JERZY KLAMKA
ZBIGNIEW OGOŃOWSKI
MICHAŁ JAMICKI
MARIUSZ STASIK

METODY NUMERYCZNE

WYDAWNICTWO POLITECHNIKI ŚLĄSKIEJ

powered by
Misiol

SKRYPT 2068

Jerzy Klamka, Zbigniew Ogonowski,
Michał Jamicki, Mariusz Stasik

METODY NUMERYCZNE

WYDAWNICTWO POLITECHNIKI ŚLĄSKIEJ
GLIWICE 1998

OPINIODAWCA
Dr hab. inż. Andrzej Królikowski -
Profesor Politechniki Poznańskiej

KOLEGIUM REDAKCYJNE

REDAKTOR NACZELNY – Prof. dr hab. Zygmunt Kleszczewski
REDAKTOR DZIAŁU – Dr inż. Anna Skrzywan-Kosek
SEKRETARZ REDAKCJI – Mgr Elżbieta Leśko

REDAKCJA
Mgr Kazimiera Szafir

REDAKCJA TECHNICZNA
Alicja Nowacka

Wydano za zgodą
Rektora Politechniki Śląskiej

Skrypt przeznaczony dla studentów Wydziału
Automatyki, Elektroniki i Informatyki

PL ISSN 0434 - 0825

Wydawnictwo Politechniki Śląskiej
ul. Kujawska 3, 44 - 100 Gliwice

Nakł. 500 + 55 Ark. wyd. 12 Ark. druk. 12 Papier offset. kl. III 70 x 100, 80 g.
Oddano do druku 27.01.1998 r. Podpis. do druku 27.01.1998 r. Druk ukończ. w lutym 1998 r.

Fotokopie, druk i oprawę wykonał „ROLEK", Gliwice, ul. Kazimierza Wielkiego 4

powered by
Misiol

Spis treści

1. Podstawy teoretyczne podstawowych metod numerycznych 7
1.1. Teoria błędów . . . 7
1.1.1. Definicje i pojęcia związane z błędami . . . 7
1.1.2. Podstawowe źródła błędów . . . 8
1.1.3. Metody obliczania błędów . . . 10
1.1.4. Błędy działań arytmetycznych na liczbach przybliżonych . . . 13
1.1.5. Problem odwrotny teorii błędów . . . 15
1.2. Obliczanie wartości funkcji . . . 16
1.2.1. Obliczanie wartości wielomianu. Schemat Hőrnera . . . 16
1.2.2. Obliczanie wartości funkcji analitycznej . . . 19
1.2.3. Iteracyjne obliczanie wartości funkcji . . . 20
1.3. Interpolacja funkcji . . . 22
1.3.1. Sformułowanie zagadnienia interpolacji . . . 22
1.3.2. Wzór interpolacyjny Lagrange'a . . . 25
1.3.3. Operatory różnicowe . . . 27
1.3.4. Wzory interpolacyjne Newtona . . . 30
1.3.5. Optymalny dobór węzłów interpolacji . . . 35
1.3.6. Wzory interpolacyjne z operatorami różnicy symetrycznej . . . 36
1.3.7. Przykład . . . 38
1.3.8. Uwagi końcowe . . . 41
1.4. Różniczkowanie numeryczne . . . 41
1.4.1. Sformułowanie problemu . . . 41
1.4.2. Wzór Taylora . . . 42
1.4.3. Wzór Stirlinga . . . 45
1.4.4. Podsumowanie . . . 49
1.5. Całkowanie numeryczne . . . 49
1.5.1. Wprowadzenie . . . 49
1.5.2. Metoda Newtona-Cotesa . . . 50
1.5.3. Metoda Czebyszewa . . . 54
1.5.4. Kwadratury złożone . . . 55
1.5.5. Podsumowanie . . . 57
1.6. Aproksymacja . . . 58
1.6.1. Sformułowanie problemu aproksymacji . . . 58
1.6.2. Aproksymacja średniokwadratowa . . . 58
1.6.3. Aproksymacja jednostajna . . . 62
1.6.4. Aproksymacja punktowa . . . 65
1.7. Metody rozwiązywania układów równań liniowych . . . 68
1.7.1. Wprowadzenie . . . 68
1.7.2. Metody dokładne . . . 70
1.7.3. Metody przybliżone . . . 76
1.7.4. Błędy rozwiązań układów równań liniowych . . . 80
1.7.5. Uwagi . . . 83
1.8. Wyznaczanie wartości własnych i wektorów własnych macierzy . . . 84

1.8.1. Wprowadzenie . . . 84
1.8.2. Metoda Kryłowa . . . 85
1.8.3. Iteracyjne wyznaczanie wartości własnej o największym module . . 89
1.8.4. Metoda iteracyjna wyznaczania wartości własnych i wektorów własnych macierzy symetrycznej dodatnio określonej . . . 93
1.9. Przybliżone rozwiązywanie układów równań nieliniowych . . . 97
1.9.1. Wprowadzenie . . . 97
1.9.2. Metoda połowienia przedziałów . . . 97
1.9.3. Metoda siecznych . . . 100
1.9.4. Metoda stycznych (Newtona) . . . 103
1.9.5. Metoda Bernoulliego . . . 105
1.10. Przybliżone rozwiązywanie równań różniczkowych zwyczajnych . . . 108
1.10.1. Wprowadzenie . . . 108
1.10.2. Sformułowanie zadania przybliżonego rozwiązywania równań różniczkowych zwyczajnych . . . 108
1.10.3. Metoda Eulera . . . 109
1.10.4. Metoda Rungego-Kutty . . . 111
1.10.5. Rozwiązywanie układów równań różniczkowych . . . 114
1.10.6. Rozwiązywanie równań różniczkowych wyższych rzędów . . . 114
1.10.7. Metoda szeregów Taylora . . . 115
1.10.8. Metoda Picarda . . . 117
1.11. Równania różniczkowe cząstkowe typu parabolicznego . . . 118
1.11.1. Zagadnienie Dirichleta dla równania parabolicznego . . . 118
1.11.2. Metoda siatek . . . 119
1.11.3. Algorytm rozwiązywania przybliżonego . . . 121
1.11.4. Błąd metody . . . 122
1.11.5. Przykład . . . 122
1.12. Równania całkowe typu Fredholma i Volterry . . . 124
1.12.1. Równania Fredholma i Volterry . . . 124
1.12.2. Równanie Volterry z jądrem splotowym . . . 125
1.12.3. Równanie różniczkowe zwyczajne liniowe pierwszego rzędu ze zmiennymi współczynnikami . . . 125
1.12.4. Metoda Neumanna . . . 126
1.12.5. Przykłady . . . 128

