawności: algorytm jest *numerycznie poprawny*, jeżeli dla dokładnych danych dostarcza rozwiązań będących dokładnymi rozwiązaniami zadania dla „nieco” zaburzonych danych [J1].

Przykład 2.8. Algorytm obliczania wartości wyrażenia $y = x_1^2 + x_2^2$ jest numerycznie poprawny, ponieważ:

$$\tilde{y} = \left[x_1^2(1+\eta_1) + x_2^2(1+\eta_2) \right] (1+\eta_s)$$

a zgodnie z regułą (2.10a):

$$\tilde{y} \approx \left[x_1^2(1+\eta_1 + \eta_s) + x_2^2(1+\eta_1 + \eta_s) \right]$$

zaś zgodnie z regułą (2.10b):

$$\tilde{y} \approx \left[x_1 \left(1 + \frac{1}{2}\eta_1 + \frac{1}{2}\eta_s \right) \right]^2 + \left[x_2 \left(1 + \frac{1}{2}\eta_2 + \frac{1}{2}\eta_s \right) \right]^2$$

Z tego ostatniego wyrażenia wynika, że skutek błędów zaokrągleń wyników podnoszenia do kwadratu i sumowania jest taki sam jak skutek zaburzenia danych na poziomie eps .

Metoda badania numerycznej poprawności algorytmu, zastosowana w przykładzie 2.8, nie zawsze prowadzi do rozstrzygnięcia. Jeżeli nie udaje się wyznaczyć zastępczych zaburzeń danych, których skutek byłby równoważny wpływowi na wynik błędów zaokrągleń, to nie można stąd wyciągać wniosku, że algorytm jest numerycznie niepoprawny. Można jedynie domniemywać, że tak jest i podjąć próbę potwierdzenia tej hipotezy na drodze poszukiwania takiego przykładu danych w zbiorze D , dla którego błąd spowodowany zaokrągleniami rośnie nieograniczenie, chociaż błąd pochodzący od danych jest ograniczony.

Przykład 2.9. Zbadać numeryczną poprawność następującego algorytmu rozwiązywania równania kwadratowego z przykładu 2.6:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ v = \sqrt{x_1^2 - x_2} \end{bmatrix} \rightarrow \begin{bmatrix} y_1 = x_1 + v \\ y_2 = x_1 - v \end{bmatrix}$$

Źródłem numerycznej niepoprawności może być w tym przypadku odejmowanie liczb bliskich sobie, które ma miejsce, gdy $x_2 \rightarrow 0$; wówczas bowiem błąd wyznaczania pierwiastka y_2 może rosnąć nieograniczenie, jako że nieograniczenie rośnie współczynnik przenoszenia błędu wyniku pierwiastkowania:

$$K_{2,\sqrt{}} = \frac{\sqrt{x_1^2 - x_2}}{y_2} \cdot \frac{\partial y_2}{\partial (\sqrt{x_1^2 - x_2})} = -\frac{\sqrt{x_1^2 - x_2}}{x_1 - \sqrt{x_1^2 - x_2}} \xrightarrow{x_2 \rightarrow 0} \infty$$

Klasyczny algorytm rozwiązywania równania kwadratowego nie jest więc numerycznie poprawny. Alternatywą jest algorytm wyznaczania y_2 wynikający z zastosowania wzoru Viéte'a na iloczyn pierwiastków:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_2 \\ y_1 = x_1 + \sqrt{x_1^2 - x_2} \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 = x_2 / y_1 \end{bmatrix}$$

Nietrudno się przekonać, że eliminacja odejmowania doprowadziła do powstania algorytmu numerycznie poprawnego.



Zadanie 2.7. Skonstruować numerycznie poprawny algorytm obliczenia wartości wyrażenia:

$$y = \frac{1}{x} \left(1 - \frac{1}{\sqrt{1+x}} \right) \quad \text{dla } |x| \ll 1$$

Oszacować błąd obliczania y według tego algorytmu.



2.4. ANALIZA DOKŁADNOŚCI ALGORYTMÓW ZA POMOCĄ KOMPUTERA

Podstawą analizy zadań i algorytmów numerycznych jest wyznaczanie współczynników przenoszenia błędów w funkcji danych i szacowanie ich wartości największych. Upowszechniły się cztery główne metody wyznaczania współczynników przenoszenia błędów za pomocą komputera:

- różniczkowanie numeryczne;
- symulacja statystyczna;
- różniczkowanie symboliczne;
- arytmetyka interwałowa.

Pierwsza z nich opiera się na wykorzystaniu wzoru (2.8) i może być zrealizowana nawet w bardzo ubogim środowisku oprogramowania. Ze względu na łatwość implementacji zostanie tutaj przedstawiona w zarysie. Współczynnik przenoszenia błędów zaokrągleń $K_{i,j}^{(k)}$ wyznacza się dla wybranych wartości wektora danych \hat{x} , reprezentujących klasę rozwiązywanych zadań numerycznych, według następującego schematu:

- ❶ obliczyć \hat{y} za pomocą badanego algorytmu;
- ❷ obliczyć \tilde{y} za pomocą badanego algorytmu, zaburzając $v_j^{(k)}$ na poziomie np. $\eta = 1\%$:

$$\tilde{v}_j^{(k)} = v_j^{(k)}(1 + \eta) \quad \text{dla } j, k = 1, 2, \dots \quad (2.14)$$

- ❸ wyznaczyć estymaty współczynników przenoszenia błędów zaokrągleń według wzoru:

$$\widehat{K}_{i,j}^{(k)} = \frac{\tilde{y}_i - \hat{y}_i}{\tilde{y}_i \eta} \quad \text{dla } i, j, k = 1, 2, \dots \quad (2.15)$$

Analogicznie można wyznaczyć estymaty $\widehat{T}_{i,j}$ współczynników przenoszenia błędów danych $T_{i,j}$. Błąd całkowity j -tej składowej wektora wyników, wyrażający się kombinacją liniową:

$$\delta[\tilde{y}_i] \approx \sum_j \widehat{T}_{i,j} \varepsilon_j + \sum_k \sum_j \widehat{K}_{i,j}^{(k)} \eta_j^{(k)} \quad (2.16)$$

szacuje się metodą najgorszego przypadku

$$|\delta[\tilde{y}_i]| \leq \left(\sum_j |\widehat{T}_{i,j}| + \sum_k \sum_j |\widehat{K}_{i,j}^{(k)}| \right) \text{eps} \quad (2.17)$$

Przykład 2.10. Dane są dwa algorytmy obliczania wartości wyrażenia:

$$y = \sqrt{1+x} - \sqrt{1-x} \quad \text{dla } |x| \ll 1$$

Pierwszy wynika wprost z zapisu tego wyrażenia:

$$\mathcal{A}_1: [x] \rightarrow \begin{bmatrix} v_1 = \sqrt{1+x} \\ v_2 = \sqrt{1-x} \end{bmatrix} \rightarrow [y = v_1 - v_2]$$

drugi – z pomnożenia i podzielenia tego wyrażenia przez sumę odpowiednich pierwiastków:

$$\mathcal{A}_2 : [x] \rightarrow \begin{bmatrix} v_1 = \sqrt{1+x} \\ v_2 = \sqrt{1-x} \end{bmatrix} \rightarrow \begin{bmatrix} y = 2 \frac{x}{v_1 + v_2} \end{bmatrix}$$

Wyznaczyć współczynnik przenoszenia błędu zaokrąglenia wyniku pośredniego v_2 w obydwu algorytmach.

W tablicy 2.1 zestawiono wartości współczynników K_1 i K_2 , wyznaczone według wzoru (2.7):

$$\mathcal{A}_1 : \tilde{y}_1 = v_1 - v_2(1+\eta) = (v_1 - v_2) \left[1 - \frac{v_2\eta}{v_1 - v_2} \right] \Rightarrow K_1 = -\frac{v_2}{y}$$

$$\begin{aligned} \mathcal{A}_2 : \tilde{y}_2 &= 2 \frac{\dot{x}}{v_1 + v_2(1+\eta)} = 2 \frac{\dot{x}}{(v_1 + v_2) \left[1 + \frac{v_2\eta}{v_1 + v_2} \right]} = \\ &= y \left(1 - \frac{v_2\eta}{v_1 + v_2} \right) \Rightarrow K_2 = \frac{-v_2}{v_1 + v_2} \end{aligned}$$

oraz ich estymaty \hat{K}_1 i \hat{K}_2 uzyskane za pomocą programu realizującego następującą sekwencję działań:

$$v_1 := \sqrt{1+x}, \quad v_2 := \sqrt{1-x}$$

$$\dot{y} := 2 \frac{\dot{x}}{v_1 + v_2}$$

$$\tilde{v}_2 := v_2 \cdot 1.01$$

$$\tilde{y}_1 := v_1 - \tilde{v}_2, \quad \hat{K}_1 := 100 \frac{\tilde{y}_1 - \dot{y}}{\dot{y}}$$

$$\tilde{y}_2 := 2 \frac{\dot{x}}{v_1 + v_2}, \quad \hat{K}_2 := 100 \frac{\tilde{y}_2 - \dot{y}}{\dot{y}}$$

Tablica 2.1

Wyniki obliczeń przeprowadzonych w przykładzie 2.10

\dot{x}	10^{-1}	10^{-2}	10^{-3}	10^{-4}
K_1	$-0.994975 \cdot 10^2$	$-0.999956 \cdot 10^4$	$-0.999759 \cdot 10^6$	$-0.107374 \cdot 10^9$
\hat{K}_1	$-0.994974 \cdot 10^2$	$-0.999956 \cdot 10^4$	$-0.999759 \cdot 10^6$	$-0.107374 \cdot 10^9$
K_2	-0.497499	-0.499975	-0.499999	-0.500000
\hat{K}_2	-0.497503	-0.497488	-0.497512	-0.497512

2.5. POZANUMERYCZNE ZASTOSOWANIA METOD ANALIZY DOKŁADNOŚCI

Metody i techniki analizy dokładności algorytmów, opisane w niniejszym rozdziale, pozwalają rozwiązać wiele zadań związanych z realizacją tych algorytmów za pomocą procesorów zmiennopozycyjnych, takich jak:

- szacowanie granicznych błędów, jakie mogą wystąpić podczas obliczeń prowadzonych za pomocą procesora o danej liczbie cyfr znaczących mantysy L ;
- wyznaczanie minimalnej liczby cyfr znaczących mantysy L , zapewniającej utrzymanie błędów obliczeń poniżej zadanej wartości maksymalnej;
- wyznaczanie dopuszczalnych błędów danych pochodzących z pomiarów w sposób zapewniający utrzymanie błędów obliczeń poniżej zadanej wartości maksymalnej;
- wybór algorytmów zapewniających najdokładniejsze wyniki;
- korekcja algorytmów wadliwych numerycznie.

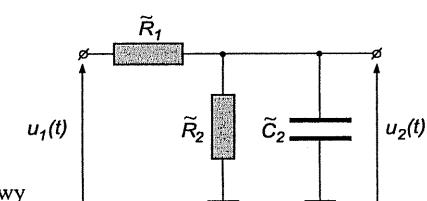
Dodatkową motywacją do nabycia sprawności w posługiwaniu się rachunkiem epsilonów może być fakt, że stanowi on także wygodne narzędzie rozwiązywania wielu innych zadań inżynierskich, takich jak:

- analiza dokładności przyrządów i systemów pomiarowych;
- analiza dokładności przetwarzania sygnałów w układach elektronicznych, takich jak filtry, mostki czy przetworniki cyfrowo-analogowe;
- badanie wrażliwości układów elektronicznych na rozrzut parametrów elementów składowych;
- małosygnałowa i szumowa analiza układów elektronicznych.

Przykład 2.11. Parametry elementów, z których zbudowany jest filtr dolnoprzepustowy przedstawiony na rys. 2.1:

$$\tilde{R}_1 = R_1(1 + \rho_1), \quad \tilde{R}_2 = R_2(1 + \rho_2) \quad \text{oraz} \quad \tilde{C}_2 = C_2(1 + \zeta_2)$$

zostały dobrane z tolerancją jednoprocentową, tzn. $|\rho_1|, |\rho_2|, |\zeta_2| \leq 1\%$. Oszacować rozrzut trzydecybelowej pulsacji granicznej filtru, spowodowany rozrzutem parametrów jego elementów.



Rysunek 2.1. Filtr dolnoprzepustowy

Napięciowo-napięciowa transmitancja filtru wyraża się wzorem:

$$K(s) = \frac{U_2(s)}{U_1(s)} = \frac{1}{(1 + R_1/R_2) \left(1 + s \frac{R_1 C_2}{1 + R_1/R_2} \right)}$$

gdzie $U_1(s)$ jest transformatą Laplace'a napięcia $u_1(t)$, zaś $U_2(s)$ – transformatą Laplace'a napięcia $u_2(t)$. Trzydecybelową pulsację graniczną tego filtru można wyznaczyć więc jako odwrotność współczynnika przy zmiennej s w mianowniku transmitancji:

$$\omega_{3\text{dB}} = \frac{1 + R_1/R_2}{R_1 C_2}$$

Pulsacja graniczna, zaburzona z powodu rozrzutu parametrów elementów filtru, wyraża się zatem wzorem:

$$\tilde{\omega}_{3\text{dB}} = \frac{1 + \tilde{R}_1/\tilde{R}_2}{\tilde{R}_1 \tilde{C}_2} = \frac{1 + \frac{R_1(1 + \rho_1)}{R_2(1 + \rho_2)}}{R_1(1 + \rho_1) C_2(1 + \zeta_2)}$$

Stosując reguły rachunku epsilonów, wyrażeniu temu można nadać postać:

$$\tilde{\omega}_{3\text{dB}} = \omega_{3\text{dB}} \left[1 + \frac{R_1}{R_1 + R_2} (\rho_1 - \rho_2) - \rho_1 - \zeta_2 \right]$$

a stąd wywnioskować, że:

$$\delta[\tilde{\omega}_{3\text{dB}}] = \frac{R_1}{R_1 + R_2} (\rho_1 - \rho_2) - \rho_1 - \zeta_2 = -\frac{R_2}{R_1 + R_2} \rho_1 - \frac{R_1}{R_1 + R_2} \rho_2 - \zeta_2$$

Oszacowanie rozrzutu trzydecybelowej pulsacji granicznej wynika więc z nierówności:

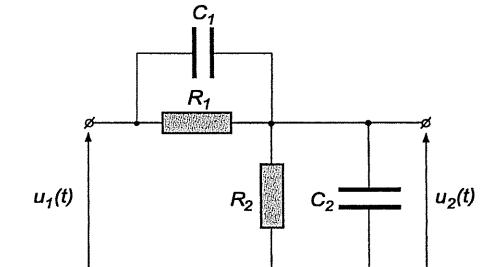
$$\begin{aligned} |\delta[\tilde{\omega}_{3\text{dB}}]| &\leq \left| \frac{R_2}{R_1 + R_2} \right| |\rho_1| + \left| \frac{R_1}{R_1 + R_2} \right| |\rho_2| + |\zeta_2| \leq \left(\frac{2R_2}{R_1 + R_2} + 1 \right) \cdot 1\% = \\ &= \frac{R_1 + 3R_2}{R_1 + R_2} \cdot 1\% \end{aligned}$$

Zadanie 2.8. Parametry elementów, z których zbudowany jest dzielnik przedstawiony na rys. 2.2:

$$\tilde{R}_1 = R_1(1 + \rho_1), \quad \tilde{R}_2 = R_2(1 + \rho_2), \quad \tilde{C}_1 = C_1(1 + \zeta_1) \quad \text{oraz} \quad \tilde{C}_2 = C_2(1 + \zeta_2)$$

zostały dobrane z tolerancją jednoprocentową, tzn. $|\rho_1|, |\rho_2|, |\zeta_1|, |\zeta_2| \leq 1\%$. Oszacować nierównomierność charakterystyki amplitudowo-częstotliwościowej

dzielnika, spowodowaną rozrzutem parametrów jego elementów przy założeniu, że $R_1 = R_2$ i $C_1 = C_2$.



Rysunek 2.2. Dzielnik napięcia

Odp.: $\delta[|K(j\omega)|] \leq 1\% \text{ dla } \omega \in (-\infty, +\infty)$.

ELEMENTY ANALIZY ALGORYTMÓW ITERACYJNYCH

3.1. INFORMACJE WSTĘPNE

Istotą algorytmu iteracyjnego jest powtarzanie pewnych operacji matematycznych (od łac. *iteratio* – powtarzanie). Szczególne znaczenie algorytmów iteracyjnych we współczesnych metodach numerycznych wynika z iteracyjnego charakteru działania komputera, ujawniającego się na wielu poziomach jego funkcjonowania. Przez powtarzanie kilku elementarnych operacji logicznych realizowane są cztery działania arytmetyczne, a przez powtarzanie tych działań – funkcje standardowe, takie jak $\sin(x)$, $\log(x)$ czy $\exp(x)$. W sposób iteracyjny – przez powtarzanie odpowiednich sekwencji operacji logicznych, działań arytmetycznych i funkcji standardowych – rozwiązywane są tak podstawowe zadania numeryczne, jak wyznaczanie zer lub ekstremów funkcji nieliniowych. Metodą *prób i błędów*, a więc również iteracyjnie, rozwiązuje się wiele – jeśli nie większość – problemów projektowania inżynierskiego, coraz częściej przy użyciu komputerów. Wszystko to uzasadnia celowość poświęcenia odrębnego rozdziału metodom analizy algorytmów iteracyjnych.

Algorytm iteracyjny definiuje się zwykle wzorem:

$$\mathbf{y}^{(i+1)} = \varphi_i(\mathbf{y}^{(i)}, \mathbf{y}^{(i-1)}, \dots; \mathbf{x}) \quad \text{dla } i = 0, 1, \dots \quad (3.1)$$

w którym φ_i jest operatorem znanym z dokładnością do algorytmu obliczeniowego. Wektory $\mathbf{y}^{(i+1)}, \mathbf{y}^{(i)}, \dots$ w tym wzorze są kolejnymi przybliżeniami rozwiązania zadania numerycznego dla danych $\mathbf{x} \in \mathbb{D}$. Analiza algorytmów iteracyjnych w tak ogólnej postaci jest zadaniem złożonym, wykraczającym poza założone ramy niniejszego podręcznika. Metodyka tej analizy zostanie tutaj zilustrowana na przykładzie algorytmów stacjonarnych, tzn. takich, dla których:

$$\varphi_0(\cdot) \equiv \varphi_1(\cdot) \equiv \varphi_2(\cdot) \equiv \dots \equiv \varphi \quad (3.2)$$

i zarazem skalarnych, dla których wynik jest skalarem:

$$y^{(i+1)} = \varphi(y^{(i)}, y^{(i-1)}, \dots; \mathbf{x}) \quad \text{dla } i = 0, 1, \dots \quad (3.3a)$$

Ze względu na ostatnie założenie możliwe jest zastosowanie w niniejszym podręczniku uproszczonej notacji:

$$y_{i+1} = \varphi(y_i, y_{i-1}, \dots; \mathbf{x}) \quad \text{dla } i = 0, 1, \dots \quad (3.3b)$$

gdzie $y_i \equiv y^{(i)}$ dla $i = 0, 1, \dots$ Algorytm realizacji operatora φ może być dany w dowolnej z form przedstawionych w podrozdziale 1.4; w przykładach i zadaniach, zawartych w niniejszym rozdziale, zakłada się, że wynika on jednoznacznie z postaci matematycznego zapisu tego operatora.

Przykład 3.1. Algorytm Herona wyznaczania pierwiastka kwadratowego ma postać:

$$y_0 = \max\{x, 1\}, \quad y_{i+1} = \frac{1}{2} \left(y_i + \frac{x}{y_i} \right) \quad \text{dla } i = 0, 1, \dots$$

Przykład 3.2. Algorytm rozwiązywania równania algebraicznego $f(y) = 0$ metodą stycznych Newtona ma postać:

$$y_{i+1} = y_i - \frac{f(y_i)}{f'(y_i)} \quad \text{dla } i = 0, 1, \dots$$

Zakłada się, że algorytmy wyznaczania wartości funkcji $f(y)$ i jej pochodnej $f'(y)$ są dane.

Cechą decydującą o praktycznej użyteczności algorytmu iteracyjnego jest jego zbieżność, tzn. zdolność do generowania zbieżnych ciągów kolejnych przybliżeń $\{y_i \mid i = 0, 1, \dots\}$ przynajmniej dla jednej wartości przybliżenia początkowego y_0 . Algorytm iteracyjny jest więc zbieżny, jeśli:

$$\forall \mathbf{x} \in \mathbb{D} \quad \exists y_\infty \in \mathbb{Y}_0 \neq \emptyset : y_i \xrightarrow{i \rightarrow \infty} y_\infty \quad (3.4)$$

gdzie y_∞ jest punktem zbieżności tego algorytmu, zaś \mathbb{Y}_0 – zbiorem przybliżeń początkowych, dla których algorytm ten jest zbieżny do y_∞ . Cechą istotną dla praktycznej użyteczności algorytmu iteracyjnego jest jego zbieżność do rozwiązania zadania numerycznego.

Zbieżność algorytmu iteracyjnego do wartości y_∞ – będącej rozwiązaniem zadania numerycznego – oznacza, że kolejne przybliżenia y_i są coraz bliższe tego rozwiązania, a więc bezwzględny błąd metody iteracyjnej $\Delta_i = y_i - y_\infty$ maleje do zera z liczbą iteracji. Szybkość tej zbieżności charakteryzuje funkcja:

$$\Delta_{i+1} = \Phi_i(\Delta_i) \quad \text{dla } i = 0, 1, \dots \quad (3.5)$$

wiązająca błędy bezwzględne dwóch kolejnych przybliżeń (Δ_i oraz Δ_{i+1}) lub parametry tej funkcji. Ocena szybkości zbieżności dla wszystkich wartości $y_0 \in \mathbb{Y}_0$

i wszystkich iteracji ($i = 0, 1, \dots$) jest zadaniem trudnym i nieczęsto podejmowanym. Na ogół analizę zbieżności ogranicza się do dostatecznie małego otoczenia punktu zbieżności y_∞ (dostatecznie dużych wartości indeksu iteracji i), aby funkcję (3.5) można było aproksymować z zadowalającą dokładnością dwuparametrową funkcją:

$$\Delta_{i+1} = C \Delta_i^\rho \quad (3.6)$$

Parametr $C \in (-\infty, +\infty)$ nosi nazwę *współczynnika lokalnej zbieżności algorytmu iteracyjnego*, zaś parametr $\rho \in [0, \infty)$ – *wykładnika lokalnej zbieżności*.

W przypadku komputerowej realizacji algorytmu iteracyjnego moduł błędu Δ_i na ogół nie maleje do zera. Gdy jego wartości stają się porównywalne z błędami reprezentacji danych i błędami zaokrągleń operacji zmiennopozycyjnych, proces zbieżności ulega zakłóceniu: kolejne przybliżenia \tilde{y}_i (gdzie tylka wskaże na zmienopozycyjną realizację obliczeń) obarczone są błędami w sposób niekontrolowany zmieniającym się w pewnym przedziale $[-K \cdot \text{eps}, +K \cdot \text{eps}]$, wyznaczającym tzw. *osiągalną dokładność* algorytmu iteracyjnego. Współczynnik K jest umowną miarą tej dokładności.

3.2. ALGORYTMY ITERACYJNE JEDNOARGUMENTOWE

Algorytm jednoargumentowy ma postać:

$$y_{i+1} = \varphi(y_i) \quad \text{dla } i = 0, 1, \dots \quad (3.7)$$

Ocena jego zbieżności lokalnej opiera się na analizie rozwinięcia funkcji $\varphi(y)$ w szereg Taylora wokół punktu y_∞ :

$$y_{i+1} = \varphi(y_\infty) + \varphi'(y_\infty)(y_i - y_\infty) + \frac{1}{2}\varphi''(y_\infty)(y_i - y_\infty)^2 + \dots \quad (3.8)$$

który – po odjęciu stronami y_∞ :

$$y_{i+1} - y_\infty = \varphi(y_\infty) - y_\infty + \varphi'(y_\infty)(y_i - y_\infty) + \frac{1}{2}\varphi''(y_\infty)(y_i - y_\infty)^2 + \dots \quad (3.9)$$

i uwzględnieniu, że – z definicji punktu zbieżności – $y_\infty = \varphi(y_\infty)$, przybiera postać równania błędu bezwzględnego (3.5):

$$\Delta_{i+1} = \varphi'(y_\infty)\Delta_i + \frac{1}{2}\varphi''(y_\infty)\Delta_i^2 + \dots \quad (3.10)$$

Dla *dostatecznie małych* Δ_i ($\Delta_i \rightarrow 0$) równanie to można aproksymować funkcją (3.6) według następujących zasad:

$$\varphi'(y_\infty) \neq 0 \Rightarrow \Delta_{i+1} \approx \varphi'(y_\infty)\Delta_i \quad (3.11a)$$

$$\varphi'(y_\infty) = 0 \text{ i } \varphi''(y_\infty) \neq 0 \Rightarrow \Delta_{i+1} \approx \frac{1}{2}\varphi''(y_\infty)\Delta_i^2 \quad (3.11b)$$

$$\varphi'(y_\infty) = \varphi''(y_\infty) = 0 \text{ i } \varphi'''(y_\infty) \neq 0 \Rightarrow \Delta_{i+1} \approx \frac{1}{6}\varphi'''(y_\infty)\Delta_i^3 \quad (3.11c)$$

itd.

Ogólnie zatem dla algorytmów jednoargumentowych wykładnik lokalnej zbieżności jest liczbą naturalną, $\rho \in \mathbb{N}$, zaś współczynnik lokalnej zbieżności przybiera wartość:

$$C = \frac{1}{\rho!} \varphi^{(\rho)}(y_\infty) \quad (3.12)$$

Wynikają stąd następujące warunki lokalnej zbieżności tych algorytmów:

$$|C| < 1 \quad \text{dla } \rho = 1 \quad (3.13a)$$

$$|C| < \infty \quad \text{dla } \rho = 2, 3, \dots \quad (3.13b)$$

Przykład 3.3. Zbadać zbieżność lokalną algorytmu:

$$y_{i+1} = y_i - \frac{2}{15}(y_i^3 - x) \quad \text{dla } x \in [1, 8]$$

Punkty stacjonarne tego algorytmu spełniają równanie $y_\infty = \varphi(y_\infty)$, a zatem:

$$y_\infty = y_\infty - \frac{2}{15}(y_\infty^3 - x) \Rightarrow y_\infty^3 = x \Rightarrow y_\infty = \sqrt[3]{x} \in [1, 2]$$

W punkcie $y_\infty = \sqrt[3]{x}$:

$$\varphi'(y_\infty) = 1 - \frac{2}{15}(3y_\infty^2 - 0) = 1 - \frac{2}{5}y_\infty^2 \neq 0$$

a zatem – zgodnie ze wzorem (3.11a) – $\rho = 1$, tzn. badany algorytm może być zbieżny liniowo, o ile współczynnik C spełnia warunek (3.13a). Zgodnie ze wzorem (3.12) jest on funkcją danej x :

$$C = 1 - \frac{2}{5}y_\infty^2 = 1 - \frac{2}{5}x^{\frac{2}{3}}$$

Dla $x \in [1, 8]$ zachodzi nierówność $|C| < 1$, ponieważ $C \in \left[-\frac{3}{5}, +\frac{3}{5}\right]$; badany algorytm jest więc zbieżny liniowo.

Przykład 3.4. Zbadać zbieżność lokalną algorytmu Herona, zdefiniowanego w przykładzie 3.1.

Punkty stacjonarne tego algorytmu spełniają równość:

$$y_\infty = \frac{1}{2} \left(y_\infty + \frac{x}{y_\infty} \right)$$

z której wynika, że $y_\infty = \pm\sqrt{x}$. Odpowiadające temu punktowi parametry zbieżności wynikają z następującej implikacji opartej na wzorach (3.11b) i (3.12):

$$\begin{cases} \varphi'(y_\infty) = \frac{1}{2} \left(1 - \frac{x}{y_\infty^2} \right) = 0 \\ \varphi''(y_\infty) = \frac{x}{y_\infty^3} = \pm \frac{1}{\sqrt{x}} \neq 0 \end{cases} \Rightarrow \rho = 2 \quad \text{oraz} \quad C = \pm \frac{1}{2\sqrt{x}}$$

Algorytm Herona jest więc zbieżny kwadratowo.

Przykład 3.5. Zbadać zbieżność lokalną algorytmu Newtona, zdefiniowanego w przykładzie 3.2.

Zależność Δ_{i+1} od Δ_i wyznaczamy, dokonując przekształcenia:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \Rightarrow x_{i+1} - x_\infty = x_i - x_\infty - \frac{f(x_i)}{f'(x_i)} = \Delta_i - \frac{f(x_\infty + \Delta_i)}{f'(x_\infty + \Delta_i)}$$

a następnie rozwijając $f(y_i)$ oraz $f'(y_i)$ w szeregi Taylora wokół y_∞ :

$$\Delta_{i+1} = \Delta_i - \frac{f(x_\infty) + f'(x_\infty)\Delta_i + \frac{1}{2}f''(x_\infty)\Delta_i^2 + \dots}{f'(x_\infty) + f''(x_\infty)\Delta_i + \dots}$$

Ponieważ $f(y_\infty) = 0$, więc dla odpowiednio małych Δ_i , tzn. dla $i \rightarrow \infty$:

$$\Delta_{i+1} \approx \frac{\left[f'(x_\infty) + f''(x_\infty)\Delta_i \right] \Delta_i - \left[f(x_\infty) + f'(x_\infty)\Delta_i + \frac{1}{2}f''(x_\infty)\Delta_i^2 \right]}{f'(x_\infty) + f''(x_\infty)\Delta_i}$$

Stąd:

$$\Delta_{i+1} \approx \frac{\frac{1}{2}f''(x_\infty)\Delta_i^2}{f'(x_\infty)} = \frac{1}{2} \frac{f''(x_\infty)}{f'(x_\infty)} \Delta_i^2 \Rightarrow C = \frac{1}{2} \frac{f''(x_\infty)}{f'(x_\infty)}, \quad \rho = 2$$

Algorytm Newtona jest więc zbieżny kwadratowo.

Metodyka oceny osiągalnej dokładności algorytmu jednoargumentowego opiera się na założeniu, że dla odpowiednio dużej liczby iteracji błąd metody iteracyjnej Δ , staje się pomijalny względem błędu $y_i \vartheta_i$, spowodowanego zmiennopozycyjną reprezentacją danych i zaokrąglaniem wyników operacji zmiennopozycyjnych:

$$\tilde{y}_i = y_i(1 + \vartheta_i) \xrightarrow{i \rightarrow \infty} y_\infty(1 + \vartheta_i) \quad (3.14)$$

Błąd ϑ_{i+1} jest wynikiem akumulacji błędów powstających w kolejnych iteracjach w związku z wykonywaniem operacji zmiennopozycyjnych. Można pokazać, że jeśli błąd ten jest dostatecznie mały ($|\vartheta_{i+1}| \ll 1$), to dobrą jego aproksymację daje wzór rekurencyjny postaci:

$$\vartheta_{i+1} = Cy_\infty^{\rho-1} \vartheta_i^\rho + \Delta \vartheta_i \quad (3.15)$$

gdzie $Cy_\infty^{\rho-1} \vartheta_i^\rho$ jest składową błędu przeniesioną z poprzedniej iteracji, zaś $\Delta \vartheta_i$ – składową wniesioną podczas wykonywania bieżącej. Błędy danych przenoszą się na wynik obliczeń w sposób niezależny od algorytmu obliczeniowego, dlatego – dla uproszczenia analizy – będą pomijane w przykładach oceny osiągalnej dokładności algorytmów iteracyjnych.

Przykład 3.6. Oszacować osiągalną dokładność algorytmu z przykładu 3.3.

Zmiennopozycyjna realizacja pojedynczej iteracji ma postać:

$$\tilde{y}_{i+1} = \left\{ \tilde{y}_i - \frac{2}{15} \left[\tilde{y}_i^3 (1 + \eta_{pi}) - x \right] (1 + \eta_{mi}) (1 + \eta'_{oi}) \right\} (1 + \eta''_{oi})$$

gdzie η_{pi} jest błędem zaokrąglenia wyniku potęgowania, η_{mi} – błędem zaokrąglenia wyniku mnożenia zawartości nawiasu [...] przez 2/15, η'_{oi} – błędem zaokrąglenia wyniku odejmowania wskazanego nawiasem [...], zaś η''_{oi} – błędem zaokrąglenia wyniku odejmowania wskazanego nawiasem {...}. Po uwzględnieniu wzoru (3.14) i zastosowaniu reguły rachunku epsilonów (2.10), dla $i \rightarrow \infty$ otrzymuje się:

$$\tilde{y}_{i+1} = \left\{ y_\infty (1 + \vartheta_i) - \frac{2}{15} \left[y_\infty^3 (1 + 3\vartheta_i + \eta_{pi}) - x \right] (1 + \eta_{mi} + \eta'_{oi}) \right\} (1 + \eta''_{oi})$$

a stąd:

$$\tilde{y}_{i+1} = y_\infty \left\{ 1 + \vartheta_i - \frac{2}{15} \frac{x}{y_\infty} (3\vartheta_i + \eta_{pi}) (1 + \eta_{mi} + \eta'_{oi}) \right\} (1 + \eta''_{oi})$$

Po podstawieniu $x = y_\infty^3$ i wyłączeniu przed nawias dokładnej wartości rozwiązań, w trzech krokach otrzymuje się zależność (3.15):

$$\tilde{y}_{i+1} = y_\infty \left\{ 1 + \vartheta_i - \frac{2}{15} y_\infty^2 (3\vartheta_i + \eta_{pi}) (1 + \eta_{mi} + \eta'_{oi}) \right\} (1 + \eta''_{oi})$$

$$\tilde{y}_{i+1} = y_\infty \left\{ 1 + \left(1 - \frac{2}{5} y_\infty^2 \right) \vartheta_i - \frac{2}{15} y_\infty^2 \eta_{pi} + \eta''_{oi} \right\}$$

$$\vartheta_{i+1} = C \vartheta_i - \frac{2}{15} y_\infty^2 \eta_{pi} + \eta''_{oi}$$

z której wynika oszacowanie:

$$|\vartheta_{i+1}| \leq \frac{3}{5} |\vartheta_i| + \frac{23}{15} \text{eps} \leq \dots \leq \left(\frac{3}{5} \right)^{i+1} |\vartheta_0| + \left[\sum_{v=0}^i \left(\frac{3}{5} \right)^v \right] \frac{23}{15} \text{eps}$$

pozwalające wyznaczyć wskaźnik osiągalnej dokładności K :

$$|\vartheta_{i+1}| \leq \frac{\frac{23}{15} \text{eps}}{1 - \frac{3}{5}} = \frac{23}{6} \text{eps} \approx 4 \text{eps} \Rightarrow K = 4$$

Wynik uzyskany w tym przykładzie nietrudno uogólnić na wszystkie algorytmy zbieżne liniowo ($\rho = 1$):

$$K \text{eps} \approx \frac{\sup \{ |\Delta \vartheta_i(x)| \mid x \in \mathbb{D} \}}{1 - \sup \{ |C(x)| \mid x \in \mathbb{D} \}} \quad (3.16)$$

Osiągalna dokładność tych algorytmów jest więc uwarunkowana zarówno błędem zaokrągleń wyników operacji zmiennopozycyjnych, składających się na bieżącą iterację, jak efektem kumulacji błędu, związanym z przenoszeniem błędu od iteracji do iteracji. Dla algorytmów zbieżnych kwadratowo lub szybciej ($\rho = 2, 3, \dots$):

$$K \text{eps} \approx \sup \{ |\Delta \vartheta_i| \mid x \in \mathbb{D} \} \quad (3.17)$$

ponieważ błąd przeniesiony $C y_\infty^{\rho-1} \vartheta_i^\rho$ jest pomijalny względem błędu bieżącej iteracji $\Delta \vartheta_i$.

Przykład 3.7. Oszacować osiągalną dokładność algorytmu Herona.

Stosując mnemoniczne oznaczenia błędów zaokrągleń wyników operacji zmiennopozycyjnych, przebieg analizy można zapisać następującymi wzorami:

$$\tilde{y}_{i+1} = \frac{1}{2} \left[y_i (1 + \vartheta_i) + \frac{x}{y_i (1 + \vartheta_i)} (1 + \eta_{di}) \right] (1 + \eta_{si}) (1 + \eta_{mi})$$

$$\tilde{y}_{i+1} = \frac{1}{2} \left[\sqrt{x} + \sqrt{x} (1 + \eta_{di}) \right] (1 + \eta_{si} + \eta_{mi}) = \sqrt{x} \left(1 + \frac{\eta_{di}}{2} + \eta_{si} + \eta_{mi} \right)$$

$$|\vartheta_{i+1}| \leq \frac{5}{2} \text{eps} \Rightarrow K = \frac{5}{2}$$

Zadanie 3.1. Dany jest algorytm iteracyjny:

$$y_{i+1} = 0.1 [y_i (10 - x) + 1] \quad \text{dla } i = 0, 1, \dots$$

zbieżny do odwrotności liczby $x \in [1, 19]$. Wyznaczyć parametry zbieżności tego algorytmu; oszacować osiągalną dokładność (przy założeniu, że dane $x, 0.1, 1$ i 10 są reprezentowane dokładnie).

Odp.: $C = 1 - x/10$, $\rho = 1$, $K \approx 38$.

Zadanie 3.2. Dany jest algorytm iteracyjny:

$$y_{i+1} = y_i - \frac{2}{3} [\operatorname{tg}(y_i) - x] \quad \text{dla } i = 0, 1, \dots$$

zbieżny do $\operatorname{arctg}(x)$ dla $x \in [0, 1]$. Wyznaczyć parametry zbieżności tego algorytmu; oszacować osiągalną dokładność (przy założeniu, że dane x i $2/3$ są reprezentowane dokładnie).

Odp.: $C = (1 - 2x^2)/3$, $\rho = 1$, $K \approx 2.77$.

Zadanie 3.3. Pokazać, że algorytm iteracyjny:

$$y_{i+1} = \frac{2y_i + \frac{x}{y_i^2}}{3} \quad \text{dla } i = 0, 1, \dots$$

jest zbieżny do $\sqrt[3]{x}$ dla $x > 0$ oraz $y_0 > 0$. Wyznaczyć wskaźniki zbieżności tego algorytmu; oszacować osiągalną dokładność (przy założeniu, że dane $x, 2$ i 3 są reprezentowane dokładnie).

Odp.: $C = \frac{1}{\sqrt[3]{x}}$, $\rho = 2$, $K \approx 3 \frac{1}{3}$.

3.3. ALGORYTMY ITERACYJNE WIELOARGUMENTOWE

Analiza algorytmów wieloargumentowych jest na ogół zadaniem znacznie bardziej skomplikowanym niż analiza algorytmów jednoargumentowych i jej scisłe przedstawienie istotnie wykracza poza ramy niniejszego podręcznika. Metodyka wyznaczania parametrów zbieżności oraz osiągalnej dokładności tych algorytmów zostanie więc jedynie zarysowana i w sposób intuicyjny uzasadniona.

Procedura wyznaczania parametrów zbieżności, C i ρ , dla danego punktu stacjonarnego y_∞ wieloargumentowego algorytmu iteracyjnego składa się z następujących kroków:

- ❶ Podstawiając $y_{i-k} = y_\infty + \Delta_{i-k}$ ($k = -1, 0, 1, \dots$) do równania definiującego algorytm, $y_{i+1} = \varphi(y_i, y_{i-1}, \dots)$, wyznaczyć równanie błędów bezwzględnych:

$$\Delta_{i+1} = \Phi(\Delta_i, \Delta_{i-1}, \dots) \quad (3.18)$$

- ❷ Rozwinąć funkcję Φ w szereg potęgowy:

$$\Delta_{i+1} = \sum_{k_1} C_{k_1} \Delta_{i-k_1} + \sum_{k_1} \sum_{k_2} C_{k_1 k_2} \Delta_{i-k_1} \Delta_{i-k_2} + \dots \quad (3.19)$$

- ❸ Uprościć ten szereg, korzystając – być może, wielokrotnie – z nierówności prawdziwej dla dostatecznie małych błędów bezwzględnych:

$$(\Delta_{i-k_1} \Delta_{i-k_2} \dots) \Delta_{i-k} \ll (\Delta_{i-k_1} \Delta_{i-k_2} \dots) \quad (3.20)$$

- ❹ Korzystając z zależności wynikającej z definicji parametrów zbieżności:

$$\Delta_{i-k-1} = \left(\frac{1}{C} \Delta_{i-k} \right)^{\frac{1}{\rho}} \quad \text{dla } k = -1, 0, 1, \dots \quad (3.21)$$

nadać uproszczonemu szeregowi postać:

$$\Delta_{i+1} = \sum_k F_k(C, \rho) \Delta_i^{f_k(\rho)} \quad (3.22)$$

- ❺ Wyznaczyć parametr ρ z warunku:

$$\lim_{i \rightarrow \infty} \frac{\Delta_{i+1}}{C \Delta_i^\rho} = 1$$

równoważnego definicji parametrów zbieżności. W tym celu warunkowi temu nadać postać:

$$\lim_{\Delta_i \rightarrow 0} \sum_k \frac{F_k(C, \rho)}{C} \Delta_i^{f_k(\rho)-\rho} = 1 \quad (3.23)$$

a następnie wyznaczyć najmniejszą wartość ρ , dla której jednocześnie:

$$f_k(\rho) - \rho = 0 \quad \text{dla } k = k_1, k_2, \dots, k_J \quad (J \geq 1) \quad (3.24a)$$

oraz

$$f_k(\rho) - \rho > 0 \quad \text{dla } k \neq k_1, k_2, \dots, k_J \quad (3.24b)$$

- ❻ Wyznaczyć parametr C jako największe co do modułu rozwiązanie równania:

$$\sum_{j=1}^J F_k(C, \rho) C^{-1} = 1 \quad (3.25)$$

Przykład 3.8. Wyznaczyć parametry zbieżności algorytmu iteracyjnego zdefiniowanego wzorem:

$$y_{i+1} = y_{i-1} + \frac{(y_{i-1} - x)(y_{i-2} - 2x)}{y_i} \quad \text{dla } i = 2, 3, \dots$$

przyjmując (dla uproszczenia) $x = 1$.

Algorytm ma dwa punkty stacjonarne, $y_\infty = 1$ i $y_\infty = 2$, które należy rozpatrzyć osobno. Analiza działania algorytmu w pobliżu punktu $y_\infty = 1$ ma następujący przebieg:

$$\textcircled{1} \quad 1 + \Delta_{i+1} = 1 + \Delta_{i-1} + \frac{(1 + \Delta_{i-1} - 1)(1 + \Delta_{i-2} - 2)}{1 + \Delta_i}$$

$$\Delta_{i+1} = \frac{\Delta_{i-1} \Delta_i + \Delta_{i-2} \Delta_{i-1}}{1 + \Delta_i}$$

$$\textcircled{2} \quad \Delta_{i+1} \approx \Delta_{i-1} \Delta_i + \Delta_{i-2} \Delta_{i-1}$$

$$\textcircled{4} \quad \Delta_{i+1} = \left(\frac{1}{C} \Delta_i \right)^{\frac{1}{\rho}} \Delta_i + \left[\frac{1}{C} \left(\frac{1}{C} \Delta_i \right)^{\frac{1}{\rho}} \right]^{\frac{1}{\rho}} \left(\frac{1}{C} \Delta_i \right)^{\frac{1}{\rho}}$$

$$\Delta_{i+1} = C^{\frac{1}{\rho}} \Delta_i^{\frac{1+1}{\rho}} + C^{\frac{2}{\rho}} \Delta_i^{\frac{1}{\rho^2}} \Delta_i^{\frac{1+1}{\rho}}$$

$$\textcircled{5} \quad \frac{\Delta_{i+1}}{C \Delta_i^\rho} = C^{\frac{1}{\rho}-1} \Delta_i^{\frac{1+1-\rho}{\rho}} + C^{\frac{2}{\rho}-\frac{1}{\rho^2}-1} \Delta_i^{\frac{1}{\rho^2}+\frac{1}{\rho}-\rho} \xrightarrow[\Delta_i \rightarrow 0]{} 1 \Rightarrow \rho = 1.325$$

$$\textcircled{6} \quad C^{\frac{2}{\rho}-\frac{1}{\rho^2}-1} = 1 \Rightarrow C = 1$$

Pewnego komentarza wymaga jedynie krok ❾ powyższej procedury. Hipoteza:

$$1 + \frac{1}{\rho} - \rho = 0$$

która prowadzi do wniosku, że

$$\rho = \rho_1 = \frac{1}{2}(1 + \sqrt{5})$$

musi być odrzucona, ponieważ

$$\frac{1}{\rho_1^2} + \frac{1}{\rho_1} - \rho_1 < 0$$

Do rozwiązania prowadzi natomiast hipoteza:

$$\frac{1}{\rho^2} + \frac{1}{\rho} - \rho = 0$$

z której wynika wartość $\rho_2 \approx 1.325$ spełniająca nierówność:

$$1 + \frac{1}{\rho_2} - \rho_2 > 0$$

Otrzymane wartości parametrów zbieżności wskazują na to, że $y_\infty = 1$ jest punktem zbieżności badanego algorytmu.

Analiza działania algorytmu w pobliżu punktu $y_\infty = 2$ ma analogiczny przebieg:

$$\textcircled{1} \quad 2 + \Delta_{i+1} = 2 + \Delta_{i-1} + \frac{(2 + \Delta_{i-1} - 1)(2 + \Delta_{i-2} - 2)}{2 + \Delta_i}$$

$$\Delta_{i+1} = \frac{\Delta_{i-1} + \frac{1}{2}\Delta_{i-2} + \frac{1}{2}\Delta_{i-1}\Delta_{i-2} + \frac{1}{2}\Delta_i\Delta_{i-1}}{2 + \Delta_i}$$

$$\textcircled{2} \quad \Delta_{i+1} \approx \Delta_{i-1} + \frac{1}{2}\Delta_{i-2} + \frac{1}{2}\Delta_{i-1}\Delta_{i-2} + \frac{1}{2}\Delta_i\Delta_{i-1}$$

$$\textcircled{3} \quad \Delta_{i+1} \approx \Delta_{i-1} + \frac{1}{2}\Delta_{i-2}$$

$$\textcircled{4} \quad \Delta_{i+1} = \left(\frac{1}{C}\Delta_i\right)^{\frac{1}{\rho}} + \frac{1}{2} \left[\frac{1}{C} \left(\frac{1}{C}\Delta_i \right)^{\frac{1}{\rho}} \right]^{\frac{1}{\rho}}$$

$$\textcircled{5} \quad \frac{\Delta_{i+1}}{C\Delta_i^\rho} = C^{-\frac{1}{\rho}-1} \Delta_i^{\frac{1}{\rho}-\rho} + \frac{1}{2} C^{-\frac{1}{\rho}-\frac{1}{\rho^2}} \Delta_i^{\frac{1}{\rho^2}-\rho}$$

$$\frac{1}{\rho} - \rho = 0 \quad i \quad \frac{1}{\rho^2} - \rho = 0 \quad \text{dla } \rho = 1$$

$$\textcircled{6} \quad C^{-2} + \frac{1}{2}C^{-3} = 1 \Rightarrow C^3 - C - \frac{1}{2} = 0 \Rightarrow C > 1$$

Z przeprowadzonej analizy wynika, że $y_\infty = 2$ nie jest punktem zbieżności badanego algorytmu, ponieważ nie spełnia warunku (3.13a).

Przykład 3.9. Wyznaczyć wskaźniki zbieżności algorytmu rozwiązywania nieliniowych równań algebraicznych $f(y)$ metodą siecznych:

$$y_{i+1} = y_i - \frac{f(y_i)(y_i - y_{i-1})}{f(y_i) - f(y_{i-1})} \quad \text{dla } i = 1, 2, \dots$$

Funkcję Φ wyznaczamy, rozwijając $f(y_i)$ i $f(y_{i-1})$ w szeregi Taylora wokół y_∞ :

$$\Delta_{i+1} = \Delta_i - \frac{\left[f(y_\infty) + f'(y_\infty)\Delta_i + \frac{1}{2}f''(y_\infty)\Delta_i^2 + \dots \right] (\Delta_i - \Delta_{i-1})}{f(y_\infty) + f'(y_\infty)\Delta_i + \frac{1}{2}f''(y_\infty)\Delta_i^2 + \dots - f(y_\infty) - f'(y_\infty)\Delta_{i-1} - \frac{1}{2}f''(y_\infty)\Delta_{i-1}^2 - \dots}$$

Ponieważ $f(y_\infty) = 0$; więc dla odpowiednio małych Δ_{i-1} i Δ_i , tzn. dla $i \rightarrow \infty$:

$$\Delta_{i+1} \approx \Delta_i - \frac{\left[f'(y_\infty)\Delta_i + \frac{1}{2}f''(y_\infty)\Delta_i^2 \right] (\Delta_i - \Delta_{i-1})}{f'(y_\infty)(\Delta_i - \Delta_{i-1}) + \frac{1}{2}f''(y_\infty)(\Delta_i^2 - \Delta_{i-1}^2)}$$

Po podzieleniu licznika i mianownika przez $\Delta_i - \Delta_{i-1}$ i sprowadzeniu do wspólnego mianownika wyrażenie powyższe przybiera postać:

$$\Delta_{i+1} \approx \widehat{C} \Delta_i \Delta_{i-1} = \widehat{C} \Delta_i \left(\frac{1}{C} \Delta_i \right)^{\frac{1}{\rho}}$$

gdzie $\widehat{C} = \frac{1}{2} \frac{f''(y_\infty)}{f'(y_\infty)}$. Wynika stąd równanie:

$$\frac{\Delta_{i+1}}{C\Delta_i^\rho} = \widehat{C} C^{\frac{-1}{\rho}-1} \Delta_i^{\frac{1}{\rho}-\rho}$$

umożliwiające wyznaczenie wykładnika zbieżności:

$$1 + \frac{1}{\rho} - \rho = 0 \Rightarrow \rho = \frac{1}{2}(1 + \sqrt{5})$$

a następnie współczynnika zbieżności:

$$\widehat{C} C^{\frac{-1}{\rho}-1} = 1 \Rightarrow C = \widehat{C}^{\frac{1}{2}(\sqrt{5}-1)}$$

Zadanie 3.4. Wyznaczyć parametry zbieżności algorytmu:

$$y_{i+1} = \frac{y_i y_{i-1} (y_i + y_{i-1}) + 1}{y_i^2 + y_i y_{i-1} + y_{i-1}^2} \quad \text{dla } i = 1, 2, \dots$$

Odp.: $y_\infty = 1$, $C = 1$, $\rho \approx 1.62$.

Zadanie 3.5. Wyznaczyć parametry zbieżności algorytmu:

$$y_{i+1} = \frac{y_i - y_{i-1} + y_{i-2}}{y_i y_{i-2} - y_{i-1} + 1} \quad \text{dla } i = 2, 3, \dots$$

Odp.: $y_\infty = 1$, $C = 1$, $\rho \approx 1.47$.

Jak wynika ze wzoru (3.15) oraz definicji $\text{eps} = 5 \cdot 10^{-L}$ (dla reprezentacji dziesiętnej), efekt kumulacji błędu można w pierwszym przybliżeniu scharakteryzować funkcją $(5 \cdot 10^{-L})^\rho$. Efekt ten można uznać za zaniedbywalny, gdy wartość tej funkcji jest istotnie (np. dziesięciokrotnie) mniejsza od granicznego błędu reprezentacji:

$$(5 \cdot 10^{-L})^\rho \leq \frac{1}{10} \cdot 5 \cdot 10^{-L} \quad (3.26)$$

tzn. gdy

$$\rho \geq 1 + \frac{1}{L - \log(5)} \quad (3.27)$$

Przykład 3.10. Wyznaczyć osiągalną dokładność algorytmu z zadania 3.4.

Ponieważ $\rho = 1.62$, więc nierówność (3.27) jest spełniona dla $L \geq 3$. Oznacza to, iż efekt kumulacji błędu można zaniedbać nawet dla mało precyzyjnego komputera. Wniosek ten istotnie upraszcza procedurę oceny osiągalnej dokładności, sprowadzając ją do wyznaczenia błędu pojedynczej iteracji. Stosując mnemoniczne oznaczenia błędów zaokrągleń wyników operacji zmiennopozycyjnych i korzystając z założenia (3.14), otrzymuje się:

$$\tilde{y}_{i+1} = \frac{[y_\infty y_\infty (1 + \eta'_{mi}) (y_\infty + y_\infty) (1 + \eta'_{si}) (1 + \eta''_{mi}) + 1] (1 + \eta_{li})}{\{[y_\infty^2 (1 + \eta'_{pi}) + y_\infty y_\infty (1 + \eta'_{mi})] (1 + \eta''_{si}) + y_\infty^2 (1 + \eta''_{pi})\} (1 + \eta'''_{si})} (1 + \eta_d)$$

a stąd – po podstawieniu $y_\infty = 1$:

$$\tilde{y}_{i+1} = \frac{[2(1 + \eta'_{mi} + \eta'_{si} + \eta''_{mi}) + 1] (1 + \eta_{li} + \eta_d)}{3 \left(1 + \frac{1}{3} \eta'_{pi} + \frac{1}{3} \eta'_{mi} + \frac{2}{3} \eta''_{si} + \frac{1}{3} \eta''_{pi} + \eta'''_{si} \right)}$$

W wyniku wielokrotnego zastosowania reguł rachunku epsilonów (2.10) wyrażenie to przekształca się do postaci umożliwiającej odczytanie błędu pojedynczej iteracji:

$$\Delta \vartheta_i = \frac{1}{3} \eta'_{mi} + \frac{2}{3} \eta'_{si} + \frac{2}{3} \eta''_{mi} + \eta_{li} + \eta_d - \frac{1}{3} \eta'_{pi} - \frac{2}{3} \eta''_{si} - \frac{1}{3} \eta''_{pi} - \eta'''_{si}$$

Oszacowanie tego błędu metodą najgorszego przypadku prowadzi do wskaźnika osiągalnej dokładności badanego algorytmu iteracyjnego:

$$K \approx \left(\frac{1}{3} + \frac{2}{3} + \frac{2}{3} + 1 + 1 + \frac{1}{3} + \frac{2}{3} + \frac{1}{3} + 1 \right) = 6$$

Zadanie 3.6. Wyznaczyć osiągalną dokładność algorytmu z zadania 3.5.

Odp.: $K \approx 4$.

3.4. ANALIZA ALGORYTMÓW ITERACYJNYCH WSPOMAGANA KOMPUTEREM

Jako wprowadzenie do bogatej problematyki wykorzystania komputerów w analizie algorytmów iteracyjnych przedstawiona zostanie intuicyjno-empiryczna metoda wyznaczania parametrów zbieżności, oparta na analizie działania algorytmu zastosowanego do rozwiązywania zadania testowego, reprezentatywnego dla całej klasy zadań numerycznych. Nie jest to metoda pewna, ponieważ uzyskany przy jej użyciu wynik nie musi być adekwatną charakterystyką całej klasy zadań. Jej zaletą, z inżynierskiego punktu widzenia, jest prostota logiczna, łatwość implementacji i szybkość działania. Matematyczne podstawy tej metody wynikają ze wzorów definiujących parametry zbieżności:

$$\Delta_{i+2} = C \Delta_{i+1}^\rho \quad \text{oraz} \quad \Delta_{i+1} = \tilde{C} \Delta_i^\rho \quad (3.28)$$

Dzieląc te wzory stronami, eliminuje się parametr C :

$$\frac{\Delta_{i+2}}{\Delta_{i+1}} = \left(\frac{\Delta_{i+1}}{\Delta_i} \right)^\rho \Rightarrow \ln \left| \frac{\Delta_{i+2}}{\Delta_{i+1}} \right| = \rho \ln \left| \frac{\Delta_{i+1}}{\Delta_i} \right| \quad (3.29)$$

a stąd wyznacza wykładnik zbieżności:

$$\rho = \lim_{i \rightarrow \infty} \frac{s_{i+1}}{s_i} \quad (3.30)$$

gdzie:

$$s_i = \ln \left| \frac{\Delta_{i+1}}{\Delta_i} \right| \quad \text{dla } i = 0, 1, \dots$$

Wspomagana komputerem procedura wyznaczania parametrów zbieżności algorytmu iteracyjnego ma więc postać:

- opracować zadanie testowe, którego dokładne rozwiązanie y_∞ jest znane;
- stosując badany algorytm do rozwiązywania zadania testowego, wyznaczyć ciągi $\{y_i\}$, $\{\Delta_i\}$ i $\{s_i\}$;

- obliczyć estymatę wykładnika zbieżności według wzoru (3.30) oraz współczynnik zbieżności według wzoru wynikającego z definicji parametrów zbieżności;

$$C = \lim_{i \rightarrow \infty} \frac{\Delta_{i+1}}{\Delta_i^p} \quad (3.31)$$

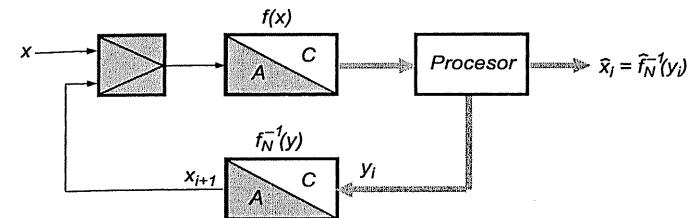
Badanie osiągalnej dokładności algorytmów iteracyjnych za pomocą komputera opiera się na wykorzystaniu wzorów (3.15), (3.16) i (3.17) oraz zastosowaniu metod analizy dokładności opisanych w podrozdziale 2.6 do wyznaczania błędu pojedynczej iteracji.

3.5. POZANUMERYCZNE ZASTOSOWANIA METOD ANALIZY ALGORYTMÓW ITERACYJNYCH

Metody analizy algorytmów iteracyjnych opisane w niniejszym rozdziale pozwalają rozwiązać wiele zadań dotyczących realizacji tych algorytmów za pomocą procesorów zmiennopozycyjnych – zadań analogicznych do wymienionych w podrozdziale 2.5. Dodatkową motywacją do nabycia sprawności w posługiwaniu się tymi metodami może być fakt, że stanowią one także wygodne narzędzie rozwiązywania wielu innych zadań inżynierskich. Wynika to stąd, iż postępowanie iteracyjne leży u podstaw funkcjonowania wielu obiektów technicznych, odznaczających się zdolnością do samoptymalizacji czy adaptacyjnością względem zmieniających się warunków zewnętrznych.

Przykład 3.11. W układzie przedstawionym na rys. 3.1 realizowana jest iteracyjna korekcja nieliniowości i niestalości charakterystyki statycznej $f(x)$ bloku analogowo-cyfrowego (blok A/C na rysunku). Mierzona wielkość x może być wielkością elektryczną, np. napięciem przemiennym, albo wielkością niesektryczną, np. natężeniem światła. Blok A/C składa się z przetwornika tej wielkości na napięcie stałe oraz z analogowo-cyfrowego przetwornika napięcia stałego. Charakterystyka statyczna tego bloku jest bardzo często nieliniowa, a co gorsza zmienia się w czasie jego eksploatacji, co utrudnia rozwiązanie problemu jej liniaryzacji wyłącznie za pomocą procesora cyfrowego realizującego odwrotność funkcji $f(x)$. W układzie przedstawionym na rys. 3.1 każdy pomiar wielkości x związany jest ze sprawdzeniem charakterystyki bloku A/C za pomocą bloku C/A realizującego przetwarzanie kodu cyfrowego na wielkość mierzoną. Zakłada się przy tym, że charakterystyka statyczna tego bloku odznacza się znacznie lepszą stałością w czasie niż charakterystyka bloku A/C. Dla wygody formalnej (prostoty analizy matematycznej problemu korekcji) charakterystyka ta jest interpreto-

wana jako odwrotność pewnej hipotetycznej charakterystyki bloku A/C – $f_N(x)$, zwanej charakterystyką nominalną.



Rysunek 3.1. Uk艂ad iteracyjnej korekcji nieliniowości charakterystyki statycznej przetwornika A/C

W pierwszym kroku algorytmu pomiarowego w procesorze zapami臋tywany jest wynik przetwarzania wielkości x przez blok A/C:

$$y_0 = f(x)$$

Dalsze kroki wykonywane s膮 po prze\u0144eczeniu komutatora wej\u0144ciowego w taki sposób, aby sygna\u0144 wyj\u0144ciowy bloku C/A:

$$x_{i+1} = f_N^{-1}(y_i) \quad \text{dla } i = 0, 1, \dots$$

by\u0144 podawany na wej\u0144cie bloku A/C. Na podstawie bież\u0144cego wyniku przetwarzania A/C procesor oblicza warto\u0144:

$$y_{i+1} = y_0 + y_i - f(x_{i+1}) \quad \text{dla } i = 0, 1, \dots$$

Mo\u0144na pokaza\u0144, \u0144e przy \u0144atwych do spe\u0144nienia wymaganiach dotyczących charakterystyki $f_N(x)$ oraz ró\u0144icy charakterystyk, nominalnej i rzeczywistej:

$$\Delta f(x) = f(x) - f_N(x)$$

ci\u0144 $\{y_i\}$ jest zbie\u0144ny do $y_\infty = f_N(x)$. Po podstawieniu wyrażenia definiuj\u0144cego x_{i+1} do wzoru definiuj\u0144cego y_{i+1} , otrzymuje si\u0144 standardowy zapis algorytmu jednoargumentowego:

$$y_{i+1} = y_0 + y_i - f(f_N^{-1}(x))$$

Bior\u0144 z kolei pod uwag\u00e3 definicj\u00e3 $\Delta f(x)$, zapis ten upraszcza si\u0144 do postaci:

$$y_{i+1} = y_0 + y_i - f_N(f_N^{-1}(y_i)) - \Delta f(f_N^{-1}(y_i)) = y_0 - \Delta f(f_N^{-1}(y_i))$$

Po podstawieniu $y_i = y_\infty = f_N(x)$ oraz $y_0 = f(x)$, otrzymuje si\u0144:

$$y_{i+1} = f_N(x) = y_\infty$$

co oznacza, \u0144e y_∞ jest punktem stacjonarnym algorytmu korekcji. Badanie pochodnej funkcji definiuj\u0144cej algorytm:

$$\varphi(y) = y_0 - \Delta f(f_N^{-1}(y))$$

w punkcie y_∞ , pozwala, przy użyciu wzorów (3.11)–(3.13), zbadać jego zbieżność do tego punktu:

$$\varphi'(y)\Big|_{y=y_\infty} = -\left[\Delta f'(z) \Big|_{z=f_N^{-1}(y)} \cdot (f_N^{-1}(y))' \right] \Big|_{y=f_N(x)} = -\frac{\Delta f'(x)}{f'_N(x)}$$

Pochodna ta jest na ogół niezerowa i co do modułu mniejsza od jedności, ponieważ korygowane względne różnice między pochodną charakterystyki rzeczywistej i nominalnej są zwykle niewielkie. Oznacza to, że badany algorytm korekcji jest liniowo zbieżny do y_∞ . Ponieważ $y_\infty = f_N(x)$, więc $x = f_N^{-1}(y_\infty)$; a zatem na podstawie przechowywanej w procesorze numerycznej reprezentacji $\tilde{f}_N(x)$ charakterystyki nominalnej $f_N(x)$ można wyznaczyć ciąg przybliżonych wartości x :

$$\hat{x}_i = \tilde{f}_N^{-1}(y_i) \quad \text{dla } i = 0, 1, \dots$$

zbieżny do $\hat{x}_\infty = \tilde{f}_N^{-1}(y_\infty) \approx x$.

ROZWIĄZYwanIE LINIOWYCH RÓWNAŃ ALGEBRAICZNYCH

Rozwiązywanie układów równań liniowych jest jednym z najczęściej występujących w praktyce inżynierskiej zagadnień numerycznych. Do układów równań liniowych prowadzą, między innymi, następujące zadania:

- analiza stanów ustalonych w liniowych obwodach elektrycznych;
- interpolacja danych pomiarowych;
- analiza pól elektromagnetycznych metodą rozwiązywania równań różniczkowych cząstkowych;
- aproksymacja charakterystyk statycznych czujników pomiarowych.

Rozważać będziemy metody numerycznego rozwiązywania układów równań liniowych:

$$\mathbf{Ax} = \mathbf{b} \tag{4.1}$$

gdzie \mathbf{x} jest N -wymiarowym wektorem niewiadomych, \mathbf{A} jest prostokątną macierzą danych o wymiarach $M \times N$, zaś \mathbf{b} – M -wymiarowym wektorem danych:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \cdots & a_{M,N} \end{bmatrix} \quad \text{oraz} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix}$$

W podrozdziałach 4.1–4.3 będziemy zakładać ponadto, że $M = N$, a \mathbf{A} jest macierzą nieosobliwą. W zależności od tego, czy macierz \mathbf{A} jest „mała” czy „duża” oraz uwarunkowania zadania stosuje się różne metody rozwiązywania układu równań liniowych. W przypadku zadań źle uwarunkowanych oraz „małych” macierzy stosuje się na ogół tzw. metody *bezpośrednie*, natomiast w przypadku dobrze warunkowych i „dużych” macierzy – metody *iteracyjne*. Warto dodać, że rozwiązywanie układów równań przy użyciu znanych z algebry wzorów Cramera, w których niezbędne wyznaczniki liczone byłyby z definicji, nie wchodzi w rachubę, gdyż już dla niewielkich macierzy o rozmiarach 20×20 komputer z procesorem Intel Centrino 1,8 GHz potrzebowałby na to około 86 000 lat.

4.1. POJĘCIA PODSTAWOWE

4.1.1. MACIERZE

Macierzą rozmiaru $M \times N$ jest nazywana tablica prostokątna zawierająca MN liczb rzeczywistych lub zespolonych. Przestrzeń macierzy o elementach rzeczywistych będzie oznaczana przez $\mathbb{R}^{M \times N}$, a przestrzeń macierzy zespolonych – przez $\mathbb{C}^{M \times N}$. Symbol \mathbf{A}^H będzie oznaczać macierz hermitowsko sprzężoną względem macierzy $\mathbf{A} = [a_{m,n}] \in \mathbb{C}^{M \times N}$, tzn. macierz:

$$\mathbf{A}^H = \begin{bmatrix} a_{1,1}^* & \cdots & a_{M,1}^* \\ \vdots & \ddots & \vdots \\ a_{1,N}^* & \cdots & a_{M,N}^* \end{bmatrix} \in \mathbb{C}^{N \times M} \quad (4.2)$$

gdzie „gwiazdka” jest operatorem sprzężenia: $a_{n,m}^* = \operatorname{Re}(a_{n,m}) - j\operatorname{Im}(a_{n,m})$, $j^2 = -1$. Odpowiednikiem macierzy hermitowsko sprzężonej w przestrzeni macierzy o elementach rzeczywistych jest macierz transponowana, oznaczana symbolem \mathbf{A}^T . Jeżeli dla macierzy kwadratowej zachodzi równość $\mathbf{A} = \mathbf{A}^H$ lub $\mathbf{A} = \mathbf{A}^T$, to macierz tę nazywa się, odpowiednio, macierzą zespoloną *hermitowską* lub *symetryczną*. Wśród macierzy kwadratowych wyróżnia się wiele specyficznych rodzajów, m.in.:

- macierz *diagonalną*, jeżeli jej wszystkie elementy poza główną przekątną są zerami;
- macierz *trójkątną górną (dolną)*, jeżeli jej wszystkie elementy leżące poniżej (powyżej) głównej przekątnej są zerami;
- macierz *redukowalną*, jeżeli istnieje macierz permutacji \mathbf{P} taka, że:

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix} \quad (4.3)$$

gdzie \mathbf{A}_{11} i \mathbf{A}_{22} są macierzami kwadratowymi. W przeciwnym przypadku macierz nazywa się *nieredukowalną*;

- macierz *diagonalnie słabo dominującą* $\mathbf{A} = [a_{m,n}] \in \mathbb{C}^{N \times N}$, jeżeli dla $m = 1, 2, \dots, N$ zachodzą nierówności:

$$|a_{m,m}| \geq \sum_{n=1, n \neq m}^N |a_{m,n}| \quad (4.4a)$$

oraz istnieje co najmniej jedna wartość m , dla której:

$$|a_{m,m}| > \sum_{n=1, n \neq m}^N |a_{m,n}| \quad (4.4b)$$

- macierz *diagonalnie dominującą* $\mathbf{A} = [a_{m,n}] \in \mathbb{C}^{N \times N}$, jeżeli spełnia ona warunek (4.4b) dla $m = 1, 2, \dots, N$;

- macierz \mathbf{A} ma własność A, jeżeli istnieje taka zero-jedynkowa macierz permutacji \mathbf{P} , że:

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{B} \\ \mathbf{C} & \mathbf{D}_2 \end{bmatrix} \quad (4.5)$$

gdzie \mathbf{D}_1 i \mathbf{D}_2 są macierzami diagonalnymi;

- macierz dodatnio określona A o elementach rzeczywistych, jeżeli dla każdego rzeczywistego, niezerowego wektora \mathbf{x} zachodzi nierówność: $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0$.

Niezerowy wektor \mathbf{x} i liczbę λ nazywamy odpowiednio *wektorem własnym* i *wartością własną* macierzy kwadratowej \mathbf{A} o rozmiarach $N \times N$, jeżeli spełniają równanie:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \quad (4.6)$$

Zbiór wszystkich wartości własnych macierzy \mathbf{A} nazywamy *widmem (spectrum)* tej macierzy i oznaczamy symbolem $\operatorname{Spect}(\mathbf{A})$. Liczbę $\rho(\mathbf{A}) = \{\max |\lambda| \mid \lambda \in \operatorname{Spect}(\mathbf{A})\}$ nazywamy promieniem spektralnym macierzy \mathbf{A} .

Z zależności (4.6) wynika, że:

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0 \quad (4.7)$$

Powyższy układ równań ma niezerowe rozwiązania, jeżeli wyznacznik macierzy $\mathbf{A} - \lambda\mathbf{I}$ przyjmuje wartość równą zeru. Zauważmy, że wyznacznik ten jest wielomianem stopnia N ze względu na wartości własne λ :

$$P(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}) \quad (4.8)$$

Wielomian ten, zwany wielomianem charakterystycznym macierzy \mathbf{A} , ma pierwiastki identyczne z wartościami własnymi macierzy \mathbf{A} .

Przykład 4.1. Macierz

$$\mathbf{A} = \begin{bmatrix} 1 & 5 & 2 & 6 \\ 0 & 5 & 0 & 6 \\ 3 & 6 & 4 & 5 \\ 0 & 6 & 0 & 5 \end{bmatrix}$$

jest macierzą redukowalną, ponieważ przy użyciu macierzy permutacji:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

można ją przekształcić do postaci:

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \begin{bmatrix} 4 & 3 & 6 & 5 \\ 2 & 1 & 5 & 6 \\ 0 & 0 & 5 & 6 \\ 0 & 0 & 6 & 5 \end{bmatrix}$$

która spełnia warunek (4.3). Macierz

$$\mathbf{B} = \begin{bmatrix} 1 & 3 & 0 & 3 \\ 2 & 2 & 2 & 0 \\ 0 & 3 & 1 & 3 \\ 2 & 0 & 2 & 2 \end{bmatrix}$$

ma własność A, ponieważ za pomocą tej samej macierzy permutacji \mathbf{P} można przekształcić ją do postaci:

$$\mathbf{P}^{-1}\mathbf{B}\mathbf{P} = \begin{bmatrix} 1 & 0 & 3 & 3 \\ 0 & 1 & 3 & 3 \\ 2 & 2 & 2 & 0 \\ 2 & 2 & 0 & 2 \end{bmatrix}$$

która spełnia warunek (4.5). Macierz

$$\mathbf{C} = \begin{bmatrix} 4 & 1 & 1 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 3 \end{bmatrix}$$

jest macierzą diagonalnie słabo dominującą, ponieważ jej elementy diagonalny w pierwszym wierszu jest większy od sumy elementów pozadiagonalnych, a elementy diagonalne w pozostałych wierszach są równe sumom odpowiednich elementów pozadiagonalnych; spełnione są więc warunki określone wzorem (4.4).



4.1.2. NORMY WEKTORÓW I MACIERZY

Niech \mathbb{R}^N (\mathbb{C}^N) oznacza przestrzeń liniową rzeczywistą (zespoloną), której elementami są N -wymiarowe wektory, o elementach rzeczywistych (zespolonych). Normy Höldera zdefiniowane są w tej przestrzeni następująco:

$$\|\mathbf{x}\|_p = \left(\sum_{n=1}^N |x_n|^p \right)^{1/p} \quad \text{dla } p = 1, 2, \dots, \infty \quad (4.9)$$

Szczególnymi przypadkami norm Höldera są:

- norma pierwsza:

$$\|\mathbf{x}\|_1 = \sum_{n=1}^N |x_n| \quad (4.10)$$

- norma druga (euklidesowa):

$$\|\mathbf{x}\|_2 = \left(\sum_{n=1}^N |x_n|^2 \right)^{1/2} \quad (4.11)$$

- norma maksimum:

$$\|\mathbf{x}\|_\infty = \sup \{ |x_n| \mid n = 1, 2, \dots, N \} \quad (4.12)$$

W analizie algorytmów numerycznych szczególnie użyteczne są, oprócz zdefiniowanych wyżej norm wektorów, normy *macierzy indukowane* przez owe normy wektorów:

$$\|\mathbf{A}\|_p = \sup \left\{ \frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p} \mid \mathbf{x} \neq \mathbf{0} \right\} \quad \text{dla } p = 1, 2, \dots, \infty \quad (4.13)$$

Zauważmy, że norma ta jest miarą maksymalnego „wydłużenia” wektora przez przekształcenie liniowe o macierzy \mathbf{A} . Z jej definicji wynika nierówność:

$$\|\mathbf{Ax}\|_p \leq \|\mathbf{A}\|_p \|\mathbf{x}\|_p \quad (4.14)$$

Najczęściej wykorzystywane są następujące normy indukowane:

$$\|\mathbf{A}\|_1 = \sup \left\{ \sum_{m=1}^M |a_{m,n}| \mid n = 1, 2, \dots, N \right\} \quad (4.15)$$

$$\|\mathbf{A}\|_2 = \sup \left\{ \frac{\|\mathbf{Ax}\|_2}{\|\mathbf{x}\|_2} \mid \mathbf{x} \neq \mathbf{0} \right\} = \sup \left\{ \frac{\sqrt{\mathbf{x}^H \mathbf{A}^H \mathbf{A} \mathbf{x}}}{\sqrt{\mathbf{x}^H \mathbf{x}}} \mid \mathbf{x} \neq \mathbf{0} \right\} \quad (4.16)$$

$$\|\mathbf{A}\|_\infty = \sup \left\{ \sum_{n=1}^N |a_{m,n}| \mid m = 1, 2, \dots, M \right\} \quad (4.17)$$

Norma zdefiniowana wzorem (4.16) nazywana jest często *normą spektralną*, ponieważ:

$$\|\mathbf{A}\|_2 = \sup \{ \sqrt{\lambda} \mid \lambda \in \text{Spect}(\mathbf{A}^H \mathbf{A}) \} \quad (4.18)$$

4.1.3. UWARUNKOWANIE ZADANIA ROZWIĄZYWANIA UKŁADU LINIOWYCH RÓWNAŃ ALGEBRAICZNYCH

Rozwiążając układ równań liniowych (4.1) w arytmetyce zmiennopozycyjnej, należy brać pod uwagę konsekwencje przybliżonej reprezentacji danych. Zbadajmy najpierw wpływ wynikającego stąd błędu $\Delta\mathbf{b}$ reprezentacji prawej strony równania na zaburzenie rozwiązań $\Delta\mathbf{x}$, przy założeniu, że \mathbf{A} jest nieosobliwą macierzą kwadratową:

$$\mathbf{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b} \quad (4.19)$$

Po odjęciu stronami równania (4.1) od równania (4.19), otrzymuje się $\mathbf{A}\Delta\mathbf{x} = \Delta\mathbf{b}$, a stąd:

$$\Delta\mathbf{x} = \mathbf{A}^{-1}\Delta\mathbf{b} \quad (4.20)$$

Ze wzoru (4.14) wynika, że dla odpowiadających sobie norm wektorów i indukowanych norm macierzy zachodzą nierówności:

$$\|\Delta\mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \|\Delta\mathbf{b}\| \quad (4.21)$$

$$\frac{\|\mathbf{b}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\| \quad (4.22)$$

Po pomnożeniu (4.21) i (4.22) stronami i podzieleniu przez $\|\mathbf{b}\|$, otrzymuje się:

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}^{-1}\| \|\mathbf{A}\| \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} = \text{cond}(\mathbf{A}) \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} \quad (4.23)$$

gdzie wielkość:

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\| \|\mathbf{A}^{-1}\| \quad (4.24)$$

nazywana *wskaznikiem uwarunkowania* układu (4.1) lub *wskaznikiem uwarunkowania* macierzy \mathbf{A} , charakteryzuje wrażliwość rozwiązania układu równań na zaburzenie wektora \mathbf{b} . Można udowodnić [D2], że charakteryzuje ona także wrażliwość rozwiązania na zaburzenie macierzy układu równań. Łączny skutek zaburzenia macierzy i wektora prawej strony układu równań podlega następującemu oszacowaniu:

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\text{cond}(\mathbf{A}) \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}}{1 - \text{cond}(\mathbf{A}) \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}} + \text{cond}(\mathbf{A}) \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} \quad (4.25)$$

Nierówność powyższa jest prawdziwa dla dowolnej normy wektora oraz odpowiedniej indukowanej normy macierzy, o ile $\|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\| < 1$.

Przykład 4.2. Bardzo źle uwarunkowane są układy równań z macierzą Hilberta o elementach:

$$a_{m,n} = \frac{1}{m+n-1}, \quad m, n = 1, 2, \dots, N \quad (4.26)$$

W tablicy 4.1 przedstawiono wartości wskaźnika uwarunkowania macierzy Hilberta o różnych rozmiarach, obliczone w środowisku MATLAB przy użyciu procedury `cond`. Jak widać, już dla macierzy 10×10 względny błąd rozwiązań jest około 10^{13} razy większy od względnego błędu wektora prawych stron układu równań.

Tablica 4.1

Zależność wskaźnika uwarunkowania od rozmiaru macierzy Hilberta

N	2	5	10	50
$\text{cond}(\mathbf{A})$	19.28	$4.77 \cdot 10^5$	$1.60 \cdot 10^{13}$	$9.37 \cdot 10^{18}$

Należy podkreślić, że oszacowania błędu rozwiązań oparte na wskaźniku uwarunkowania są oszacowaniami „najgorszego przypadku” i w praktyce należy liczyć się z mniejszymi błędami. Z drugiej jednak strony, są to oszacowania w sensie normy wektora rozwiązań, co oznacza, że poszczególne elementy wektora rozwiązań, a zwłaszcza o małych wartościach bezwzględnych, mogą być obarczone znacznymi błędami względnymi.

Przykład 4.3. Dla $N = 1, 2, \dots, 10$ rozwiązać układ równań algebraicznych (4.1), w którym:

- macierz \mathbf{A} jest macierzą Vandermonde'a z przedostatnią kolumną postaci $[1 \ 2 \ \dots \ N]^T$, utworzoną w środowisku MATLAB za pomocą procedury `vander`;
- wektor \mathbf{b} jest iloczynem macierzy \mathbf{A} i wektora dokładnych rozwiązań \mathbf{x} , składającego się z samych jedynek.

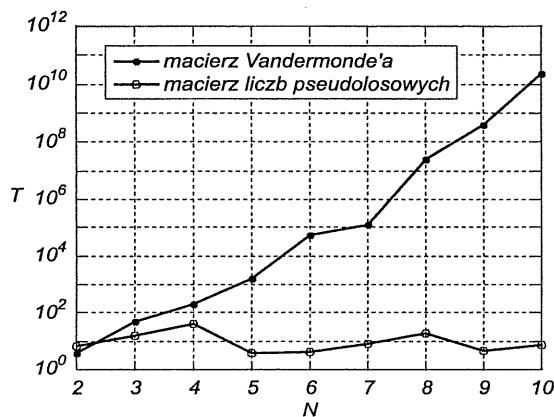
Wyznaczyć numerycznie i wykreślić w skali półilogarytmicznej zależność współczynnika przenoszenia względnego błędu elementów wektora \mathbf{b} na względny błąd rozwiązań układu równań od rozmiaru tego układu N . W tym celu zaburzać mnożnikowo wszystkie elementy wektora \mathbf{b} przy użyciu generatora liczb pseudolosowych o rozkładzie równomiernym w przedziale $[-0.005, +0.005]$ (wykorzystując funkcję `rand` środowiska MATLAB), a następnie obliczać współczynnik przenoszenia błędu według wzoru:

$$T = \frac{\|\Delta\mathbf{x}\|}{\frac{\|\mathbf{x}\|}{\|\Delta\mathbf{b}\|} \|\mathbf{b}\|}$$

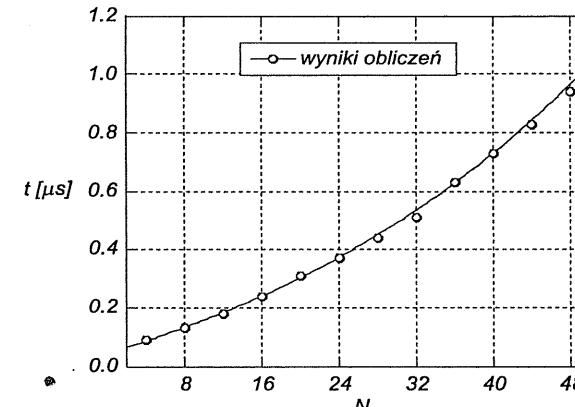
Powtórzyć obliczenia dla przypadku, gdy macierz A składa się z elementów pseudolosowych o wartościach z przedziału $(0,1)$.

Wyznaczyć i wykreślić zależność czasu rozwiązywania t układu równań z macierzą A o elementach pseudolosowych od rozmiaru tej macierzy dla $N = 4, 8, 12, 16, \dots, 48$.

Wyniki obliczeń przedstawiono na rys. 4.1 i rys. 4.2. Jak widać na rys. 4.1, wartości współczynnika przenoszenia błędu dla układów z macierzą Vandermonde'a wzrastają bardzo szybko z rozmiarem macierzy, co jest zgodne z przewidywaniami, gdyż wskaźnik uwarunkowania tej macierzy rośnie szybko w funkcji jej rozmiaru (podobnie jak dla macierzy Hilberta). Dla dobrze uwarunkowanych układów z macierzami o elementach pseudolosowych współczynniki przenoszenia błędu są niewielkie i praktycznie nie zależą od rozmiaru układu równań. Należy dodać, że oszacowania błędów przedstawione na rys. 4.1 mają charakter niejednoznaczny, z powodu losowego charakteru zaburzeń wektora b . W przypadku macierzy o elementach pseudolosowych dochodzi dodatkowy element niejednoznaczności związany z losowością realizacji macierzy A . Z tego powodu przy kolejnym powtórzeniu obliczeń wykresy błędów mogą się różnić nieco od wyników z rys. 4.1, gdyż za każdym razem są one realizowane na nieco innych zestawach danych. Z wykresu pokazanego na rys. 4.2 wynika, że czas rozwiązywania układu równań jest funkcją jej rozmiaru, rosnącą szybciej niż liniowo. Dokładniejsza analiza tej funkcji pokazuje, że czas obliczeń można aproksymować wielomianem trzeciego stopnia.