2. Narzędzia programowania metod numerycznych 130
2.1. Sprzęt . . . 130
2.2. Języki programowania . . . 130
2.2.1. Język Fortran . . . 131
2.2.2. Język Pascal . . . 138
2.2.3. Język C . . . 146
2.3. Biblioteki metod numerycznych . . . 153
2.3.1. Biblioteki NAG . . . 153
2.3.2. Biblioteki Numerical Recipes . . . 155
2.3.3. Biblioteki IMSL . . . 157
2.3.4. Biblioteki Harwell . . . 159

powered by Misiol

2.4. Pakiety programowe . . . 160
2.4.1. Matlab . . . 167
2.4.2. Mathematica . . . 169

3. Ćwiczenia laboratoryjne 172
3.1. Błędy obliczeń numerycznych . . . 172
3.1.1. Cel ćwiczenia . . . 172
3.1.2. Zestaw pytań kontrolnych . . . 173
3.1.3. Tematy zadań do wykonania w trakcie ćwiczenia . . . 173
3.1.4. Uwagi dotyczące sprawozdania . . . 174
3.2. Metody numerycznego obliczania wartości funkcji . . . 174
3.2.1. Cel ćwiczenia . . . 174
3.2.2. Zestaw pytań kontrolnych . . . 174
3.2.3. Tematy zadań do wykonania w trakcie ćwiczenia . . . 175
3.2.4. Uwagi dotyczące sprawozdania . . . 175
3.3. Metody interpolacji . . . 176
3.3.1. Cel ćwiczenia . . . 176
3.3.2. Zestaw pytań kontrolnych . . . 176
3.3.3. Tematy zadań do wykonania w trakcie ćwiczenia . . . 177
3.3.4. Uwagi dotyczące sprawozdania . . . 177
3.4. Metody różniczkowania numerycznego . . . 178
3.4.1. Cel ćwiczenia . . . 178
3.4.2. Zestaw pytań kontrolnych . . . 178
3.4.3. Tematy zadań do wykonania w trakcie ćwiczenia . . . 178
3.4.4. Uwagi dotyczące sprawozdania . . . 178
3.5. Metody całkowania numerycznego . . . 179
3.5.1. Cel ćwiczenia . . . 179
3.5.2. Zestaw pytań kontrolnych . . . 179
3.5.3. Tematy zadań do wykonania w trakcie ćwiczenia . . . 179
3.5.4. Uwagi dotyczące sprawozdania . . . 180
3.6. Metody aproksymacji . . . 180
3.6.1. Cel ćwiczenia . . . 180
3.6.2. Zestaw pytań kontrolnych . . . 180
3.6.3. Tematy zadań do wykonania w trakcie ćwiczenia . . . 181
3.6.4. Uwagi dotyczące sprawozdania . . . 181
3.7. Metody rozwiązywania układów równań liniowych . . . 182
3.7.1. Cel ćwiczenia . . . 182
3.7.2. Zestaw pytań kontrolnych . . . 182
3.7.3. Tematy zadań do wykonania w trakcie ćwiczenia . . . 183
3.7.4. Uwagi dotyczące sprawozdania . . . 183
3.8. Numeryczne obliczanie wartości i wektorów własnych macierzy . . . 183
3.8.1. Cel ćwiczenia . . . 183
3.8.2. Zestaw pytań kontrolnych . . . 184
3.8.3. Tematy zadań do wykonania w trakcie ćwiczenia . . . 184
3.8.4. Uwagi dotyczące sprawozdania . . . 185
3.9. Metody przybliżonego rozwiązywania równań nieliniowych . . . 185

3.9.1. Cel ćwiczenia 185
3.9.2. Zestaw pytań kontrolnych 185
3.9.3. Tematy zadań do wykonania w trakcie ćwiczenia 186
3.9.4. Uwagi dotyczące sprawozdania 186
3.10. Metody całkowania równań różniczkowych zwyczajnych 186
3.10.1. Cel ćwiczenia 186
3.10.2. Zestaw pytań kontrolnych 187
3.10.3. Tematy zadań do wykonania w trakcie ćwiczenia 187
3.10.4. Uwagi dotyczące sprawozdania 188
3.11. Metody całkowania równań różniczkowych cząstkowych typu parabolicznego 188
3.11.1. Cel ćwiczenia 188
3.11.2. Zestaw pytań kontrolnych 188
3.11.3. Tematy zadań do wykonania w trakcie ćwiczenia 189
3.11.4. Uwagi dotyczące sprawozdania 189
3.12. Równania całkowe typu Fredholma i Volterry 189
3.12.1. Cel ćwiczenia 189
3.12.2. Zestaw pytań kontrolnych 189
3.12.3. Tematy zadań do wykonania w trakcie ćwiczenia 190
3.12.4. Uwagi dotyczące sprawozdania 190

Literatura **191**

powered by Misiol

Rozdział 1

Podstawy teoretyczne podstawowych metod numerycznych

1.1. Teoria błędów

1.1.1. Definicje i pojęcia związane z błędami

Liczbą przybliżoną a nazywa się liczbę różniącą się nieznacznie od dokładnej liczby A i zastępującą ją w obliczeniach. Gdy $a < A$ $(a > A)$, liczba a jest wartością przybliżoną z niedomiarem (nadmiarem) liczby A.

Różnicę $\Delta a = A - a$ nazywamy błędem liczby a. Jeśli $A > a$, to błąd jest dodatni; gdy $A < a$, błąd jest ujemny. Żeby otrzymać dokładną liczbę A, należy do przybliżonej liczby a dodać błąd tej liczby, tj. $A = a + \Delta a$. Często znak błędu nie jest znany. Wówczas stosuje się pojęcie błędu bezwzględnego $\Delta = |\Delta a|$.

Definicja 1

Błędem bezwzględnym Δ liczby przybliżonej a nazywa się wartość bezwzględną różnicy pomiędzy liczbą dokładną A i liczbą przybliżoną a

$$\Delta = |A - a|. \tag{1.1}$$

Wyróżniamy tu dwa przypadki:

- liczba A jest znana (wtedy Δ oblicza się ze wzoru (1.1)),
- liczba A jest nieznana (wtedy zamiast nieznanego błędu Δ, który praktycznie trudno jest wyznaczyć, wprowadza się jego oszacowanie z góry, tzw. kres górny błędu bezwzględnego).

Definicja 2

Kresem górnym błędu bezwzględnego Δ_a liczby przybliżonej a nazywa się każdą liczbę nie mniejszą od błędu bezwzględnego tej liczby

$$\Delta = |A - a| < \Delta_a. \tag{1.2}$$

Wynika stąd, że liczba A zawarta jest w granicach:

$$a - \Delta_a \leq A \leq a + \Delta_a, \tag{1.3}$$

czyli $a - \Delta_a$ jest przybliżeniem z niedomiarem, zaś $a + \Delta_a$ przybliżeniem liczby A z nadmiarem.

Pojęcie kresu górnego błędu bezwzględnego obejmuje szeroką klasę liczb. Każda liczba większa od Δ_a może być również nazwana kresem górnym. W praktyce za Δ_a wybiera się możliwie najmniejszą w danych warunkach liczbę spełniającą nierówność (1.2). Zwykle kres górny identyfikuje się z błędem.

Błąd bezwzględny (albo jego kres) nie charakteryzuje dokładności pomiarów czy obliczeń. Wprowadza się więc pojęcie błędu względnego.

Definicja 3

Błędem względnym δ liczby przybliżonej a nazywa się stosunek błędu bezwzględnego Δ tej liczby do wartości bezwzględnej liczby dokładnej A ($A \neq 0$)

$$\delta = \frac{\Delta}{|A|}. \tag{1.4}$$

Gdy $\Delta \ll A$, to $\delta \cong \frac{\Delta}{|a|}$. Podobnie jak dla błędu bezwzględnego, definiuje się kres górny błędu względnego.