Rysunek 4.1. Zależność współczynnika przenoszenia błędu wektora b od rozmiaru układu równań N dla układu z macierzą Vandermonde'a (\bullet) i dla przykładowej realizacji układu z macierzą o elementach pseudolosowych (\circ); linią ciągłą zaznaczono średnią wyników uzyskanych dla 1000 realizacji



Rysunek 4.2. Zależności czasu obliczeń od rozmiaru układu równań N dla układu z macierzą o elementach pseudolosowych; wyniki pomiaru (\circ) oraz wykres wielomianu trzeciego stopnia aproksymującego te wyniki (linia ciągła)

4.2. METODA ELIMINACJI GAUSSA

Metoda eliminacji Gaussa polega na przekształcaniu wyjściowego układu równań:

$$\begin{aligned} a_{1,1}^{(1)}x_1 + a_{1,2}^{(1)}x_2 + \dots + a_{1,N}^{(1)}x_N &= b_1^{(1)} \\ a_{2,1}^{(1)}x_1 + a_{2,2}^{(1)}x_2 + \dots + a_{2,N}^{(1)}x_N &= b_2^{(1)} \\ \dots \\ a_{N,1}^{(1)}x_1 + a_{N,2}^{(1)}x_2 + \dots + a_{N,N}^{(1)}x_N &= b_N^{(1)} \end{aligned} \quad (4.27)$$

w równoważny układ o macierzy trójkątnej górnej. W pierwszym kroku – przy użyciu pierwszego z równań – eliminuje się niewiadomą x_1 z pozostałych równań, przez odjęcie stronami od tych równań pierwszego równania pomnożonego przez współczynniki:

$$l_{m,1} = \frac{a_{m,1}^{(1)}}{a_{1,1}^{(1)}} \quad \text{dla } m = 2, 3, \dots, N \quad (4.28)$$

przy założeniu, że $a_{1,1}^{(1)} \neq 0$. W ten sposób otrzymuje się równoważny układ równań:

$$\begin{aligned} a_{1,1}^{(1)}x_1 + a_{1,2}^{(1)}x_2 + \dots + a_{1,N}^{(1)}x_N &= b_1^{(1)} \\ 0 + a_{2,2}^{(2)}x_2 + \dots + a_{2,N}^{(2)}x_N &= b_2^{(2)} \\ \dots \\ 0 + a_{N,2}^{(2)}x_2 + \dots + a_{N,N}^{(2)}x_N &= b_N^{(2)} \end{aligned} \quad (4.29)$$

W drugim kroku – przy użyciu drugiego równania powyższego układu – eliminuje się z dalszych równań niewiadomą x_2 i otrzymuje:

$$\begin{aligned} a_{1,1}^{(1)}x_1 + a_{1,2}^{(1)}x_2 + a_{1,3}^{(1)}x_3 + \dots + a_{1,N}^{(1)}x_N &= b_1^{(1)} \\ 0 + a_{2,2}^{(2)}x_2 + a_{2,3}^{(2)}x_3 + \dots + a_{2,N}^{(2)}x_N &= b_2^{(2)} \\ 0 + 0 + a_{3,3}^{(3)}x_3 + \dots + a_{3,N}^{(3)}x_N &= b_3^{(3)} \\ \dots &\dots \dots \dots \dots \dots \\ 0 + 0 + a_{N,3}^{(3)}x_3 + \dots + a_{N,N}^{(3)}x_N &= b_N^{(3)} \end{aligned} \quad (4.30)$$

Po $N-1$ krokach uzyskuje się układ równań, którego macierz współczynników jest macierzą trójkątną górną:

$$\begin{aligned} a_{1,1}^{(1)}x_1 + a_{1,2}^{(1)}x_2 + a_{1,3}^{(1)}x_3 + \dots + a_{1,N}^{(1)}x_N &= b_1^{(1)} \\ 0 + a_{2,2}^{(2)}x_2 + a_{2,3}^{(2)}x_3 + \dots + a_{2,N}^{(2)}x_N &= b_2^{(2)} \\ 0 + 0 + a_{3,3}^{(3)}x_3 + \dots + a_{3,N}^{(3)}x_N &= b_3^{(3)} \\ \dots &\dots \dots \dots \dots \dots \\ 0 + 0 + 0 + \dots + a_{N,N}^{(N)}x_N &= b_N^{(N)} \end{aligned} \quad (4.31)$$

Układ ten można formalnie zapisać jako:

$$\mathbf{A}^{(N)}\mathbf{x} = \mathbf{b}^{(N)} \quad (4.32)$$

gdzie $\mathbf{A}^{(N)}$ jest macierzą trójkątną górną. Rozwiązywanie tego układu jest już bardzo proste: z ostatniego równania można bezpośrednio wyznaczyć niewiadomą x_N , a następnie – znając jej wartość – wyznaczyć z przedostatniego równania niewiadomą x_{N-1} itd. Opisana metoda eliminacji Gaussa może być zrealizowana, jeżeli $a_{kk}^{(k)} \neq 0$ dla $k=1, 2, \dots, N$; w przeciwnym przypadku, metoda wymaga modyfikacji, polegającej na odpowiednim przestawianiu równań układu. Z punktu widzenia dokładności obliczeń w arytmetyce zmiennopozycyjnej korzystne jest, aby moduły elementów $a_{kk}^{(k)}$ były jak największe. Modyfikacja algorytmu eliminacji Gaussa spełniająca to wymaganie nazywa się algorytmem eliminacji Gaussa z wyborem elementu głównego lub podstawowego. Częściowy wybór w k -tym kroku eliminacyjnym polega na znalezieniu wśród elementów $a_{m,k}^{(k)}$ ($m=k, k+1, \dots, N$) elementu o największym module, a następnie przestawieniu w macierzy $[a_{m,n}^{(k)}]$ wiersza, w którym się on znajduje, z wierszem k -tym. Pełny wybór ma miejsce, jeżeli element największy wyszukiwany jest w całej przekształcanej części macierzy. Jeżeli $a_{\mu,v}^{(k)} = \max \{a_{m,n}^{(k)} \mid k \leq m, n \leq N\}$, to zamienia się wiersz μ -ty z k -tym oraz kolumnę v -tą z k -tą. Algorytm eliminacji Gaussa z pełnym wyborem elementu głównego ma najlepsze właściwości numeryczne, jednak w praktyce stosuje się go rzadko, gdyż w większości zadań inżynierskich wystarcza algorytm z częściowym wyborem elementu głównego. Zauważmy przy okazji, że efektem zastosowania metody eliminacji Gaussa jest uzyskanie

równoważnego układu równań, którego macierz współczynników jest macierzą trójkątną górną (\mathbf{U}). Jeżeli mamy wiele układów równań o identycznych współczynnikach (różniących się jedynie wyrazami wolnymi), to warto dokonać rozkładu macierzy współczynników na iloczyn macierzy trójkątnych dolnej (\mathbf{L}) i górnej (\mathbf{U}); rozwiązywanie układu równań $\mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{b}$ wymaga bowiem znacznie mniejszej liczby operacji elementarnych niż rozwiązywanie układu oryginalnego. Algorytm wykorzystujący rozkład LU jest podstawowym algorytmem rozwiązywania dobrze uwarunkowanych układów równań liniowych. Zwrócić uwagę, że metoda eliminacji Gaussa jest równoważna przedstawieniu macierzy \mathbf{A} w postaci iloczynu:

$$\mathbf{A} = \mathbf{LU} \quad (4.33)$$

przy czym $\mathbf{U} = \mathbf{A}^{(N)}$ oraz

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ l_{2,1} & 1 & \dots & 0 \\ l_{3,1} & l_{3,2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{N,1} & l_{N,2} & & 1 \end{bmatrix} \quad (4.34)$$

Rozwiązywanie układu $\mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{b}$ jest równoważne rozwiązyaniu dwóch układów o macierzach trójkątnych – układu

$$\mathbf{Ly} = \mathbf{b} \quad (4.35)$$

względem \mathbf{y} , a następnie układu

$$\mathbf{Ux} = \mathbf{y} \quad (4.36)$$

względem \mathbf{x} .

W praktyce do uzyskania rozkładu LU częściej niż metodę Gaussa wykorzystuje się przekształcenia elementarne macierzy, np. przekształcenie Hauseholdera lub obroty Givensa, a także metodę Doolittle'a. Zagadnienia te są szeroko omówione w dostępnej literaturze – np. [F2] – wykraczają poza ramy niniejszego podręcznika i dlatego zostaną jedynie zilustrowane przykładem rozwiązywania układu (4.1) w systemie MATLAB.

Przykład 4.4. Rozwiązać układ równań (4.1) dla danych:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1.5 & 2 & 4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

metodą rozkładu LU.

Rozkład LU realizuje się w środowisku MATLAB przy użyciu procedury **lu**. Użycie tej procedury z dwoma argumentami wyjściowymi, $[L, U] = \text{lu}(A)$, powoduje uzyskanie macierzy trójkątnej górnej U oraz macierzy L będącej iloczynem macierzy trójkątnej dolnej i macierzy permutacji P zawierającej informacje o przedstawieniach wierszy i kolumn macierzy A dokonanych podczas realizacji procedury **lu** w związku z wyborem elementu głównego. Macierze A , L oraz U spełniają, oczywiście, równość $A = LU$.

Dla podanych w przykładzie danych uzyskuje się następujący rozkład:

$$L = \begin{bmatrix} 0.6667 & -0.5000 & 1.0000 \\ 0.6667 & 1.0000 & 0 \\ 1.0000 & 0 & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 1.5000 & 2.0000 & 4.0000 \\ 0 & 0.6667 & 0.3333 \\ 0 & 0 & -1.5000 \end{bmatrix}$$

Po dokonaniu rozkładu rozwiązywanie wyznacza się w dwóch krokach:

- rozwiązanie układu równań $Ly = b$ względem wektora y ;
- rozwiązanie układu równań $Ux = y$ względem wektora x .

Daje to: $y = [1.0000 \ 0.3333 \ 0.5000]^T$ oraz $x = [0.6667 \ 0.6667 \ -0.3333]^T$.

Szybkość zbieżności tego algorytmu zależy od parametru γ , ponieważ:

$$\dot{x} + \Delta x^{(i+1)} = \dot{x} + \Delta x^{(i)} + \gamma[a(\dot{x} + \Delta x^{(i)}) - b] \quad (4.38)$$

gdzie $\Delta x^{(i)} = x^{(i)} - \dot{x}$, $\Delta x^{(i+1)} = x^{(i+1)} - \dot{x}$, $\Delta x^{(i+1)} = \Delta x^{(i+1)} + \gamma a \Delta x^{(i)} = (1 + \gamma a) \Delta x^{(i)}$. Analizowany algorytm jest zbieżny, jeżeli:

$$|1 + \gamma a| < 1 \quad (4.39)$$

wówczas bowiem:

$$|\Delta x^{(i+1)}| < |\Delta x^{(i)}| \quad (4.40)$$

Z warunku zbieżności (4.39) wynikają wartości parametru γ , dla których ma miejsce zbieżność, a mianowicie:

$$\gamma \in \left(0, \frac{2}{|a|}\right), \quad \text{dla } a < 0 \quad \text{oraz} \quad \gamma \in \left(-\frac{2}{|a|}, 0\right), \quad \text{dla } a > 0$$

Naturalnym uogólnieniem rozważanego przypadku skalarnego są iteracyjne metody rozwiązywania układu równań liniowych $\mathbf{Ax} = \mathbf{b}$, którego dokładnym rozwiązaniem jest $\dot{\mathbf{x}} = \mathbf{A}^{-1}\mathbf{b}$:

$$\mathbf{x}^{(i+1)} = \mathbf{M}\mathbf{x}^{(i)} + \mathbf{w} \quad \text{dla } i = 0, 1, \dots \quad (4.41)$$

przy czym $\mathbf{x}^{(0)}$ jest przybliżeniem początkowym, zaś \mathbf{M} i \mathbf{w} spełniają następujące warunki:

- warunek zbieżności: $\rho(\mathbf{M}) < 1$;
- warunek zgodności: $\dot{\mathbf{x}} = \mathbf{M}\dot{\mathbf{x}} + \mathbf{w}$.

Szybkość zbieżności algorytmu iteracyjnego postaci (4.41) jest tym większa, im mniejszy jest promień spektralny macierzy \mathbf{M} – $\rho(\mathbf{M})$. Obliczenia są przerywane, gdy spełniony jest warunek zakończenia obliczeń; najczęściej ma on jedną z dwóch postaci:

- $\|\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\| \leq \Delta \mathbf{x}$, gdzie $\Delta \mathbf{x}$ jest założonym wskaźnikiem dopuszczalnego bezwzględnego błędu rozwiązania;
- $\frac{1}{\|\mathbf{b}\|} \|\mathbf{Ax}^{(i+1)} - \mathbf{b}\| \leq \delta \mathbf{b}$, gdzie $\delta \mathbf{b}$ jest założonym wskaźnikiem dopuszczalnego względnego błędu niedopasowania stron równania.

4.3. METODY ITERACYJNE

Metody bezpośrednie, o których była mowa w poprzednim podrozdziale, wymagają wykonania około N^3 działań arytmetycznych. Jeżeli wartość N jest bardzo duża, np. rzędu 10^4 , często konieczne staje się użycie metod iteracyjnych wymagających mniejszych nakładów obliczeniowych. Trzeba jednak pamiętać, że algorytmy iteracyjne mogą być stosowane tylko dla takich układów równań, dla których są zbieżne. Tytułem wprowadzenia do metod iteracyjnych rozważmy zastosowanie następującego algorytmu iteracyjnego do rozwiązywania skalarnego liniowego równania algebraicznego $a\mathbf{x} = \mathbf{b}$, którego dokładnym rozwiązaniem jest $\dot{\mathbf{x}} = \mathbf{b} / a$:

$$x^{(i+1)} = x^{(i)} + \gamma(a x^{(i)} - b) \quad \text{dla } i = 0, 1, \dots \quad (4.37)$$

4.3.1. METODA JACOBIEGO

Metoda Jacobiego opiera się na przedstawieniu macierzy \mathbf{A} w postaci sumy trzech macierzy:

$$\mathbf{A} \triangleq \mathbf{L} + \mathbf{D} + \mathbf{U} \quad (4.42)$$

gdzie \mathbf{D} jest macierzą diagonalną, \mathbf{L} jest dolną macierzą trójkątną o zerowych elementach diagonalnych, a \mathbf{U} jest górną macierzą trójkątną o zerowych elementach diagonalnych.

Przykład 4.5

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix} + \begin{bmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{bmatrix}$$

\mathbf{A} \mathbf{L} \mathbf{D} \mathbf{U}

Heurystyczne wyprowadzenie metody Jacobiego obejmuje następujące kroki:

$$\begin{aligned} \mathbf{D}\mathbf{x} &= -(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{b} \\ \mathbf{x} &= -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b} \\ \mathbf{x}^{(i+1)} &= \mathbf{M}\mathbf{x}^{(i)} + \mathbf{w} \quad \text{dla } i = 0, 1, \dots \end{aligned} \quad (4.43)$$

gdzie $\mathbf{M} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$ oraz $\mathbf{w} = \mathbf{D}^{-1}\mathbf{b}$. Warunkiem koniecznym i wystarczającym zbieżności algorytmu iteracyjnego zdefiniowanego wzorem (4.43) jest, aby macierz \mathbf{A} była nieredukowalna i diagonalnie dominująca lub słabo dominująca (por. wzór (4.4)).

4.3.2. METODA RICHARDSONA

Heurystyczne wyprowadzenie metody Richardsona opiera się na następujących przekształceniach układu równań (4.1):

$$\begin{aligned} \mathbf{x} + p\mathbf{A}\mathbf{x} &= \mathbf{x} + p\mathbf{b} \\ \mathbf{x} &= \mathbf{I}\mathbf{x} - p\mathbf{A}\mathbf{x} + p\mathbf{b} \\ \mathbf{x}^{(i+1)} &= \mathbf{M}\mathbf{x}^{(i)} + \mathbf{w} \quad \text{dla } i = 0, 1, \dots \end{aligned} \quad (4.44)$$

gdzie $\mathbf{M} = \mathbf{I} - p\mathbf{A}$ oraz $\mathbf{w} = p\mathbf{b}$. Algorytm zdefiniowany wzorem (4.44) jest zbieżny do rozwiązania dokładnego, jeżeli promień spektralny macierzy $\mathbf{I} - p\mathbf{A}$ jest mniejszy od jedności. Aby zapewnić jak najlepszą zbieżność, parametr p dobiera się w taki sposób, aby promień ten był jak najmniejszy. Jeżeli macierz \mathbf{A} ma rzeczywiste wartości własne tego samego znaku (jest np. macierzą symetryczną dodatnio określona), to promień spektralny macierzy $\mathbf{I} - p\mathbf{A}$ jest najmniejszy,

gdy $p = 2 / (\lambda_{\min} + \lambda_{\max})$ [D2]. Odpowiadający tej wartości parametru p promień spektralny spełnia następującą nierówność:

$$\rho(\mathbf{I} - p\mathbf{A}) = \left| \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} \right| = \left| \frac{\lambda_{\max} / \lambda_{\min} - 1}{\lambda_{\max} / \lambda_{\min} + 1} \right| < 1 \quad (4.45)$$

Ponieważ dla macierzy symetrycznych $|\lambda_{\max} / \lambda_{\min}| = \text{cond}(\mathbf{A})$, promień spektralny można wyrazić przez wskaźnik uwarunkowania macierzy \mathbf{A} :

$$\rho(\mathbf{I} - p\mathbf{A}) = \frac{\text{cond}(\mathbf{A}) - 1}{\text{cond}(\mathbf{A}) + 1} \quad (4.46)$$

4.3.3. METODA GAUSSA-SEIDLIA

Pierwszy krok związany z wyprowadzeniem metody Gaussa-Seidla jest taki sam jak w metodzie Jacobiego (4.39). Kolejne kroki metody są następujące:

$$\begin{aligned} (\mathbf{L} + \mathbf{D})\mathbf{x} &= -\mathbf{U}\mathbf{x} + \mathbf{b} \\ (\mathbf{L} + \mathbf{D})\mathbf{x}^{(i+1)} &= -\mathbf{U}\mathbf{x}^{(i)} + \mathbf{b} \quad \text{dla } i = 0, 1, \dots \\ \mathbf{D}\mathbf{x}^{(i+1)} &= -\mathbf{L}\mathbf{x}^{(i+1)} - \mathbf{U}\mathbf{x}^{(i)} + \mathbf{b} \quad \text{dla } i = 0, 1, \dots \\ \mathbf{x}^{(i+1)} &= -\mathbf{D}^{-1}\mathbf{L}\mathbf{x}^{(i+1)} - \mathbf{D}^{-1}\mathbf{U}\mathbf{x}^{(i)} + \mathbf{D}^{-1}\mathbf{b} \quad \text{dla } i = 0, 1, \dots \end{aligned} \quad (4.47)$$

Aby zrozumieć działanie tego algorytmu, mimo występowania wektora $\mathbf{x}^{(i+1)}$ po lewej i po prawej stronie równości, warto wzór (4.47) przedstawić w postaci:

$$x_m^{(i+1)} = x_m^{(i)} - \frac{1}{a_{m,m}} \left(\sum_{n=1}^{m-1} a_{m,n} x_n^{(i+1)} + \sum_{n=m}^N a_{m,n} x_n^{(i)} - b_m \right) \quad \text{dla } m = 1, 2, \dots, N \quad (4.48)$$

Jak widać, do obliczenia $x_m^{(i+1)}$ wykorzystywane są wyłącznie wartości $x_1^{(i+1)}, \dots, x_{m-1}^{(i+1)}$ obliczone wcześniej. Metoda Gaussa-Seidla jest zbieżna, między innymi, dla macierzy nieredukowalnych i diagonalnie słabo dominujących oraz dla macierzy symetrycznych dodatnio określonych.

4.3.4. METODA SOR

Metoda SOR (od ang. *Successive Overrelaxation*) jest uogólnieniem metody Gaussa-Seidla polegającym na wprowadzeniu mnożnika ω we wzorze (4.48) w celu przyspieszenia zbieżności:

$$x_m^{(i+1)} = x_m^{(i)} - \frac{\omega}{a_{m,m}} \left(\sum_{n=1}^{m-1} a_{m,n} x_n^{(i+1)} + \sum_{n=m}^N a_{m,n} x_n^{(i)} - b_m \right) \quad \text{dla } m = 1, 2, \dots, N \quad (4.49)$$

W zapisie macierzowym algorytm iteracyjny metody SOR można przedstawić następująco:

$$\mathbf{D}\mathbf{x}^{(i+1)} = (1-\omega)\mathbf{D}\mathbf{x}^{(i)} - \omega(\mathbf{L}\mathbf{x}^{(i+1)} + \mathbf{U}\mathbf{x}^{(i)} - \mathbf{b}) \quad \text{dla } i = 0, 1, \dots \quad (4.50)$$

Po rozwiązaniu (4.50) względem $\mathbf{x}^{(i+1)}$ otrzymuje się ostatecznie definicję metody SOR w postaci:

$$\mathbf{x}^{(i+1)} = \mathbf{M}_\omega \mathbf{x}^{(i)} + \mathbf{w} \quad (4.51)$$

gdzie $\mathbf{M}_\omega = (\mathbf{D} + \omega\mathbf{L})^{-1}((1-\omega)\mathbf{D} - \omega\mathbf{U})$ oraz $\mathbf{w} = \omega(\mathbf{D} + \omega\mathbf{L})^{-1}\mathbf{b}$. Można udowodnić, że:

$$\rho(\mathbf{M}_\omega) \geq |\omega - 1| \quad (4.52)$$

dla dowolnej nieosobliwej macierzy \mathbf{A} i dowolnej liczby ω [D2]. Aby metoda zdefiniowana wzorem (4.49) była zbieżna, parametr ω musi przyjmować wartości z przedziału $(0, 2)$. Dla $\omega > 1$ metoda nosi nazwę kolejnych nadrelaksacji – SOR. Jeżeli macierz \mathbf{A} jest symetryczna i dodatnio określona, to metoda ta jest zbieżna dla dowolnej wartości $\omega \in (0, 2)$. Wybór optymalnej wartości parametru ω , dla której zbieżność jest najszybsza, przeprowadza się na ogół doświadczalnie, chociaż dla niektórych klas macierzy można tę wartość z góry oszacować. Na przykład, dla układu równań o macierzy symetrycznej, dodatnio określonej i mającej własność A wartość optymalna [D2]:

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \lambda_{\max}^2}} \quad (4.53)$$

W tym przypadku promień spektralny macierzy \mathbf{M}_ω jest określony następująco [D2]:

$$\rho(\mathbf{M}_\omega) = \omega_{\text{opt}} - 1 = \left(\frac{\sqrt{\text{cond}(\mathbf{A})} - 1}{\sqrt{\text{cond}(\mathbf{A})} + 1} \right)^2 \quad (4.54)$$

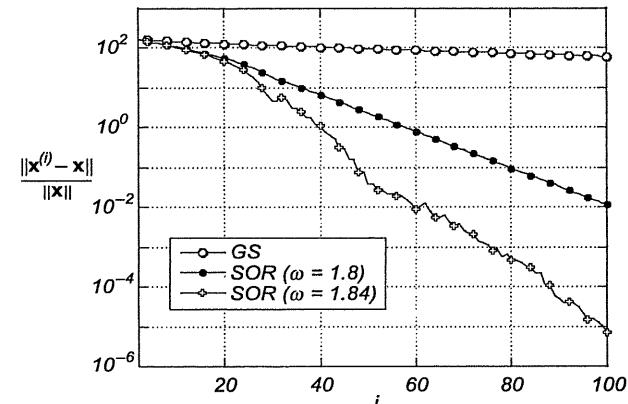
4.3.5. PORÓWNANIE ZBIEŻNOŚCI METOD ITERACYJNYCH

Metody Jacobiego, Richardsoна i Gaussa-Seidla można stosować jedynie do rozwiązywania układów bardzo dobrze uwarunkowanych, o wskaźnikach uwarunkowania nieprzekraczających 100. Zdecydowanie lepsze właściwości ma metoda SOR, która może być stosowana nawet w przypadku macierzy o wskaźnikach uwarunkowania rzędu 10000.

Przykład 4.6. Układ równań:

$$\begin{bmatrix} -4 & 1 & 1 & 0 & \cdots & 0 \\ 1 & -4 & 1 & 1 & \cdots & 0 \\ 1 & 1 & -4 & 1 & \cdots & 0 \\ 0 & 1 & 1 & -4 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \cdots & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{50} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

rozwijano metodą SOR (z parametrem $\omega = 1.8$ i 1.84) oraz metodą Gaussa-Seidla. Wyznaczono błędy rozwiązań w funkcji liczby iteracji, jako przybliżenie początkowe rozwiązania, przyjmując wektor $[1 \ 2 \ \dots \ 50]^T$. Wyniki przedstawiono na rys. 4.3.



Rysunek 4.3. Względny błąd rozwiązań układowów równań w funkcji liczby iteracji: GS – metoda Gaussa-Seidla; SOR($\omega = 1.8$) – metoda nadrelaksacji z parametrem $\omega = 1.8$; SOR($\omega = 1.84$) – metoda nadrelaksacji z parametrem $\omega = 1.84$

Jak widać, dla rozważanego układu równań, wśród trzech porównywanych metod, najszybciej zbieżna jest metoda nadrelaksacji z parametrem $\omega = 1.84$. Oczywiście, dla innych układów optymalne mogą okazać się inne wartości tego parametru.

Liczba iteracji niezbędna do uzyskania rozwiązania przy założonym poziomie błędów zależy od przyjętego przybliżenia początkowego. W praktyce rzadko jest ono dane i wybiera się je w zasadzie dość dowolnie, w taki jednak sposób, aby zapewnić zbieżność ciągu rozwiązań. Na koniec należy wspomnieć o jeszcze jednej często stosowanej metodzie iteracyjnej, jaką jest metoda gradientów sprzężonych

(ang. *Conjugate Gradient*) stosowana do rozwiązywania dużych układów równań i niewymagająca znajomości parametrów określających jej zbieżność, jak metoda SOR. Szczegóły można znaleźć w literaturze [D2].

4.4. LINIOWE ZADANIE NAJMIEJSZYCH KWADRATÓW

Rozważmy nadokreślony układ liniowych równań algebraicznych (4.1), tzn. układ składający się z większej liczby równań niż liczba niewiadomych ($M > N$). Liniowe zadanie najmniejszych kwadratów, związane z takim układem równań, polega na znalezieniu wśród wektorów \mathbf{x} minimalizujących normę euklidesową wektora błędu:

$$\|\mathbf{b} - \mathbf{Ax}\|_2 = \inf\{\|\mathbf{b} - \mathbf{Ay}\| \mid \mathbf{y} \in \mathbb{R}^N\} \quad (4.55)$$

takiego wektora $\hat{\mathbf{x}}$, który ma najmniejszą normę:

$$\|\hat{\mathbf{x}}\|_2 = \inf\{\|\mathbf{x}\|_2 \mid \mathbf{x} \in \mathbb{R}^M\} \quad (4.56)$$

W praktyce inżynierskiej liniowe zadanie najmniejszych kwadratów wiąże się najczęściej z aproksymacją danych pomiarowych. Indywidualne pomiary są, na ogół, obarczone pewnymi błędami wynikającymi ze skończonej precyzji przyrządów pomiarowych. Wielokrotne powtarzanie pomiarów prowadzi do uśrednienia błędów przypadkowych, także w pomiarach pośrednich.

Jedna z najlepszych metod rozwiązywania zadania (4.1) dla macierzy (4.50) polega na rozkładzie macierzy \mathbf{A} według wartości szczególnych – ang. *Singular Value Decomposition* (SVD). Rozkład ten ma postać:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \quad (4.57)$$

gdzie $\mathbf{U} \in \mathbb{R}^{M \times M}$ i $\mathbf{V} \in \mathbb{R}^{N \times N}$ są macierzami ortogonalnymi, natomiast:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_N \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Liczby $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \sigma_{r+2} = \dots = \sigma_N = 0$ nazywane są *wartościami szczególnymi* macierzy \mathbf{A} , przy czym r jest rzędem macierzy \mathbf{A} . Wartości szczególne są pierwiastkami wartości własnych macierzy $\mathbf{A}^T\mathbf{A}$; kolumny macierzy \mathbf{V}

są wektorami własnymi tej macierzy, a kolumny macierzy \mathbf{U} są wektorami własnymi macierzy $\mathbf{A}\mathbf{A}^T$ [D2]. Znając rozkład macierzy według wartości szczególnych, można łatwo wyznaczyć rozwiązanie liniowego zadania najmniejszych kwadratów, posługując się tzw. pseudoodwrotnością macierzy \mathbf{A} :

$$\mathbf{A}^+ = \mathbf{V}\Sigma^+\mathbf{U}^T \quad (4.58)$$

gdzie

$$\Sigma^+ = \begin{bmatrix} \sigma_1^+ & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2^+ & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_N^+ & \cdots & 0 \end{bmatrix}$$

przy czym:

$$\sigma_n^+ = \begin{cases} 1/\sigma_n, & \text{gdy } \sigma_n \neq 0 \\ 0, & \text{gdy } \sigma_n = 0 \end{cases}$$

Rozwiązanie zadania najmniejszych kwadratów ma postać:

$$\hat{\mathbf{x}} = \mathbf{A}^+\mathbf{b} \quad (4.59)$$

Wektor $\hat{\mathbf{x}}$ spełnia układ równań normalnych o postaci:

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b} \quad (4.60)$$

z symetryczną i dodatnio określona macierzą $\mathbf{A}^T\mathbf{A}$. Układ ten może być efektywnie rozwiązywany przy użyciu metod numerycznych opisanych w podrozdziale 4.2 lub 4.3, o ile wskaźnik uwarunkowania macierzy $\mathbf{A}^T\mathbf{A}$ nie osiąga zbyt dużych wartości. W przeciwnym przypadku realną alternatywą dla metody SVD jest rozkład macierzy \mathbf{A} na iloczyn macierzy o ortogonalnych kolumnach i macierzy trójkątnej górnej, czyli tzw. rozkład QR [D2]. Idea wykorzystania rozkładu QR do rozwiązywania układów liniowych równań algebraicznych zilustrowana zostanie przykładem.

Przykład 4.7. Rozwiązać układ równań z przykładu 4.4, wykorzystując rozkład QR:

$$\mathbf{A} = \mathbf{QR}$$

gdzie \mathbf{Q} jest macierzą ortogonalną (tzn. macierzą spełniającą warunek $\mathbf{QQ}^T = \mathbf{I}$), zaś \mathbf{R} jest macierzą trójkątną górną. Rozkładu QR macierzy \mathbf{A} z przykładu 4.4 w środowisku MATLAB można dokonać przy użyciu procedury qr:

$$\mathbf{Q} = \begin{bmatrix} -0.4851 & 0.5659 & -0.6667 \\ -0.4851 & -0.8085 & -0.3333 \\ -0.7276 & 0.1617 & 0.6667 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} -2.0616 & -2.9104 & -4.8507 \\ 0 & -0.7276 & -1.2127 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

Rozwiązywanie układu równań wyznacza się w dwóch krokach:

- oblicza wektor $\mathbf{y} = \mathbf{Q}^T \mathbf{b}$;
- rozwiązuje układ równań $\mathbf{R}\mathbf{x} = \mathbf{y}$ względem wektora \mathbf{x} .

Daje to: $\mathbf{y} = [-1.6977 \ -0.0808 \ -0.3333]^T$

oraz $\mathbf{x} = [0.6667 \ 0.6667 \ -0.3333]^T$.

Zalety przedstawionego sposobu rozwiązywania równań algebraicznych są szczególnie widoczne w sytuacji, gdy układ (4.1) o niezmienionej macierzy \mathbf{A} należy rozwiązać dla wielu wersji wektora \mathbf{b} ; wówczas bowiem rozkład QR jest wykonywany tylko raz, a dla każdej wersji tego wektora powtarzane są jedynie opisane operacje, charakteryzujące się niewielką złożonością obliczeniową.

Przykład 4.8. W wyniku wstępnych badań statystycznych stwierdzono, że wpływ temperatury a_1 , ciśnienia atmosferycznego a_2 i wilgotności względnej a_3 na wydajność b pewnego procesu produkcyjnego można, w małym zakresie zmienności tych wielkości, z wystarczającą dokładnością dla potrzeb sterowania procesem produkcyjnym, modelować zależnością liniową:

$$b = x_1 a_1 + x_2 a_2 + x_3 a_3$$

Ze względów technologicznych pomiary wydajności mogą być wykonywane jedynie dla trzech układów wartości wielkości a_1 , a_2 i a_3 :

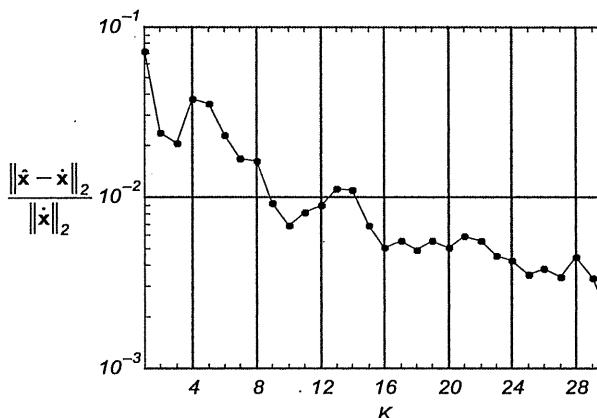
$$\tilde{\mathbf{A}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}, \quad \tilde{\mathbf{b}} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

przy czym pomiary a_1, a_2, a_3 i b mogą być wykonane z dokładnością nie większą niż 1%. Należy zbadać wpływ krotności powtarzania pomiarów K na osiąganą dokładność estymacji parametrów x_1, x_2 i x_3 , zakładając, że macierz układu równań i wektor prawych stron równań – wiążących wektor parametrów z wynikami pomiarów – mają postać:

$$\tilde{\mathbf{A}} = \begin{bmatrix} \tilde{\mathbf{A}}_1 \\ \tilde{\mathbf{A}}_2 \\ \dots \\ \tilde{\mathbf{A}}_K \end{bmatrix}, \quad \tilde{\mathbf{b}} = \begin{bmatrix} \tilde{\mathbf{b}}_1 \\ \tilde{\mathbf{b}}_2 \\ \dots \\ \tilde{\mathbf{b}}_K \end{bmatrix}$$

przy czym $\tilde{\mathbf{A}}_K$ różnią się od $\tilde{\mathbf{A}}$ tylko błędami pomiarów; podobnie $\tilde{\mathbf{b}}_K$ różnią się od $\tilde{\mathbf{b}}$.

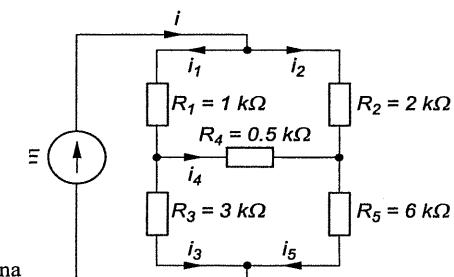
W celu wyznaczenia zależności błędu rozwiązania $\hat{\mathbf{x}}$ od liczby pomiarów $M = 3K$ wyznaczono najpierw rozwiązanie układu niezaburzonego ($\hat{\mathbf{x}}$), a następnie dokonano symulacji danych zaburzonych $\tilde{\mathbf{A}}$ i $\tilde{\mathbf{b}}$, stosując generatory liczb pseudolosowych (o których mowa w rozdziale 9) do symulacji błędów pomiaru wielkości a_1, a_2, a_3 i b . Równania normalne dla każdej wartości K rozwiązano metodą rozkładu QR przy użyciu środowiska MATLAB. Wyniki obliczeń przedstawiono na rys. 4.4.



Rysunek 4.4. Względny błąd wyznaczenia wektora rozwiązań w funkcji liczby równań, $M = 3K$

Jak widać, błędy wyznaczenia współczynników zmniejszają się o rzad wielkości, gdy liczba powtórzeń K wzrasta od 2 do 30.

Zadanie 4.1. Wyznaczyć wartości prądów w gałęziach sieci elektrycznej przedstawionej na rys. 4.5. Zbadać wpływ jednoprocentsowego rozrzutu wartości poszczególnych rezystancji na rozrzut wartości tych prądów. Określić współczynniki przenoszenia błędów rezystancji R_i na względne zmiany wartości prądu i_4 oraz na względne zmiany normy wektora prądów i_1, \dots, i_5 .



Rysunek 4.5. Liniowa sieć elektryczna

4.5. PODSUMOWANIE

Syntetyczne porównanie właściwości najważniejszych metod rozwiązywania układów równań liniowych przedstawiono w tablicy 4.2. Wynika z niego, że do rozwiązywania dobrze uwarunkowanych układów równań najlepiej jest stosować metodę LU. Gorzej uwarunkowane oraz nadokreślone układy równań rozwiązuje się zazwyczaj metodami QR lub SVD. Czas rozwiązywania układów równań z tzw. macierzami pełnymi (tzn. takimi, dla których większość elementów macierzy A jest różna od zera), za pomocą wymienionych metod nieiteracyjnych, jest proporcjonalny do N^3 (gdzie N jest rozmiarem układu równań). Z tego względu maksymalny rozmiar rozwiązywanych tymi metodami układów nie przekracza zazwyczaj kilku tysięcy. Bardzo duże układy równań liniowych powstają w wyniku zastosowanie numerycznych metod rozwiązywania równań różniczkowych cząstkowych. Dotyczy to w szczególności metod różnic skończonych i elementów skończonych. Struktura powstających w wyniku zastosowania tych metod macierzy jest zazwyczaj pasmowa i są to tzw. macierze rzadkie (tzn. niewielki procent ich elementów jest różny od zera). Są to zwykle macierze o niewielkich wskaźnikach uwarunkowania. Do rozwiązywania takich układów równań stosuje się metody iteracyjne, takie jak SOR i metoda gradientów sprzężonych.

Tablica 4.2

Właściwości najważniejszych metod rozwiązywania układów równań liniowych

METODA	ZNACZENIE	ZALETY	WADY	ZASTOSOWANIA
eliminacji Gaussa	stosowana do uzyskania rozkładu LU	prostota	kosztowna dla wielu równań o identycznych macierzach	samodzielnie dla pojedynczych układów równań
rozkład LU	duże	niski koszt obliczeń	możliwa niestabilność dla źle uwarunkowanych układów równań	podstawowy algorytm dla dobrze uwarunkowanych układów równań
rozkład QR	duże	stabilność numeryczna	wyższy koszt niż LU	układy źle uwarunkowane i układy nadokreślone
rozkład SVD	duże	stabilność numeryczna	wyższy koszt niż QR	układy bardzo źle uwarunkowane oraz układy nad- i podokreślone
SOR	średnie	niski koszt obliczeń	wymagana znajomość współczynnika nadrelaksacji, ograniczenie do szczególnej klasy równań	algorytm iteracyjny rozwiązywania układów równań dla macierzy symetrycznych, dodatnio określonych
gradientów sprzężonych	duże	nie wymaga znajomości żadnych parametrów	ograniczenie do szczególnej klasy równań	algorytm iteracyjny rozwiązywania układów równań dla macierzy symetrycznych, dodatnio określonych

ROZWIĄZYwanIE NIELINIOWYCH RÓWNAŃ ALGEBRAICZNYCH



Pojedyncze nieliniowe równania algebraiczne i układy nieliniowych równań algebraicznych pojawiają się bardzo często w praktyce inżynierskiej, gdyż są one podstawową formą matematycznego opisu wielu zagadnień fizycznych albo wynikiem „algebraizacji” równań różniczkowych. W praktyce inżyniera elektronika równania nieliniowe pojawiają się między innymi:

- w opisie układów elektronicznych zawierających elementy nieliniowe bierne, takie jak diody, oraz elementy aktywne, takie jak tranzystory;
- w analizie obwodów magnetycznych, takich jak magnesy, transformatory, dławiki oraz obwody magnetyczne maszyn elektrycznych;
- w wyniku „algebraizacji” równań różniczkowych cząstkowych, opisujących różnego rodzaju urządzenia mikrofalowe, takie jak rezonatory z niejednorodnym wypełnieniem, cyrkulatory czy przesuwniuki fazy.

Ogólnie, układ nieliniowych równań algebraicznych ma postać:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \quad (5.1)$$

gdzie $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}) \ f_2(\mathbf{x}) \dots \ f_N(\mathbf{x})]^T$ jest N -wymiarową rzeczywistą lub zespoloną funkcją N zmiennych (rzeczywistych lub zespolonych) x_1, x_2, \dots, x_N , tzn. funkcją wektora: $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_N]^T$. Zakładamy dalej, że funkcja ta jest określona w pewnym podzbiorze przestrzeni oraz że rozwiązaniem równania (5.1) jest wektor α .

5.1. ROZWIĄZYwanIE SKALARNYCH RÓWNAŃ NIELINIOWYCH

5.1.1. METODA BISEKCJI

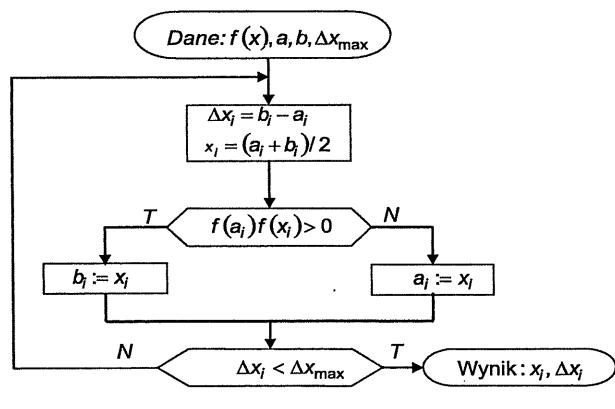
Niech $f(x)$ będzie rzeczywistą skalarną funkcją zmiennej skalarnej, ciągłą w przedziale $[a, b]$ i przyjmującą różne znaki na krańcach tego przedziału.

Metoda bisekcji polega na konstrukcji ciągu przybliżonych wartości rozwiązania $\{x_i\}$ oraz ciągu przedziałów $\{[a_i, b_i]\}$ zdefiniowanych następująco:

$$x_i := \frac{1}{2}(a_i + b_i) \quad (5.2)$$

$$[a_i, b_i] := \begin{cases} [a_i, x_i], & \text{gdy } f(a_i)f(x_i) \leq 0 \\ [x_i, b_i], & \text{gdy } f(a_i)f(x_i) > 0 \end{cases} \quad (5.3)$$

dla $i = 0, 1, \dots$. Zauważmy, że tak skonstruowany ciąg kolejnych przybliżeń $\{x_i\}$ dąży do wartości $x = \alpha$, dla której funkcja $f(x)$ jest równa zeru, przy czym miara bezwzględnego błędu wyznaczenia tej wartości jest różnicą $\Delta x_i = b_i - a_i$. Metoda bisekcji jest zbieżna liniowo (wykładnik lokalnej zbieżności $\rho = 1$, współczynnik lokalnej zbieżności $C = 0.5$), przy czym zbieżność ma miejsce dla wszystkich rzeczywistych funkcji ciągłych w przedziale $[a, b]$ i przyjmujących na jego krańcach różne znaki. Sieć działań, ułatwiającą zaprogramowanie algorytmu rozwiązywania równania nieliniowego metodą bisekcji, przedstawiono na rys. 5.1.



Rysunek 5.1. Sieć działań dla metody bisekcji

Dla metody bisekcji łatwo można wyznaczyć liczbę iteracji I wystarczających do uzyskania błędu bezwzględnego nieprzekraczającego Δx_{\max} . Początkowa wartość tego błędu jest nie większa niż długość przedziału $[a, b]$, a po każdej kolejnej iteracji błąd ten maleje dwukrotnie, więc:

$$I = \log_2 \frac{|b - a|}{\Delta x_{\max}}$$

5.1.2. METODA STYCZNYCH (NEWTONA) I METODA SIECZNYCH

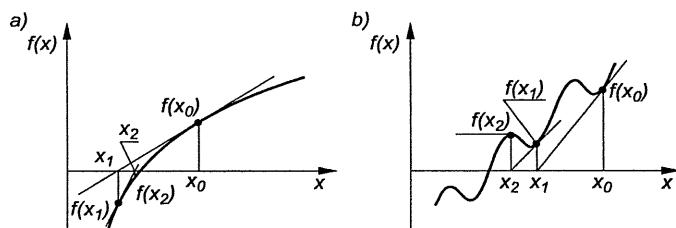
Metoda Newtona jest metodą iteracyjną, która polega na wyznaczeniu przybliżenia x_{i+1} miejsca zerowego α funkcji $f(x)$ jako punktu przecięcia stycznej do wykresu tej funkcji w punkcie x_i z osią odciętych:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad \text{dla } i = 0, 1, \dots \quad (5.4)$$

Jest to zilustrowane graficznie na rys. 5.2a. Metoda Newtona jest lokalnie zbieżna kwadratowo ($\rho = 2$) ze współczynnikiem zbieżności:

$$C = -\frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)} \quad (5.5)$$

Jej zbieżność jest więc zazwyczaj szybsza od metody bisekcji dla funkcji dostatecznie regularnych w otoczeniu miejsca zerowego. Bywają jednak funkcje (przykład na rys. 5.2b), dla których metoda Newtona nie jest zbieżna.



Rysunek 5.2. Przykład funkcji, dla której metoda Newtona; a) jest zbieżna, b) nie jest zbieżna

Zastępując pochodną $f'(x_i)$ we wzorze (5.4) ilorazem różnicowym $\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$, otrzymuje się definicję metody siecznych:

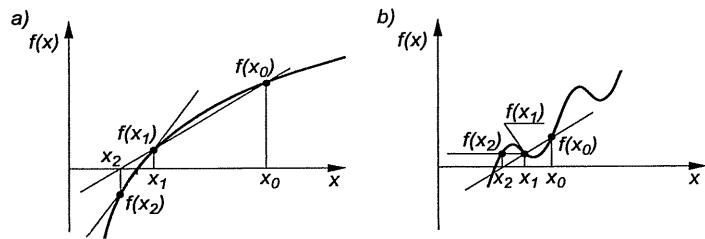
$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \quad \text{dla } i = 1, 2, \dots \quad (5.6)$$

Kolejne przybliżenie wyznaczone metodą siecznych jest punktem przecięcia osi odciętych z sieczną wykresu funkcji $f(x)$, poprowadzoną przez punkty x_{i-1} i x_i , co pokazano na rys. 5.3a.

Parametry zbieżności metody siecznych wyznaczono w przykładzie 3.8. Są one następujące:

$$\rho = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618 \quad \text{oraz} \quad C = \left[\frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)} \right]^{\frac{1}{2}(\sqrt{5}-1)} \quad (5.7)$$

Lokalna zbieżność metody siecznych jest więc dla dostatecznie regularnych funkcji szybsza od zbieżności metody bisekcji (dla której $\rho = 1$), lecz nieco wolniejsza niż dla metody Newtona (dla której $\rho = 2$). Podobnie jak metoda Newtona, metoda siecznych może nie być zbieżna dla niektórych funkcji, np. takiej jak przedstawiona na rys. 5.3b.



Rysunek 5.3. Przykład funkcji, dla której metoda siecznych: a) jest zbieżna, b) nie jest zbieżna

Oprócz wersji podstawowej, zdefiniowanej wzorem (5.6), znane są następujące uproszczone wersje metody siecznych:

- metoda *regula falsi*, polegająca na zastąpieniu punktu x_{i-1} ustaloną liczbą c :

$$x_{i+1} = x_i - \frac{x_i - c}{f(x_i) - f(c)} f(x_i) \quad \text{dla } i = 0, 1, \dots \quad (5.8)$$

- metoda *Steffensena*, powstająca w wyniku zastąpienia punktu x_{i-1} punktem $x_i + f(x_i)$:

$$x_{i+1} = x_i - \frac{f(x_i)}{f(x_i + f(x_i)) - f(x_i)} f(x_i) \quad \text{dla } i = 0, 1, \dots \quad (5.9)$$

Metoda *regula falsi* jest zbieżna liniowo ($\rho = 1$), jeżeli liczba c leży dostatecznie blisko miejsca zerowego, natomiast metoda Steffensena ma wykładnik lokalnej zbieżności $\rho = 2$, lecz złe własności numeryczne; dlatego obie te metody są rzadko stosowane w praktyce.

Porównując opisane dotychczas metody, można stwierdzić, że dla dostatecznie regularnych funkcji metoda Newtona i metoda siecznych są szybciej zbieżne od metody bisekcji, jednak ta ostatnia jest bardziej „niezawodna”, gdyż dla każdej funkcji ciągłej zapewnia dwukrotny wzrost dokładności rozwiązania przybliżonego w kolejnych iteracjach. Istnieją metody łączące niezawodność metody bisekcji z szybkością zbieżności charakterystyczną dla metod o wykładniku lokalnej zbieżności $\rho > 1$. Przykładem jest metoda Brendta, której opis można znaleźć w literaturze [F2].

Przykład 5.1. Równanie $x^2 - x - 2 = 0$ rozwiązyano kolejno metodą bisekcji, metodą Newtona i metodą siecznych; jako punkt początkowy w metodzie Newtona przyjęto $x_0 = 3$, zaś w metodach siecznych i bisekcji: $x_0 = 3$ oraz $x_1 = 1.5$. Wyznaczono zależność względnego błędu rozwiązania w funkcji liczby iteracji. Przedstawione w tablicy 5.1 wyniki obliczeń potwierdzają stwierdzone wcześniej prawidłowości dotyczące szybkości zbieżności poszczególnych metod.

Tablica 5.1

Porównanie zbieżności (błąd względny wyznaczenia pierwiastka $\alpha = 2$) dla trzech metod rozwiązywania równań nieliniowych

METODA	LICZBA ITERACJI					
	1	2	3	4	5	6
Bisekcji	$1.25 \cdot 10^{-1}$	$6.25 \cdot 10^{-2}$	$3.13 \cdot 10^{-2}$	$1.56 \cdot 10^{-2}$	$7.81 \cdot 10^{-3}$	$3.91 \cdot 10^{-3}$
Siecznych	$7.14 \cdot 10^{-2}$	$1.52 \cdot 10^{-2}$	$7.50 \cdot 10^{-4}$	$7.50 \cdot 10^{-6}$	$3.75 \cdot 10^{-9}$	$1.88 \cdot 10^{-14}$
Newtona	$1.00 \cdot 10^{-1}$	$5.88 \cdot 10^{-3}$	$2.29 \cdot 10^{-5}$	$3.49 \cdot 10^{-10}$	<i>eps</i>	<i>eps</i>

Zadanie 5.1. Częstotliwości f_n ($n = 1, 2, \dots$) drgań własnych rodzajów TE_{0,1,n} cylindrycznego rezonatora wnęgowego o promieniu a i długości L – wypełnionego dielektrykiem o grubości h i stałej dielektrycznej ϵ_r w całym przekroju poprzecznym, przylegającym do jednego z denek – spełniają układ równań:

$$\frac{\operatorname{tg}(kh)}{k} + \frac{\operatorname{tg}(k_0(L-h))}{k_0} = 0$$

$$k^2 = \frac{(2\pi f)^2}{c^2} \epsilon_r - \left(\frac{3.83171}{a} \right)^2$$

$$k_0^2 = \frac{(2\pi f)^2}{c^2} - \left(\frac{3.83171}{a} \right)^2$$

gdzie $c = 3 \cdot 10^8$ m/s jest prędkością światła w próżni. Wyznaczyć kilka najmniejszych częstotliwości f_n , sprowadzając ten układ do jednego równania względem częstotliwości f . Obliczenia przeprowadzić dla: $a = 25$ mm, $L = 25$ mm, $h = 3$ mm, $\epsilon_r = 10$.

Odp.: $f_1 = 8.1668$ GHz, $f_2 = 10.634$ GHz, $f_3 = 15.525$ GHz.

5.1.3. METODY WYZNACZANIA ZER WIELOMIANÓW

Każdy wielomian algebraiczny:

$$P(x) = \sum_{n=0}^N a_n x^n \quad (5.10)$$

którego współczynniki a_n są liczbami rzeczywistymi, może mieć zera pojedyncze i wielokrotne, przy czym mogą one być zarówno rzeczywiste, jak zespolone. Do wyznaczenia pojedynczych zer wielomianów stosowane są metody o dużym wykładowisku zbieżności, takie jak iteracyjna metoda Mullera opierająca się na aproksymacji funkcji $f(x)$ w otoczeniu jej miejsca zerowego funkcją kwadratową:

$$x_{i+1} = x_i - \frac{2f(x_i)}{f'(x_i) + \operatorname{sgn}(f'(x_i)) \sqrt{(f'(x_i))^2 - 2f(x_i)f''(x_i)}} \quad \text{dla } i = 0, 1, \dots \quad (5.11)$$

Wykładowik lokalnej zbieżności tej metody wynosi $\rho = 3$. W przypadku wielomianów bardzo często zachodzi potrzeba wyznaczenia nie jednego zera lecz wszystkich. Aby uniknąć wielokrotnego wyznaczania tego samego zera, eliminuje się je z wielomianu zaraz po jego wyznaczeniu. Operacja ta nazywa się *deflakcją*.

W przypadku zera rzeczywistego deflakcja polega na obniżeniu stopnia wielomianu $P(x)$ przez podzielenie go przez czynnik liniowy $(x - \alpha)$. Współczynniki wielomianu:

$$Q(x) = \sum_{n=0}^{N-1} b_n x^n \quad (5.12)$$

będącego ilorazem wielomianu $P(x)$ i czynnika liniowego $(x - \alpha)$:

$$P(x) = Q(x)(x - \alpha) \quad (5.13)$$

można wyznaczyć za pomocą algorytmu Hornera:

$$b_{N-1} = a_N \quad \text{oraz} \quad b_{n-1} = a_n + \alpha b_n \quad \text{dla } n = N-1, \dots, 1 \quad (5.14)$$

W przypadku ogólnym wielomian o współczynnikach rzeczywistych może być rozłożony na czynniki rzeczywiste pierwszego i drugiego stopnia. Wówczas deflakcję przeprowadza się czynnikami drugiego stopnia (czynniki pierwszego stopnia można wyznaczyć później, znając wszystkie czynniki stopnia drugiego). Algorytm deflakcji czynnikiem $x^2 + ux + v$ wyprowadza się w następujący sposób:

Dzielenie wielomianu $P(x)$ przez dwumian $x^2 + ux + v$ daje wielomian:

$$Q(x) = \sum_{n=0}^{N-2} b_n x^n \quad (5.15)$$

i resztę $cx + d$, a zatem:

$$P(x) = (x^2 + ux + v) \left(\sum_{n=0}^{N-2} b_n x^n \right) + c(u, v)x + d(u, v) \quad (5.16)$$

Współczynniki b_n są również funkcjami u i v i mogą być wyznaczone według następującego schematu:

$$b_n = a_{n+2} - ub_{n+1} - vb_{n+2} \quad \text{dla } n = N-2, N-3, \dots, 0 \quad (5.17)$$

Aby trójmian $x^2 + ux + v$ był dzielnikiem wielomianu, reszta z dzielenia:

$$c(u, v) = a_1 - ub_0 - vb_1$$

$$d(u, v) = a_0 - vb_0$$

musi być równa零, co prowadzi do następującego układu dwóch równań nieliniowych ze względu na współczynniki u i v :

$$\begin{cases} c(u, v) = 0 \\ d(u, v) = 0 \end{cases} \quad (5.18)$$

Do rozwiązania tego układu stosuje się *metodę Bairstowa*, która jest szczególnym przypadkiem dwuwymiarowej metody Newtona, omówionej szczegółowo w podrozdziale 5.2.1:

$$\begin{bmatrix} u_{i+1} \\ v_{i+1} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix} - \begin{bmatrix} \frac{\partial c}{\partial u} & \frac{\partial c}{\partial v} \\ \frac{\partial d}{\partial u} & \frac{\partial d}{\partial v} \end{bmatrix}^{-1} \begin{bmatrix} c(u_i, v_i) \\ d(u_i, v_i) \end{bmatrix} \quad \text{dla } i = 0, 1, \dots \quad (5.19)$$

Po wyznaczeniu współczynników u i v przeprowadza się deflakcję czynnikiem stopnia drugiego, redukując tym samym stopień wielomianu o dwa. W kolejnym kroku stosuje się metodę Bairstowa do zredukowanego wielomianu w celu dokonania kolejnej deflakcji. Proces ten kontynuuje się tak długo, aż uzyska się wielomian stopnia drugiego lub pierwszego. W ten sposób można uzyskać pełny rozkład wielomianu na czynniki stopnia nie wyższego niż drugi. Pierwiastki wielomianu wyznacza się następnie jako zera (być może zespolone) wszystkich czynników stopnia drugiego; w przypadku wielomianu nieparzystego stopnia jedno z zer rzeczywistych jest zerem czynnika stopnia pierwszego.

Przykład 5.2. Wyznaczyć wszystkie pierwiastki zespolone równania $x^4 + 1 = 0$, dokonując jego rozkładu na czynniki stopnia drugiego metodą Bairstowa, a następnie wyznaczając zera czynników stopnia drugiego, korzystając ze wzorów na pierwiastki równania kwadratowego.

Dzielenie wielomianu $P(x) = x^4 + 1$ przez dwumian $x^2 + ux + v$ daje następujący układ równań nieliniowych względem współczynników u i v :

$$2uv - u^3 = 0$$

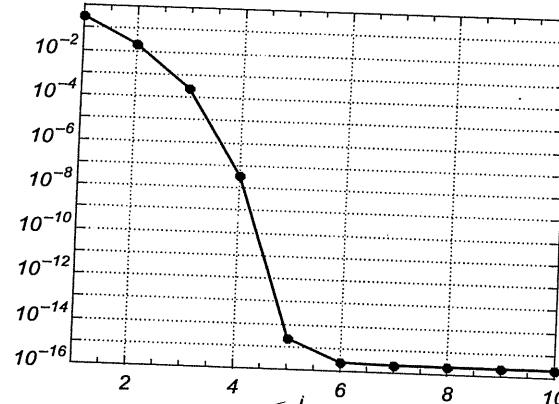
$$v^2 - u^2v + 1 = 0$$

Dla tych danych wzór (5.19) przyjmuje następującą postać:

$$\begin{bmatrix} u_{i+1} \\ v_{i+1} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix} - \begin{bmatrix} -3u_i^2 + 2v_i & 2u \\ -2uv & 2v - u^2 \end{bmatrix}^{-1} \begin{bmatrix} 2u_i v_i - u_i^3 \\ v_i^2 - u_i^2 v_i + 1 \end{bmatrix} \quad \text{dla } i = 0, 1, \dots$$

Zbieżność tego procesu iteracyjnego zależy od wyboru punktu startowego. Dla $u_0 = 1$ i $v_0 = 2$ jest on zbieżny do $u_\infty = \sqrt{2}$ i $v_\infty = 1$ w sposób przedstawiony na rys. 5.4. Jak widać, błąd na poziomie reprezentacji liczb zmiennoprzecinkowych uzyskuje się po ośmiu iteracjach. Dla $u_0 = -2$ i $v_0 = 2$ proces iteracyjny jest zbieżny do $u_\infty = -\sqrt{2}$ i $v_\infty = 1$. Oznacza to, że wielomian $P(x)$ może być przedstawiony w postaci następującego iloczynu dwóch czynników kwadratowych:

$$P(x) = x^4 + 1 = (x^2 - \sqrt{2}x + 1)(x^2 + \sqrt{2}x + 1)$$



Rysunek 5.4. Zależność normy błędu bezwzględnego $\Delta_i = \sqrt{(u_i - u_\infty)^2 + (v_i - v_\infty)^2}$ od liczby iteracji i dla punktu startowego $u_0 = 1$, $v_0 = 2$

Obydwa czynniki kwadratowe mają jedynie pierwiastki zespolone:

$$x_1 = -\frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}, \quad x_2 = -\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}, \quad x_3 = \frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}, \quad x_4 = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$$

Alternatywny sposób wyznaczania zer wielomianów wynika ze spostrzeżenia, że wartości własne macierzy A są tożsame z zerami jej wielomianu charakterystycznego $P(\lambda) = \det(A - \lambda I)$. Znając współczynniki wielomianu $P(x)$, można utworzyć tzw. macierz stwarzoszoną C (ang. *companion matrix*), której wartości własne są zerami tego wielomianu:

$$\mathbf{C} = \begin{bmatrix} -\frac{a_{N-1}}{a_N} & -\frac{a_{N-2}}{a_N} & \dots & \dots & -\frac{a_1}{a_N} & -\frac{a_0}{a_N} \\ a_N & a_N & \dots & \dots & a_N & a_N \\ 1 & 0 & \dots & \dots & 0 & 0 \\ 0 & 1 & \dots & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 1 & 0 \end{bmatrix} \quad (5.20)$$

a następnie wyznaczyć wartości własne tej macierzy. Tak, między innymi, skonstruowana jest procedura **roots** w środowisku MATLAB. Dla rozpatrywanego w przykładzie 5.2 równania macierz C przyjmuje postać:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Jej wartości własne, wyznaczone za pomocą służącej do wyznaczania wartości własnych macierzy procedury **eig** (która jest wywoływana wewnątrz procedury **roots**), to:

$$\lambda_1 = -\frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}, \quad \lambda_2 = -\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}, \quad \lambda_3 = \frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}, \quad \lambda_4 = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$$

5.2. ROZWIĄZYWANIE UKŁADÓW RÓWNAŃ NIELINIOWYCH

5.2.1. WIELOWYMIAROWA METODA NEWTONA

W wielowymiarowej metodzie Newtona układ równań nieliniowych o postaci (5.1) jest rozwiązywany przy założeniu, że znane jest przybliżenie początkowe miejsca zerowego α , zapewniające zbieżność lokalną algorytmu. Aby to miało miejsce, funkcja wektorowa $\mathbf{f}(\mathbf{x})$ musi być dostatecznie regularna w zawierającym to przybliżenie otoczeniu miejsca zerowego. Kolejne iteracje wielowymiarowej metody Newtona są zdefiniowane zależnością formalnie analogiczną do (5.4):

$$\mathbf{x}_{i+1} = \mathbf{x}_i - [\mathbf{f}'(\mathbf{x}_i)]^{-1} \mathbf{f}(\mathbf{x}_i) \quad \text{dla } i = 0, 1, \dots \quad (5.21)$$

gdzie

$$\mathbf{f}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \dots & \dots & \frac{\partial f_1}{\partial x_N}(\mathbf{x}) \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1}(\mathbf{x}) & \dots & \dots & \frac{\partial f_N}{\partial x_N}(\mathbf{x}) \end{bmatrix} \quad (5.22)$$

Oznacza to, że w celu wykonania jednej iteracji należy rozwiązać układ równań liniowych:

$$[\mathbf{f}'(\mathbf{x}_i)]\Delta\mathbf{x}_i = -\mathbf{f}(\mathbf{x}_i) \quad (5.23)$$

względem $\Delta\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$, a następnie wyznaczyć $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}_i$. Każda iteracja wielowymiarowej metody Newtona wymaga więc wykonania następujących czynności:

- wyznaczenia wektora wartości funkcji $\mathbf{f}(\mathbf{x}_i)$;
- wyznaczenia macierzy pochodnych $\mathbf{f}'(\mathbf{x}_i)$;
- rozwiązania układu równań liniowych (5.23);
- obliczenia $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}_i$.

Zakończenie obliczeń ma miejsce wtedy, gdy norma poprawki $\|\Delta\mathbf{x}_i\|$ staje się dostatecznie mała lub jeżeli następuje jej wzrost w kolejnych krokach algorytmu. Podobnie jak w przypadku skalarnym, wykładnik lokalnej zbieżności wielowymiarowej metody Newtona $p = 2$.

5.2.2. WIELOWYMIAROWA METODA SIECZNYCH

Układ równań nieliniowych o postaci (5.1) jest rozwiązywany metodą siecznych przy założeniu, że znane są przybliżenia początkowe miejsca zerowego, zapewniające zbieżność lokalną algorytmu. W wielowymiarowej metodzie siecznych kolejne przybliżenia rozwiązania układu równań (5.1) wynikają z zależności:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \Delta\mathbf{X}_i [\Delta\mathbf{F}_i]^{-1} \mathbf{f}(\mathbf{x}_i) \quad \text{dla } i = N, N+1, \dots \quad (5.24)$$

gdzie $\Delta\mathbf{X}_i$ jest macierzą o kolumnach postaci:

$$\mathbf{x}_{i-N+1} - \mathbf{x}_{i-N}, \mathbf{x}_{i-N+2} - \mathbf{x}_{i-N+1}, \dots, \mathbf{x}_i - \mathbf{x}_{i-1}$$

zaś $\Delta\mathbf{F}_i$ jest macierzą o kolumnach postaci:

$$\mathbf{f}(\mathbf{x}_{i-N+1}) - \mathbf{f}(\mathbf{x}_{i-N}), \mathbf{f}(\mathbf{x}_{i-N+2}) - \mathbf{f}(\mathbf{x}_{i-N+1}), \dots, \mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_{i-1})$$

Wzór (5.24) definiuje tzw. $(N+1)$ -punktową metodę siecznych, gdyż do wyznaczenia każdego kolejnego przybliżenia niezbędna jest znajomość wartości funkcji $\mathbf{f}(\mathbf{x})$ dla $N+1$ uprzednio wyznaczonych przybliżeń \mathbf{x}_i . Nietrudno zauważyc, że przyrost rozwiązania w kolejnej iteracji spełnia równanie analogiczne do (5.23):

$$\Delta\mathbf{F}_i [\Delta\mathbf{X}_i]^{-1} \Delta\mathbf{x}_i = -\mathbf{f}(\mathbf{x}_i) \quad (5.25)$$

w którym $\Delta\mathbf{F}_i [\Delta\mathbf{X}_i]^{-1}$ jest różnicową aproksymacją macierzy pochodnych występującą we wzorze (5.23). Każdy krok wielowymiarowej metody siecznych wymaga zatem wykonania następujących czynności:

- wyznaczenia wektora $\mathbf{f}(\mathbf{x}_i)$;
- wyznaczenia macierzy $\Delta\mathbf{F}_i$ na podstawie $\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}_{i-1}), \dots, \mathbf{f}(\mathbf{x}_{i-N})$;
- rozwiązania układu równań liniowych $\Delta\mathbf{F}_i \mathbf{v} = \mathbf{f}(\mathbf{x}_i)$ względem \mathbf{v} ;
- obliczenia $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}_i$.

5.2.3. UWAGI PRAKTYCZNE

Opisane w podrozdziałach 5.2.1 i 5.2.2 metody rozwiązywania nieliniowych równań algebraicznych są zbieżne pod warunkiem właściwego wyboru początkowego przybliżenia zera funkcji $\mathbf{f}(\mathbf{x})$. Zbieżne globalnie metody wyznaczania zer takich funkcji związane są ze zbieżnymi globalnie metodami poszukiwania minimum funkcji wielu zmiennych. Zauważmy bowiem, że równość (5.1) implikuje równość:

$$J(\mathbf{x}) \equiv \|\mathbf{f}(\mathbf{x})\|^2 = 0 \quad (5.26)$$

Zera funkcji $\mathbf{f}(\mathbf{x})$ są zatem także minimami funkcji $J(\mathbf{x})$, ale niekoniecznie na odwrót, gdyż wśród minimów funkcji $J(\mathbf{x})$ mogą istnieć takie, które nie są zerami $\mathbf{f}(\mathbf{x})$. Zbieżne globalnie metody poszukiwania minimum funkcji wielu zmiennych, mogą więc być stosowane do wyznaczania zer układów równań nieliniowych. W praktyce inżynierskiej korzysta się często z gotowych procedur służących do tego celu. W środowisku MATLAB do rozwiązywania skalarnych równań nieliniowych służy procedura o nazwie **fzero** (metoda Brendta), a do rozwiązywania układów równań nieliniowych – procedura **fsolve** realizująca metodę optymalizacji Gaussa-Newtona lub Levenberga-Marquardta. Używając tych procedur szczególną uwagę należy zwrócić na odpowiedni wybór przybliżeń początkowych. Równania nieliniowe oraz ich układy często mają wiele rozwiązań; ich liczba może być nawet nieskończona i wszystkie mogą opisywać realne zagadnienie fizyczne. Przykładem są równania opisujące zagadnienia rezonansu pól elektromagnetycznych w rezonatorach mikrofalowych. Analizując sieci elektryczne z elementami nieliniowymi już na etapie formułowania problemu możemy zmniejszyć liczbę równań, eliminując niektóre z niewiadomych. Prowadzi to zwykle do zwiększenia dokładności obliczeń i często pozwala na uniknięcie problemów związanych z wyborem przybliżenia początkowego.

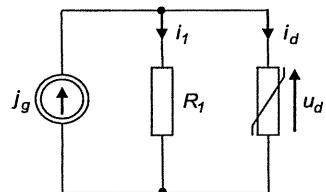
Przykład 5.3. Wyznaczyć prądy i napięcia w układzie przedstawionym na rys. 5.5. Elementem nieliniowym w tym układzie jest dioda tunelowa o charakterystyce statycznej:

$$i_d = f(u_d) = ku_d \left(\frac{u_d^2}{3} - 1.5u_d^2 + 2 \right)$$

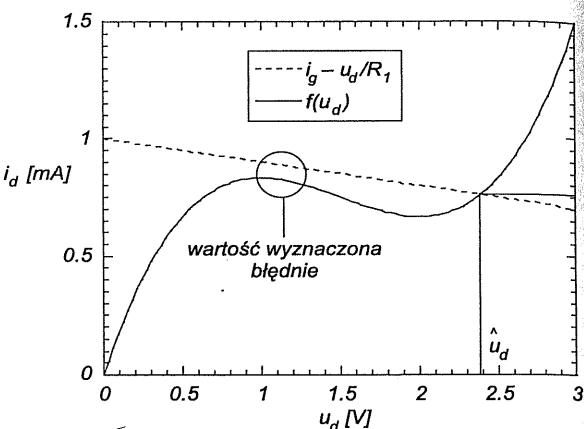
gdzie $k = 1 \text{ mA/V}^3$. Założyć, że $j_g = 1 \text{ mA}$ oraz $R_i = 10 \text{ k}\Omega$.

Prąd i_d i napięcie u_d można wyznaczyć, rozwiązując układ równań:

$$\begin{cases} i_d = ku_d \left(\frac{u_d^2}{3} - 1.5u_d + 2 \right) \\ j_g = \frac{u_d}{R_1} + i_d \end{cases} \quad (5.27)$$



Rysunek 5.5. Układ z diodą tunelową



Rysunek 5.6. Graficzne rozwiązanie układu równań (5.27); wykres charakterystyki $f(u_d)$ (linia ciągła) oraz wykres $j_g - \frac{u_d}{R_1}$ (linia przerywana)

przy użyciu odpowiedniej procedury numerycznej. Lepiej jednak najpierw podstawić prawą stronę pierwszego równania zamiast i_d w drugim równaniu, a następnie rozwiązać pojedyncze równanie nieliniowe. Rozwiązanie wyznaczone za pomocą procedury **fzero**

$$\hat{u}_d = 2.3877 \text{ V}, \quad \hat{i}_d = 0.7612 \text{ mA}$$

praktycznie nie zależy od wyboru początkowej wartości napięcia. Rozwiązanie układu równań przy użyciu procedury **fsolve** daje taki sam wynik, ale przy wyborze wartości początkowych napięcia u_d i prądu i_d położonych dostatecznie blisko wartości dokładnej. Wybór $u_{d,0}$ w zakresie 0–1.4 V oraz $i_{d,0}$ w zakresie 0–1.0 mA prowadzi do uzyskania fałszywego „rozwiązania”:

$$u_d \approx 1.1127 \text{ V} \quad i \quad i_d \approx 0.8581 \text{ mA}$$

będącego lokalnym minimum funkcji:

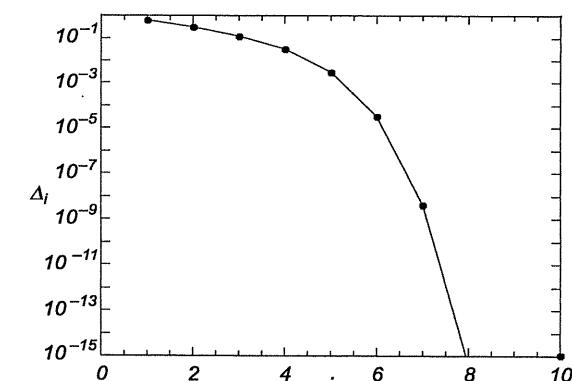
$$J(i_d, u_d) = \left[i_d - ku_d \left(\frac{u_d^2}{3} - 1.5u_d + 2 \right) \right]^2 + \left(j_g - \frac{u_d}{R_1} - i_d \right)^2$$

Graficzne rozwiązanie układu równań przedstawiono na rys. 5.6.

Przykład 5.4. Na rys. 5.7 przedstawiono zależność błędu rozwiązywania układu równań (5.27) metodą Newtona od liczby iteracji. Jako przybliżenie początkowe przyjęto:

$$u_d^{(0)} = 5 \text{ V} \quad i \quad i_d^{(0)} = 0 \text{ mA}$$

Z przedstawionego wykresu wynika, że dopiero w bardzo bliskim otoczeniu rozwiązania następuje istotne przyspieszenie zbieżności.

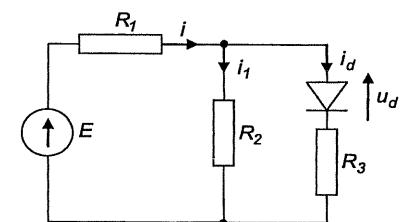


Rysunek 5.7. Zależność błędu rozwiązywania układu równań (5.27) metodą Newtona od liczby iteracji

Zadanie 5.2. Wyznaczyć wartości prądów w układzie przedstawionym na rys. 5.8 dla $E = 5 \text{ V}$, $R_1 = 0.1 \text{ k}\Omega$, $R_2 = 1 \text{ k}\Omega$ oraz: a) $R_3 = 0.1 \text{ k}\Omega$, b) $R_3 = 0 \text{ k}\Omega$. Założyć, że charakterystyka statyczna diody ma postać:

$$i_d = i_s \left\{ \exp \left(\frac{u_d}{u_T} \right) - 1 \right\}$$

gdzie $i_s = 10^{-12} \text{ mA}$ oraz $u_T = 0.025 \text{ V}$.



Rysunek 5.8. Układ zawierający element nieliniowy (diódę)

INTERPOLACJA I APROKSYMACJA FUNKCJI

Zadanie interpolacji, o którym mowa w podrozdziałach 6.1–6.3, można sformułować następująco. W przedziale $[a, b]$ dane są punkty o współrzędnych x_0, x_1, \dots, x_N oraz $f(x_0), f(x_1), \dots, f(x_N)$. Wyznaczyć taką funkcję ciągłą w przedziale $[a, b]$, aby jej wykres przechodził przez wszystkie te punkty. Rozpatrywać będziemy jedynie niektóre klasy funkcji interpolacyjnych, a mianowicie: wielomiany Lagrange'a, wielomiany Newtona oraz funkcje sklejane i funkcje trygonometryczne.

Zadanie aproksymacji, o którym mowa w podrozdziałach 6.4–6.6, można sformułować następująco. W przedziale $[a, b]$ dana jest funkcja $f(x)$. Wyznaczyć taką funkcję $\hat{f}(x; p)$ z parametrami $p = [p_1 \ p_2 \ \dots]^T$, której wartości w przedziale $[a, b]$ dostatecznie mało się różnią, w sensie przyjętego kryterium, od odpowiednich wartości funkcji $f(x)$. Rozpatrywać będziemy jedynie najważniejsze rodzaje aproksymacji: aproksymację średniokwadratową, aproksymację jednostajną i aproksymację Padégo.

6.1. INTERPOLACJA PRZY UŻYCIU WIELOMIANÓW NEWTONA I LAGRANGE'A

6.1.1. ZALEŻNOŚCI OGÓLNE

Interpolacja Lagrange'a polega na znalezieniu wielomianu interpolacyjnego $L_N(x)$ stopnia nie wyższego niż N , którego wartości w $N+1$ węzłach interpolacji są takie same jak wartości interpolowanej funkcji $f(x)$, tzn.:

$$L_N(x_n) = f(x_n) \quad \text{dla } n = 0, 1, \dots, N \quad (6.1)$$

Zadanie to ma jednoznaczne rozwiązanie postaci:

$$L_N(x) = \sum_{n=0}^N f(x_n) \prod_{v=0, v \neq n}^N \frac{x - x_v}{x_n - x_v} \quad (6.2)$$

Posługiwając się tym wzorem w obliczeniach, np. w związku z numerycznym obliczaniem całek czy pochodnych, jest uciążliwe. Wartość wielomianu Lagrange'a można obliczyć, korzystając z jego postaci naturalnej:

$$L_N(x) = \sum_{n=0}^N a_n x^n \quad (6.3)$$

Współczynniki a_n wyznacza się, rozwiązyując układ równań liniowych:

$$a_0 + a_1 x_n + \dots + a_N x_n^N = f(x_n) \quad \text{dla } n = 0, 1, \dots, N \quad (6.4)$$

Współczynnik uwarunkowania tego układu równań szybko rośnie z ich liczbą N . Na przykład, dla $N = 10$, $x_1 = 1$, $x_2 = 2, \dots, x_{10} = 10$ współczynnik oszacowania osiąga wartość $2.1 \cdot 10^{12}$. Oznacza to, że wartości współczynników a_0, a_1, \dots, a_N , wyznaczone z układu równań (6.4), mogą być obarczone znacznymi błędami. Z tego powodu do ich wyznaczania używa się wielomianów Newtona o postaci:

$$L_N(x) = \sum_{n=0}^N b_n p_n(x) \quad (6.5)$$

gdzie $p_0(x) = 1$ oraz $p_n(x) = (x - x_0)(x - x_1) \cdot \dots \cdot (x - x_{n-1})$ dla $n = 1, 2, \dots, N$. Zadanie wyznaczania współczynników b_n jest lepiej uwarunkowane numerycznie niż zadanie wyznaczania współczynników a_n . Zauważmy, że między wielomianami interpolacyjnymi tej samej funkcji L_{m-1} i L_m – rozpiętymi na węzłach x_0, x_1, \dots, x_{m-1} i x_0, x_1, \dots, x_m – zachodzi związek:

$$L_m(x) = L_{m-1}(x) + b_m p_m(x) \quad (6.6)$$

z którego wynika następująca zależność współczynników b_n od danych:

$$b_m = \frac{L_m(x) - L_{m-1}(x)}{p_m(x)} = \dots = \sum_{n=0}^m \frac{f(x_n)}{\prod_{v=0, v \neq n}^m (x_n - x_v)} \quad (6.7)$$

Powyższy algorytm wyznaczania współczynników b_m ma dobre właściwości numeryczne. Współczynniki b_m nazywane są niekiedy ilorazami różnicowymi funkcji opartymi na węzłach x_0, x_1, \dots, x_m , ponieważ są one równe elementom diagonalnym następującej macierzy trójkątnej:

$$\begin{aligned} & f(x_0) \\ & f(x_1) \quad f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ & f(x_2) \quad f[x_1, x_2] = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\ & \vdots \\ & f(x_N) \quad f[x_{N-1}, x_N] = \frac{f(x_N) - f(x_{N-1})}{x_N - x_{N-1}} \quad \dots \quad f[x_0, x_1, \dots, x_N] = \frac{f[x_1, \dots, x_N] - f[x_0, \dots, x_{N-1}]}{x_N - x_0} \end{aligned} \quad (6.8)$$

Właściwości ilorazów różnicowych omówione są szczegółowo w podręczniku [J1].

6.1.2. ZALEŻNOŚCI DLA WĘZŁÓW RÓWNODLEGŁYCH

Jeżeli węzły są rzeczywiste i równoodległe: $x_n = x_0 + nh$ dla $n = 1, \dots, N$, gdzie h jest odlegością między węzłami, to wielomian interpolacyjny Lagrange'a upraszcza się do postaci:

$$L_N(x) = \sum_{n=0}^N f(x_i) \prod_{v=0, v \neq n}^N \frac{\xi - v}{n - v} \quad (6.9)$$

gdzie $\xi = (x - x_0)/h$. W przypadku równoodległych węzłów szczególnie prostą postać przyjmują wielomiany Newtona:

$$p_n(x) = h^n \prod_{v=0}^{n-1} (\xi - v) \quad (6.10)$$

oraz ilorazy różnicowe:

$$f[x_0, x_1, \dots, x_n] = \frac{\Delta^n f(x_0)}{n! h^n} \quad (6.11)$$

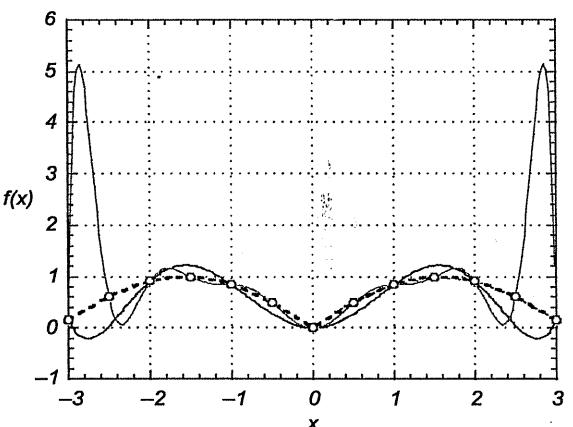
gdzie

$$\Delta^0 f(x) = f(x), \quad \Delta^n f(x) = \Delta^{n-1} f(x+h) - \Delta^{n-1} f(x) \quad \text{dla } n = 1, 2, \dots \quad (6.12)$$

Przy ich użyciu wielomian Newtona można zapisać w postaci:

$$L_N(x_0 + \xi h) = \Delta^0 f(x_0) + \Delta^1 f(x_0) q_1(\xi) + \frac{\Delta^2 f(x_0)}{2!} q_2(\xi) + \dots + \frac{\Delta^N f(x_0)}{n!} q_n(\xi) \quad (6.13)$$

gdzie $q_n(t) = p_n(t)/h = \prod_{v=0}^{n-1} (\xi - v)$.



Rysunek 6.1. Wykres funkcji $f(x) = |\sin(x)|$ (linia czerwona-przerywana) oraz wielomianów Lagrange'a szóstego (linia niebieska) i dwunastego stopnia (linia zielona) interpolujących tę funkcję

Przykład 6.1. Dokonano interpolacji funkcji $f(x) = |\sin(x)|$ w przedziale $[-3, +3]$ przy użyciu wielomianów Lagrange'a szóstego i dwunastego stopnia, dla węzłów odległych o 0.5. Wykresy tych wielomianów przedstawiono na rys. 6.1. Jak widać, dokładność interpolacji wielomianem dwunastego stopnia jest gorsza niż dokładność interpolacji wielomianem szóstego stopnia. Pogarszanie się dokładności interpolacji ze wzrostem stopnia wielomianu jest charakterystyczne dla interpolacji funkcji, które nie są ciągłe lub mają nieciągłe pochodne (tzw. efekt Rungego).

6.2. INTERPOLACJA PRZY UŻYCIU WIELOMIANOWYCH FUNKCJI SKLEJANYCH

W dotychczasowych rozważaniach zakładano, że funkcja $f(x)$ jest przybliżana jednym wielomianem odpowiednio wysokiego stopnia w całym przedziale interpolacji $[a, b]$. Jednak, jak pokazano w przykładzie 6.1, wzrost stopnia wielomianu niekoniecznie musi prowadzić do zwiększenia dokładności interpolacji. Z tego względu w praktyce dzieli się przedział interpolacji $[a, b]$ na kilka podprzedziałów i stosuje się do interpolacji wielomiany niskiego stopnia ($N = 2, \dots, 5$), o różnych współczynnikach w każdym z podprzedziałów. Jeżeli tak utworzona funkcja interpolująca $s(x)$ jest ciągła w przedziale $[a, b]$ wraz z co najmniej pierwszą pochodną, to nosi ona nazwę funkcji sklejanej (ang. *spline function*). W praktyce najczęściej używa się wielomianowych funkcji sklejanych stopnia trzeciego. Funkcję taką można w każdym z podprzedziałów przedstawić w postaci:

$$s_n(x) = a_n(x - x_n)^3 + b_n(x - x_n)^2 + c_n(x - x_n) + d_n \quad \text{dla } x \in [x_n, x_{n+1}] \quad (6.14)$$

gdzie $n = 0, 1, \dots, N-1$, przy czym $x_0 = a$ oraz $x_N = b$. Wymagane jest, aby funkcja ta spełniała następujące warunki:

$$s(x_n) = f(x_n) = d_n \quad \text{dla } n = 0, 1, \dots, N-1 \quad (6.15)$$

$$s(x_{n+1}) = f(x_{n+1}) \quad \text{dla } n = 0, 1, \dots, N-1 \quad (6.16)$$

$$\left. \frac{ds_n(x)}{dx} \right|_{x=x_{n+1}} = \left. \frac{ds_{n+1}(x)}{dx} \right|_{x=x_{n+1}} \quad \text{dla } n = 0, 1, \dots, N-2 \quad (6.17)$$

$$\left. \frac{d^2 s_n(x)}{dx^2} \right|_{x=x_{n+1}} = \left. \frac{d^2 s_{n+1}(x)}{dx^2} \right|_{x=x_{n+1}} \quad \text{dla } n = 0, 1, \dots, N-2 \quad (6.18)$$

Wzory (6.15)–(6.18) określają $4N-2$ równań algebraicznych względem $4N$ współczynników wielomianów postaci (6.14). Istnieje zatem możliwość swobod-

nego wyboru dwóch dowolnych współczynników. Funkcja $s(x)$ spełniająca warunki:

$$\frac{d^2 s_0(x)}{dx^2} \Big|_{x=x_0} = \frac{d^2 s_{N-1}(x)}{dx^2} \Big|_{x=x_N} = 0 \quad (6.19)$$

nazywana jest *naturalną funkcją sklejaną*. Z warunków tych wynikają wartości dwóch współczynników: $b_0 = b_{N-1} = 0$. Po wyeliminowaniu z pozostałych $4N-2$ równań współczynników a_n , c_n oraz d_n , otrzymuje się następujący układ równań liniowych:

$$\begin{bmatrix} 2 & w_1 & 0 & \dots & \dots & 0 \\ u_2 & 2 & w_2 & \dots & \dots & 0 \\ 0 & u_3 & 2 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 2 & w_{N-4} \\ 0 & \dots & \dots & \dots & u_{N-3} & 2 \\ 0 & \dots & \dots & \dots & 0 & u_{N-2} \end{bmatrix} \begin{bmatrix} b_1^* \\ b_2^* \\ b_3^* \\ \vdots \\ b_{N-4}^* \\ b_{N-3}^* \\ b_{N-2}^* \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{N-4} \\ v_{N-3} \\ v_{N-2} \end{bmatrix} \quad (6.20)$$

w którym:

$$b_n^* = \frac{b_n}{3}$$

$$u_n = \frac{h_{n-1}}{h_{n-1} + h_n}, \quad w_n = \frac{h_n}{h_{n-1} + h_n}$$

$$v_n = \frac{\frac{f(x_{n+1}) - f(x_n)}{h_n} - \frac{f(x_n) - f(x_{n-1})}{h_n}}{h_{n-1} + h_n}, \quad h_n = x_{n+1} - x_n$$

Rozwiążanie tego układu umożliwia wyznaczenie współczynników b_n^* , a następnie b_n oraz, przy użyciu pozostałych przedstawionych wcześniej równań, współczynników a_n , c_n i d_n . W pracy [J1] wykazano, że dla funkcji $f(x)$ ciągłej wraz z drugą pochodną w przedziale $[x_0, x_N]$ otrzymana w powyższy sposób funkcja sklejana spełnia zależność

$$\sup \{|s(x) - f(x)| \mid x \in [x_0, x_N]\} \leq 5M_2 \sup \{h_n^2 \mid n = 0, \dots, N-1\} \quad (6.21)$$

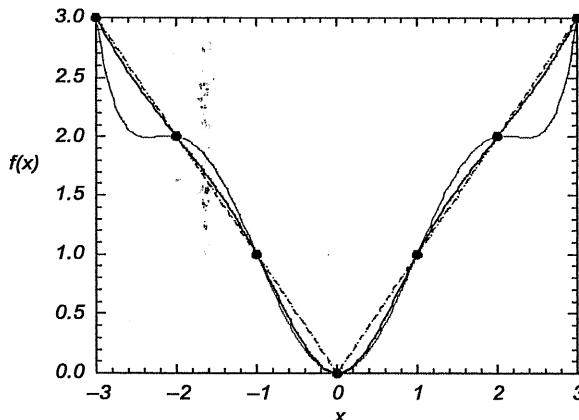
gdzie $M_2 = \max \{|f''(x)| \mid x \in [x_0, x_N]\}$.

Przykład 6.2. Zbiór punktów, których współrzędne określone są wektorami:

$$\mathbf{x} = [-3 -2 -1 0 1 2 3]^T \quad \text{oraz} \quad \mathbf{y} = [3 2 1 0 1 2 3]^T$$

reprezentuje dyskretne wartości funkcji $y = |x|$. Dokonać interpolacji tych punktów przy użyciu wielomianu szóstego stopnia oraz funkcji sklejanej trzeciego stopnia.

Współczynniki wielomianu w postaci naturalnej oraz współczynniki funkcji sklejanej wyznaczyć można w środowisku MATLAB przy użyciu procedur **polyfit** oraz **spline**. Procedura **p=polyfit(x,y,N)** służy do wyznaczenia wektora **p** zawierającego $N+1$ współczynników wielomianu N -tego stopnia w postaci naturalnej, interpolującego dane **x** i **y**. Wektor **v** wartości wielomianu, odpowiadający wektorowi wartości zmiennej niezależnej **u**, można obliczyć przy użyciu procedury **v=polyval(p,u)**. Wywołanie procedury **spline**: **v=spline(x,y,u)**, umożliwia bezpośrednie wyznaczenie wartości wielomianowej funkcji sklejanej trzeciego stopnia, interpolującej punkty zdefiniowane danymi **x** i **y** na (rys. 6.2).



Rys. 6.2. Funkcja $f(x) = |x|$ (linia czerwona) oraz wynik jej interpolacji za pomocą wielomianu Lagrange'a szóstego stopnia (linia zielona) i wielomianowej funkcji sklejanej trzeciego stopnia (linia niebieska). Jak widać, w tym przypadku interpolacja za pomocą funkcji sklejanej jest o wiele dokładniejsza niż interpolacja wielomianem Lagrange'a szóstego stopnia

6.3. INTERPOLACJA TRYGONOMETRYCZNA

Interpolacja trygonometryczna funkcji $f(x)$ o okresie 2π polega na wyznaczeniu nieznanych współczynników liniowej kombinacji zespolonych funkcji wykładniczych:

$$\hat{f}(x) = \sum_{m=0}^{N-1} c_m \exp(jmx), \quad \text{gdzie } j^2 = -1 \quad (6.22)$$

która w N punktach przedziału $[0, 2\pi]$ przyjmuje te same wartości co funkcja $f(x)$. Najczęściej stosuje się interpolację trygonometryczną opartą na równoodległych węzłach:

$$x_n = n \frac{2\pi}{N} \quad \text{dla } n = 0, 1, \dots, N-1 \quad (6.23)$$

Zespolone współczynniki:

$$c_m = \frac{1}{N} \sum_{n=0}^{N-1} f(x_n) \exp\left(-jm\frac{2\pi}{N}\right) \quad \text{dla } m = 0, 1, \dots, N-1 \quad (6.24)$$

nazywane są *współczynnikami rozwinięcia funkcji $f(x)$ w szereg Fouriera*, a zadanie ich wyznaczania – *zagadnieniem analizy fourierowskiej*. Zadanie odwrotne, tzw. *zagadnienie syntezy fourierowskiej*, polega na wyznaczeniu wartości funkcji według wzoru (6.22) przy podstawieniu za x wartości określonych wzorem (6.23):

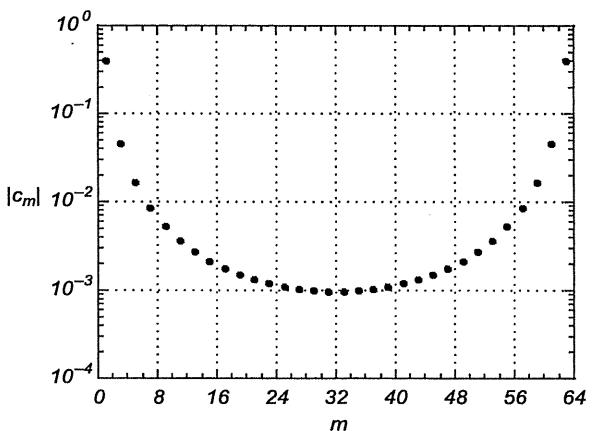
$$f(x_n) = \sum_{m=0}^{N-1} c_m \exp\left(jmn\frac{2\pi}{N}\right) \quad \text{dla } n = 0, 1, \dots, N-1 \quad (6.25)$$

Bezpośrednie użycie wzoru (6.24) lub (6.25) wymaga wykonania około N^2 zespolonych mnożeń i dodawień (wartości zespolonych funkcji wykładniczych mogą być stabilizowane). Nakłady obliczeniowe można istotnie zmniejszyć, posługując się algorytmami tzw. szybkiej transformacji Fouriera (ang. *Fast Fourier Transform*, w skrócie – FFT). Korzyści stąd płynące są największe, gdy $N = 2^P$, gdzie P jest liczbą naturalną. Wówczas algorytm FFT wymaga wykonania rzędu NP działań. W związku z tak dużą efektywnością tego algorytmu jest on stosowany w wielu zagadnieniach analizy numerycznej. W praktyce inżynierskiej szybka transformacja Fouriera jest wykorzystywana, między innymi, do:

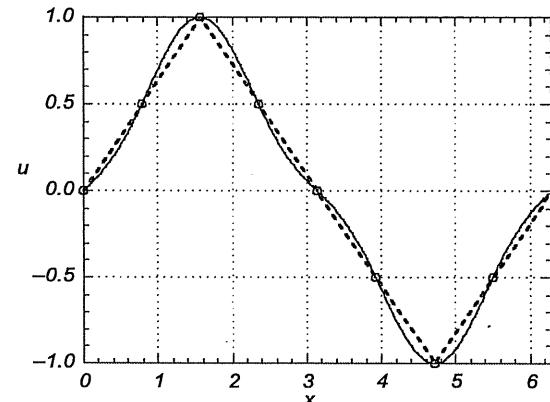
- analizy widma sygnałów;
- częstotliwościowej filtracji sygnałów;
- przetwarzania obrazów;
- rozwiązywania niektórych równań różniczkowych, np. równania Poissona.

Przykład 6.3. Wyznaczyć moduły współczynników $|c_m|$ dla przebiegu napięcia $u(t)$, tj. zależności napięcia u od czasu t , pokazanego na rys. 6.4 – dla $N = 8$ i $N = 64$. Dla $N = 8$ wyznaczyć przebieg napięcia $\hat{u}_8(t)$ będący odwrotną transformacją Fouriera wektora $\mathbf{c} = [c_0 \ c_1 \ \dots \ c_7]^T$. Dla $N = 64$ wyznaczyć estymatę napięcia $\hat{u}_{64}(t)$ jako odwrotną transformację Fouriera wektora $\mathbf{c} = [c_0 \ c_1 \ \dots \ c_{63}]^T$, w którym wyzerowano wszystkie składowe poza c_1 i c_3 oraz c_{61} i c_{63} .

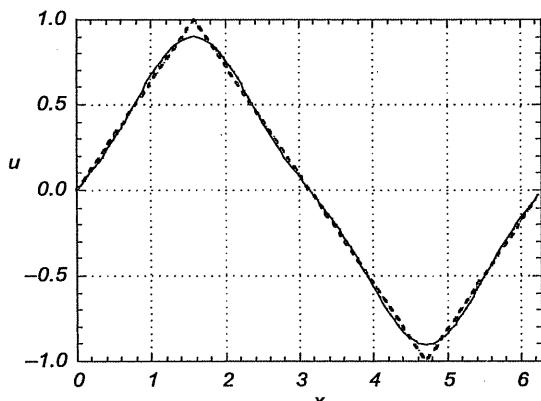
Ponieważ $u(t)$ jest funkcją nieparzystą, co drugi współczynnik c_m jest dla niej równy zeru. Z tego względu parzyste współczynniki Fouriera dla tej funkcji znikają, a wyznaczanie $\hat{u}_8(t)$ jest równoważne interpolacji $u(t)$ za pomocą sumy dwóch jego harmonicznych: pierwszej i trzeciej. Natomiast $\hat{u}_{64}(t)$ jest równoważne aproksymacji napięcia. Wyniki obliczeń przedstawione na rys. 6.3, rys. 6.4 i rys. 6.5 uzyskano przy użyciu procedur `fft` oraz `ifft` w środowisku MATLAB. Jak widać, dla $N = 8$ funkcja interpolująca przechodzi przez węzły interpolacji, których jest 8. Dla $N = 64$ mamy do czynienia z aproksymacją opisaną w następującym podrozdziale.



Rysunek 6.3. Moduły współczynników c_m rozwinięcia w szereg Fouriera napięcia $u(t)$ dla $N = 64$



Rysunek 6.4. Napięcie $u(t)$ (linia czerwona-przerywana) oraz wynik jego interpolacji $\hat{u}_8(t)$ (linia czarna-ciągła) za pomocą sumy dwóch harmonicznych, pierwszej i trzeciej, dla $N = 8$



Rysunek 6.5. Napięcie $u(t)$ (linia czerwona-przerywana) oraz wynik jego aproksymacji $\hat{u}(t)$ (linia czarna-ciągła) za pomocą sumy dwóch harmonicznych, pierwszej i trzeciej, dla $N = 64$

6.4. APROKSYMACJA ŚREDNIOKWADRATOWA

6.4.1. APROKSYMACJA FUNKCJI DANEJ W POSTACI ANALITYCZNEJ

Klasyczne zadanie aproksymacji polega na znalezieniu funkcji $\hat{f}(x; \mathbf{p})$, która w sensie założonego kryterium najlepiej przybliża daną funkcję ciągłą $f(x)$ w przedziale $[a, b]$. Poszukiwana funkcja ma najczęściej postać:

$$\hat{f}(x; \mathbf{p}) = \sum_{k=1}^K p_k \varphi_k(x) \quad (6.26)$$

gdzie $\{\varphi_k(x) | k = 1, 2, \dots, K\}$ jest ciągiem zadanych funkcji, a $\mathbf{p} = [p_1 \ p_2 \ \dots \ p_K]^T$ jest wektorem parametrów, które należy wyznaczyć. Najczęściej stosowanym kryterium aproksymacji jest błąd średniokwadratowy zdefiniowany wzorem:

$$J'_2(\mathbf{p}) = \sqrt{\frac{1}{b-a} \int_a^b [\hat{f}(x; \mathbf{p}) - f(x)]^2 dx} \quad (6.27)$$

przy czym w praktyce minimalizuje się funkcjonal:

$$J_2(\mathbf{p}) = (b-a)[J'_2(\mathbf{p})]^2 \quad (6.28)$$

który osiąga minimum dla tego samego wektora \mathbf{p} . Minimum to wyznacza się z warunku koniecznego:

$$\frac{\partial J_2(\mathbf{p})}{\partial p_k} = 0 \quad \text{dla } k = 1, 2, \dots, K \quad (6.29)$$

który prowadzi do układu liniowych równań algebraicznych o postaci:

$$\mathbf{Ap} = \mathbf{b} \quad (6.30)$$

Elementy macierzy \mathbf{A} mają postać:

$$a_{kj} = \int_a^b \varphi_k(x) \varphi_j(x) dx \quad \text{dla } k, j = 1, 2, \dots, K \quad (6.31)$$

natomiast elementy wektora \mathbf{b} – postać:

$$b_k = \int_a^b f(x) \varphi_k(x) dx \quad \text{dla } k = 1, 2, \dots, K \quad (6.32)$$

Zadanie średniokwadratowej aproksymacji funkcji ma więc szczególnie proste rozwiązanie, jeśli macierz układu (6.30) jest diagonalna, co ma miejsce wtedy, gdy całki określone wzorem (6.31) zerują się dla $k \neq j$.

W dalszej części zostaną sformułowane podstawowe pojęcia i pokazane sposoby konstruowania funkcji $\varphi_k(x)$, dla których macierz układu (6.30) jest diagonalna (funkcje takie będą nazywane ortogonalnymi).

Przez $L_p^2[a, b]$ oznaczymy przestrzeń funkcji mieralnych na $[a, b]$ i takich, że:

$$\int_a^b f^2(x) w(x) dx < \infty$$

gdzie $w(x)$ jest funkcją nieujemną taką, że:

$$\int_a^b w(x) dx < \infty$$

Przestrzeń $L_p^2[a, b]$ jest przestrzenią liniową unormowaną, z normą określoną przez wyrażenie:

$$\|f(x)\| = \sqrt{\int_a^b f^2(x) w(x) dx} \quad (6.33)$$

Przestrzeń $L_p^2[a, b]$ jest przestrzenią unitarną, z iloczynem skalarnym zdefiniowanym następująco:

$$\langle f | g \rangle = \int_a^b f(x) g(x) w(x) dx \quad (6.34)$$

Jeżeli iloczyn skalarny dwóch funkcji należących do tej samej przestrzeni unitarnej jest równy零, to mówimy, że funkcje te są *ortogonalne*. Skończony lub

nieskończony zbiór niezerowych funkcji $\{\varphi_1, \varphi_2, \dots\}$ należących do tej samej przestrzeni unitarnej nazywamy *układem ortogonalnym*, jeżeli dowolna para tych funkcji jest ortogonalna, tzn. jeżeli

$$\langle \varphi_k | \varphi_j \rangle = \begin{cases} 0 & \text{dla } k \neq j \\ \|\varphi_k\|_2^2 & \text{dla } k = j \end{cases} \quad (6.35)$$

Jeżeli ponadto $\|\varphi_k\|_2^2 = 1$ dla $k = 1, 2, \dots$, to układ taki nazywamy *ortonormalnym*.

Dowolny układ liniowo niezależnych funkcji $\{\varphi_1, \varphi_2, \dots\}$, należących do przestrzeni unitarnej, można przekształcić w układ ortogonalny $\{\psi_1, \psi_2, \dots\}$, którego elementami są funkcje wyrażające się kombinacjami liniowymi funkcji $\{\varphi_1, \varphi_2, \dots\}$. Procedura tworzenia tego układu znana jest pod nazwą *ortogonalizacji Grama-Schmidta*:

$$\psi_1 = \varphi_1, \quad \psi_k = \varphi_k - \sum_{j=1}^{k-1} \frac{\langle \varphi_k | \psi_j \rangle}{\langle \psi_j | \psi_j \rangle} \psi_j \quad \text{dla } k = 2, 3, \dots \quad (6.36)$$

Ortogonalizacja ciągu funkcji potęgowych:

$$\varphi_k(x) = x^k \quad \text{dla } k = 0, 1, \dots \quad (6.37)$$

przy różnych założeniach dotyczących przedziału $[a, b]$ oraz funkcji wagowej $w(x)$, prowadzi do konstrukcji różnych układów wielomianów algebraicznych postaci:

$$\psi_k(x) = a_k^{(k)} x^k + \dots + a_1^{(k)} x + a_0^{(k)} \quad \text{dla } k = 0, 1, \dots \quad (6.38)$$

W ustalonej przestrzeni $L_p^2[a, b]$ ciąg wielomianów ortogonalnych jest określony jednoznacznie z dokładnością do stałych. Pierwiastki wielomianów ortogonalnych w przestrzeni $L_p^2[a, b]$ są rzeczywiste, jednokrotne i położone we wnętrzu przedziału $[a, b]$.

Przykład 6.4. Wielomiany Legendre'a, zdefiniowane następującym wzorem rekurencyjnym:

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

są ortogonalne w przestrzeni $L_p^2[-1, +1]$ z wagą $w(x) = 1$, a kwadrat ich normy wyraża się wzorem:

$$\|P_k\|^2 = \int_{-1}^1 P_k^2(x) dx = \frac{2}{2k+1}$$

Wielomiany Hermite'a, określone wzorami:

$H_0(x) = 1, \quad H_1(x) = 2x, \quad H_k(x) = 2xH_{k-1}(x) - (2k-2)H_{k-2}(x)$ dla $k = 2, 3, \dots$, są ortogonalne w przestrzeni $L_p^2(-\infty, +\infty)$ z funkcją wagową $w(x) = \exp(-x^2)$, a kwadrat ich normy wyraża się wzorem:

$$\|H_k\|^2 = \int_{-\infty}^{+\infty} \exp(-x^2) H_k^2(x) dx = \sqrt{\pi} 2^k k!$$

Wielomiany Czebyszewa pierwszego rodzaju zdefiniowane następująco:

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

są ortogonalne w przedziale $L_p^2[-1, +1]$ z funkcją wagową: $w(x) = 1/\sqrt{1-x^2}$, a kwadrat ich normy wyraża się wzorem:

$$\|T_k(x)\|^2 = \int_{-1}^1 T_k^2(x) p(x) dx = \begin{cases} \pi & \text{dla } k = 0 \\ \pi/2 & \text{dla } k \neq 0 \end{cases}$$

Wielomiany Czebyszewa mają wszystkie zera rzeczywiste, jednokrotne leżące w przedziale $[-1, +1]$, określone następującym wzorem:

$$z_j = \cos \left[\frac{(2j-1)\pi}{2k} \right] \quad \text{dla } j = 1, 2, \dots, k$$

Praktyczne wykorzystanie wielomianów ortogonalnych do aproksymacji średniokwadratowej wymaga zwykle zamiany zmiennej x w taki sposób, aby dostosować przedział aproksymacji funkcji $f(x)$ do przedziału ortogonalności wielomianów albo postać wielomianów ortogonalnych – do przedziału aproksymacji.

Przykład 6.5. W przypadku wyboru wielomianów Legendre'a przedział aproksymacji można dostosować do przedziału ortogonalności tych wielomianów, tzn. $[-1, +1]$, za pomocą następującego przekształcenia:

$$f(x) = f\left(\frac{b-a}{2}x' + \frac{a+b}{2}\right) \quad \text{dla } x' \in [-1, +1]$$

W przypadku wielomianów $\{\varphi_k(x) \mid k = 0, 1, \dots\}$, ortogonalnych względem iloczynu skalarnego z wagą $w(x) \neq 1$, do aproksymacji średniokwadratowej wygodnie jest przyjąć jako bazę układ funkcji:

$$\psi_k(x) = \sqrt{w(x)} \varphi_k(x) \quad \text{dla } k = 0, 1, \dots, K \quad (6.39)$$

ortogonalnych w sensie iloczynu skalarnego z wagą jednostkową.

Ortogonalny układ funkcji $\{\varphi_k(x) \mid k = 0, 1, \dots\}$ staje się układem ortonormalnym po dokonaniu normalizacji:

$$\psi_k(x) = \frac{\varphi_k(x)}{\|\varphi_k\|} \quad \text{dla } k = 0, 1, \dots, K \quad (6.40)$$

Wynik aproksymacji średniokwadratowej za pomocą funkcji ortonormalnych można zapisać w postaci:

$$p_k = \langle f | \psi_k \rangle \quad \text{dla } k = 0, 1, \dots, K \quad (6.41)$$

Adekwatnym narzędziem aproksymacji funkcji okresowych, danych w postaci analitycznej, są wielomiany trygonometryczne, tzn. liniowe kombinacje funkcji:

$$1, \cos(x), \sin(x), \cos(2x), \sin(2x), \dots \quad (6.42)$$

które w przestrzeni $L_p^2[0, 2\pi]$ tworzą układ ortogonalny. Rozwinięcie funkcji $f(x)$ w szereg tych funkcji, zwany szeregiem Fouriera, ma postać:

$$\hat{f}(x; \mathbf{p}) = a_0 + \sum_{k=1}^K [a_k \cos(kx) + b_k \sin(kx)] \quad (6.43)$$

przy czym elementy wektora parametrów $\mathbf{p} = [a_0 \ a_1 \ b_1 \ \dots \ a_K \ b_K]^T$ wyrażają się następującymi wzorami:

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x) dx \quad (6.44)$$

$$a_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(kx) dx \quad (6.45)$$

$$b_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(kx) dx \quad (6.46)$$

Zależności te zostały wyrowadzone na podstawie wzorów (6.26)–(6.32). Tak więc szereg Fouriera jest aproksymacją średniokwadratową funkcji $f(x)$ dla bazy (6.42).

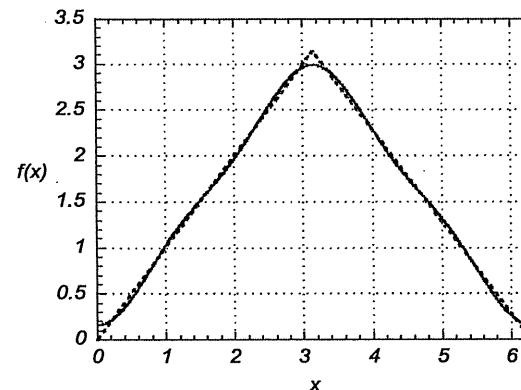
Przykład 6.6. Na rys. 6.6 przedstawiono wynik aproksymacji funkcji:

$$f(x) = \begin{cases} x & \text{dla } 0 < x \leq \pi \\ 2\pi - x & \text{dla } \pi x \leq 2\pi \end{cases}$$

szeregiem Fouriera dla $K = 4$ i $K = 64$. Współczynniki Fouriera obliczone według wzorów (6.44), (6.45) i (6.46) są w tym przypadku następujące:

$$a_0 = \frac{\pi}{2}, \quad a_k = \frac{2}{\pi k^2} [(-1)^k - 1], \quad b_k = 0$$

Warto zwrócić uwagę, że wyniki aproksymacji mogą różnić się nieco od wyników aproksymacji uzyskanych przy użyciu fft i ifft w przykładzie 6.3, ponieważ do wyznaczenia współczynników Fouriera metodą fft korzystano z informacji o funkcji w dyskretnej liczbie punktów, podczas gdy wzorach (6.44), (6.45) i (6.46) wykorzystywana jest pełna informacja o funkcji w przedziale $[0, 2\pi]$.



Rysunek 6.6. Funkcja $f(x)$ (linia czerwona) oraz wynik jej aproksymacji $\hat{f}(x)$ (linia czarna) dla $K = 4$; wykres aproksymacji funkcji dla $K = 64$ pokrywa się z wykresem funkcji $f(x)$

6.4.2. APROKSYMACJA FUNKCJI NA PODSTAWIE CIĄGU JEJ DYSKRETNYCH WARTOŚCI

Celem aproksymacji w zadaniach inżynierskich jest najczęściej znalezienie funkcji $\hat{f}(x; \mathbf{p})$, która w sensie założonego kryterium najlepiej przybliża ciąg danych pomiarowych:

$$\{x_n, f(x_n) \mid n = 1, 2, \dots, N\} \quad (6.47)$$

Poszukiwana funkcja ma najczęściej postać:

$$\hat{f}(x; \mathbf{p}) = \sum_{k=1}^K p_k \varphi_k(x) \quad (6.48)$$

gdzie $\{\varphi_k(x) \mid k = 1, 2, \dots, K\}$ jest ciągiem zadanych funkcji, a \mathbf{p} jest wektorem parametrów, które należy wyznaczyć, przy czym $K < N$. Najczęściej stosowanym kryterium aproksymacji jest błąd średniokwadratowy zdefiniowany wzorem:

$$J'_2(\mathbf{p}) = \sqrt{\frac{1}{N} \sum_{n=1}^N [\hat{f}(x_n; \mathbf{p}) - f(x_n)]^2} \quad (6.49)$$

przy czym w praktyce minimalizuje się funkcjonał:

$$J_2(\mathbf{p}) = N [J'_2(\mathbf{p})]^2 \quad (6.50)$$

który osiąga minimum dla tego samego wektora \mathbf{p} . Minimum to wyznacza się z warunku koniecznego:

$$\frac{\partial J_2(\mathbf{p})}{\partial p_k} = 0 \quad \text{dla } k = 1, 2, \dots, K \quad (6.51)$$

który prowadzi do nadokreślonego (jeżeli $N > K$) układu liniowych równań algebraicznych postaci:

$$\Phi \mathbf{p} = \mathbf{y} \quad (6.52)$$

gdzie $\mathbf{y} = [y_1 \ y_2 \ \dots \ y_N]^T$, zaś macierz Φ ma postać:

$$\Phi = \begin{bmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_K(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_K(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_N) & \varphi_2(x_N) & \cdots & \varphi_K(x_N) \end{bmatrix}$$

Układ (6.52) może być rozwiązywany metodami omówionymi w podrozdziale 4.4, np. metodą SVD lub QR po sprowadzeniu do układu równań normalnych: $\Phi^T \Phi \mathbf{p} = \Phi^T \mathbf{y}$. Numeryczne uwarunkowanie tego zadania istotnie zależy od wyboru bazy. Najłatwiej je rozwiązać, gdy kolumny macierzy Φ są ortogonalne. Dominację głównej przekątnej macierzy $\Phi^T \Phi$ zapewnia użycie wielomianów ortogonalnych omówionych w podrozdziale 6.4.1, a jej diagonalność – odpowiedni wybór punktów $\{x_n\}$.

Przykład 6.7. Informacja o charakterystyce prądowo-napięciowej pewnego elementu nieliniowego dana jest w postaci wyników $N = 9$ niezależnych pomiarów:

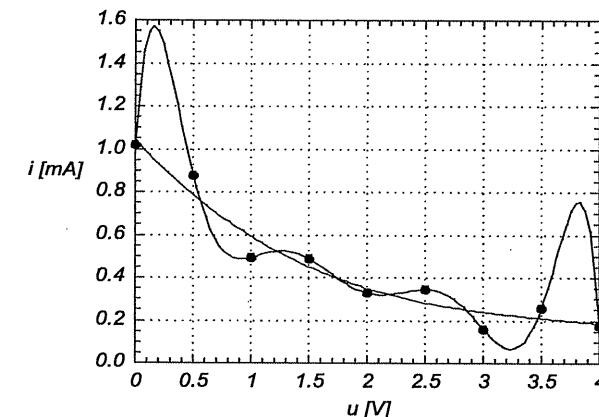
$$\mathbf{u} = [0 \ 0.5 \ 1.0 \ 1.5 \ 2.0 \ 2.5 \ 3.0 \ 3.5 \ 4.0]^T \text{ [V]}$$

$$\mathbf{i} = [1.017 \ 0.8788 \ 0.4925 \ 0.4883 \ 0.3281 \ 0.3452 \ 0.1613 \ 0.2575 \ 0.1747]^T \text{ [mA]}$$

obarczonych błędami przypadkowymi. Dokonać aproksymacji tej charakterystyki wielomianami trzeciego oraz ósmego stopnia.

Współczynniki wielomianu w postaci naturalnej – aproksymującego dane (\mathbf{u}, \mathbf{i}) – można wyznaczyć w środowisku MATLAB przy użyciu procedury `polyfit`. Procedura ta służy do wyznaczenia wektora $\mathbf{p}=polyfit(\mathbf{u}, \mathbf{i}, N)$ zawierającego $N+1$ współczynników wielomianu N -tego stopnia w postaci naturalnej, aproksymującego dane (\mathbf{u}, \mathbf{i}) w sensie średniokwadratowym. W szczególnym przypadku, gdy rozmiar wektorów \mathbf{u} i \mathbf{i} jest równy $N+1$, mamy do czynienia z interpolacją

danych (porównaj przykład 6.2). Wykresy wielomianów trzeciego i ósmego stopnia aproksymujących dane (\mathbf{u}, \mathbf{i}) przedstawiono na rys. 6.7.



Rysunek 6.7. Charakterystyka prądowo-napięciowa elementu nieliniowego: dane pomiarowe (●), wynik ich aproksymacji za pomocą wielomianu trzeciego stopnia (linia czerwona-ciągła) i wynik interpolacji za pomocą wielomianu ósmego stopnia (linia niebieska)

Jak widać, charakterystyka opisana wielomianem trzeciego stopnia jest krzywą „glądką”, lecz nieprzechodzącą przez punkty pomiarowe. Wielomian ósmego stopnia przechodzi co prawda przez wszystkie punkty pomiarowe, lecz poza nimi wykazuje tendencję do oscylacji.

Przykład 6.8. Na podstawie wartości funkcji:

$$f_1(x) = 2e^x \sin(x)$$

oraz

$$f_2(x) = |x - 1| + 1$$

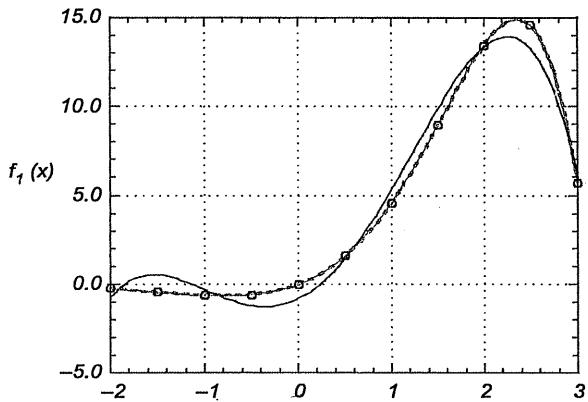
w 11 równoodległych punktach przedziału $[-2, +3]$ wyznaczono:

- wielomiany stopnia $K = 4, 6, 8$ i 10 interpolujące funkcje $f_1(x)$ i $f_2(x)$;
- wielomiany stopnia $K = 4, 6, 8$ i 10 aproksymujące te funkcje w sensie najmniejszych kwadratów;
- funkcje sklejane trzeciego stopnia interpolujące te funkcje;
- funkcje sklejane trzeciego stopnia aproksymujące te funkcje w sensie najmniejszych kwadratów.

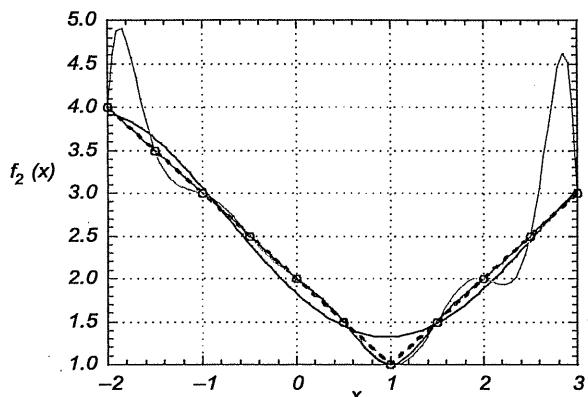
Wykresy niektórych spośród wyznaczonych funkcji przedstawiono na rys. 6.8 i rys. 6.9. Na rys. 6.10 natomiast przedstawiono wskaźnik:

$$\Delta_2 = \sqrt{\frac{1}{N} \sum_{n=1}^N \left| \widehat{f}_j(x_n; \mathbf{p}) - f_j(x_n) \right|^2} \quad \text{dla } j=1, 2$$

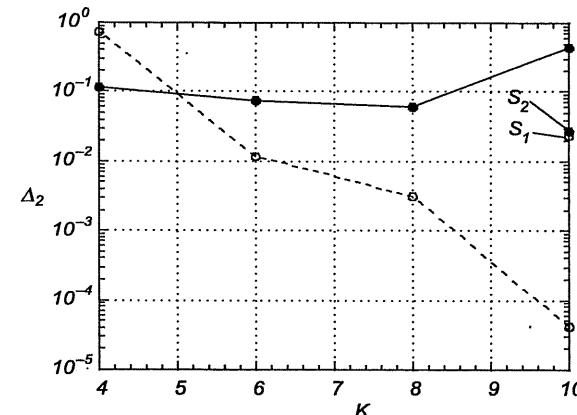
charakteryzujący dokładność interpolacji lub aproksymacji dla wszystkich wyznaczonych funkcji. W powyższej definicji wskaźnika $\widehat{f}_j(x; \mathbf{p})$ oznacza dowolną funkcję interpolującą lub aproksymującą $f_j(x)$ ($j=1$ lub 2), a $x_n = -2 + 0.05(n-1)$ dla $n=1, 2, \dots, 101$.



Rysunek 6.8. Funkcja $f_1(x)$ (linia czerwona), wielomian interpolacyjny dziesiątego stopnia (linia czarna pokrywająca się z linią czerwoną), wielomian aproksymujący czwartego stopnia (linia niebieska) i wielomianowa funkcja sklejana trzeciego stopnia (linia zielona); kółkami zaznaczono węzły interpolacji dla wielomianu dziesiątego stopnia



Rysunek 6.9. Funkcja $f_2(x)$ (linia czerwona), wielomian interpolacyjny dziesiątego stopnia (linia zielona), wielomian aproksymujący czwartego stopnia (linia czarna) i wielomianowa funkcja sklejana trzeciego stopnia (linia niebieska – widoczna tylko dla $x \in [-0.5, 0.5]$); kółkami zaznaczono węzły interpolacji dla wielomianu dziesiątego stopnia



Rysunek 6.10. Błędy aproksymacji ($K < 10$) oraz interpolacji ($K = 10$) funkcji $f_1(x)$ (linia przerywana) i funkcji $f_2(x)$ (linia ciągła) w zależności od stopnia wielomianu aproksymującego (interpolującego); symbolami S_1 i S_2 wskazano wartości błędu interpolacji wielomianowymi funkcjami sklejanyimi trzeciego stopnia, odpowiednio, dla pierwszej i drugiej funkcji

Na rys. 6.8 przedstawiono dane i wyniki uzyskane dla funkcji $f_1(x)$. Widać duże różnice między wykresami wielomianu czwartego stopnia i funkcji $f_1(x)$, podczas gdy zarówno wykresy wielomianu interpolującego dziesiątego stopnia, jak i wielomianowej funkcji sklejanej trzeciego stopnia praktycznie pokrywają się z wykresem $f_1(x)$. Na rys. 6.9 przedstawiono dane i wyniki uzyskane dla funkcji $f_2(x)$. Zarówno wykres wielomianu interpolacyjnego dziesiątego stopnia, jak i wielomianu aproksymującego czwartego stopnia różni się znacznie od wykresu $f_2(x)$; dla tej funkcji najlepsze wyniki daje interpolacja wielomianową funkcją sklejaną trzeciego stopnia. Porównanie błędów aproksymacji (interpolacji) w zależności od stopnia wielomianu aproksymującego (rys. 6.10) pokazuje, że dla funkcji $f_1(x)$ błędy aproksymacji maleją ze wzrostem stopnia wielomianu, natomiast dla funkcji $f_2(x)$ uzyskuje się minimum błędu dla wielomianu aproksymującego siódmeego stopnia. Interpolacja funkcjami sklejonymi daje najmniejsze błędy dla funkcji $f_2(x)$, ale jest gorsza od aproksymacji wielomianami stopni $6 < N < 9$ oraz interpolacji wielomianem stopnia $N=10$ dla funkcji $f_1(x)$. Można podać ogólne wnioski dotyczące interpolacji i aproksymacji wielomianowych opartych na równoodległych węzłach:

- dla funkcji ciągłych wraz ze wszystkimi pochodnymi (do takiej klasy należy funkcja $f_1(x)$) błędy zarówno interpolacji, jak i aproksymacji maleją w sposób monotoniczny ze wzrostem stopnia wielomianu;
- dla funkcji o nieciągłych pochodnych (takich jak funkcja $f_2(x)$) największe błędy interpolacji uzyskuje się zazwyczaj przy zastosowaniu wielomianów sklejanych, natomiast wzrost stopnia wielomianu interpolującego lub aproksymującego nie musi poprawiać jakości interpolacji (bądź aproksymacji).

6.5. INNE RODZAJE APROKSYMACJI

6.5.1. APROKSYMACJA JEDNOSTAJNA

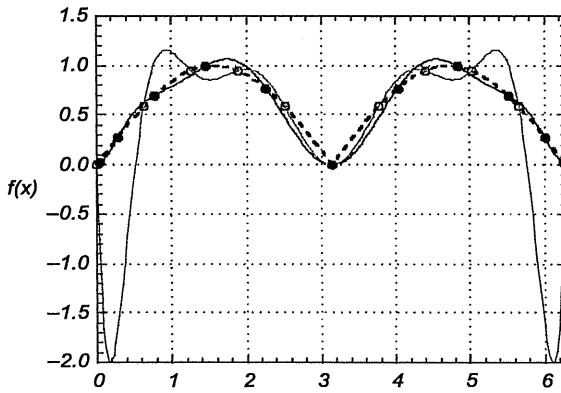
Aproksymacja jednostajna polega na minimalizacji błędu maksymalnego:

$$J_{\infty}(\mathbf{p}) = \sup \{ |\hat{f}(x_n; \mathbf{p}) - f(x_n)| \mid n=1, \dots, N \} \quad (6.53)$$

lub

$$J_{\infty}(\mathbf{p}) = \sup \{ |\hat{f}(x; \mathbf{p}) - f(x)| \mid x \in [a, b] \} \quad (6.54)$$

Warunki jednoznaczności wyznaczenia optymalnego wektora parametrów \mathbf{p} określa twierdzenie Czebyszewa [J1]. Twierdzenie to stanowi teoretycznie podstawę konstrukcji metod rozwiązywania zadania aproksymacji jednostajnej. Metody te są jednak znacznie trudniejsze niż metody aproksymacji średniokwadratowej; dlatego w praktyce inżynierskiej bardzo często poprzestaje się na poszukiwaniu rozwiązania przybliżonego, polegającego na aproksymacji średniokwadratowej wielomianami Czebyszewa pierwszego rodzaju. Tak wyznaczone rozwiązanie nieznacznie tylko odbiega od rozwiązania optymalnego, a jego uzyskanie jest o wiele łatwiejsze [J1, P]. Jeszcze prostszym, dającym dobre rezultaty rozwiązaniem jest interpolacja wielomianami Lagrange'a oparta na węzłach Czebyszewa (będących zerami wielomianu Czebyszewa odpowiedniego stopnia). Zilustrowano to na rys. 6.11. Jak widać, użycie interpolacji wielomianowej opartej na węzłach Czebyszewa wyeliminowało efekt wzrostu błędów na krańcach dziedziny funkcji występujący przy interpolacji funkcji o nieciągłych pochodnych wielomianami opartymi na równoodległych węzłach (porównaj przykład 6.1).



Rysunek 6.11. Wynik interpolacji funkcji $f(x) = |\sin(x)|$ (linia czerwona) wielomianem dziesiątego stopnia opartym na węzłach równoodległych (linia zielona) i wielomianem dziesiątego stopnia opartym na węzłach Czebyszewa (linia niebieska)

6.5.2. APROKSYMACJA FUNKCJAMI NIELINIOWYMI WZGLĘDEM PARAMETRÓW

W wielu zagadnieniach praktycznych użycie do aproksymacji funkcji $\hat{f}(x; \mathbf{p})$ nieliniowych względem wektora parametrów \mathbf{p} daje lepsze wyniki niż użycie wielomianów algebraicznych lub trygonometrycznych: przy tej samej dokładności aproksymacji liczba parametrów jest istotnie mniejsza niż liczba współczynników wielomianu. Klasycznym przykładem jest *aproksymacja Padégo* za pomocą funkcji wymiernych o postaci:

$$\hat{f}(x; \mathbf{p}) = \frac{a_0 + a_1 x + \dots + a_L x^L}{1 + b_1 x + \dots + b_M x^M} \quad \text{dla } L = 0, 1, \dots, M = 1, 2, \dots, L \leq M \quad (6.55)$$

gdzie $\mathbf{p} = [a_0 \dots a_L \ b_1 \dots b_M]^T$. Parametry \mathbf{p} wyznacza się w tym przypadku tak, aby rozwinięcie funkcji aproksymującej $\hat{f}(x; \mathbf{p})$ w szereg Maclaurina było zgodne z odpowiednim rozwinięciem funkcji aproksymowanej:

$$f(x) = \sum_{k=0}^{\infty} c_k x^k \quad \text{dla } x \in [0, \varepsilon] \quad (6.56)$$

z dokładnością do $L + M + 1$ wyrazów. Żąda się więc, aby zachodziła równość:

$$\hat{f}(x; \mathbf{p}) + O(x^{L+M+1}) = f(x) \quad (6.57)$$

w pewnym otoczeniu zera. Jest ona równoważna następującej równości:

$$a_0 + a_1 x + \dots + a_L x^L + O(x^{L+M+1}) = (1 + b_1 x + \dots + b_M x^M) \sum_{k=0}^{\infty} c_k x^k \quad (6.58)$$

z której – po porównaniu współczynników przy jednakowych potęgach x po lewej i prawej stronie – wynika układ liniowy równań algebraicznych:

$$\left. \begin{array}{l} c_l + \sum_{k=0}^{l-1} c_k b_{l-k} = a_l \quad \text{dla } l = 0, 1, \dots, L \\ c_l + \sum_{k=0}^{l-1} c_k b_{l-k} = 0 \quad \text{dla } l = L+1, \dots, M \\ c_l + \sum_{k=l-M}^{l-1} c_k b_{l-k} = 0 \quad \text{dla } l = M+1, \dots, M+L \end{array} \right\} \quad (6.59)$$

Choć funkcja wymierna (6.55) jest przykładem funkcji nieliniowej względem parametrów, to jej użycie do aproksymacji nie wymaga nieliniowych narzędzi numerycznych, ponieważ pomnożenie obu stron wzoru (6.55) przez mianownik funkcji wymiernej w pewnym sensie linearyzuje zadanie. W wielu praktycznie ważnych przypadkach celowe jest jednak użycie do aproksymacji funkcji $\hat{f}(x; \mathbf{p})$ nieliniowej względem parametrów, która takich możliwości nie daje, a jej użycie

jest pożądane ze względu na fizyczne lub techniczne znaczenie jej parametrów. Wówczas do minimalizacji funkcjonału $J_2(\mathbf{p})$ lub $J_\infty(\mathbf{p})$ trzeba użyć procedury optymalizacji ogólnego przeznaczenia, jaką w środowisku MATLAB jest procedura **fminsearch**. Metodyka postępowania w tym przypadku zostanie pokazana na przykładzie.

Przykład 6.9. Zmierzono następujący ciąg wartości modułu impedancji $z(f)$ mikrofalowego obwodu rezonansowego:

$$\tilde{z}_n \approx z(f_n) \quad \text{dla } n=1, \dots, N$$

gdzie f_n jest wartością częstotliwości, dla której wykonano n -ty pomiar. Wyznaczyć częstotliwość rezonansową f_0 i dobroć Q obwodu rezonansowego oraz wartość jego impedancji dla częstotliwości rezonansowej – z_0 . Dane pomiarowe obwodu o parametrach $f_r = 10^9 \text{ Hz}$, $Q = 5000$, $z_0 = 1$, były symulowane na podstawie obliczeń wartości impedancji wyliczonej z wzoru (6.60) w 100 równoodległych punktach z zakresu częstotliwości od $0.9995 \cdot 10^9 \text{ Hz}$ do $1.0005 \cdot 10^9 \text{ Hz}$, przez obarczenie tych wartości multiplikatywnymi błędami przypadkowymi (generator liczb pseudolosowych **rand**) w granicach $\pm 4.5\%$.

Rozwiążanie tego zadania polega na aproksymacji ciągu danych pomiarowych następującą funkcją modelującą obwód rezonansowy:

$$\hat{z}(f; f_0, Q, z_0) = \frac{z_0}{\sqrt{1 + Q^2 \left(\frac{f}{f_0} - \frac{f_0}{f} \right)^2}} \quad (6.60)$$

przez minimalizację funkcjonału:

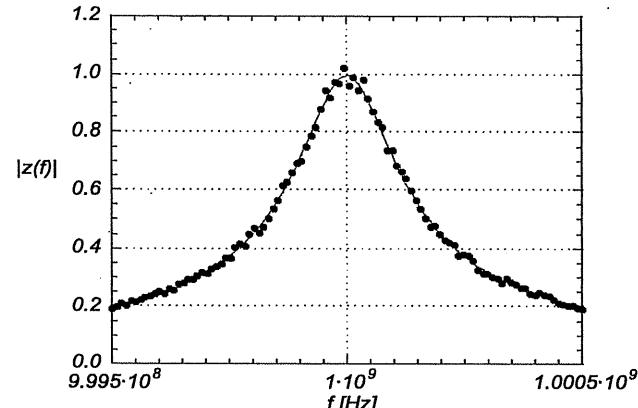
$$J_2(\mathbf{p}) = \sum_{n=1}^N \left[\hat{z}(f_n; \mathbf{p}) - \tilde{z}_n \right]^2 \quad (6.61)$$

gdzie $\mathbf{p} = [f_0 \ Q \ z_0]^T$. Do minimalizacji $J_2(\mathbf{p})$ w środowisku MATLAB użyto procedury **fminsearch**. W najprostszej wersji tej procedury wektor parametrów \mathbf{p} , najlepiej dopasowanych do danych pomiarowych, uzyskuje się przez wywołanie: $\mathbf{p} = \text{fminsearch}(@\text{fun}, \mathbf{p}^{(0)})$, gdzie: *fun* oznacza zapisaną w kodzie MATLABa funkcję (6.61), a $\mathbf{p}^{(0)}$ jest przybliżeniem początkowym wektora parametrów. Przy odpowiednim wyborze punktu startowego proces poszukiwania minimum jest zbieżny do:

$$\mathbf{p}^{(\infty)} = [f_r^{(\infty)} \ Q^{(\infty)} \ z_0^{(\infty)}]^T = [1.00000 \cdot 10^9 \ 5007.0 \ 0.99744]^T$$

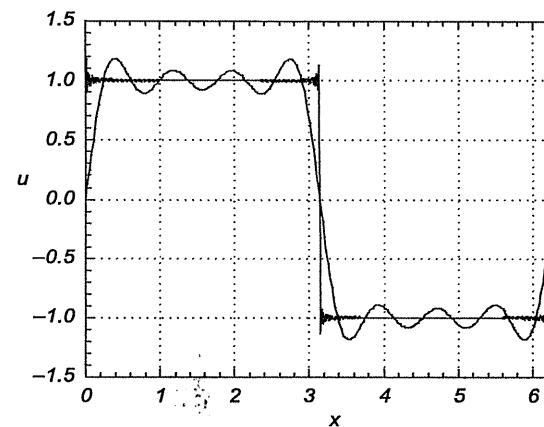
Na rys. 6.12 przedstawiono wyniki obliczeń. Jak widać, parametry obwodu rezonansowego wyznaczone na podstawie znacznie zaburzonych danych pomiarowych są obarczone stosunkowo niewielkimi błędami (w stosunku do modelu niezaburzonego). Wynika to ze stosunkowo dużej liczby danych pomiarowych i uśredniania błędów losowych. Można stąd wyciągnąć ogólny wniosek (który był

już sformułowany wcześniej dla zagadnień liniowych w rozdziale 4), że na podstawie wielu danych pomiarowych charakteryzujących obiekt fizyczny, obarczonych nawet znacznymi błędami, można wyznaczyć parametry tego obiektu z zadowalającą dokładnością – na ogół tym lepszą, im więcej jest danych pomiarowych.



Rysunek 6.12. Dane pomiarowe charakteryzujące mikrofalowy obwód rezonansowy (●) oraz wynik ich aproksymacji (linia czerwona) za pomocą funkcji (6.60), której parametry wyznaczono z wykorzystaniem procedury **fminsearch**

Zadanie 6.1. Wyznaczyć szereg Fouriera aproksymujący prostokątny przebiegu napięcia o okresie równym 2π i jednostkowej amplitudzie. Oliczenia przeprowadzić dla $K = 8$ i $K = 256$; wyniki obliczeń przedstawić na wykresie. Odpowiedź przedstawiono na rys. 6.13.



Rysunek 6.13. Wynik aproksymacji prostokątnego przebiegu napięcia za pomocą szeregu Fouriera dla $K = 8$ (linia niebieska) i $K = 256$ (linia czerwona)

CAŁKOWANIE I RÓZNICZKOWANIE FUNKCJI – METODY KLASYCZNE

W rozdziale tym omówione zostaną najważniejsze niestatystyczne metody przybliżonego obliczania całek oznaczonych oraz pochodnych funkcji rzeczywistych jednej i wielu zmiennych.

Do zadania numerycznego wyznaczania całki prowadzą, między innymi, następujące problemy inżynierskie:

- obliczanie powierzchni lub objętości obszaru o zadanym brzegu;
- obliczanie masy, współrzędnych środków ciężkości niejednorodnych i niregularnych brył;
- obliczanie mocy wydzielonej w dwójniku elektrycznym w zadanym przedziale czasu (por. przykład 1.1), obliczenia napięcia skutecznego lub ładunku zgromadzonego w pojemności nieliniowej;
- obliczanie drogi przebytej przez obiekt poruszający się ze zmienną prędkością.

Warto zauważyć, że całkę oznaczoną funkcji czasu można traktować jako rozwiązanie tzw. zagadnienia początkowego dla pewnego równania różniczkowego. W rozdziale 9 przedstawiono specjalne metody rozwiązywania tego typu zadań.

Różniczkowanie numeryczne jest przydatne przy rozwiązywaniu następujących zadań technicznych:

- analizie wrażliwościowej systemu fizycznego lub algorytmu numerycznego (por. rozdz. 2.4);
- realizacji algorytmu wymagającego użycia pochodnej funkcji znanych niejawnie (np. będących rozwiązaniem nieliniowego układu równań algebraicznych lub układu równań różniczkowych).

7.1. CAŁKOWANIE FUNKCJI JEDNEJ ZMIENNEJ

Zajmiemy się obliczaniem całki funkcji $f(x)$ z wagą $w(x)$ w przedziale $[a, b]$:

$$I(f) = \int_a^b w(x)f(x)dx \quad (7.1)$$

przy założeniu, że funkcja wagowa $w(x)$ jest nieujemna i całkowalna w tym przedziale, a zeruje się tylko w skończonej liczbie punktów tego przedziału. Całka $I(f)$ będzie wyznaczona na podstawie wartości całkowanej funkcji $f(x)$ w odpowiednio dobranych punktach (węzłach):

$$\{x_n\} \equiv \{x_n \mid n = 0, 1, \dots, N\}$$

przy użyciu formuły zwanej kwadraturą liniową:

$$\hat{I}_N(f) = \sum_{n=0}^N A_n f(x_n) \quad (7.2)$$

Współczynniki A_n i węzły x_n tej kwadratury powinny być dobrane tak, aby błąd przybliżenia całki przez kwadraturę, zwany resztą kwadratury:

$$R_N(f) = \hat{I}_N(f) - I(f) \quad (7.3)$$

był możliwie mały – dla ustalonej wagi $w(x)$ i przedziału całkowania $[a, b]$. Przede wszystkim jednak kwadratura powinna być zbieżna, tzn. umożliwiać (dla danej klasy funkcji f) osiągnięcie dowolnie małego błędu przy wykorzystaniu dostatecznie dużej liczby węzłów, tzn.

$$R_N(f) \xrightarrow{N \rightarrow \infty} 0$$

dla normalnego ciągu podziałów przedziału całkowania $[a, b]$ na podprzedziały $[x_{n-1}, x_n]$ (tj. dla takiego wyboru węzłów, że $\max_{n=1, \dots, N} \{x_n - x_{n-1}\} \xrightarrow{N \rightarrow \infty} 0$).

Szybkość tej zbieżności charakteryzuje rząd kwadratury. Kwadratura jest rzędu p , jeżeli jest dokładna dla wszystkich wielomianów stopnia niższego niż p , ale nie dla wszystkich wielomianów stopnia p .

Podstawowym sposobem tworzenia kwadratur jest całkowanie funkcji interpolującej funkcję $f(x)$ – najczęściej wielomianu interpolacyjnego Lagrange'a lub funkcji sklejanej.

7.1.1. PROSTE KWADRATURY INTERPOLACYJNE NEWTONA-COTESA

Kwadratury Newtona-Cotesa służą do przybliżonego wyznaczania całki (7.1) z funkcją wagową $w(x) \equiv 1$. Mają one postać $\hat{I}_N(f) = I(L_N)$, gdzie L_N jest wielomianem interpolacyjnym Lagrange'a zbudowanym dla funkcji $f(x)$, opartym na $(N+1)$ równoodległych węzłach $x_n = a + nh$, $n = 0, 1, \dots, N$, $h = (b - a) / N$. Ponieważ wielomian ten ma postać:

$$L_N(x) = \sum_{n=0}^N f(x_n) \prod_{v=0, v \neq n}^N \frac{x - x_v}{x_n - x_v} \quad (7.4)$$

w wyniku całkowania otrzymuje się kwadraturę liniową o współczynnikach:

$$\begin{aligned} A_{n,N} &= \int_a^b \prod_{v=0, v \neq n}^N \frac{x - x_v}{x_n - x_v} dx = \left| \begin{array}{l} z = (x - a) / h \\ dx = h dz \end{array} \right| = \\ &= (b - a) \frac{1}{N} \int_0^N \prod_{v=0, v \neq n}^N \frac{z - v}{n - v} dz = (b - a) \frac{l_{n,N}}{m_N} \end{aligned} \quad (7.5)$$

których wartości można wyznaczyć, korzystając z tablicy 7.1. Kwadraturę $\hat{I}_N(f)$ można więc zapisać w postaci:

$$\hat{I}_N(f) = \sum_{n=0}^N A_{n,N} f(x_n) = (b - a) \sum_{n=0}^N \frac{l_{n,N}}{m_N} f(x_n) \quad (7.6)$$

Jeżeli funkcja f ma ciągłe pochodne rzędu p_N w przedziale całkowania (tzn. $f \in C_{[a,b]}^{p_N}$), to reszta kwadratury ma postać:

$$R_N(f) = C_N h^{p_N+1} f^{(p_N)}(\xi) \quad (7.7)$$

gdzie $\xi \in [a, b]$. Wartości stałej C_N podano w tablicy 7.1. Rząd kwadratury jest równy p_N , ponieważ $f^{(p_N)}(x) \equiv 0$ dla wielomianów stopnia nie wyższego niż $(p_N - 1)$. Można pokazać, że $p_N \geq N + 1$ [F2].

Tablica 7.1

Parametry wybranych kwadratur Newtona-Cotesa

N	$l_{n,N}$	m_N	p_N	C_N	Kwadratura
1	1, 1	2	2	1/12	trapezów
2	1, 4, 1	6	4	1/90	Simpsona
3	1, 3, 3, 1	8	4	3/80	trzech ósmych
4	7, 32, 12, 32, 7	90	6	8/945	Milne'a
5	19, 75, 50, 50, 75, 19	288	6	275/12096	Bode'a
6	41, 216, 27, 272, 27, 216, 41	840	8	9/1400	Weddle'a

Przykład 7.1. Dwupunktową ($N = 1$) kwadraturę Newtona-Cotesa, zwaną kwadraturą trapezów (rys. 7.1), można uzyskać przez całkowanie w przedziale $[a, b]$ wielomianu pierwszego stopnia, interpolującego funkcję podcałkową $f(x)$:

$$L_1(x) = f(a) \frac{x - b}{a - b} + f(b) \frac{x - a}{b - a}$$

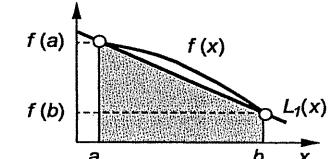
Kwadratura trapezów ma postać:

$$\hat{I}(f) = \int_a^b L_1(x) dx = \int_a^b \left[f(a) \frac{x - b}{a - b} + f(b) \frac{x - a}{b - a} \right] dx = \frac{f(a) + f(b)}{2} (b - a) \quad (7.8)$$

Błąd bezwzględny tej kwadratury:

$$R_1(f) = \hat{I}_N(f) - I(f) = \int_a^b [L_1(x) - f(x)] dx$$

Rysunek 7.1. Ilustracja konstrukcji dwupunktowej kwadratury Newtona-Cotesa



Jeśli funkcja podcałkowa ma w przedziale $[a, b]$ ciągłą pochodną rzędu $N + 1$, to błąd interpolacji wielomianem N -tego stopnia wyraża się wzorem:

$$L_N(x) - f(x) = -\frac{1}{(N+1)!} \prod_{n=0}^N (x - x_n) f^{(N+1)}(\xi)$$

gdzie $\xi \in [a, b]$. Dla $N = 1$

$$R_1(f) = -\frac{1}{2} \int_a^b (x - a)(x - b) f^{(2)}(\xi) dx = \frac{(b - a)^3}{12} f^{(2)}(\xi) = \frac{1}{12} h^3 f^{(2)}(\xi) \quad (7.9)$$

gdzie $h = b - a$. Kwadratura jest rzędu $p = 2$, bo jedynie dla wielomianów stopnia $n \leq 1$ zachodzi równość $f^{(2)}(\xi) = 0$ dla każdej wartości ξ , a reszta $R_1(f) \equiv 0$; dla wielomianów wyższego stopnia reszta $R_1(f)$ może być niezerowa, ponieważ druga pochodna $f^{(2)}(\xi)$ nie jest równa zeru w całym przedziale całkowania.

Przykład 7.2. Dla porównania dokładności całkowania numerycznego za pomocą prostych kwadratur Newtona-Cotesa różnych rzędów wyznaczono przybliżone wartości całek postaci:

$$I_i = \int_{-1}^1 f_i(x) dx \quad \text{dla } k = 1, 2, 3, 4$$

dla następujących funkcji testowych (w nawiasach podano przybliżone wartości całek):

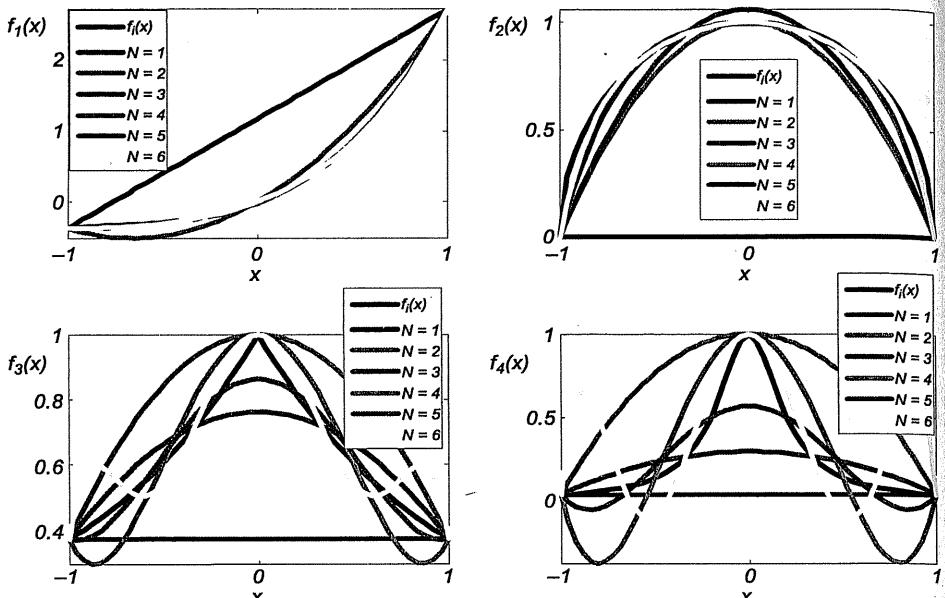
$$f_1(x) = x \exp(x) \quad (I_2 = 2 / e \approx 0.73575888)$$

$$f_2(x) = \sqrt{1 - x^2} \quad (I_2 = \pi / 2 \approx 1.5707963)$$

$$f_3(x) = |\exp(-|x|)| \quad (I_3 = 2 - 2 / e \approx 1.264241118)$$

$$f_4(x) = \frac{1}{1 + 25x^2} \quad (I_4 = 0.4 \arctg(5) \approx 0.54936030678)$$

Na rys. 7.2 przedstawiono wykresy całkowanych funkcji oraz wielomianów interpolacyjnych stopnia od 1 do 6, użytych do konstrukcji kwadratur Newtona-Cotesa (o współczynnikach wyznaczonych na podstawie tablicy 7.1). W tablicy 7.2 zamieszczone zostały uzyskane wartości błędów względnych tych kwadratur.



Rysunek 7.2. Wykresy funkcji podcałkowych z przykładu 7.2 oraz wielomianów interpolujących stopni $N = 1, 2, \dots, 6$ ($f_i(x)$ oznacza funkcję podcałkową)

Tablica 7.2

Błędy względne $\delta[\hat{I}_i] = (\hat{I}_i - I_i) / I_i$ prostych kwadratur Newtona-Cotesa dla funkcji testowych z przykładu 7.2

i	N					
	1	2	3	4	5	6
1	219.5%	6.484%	2.937%	0.05653%	0.032%	0.00037%
2	-100%	-15.12%	-9.968%	-4.612%	-3.632%	-2.248%
3	-41.80%	24.86%	-0.4354%	-1.622%	-2.129%	5.965%
4	-86.00%	147.4%	-24.22%	-13.57%	-15.99%	40.91%

Porównując wykresy na rys. 7.2 z wynikami zamieszczonymi w tablicy 7.2, można zauważać bezpośredni związek między błędem kwadratur a błędem interpolacji całkowanej funkcji wielomianami użytymi do konstrukcji kwadratury. Zauważmy przy tym, że źródła niedokładności interpolacji są różne dla każdej

z funkcji: funkcja f_2 ma nieskończoną pochodną na krańcach przedziału całkowania, a f_3 jest nieróżniczkowalna dla $x = 0$; z kolei dla funkcji f_4 wielomiany interpolacyjne wykazują oscylacje w pobliżu krańców przedziału $[-1, 1]$ – rosnące ze stopniem wielomianu (tzw. efekt Rungego – por. rozdz. 6.1).

7.1.2. ZŁOŻONE KWADRATURY INTERPOLACYJNE NEWTONA-COTESA

Zwiększanie liczby węzłów ($N + 1$) prostych interpolacyjnych kwadratur Newtona-Cotesa może zwiększyć dokładność całkowania – jednak tylko wówczas, gdy wzrost stopnia wielomianu interpolującego zwiększa dokładność przybliżenia funkcji podcałkowej (np. nie występuje efekt Rungego). Jeżeli uzyskana w ten sposób dokładność jest niewystarczająca, warto rozważyć zastosowanie złożonej kwadratury Newtona-Cotesa. Dzieli się wówczas przedział całkowania $[a, b]$ na M równych podprzedziałów $[x_m, x_{m+1}]$ dla $m = 0, 1, \dots, M - 1$, gdzie $x_m = a + mH$ oraz $H = (b - a) / M$; w każdym z nich stosuje się proste kwadratury Newtona-Cotesa $\hat{I}_i(f)$ (najczęściej niskiego rzędu – wzory trapezów lub Simpsona) i sumuje uzyskane przybliżenia:

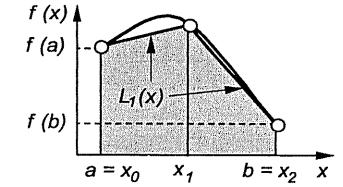
$$I(f) = \int_{a=x_0}^{b=x_N} f(x) dx = \sum_{m=0}^{M-1} \hat{I}_m(f) \approx \hat{I}_{M,N}(f) = \sum_{m=0}^{M-1} \hat{I}_m(f) \quad (7.10)$$

Przykład 7.3. Dwupunktowa kwadratura trapezów odpowiada interpolacji funkcji podcałkowej za pomocą wielomianu pierwszego stopnia (rys. 7.1). Podzielmy przedział całkowania na M równych podprzedziałów $[x_m, x_{m+1}]$ (na rys. 7.3: $M = 2$). Cały całkowany można przybliżyć za pomocą wzoru trapezów (7.8), tj.:

$$I_m = \int_{x_m}^{x_{m+1}} f(x) dx \approx \hat{I}_m = \frac{H}{2} [f(x_m) + f(x_{m+1})]$$

gdzie $H = x_{m+1} - x_m$. Złożona kwadratura trapezów, przybliżająca całkę, ma więc ostatecznie postać:

$$\hat{T}_M(f) = \frac{H}{2} \left(f(a) + 2 \sum_{m=1}^{M-1} f(a + mH) + f(b) \right) \quad (7.11)$$



Rysunek 7.3. Złożona ($M = 2$) kwadratura trapezów

Reszta kwadratury $R_{M,1}(f) = \hat{I}_{M,1}(f) - I(f)$ jest sumą reszt cząstkowych kwadratur (7.9), tj.

$$R_{M,1}(f) = -\frac{H^3}{12} \sum_{m=0}^{M-1} f^{(2)}(\xi_m) \quad (7.12)$$

gdzie $\xi_m \in [x_m, x_{m+1}]$. Jeżeli całkowana funkcja ma ciągłą drugą pochodną w przedziale $[a, b]$, to

$$R_{M,1}(f) = \left(\frac{b-a}{M}\right)^3 \frac{1}{12} \sum_{m=0}^{M-1} f^{(2)}(\xi_m) = \frac{(b-a)^3}{12M^2} f^{(2)}(\xi) \quad (7.13)$$

gdzie $\xi \in [a, b]$. Złożona kwadratura trapezów ma więc ten sam rząd ($p = 2$) co kwadratura prosta.

Resztę kwadratury złożonej dla $N > 1$ można oszacować podobnie jak resztę dla wzoru trapezów:

$$R_{M,N}(f) = MC_N h^{p_N+1} f^{(p_N)}(\xi) = C_N f^{(p_N)}(\xi)(b-a)^{p_N+1} M(MN)^{-(p_N+1)} \quad (7.14)$$

przy założeniu, że funkcja f ma w przedziale całkowania ciągłą p_N -tą pochodną.

Zauważmy, że złożoną kwadraturę (w przykładzie 7.3: kwadraturę trapezów) można uważać za kwadraturę interpolacyjną (prostą w szerszym sensie). Powstaje ona bowiem przez całkowanie interpolacyjnego przybliżenia funkcji podcałkowej za pomocą funkcji sklejanej (w przykładzie 7.3: funkcji odcinkowo-liniowej).

Przykład 7.4. W przykładzie 1.1 opisano metodę wyznaczania średniej mocy wydzielanej w nieliniowym obciążeniu na podstawie pomiarów chwilowych wartości napięcia $u(t)$ i prądu $i(t)$ w przedziale czasu $t \in [0, T]$. Polega ona na całkowaniu funkcji interpolującej wartości mocy chwilowej, $\{t_m, u_m, i_m | m = 1, 2, \dots, M\}$, za pomocą pewnej metody numerycznej (w przykładzie 1.1 użyto metody prostokątów). Niech $i = \hat{F}(u)$ będzie odcinkowo-liniową interpolacją charakterystyki statycznej $i = F(u)$ wyznaczoną na podstawie danych $\{t_m, u_m, i_m | m = 0, 1, \dots, M\}$, gdzie $u_m = u(t_m)$ oraz $i_m = i(t_m)$. Przybliżoną wartość mocy średniej za okres $[0, T]$ można wówczas wyznaczyć następująco:

$$\begin{aligned} P &\equiv \frac{1}{T} \int_0^T u(t) i(t) dt \approx \hat{P} \equiv \frac{1}{T} \int_0^T u(t) \hat{F}[u(t)] dt = \\ &= \frac{1}{T} \sum_{m=0}^{M-1} \int_{t_m}^{t_{m+1}} u(t) \hat{F}[u(t)] dt = \frac{1}{T} \sum_{m=0}^{M-1} \int_{t_m}^{t_{m+1}} f_m(t) dt \end{aligned}$$

gdzie $f_m(t) \equiv u(t) \hat{F}[u(t)]$ dla $t \in [t_m, t_{m+1}]$. Postać całek składowych

$$\hat{P}_m \equiv \int_{t_m}^{t_{m+1}} f_m(t) dt$$

można wyznaczyć, korzystając z warunków interpolacji:

$$\hat{F}(u) = i_m + g_m(u - u_m)$$

gdzie $g_m \equiv \frac{i_{m+1} - i_m}{u_{m+1} - u_m}$ dla $t \in [t_m, t_{m+1}]$. Zakładając, że $u(t) = \sin(t)$, otrzymuje się:

$$\hat{P}_m = \int_{t_m}^{t_{m+1}} [i_m + g_m(\sin(t) - u_m)] \sin(t) dt =$$

$$= (g_m u_m - i_m) [\cos(t_{m+1}) - \cos(t_m)] + \frac{g_m}{2} (t_{m+1} - t_m) - \frac{g_m}{4} [\sin(2t_{m+1}) - \sin(2t_m)]$$

Przybliżoną wartość mocy średniej można więc obliczyć jako: $\hat{P} = \sum_{m=0}^{M-1} \hat{P}_m$. Zaproponowana tu procedura przybliżonego całkowania jest w istocie realizacją złożonej kwadratury trapezów.

Całkowanie za pomocą kwadratur złożonych wymaga podjęcia decyzji o liczbie podprzedziałów M . Racjonalne postępowanie może polegać na maksymalizacji dokładności uzyskanej kwadratury złożonej przy zadanej liczbie węzłów bądź też minimalizacji liczby węzłów wykorzystanych przez kwadraturę o złożonej dokładności. Rozwiążanie tak postawionych problemów nie jest jednak łatwe, o czym przekonują eksperymenty numeryczne przedstawione w następującym przykładzie.

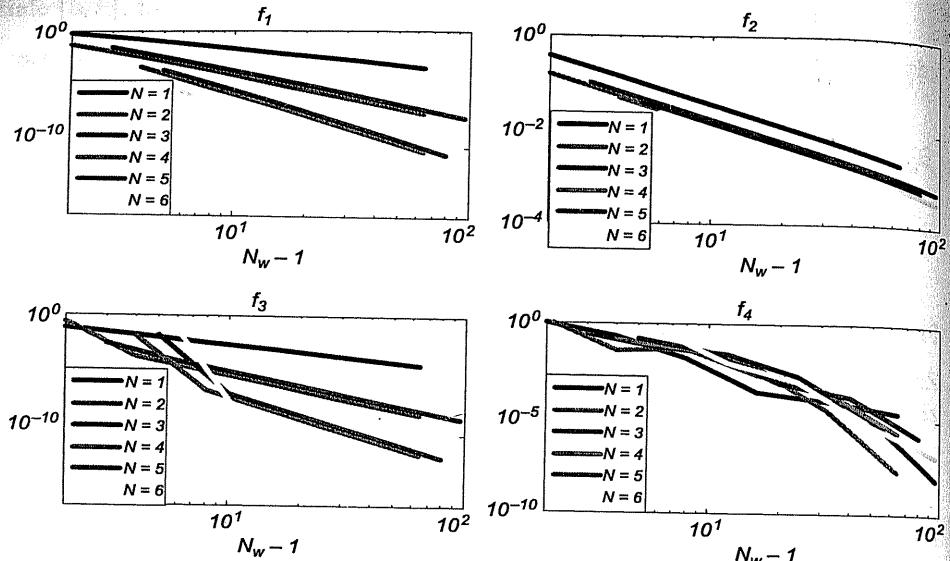
Przykład 7.5. Przybliżone wartości całek czterech funkcji z przykładu 7.2 zostały obliczone za pomocą złożonych kwadratur Newtona-Cotesa o różnej liczbie podprzedziałów M i różnej liczbie punktów wykorzystywanych przez kwadraturę w każdym podprzedziale ($N+1$).

Na rys. 7.4 wykreślono zależność błędów względnych tych kwadratur od liczby węzłów $N_w = NM + 1$. Widać, że zależność błędów kwadratury od liczby węzłów N_w i stopnia interpolacji N jest dla każdej funkcji inna. Z dotychczasowych rozważań wiadomo, że błąd kwadratury można oszacować w przypadku, gdy funkcja podcałkowa ma w przedziale całkowania ciągłą pochodną rzędu p_N . Wówczas, zgodnie z (7.14), mamy:

$$|\delta[\hat{I}(f_i)]| = \left| \frac{\hat{I}(f_i) - I(f_i)}{I(f_i)} \right| \leq \frac{C_N f_i^{(p_N)} (\xi)(b-a)^{p_N+1}}{I(f_i)} \frac{1}{N(N_w-1)^{p_N}} \equiv \delta_{\max}$$

Po zlogarytmowaniu nierówność ta przybiera postać:

$$\log(|\delta[\hat{I}(f_i)]|) \leq \log(K_N) - \log(N) - p_N \log(N_w - 1) \quad (7.15)$$



Rysunek 7.4. Zależność błędu względnego $|\delta[\hat{I}(f_i)]|$ złożonych kwadratur Newtona-Cotesa od $N_w - 1$ dla funkcji f_1, f_2, f_3 i f_4 z przykładu 7.5; N oznacza stopień wielomianu interpolującego (skojarzonego z kwadraturą), a N_w jest liczbą wykorzystanych węzłów

Prawa strona tego oszacowania dla ustalonej wartości N opisuje prostą o nachyleniu $-p_N$ (jeśli jako zmienną niezależną wybrać $\log(N_w - 1)$). Taką zależność błędu od $\log(N_w - 1)$ wykazują kwadratury dla funkcji f_2 – co nie dziwi, gdyż funkcja ta ma nieskończonie wiele pochodnych i wyprowadzone oszacowania są poprawne. Dla tej funkcji największą dokładność uzyskuje się dla kwadratur złożonych z możliwie dużą wartością parametru N . Natomiast funkcje f_3 i f_4 nie mają pochodnych we wszystkich punktach przedziału całkowania; w konsekwencji oszacowanie (7.15) nie jest poprawne. Pomimo to kwadratury dla f_3 wykazują asymptotycznie zachowanie podobne do obserwowanego dla funkcji f_2 , gdyż funkcja podcałkowa f_3 jest tak gładka jak f_2 w obydwóch podprzedziałach $[-1, 0]$ i $[0, 1]$. Tak więc dla każdego podziału przedziału całkowania, w którym punkt $x=0$ jest węzłem, wszystkie całki cząstkowe funkcji f_3 spełniają wa-

runki oszacowania (7.7), tak jak dla funkcji f_2 . Funkcja f_4 , jakkolwiek gładka, nie jest dobrze interpolowana za pomocą wielomianów wysokiego stopnia z powodu efektu Rungego; dlatego do jej całkowania trzeba wykorzystać wielomiany interpolacyjne niskiego stopnia. Z obliczeń wynika, że najlepsza okazuje się kwadratura z $N=3$, ale taki wynik nie jest łatwy do przewidzenia. Zauważmy też, że kształt wykresów dla f_4 jest inny niż dla pozostałych funkcji; najwidoczniej oszacowanie (7.15) jest zbyt niedokładne ze względu na dużą zmienność pochodnych f_4 w obszarze całkowania.

Oszacowanie z góry liczby przedziałów M , wystarczającej do uzyskania przybliżenia całki zadaną dokładnością bezwzględną ΔI_{\max} jest zazwyczaj niemożliwe ze względu na nieznajomość oszacowania wartości pochodnych $f^{(r)}(\xi)$, a czasem brak pewności co do gładkości funkcji podcałkowej, która może być „poznana” jedynie przez wyznaczenie jej wartości dla skończonej liczby wartości argumentu. Zalecanym sposobem postępowania jest wówczas obliczanie kwadratur złożonych $\hat{I}_{M,N}(f)$ dla zwiększającej się liczby podziałów przedziału całkowania $M_0 < M_1 < M_2 < \dots$, przy czym $M_i = K^i$, tzn. dzieli się poprzednie podprzedziały na K części, a K dobiera tak, aby wykorzystać poprzednio obliczone wartości funkcji podcałkowej. Podział prowadzi się dotąd, aż względna różnica wartości kolejnych przybliżeń $\hat{I}_{M,N}(f)$ stanie się mniejsza niż ΔI_{\max} . Lepszą efektywność obliczeń można uzyskać, jeśli podziałowi podlegają nie wszystkie podprzedziały całkowania, a jedynie te, dla których nie uzyskano jeszcze odpowiedniej dokładności kwadratury cząstkowej.

Przykład 7.6. Rozważmy całkowanie za pomocą złożonej kwadratury Simpsona. Pierwsze przybliżenie całki uzyskuje się za pomocą kwadratury prostej:

$$I = \int_a^b f(x) dx \approx I_1 = \frac{b-a}{6} [f(a) + 4f(c) + f(b)]$$

gdzie $c = (a+b)/2$. Następnie dzieli się podprzedziały $[a,c]$ oraz $[c,b]$ na połowy, uzyskując punkty $d = (a+c)/2$ i $e = (b+c)/2$. W podprzedziałach $[a,c]$ i $[c,b]$ wyznacza się przybliżone wartości całek:

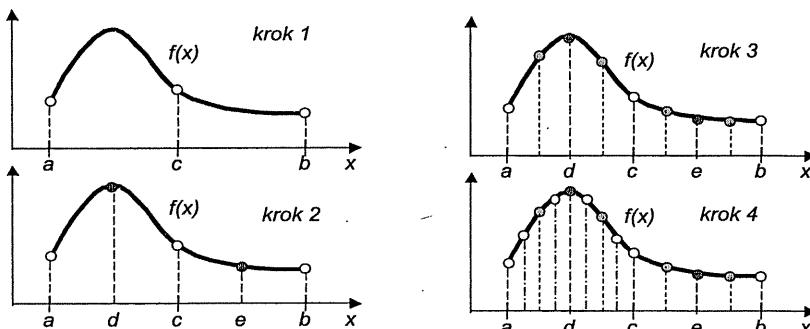
$$I_1^{[a,c]} = \int_a^{c} f(x) dx \approx \hat{I}_1^{[a,c]} = \frac{c-a}{6} [f(a) + 4f(d) + f(c)]$$

$$I_1^{[c,b]} = \int_c^b f(x) dx \approx \hat{I}_1^{[c,b]} = \frac{b-c}{6} [f(b) + 4f(e) + f(c)]$$

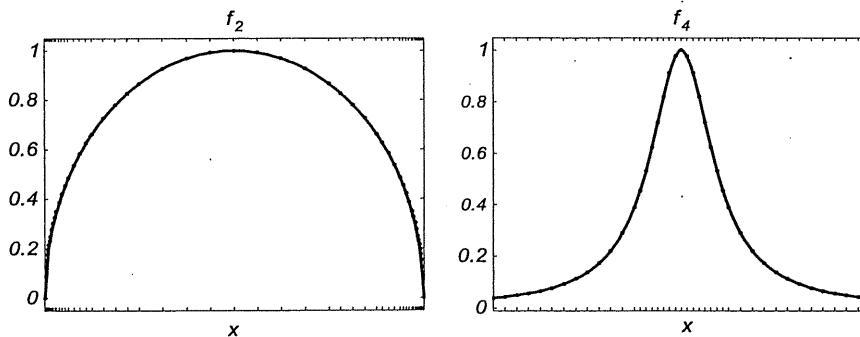
Wynika stąd drugie przybliżenie całki I o postaci:

$$\hat{I}_2 = \hat{I}_1^{[a,c]} + \hat{I}_1^{[c,b]} = \frac{b-a}{12} [f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)]$$

Jeśli $|\hat{I}_1 - \hat{I}_2| < \Delta I_{\max}$ (warunek dokładności), to \hat{I}_2 można przyjąć za przybliżenie całki I . W przeciwnym przypadku całkę I rozbijamy na dwie całki $I^{[a,c]}$ i $I^{[c,b]}$, tzn. $I = I^{[a,c]} + I^{[c,b]}$, zdefiniowane odpowiednio w przedziałach $[a,c]$ oraz $[c,b]$. Dla każdej z tych całek stosowana jest powyższa procedura podziału – aż do uzyskania dla wszystkich całek cząstkowych warunku dokładności (por. rys. 7.5). Wynik zastosowania opisanej procedury pokazano na rys. 7.6 dla funkcji f_2 i f_4 z przykładu 7.3. Widać, że siatka węzłów jest nierównomierna; są one zagęszczone tam, gdzie funkcja ma dużą zmienność. Warto dodać, że przy założonej wartości parametru $\Delta I_{\max} = 10^{-4}$ uzyskano błąd względny ok. $-4.3 \cdot 10^{-5}$ dla f_2 oraz $1.6 \cdot 10^{-5}$ dla f_4 , przy czym liczba węzłów wyniosła, odpowiednio, 57 i 41.



Rysunek 7.5. Ilustracja adaptacyjnego doboru węzłów złożonej kwadratury Simpsona



Rysunek 7.6. Rozkład węzłów całkowania procedury z przykładu 7.6 dla funkcji f_2 i f_4

7.1.3. KWADRATURY INTERPOLACYJNE GAUSSA

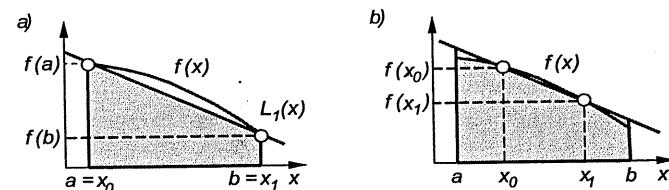
Dokładność kwadratur interpolacyjnych postaci (7.2) można poprawić, rezygnując z założenia, że funkcja wagowa $w(x) \equiv 1$, a węzły kwadratury są równoodległe.

Przykład 7.7. Wyznaczyć pole pod wykresem funkcji:

$$y = f(x) \text{ dla } [-1, 1], \quad a = -1, \quad b = 1$$

wiedząc, że wartości tej funkcji są dodatnie.

Z wykresów pokazanych na rys. 7.7 wynika, że błąd metody trapezów jest znaczny, jeśli funkcja podcałkowa jest silnie wypukła bądź wklęsła. Błąd przybliżenia pola powierzchni pod krzywą przez trapez maleje, gdy węzły leżą wewnątrz przedziału całkowania (rys. 7.7b).



Rysunek 7.7. Kwadratura: a) trapezów i b) dwupunktowa z węzłami wewnętrznych przedziału całkowania

Spróbujmy dobrać węzły x_0 i x_1 oraz współczynniki A_0 i A_1 dwupunktowej kwadratury liniowej

$$\hat{I}_{GL}(f) = A_0 f(x_0) + A_1 f(x_1)$$

w taki sposób, aby kwadratura ta była dokładna dla wielomianów do stopnia trzeciego włącznie. Biorąc pod uwagę addytywność całkowania, wystarczy sprawdzić jej dokładność dla $f(x) = 1, x, x^2, x^3$:

$$\int_{-1}^1 dx = 2 = A_0 + A_1,$$

$$\int_{-1}^1 x^2 dx = \frac{2}{3} = A_0 x_0^2 + A_1 x_1^2,$$

$$\int_{-1}^1 x dx = 0 = A_0 x_0 + A_1 x_1$$

$$\int_{-1}^1 x^3 dx = 0 = A_0 x_0^3 + A_1 x_1^3$$

Po rozwiązaniu powyższego układu równań otrzymujemy: $x_0 = -1/\sqrt{3}$, $x_1 = -x_0$, $A_0 = A_1 = 1$.