Definicja 4

Kresem górnym błędu względnego δ_a liczby przybliżonej a nazywa się każdą liczbę nie mniejszą od błędu względnego tej liczby

$$\delta \leq \delta_a, \tag{1.5}$$

tj. $\frac{\Delta}{|A|} \leq \delta_a$, czyli $\Delta \leq |A|\delta_a$. Zatem za kres górny błędu bezwzględnego liczby a można przyjąć

$$\Delta_a = |A|\delta_a. \tag{1.6}$$

W praktyce $A \approx a$, więc $\Delta_a = |a|\delta_a$. Znając δ_a otrzymujemy przedział, w którym zawarta jest liczba dokładna $A = a(1 \pm \delta_a)$.

1.1.2. Podstawowe źródła błędów

Wszystkie zagadnienia rozwiązywane metodami numerycznymi są obarczone na ogół pewnymi błędami. Błędy te można podzielić na pięć grup.

Błędy metody

Błędy te są związane z samym sformułowaniem zagadnienia matematycznego. Sformułowania matematyczne rzadko opisują realne zjawiska. Zazwyczaj dają ich bardziej lub mniej wyidealizowane modele. Często zmuszeni jesteśmy do przyjęcia pewnych warunków upraszczających. Błędy metody musimy uwzględnić w końcowym wyniku, tzn. musimy je oszacować.

powered by Misiol

Rozpatrzmy funkcję $f(x)$, która w punkcie x_0 posiada wszystkie pochodne aż do rzędu n włącznie. Dla funkcji $f(x)$ można utworzyć wielomian $W_n(x)$ postaci następującej

$$W_n(x) = f(x_0) + \frac{f^{(1)}(x_0)}{1!}(x-x_0) + \frac{f^{(2)}(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n. \tag{1.7}$$

Wielomian $W_n(x)$ tworzy $(n+1)$ pierwszych składników rozwinięcia funkcji $f(x)$ w szereg Taylora. Interesuje nas błąd przybliżenia

$$r_n(x) = f(x) - W_n(x). \tag{1.8}$$

Jest sprawą oczywistą, że $r_n(x)$ maleje ze wzrostem indeksu n. Różnicę $r_n(x)$ o postaci

$$r_n(x) = \frac{\alpha}{n!}(x - x_0)^n \tag{1.9}$$

nazywamy resztą Peano. Wzór Taylora z resztą Peano ma liczne zastosowania, ale odnoszą się one do samego punktu x_0 lub do punktów x dostatecznie bliskich punktowi x_0.

Istnieją jeszcze inne postacie reszty szeregu Taylora, np. reszta Lagrange'a, wyrażona następującym wzorem

$$r_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x - x_0)^{n+1}, \tag{1.10}$$

przy założeniu, że funkcja $f(x)$ jest $(n+1)$-krotnie różniczkowalna. Tak zwany punkt pośredni c spełnia jeden z poniższych warunków:

$$x_0 < c < x \quad \text{lub} \quad x < c < x_0.$$

Resztę w tej postaci można dokładnie oszacować dla różnych wartości x.

Błędy obcięcia

Błędy obcięcia są związane z obecnością procesów nieskończonych w analizie matematycznej. Często funkcje występujące we wzorach są określone jako granice ciągów nieskończonych lub sumy szeregów (np. $\sin x = x - \frac{x^3}{3!} + \cdots$). Na ogół powstaje konieczność zakończenia obliczeń na pewnym wyrazie ciągu. Przerwanie takie powoduje powstanie błędu obcięcia.

Błędy zaokrąglenia

Zaokrągleniem liczby nazywamy odrzucenie w niej wszystkich cyfr poczynając od pewnego miejsca. Powstaje przy tym błąd zaokrąglenia, który należy dodawać do początkowego błędu bezwzględnego liczby. Przy zaokrąglaniu liczb obowiązują następujące zasady:

- jeśli pierwsza z odrzuconych cyfr jest mniejsza od 5, to pozostałych cyfr dziesiętnych liczby nie zmienia się,
- jeśli pierwsza z odrzuconych cyfr jest większa od 5, to do ostatniej pozostawionej cyfry dodaje się 1,
- jeśli pierwsza z odrzuconych cyfr jest równa 5, a wśród odrzuconych cyfr są cyfry różne od zera, to ostatnią pozostałą cyfrę zwiększa się o 1,
- jeśli pierwsza z odrzuconych cyfr jest równa 5, a pozostałe odrzucone cyfry są zerami, to do ostatniej pozostawionej cyfry dodaje się 1, gdy jest ona nieparzysta, a pozostawia się ją bez zmian, gdy jest ona parzysta (jest to reguła cyfry parzystej).

Błędy początkowe

Błędy te występują w związku z obecnością we wzorach matematycznych parametrów liczbowych, których wartości można określić jedynie w przybliżeniu. Należą do niego np. wszystkie stałe fizyczne.

Błędy działań arytmetycznych

Ta grupa błędów jest związana z wykonywaniem działań arytmetycznych na liczbach przybliżonych. Jest rzeczą oczywistą, że przy wykonywaniu działań na liczbach przybliżonych przenosimy do wyników obliczeń błędy danych początkowych. W tym sensie błędy działań są nie do usunięcia. Przy rozwiązywaniu konkretnych zagadnień pewne błędy mogą w ogóle nie występować, albo mogą wywierać znikomy wpływ na całość rachunku. Jednakże w celu przeprowadzenia pełnej analizy błędów należy brać pod uwagę wszystkie ich rodzaje.

1.1.3. Metody obliczania błędów

Metoda różniczki zupełnej

Podstawowe zadanie teorii błędów można sformułować następująco: wyznaczyć błąd danej funkcji, gdy dane są błędy wszystkich jej argumentów.

Niech będzie dana funkcja różniczkowalna $y = f(x_1, x_2, \cdots, x_n)$. Oznaczmy przez $|\Delta x_i|$ $(i = 1, 2, \ldots, n)$ błędy bezwzględne argumentów tej funkcji. Wtedy błąd bezwzględny tej funkcji można zapisać w postaci

$$|\Delta y| = |f(x_1 + \Delta x_1, x_2 + \Delta x_2, \cdots, x_n + \Delta x_n) - f(x_1, x_2, \cdots, x_n)|. \tag{1.11}$$

W praktyce $|\Delta x_i|$ są wielkościami małymi, których kwadratów i wyższych potęg można nie brać pod uwagę. Można zatem przyjąć

$$|\Delta y| \approx |df(x_1, x_2, \cdots, x_n)| = \left|\sum_{i=1}^{i=n} \frac{\partial f}{\partial x_i} \Delta x\right| \leq \sum_{i=1}^{i=n} \left|\frac{\partial f}{\partial x_i}\right| |\Delta x_i|, \tag{1.12}$$

a więc

$$|\Delta y| \leq \sum_{i=1}^{i=n} \left|\frac{\partial f}{\partial x_i}\right| |\Delta x_i|. \tag{1.13}$$

Stąd dla kresów górnych można zapisać następującą zależność

$$\Delta y = \sum_{i=1}^{i=n} \left|\frac{\partial f}{\partial x_i}\right| \Delta x_i. \tag{1.14}$$

Błąd względny rozpatrywanej funkcji wyrazi się wzorem

$$\delta y = \frac{|\Delta y|}{|y|} \leq \sum_{i=1}^{i=n} \frac{|\frac{\partial f}{\partial x_i}|}{|y|} |\Delta x_i| = \sum_{i=1}^{i=n} |\frac{\partial}{\partial x_i} \ln f(x_1, x_2, \cdots, x_n)| |\Delta x_i|, \tag{1.15}$$

przy założeniu, że $f(x_1, x_2, \cdots, x_n) \geq 0$.

powered by Misio

Przykład 1

Znaleźć kres górny błędu bezwzględnego objętości kuli, jeśli średnica kuli wynosi $d = 3.7cm \pm 0.05cm$; $\pi = 3.14 \pm 0.0016$.