Dwupunktowa kwadratura uzyskana w przykładzie 7.7 jest rzędu 4, podczas gdy dwupunktowa kwadratura Newtona-Cotesa jest rzędu 2. Interesująca jest odpowiedź na pytanie o maksymalny rząd kwadratury liniowej (7.2) o współczynnikach:

$$A_n = \int_a^b w(x) \prod_{v=0, v \neq n}^N \frac{x - x_v}{x_n - x_v} dx \quad (7.16)$$

dla całki $I = \int_a^b w(x)f(x) dx$. Wiadomo [F2], że rząd ten jest równy co najmniej liczbie współczynników kwadratury, tj. $(N+1)$. Swoboda wyboru położenia węzłów zwiększa liczbę parametrów (x_n, A_n) określających kwadraturę (7.16) do $2(N+1)$, co daje potencjalną możliwość uzyskania kwadratury dokładnej dla wielomianów do stopnia $2N+1$ włącznie, czyli rzędu $p = 2N+2$. Okazuje się, że maksymalny rząd jest osiągalny dla węzłów x_n , które są pierwiastkami wielomianu $\psi_{N+1}(x)$ z ciągu wielomianów ortogonalnych w przedziale $[a, b]$ z wagą $w(x)$. Kwadraturę ze współczynnikami (7.16) o takim rozłożeniu węzłów nazywa się kwadraturą Gaussa. Wszystkie współczynniki takiej kwadratury są dodatnie; można je obliczyć za pomocą formuły [J1]:

$$A_n = \frac{a_{N+1}}{a_N} \frac{\|\psi_N\|^2}{\psi'_{N+1}(x_n) \psi_N(x_n)} \quad \text{dla } n = 0, \dots, N \quad (7.17)$$

gdzie a_N oznacza współczynnik przy najwyższej potędze argumentu wielomianu ortogonalnego $\psi_N(x)$, zaś a_{N+1} – współczynnik przy najwyższej potędze argumentu wielomianu $\psi_{N+1}(x)$. Reszta kwadratury Gaussa dla funkcji ciągłej pochodnej rzędu $2N+2$ w przedziale $[a, b]$ wyraża się wzorem:

$$R_N(f) = \frac{1}{(2N+2)!} f^{(2N+2)}(\xi) \int_a^b w(x) \prod_{n=0}^N (x - x_n) dx \quad (7.18)$$

gdzie $\xi \in [a, b]$. Maksymalny rząd nie jest jedyną zaletą kwadratury Gaussa. Dla dowolnej funkcji $f(x)$ ciągłej w przedziale $[a, b]$ i dla dowolnej funkcji wagowej $w(x) \geq 0$ ciąg $\hat{I}_N(f)$ kwadratur (7.2) ze współczynnikami (7.17) jest zbieżny do całki $I(f)$, gdy $N \rightarrow \infty$ [J1]. Kwadratury Newtona-Cotesa nie mają takiej własności.

Przykład 7.8. Wyprowadzić dwupunktowy wzór Gaussa dla całki

$$I(f) = \int_0^1 f(x) dx \quad (7.19)$$

Ponieważ funkcja wagowa $w(x) \equiv 1$, więc odpowiednim wielomianem ortogonalnym dla $N=1$ jest wielomian Legendre'a $P_2(x)$ zdefiniowany w przedziale $[-1, 1]$ (por. przykład 6.4). Wielomian ten można zastosować bezpośrednio do

konstrukcji kwadratury Gaussa (nazywanej też kwadraturą Gaussa-Legendre'a – dla podkreślenia roli ortogonalnych wielomianów Legendre'a w jej konstrukcji) o postaci:

$$\hat{I}(g) \approx \int_{-1}^1 g(x) dx \quad (7.20)$$

W celu obliczenia całki (7.19) należy więc dokonać liniowej zamiany zmiennej x w taki sposób, aby przedział $[0, 1]$ odwzorować na przedział $[-1, 1]$, a funkcję $f(x)$ na funkcję $g(x)$. Węzłami pomocniczej kwadratury (7.20) będą zera tego wielomianu Legendre'a $P_2(x) = (3x^2 - 1)/2$, tzn. $x_0 = -1/\sqrt{3}$ i $x_1 = 1/\sqrt{3}$. Współczynniki A_n można obliczyć ze wzoru (7.17). Biorąc pod uwagę, że:

$$P_1(x) = x, \quad \|P_1\|^2 = \int_{-1}^1 P_1^2(x) dx = \int_{-1}^1 x^2 dx = \frac{2}{3} \quad \text{oraz} \quad P_2'(x) = 3x$$

otrzymujemy $A_n = 1/3x_n^2$, a stąd $A_0 = A_1 = 1$. Pomocnicza kwadratura, przybliżająca całkę (7.20), ma więc postać:

$$\hat{I}(g) = g(-1/\sqrt{3}) + g(1/\sqrt{3})$$

Wykorzystując zamianę zmiennych: $y = 2x - 1$, można zapisać poszukiwaną całkę (7.19) jako:

$$I(f) = \int_0^1 f(x) dx = \int_{-1}^1 \frac{1}{2} f\left(\frac{y+1}{2}\right) dy = I(g) = \int_{-1}^1 g(y) dy$$

gdzie $g(x) = \frac{1}{2} f\left(\frac{x+1}{2}\right)$. Ostatecznie więc

$$\int_0^1 f(x) dx = \hat{I}_{GL}(f) = \frac{1}{2} \left[f\left(-\frac{1}{2\sqrt{3}} + \frac{1}{2}\right) + f\left(\frac{1}{2\sqrt{3}} + \frac{1}{2}\right) \right] \quad (7.21)$$



Podobnie jak złożone kwadratury Newtona-Cotesa, można budować złożone kwadratury Gaussa, dzieląc przedział całkowania na podprzedziały i stosując w każdym z podprzedziałów prostą kwadraturę Gaussa. Jeżeli zachodzi potrzeba zwiększenia dokładności kwadratury, należy zwiększyć liczbę podprzedziałów, tak jak dla kwadratur Newtona-Cotesa. Niestety, w przypadku kwadratury Gaussa nie można użyć wartości funkcji podcałkowej obliczonych dla mniejszej liczby podprzedziałów, gdyż węzły nie są równoodległe. W celu usunięcia tej niedogodności wprowadzane są modyfikacje powyższego sposobu zwiększania dokładności kwadratury Gaussa. Na przykład, dla kwadratury Gaussa-Kronroda [P] modyfikacje tworzone są przez dodanie $n+1$ węzłów do zbioru węzłów n -punktowej kwadra-

tury Gaussa w taki sposób, że uzyskuje się kwadraturę rzędu $3n+1$. Kwadratura Gaussa o takiej samej liczbie punktów ($2n+1 = N+1$) ma rzad $2(N+1) = 4n+2$, co oznacza, że możliwość użycia węzłów mniej dokładnej kwadratury przez kwadraturę wyższego rzędu odbyła się kosztem rzędu nowej kwadratury – jest niższy niż najwyższy możliwy.

Przykład 7.9. W tablicy 7.3 zestawiono względne błędy $\delta[\hat{I}_i]$ przybliżeń całek czterech funkcji testowych $f_i(x)$ ($i=1,\dots,4$) z przykładu 7.2, uzyskanych za pomocą prostych kwadratur Gaussa-Legendre'a o liczbie węzłów $N+1$, gdzie $N=1,\dots,6$.

Tablica 7.3

Porównanie względnych błędów całkowania $|\delta[\hat{I}_i]|$ funkcji z przykładu 7.2 za pomocą prostych kwadratur Gaussa

i	N					
	1	2	3	4	5	6
1	4.27%	0.054%	$3.2 \cdot 10^{-4}\%$	$1.1 \cdot 10^{-6}\%$	$2.6 \cdot 10^{-9}\%$	$7.7 \cdot 10^{-12}\%$
2	3.96%	1.33%	0.604%	0.325%	0.195%	0.126%
3	11.2%	10.8%	3.31%	4.34%	1.56%	2.32%
4	61.0%	74.4%	32.5%	28.7%	16.0%	12.2%

Porównując wyniki z tablicy 7.2 i tablicy 7.3 można zauważyć, że dla funkcji f_1 i f_2 dokładność prostych kwadratur Gaussa-Legendre'a jest istotnie większa niż dokładność kwadratur Newtona-Cotesa. Dla pozostałych funkcji błędy obydwu kwadratur są porównywalne i znaczne; świadczy to o małej przydatności prostych kwadratur do całkowania takich funkcji.

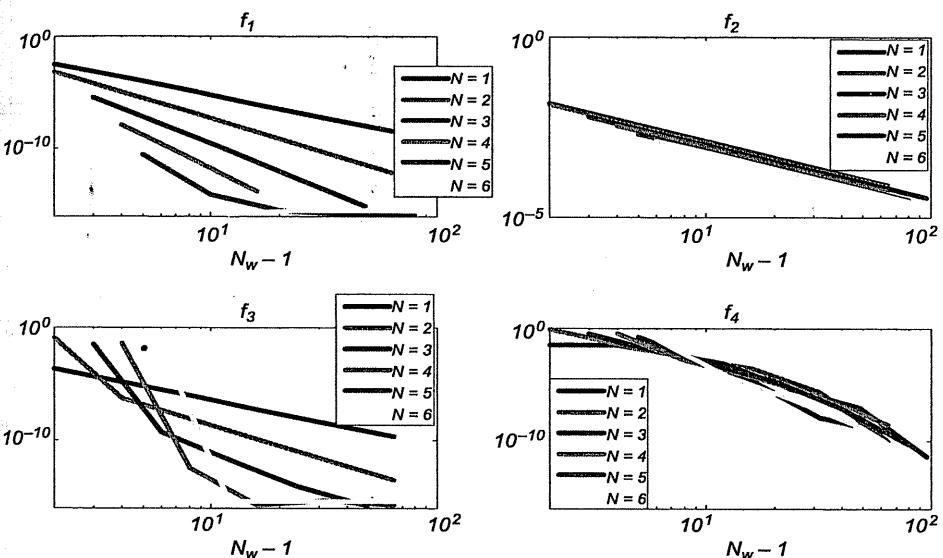
Wykonano również obliczenia dokładności całkowania numerycznego za pomocą złożonych kwadratur Gaussa-Legendre'a dla różnej liczby podprzedziałów całkowania M oraz różnej liczby punktów w każdym podprzedziale $N+1$. Dla ułatwienia porównań z wynikami uzyskanymi w przykładzie 7.5, na rys. 7.8 pokazano zależność błędu kwadratur złożonych od liczby węzłów:

$$N_w = (N+1)M$$

Dla gładkiej funkcji f_1 szybkość zmniejszania się błędu kwadratury ze wzrostem liczby węzłów N_w zależy od rzędu kwadratury; dlatego błąd kwadratur Gaussa maleje dużo szybciej niż błąd kwadratur Newtona-Cotesa o takiej samej liczbie węzłów. Dla funkcji f_2 (o nieograniczonej pochodnej na krańcach przedziału całkowania) kwadratury Gaussa są wprawdzie dokładniejsze, ale szybkość zmniejszania się błędu całkowania jest w przybliżeniu taka jak dla kwadratur Newtona-Cotesa i niezależna od liczby węzłów w podprzedziale całkowania

($N+1$). Podobnie jest dla funkcji f_4 (dla której występuje efekt Rungego). Błąd całkowania funkcji f_3 jest również mniejszy dla kwadratur Gaussa, a szybkość zbieżności jest – co może być pewnym zaskoczeniem – taka jak dla f_1 , mimo nieróżniczkowalności tej funkcji dla $x=1/2$. Jak widać z porównania, przewidywanie charakteru zbieżności kwadratury jest łatwe jedynie dla funkcji gładkich. Z tego względu procedury całkowania spotykane w bibliotekach numerycznych nie wyznaczają liczby węzłów na podstawie zadanej dokładności całkowania przed rozpoczęciem całkowania, ale dostosowują liczbę i położenie węzłów adaptacyjnie, posługując się lokalnym oszacowaniem dokładności całkowania (w podprzedziałach – por. przykład 7.6). Warto przy okazji zauważyć, że całka funkcji f_2 może być obliczona dokładnie (zakładając idealną arytmetykę), jeśli zastosowane będzie przekształcenie:

$$I(f_2) = \int_{-1}^1 \sqrt{1-x^2} dx = \int_{-1}^1 \frac{1}{\sqrt{1-x^2}} \underbrace{(1-x^2)}_{f(x)} dx = \int_{-1}^1 w(x) f(x) dx$$



Rysunek 7.8. Zależność względnego błędu całkowania złożonych kwadratur Gaussa-Legendre'a od $N_w - 1$; N oznacza stopień wielomianu interpolującego (skojarzonego z kwadraturą), a N_w jest liczbą wykorzystanych węzłów

Ponieważ wielomiany Czebyszewa (przykład 6.5) są ortogonalne w przedziale $[-1, 1]$ z funkcją wagową $w(x)$, więc powyższą całkę można obliczyć za pomo-

cą kwadratur Gaussa-Czebyszewa (patrz przykład 7.11). Rząd kwadratur Gaussa wyraża się wzorem $p = 2(N+1)$ i dlatego już dla $N+1=2$ węzłów całka będzie wyznaczona dokładnie.

7.1.4. PRZYSPIESZANIE ZBIEŻNOŚCI KWADRATUR METODĄ EKSTRAPOLACJI RICHARDSONA

Wzory określające reszty kwadratur mogą służyć nie tylko do szacowania ich dokładności, ale i do konstrukcji nowego ciągu kwadratur, szybciej zbieżnego do obliczanej całki. Założymy, że dla pewnego ciągu kwadratur zachodzi równość:

$$I(f) = \hat{I}_N(f) + \sum_{m=1}^M \frac{c_m}{N^{\alpha_m}} + o\left(\frac{1}{N^{\alpha_{M+1}}}\right) \quad (7.22)$$

gdzie α_m są dodatnimi wykładnikami uporządkowanymi według wartości: $0 < \alpha_1 < \alpha_2 < \dots < \alpha_{M+1}$, zaś stałe c_m zależą od całkowanej funkcji, ale nie zależą od N . Metoda ekstrapolacji Richardsona przyspieszania zbieżności kwadratury polega na wyznaczeniu takiej kombinacji liniowej dwóch kwadratur

$$\hat{I}_N^1(f) = \lambda \hat{I}_N + (1-\lambda) \hat{I}_{sN}$$

gdzie $\lambda \in (0,1)$ oraz $s \in \{2, 3, \dots\}$, aby odpowiadająca jej reszta nie zawierała składnika c_1 / N^{α_1} . Wynika stąd warunek:

$$\frac{\lambda}{N^{\alpha_1}} + \frac{1-\lambda}{(sN)^{\alpha_1}} = 0$$

z którego można obliczyć mnożnik:

$$\lambda = \frac{1}{1-s^{\alpha_1}} \quad (7.23)$$

Reszty tak utworzonego ciągu kwadratur $\{\hat{I}_N^1\}$ są rzedu $1/N^{\alpha_2}$, a nie $1/N^{\alpha_1}$. Postępowanie takie można kontynuować, tworząc kombinację liniową kwadratur \hat{I}_N^1 i \hat{I}_{sN}^1 .

Przykład 7.10. Całkę oznaczoną $I = \int_0^1 \sin(\pi x) dx$ obliczano numerycznie za pomocą złożonej kwadratury trapezów z $N+1$ węzłami (patrz przykład 7.3):

$$\hat{T}_N(f) = \frac{H}{2} \left[f(a) + 2 \sum_{n=1}^{N-1} f(a+nH) + f(b) \right]$$

gdzie $H = (b-a)/N$. Błąd tej kwadratury określa reszta (7.12):

$$R_N(f) = -\frac{b-a}{12} H^2 f^{(2)}(\xi)$$

gdzie $\xi \in [a, b]$, a więc kwadratura jest rzędu drugiego. Stosując do ciągu kwadratur $\{\hat{T}_N(f)\}$ ekstrapolację Richardsona z $s=2$ (tj. podwajanie liczby węzłów), otrzymujemy z równania (7.23) mnożnik $\lambda = 1/(1-s^{\alpha_1}) = -1/3$, ponieważ wykładnik $\alpha_1 = 2$. Wynika stąd nowy ciąg kwadratur postaci:

$$\hat{T}_N^1(f) = \frac{4}{3} \hat{T}_{2N}(f) - \frac{1}{3} \hat{T}_N(f) \quad \text{dla } N = 1, 2, \dots$$

Można pokazać, że jest to ciąg złożonych kwadratur Simpsona.

Tablica 7.4

Względne błędy kwadratur Romberga $\delta[\hat{T}_N^k] \equiv (\hat{T}_N^k - I)/I$ dla całki z przykładu 7.10

N	k						
	0	1	2	3	4	5	6
1	$1.00 \cdot 10^0$	$4.72 \cdot 10^{-2}$	$-7.15 \cdot 10^{-4}$	$2.77 \cdot 10^{-6}$	$-2.71 \cdot 10^{-9}$	$6.60 \cdot 10^{-13}$	$-0.174 \cdot 10^{-16}$
2	$2.15 \cdot 10^{-1}$	$2.28 \cdot 10^{-3}$	$-8.43 \cdot 10^{-6}$	$8.14 \cdot 10^{-9}$	$-1.98 \cdot 10^{-12}$	$-1.74 \cdot 10^{-16}$	$1.05 \cdot 10^{-15}$
4	$5.19 \cdot 10^{-2}$	$1.35 \cdot 10^{-4}$	$-1.24 \cdot 10^{-7}$	$2.98 \cdot 10^{-11}$	$-2.09 \cdot 10^{-15}$	$1.05 \cdot 10^{-15}$	$1.74 \cdot 10^{-16}$
8	$1.29 \cdot 10^{-2}$	$8.30 \cdot 10^{-6}$	$-1.90 \cdot 10^{-9}$	$1.15 \cdot 10^{-13}$	$1.05 \cdot 10^{-15}$	$1.74 \cdot 10^{-16}$	$6.98 \cdot 10^{-16}$
16	$3.21 \cdot 10^{-3}$	$5.17 \cdot 10^{-7}$	$-2.96 \cdot 10^{-11}$	$1.40 \cdot 10^{-15}$	$1.74 \cdot 10^{-16}$	$6.98 \cdot 10^{-16}$	
32	$8.03 \cdot 10^{-4}$	$3.23 \cdot 10^{-8}$	$-4.62 \cdot 10^{-13}$	$1.74 \cdot 10^{-16}$	$6.98 \cdot 10^{-16}$		
64	$2.01 \cdot 10^{-4}$	$2.02 \cdot 10^{-9}$	$-6.98 \cdot 10^{-15}$	$6.98 \cdot 10^{-16}$			
128	$5.02 \cdot 10^{-5}$	$1.26 \cdot 10^{-10}$	$5.23 \cdot 10^{-16}$				

Dokładniejsza analiza reszt oryginalnej formuły trapezów (por. formuła Eulera-Maclaurina, np. w [J1]) pokazuje, że reszty ciągu $\{\hat{T}_N(f)\}$ są rzędu H^4 , a nie H^2 jak dla ciągu $\{\hat{T}_N^1(f)\}$, tak więc ekstrapolację można powtórzyć. Dla k -krotnej ekstrapolacji, otrzymamy rodzinę tzw. kwadratur Romberga:

$$\hat{T}_N^k = \frac{4^k \hat{T}_{2N}^{k-1} - \hat{T}_N^{k-1}}{4^k - 1} \quad (7.24)$$

gdzie $\hat{T}_N^0 = \hat{T}_N$. Dla rozważanej tu całki otrzymujemy wartości błędów kwadratur Romberga $\delta[\hat{T}_N^k] \equiv (\hat{T}_N^k - I)/I$ przedstawione w tablicy 7.4. Jak widać, ekstrapolacja umożliwia uzyskanie dużo większej dokładności niż prosta metoda trapezów – bez użycia dodatkowych wartości całkowanej funkcji. Na przykład dla 16 węzłów metoda trapezów daje błąd ok. $3.21 \cdot 10^{-3}$, a po trzech krokach ekstrapolacji błąd maleje do poziomu 10^{-15} .

7.1.5. OBLCZANIE CAŁEK Z OSOBLIWOŚCIAMI I CAŁEK NIEWŁAŚCIWYCH

Zastosowanie kwadratur do całkowania numerycznego opiera się na założeniu, że funkcja podcałkowa ma ograniczoną pochodną, a użyty wielomian interpolacyjny pozwala na dokładne jej przybliżenie. Gdy założenie to nie jest spełnione, zachodzi potrzeba odpowiedniej transformacji całki. W tym podrozdziale omówione będą cztery takie transformacje:

- włączanie osobliwości w funkcję wagową;
- podział przedziału całkowania na podprzedziały;
- zamiana zmiennych;
- całkowanie przez części.

Pierwsza transformacja polega na przedstawieniu funkcji podcałkowej w postaci iloczynu dwóch funkcji $g(x) = w(x)f(x)$, z których jedna $w(x)$ może być funkcją wagową.

Przykład 7.11. Obliczyć numerycznie wartość całki

$$I(f) = \int_{-1}^1 \frac{1-x^2}{\sqrt{1-x^2}} dx \quad (7.25)$$

Funkcja podcałkowa jest osobliwa na końcach przedziału całkowania. Ponieważ funkcja $w(x) = 1/\sqrt{1-x^2}$ jest wagą dla wielomianów Czebyszewa pierwszego rodzaju, ortogonalnych w przedziale $[-1, 1]$ (por. przykład 6.5), więc poszukiwaną całkę można zapisać w równoważnej postaci:

$$I = \int_{-1}^1 w(x)f(x) dx \quad (7.26)$$

gdzie $f(x) = 1 - x^2$, i przybliżyć za pomocą kwadratur Gaussa-Czebyszewa. Węzłami tych kwadratur będą zera wielomianu Czebyszewa:

$$x_n = \cos \frac{(2n+1)\pi}{2N+2} \quad \text{dla } n = 0, 1, \dots, N \quad (7.27)$$

Współczynniki kwadratur można obliczyć z równania (7.17):

$$A_n = \frac{\pi}{N+1} \quad \text{dla } n = 0, 1, \dots, N \quad (7.28)$$

Rząd kwadratur Gaussa wyraża się wzorem $p = 2(N+1)$ i dlatego już dla liczby węzłów równej $N+1=2$ całka (7.25) będzie wyznaczona dokładnie, gdyż $f(x)=1-x^2$ jest wielomianem drugiego stopnia

$$I(f) = \hat{I}_2(f) = \hat{I}_2(1-x^2) = \frac{\pi}{2} f\left(-\frac{1}{\sqrt{2}}\right) + \frac{\pi}{2} f\left(\frac{1}{\sqrt{2}}\right) = \frac{\pi}{2} \quad (7.29)$$

Całkę niewłaściwą można niekiedy przedstawić w postaci sumy dwóch całek:

$$I(f) = \int_a^c f(x) dx + \int_c^b f(x) dx \quad (7.30)$$

z których co najmniej jedną – ze względu na szczególne właściwości funkcji $f(x)$ w odpowiednim podprzedziale – można wyznaczyć analitycznie lub za pomocą kwadratur. Druga z całek, jeżeli nie może być wyznaczona w ten sam sposób, może podlegać dalszej dekompozycji.

Przykład 7.12. [D1]. Obliczymy całkę: $I(f) = \int_{-\infty}^{\infty} e^{-x^2} dx$ z dokładnością względną $2 \cdot 10^{-8}$. Całkę tę można przedstawić jako sumę trzech całek:

$$I_1(f) = \int_{-\infty}^{-4} e^{-x^2} dx, \quad I_2(f) = \int_{-4}^4 e^{-x^2} dx, \quad I_3(f) = \int_4^{\infty} e^{-x^2} dx$$

Wartości całek składowych $I_1(f)$ i $I_3(f)$ można oszacować następująco:

$$\begin{aligned} I_1(f) &= I_3(f) = \int_4^{\infty} e^{-x^2} dx = \int_{16}^{\infty} e^{-t} \frac{1}{2} t^{-1/2} dt < \frac{1}{2} 16^{-1/2} \int_{16}^{\infty} e^{-t} dt = \\ &= \frac{1}{8} (-e^{-t}) \Big|_{16}^{\infty} < 1.5 \cdot 10^{-8} \end{aligned}$$

Ponieważ $I(f) = \sqrt{\pi} \approx 1.772454$, więc pominięcie $I_1(f)$ i $I_3(f)$ powoduje błąd względny ok. $1.5 \cdot 10^{-8}$, a więc mniejszy niż dopuszczalny. Do wyznaczenia wartości całki $I(f)$ wystarczy więc obliczenie wartości całki $I_2(f)$ z dokładnością lepszą niż $2 \cdot 10^{-8}$.



Zamiany zmiennych używa się zarówno do transformacji przedziału całkowania (np. z $[-\infty, \infty]$ na $[-1, 1]$), jak i do usuwania osobliwości funkcji podcałkowej. Oto użyteczne transformacje [P]:

- dla osobliwości typu $(x-a)^\gamma$, gdzie $0 \leq \gamma < 1$:

$$\int_a^b f(x) dx = \frac{1}{1-\gamma} \int_0^{(b-a)^{1-\gamma}} t^{\frac{\gamma}{1-\gamma}} f\left(t^{\frac{1}{1-\gamma}} + a\right) dt, \quad \text{gdzie } b > a \quad (7.31)$$

- dla osobliwości typu $(b-x)^\gamma$, gdzie $0 \leq \gamma < 1$:

$$\int_a^b f(x) dx = \frac{1}{1-\gamma} \int_0^{(b-a)^{1-\gamma}} t^{\frac{\gamma}{1-\gamma}} f\left(b - t^{\frac{1}{1-\gamma}}\right) dt, \quad \text{gdzie } b > a \quad (7.32)$$

- dla całki w przedziale półnieskończonym:

$$\int_0^\infty f(x) dx = \int_0^1 \frac{f(-\ln t)}{t} dt \quad (7.33)$$

$$\int_0^\infty x^{-a} f(x) dx = a \int_0^1 t^{a-2} f(t^a) dt \quad \text{dla } f \in C_{[0,\infty]} \text{ i } a \geq 2 \quad (7.34)$$

- dla funkcji malejącej przy $x \rightarrow b$ nie wolniej niż $1/x^2$:

$$\int_a^b f(x) dx = \int_{1/b}^{1/a} \frac{f(1/t)}{t^2} dt, \quad \text{gdzie } b > a, ab > 0 \quad (7.35)$$

Całkowanie przez części polega na wykorzystaniu następującego wzoru:

$$\int_a^b u(x)v'(x) dx = [u(x)v(x)]_a^b - \int_a^b u'(x)v(x) dx \quad (7.36)$$

przy założeniu, że funkcje $u(x)$ i $v(x)$ mają ciągłe pochodne w przedziale $[a,b]$.

Przykład 7.13. [D1]. Bezpośrednie zastosowanie kwadratury do całki:

$$I(f) = \int_0^1 e^x x^{-1/2} dx$$

nie jest możliwe ze względu na osobliwość funkcji podcałkowej w zerze. Całkowanie przez części daje natomiast:

$$I(f) = \int_0^1 e^x x^{-1/2} dx = [2e^x x^{1/2}]_0^1 - \int_0^1 e^x (2x^{1/2}) dx = 2e - 2 \int_0^1 e^x x^{1/2} dx$$

Funkcja podcałkowa w ostatniej całce jest ciągła w zerze i całka ta może być wyznaczona numerycznie. W celu uzyskania lepszej dokładności kwadratur należy poprawić gładkość tej funkcji przez ponowne całkowanie przez części. ♣

Zadanie 7.1. Zrealizować adaptacyjną kwadraturę Simpsona (por. przykład 7.6) w programie MATLAB. Wykreślić zależność faktycznie uzyskanej dokładności całkowania od wartości parametru δI_{\max} – dla funkcji z przykładu 7.2.

Zadanie 7.2. Wykazać, że złożony wzór trapezów dla całkowania w przedziale o długości 2π i dla $h = 2\pi/(N+1)$ jest dokładny dla wszystkich wielomianów trygonometrycznych stopnia n o okresie 2π .

7.2. CAŁKOWANIE FUNKCJI WIELU ZMIENNYCH

Całkowanie funkcji wielu zmiennych jest znacznie trudniejsze niż całkowanie funkcji jednej zmiennej, ponieważ:

- konstrukcja wielomianów interpolacyjnych jest możliwa tylko dla odpowiednio położonych węzłów i odpowiednio regularnych obszarów całkowania;
- nakłady obliczeniowe rosną bardzo szybko z liczbą zmiennych.

Jedyną ogólną metodą całkowania, którą można użyć dla dużej liczby zmiennych, jest statystyczna metoda Monte Carlo, omówiona w rozdziale 8. W tym podrozdziale rozważymy tylko jedną niestatystyczną metodę całkowania, która może być konkurencyjna w stosunku do metody Monte Carlo, gdy wymagana jest duża dokładność, a liczba zmiennych nie przekracza 3.

Jeżeli obszar całkowania $\mathbb{A} \subset \mathbb{R}^M$ da się opisać układem nierówności:

$$a_1 \leq x_1 \leq b_1$$

$$a_2(x_1) \leq x_2 \leq b_2(x_1)$$

.....

$$a_M(x_1, x_2, \dots, x_{M-1}) \leq x_M \leq b_M(x_1, x_2, \dots, x_{M-1})$$

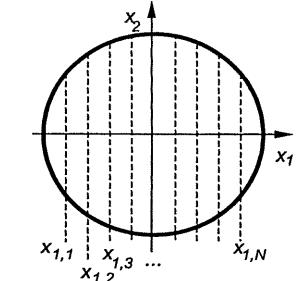
to całkę wielokrotną można przekształcić w całkę iterowaną:

$$\begin{aligned} I(f) &= \int_{\Omega} \cdots \int f(x_1, \dots, x_M) dx_1 \cdots dx_M = \\ &= \int_{a_1}^{b_1} dx_1 \int_{a_2(x_1)}^{b_2(x_1)} dx_2 \cdots \int_{a_n(x_1, x_2, \dots, x_{M-1})}^{b_n(x_1, x_2, \dots, x_{M-1})} f(x_1, \dots, x_M) dx_M \end{aligned} \quad (7.37)$$

której numeryczne wyznaczenie sprowadza się do M -krotnego użycia kwadratur jednowymiarowych.

Przykład 7.14. Rozważmy całkowanie funkcji $f(x_1, x_2)$ po obszarze koła jednostkowego (rys. 7.9):

$$\mathbb{A} = \{\mathbf{x} \in \mathbb{R}^2 \mid -1 \leq x_1 \leq 1, -\sqrt{1-x_1^2} \leq x_2 \leq \sqrt{1-x_1^2}\} \quad (7.38)$$



Rysunek 7.9. Całkowanie po obszarze koła

W tym celu przedstawmy całkę w postaci:

$$I(f) = \int_{-1}^1 g(x_1) dx_1, \quad g(x_1) = \int_{-\sqrt{1-x_1^2}}^{\sqrt{1-x_1^2}} f(x_1, x_2) dx_2 \quad (7.39)$$

Funkcja $g(x_1)$ nie jest wprawdzie znana w postaci analitycznej, ale jej wartości w węzłach $x_{1,i}$ kwadratury:

$$\hat{I}_{N_1}(g) = \sum_{n=0}^{N_1} A_n g(x_{1,n})$$

mogą być przybliżone za pomocą kwadratur o postaci:

$$\hat{I}_{N_2,n}(f_n) = \sum_{v=0}^{N_2,n} B_{v,n} f(x_{1,n}, x_{2,v})$$

gdzie $f_n(x_2) = f(x_{1,n}, x_2)$. Ostatecznie więc:

$$I(f) = \iint_A f(x_1, x_2) dx_1 dx_2 = \sum_{n=0}^{N_1} \sum_{v=0}^{N_2,n} B_{v,n} f(x_{1,n}, x_{2,v}) + R_{N_1}(g) + \sum_{n=0}^{N_2,n} A_n R_{N_2,n}(f_n)$$

gdzie $R_{N_1}(g)$ jest resztą kwadratury $\hat{I}_{N_1}(g)$, zaś $R_{N_2,n}(f_n)$ – resztą kwadratury $\hat{I}_{N_2,n}(f_n)$. Warto zauważyc, że przedział całkowania względem zmiennej x_2 zależy od wartości x_1 . Liczba węzłów kwadratur $\hat{I}_{N_2,n}(f_n)$ może więc być inna dla każdego węzła $x_{1,n}$.



W przykładzie tym liczba użytych węzłów wynosi:

$$\sum_{n=0}^{N_1} (N_{2,n} + 1)$$

Jeśli liczba węzłów w każdej kwadraturze byłaby równa $N+1$, to trzeba by obliczyć $(N+1)^2$ wartości funkcji $f(x_1, x_2)$. Ogólnie dla M wymiarów potrzeba $(N+1)^M$ obliczeń funkcji. Na przykład, dla $N=1$ (kwadratura dwupunktowa) i $M=10$ mamy $(1+1)^{10} = 1024$ obliczeń, a dla $M=20$ $(1+1)^{20} = 1048576$, co daje wyobrażenie o ograniczeniach przedstawionej techniki obliczania całki wielokrotnej.

Zadanie 7.3. Zaprogramować obliczanie całki funkcji $f(x_1, x_2) = \sqrt{4 - x_1^2 - x_2^2}$ w obszarze \mathbb{A} zdefiniowanym wzorem (7.38), wykorzystując adaptacyjną kwadraturę Simpsona do całkowania względem jednej zmiennej.

7.3. RÓŻNICZKOWANIE FUNKCJI JEDNEJ ZMIENNEJ

7.3.1. FORMUŁY RÓŻNICOWE

Numeryczne wyznaczanie pochodnych funkcji $f(x)$ – na podstawie wartości tej funkcji w danej liczbie punktów x_n , $n = 0, 1, \dots$, występuje często jako podzadanie algorytmów optymalizacji lub algorytmów rozwiązywania równań nieliniowych. Może być również wykorzystywane do analizy wrażliwości układów fizycznych (np. elektronicznych) na niewielkie zaburzenia fizycznych parametrów procesu produkcyjnego lub warunków eksploatacji obiektów technicznych (temperatury, napięcia zasilania itp.).

Przykład 7.15. W zadaniu 2.8 należy oszacować rozrzutu modułu wzmacnienia filtru $|K(j\omega, p)|$, wynikające z rozrzutów produkcyjnych parametrów jego elementów $p = [\tilde{R}_1 \tilde{R}_2 \tilde{C}_1 \tilde{C}_2]^T$; $\tilde{p}_i = p_i(1 + \delta_i)$, $|\delta_i| \leq 1\%$. Jedna z użytecznych metod szacowania skutków rozrzutów parametrów p polega na zastosowaniu liniowego przybliżenia funkcji $f(\omega, p)$:

$$\Delta f(\omega, \tilde{p}) = f(\omega, \tilde{p}) - f(\omega, p) \approx \sum_i \frac{\partial f}{\partial p_k} p_k \delta_k \quad (7.40)$$

Należy więc obliczyć wrażliwość (pochodnych) odpowiedzi układu $|K(j\omega, p)|$ względem parametrów p . Jeśli te wrażliwości nie są dostępne w postaci jawnych formuł, konieczne jest użycie numerycznego różniczkowania funkcji, np. technik omawianych w tym rozdziale. Warto przy tym pamiętać, że istnieją też inne, specjalizowane metody obliczania wrażliwości układów elektronicznych, które są dokładniejsze od tutaj omawianych. Zainteresowanych odsyłamy do literatury specjalistycznej [O, S2].



Pierwsza pochodna funkcji $f(x)$ jest z definicji granicą ilorazu różnicowego:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \quad (7.41)$$

co sugeruje możliwość przybliżania wartości tej pochodnej za pomocą różnicy progresywnej:

$$D_F(f, x, h) = \frac{f(x+h) - f(x)}{h} \quad (7.42)$$

czy też różnicy wstecznej:

$$D_B(f, x, h) = \frac{f(x) - f(x-h)}{h} \quad (7.43)$$

dla odpowiednio małej wartości kroku h . Przybliżenie pochodnej uzyskujemy również dla różnicy centralnej, która jest średnią arytmetyczną różnicy progresywnej i wstecznej:

$$D_C(f, x, h) = \frac{f(x+h) - f(x-h)}{2h} \quad (7.44)$$

Nawet w przypadku obliczeń dokładnych błędów aproksymacji pochodnej przez każdą z tych różnic, tzw. błąd obcięcia, jest na ogół niezerowy, ale dla funkcji różniczkowalnej w punkcie x maleje do zera przy zmniejszaniu kroku h do zera. Błąd ten szacuje się, podstawiając za $f(x+h)$ rozwinięcie tej funkcji w szereg Taylora z resztą. Dla formuły (7.42) otrzymuje się w ten sposób następujące oszacowanie błędu obcięcia:

$$D_F(f, x, h) - f'(x) = \frac{f(x) + f'(x)h + f''(\xi)h^2/2 - f(x)}{h} - f'(x) = f''(\xi)\frac{h}{2} \quad (7.45)$$

gdzie $\xi \in [x, x+h]$. W przypadku realizacji formuły (7.42) w arytmetyce zmiennopozycyjnej niedokładność obliczeń pochodnej zawiera dodatkową składową związaną z niedokładnością numerycznej realizacji funkcji f oraz błędami zaokrągleń:

$$\begin{aligned} \tilde{D}_F(f, x, h) &= \frac{[f(x+h)(1+\varepsilon_1) - f(x)(1+\varepsilon_0)](1+\eta_o)}{h}(1+\eta_d) \approx \\ &\approx \frac{f(x+h) - f(x) + f(x+h)\varepsilon_1 - f(x)\varepsilon_0}{h}(1+\eta_o + \eta_d) = \\ &= \frac{f(x+h) - f(x)}{h} \left[1 + \frac{f(x+h)\varepsilon_1 - f(x)\varepsilon_0}{f(x+h) - f(x)} + \eta_o + \eta_d \right] = \\ &= \left[f'(x) + f''(\xi)\frac{h}{2} \right] \left[1 + \underbrace{\frac{f(x+h)\varepsilon_1 - f(x)\varepsilon_0}{f(x+h) - f(x)}}_{\eta_f} + \eta_o + \eta_d \right] = \\ &= \left[f'(x) + f''(\xi)\frac{h}{2} \right] (1 + \eta_f + \eta_o + \eta_d) \end{aligned}$$

gdzie ε_0 i ε_1 są względnymi błędami obliczania $f(x)$ i $f(x+h)$, natomiast η_o i η_d są względnymi błędami zaokrąglenia wyników odejmowania i dzielenia. Przy założeniu, że $|\varepsilon_0|, |\varepsilon_1| \leq EPS$ (górnego oszacowania niedokładności obliczenia funkcji $f(x)$; zazwyczaj $EPS \gg eps$) oraz $|\eta_o|, |\eta_d| \leq eps$ można uzyskać oszacowanie:

$$|\eta_f| \leq \frac{|f(x+h)|EPS + |f(x)|EPS}{\left|f'(x) + f''(\xi)\frac{h}{2}\right|h} \approx 2 \frac{|f(x)|}{|f'(x)|} \frac{EPS}{h} \quad (7.46)$$

Ze względu na małą wartość h w mianowniku oraz nierówność: $EPS \gg eps$, na ogół można pominąć η_o i η_d , ponieważ ich oszacowania są małe względem oszacowania η_f . Błąd $\tilde{D}_F(f, x, h)$ można wówczas oszacować następująco:

$$|\tilde{D}_F(f, x, h) - f'(x)| \leq \frac{h}{2}|f''(\xi)| + \frac{2}{h}|f(x)|EPS \quad (7.47)$$

Najmniejszą wartość prawej strony tej nierówności uzyskuje się dla kroku:

$$\hat{h}_F = 2 \cdot \sqrt{\frac{|f(x)|}{|f''(\xi)|}} EPS \quad (7.48)$$

Występującą w tym wzorze wartość $f''(\xi)$ trzeba oszacować na podstawie dotychczasowych obliczeń pierwszej pochodnej w kilku punktach albo korzystając z formuły różnicowej, np. drugiego rzędu:

$$f''(\xi) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \quad (7.49)$$

Optymalizacja kroku różniczkowania numerycznego za pomocą różnic wstecznych, analogiczna do tej przeprowadzonej dla różnicy progresywnej, daje $\hat{h}_B = \hat{h}_F$. Natomiast optymalizacja kroku dla różnicy centralnej, dla której błąd obcięcia:

$$D_C(f, x, h) - f'(x) = f'''(\xi)\frac{h^2}{6} \quad (7.50)$$

prowadzi do wzoru:

$$\hat{h}_C = \sqrt[3]{\frac{3|f(x)|}{|f'''(\xi)|}} EPS \quad (7.51)$$

Przykład 7.16. Wyznaczyć numerycznie wartość pochodnej funkcji:

$$f(x) = \left(\frac{1}{\sqrt{1+x^2}} - 1 \right)^2 \quad (7.52)$$

w punkcie $x=1$, korzystając z formuł różnicy progresywnej i różnicy centralnej dla różnych wartości kroku h . Oszacować optymalną wartość kroku h , przyjmując $eps = 1.11 \cdot 10^{-16}$ (obliczenia w podwójnej precyzyji wg IEEE 754).

Wartości dokładne funkcji i jej pochodnych są następujące:

$$f(1) = (\sqrt{2} - 1)^2 / 2 \approx 0.085786\dots$$

$$f'(1) = (\sqrt{2} - 1) / 2 \approx 0.207106781,$$

$$f''(1) = (\sqrt{2} - 1) / (2\sqrt{2}) \approx 0.1464466\dots$$

$$f'''(1) = -3\sqrt{2} / 8 \approx -0.53033008\dots$$

Zakładając, że wszystkie jedynki we wzorze (7.52) reprezentowane są dokładnie, błąd zmiennopozycyjnej realizacji obliczeń wartości funkcji $f(x)$ dla $x=1$ można oszacować na podstawie następującego wzoru:

$$\tilde{f}(x) = \left\{ \left[\frac{1 + \eta_d}{\sqrt{[1 + [x(1 + \varepsilon_x)]^2(1 + \eta'_p)](1 + \eta_s)(1 + \eta_{\sqrt{}})}} - 1 \right] (1 + \eta''_p) \right\}^2 (1 + \eta''_p)$$

gdzie: ε_x – błąd reprezentacji danej x ,

η'_p – błąd zaokrąglenia wyniku potęgowania x^2 ,

η_s – błąd zaokrąglenia wyniku sumowania $v_1 = 1 + x^2$,

$\eta_{\sqrt{}}$ – błąd zaokrąglenia wyniku pierwiastkowania $v_2 = \sqrt{v_1}$,

η_d – błąd zaokrąglenia wyniku dzielenia $v_3 = 1/v_2$,

η_o – błąd zaokrąglenia wyniku odejmowania $v_4 = v_3 - 1$,

η''_p – błąd zaokrąglenia wyniku potęgowania v_4^2 .

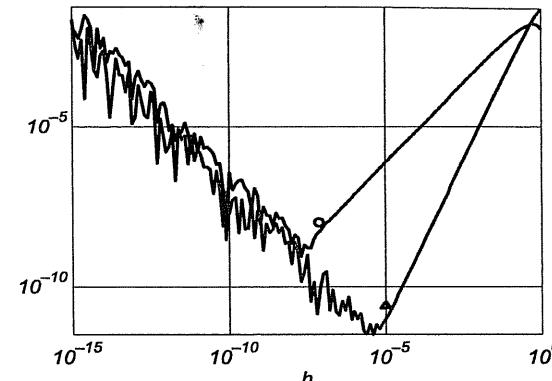
Oszacowanie względnego błędu obliczania wartości funkcji $f(x)$ ma postać:

$$|\delta[f(x)]| \leq |\eta_d| + \frac{|\eta_s|}{2} + |\eta_{\sqrt{}}| + \frac{x^2}{1+x^2} \left(|\varepsilon_x| + \frac{|\eta'_p|}{2} \right)$$

Przy założeniu, że $|\varepsilon_x|, |\eta'_p|, |\eta''_p|, |\eta_s|, |\eta_d|, |\eta_o|, |\eta_{\sqrt{}}| \leq \text{eps} = 1.11 \cdot 10^{-16}$:

$$|\delta[\tilde{f}(1)]| \leq \left(3 + \frac{13/2}{\sqrt{2}-1} \right) \text{eps} \approx 18.69 \text{eps} \equiv \text{EPS} \approx 10^{-8} \quad (7.53)$$

Odpowiadająca temu oszacowaniu optymalna długość kroku, obliczona według wzoru (7.48), wynosi zatem $\hat{h}_F \approx 1.6 \cdot 10^{-8}$, a wartość błędu obliczania pierwsiowej pochodnej wynosi ok. 10^{-8} . Przeprowadzając obliczenia wg wzoru (7.51), otrzymuje się oszacowanie optymalnej długości kroku dla formuły różnicycznej centralnej $\hat{h}_C \approx 10^{-5}$, przy czym względny błąd wyznaczenia pochodnej jest rzędu $2.66 \cdot 10^{-11}$. W celu weryfikacji powyższych przewidywań wykonano obliczenia w systemie MATLAB, a ich wyniki przedstawiono na rys. 7.10.



Rysunek 7.10. Zależność względnego błędu obliczania $f'(1)$ od długości kroku h dla metody różnicycznej progresywnej i metody różnicicy centralnej; kółkiem i trójkątem oznaczono najlepsze długości kroku, \hat{h}_F i \hat{h}_C , wyznaczone za pomocą formuł (7.48) i (7.51)

Wyznaczone za pomocą wzorów (7.48) i (7.51) optymalne długości kroku są nieco większe od rzeczywistych wartości uzyskanych w obliczeniach numerycznych, ponieważ oszacowanie błędu obliczeń funkcji (7.46) jest oszacowaniem z góry (na najgorszy przypadek).

7.3.2. RÓŻNICZKOWANIE FORMUŁ INTERPOLACYJNYCH

Niech

$$\hat{f}(x) = \tilde{f}(x; \hat{\mathbf{p}}) = \sum_{k=1}^K \hat{p}_k \varphi_k(x) \quad (7.54)$$

gdzie $\{\varphi_k(x)\}$ jest układem funkcji liniowo niezależnych, będzie funkcją interpolującą funkcję $f(x)$ w węzłach x_0, \dots, x_N , przy czym $K = N + 1$, a wektor $\hat{\mathbf{p}}$ spełnia warunki interpolacji: $\tilde{f}(x_n; \hat{\mathbf{p}}) = f(x_n)$ dla $n = 0, 1, \dots, N$. Do przybliżonego obliczenia wartości pochodnej funkcji $f(x)$ można wykorzystać pochodną funkcji interpolującej, tj.:

$$\hat{f}'(x) = \frac{\partial \tilde{f}(x; \hat{\mathbf{p}})}{\partial x} = \sum_{k=1}^K \hat{p}_k \varphi'_k(x) \quad (7.55)$$

Błąd przybliżenia $f'(x)$ przez $\hat{f}'(x)$ zależy od jakości przybliżenia funkcji $f(x)$ przez funkcję $\tilde{f}(x)$ w otoczeniu punktu różniczkowania x , a więc zależy zarówno od własności $f(x)$, jak i od wyboru układu funkcji $\{\varphi_k(x)\}$.

Przykład 7.17. Interpolacja liniowa Lagrange'a, oparta na dwóch węzłach \check{x} i $\check{x} + h$:

$$\begin{aligned}\hat{f}(x) &= f(\check{x}) \frac{x - (\check{x} + h)}{\check{x} - (\check{x} + h)} + f(\check{x} + h) \frac{x - \check{x}}{(\check{x} + h) - \check{x}} = \\ &= f(\check{x}) \frac{x - (\check{x} + h)}{-h} + f(\check{x} + h) \frac{x - \check{x}}{h}\end{aligned}$$

po zróżniczkowaniu daje formułę różnicy progresywnej: $f'(x) = [f(\check{x} + h) - f(\check{x})]/h$ z błędem obcięcia $O(h)$. Interpolacja paraboliczna Lagrange'a, oparta na trzech węzłach $\check{x} - h$, \check{x} i $\check{x} + h$:

$$\begin{aligned}\hat{f}(x) &= f(\check{x} - h) \cdot \frac{(x - \check{x})(x - (\check{x} + h))}{((\check{x} - h) - \check{x})((\check{x} - h) - (\check{x} + h))} + f(\check{x}) \cdot \frac{(x - (\check{x} - h))(x - (\check{x} + h))}{(\check{x} - (\check{x} - h))(\check{x} - (\check{x} + h))} + \\ &\quad + f(\check{x} + h) \cdot \frac{(x - (\check{x} - h))(x - \check{x})}{((\check{x} + h) - (\check{x} - h))((\check{x} + h) - \check{x})}\end{aligned}$$

po zróżniczkowaniu daje formułę różnicy centralnej z błędem obcięcia $O(h^2)$, która po powtórnym zróżniczkowaniu daje:

$$D_C^2(f; x, h) = \frac{f(x - h) - 2f(x) + f(x + h)}{h^2} \quad (7.56)$$

przybliżającą drugą pochodną funkcji $f(x)$ z błędem obcięcia $O(h^2)$. Interpolacja wielomianem Lagrange'a czwartego stopnia, oparta na pięciu węzłach $\check{x} - 2h$, $\check{x} - h$, \check{x} , $\check{x} + h$ i $\check{x} + 2h$, po zróżniczkowaniu daje formułę numeryczną różniczkowania z błędem obcięcia $O(h^4)$:

$$D_{C5}(f; x, h) = \frac{f(x - 2h) - 8f(x - h) + 8f(x + h) - f(x + 2h)}{12h} \quad (7.57)$$

Do tworzenia formuł numerycznego różniczkowania można wykorzystać także wielomiany interpolacyjne Newtona. Różniczkowanie takich wielomianów z węzłami równoodległymi (por. wzór (6.13)) daje formułę:

$$\begin{aligned}D_N(f; x, h) &= \frac{1}{h} \frac{dL_N(x + \xi h)}{d\xi} \Big|_{\xi=0} = \\ &= \frac{1}{h} \left(\Delta^1 f(x) \frac{dq_1(\xi)}{d\xi} + \frac{\Delta^2 f(x)}{2!} \frac{dq_2(\xi)}{d\xi} + \dots + \frac{\Delta^N f(x)}{N!} \frac{dq_N(\xi)}{d\xi} \right) \Big|_{\xi=0}\end{aligned}$$

gdzie $q_n(\xi) \equiv \prod_{k=0}^{n-1} (\xi - k)$, a ilorazy różnicowe $\Delta^n f(x)$ zdefiniowane są wzorami (6.11) i (6.12). Dla $N = 2$ przybiera ona postać:

$$\begin{aligned}f'(x) &\approx \frac{1}{h} \left[\Delta^1 f(x) \frac{dq_1(\xi)}{d\xi} + \frac{\Delta^2 f(x)}{2!} \frac{dq_2(\xi)}{d\xi} \right] \Big|_{\xi=0} = \\ &= \frac{1}{h} \left[\Delta^1 f(x) + \frac{\Delta^2 f(x)}{2!} (2\xi - 1) \right] \Big|_{\xi=0} = \frac{2\Delta^1 f(x) - \Delta^2 f(x)}{2h}\end{aligned}$$

gdzie $\Delta^1 f(x) = f(x + h) - f(x)$ oraz $\Delta^2 f(x) = \Delta^1 f(x + h) - \Delta^1 f(x)$.

Ponieważ dokładność obliczania pochodnej zależy od dokładności interpolacji, więc czasami można poprawić dokładność, dokonując odpowiedniej transformacji różniczkowanej funkcji.

Przykład 7.18. [D2]. Wyznaczyć numerycznie pochodną funkcji $f(x) = \Gamma(x+1)$ dla $x \in [10, 16]$. Jest to funkcja bardzo silnie rosnąca w tym przedziale (dla argumentów całkowitych $\Gamma(x+1) = x!$), którą źle się przybliża wielomianem. Znacznie lepiej natomiast przybliża się funkcję $\log(f(x))$ dla $x \in [10, 16]$. Pochodną funkcji $f(x)$ można wyrazić za pomocą wzoru:

$$f'(x) = \frac{f(x)}{\log(e)} \frac{d(\log(f(x)))}{dx}$$

gdzie drugi czynnik można obliczyć, różniczkując wielomian interpolujący $\log(f(x))$.

7.3.3. ZWIĘKSZANIE DOKŁADNOŚCI RÓŻNICZKOWANIA METODĄ EKSTRAPOLACJI RICHARDSONA

Załóżmy, że $(N+1)$ -punktowa formuła różniczkowania pewnej funkcji $f(x)$ gładkiej może być zapisana następująco:

$$D_0(f; x, h) = f'(x) + \sum_{k=K}^{\infty} \alpha_k h^k \quad (7.58)$$

Rozważmy następującą kombinację liniową dwóch przybliżonych wartości pochodnej $f'(x)$, uzyskanych dla dwóch odległości węzłów interpolacji: h oraz qh :

$$\begin{aligned}D_1(f; x, h) &\equiv \lambda D_0(f; x, h) + (1 - \lambda) D_0(f; x, qh) = \\ &= f'(x) + \sum_{k=K}^{\infty} \alpha_k [\lambda + (1 - \lambda) q^k] h^k\end{aligned} \quad (7.59)$$

Otrzymana w ten sposób formuła różniczkowania $D_1(f; x, h)$ może być dokładniejsza od $D_0(f; x, h)$, jeśli wartość parametru λ zostanie dobrana tak, aby zerował się pierwszy składnik sumy w (7.59). Dla wartości:

$$\lambda = \frac{q^k}{q^k - 1} \quad (7.60)$$

uzyskuje się formułę różniczkowania $D_1(f; x, h)$ z błędem obcięcia rzędu co najmniej $O(h^{K+1})$, podczas gdy dla $D_0(f; x, h)$ błąd ten jest rzędu $O(h^K)$.

Dla formuły różnicicy centralnej wzór (7.58) przybiera postać:

$$D_{C,0}(f; x, qh) \equiv \frac{f(x+h) - f(x-h)}{2h} = f'(x) + \sum_{k=2}^{\infty} \alpha_k h^k \quad (7.61)$$

przy czym współczynniki $\alpha_k = \frac{1 - (-1)^{k+1}}{2} \frac{f^{(k+1)}(x)}{(k+1)!}$ zerują się dla nieparzystych wartości indeksu k . Ponieważ dla formuły różnicicy centralnej mamy $K = 2$, więc $\lambda = q^2/(q^2 - 1)$. Wzory (7.59)–(7.61) dają w tym przypadku formułę:

$$D_{C,1}(f; x, h) \equiv \frac{q^2 D_{C,0}(f; x, h) - D_{C,0}(f; x, qh)}{q^2 - 1} = f'(x) + \sum_{k=2}^{\infty} \frac{\alpha_k (q^2 - q^k)}{q^2 - 1} h^k \quad (7.62)$$

Ponieważ w rozwinięciu (7.62) nie ma składowych z nieparzystymi potęgami h , błąd obcięcia uzyskanej formuły różniczkowania jest rzędu $O(h^4)$. Opisany proces tworzenia nowej formuły różniczkowania, nazywany ekstrapolacją Richardsoна, można prowadzić dalej w celu uzyskania ciągu przybliżeń pochodnej coraz wyższego rzędu. Schemat obliczeń można zapisać w postaci macierzy, której strukturę pokazano w tablicy 7.5. Warto zwrócić uwagę na to, że liczba węzłów, wykorzystanych do obliczenia przybliżonej wartości pochodnej znajdującej się w n -tej kolumnie tej tablicy wynosi $2n$.

Tablica 7.5

Zależności między formułami przybliżającymi pochodne uzyskiwanymi w wyniku rekurencyjnego zastosowania ekstrapolacji Richardsoна: $D_{C,n}(f; x, h_n)/q = \frac{q^2 D_{C,n-1}(f; x, h_n) - D_{C,1}(f; x, qh_n)}{q^2 - 1}$

$D_{C,0}(f; x, h)$				
$D_{C,0}(f; x, hq)$	$D_{C,1}(f; x, hq)$			
$D_{C,0}(f; x, hq^2)$	$D_{C,1}(f; x, hq^2)$	$D_{C,2}(f; x, hq^2)$		
$D_{C,0}(f; x, hq^3)$	$D_{C,1}(f; x, hq^3)$	$D_{C,2}(f; x, hq^3)$	$D_{C,3}(f; x, hq^3)$	
$D_{C,0}(f; x, hq^4)$	$D_{C,1}(f; x, hq^4)$	$D_{C,2}(f; x, hq^4)$	$D_{C,3}(f; x, hq^4)$	$D_{C,4}(f; x, hq^4)$

Przykład 7.19. Wyznaczmy ciąg przybliżeń pochodnej funkcji $f(x) = \sin(x)$ w punkcie $x = 1$ za pomocą ciągu formuł różniczkowych wynikających ze wzoru (7.59) dla początkowej odległości węzłów: $h = 0.5$ i $q = \sqrt{2}$. W tablicy 7.6 zestawiono wartości błędów numerycznego różniczkowania funkcji $f(x)$, odpowiadających formułom zestawionym w analogicznym układzie w tablicy 7.5.

Tablica 7.6

Błędy przybliżeń pochodnej funkcji $f(x) = \sin(x)$ w punkcie $x = 1$, uzyskanych za pomocą formuły różnicicy centralnej z ekstrapolacją Richardsoна, zdefiniowanych w odpowiednich komórkach tablicy 7.5

$-2.22 \cdot 10^{-2}$				
$-1.12 \cdot 10^{-2}$	$-1.39 \cdot 10^{-4}$			
$-5.61 \cdot 10^{-3}$	$-3.50 \cdot 10^{-5}$	$-2.08 \cdot 10^{-7}$		
$-2.81 \cdot 10^{-3}$	$-8.77 \cdot 10^{-6}$	$-2.61 \cdot 10^{-8}$	$-9.05 \cdot 10^{-11}$	
$-1.41 \cdot 10^{-3}$	$-2.20 \cdot 10^{-6}$	$-3.27 \cdot 10^{-9}$	$-5.67 \cdot 10^{-12}$	$-1.23 \cdot 10^{-14}$

7.3.4. RÓŻNICZKOWANIE FORMUŁ APROKSYMACJI WYGŁADZAJĄCEJ

Jeżeli wartości funkcji obarczone są niepowiązonym błędem przypadkowym (np. błędem pomiaru), to bezpośrednie użycie opisanych dotychczas formuł numerycznego różniczkowania nie jest celowe. Zmniejszanie kroku h prowadzi bowiem do szybkiego wzrostu wpływu tych błędów na niedokładność wyniku różniczkowania. Opis statystyczny tego zjawiska ilustruje kolejny przykład.

Przykład 7.20. Założymy, że wartości funkcji $f(x) = ax^2 + bx + c$ są mierzone z addytywnym błędem losowym η o zerowej wartości oczekiwanej i wariancji σ^2 . Należy wyznaczyć wartość oczekiwana i wariancję przybliżeń $f'(x)$ – uzyskanych za pomocą formuły różnicicy progresywnej oraz formuły różnicicy centralnej – przy założeniu, że realizacje błędów losowych są statystycznie niezależne.

Dla ustalonej wartości x wynik zastosowania formuły różnicicy progresywnej (7.42) do wartości $f(x)$ i $f(x+h)$ zaburzonych losowo przez niezależne zmienne losowe η_1 i η_2 ma postać:

$$\begin{aligned} \tilde{f}'_F(x) &= \frac{f(x+h) + \eta_1 - (f(x) + \eta_2)}{h} = \frac{f(x+h) - f(x)}{h} + \frac{\eta_1 - \eta_2}{h} = \\ &= \tilde{f}'_F(x) + \Delta f' \end{aligned} \quad (7.63)$$

gdzie $\Delta f' = (\underline{\eta}_1 - \underline{\eta}_2) / h$. Następstwem losowego zaburzenia danych pomiarowych jest więc pojawienie się składowej losowej błędu różniczkowania numerycznego $\underline{\Delta f'}$ o zerowej wartości oczekiwanej i wariancji:

$$\text{Var}\{\underline{\Delta f'}\} = \frac{1}{h^2} \text{Var}\{\underline{\eta}_1\} + \frac{1}{h^2} \text{Var}\{\underline{\eta}_2\} = \frac{2\sigma^2}{h^2} \quad (7.64)$$

Analogiczne rozumowanie dla formuły różnicicy centralnej (7.44) prowadzi do wniosku, że:

$$\tilde{f}'_C(x) = \frac{f(x+h) + \underline{\eta}_1 - (f(x-h) + \underline{\eta}_2)}{2h} = \tilde{f}'_C(x) + \underline{\Delta f'} \quad (7.65)$$

$$\text{Var}\{\underline{\Delta f'}\} = \frac{1}{4h^2} \text{Var}\{\underline{\eta}_1\} + \frac{1}{4h^2} \text{Var}\{\underline{\eta}_2\} = \frac{\sigma^2}{2h^2} \quad (7.66)$$

przy czym $\Delta f' = (\underline{\eta}_1 - \underline{\eta}_2) / 2h$. Wariancja błędów obydwu estymatorów pochodzącej rośnie nieograniczenie przy zmniejszaniu kroku h do 0; jest jednak czterokrotnie mniejsza dla formuły różnicicy centralnej.

Ze względu na nieograniczony wzrost błędów numerycznego różniczkowania przy zmniejszaniu kroku h do 0, gdy dane są zaburzone statystycznymi zakłóceniami, zaleca się aproksymację tych danych za pomocą jakiejś funkcji wygładzającej (filtrującej) błędy, a następnie różniczkowanie tej funkcji. Szczegółowe omówienie tego zagadnienia wykracza jednak poza ramy niniejszego podręcznika.

Zadanie 7.4. [D1]. Niech $P(x)$ będzie wielomianem drugiego stopnia aproksymującym funkcję $f(x)$ w sensie błędu średniokwadratowego, przy czym $x_n = x_0 + nh$ ($n = -2, -1, 0, 1, 2$) są węzłami aproksymacji. Wykazać, że:

$$P(x_0) = \frac{1}{5} \sum_{n=-2}^2 f(x_n) - h^2 P''(x_0)$$

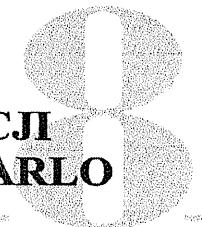
oraz

$$P'(x_0) = \frac{f(x_1) - f(x_{-1})}{2h} + \frac{f(x_2) - f(x_{-2}) + 2f(x_1) - 2f(x_{-1})}{5h}$$

Zadanie 7.5. Wyprowadzić siedmiopunktową formułę różnicową, wykorzystującą wygładzającą aproksymację paraboliczną. Zbadać zależność średniego i średniokwadratowego błędu różniczkowania numerycznego funkcji $f(x) = \sin(x)$ dla $x \in [-2, 2]$ od kroku $h \in [10^{-15}, 10^{-1}]$ dla:

- dwupunktowej formuły różnicicy progresywnej (7.42);
- trójpunktowej formuły różnicicy centralnej (7.44);
- pięciopunktowej formuły różnicowej (7.57);
- wyprowadzonej siedmiopunktowej formuły różnicowej.

CALKOWANIE FUNKCJI – METODA MONTE CARLO



Rozdział ten ma charakter uzupełniający w stosunku do rozdziału 7, w którym przedstawiono problem numerycznego wyznaczania wartości całek oznaczonych funkcji jednej zmiennej i całek iterowanych oraz klasyczne metody rozwiązywania tego problemu, jakimi są kwadratury liniowe. Metody te dają zadowalające wyniki w przypadku całkowania funkcji jednej, dwóch, a nawet trzech zmiennych, o ile funkcja ta jest odpowiednio gładka. W innych przypadkach można odwołać się do metody Monte Carlo.

8.1. WPROWADZENIE DO METODY MONTE CARLO

Metoda Monte Carlo opracowana została jako narzędzie estymacji wartości oczekiwanej i innych parametrów zmiennych losowych na podstawie ich realizacji. Szybko okazało się jednak, że przy użyciu tej metody można rozwiązać wiele innych zadań, w tym zadań związanych z wyznaczaniem wartości całek. Zazwyczaj nie jest to metoda konkurencyjna względem kwadratur interpolacyjnych jednej zmiennej – ze względu na stosunkowo powolne zwiększenie dokładności ze wzrostem liczby węzłów. Jest to natomiast metoda bardzo prosta i skuteczna dla całek funkcji wielu zmiennych, całek funkcji niegładkich, a nawet nieciągłych. Istotę dwóch podstawowych wariantów metody Monte Carlo, zastosowanej do wyznaczania wartości całek oznaczonych, ilustrują kolejne dwa przykłady. Bardziej szczegółowy opis metod zawiera podrozdział 8.3.

Przykład 8.1. Założmy, że należy wyznaczyć przybliżoną wartość całki funkcji $F(x)$ jednej zmiennej w przedziale $[a, b]$, tj.:

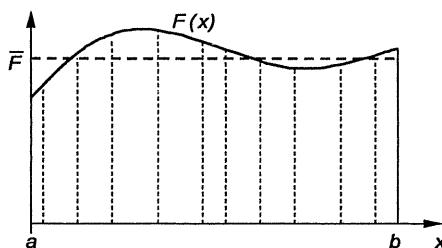
$$I = \int_a^b F(x) dx \quad (8.1)$$

Metoda Monte Carlo w tzw. wariancie podstawowym opiera się na przedstawieniu tej całki w postaci iloczynu wartości średniej funkcji $F(x)$:

$$\bar{F} = \frac{\int_a^b F(x) dx}{b-a}$$

oraz długości przedziału całkowania:

$$I = (b-a)\bar{F}$$



Rysunek 8.1. Ilustracja do przykładu 8.1

Wartość średnią \bar{F} można oszacować statystycznie, losując N punktów x_n ($n=1, \dots, N$) o rozkładzie równomiernym (w sensie statystycznym) w przedziale $[a, b]$ i wyznaczając średnią arytmetyczną odpowiadających im wartości funkcji $F(x)$ (rys. 8.1):

$$\hat{F} = \frac{\sum_{n=1}^N F(x_n)}{N} \quad (8.3)$$

Dla dużej liczby punktów $\hat{F} \approx \bar{F}$. Przybliżona wartość całki (8.2) może być więc obliczona następująco:

$$\hat{I} = (b-a) \frac{\sum_{n=1}^N F(x_n)}{N} \quad (8.4)$$

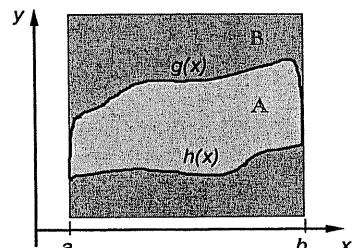
Przykład 8.2. Zastosujmy metodę Monte Carlo do oszacowania pola powierzchni $\mu(\mathbb{A})$ obszaru \mathbb{A} zawartego w obszarze \mathbb{B} o znanej powierzchni $\mu(\mathbb{B})$ (rys. 8.2). Metoda Monte Carlo w tzw. wersji „orzeł-reszka” polega na wylosowaniu N_B punktów rozłożonych równomiernie (w sensie statystycznym) w obszarze \mathbb{B} i oszacowaniu prawdopodobieństwa η tego, że wylosowany punkt należy również do obszaru \mathbb{A} . Jeśli N_A jest liczbą wylosowanych punktów należących do

tego obszaru, to $\eta \approx N_A / N_B$. Z drugiej strony, prawdopodobieństwo η jest równe stosunkowi pól powierzchni obszarów \mathbb{A} i \mathbb{B} :

$$\eta = \frac{\mu(\mathbb{A})}{\mu(\mathbb{B})} \quad (8.5)$$

a zatem poszukiwane pole powierzchni można oszacować następująco:

$$\mu(\mathbb{A}) \approx \mu(\mathbb{B}) \frac{N_A}{N_B} \quad (8.6)$$



Rysunek 8.2. Ilustracja do przykładu 8.2

Zadanie oszacowania powierzchni obszaru \mathbb{A} można sformułować jako zadanie wyznaczania całki:

$$\mu(\mathbb{A}) = \iint_B \mathbf{1}_{\mathbb{A}}(x, y) dx dy \quad (8.7)$$

gdzie:

$$\mathbf{1}_{\mathbb{A}}(x, y) = \begin{cases} 1 & \text{dla } (x, y) \in \mathbb{A} \\ 0 & \text{dla } (x, y) \notin \mathbb{A} \end{cases}$$

Opis obszaru \mathbb{A} za pomocą nierówności:

$$\mathbb{A} = \{(x, y) \mid x \geq a \wedge x \leq b \wedge y \leq g(x) \wedge y \geq h(x)\} \quad (8.8)$$

umożliwia sprowadzenie zadania szacowania powierzchni obszaru \mathbb{A} do obliczania całki pojedynczej:

$$\mu(\mathbb{A}) = \int_a^b [g(x) - h(x)] dx = \int_a^b F(x) dx \quad (8.9)$$

Jest to sformułowanie identyczne z użytym w przykładzie 8.1, a więc przybliżoną wartość pola powierzchni $\mu(\mathbb{A})$ można w tym przypadku wyznaczyć również za pomocą wariantu podstawowego metody Monte Carlo.

8.2. METODY ESTYMACJI WARTOŚCI OCZEKIWANEJ ZMIENNEJ LOSOWEJ

Niech \underline{y} będzie skalarną zmienną losową o funkcji gęstości prawdopodobieństwa (f.g.p.) $f_y(y)$. Wartość oczekiwana tej zmiennej wyraża się wówczas całką oznaczoną z funkcją wagową $f_y(y)$ postaci:

$$\mu_y = E\{\underline{y}\} \equiv \int_{-\infty}^{\infty} y f_y(y) dy \quad (8.10a)$$

zaś jej wariancja σ_y^2 – całką postaci:

$$\sigma_y^2 = \text{Var}\{\underline{y}\} \equiv E\{(\underline{y} - \mu_y)^2\} = \int_{-\infty}^{\infty} (y - \mu_y)^2 f_y(y) dy \quad (8.10b)$$

Dla metod Monte Carlo zasadniczą rolę odgrywa możliwość estymacji (szacowania) wartości oczekiwanej i wariancji zmiennej losowej \underline{y} na podstawie skończonej liczby niezależnych próbek (realizacji) tej zmiennej:

$$\{y_n\} \equiv \{y_n | n=1,2,\dots,N\}$$

Do tego celu służą *estymatory*. Ich konstrukcja opiera się na modelowaniu zbioru wszystkich możliwych próbek, jakie mogą pojawić się na n -tej pozycji ciągu $\{y_n\}$ za pomocą zmiennej losowej y_n o rozkładzie identycznym z rozkładem zmiennej losowej \underline{y} . Estymatory są funkcjami zmiennych losowych y_1, \dots, y_N . Wynikiem ich działania są więc zmienne losowe, których parametry zależą od parametrów zmiennych y_1, \dots, y_N . Najczęściej używanym *estymatorem wartości oczekiwanej* $\hat{\mu}_y$ jest średnia arytmetyczna tych zmiennych:

$$\hat{\mu}_y = \frac{1}{N} \sum_{n=1}^N y_n \quad (8.11)$$

przy czym zachodzi równość $E\{\hat{\mu}_y\} = \mu_y$, która oznacza, że tak zdefiniowany estymator jest *nieobciążony*. Jeżeli wariancja zmiennej losowej \underline{y} jest skończona, to wariancja tego estymatora:

$$\text{Var}\{\hat{\mu}_y\} = \frac{1}{N^2} \sum_{n=1}^N \text{Var}\{y_n\} = \frac{\sigma_y^2}{N} \quad (8.12)$$

Zauważmy, że wariancja ta jest N -krotnie mniejsza od wariancji zmiennej losowej \underline{y} . Ponieważ jest ona miarą rozrzutu $\hat{\mu}_y$ wokół wartości średniej μ_y , więc może być uważana za miarę niedokładności estymacji wartości oczekiwanej. Istotne jest również to, że wariancja $\hat{\mu}_y$ maleje do 0, gdy $N \rightarrow \infty$. Oznacza to, że dokładność estymacji może być dowolnie zwiększena przez zwiększenie liczby N użytych próbek losowych y_n .

Dla każdej skończonej liczby próbek N i dla każdej f.g.p. $f_y(y)$ zmiennej losowej \underline{y} rozproszenie estymatora (8.11) można scharakteryzować probabilistycznie za pomocą nierówności Czebyszewa:

$$\Pr\left\{ \left| \hat{\mu}_y - \mu_y \right| \leq \frac{k\sigma_y}{\sqrt{N}} \right\} \geq 1 - \frac{1}{k^2} \quad \text{dla } k \geq 1 \quad (8.13)$$

gdzie $\Pr\{\mathbb{X}\}$ oznacza prawdopodobieństwo zdarzenia \mathbb{X} . Jeżeli dla f.g.p. zmiennej losowej \underline{y} spełnione są założenia centralnego twierdzenia granicznego, to dla $N \rightarrow \infty$ rozkład prawdopodobieństwa średniej arytmetycznej $\hat{\mu}_y$ dąży do rozkładu normalnego (Gaussa) z wartością średnią μ_y oraz wariancją σ_y^2 / N . Dla tego zaś rozkładu:

$$\Pr\left\{ \left| \hat{\mu}_y - \mu_y \right| \leq \frac{k\sigma_y}{\sqrt{N}} \right\} \approx \begin{cases} 68.3\% & \text{dla } k=1 \\ 95.4\% & \text{dla } k=2 \\ 99.7\% & \text{dla } k=3 \end{cases} \quad (8.14)$$

Oszacowania (8.14) są praktycznie użyteczne dla $N \geq 30$.

Estymatory parametrów rozkładu są funkcjami zmiennych losowych. Jeżeli w miejsce zmiennych wstawimy niezależne realizacje tych zmiennych, uzyskamy estymaty wartości odpowiednich parametrów rozkładu. W ten właśnie sposób możemy obliczyć estymatę wartości oczekiwanej μ_y :

$$\hat{\mu}_y = \frac{1}{N} \sum_{n=1}^N y_n \quad (8.15)$$

Wartość tej estymaty zależy od użytych próbek będących realizacjami zmiennych losowych y_n (jakkolwiek przyjęty symbol $\hat{\mu}_y$ nie podkreśla tego związku – dla uproszczenia notacji). Jeśli więc użyjemy innych realizacji zmiennych losowych, uzyskamy inny wynik. Pożądaną cechą estymatorów jest zdolność do generowania przez nie estymat możliwie nieznacznie różniących się od dokładnej wartości estymowanego parametru rozkładu. W języku statystyki oznacza to wymaganie, aby estymator był nieobciążony i miał możliwie małą wariancję.

W metodzie Monte Carlo interesuje nas estymacja wartości oczekiwanej pewnej zmiennej losowej z założoną dokładnością – zazwyczaj charakteryzowaną przez wariancję estymatora bądź pierwiastek kwadratowy z tej wariancji, zwany odchyleniem standardowym. Wiemy też, że wariancję estymatora możemy zmniejszyć, stosując większą liczbę próbek. Na podstawie dostępnych próbek losowych y_n możemy wyznaczyć zarówno wartość oczekiwana, jak i wariancję estymatora.

Nieobciążoną estymatę wariancji estymatora definiuje wzór:

$$\hat{\sigma}_{\hat{\mu}_y}^2 \equiv \frac{1}{N(N-1)} \sum_{n=1}^N (y_n - \hat{\mu}_y)^2 = \frac{1}{N(N-1)} \left[\frac{1}{N} \sum_{n=1}^N y_n^2 - \left(\frac{1}{N} \sum_{n=1}^N y_n \right)^2 \right] \quad (8.16)$$

Estymata wariancji zmiennej losowej \underline{y} jest oczywiście N -krotnie większa, tzn.

$$\hat{\sigma}_y^2 \stackrel{1}{=} \frac{1}{N-1} \sum_{n=1}^N (y_n - \hat{\mu}_y)^2 = \frac{1}{N-1} \left[\sum_{n=1}^N y_n^2 - \frac{1}{N} \left(\sum_{n=1}^N y_n \right)^2 \right] \quad (8.17)$$

Warto zauważyć, że komputerowe obliczanie estymat w (8.16) i (8.17) według wzorów następujących po znaku $\stackrel{1}{=}$ wymaga dwóch odwołań do każdej próbki: raz przy wyznaczaniu średniej arytmetycznej $\hat{\mu}_y$, i drugi raz przy obliczaniu odchyłek od tej średniej $y_n - \hat{\mu}_y$; nieodzowne jest więc pamiętanie wszystkich próbek.

Alternatywna postać wzorów, następująca po znaku $\stackrel{2}{=}$, umożliwia obliczanie obydwu sum jednocześnie, bez konieczności pamiętania próbek. Trzeba się jednak wówczas liczyć z tym, że w arytmetyce zmiennopozycyjnej, na skutek błędów zaokrągleń, estymata wariancji może okazać się ujemna.

Inny sposób reprezentacji rozrzutów estymat polega na konstrukcji tzw. *przedziału ufności*. Dla wartości średniej chodzi o konstrukcję przedziału, w którym znajdzie się zadana (zwykle rzędu 95%) część wszystkich realizacji estymat wartości średniej. Dla naszych celów do konstrukcji tego przedziału można wykorzystać oszacowanie (8.14). W szczególności, ok. 99.7% estymat wartości oczekiwanej przyjmuje (dla $N > 30$) wartości z przedziału $[\mu_y - 3\sigma_y, \mu_y + 3\sigma_y]$, nazywanego *trzysigmowym przedziałem ufności*.

Przykład 8.3. Niech \underline{y} będzie zmienną losową o rozkładzie równomiernym w przedziale $[0, 1]$. Wartość oczekiwana tej zmiennej to:

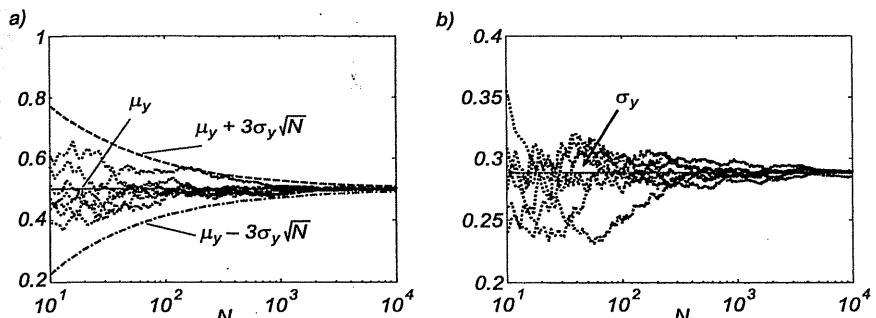
$$\mu_y = \int_0^1 y \, dy = \frac{1}{2}$$

zaś wariancja:

$$\sigma_y^2 = \int_0^1 \left(y - \frac{1}{2} \right)^2 \, dy = \int_{-1/2}^{1/2} z^2 \, dz = \frac{1}{12}$$

W celu ilustracji charakteru zbieżności estymat wartości oczekiwanej i wariancji zmiennej losowej \underline{y} do wartości dokładnych siedmiokrotnie powtórzono następujący eksperyment: wylosowano ciąg 10000 liczb losowych rozłożonych równomiernie w przedziale $[0, 1]$ (generator *rand* programu MATLAB) i wyznaczono ciąg estymat (wartości oczekiwanej i odchylenia standardowego) odpowiadających podciagom N pierwszych próbek dla $N = 10, \dots, 10000$. Na rys. 8.3a widać, że trzysigmowe przedziały ufności (uzyskane dla dokładnej wartości wariancji) dobrze ograniczają przebieg wartości ciągów estymat wartości oczekiwanej dla większych wartości N (poziom ufności 99.7% wg (8.14)). Na rys. 8.3b pokazano natomiast rozrzutu wartości estymat odchylenia standardo-

go zmiennej losowej \underline{y} wokół wartości dokładnej ($1/\sqrt{12}$) i zmniejszanie się tych rozrzutów ze wzrostem N podobne w charakterze do tego z rys. 8.3a.



Rysunek 8.3. Zależność: a) estymat wartości oczekiwanej oraz b) estymat odchylenia standardowego estymatora wartości oczekiwanej od liczby próbek N dla siedmiu ciągów liczb losowych o rozkładzie równomiernym; liniami przerywanymi oznaczono trzysigmowe przedziały ufności wokół wartości średniej

8.3. PROSTE METODY MONTE CARLO

Prosta metoda Monte Carlo (metody złożone będą omawiane w podrozdziale 8.4) służy do wyznaczenia przybliżonej wartości całki o postaci:

$$I = \int_A G(\mathbf{x}) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \quad (8.18)$$

na podstawie próbek $\mathbf{x}_1, \dots, \mathbf{x}_N$ zmiennej wektorowej $\mathbf{x} \in \mathbb{R}^M$, przy założeniu, że funkcja $f_{\mathbf{x}}(\mathbf{x})$ spełnia wymagania stawiane f.g.p. wektorowej zmiennej losowej, tzn. jest nieujemna i spełnia warunek:

$$\int_{\mathbb{R}^M} f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} = 1$$

8.3.1. WARIANT PODSTAWOWY W WERSJI OGÓLNEJ

Wariant podstawowy metody Monte Carlo opiera się na interpretacji całki (8.18) jako wartości oczekiwanej zmiennej losowej $\underline{y} = G(\mathbf{x})$, która jest funkcją wektora zmiennych losowych \mathbf{x} o f.g.p. $f_{\mathbf{x}}(\mathbf{x})$:

$$I = E\{\underline{y}\} = \int_A G(\mathbf{x}) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} = \underbrace{\int_{\mathbb{R}^M} \mathbf{1}_A(\mathbf{x}) G(\mathbf{x}) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}}_{F(\mathbf{x})} \quad (8.19)$$

gdzie $\mathbf{1}_A(\mathbf{x})$ jest funkcją przynależności zbioru $A \subset \mathbb{R}^M$, zdefiniowaną wzorem:

$$\mathbf{1}_A(\mathbf{x}) = \begin{cases} 1 & \text{dla } \mathbf{x} \in A \\ 0 & \text{dla } \mathbf{x} \notin A \end{cases} \quad (8.20)$$

Do estymacji poszukiwanej wartości oczekiwanej zmiennej losowej y można wykorzystać estymator wartości oczekiwanej, uzyskując formułę numerycznego całkowania o postaci:

$$\hat{I} = \frac{1}{N} \sum_{n=1}^N F(\mathbf{x}_n) \quad (8.21)$$

generującą estymaty o wariancji:

$$\hat{\sigma}_i^2 = \frac{1}{N(N-1)} \sum_{n=1}^N \left[F(\mathbf{x}_n) - \frac{1}{N} \sum_{v=1}^N F(\mathbf{x}_v) \right]^2 \quad (8.22)$$

Wzory (8.21) i (8.22) umożliwiają realizację następującej wieloetapowej procedury, zapewniającej uzyskanie przybliżonej wartości całki zadaną trzysigmową dokładnością ΔI_{\max} :

- ① $N = 0$; przyjmij wartość liczby próbek $\Delta N > 30$ pierwszego etapu metody.
- ② Wylosuj ΔN realizacji zmiennej losowej \underline{x} o f.g.p. $f_x(\mathbf{x})$: $\mathbf{x}_{N+1}, \dots, \mathbf{x}_{N+\Delta N}$, a na tej podstawie oblicz wartości $G(\mathbf{x}_{N+1}), \dots, G(\mathbf{x}_{N+\Delta N})$.
- ③ Podstaw $N := N + \Delta N$, a następnie wyznacz estymatę \hat{I} wg wzoru (8.21) oraz estymatę $\hat{\sigma}_i^2$ wg wzoru (8.22).
- ④ Jeżeli $9\hat{\sigma}_i^2 \leq \Delta I_{\max}^2$, to zakończ, ponieważ rozwiązywanie jest już wystarczająco dokładne.
- ⑤ Oszacuj przybliżoną liczbę próbek N koniecznych do uzyskania zadanej dokładności ΔI_{\max} , pamiętając, że wariancja estymaty jest w przybliżeniu odwrotnie proporcjonalna do liczby próbek. Stąd wyznacz liczbę dodatkowych próbek ΔN i przejdź do kroku ②.

8.3.2. WARIANT PODSTAWOWY W WERSJI „ORZEŁ-RESZKA”

Wersja „orzeł-reszka” jest szczególnym przypadkiem wariantu podstawowego metody Monte Carlo, opisanego w podrozdziale 8.2.1, odpowiadającym $G(\mathbf{x}) \equiv 1$:

$$I = \int_A f_x(\mathbf{x}) d\mathbf{x} = \int_{\mathbb{R}^M} \mathbf{1}_A(\mathbf{x}) f_x(\mathbf{x}) d\mathbf{x} = \int_{\mathbb{R}^M} F(\mathbf{x}) f_x(\mathbf{x}) d\mathbf{x} \quad (8.23)$$

Ze względu na to, że funkcja $F(\mathbf{x}) \equiv \mathbf{1}_A(\mathbf{x})$ jest dwuwartościowa, estymaty wartości oczekiwanej oraz odchylenia standardowego mają szczególnie prostą postać:

$$\hat{I} = \frac{1}{N} \sum_{n=1}^N \mathbf{1}_A(\mathbf{x}_n) = \frac{N_A}{N} \quad (8.24)$$

$$\hat{\sigma}_i^2 = \frac{1}{N(N-1)} \left\{ \sum_{n=1}^N [\mathbf{1}_A(\mathbf{x}_n)]^2 - \frac{1}{N} \left[\sum_{n=1}^N \mathbf{1}_A(\mathbf{x}_n) \right]^2 \right\} = \frac{N\hat{I} - (N\hat{I})^2 / N}{N(N-1)} = \frac{\hat{I}(1-\hat{I})}{N-1} \quad (8.25)$$

gdzie N_A jest liczbą wylosowanych próbek \mathbf{x}_n , które należą do zbioru A , a N – liczbą wszystkich próbek.

Metodę Monte Carlo w wersji „orzeł-reszka” można również zrealizować w postaci wieloetapowej procedury, zapewniającej uzyskanie estymaty o zadanej trzysigmowej dokładności ΔI_{\max} .

- ① $N = 0$; $N_A = 0$; przyjmij wartość liczby próbek $\Delta N > 30$ pierwszego etapu metody.
- ② Wylosuj ΔN realizacji zmiennej losowej \underline{x} o f.g.p. $f_x(\mathbf{x})$: $\mathbf{x}_{N+1}, \dots, \mathbf{x}_{N+\Delta N}$, oblicz wartości $\mathbf{1}_A(\mathbf{x}_{N+1}), \dots, \mathbf{1}_A(\mathbf{x}_{N+\Delta N})$, a ich sumę zapamiętaj w zmiennej ΔN_A .
- ③ Podstaw $N := N + \Delta N$ i $N_A := N_A + \Delta N_A$, a następnie wyznacz estymatę \hat{I} wg wzoru (8.24) oraz estymatę $\hat{\sigma}_i^2$ wg wzoru (8.25).
- ④ Jeżeli $9\hat{\sigma}_i^2 \leq \Delta I_{\max}^2$, to zakończ, ponieważ rozwiązywanie jest już wystarczająco dokładne.
- ⑤ Na podstawie (8.25) oszacuj przybliżoną liczbę nowych próbek $\Delta N = 9\hat{I}(1-\hat{I}) / \Delta I_{\max}^2 - N$ koniecznych do uzyskania zadanej dokładności ΔI_{\max} . Przejdź do kroku ②.

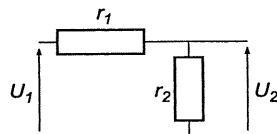
Zauważmy, że metoda Monte Carlo w wersji „orzeł-reszka” nadaje się do rozwiązania zadania sformułowanego w przykładzie 8.2, tj. do estymacji powierzchni figury płaskiej. Jednym z praktycznie ważnych zastosowań tego wariantu jest ponadto estymacja uzysku produkcyjnego, co ilustruje kolejny przykład.

Przykład 8.4. Dzielnik napięcia przedstawiony na rys. 8.4 powinien zapewniać tłumienie napięciowe bez obciążenia:

$$k(\mathbf{r}) = \frac{U_2}{U_1} = \frac{r_1}{r_1 + r_2} \quad (8.26)$$

o wartości 0.5 z dokładnością 2%; $\mathbf{r} = [r_1 \ r_2]^T$. Należy przyjąć, że rezystancje r_1 i r_2 mają rozrzuty produkcyjne na poziomie 5%, które można przedstawić za pomocą niezależnych zmiennych losowych r_1 i r_2 o f.g.p. odpowiednio $f_{r_1}(r_1)$ i $f_{r_2}(r_2)$. Trzeba wyznaczyć estymatę uzysku produkcyjnego Y , czyli średnie-

go odsetka układów sprawnych, z trzysigmową dokładnością $\Delta Y_{\max} = 10^{-2}$ oraz $\Delta Y_{\max} = 10^{-3}$.



Rysunek 8.4. Rezystorowy dzielnic napięcia z przykładu 8.4

Uzysk produkcyjny można zapisać w postaci całkowej jako:

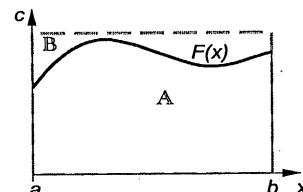
$$Y = \int_{\mathbb{R}^2} \mathbf{1}_A(k(\mathbf{r})) f_r(\mathbf{r}) d\mathbf{r} \quad (8.27)$$

gdzie $f(\mathbf{r}) = f_{r_1}(r_1)f_{r_2}(r_2)$, a obszar sprawności A jest przedziałem $[0.49, 0.51]$. Stosując do tak sformułowanego zadania wieloetapową procedurę Monte Carlo, uzyskano: $\hat{Y} \approx 0.637$, $3\sigma_{\hat{Y}} \approx 0.0098$, $N = 21772$ dla $\Delta I_{\max} = 10^{-2}$ oraz $\hat{Y} \approx 0.6398$, $3\sigma_{\hat{Y}} \approx 9.8 \cdot 10^{-4}$, $N = 2177101$ dla $\Delta I_{\max} = 10^{-3}$. Zgodnie z teoretycznymi przewidywaniami dziesięciokrotne zwiększenie dokładności wymagało około stu-krotnego zwiększenia liczby próbek.



Metodę Monte Carlo w wersji „orzeł-reszka” można zastosować również w przypadku, gdy opis brzegu obszaru A nie ma jawniej postaci funkcyjnej, ale dla każdego punktu obszaru B , zawierającego A , możemy stwierdzić algorytmicznie, czy należy on również do A . Można ją także stosować do niektórych ogólnych zadań całkowania (8.18), w których funkcja podcałkowa nie jest dwuwartościowa, jak pokazuje kolejny przykład.

Przykład 8.5. Rozważmy zadanie całkowania z przykładu 8.1, zakładając, że funkcja $F(x)$ we wzorze (8.1) jest nieujemna w przedziale $[a, b]$ i znana jest taka liczba c , że $c \geq \sup \{F(x) \mid a \leq x \leq b\}$.



Rysunek 8.5. Ilustracja zastosowania metody Monte Carlo w wersji „orzeł-reszka” do całkowania funkcji

Obszar całkowania A zawiera się więc w prostokącie B , którego pole powierzchni wyraża się wzorem: $\mu(B) = (b-a)(c-0)$ (rys. 8.5). Do oszacowania pola powierzchni $\mu(A) = I$ obszaru A możemy więc posłużyć się metodą Monte Carlo w wersji „orzeł-reszka”, przeprowadzając losowanie z rozkładem równomiernym w obszarze B .



8.4. ZŁOŻONE METODY MONTE CARLO

Z przedstawionych dotychczas przykładów wynikają następujące wnioski:

- Realizacja metody Monte Carlo nie wymaga znajomości postaci f.g.p. zmiennych losowych modelujących próbki, a jedynie umiejętności generacji niezależnych próbek.
- Dokładność metody Monte Carlo nie zależy od krotności całki, ani od gładkości całkowanej funkcji.
- Dokładność estymacji można szacować w trakcie prowadzenia obliczeń, co pozwala przerwać je z chwilą osiągnięcia założonej dokładności.

Do istotnych wad metody Monte Carlo należy zaliczyć wolną zbieżność, rozumianą jako zależność wariancji estymatora \hat{I} całki I od liczby próbek N , która ma postać: $\text{Var}\{\hat{I}\} \approx c/N$. Poprawę dokładności można uzyskać, modyfikując prostą metodę w taki sposób, aby zmniejszyć wartość współczynnika c w tym oszacowaniu. W dalszej części tego podrozdziału zostaną przedstawione najważniejsze modyfikacje służące temu celowi.

8.4.1. METODA LOSOWANIA WAŻONEGO

Dokładność wyznaczania całki (8.19) przy użyciu podstawowej metody Monte Carlo zależy zarówno od liczby próbek N , jak i od wariancji zmiennej losowej $y = \mathbf{1}_A(\mathbf{x})G(\mathbf{x})$. Przy danej liczbie próbek zwiększenie dokładności można uzyskać przez taką transformację całki (8.19), aby mieć do czynienia z nową zmienią losową o mniejszej wariancji, ale tej samej wartości oczekiwanej. Niech $g_x(\mathbf{x})$ będzie f.g.p. taką, że zmienią $z(\mathbf{x}) \equiv G(\mathbf{x})f_x(\mathbf{x})/g_x(\mathbf{x})$ jest ograniczona dla każdego $\mathbf{x} \in A$.

Metoda losowania ważonego polega na przekształceniu całki (8.19) do postaci:

$$I = \int_A \left[G(\mathbf{x}) \frac{f_x(\mathbf{x})}{g_x(\mathbf{x})} \right] g_x(\mathbf{x}) d\mathbf{x} = E\{z(\underline{\mathbf{x}})\} \quad (8.28)$$

Wynika z niej możliwość estymacji całki I za pomocą wzoru:

$$\hat{I} = \frac{1}{N} \sum_{n=1}^N z(\mathbf{x}_n) = \frac{1}{N} \sum_{n=1}^N \frac{G(\mathbf{x}_n) f_x(\mathbf{x}_n)}{g_x(\mathbf{x}_n)} \quad (8.29)$$

Zmienna losowa z ma taką samą wartość oczekiwana jak zmienna losowa $y = \mathbf{1}_A(\mathbf{x})G(\mathbf{x})$ oraz wariancję zależną od f.g.p. $g_x(\mathbf{x})$. Okazuje się, że najmniejszą wartość wariancja ta osiąga dla:

$$g_x(\mathbf{x}) = \frac{\mathbf{1}_A(\mathbf{x}) |G(\mathbf{x})| f_x(\mathbf{x})}{\int_A |G(\mathbf{x})| f_x(\mathbf{x}) d\mathbf{x}} \quad (8.30)$$

Jeżeli $G(x)$ jest funkcją nieujemną, to minimalna wariancja estymatora ważonego jest równa 0. Niestety, aby skonstruować estymator z zerową wariancją, należałoby z góry znać wartość całki w mianowniku powyższego wzoru. Zastosowanie wzoru (8.30) do wyznaczenia funkcji $g_x(x)$, pozwalającej na redukcję wariancji, wymaga w praktyce odpowiednio dobrego przybliżenia funkcji $G(x)$ przez pewną funkcję $\hat{G}(x)$, której całkę można łatwo obliczyć. Znane są ważne przykłady praktycznego wykorzystania omawianej metody, między innymi w dziedzinie statystycznej optymalizacji układów elektronicznych oraz w dziedzinie statystycznej symulacji cyfrowych systemów telekomunikacyjnych [S2].

8.4.2. METODA ZMIENNEJ KONTROLNEJ

Metoda zmiennej kontrolnej opiera się na następującej dekompozycji całki (8.19):

$$I = \int_{\mathbb{A}} \hat{G}(x) f_x(x) dx + \int_{\mathbb{A}} [G(x) - \hat{G}(x)] f_x(x) dx \quad (8.31)$$

gdzie $\hat{G}(x)$ jest taką aproksymacją funkcji $G(x)$, że pierwszy składnik prawej strony (8.31) jest dużo większy od drugiego i może być łatwo wyznaczony analitycznie lub numerycznie. Zakłada się przy tym, że zmieniona losowa $G(x) - \hat{G}(x)$ ma znacznie mniejszą wariancję niż $G(x)$ i dlatego drugi składnik prawej strony (8.31) stosunkowo dokładnie może być przybliżony za pomocą metody Monte Carlo. Efektywność metody zilustrowano w przykładzie 8.6.

8.4.3. METODA OPARTA NA OBNIŻANIU KROTNOSCI CAŁKI

Obniżenie krotności całki I jest możliwe wtedy, gdy istnieje taka dekompozycja wektora oryginalnych zmiennych losowych:

$$\underline{x} = \begin{bmatrix} \underline{u} \\ \underline{v} \end{bmatrix}$$

i obszaru całkowania $\mathbb{A} = \mathbb{A}_{\underline{u}} \times \mathbb{A}_{\underline{v}}$, że $f_x(\underline{x}) = f_{\underline{u}}(\underline{u}) f_{\underline{v}}(\underline{v})$ dla $\underline{u} \in \mathbb{A}_{\underline{u}}$ oraz $\underline{v} \in \mathbb{A}_{\underline{v}}$, a w konsekwencji:

$$I = \int_{\mathbb{A}_{\underline{u}}} \left\{ \int_{\mathbb{A}_{\underline{v}}} G(\underline{x}(\underline{u}, \underline{v})) f_{\underline{v}}(\underline{v}) d\underline{v} \right\} f_{\underline{u}}(\underline{u}) d\underline{u} \quad (8.32)$$

Zmienna losowa $\underline{z} = \int_{\mathbb{A}_{\underline{v}}} G(\underline{x}(\underline{u}, \underline{v})) f_{\underline{v}}(\underline{v}) d\underline{v}$ ma na ogół mniejszą wariancję niż $G(\underline{x})$ i dlatego całka zewnętrzna we wzorze (8.32) może być stosunkowo łatwo

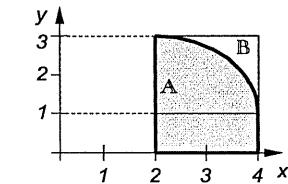
wyznaczona metodą Monte Carlo, jeśli tylko całka wewnętrzna może być wyznaczona dostatecznie dokładnie i szybko metodą analityczną lub numeryczną. Efektywność tej metody ilustruje następujący przykład.

Przykład 8.6. Należy wyznaczyć przybliżoną wartość całki:

$$I = \int_{\mathbb{A}} dy dx \quad (8.33)$$

gdzie $\mathbb{A} = \{(x, y) | 2 \leq x \leq 4, G(x) \geq y\}$, przy czym $G(x) = \sqrt{4 - (x - 2)^2}$.

Można to zrobić przy użyciu prostej metody Monte Carlo w wersji „orzel-reszka” i w wariancie *podstawowym*, a następnie przy użyciu metody *zmiennej kontrolnej*.



Rysunek 8.6. Obszar całkowania w przykładzie 8.6

Poszukiwana całka jest równa polu $\mu(\mathbb{A})$ powierzchni obszaru \mathbb{A} na rys. 8.6. Metoda Monte Carlo w wersji „orzel-reszka” przybliża stosunek η pola tego obszaru do pola $\mu(\mathbb{B})$ prostokąta \mathbb{B} za pomocą częstości trafiania w obszarze \mathbb{A} w przypadku statystycznie równomiernego losowania punktów z obszarem \mathbb{B} :

$$\eta = \frac{\mu(\mathbb{A})}{\mu(\mathbb{B})} = \frac{\int_0^1 \int_0^1 \mathbf{1}_{\mathbb{A}}(x, y) dy dx}{\mu(\mathbb{B})} = \int_0^1 \int_0^1 \mathbf{1}_{\mathbb{A}}(x, y) f_{xy}(x, y) dy dx \quad (8.34)$$

gdzie $f_{xy}(x, y) = 1 / \mu(\mathbb{B}) = 1 / 6$, ponieważ $\mu(\mathbb{B}) = 6$. W celu przeprowadzenia odpowiedniego eksperymentu statystycznego należy więc posłużyć się generatorem liczb pseudolosowych podlegających rozkładowi opisanemu następującą f.p.g.:

$$f_{x,y}(x, y) = \begin{cases} 1 & \text{dla } x \in [2, 4] \wedge y \in [0, 3] \\ 0 & \text{dla } x \notin [2, 4] \vee y \notin [0, 3] \end{cases}$$

Wówczas $\hat{I} = \hat{\eta} \mu(\mathbb{B}) = 6N_{\mathbb{A}} / N_{\mathbb{B}}$, gdzie $N_{\mathbb{B}}$ jest liczbą wszystkich wylosowanych punktów, a $N_{\mathbb{A}}$ jest liczbą punktów należących do obszaru \mathbb{A} . Przeprowadzając obliczenia analogiczne do tych we wzorze (8.25), ale funkcji $G(x)$ przybierającej wartości 6 i 0, zamiast 1 i 0, można wyznaczyć odchylenie standardowe powyżej estymaty:

$$\sigma_i = \mu(\mathbb{B}) \sqrt{\frac{\hat{\eta}(1 - \hat{\eta})}{N - 1}} = 6 \sqrt{\frac{\hat{\eta}(1 - \hat{\eta})}{N - 1}} \quad (8.35)$$

Metodę podstawową stosuje się do całki o postaci (8.18). Dlatego całkę (8.33) trzeba doprowadzić do tej postaci, używając techniki obniżania krotności całki:

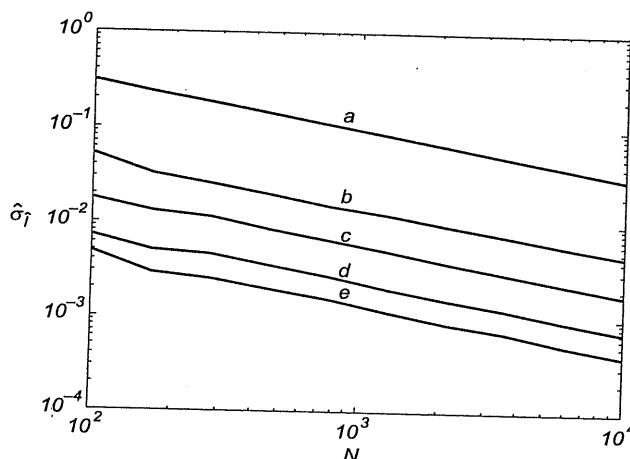
$$\begin{aligned} I &= \int_{\mathbb{A}} dy dx = \int_2^4 \left[\int_0^3 \mathbf{1}_{\mathbb{A}}(x, y) dy \right] dx = \int_{-\infty}^{\infty} \frac{[(4-2)G(x)]}{F(x)} \cdot \left[\frac{\mathbf{1}_{[2,4]}(x)}{4-2} \right] dx = \\ &= \int_{-\infty}^{\infty} F(x) \cdot f_x(x) dx \end{aligned} \quad (8.36)$$

Do estymacji tej całki można wykorzystać losowanie wartości x zgodnie z rozkładem $f_x(x)$, równomiernym w przedziale $[2, 4]$. Odpowiednie estymaty całki i odchylenia standardowego estymatora można wyznaczyć na podstawie formuł (8.21) i (8.22):

$$\hat{I} = \frac{1}{N} \sum_{n=1}^N F(x_n) = \frac{2}{N} \sum_{n=1}^N \left[\sqrt{4 - (x_n - 2)^2} + 1 \right]$$

$$\hat{\sigma}_I^2 = \frac{1}{N(N-1)} \sum_{n=1}^N \left[F(x_n) - \frac{1}{N} \sum_{m=1}^N F(x_m) \right]^2$$

Metodę zmiennej kontrolnej, omówioną w podrozdziale 8.3.2, zastosowano również do postaci (8.36) całki. Jako funkcje aproksymujące wybrano wielomiany stopnia $r = 1, 2, 4$ oparte na 21 węzłach równomiernie rozłożonych w przedziale całkowania $[2, 4]$.



Rysunek 8.7. Zależność etymaty odchylenia standardowego $\hat{\sigma}_I$ estymat całki \hat{I} od liczby próbek N : a – metoda podstawowa w wersji „orzel-reszka”; b – metoda podstawowa w wersji ogólnej; c, d i e – metoda zmiennej kontrolnej z wielomianami aproksymującymi opartymi na 21 węzłach rzędu odpowiednio 1, 2 i 4

Na rys. 8.7 przedstawiono zależność odchylenia standardowego $\hat{\sigma}_I$ estymat \hat{I} całki I od liczby próbek N , otrzymanych za pomocą trzech omówionych metod całkowania. Widoczna jest lepsza dokładność metod wykorzystujących wzór (8.36) niż metody podstawowej w wersji „orzel-reszka” oraz lepsza dokładność metody zmiennej kontrolnej niż metody podstawowej w wersji ogólnej. Najbardziej dokładną z badanych metod (i jednocześnie bardzo prostą w implementacji) okazała się metoda zmiennej kontrolnej z wielomianowym przybliżeniem całkowanej funkcji czwartego stopnia.

8.4.4. METODA LOSOWANIA WARSTWOWEGO

W metodzie losowania warstwowego obszar całkowania \mathbb{A} dzieli się na K rozłącznych obszarów $\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_K$ i wyznacza całkę I jako sumę całek:

$$I_k = \int_{\mathbb{A}_k} G(\mathbf{x}) f_x(\mathbf{x}) d\mathbf{x} = \mu(\mathbb{A}_k) \int_{\mathbb{A}_k} \frac{G(\mathbf{x}) f_x(\mathbf{x})}{\mu(\mathbb{A}_k)} d\mathbf{x} \quad \text{dla } k = 1, 2, \dots, K \quad (8.37)$$

gdzie $\mu(\mathbb{A}_k) = \int f_x(\mathbf{x}) d\mathbf{x}$. Całki I_k oblicza się za pomocą dowolnej wersji metody Monte Carlo, np. za pomocą podstawowego wariantu prostej metody Monte Carlo:

$$\hat{I}_k = \frac{\mu(\mathbb{A}_k)}{N_k} \sum_{n=1}^{N_k} \mathbf{1}_{\mathbb{A}_k}(\mathbf{x}_n^{(k)}) G(\mathbf{x}_n^{(k)})$$

przy czym wektory $\{\mathbf{x}_n^{(k)} | n = 1, 2, \dots, N_k\}$ są próbami wylosowanymi w celu wyznaczenia estymat odpowiednich I_k zgodnie z rozkładem, którego f.g.p ma postać:

$$f_{x,k}(\mathbf{x}) = \mathbf{1}_{\mathbb{A}_k}(\mathbf{x}) \frac{f_x(\mathbf{x})}{\mu(\mathbb{A}_k)}$$

Mogą pokazać [F1], że jeśli liczby próbek wyrażają się wzorem $N_k = N \mu(\mathbb{A}_k)$ i są one liczbami całkowitymi, to suma estymat cząstkowych:

$$\hat{I} = \sum_{k=1}^K \hat{I}_k$$

ma dokładność nie gorszą niż estymata metody prostej dla całki pierwotnej.

8.5. METODY GENERACJI ZMIENNYCH LOSOWYCH

Do realizacji metody Monte Carlo niezbędna jest umiejętność generacji ciągu liczb losowych o zadanym rozkładzie i dowolnej (a przynajmniej bardzo dużej) długości. W komputerowej symulacji statystycznej używa się zazwyczaj liczb pseudo-

losowych, tj. okresowych ciągów liczbowych, jednoznacznie określonych przez wartość początkową (lub wartości początkowe) oraz pewien algorytm iteracyjny, zwany *generatorem liczb pseudolosowych*. Ciągi liczb pseudolosowych, choć są w istocie zdeterminowane (nielosowe), mogą spełniać statystyczne testy zgodności z założonym rozkładem i dlatego są używane w algorytmach wymagających liczb losowych w ścisłym sensie. We współczesnych systemach komputerowych bywają też dostępne generatory liczb losowych w ścisłym sensie, w których wykorzystuje się zjawiska fizyczne ze swojej natury losowe, takie jak szum elektryczny w specjalnych elementach elektronicznych czy momenty wystąpienia różnych nieprzewidywalnych zdarzeń w aktywnym systemie komputerowym.

W tym rozdziale zostaną omówione wybrane sposoby algorytmicznej generacji liczb pseudolosowych.

8.5.1. GENERATORY LICZB PSEUDOLOSOWYCH O ROZKŁADZIE RÓWNOMIERNYM

Generator liczb pseudolosowych o rozkładzie równomiernym w przedziale $[0, 1]$ ma charakter podstawowy, ponieważ służy do konstrukcji generatorów liczb pseudolosowych o innych rozkładach. Większość obecnie używanych generatorów liczb pseudolosowych o rozkładzie równomiernym, to generatorzy liniowe oparte na metodzie kongruencyjnej (Lehmera):

$$x_{i+1} = (a_0 x_i + a_1 x_{i-1} + \dots + a_k x_{i-k} + b) \pmod{M} \quad (8.38)$$

gdzie wszystkie zmienne mogą przybierać jedynie wartości całkowite, zaś (\pmod{M}) oznacza obliczanie wartości wyrażenia modulo M . Generowane liczby mają rozkład równomierny w przedziale $[0, M-1]$, a zatem w celu sprowadzenia ich do przedziału $[0, 1]$ należy podzielić je przez $M-1$. Najczęściej używane są generatorzy multiplikatywne:

$$x_{i+1} = a x_i \pmod{M} \quad (8.39)$$

oraz mieszane:

$$x_{i+1} = (a x_i + b) \pmod{M} \quad (8.40)$$

Obecnie uważa się, że generatorzy mieszane nie są lepsze od multiplikatywnych. Przykłady parametrów generatorów multiplikatywnych, które dają dobre wyniki wg [P], to:

$$M = 2147483647, \quad a = 16807, \quad 48271, \quad 69621$$

Wszystkie one mają okres równy $M-1$, tj. 2147483646. Z analizy generatorów multiplikatywnych wynika, że mogą one być używane tylko wtedy, gdy końcowe bity reprezentacji generowanych liczb nie odgrywają istotnej roli. Ostatnie bity,

tzn. najmłodsze, są bowiem znacznie mniej losowe niż najstarsze (np. ostatni bit jest stale równy 1).

W praktyce trzeba liczyć się z istnieniem autokorelacji w ciągu liczb pseudolosowych. W ciągu pochodząącym z generatora multiplikatywnego bierze się ona stąd, że kolejne próbki różnią się a -krotnie. Jeżeli $x_i \in (1, a)$, tzn. x_i jest liczbą małą w porównaniu z zakresem zmienności M , to $x_{i+1} \in (a, a^2)$ jest również liczbą małą w tym sensie. Tymczasem w idealnym generatorze prawdopodobieństwo uzyskania wartości „dużych” po wartości „małej” jest takie samo jak prawdopodobieństwo uzyskania wartości „małych”. O sposobach redukcji autokorelacji ciągów liczb pseudolosowych przeczytać można np. w książce [P].

Jeżeli trzeba wygenerować wektor zmiennych losowych o rozkładzie równomiernym w wielowymiarowej kostce jednostkowej $[0, 1] \times [0, 1] \times \dots$, to za kolejne składowe można przyjąć dowolne niezależne realizacje zmiennej losowej o rozkładzie równomiernym w przedziale $[0, 1]$.

8.5.2. GENERATORY LICZB PSEUDOLOSOWYCH O ZADANYM ROZKŁADZIE PRAWDOPODOBIEŃSTWA

Omówione zostaną dwa najważniejsze sposoby otrzymywania ciągów liczb pseudolosowych o rozkładzie zadanym dystrybuantą lub f.g.p. – metoda odwracania dystrybuanty oraz metoda transformacji zmiennych. Metody te zakładają, że dostępne jest źródło liczb (pseudo)losowych $\{u_1, u_2, \dots\}$ o rozkładzie równomiernym w przedziale $[0, 1]$.

Dystrybuanta $F_x(x)$ skalarnej zmiennej losowej x jest ciągłą i ściśle rosnącą funkcją taką, że $F_x(-\infty) = 0$ i $F_x(+\infty) = 1$; w przedziale $[0, 1]$ istnieje zatem monotoniczna funkcja odwrotna $F_x^{-1}(u)$. Metoda odwracania dystrybuanty opiera się na spostrzeżeniu, że zmienna losowa $x = F_x^{-1}(u)$ ma rozkład o dystrybuancie $F_x(x)$, ponieważ:

$$\Pr\{\underline{x} \leq x\} = \Pr\{F_x^{-1}(\underline{u}) \leq x\} = \Pr\{\underline{u} \leq F_x(x)\} = F_x(x) \quad (8.41)$$

Ciąg liczb pseudolosowych $\{F_x^{-1}(u_1), F_x^{-1}(u_2), \dots\}$ ma więc pożądany rozkład. Główna trudność tej metody wiąże się z odwracaniem dystrybuanty. Jeżeli bowiem $F_x^{-1}(u)$ nie można wyrazić analitycznie, to konieczne jest rozwiążanie nieliniowego równania algebraicznego $u_n = F_x(x_n)$ dla $n = 1, 2, \dots$

Przykład 8.7. Funkcja odwrotna do dystrybuanty rozkładu wykładniczego:

$$F_x(x) = \begin{cases} 1 - e^{-\lambda x} & \text{dla } x \geq 0 \\ 0 & \text{dla } x < 0 \end{cases}$$

gdzie $\lambda > 0$, ma postać:

$$x = -\lambda^{-1} \ln(1-u) \quad \text{dla } u \in (0, 1)$$

Ponieważ ciąg liczb $\{1-u_1, 1-u_2, \dots\}$ ma rozkład równomierny w przedziale $[0, 1]$, więc do generacji liczb o rozkładzie wykładniczym można użyć uproszczonej formuły: $x_n = -\lambda^{-1} \ln(u_n)$.

Przykład 8.8. Do generacji rozkładu dyskretnego:

$$\Pr\{\underline{x} = k\} = p_k \quad \text{dla } k = 0, 1, \dots$$

gdzie $p_k \in [0, 1]$ i $\sum_k p_k = 1$, można wykorzystać transformację:

$$x_n = \min_k \left\{ k \in \mathbb{N} \mid u_n \leq \sum_{j=0}^k p_j \right\}$$

Metoda transformacji zmiennych, w swej ogólnej postaci, polega na przekształceniu wektora zmiennych losowych \underline{u} , opisanego znaną f.g.p. $f_{\underline{u}}(\underline{u})$, w wektor zmiennych losowych :

$$\underline{x} = \Phi(\underline{u}) \quad (8.42)$$

którego rozkład charakteryzuje zadana f.g.p. $f_{\underline{x}}(\underline{x})$. Punktem wyjścia do raczej heurystycznych poszukiwań odpowiedniej transformacji jest zależność znana z rachunku prawdopodobieństwa:

$$f_{\underline{x}}(\underline{x}) = \det(\mathbf{J}) f_{\underline{u}}(\Phi^{-1}(\underline{x})) \quad (8.43)$$

gdzie \mathbf{J} jest jacobianem przekształcenia odwrotnego, tzn. macierzą zawierającą wszystkie pochodne składowych wektora \underline{u} względem składowych wektora \underline{x} .

Przykład 8.9. Zbadajmy następującą transformację zmiennych:

$$x_i = \sqrt{-2 \ln(u_i)} \cos(2\pi u_{i+1}), \quad x_{i+1} = \sqrt{-2 \ln(u_i)} \sin(2\pi u_{i+1}) \quad (8.44)$$

Przekształcenie odwrotne ma postać:

$$u_i = \exp\left(-\frac{x_i^2 + x_{i+1}^2}{2}\right), \quad u_{i+1} = \frac{1}{2\pi} \arctg\left(\frac{x_{i+1}}{x_i}\right)$$

Można sprawdzić, że wyznacznik macierzy Jacobiego ma w tym przypadku postać:

$$\det(\mathbf{J}) = -\left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x_i^2}{2}\right) \right] \left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x_{i+1}^2}{2}\right) \right]$$

co oznacza, że w wyniku transformacji uzyskaliśmy parę niezależnych zmiennych losowych o rozkładach normalnych z zerową średnią i jednostkową warian-

cją. Posługując się wzorem (8.44) oraz generatorem liczb pseudolosowych o rozkładzie równomiernym $\{u_1, u_2, \dots\}$, można uzyskać ciąg liczb pseudolosowych o rozkładzie normalnym $\{x_1, x_2, \dots\}$.

Przykład 8.10. W praktyce często zachodzi potrzeba przekształcenia N -wymiarowego wektora zmiennych losowych \underline{u} o standardowym rozkładzie normalnym, którego f.g.p. ma postać:

$$f_{\underline{x}}(\underline{u}) = \frac{1}{(2\pi)^{N/2}} \exp\left(-\frac{\underline{u}^T \underline{u}}{2}\right)$$

w wektor skorelowanych zmiennych losowych \underline{x} o rozkładzie normalnym z zerową średnią i macierzą kowariancji \mathbf{C} . Można do tego celu użyć transformacji $\underline{x} = \mathbf{L}\underline{u}$, gdzie \mathbf{L} jest dolną macierzą trójkątną taką, że $\mathbf{L}\mathbf{L}^T = \mathbf{C}$. Wyznacznik macierzy Jacobiego przekształcenia odwrotnego jest wówczas równy: $\det(\mathbf{L}^{-1}) = [\det(\mathbf{L})]^{-1}$. Ponieważ $\det(\mathbf{C}) = \det(\mathbf{L})\det(\mathbf{L}^T) = [\det(\mathbf{L})]^2$, więc zgodnie ze wzorem (8.43) otrzymujemy się pożądanego wektora skorelowanych zmiennych losowych o f.g.p.:

$$f_{\underline{x}}(\underline{x}) = \frac{1}{(2\pi)^{N/2} \sqrt{\det(\mathbf{C})}} \exp\left(-\frac{\underline{x}^T \mathbf{C}^{-1} \underline{x}}{2}\right)$$

Zadanie 8.1. Korzystając z metody Monte Carlo, znaleźć objętość torusa zdefiniowanego nierównością $(\sqrt{x^2 + y^2} - 3)^2 + z^2 \leq 1$.

Odp.: $6\pi^2$.

Zadanie 8.2. Objętość K -wymiarowej kuli o promieniu 1 wyraża się wzorem:

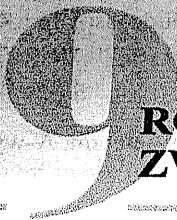
$$V_K = \frac{\pi^{K/2}}{\Gamma\left(\frac{K}{2} + 1\right)}$$

Wyznaczyć zależność odchylenia standardowego estymat tej objętości, uzyskanych za pomocą metody Monte Carlo w wersji „orzeł-reszka”, od wymiarowości kuli dla $K = 1, 2, \dots, 10$.

Zadanie 8.3. Zbadać, na przykładzie dwóch całek:

$$I_1 = \int_{-1}^1 x^2 dx, \quad I_2 = \int_{-1}^1 |x| dx,$$

czy szybkość zmniejszania się odchylenia standardowego wybranego wariantu metody Monte Carlo przy zwiększaniu liczby próbek zależy od gładkości funkcji podcałkowej.



ROZWIĄZYWANIE RÓWNAŃ RÓŻNICZKOWYCH ZWYCZAJNYCH

Umiejętność przewidywania odpowiedzi układu dynamicznego na pobudzenia zewnętrzne jest niezbędna w projektowaniu inżynierskim, np. w projektowaniu układów impulsowych, cyfrowych, czy też systemów automatycznego sterowania. Wynika stąd potrzeba rozwiązywania (całkowania) równań różniczkowych zwyczajnych, przy czym w znakomitej większości praktycznie ważnych zadań inżynierskich można to zrobić jedynie metodami numerycznymi. W tym rozdziale zajmować się będziemy takimi właśnie metodami przeznaczonymi do rozwiązywania tzw. *zagadnienia początkowego* (zagadnienia Cauchy'ego), polegającego na wyznaczaniu M funkcji $y_1(t), \dots, y_M(t)$, które w przedziale $t \in [t_0, t_0 + T]$ spełniają układ M równań różniczkowych zwyczajnych:

$$\frac{dy_m}{dt} = f_m(t, y_1, \dots, y_M) \quad \text{dla } m=1, \dots, M \quad (9.1)$$

oraz zestaw M warunków początkowych: $y_1(t_0) = y_{1,0}, \dots, y_M(t_0) = y_{M,0}$. W zapisie wektorowym zagadnienie to można sformułować następująco: wyznaczyć wektor funkcji $\mathbf{y}(t) = [y_1(t) \ y_2(t) \ \dots \ y_M(t)]^T$, który w przedziale $t \in [t_0, t_0 + T]$ spełnia równanie wektorowe:

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}) \quad (9.2)$$

oraz warunek $\mathbf{y}(t_0) = \mathbf{y}_0$, gdzie $\mathbf{f}(t, \mathbf{y}) = [f_1(t, \mathbf{y}) \ f_2(t, \mathbf{y}) \ \dots \ f_M(t, \mathbf{y})]^T$ i $\mathbf{y}_0 = [y_{0,1} \ y_{0,2} \ \dots \ y_{0,M}]^T$. Rozwiązywanie zagadnienia początkowego istnieje, jest jednoznaczne i różniczkowalne w przedziale $t \in [t_0, t_0 + T]$, jeśli funkcje $f_m(t, \mathbf{y})$ we wszystkich punktach zbioru $\mathbb{D} = \{(t, \mathbf{y}) \mid 0 \leq t - t_0 \leq T, \mathbf{y} \in \mathbb{R}^M\}$ są ciągłe oraz spełniają warunek Lipschitza względem \mathbf{y} :

$$\exists L > 0, \forall t \in [0, T], \forall \mathbf{y}, \bar{\mathbf{y}} \in \mathbb{D} : |f_m(t, \mathbf{y}) - f_m(t, \bar{\mathbf{y}})| \leq L \|\mathbf{y} - \bar{\mathbf{y}}\| \quad (9.3)$$

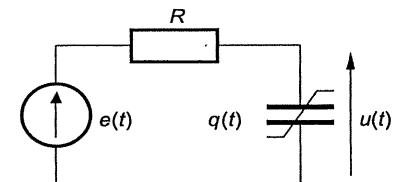
W elektronice zmienna niezależna t ma zazwyczaj sens czasu, natomiast funkcje $y_m(t)$ reprezentują zmienne w czasie: ładunki elektryczne zgromadzone w kondensatorach i strumienie magnetyczne skojarzone z cewkami albo też napięcia i prądy gałęziowe [O]. Zagadnienie początkowe można więc zinterpretować jako

wyznaczanie odpowiedzi czasowej układu elektronicznego w przedziale czasu $[t_0, t_0 + T]$ dla znanych warunków początkowych.

Przykład 9.1. Równanie dynamiki nieliniowego układu RC, przedstawionego na rys. 9.1, ma postać:

$$e(t) = \frac{dq(t)}{dt} R + u(t)$$

gdzie $q(t) = Q(u(t))$ oznacza ładunek zgromadzony w kondensatorze.



Rysunek 9.1. Nieliniowy układ RC z przykładu 9.1

Równanie można nadać postać (9.1):

$$\frac{dq(t)}{dt} = \frac{e(t) - u(t)}{R}, \quad q(t_0) = Q(u(t_0)) \quad (9.4)$$

Nawet dla tak prostego układu analityczną postać odpowiedzi można wyznaczyć tylko w niektórych przypadkach – w szczególności, gdy kondensator jest liniowy, tzn. jego pojemność C nie zależy od napięcia u (wówczas $Q(u) = Cu$), a źródło napięcia zmienia się w chwili t_0 skokowo z wartości 0 do wartości E_0 , tzn. $e(t) = E_0 \cdot \mathbf{1}(t - t_0)$. Równanie (9.4) można wówczas przekształcić do liniowego niejednorodnego równania różniczkowego względem zmiennej $u(t)$:

$$u'(t) = -\frac{1}{\tau}u(t) + \frac{1}{\tau}E_0 \quad \text{dla } t \geq t_0, \quad u(t_0) = 0 \quad (9.5)$$

gdzie $\tau = RC$ oznacza tzw. stałą czasową. Rozwiązywanie tego równania ma dobrze znaną elektronikom postać:

$$u(t) = E_0[1 - e^{-(t-t_0)/\tau}] \cdot \mathbf{1}(t - t_0)$$

W kolejnych przykładach, zamieszczonych w tym rozdziale, badane są właściwości metod rozwiązywania zagadnienia początkowego dla układu z rys. 9.1, przy założeniu, że $R = 1 \text{ M}\Omega$, $C = 1 \mu\text{F}$, $t_0 = 0 \text{ s}$, $T = 5 \text{ s}$ oraz:

$$e(t) = \begin{cases} 0 & \text{dla } t \leq 0 \\ 1 & \text{dla } t > 0 \end{cases}$$

W takim układzie, który nazywać będziemy „standardowym”, napięcie na kondensatorze spełnia następujące równanie różniczkowe:

$$u'(t) = -u(t) + 1 \quad \text{dla } t \geq 0 \quad \text{i} \quad u(0) = 0 \quad (9.6)$$

Metody numerycznego rozwiązywania zagadnień początkowych można podzielić na dwie grupy:

- metody oparte na reprezentacji rozwiązania w postaci szeregu funkcyjnego (np. szeregu potęgowego lub szeregu Fouriera);
- metody oparte na dyskretyzacji rozwiązania, zwane metodami różnicowymi.

W dalszym ciągu zajmować się będziemy wyłącznie metodami różnicowymi. Umożliwiają one wyznaczenie przybliżonych wartości rozwiązania $y(t)$ w pewnych tylko punktach t_n należących do przedziału całkowania $[t_0, t_0 + T]$: $y_n \approx y(t_n)$. Najczęściej będziemy zakładać, że $t_n = t_0 + nh$ dla $n = 0, 1, \dots, N$, gdzie tzw. *krok całkowania* $h = t_n - t_{n-1}$ jest stały: $h = T/N$. Pokażemy również, w jaki sposób krok ten można automatycznie dostosowywać do własności rozwiązania w celu poprawienia efektywności obliczeń.

Metody różnicowe, przeznaczone do rozwiązywania zagadnienia początkowego, można podzielić na jednokrokwne (liniowe i nieliniowe) oraz wielokrokwne (liniowe i nieliniowe). W niniejszym rozdziale przedstawimy wybrane metody jednokrokwne oraz wybrane liniowe metody wielokrokwne. *Metody jednokrokwne (typu Rungego-Kutty)* wykorzystują do wyznaczenia przybliżonego rozwiązania y_{n+1} w kolejnym punkcie czasowym t_{n+1} , jedną wartość rozwiązania „z przeszłości”, a mianowicie tę uzyskaną w poprzednim kroku czasowym, tj. y_n :

$$y_{n+1} = y_n + h \Phi_f(t_n, y_n, h) \quad \text{dla } n = 0, 1, \dots, N-1 \quad (9.7)$$

przy czym $y_0 = y(t_0)$, a Φ_f jest funkcją zależną (w sposób liniowy lub nielinowy) od funkcji f . *Liniowe metody wielokrokwne* polegają na wyrażeniu przybliżonego rozwiązania y_{n+1} w kolejnym punkcie czasowym t_{n+1} w postaci liniowej kombinacji wartości rozwiązania i jego pochodnych, uzyskanych w $K > 1$ poprzednich krokach:

$$y_{n+1} = \sum_{k=0}^K \alpha_k y_{n+1-k} + h \sum_{k=0}^K \beta_k f_{n+1-k} \quad \text{dla } n = K-1, \dots, N-1, |\alpha_K| + |\beta_K| \neq 0 \quad (9.8)$$

gdzie $f_n \equiv f(t_n, y_n)$. Jeżeli $\beta_0 \neq 0$, to metodę nazywamy *zamkniętą*, w przeciwnym przypadku – *otwartą*.

Przykład 9.2. Najprostsze metody rozwiązywania zagadnienia początkowego (9.1) można uzyskać, przybliżając pochodną rozwiązania za pomocą różnic skończonych, o których była mowa w podrozdziale 7.3, bądź przedstawiając rozwiązanie w postaci całkowej:

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t)) dt \quad \text{dla } n = 0, 1, \dots \quad (9.9)$$

i zastępując całkę we wzorze (9.9) formułą numerycznego całkowania (kwadraturą).

Zastępując, na przykład, pochodną $y'_n \equiv \left. \frac{dy(t)}{dt} \right|_{t=t_n}$ formułą różnicę progresywnej $\frac{y(t_{n+1}) - y(t_n)}{t_{n+1} - t_n}$ w równaniu $y'(t_n) = f(t_n, y(t_n))$, otrzymuje się *metodę otwartą Eulera*:

$$y_{n+1} = y_n + h f(t_n, y(t_n)), \quad y_0 = y(t_0) \quad (9.10)$$

Ten sam wynik można uzyskać, zastępując $\int_{t_n}^{t_{n+1}} f(t, y(t)) dt$ we wzorze (9.9) kwadraturą prostokątów: $f(t_n, y(t_n)) \cdot h$. Wzór (9.10) można uznać za najprostszą formułę jednokrokwą o postaci (9.7) bądź szczególną postać otwartej formuły wielokrokwej (9.8), odpowiadającą $K = 1$. Jeśli jednak zastąpimy pochodną $y'_{n+1} \equiv \left. \frac{dy(t)}{dt} \right|_{t=t_{n+1}}$ formułą różnicę wsteczną $\frac{y(t_{n+1}) - y(t_n)}{t_{n+1} - t_n}$ w równaniu $y'(t_{n+1}) = f(t_{n+1}, y(t_{n+1}))$, to otrzymamy *metodę zamkniętą Eulera*:

$$y_{n+1} = y_n + h f(t_{n+1}, y(t_{n+1})), \quad y_0 = y(t_0) \quad (9.11)$$

która jest szczególną postacią zamkniętej formuły wielokrokwej (9.8), odpowiadającą $K = 1$.

9.1. PODSTAWOWE WŁASNOŚCI METOD ROZWIĄZYWANIA RÓWNAŃ RÓŻNICZKOWYCH ZWYCZAJNYCH

Od metody różnicowej rozwiązywania zadania (9.1) wymaga się przede wszystkim zbieżności. Metoda jest *zbieżna*, jeśli dla dowolnego zadania (9.1), którego jedynym rozwiązaniem w przedziale $[t_0, t_0 + T]$ jest $y(t)$, rozwiązanie przybliżone y_n , uzyskane po n krokach ($n = 1, \dots, N = T/h$), jest zbieżne do rozwiązania dokładnego $y(t_n)$, gdy krok $h = t_n - t_{n-1}$ ($n = 1, 2, \dots$) maleje do zera. Innymi słowy: błędy przybliżenia, $e_n = |y(t_n) - y_n|$, stałokrokwnej metody zbieżnej maleją *jednostajnie* do zera, gdy krok h maleje do zera:

$$e_n \xrightarrow{h \rightarrow 0} 0 \quad \text{dla } n = 1, \dots, N = T/h \quad (9.12)$$

W praktyce ważna jest nie tylko sama zbieżność metody, ale i szybkość tej zbieżności. Miarą szybkości zbieżności jest *rzqd metody*, rozumiany jako największa

liczba całkowita p taka, że dla dostatecznie małych długości kroku h zachodzi nierówność:

$$\sup \{ \| \mathbf{e}_n \| \mid 0 \leq n \leq N \} \leq c h^p \quad (9.13)$$

gdzie c jest stałą niezależną od h , a lewa strona nierówności jest *globalnym błędem metody*.

Oznaczmy przez \mathcal{D} operator rozwiązywania układu równań różniczkowych zwyczajnych za pomocą wielokrokowej metody różnicowej. Wówczas:

$$\mathbf{y}_{n+1} = \mathcal{D}(\mathbf{y}_n, \mathbf{y}_{n-1}, \dots) \quad (9.14)$$

Lokalny błąd obcięcia (błąd dyskretyzacji) metody różnicowej, działającej według tego schematu, jest definiowany jako różnica między rozwiązaniem dokładnym $\mathbf{y}(t_{n+1})$ a przybliżonym \mathbf{y}_{n+1} , przy założeniu, że argumenty operatora \mathcal{D} w (9.14) są dokładne, tzn. $\mathbf{y}_n = \mathbf{y}(t_n)$, $\mathbf{y}_{n-1} = \mathbf{y}(t_{n-1})$, ...:

$$\mathbf{r}_{n+1} = \mathbf{y}(t_{n+1}) - \mathcal{D}(\mathbf{y}(t_n), \mathbf{y}(t_{n-1}), \dots) \quad (9.15)$$

Zależność lokalnego błędu obcięcia od kroku całkowania h :

$$\mathbf{r}_{n+1} \equiv \mathbf{r}_{n+1}(h) = \mathbf{r}_n(0) + \mathbf{r}'_n(0)h + \mathbf{r}''_n(0)h^2/2 + \dots \quad (9.16)$$

jest wykorzystywana do określenia *rzędu metody*. Metoda jest rzędu p , jeśli:

$$\mathbf{r}_n(0) = 0, \quad \mathbf{r}'_n(0) = 0, \quad \mathbf{r}''_n(0) = 0, \dots, \quad \mathbf{r}_n^{(p)}(0) = 0, \quad \text{ale} \quad \mathbf{r}_n^{(p+1)}(0) \neq 0 \quad (9.17)$$

Dla metody rzędu p lokalny błąd obcięcia ma postać:

$$\mathbf{r}_{n+1}(h) = \mathbf{C}_p h^{p+1} + O(h^{p+2}) \quad (9.18)$$

gdzie $\mathbf{C}_p = \mathbf{r}_{n+1}^{(p)}(0)/(p+1)!$. Metodę, dla której $\mathbf{r}_n(0) = 0$ oraz $\mathbf{r}'_n(0) = 0$, nazywamy *zgodną*. Zgodność jest warunkiem koniecznym, ale niewystarczającym zbieżności wielokrokowej formuły całkowania [F2]. Warunkiem koniecznym i wystarczającym zbieżności metody zgodnej jest bowiem D-stabilność – właściwość metod różnicowych omówiona w podrozdziale 9.3.2.

9.2. METODY JEDNOKROKOWE TYPU RUNGEGO-KUTTY

9.2.1. KONSTRUKCJA I WŁASNOŚCI METOD JEDNOKROKOWYCH

Najprostszą metodą jednokrokową jest, wprowadzona już wcześniej, metoda otwarta Eulera (9.10). Zbadajmy lokalną i globalną dokładność tej metody, zakładając, że rozwiązanie dokładne $y(t)$ skalarnego równania różniczkowego

$y'(t) = f(t, y)$ ma ciągłą drugą pochodną w przedziale $[t_0, t_0 + T]$. Rozkładając to rozwiązanie w szereg Taylora wokół punktu t_n , otrzymujemy:

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + r_{n+1} = y(t_n) + hf(t_n, y(t_n)) + r_{n+1} = y_{n+1} + r_{n+1} \quad (9.19)$$

gdzie $r_{n+1} = y''(\tau_n)h^2/2$ jest resztą szeregu Taylora, a $\tau_n \in [t_n, t_{n+1}]$. Reszta r_{n+1} jest różnicą między rozwiązaniem dokładnym a przybliżonym w chwili t_{n+1} (przy założeniu, że znane jest dokładne rozwiązanie $y(t_n)$ w punkcie poprzednim t_n), czyli poszukiwanym *lokalnym błędem obcięcia* w punkcie t_{n+1} . Zgodnie z (9.17) metoda otwarta Eulera ma więc rzad $p=1$.

Oszacujmy teraz *globalny błąd obcięcia*, tzn. błąd charakteryzujący niedokładność przyblizonego rozwiązania $y(t)$ w przedziale $[t_0, t_0 + T]$ uzyskanego za pomocą formuły po N krokach całkowania. Błąd ten nie jest oczywiście równy lokalnemu błędowi w ostatnim kroku $r_N = y''(\tau_N)h^2/2$ ani sumie błędów lokalnych $r_1 + r_2 + \dots + r_N$, ponieważ w każdym kroku (poza pierwszym) do błędu lokalnego metody dodaje się pewna składowa błędu spowodowana niedokładnością obliczenia rozwiązania w poprzednim kroku:

$$y_{n+1} = y_n + hf(t_n, y_n) = y(t_n) + e_n + h[f(t_n, y(t_n)) + e_n] \quad (9.20)$$

gdzie $e_n = y(t_n) - y_n$ oznacza błąd metody po n krokach (przy czym $e_0 = 0$). Stąd:

$$\mathbf{e}_{n+1} \equiv y(t_{n+1}) - y_{n+1} = \underbrace{y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\tilde{t}_n)}_{y(t_{n+1})} - \underbrace{[y(t_n) + e_n + hf(t_n, y(t_n)) + e_n]}_{y_{n+1}} \quad (9.21a)$$

$$\mathbf{e}_{n+1} \equiv \frac{h^2}{2}y''(\tilde{t}_n) - e_n + h \left[\underbrace{y'(t_n) - f(t_n, y(t_n))}_{=0} - f'_y(t_n, \tilde{y}(t_n)) \cdot e_n \right] \quad (9.21b)$$

gdzie $\tilde{t}_n \in [t_n, t_n + h]$, $\tilde{y}(t_n) \in [y(t_n), y(t_n + e_n)]$, zaś $f'_y(t, y)$ jest pochodną funkcji $f(t, y)$ względem y . Oszacowanie globalnego błędu obcięcia wykonuje się zazwyczaj przy założeniu, że funkcja $f(t, y)$ spełnia warunek Lipschitza (9.3) ze stałą:

$$L = \sup \left\{ \frac{|f(t, y) - f(t, \bar{y})|}{|y - \bar{y}|} \mid t \in [t_0, t_0 + T], y, \bar{y} \in [y_{\min}, y_{\max}] \right\}$$

a moduł drugiej pochodnej rozwiązania $y(t)$ jest ograniczony, tzn.:

$$|y''(t)| \leq M_2 \equiv \sup \{ |y''(t)| \mid t \in [t_0, t_0 + T] \} < \infty$$

Zbieżność metod jednokrokowych (9.7) zależy od własności funkcji Φ_f . Okazuje się [J1], że głównym warunkiem koniecznym zbieżności jest warunek aproksymacji:

$$\Phi_f(t, y, 0) = \mathbf{f}(t, y) \quad (9.27)$$

(pozostałe warunki dotyczą ciągłości funkcji Φ_f i \mathbf{f}). Z kolei szybkość zbieżności zależy od zachowania funkcji $\Phi_f(t, y(t), h)$ dla $h > 0$. Można udowodnić [J1], że metoda jednokrokowa jest rzędu p , gdy lokalny błąd obcięcia pochodnej:

$$\tau(t, y, h) \equiv \frac{y(t+h) - y(t)}{h} - \Phi_f(t, y, h) \quad (9.28)$$

jest rzędu h^p , tzn. $\tau(t, y, h) = O(h^p)$. Jednokrokowe metody rzędu $p > 1$ wymagają więc bardziej złożonej funkcji Φ_f , lepiej przybliżającej iloraz $\Delta(t, h) \equiv \frac{y(t+h) - y(t)}{h}$ niż metoda Eulera. Jako przykład rozważmy formułę opartą na funkcji Φ_f postaci:

$$\Phi_f(t, y, h) = \alpha_1 f(t, y) + \alpha_2 f(t + \gamma_1 h, y + \gamma_2 h f(t, y)) \quad (9.29)$$

gdzie α_1 , α_2 , γ_1 i γ_2 są stałymi współczynnikami. Rozwinięcie ilorazu $\Delta(t, h)$ w szereg Taylora względem h ma postać:

$$\begin{aligned} \Delta(t, h) &\equiv \frac{y(t+h) - y(t)}{h} = y'(t) + \frac{h}{2} y''(t) + \frac{h^2}{6} y'''(t) + O(h^3) = \\ &= f + \frac{h}{2} (f'_t + f'_y f) + \frac{h^2}{6} (f''_t + 2f''_{ty} f + f''_{yy} f^2 + f'_y f'_t + (f'_y)^2 f) + O(h^3) \end{aligned}$$

gdzie:

$$\begin{aligned} f &\equiv f(t, y(t)), \quad f'_t \equiv \frac{\partial f(t, y)}{\partial t}, \quad f'_y \equiv \frac{\partial f(t, y)}{\partial y}, \quad f''_t \equiv \frac{\partial^2 f(t, y)}{\partial t^2}, \quad f''_y \equiv \frac{\partial^2 f(t, y)}{\partial t \partial y}, \\ f''_{yy} &\equiv \frac{\partial^2 f(t, y)}{\partial y^2} \end{aligned}$$

natomaiast rozwinięcie funkcji Φ_f :

$$\begin{aligned} \Phi_f(t, y, h) &= \Phi_f(t, y, 0) + h \left. \frac{\partial \Phi_f(t, y, h)}{\partial h} \right|_{h=0} + \frac{h^2}{2} \left. \frac{\partial^2 \Phi_f(t, y, h)}{\partial h^2} \right|_{h=0} + O(h^3) = \\ &= (\alpha_1 + \alpha_2) f + \alpha_2 h \frac{df}{dh} + \frac{\alpha_2 h^2}{2} \frac{d^2 f}{dh^2} + O(h^3) \end{aligned}$$

gdzie:

$$\begin{aligned} \frac{df}{dh} &= \left. \frac{df(t + \gamma_1 h, y + \gamma_2 h f(t, y))}{dh} \right|_{h=0} = \gamma_1 f'_t + \gamma_2 f'_y f(t, y) \\ \frac{d^2 f}{dh^2} &= \left. \frac{d^2 f(t + \gamma_1 h, y + \gamma_2 h f(t, y))}{dh^2} \right|_{h=0} = \\ &= \gamma_1 f''_t + [(\gamma_1 + \gamma_2) f''_y + \gamma_2 f''_{yy} f(t, y)] f(t, y) \end{aligned}$$

Wynika stąd rozwinięcie w szereg Taylora formuły lokalnego błędu obcięcia pochodnej:

$$\begin{aligned} \tau(t, y, h) &\equiv f + \frac{h}{2} (f'_t + f'_y f) + \frac{h^2}{6} (f''_t + 2f''_{ty} f + f''_{yy} f^2 + f'_y f'_t + (f'_y)^2 f) + \\ &\quad - (\alpha_1 + \alpha_2) f - \alpha_2 h (\gamma_1 f'_t + \gamma_2 f'_y f) + \\ &\quad - \frac{\alpha_2 h^2}{2} (\gamma_1 f''_t + ((\gamma_1 + \gamma_2) f''_y + \gamma_2 f''_{yy} f) f) + O(h^3) \end{aligned}$$

Zauważmy, że możliwe jest wyzerowanie składników rozwinięcia rzędu $O(1)$ i $O(h)$, o ile spełnione są warunki: $\alpha_1 + \alpha_2 = 1$, $\alpha_2 \gamma_1 = 1/2$, $\alpha_2 \gamma_2 = 1/2$. Nie jest możliwe jednoczesne wyzerowanie składnika $O(h^2)$; dlatego podane warunki określają ograniczenia na współczynniki α_1 , α_2 , γ_1 i γ_2 metod RK postaci (9.29) rzędu drugiego. Jednym z rozwiązań tych równań jest zestaw współczynników: $\alpha_1 = 1/2$, $\alpha_2 = 1/2$, $\gamma_1 = 1$ i $\gamma_2 = 1$, definiujący metodę Heuna:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2} (\mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}(t_n + h, \mathbf{y}_n + h \mathbf{f}(t_n, \mathbf{y}_n))) \quad (9.30)$$

Innym rozwiązaniem jest zestaw: $\alpha_1 = 0$, $\alpha_2 = 1$, $\gamma_1 = 1/2$, $\gamma_2 = 1/2$, definiujący zmodyfikowaną metodę Eulera:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \mathbf{f} \left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2} \mathbf{f}(t_n, \mathbf{y}_n) \right) \quad (9.31)$$

Formuły (9.30) i (9.31) istotnie mają wyższy rzad od metody otwartej Eulera; wzrost tego rzędu uzyskaliśmy jednak kosztem dwukrotnie większej liczby wartości funkcji \mathbf{f} , które muszą być wyznaczone w każdym kroku.

Powyższe postępowanie można uogólnić, tworząc formuły jeszcze wyższego rzędu. Klasa metod Rungego-Kutty (w skrócie RK) jest zdefiniowana następująco:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{r=1}^R w_r \mathbf{k}_r \quad (9.32a)$$

gdzie:

$$\mathbf{k}_r = \mathbf{k}_r(t_n, \mathbf{y}_n, h) = \mathbf{f}\left(t_n + h a_r, \mathbf{y}_n + \sum_{p=1}^R b_{r,p} \mathbf{k}_p\right) \quad \text{dla } r = 1, \dots, R \quad (9.32b)$$

natomiast a_r , $b_{r,p}$ i w_r są stałymi, których wartości można znaleźć np. w [F2]. Jeżeli $b_{r,p} = 0$ dla $p \geq r$ oraz $r = 1, \dots, R$, to metoda jest *otwarta* (jawna). W przeciwnym przypadku metoda jest *zamknięta* (niejawna), tzn. nieznany jeszcze wektor \mathbf{y}_{n+1} występuje po lewej i prawej stronie wyrażenia (9.32a); wyznaczenie jego wartości wymaga rozwiązyania układu nieliniowych równań algebraicznych.

Maksymalny rząd otwartej metody RK, wykorzystującej R wartości funkcji wynosi $p(R) \leq R$, przy czym $p(R) = R$ dla $R = 1, 2, 3$ i 4 . Maksymalny rząd zamkniętej metody RK może być równy $2R$, jednak ze względu na konieczność rozwiązywania nieliniowych równań algebraicznych nakład obliczeniowy niezbędny do wykonania jednego kroku jest dla takiej metody znacznie większy niż dla odpowiedniej metody otwartej.

Zauważmy, że metodę Heuna (9.30) i zmodyfikowaną metodę Eulera (9.31) można uważać za metody Rungego-Kutty drugiego rzędu – maksymalnego w swojej klasie ($R = 2$), a metodę otwartą Eulera (9.10) – za metodę Rungego-Kutty pierwszego rzędu. Spośród metod wyższego rzędu najczęściej stosowana jest metoda czwartego rzędu (oznaczana akronimem RK4), ze współczynnikami:

$$\left. \begin{aligned} w_1 &= w_4 = \frac{1}{6}, \quad w_2 = w_3 = \frac{1}{3} \\ \mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n) \\ \mathbf{k}_2 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2} \mathbf{k}_1\right) \\ \mathbf{k}_3 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2} \mathbf{k}_2\right) \\ \mathbf{k}_4 &= \mathbf{f}(t_n + h, \mathbf{y}_n + h \mathbf{k}_3) \end{aligned} \right\} \quad (9.33)$$

Przykład 9.4. Porównajmy dokładność rozwiązania równania różniczkowego (9.6), opisującego „standardowy” liniowy układ RC z przykładu 9.1, za pomocą trzech metod całkowania: otwartej metody Eulera (9.10), metody Heuna (9.30) i metody RK4 (9.32) i (9.33) z taką samą liczbą kroków czasowych $N = 20$. Otwarta metoda Eulera prowadzi w tym przypadku do rozwiązania różnicowego:

$$u_0 = 0, \quad u_{n+1} = u_n + h(1 - u_n) \quad \text{dla } n = 0, 1, \dots, N-1 \quad (9.34a)$$

metoda Heuna – do równania:

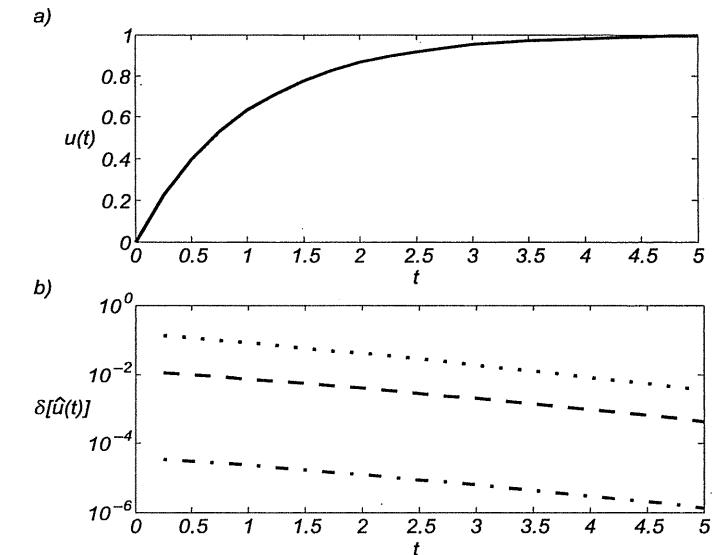
$$u_0 = 0, \quad u_{n+1} = u_n + h\left(1 - \frac{h}{2}\right)(1 - u_n) \quad \text{dla } n = 0, 1, \dots, N-1 \quad (9.34b)$$

metoda RK4 – do równania:

$$u_0 = 0, \quad u_{n+1} = u_n + h\left(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4\right) \quad \text{dla } n = 0, 1, \dots, N-1 \quad (9.34c)$$

przy czym:

$$k_1 = 1 - u_n, \quad k_2 = 1 - \left(u_n + \frac{h}{2}k_1\right), \quad k_3 = 1 - \left(u_n + \frac{h}{2}k_2\right) \text{ oraz } k_4 = 1 - (u_n + h k_3)$$



Rysunek 9.3. Wyniki obliczeń przeprowadzonych w przykładzie 9.4: a) rozwiązanie dokładne; b) względne błędy rozwiązań uzyskanych za pomocą otwartej metody Eulera (linia kropkowana), metody Heuna (linia przerywana) i metody Rungego-Kutty rzędu 4 (linia typu „kreska-kropka”)

Na rys. 9.3a pokazano rozwiązanie dokładne: $u(t) = 1 - e^{-t}$ dla $t \geq 0$, zaś na rys. 9.3b – przebiegi błędu względnego $\delta[\hat{u}(t)] \equiv [\hat{u}(t) - u(t)]/u(t)$ rozwiązań przybliżonych $\hat{u}(t)$. Widać wyraźnie, że metody wyższego rzędu umożliwiły osiągnięcie istotnie większej dokładności rozwiązania. Trzeba jednak podkreślić, że uogólnienie tego wniosku jest możliwe jedynie dla dostatecznie małego kroku całkowania h , dla większych bowiem wartości tego kroku (wynikających z mniejszej wymaganej dokładności) metody niższych rzędów mogą okazać się dokładniejsze.

9.2.2. WYBÓR KROKU CAŁKOWANIA

Krok całkowania wpływa na dokładność rozwiązywania dwojako. Do tej pory skupialiśmy się na błędzie globalnym metody całkowania, malejącym asymptotycznie do zera tak jak ch^p , gdzie p jest rzędem metody rozwiązywania równań różniczkowych. Przy realizacji zmienopozycyjnej tej metody występuje ponadto błąd zaokrągleń, który dodatkowo pogarsza łączną dokładność rozwiązywania numerycznego. Analiza tego problemu pokazuje [J1, S1], że zmniejszanie długości kroku zmniejsza wprawdzie błąd obcięcia metody, ale zwiększa składową błędu spowodowaną zaokrąglaniami wyników operacji zmienopozycyjnych. Istnieje więc teoretyczna możliwość minimalizacji łącznego błędu metody przez odpowiedni dobór długości tego kroku w sposób równoważący wpływ błędów zaokrągleń i błędów obcięcia. Precyzyjne określenie osiągalnej dokładności całkowania jest jednak trudne ze względu na zgrubne oszacowanie globalnego błędu obcięcia (por. przykład 9.3).

Ponieważ w praktyce inżynierskiej wymagana dokładność rozwiązywania równań różniczkowych jest zazwyczaj znacznie niższa niż możliwa do osiągnięcia w danej arytmetyce zmienopozycyjnej, nie będziemy dalej rozważać problemu oszacowania osiągalnej dokładności i wynikającej stąd długości kroku. Zajmiemy się natomiast bardzo ważnym praktycznie zagadnieniem automatycznego wyboru kroku całkowania, który umożliwi osiągnięcie założonej dokładności rozwiązania przy jak najmniejszych nakładach obliczeniowych. Dla uproszczenia zapisu rozważymy zagadnienie wyboru długości kroku metody całkowania rzędu p dla pojedynczego równania różniczkowego przy założeniu, że o niedokładności decyduje błąd obcięcia metody. Będziemy wymagać, aby błąd lokalny obcięcia dla podprzedziału całkowania o długości h wynosił nie więcej niż $r_{\max} = e_{\max} h/T$, gdzie e_{\max} jest zadanym parametrem dokładności globalnej.

Jednym z praktycznie użytecznych sposobów oszacowania lokalnego błędu obcięcia jest tzw. zasada Rungego, zgodnie z którą następny punkt rozwiązywania wyznacza się dwukrotnie: najpierw przy użyciu formuły z krokiem h (na rys. 9.4 wynik oznaczony przez $y_{n+1,1}$), a następnie – przez dwukrotne użycie tej samej formuły z krokiem $h/2$ (na rys. 9.4 wynik oznaczony przez $y_{n+1,2}$). Po uwzględnieniu ogólnej postaci lokalnego błędu obcięcia (9.18), otrzymujemy:

$$\begin{aligned} y_{n+1,1} &= y(t+h) + C_{p+1} h^{p+1} + O(h^{p+2}) \\ y_{n+1,2} &= y(t+h) + 2C_{p+1} \left(\frac{h}{2}\right)^{p+1} + O(h^{p+2}) \end{aligned} \quad (9.35)$$

gdzie C_{p+1} jest współczynnikiem zależnym od $(p+1)$ -szej pochodnej funkcji f (np. $C_2 = y''(\tau_n)/2$ dla metody otwartej Eulera – por. (9.19)). Różnica

$\Delta_{n+1} \equiv y_{n+1,1} - y_{n+1,2}$ może służyć do oszacowania lokalnego błędu obcięcia. Z dokładnością $O(h^{p+2})$ mamy bowiem:

$$\Delta_{n+1} = C_{p+1} h^{p+1} - 2C_{p+1} \left(\frac{h}{2}\right)^{p+1} = C_{p+1} h^{p+1} (1 - 2^{-p})$$

a więc:

$$C_{p+1} \approx \frac{\Delta_{n+1}}{h^{p+1} (1 - 2^{-p})}$$

Wynika stąd oszacowanie błędu popełnionego przy obliczaniu $y_{n+1,2}$:

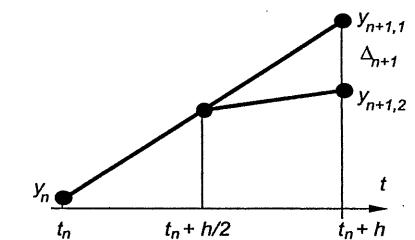
$$r_{n+1} \approx 2C_{p+1} \left(\frac{h}{2}\right)^{p+1} = \frac{\Delta_{n+1}}{2^p - 1}. \quad (9.36)$$

Załóżmy teraz, że błąd lokalny dla podprzedziału całkowania o długości h ma wynieść nie więcej niż

$$r_{\max} = \frac{h}{T} e_{\max} \quad (9.37)$$

Na podstawie oszacowania (9.36) można wyznaczyć długość kroku całkowania \hat{h} , który spełnia wymagania dokładności; rozwiązując równanie $r \approx 2C_{p+1} \hat{h}^{p+1} / 2^{p+1}$ względem \hat{h}^{p+1} :

$$\hat{h}^{p+1} \approx \frac{2^p r_{\max}}{C_{p+1}} \approx \frac{(2^p - 1)r_{\max}}{\Delta_{n+1}} \hat{h}^{p+1} = \frac{r_{\max}}{r_{n+1}} h^{p+1} \quad (9.38)$$



Rysunek 9.4. Znaczenie zmiennych wykorzystywanych do szacowania lokalnego błędu obcięcia przy użyciu zasady Rungego

Powyższe przybliżenie lokalnego błędu może być wykorzystane do doboru długości kroku całkowania. Jeden z możliwych do realizacji algorytmów sterowania krokiem na bieżąco ma postać:

- ❶ Wyznacz błąd r_{n+1} dla długości kroku h_{n+1} według wzoru (9.36) oraz r_{\max} wg wzoru (9.37).
- ❷ Jeśli $|r_{n+1}| \leq r_{\max}$, to wynik tego kroku $y_{n+1,2}$ uznaj za wystarczająco dokładny. Kontynuuj obliczenia z krokiem:

$$h_{n+2} = \begin{cases} h_{n+1}, & \text{gdy } \rho r_{\max} < |r_{n+1}| \leq r_{\max} \\ \frac{h_{n+1}}{\max \left\{ \frac{1}{10}, \sqrt[p+1]{\frac{|r_{n+1}|}{r_{\max}}} \right\}}, & \text{gdy } |r_{n+1}| \leq \rho r_{\max} \end{cases} \quad (9.39)$$

gdzie $\rho \in (0, 1]$.

- ❸ Jeśli $|r_{n+1}| > r_{\max}$, to powtóż punkt ❶ algorytmu ze skróconym krokiem, np.:

$$h_{n+1} := \frac{h_{n+1}}{\min \left\{ 10, \sqrt[p+1]{\frac{|r_{n+1}|}{r_{\max}}} \right\}} \quad (9.40)$$

Gdy rozwiązywany jest układ równań różniczkowych, symbol r_{n+1} w tym algorytmie należy rozumieć jako największy (co do modułu) z lokalnych błędów obcięcia składowych wektora rozwiązań.

Słosowanie automatycznego doboru kroku prowadzi na ogół do mniejszych nakładów obliczeniowych niż metody stałokrokowe. Ilustracją niech będzie poniższy przykład obliczeniowy.

Przykład 9.5. Porównajmy dokładność rozwiązywania równania różniczkowego (9.6), opisującego „standardowy” liniowy układ RC z przykładu 9.1, za pomocą dwóch wariantów otwartej metody Eulera. W wariantie zmienkokrokowym przyjmijmy krok początkowy $h = 2 \cdot 10^{-3}$ oraz parametry algorytmu adaptacji kroku: $e_{\max} = 10^{-3}$ oraz $\rho = 0.5$. Krok całkowania dla drugiego wariantu metody jest stały i eksperymentalnie dobrany ($h \approx 6.4 \cdot 10^{-4}$) w taki sposób, aby globalny błąd był w przybliżeniu równy uzyskanemu dla metody zmienkokrokowej. Na rys. 9.5 widać, że obydwie metody dają rozwiązania z podobnym błędem globalnym ($< 1.2 \cdot 10^{-4}$), znacznie mniejszym niż wartość parametru $e_{\max} = 10^{-3}$. Dzięki sterowaniu krokiem liczba obliczeń wartości funkcji $f(t, u)$ zostaje zmniejszona z 7812 do 5832. Takie zachowanie algorytmu ma szczególnie istotne znaczenie dla czasu rozwiązywania zadań, w których złożoność obliczeniowa funkcji $f(t, u)$ jest znacznie większa niż złożoność wyznaczenia kolejnego punktu rozwiązania.



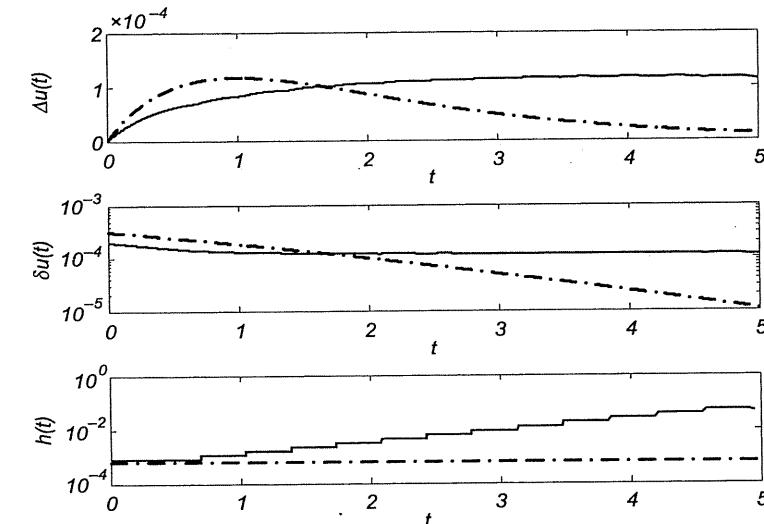
Do szacowania lokalnego błędu obcięcia w metodach Rungego-Kutty stosowana jest też technika zaproponowana przez Fehlberga. Przedstawimy ją na przykładzie pary metod składającej się z metody Rungego-Kutty trzeciego rzędu (oznaczanej akronimem RK3):

$$\left. \begin{aligned} \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{6} (\mathbf{k}_1 + \mathbf{k}_2 + 4\mathbf{k}_3) \\ \mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n) \\ \mathbf{k}_2 &= \mathbf{f}(t_n + h, \mathbf{y}_n + \mathbf{k}_1) \\ \mathbf{k}_3 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{\mathbf{k}_1 + \mathbf{k}_2}{4}\right) \end{aligned} \right\} \quad (9.41)$$

oraz metody Heuna, zdefiniowanej wzorem (9.30), który wykorzystuje takie same współczynniki \mathbf{k}_1 i \mathbf{k}_2 :

$$\mathbf{y}_{n+1}^* = \mathbf{y}_n + \frac{h}{2} (\mathbf{k}_1 + \mathbf{k}_2) \quad (9.42)$$

Ponieważ metody te charakteryzują się różnym błędem obcięcia, różnicę $\Delta \mathbf{y}_{n+1} \equiv \mathbf{y}_{n+1} - \mathbf{y}_{n+1}^*$ można wykorzystać do szacowania lokalnego błędu obcięcia \mathbf{r}_{n+1} . Dla tej pary algorytmów $\mathbf{r}_{n+1} = \Delta \mathbf{y}_{n+1}$.



Rysunek 9.5. Porównanie błędu bezwzględnego $\Delta u(t)$, błędu względnego $\delta u(t)$ i długości kroku $h(t)$ dla metody Eulera ze stałym krokiem (linia ciągła) i zmiennym krokiem (linia przerwana)

W procedurze **ode23** (w programie MATLAB) wykorzystywana jest inna para metod RK, a mianowicie para Bogackiego-Shampine, składająca się z metody trzeciego rzędu:

$$\left. \begin{array}{l} \mathbf{y}_{n+1} = \mathbf{y}_n + h \left(\frac{2}{9} \mathbf{k}_1 + \frac{1}{3} \mathbf{k}_2 + \frac{4}{9} \mathbf{k}_3 \right) \\ \mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{y}_n) \\ \mathbf{k}_2 = \mathbf{f}(t_n + h, \mathbf{y}_n + \mathbf{k}_1) \\ \mathbf{k}_3 = \mathbf{f}\left(t_n + \frac{3}{4}h, \mathbf{y}_n + \frac{3}{4}\mathbf{k}_2\right) \end{array} \right\} \quad (9.43)$$

oraz zmodyfikowanej metody Eulera (drugiego rzędu) zdefiniowanej wzorem (9.31), któremu można nadać postać:

$$\mathbf{y}_{n+1}^* = \mathbf{y}_n + h \mathbf{k}_2 \quad (9.44)$$

Można pokazać, że oszacowanie lokalnego błędu obcięcia ma w tym przypadku postać:

$$r_{n+1} = \frac{h}{72} (-5\mathbf{k}_1 + 6\mathbf{k}_2 + 8\mathbf{k}_3 - 9\mathbf{k}_4) \quad (9.45)$$

gdzie $\mathbf{k}_4 = \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$.

Do bardziej znanych jednokrokowych metod całkowania należy metoda mieszana, zwana metodą Rungego-Kutty-Fehlberga rzędu 5 i oznaczana akronimem RKF45. Istnieje wiele wariantów tej metody, które wykorzystują metodę Rungego-Kutty piątego rzędu:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \sum_{i=1}^6 c_i \mathbf{k}_i \quad (9.46a)$$

oraz metodę Rungego-Kutty czwartego rzędu:

$$\mathbf{y}_{n+1}^* = \mathbf{y}_n + \sum_{i=1}^6 c_i^* \mathbf{k}_i \quad (9.46b)$$

gdzie:

$$\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{y}_n), \quad \mathbf{k}_i = \mathbf{f}\left(t_n + a_i h, \mathbf{y}_n + h \sum_{j=1}^{i-1} b_{i,j} \mathbf{k}_j\right) \quad \text{dla } i = 2, \dots, 6$$

Oszacowanie lokalnego błędu obcięcia ma postać:

$$\mathbf{r}_{n+1} = \mathbf{y}_{n+1} - \mathbf{y}_{n+1}^* = \sum_{i=1}^6 (c_i - c_i^*) \mathbf{k}_i \quad (9.46c)$$

Procedura **ode45** programu MATLAB nie wykorzystuje oryginalnych współczynników a_i , $b_{i,j}$, c_i i c_i^* podanych przez Fehlberga. Kolejne punkty rozwiązań \mathbf{y}_{n+1} są wyznaczane wg formuły (9.46a) ze współczynnikami zaproponowanymi

przez Dormanda i Prince'a [P], podanymi w tablicy 9.1. Oszacowanie lokalnego błędu obcięcia ma natomiast postać:

$$\mathbf{r}_{n+1} = \sum_{i=1}^7 w_i \mathbf{k}_i \quad (9.47)$$

gdzie $\mathbf{k}_7 = \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$, $w_7 = -1/40$. Charakterystyczne dla tej metody jest to, że ostatni (siódmy) punkt, w którym procedura wyznacza wartość funkcji w danym kroku, jest jednocześnie pierwszym punktem w następnym kroku.

Tablica 9.1
Współczynniki metody RKF45 według Dormanda i Prince'a

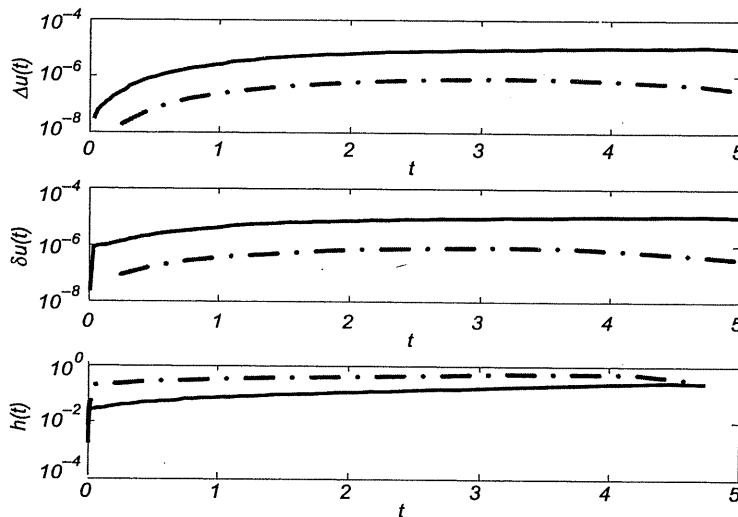
i	a_i	$b_{i,1}$	$b_{i,2}$	$b_{i,3}$	$b_{i,4}$	$b_{i,5}$	c_i	w_i
1							$\frac{35}{384}$	$\frac{71}{57600}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{500}{1113}$	$\frac{71}{16695}$
4	$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$			$\frac{125}{192}$	$\frac{71}{1920}$
5	$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$		$-\frac{2187}{6784}$	$-\frac{17253}{339200}$
6	1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$-\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$	$\frac{11}{84}$	$\frac{22}{525}$

Przykład 9.6. Równanie różniczkowe (9.6), opisujące „standardowy” liniowy układ RC z przykładu 9.1, było rozwiązywane w przykładzie 9.5 za pomocą zmiennokrokowej otwartej metody Eulera. Dla porównania to samo zadanie zostało rozwiązane za pomocą procedur **ode23** i **ode45** programu MATLAB. W procedurach tych wartość parametru e_{\max} , sterującego krokiem całkowania, nie jest wyznaczana według równania (9.37), lecz określana oddziennie dla każdej składowej rozwiązania w bieżącym kroku:

$$e_{\max,m} = \max(\text{RelTol} \cdot |\mathbf{y}_m(t_n)|, \text{AbsTol}_m) \quad (9.48)$$

gdzie RelTol jest parametrem dokładności względnej, a AbsTol_m – parametrem dokładności bezwzględnej (dla naszego zadania jest tylko jedna składowa rozwiązania). Początkowy krok całkowania był wybierany przez procedury automatycznie.

Przebieg rozwiązań dla domyślnych ustawień parametrów ($\text{RelTol} = 0.001$, $\text{AbsTol} = 10^{-6}$) przedstawiono narys. 9.6. Wartości funkcji $f(t, u) \equiv 1 - u$ były obliczane 55 razy w procedurze **ode23** i 91 razy w procedurze **ode45**, przy czym pierwsza z nich dała globalny błąd wyznaczenia $u(T)$ rzędu $7.3 \cdot 10^{-4}$ (a błąd wzajemny ok. $7.9 \cdot 10^{-4}$), a druga – rzędu $2.1 \cdot 10^{-5}$ (błąd wzajemny ok. $6.2 \cdot 10^{-5}$). Obydwie procedury wykonały zdecydowanie mniej obliczeń wartości funkcji $f(t, u)$ niż metoda otwarta Eulera, użytą w przykładzie 9.5, co można tłumaczyć wyższym rzędem procedur **ode23** i **ode45**. Dla wszystkich trzech procedur uzyskany błąd rozwiązania jest znacznie mniejszy niż wartości ich parametrów definiujących wymaganą dokładność (w przypadku **ode23** i **ode45** chodzi o RelTol).



Rysunek 9.6. Porównanie błędu bezwzględnego $\Delta u(t)$, błędu wzajemnego $\delta u(t)$ i długości kroku $h(t)$ dla procedur **ode23** (linia ciągła) i **ode45** (linia przerywana) zastosowanych w przykładzie 9.6 ($\text{RelTol} = 0.001$, $\text{AbsTol} = 10^{-6}$)

Obliczenia powtórzone dla wartości parametru $\text{RelTol} = 10^{-5}$. Tym razem mniej (91) obliczeń wartości funkcji $f(t, u)$ wymagała procedura **ode45**, uzyskując maksymalny błąd bezwzględny ok. $1.0 \cdot 10^{-6}$ i wzajemny ok. $1.5 \cdot 10^{-6}$ niż procedura **ode23** (151), która osiągnęła maksymalny błąd bezwzględny ok. $1.1 \cdot 10^{-5}$ i wzajemny ok. $1.1 \cdot 10^{-5}$. Jest to zachowanie zgodne z oczekiwaniem: metoda wysokiego rzędu wyznacza rozwiązanie dokładniej od metody niższego rzędu wtedy, gdy krok jest odpowiednio mały, tzn. gdy wymagania dotyczące dokładności są odpowiednio wysokie.