Rozwiązanie

Objętość kuli wynosi $V = \frac{1}{6}\pi d^3$. Traktując π oraz d jako wielkości zmienne, obliczamy pochodne cząstkowe:

$$\frac{\partial V}{\partial \pi} = \frac{1}{6} d^3 = 8.44, \qquad \frac{\partial V}{\partial d} = \frac{1}{6}\pi d^2 = 21.5.$$

Korzystając ze wzoru (1.14) otrzymamy:

$$\Delta_V = \left|\frac{\partial V}{\partial \pi}\right| |\Delta \pi| + \left|\frac{\partial V}{\partial d}\right| |\Delta d| = 8.44 \cdot 0.0016 + 21.5 \cdot 0.05 \approx 1.1cm^3.$$

Zatem ostatecznie otrzymujemy

$$V = \frac{1}{6}\pi d^3 \approx 27.4cm^3 \pm 1.1cm^3.$$

Metoda przedziałów wartości

Załóżmy, że dany jest przedział $[\underline{X}, \overline{X}]$ wartości, jakie może przyjąć liczba rzeczywista X $(\underline{X} \leq X \leq \overline{X})$. Jeżeli każdą liczbę rzeczywistą można przedstawić w postaci przedziału jej skrajnych wartości, to można wykazać, że zachodzą następujące zależności określające przedziały wartości poszczególnych działań arytmetycznych:

$$-[\underline{X}, \overline{X}] = [-\overline{X}, -\underline{X}], \tag{1.16}$$

$$[\underline{X}, \overline{X}] + [\underline{Y}, \overline{Y}] = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}], \tag{1.17}$$

$$[\underline{X}, \overline{X}] - [\underline{Y}, \overline{Y}] = [\underline{X} - \underline{Y}, \overline{X} - \overline{Y}], \tag{1.18}$$

$$[\underline{X}, \overline{X}][\underline{Y}, \overline{Y}] = [\min(\underline{XY}, \underline{X}\overline{Y}, \overline{XY}); \max(\underline{XY}, \underline{X}\overline{Y}, \overline{Y}, \underline{Y}, \overline{XY})], \tag{1.19}$$

$$[\underline{X}, \overline{X}] : [\underline{Y}, \overline{Y}] = [\min(\frac{\underline{X}}{\underline{Y}}, \frac{\underline{X}}{\overline{Y}}, \frac{\overline{X}}{\overline{Y}}); \max(\frac{\underline{X}}{\underline{Y}}, \frac{\underline{X}}{\overline{Y}}, \frac{\overline{Y}}{\underline{Y}}, \frac{\overline{X}}{\overline{Y}})], \text{ dla } \underline{Y}, \overline{Y} \neq 0. \tag{1.20}$$

Dla liczb dodatnich otrzymamy na podstawie (1.19) i (1.20) następujące zależności:

$$[\underline{X}, \overline{X}][\underline{Y}, \overline{Y}] = [\underline{XY}, \overline{XY}], \tag{1.21}$$

$$[\underline{X}, \overline{X}] : [\underline{Y}, \overline{Y}] = [\frac{\underline{X}}{\overline{Y}}, \frac{\overline{X}}{\underline{Y}}]. \tag{1.22}$$

Posługując się przedziałami wartości można prowadzić obliczenia również dla funkcji złożonych.

Przykład 2

Dane są przedziały, w których mieszczą się wartości tworzącej i promieni ściętego stożka prostego:

$$\underline{r_1} = 73.5;\ \overline{r_1} = 74.5;\ \underline{r_2} = 60.5;\ \overline{r_2} = 61.5;\ \underline{l} = 81.5;\ \overline{l} = 82.5$$

(l - tworząca stożka). Wyznaczyć przedział, w którym mieści się wartość wysokości tego stożka.

Rozwiązanie

Korzystając ze wzoru na wysokość stożka ściętego

$$h = \sqrt{l^2 - (r_1 - r_2)^2}$$

obliczamy kolejno:

$$\underline{r_1 - r_2} = 12.0$$

$$\overline{r_1 - r_2} = 14.0$$

$$(\underline{r_1 - r_2})^2 = 144.0$$

$$(\overline{r_1 - r_2})^2 = 196.0$$

$$\underline{l}^2 = 6642.25$$

$$\overline{l}^2 = 6806.25$$

$$\overline{h}^2 = 6662.25$$

$$\underline{h}^2 = 6446.25$$

$$\underline{h} = 80.28$$

$$\overline{h} = 81.63$$

Wartość wysokości stożka mieści się w granicach [80.28; 81.63].
Uwaga: w przykładzie korzystano z równości

$$\sqrt{[\underline{X}, \overline{X}]} = [\sqrt{\underline{X}}, \sqrt{\overline{X}}].$$

powered by Misiol

1.1.4. Błędy działań arytmetycznych na liczbach przybliżonych

Błąd sumy

Niech $x_1, x_2, \cdots, x_n$ będą danymi liczbami przybliżonymi. Rozpatrzmy ich sumę algebraiczną

$$y = \sum_{i=1}^{i=n} x_i.$$

Korzystając ze wzoru (1.13) zapiszemy następującą zależność na błąd bezwzględny sumy liczb przybliżonych

$$\Delta y \leq \sum_{i=1}^{i=n} |\Delta x_i|, \tag{1.23}$$

ponieważ dla każdego $i = 1, 2, \ldots, n$ $|\frac{\partial y}{\partial x_i}| = 1$. Można zatem powiedzieć, że błąd bezwzględny skończonej sumy algebraicznej liczb przybliżonych nie przewyższa sumy błędów bezwzględnych tych liczb.

Dla błędu względnego sumy algebraicznej liczb przybliżonych prawdziwa jest zależność

$$\delta_y \leq \sum_{i=1}^{i=n} \frac{|\Delta x_i|}{|\sum_{i=1}^{i=n} x_i|}. \tag{1.24}$$

Błąd różnicy

Rozpatrzmy różnicę dwóch liczb przybliżonych

$$y = x_1 - x_2. \tag{1.25}$$

Ponieważ $|\frac{\partial y}{\partial x_1}| = |\frac{\partial y}{\partial x_2}| = 1$, więc zgodnie z (1.13) błąd bezwzględny różnicy tych liczb wyrazi się wzorem

$$|\Delta y| \leq |\Delta x_1| + |\Delta x_2|. \tag{1.26}$$

Zatem błąd względny różnicy jest równy

$$\delta_y = \frac{\Delta y}{y} \leq \frac{|\Delta x_1| + |\Delta x_2|}{|x_1 - x_2|}. \tag{1.27}$$

W przypadku odejmowania liczb przybliżonych należy zwrócić uwagę na mogącą wystąpić tzw. stratę dokładności. Jeżeli liczby przybliżone x_1 i x_2 różnią się nieznacznie i są obarczone małymi błędami bezwzględnymi, to ich różnica $|x_1 - x_2|$ jest mała. Ze wzoru (1.27) wynika, że błąd względny różnicy będzie znaczny (ma tu miejsce strata dokładności). Z tego względu w obliczeniach przybliżonych lepiej jest przekształcać wyrażenia, których obliczanie nie prowadzi do odejmowania bliskich liczb. Jeżeli już zaistnieje konieczność odejmowania prawie równych liczb przybliżonych, to liczby te należy wziąć z wystarczającą liczbą cyfr znaczących.

Błąd iloczynu i ilorazu

Niech $y = x_1 \cdot x_2 \cdot \ldots \cdot x_n$. Wówczas $|\frac{\partial f}{\partial x_i}| = |x_1 \cdot x_2 \cdot \ldots \cdot x_{i-1} \cdot x_{i+1} \cdot \ldots \cdot x_n|$. Korzystając ze wzoru (1.13) otrzymamy następującą zależność na błąd bezwzględny iloczynu liczb przybliżonych

$$|\Delta y| \leq |\Delta x_1 \cdot x_2 \cdot \ldots \cdot x_n| + |x_1 \cdot \Delta x_2 \cdot x_3 \cdot \ldots \cdot x_n| + \cdots + |x_1 \cdot x_2 \cdot \ldots \cdot \Delta x_n|. \quad (1.28)$$

Błąd względny iloczynu n liczb przybliżonych różnych od zera nie przewyższa sumy błędów względnych tych liczb. A zatem

$$\delta_y \leq \sum_{i=1}^{i=n} \frac{|\Delta x_i|}{|x_i|} = \sum_{i=1}^{i=n} \delta x_i. \quad (1.29)$$

Jeżeli $y = \frac{x_1}{x_2}$, to wzory na błąd bezwzględny i względny ilorazu tych liczb są następujące:

$$|\Delta y| \leq \frac{|\Delta x_1|}{|x_2|} + \frac{|x_1||\Delta x_2|}{x_2^2} = \frac{|\Delta x_1|}{|x_2|} + |y| \cdot \frac{|\Delta x_2|}{|x_2|}, \quad (1.30)$$

$$\delta y \leq \frac{|\Delta x_1|}{|x_1|} + \frac{|\Delta x_2|}{|x_2|} = \delta x_1 + \delta x_2. \quad (1.31)$$

Błąd potęgi i pierwiastka

Niech

$$y = x^m.$$

Ponieważ

$$\left|\frac{\partial f}{\partial x}\right| = |m| \cdot |x^{m-1}|,$$

to błąd bezwzględny funkcji potęgowej będzie miał postać

$$|\Delta y| \leq |m| \cdot \frac{|y|}{|x|} \cdot |\Delta x|. \quad (1.32)$$

Błąd względny m-tej potęgi liczby przybliżonej jest m razy większy niż błąd względny tej liczby. Stąd

$$\delta_y \leq |m| \cdot \frac{|\Delta x|}{|x|} = |m| \cdot \delta x. \quad (1.33)$$

W przypadku gdy $y = \sqrt[m]{x}$ $(m > 0)$, prawdziwe są zależności:

$$\Delta y \leq \frac{1}{|m|} \frac{|y|}{|x|} \cdot |\Delta x| \quad (1.34)$$

oraz

$$\delta y \leq \frac{1}{m} \cdot \frac{|\Delta x|}{|x|} = \frac{1}{m} \cdot \delta x. \quad (1.35)$$

powered by Misiol

1.1.5. Problem odwrotny teorii błędów

W praktyce ważny jest również problem odwrotny do problemu omówionego przy przedstawieniu metody różniczki zupełnej. Można go sformułować następująco:

Jakie mogą być błędy bezwzględne argumentów funkcji, aby błąd bezwzględny funkcji nie przekraczał zadanej wartości?

Dla uproszczenia rozważań zakładamy, że wszystkie pochodne cząstkowe $\frac{\partial f}{\partial x_i}$ $(i = 1, 2, \ldots, n)$ wpływają jednakowo na powstanie ogólnego błędu bezwzględnego funkcji $y = f(x_1, x_2, \ldots, x_n)$. Jest to tzw. zasada równego wpływu. Zakładając, że składniki sumy we wzorze (1.14) są równe, czyli

$$\left|\frac{\partial f}{\partial x_1}\right| \cdot \Delta x_1 = \left|\frac{\partial f}{\partial x_2}\right| \cdot \Delta x_2 = \cdots = \left|\frac{\partial f}{\partial x_n}\right| \cdot \Delta x_n, \quad (1.36)$$

otrzymujemy

$$\Delta y = \sum_{i=1}^{i=n} \left|\frac{\partial f}{\partial x_i}\right| \Delta x_i = n \cdot \left|\frac{\partial f}{\partial x_i}\right| \cdot \Delta x_i. \quad (1.37)$$

Z powyższej zależności otrzymujemy następujący wzór

$$\Delta x_i = \frac{\Delta y}{n|\frac{\partial f}{\partial x_i}|} \qquad i = 1, 2, \ldots, n. \quad (1.38)$$

Przykład 3

Promień podstawy walca wynosi $r \approx 2cm$, a wysokość walca $h \approx 3cm$. Z jaką dokładnością bezwzględną należy określić r oraz h, żeby objętość walca V obliczyć z dokładnością do $0.1cm^3$?

Rozwiązanie

Dane: $\Delta_V = 0.1cm^3$, $r \approx 2cm$, $h \approx 3cm$, $\pi \approx 3.14$, $V = \pi r^2 h$

Szukane: Δ_r, Δ_h, Δ_π

Dla powyższych danych otrzymujemy w przybliżeniu:

$$\frac{\partial V}{\partial r} = 2\pi r h = 37.7cm^2,$$

$$\frac{\partial V}{\partial h} = \pi r^2 = 12.6cm^2,$$

$$\frac{\partial V}{\partial \pi} = r^2 h = 12.0cm^3.$$

Ponieważ n=3, to na podstawie wzoru (1.38) otrzymamy:

$$\Delta_r = \frac{0.1}{3 \cdot 37.7} < 0.001,$$

$$\Delta_h = \frac{0.1}{3 \cdot 12.6} < 0.003,$$

$$\Delta_\pi = \frac{0.1}{3 \cdot 12} < 0.003.$$

1.2. Obliczanie wartości funkcji

1.2.1. Obliczanie wartości wielomianu. Schemat Hőrnera

Niech dany będzie wielomian stopnia n, o współczynnikach rzeczywistych a_i, $(i = 0, 1, \ldots, n)$

$$W_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = \sum_{i=0}^{n} a_i x^i, \quad a_i \in R. \tag{2.1}$$

Należy obliczyć jego wartość dla $x = \xi$

$$W_n(\xi) = a_n \xi^n + a_{n-1} \xi^{n-1} + \cdots + a_0. \tag{2.2}$$

Najwygodniej jest obliczać wartość $W_n(\xi)$ korzystając z przekształconej postaci wielomianu

$$W_n(\xi) = (\cdots((a_n \xi + a_{n-1})\xi + a_{n-2})\xi + \cdots + a_1)\xi + a_0. \tag{2.3}$$

Wtedy obliczając kolejno:

$$\begin{aligned} b_n &= a_n, \\ b_{n-1} &= a_{n-1} + b_n \xi, \\ b_{n-2} &= a_{n-2} + b_{n-1} \xi, \\ &\vdots \\ b_i &= a_i + b_{i+1} \xi, \\ &\vdots \\ b_0 &= a_0 + b_1 \xi = W_n(\xi), \end{aligned} \tag{2.4}$$

otrzymujemy poszukiwaną wartość.