♣

9.3. METODY WIELOKROKOWE

9.3.1. KONSTRUKCJA I WŁASNOŚCI METOD WIELOKROKOWYCH

Liniowe metody wielokrokowe (9.8) wykorzystują informację z K poprzednich kroków: przybliżone wartości rozwiązania y_n i pochodnych rozwiązania $f_n \equiv f(t_n, y_n)$:

$$y_{n+1} = \sum_{k=1}^K \alpha_k y_{n+1-k} + h \sum_{k=0}^K \beta_k f_{n+1-k} \quad \text{dla } n = K-1, \dots, N-1 \quad (9.49)$$

przy czym $|\alpha_K| + |\beta_K| \neq 0$. Dzięki temu możliwe jest osiągnięcie wyższego rzędu metody bez pomocniczych obliczeń funkcji $f(t, y)$ w punktach pośrednich, jak to ma miejsce w metodach jednokrokowych (Rungego-Kutty) rzędu wyższego niż pierwszy.

Jeżeli $\beta_0 = 0$, to metoda nazywana jest *otwartą*, ponieważ wektor y_{n+1} zależy wyłącznie od znanych już wektorów y_n, y_{n-1}, \dots i y_{n-K+1} oraz f_n, f_{n-1}, \dots i f_{n-K+1} . Jeżeli $\beta_0 \neq 0$, to metoda nazywana jest *zamkniętą*, ponieważ y_{n+1} zależy również od $f_{n+1} \equiv f(t_{n+1}, y_{n+1})$, co oznacza, że w każdym kroku trzeba rozwiązać układ równań algebraicznych:

$$y_{n+1} - h\beta_0 f(t_{n+1}, y_{n+1}) = \sum_{k=1}^K \alpha_k y_{n+1-k} + h \sum_{k=1}^K \beta_k f_{n+1-k} \quad (9.50)$$

Zakładamy dalej, że układ ten ma jednoznaczne rozwiązanie. Zauważmy, że metoda K -krokowa wymaga znajomości K wektorów startowych y_0, \dots, y_{K-1} , a sformułowanie zadania (9.2) definiuje jedynie pierwszy z tych wektorów. Dalej będziemy zakładać na ogół, że wektory startowe są dokładnymi wartościami rozwiązania w punktach czasowych t_0, \dots, t_{K-1} . Wpływ dokładności wektorów startowych na dokładność rozwiązania jest omówiony zwięzle w podrozdziale 9.3.3.

Do metod wielokrokowych stosują się wprowadzone w podrozdziale 9.1 definicje lokalnego błędu obcięcia i rzędu metody. Lokalny błąd obcięcia metody wielokrokowej:

$$y_{n+1} = \sum_{k=1}^K \alpha_k y_{n+1-k} + h \sum_{k=0}^K \beta_k f_{n+1-k} \quad \text{dla } n = K-1, \dots, N-1 \quad (9.51)$$

jest funkcją kroku całkowania h , gdyż $t_n = t_0 + nh$. Dla metody rzędu p lokalny błąd obcięcia spełnia warunek (9.17) zerowania kolejnych p pochodnych funkcji $r_{n+1}(h)$ w punkcie $h = 0$, tzn. lokalny błąd obcięcia jest rzędu p . Dahlquist udowodnił, że maksymalny rząd liniowej K -krokowej metody zbieżnej wynosi $K+2$, gdy K jest liczbą parzystą, lub $K+1$, gdy K jest liczbą nieparzystą [F2, S1].

Do wyznaczenia wartości wszystkich $2K+1$ współczynników α_k i β_k formuły wielokrokowej (9.49) potrzeba $2K+1$ niezależnych warunków algebraicz-

nych. Jeśli założyć rzad formuły (p), to uzyskujemy $p+1$ warunków wynikających z rozwinięcia lokalnego błędu obcięcia $r_{n+1}(h)$ wokół t_n i przyrównania do 0 współczynników rozwinięcia przy kolejnych potęgach kroku całkowania – do p -tej włącznie. Dla formuły (9.49) rozwinięcie $r_{n+1}(h)$ wokół t_n , dla odpowiednio gładkiego rozwiązania $y(t)$, ma postać:

$$r_{n+1}(h) = \sum_{m=0}^{p+1} h^m C_m y_n^{(m)} + O(h^{p+2}) \quad (9.52)$$

gdzie:

$$\left. \begin{aligned} C_0 &= \sum_{k=1}^K \alpha_k - 1 \\ C_1 &= \sum_{k=1}^K (1-k) \alpha_k + \sum_{k=0}^K \beta_k - 1 \\ C_m &= \frac{1}{m!} \sum_{k=1}^K (1-k)^m \alpha_k + \frac{1}{(m-1)!} \sum_{k=1}^K (1-k)^{(m-1)} \beta_k - \frac{1}{m!} \quad \text{dla } m \geq 2 \end{aligned} \right\} \quad (9.53)$$

Tak więc dla formuły rzędu p uzyskujemy $p+1$ równań:

$$C_m = 0 \quad \text{dla } m = 0, 1, \dots, p \quad (9.54)$$

Jeśli $2K = p$, to warunki (9.54) określają współczynniki formuły jednoznacznie – jak w poniższym przykładzie.

Przykład 9.7. Parametry α_1 , α_2 i β_1 następującej otwartej metody dwukrokowej ($K = 2$):

$$y_{n+1} = \alpha_1 y_n + \alpha_2 y_{n-1} + h \beta_1 f_n \quad (9.55)$$

można wyznaczyć, zapisując wyrażenie na lokalny błąd obcięcia:

$$\begin{aligned} r_{n+1} &= \alpha_1 y(t_n) + \alpha_2 y(t_{n-1}) + h \beta_1 f(t_n, y(t_n)) - y(t_{n+1}) = \\ &= \alpha_1 y(t_n) + \alpha_2 y(t_{n-1}) + h \beta_1 y'(t_n) - y(t_{n+1}) \end{aligned}$$

Po rozwinięciu tego wyrażenia w szereg Taylora uzyskujemy:

$$\begin{aligned} r_{n+1}(h) &= \alpha_1 y(t_n) + \alpha_2 y(t_n - h) + h \beta_1 y'(t_n) - y(t_n + h) = \\ &= \alpha_1 y(t_n) + \alpha_2 \left[y(t_n) - y'(t_n)h + \frac{y''(t_n)h^2}{2} - \frac{y'''(t_n)h^3}{6} + O(h^4) \right] + \\ &\quad + h \beta_1 y'(t_n) - \left[y(t_n) + y'(t_n)h + \frac{y''(t_n)h^2}{2} + \frac{y'''(t_n)h^3}{6} + O(h^4) \right] \end{aligned}$$

Zerowanie kolejnych współczynników przy $y(t_n)$, hy' i h^2y'' prowadzi do warunków:

$$\left. \begin{aligned} \alpha_1 + \alpha_2 - 1 &= 0 \\ -\alpha_2 + \beta_1 - 1 &= 0 \\ \frac{1}{2}\alpha_2 - \frac{1}{2} &= 0 \end{aligned} \right\} \quad (9.56)$$

z których można wyznaczyć poszukiwane wartości współczynników formuły: $\alpha_1 = 0$, $\alpha_2 = 1$ i $\beta_1 = 2$. Dla takich współczynników estymata lokalnego błędu obcięcia może być wyrażona (z błędem $O(h^4)$) w postaci:

$$r_{n+1}(h) \approx h^3 y_n''' \left(-\frac{1}{6}\alpha_2 - \frac{1}{6} \right) = -\frac{1}{3} y''' h^3 \quad (9.57)$$

z której wynika, że formuła jest (jedynie) rzędu $p = 2$.

Ten sam wynik można uzyskać, rozwiązuając układ równań (9.53) i (9.54) oraz zauważając, że (9.57) jest pierwszym niezerowym składnikiem rozwinięcia (9.52), przy czym $C_3 = \alpha_1/3! - 1/3! = -1/3$.

Jeśli $2K > p$, to warunki (9.54) nie określają współczynników formuły jednoznacznie i trzeba dodatkowo narzucić $2K - p$ warunków. Wśród algorytmów wielokrokowych szczególną popularność zdobyły formuły Adamsa, w których zakłada się $K - 1$ dodatkowych warunków:

$$\alpha_2 = \alpha_3 = \dots = \alpha_K = 0 \quad (9.58)$$

Formuły otwarte tej klasy (Adamsa-Bashfortha) narzucają jeszcze jeden warunek: $\beta_0 = 0$. Wynika stąd, że ich rzad wynosi $p = K$, ponieważ $2K + 1$ współczynników może spełniać co najwyżej $p + 1 + K$ warunków algebraicznych. Formuły zamknięte (Adamsa-Moultona) narzucają z kolei warunek $\beta_K = 0$, a więc ich rzad wynosi również $p = K$. Współczynniki formuł Adamsa oraz wartość stałej C_{p+1} w wyrażeniu (9.52) na błąd obcięcia podano w tablicy 9.2.

Tablica 9.2

Wybrane formuły typu Adamsa-Bashfortha i Adamsa-Moultona wraz z rzędem dokładności p i stałą do szacowania błędu obcięcia C_{p+1} ; $f_{n-k} \equiv f(t_{n-k}, y_{n-k})$

Metody Adamsa-Bashfortha	p	C_{p+1}
$y_{n+1} = y_n + hf_n$	1	$\frac{1}{2}$
$y_{n+1} = y_n + \frac{h(3f_n - f_{n-1})}{2}$	2	$\frac{5}{12}$
$y_{n+1} = y_n + \frac{h(23f_n - 16f_{n-1} + 5f_{n-2})}{12}$	3	$\frac{3}{8}$
$y_{n+1} = y_n + \frac{h(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})}{24}$	4	$\frac{251}{720}$
Metody Adamsa-Moultona	p	C_{p+1}
$y_{n+1} = y_n + hf_{n+1}$	1	$-\frac{1}{2}$
$y_{n+1} = y_n + \frac{h(f_{n+1} + f_n)}{2}$	2	$-\frac{1}{12}$
$y_{n+1} = y_n + \frac{h(5f_{n+1} + 8f_n - f_{n-1})}{12}$	3	$-\frac{1}{24}$
$y_{n+1} = y_n + \frac{h(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})}{24}$	4	$-\frac{19}{720}$

9.3.2. STABILNOŚĆ NUMERYCZNA METOD WIELOKROKOWYCH

Z dotychczasowych rozważań zdaje się wypływać wniosek, że metody wielokrokowe są tym dokładniejsze, im wyższy jest rzad lokalnego błędu obcięcia (9.52). Ogólnie jest to przypuszczenie nieuzasadnione, o czym przekonuje następujący przykład.

Przykład 9.8. Skonstrujmy otwartą metodę dwukroikową z lokalnym błędem obcięcia możliwie wysokiego rzędu. Lokalny błąd obcięcia dla takiej metody ma postać:

$$r_{n+1}(h) = \left[\sum_{k=1}^2 \alpha_k y(t_n + (1-k)h) + h \sum_{k=1}^2 \beta_k y'(t_n + (1-k)h) \right] - y(t_n + h)$$

Po rozwinięciu tego wyrażenia w szereg Taylora względem h otrzymujemy:

$$\begin{aligned} r_{n+1}(h) &= \alpha_1 y(t_n) + \alpha_2 \left[y(t_n) - y'(t_n)h + y''(t_n) \frac{h^2}{2} - y'''(t_n) \frac{h^3}{6} + y^{(4)}(t_n) \frac{h^4}{24} \right] + h\beta_1 y'(t_n) + \\ &+ h\beta_2 \left[y'(t_n) - y''(t_n)h + y'''(t_n) \frac{h^2}{2} - y^{(4)}(t_n) \frac{h^3}{6} \right] + \\ &- \left[y_n(t_n) + y'(t_n)h + y''(t_n) \frac{h^2}{2} + y'''(t_n) \frac{h^3}{6} + y^{(4)}(t_n) \frac{h^4}{24} \right] + O(h^5) \end{aligned}$$

Dwukroikową metodę najwyższego rzędu uzyskuje się, dobierając współczynniki α_i, β_j tak, aby wyzerować możliwie wiele składników w tym rozwinięciu. Po przyrównaniu do zera współczynników przy $y_n, hy'_n, h^2y''_n$ i $h^3y'''_n$, uzyskujemy układ równań:

$$\left. \begin{array}{rcl} \alpha_1 & + \alpha_2 & = 0 \\ -\alpha_2 & + \beta_1 & + \beta_2 - 1 = 0 \\ \alpha_2 & - \beta_2 & - 1 = 0 \\ -\alpha_2 & + \beta_2 & - 1 = 0 \end{array} \right\}$$

którego rozwiązaniem są wartości współczynników: $\alpha_1 = -4$, $\alpha_2 = 5$, $\beta_1 = 4$ i $\beta_2 = 2$. Uzyskana w ten sposób zgodna metoda otwarta ma postać:

$$y_{n+1} = -4y_n + 5y_{n-1} + h(4y'_n + 2y'_{n-1}) \quad (9.59)$$

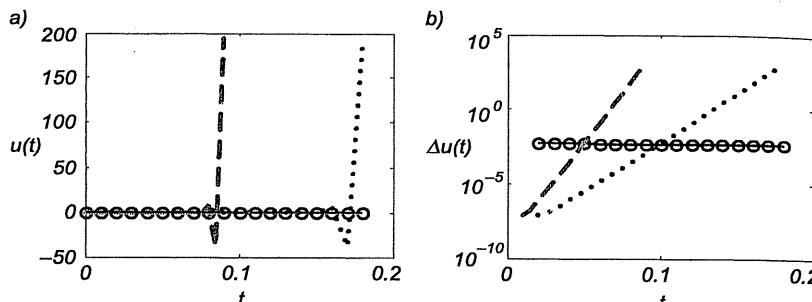
Dla przyjętych wartości współczynników lokalny błąd obcięcia jest rzędu 4, bo:

$$r_{n+1}(h) = y^{(4)}(t_n)h^4 \left(\frac{\alpha_2}{24} - \frac{\beta_2}{6} + \frac{1}{24} \right) + O(h^5) = -\frac{h^4}{12}y^{(4)}(t_n)$$

a więc metoda (9.59) jest potencjalnie rzędu trzeciego.

Porównajmy teraz dokładność rozwiązania równania różniczkowego (9.6), opisującego „standardowy” liniowy układ RC z przykładu 9.1, za pomocą uzyskanej metody otwartej (9.59) z dokładnymi wartościami startowymi: $y_0 = 0$, $y_1 = 1 - e^{-h}$ oraz za pomocą metody otwartej Eulera (rzędu 1). Na rys. 9.7 przedstawiono początkowy fragment rozwiązania dokładnego i przybliżonego. Widać, że rozwiązanie wyznaczone za pomocą metody dwupunktowej jest nie tylko gorsze od rozwiązania uzyskanego metodą jednopunktową, ale wręcz nieużyteczne – już po kilkunastu krokach w rozwiązaniu przybliżonym pojawiają się narastające oscylacje, a dwukrotne zmniejszenie kroku nie tylko ich nie usuwa, ale nawet przyspiesza ich pojawienie się. Jest to potwierdzenie sformułowanej wcześniej tezy, że zgodność metody nie jest warunkiem wystarczającym jej zbieżności; potrzebna jest jeszcze dodatkowa własność – stabilność.

Stabilność metody całkowania opisuje zachowanie błędu rozwiązania przybliżonego e_n dla dużych wartości n . Jeżeli rozwiązanie dokładne $y(t)$ maleje, to y_n będzie praktycznie przydatnym rozwiązaniem wówczas, gdy błąd e_n nie będzie wzrastał wraz z n . Jeżeli rozwiązanie dokładne $y(t)$ rośnie, to błąd nie powinien zwiększać się szybciej niż rozwiązanie. Metoda spełniająca tak postawione wymaganie w pewnym przedziale wartości kroku całkowania h może być nazywana stabilną.



Rysunek 9.7. Wyniki obliczeń do przykładu 9.8: a) rozwiązanie dokładne (linia ciągła) oraz rozwiązania przybliżone uzyskane za pomocą metody Eulera (kółka), metody dwupunktowej z krokiem $h = 0.01$ (linia przerwana) i $h = 0.005$ (linia kropkowana), b) moduły błędów bezwzględnych rozwiązań przybliżonych

Do badania stabilności metod całkowania stosuje się kilka zadań testowych. Spośród nich najważniejsze jest *liniowe zadanie testowe* o postaci:

$$\mathbf{y}'(t) = \mathbf{A} \cdot \mathbf{y}(t) + \mathbf{B} \cdot \mathbf{u}(t) \quad (9.60)$$

Jeżeli macierz \mathbf{A} ma różne wartości własne $\lambda_1, \lambda_2, \dots \in \mathbb{C}$, to istnieje macierz kwadratowa \mathbf{T} taka, że $\mathbf{A} = \mathbf{T}^{-1} \cdot \Lambda \cdot \mathbf{T}$, gdzie $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots)$ jest macierzą diagonalną. Macierz \mathbf{T} umożliwia przekształcenie układu równań różniczkowych zwyczajnych w zbiór niezależnych skalarnych równań różniczkowych zwyczajnych względem nowych zmiennych $\mathbf{z}(t) = \mathbf{T} \cdot \mathbf{y}(t)$:

$$\mathbf{z}'(t) = \Lambda \cdot \mathbf{z}(t) + \mathbf{T} \cdot \mathbf{B} \cdot \mathbf{u}(t)$$

Oznacza to, że wynik badania stabilności równania, zwanego *skalarnym równaniem testowym*:

$$y'(t) = \lambda y(t), \quad y(t_0) = y_0, \quad \lambda \in \mathbb{C} \quad (9.61)$$

może być uogólniony na układ równań (9.60). Wielokrokowa metoda dyskretna (9.49) przekształca równanie testowe (9.61) w równanie różnicowe:

$$y_{n+1} = \frac{1}{1 - h\lambda\beta_0} \sum_{k=1}^K (\alpha_k + h\lambda\beta_k) y_{n+1-k} \quad (9.62)$$

którego rozwiązania są jednoznacznie określone przez warunki początkowe oraz pierwiastki wielomianu charakterystycznego:

$$w(z) = z^K - \frac{1}{1 - h\lambda\beta_0} \sum_{k=1}^K (\alpha_k + h\lambda\beta_k) z^{K-k} \quad (9.63)$$

Położenie pierwiastków wielomianu charakterystycznego (na płaszczyźnie zespolonej) ma zasadnicze znaczenie dla stabilności rozwiązania równania różnicowego.

Przykład 9.9. Stosując formułę różnicową (9.59) do równania różniczkowego (9.6) opisującego „standardowy” liniowy układ RC z przykładu 9.1 – tj. do zadania początkowego $u'(t) = 1 - u(t)$, $u(0) = 1$ – otrzymujemy równanie różnicowe:

$$u_{n+1} + 4(1+h)u_n - (5-2h)u_{n-1} = 0, \quad u_0 = 1$$

któremu odpowiada wielomian charakterystyczny $w(z) = z^2 + 4(1+h)z - (5-2h)$ o dwóch pierwiastkach:

$$z_1 = -2 - 2h + 3\sqrt{1 + \frac{2}{3}h + \frac{4}{9}h^2} \quad \text{i} \quad z_2 = -2 - 2h - 3\sqrt{1 + \frac{2}{3}h + \frac{4}{9}h^2}$$

Gdyby równanie różniczkowe opisujące układ z przykładu 9.1 było jednorodne, co miałoby miejsce dla:

$$e(t) = \begin{cases} 1 & \text{dla } t < 0 \\ 0 & \text{dla } t \geq 0 \end{cases}$$

to rozwiązanie przybliżone miałoby postać:

$$u_n = c_1 z_1^n + c_2 z_2^n$$

Ponieważ jednak $e(t) \geq 0$ dla $t \geq 0$, należy jeszcze znaleźć jakiekolwiek rozwiązanie równania niejednorodnego – w rozważanym przypadku może to być $u_n = 1$ – i dodać je do rozwiązania równania jednorodnego:

$$u_n = 1 + c_1 z_1^n + c_2 z_2^n$$

Wartości stałych:

$$c_1 = -\frac{e^{-h}}{\sqrt{1 + \frac{2}{3}h + \frac{4}{9}h^2}} \quad \text{i} \quad c_2 = -c_1$$

wynikają z przyjętych warunków początkowych, tzn.:

$$u_0 = 1 + c_1 + c_2 = 1 \quad \text{i} \quad u_1 = 1 + c_1 z_1 + c_2 z_2 = 1 - e^{-h}$$

Dla małych długości kroku:

$$z_1 = 1 - h + \frac{1}{2}h^2 - \frac{1}{6}h^3 + \frac{1}{72}h^4 + O(h^5) \quad i \quad z_2 = -5 - 3h + O(h^2)$$

a zatem przy $h \rightarrow 0$ składnik $c_1 z_1^n$ dąży do składowej rozwiązania dokładnego $-e^{-t_n}$, natomiast składnik $c_2 z_2^n$ zachowuje się w granicy jak rozbieżny ciąg $c_2(-5)^n$. To właśnie obecność w rozwiązaniu numerycznym tego składnika (zwanej *rozwiązaniem pasezytniczym*) tłumaczy oscylacje obserwowane w przykładzie 9.8. Ponieważ rozwiązanie dokładne jest stabilne, więc niestabilność przybliżonego rozwiązania jest dowodem niestabilności (a więc i niezbieżności) metody całkowania. Widać, że o zbieżności metody decyduje położenie pierwiastków równania charakterystycznego.

Warunkiem koniecznym i wystarczającym zbieżności zgodnej liniowej metody różnicowej jest, aby wszystkie pierwiastki wielomianu charakterystycznego dla $h=0$ spełniały nierówność $|z_j| \leq 1$, a pierwiastki wielokrotne – nierówność $|z_j| < 1$ [F2]. Metodę spełniającą warunek konieczny i wystarczający zbieżności nazywamy metodą *D-stabilną* albo *stabilną w sensie Dahlquiista*, albo *0-stabilną* [O].

Ponieważ pierwiastki wielomianu charakterystycznego zależą od długości kroku, więc nie można powiedzieć, czy metoda D-stabilna będzie stabilna dla długości kroku $h > 0$. Przekonuje o tym kolejny przykład.

Przykład 9.10. [D2, F2]. Wyznaczmy rozwiązanie równania $y' = -y$, z warunkiem początkowym $y(0) = 1$ za pomocą otwartej metody punktu środkowego:

$$y_{n+1} = y_{n-1} + 2hy'_n \quad \text{dla } y(0) = 1 \quad i \quad y(h) = e^{-h} \quad (9.64)$$

Wielomian charakterystyczny dla tej metody:

$$z^2 + 2hz - 1 = 0$$

ma pierwiastki:

$$z_1 = -h + \sqrt{h^2 + 1} \xrightarrow[h \rightarrow 0]{} 1 \quad i \quad z_2 = -h - \sqrt{h^2 + 1} \xrightarrow[h \rightarrow 0]{} -1$$

Metoda ta jest więc D-stabilna. Rozwiązanie ogólne analizowanego równania ma postać $y_n = c_1 z_1^n + c_2 z_2^n$, przy czym z warunków początkowych wynika, że:

$$c_1 = \frac{e^{-h} + h + \sqrt{h^2 + 1}}{2\sqrt{h^2 + 1}} \xrightarrow[h \rightarrow 0]{} 1 \quad i \quad c_2 = 1 - c_1 \xrightarrow[h \rightarrow 0]{} 0$$

Z drugiej strony:

$$z_1^n \xrightarrow[n \rightarrow \infty]{h=T/n} e^{-T} \quad i \quad z_2^n \xrightarrow[n \rightarrow \infty]{h=T/n} e^T$$

a zatem rozwiązanie przybliżone jest zbieżne do rozwiązania dokładnego przy $h \rightarrow 0$. Dla każdej większej od zera wartości h wystąpi jednak błąd rosnący wykładniczo z długością przedziału całkowania.

Badanie stabilności metody całkowania za pomocą równania testowego sprawdza się do określenia zależności największego z modułów pierwiastków równania charakterystycznego od iloczynu kroku całkowania h i parametru λ skalarnego równania testowego (9.61). Zbiór tych wartości $h\lambda$, dla których równanie różnicowe (powstałe przez dyskretyzację równania testowego za pomocą badanej metody rozwiązywania równań różniczkowych) jest stabilne, nazywamy *obszarem stabilności* metody całkowania.

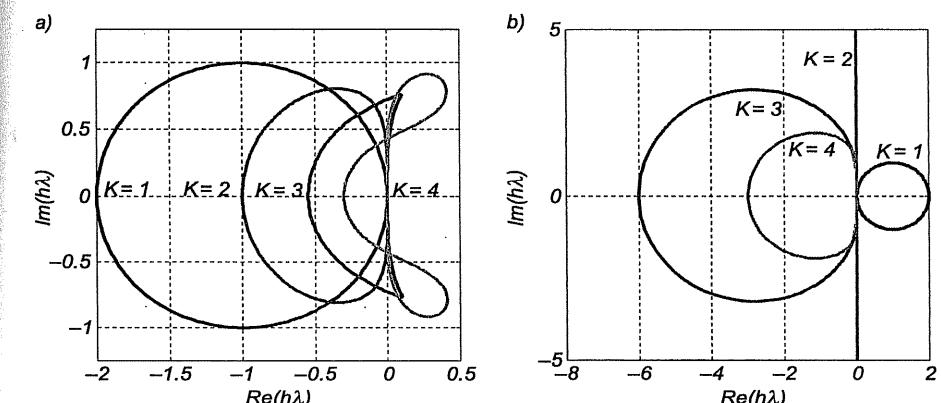
Minimalnym wymaganiem wobec użytecznej metody całkowania jest to, by była ona D-stabilna, to znaczy, aby obszar stabilności zawierał punkt 0. Metodę całkowania nazywamy *absolutnie stabilną* w pewnym obszarze \mathbb{A} płaszczyzny zespolonej \mathbb{C} , jeśli wszystkie pierwiastki wielomianu charakterystycznego spełniają warunek $|z_j| \leq 1$ (a pierwiastki wielokrotne – nierówność $|z_j| < 1$) dla każdego $h\lambda \in \mathbb{A}$.

Jeśli ten obszar stabilności \mathbb{A} zawiera lewą półpłaszczyznę zmiennej zespolonej $h\lambda$, tzn.:

$$h\lambda \in \{z \in \mathbb{C} \mid \operatorname{Re}(z) \leq 0\} \quad (9.65)$$

to metodę nazywamy *A-stabilną*.

Przykład 9.11. Zbadajmy stabilność kilku metod Adamsa, których współczynniki są podane w tablicy 9.2. Metoda Adamsa-Bashfortha rzędu 1 jest otwartą metodą Eulera. Odpowiadający jej obszar absolutnej stabilności określony jest przez nierówność $|1 + h\lambda| \leq 1$. Na rys. 9.8a jest to domknięte koło o środku w punkcie $(-1, j0)$, oznaczone $K = 1$. Metoda ta nie jest więc A-stabilna.

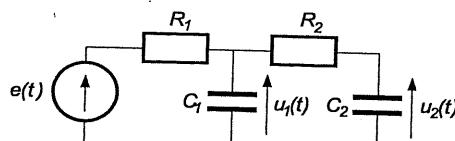


Rysunek 9.8. Obszary absolutnej stabilności dla metod Adamsa: a) otwartych, b) zamkniętych

Metoda Adamsa-Moultona rzędu 1 jest zamkniętą metodą Eulera. Jedy-
nym pierwiastkiem odpowiadającym jej wielomianu charakterystycznego jest
 $z = 1/(1 - h\lambda)$; dlatego obszar absolutnej stabilności opisany jest nierównością
 $|1 - h\lambda| \geq 1$. Na rys. 9.8b jest to domknięte zewnętrze koła, o środku w punkcie
(1, j0), oznaczone $K = 1$. Metoda ta jest więc A-stabilna. Metoda Adamsa-Mo-
ultonowa rzędu 2 jest zamkniętą metodą trapezów. Jedynym pierwiastkiem odpo-
wiadającym jej wielomianu charakterystycznego jest $z = (1 + h\lambda/2) / (1 - h\lambda/2)$.
Ponieważ $|z| \leq 1$ dla wszystkich $h\lambda$ takich, że $\operatorname{Re}(h\lambda) \leq 0$ (lewa półpłaszczyzna
na rys. 9.8b), więc metoda ta jest A-stabilna.

A-stabilność jest silną właściwością, gdyż wymaga stabilności ciągu generowa-
nego przez metodę dyskretyzacji dla każdej długości kroku h , jeżeli tylko cał-
kowany układ równań jest stabilny. Jak wynika z rys. 9.8, wśród metod Adamsa
jedynie metody zamknięte rzędu 1 i 2 są A-stabilne. Udowodniono, że liniowe
metody wielokrokowe rzędu wyższego niż 2 nie mogą być A-stabilne.

Przykład 9.12. Wyznaczyć odpowiedź sieci elektrycznej z rys. 9.9 za pomocą
otwartej metody Eulera, przyjmując następujące wartości parametrów:
 $R_1 = 1 \text{ k}\Omega$, $R_2 = 1 \text{ M}\Omega$, $C_1 = C_2 = 1 \mu\text{F}$ oraz $e(t) = 1(t) \text{ V}$.



Rysunek 9.9. Schemat sieci analizowanej w przykładzie 9.12

Układ równań stanu tej sieci:

$$\begin{bmatrix} u'_1 \\ u'_2 \end{bmatrix} = \begin{bmatrix} -\frac{R_1 + R_2}{R_1 R_2 C_1} & \frac{1}{R_2 C_1} \\ \frac{1}{R_2 C_2} & -\frac{1}{R_2 C_2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} \frac{e(t)}{R_1 C_1} \\ 0 \end{bmatrix} \quad (9.66)$$

po podstawieniu wartości parametrów przybiera postać:

$$\begin{bmatrix} u'_1(t) \\ u'_2(t) \end{bmatrix} = \begin{bmatrix} -1001 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} + \begin{bmatrix} 10^3 \\ 0 \end{bmatrix} \quad \text{dla } t \in [0, T], \quad \begin{bmatrix} u_1(0) \\ u_2(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (9.67)$$

Macierz tego układu równań ma wartości własne: $\lambda_1 = -501 + \sqrt{250001} \approx -0.999$
oraz $\lambda_2 = -501 - \sqrt{250001} \approx -1001$. W celu wyznaczenia równania charaktery-

stycznego przekształcamy układ równań różniczkowych (9.67) w jedno równanie
drugiego rzędu [\dot{Z}]:

$$u''_2(t) + 1002u'_2(t) + 1000u_2(t) = 10^3, \quad u_2(0) = 0, \quad u'_2(0) = 0$$

Jego wielomian charakterystyczny:

$$z^2 + 1002z + 1000 = 0$$

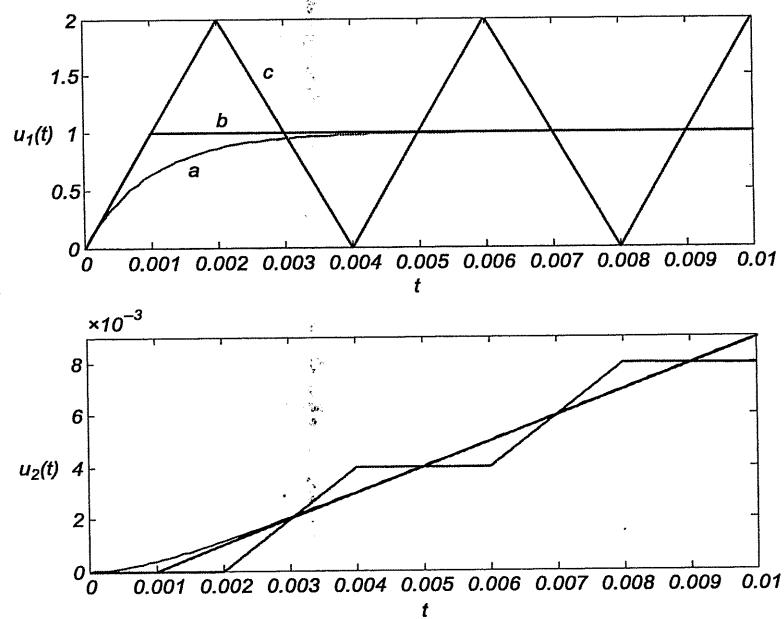
ma dwa pierwiastki: $z_1 = \lambda_1$ i $z_2 = \lambda_2$. Rozwiązanie ogólne można więc zapisać
następująco:

$$u_2(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} + 1 \quad \text{dla } t \geq 0 \quad (9.68a)$$

przy czym z warunków początkowych wynika, że $c_1 = \lambda_2 / (\lambda_1 - \lambda_2) \approx -1.001$
i $c_2 = -1 - c_1 \approx 0.001$. Po wstawieniu $u_2(t)$ do drugiego z równań układu
(9.67) otrzymujemy:

$$u_1(t) = u'_2(t) + u_2(t) = c_1(1 - \lambda_1) e^{\lambda_1 t} + c_2(1 - \lambda_2) e^{\lambda_2 t} + 1 \quad \text{dla } t \geq 0 \quad (9.68b)$$

Na rys. 9.10 przedstawiono dokładne przebiegi obydwu napięć węzłowych,
a także wyniki obliczeń przeprowadzonych za pomocą metody Eulera dla dwóch
długości kroku: $h = 2 \cdot 10^{-3}$ i $h = 10^{-3}$.



Rysunek 9.10. Rozwiązań układu równań z przykładu 9.12: a – dokładne, b – przybliżone dla
kroku $h = 10^{-3}$, c – przybliżone dla kroku $h = 2 \cdot 10^{-3}$

Widac, że dla dłuższego z tych kroków w rozwiązyaniu pojawiają się *pasozytne oscylacje*. W celu wyjaśnienia takiego zachowania metody Eulera rozpatrzmy jej równanie charakterystyczne:

$$z = 1 + h\lambda$$

Jego jedynym pierwiastkiem jest $z_1 = 1 + h\lambda$, a zatem warunek absolutnej stabilności ma postać:

$$|1 + h\lambda| \leq 1$$

Z warunku tego wynika ograniczenie długości kroku zapewniającego stabilność rozwiązyania układu równań (9.67), którego macierz ma dwie rzeczywiste wartości własne ($\lambda_1 \approx -0.999$ i $\lambda_2 \approx -1001$):

$$|1 + h\lambda_j| \leq 1 \Rightarrow -1 \leq 1 + h\lambda_j \leq 1 \Rightarrow -2 \leq h\lambda_j \leq 0 \Rightarrow |h\lambda_j| \leq 2 \Rightarrow h \leq \frac{2}{|\lambda_j|} \quad \text{dla } j = 1, 2$$

Biorąc pod uwagę wartości własne układu równań (9.67), uzyskujemy ograniczenie $h \leq 2 / \max\{|\lambda_1|, |\lambda_2|\} \approx 2 \cdot 10^{-3}$. Ponieważ krok $h = 2 \cdot 10^{-3}$ leży w pobliżu brzegu obszaru stabilności, to w rozwiązyaniu pojawiają się wyraźne oscylacyjne składowe pasozytnicze. Dla $h > 2 \cdot 10^{-3}$ oscylacje te miałyby charakter narastający.

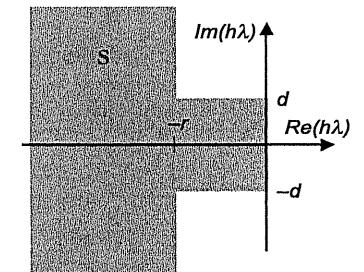
W języku układów elektronicznych ograniczenie długości kroku można wyrazić przez najmniejszą stałą czasową odpowiedzi, tj. stałą czasową ładowania kondensatora C_1 , równą 1 ms. Zauważmy, że czas ustalania się odpowiedzi w całym układzie jest określony przez najdłuższą stałą czasową, tzn. przez stałą czasową ładowania kondensatora C_2 , ok. 1000 razy większą od stałej czasowej ładowania kondensatora C_1 , decydującej o stabilności metody całkowania. Z tych szacunków wynika, że wyznaczenie całego procesu ładowania kondensatorów w układzie z rys. 9.9 wymaga kilku tysięcy kroków całkowania; liczba tych kroków zależy przy tym od stosunku największej do najmniejszej stałej czasowej układu. W układach elektronicznych rozrzucona stała czasowa może być jeszcze większa i dlatego stałokrokowa otwarta metoda Eulera nie jest używana do ich analizy (przynajmniej samodzielnie) ze względu na zbyt małą efektywność.

*

Układy równań różniczkowych zwyczajnych, opisujące sieci elektryczne o znacznie różniących się „stałych czasowych” odpowiedzi, noszą nazwę *równań sztywnych* (ang. *stiff*). Do rozwiązywania takich układów należy używać metod A-stabilnych, które nie wymagają ograniczenia kroku ze względu na stabilność. Trzeba jednak pamiętać, że rzad takich metod nie może przekraczać 2, a więc ograniczenie długości kroku bierze się z wymagań dokładności obliczeń. Zwiększenie efektywności rozwiązywania równań sztywnych, występujących np. w symulacji układów elektronicznych, było motywacją dla wprowadzenia mniej rygorystycz-

nego od A-stabilności, a jednocześnie użytecznego praktycznie, wymagania stabilności zwanego *S-stabilnością*. Pojęcie to, wprowadzone przez Geara, oznacza stabilność metody dla pewnego podobszaru S (rys. 9.11) lewej domkniętej pół-płaszczyzny liczb zespolonych, gdzie leżą wartości $h\lambda$ dla praktycznie istotnych równań sztywnych [O1]:

$$S = \{h\lambda \in \mathbb{C} \mid |\operatorname{Im}(h\lambda)| \leq d \quad \text{i} \quad -r \leq \operatorname{Re}(h\lambda) \leq 0 \quad \text{lub} \quad \operatorname{Re}(h\lambda) \leq -r\} \quad (9.69)$$



Rysunek 9.11. Obszar S-stabilności na płaszczyźnie liczb zespolonych

Gear wprowadził też ważną grupę wielokrotkowych metod S-stabilnych o postaci:

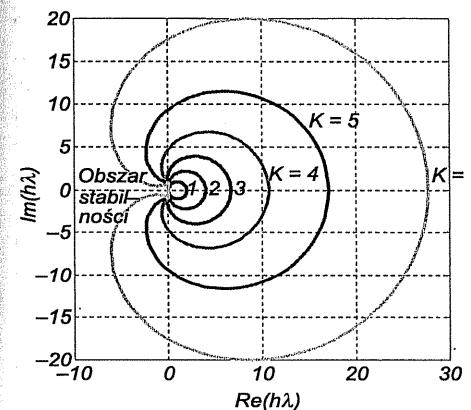
$$\mathbf{y}_{n+1} = h\beta_0 \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}) + \sum_{k=1}^K \alpha_k \mathbf{y}_{n+1-k} \quad (9.70)$$

których współczynniki α_k i β_0 są dobrane tak (tablica 9.3), aby metody te były rzędu K :

$$\beta_0 = \left(\sum_{k=1}^K \frac{1}{k} \right)^{-1}, \quad \alpha_k = \frac{\beta_0}{k} \prod_{j=1, j \neq k}^K \frac{j}{j-k} = (-1)^k \frac{\beta_0}{k} \frac{K!}{k!(K-k)!} \quad \text{dla } k = 1, \dots, K \quad (9.71)$$

Dla $K = 1$ otrzymuje się zamkniętą metodę Eulera, a dla $K = 2$ metodę drugiego rzędu postaci:

$$\mathbf{y}_{n+1} = \frac{2h\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}) + 4\mathbf{y}_n - \mathbf{y}_{n-1}}{3} \quad (9.72)$$



Rysunek 9.12. Obszary absolutnej stabilności metod Geara

Tablica 9.3

Współczynniki zamkniętych K -krokowych metod Geara wraz z rzędem dokładności p i stałą do szacowania błędu obcięcia C_{p+1}

K	β_0	α_1	α_2	α_3	α_4	α_5	α_6	p	C_{p+1}
1	1	1						1	$\frac{1}{2}$
2	$\frac{2}{3}$	$\frac{4}{3}$	$-\frac{1}{3}$					2	$\frac{2}{9}$
3	$\frac{6}{11}$	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$				3	$\frac{3}{22}$
4	$\frac{12}{25}$	$\frac{48}{25}$	$-\frac{36}{25}$	$\frac{16}{25}$	$-\frac{3}{25}$			4	$\frac{12}{125}$
5	$\frac{60}{137}$	$\frac{300}{137}$	—	$\frac{200}{137}$	$-\frac{75}{137}$	$\frac{12}{137}$		5	$\frac{10}{137}$
6	$\frac{60}{147}$	$\frac{360}{147}$	—	$\frac{400}{147}$	—	$\frac{72}{147}$	$-\frac{10}{147}$	6	$\frac{20}{343}$

9.3.3. SCHEMAT PREDYKTOR-KOREKTOR

W praktyce metody zamknięte wypierają otwarte – ze względu na lepszą stabilność i mniejszy wpływ błędów zaokrągleń. Użycie metody zamkniętej wymaga jednak rozwiązania algebraicznego o postaci:

$$y_n = F(t_{n-q}, \dots, t_n; y_{n-q}, \dots, y_n) \quad (9.73)$$

w każdym kroku całkowania. Jeżeli całkowany jest sztywny układ równań, to używa się do tego celu metody Newtona-Raphsona (lub jej modyfikacji). Jako punkt startowy w najprostszym przypadku przyjmuje się rozwiązanie z poprzedniego kroku albo wynik ekstrapolacji rozwiązań w krokach poprzednich na krok bieżący. *Metoda predyktor-korektor* (ekstrapolacyjno-interpolacyjna, przewidywania i korekcji) polega na użyciu zestawu dwóch metod tego samego rzędu: otwartej i zamkniętej, przy czym rozwiązanie uzyskane za pomocą metody otwartej jest dobrym przybliżeniem początkowym dla procesu iteracyjnego rozwiązywania równania (9.73) uzyskanego za pomocą metody zamkniętej.

Spośród omówionych metod w pary łączy się metody Adamsa-Bashfortha z metodami Adamsa-Moultona, a także metody zamknięte Geara z metodami otwartymi (nazywanymi też otwartymi metodami Geara [O]) o postaci:

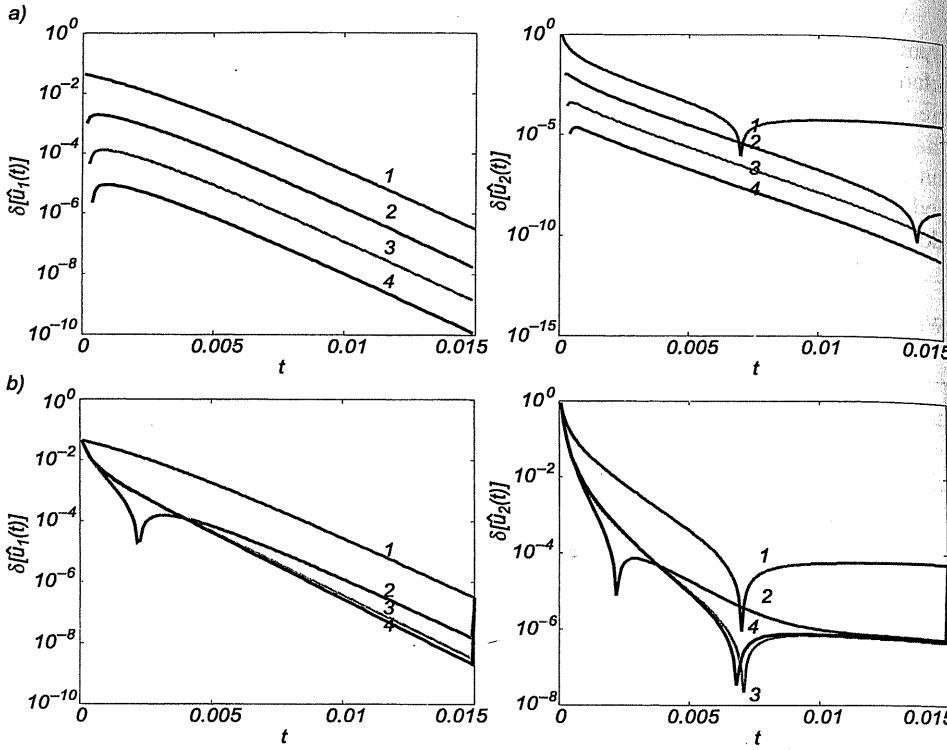
$$\mathbf{y}_{n+1} = h\beta_1 f_n = \sum_{k=1}^K \alpha_k \mathbf{y}_{n+1-k} \quad (9.74)$$

Współczynniki tych metod podano w tablicy 9.4. W każdym przypadku maksymalna długość kroku h jest ograniczona nie tylko względami stabilności i dokładności, ale również zbieżnością metody rozwiązywania równania algebraicznego (9.73). Ponieważ równanie to rozwiązywane jest ze skończoną dokładnością, występuje dodatkowy błąd, wpływający nie tylko na dokładność, ale nawet na stabilność metody całkowania [F2]. Ograniczymy się tutaj do zilustrowania jedynie wpływu niedokładności wartości startowych procedury wielokrokowej na dokładność uzyskanego rozwiązania.

Tablica 9.4
Współczynniki otwartych K -krokowych metod Geara wraz z rzędem dokładności p i stałą do szacowania błędu obcięcia C_{p+1}

K	β_0	α_1	α_2	α_3	α_4	α_5	α_6	p	C_{p+1}
1	1	1						1	$-\frac{1}{2}$
2	2	0	1					2	$-\frac{1}{3}$
3	3	$-\frac{3}{2}$	3	$-\frac{1}{2}$				3	$-\frac{1}{4}$
4	4	$-\frac{10}{3}$	6	$-\frac{2}{3}$	$\frac{1}{3}$			4	$-\frac{1}{5}$
5	5	$-\frac{65}{12}$	10	$-\frac{5}{3}$	$\frac{5}{3}$	$-\frac{1}{4}$		5	$-\frac{1}{6}$
6	6	$-\frac{77}{10}$	15	$-\frac{10}{3}$	5	$-\frac{3}{2}$	$\frac{1}{5}$	6	$-\frac{1}{7}$

Przykład 9.13. Równania (9.67) sieci elektrycznej z przykładu 9.12 rozwiązyano za pomocą K -krokowych zamkniętych metod Geara ($K=1, \dots, 4$) ze stałą długością kroku $h=0.1$ ms. Metody wielokrokowe rzędu K wymagają do rozpoczęcia obliczeń podania wartości rozwiązania w K punktach: $\mathbf{y}_0, \dots, \mathbf{y}_{K-1}$. Eksperymentalnie sprawdzono wpływ dokładności wartości startowych na dokładność rozwiązania. Na rys. 9.13a pokazano względne błędy $\delta[\hat{u}_1(t)]$ i $\delta[\hat{u}_2(t)]$ rozwiązań numerycznych $\hat{u}_1(t)$ i $\hat{u}_2(t)$ dla dokładnych wartości startowych (uzyskanych w przykładzie 9.12). Na rys. 9.13b pokazano analogiczne błędy dla tylej niedokładnej procedury startowej, gdy wartość \mathbf{y}_1 uzyskuje się w wyniku wykonania jednego kroku metody rzędu 1, a następnie wartości \mathbf{y}_0 i \mathbf{y}_1 używa się do wyznaczenia \mathbf{y}_2 metodą rzędu 2 itd. Widoczne jest istotne pogorszenie dokładności wywołane przez niedokładne wartości startowe metod wielokrokowych – większe dla wyższych rzędów metody całkowania.



Rysunek 9.13. Względne błędy rozwiązań numerycznych równań z przykładu 9.12: a) z dokładną i b) niedokładną procedurą startową; rzad zastosowanej metody numerycznej ($p = 1, 2, 3, 4$) wskaźuje cyfra przy odpowiedniej linii na wykresach

9.3.4. WYBÓR KROKU CAŁKOWANIA I RZĘDU METODY

Ogólne zasady wyboru kroku całkowania, dla którego globalny błąd rozwiązania nie przekracza zadanej wartości, są dla metod wielokrokowych podobne jak dla metod jednokrokowych. Błąd lokalny szacuje się według tych samych zasad jak dla metod jednokrokowych, tzn. wykonując ten sam krok za pomocą dwóch metod i wnioskując o błędzie na podstawie różniczki otrzymanych w ten sposób rozwiązań. Dla metod wielokrokowych najwygodniej jest to robić przy użyciu schematu przewidywania i korekcji, w którym stosuje się parę metod (otwartą i zamkniętą) tego samego rzędu, wykorzystujących te same dane o rozwiązaaniu z poprzednich kroków czasowych. Trzeba przy tym zauważyć, że zmiana kroku na bieżąco jest o tyle bardziej skomplikowana dla metod wielokrokowych, że w obliczeniach należy posłużyć się formułami bardziej ogólnymi niż przedstawione w tym rozdziale dla metod stałokrokowych.

Przykład 9.14. Użycie otwartej i zamkniętej metody Eulera ($p=1$) według schematu predyktor–korektor umożliwia oszacowanie lokalnego błędu obcięcia zamkniętej metody Eulera dla wszystkich składowych rozwiązań według następującego wzoru:

$$r_{n+1,m} \approx \frac{1}{2} (y_{n+1,m}^{(0)} - y_{n+1,m}) \quad \text{dla } m=1, \dots, M$$

który wynika z oszacowania różnicy między obydwoma rozwiązaniami:

$$\begin{aligned} y_{n+1,m}^{(0)} - y_{n+1,m} &= \left[y_m(t_{n+1}) + \frac{1}{2} y''_{n,m} h^2 + o(h^3) \right] - \left[y_m(t_{n+1}) - \frac{1}{2} y''_{n,m} h^2 + o(h^3) \right] \approx \\ &\approx y''_{n,m} h^2 \end{aligned}$$

Jako błąd lokalny wektora rozwiązań można przyjąć $|r_{n+1}| = \max\{|r_{n+1,m}| \mid m=1, \dots, M\}$. Wielkość tę można wykorzystać do bieżącej modyfikacji kroku całkowania według następującego algorytmu:

- ① Oblicz przewidywane rozwiązanie w chwili $t_n + h_n$ przy użyciu otwartej metody Eulera:

$$\mathbf{y}_{n+1}^{(0)} = \mathbf{y}_n + h_n \mathbf{f}(t_n, \mathbf{y}_n)$$

- ② Oblicz skorygowane rozwiązanie w chwili $t_n + h_n$ przy użyciu zamkniętej metody Eulera:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$$

Wykorzystaj $\mathbf{y}_{n+1}^{(0)}$ jako przybliżenie początkowe do rozwiązań tego równania algebraicznego względem \mathbf{y}_{n+1} .

- ③ Na podstawie $|\mathbf{y}_{n+1}^{(0)} - \mathbf{y}_{n+1}|$ wyznacz $|r_{n+1}|$. Oblicz maksymalny dopuszczalny błąd dla podprzedziału całkowania o długości h_n :

$$r_{\max} = \frac{h_n}{T} e_{\max}$$

- ④ Jeśli $|r_{n+1}| \leq r_{\max}$, to wynik skorygowany uznaj za wystarczająco dokładny. Kontynuuj obliczenia z krokiem:

$$h_{n+2} = \begin{cases} h_{n+1}, & \text{gdy } \rho r_{\max} < |r_{n+1}| \leq r_{\max} \\ \frac{h_{n+1}}{\max \left\{ \frac{1}{10}, \sqrt[p+1]{\frac{|r_{n+1}|}{r_{\max}}} \right\}}, & \text{gdy } |r_{n+1}| \leq \rho r_{\max} \end{cases}$$

gdzie $\rho \in (0, 1]$.

- 5 Jeśli $|r_{n+1}| > r_{\max}$, to powtóż punkty 1, 2 i 3 algorytmu ze skróconym krokiem, np.:

$$h_{n+1} := \min \left\{ 10, \sqrt[p+1]{\frac{|r_{n+1}|}{r_{\max}}} \right\}$$

Jeżeli całkowane funkcje są odpowiednio gładkie, to wydawać by się mogło, że najlepiej jest użyć stabilnej metody całkowania możliwie wysokiego rzędu. Okazuje się, że nie zawsze tak jest. Jeżeli bowiem mamy dwie metody całkowania, nazwijmy je „a” i „b”, charakteryzujące się lokalnym błędem obcięcia, odpowiednio, $r_a = C_a h^{p_a}$ i $r_b = C_b h^{p_b}$, przy czym $p_a < p_b$, to istnieje taka długość kroku:

$$\hat{h} = (p_b - p_a) \sqrt{\frac{C_a}{C_b}}$$

dla którego $r_a = r_b$. Im stosunek C_b / C_a jest większy, tym mniejsza jest wartość \hat{h} , przy czym metoda wyższego rzędu ma przewagę dla kroków krótszych od \hat{h} . Jeżeli narzucone jest wymaganie dokładności, to dopiero wówczas można wybrać rzad metody pozwalającej na dostatecznie długim kroku całkowania. W praktyce sytuacja jest bardziej złożona, gdyż stałe C_a i C_b trzeba szacować w trakcie całkowania. Dla metod Geara istnieją heurystyczne techniki automatycznego wyboru rzędu i długości kroku [O].

9.3.5. ROZWIĄZYwanIE UKŁADÓW RÓWNAŃ RÓZNICZKOWO-ALGEBRAICZNYCH

Opis układu dynamicznego w postaci tzw. równań stanu (9.1) i (9.2) jest niewygodny w przypadku układów elektronicznych z wieloma elementami niegromadzącymi energię. Bardziej naturalny jest wówczas opis za pomocą układu równań różniczkowo-algebraicznych:

$$g(\mathbf{y}, \mathbf{y}', t) = \mathbf{0} \quad (9.75)$$

Metoda rozwiązywania tego układu równań polega na jego dyskretyzacji, np. przy użyciu wielokrokowej metody różnicowej (9.8):

$$\mathbf{y}'_{n+1} = \gamma_0 \mathbf{y}_{n+1} + \mathbf{d}_n \quad (9.76)$$

gdzie:

$$\gamma_0 = \frac{1}{h\beta_0}, \quad \mathbf{d}_n = -\gamma_0 \sum_{k=1}^K (\alpha_k \mathbf{y}_{n+1-k} + h\beta_k \mathbf{y}'_{n+1-k})$$

Po wstawieniu do tego wzoru przybliżeń pochodnych obliczonych w poprzednich krokach uzyskuje się układ równań algebraicznych, na ogół nieliniowych, postaci:

$$g(\mathbf{y}_{n+1}, \gamma_0 \mathbf{y}_{n+1} + \mathbf{d}_n, t_{n+1}) = \mathbf{0} \quad (9.77)$$

Ponieważ tzw. wektor przeszłości \mathbf{d}_n jest znany, więc po znalezieniu przybliżonego rozwiązania dla $t \leq t_n$, układ ten można rozwiązać względem szukanej odpowiedzi \mathbf{y}_{n+1} , posługując się np. metodą Newtona-Raphsona.

Przykład 9.15. Dla układu RC z przykładu 9.1 mamy następujący układ równań:

$$\begin{cases} iR + u = e \\ Q(u) - q = 0 \\ i - q' = 0 \end{cases} \quad (9.78)$$

Po podstawieniu wartości siły elektromotorycznej e , napięcia u , prądu i i ładunku q w chwili t_{n+1} , a następnie zastosowaniu zamkniętej metody trapezów (rzędu 2) do dyskretyzacji pochodnej ładunku w trzecim równaniu:

$$y'_{n+1} = -y'_n + \frac{2}{h}(y_{n+1} - y_n)$$

uzyskujemy układ równań różniczkowo-algebraicznych o postaci:

$$\begin{cases} i_{n+1}R + u_{n+1} = e_{n+1} \\ Q(u_{n+1}) - q_{n+1} = 0 \\ i_{n+1} - [-q'_n + 2(q_{n+1} - q_n)/h] = 0 \end{cases} \quad (9.79)$$

Zauważmy, że równania te opisują nieliniową sieć elektryczną przedstawioną na rys. 9.14. Ogólnie, nowa sieć elektryczna ma tę samą strukturę co oryginalna, chociaż gałęzie z elementami gromadzącymi energię podlegają transformacji. W naszym przypadku nieliniowy kondensator zostaje zastąpiony przez tzw. stwarzyszony model dyskretny, składający się z nieliniowej konduktancji:

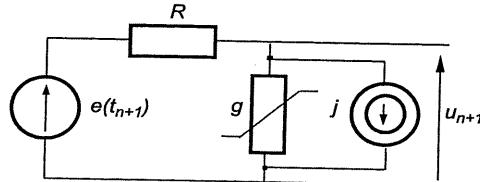
$$g(u_{n+1}) = \frac{2}{h} Q(u_{n+1})$$

równolegle połączonej ze źródłem prądowym o wydajności:

$$j = q'_n - \frac{2}{h} q_n$$

W ten sposób analiza stanów dynamicznych układu została sprowadzona do ciągu analiz statycznych (stałoprądowych) zastępczych układów nieliniowych. Z kolei każda taka analiza statyczna, realizowana za pomocą metody Newtona-

-Raphsona, sprowadza się do ciągu analiz układów zlinearyzowanych [O]. Punkt startowy dla analizy statycznej można uzyskać np. za pomocą dowolnego algorytmu otwartego rzędu 2. Różnice między wartościami uzyskanymi z predykcji a rozwiązaniem końcowym układu zlinearyzowanego można wykorzystać równocześnie do kontroli kroku całkowania – w sposób analogiczny do tego, który opisano w przykładzie 9.14 [O].



Rysunek 9.14. Schemat układu elektrycznego opisanego równaniami (9.79)

Zadanie 9.1. Korzystając z programu MATLAB, wykreślić brzeg obszaru stabilności absolutnej dla wszystkich formuł Adamsa-Bashfortha i Adamsa-Moultona z tablicy 9.2.

Wskazówka. Każdy pierwiastek z wielomianu charakterystycznego $w(z)$, zdefiniowanego wzorem (9.22), zależy od $h\lambda$, tzn. $z = f(h\lambda)$, gdzie f jest pewną funkcją nieliniową. Zależność odwrotna $h\lambda = f^{-1}(z)$ dla $z = \exp(j\varphi)$, gdzie $\varphi \in [0, 2\pi]$, określa parametrycznie brzeg obszaru stabilności. Rozwiązanie zadania przedstawiono na rys. 9.8.

Zadanie 9.2. Korzystając z programu MATLAB, wykreślić brzeg obszaru stabilności absolutnej dla metod Rungego-Kutty, omawianych w podrozdziale 9.2.

Zadanie 9.3. Dla układu z przykładu 9.12 pokazać, że metoda Newtona-Raphsona, zastosowana do rozwiązywania nieliniowych równań sieci elektrycznej (powstałej w wyniku dyskretyzacji układu dynamicznego), prowadzi do rozwiązywania ciągu układów liniowych równań sieci elektrycznych.

Zadanie 9.4. Wyprowadzić formułę otwartej i zamkniętej metody Adamsa drugiego rzędu dla zmiennej długości kroku całkowania. Sprawdzić poprawność metody, rozwiązując równanie z przykładu 9.1. Zastosować następujący algorytm zmiany kroku:

- jeśli $|r_n| < 2^{-p} r_{\max}$, gdzie $p = 2$ jest rzędem algorytmu, to krok należy podwoić;
- jeśli $|r_n| > 2^{-p} r_{\max}$, ale $|r_n| < r_{\max}$, to należy utrzymać dotychczasowy krok;
- jeśli $|r_n| \geq r_{\max}$, to krok należy skrócić do połowy;

gdzie r_n oznacza oszacowanie lokalnego błędu obcięcia, a r_{\max} maksymalną wartość tego błędu.

Zadanie 9.5. Stosując schemat predyktor-korekтор, oparty na parze metod Adamsa rzędu 1, wyznaczyć napięcie na nieliniowym kondensatorze w układzie elektrycznym, przedstawionym na rys. 9.1. Założyć, że $e(t) = 1(t)$, a adekwatnym modelem matematycznym zależności pojemności kondensatora od napięcia jest równanie:

$$C(x) = \frac{C_0}{\sqrt{1 + \frac{x}{U_0}}}$$

gdzie $C_0 = 1 \mu\text{F}$ oraz $U_0 = 0,6 \text{ V}$. Do sterowania długością kroku wykorzystać algorytm z przykładu 9.14. Powtórzyć obliczenia przy użyciu schematu predyktor-korekтор, opartego na parze metod Adamsa rzędu 2. Uzyskane wyniki porównać z rozwiązaniem uzyskanym za pomocą procedury **ode23** programu MATLAB.

PROGRAMY W JĘZYKU MATLAB

Niniejszy dodatek zawiera listingi programów w języku systemu MATLAB (wersja 7.5), zapisanych na płycie CD dołączonej do podręcznika. Są to programy ilustrujące wszystkie te przykłady i zadania, opisane w kolejnych rozdziałach podręcznika, które wymagają użycia komputera. Korzystając z tych programów, należy uwzględnić następujące okoliczności:

- Ich wykonanie w wersjach systemu MATLAB starszych niż wersja 7.5 wymaga zastąpienia definicji funkcji anonimowych plikami funkcyjnymi, umieszczonymi w lokalnym katalogu roboczym.
- Listingi wszystkich funkcji, wykorzystywanych przez dany program, znajdują się pod jego listingiem (nawet wówczas, gdy pojawiły się już wcześniej).
- Rozwiązania graficzne dostosowane zostały do rozdzielczości ekranu komputera 1024×768 ; użycie ekranu o innej rozdzielczości może dawać efekt graficzny niezupełnie zgodny z intencją autorów, ale łatwy do skorygowania.
- Długość wiersza programu ograniczona została do 80 znaków; dlatego ciągi instrukcji i komentarze zawierające więcej niż 80 znaków zostały podzielone na dwa lub trzy wiersze.
- Ze względów dydaktycznych złożoność programów na ogół wzrasta z numerami przykładów i zadań: stosowane są coraz bardziej zaawansowane techniki programowania; nierzadko pokazywane są również inne rozwiązania problemów wcześniej już rozwiązanych.

PROGRAMY DO ROZDZIAŁU 4

Przykład 4.1

```
clear all
clc

% BADANIE WŁASNOŚCI MACIERZY
%-----

% Sprawdzenie czy macierz A jest redukowalna
%-----
A=[1 5 2 6
   0 5 0 6
   3 6 4 5
   0 6 0 5];
[Ar,Pp]=CzyMacierzRedukowalna(A); % odwołanie do zdefiniowanej
                                     % niżej funkcji „CzyMacierzRedukowalna”
fprintf('\nAby kontynuować,naciśnij klawisz\n')
pause

% Sprawdzenie czy macierz B ma własność A
%-----
clc
fprintf('Macierz\n')
B=[1 3 0 3
   2 2 2 0
   0 3 1 3
   2 0 2 2];
fprintf('ma własność A, ponieważ macierz P''*B*B*P, mająca postać\n\n')
PBP=Pp'*B*B*p;
fprintf('%6i%6i%6i%6i\n\n',PBP)
fprintf('spełnia równanie (4.5)\n')
fprintf('\nAby kontynuować,naciśnij klawisz\n')
pause

% Przykład macierzy C diagonalnie słabo dominującej
%-----
clc
fprintf('Macierz\n')
C=diag([3,2,2,2])+ones(4)
fprintf('jest macierzą diagonalnie słabo dominującą\n')

%-----
```

```
function [A,P]=CzyMacierzRedukowalna(A)
% Funkcja sprawdza, czy macierz kwadratowa A o wymiarach NxN jest redukowalna:
% - generuje wszystkich możliwe macierze permutacji P;
% - sprawdza, czy istnieje co najmniej jeden iloczyn P'*A*P mający postać (4.3).
% Dla następującej macierzy redukowalnej o wymiarach 9x9:
%
%    11    0    0    15    9    0    0    0    12
%    10    6    7    13   10    4    5   10   10
%    7    6    6    1    4    13   10   13    3
%   14    0    0   12   13    0    0    0    9
%    6    0    0   15    2    0    0    0    1
%    7    8   11    6    7    3    5    3    6
%   12    9    9    9   11    8   12    0   10
%    3    5    8    5   14   15    4    4   12
%    5    0    0    0    3    0    0    0    7
```

```

% czas wykonania tej funkcji na komputerze z procesorem Pentium 1.8 GHz
% wynosi ponad 18 sek, a dla macierzy o wymiarach 10x10 ponad 7 minut.

if size(A,1)==size(A,2) % sprawdzenie czy macierz A jest kwadratowa
N=size(A,1);
Pall=perms([1:N]); % generowanie macierzy Pall o wymiarach N! x N
% zawierającej wszystkie permutacje liczb od 1 do N
M=size(Pall,1);
Ared=A;
ielmax=0;

% Tworzenie kolejnych macierzy permutacji P i badanie struktury macierzy P'*A*p
%-----


for m=1:M
P=zeros(N);
Pw=Pall(m,:);
for n=1:N
P(n,Pw(n))=1;
end
PAP=P'*A*p;

% Szukanie zerowej podmacierzy o największej liczbie elementów
%-----
for ik=N-1:-1:1
if all(PAP(ik+1:N,1:ik)==0)
wmax=N-ik;
kmax=ik;
iemax=wmax*kmax;
if iemax>ielmax
ielmax=iemax;
Ared=PAP;
Pper=P;
end
end
end

% Prezentacja wyników
%-----


if all(Ared==A)
fprintf('Macierz nie jest redukowalna\n')
else
fprintf('\nMacierz\n')
A
fprintf('jest redukowalna, bowiem iloczyn P''*A*p dla macierzy permutacji')
fprintf('\n')
P=Pper
fprintf('daje macierz zredukowaną\n')
A=Ared
end
else
fprintf('Macierz musi być kwadratowa\n')
end
%-----

```

Przykład 4.2

```

clear all
warning on % włączenie komunikatów ostrzegających
clc

% WYZNACZANIE ZALEŻNOŚCI WSKAŹNIKA UWARUNKOWANIA MACIERZY HILBERTA OD WYMIARU
%-----


n=[2 5 10 50];
k=1;
for N=n % realizacja pętli „for” dla kolejnych elementów wektora n
A=hilb(N); % generowanie macierzy Hilberta o wymiarach NxN
ws(k)=cond(A); % wyznaczenie wskaźnika uwarunkowania
mo=norm(A*inv(A)-eye(N)); % wyznaczanie A*inv(A) i porównanie z macierzą I
fprintf(['Rozmiar macierzy = %2i\n'...
'Wskaźnik uwarunkowania = %10.4e\n'...
'|| A*inv(A)-I || = %10.4e\n'],N,ws(k),mo);
if N<n(length(n))
fprintf('\nAby kontynuować, naciśnij klawisz\n');
k=k+1;
pause
end
end
fprintf('\n\n N %8i %9i %10i %10i\n',n);
fprintf('cond(A) %5.2f %10.2e %10.2e %10.2e\n',ws);
%-----



```

Przykład 4.3

```

clear all
close all
clc
warning off

% WYZNACZANIE WSPÓŁCZYNNIKA PRZENOSZENIA BŁĘDU WEKTORA PRAWYCH STRON b UKŁADU
% RÓWNAŃ LINIOWYCH Z MACIERZĄ A, KTÓREGO ROZWIĄZANIEM JEST WEKTOR x=[1 1...1]':
%-----


% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
        figx+15 5 figx figy
        5 100+figy figx figy
        figx+15 100+figy figx figy]; % położenia rysunków

n=1:10;
R=10000;
for N=n
A1=vander((1:N)); % A=A1 jest macierzą Vandermonde'a
x=ones(N,1);
b1=A1*x;
T1(N)=0;
T2(N)=0;

```

```

A2=rand(N); % A=A2 jest macierzą o elementach pseudolosowych
b2=A2*x;

% Wyznaczanie współczynnika przenoszenia błędu
% dla R realizacji zaburzenia wektora b.
%-----
for r=1:R

    % Generowanie zaburzonych prawych stron układu b1zab i b2zab
    %-----
    b1zab=b1.*(1+(rand(N,1)-0.5)*1e-3);
    b2zab=b2.*(1+(rand(N,1)-0.5)*1e-3);

    % Rozwiązywanie układów równań z zaburzonym wektorem prawych stron
    %-----
    x1zab=A1\b1zab;
    x2zab=A2\b2zab;

    % Wyznaczenie współczynników przenoszenia błędu
    %-----
    T1(N)=T1(N)+(norm(x1zab-x)/norm(x))/(norm(b1zab-b1)/norm(b1));
    T2(N)=T2(N)+(norm(x2zab-x)/norm(x))/(norm(b2zab-b2)/norm(b2));
end

% Obliczenie wartości średniej współczynników przenoszenia błędu
%-----
T1r(N)=(norm(x1zab-x)/norm(x))/(norm(b1zab-b1)/norm(b1));
T2r(N)=(norm(x2zab-x)/norm(x))/(norm(b2zab-b2)/norm(b2));
T1(N)=T1(N)/R;
T2(N)=T2(N)/R;

end

% Prezentacja wyników
%-----
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
semilogy(n,T1r(n),'ok','MarkerFaceColor','k','MarkerSize',4)
hold on
semilogy(n,T1(n),'-k','MarkerFaceColor','k','MarkerSize',4)
semilogy(n,T2r(n),'or','MarkerFaceColor','w','MarkerSize',4)
semilogy(n,T2(n),'-r','MarkerFaceColor','w','MarkerSize',4)
title('Zależność współczynnika przenoszenia błędu od liczby równań');
h=xlabel('\itN');
set(h,'FontName','Times','FontSize',12);
h=ylabel ('\itT ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
l=legend('macierz Vandermonde''a, jedna realizacja zaburzeń',...
    'macierz Vandermonde''a, 10000 realizacji zaburzeń',...
    'macierz pseudolosowa, jedna realizacja zaburzeń',...
    'macierz pseudolosowa, 10000 realizacji zaburzeń',2);
set(l,'FontSize',8);
grid on; zoom on

% Wyznaczanie czasu rozwiązywania układu równań
%-----
R=10000;
n=4:4:48;

```

```

for N=n
    A=rand(N);
    b=sum(A,2);
    tic
    for r=1:R % R-krotne rozwiązywanie układu równań
        x=A\b;
    end
    czas(N)=toc/R; % średni czas jednokrotnego rozwiązywania układu równań w sek.
end

% Prezentacja wyników
%-----
f=figure(2);
set(f,'Pos',figpos(2,:));
hold off
plot(n,czas(n)*1e6,'ok','MarkerFaceColor','w','MarkerSize',4) % przeliczenie
hold on % na mikrosekundy
plot((2:50),polyval(polyfit(n,czas(n)*1e6,2),(2:50)),'r'); % aproksymacja
title('Czas rozwiązywania układu równań'); % wielomianowa
h=xlabel('\itN');
set(h,'FontName','Times','FontSize',12);
h=ylabel ('\it\rm {\it\mu\it s\it\rm } ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
l=legend('średnia czasu 10000 rozwiązań układu równań',...
    'aproksymacja wielomianem drugiego stopnia',2);
set(l,'FontSize',8);
grid on; zoom on

```

Przykład 4.4

```

clear all
format short
warning off
clc

% ROZWIĄZANIE UKŁADU RÓWNAŃ METODĄ LU
%-----

A=[ 1 1 1
     1 2 3
     1.5 2 4]; % macierz układu równań
b=[1
    1
    1]; % wektor prawych stron układu równań
[L,U]=lu(A)
y=L\b % rozwiązywanie układu Ly=b względem wektora y
x=U\y % rozwiązywanie układu Ux=y względem wektora x

```

Przykład 4.6

```

clear all
close all
clc

```

```

% ROZWIĄZYwanIE UKŁADÓw RÓWNAŃ LINIOWYCH METODAMI ITERACYJNYMI
%-----
% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenie rysunku

N=50;
D=-4*eye(N); % macierz diagonalna o elementach równych -4
L=diag(ones(N-1,1),-1); % dolna macierz trójkątna o elementach niezerowych
% (równych 1) wyłącznie na podprzekątnej
L=L+diag(ones(N-2,1),-2); % dolna macierz trójkątna o elementach niezerowych
% (równych 1) wyłącznie na drugiej podprzekątnej
U=L'; % górna macierz trójkątna będąca transpozycją macierzy L

A=L+D+U; % macierz układu równań
x_dokl=ones(N,1); % dokładne rozwiązywanie układu równań
b=A*x_dokl; % wektor prawych stron układu równań

% Metoda Gaussa-Seidla wg wzoru (4.48)
% Wyznaczanie błędu rozwiązywania dla 100 iteracji
%-----
x=(1:N)'; % przybliżenie początkowe
bw_gs(1)=norm(x-x_dokl)/norm(x_dokl); % wskaźnik błędu przybliżenia początkowego
for iter=2:100
    for m=1:N
        x(m)=x(m)-1/A(m,m)*(A(m,1:m-1)*x(1:m-1)+A(m,m:N)*x(m:N)-b(m));
    end
    bw_gs(iter)=norm(x-x_dokl)/norm(x_dokl); % wskaźnik błędu iter-tego
end % przybliżenia
figure(1);
set(f,'Pos',figpos(1,:));
hold off
f=semilogy((4:4:100),bw_gs(4:4:100),'o-k','MarkerFaceColor','w','MarkerSize',4);
hold on
h=xlabel('\iti\rm');
set(h,'FontName','Times','FontSize',12);
figure(1)
grid on
g=get(f);
set(g.Parent,'MinorGridLineStyle','none');
pause(0.1)

% Metoda SOR wg wzoru (4.51)
% Wyznaczanie błędu rozwiązywania dla 100 iteracji
%-----
omega=1.80; % parametr metody SOR
M_omega=inv(D+omega*L)*((1-omega)*D-omega*U); % macierz zdefiniowana pod (4.51)
w_omega=omega*inv(D+omega*L)*b; % wektor zdefiniowany pod wzorem (4.51)
x=(1:N)'; % przybliżenie początkowe
bw_sor180(1)=norm(x-x_dokl)/norm(x_dokl); % wskaźnik błędu przybliżenia
for iter=2:100 % początkowego
    x=M_omega*x+w_omega; % iter-te przybliżenie
    bw_sor180(iter)=norm(x-x_dokl)/norm(x_dokl); % wskaźnik błędu iter-tego
end % przybliżenia
f=semilogy((4:4:100),bw_sor180(4:4:100),'s-k','MarkerFaceCol','w','MarkerSize',4);
hold on
h=xlabel('(\iti\rm)^{1-1.80} / (\iti\rm)^{1-1.84}');
set(h,'FontName','Times','FontSize',12);
figure(1)
grid on
g=get(f);
set(g.Parent,'MinorGridLineStyle','none');
pause(0.1)

```

```

%-----%
bw_sor180(iter)=norm(x-x_dokl)/norm(x_dokl); % wskaźnik błędu iter-tego
% przybliżenia
semilogy((4:4:100),bw_sor180(4:4:100),'o-k','MarkerFaceCol','k','MarkerSize',4);
figure(1);
pause(0.1)

% Metoda SOR wg wzoru (4.51)
% Wyznaczanie błędu rozwiązywania dla 100 iteracji
%-----
omega=1.84; % parametr metody SOR
M_omega=inv(D+omega*L)*((1-omega)*D-omega*U); % macierz zdefiniowana pod (4.51)
w_omega=omega*inv(D+omega*L)*b; % wektor zdefiniowany pod wzorem (4.51)
x=(1:N)'; % przybliżenie początkowe
bw_sor184(1)=norm(x-x_dokl)/norm(x_dokl); % wskaźnik błędu przybliżenia
for iter=2:100 % początkowego
    x=M_omega*x+w_omega; % iter-te przybliżenie
    bw_sor184(iter)=norm(x-x_dokl)/norm(x_dokl); % wskaźnik błędu iter-tego
end % przybliżenia
semilogy((4:4:100),bw_sor184(4:4:100),'s-k','MarkerFaceCol','w','MarkerSize',4);
legend('GS','SOR - 1.80','SOR - 1.84',3);
h=ylabel('||\bf{x}\rm^{(\iti\rm)} - \bf{x}\rm^{(\iti\rm)||} ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
t1=text(-19,0.019,'_\_\_\_\_\_');
t2=text(-15,0.005,'|\bf{x}\rm||');
set(t1,'FontName','Times','FontSize',12);
set(t2,'FontName','Times','FontSize',12);
title('Błędy względne rozwiązywania w funkcji liczby iteracji');
axis([0 100 1e-6 1e2]);
figure(1)
%-----%

```

Przykład 4.7

```

clear all
clc
format short
warning off

% ROZWIĄZYwanIE UKŁADÓw RÓWNAŃ LINIOWYCH A*x=b METODA QR
%-----
A=[ 1 1 1
     1 2 3
     1.5 2 4]; % macierz układu równań
b=[1
    1
    1]; % wektor prawych stron układu równań
[Q,R]=qr(A)
y=Q'*b % obliczenie wektora y=Q'b
x=R\y % rozwiązywanie układu równań z górną macierzą trójkątną Rx=y względem x
%-----%

```

Przykład 4.8

```

clear all
close all
clc
warning off

% ROZWIĄZANIE NADOKREŚLONEGO UKŁADU RÓWNAŃ A*x=b, KTÓREGO ROZWIĄZANIEM
% DOKŁADNYM JEST WEKTOR x=[-1 1 1]', METODA NAJMIEJSZYCH KWADRATÓW,
% A JEST MACIERZĄ O WYMIAΡACH 3Kx3, b JEST WEKTOREM 3K-WYMIAROWYM; K=1,2,...,30
%=====

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenie rysunku

% Definicja zadania
%-----
A1=[1 1 1
     1 2 1
     1 1 3]; % kwadratowa (3x3) podmacierz macierzy układu równań
x_dokl=[-1
         1
         1]; % rozwiązanie dokładne
b1=A1*x_dokl; % wektor prawych układow równań

% Formowanie i rozwiązywanie układu równań dla K=1,2,...,30
%-----
A_zab=[];
b_zab=[];
zab=0.01;
K=30;
for k=1:K
    A_zab1=A1.* (1+(rand(3,3)-0.5)*zab); % k-ta wersja zaburzonej podmacierzy A1
    A_zab=[A_zab;A_zab1]; % k-ta wersja zaburzonej macierzy A
    b_zab1=b1.* (1+(rand(3,1)-0.5)*zab); % k-ta wersja zaburzonego podwektora b1
    b_zab=[b_zab;b_zab1]; % k-ta wersja zaburzonego wektora b1
    [Q,R]=qr(A_zab'*A_zab); % rozkład QR macierzy A_zab
    y=Q'*A_zab'*b_zab; % rozwiązanie układu równań z macierzą ortogonalną
    % Q*y=b_zab względem wektora y
    x=R\y; % rozwiązanie układu równań z górną macierzą trójkątną Rx=y względem x
    bw(k)=norm(x-x_dokl)/norm(x_dokl);
end

% Prezentacja uzyskanych wyników
%-----
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
f=semilogy((1:K),bw,'*':k');
axis([0,30,1e-4,1e-1]);
g=get(f);
set(g.Parent,'MinorGridLineStyle','none');
h=xlabel('itK');
set(h,'FontName','Times','FontSize',12);

```

```

h=xlabel('||\bfx\rm^{\{(\it{i}\}\rm)}-\bfx\rm||');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
t1=text(-5,0.004,'_____');
set(t1,'FontName','Times','FontSize',12);
t2=text(-4,0.0024,'||\bfx\rm||');
set(t2,'FontName','Times','FontSize',12);
title('Względny błąd rozwiązania');
grid on
%=====

```

Zadanie 4.1

```

clear all
clc

% WYZNACZANIE PRĄDÓW W REZYSTANCYJNEJ SIECI ELEKTRYCZNEJ
%-----

% Formowanie układu równań opisujących sieć
% -R1*i(1)+R2*i(2) -R4*i(4) =0 - suma napięć w oczku R1-R2-R4
% -R3*i(3)+R4*i(4)+R5*i(5)=0 - suma napięć w oczku R3-R4-R5
% R1*i(1) +R3*i(3) =e - suma napięć w oczku e-R1-R3
% i(1) -i(3) -i(4) =0 - suma prądów w węźle R1-R3-R4
% i(2) -i(4) -i(5)=0 - suma prądów w węźle R2-R4-R5

R1=1;R2=2;R3=3;R4=0.5;R5=6;e=1; % parametry sieci

A=[-R1 R2 0 -R4 0;
   0 0 -R3 R4 R5;
   R1 0 R3 0 0 ;
   1 0 -1 -1 0;
   0 1 0 -1 -1]; % macierz układu równań opisujących sieć
b=[0;0;e;0;0]; % wektor prawych stron układu równań opisujących sieć

fprintf('\n Rozwiązanie dokładne układu równań:\n')
i=A\b % rozwiązanie układu równań A*i=b względem wektora prądów i
fprintf('\n Aby kontynuować, naciśnij klawisz\n')
pause

% Badania wrażliwości wektora prądów na rozrzut parametrów układu
%-----
R1=R1*1.01; % wartość parametru R1 zaburzona o 1%
A=[-R1 R2 0 -R4 0;
   0 0 -R3 R4 R5;
   R1 0 R3 0 0 ;
   1 0 -1 -1 0;
   0 1 0 -1 -1]; % macierz układu równań opisujących sieć z zaburzoną R1
fprintf('\n Rozwiązanie układu równań przy zmianie rezystancji R1 o 1%:\n')
il=A\b

% Prezentacja wyników
fprintf('\n Zmiana rezystancji R1 o 1% powoduje\n')
fprintf('względna zmiana prądu il: o %5.3f%%\n',(il(1)-i(1))/i(1)*100);
Ti=(abs(norm(il)-norm(i))/norm(i)*100; % wskaźnik wrażliwości normy
% wektora prądów na zaburzenie R1
fprintf('oraz względna zmiana normy wektora prądów i o %6.3f%%\n',Ti)
fprintf('\n Aby kontynuować, naciśnij klawisz\n')
pause

```

```

R1=1;
R2=R2*1.01; % wartość parametru R2 zaburzona o 1%
A=[-R1 R2 0 -R4 0;
  0 0 -R3 R4 R5;
  R1 0 R3 0 0 ;
  1 0 -1 -1 0;
  0 1 0 -1 -1]; % macierz układu równań opisujących sieć z zaburzonym R2
fprintf('\n Rozwiązańe układu równań przy zmianie rezystancji R2 o 1%:\n')
i2=A\b
fprintf('\nZmiana rezystancji R2 o 1% powoduje\n')
fprintf('względna zmiana prądu i2 o %5.3f%\n',(i2(2)-i(2))/i(2)*100);
Ti=(abs(norm(i2)-norm(i)))/norm(i)*100; % wskaźnik wrażliwości normy
% wektora prądów na zaburzenie R2
fprintf('oraz względna zmiana normy wektora prądów i o %6.3f%\n',Ti)
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

R2=2;
R3=R3*1.01; % wartość parametru R3 zaburzona o 1%
A=[-R1 R2 0 -R4 0;
  0 0 -R3 R4 R5;
  R1 0 R3 0 0 ;
  1 0 -1 -1 0;
  0 1 0 -1 -1]; % macierz układu równań opisujących sieć z zaburzonym R3
fprintf('\n Rozwiązańe układu równań przy zmianie rezystancji R3 o 1%:\n')
i3=A\b
fprintf('\nZmiana rezystancji R3 o 1% powoduje\n')
fprintf('względna zmiana prądu i3 o %7.5f%\n',(i3(3)-i(3))/i(3)*100);
Ti=(abs(norm(i3)-norm(i)))/norm(i)*100; % wskaźnik wrażliwości normy
% wektora prądów na zaburzenie R3
fprintf('oraz względna zmiana normy wektora prądów i o %6.3f%\n',Ti)
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

R3=3;
R4=R4*1.01; % wartość parametru R4 zaburzona o 1%
A=[-R1 R2 0 -R4 0;
  0 0 -R3 R4 R5;
  R1 0 R3 0 0 ;
  1 0 -1 -1 0;
  0 1 0 -1 -1]; % macierz układu równań opisujących sieć z zaburzonym R4
fprintf('\n Rozwiązańe układu równań przy zmianie rezystancji R4 o 1%:\n')
i4=A\b
fprintf('\nZmiana rezystancji R4 o 1% nie powoduje\n')
fprintf('zmiany rozwiązania\n');
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

R4=0.5;
R5=R5*1.01; % wartość parametru R5 zaburzona o 1%
A=[-R1 R2 0 -R4 0;
  0 0 -R3 R4 R5;
  R1 0 R3 0 0 ;
  1 0 -1 -1 0;
  0 1 0 -1 -1]; % macierz układu równań opisujących sieć z zaburzonym R5
fprintf('\n Rozwiązańe układu równań przy zmianie rezystancji R5 o 1%:\n')
i5=A\b
fprintf('\nZmiana rezystancji R5 o 1% powoduje\n')
fprintf('względna zmiana prądu i5 o %5.3f%\n',(i5(5)-i(5))/i(5)*100);
Ti=(abs(norm(i5)-norm(i)))/norm(i)*100; % wskaźnik wrażliwości normy
% wektora prądów na zaburzenie R5
fprintf('oraz względna zmiana normy wektora prądów i o %6.3f%\n',Ti)