Liczby $b_n, b_{n-1}, \cdots, b_0$ są współczynnikami wielomianu $Q_{n-1}(x)$, który powstaje w wyniku podzielenia wielomianu $W_n(x)$ przez dwumian $x - \xi$. Na podstawie twierdzenia Bezouta reszta z tego dzielenia jest $\beta_0 = W_n(\xi)$. Czyli

$$\sum_{i=0}^{n} a_i x^i = \left(\sum_{i=0}^{n-1} \beta_{i+1} x^i \right)(x - \xi) + \beta_0. \tag{2.5}$$

powered by Misiol

Stąd po wykonaniu odpowiednich działań, z porównania współczynników przy jednakowych potęgach mamy:

$$\begin{aligned} a_n &= \beta_n, \\ a_{n-1} &= \beta_{n-1} - \beta_n \xi, \\ a_{n-2} &= \beta_{n-2} - \beta_{n-1} \xi, \\ &\vdots \\ a_i &= \beta_i - \beta_{i+1} \xi, \\ &\vdots \\ a_0 &= \beta_0 - \beta_1 \xi. \end{aligned} \tag{2.6}$$

Po ich przekształceniu otrzymujemy zależności, które należało wykazać:

$$\begin{aligned} \beta_n &= a_n = b_n, \\ \beta_{n-1} &= a_{n-1} + b_n \xi = b_{n-1}, \\ \beta_{n-2} &= a_{n-2} + b_{n-1} \xi = b_{n-2}, \\ &\vdots \\ \beta_i &= a_i + b_{i+1} \xi = b_i, \\ &\vdots \\ \beta_0 &= a_0 + b_1 \xi = b_0 = W_n(\xi). \end{aligned} \tag{2.7}$$

W praktyce obliczenia wartości wielomianu zgodnie ze wzorem (2.4) przeprowadza się według tzw. schematu Hőrnera w sposób następujący

$$\begin{array}{cccccc} & a_n & a_{n-1} & a_{n-2} & \cdots & a_0 \\ + & & b_n \xi & b_{n-1} \xi & \cdots & b_1 \xi \\ \hline & b_n & b_{n-1} & b_{n-2} & \cdots & b_0 = W_n(\xi). \end{array} \tag{2.8}$$

Można udowodnić, że schemat Hőrnera jest algorytmem o minimalnej złożoności obliczeniowej, minimalizującym liczbę potrzebnych mnożeń i dodawań przy obliczaniu wartości wielomianu danego w postaci naturalnej.

Błędy numeryczne przy obliczaniu wartości wielomianu ze schematu Hőrnera zależą tylko od błędów zaokrągleń. Załóżmy, że współczynniki $a_n, a_{n-1}, \cdots, a_0$ oraz liczba ξ są liczbami dokładnymi i obliczenia prowadzone są z dokładnością do k miejsc po przecinku. Wtedy oszacowanie błędu obliczenia wartości wielomianu wyraża się wzorem

$$\Delta W_n(\xi) = \frac{|\xi|^k - 1}{|\xi| - 1} \cdot 0.5 \cdot 10^k, \tag{2.9}$$

gdzie: n - stopień wielomianu, k - rząd ostatniej zapisywanej cyfry.

Za pomocą schematu Hőrnera można określić przedział, w którym zawarte są pierwiastki rzeczywiste wielomianu $W_n(x)$. W tym celu należy znaleźć metodą prób $x = \beta > 0$ takie, by wszystkie współczynniki $b_i (i = 0, 1, \ldots, n-1)$ ze schematu Hőrnera były nieujemne:

$$b_n = a_n > 0, \quad b_i \geq 0, \quad i = 0, 1, \ldots, n-1. \tag{2.10}$$

Wtedy wszystkie pierwiastki rzeczywiste x_k $(k = 1, 2, \ldots, m;\ m \leq n)$ wielomianu $W_n(x)$ nie są większe od β, tj. $x_k \leq \beta$, $(k = 1, 2, \ldots, m)$, ponieważ

$$W_x(x) = (b_n x^{n-1} + b_{n-1} x^{n-2} + \cdots + b_1)(x - \beta) + b_0 > 0 \tag{2.11}$$

dla dowolnego $x > \beta$, a więc dowolna liczba większa od β na pewno nie jest pierwiastkiem wielomianu.

W celu otrzymania oszacowania z dołu wartości pierwiastków x_k rozpatrujemy wielomian

$$(-1)^n W_n(-x) = a_n x^n - a_{n-1} x^{n-1} + \cdots + (-1)^n a_0. \tag{2.12}$$

Dla tego wielomianu znajdujemy taką liczbę $x = \alpha > 0$, żeby był spełniony warunek (2.10). Wtedy pierwiastki rzeczywiste wielomianu $(-1)^n W_n(-x)$, równe $-x_k$, $(k = 1, 2, \ldots, m)$, spełniają nierówność $-x_k \leq \alpha$, a zatem $x_k \geq -\alpha$, $(k = 1, 2, \ldots, m)$. Wszystkie zatem pierwiastki rzeczywiste wielomianu $W_n(x)$ leżą w przedziale domkniętym $[-\alpha, \beta]$.

Przykład 1

Obliczyć $W_4(x) = x^4 - 3.5x^3 + 2x - 1$, dla $x = 2.5$.

	1	−3.5	0	2	−1
+		2.5	−2.5	−6.25	−10.625
	1	−1.0	−2.5	−4.25	−11.625 = $W_4(2.5)$

Odp. $W_4(2.5) = -11.625$.

Przykład 2

Obliczyć, z jaką dokładnością należy prowadzić obliczenia wielomianu z przykładu 1 dla $x = 2.7555$, aby otrzymać wynik z dokładnością do 0.001.

$$W_4(2.755) \leq \frac{3^4 - 1}{3 - 1} \cdot 0.5 \cdot 10^k < 0.001$$

$$k \leq -5$$

Odp. Obliczenia wystarczy prowadzić z dokładnością do 5 miejsc po przecinku.

Przykład 3

Znaleźć przedział, w którym zawarte są pierwiastki rzeczywiste wielomianu

$$W_4(x) = x^4 - 2x^3 + x^2 - 3x - 1.$$

Dla $x = 3$ otrzymujemy

	1	−2	1	−3	−1
+		3	3	12	27
	1	1	4	9	26.

powered by Misio]

Warunek (2.10) jest spełniony, a zatem $x_k \leq 3$. Przyjmując dla wielomianu

$$Q_4(x) = (-1)^4 W_4(-x) = x^4 + 2x^3 + x^2 + 3x - 1$$

$x = 0.5$ uzyskujemy

	1	2	1	3	−1
+		0.5	1.25	1.125	2.0625
	1	2.5	2.25	4.125	1.0625.

Odp. Wszystkie pierwiastki rzeczywiste wielomianu leżą wewnątrz przedziału domkniętego $[-0.5, 3]$.

1.2.2. Obliczanie wartości funkcji analitycznej

Funkcja rzeczywista $f(x)$ nazywa się funkcją analityczną w punkcie ξ, jeśli w otoczeniu $|x - \xi| < R$ tego punktu można ją rozwinąć w zbieżny szereg potęgowy (szereg Taylora)

$$f(x) = f(\xi) + f^{(1)}(\xi)(x - \xi) + \frac{f^{(2)}(\xi)}{2!}(x - \xi)^2 + \cdots + \frac{f^{(n)}(\xi)}{n!}(x - \xi)^n + \cdots \tag{2.13}$$

Zastępując sumę szeregu nieskończonego sumą n pierwszych jego wyrazów otrzymujemy wielomian Taylora

$$W_n(x) = \sum_{i=0}^{n} \frac{f^{(i)}(\xi)}{i!}(x - \xi)^i. \tag{2.14}$$

W szczególnym przypadku przy $\xi = 0$ otrzymujemy wielomian Maclaurina

$$W_n(x) = \sum_{i=0}^{n} \frac{f^{(i)}(0)}{i!} x^i. \tag{2.15}$$

Różnicę wyrażającą błąd obcięcia, który powstaje przy zastąpieniu funkcji $f(x)$ wielomianem Taylora, nazywamy resztą szeregu Taylora i obliczamy następująco

$$R_n(x) = f(x) - \sum_{i=0}^{n} \frac{f^{(i)}(\xi)}{i!}(x - \xi)^i = \sum_{i=n+1}^{\infty} \frac{f^{(i)}(\xi)}{i!}(x - \xi)^i. \tag{2.16}$$

Można dowieść, że błąd obcięcia wyraża się następującym wzorem

$$R_n(x) = \frac{f^{(n+1)}(\xi + \Theta(x - \xi))}{(n+1)!}(x - \xi)^{n+1}, \tag{2.17}$$

gdzie $0 < \Theta < 1$ jest parametrem, natomiast $\xi + \Theta(x - \xi)$ jest punktem pośrednim. Ponieważ w praktyce nie znamy wartości Θ, to w przypadku gdy $x - \xi = h$, czyli $f(x) = f(\xi + h)$, można oszacować moduł błędu obcięcia w sposób następujący

$$|R_n(h)| < \frac{\max\limits_{0<\Theta<1} |f^{(n+1)}(\xi + h)|}{(n+1)!}|h|^{n+1}. \tag{2.18}$$

Przykład 4

Obliczyć wartość $e^{\frac{1}{4}}$ uwzględniając 4 pierwsze wyrazy szeregu i oszacować błąd obcięcia

$$f(x) = f^{(1)}(x) = f^{(2)}(x) = f^{(3)}(x) = f^{(4)}(x) = e^x.$$

Przyjmijmy: $\xi = 0$, $h = 1/4$. Stąd:

$$\begin{aligned} f(\xi + h) &= 1 + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{2!} \cdot \left(\frac{1}{4}\right)^2 + 1 \cdot \frac{1}{3!} \cdot \left(\frac{1}{4}\right)^3 + 1 \cdot \frac{1}{4!} \cdot \left(\frac{1}{4}\right)^4 + R_4(h) = \\ &= 1 + 0.25 + 0.03125 + 0.006944 + 0.000162 + R_4(h) = \\ &= 1.28836 + R_4(h), \end{aligned}$$

$$|R_4(h)| \le \frac{\max\limits_{0<\Theta<1} \left|e^{\frac{\Theta}{4}}\right|}{5!} \left(\frac{1}{4}\right)^5 \le \frac{2}{5!\,4^5} \approx 2 \cdot 10^{-5}.$$

1.2.3. Iteracyjne obliczanie wartości funkcji

Załóżmy, że dla zadanej wartości argumentu x należy obliczyć wartość funkcji ciągłej

$$y = f(x). \tag{2.19}$$

Zapiszemy funkcję w postaci uwikłanej, tak aby $f(x)$ nie występowała w sposób jawny

$$F(x, y) = 0. \tag{2.20}$$

Załóżmy, że $F(x, y)$ jest funkcją ciągłą i ma ciągłą pochodną cząstkową oraz $F_y^{(1)}(x, y) \neq 0$. Niech y_n będzie wartością przybliżoną y. Wtedy stosując twierdzenie Lagrange'a mamy

$$F(x, y_n) = F(x, y_n) - F(x, y) = (y_n - y) F_y^{(1)}(x, \overline{y}_n), \tag{2.21}$$

gdzie $\overline{y}_n$ jest pewną wartością pośrednią pomiędzy y_n i y. Stąd

$$y = y_n - \frac{F(x, y_n)}{F_y^{(1)}(x, \overline{y}_n)}. \tag{2.22}$$

Wartość $\overline{y}_n$ nie jest znana, ale przyjmując $\overline{y}_n \approx y_n$, otrzymujemy do obliczania wartości y proces iteracyjny

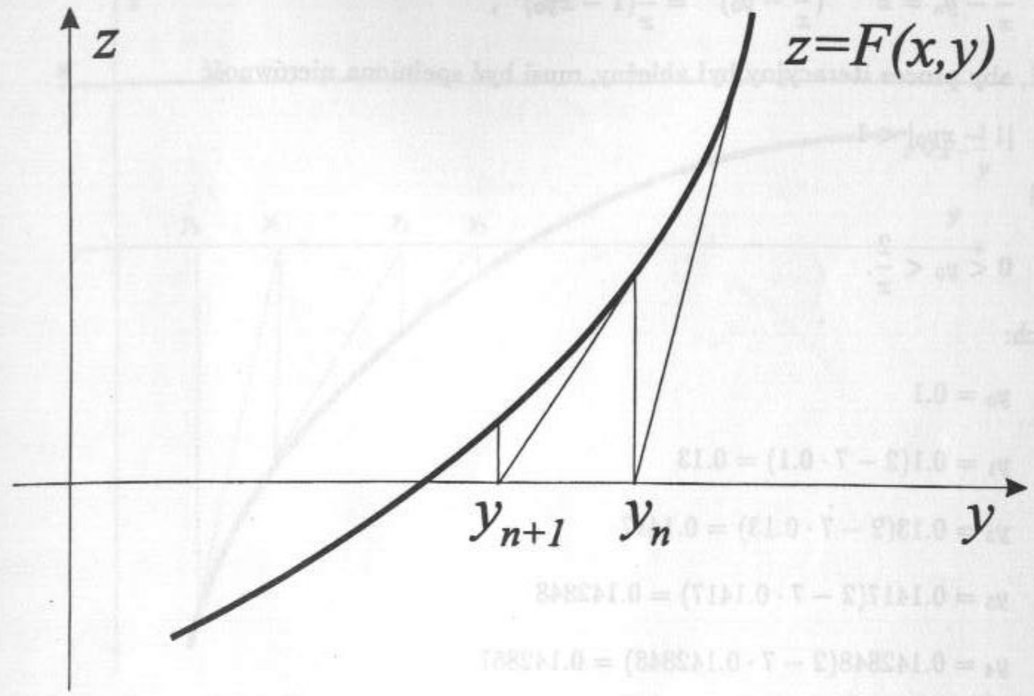
$$y_{n+1} = y_n - \frac{F(x, y_n)}{F_y^{(1)}(x, y_n)} \qquad n = 1, 2, \ldots. \tag{2.23}$$

Proces ten przedstawia metodę Newtona zastosowaną do funkcji $z = F(x, y)$, gdzie x jest zadanym parametrem. Interpretacja geometryczna procesu iteracji przedstawiona jest na rysunku 1.1.