```

PROGRAMY DO ROZDZIAŁU 5

Przykład 5.1

```

clear all
close all
warning off
clc

% ROZWIAZYWANIE NIELINIOWEGO RÓWNANIA ALGEBRAICZNEGO x*x-x-2=0
% METODA BISEKCJI, METODA SIECZNYCH I METODA NEWTONA
%=====

% Metoda bisekcji wg sieci działań z rys. 5.1
%-----

x_dokl=2; % rozwiązanie dokładne
a=1.5; b=3; % lewa i prawa granica przedziału poszukiwania rozwiązania
dokladosc=4e-15; % wskaźnik pożądanej dokładności rozwiązania
deltax=(b-a)/2; % oszacowanie błędu bezwzględnego pierwszego przybliżenia

iter=0; % początek pętli iteracyjnej (metody bisekcji)
while deltax>dokladosc
    iter=iter+1;
    x=(a+b)/2;
    if (f51(a)*f51(x)>0) % odwołanie do zdefiniowanej niżej funkcji f51
        a=x;
    else b=x;
    end
    deltax=b-a; % oszacowanie błędu bezwzględnego kolejnego przybliżenia
    dx_bis(iter)=abs(x_dokl-x)/x_dokl; % moduł błędu względnego kolejnego
end % koniec pętli iteracyjnej (metody bisekcji)           przybliżenia

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenie rysunku

f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
semilogy(dx_bis,:x')
hold on

% Metoda siecznych wg wzoru (5.6)
%-----

ximinus1=3;
xi=1.5;
deltax=abs(f51(xi));
dokladosc=1e-15; % wskaźnik pożąданiej dokładności rozwiązania

iter=0; % początek pętli iteracyjnej (metody siecznych)
while deltax>dokladosc
    iter=iter+1;
    xiplus1=xi-(xi-ximinus1)/(f51(xi)-f51(ximinus1))*f51(xi);
    deltax=abs(f51(xiplus1));

```

```

ximinus1=xi;
xi=xiplus1;
dx_siecz(iter)=abs(x_dokl-xi)/x_dokl; % moduł błędu względnego kolejnego
end % koniec pętli iteracyjnej (metody siewcznych) przybliżenia
semilogy(dx_siecz,:*r')

% Metoda Newtona wg wzoru (5.4)
%-----

xi=3;
deltax=abs(f51(xi));
dokladnosc=1e-15; % wskaźnik pożąданej dokładności rozwiązania

iter=0; % początek pętli iteracyjnej (metody Newtona)
while deltax>dokladnosc
    iter=iter+1;
    xiplus1=xi-f51(xi)/fprim51(xi); % odwołanie do zdefiniowanej
    xi=xiplus1; % niżej funkcji fprim51
    deltax=f51(xi);
    dx_newton(iter)=abs(x_dokl-xi)/x_dokl; % moduł błędu względnego kolejnego
end % koniec pętli iteracyjnej (metody Newtona) przybliżenia

% Prezentacja wyników
%-----

semilogy(dx_newton,:ok')
l=legend('metoda bisekcji','metoda siewcznych','metoda Newtona');
set(l,'FontSize',8);
title('Zależność modułu błędu względnego od liczby iteracji');
xlabel('Numer iteracji');
ylabel('Moduł błędu względnego rozwiązania');
grid on
fprintf('\n
fprintf(['METODA      1      2      3      4      5',...
       '      ... ...
       '6\n'])
fprintf('BISEKCJI %10.2e %10.2e %10.2e %10.2e %10.2e\n',dx_bis(1:6))
fprintf('SIECZNYCH %10.2e %10.2e %10.2e %10.2e %10.2e\n',dx_siecz(1:6))
fprintf('NEWTONA  %10.2e%11.2e%11.2e%11i%11i\n',[dx_newton(1:4),0,0])
%-----
```

```

function y=f51(x)
% obliczanie wartości funkcji definiującej równanie w przykładzie 5.1
y=x.*x-x-2; % mnożenie z kropką, aby funkcja działała poprawnie dla wektora arg.
%-----
```

```

function y=fprim51(x)
% obliczanie wartości pochodnej funkcji definiującej równanie w przykładzie 5.1
y=2*x-1;
%-----
```

Zadanie 5.1

```

clear all
close all
clc
format long
%-----
```

```

% ROZWIĄZYwanIE UKŁADU TRZECH RÓWNAŃ NIELINIOWYCH
% POPRZEZ SPRAWOZDZENIE TEGO UKŁADU DO JEDNEGO RÓWNANIA WZGLEDEM CZĘSTOTLIWOŚCI
% I ROZWIĄZANIE TEGO RÓWNANIA ZA POMOCĄ PROCEDURY „fzero”
%-----
```

```

for xs=8e9:1e9:5e10 % wyznaczanie i pokazywanie kolejnych rozwiązań;
    % eliminacja rozwiązań fałszywych
    [x,f]=fzero('zad51',xs); % odwołanie do zdefiniowanej niżej funkcji 'zad51'
    if abs(f)<1e-3
        fprintf('\nCzęstotliwość drgań własnych fn=%7.4f [GHz]',x*1e-9);
        fprintf('\nWartość funkcji dla uzyskanego rozwiązania %10.4e\n',f);
        fprintf('\nAby kontynuować, naciśnij klawisz\n')
        pause
    end
end
%-----
```

```

function z=zad51(f)
% Funkcja definiująca równanie nieliniowe względem częstotliwości
% równoważne rozwiązywanemu układowi trzech równań
%-----
c=3e+8;
h=3e-3;
a=0.025;
L=0.025;
h=0.003;
epsilon_r=10;
u1=(2*pi*f).^2/c/c;
u2=3.83171.^2/a/a;
k2=u1*epsilon_r-u2;
k=k2.^0.5;
k02=u1-u2;
k0=k02.^0.5;
z=sin(k*h).*cos(k0*(L-h))./k+sin(k0*(L-h)).*cos(k*h)./k0;
%-----
```

Przykład 5.2

```

clear all
close all
clc
format long
% WYZNACZANIE PIERWIASTKÓW ZESPOLONYCH WIELOMIANU x^4+1=0
%-----
```

```

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenie rysunku
% WARIANT 1: rozkład wielomianu na trójmiany kwadratowe metodą Bairstowa
%-----
```

```

% Punkt startowy: u0=1, v0=2
%-----
```

```

uv0=[1;2]; % punkt startowy
dok=1; % wskaźnik pożąданie dokładności rozwiązania
uvi=uv0;
iter=0;
uinf=sqrt(2);vinf=1; % rozwiązanie dokładne

while dok>1e-15 % początek pętli iteracyjnej realizowanej wg wzoru (5.19)
    uvip1=uvi-muvp52(uvi)*fuvp52(uvi); % odwołanie do zdefiniowanych niżej funkcji
        % muvp52 i fuvp52
    dok=norm(uvip1-uvi); % norma różnicy między dwoma kolejnymi przybliżeniami
    uvi=uvip1;
    iter=iter+1;
    delta(iter)=sqrt((uvi(1)-uinf)^2+(uvi(2)-vinf)^2);
end % koniec pętli iteracyjnej

% Prezentacja wyników dla punktu startowego u0=1, v0=2
%-----

delta(find(delta<eps))=eps; % zastąpienie zer liczbą eps
fprintf('\nu=\%17.15f',uvi(1)); % (ze względu na wykres logarytmiczny)
fprintf('v=\%17.15f\n',uvi(2));
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
semilogy(1:iter,delta,'o-k');
h=xlabel('\it{i}'); % etykieta osi x
set(h,'FontName','Times','FontSize',12);
h=ylabel('\it{Delta}_i ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
t=title(['Zależność błędu od liczby iteracji dla punktu startowego ',...
    '\it{u}\rm_0=-2, \it{v}\rm_0=2']);
set(t,'FontSize',8);
grid on
%=====

% WARIANT 2: wyznaczanie wartości własnych macierzy stwarzyszonej
%-----

clear delta
uv0=[-2;2];
dok=1;
uvi=uv0;
iter=0;
uinf=-sqrt(2);
vinf=1;

while dok>1e-15 % początek pętli iteracyjnej realizowanej wg wzoru (5.19)
    uvip1=uvi-muvp52(uvi)*fuvp52(uvi); % odwołanie do zdefiniowanych niżej funkcji
        % muvp52 i fuvp52
    dok=norm(uvip1-uvi); % norma różnicy między dwoma kolejnymi przybliżeniami
    uvi=uvip1;
    iter=iter+1;
    delta(iter)=sqrt((uvi(1)-uinf)^2+(uvi(2)-vinf)^2);
end % koniec pętli iteracyjnej

% Prezentacja wyników dla punktu startowego u0=-2, v0=2
%-----

delta(find(delta<eps))=eps;% zastąpienie zer liczbą eps
fprintf('\nu=\%17.15f',uvi(1)); % (ze względu na wykres logarytmiczny)

```

```

fprintf('v=\%17.15f\n',uvi(2));
figure(1)
hold off
semilogy(1:iter,delta,'o-k');
h=xlabel('\it{i}'); % etykieta osi x
set(h,'FontName','Times','FontSize',12);
h=ylabel('\it{Delta}_i ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
t=title(['Zależność błędu od liczby iteracji dla punktu startowego ',...
    '\it{u}\rm_0=-2, \it{v}\rm_0=2']);
set(t,'FontSize',8);
grid on
%=====

% WARIANT 2: wyznaczanie wartości własnych macierzy stwarzyszonej
%-----

C=[ 0 0 0 -1 % macierz stwarzyszona wg wzoru (5.20)
    1 0 0 0
    0 1 0 0
    0 0 1 0];
fprintf('\nWyznaczone wartości własne macierzy stwarzyszonej\n');
eig(C)
format short
%=====

function [cdprim]=muvp52(uvi)
% Wyznaczanie macierzy odwrotnej do macierzy pochodnych wg wzoru (5.19)
%
ui=uvi(1);
vi=uvi(2);
J=[-3*ui.^2+2*vi,2*ui;-2*ui*vi,2*vi-ui.^2];
cdprim=inv(J);
%=====

function [cd]=fuvp52(uvi)
% Wyznaczanie wartości funkcji definiujących układ równań (5.18)
%
ui=uvi(1);
vi=uvi(2);
cd=[2*ui.*vi-ui.^3;vi.^2-ui.^2.*vi+1];
%=====

Przykład 5.3

clear all
close all
clc

% ANALIZA UKŁADU ELEKTRYCZNEGO Z DIODĄ TUNELOWĄ
%=====

% Parametry rysunków
%
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków

```

```

figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenie rysunku

% WARIANT 1: sprowadzenie układu równań do jednego równania względem napięcia
% na diodzie i rozwiązanie tego równania za pomocą procedury „fzero”
%-----
ud=fzero('f53',0); % odwołanie do zdefiniowanej niżej funkcji f53
k=1;
jg=1;
Rl=10;
id=k*ud*(ud*ud/3-3*ud/2+2); % wyznaczenie prądu id dla znalezionego rozwiązania. Ud
udr=ud;
idr=id;
fprintf('\nRozwiązań uzyskane za pomocą procedury „fzero”:\n')
fprintf('      ud=%6.4f id=%6.4f\n',ud,id)
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

% WARIANT 2: rozwiązanie układu równań za pomocą procedury „fsolve”
%-----

% punkt startowy (2, 0.5)
%-----
r1=fsolve('u53',[2;0.5],optimset('fsolve')); % odwołanie do zdefiniowanej niżej
funkcji „u53”
r1=fsolve('u53',[2;0.5]); % odwołanie do zdefiniowanej niżej funkcji „u53”
ud=r1(1);
id=r1(2);
fprintf('\nRozwiązań uzyskane za pomocą procedury „fsolve”:\n')
fprintf('      dla punktu startowego [2;0.5]:\n')
fprintf('      ud=%6.4f id=%6.4f\n',ud,id)

% punkt startowy (0.4, 0.05)
%-----
r2=fsolve('u53',[0.4,0.05],optimset('fsolve'));
ud=r2(1);
id=r2(2);
fprintf('\n      dla punktu startowego [0.4;0.05]:\n')
fprintf('      ud=%6.4f id=%6.4f\n',ud,id)
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

% WARIANT 3: rozwiązanie układu równań metodą graficzną
%-----
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
ud=(0:0.1:3);
id=k*ud.* (ud.*ud/3-3*ud/2+2);
hold off
plot(ud,id)
id=jg-ud/Rl;
hold on
plot(ud,id,'.r');
h=line([udr udr],[0 idr]);
set(h,'Color','g','LineStyle',':');
h=line([0 udr],[idr idr]);
set(h,'Color','g','LineStyle',':');


```

```

h=xlabel('\it{u}_d\backslash rm [V]');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\it{i}_d\backslash rm [mA]');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
text(0.5,1,'ig-ud/Rl');
h=line([r2(1) r2(1)], [0 r2(2)]);
set(h,'Color','k','LineStyle','');
text(1.2,0.5,'minimum lokalne');
annotation('arrow',[0.5 r2(1)/2.68],[0.42 r2(2)/3*2]);
title('Graficzne rozwiązanie układu równań')

%-----
function y=f53(Ud)
% Wyznaczanie wartości funkcji reprezentującej równanie
% z jedną niewiadomą, do którego został sprowadzony układ równań
%-----
k=1;jg=1;Rl=10;
id=k*Ud*(Ud*Ud/3-3*Ud/2+2);
y=Ud/Rl+id-ig;

%-----
function y=u53(x)
% Wyznaczanie wartości dwóch funkcji definiujących układ równań
%-----
k=1;jg=1;Rl=10; % parametry układu
ud=x(1); id=x(2); % zmiana nazw zmiennych wejściowych ułatwiająca interpretację
y(1,1)=id-k*ud*(ud*ud/3-3*ud/2+2);
y(2,1)=ud/Rl+id-ig;

%-----
```

Przykład 5.4

```

clear all
close all
clc

% ROZWIĄZANIE UKŁADU RÓWNAŃ ZA POMOCĄ DWUWYMIAROWEJ METODY NEWTONA
%-----

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenie rysunku

% Obliczenia
%-----
k=1;jg=1;Rl=10; % parametry równania
dokladosc=1e-15; % wskaźnik pożądanej dokładności rozwiązania
xinf=[2.3876725061573505;0.761232749384265]; % rozwiązanie dokładne
x=[5;0]; % punkt startowy
delta_i=1;

iter=0; % początek pętli iteracyjnej (dwuwymiarowa metoda Newtona)
```

```

while delta_i>dokladnosc
    iter=iter+1;
    f=u53(x); % odwołanie do zdefiniowanej niżej funkcji „u53”
    fprim=fprim53(x); % odwołanie do zdefiniowanej niżej funkcji „fprim53”
    d_x=fprim\(-f); % rozwiązywanie układu równań 5.23
    x=x+d_x;
    delta_i(iter)=sqrt((x(1)-xinf(1))^2+(x(2)-xinf(2))^2);
end

% Prezentacja wyników
%=====

delta_i(find(delta_i<eps))=eps;
fprintf('nRozwiązań dwuwymiarowa metoda Newtona:\n')
fprintf('Ud=%6.4f id=%6.4f\n',x(1),x(2))
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
semilogy(delta_i,'*k')
title('Błąd rozwiązań dwuwymiarowa metoda Newtona');
h=xlabel('iti');
set(h,'FontName','Times','FontSize',12);
h=ylabel('||\Delta\it_i|rml|');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
axis([0,10,1e-16,1e1]);
%=====

function y=u53(x)
% Wyznaczanie wartości dwóch funkcji definiujących układ równań
%=====
k=1;ig=1;R1=10; % parametry układu
ud=x(1); id=x(2); % zmiana nazw zmiennych wejściowych ułatwiająca interpretację
y(1,1)=id-k*ud*(ud*ud/3-3*ud+2);
y(2,1)=ud/R1+id-ig;

%=====

function y=fprim53(x)
% Obliczanie wartości pochodnych cząstkowych dwóch funkcji,
% definiujących układ równań (5.3), względem zmiennych Ud i id
%=====
k=1;jg=1;R1=10;
ud=x(1); id=x(2); % zmiana nazw zmiennych wejściowych ułatwiająca interpretację
y(1,1)=-k*(ud*ud-3*ud+2); % pochodna pierwszego równania względem Ud
y(1,2)=1; % pochodna pierwszego równania względem id
y(2,1)=1/R1; % pochodna drugiego równania względem Ud
y(2,2)=1; % pochodna drugiego równania względem id
%=====

```

Zadanie 5.2

```

clear all
close all
clc
global R3 % wartość R3 przekazywana do uz52 jako zmienna globalna

```

```

% WYZNACZANIE WARTOŚCI PRĄDÓW W UKŁADZIE ELEKTRYCZNYM PRZEDSTAWIONYM NA RYS. 5.8
%=====

% WARIANT A: wyznaczanie wartości prądów dla R3=0.1
%=====
R3=0.1;
options=optimset('fsolve');
options=optimset(options,'MaxIter',100000,'MaxFunEvals',100000,'TolFun',1e-12);
i=fsolve('uz52',[4,42,0.5],options); % odwołanie do zdefiniowanej niżej funkcji
i1=i(1); % „uz52”
id=i(2)-i(1);
fprintf('nRozwiązań uzyskane dla R3=%3.1f kOhm:\n',R3)
fprintf('id=%5.2f mA il=%6.4f mA\n',i,id,i1)

% WARIANT B: wyznaczanie wartości prądów dla R3=0
%=====

R3=0;
options=optimset('fsolve');
options=optimset(options,'MaxIter',100000,'MaxFunEvals',100000,'TolFun',1e-12);
i=fsolve('uz52',[0.8,142,0.8],options);
i1=i(1);
id=i(2)-i(1);
fprintf('nRozwiązań uzyskane dla R3=%3.1f kOhm:\n',R3)
fprintf('id=%5.2f mA il=%6.4f mA\n',i,id,i1)

%=====

function u=uz52(p)
% Układ równań do zadania 5.2
%=====
global R3
E=5;
R1=0.1;
R2=1;
is=1e-12;
ut=0.025;

i1=p(1);
i2=p(2);
ud=p(3);
id=is*(exp(ud/ut)-1);
u(1)=i1-id;
u(2)=E-i1*R1-i1*R2;
u(3)=i1*R2-ud-id*R3;

```

PROGRAMY DO ROZDZIAŁU 6

Przykład 6.1

```

clear all
close all
clc

%INTERPOLACJA FUNKCJI
%-----

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków

% Interpolacja funkcji y=|sin(x)| wielomianem stopnia N=6 w przedziale [-3,3]
%-----


N=6;
xwezly=linspace(-3,3,N+1)'; % wyznaczenie węzłów interpolacji
ywezly=abs(sin(xwezly)); % wyznaczenie wartości funkcji w węzłach
xplot=linspace(-3,3,700)';
yplot=abs(sin(xplot));
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
plot(xplot,yplot)
hold on
plot(xwezly,ywezly,'*r')
wspwiel=polyfit(xwezly,ywezly,N); % wyznaczenie współczynników wielomianu
                                         % interpolacyjnego stopnia N
ywiel=polyval(wspwiel,xplot); % wyznaczenie wielomianu interpolacyjnego dla
                                 % argumentu xplot
blad6=norm(ywiel-yplot)/sqrt(700);

% Prezentacja wykresów: określenie rodzaju i wielkości czcionki użytej do opisu
%-----


legend('funkcja interpolowana','węzły interpolacji',...
       'wielomian interpolujacy, N=6','Location','South');
title(['Interpolacja wielomianem 6-go stopnia']);

fprintf(['\nBłąd średnickwadratowy interpolacji wielomianem 6-go stopnia =',...
         '%7.4f\n'],blad6);
h=xlabel('\itx');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\itf\rm(\itx\rm) ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')
axis([-3.1,3.1,-0.5,1.5]);
figure(1);
pause

```

```

% Interpolacja funkcji y=|sin(x)| wielomianem stopnia N=12 w przedziale [-3,3]
%-----


N=12;
xwezly=linspace(-3,3,N+1)';
ywezly=abs(sin(xwezly));
plot(xwezly,ywezly,'sg')
wspwiel=polyfit(xwezly,ywezly,N);
ywiel=polyval(wspwiel,xplot);
plot(xplot,ywiel,'g')
blad12=norm(ywiel-yplot)/sqrt(700);
legend('funkcja interpolowana','węzły interpolacji, N=6',...
       'wielomian interpolujacy, N=6','węzły interpolacji, N=12',...
       'wielomian interpolujacy, N=12','Location','North')
title(['Interpolacja wielomianem 6-go stopnia i 12-go stopnia']);
fprintf(['\nBłąd średnickwadratowy interpolacji wielomianem 12-go stopnia =',...
         '%7.4f\n'],blad12)

axis([-3.1,3.1,-0.5,5.5]);
grid on
zoom on
figure(1)
%-----


Przykład 6.2

clear all
close all
clc

%INTERPOLACJA FUNKCJI
%-----


% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków

% Interpolacja funkcji y=|x| wielomianem stopnia N=6 w przedziale [-3,3]
%-----


N=6;
xwezly=linspace(-3,3,N+1)'; % wyznaczenie węzłów interpolacji
ywezly=abs(xwezly); % wyznaczenie wartości funkcji w węzłach
xplot=linspace(-3,3,300)';
yplot=abs(xplot);
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
plot(xplot,yplot)
hold on
plot(xwezly,ywezly,'or')
wspwiel=polyfit(xwezly,ywezly,6); % wyznaczenie współczynników wielomianu
                                         % interpolacyjnego
ywiel=polyval(wspwiel,xplot); % wyznaczenie wielomianu interpolacyjnego dla
                                 % argumentu xplot

```

```

bladw=norm(ywiel-yplot)/sqrt(300);
fprintf(['\nBłąd średniokwadratowy interpolacji wielomianem 6-go stopnia =',...
' %7.4f\n'],bladw)

% Prezentacja wykresów: określenie rodzaju i wielkości czcionki użytej do opisu
%-----

legend('funkcja interpolowana','węzły interpolacji',...
'wielomian interpolacyjny','Location','North');
title(['Interpolacja wielomianem 6-go stopnia']);
h=xlabel('\itx');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\itf\rm(\itx\rm) ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
grid on
zoom on
fprintf('Aby kontynuować, naciśnij klawisz\n')
axis([-3.1,3.1,-0.05,3.1])
pause

% Interpolacja funkcji y=|x| funkcją sklejaną w przedziale [-3,3]
%-----



yspline=spline(xwezly,ywezly,xplot); % wyznaczenie wartości funkcji sklejanej
% dla wektora argumentu xplot
plot(xplot,yspline,'g')
blads=norm(yspline-yplot)/sqrt(300);
fprintf('Błąd średniokwadratowy interpolacji funkcją sklejaną =%7.4f\n',blads)
title('Interpolacja wielomianem 6-go stopnia i funkcją sklejana');
legend('funkcja interpolowana','węzły interpolacji, N=6',...
'wielomian interpolacyjny, N=6','funkcja sklejana','Location','North')
grid on
zoom on
figure(1)
%-----


Przykład 6.3

clear all
close all
clc

% INTERPOLACJA I APROKSIMACJA TRYGONOMETRYCZNA
%-----



% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
        figx+15 5 figx figy
        5 100+figy figx figy
        figx+15 100+figy figx figy]; % położenia rysunków

% Interpolacja trygonometryczna
% przebiegu trójkątnego o amplitudzie 1 i okresie 2pi
%-----



N=64;
tmax=2*pi;
t=linspace(0,tmax,N+1)';
t1=find(t<=pi/2);
u1=2*pi*t(t1);
t2=find(t>pi/2 & t<=3*pi/2);
u2=-2*pi*t(t2)+2;
t3=find(t>3*pi/2);
u3=2*pi*t(t3)-4;
u=[u1;u2;u3];
f=figure(3);
set(f,'Pos',figpos(3,:));
hold off
plot(t,u)
h=xlabel('\itt');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\itu');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title('Napięcie \itu\rm(\itt\rm)')
hold on

cm=fft(u(1:N)); % wyznaczenie współczynników transformaty Fouriera wg wz. (6-24)

% Prezentacja wykresów: określenie rodzaju i wielkości czcionki użytej do opisu
%-----



f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
f=semilogy((2:2:N),abs(cm(2:2:N)),'ok','MarkerFaceColor','k','MarkerSize',4);
h=xlabel('\itm');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\mid\itc_m\rm\mid');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
grid on
zoom on
g=get(f);
set(g.Parent,'MinorGridLineStyle','none');
title(['Moduły współczynników \itc_m\rm rozwinienia',...
'\itu\rm(\itt\rm) w szeregu Fouriera'])
fprintf('Aby kontynuować, naciśnij klawisz\n')
figure(1)
pause

% Aproksymacja napięcia przy użyciu transformacji odwrotnej FFT
%-----



cmo=[cm(1:4);zeros(57,1);cm(62:64)]; % wyzerowanie wszystkich składowych poza
% dwiema pierwszymi harmonicznymi
uest=real(ifft(cmo));
figure(3)
plot(t,[uest;uest(1)],'r')
legend('przebieg trójkątny','aproksymacja trygonometryczna',1)
h=xlabel('\itt');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\itu');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title('Napięcie \itu\rm(\itt\rm) oraz jego aproksymacja trygonometryczna')
grid on
fprintf('Aby kontynuować, naciśnij klawisz\n')

```

```

figure(3)
pause

N=8;
tmax=2*pi;
t=linspace(0,tmax,N+1)';
t1=find(t<=pi/2);
u1=2/pi*t(t1);
t2=find(t>pi/2 & t<=3*pi/2);
u2=-2/pi*t(t2)+2;
t3=find(t>3*pi/2);
u3=2/pi*t(t3)-4;
u=[u1;u2;u3];
figure(2)
hold off
plot(t,u,'k',t(1:N),u(1:N),'ok')
hold on

% Wyznaczenie transformaty Fouriera dla N=8
% i interpolacja trygonometryczna oparta na 64 punktach
%-----

cm8=fft(u(1:N));
cm8r=[cm8(1:4);zeros(56,1);cm8(5:8)]*8; % uzupełnienie zerami do 64 punktów
f=ifft(cm8r);
M=64;
t=linspace(0,tmax,M+1);
plot(t,[f;f(1)],'g');
f=figure(2);
set(f,'Pos',figpos(2,:));
legend('przebieg trójkątny','wzły interpolacji',...
    'interpolacja trygonometryczna',1)
h=xlabel('\it{t}');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\it{u} ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title('Napięcie \it{u}\rm(\it{t}\rm\rm) oraz jego interpolacja trygonometryczna')
grid on
zoom on
figure(2)

=====

```

Przykład 6.4

```

clear all
close all
clc

% WYZNACZANIE WARTOŚCI WIELOMIANÓW ORTOGONALNYCH
%-----

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
f=figure(1);
set(f,'Pos',figpos(1,:));

```

```

% Wyznaczanie wartości wielomianów Legendre'a w przedziale [-1,1] dla K=0,1,...,9
%-----
x=linspace(-1,1,1000)';
for K=0:9
    L=legendre(x,K); % odwołanie do zdefiniowanej niżej funkcji „legendre”
    figure(1);
    hold off
    plot(x,L)
    title(['Wielomian Legendre''a stopnia ',num2str(K)])
    h=xlabel('\it{x}');
    set(h,'FontName','Times','FontSize',12);
    h=ylabel(['\it{P}\rm_{\it{K}}\rm(\it{x})']);
    set(h,'Rotation',0,'FontName','Times','FontSize',12);
    grid on
    fprintf('\nAby kontynuować, naciśnij klawisz\n')
    pause
end

% Wyznaczanie wartości wielomianów Hermite'a w przedziale [-10,10] dla K=0,1,...,9
%-----
x=linspace(-10,10,1000)';
for K=0:9
    H=hermite(x,K); % odwołanie do zdefiniowanej niżej funkcji „hermite”
    figure(1);
    hold off
    plot(x,H)
    title(['Wielomian Hermite''a stopnia ',num2str(K)])
    h=xlabel('\it{x}');
    set(h,'FontName','Times','FontSize',12);
    h=ylabel(['\it{H}\rm_{\it{K}}\rm(\it{x})']);
    set(h,'Rotation',0,'FontName','Times','FontSize',12);
    grid on
    fprintf('\nAby kontynuować, naciśnij klawisz\n')
    pause
end

% Wyznaczanie wartości wielomianów Czebyszewa w przedziale [-1,1] dla K=0,1,...,9
%-----
x=linspace(-1,1,1000)';
for K=0:9
    T=czebyszew(x,K); % odwołanie do zdefiniowanej niżej funkcji „czebyszew”
    figure(1);
    hold off
    plot(x,T)
    title(['Wielomian Czebyszewa stopnia ',num2str(K)])
    h=xlabel('\it{x}');
    set(h,'FontName','Times','FontSize',12);
    h=ylabel(['\it{T}\rm_{\it{K}}\rm(\it{x})']);
    set(h,'Rotation',0,'FontName','Times','FontSize',12);
    grid on
    if (K<9)
        fprintf('\nAby kontynuować, naciśnij klawisz\n')
        pause
    end
end

=====

```

```

function P=legendre(x,K)
% Wyznaczanie wartości wielomianu Legendre'a stopnia K dla x z przedziału [-1,1]
% przy użyciu trójczłonowej formuły rekurencyjnej
%
PK2=ones(size(x)); % wielomian stopnia K-2 (początkowo zerowego stopnia)
PK1=x; % wielomian stopnia K-1 (początkowo pierwszego stopnia)
if K==0, P=PK2; return; end
if K==1, P=PK1; return; end
for k=2:K
    P=(2*k-1)/k*x.*PK1-(k-1)/k*PK2;
    PK2=PK1;
    PK1=P;
end

=====
function H=hermite(x,K)
% Wyznaczanie wartości wielomianu Hermite'a stopnia K dla x z przedziału [-1,1]
% przy użyciu trójczłonowej formuły rekurencyjnej
%
HK2=ones(size(x)); % wielomian stopnia K-2 (początkowo zerowego stopnia)
HK1=2*x; % wielomian stopnia K-1 (początkowo pierwszego stopnia)
if K==0, H=HK2; return; end
if K==1, H=HK1; return; end
for k=2:K
    H=2*x.*HK1-(2*k-2)*HK2;
    HK2=HK1;
    HK1=H;
end

=====
function T=czebyszew(x,K)
% Wyznaczanie wartości wielomianu Czebyszewa stopnia K dla x z przedziału [-1,1]
% przy użyciu trójczłonowej formuły rekurencyjnej
%
TK2=ones(size(x)); % to wielomian stopnia K-2 (początkowo zerowego stopnia)
TK1=x; % wielomian stopnia K-1 (początkowo pierwszego stopnia)
if K==0, T=TK2; return; end
if K==1, T=TK1; return; end
for k=2:K
    T=2*x.*TK1-TK2;
    TK2=TK1;
    TK1=T;
end

```

Przykład 6.6

```

clear all
close all
clc

% APROKSYMACJA TRYGONOMETRYCZNA FUNKCJI
%
% Parametry rysunków
%
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków

```

```

figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
figaxis=[-0.1,2*pi+0.1,-0.1,pi+0.1];
f=figure(1);
hold off
set(f,'Pos',figpos(1,:));

N=64;
xmax=2*pi;
x=linspace(0,xmax,N+1);
u=pi*triang(N-1); % definiowanie przebiegu trójkątnego za pomocą f-cji „triang”
plot(x,[0;u;0],'k')
hold on

% Wyznaczanie współczynników transformaty Fouriera przebiegu okresowego; K=4
%
K=4;
a0=pi/2; % współczynnik a0 wyznaczony wg wzoru (6-44)
k=1:K;
a=2/pi./k./k.*((-1).^k-1); % współczynniki ak wyznaczone wg wzoru (6-45)
b=k*0; % współczynniki bk wyznaczone wg wzoru (6-46)
NP=101;
x=linspace(0,2*pi,NP); % NP wartości argumentu aproksymowanej funkcji dla K=4
for n=1:NP
    f(n)=a0+sum(a.*cos(k*x(n))+b.*sin(k*x(n)));
end
figure(1)
plot(x,f,'-b')
h=xlabel('\itx');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\itf\rm(\itx\rm)');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
legend('funkcja f(x)', 'aproksymacja dla K=4', 'Location', 'South')
title('Aproksymacja trygonometryczna funkcji niegładkiej')
axis(figaxis(1,:));
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')

%
% Wyznaczenie współczynników transformaty Fouriera przebiegu okresowego; K=64
%
K=64;
a0=pi/2;
k=1:K;
a=2/pi./k./k.*((-1).^k-1);
b=k*0;
NP=101;
x=linspace(0,2*pi,NP);
for n=1:NP
    f(n)=a0+sum(a.*cos(k*x(n))+b.*sin(k*x(n)));
end
figure(1)
plot(x,f,'-r')
legend('funkcja f(x)', 'aproksymacja dla K=4', 'aproksymacja dla K=64',...
    'Location', 'South')
grid on
zoom on

```

Przykład 6.7

```

clear all
close all
clc

% APROKSYMACJA FUNKCJI NA PODSTAWIE CIĄGU JEJ DYSKRETNYCH WARTOŚCI
%=====

ug=linspace(0,4,200); % definicja punktów w których rysowany
% będzie wynik aproksymacji
u=(0:0.5:4)'; % definicja punktów pomiarowych
i=[1.017;0.8788;0.4925;0.4883;0.3281;0.3452;0.1613;0.2575;0.1747];

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
f=figure(1);
hold off
set(f,'Pos',figpos(1,:));
plot(u,i,'ok','MarkerFaceColor','k','MarkerSize',4);
h=xlabel('\it{u}\rm{[V]}' );
set(h,'FontName','Times','FontSize',12);
h=ylabel('\it{i}\rm{[mA]}');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
legend ('punkty pomiarowe')
title('Wielomianowa aproksymacja charakterystyki prądowo-napięciowej')
hold on
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')
figure(1)
pause

% Aproksymacja wielomianem 3-go stopnia
%-----
w3=polyfit(u,i,3);
i3=polyval(w3,ug);
h=plot(ug,i3,'k');
set(h,'LineWidth',2);
legend ('punkty pomiarowe','aproksymacja wielomianem 3-go stopnia')
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')
figure(1)
pause

% Interpolacja wielomianem 8-go stopnia
%-----
w8=polyfit(u,i,8);
i8=polyval(w8,ug);
h=plot(ug,i8,'g');
set(h,'LineWidth',2);
legend ('punkty pomiarowe','aproksymacja wielomianem 3-go stopnia',...
'interpolacja wielomianem 8-go stopnia')
grid on
zoom on
figure(1)
%=====

```

Przykład 6.8

```

clear all
close all
clc

% APROKSYMACJA FUNKCJI NA PODSTAWIE CIĄGU JEJ DYSKRETNYCH WARTOŚCI
%=====

x=linspace(-2,3,11)'; % wektor wartości argumentu aproksymowanej funkcji,
% używanych do aproksymacji
N=101;
xn=linspace(-2,3,N)'; % wektor wartości argumentu aproksymowanej funkcji,
% używanych do wyznaczania błędu aproksymacji
f1=@(x) exp(x).*sin(x); % definicja funkcji f1 jako funkcji anonimowej
f2=@(x) abs(x-1)+1; % definicja funkcji f2 jako funkcji anonimowej

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
        figx+15 5 figx figy
        5 100+figy figx figy
        figx+15 100+figy figx figy]; % położenia rysunków
f=figure(1);
hold off
set(f,'Position',figpos(1,:));
plot(x,f1(x),'k');
hold on
plot(xn,f1(xn),'k')
h=xlabel('\it{x}\rm{'} );
set(h,'FontName','Times','FontSize',12);
l=legend ('węzły aproksymacji','funkcja aproksymowana',2);
set(l,'FontSize',8);
title('Aproksymacja i interpolacja \it{f_1}\rm{(\it{x}\rm{)}}')
grid on
zoom on
f=figure(2);
hold off
set(f,'Position',figpos(2,:));
plot(x,f2(x),'ok');
hold on
plot(xn,f2(xn),'k');
h=xlabel('\it{x}\rm{'} );
set(h,'FontName','Times','FontSize',12);
l=legend ('węzły aproksymacji','funkcja aproksymowana','Location','North');
set(l,'FontSize',8);
title('Aproksymacja i interpolacja \it{f_2}\rm{(\it{x}\rm{)}}')
hold on
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

% Wyznaczanie wielomianów aproksymujących
% stopnia 4,6,8 i 10 oraz błędów aproksymacji
%=====

```

```

for K=4:2:10
wspwiel1=polyfit(x,f1(x),K);
wiel1=polyval(wspwiel1,xn);
roznical=abs(f1(xn)-wiel1);
Delta1(K)=sqrt(1/N*roznical'*roznical);
if (K==4)
    figure(1)
    plot(xn,wiel1,'r');
    l=legend ('węzły aproksymacji','funkcja aproksymowana',...
        'wielomian aproksymujacy 4-go stopnia',2);
    set(l,'FontSize',8);
end
if (K==10)
    figure(1)
    plot(xn,wiel1,'-k');
    l=legend ('węzły aproksymacji','funkcja aproksymowana',...
        'wielomian aproksymujacy 4-go stopnia',...
        'wielomian aproksymujacy 10-go stopnia',2);
    set(l,'FontSize',8);
end

wspwiel2=polyfit(x,f2(x),K);
wiel2=polyval(wspwiel2,xn);
roznica2=abs(f2(xn)-wiel2);
Delta2(K)=sqrt(1/N*roznica2'*roznica2);
if (K==4)
    figure(2)
    plot(xn,wiel2,'r');
    l=legend ('węzły aproksymacji','funkcja aproksymowana',...
        'wielomian aproksymujacy 4-go stopnia','Location','North');
    set(l,'FontSize',8);
    fprintf('\nAby kontynuować, naciśnij klawisz\n')
    pause
end
if (K==10)
    figure(2)
    plot(xn,wiel2,'-k');
    l=legend ('węzły aproksymacji','funkcja aproksymowana',...
        'wielomian aproksymujacy 4-go stopnia',...
        'wielomian aproksymujacy 10-go stopnia','Location','North');
    set(l,'FontSize',8);
    fprintf('\nAby kontynuować, naciśnij klawisz\n')
    pause
end

% Interpolacja funkcją sklejaną i wyznaczenie błędu interpolacji
%-----
wiel1=spline(x,f1(x),xn);
roznicalS=abs(f1(xn)-wiel1);
Delta1S=sqrt(1/N*roznicalS'*roznicalS);
figure(1)
plot(xn,wiel1,'--g');
l=legend ('węzły aproksymacji','funkcja aproksymowana',...
    'wielomian aproksymujacy 4-go stopnia',...
    'wielomian aproksymujacy 10-go stopnia',...
    'interpolująca funkcja sklejana',2);
set(l,'FontSize',8);
wiel2=spline(x,f2(x),xn);

```

```

roznica2S=abs(f2(xn)-wiel2);
Delta2S=sqrt(1/N*roznica2S'*roznica2S);
figure(2)
plot(xn,wiel2,'--g');
l=legend ('węzły aproksymacji','funkcja aproksymowana',...
    'wielomian aproksymujacy 4-go stopnia',...
    'wielomian aproksymujacy 10-go stopnia',...
    'interpolująca funkcja sklejana','Location','North');
set(l,'FontSize',8);
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause
% Sporządzanie wykresu błędów
%-----
f=figure(3);
set(f,'Position',figpos(3,:));
hold off
K=(4:2:10);
semilogy(K,Delta1(K),'o--k','MarkerSize',4);
hold on
semilogy(K,Delta2(K),'o-k','MarkerFaceColor','k','MarkerSize',4);
semilogy(10,Delta1S,'or','MarkerSize',6)
f=semilogy(10,Delta2S,'or','MarkerFaceColor','r','MarkerSize',6);
grid on
g=get(f);
set(g.Parent,'MinorGridLineStyle','none');
h1=label('\it{K}');
set(h1,'FontName','Times','FontSize',12);
h2=label('\it{\Delta}\rm_2');
set(h2,'Rotation',0,'FontName','Times','FontSize',12);
title('Błędy średniokwadratowe aproksymacji i interpolacji');
legend('aproksymacja f1 wielomianem','aproksymacja f2 wielomianem',...
    'interpolacja f1 funkcją sklejana','interpolacja f2 funkcją sklejana',3);
%-----

```

Przykład 6.9

```

clear all
close all
clc
% APROKSYMACJA FUNKCJAMI NIELINIOWYMI WZGLEDEM PARAMETRÓW
%-----
% Wyznaczenie teoretycznej wartości impedancji obwodu rezonansowego
%-----
f0=1e9;
Q=5000;
z0=1;
fn=linspace(0.9995e9,1.0005e9,100)'; % wektor wartości argumentu aproksymowanej
                                            % funkcji
zn=modulimpedancji([f0;Q;z0],fn); % odwołanie do zdefiniowanej niżej funkcji
                                            % „modulimpedancji”

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków

```

```

figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
f=figure(1);
hold off
set(f,'Position',figpos(1,:));
plot(fn,zn,'k');
hold on

% Symulacja niepewnych wyników pomiaru
%-----
zn=zn.*(1+(rand(size(zn))*2-1)*0.045;
plot(fn,zn,'ok','MarkerFaceColor','k','MarkerSize',4);
h=xlabel('\itf \rm [Hz]');
set(h,'FontName','Times','FontSize',12);
h=ylabel('|\itx\rm(\itf\rm) |');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
legend ('charakterystyka idealna','punkty pomiarowe','Location','South');
title('Wyznaczanie optymalnych parametrów krzywej rezonansowej')
grid on
zoom on

% Minimalizacja funkcjonalu (6.61)
% przy użyciu standardowej procedury „fminsearch”
%-----
p0=[f0;Q;z0];
options=optimset('fminsearch');
p=fminsearch(@modulimpedancji,[p0].options,fn,zn);
fprintf('\nUzyskane rozwiązanie zmienia się z losowym zaburzeniem danych\n');
fprintf(['fr=%15.8e\nQ =%7.2f\nz =%7.5f\n'],p(1),p(2),p(3));
%=====

function z=modulimpedancji(p,fn,zn)
% Wyznaczanie modułu impedancji z(f) mikrofalowego obwodu rezonansowego
% lub funkcjonalu J2 (6.61) w zależności od liczby parametrów wejściowych
f0=p(1);
Q =p(2);
z0=p(3);
z=z0.*((1+Q*Q*(fn/f0-f0./fn).^2).^(0.5));
if nargin==3
    J2=(z-zn).*(z-zn);
    z=J2;
end
%=====

```

Zadanie 6.1

```

clear all
close all
clc

% APROKSYMACJA TRYGONOMETRYCZNA PRZEBIEGU PROSTOKĄTNEGO
%=====

% Definicja przebiegu prostokątnego o amplitudzie 1 i okresie 2pi
%-----
N=1000;
xmax=2*pi;

```

```

x=linspace(0,xmax,N+1)';
u=sign(sin(x));

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
f=figure(1);
hold off
set(f,'Position',figpos(1,:));
plot(x,u,'k')
grid on; zoom on; hold on

% Wyznaczanie parametrów wg wzorów (6.44) - (6.46)
% Wyznaczenie aproksymacji trygonometrycznej wg wzoru (6.43)
%-----
K=8;
a0=0;
k=(1:K)';
a=0;
b=2/pi./k.*(1-(-1).^k);
NP=1001;
x=linspace(0,2*pi,NP)';
for n=1:NP
    u(n,1)=a0+sum(a.*cos(k*x(n))+b.*sin(k*x(n)));
end
figure(1)
plot(x,u,:b')
h=xlabel('\itx');
set(h,'FontName','Times','FontSize',12);
h=ylabel('|\itu|');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title('Aproksymacja trygonometryczna')
l=legend('funkcja u(x)', 'aproksymacja u(x) dla K=8');
set(l,'FontSize',8);
grid on
zoom on
fprintf('\nKontynuacja - naciśnij klawisz\n')
pause

% Wyznaczanie parametrów wg wzorów (6.44) - (6.46)
% Wyznaczenie aproksymacji trygonometrycznej wg wzoru (6.43)
%-----
K=256;
a0=0;
k=(1:K)';
a=0;
b=2/pi./k.*(1-(-1).^k);
NP=1001;
x=linspace(0,2*pi,NP)';
for n=1:NP
    u(n)=a0+sum(a.*cos(k*x(n))+b.*sin(k*x(n)));
end
figure(1)
plot(x,u,'r')
l=legend('funkcja u(x)', 'aproksymacja u(x) dla K=8',...
    'aproksymacja u(x) dla K=256');
set(l,'FontSize',8);
grid on
zoom on

```

PROGRAMY DO ROZDZIAŁU 7

Przykład 7.2

```

clear all
close all
clc
kolory='ymgbrk';

% CAŁKOWANIE FUNKCJI ZA POMOCĄ PROSTYCH KWADRATUR NEWTONA-COTESA (KNC)
%=====

% Definicje całkowanych funkcji (f1, f2, f3 i f4) jako funkcji anonimowych
%-----
f1=@(x) x.*exp(x);
f2=@(x) (1-x.*x).^(0.5);
f3=@(x) exp(-abs(x));
f4=@(x) 1./(1+25*x.*x);

% Dokładne wartości całek funkcji f1, f2, f3 i f4, wyznaczone analitycznie
%-----
I1=2/exp(1);
I2=pi/2;
I3=2-2/exp(1);
I4=0.4*atan(5);
a=-1;
b=1;
x=linspace(a,b,1000);

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
        figx+15 5 figx figy
        5 100+figy figx figy
        figx+15 100+figy figx figy]; % położenia rysunków

% Prezentacja wykresów funkcji f1, f2, f3 i f4
%-----
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
plot(x,f1(x));
h=xlabel('itx');
set(h,'FontName','Times','FontSize',12);
h ylabel('itf_1\rm(itx\rm)');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Funkcja \itf_1\rm(itx\rm)']);
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

f=figure(2);
set(f,'Pos',figpos(2,:));
hold off

```

```

plot(x,f2(x));
h=xlabel('itx');
set(h,'FontName','Times','FontSize',12);
h ylabel('itf_2\rm(itx\rm)');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Funkcja \itf_2\rm(itx\rm)']);
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

f=figure(3);
hold off
set(f,'Pos',figpos(3,:));
plot(x,f3(x));
h=xlabel('itx');
set(h,'FontName','Times','FontSize',12);
h ylabel('itf_3\rm(itx\rm)');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Funkcja \itf_3\rm(itx\rm)']);
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

f=figure(4);
set(f,'Pos',figpos(4,:));
hold off
plot(x,f4(x));
h=xlabel('itx');
set(h,'FontName','Times','FontSize',12);
h ylabel('itf_4\rm(itx\rm)');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Funkcja \itf_4\rm(itx\rm)']);
grid on
zoom on
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

% Definicja macierzy współczynników prostej KNC
%-----
ln=[1,1,0,0,0,0;
     1,4,1,0,0,0,0;
     1,3,3,1,0,0,0;
     7,32,12,32,7,0,0;
     19,75,50,50,75,19,0;
     41,216,27,272,27,216,41];

% Wyznaczenie wartości prostej KNC dla funkcji f1.
% Ilustracja procesu obliczeniowego, polegajaca na prezentacji
% wielomianu interpolacyjnego, na podstawie którego jest ona wyznaczana
%-----
for N=1:6
    wezly=linspace(a,b,N+1); % wyznaczenie wezłów kwadratury
    yl=f1(wezly);
    figure(1)
    hold off
    plot(x,f1(x))
    hold on
    plot(wezly,yl,['o' kolory(N)],'MarkerFaceColor',kolory(N),'MarkerSize',8)

```

```

wsp=polyfit(wezly,y1,N); % wyznaczenie wielomianu interpolacyjnego
w1=polyval(wsp,x);
h=plot(x,w1,[': kolor(N)]);
set(h,'LineWidth',3);
h=xlabel('\itx');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\itf_1\rm(\itx\rm) ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Funkcja \itf_1\rm(\itx\rm) i jej interpolacja wielomianem',...
'Lagrange''a ',num2str(N),'-go stopnia'])
grid on
zoom on

mn=sum(ln(N,:)); % mnożnik prostej KNC wg Tablicy 7.1
KW1(N)=(ln(N,1:N+1)*f1(wezly))*(b-a)/mn; % wartość prostej KNC wg wzoru (7.6)
deltaI(1,N)=(KW1(N)-I1)/I1*100; % względny błąd prostej KNC w %
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause
end

% Wyznaczenie wartości prostych KNC dla funkcji f2, f3 i f4.
% Ilustracja procesu obliczeniowego, polegająca na prezentacji
% wielomianów interpolacyjnych, na podstawie których są one wyznaczane.
%-----

for N=1:6
    wezly=linspace(a,b,N+1);
    y2=f2(wezly);
    figure(2)
    hold off
    plot(x,f2(x));
    hold on
    plot(wezly,y2,['o' kolor(N)],'MarkerFaceColor',kolor(N),'MarkerSize',8)
    wsp=polyfit(wezly,y2,N);
    w2=polyval(wsp,x);
    h=plot(x,w2,[': kolor(N)]);
    set(h,'LineWidth',3);
    h=xlabel('\itx');
    set(h,'FontName','Times','FontSize',12);
    h=ylabel('\itf_2\rm(\itx\rm) ');
    set(h,'Rotation',0,'FontName','Times','FontSize',12);
    title(['Funkcja \itf_2\rm(\itx\rm) i jej interpolacja wielomianem',...
'Lagrange''a ',num2str(N),'-go stopnia'])
    mn=sum(ln(N,:));
    KW2(N)=(ln(N,1:N+1).*f2(wezly))*(b-a)/mn;
    deltaI(2,N)=(KW2(N)-I2)/I2*100;
    fprintf('\nAby kontynuować, naciśnij klawisz\n')
    pause
end

for N=1:6
    wezly=linspace(a,b,N+1);
    y3=f3(wezly);
    figure(3)
    hold off
    plot(x,f3(x));
    hold on
    plot(wezly,y3,['o' kolor(N)],'MarkerFaceColor',kolor(N),'MarkerSize',8)
    wsp=polyfit(wezly,y3,N);
    w3=polyval(wsp,x);
    h=plot(x,w3,[': kolor(N)]);

```

```

    set(h,'LineWidth',3);
    h=xlabel('\itx');
    set(h,'FontName','Times','FontSize',12);
    h=ylabel('\itf_3\rm(\itx\rm) ');
    set(h,'Rotation',0,'FontName','Times','FontSize',12);
    title(['Funkcja \itf_3\rm(\itx\rm) i jej interpolacja wielomianem',...
'Lagrange''a ',num2str(N),'-go stopnia'])
    mn=sum(ln(N,:));
    KW3(N)=sum(ln(N,1:N+1).*f3(wezly))*(b-a)/mn;
    deltaI(3,N)=(KW3(N)-I3)/I3*100;
    fprintf('\nAby kontynuować, naciśnij klawisz\n')
    pause
end

for N=1:6
    wezly=linspace(a,b,N+1);
    y4=f4(wezly);
    figure(4)
    hold off
    plot(x,f4(x));
    hold on
    plot(wezly,y4,['o' kolor(N)],'MarkerFaceColor',kolor(N),'MarkerSize',8)
    wsp=polyfit(wezly,y4,N);
    w4=polyval(wsp,x);
    h=plot(x,w4,[': kolor(N)]);
    set(h,'LineWidth',3);
    h=xlabel('\itx');
    set(h,'FontName','Times','FontSize',12);
    h=ylabel('\itf_4\rm(\itx\rm) ');
    set(h,'Rotation',0,'FontName','Times','FontSize',12);
    title(['Funkcja \itf_4\rm(\itx\rm) i jej interpolacja wielomianem',...
'Lagrange''a ',num2str(N),'-go stopnia'])
    mn=sum(ln(N,:));
    KW4(N)=sum(ln(N,1:N+1).*f4(wezly))*(b-a)/mn;
    deltaI(4,N)=(KW4(N)-I4)/I4*100;
    fprintf('\nAby kontynuować, naciśnij klawisz\n')
    pause
end

% Drukowanie wyników przedstawionych w Tablicy 7.2
fprintf(['\nBłędy względne prostych kwadratur Newtona-Cotesa dla funkcji',...
'testowych\n'])
fprintf(['      N=1      N=2      N=3      N=4      ','...'])
fprintf(['      'N=5      N=6\n'])
fprintf('f1 $11.3e $11.3e $11.3e $11.3e $11.3e\n',deltaI(1,1:6));
fprintf('f2 $11.3e $11.3e $11.3e $11.3e $11.3e\n',deltaI(2,1:6));
fprintf('f3 $11.3e $11.3e $11.3e $11.3e $11.3e\n',deltaI(3,1:6));
fprintf('f4 $11.3e $11.3e $11.3e $11.3e $11.3e\n',deltaI(4,1:6));
%=====

Przykład 7.5

clear all
close all
clc
kolor='ymgbkr';

% CAŁKOWANIE FUNKCJI ZA POMOCĄ ZEZOŻONYCH KWADRATUR NEWTONA-COTESA (KNC)
%=====
```

```

% Definicje całkowanych funkcji (f1, f2, f3 i f4) jako funkcji anonimowych
% ----
f1=@(x) x.*exp(x);
f2=@(x) (1-x.*x).^0.5;
f3=@(x) exp(-abs(x));
f4=@(x) 1./(1+25*x.*x);

% Dokładne wartości całek funkcji f1, f2, f3 i f4, wyznaczone analitycznie
% ----

I1=2/exp(1);
I2=pi/2;
I3=2-2/exp(1);
I4=0.4*atan(5);

a=-1;
b=1;
x=linspace(a,b,1000);

% Definicja macierzy współczynników prostych KNC
% ----
ln=[1,1,0,0,0,0;
    1,4,1,0,0,0;
    1,3,3,1,0,0;
    7,32,12,32,7,0;
    19,75,50,50,75,19;
    41,216,27,272,27,216,41];

% Parametry rysunków
% ----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figy figx figy
        figx+15 100+figy figx figy
        5 5 figx figy
        figx+15 figx figy]; % położenia rysunków
figaxis=[1 100 1e-15 1;1 100 3e-4 1;1 100 1e-15 1;1 100 2e-8 1];
for i=1:4 % realizacja obliczeń kolejno dla f1, f2, f3 i f4
    f=figure(i);
    set(f,'Position',figpos(i,:));
    hold off
    for N=1:6
        ind=0;
        mn=sum(ln(N,:));
        M=1; % inicjacja obliczeń przy użyciu prostej KNC
        st=(b-a)/M;
        while N*M+1<=100 % graniczna liczba węzłów = 100
            ind=ind+1; % indeks kolejnej kwadratury, niezbędny do zapisywania wyników
            ab=linspace(a,b,M+1);
            Nw(ind)=N*M+1;
            eval(['K' N,'num2str(i)', '(N,ind)=0;']);
            for pp=1:M % wyznaczanie prostych KNC dla kolejnych podprzedziałów
                wezly=linspace(ab(pp),ab(pp+1),N+1);
                eval(['K' N,'num2str(i)', '(N,ind)=K' N,'num2str(i)', '(N,ind)+sum(ln(N,1x',...
                    'N+1).*f', 'num2str(i)', '(wezly)*(ab(pp+1)-ab(pp))/mn;');
                % sumowanie prostych KNC
            end
        end
    end

```

```

eval(['bladKW','num2str(i)', '(N,ind)=abs(I', 'num2str(i)', '-KW', 'num2str(i),...
       '(N,ind))/I', 'num2str(i)', ']); % wyznaczanie błędów kwadratury
M=M*2; % powiększanie liczby podprzedziałów
end
eval(['h=loglog(Nw,bladKW','num2str(i)', '(N,:),[''-'' kolory(N)],']);
set(h,'LineWidth',2);
hold on
end
legend('N=1','N=2','N=3','N=4','N=5','N=6','Location','SouthWest')
axis(figaxis(i,:));
h xlabel('itN_w');
set(h,'FontName','Times','FontSize',12);
eval(['h=ylabel(''\mid\delta[\itI]\rm(\itf_,', 'num2str(i), '\rm)\mid'')']);
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Zależność błędu względnego od liczby węzłów'])
pause(0.1)
end
% ----

Przykład 7.6

clear all
close all
clc

% CAŁKOWANIE FUNKCJI ZA POMOCĄ ADAPTACYJNEJ ZŁOŻONEJ KWADRATURY SIMPSONA (KS)
% ----

% Definicje całkowanych funkcji (f1, f2, f3 i f4) jako funkcji anonimowych
% ----
f1=@(x) x.*exp(x);
f2=@(x) (1-x.*x).^0.5;
f3=@(x) exp(-abs(x));
f4=@(x) 1./(1+25*x.*x);

% Dokładne wartości całek funkcji f1, f2, f3 i f4, wyznaczone analitycznie
% ----
I1=2/exp(1);
I2=pi/2;
I3=2-2/exp(1);
I4=0.4*atan(5);

% Wyznaczanie wartości całek za pomocą złożonej KS i prezentacja wyników
% ----
a=-1;
b=1;
x=linspace(a,b,1000);

% Parametry rysunków
% ----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figy figx figy
        figx+15 100+figy figx figy
        5 5 figx figy
        figx+15 figx figy]; % położenia rysunków
figaxis=[-1 1 -1 3;-1 1 0 1;-1 1 0.3 1.1;-1 1 0 1];

```

```

for i=1:4
    f=figure(i);
    set(f,'Position',figpos(i,:));
    hold off
    eval(['[I,w]=simpson(f',num2str(i),',-1,1,1e-4);']); % odwołanie do
    % zdefiniowanej niżej funkcji "simpson"
    eval(['bladKW=(I-I',num2str(i),')/I',num2str(i),']); % błąd względny całkowania
    eval(['h=plot(w,f',num2str(i),',(w)','.b');']);
    axis(figaxis(i,:));
    h=xlabel('\itx');
    set(h,'FontName','Times','FontSize',12);
    h=title(['Rozkład \itN_w\rm ',num2str(length(w)), ' węzłów dla \itf_',...
        num2str(i)]);
    fprintf('Błąd bezwzględny całkowania funkcji f%li =%10.2e (Nw=%2i)\n',i,',...
        bladKW,length(w));
    set(h,'FontSize',10);
    pause(0.1)
end

=====
function [I,w]=simpson(f,a,b,deltaImax)
% Wyznaczanie adaptacyjnej złożonej KS dla funkcji f
% I - wartość kwadratury; w - węzły wykorzystywane przez kwadraturę;
% f - wskaźnik do całkowanej funkcji; a, b - krańce przedziału całkowania;
% deltaImax - dopuszczalny bezwzględny błąd całkowania
% ----

% Wyznaczanie wartości I1 prostej KS opartej na węzłach a, c, b
% ----

c=(a+b)/2;
I1=(b-a)/6*(f(a)+4*f(c)+f(b));

% Wyznaczanie wartości I2 złożonej KS opartej na węzłach a, d, c, e, b
% ----

d=(a+c)/2;
e=(c+b)/2;
I2=(b-a)/12*(f(a)+4*f(d)+2*f(c)+4*f(e)+f(b));
if (abs(I1-I2)<deltaImax) % jeżeli różnica między I1 i I2 jest mniejsza
    I=I2; % od założonej dokładności to kwadratura przyjmuje wartość I2
    w=[a;d;c;e;b]; % a węzłami sa [a;d;c;e;b]
else % jeżeli warunek nie jest spełniony to rekurencyjne wywołanie procedury
    [I1,w1]=simpson(f,a,c,deltaImax); % "simpson" dla dwóch podprzedziałów: a - c
    [I2,w2]=simpson(f,c,b,deltaImax); % i c - b
    I=I1+I2; % ostateczna wartość kwadratury jest sumą kwadratur dla dwóch
    w=[w1;w2(2:length(w2))]; % podprzedziałów, a węzły są sumą węzłów
end
% ----

Przykład 7.9
clear all
close all
clc
kolory='ymgbrk';

% CAŁKOWANIE FUNKCJI ZA POMOCĄ PROSTYCH I ZŁOŻONYCH
% KWADRATUR GAUSSA-LEGENDRE'A (KGL)
% -----

```

```

% Definicje całkowanych funkcji (f1, f2, f3 i f4) jako funkcji anonimowych
% -----
f1=@(x) x.*exp(x);
f2=@(x) (1-x.*x).^(0.5);
f3=@(x) exp(-abs(x));
f4=@(x) 1./(1+25*x.*x);

% Dokładne wartości całek funkcji f1, f2, f3 i f4, wyznaczone analitycznie
% -----
I1=2/exp(1);
I2=pi/2;
I3=2-2/exp(1);
I4=0.4*atan(5);

% Współczynniki wielomianów Legendre'a stopnia od 0 do 7
% -----
Lcoefs=[[0 0 0 0 0 0 0 1]
        [0 0 0 0 0 1 0]
        [0 0 0 0 3 0 -1]/2
        [0 0 0 5 0 -3 0]/2
        [0 0 35 0 -30 0 3]/8
        [0 0 63 0 -70 0 15 0]/8
        [0 231 0 -315 0 105 0 -5]/16
        [429 0 -693 0 315 0 -35 0]/16];

% Wyznaczanie prostej kwadratury KGL kolejno dla funkcji f1, f2, f3 i f4
% -----
for i=1:4
    for N=1:6
        aN=Lcoefs(N+2,7-N); % współczynnik a(N+1) we wzorze (7-17)
        aN=Lcoefs(N+1,8-N); % współczynnik a(N) we wzorze (7-17)
        normN=2/(2*N+1); % norma wielomianu Legendre'a stopnia N
        wezly=roots(Lcoefs(N+2,7-N:8)); % pierwiastki wielomianu Legendre'a
        Lprim=polyder(Lcoefs(N+2,7-N:8)); % współczynniki pochodnej
        % wielomianu Legendre'a
        An=aN1/aN*normN./polyval(Lprim,wezly)./polyval(Lcoefs(N+1,8-N:8),wezly);
        % wzór (7-17)
        eval(['I(i,N)=sum(An.*f',num2str(i),',(wezly));']); % wartość prostej KGL
        eval(['deltaI(i,N)=abs((I',num2str(i),'-I(i,N))/I',num2str(i),')*100;']);
        % względny błąd prostej KGL
        pause(0.1)
    end
end

% Drukowanie wyników przedstawionych w Tablicy 7.3
% -----
fprintf(['\nBłędy względne (%%) całkowania za pomocą kwadratur',...
    'Gaussa-Legendre''a\n']);
fprintf('      N=1      N=2      N=3      N=4      N=5      N=6\n');
fprintf('f1  %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',deltaI(1,1:6));
fprintf('f2  %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',deltaI(2,1:6));
fprintf('f3  %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',deltaI(3,1:6));
fprintf('f4  %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',deltaI(4,1:6));

a=-1;
b=1;

```

```

% Wyznaczanie złożonej kwadratury KGL kolejno dla funkcji f1, f2, f3 i f4
%-----

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figy figx figy
        figx+15 100+figy figx figy
        5 5 figx figy
        figx+15 5 figx figy]; % położenia rysunków
figaxis=[1 100 1e-15 1;1 100 1e-5 1e-1;1 100 1e-15 1;1 100 1e-15 1];

for i=1:4
    f=figure(i);
    set(f,'Position',figpos(i,:));
    hold off
    for N=1:6
        ind=0;
        aN1=Lcoefs(N+2,7-N);
        aN =Lcoefs(N+1,8-N);
        normN=2/(2*N+1);
        wezly=roots(Lcoefs(N+2,7-N:8));
        Lprim=polyder(Lcoefs(N+2,7-N:8));
        An=aN1/normN./polyval(Lprim,wezly)./polyval(Lcoefs(N+1,8-N:8),wezly);
        M=1; % inicjacja obliczeń przy użyciu prostej KGL
        while N*M+1<=100 % ograniczenie liczby węzłów do 100
            ind=ind+1; % indeks kolejnej kwadratury niezbędny do zapisywania wyników
            ab=linspace(a,b,M+1);
            Nw(ind)=N*M+1;
            eval(['Kw',num2str(i),'(N,ind)=0;']);
            for pp=1:M % wyznaczenie prostych KGL dla każdego podprzedziału
                % i - sumowanie
                wezlyl=(wezly-((ab(pp)+ab(pp+1))/(ab(pp)-ab(pp+1))))*...
                    (ab(pp+1)-ab(pp))/2;
                eval(['Kw',num2str(i),'(N,ind)=Kw',num2str(i),'(N,ind)+sum(An.*f',...
                    num2str(i),'(wezlyl)*(ab(pp+1)-ab(pp))/2;']);
            end
            eval(['bladKw',num2str(i),'(N,ind)=abs(I',num2str(i),'-Kw',num2str(i),...
                '(N,ind))/I',num2str(i),';']); % wzajemny błąd złożonej KGL
            M=M*2; % powiększanie liczby podprzedziałów
        end
        eval(['h=loglog(Nw,bladKw',num2str(i),'(N,:),[''-'' kolory(N)])']);
        set(h,'LineWidth',2);
        title('Zależność względnego błędu całkowania od liczby węzłów','FontSize',8)
        hold on
        pause(0.1)
    end
    legend('N=1','N=2','N=3','N=4','N=5','N=6','Location','SouthWest')
    axis(figaxis(i,:));
    xlabel('\itN_w');
    set(h,'FontName','Times','FontSize',12);
    eval(['h=ylabel(''\mid\delta\mid\rm{\Delta}\mid\itf_\itN_w\mid'')']);
    set(h,'Rotation',0,'FontName','Times','FontSize',12);
    pause(0.1)
end

% Wyznaczenie wartości całki I2 za pomocą kwadratury Gaussa-Czebyszewa
%-----

```

```

f2n=@(x) (1-x.*x); % modyfikacja całkowanej funkcji f2 po usunięciu funkcji wagi
wezly=[-1/sqrt(2);1/sqrt(2)];
An=[pi/2;pi/2];
Igc2=sum(An.*f2n(wezly));
fprintf(['\nWAGA:\nBłąd kwadratury Gaussa-Czebyszewa opartej na dwóch',...
        'węzłach dla funkcji f2 wynosi %10.2e\n'],abs((Igc2-I2)/I2));
%=====

Przykład 7.10
clear all
close all
cic

% CAŁKOWANIE FUNKCJI ZA POMOCĄ KWADRATUR ROMBERGA (KR)
%-----
f=@(x) sin(pi*x); % całkowana funkcja
a=0; b=1; % przedział całkowania
I=2/pi; % dokładna wartość całki

% Wyznaczanie KR (złożonej kwadratury trapezów) dla 1, ..., 128 podprzedziałów
%-----
for w=1:10
    N=2^(w-1); % liczba węzłów
    wezly=linspace(a,b,N+1);
    H=(b-a)/N;
    T(w,1)=H/2*(f(a)+2*sum(f(wezly(2:N)))+f(b));
    B(w,1)=abs((T(w)-I)/I);
end
for k=1:6 % k - krotność ekstrapolacji
    for w=1:k
        T(w,k+1)=(4^k*T(w+1,k)-T(w,k))/(4^k-1); % KR dla N węzłów
        B(w,k+1)=(T(w,k+1)-I)/I; % błąd wzajemny KR
    end
end
% Drukowanie wyników przedstawionych w Tablicy 7.4
%-----
fprintf('\nWzględne błędy kwadratur Romberga')
fprintf(['\n N      k=0      k=1      k=2      k=3      k=4      ','...'])
    'k=5      k=6\n']);
fprintf(' 1 %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',B(1,1:7))
fprintf(' 2 %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',B(2,1:7))
fprintf(' 4 %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',B(3,1:7))
fprintf(' 8 %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',B(4,1:7))
fprintf('16 %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',B(5,1:6))
fprintf('32 %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',B(6,1:5))
fprintf('64 %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',B(7,1:4))
fprintf('128 %10.2e %10.2e %10.2e %10.2e %10.2e %10.2e\n',B(8,1:3))
%=====

%%%%%%%
% ZADANIE 7.1 %
%-----

```

```

%%%%%
clear all;close all;clc

% CAŁKOWANIE FUNKCJI ZA POMOCĄ ADAPTACYJNEJ ZŁOŻONEJ KWADRATURY SIMPSONA (KS)
%=====

% Definicje całkowanych funkcji (f1, f2, f3 i f4) jako funkcji anonimowych
%-----
f1=@(x) x.*exp(x);
f2=@(x) (1-x.*x).^(0.5);
f3=@(x) exp(-abs(x));
f4=@(x) 1./(1+25*x.*x);

% Dokładne wartości całek funkcji f1, f2, f3 i f4, wyznaczone analitycznie
%-----
a=-1;b=1; % przedział całkowania
I1=2/exp(1);
I2=pi/2;
I3=2-2/exp(1);
I4=0.4*atan(5);

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
        figx+15 5 figx figy
        5 100+figy figx figy]; % położenia rysunków

figaxis=[1e-10 1e-1 1e-16 1e-1; % zakresy osi na poszczególnych rysunkach
         1e-10 1e-1 1e-16 1e-1;
         1e-10 1e-1 1e-16 1e-1;
         1e-10 1e-1 1e-16 1e-1];

% Wyznaczanie adaptacyjnych złożonych KS kolejno funkcji f1, f2, f3 i f4
%-----
x=linspace(a,b,1000);
for i=1:4
    f=figure(i);
    set(f,'Position',figpos(i,:));
    hold off
    for w=-1:-1:-10 % w - wykładnik kryterium dokładności kwadratury
        eval(['I=simpson(f',num2str(i),',-1,1,10^w);']); % wartość złożonej KS
        wa(-w)=10^w;% wyznaczona przy użyciu zdefiniowanej poniżej funkcji „simpson”
        eval(['bladKW(i,-w)=abs((I',num2str(i),'-I',num2str(i),'))/I']); % względny
        % błąd złożonej KS
    end
    eval(['h=loglog(wa,bladKW(i,:),''.b'');']);
    axis(figaxis(i,:));
    h=xlabel('\delta\alpha_{itI\_m\_a\_x}');
    set(h,'FontName','Times','FontSize',12);
    eval(['h=ylabel(''\delta\alpha_{itI\_},num2str(i),''');']);
    set(h,'Rotation',0,'FontName','Times','FontSize',12);
    title({'Zależność względnego błędu całkowania';'od wskaźnika dokładności'});
    pause(0.1)
end
%=====

```

```

function [I,w]=simpson(f,a,b,deltaImax)
% Wyznaczanie adaptacyjnej złożonej KS dla funkcji f
% I - wartość kwadratury; w - węzły wykorzystywane przez kwadraturę;
% f - wskaźnik do całkowanej funkcji; a, b - krańce przedziału całkowania;
% deltaImax - dopuszczalny bezwzględny błąd całkowania
%-----

% Wyznaczanie wartości I1 prostej KS opartej na węzłach a, c, b
% -----
c=(a+b)/2;
I1=(b-a)/6*(f(a)+4*f(c)+f(b));

% Wyznaczanie wartości I2 złożonej KS opartej na węzłach a, d, c, e, b
% -----
d=(a+c)/2;
e=(c+b)/2;
I2=(b-a)/12*(f(a)+4*f(d)+2*f(c)+4*f(e)+f(b));
if (abs(I1-I2)<deltaImax) % jeżeli różnica między dwiema I1 i I2 jest mniejsza
    I=I2; % od założonej dokładności to kwadratura przyjmuje wartość I2
    w=[a;d;c;e;b]; % a węzłami są [a;d;c;e;b]
else % jeżeli warunek nie jest spełniony to rekurencyjne wywołanie procedury
    [I1,w1]=simpson(f,a,c,deltaImax); % „simpson” dla dwóch podprzedziałów: a - c
    [I2,w2]=simpson(f,c,b,deltaImax); % i c - b
    I=I1+I2; % ostateczna wartość kwadratury jest sumą kwadratur dla dwóch
    w=[w1;w2(2:length(w2))]; % podprzedziałów, a węzły są sumą węzłów
end
%=====

```

Zadanie 7.3

```

clear all
close all
clc

% WYZNACZANIE CAŁKI DWUWYMIAROWEJ
% ZA POMOCĄ ADAPTACYJNEJ ZŁOŻONEJ KWADRATURY SIMPSONA (KS)
%=====

x1=(-1:0.1:1);
x2=(-1:0.1:1)';
for i=1:length(x1)
    for j=1:length(x2)
        f(i,j)=(4*x1(i)*x1(i)-x2(j)*x2(j))^0.5;
    end
end
mesh(x1,x2,f) % wykres funkcji pocałkowej
title('Funkcja podcałkowa dla x_1 i x_2 z przedziału (-1,1)')
[Is,w]=simpson2w(-1,1,1e-4); % odwołanie do zdefiniowanej niżej funkcji
fprintf('\nEstymata całki = %6.4f\n',Is) % "simpson2w"
%=====

function [I,w]=simpson2w(a,b,deltaImax)
% Funkcja wyznaczająca adaptacyjną złożoną kwadraturę Simpsona
% zwraca wartość kwadratury oraz węzły wykorzystywane przez kwadraturę
% Funkcja analogiczna do funkcji „simpson”
%=====

```

```

% ale zaadaptowana do całkowania po obszarze A w dwóch wymiarach
% wykorzystuje funkcję „simpson” do całkowania po jednej zmiennej
%-----

c=(a+b)/2;
x1=[a;c;b];
for k=1:length(x1)
    f=@(x) (4*x1(k)*x1(k)-x*x)^0.5; % ustalenie jednej zmiennej
    g(k,1)=simpson(f,-(1-x1(k)^2)^0.5,(1-x1(k)^2)^0.5,deltaImax); % całkowanie
    end % po drugiej
I1=(b-a)/6*[1,4,1]*g; % formuła KS dla jednego podprzedziału

d=(a+c)/2;
e=(c+b)/2;
x1=[a;d;c;d;b];
for k=1:length(x1)
    f=@(x) (4*x1(k)*x1(k)-x*x)^0.5;
    g(k,1)=simpson(f,-(1-x1(k)^2)^0.5,(1-x1(k)^2)^0.5,deltaImax);
end

I2=(b-a)/12*[1,4,2,4,1]*g; % formuła KS dla dwóch podprzedziałów
if (abs(I1-I2)<deltaImax)
    I=I2;
    w=[a;d;c;e;b];
else
    [I1,w1]=simpson2w(a,c,deltaImax);
    [I2,w2]=simpson2w(c,b,deltaImax);
    I=I1+I2;
    w=[w1;w2(2:length(w2))];
end
%-----


function [I,w]=simpson(f,a,b,deltaImax)
% Wyznaczanie adaptacyjnej złożonej KS dla funkcji f
% I - wartość kwadratury; w - węzły wykorzystywane przez kwadraturę;
% f - wskaźnik do całkowanej funkcji; a, b - krańce przedziału całkowania;
% deltaImax - dopuszczalny bezwzględny błąd całkowania
%-----


% Wyznaczanie wartości I1 prostej KS opartej na węzłach a, c, b
% -----


c=(a+b)/2;
I1=(b-a)/6*(f(a)+4*f(c)+f(b));

% Wyznaczanie wartości I2 złożonej KS opartej na węzłach a, d, c, e, b
% -----


d=(a+c)/2;
e=(c+b)/2;
I2=(b-a)/12*(f(a)+4*f(d)+2*f(c)+4*f(e)+f(b));
if (abs(I1-I2)<deltaImax) % jeżeli różnica między dwiema I1 i I2 jest mniejsza
    I=I2; % od założonej dokładności to kwadratura przyjmuje wartość I2
    w=[a;d;c;e;b]; % a węzłami są [a;d;c;e;b]
else % jeżeli warunek nie jest spełniony to rekurencyjne wywołanie procedury
    [I1,w1]=simpson(f,a,c,deltaImax); % „simpson” dla dwóch podprzedziałów: a - c
    [I2,w2]=simpson(f,c,b,deltaImax); % i c - b
    I=I1+I2; % ostateczna wartość kwadratury jest sumą kwadratur dla dwóch
    w=[w1;w2(2:length(w2))]; % podprzedziałów, a węzły są sumą węzłów
end
%-----

```

Przykład 7.16

```

clear all;close all;clc;format long

% WYZNACZENIE POCHODNEJ ZA POMOCĄ FORMUŁ RÓŻNICOWYCH
%-----


% Różniczkowana funkcja i wartości jej pochodnych
%-----


f=@(x) (1./((1+x.*x).^0.5)-1).^2; % różniczkowana funkcja f(x)
f1=(sqrt(2)-1)/2; % dokładna wartość pierwsiowej pochodnej f(x) dla x=1
f2=(sqrt(2)-1)/2*sqrt(2); % dokładna wartość drugiej pochodnej f(x) dla x=1
f3=-0.53033008; % dokładna wartość trzeciej pochodnej f(x) dla x=1

% Parametry numerycznego różniczkowania
%-----


eps=1.11e-16;
EPS=13.25*eps;
hf_opt=2*sqrt(abs(f1))/abs(f2)*EPS; % optymalny krok dla formuły RP
df_opt=1e-8; % błąd wyznaczenia pochodnej dla formuły progresywnej (RP)
hc_opt=(3*abs(f1))*EPS/abs(f3); % optymalny krok dla formuły RC
dc_opt=2.66e-11; % błąd wyznaczenia pochodnej dla formuły różnicowej centralnej (RC)
h=logspace(-15,0,150); % definicja 150 punktów równomiernie rozłożonych
Df=(f(1+h)-f(1))./h; % wartość pochodnej wyznaczona za pomocą formuły RP
Dc=(f(1+h)-f(1-h))/2./h; % wartość pochodnej wyznaczona za pomocą formuły RC

% Prezentacja wyników
%-----


% Parametry rysunków
%-----


figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
errf=abs(Df-f1)/f1;
errc=abs(Dc-f1)/f1;
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
hh=loglog(h,errf,'-b',h,errc,'-r');
set(hh,'LineWidth',1);
hold on
hh=loglog(hf_opt,df_opt,'ob');
set(hh,'LineWidth',2);
hh=loglog(hc_opt,dc_opt,'^r');
set(hh,'LineWidth',2);
hh=xlabel('1/itf');
set(hh,'FontSize',12);
hh=ylabel('1/\delta');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
title('Zależność względnego błędu różniczkowania od długości kroku')
axis([1e-15 1e0 1e-13 1e-1]);
grid on
zoom on
%-----



```

Przykład 7.18

```

clear all;close all;clc;format long

% WYZNACZENIE POCHODNEJ FUNKCJI GAMMA POPRZEZ RÓŻNICZKOWANIE WIELOMIANU
% INTERPOLUJĄCEGO LOGARYTM TEJ FUNKCJI
%=====

x=(11:0.001:17)';
N=8; % stopień wielomianu interpolującego logarytm funkcji gamma
wezly=linspace(11,17,N+1);
coef=polyfit(wezly,log10(gamma(wezly)),N); % współczynniki wielomianu
                                                % interpolującego logarytm funkcji gamma
coefd=polyder(coef); % współczynniki pochodnej wielomianu interpolującego
                        % logarytm funkcji gamma
fprim=gamma(x)/log10(exp(1)).*polyval(coefd,x); % pochodna funkcji gamma
fprimmatlab=psi(x).*gamma(x); % pochodna funkcji gamma wyznaczona przy użyciu
                                % funkcji Matlaba

% Parametry rysunków
%=====
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunku
f=figure(1);
set(f,'Pos',figpos(1,:));
hold off
plot(x,(fprim-fprimmatlab)./fprimmatlab,'r') % porównanie dwóch pochodnych
hh=xlabel('\itx');
set(hh,'FontName','Times','FontSize',12);
title({'Względna różnica wyników różniczkowania funkcji gamma(x)'...
        'uzyskanych za pomocą dwóch metod numerycznych'});
grid on
zoom on
%
```

Przykład 7.19

```

clear all;close all;clc;format long

% WYZNACZANIE POCHODNEJ ZA POMOCĄ FORMUŁY
% RÓŻNICY CENTRALNEJ Z EKSTRAPODACJĄ RICHARDSONA
%-----

% Różniczkowana funkcja i parametry ekstrapolacji Richardsoa
%-----

f=@(x) sin(x); x=1; % różniczkowana funkcja
h=0.5;
q=sqrt(2); % parametry ekstrapolacji Richardsoa

% Różniczkowanie przy użyciu formuły różnicy centralnej
% z jedno-, dwu-, trzy- i czterokrotną ekstrapolacją Richardsoa
%-----

for w=1:5
    Dc(w,1)=(f(x+h)-f(x-h))/2/h;

```

```

h=h/q;
end
q2=q*q;
for c=1:4
    for w=1:4
        if w>=c
            Dc(w+1,c+1)=(q2*Dc(w+1,c)-Dc(w,c))/(q2-1);
        end
    end
    q2=q2*q*q;
end
B=(Dc-cos(1));
fprintf('\nBłedy bezwzględne przybliżonych wartości pochodnej uzyskanych\n')
fprintf('za pomocą formuły różniczki centralnej z ekstrapolacją Richardsoa\n')
fprintf('\n')
fprintf('%10.3e \n',B(1,1:1))
fprintf('%10.3e %10.3e. \n',B(2,1:2))
fprintf('%10.3e %10.3e %10.3e \n',B(3,1:3))
fprintf('%10.3e %10.3e %10.3e %10.3e \n',B(4,1:4))
fprintf('%10.3e %10.3e %10.3e %10.3e %10.3e\n',B(5,1:5))

```

Zadanie 7.5

```

clear all;close all;clc; warning off

% ZALEŻNOŚĆ BŁĘDU RÓZNICZKOWANIA OD KROKU DLA RÓŻNYCH FORMUŁ
%=====

f=@(x) sin(x); % definicja testowanej funkcji
fprom=@(x) cos(x); % definicja pochodnej testowanej funkcji
h=logspace(-15,-1,15);
xL=-2;
xR=2;
N=100;
x=linspace(xL,xR,N);
for i=1:length(h)
    for j=1:length(x)
        Df(j)=(f(x(j)+h(i))-f(x(j)))/h(i); % różnica progresywna
        Bf(j)=abs(Df(j)-fprom(x(j)));
    end
    MBf(i)=mean(Bf);
    SBf(i)=norm(Bf);
end
% Parametry rysunku
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
        figx+15 5 figx figy]; % położenie rysunku
ff=figure(1);
set(ff,'Pos',figpos(1,:));
loglog(h,MBf)
hold on
ff=figure(2);
set(ff,'Pos',figpos(2,:));
hold off
loglog(h,SBf)

```

```

hold on

for i=1:length(h)
    for j=1:length(x)
        Dc(j)=(f(x(j)+h(i))-f(x(j)-h(i))/2/h(i); % różnica centralna
        Bc(j)=abs(Dc(j)-fprime(x(j)));
    end
    MBc(i)=mean(Bc);
    SBc(i)=norm(Bc);
end
figure(1)
loglog(h,MBc,'r')
figure(2)
loglog(h,SBc,'r')

for i=1:length(h)
    for j=1:length(x)
        Dc5(j)=(f(x(j)-2*h(i))-8*f(x(j)-h(i))+8*f(x(j)+h(i))-f(x(j)+...
            2*h(i))/12/h(i);
        Bc5(j)=abs(Dc5(j)-fprime(x(j))); % formuła pięciopunktowa
    end
    MBc5(i)=mean(Bc5);
    SBc5(i)=norm(Bc5);
end
figure(1)
loglog(h,MBc5,'g')
figure(2)
loglog(h,SBc5,'g')

for i=1:length(h)
    for j=1:length(x)
        wezly=[x(j)-3*h(i);x(j)-2*h(i);x(j)-h(i);x(j);x(j)+h(i);...
            x(j)+2*h(i);x(j)+3*h(i)];
        c=polyfit(wezly-x(j),f(wezly),2);
        Dc7(j)=c(2); % formuła siedmiopunktowa
        Bc7(j)=abs(Dc7(j)-fprime(x(j)));
    end
    MBc7(i)=mean(Bc7);
    SBc7(i)=norm(Bc7);
end
figure(1)
loglog(h,MBc7,'c')
title('Zależność średniego błędu różniczkowania od długości kroku')
legend('formuła dwupunktowa','formuła trójpunktowa','formuła pięciopunktowa',...
    'formuła siedmiopunktowa',3)
hh=xlabel('\it h');
set(hh,'FontName','Times','FontSize',12);
figure(2)
loglog(h,SBc7,'c')
title('Zależność średniokwadratowego błędu różniczkowania od długości kroku')
legend('formuła dwupunktowa','formuła trójpunktowa','formuła pięciopunktowa',...
    'formuła siedmiopunktowa',3)
hh=xlabel('\it h');
set(hh,'FontName','Times','FontSize',12);

```

PROGRAMY DO ROZDZIAŁU 8

Przykład 8.3

```

clear all;close all;clc
% WYZNACZANIE WARTOŚCI OCZEKIWANEJ I ODCHYLENIA STANDARDOWEGO ZMIENNEJ LOSOWEJ
%=====
% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
        figx+15 5 figx figy]; % położenia rysunków
N=10000;
figaxis=[10,N,0.2,0.8;10,N,0.15,0.45]; % osie rysunków
f=figure(1);
hold off
set(f,'Position',figpos(1,:));
f=figure(2);
hold off
set(f,'Position',figpos(2,:));
for iter=1:7
    y=rand(N,1); % siedem ciągów liczb losowych
    for n=10:N
        yn=y(1:n);
        miy(n-9)=sum(yn)/n; % estymata wartości oczekiwanej
        sigmay(n-9)=sqrt(1/(n-1)*sum((yn-miy(n-9)).^2)); % estymata odchylenia
        % standardowego
    end
    figure(1)
    h1=semilogx((10:N),miy,:');
    hold on
    figure(2)
    h5=semilogx((10:N),sigmay,:');
    hold on
end
figure(1)
h2=semilogx([10,N],[0.5,0.5],'g');
h3=semilogx((10:N),0.5+3*sqrt(1/12)./((10:N).^0.5),'--r');
h4=semilogx((10:N),0.5+3*-sqrt(1/12)./((10:N).^0.5),'--r');
title('Estymaty wartości oczekiwanej i przedziały ufności')
axis(figaxis(1,:));
h=xlabel('\it N');
set(h,'FontName','Times','FontSize',12);
legend([h1,h2,h3,h4],'realizacje','wartość średnia','przedział ufności')

figure(2)
h6=semilogx([10,N],[sqrt(1/12),sqrt(1/12)],'g');
title('Estymaty odchylenia standardowego')
legend([h5,h6],'realizacje','odchylenie standardowe')
h=xlabel('\it N');
set(h,'FontName','Times','FontSize',12);
axis(figaxis(2,:));
%=====
```

Przykład 8.4

```

clear all; close all;clc

% WYZNACZANIE UZYSKU PRODUKCYJNEGO DZIELNIKA
%=====

p=5; % dokładność rezystorów w %
k_dokl=2; % dokładność tłumienia w %
k_min=0.5*(1-k_dokl/100);
k_max=0.5*(1+k_dokl/100);
DeltaImax=1e-2;
N=0;
DN=50;
Na=0;
estsigmaI=1;
while estsigmaI>DeltaImax/3 % wieloetapowa procedura Monte-Carlo
    z1=(rand(DN,1)-0.5)*2*p/100; % wylosowanie odchyłki rezystora R1
    z2=(rand(DN,1)-0.5)*2*p/100; % wylosowanie odchyłki rezystora R2
    k=(1+z1)./(2+z1+z2); % tłumienie napięciowe
    DNa=length(find(k>=k_min & k<=k_max)); % liczba sprawnych w tej iteracji
    N=N+DN; % liczba sprawnych ogólnie
    Na=Na+DNa; % liczba eksperymentów ogólnie
    estI=Na/N; % estymata uzysku
    estsigmaI=sqrt(estI*(1-estI)/(N-1)); % estymata odchylenia standardowego
    DN=round(9*estI*(1-estI)/DeltaImax/DeltaImax-N); % oszacowanie potrzebnej
    if (DN<1),DN=50;end % liczby testów
end
fprintf('\nWyniki dla DeltaImax = 1e-2:\n')
fprintf('    estymata uzysku =%6.3f\n',estI)
fprintf('    N =%6i\n',N)
fprintf('    trzysigmowy poziom ufności =%9.6f\n\n',3*estsigmaI)

% Powtórzenie eksperymentu dla większej dokładności
%=====

DeltaImax=1e-3;
N=0;
DN=50;
Na=0;
estsigmaI=1;
while estsigmaI>DeltaImax/3
    z1=(rand(DN,1)-0.5)*2*p/100; % wylosowanie rezystora R1
    z2=(rand(DN,1)-0.5)*2*p/100; % itd.
    k=(1+z1)./(2+z1+z2);
    DNa=length(find(k>=k_min & k<=k_max));
    N=N+DN;
    Na=Na+DNa;
    estI=Na/N;
    estsigmaI=sqrt(estI*(1-estI)/(N-1));
    DN=round(9*estI*(1-estI)/DeltaImax/DeltaImax-N);
    if (DN<1),DN=50;end
end
fprintf('\nWyniki dla DeltaImax = 1e-3:\n')
fprintf('    estymata uzysku =%6.3f\n',estI)
fprintf('    N =%6i\n',N)
fprintf('    trzysigmowy poziom ufności =%9.6f\n\n',3*estsigmaI)
%=====

```

Przykład 8.6

```

clear all; close all;clc

% WYZNACZANIE PRZYBLIŻONEJ WARTOŚCI CAŁKI
%=====

f=@(x) (4-(x-2).^2).^0.5;

% Metoda orzeł-reszka
%-----
N=round(logspace(2,4,10));
I_dokl=2*pi;
for k=1:length(N)
    y=rand(N(k),1)*3; % losowanie x
    x=rand(N(k),1)*2+2; % losowanie y
    uzysk=length(find(y<=f(x))/N(k)); % ile wylosowanych punktów jest pod wykresem
    I_or(k)=uzysk*6;
    sigma_or(k)=6*sqrt(uzysk*(1-uzysk)/(N(k)-1));
end

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
ff=figure(1);
hold off
set(ff,'Pos',figpos)
loglog(N,sigma_or)
hold on
title('Zależność odchylenia standardowego estymat całki od liczby próbek');
pause(0.1)

% Metoda podstawowa
%-----
for k=1:length(N)
    x=rand(N(k),1)*2+2;
    I_po(k)=2*sum(f(x))/N(k);
    sigma_po(k)=sqrt(1/(N(k)-1)*(1/N(k)*sum(f(x).*f(x))-(1/N(k)*sum(f(x))).^2));
end
loglog(N,sigma_po,'g')
pause(0.1)

% Metoda zmiennej kontrolnej r=1
%-----
r=1;
wezly=linspace(2,4,21); % wezły aproksymacji
coef=polyfit(wezly,f(wezly),r); % wyznaczenie współczynników aproksymacji
lg=polyval(polyint(coef),4)-polyval(polyint(coef),2);
for k=1:length(N)
    x=rand(N(k),1)*2+2;
    gg=f(x)-polyval(coef,x);
    I_pzk1(k)=2*sum(gg)/N(k);
    sigma_pzk1(k)=sqrt(1/(N(k)-1)*(1/N(k)*sum(gg.*gg)-(1/N(k)*sum(gg)).^2));
end
loglog(N,sigma_pzk1,'c')
pause(0.1)

```

```

% Metoda zmiennej kontrolnej r=2
%-----
r=2;
coef=polyfit(wezly,f(wezly),r); % wyznaczenie współczynników aproksymacji
Ig=polyval(polyint(coef),4)-polyval(polyint(coef),2);
for k=1:length(N)
    x=rand(N(k),1)*2+2;
    gg=f(x)-polyval(coef,x);
    I_pzk2(k)=2*sum(gg)/N(k);
    sigma_pzk2(k)=sqrt(1/(N(k)-1)*(1/N(k)*sum(gg.*gg)-(1/N(k)*sum(gg))^2));
end
loglog(N,sigma_pzk2,'m')
pause(0.1)

% Metoda zmiennej kontrolnej r=4
%-----
r=4;
coef=polyfit(wezly,f(wezly),r);
Ig=polyval(polyint(coef),4)-polyval(polyint(coef),2);
for k=1:length(N)
    x=rand(N(k),1)*2+2;
    gg=f(x)-polyval(coef,x);
    I_pzk1(k)=2*sum(gg)/N(k);
    sigma_pzk4(k)=sqrt(1/(N(k)-1)*(1/N(k)*sum(gg.*gg)-(1/N(k)*sum(gg))^2));
end
loglog(N,sigma_pzk4,'k')
h=xlabel('itN');
set(h,'FontName','Times','FontSize',12);
h=ylabel('it\sigma_I');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
legend('metoda „orzeł-reszka”','metoda podstawowa','metoda zm.kontrolnej',...
'r=1','metoda zm.kontrolnej, r=2','metoda zm.kontrolnej, r=4');

=====
```

Zadanie 8.1

```

clear all; close all; clc

% WYZNACZANIE OBĘTOŚCI TORUSA
% ZDEFINIOWANEGO RÓWNANIEM  $((x^2+y^2)^{0.5}-R)^2+z^2 \leq r^2$ 
%-----

N=1e6;
x=rand(N,1)*8-4; % losowanie x z przedziału <-4,4>
y=rand(N,1)*8-4; % losowanie y z przedziału <-4,4>
z=rand(N,1)*2-1; % losowanie z z przedziału <-1,1>
R=3;
r=1;
Na=length(find(((x.*x+y.*y).^0.5-R).^2+z.^2<=r.^2)); % ile punktów spełnia
Vt=Na/N*8*8*2; % objętość torusa
sigma=8*8*2*sqrt(Na/N*(1-Na/N)/(N-1)); % odchylenie standardowe
fprintf('\nWyniki obliczeń dla N=%4i\n',N)
fprintf(' estymata objętości torusa = %8.5f\n',Vt);
fprintf(' błąd względny estymaty = %8.5f %%\n', (Vt/(6*pi*pi)-1)*100);
fprintf(' odchylenie standardowe estymaty = %6.4f\n',sigma);

=====
```

Zadanie 8.2

```

clear all; close all; clc

% WYZNACZANIE OBĘTOŚCI KULI K-WYMIAREWJ
%-----
N=1e6;
for K=1:10
    x=rand(N,K)*2-1; % wylosowanie K-wymiarowego x
    Na=length(find(sum(x.*x,2)<=1)); % ile x spełnia równanie kuli
    Vs(K)=Na*N^2^K; % objętość kuli
    sigma(K)=2^K*sqrt(Na/N*(1-Na/N)/(N-1)); % odchylenie standardowe
    Va(K)=(pi)^(K/2)/gamma(K/2+1); % objętość kuli analityczna
end
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630; end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412; end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
f=figure(1);
hold off
set(f,'Pos',figpos)
plot((1:10),sigma);
title('Zależność odchylenia standardowego estymaty objętości kuli'...
' od jej wymiarowości')
h=xlabel('itK');
set(h,'FontName','Times','FontSize',12);
h=ylabel ('it\sigma_it_V ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);

=====
```

Zadanie 8.3

```

clear all; close all; clc

% BADANIE ZALEŻNOŚCI
% ODCHYLENIA STANDARDOWEGO CAŁKI OD GŁADKOŚCI FUNKCJI PODCAŁKOWEJ
%-----

f1=@(x) x.*x;
f2=@(x) abs(x);

% Metoda orzeł-reszka
%-----
N=round(logspace(2,6,30)); %
I1_dokl=2/3;
for k=1:length(N)
    y=rand(N(k),1);
    x=rand(N(k),1)*2-1;
    uzysk=length(find(y<=f1(x))/N(k));
    I1_or(k)=uzysk*2; % wartości estymat całek nie są potrzebne w tym zadaniu
    sigma1_or(k)=2*sqrt(uzysk*(1-uzysk)/(N(k)-1));
end
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630; end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412; end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków

=====
```

```

f=figure(1);
set(f,'Pos',figpos)
hold off
loglog(N,sigma1_or)
hold on
I2_dokl=1;
for k=1:length(N)
    y=rand(N(k),1);
    x=rand(N(k),1)*2-1;
    uzysk=length(find(y<=f2(x)))/N(k);
    I2_or(k)=uzysk*2;
    sigma2_or(k)=2*sqrt(uzysk*(1-uzysk)/(N(k)-1));
end
figure(1)
loglog(N,sigma2_or,:b')

% Metoda podstawowa
for k=1:length(N)
    x=rand(N(k),1)*2-1;
    I1_p(k)=2*sum(f1(x))/N(k);
    sigma1_p(k)=sqrt(1/(N(k)-1)*(1/N(k)*sum(f1(x).*f1(x))-(1/N(k)*sum(f1(x)))^2));
end
figure(1)
loglog(N,sigma1_p,'g')

for k=1:length(N)
    x=rand(N(k),1)*2-1;
    I2_p(k)=2*sum(f2(x))/N(k);
    sigma2_p(k)=sqrt(1/(N(k)-1)*(1/N(k)*sum(f2(x).*f2(x))-(1/N(k)*sum(f2(x)))^2));
end
figure(1)
loglog(N,sigma2_p,:g')
legend('f1, metoda „orzeł-reszka”','f2, metoda „orzeł-reszka”',...
    'f1, metoda podstawowa','f2, metoda podstawowa');
axis([1e2,1e6,1e-4,1e-1]);
h=xlabel('\it{N}');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\it{\sigma_I} ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title('Zależność odchylenia standardowego estymaty całki od liczby próbek')

```

PROGRAMY DO ROZDZIAŁU 9

Przykład 9.3

```

clear all; close all; clc

% ROZWIAZYWANIE TESTOWEGO RÓWNANIA RÓŻNICZKOWEGO  $y'(t) = \lambda y(t)$ 
% OTWARTA METODA EULERA
%=====

% Rozwiązywanie dokładne  $y(t)$  i jego pochodne  $y'(t)$  i  $y''(t)$ 
%-----
y=@(t,lambda) exp(lambda*t);
yprim=@(t,lambda) lambda*exp(lambda*t);
ybis=@(t,lambda) lambda^2*exp(lambda*t);

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figx figy
        5 5 figx figy
        figx+15 100+figy figx figy
        figx+15 5 figx figy]; % położenia rysunków
figaxis=[0 5 0 1;0 5 1e-4 1e4;0 5 0 150;0 5 1e-4 1e4];

% Parametry rozwiązania numerycznego
% -----
T=5; % czas w [ms]
N=20;
t=linspace(0,T,N+1)';
h=t(2)-t(1);

% Zastosowanie otwartej metody Eulera do równania z parametrem  $\lambda=-1$ 
%-----
lambda=-1;
y1(1)=y(0,lambda);
M2=max(abs(ybis(t,lambda)));
L=abs(lambda);
for i=1:N
    y1(i+1,1)=y1(i)+h*lambda*y1(i);
    e24(i+1,1)=M2*h*(exp(L*h*(i))-1)/2/L; % oszacowanie błędu obcienia wg (9.24)
    e26(i+1,1)=lambda^2*h^2*2*((1+h*lambda)^(i)-exp(lambda*h*i))/...
                (1+h*lambda-exp(lambda*h)); % oszacowanie błędu obcienia wg (9.26)
end
f=figure(1);
hold off
set(f,'Position',figpos(1,:));
plot(t,y(t,lambda),'-.k')
hold on
plot(t,y1,'r')
hh=xlabel('\it{t}');
set(hh,'FontName','Times','FontSize',12);
title('Rozwiązywanie dla \lambda = -1');
legend('rozwiązanie dokładne','rozwiązywanie przybliżone')
f=figure(2);

```

```

hold off
set(f,'Position',figpos(2,:));
semilogy(t,abs(y1-y(t,lambda)),'r')
hold on
f=semilogy(t,e24,'+g','MarkerFaceColor','g','MarkerSize',4);
set(f,'LineWidth',2);
semilogy(t,e26,'ok','MarkerFaceColor','k','MarkerSize',4);
hh=xlabel('\itt');
set(hh,'FontName','Times','FontSize',12);
axis(figaxis(2,:))
legend('moduł błędu','oszacowanie błędu wg (9.24)',...
'oszacowanie błędu wg (9.26)',2)
title('Błędy rozwiązań dla \lambda = -1')

% Zastosowanie otwartej metody Eulera do równania z parametrem lambda=1
%-----
lambda=1;
y1(1)=y(0,lambda);
M2=max(ybis(t,lambda));
L=abs(lambda);
for i=1:N
    y1(i+1)=y1(i)+h*lambda*y1(i);
    e24(i+1,1)=M2*h*(exp(L*h*(i))-1)/2/L;
    e26(i+1,1)=lambda^2*h^2/2*((1+h*lambda)^(i)-exp(lambda*h*i))/...
        (1+h*lambda-exp(lambda*h));
end
f=figure(3);
hold off
set(f,'Position',figpos(3,:));
plot(t,y(t,lambda),'-.k')
hold on
plot(t,y1,'r')
hh=xlabel('\itt');
set(hh,'FontName','Times','FontSize',12);
title('Błędy rozwiązań dla \lambda = +1')
legend('rozwiązanie dokładne','rozwiązanie przybliżone',2)
axis(figaxis(3,:))
f=figure(4);
hold off
set(f,'Position',figpos(4,:));
semilogy(t,abs(y1-y(t,lambda)),'r')
hold on
f=semilogy(t,e24,'+g','MarkerFaceColor','g','MarkerSize',4);
set(f,'LineWidth',2);
semilogy(t,e26,'ok','MarkerFaceColor','k','MarkerSize',4);
axis(figaxis(4,:))
hh=xlabel('\itt');
set(hh,'FontName','Times','FontSize',12);
legend('moduł błędu','oszacowanie błędu wg (9.24)',...
'oszacowanie błędu wg (9.26)',4)
title('Błędy rozwiązań dla \lambda = +1')

=====

```

Przykład 9.4

```

clear all; close all; clc

% ROZWIĄZANIE STANDARDOWEGO UKŁADU RC
% PORÓWNANIE EFEKTYWNOŚCI STAŁOKROKOWYCH METOD CAŁKOWANIA: EULERA, HEUNA I RK4
=====
```

```

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
        figx+15 5 figx figy]; % położenia rysunków
figaxis=[0 5 0 1;0 5 1e-6 3e0];

% Definicja funkcji i parametrów układu
%-----
u=@(t,tau) 1-exp(-t/tau);
tau=1;
T=5;
N=20;
t=linspace(0,T,N+1)';
h=t(2)-t(1);

% Otwarta metoda Eulera
%-----
uE=[0;0]; % punkt startowy
for i=1:N
    uE(i+1)=uE(i)+h*f(t(i),uE(i)); % formuła metody otwartej Eulera wg wzoru
    % (9.10) odwołująca się do zdefiniowanej niżej funkcji „f”
end
duE=abs(uE-u(t,tau))./u(t,tau); % błąd względny rozwiązania

% Prezentacja wyników
%-----
ff=figure(1);
set(ff,'Position',figpos(1,:));
hold off
hh=plot(t,u(t,tau),'k');
set(hh,'LineWidth',1.5);
hh=xlabel('\itt');
set(hh,'FontName','Times','FontSize',12);
hh=ylabel('\it{u}(t)\rm{(\it{t}\rm{)}\rm{)}');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
title('Rozwiązanie dokładne')
axis(figaxis(1,:))
ff=figure(2);
set(ff,'Position',figpos(2,:));
hold off
hh=semilogy(t,duE,'r');
set(hh,'LineWidth',2);
hold on

% Metoda Heuna
%-----
uH=[0;0];
for i=1:N
    uH(i+1)=uH(i)+h/2*(f(t(i),uH(i))+... % formuła metody Heuna wg wzoru (9.30)
        f(t(i)+h,uH(i)+h*f(t(i),uH(i)))); % odwołująca się do zdefiniowanej
    % niżej funkcji „f”
end
duH=abs(uH-u(t,tau))./u(t,tau); % błąd względny rozwiązania metodą Heuna
figure(2)
hh=semilogy(t,duH,'--b');
set(hh,'LineWidth',2);

=====
```

```

% Metoda RK4
%-----
uR=[0;0];
for i=1:N
    k1=f(t(i),uR(i)); % definicje współczynników metody RK4 wg wzorów (9.33)
    k2=f(t(i)+h/2,uR(i)+h/2*k1); % odwołujace się do zdefiniowanej niżej
    k3=f(t(i)+h/2,uR(i)+h/2*k2); % funkcji „f”
    k4=f(t(i)+h,uR(i)+h*k3);
    uR(i+1)=uR(i)+h*(1/6*k1+1/3*k2+1/3*k3+1/6*k4); % formula metody RK4
end
duR=abs(uR-u(t,tau))./u(t,tau); % błąd względny rozwiązania metodą RK4
figure(2)
hh=xlabel('\itt');
set(hh,'FontName','Times','FontSize',12);
hh=semilogy(t,duR,'-.g');
set(hh,'LineWidth',2);
title('Błędy względne metod jednokrokowych');
axis(figaxis(2,:));
legend('metoda Eulera','metoda Heuna','metoda Rungego-Kutty rzędu 4')

=====
function uprim=f(t,u)
% Wyznaczanie wartości prawej strony równania różniczkowego (9.2)
% t - czas, u - napięcie na kondensatorze
% e(t) - siła elektromotoryczna w woltach odwołanie do zdefiniowanej funkcji „e”
global licznik % zmienna globalna do liczenia wywołań funkcji
licznik=licznik+1;
C=1; % pojemność w mikrofaradach
R=1; % rezystancja w kiloomach
uprim=(e(t)-u)/R/C; % wzór wynikający z zerowania się sumy napięć w obwodzie
% i równań elementowych: u(t)=R*i(t) dla rezystora
% oraz i(t)=C(u(t))u'(t) dla kondensatora

=====

function E=e(t)
% Wyznaczanie wartości siły elektromotorycznej; t - czas
E=zeros(size(t));
idx=find(t>0);
E(idx)=1;

=====

```

Przykład 9.5

```

clear all; close all; clc
global licznik
licznik=0; % zmienna globalna do zliczania wywołań funkcji

% ROZWIĄZYwanie UKŁADU RÓWNAŃ Z PRZYKŁADU 9.1 ZA POMOCĄ
% STAŁOKROKOWEJ I ZMIENNOKROKOWEJ OTWARTEJ METODY EULERA
%-----

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków

```

```

figpos=[5 100+figy figx figy
        5 5 figx figy
        figx+15 100+figy figx figy]; % położenia rysunków
figaxis=[0 5 0 1.3e-4;0 5 9e-6 4e-4;0 5 3e-4 1e-1];

% Dokładne rozwiązanie - u(t)
%-----
u=@(t,tau) 1-exp(-t/tau); tau=1;

% Stałokrokowa otwarta metoda Eulera
%-----
T=5;
hs=6.4e-4;
t=(0:hs:5)';
N=length(t)-1;
us=[0;0];
for i=1:N
    us(i+1)=us(i)+hs*f(t(i),us(i)); % formula metody otwartej Eulera wg wzoru
    % (9.10) odwołująca się do zdefiniowanej niżej funkcji „f”
end
lws=licznik; % liczba wywołań funkcji f(t,u) przez metodę stałokrokową
Duss=abs(us-u(t,tau)); % błąd bezwzględny rozwiązania numerycznego
duss=abs(us-u(t,tau))./u(t,tau); % błąd względny rozwiązania numerycznego
ff=figure(1);
set(ff,'Position',figpos(1,:));
hold off
plot(t,Dus,'-r')
hold on
ff=figure(2);
set(ff,'Position',figpos(2,:));
hold off
semilogy(t,dus,'-r')
hold on
ff=figure(3);
set(ff,'Position',figpos(3,:));
hold off
semilogy([t(1),t(N)],[hs hs],'-r')
hold on

% Zmiennokrokowa otwarta metoda Eulera
%-----
clear t
T=5;
hz=1e-3;
t=[0;0];
uz=[0;0];
i=1;
emax=1e-3;
ro=0.5;
p=1;
lw=0;
licznik=0; % zmienna globalna do zliczania wywołań funkcji
while t<=T
    y1=uz(i)+hz*f(t(i),uz(i)); % wzór (9.10) dla kroku hz
    y2=uz(i)+hz/2*f(t(i),uz(i)); % wzór (9.10) dla pierwszego kroku hz/2
    y2=y2+hz/2*f(t(i)+hz/2,y2); % wzór (9.10) dla drugiego kroku hz/2
    Deltan1=y1-y2;
    rn1=Deltan1/(2^p-1); % oszacowanie błędu lokalnego wg wzoru (9.36)
    rmax=hz/T*emax; % oszacowanie dopuszczalnego błędu lokalnego wg wzoru (9.37)
    if abs(rn1)<=rmax % jeżeli bieżące rozwiązanie jest wystarczająco dokładne:

```

```

i=i+1;
uz(i)=y2; % zapamiętanie wyniku
t(i)=t(i-1)+hz;
if abs(rn1)<rmax*ro % jeżeli błąd dostatecznie mały
    hz=hz/max(0.1,(abs(rn1)/rmax)^(1/(p+1))); % zwiększenie kroku
end
else % jeżeli bieżące rozwiązanie nie jest wystarczająco dokładne:
    hz=hz/min(10,(abs(rn1)/rmax)^(1/(p+1))); % zmniejszenie kroku
end
end
lzw=licznik; % liczba wywołań funkcji f(t,u) przez metodę zmiennokrokową;
Duz=abs(uz-u(t,tau)); % błąd bezwzględny rozwiązania numerycznego
duz=abs(uz-u(t,tau))./u(t,tau); % błąd względny rozwiązania numerycznego

% Graficzna prezentacja wyników
%-----
figure(1);
plot(t,Duz)
hh=ylabel('\Delta\|u\|');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
hh=xlabel('t\|t\|');
set(hh,'FontName','Times','FontSize',12);
axis(figaxis(1,:));
legend('metoda stałokrokowa','metoda zmiennokrokowa','Location','South')
title('Błąd bezwzględny rozwiązania')
figure(2);
semilogy(t,duz)
hh=ylabel('|\delta\|u\|');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
hh=xlabel('t\|t\|');
set(hh,'FontName','Times','FontSize',12);
axis(figaxis(2,:));
legend('metoda stałokrokowa','metoda zmiennokrokowa',3)
title('Błąd względny rozwiązania')
figure(3);
semilogy(t,[0;diff(t)])
hh=ylabel('|\dot{u}\|');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
hh=xlabel('t\|t\|');
set(hh,'FontName','Times','FontSize',12);
legend('metoda stałokrokowa','metoda zmiennokrokowa',2)
axis(figaxis(3,:));
title('Krok całkowania')

% Alfanumeryczna prezentacja wyników
%-----
fprintf('\nMetoda stałokrokowa\n');
fprintf('-----\n');
fprintf('Liczba odwołań do funkcji f(t,u): %3i\n',lws)
fprintf('Maksymalny błąd globalny u(t) : %10.3e\n',max(Dus));
fprintf('-----\n');
fprintf('Liczba odwołań do funkcji f(t,u): %3i\n',lzw);
fprintf('Maksymalny błąd globalny u(t) : %10.3e\n',max(Duz));

%-----
function uprim=f(t,u)
% Wyznaczanie wartości prawej strony równania różniczkowego (9.2)
% t - czas, u - napięcie na kondensatorze

```

```

% e(t) - siła elektromotoryczna w voltach odwołanie do zdefiniowanej funkcji "e"
global licznik % zmienna globalna do liczenia wywołań funkcji
licznik=licznik+1;
C=1; % pojemność w mikrofaradach
R=1; % rezystancja w kiloomach
uprim=(e(t)-u)/R/C; % wzór wynikający z zerowania się sumy napięć w obwodzie
% i równań elementowych: u(t)=R*i(t) dla rezystora
% oraz i(t)=C(u(t))u'(t) dla kondensatora
%=====

function E=e(t)
% Wyznaczanie wartości siły elektromotorycznej; t - czas
E=zeros(size(t));
idx=find(t>=0);
E(idx)=1;

%=====

Przykład 9.6

clear all; close all; clc; warning off
global licznik
licznik=0; % zmienna globalna do zliczania wywołań funkcji

% ROZWIAZYwanie UKŁADU RÓWNAŃ Z PRZYKŁADU 9.1 ZA POMOCĄ
% METOD RUNGEGO-KUTTY ODE23 I ODE45 DLA DWÓCH WARTOŚCI PARAMETRU RelTol
%=====

% dokładne rozwiązanie - u(t)
%-----
u=@(t,tau) 1-exp(-t/tau); tau=1;

% Metoda „ode23” dla parametru RelTol=1e-3
%-----
T=5;
options = odeset('RelTol',1e-3,'AbsTol',1e-6); % opcje procedury ode23
[t,uRK23] = ode23(@f,[0 T],0,options); % odwołanie do zdefiniowanej funkcji "f"
lw=licznik; %liczba odwołań do funkcji f
DuR=abs(uRK23-u(t,tau)); % błąd bezwzględny rozwiązania numerycznego
duR=DuR./u(t,tau); % błąd względny rozwiązania numerycznego

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figy figx figy
      5 5 figx figy
      figx+15 100+figy figx figy]; % położenia rysunków
figaxis=[0 5 1e-7 2e-3;0 5 1e-7 2e-3;0 5 1e-3 1];
ff=figure(1);
set(ff,'Position',figpos(1,:));
hold off
semilogy(t,DuR,'k')
hold on
ff=figure(2);
set(ff,'Position',figpos(2,:));
hold off

```

```

semilogy(t,duR,'k')
hold on
ff=figure(3);
set(ff,'Position',[3, :]);
hold off
semilogy(t(2:length(t)),diff(t),'k')
hold on
fprintf('\nProcedura "ode23" dla RelTol=1e-3 i AbsTol=1e-6:\n');
fprintf(' liczba odwołań do funkcji f(t,u) = %3i\n',lw)
fprintf(' maksymalny błąd bezwzględny u(t) = %8.1e\n',max(DuR));
fprintf(' maksymalny błąd względny u(t) = %8.1e\n',max(duR));
% Metoda „ode45” dla parametru RelTol=1e-3
%-----
T=5;
licznik=0;
[t,uRK45] = ode45(@f,[0 T],0,options); % odwołanie do zdefiniowanej funkcji „f”
lw=licznik;
figure(1)
DuR=abs(uRK45-u(t,tau)); % błąd bezwzględny rozwiązania numerycznego
duR=DuR./u(t,tau); % błąd względny rozwiązania numerycznego

% Prezentacja wyników dla metody „ode45” RelTol=1e-3
%-----
figure(1)
plot(t,DuR,'-r')
axis(figaxis(1,:));
h=xlabel('itt');
set(h,'FontName','Times','FontSize',12);
h=ylabel('Delta\itu');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Błąd bezwzględny rozwiązania'])
legend('procedura "ode23"', 'procedura "ode45"', 'Location', 'SouthWest')
zoom on
figure(2)
semilogy(t,duR,'-r')
axis(figaxis(2,:));
h=xlabel('itt');
set(h,'FontName','Times','FontSize',12);
h=ylabel('delta\itu');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Błąd względny rozwiązania'])
legend('procedura "ode23"', 'procedura "ode45"', 'Location', 'SouthWest')
zoom on
figure(3)
semilogy(t(2:length(t)),diff(t),'-r')
axis(figaxis(3,:));
h=xlabel('itt');
set(h,'FontName','Times','FontSize',12);
h=ylabel('ith');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Krok całkowania'])
legend('procedura "ode23"', 'procedura "ode45"', 'Location', 'SouthEast')
zoom on
fprintf('\nProcedura "ode45" dla RelTol=1e-3 i AbsTol=1e-6:\n');
fprintf(' liczba odwołań do funkcji f(t,u) = %3i\n',lw)
fprintf(' maksymalny błąd bezwzględny u(t) = %8.1e\n',max(DuR));
fprintf(' maksymalny błąd względny u(t) = %8.1e\n',max(duR));
fprintf('\nAby kontynuować, naciśnij klawisz\n')
pause

```

```

% Metoda „ode23” dla parametru RelTol=1e-5
%-----
figaxis=[0 5 1e-8 2e-5;0 5 1e-8 2e-5;0 5 1e-3 1];
licznik=0;
options = odeset('RelTol',1e-5,'AbsTol',1e-6);
[t,uRK23] = ode23(@f,[0 T],0,options); % odwołanie do zdefiniowanej funkcji „f”
lw=licznik;
figure(1)
hold off
DuR=abs(uRK23-u(t,tau));
semilogy(t,DuR,'k')
hold on
figure(2)
hold off
duR=DuR./u(t,tau);
semilogy(t,dur,'k')
hold on
figure(3)
hold off
semilogy(t(2:length(t)),diff(t),'k')
hold on
fprintf('\nProcedura "ode23" dla RelTol=1e-5 i AbsTol=1e-6:\n');
fprintf(' liczba odwołań do funkcji f(t,u) = %3i\n',lw)
fprintf(' maksymalny błąd bezwzględny u(t) = %8.1e\n',max(DuR));
fprintf(' maksymalny błąd względny u(t) = %8.1e\n',max(duR));

% Metoda „ode45” dla parametru RelTol=1e-5
%-----
licznik=0;
[t,uRK45] = ode45(@f,[0 T],0,options); % odwołanie do zdefiniowanej funkcji „f”
lw=licznik;
DuR=abs(uRK45-u(t,tau));
figure(1)
plot(t,DuR,'-r')
axis(figaxis(1,:));
h=xlabel('itt');
set(h,'FontName','Times','FontSize',12);
h=ylabel('Delta\itu');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Błąd bezwzględny rozwiązania'])
legend('procedura "ode23"', 'procedura "ode45"', 'Location', 'SouthWest')
zoom on

figure(2)
duR=DuR./u(t,tau);
semilogy(t,duR,'-r')
axis(figaxis(2,:));
h=xlabel('itt');
set(h,'FontName','Times','FontSize',12);
h=ylabel('delta\itu');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Błąd względny rozwiązania'])
legend('procedura "ode23"', 'procedura "ode45"', 'Location', 'SouthWest')
zoom on

figure(3)
semilogy(t(2:length(t)),diff(t),'-r')
axis(figaxis(3,:));
h=xlabel('itt');
set(h,'FontName','Times','FontSize',12);

```

```

h=label('\ith');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Krok całkowania'])
legend('procedura "ode23"', 'procedura "ode45"', 'Location','SouthEast')
zoom on
fprintf('\nProcedura „ode45” dla RelTol=1e-5 i AbsTol=1e-6:\n');
fprintf(' liczba odwołań do funkcji f(t,u) = %3i\n',lw)
fprintf(' maksymalny błąd bezwzględny u(t) = %8.1e\n',max(DuR));
fprintf(' maksymalny błąd wzglny u(t) = %8.1e\n',max(duR));

%=====

function uprim=f(t,u)
% Wyznaczanie wartości prawej strony równania różniczkowego (9.2)
% t - czas, u - napięcie na kondensatorze
% e(t) - siła elektromotoryczna woltach odwołanie do zdefiniowanej funkcji „e”
global licznik % zmieni globalna do liczenia wywołań funkcji
licznik=licznik+1;
C=1; % pojemność w mikrofaradach
R=1; % rezystancja w kiloomach
uprim=(e(t)-u)/R/C; % wzór wynikający z zerowania się sumy napięć w obwodzie
% i równań elementowych: u(t)=R*i(t) dla rezystora
% oraz i(t)=C(u(t))u'(t) dla kondensatora

%=====

function E=e(t)
% Wyznaczanie wartości siły elektromotorycznej; t - czas
E=zeros(size(t));
idx=find(t>=0);
E(idx)=1;

```

Przykład 9.8

```

clear all; close all; clc

% ROZWIAZYwanie UKŁADU RÓWNAŃ Z PRZYKŁADU 9.1 ZA POMOCĄ OTWARTEJ METODY EULERA
% ORAZ DWUKROKOWEJ METODY ZDEFINIOWANEJ WZOREM (9.59)
% ILUSTRACJA NIESTABILNOŚCI METODY DWUKROKOWEJ
%=====

% Dokładne rozwiązanie - u(t)
%-----
u=@(t,tau) 1-exp(-t/tau); tau=1;

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figx 2*figx figy
      5 5 2*figx figy]; % położenia rysunków
figaxis=[0 0.2 -50 200;0 0.2 1e-10 1e5];

% Otwarta metoda Eulera dla h=0.01
%-----
T=5;
h=0.01;

```

```

t=(0:h:T)';
N=length(t)-1;
uE=[0;0]; % dokładny warunek początkowy
for i=1:N
    uE(i+1)=uE(i)+h*f(t(i),uE(i));
end
DuE=abs(uE-u(t,tau)); % błąd bezwzględny rozwiązania numerycznego
ff=figure(1);
set(ff,'Position',figpos(1,:));
hold off
hh=plot(t(1:20),u(t(1:20),tau),'r');
set(hh,'LineWidth',2);
hold on
plot(t(1:20),uE(1:20),'ok')
ff=figure(2);
set(ff,'Position',figpos(2,:));
hold off
hh=semilogy(t(1:20),DuE(1:20),'ok');
set(hh,'LineWidth',2);
hold on

% Metoda dwukrokowa zdefiniowana wzorem (9.59) dla h=0.01
%-----
T=5;
h=0.01;
t=(0:h:T)';
N=length(t)-1;
ud(1)=0;ud(2,1)=u(h,tau); % dokładne warunki początkowe
for i=2:N
    ud(i+1)=-4*ud(i)+5*ud(i-1)+h*(4*f(t(i),ud(i))+2*f(t(i-1),ud(i-1))); % wzór
end
Dud=abs(ud-u(t,tau)); % błąd bezwzględny rozwiązania numerycznego
figure(1)
plot(t(1:20),ud(1:20),'-b')
figure(2)
hh=semilogy(t(1:20),Dud(1:20),'-b');
set(hh,'LineWidth',2);

% Metoda dwukrokowa zdefiniowana wzorem (9.59) dla h=0.005
%-----
T=5;
h=0.005;
t=(0:h:T)';
N=length(t)-1;
ud(1)=0;ud(2,1)=u(h,tau); % dokładne warunki początkowe
for i=2:N
    ud(i+1)=-4*ud(i)+5*ud(i-1)+h*(4*f(t(i),ud(i))+2*f(t(i-1),ud(i-1)));
end
Dud=abs(ud-u(t,tau));
idx=min(find(ud>200)); % identyfikacja momentu wzbudzenia oscylacji (> 200)
figure(1)
plot(t(1:idx),ud(1:idx),'g') % rysowanie rozwiązania do momentu wzbudzenia
axis(figaxis(1,:)); % oscylacji (> 200)
h=ylabel('\ith\rm [\itv\rm]');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
h=xlabel('\itt\rm [\itms\rm]');
set(h,'FontName','Times','FontSize',12);
title('Rozwiązań przybliżone')
legend('rozwiązań dokładne','metoda Eulera, h=0.01',...
'metoda dwukrokowa, h=0.01','metoda dwukrokowa, h=0.005',2)

```

```

figure(2)
hh=semilogy(t(1:idx),Dud(1:idx),'g');
set(hh,'LineWidth',2);
axis(figaxis(2,:));
h ylabel('|\Delta|/|u| [10^m]');
set(h,'Rotation',0,'FontSize',12);
h xlabel('|t|/s [10^m]');
set(h,'FontSize',12);
legend('metoda Eulera, h=0.01','metoda dwukrokowa, h=0.01',...
'metoda dwukrokowa, h=0.005',2)
title('Błędy bezwzględne rozwiązań przybliżonych')

=====
function uprim=f(t,u)
% Wyznaczanie wartości prawej strony równania różniczkowego (9.2)
% t - czas, u - napięcie na kondensatorze
% e(t) - siła elektromotoryczna w woltach odwołanie do zdefiniowanej funkcji „e”
global licznik % zmienna globalna do liczenia wywołań funkcji
licznik=licznik+1;
C=1; % pojemność w mikrofaradach
R=1; % rezystancja w kilomach
uprim=(e(t)-u)/R/C; % wzór wynikający z zerowania się sumy napięć w obwodzie
% i równań elementowych: u(t)=R*i(t) dla rezystora
% oraz i(t)=C(u(t))u'(t) dla kondensatora

=====
function E=e(t)
% Wyznaczanie wartości siły elektromotorycznej; t - czas
E=zeros(size(t));
idx=find(t>=0);
E(idx)=1;

=====

Przykład 9.12

clear all; close all; clc

% ROZWIAZYwanIE UKŁADU RÓWNAŃ RÓZNICZKOWYCH (9.67) OTWARTA METODA EULERA
=====

% Prawa strona układu równań (9.67) - f(t,u)
% oraz jego dokładne rozwiązanie - u=[u1 u2]
%
f=@(t,u) [-1001 1;1 -1]*u+[e(t)*1e3;0];
lambda1=-501+sqrt(250001);
lambda2=-501-sqrt(250001);
c1=lambda2/(lambda1-lambda2);
c2=-1-c1;
u2=@(t,c1,c2,lambda1,lambda2) c1*exp(lambda1*t)+c2*exp(lambda2*t)+1; % wzory
u1=@(t,c1,c2,lambda1,lambda2) c1*(1+lambda1)*exp(lambda1*t)... % (9.68a)
+ c2*(1+lambda2)*exp(lambda2*t)+1; % i (9.68b)

% Parametry rysunków
%
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków

```

```

figpos=[5 100+figy 2*figx figy
        5 2*figx figy]; % położenia rysunków
figaxis=[0 0.01 0 3;0 0.01 0 0.009];
tu=(0:le-4:0.01)'; % wektor czasu do wykreslenia u(t) dokładnego

% Rozwiązańe dla h=2e-3
-----
T=1;
h=2e-3;
t=(0:h:T)';
N=length(t)-1;
u=[0;0];
for i=1:N
    u(:,i+1)=u(:,i)+h*f(t(i),u(:,i));
end
u=u';
ff=figure(1);
hold off
set(ff,'Position',figpos(1,:));
plot(tu,u1(tu,c1,c2,lambda1,lambda2),'g')
hold on
plot(t(1:6),u(1:6,1),'r')
ff=figure(2);
hold off
set(ff,'Position',figpos(2,:));
plot(tu,u2(tu,c1,c2,lambda1,lambda2),'g')
hold on
plot(t(1:6),u(1:6,2),'r')

% Rozwiązańe dla h=1e-3
-----
h=1e-3;
t=(0:h:T)';
N=length(t)-1;
u=[0;0];
for i=1:N
    u(:,i+1)=u(:,i)+h*f(t(i),u(:,i));
end
u=u';
figure(1);
plot(t(1:30),u(1:30,1),'b')
h xlabel('|t|');
set(h,'FontName','Times','FontSize',12);
h ylabel('|\Delta|/|u|_1');
set(h,'Rotation',0,'FontSize',12);
axis(figaxis(1,:));
legend('rozwiązańe dokładne','rozwiązańe uzyskane metodą Eulera, h=0.002',...
'rozwiązańe uzyskane metodą Eulera, h=0.001','2');
title('Napięcie na kondensatorze C_1')
figure(2);
plot(t(1:30),u(1:30,2),'b')
h xlabel('|t|');
set(h,'FontName','Times','FontSize',12);
h ylabel('|\Delta|/|u|_2');
set(h,'Rotation',0,'FontSize',12);
axis(figaxis(2,:));
legend('rozwiązańe dokładne','rozwiązańe uzyskane metodą Eulera, h=0.002',...
'rozwiązańe uzyskane metodą Eulera, h=0.001','2');
title('Napięcie na kondensatorze C_2')

=====
```

```

function E=e(t)
% Wyznaczanie wartości siły elektromotorycznej; t - czas
E=zeros(size(t));
idx=find(t>=0);
E(idx)=1;
%
```

Przykład 9.13

```

clear all; close all; clc; kolory='rbgk';

% ROZWIĄZYwanIE UKŁADU RÓWNAŃ RÓŻNICZKOWYCH (9.67)
% ZA POMOCĄ ZAMKNIĘTYCH METOD GEARA RZĘDU K=1,2,3,4
%-----  

% Prawa strona układu równań (9.67) - A*u+b
% oraz jego dokładne rozwiązanie - u=[u1 u2]
%-----  

A=[-1001,1;1,-1];
b=[1000;0];
lambda1=-501+sqrt(250001);
lambda2=-501-sqrt(250001);
c1=lambda2/(lambda1-lambda2);
c2=-1-c1;
u2=@(t,c1,c2,lambda1,lambda2) c1*exp(lambda1*t)+c2*exp(lambda2*t)+1; % wzory
u1=@(t,c1,c2,lambda1,lambda2) c1*(1+lambda1)*exp(lambda1*t)... % (9.68a)
+ c2*(1+lambda2)*exp(lambda2*t)+1; % i (9.68b)

% Współczynniki zamkniętych metod Garea rzędu K=1,2,3,4
beta0=[1/2;3/6;11/12;25];
alfa=[ 1      0      0      0;
      4/3    -1/3    0      0;
     18/11   -9/11   2/11   0;
     48/25  -36/25  16/25  -3/25];

% Parametry rysunków
%-----  

figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figx figx figy
        figx+15 100+figy figx figy
        5 5 figx figy]; % położenia rysunków
figaxis=[0 0.015 1e-10 1e-1;0 5 1e-4 1e4;0 0.015 1e-10 1e-1;0 5 1e-4 1e4];

% Parametry rozwiązań numerycznych
%-----  

T=0.015;
h=1e-4;
t=(0:h:T)';
N=length(t)-1;
u=[0;0];

% Rozwiązania dla dokładnych warunków początkowych
%-----  

ts=[h,2*h,3*h];
us=[[0 u1(ts,c1,c2,lambda1,lambda2)];[0 u2(ts,c1,c2,lambda1,lambda2)]];

```

```

f=figure(1);
hold off
set(f,'Position',figpos(1,:));
f=figure(2);
hold off
set(f,'Position',figpos(2,:));
for K=1:4
    clear u
    u(:,1:K)=us(:,1:K);
    for i=K:N % realizacja zamkniętej metody Garea rzędu K
        A1=eye(2)-h*beta0(K)*A;
        Y=[];
        for j=0:K-1
            Y=[Y u(:,i-j)];
        end
        B=h*beta0(K)*b+Y*alfa(1:K,K);
        u(:,i+1)=A1\B;
    end
    u=u';
    figure(1)
    semilogy(t,abs(u(:,1)...
               u1(t,c1,c2,lambda1,lambda2))./u1(t,c1,c2,lambda1,lambda2),kolory(K))
    hold on
    figure(2)
    semilogy(t,abs(u(:,2)...
               u2(t,c1,c2,lambda1,lambda2))./u2(t,c1,c2,lambda1,lambda2),kolory(K))
    hold on
end
figure(1)
legend('K=1','K=2','K=3','K=4');
hh=xlabel('\itt');
set(hh,'FontName','Times','FontSize',12);
hh=ylabel('|\delta|/u_{rm\_1}');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
title('Dokładne warunki początkowe');
axis(figaxis(1,:));
figure(2)
legend('K=1','K=2','K=3','K=4',3);
hh=xlabel('\itt');
set(hh,'FontName','Times','FontSize',12);
hh=ylabel('|\delta|/u_{rm\_2}');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
title('Niedokładne warunki początkowe');

% Rozwiązania dla niedokładnych warunków początkowych
%-----  

f=figure(3);
hold off
set(f,'Position',figpos(3,:));
f=figure(4);
hold off
set(f,'Position',figpos(4,:));
us=[0;0];
for K=1:4
    u=us;
    for i=K:N % realizacja zamkniętej metody Garea rzędu K
        A1=eye(2)-h*beta0(K)*A;
        Y=[];
        for j=0:K-1
            Y=[Y u(:,i-j)];
        end
    end

```

```

B=h*beta0(K)*b+Y*alfa(1:K,K);
u(:,i+1)=A1\B;
end
us=u(:,1:K+1); % warunki początkowe dla metody wyższego rzędu
u=u';
figure(3)
semilogy(t,abs(u(:,1)...
    u1(t,c1,c2,lambda1,lambda2))./u1(t,c1,c2,lambda1,lambda2),kolory(K));
hold on
figure(4)
semilogy(t,abs(u(:,2)...
    u2(t,c1,c2,lambda1,lambda2))./u2(t,c1,c2,lambda1,lambda2),kolory(K));
hold on
end
figure(3)
h=xlabel('\itt');
set(h,'FontName','Times','FontSize',12);
legend('K=1','K=2','K=3','K=4',3);
hh=ylabel('|\delta|/|u_0|');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
title('Dokładne warunki początkowe');
axis(figaxis(1,:));
figure(4)
h=xlabel('\itt');
set(h,'FontName','Times','FontSize',12);
legend('K=1','K=2','K=3','K=4',1);
hh=ylabel('|\delta|/|u_0|');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
title('Niedokładne warunki początkowe');

=====

```

Zadanie 9.1

```

clear all; close all; clc;

% OBSZARY STABILNOŚCI METOD ADAMSA-BASHFORTHA I ADAMSA-MOULTONA
%-----

fi=linspace(0,2*pi,1000)';
% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-100; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy
    figx+15 5 figx figy]; % położenia rysunków
f=figure(1);
hold off
set(f,'Pos',figpos(1,:));

% Metody Adamsa-Bashfortha
%-----
%K=1
alfa=[1];
beta=[1];beta0=0;
hlambda=h_lambda(fi,alfa,beta,beta0); % odwołanie do zdefiniowanej niżej
h=plot(real(hlambda),imag(hlambda),'b'); % funkcji h_lambda
hold on

```

```

set(h,'LineWidth',1.7);
axis([-2.1 0.5,-1.1,1.1]);
axis equal
h=text(-1.98,0.05,'K');
set(h,'Color','b','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-1.88,0.05,'=1');
set(h,'Color','b','FontName','Times','FontSize',12,'FontAngle','normal');
title('Obszary absolutnej stabilności dla metod Adamsa-Bashfortha');

%K=2
alfa=[1 0];
beta=[3/2 -1/2];beta0=0;
hlambda=h_lambda(fi,alfa,beta,beta0); % odwołanie do funkcji h_lambda
h=plot(real(hlambda),imag(hlambda),'g');
set(h,'LineWidth',1.7);
h=text(-1.28,0.05,'K');
set(h,'Color','g','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-1.18,0.05,'=2');
set(h,'Color','g','FontName','Times','FontSize',12,'FontAngle','normal');

%K=3
alfa=[1 0 0];
beta=[23/12 -4/3 5/12];beta0=0;
hlambda=h_lambda(fi,alfa,beta,beta0); % odwołanie do funkcji h_lambda
h=plot(real(hlambda),imag(hlambda),'r');
set(h,'LineWidth',1.7);
h=text(-0.83,0.05,'K');
set(h,'Color','r','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-0.72,0.05,'=3');
set(h,'Color','r','FontName','Times','FontSize',12,'FontAngle','normal');

%K=4
alfa=[1 0 0 0];
beta=[55/24 -59/24 37/24 -3/8];beta0=0;
hlambda=h_lambda(fi,alfa,beta,beta0); % odwołanie do funkcji h_lambda
h=plot(real(hlambda),imag(hlambda),'c');
set(h,'LineWidth',1.7);
h=text(-0.28,0.05,'K');
set(h,'Color','c','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-0.18,0.05,'=4');
set(h,'Color','c','FontName','Times','FontSize',12,'FontAngle','normal');

line([-2.2 0.6],[0,0],'Color','k')
line([0 0],[-1.1,1.1],'Color','k')
h=xlabel('Im[\lambda] vs \lambda');
set(h,'FontName','Times','FontSize',12);
hh=ylabel('Re[\lambda]');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);

% Metody Adamsa-Moultona
%-----
f=figure(2);
set(f,'Pos',figpos(2,:));
hold off
%K=1
alfa=[1];
beta=[0];beta0=1;
hlambda=h_lambda(fi,alfa,beta,beta0); % odwołanie do funkcji h_lambda
h=plot(real(hlambda),imag(hlambda),'g');

```

```

set(h,'LineWidth',1.7);
axis([-6.5 2.2,-4,4]);
axis equal
title('Obszary absolutnej stabilności dla metod Adamsa-Moultona');
hold on
h=text(0.7,0.2,'K');
set(h,'Color','g','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(1.07,0.2,'=1');
set(h,'Color','g','FontName','Times','FontSize',12,'FontAngle','normal');

%K=1+
% alfa=[1];
% beta=[1/2];beta0=1/2;
x=[0 0];y=[-4 4];
h=plot(x,y,'b');
set(h,'LineWidth',1.7);
hold on
h=text(-1.2,3.5,'K');
set(h,'Color','b','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-0.85,3.5,'=1');
set(h,'Color','b','FontName','Times','FontSize',12,'FontAngle','normal');

%K=2
alfa=[1 0 ];
beta=[2/3 -1/12]';beta0=5/12;
h_lambda=h_lambda(fi,alfa,beta,beta0); % odwołanie do funkcji h_lambda
h=plot(real(h_lambda),imag(h_lambda),'r');
set(h,'LineWidth',1.7);
hold on
h=text(-5.9,0.2,'K');
set(h,'Color','r','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-5.55,0.2,'=2');
set(h,'Color','r','FontName','Times','FontSize',12,'FontAngle','normal');

%K=3
alfa=[1 0 0 ];
beta=[19/24 -5/24 1/24]';beta0=3/8;
h_lambda=h_lambda(fi,alfa,beta,beta0); % odwołanie do funkcji h_lambda
h=plot(real(h_lambda),imag(h_lambda),'k');
set(h,'LineWidth',1.7);
hold off
h=text(-2.9,0.2,'K');
set(h,'Color','k','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-2.55,0.2,'=3');
set(h,'Color','k','FontName','Times','FontSize',12,'FontAngle','normal');

line([-7,3],[0,0],'Color','k')
h=xlabel('Im[\lambda]');
set(h,'FontName','Times','FontSize',12);
hh=ylabel('Re[\lambda]');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
figure(1);

%=====

function h_lambda=h_lambda(fi,alfa,beta,beta0)
% Wyznaczanie zespolonej wartości funkcji wymiernej,
% której licznik jest wielomianem o współczynnikach alfa,
% a mianownik wielomianem o współczynnikach beta.
% Argumentem funkcji wymiernej jest liczba zespolona exp(-j*fi),

```

```

% przy czym fi przybiera wartości z przedziału [0, 2*pi].
%-----
N=size(fi,1);
K=size(alfa,1);
num=ones(N,1)+j*zeros(N,1);
den=beta0*ones(N,1)+j*zeros(N,1);
for k=1:K
    num=num-alfa(k)*exp(-j*k*fi);
    den=den+beta(k)*exp(-j*k*fi);
end
h_lambda=num./den;
%=====

Zadanie 9.2

clear all; close all; clc

% OBSZARY ABSOLUTNEJ STABILNOŚCI METOD RUNGEGO-KUTTY RZĘDU K=1,2,3,4
%=====

% Metoda Rungego-Kutty rzędu K=1
%-----
fi=linspace(0,2*pi,1000); % faza bieżącego punktu na brzegu obszaru stabilności
% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 5 figx figy]; % położenia rysunków
ff=figure(1);
set(ff,'Pos',figpos(1,:));
hold off
N=size(fi,1);
h_lambda=zeros(N,1)+j*zeros(N,1); % pierwszy punkt na brzegu obszaru stabilności
for n=2:N
    z0=h_lambda(n-1); % pierwsze przybliżenie n-tego punktu na brzegu obszaru stab.
    for k=1:10 % iteracyjne poprawianie n-tego punktu na brzegu obszaru stab.
        z1=z0-(1+z0-exp(-j*fi(n)));
        z0=z1;
    end
    h_lambda(n)=z0;
end

h=plot(real(h_lambda),imag(h_lambda),'g');
set(h,'LineWidth',1.7);
axis([-3 0.5,-3.1,3.1]);
axis equal
title('Obszary absolutnej stabilności dla metod Rungego-Kutty');
hold on
h=text(-1.25,0.2,'K');
set(h,'Color','g','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-1.05,0.2,'=1');
set(h,'Color','g','FontName','Times','FontSize',12,'FontAngle','normal');

% Metoda Rungego-Kutty rzędu K=2
%-----
fi=linspace(0,4*pi,1000); % faza bieżącego punktu na brzegu obszaru stabilności
N=size(fi,1);
h_lambda=zeros(N,1)+j*zeros(N,1); % pierwszy punkt na brzegu obszaru stabilności

```

```

for n=2:N
    z0=hlambda(n-1); % pierwsze przybliżenie n-tego punktu na brzegu obszaru stab.
    for k=1:10 % iteracyjne poprawianie n-tego punktu na brzegu obszaru stab.
        z1=z0-(1+z0+0.5*z0^2-exp(-j*fi(n)))/(1+z0);
        z0=z1;
    end
    hlambda(n)=z0;
end

h=plot(real(hlambda),imag(hlambda),'b');
set(h,'LineWidth',1.7);
hold on
h=text(-1.25,1.35,'K');
set(h,'Color','b','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-1.05,1.35,'=2');
set(h,'Color','b','FontName','Times','FontSize',12,'FontAngle','normal');

% Metoda Rungego-Kutty rzędu K=3
%-----
fi=linspace(0,6*pi,1000); % faza bieżącego punktu na brzegu obszaru stabilności
N=size(fi,1);
hlambda=zeros(N,1)+j*zeros(N,1); % pierwszy punkt na brzegu obszaru stabilności
for n=2:N % iteracyjne poprawianie n-tego punktu na brzegu obszaru stabilności
    z0=hlambda(n-1); % pierwsze przybliżenie n-tego punktu na brzegu obszaru stab.
    for k=1:10
        z1=z0-(1+z0+0.5*z0^2+(1/6)*z0^3-exp(-j*fi(n)))/(1+z0+0.5*z0^2);
        z0=z1;
    end
    hlambda(n)=z0;
end

h=plot(real(hlambda),imag(hlambda),'r');
set(h,'LineWidth',1.7);
hold on
h=text(-1.05,2.05,'K');
set(h,'Color','r','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-0.85,2.05,'=3');
set(h,'Color','r','FontName','Times','FontSize',12,'FontAngle','normal');

% Metoda Rungego-Kutty rzędu K=4
%-----
fi=linspace(0,8*pi,1000); % faza bieżącego punktu na brzegu obszaru stabilności
N=size(fi,1);
hlambda=zeros(N,1)+j*zeros(N,1); % pierwszy punkt na brzegu obszaru stabilności
for n=2:N
    z0=hlambda(n-1); % pierwsze przybliżenie n-tego punktu na brzegu obszaru stab.
    for k=1:10 % iteracyjne poprawianie n-tego punktu na brzegu obszaru stab.
        z1=z0-(1+z0+0.5*z0^2+(1/6)*z0^3+(1/24)*z0^4-exp(-j*fi(n)))/...
            (1+z0+0.5*z0^2+(1/6)*z0^3);
        z0=z1;
    end
    hlambda(n)=z0;
end

h=plot(real(hlambda),imag(hlambda),'k');
set(h,'LineWidth',1.7);
h=text(-0.65,2.65,'K');
set(h,'Color','k','FontName','Times','FontSize',12,'FontAngle','italic');
h=text(-0.45,2.65,'=4');
set(h,'Color','k','FontName','Times','FontSize',12,'FontAngle','normal');
line([-5.2,2.8],[0,0],'Color','k')
line([0,0],[-3.1,3.1],'Color','k')

```

```

h xlabel('Im[i\th\lambda\rm]');
set(h,'FontName','Times','FontSize',12);
h ylabel('Re[i\th\lambda\rm] ');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);

=====

Zadanie 9.4

clear all; close all; clc;
global licznik
licznik=0;

% ROZWIĄZYwanIE UKŁADU RÓWNAŃ Z PRZYKŁADU 9.1 ZA POMOCĄ
% ZMIENNOKROKOWEJ IMPLEMENTACJI OTWARTEJ I ZAMKNIĘTEJ METODY ADAMSA (K=2)
%-----

% Dokładne rozwiązanie - u(t)
%-----
u=@(t,tau) 1-exp(-t/tau); tau=1;

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-100; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figy figx figy
        figx+15 100+figy figx figy
        5 5 figx figy
        figx+15 5 figx figy]; % położenia rysunków
figaxis=[0 5 1e-6 4e-4;0 5 1e-6 1e-3;0 5 1e-3 1e-0];

% Zmiennokrokowa implementacja otwartej metody Adamsa (K=2)
%-----
clear t
T=5;
tau=1;
ho=5e-3; % początkowa długość kroku h
t(1)=0; t(2,1)=5e-3;
uo(1)=0; uo(2,1)=uo(1)+ho*f(t(1),uo(1)); % przybliżony warunek początkowy
i=2;
emax=1e-3;
p=2;
while t<=T
    y1=uo(i)+ho*(3*f(t(i),uo(i))-f(t(i-1),uo(i-1))/2; % wartość dla kroku h
    Y2=uo(i)+ho/2*(3*f(t(i),uo(i))-f(t(i-1),uo(i-1))/2; % wartość dla 1 kroku h/2
    Y2=y2+ho/2*(3*f(t(i)+ho/2,y2)-f(t(i),uo(i))/2; % wartość dla 2-go kroku h/2
    Deltan1=y1-y2;
    rn1=Deltan1/(2^p-1); % oszacowanie błędu lokalnego wg wzoru (9.36)
    rmax=ho/T*emax; % oszacowanie dopuszczalnego błędu lokalnego wg wzoru (9.37)
    if abs(rn1)<=rmax % jeżeli bieżące rozwiązanie jest wystarczająco dokładne:
        i=i+1;
        uo(i,1)=y2; % zapamiętanie wyniku
        t(i,1)=(i-1)+ho;
        if abs(rn1)<2^(-p)*rmax % jeżeli błąd jest dostatecznie mały
            ho=ho*2; % zwiększenie kroku
        end
    else % jeżeli bieżące rozwiązanie nie jest wystarczająco dokładne:
        ho=ho/2; % zmniejszenie kroku
    end
end

```

```

% Prezentacja wyników
%-----
N=length(t);
Duo=abs(uo-u(t,tau));
duo=Duo./u(t,tau);
duo(1)=0;
ff=figure(1);
hold off
set(ff,'Position',figpos(1,:));
semilogy(t,Duo)
hold on
ff=figure(2);
hold off
set(ff,'Position',figpos(2,:));
semilogy(t,duo)
hold on
ff=figure(3);
hold off
set(ff,'Position',figpos(3,:));
semilogy(t(2:length(t)),diff(t))
hold on

fprintf('\nMetoda zmiennokrokowa Adamsa-Bashfortha K=2:\n');
fprintf('Liczba odwołań do funkcji f(t,u) : %4i\n',licznik)
fprintf('Maksymalny błąd bezwzględny u(t) : %10.3e\n',max(Duo));
fprintf('Maksymalny błąd względny u(t) : %10.3e\n',max(duo));

% Zmiennokrokowa implementacja zamkniętej metody Adamsa (K=2)
% W zadaniu tym zrealizowano zamkniętą metodę Adamsa dla szczególnej postaci
% funkcji f(t,u)=1-u, co pozwala na uniknięcie wyznaczania w każdym kroku
% rozwiązania za pomocą procedury fzero (w przypadku skalarnym) lub fsolve
% (w przypadku wektorowym). W zadaniu 9.5 zrealizowano metodę zamkniętą dla
% ogólnej (skalarnej) postaci funkcji f(t,u).
%-----
clear t
T=5;
hz=5e-3;
t=[0:5e-3];
uz=[0;0];
i=1;
emax=1e-3;
p=2;
licznik=0;
while t<=T
    y1=(uz(i)*(1-hz/2)+hz)/(1+hz/2); % wartość dla kroku h
    y2=(uz(i)*(1-hz/4)+hz/2)/(1+hz/4); % wartość dla pierwszego kroku h/2
    y2=(y2*(1-hz/4)+hz/2)/(1+hz/4); % wartość dla drugiego kroku h/2
    licznik=licznik+6; % zliczanie liczby wywołań funkcji
    Deltan1=y1-y2;
    rn1=Deltan1/(2^p-1); % oszacowanie błędu lokalnego wg wzoru (9.36)
    rmax=hz/T*emax; % oszacowanie dopuszczalnego błędu lokalnego wg wzoru (9.37)
    if abs(rn1)<=rmax % jeżeli bieżące rozwiązanie jest wystarczająco dokładne:
        i=i+1;
        uz(i,1)=y2; % zapamiętanie wyniku
        t(i)=t(i-1)+hz;
        if abs(rn1)<2^(-p)*rmax % jeżeli błąd jest dostatecznie mały
            hz=hz*2; % zwiększenie kroku
        end
    else % jeżeli bieżące rozwiązanie nie jest wystarczająco dokładne:
        hz=hz/2; % zmniejszenie kroku
    end
end

```

```

% Prezentacja wyników
%-----
N=length(t);
Duz=abs(uz-u(t,tau));
duz=Duz./u(t,tau);
duz(1)=0;
f=figure(1);
set(f,'Position',figpos(1,:));
semilogy(t,Duz,'r')
axis(figaxis(1,:));
h=xlabel('\it{t}');
set(h,'FontName','Times','FontSize',12);
h=ylabel('\Delta|u|tu');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Błąd bezwzględny'])
legend('metoda Adamsa-Bashfortha','metoda Adamsa-Moultona',4)
figure(2);
semilogy(t,duz,'r')
axis(figaxis(2,:));
h=xlabel('\it{t}');
set(h,'FontName','Times','FontSize',12);
h=ylabel('|\delta|u|tu ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
title(['Błąd względny'])
legend('metoda Adamsa-Bashfortha','metoda Adamsa-Moultona',4)
figure(3);
semilogy(t(2:length(t)),diff(t),'r')
axis(figaxis(3,:));
h=xlabel('\it{t}');
set(h,'FontName','Times','FontSize',12);
h=ylabel('|\delta|u|tu ');
set(h,'Rotation',0,'FontName','Times','FontSize',12);
legend('metoda Adamsa-Bashfortha','metoda Adamsa-Moultona',4)
title(['Długość kroku'])

fprintf('\nMetoda zmiennokrokowa Adamsa-Moultona K=2:\n');
fprintf('Liczba odwołań do funkcji f(t,u) : %3i\n',licznik)
fprintf('Maksymalny błąd bezwzględny u(t) : %10.3e\n',max(Duz));
fprintf('Maksymalny błąd względny u(t) : %10.3e\n',max(duz));

%-----

function uprim=f(t,u)
% Wyznaczanie wartości prawej strony równania różniczkowego (9.2)
% t - czas, u - napięcie na kondensatorze
% e(t) - siła elektromotoryczna w voltach odwołanie do zdefiniowanej funkcji "e"
global licznik % zmienna globalna do liczenia wywołań funkcji
licznik=licznik+1;
C=1; % pojemność w mikrofaradach
R=1; % rezystancja w kiloomach
uprim=(e(t)-u)/R/C; % wzór wynikający z zerowania się sumy napięć w obwodzie
% i równań elementowych: u(t)=R*i(t) dla rezystora
% oraz i(t)=C(u(t))u'(t) dla kondensatora

%-----

function E=e(t)
% Wyznaczanie wartości siły elektromotorycznej; t - czas
E=zeros(size(t));
idx=find(t>=0);
E(idx)=1;

%
```

Zadanie 9.5

```

clear all; close all; clc; warning off

% ROZWIAZANIE UKŁADU RÓWNAŃ NIELINIOWEGO UKŁADU
% PRZY UŻYCIU SCHEMATU PREDYKTOR-KOREKTOR
% OPARTEGO NA PARZE METOD ADAMSA RZĘDU 1 I 2 ZE STEROWANIEM DŁUGOŚCIĄ KROKU
%=====

% Parametry rysunków
%-----
figsize=get(0,'ScreenSize'); % identyfikacja wielkości ekranu
figx=figsize(3)/2-10; if figx>630, figx=630;end % szerokość rysunków
figy=figsize(4)/2-10; if figy>412, figy=412;end % wysokość rysunków
figpos=[5 100+figx figx figy
        figx+15 100+figy figx figy
        5 5 figx figy
        figx+15 5 figx figy]; % położenia rysunków
figaxis=[0 5 0 1.02;0 5 -0.5e-4 1.9e-4;0 5 0 1.02;0 5 0 8e-5];

% Parametry rozwiązań numerycznych
%-----
adams_moulton1=@(un1,un0,tn1,tn0) un1-un0-(tn1-tn0)*f(tn1,un1); % schemat metody
% Adamsa-Moultona 1 rzędu odwołujący się do funkcji f prawej strony równania
u=zeros(2,1);
t=zeros(2,1);
h=1e-3*ones(2,1);
k=1;
T=5;
options=optimset('TolX',1e-6,'Display','off');
p=1;
ro=0.5;
emax=1e-3;
while(t(k)<T)
    t(k+1)=t(k)+h(k);
    u_pre=u(k)+h(k)*f(t(k),u(k)); % predyktor
    u_kor=fzero(adams_moulton1,[u_pre],options,u(k),t(k+1),t(k)); % korektor
    Deltan1=u_kor-u_pre;
    rn1=Deltan1/(2^p-1);
    rmax=h(k)/T*emax; % oszacowanie błędu wzór (9.37)
    if abs(rn1)<=rmax % jeżeli wynik wystarczająco dokładny
        k=k+1;
        u(k)=u_kor; % zapamiętanie wyniku
        t(k)=t(k-1)+h(k-1);
        h(k)=h(k-1);
        if abs(rn1)<=rmax*ro % jeżeli warunek wydłużenia kroku spełniony to
            h(k)=h(k)/max(1/10,(abs(rn1)/rmax)^(1/(p+1))); % wydłużenie kroku
        end
    else % zmniejszenie kroku
        h(k)=h(k)/min(10,(abs(rn1)/rmax)^(1/(p+1)));
    end
end
options=odeset('RelTol',1e-9,'AbsTol',1e-9);
[t,urk23]=ode23(@f,t,0,options); % rozwiązanie metodą RK23 dla tych samego t

% Prezentacja wyników
%-----
ff=figure(1);
set(ff,'Position',figpos(1,:));
hold off
hh=plot(t,urk23);

```

```

set(hh,'LineWidth',2);
hold on
hh=plot(t,u,:r');
set(hh,'LineWidth',2);
hh=ylabel('\it{u}');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
hh=xlabel('\it{t}');
set(hh,'FontName','Times','FontSize',12);
axis(figaxis(1,:));
legend('procedura „ode23”','predyktor-korektor rzędu 1',0);
title('Rozwiązań przybliżone')
ff=figure(2);
set(ff,'Position',figpos(3,:));
hold off
plot(t,u-urk23);
hh=ylabel('\Delta\it{u}');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
hh=xlabel('\it{t}');
set(hh,'FontName','Times','FontSize',12);
axis(figaxis(2,:));
title('Błąd bezwzględny rozwiązania')
pause(0.1);

% Powtórzenie eksperymentu dla metod Adamsa rzędu 2
%-----
adams_moulton2=@(un1,un0,tn1,tn0) un1-un0-(tn1-tn0)*(f(tn0,un0)+f(tn1,un1))/2;
u=u(1:2,1);
t=t(1:2,1);
h=h(1:2,1);
k=2;
T=5;
options=optimset('TolX',1e-6,'Display','off');
p=2;
ro=0.5;
emax=1e-3;
while(t(k)<T)
    t(k+1)=t(k)+h(k);
    u_pre=u(k)+h(k)*(3*f(t(k),u(k))-f(t(k-1),u(k-1)))/2; % predyktor
    u_kor=fzero(adams_moulton2,[u_pre],options,u(k),t(k+1),t(k)); % korektor
    Deltan1=u_kor-u_pre;
    rn1=Deltan1/(2^p-1);
    rmax=h(k)/T*emax; % oszacowanie błędu wzór (9.37)
    if abs(rn1)<=rmax % jeżeli wynik wystarczająco dokładny
        k=k+1;
        u(k)=u_kor; % zapamiętanie wyniku
        t(k)=t(k-1)+h(k-1);
        h(k)=h(k-1);
        if abs(rn1)<=rmax*ro % jeżeli warunek wydłużenia kroku spełniony to
            h(k)=h(k)/max(1/10,(abs(rn1)/rmax)^(1/(p+1))); % wydłużenie kroku
        end
    else % zmniejszenie kroku
        h(k)=h(k)/min(10,(abs(rn1)/rmax)^(1/(p+1)));
    end
end
options=odeset('RelTol',1e-12,'AbsTol',1e-12);
[t,urk23]=ode23(@f,t,0,options); % rozwiązanie metodą RK23 dla tego samego t

% Prezentacja wyników
%-----
ff=figure(3);
set(ff,'Position',figpos(2,:));

```

```

hold off
hh=plot(t,urk23);
set(hh,'LineWidth',2);
hold on
hh=plot(t,u,:r');
set(hh,'LineWidth',2);
hh ylabel ('\it{u}');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
hh xlabel ('\it{t}');
set(hh,'FontName','Times','FontSize',12);
axis(figaxis(3,:));
legend('procedura „ode23”','predyktor-korektor rzedu 2',0)
title('Rozwiazanie przyblizone')
ff=figure(4);
set(ff,'Position',figpos(4,:));
hold off
plot(t,u-urk23)
hh ylabel ('\Delta\it{u}');
set(hh,'Rotation',0,'FontName','Times','FontSize',12);
hh xlabel ('\it{t}');
set(hh,'FontName','Times','FontSize',12);
axis(figaxis(4,:));
title('Błęd bezwzględny rozwiązania')

=====

function uprim=f(t,u)
% Wyznaczanie wartości prawej strony równania różniczkowego (9.2)
% t - czas, u - napięcie na kondensatorze
% e(t) - siła elektromotoryczna w woltach odwołanie do zdefiniowanej funkcji „e”
global licznik % zmenna globalna do liczenia wywołań funkcji
licznik=licznik+1;
C=1; % pojemność w mikrofaradach
R=1; % rezystancja w kiloomach
uprim=(e(t)-u)/R/C; % wzór wynikający z zerowania się sumy napięć w obwodzie
% i równań elementowych: u(t)=R*i(t) dla rezystora
% oraz i(t)=C(u(t))u'(t) dla kondensatora

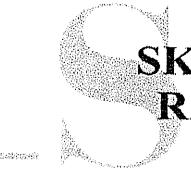
=====

function E=e(t)
% Wyznaczanie wartości siły elektromotorycznej; t - czas
E=zeros(size(t));
idx=find(t>=0);
E(idx)=1;

```

LITERATURA

- [B] Błażewicz J.: *Złożoność obliczeniowa problemów kombinatorycznych*, WNT, Warszawa 1988.
- [C] Chua L.O., Pen-Min Lin: *Komputerowa analiza układów elektronicznych*, WNT, Warszawa 1981.
- [D1] Dahlquist G., Björck A.: *Metody numeryczne*, PWN, Warszawa 1983.
- [D2] Dryja M., Jankowscy J. i M.: *Przegląd metod i algorytmów numerycznych – część 2*, WNT, Warszawa 1982.
- [F1] Fisz M.: *Rachunek prawdopodobieństwa i statystyka matematyczna*, PWN, Warszawa 1969.
- [F2] Fortuna Z., Macukow B., Wąsowski J.: *Metody numeryczne*, WNT, Warszawa 1982.
- [J1] Jankowscy J. i M.: *Przegląd metod i algorytmów numerycznych – część 1*, WNT, Warszawa 1981.
- [J2] Jermakow S.M.: *Metoda Monte Carlo i zagadnienia pokrewne*, PWN, Warszawa 1976.
- [K] Kielbasiński A., Schwetlick H.: *Numeryczna algebra liniowa*, WNT, Warszawa 1992.
- [M] Mrozek B., Mrozek Z.: *MATLAB 6 – poradnik użytkownika*, Wyd. PLJ, Warszawa 2001.
- [O] Ogrodzki J.: *Komputerowa analiza układów elektronicznych*, PWN, Warszawa 1994.
- [P] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P., *Numerical Recipes in C*, Cambridge University Press, Cambridge 1992.
- [S1] Stoer J., Bulirsch R.: *Wstęp do metod numerycznych*, PWN, Warszawa 1984.
- [S2] Stybliński M.A.: *Metody analizy i optymalizacji tolerancji parametrów układów elektronicznych*, WNT, Warszawa 1981.
- [T] Traub J.F., Wasilkowski T., Woźniakowski H.: *Information, Uncertainty, Complexity*, Addison-Wesley, Reading 1983.
- [W] Wieczorkowski R., Zieliński R.: *Komputerowe generatory liczb losowych*, WNT, Warszawa 1997.
- [Z1] Zalewski A., Cegieła R.: *MATLAB – obliczenia numeryczne i ich zastosowania*. Wyd. Naukowe, Poznań 1997.
- [Z2] Zieliński R.: *Metody Monte Carlo*, WNT, Warszawa 1970.
- [Ż] Żakowski W., Leksiński W.: *Matematyka – część IV*, Seria „Podręczniki Akademickie – Elektronika”, WNT, Warszawa 1995.



SKOROWIDZ RZECZOWY

- Algorytm 13, 25–56, 222
 - Hornera 26, 27
 - iteracyjny 50–64
 - jednoargumentowy 52, 53, 55, 57
 - stacjonarny 50
 - wieloargumentowy 57, 58
 - wielomianowy 31, 32
 - wykładowczy 31, 32
- Aproksymacja funkcji 14–16, 22, 25, 55, 244, 245
 - jednostajna 120
 - Padégo 121
 - średniokwadratowa 110
- Automatyczny wybór kroku całkowania 188, 210
- Automatyczny wybór rzędu metody całkowania 210
- Automatyczny wybór położenia węzłów kwadratury 133
- Błąd bezwzględny 39, 51, 58, 256, 261, 262, 277–283, 294, 295, 297, 298
- Błąd względny 28–30, 33, 34, 38, 225, 252, 257–260, 270, 275–282, 294, 295
- Całkowanie funkcji 14
 - jednej zmiennej 14, 250, 253, 255, 256, 259, 260
 - wielu zmiennych 145, 146
- Dokładność obliczeń 27–64
- Dyskretyzacja równań różniczkowych 212
- Efekt Rungego 105, 129, 133, 139
- Eliminacja Gaussa 75
- Estymata
 - wariancji zmiennej losowej 161, 164, 165, 169, 170, 267, 268

- wartości oczekiwanej zmiennej losowej 161, 267
- Estymator
 - wariancji zmiennej losowej 160
 - wartości oczekiwanej zmiennej losowej 160
- Funkcja
 - gestości prawdopodobieństwa (f.g.p.) 160, 161, 163, 167, 169, 173, 174
 - wagowa kwadratury 124
 - złożoności obliczeniowej 30–32
- Funkcje ortogonalne 111, 240
- Generacja zmiennej losowej 171
- Generatorzy liczb pseudolosowych 172
- Globalny błąd dyskretyzacji 180, 181
- Homomorfizm 18, 19, 23
- Identyfikacja modelu matematycznego 20, 21–23
 - parametryczna 20, 21–23
 - strukturalna 20, 21, 23
- Ilorazy różnicowe 103
- Interpolacja funkcji 236–238, 244, 245, 252, 253
 - funkcjami sklejonymi 105, 238, 246, 247
 - Lagrange'a 102, 252
 - trygonometryczna 107, 238, 240
- Komputer 216, 218
- Krok całkowania 178, 277, 278, 280, 282, 293–297
- Kwadratura interpolacyjna
 - Gaussa 256, 257
 - złożona 256, 258
 - Gaussa-Czebyszewa 140, 142
 - Gaussa-Konroda 137
 - Gaussa-Legendre'a 137

- liniowa 125
- Newtona-Cotesa 125, 250–252, 254
- złożona 253–255, 260–262
- Richardsona 141
- Simsona 126, 133, 140
- trapezów 126, 129, 140
- Liczby pseudolosowe 220
- Liniowe zadanie najmniejszych kwadratów 84
- Liniowe zadanie testowe 200
- Liniowy model przenoszenia błędów 23–49
- Lokalny błąd obcięcia (metody całkowania) 180, 181, 189
- Macierz
 - diagonalnie słabo dominująca 68
 - dodatnio określona 69
 - hermitowsko sprzężona 68
 - Hilberta 73, 219
 - redukowalna 68, 217
 - trójkątna górną (dolna) 77, 221–223
 - Vandermonde'a 73, 220
- Metoda
 - Adamsa-Bashfortha 197, 203, 288, 289, 293–295
 - Adamsa-Moultona 197, 204, 288–290, 295–297
 - bisekcji 89, 227, 228
 - całkowania
 - otwarta 186
 - zamknięta 186
 - ekstrapolacji Richardsona 140, 153, 264, 265
 - eliminacji Gaussa 75
 - Eulera
 - otwarta 179, 204, 275–277, 282–285
 - zamknięta 179, 204, 207
 - zmodyfikowana 185
 - Gaussa-Seidla 81, 222
 - Geara 207, 209, 286, 287
 - Heuna 186, 275
 - iteracyjna 222
 - Jacobiego 79
 - jednokroka 178, 195, 276
 - losowania ważonego 167, 168
 - Monte Carlo 157
 - prosta 163
 - podstawowa 158, 163, 269, 272
 - orzeł-rezka 159, 164, 268–272
 - złożona 167
 - losowania warstwowego 171
 - losowania ważonego 167
- Nieliniowe równanie algebraiczne 89
- Nierówność Czebyszewa 120
- Norma indukowana macierzy 71
- Norma wektora
 - druga (Euklidesowa) 71
 - nieskończoność 71
 - pierwsza 71
- Numeryczna poprawność algorytmu 43, 44
- Obliczanie całek
 - niewłaściwych 142
 - z osobliwościami 143
- Odchylenie standardowe zmiennej losowej 267
- Ortogonalizacja Grama-Schmidta 112
- Osiagalna dokładność 52, 55–57, 62–64, 112
- Parametry lokalnej zbieżności 52–54, 57
- Pomiar 15, 16, 21–25, 32, 37, 64
- Projektowanie 13, 17, 22, 23
- Przenoszenie błędów 30, 33–35, 37, 38, 44–46, 56
 - danych 37, 55
 - zaokrągleń 41, 44–46
- Przestrzeń
 - liniowa 111
 - unormowana 111
- Punkt zbieżności algorytmu iteracyjnego 51, 52
- Rachunek epsilonów 35, 38, 39, 47, 48, 55, 62
- Reprezentacja liczb w komputerze 28, 29
 - stałopozycyjna 28
 - zmiennopozycyjna 28, 29

- Reprezentacyjna teoria pomiaru 23
- Reszta kwadratury 125, 126, 130, 136
- Rozkład macierzy
 - LU 77, 221
 - QR 85, 223, 224
 - SVD 84
- Rozrzuty produkcyjne parametrów 147, 165
- Równanie różniczkowe zwyczajne 176
- Różnica
 - centralna 148, 152, 156, 263, 264, 266
 - progresywna 147, 155, 263, 265
 - wsteczna 147
- Różniczkowanie funkcji jednej zmiennej 147
- Różniczkowanie formuł interpolacyjnych 151
- Różniczkowanie formuł aproksymacji wygła-
dzającej 155
- Rząd
 - kwadratury 125
 - metody całkowania 180, 184, 276, 286–288,
291, 292, 296–298
- Sieć działań 26, 27
- Stabilność numeryczna liniowej metody różnic-
owej 198, 282
 - absolutna 203, 288–292
 - A-stabilność 203
 - S-stabilność 207
 - w sensie Dahlquista 202
- Stwarzyszony model dyskretny pojemności
213
- Szybka transformacja Fouriera 108, 239, 240,
243
- Układ funkcji
 - ortogonalny 112
 - ortonormalny 112
- Układ normalny równań 116
- Układ równań
 - algebraicznych 67, 219–226, 229, 231–235
 - różniczkowo-algebraicznych 212, 213
 - różniczkowych 276, 279, 282, 284, 286, 293,
296
 - różniczkowych – sztywny 206
- Uwarunkowanie numeryczne zadania 40
- Uzysk 165

- Wariancja zmiennej losowej 160
- Wartości własne macierzy 69, 222
- Wartość oczekiwana zmiennej losowej 160,
267
- Warunek aproksymacji 184
- Wektor przeszłości 213
- Wektor własny macierzy 69
- Wielomian
 - charakterystyczny 69, 201
 - Czebyszewa 113, 139, 142, 239, 240
 - Hermite'a 113, 222
 - Lagrange'a 102, 252
 - Legendre'a 103, 241, 242, 257
 - Newtona 104
- Wielomianowa funkcja sklejana 105, 238, 246,
247
- Wielowymiarowa metoda
 - Newtona 97, 233
 - siecznych 98
- Wskaźnik uwarunkowania macierzy 72, 219
- Współczynnik
 - lokalnej zbieżności 52, 53, 61, 64
 - przenoszenia błędów danych 33–35, 37, 38,
45
 - przenoszenia błędów zaokrągleń 41, 45, 46
- Wybór
 - kroku całkowania 188–190
 - rzędu metody wielokrokowej 210–212
- Wykładnik lokalnej zbieżności 52, 53, 61, 63,
64
- Zagadnienie początkowe (Cauchy'ego) 176
- Zokrąglanie wyników obliczeń w komputerze
28–30, 35, 36, 39, 41–43, 45, 46, 52, 55, 56,
62
- Zapis sekwencyjny algorytmu 26, 27, 30
- Zasada Rungego 188
- Zbieżność
 - algorytmu iteracyjnego 51, 52, 57, 59–63
 - kwadratury 125
 - różnicowej metody całkowania 179, 182
- Złożoność obliczeniowa 30–32