Dla zbieżności procesu iteracyjnego wystarczy, aby pochodne cząstkowe $F_y^{(1)}(x, y)$ oraz $F_{yy}^{(2)}(x, y)$ nie zmieniały znaku w rozpatrywanym przedziale wartości, zawierającym poszukiwany pierwiastek y. Przybliżenie początkowe y_0 należy wybierać w miarę możliwości bliskie poszukiwanej wartości y i spełniające zależność

$$F(x, y_0) \cdot F_{yy}^{(2)}(x, y_0) > 0. \tag{2.24}$$

powered by Misiol



Rys. 1.1. Geometryczna interpretacja iteracyjnego obliczania wartości funkcji

Proces iteracyjny przerywamy wtedy, gdy w granicach zadanej dokładności ε dwie kolejne wartości iterat y_{n+1} i y_n spełniają nierówność

$$|y_{n+1} - y_n| < \varepsilon. \tag{2.25}$$

Wtedy na ogół nie można niestety zagwarantować, że prawdziwa jest także nierówność

$$|y - y_{n+1}| < \varepsilon. \tag{2.26}$$

Przykład 5

Obliczyć $y = 1/x$ dla $x = 7$, bez wykonywania dzielenia, z dokładnością $\varepsilon = 0.001$. Określić przedział zbieżności procesu iteracyjnego:

$$F(x, y) = x - \frac{1}{y}, \qquad F_y^{(1)}(x, y) = \frac{1}{y^2}.$$

Korzystając z zależności (2.23) otrzymujemy

$$y_{n+1} = y_n - \frac{x - \frac{1}{y_n}}{\frac{1}{y_n^2}} = y_n(2 - x y_n), \qquad n = 0, 1, 2, \ldots.$$

Przedział zbieżności określić możemy w następujący sposób:

$$\frac{1}{x} - y_n = \frac{1}{x} - 2y_{n-1} + x y_{n-1}^2 = x\left(\frac{1}{x} - y_{n-1}\right)^2,$$

$$\frac{1}{x} - y_n = x^{2^n-1}(\frac{1}{x} - y_0)^{2^n} = \frac{1}{x}(1 - xy_0)^{2^n},$$

czyli, aby proces iteracyjny był zbieżny, musi być spełniona nierówność

$$|1 - xy_0| < 1.$$

Stąd

$$0 < y_0 < \frac{2}{x}.$$

Niech:

$$y_0 = 0.1$$

$$y_1 = 0.1(2 - 7 \cdot 0.1) = 0.13$$

$$y_2 = 0.13(2 - 7 \cdot 0.13) = 0.1417$$

$$y_3 = 0.1417(2 - 7 \cdot 0.1417) = 0.142848$$

$$y_4 = 0.142848(2 - 7 \cdot 0.142848) = 0.142857$$

$$|y_4 - y_3| < 0.001$$

Odp. $1/7 \approx 0.14286$.

Uwaga: korzystając z zależności (2.24) otrzymujemy węższy przedział, z którego należy wybrać przybliżenie początkowe

$$0 < y_0 < \frac{1}{x}.$$

Wybierając x z tego przedziału, widzimy, że kolejne przybliżenia rosną monotonicznie, co ilustruje rysunek 1.2

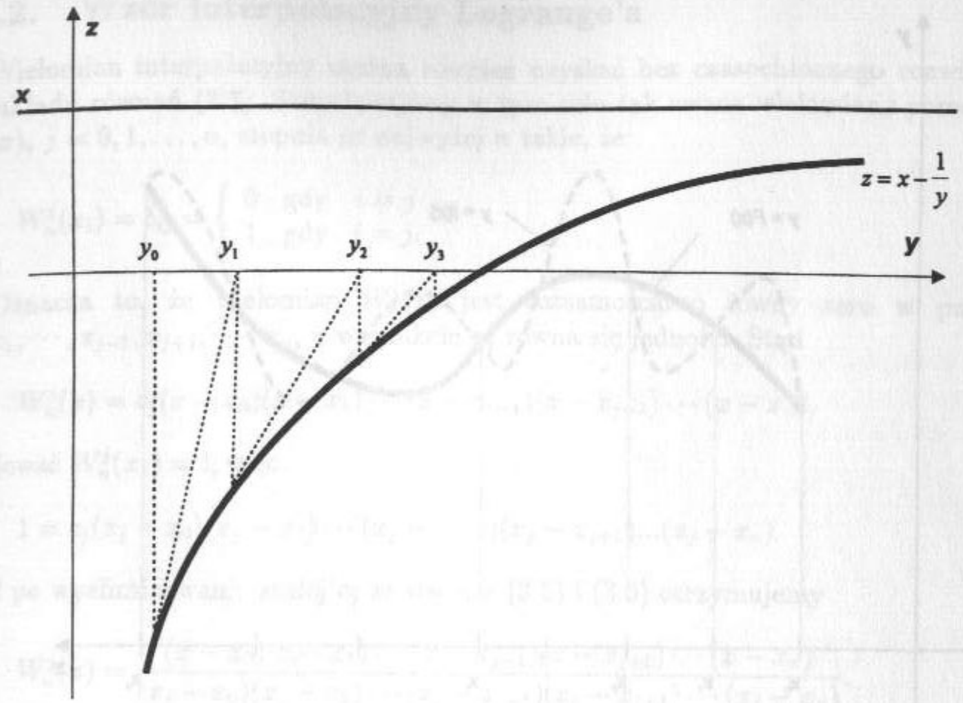
1.3. Interpolacja funkcji

1.3.1. Sformułowanie zagadnienia interpolacji

Interpolacją funkcji $f(x) : [a, b] \to R$ nazywa się wyznaczenie przybliżonych wartości funkcji $f(x)$ dla dowolnego argumentu $x \in [x_0, x_n] \subset [a, b]$ przy znanych jej wartościach $f(x_0), f(x_1), \cdots, f(x_n)$ w ustalonych punktach $a \leq x_0 \leq x_2 \leq \cdots \leq x_n \leq b$, zwanych węzłami interpolacji.

Interpolację stosujemy, gdy nie znamy postaci analitycznej funkcji $f(x)$, lecz tylko jej wartości dla skończonego zbioru argumentów lub, gdy analityczna postać funkcji $f(x)$ jest znana, ale obliczanie jej wartości jest zbyt pracochłonne.

powered by Misiol



Rys. 1.2. Iteracyjne obliczanie wartości funkcji dla przykładu 5

W celu wyznaczenia przybliżonych wartości funkcji $f(x)$ konstruujemy tzw. funkcję interpolacyjną $F(x) : [a, b] \to R$ określonej klasy, dostatecznie prostą dla obliczeń, która w węzłach interpolacji przyjmuje te same wartości co funkcja $f(x)$, czyli:

$$F(x_0) = f(x_0), \quad F(x_1) = f(x_1), \cdots, F(x_n) = f(x_n). \tag{3.1}$$

W interpretacji geometrycznej (rys. 1.3) oznacza to, że poszukujemy krzywej $F(x)$ określonego typu, przechodzącej przez zadany układ punktów: $M_i(x_i, y_i)$, $y_i = f(x_i)$, $i = 0, 1, \ldots, n$.

Tak sformułowane zagadnienie staje się jednoznaczne, jeżeli zażądamy dodatkowo, aby funkcja $F(x)$ była wielomianem stopnia co najwyżej n.

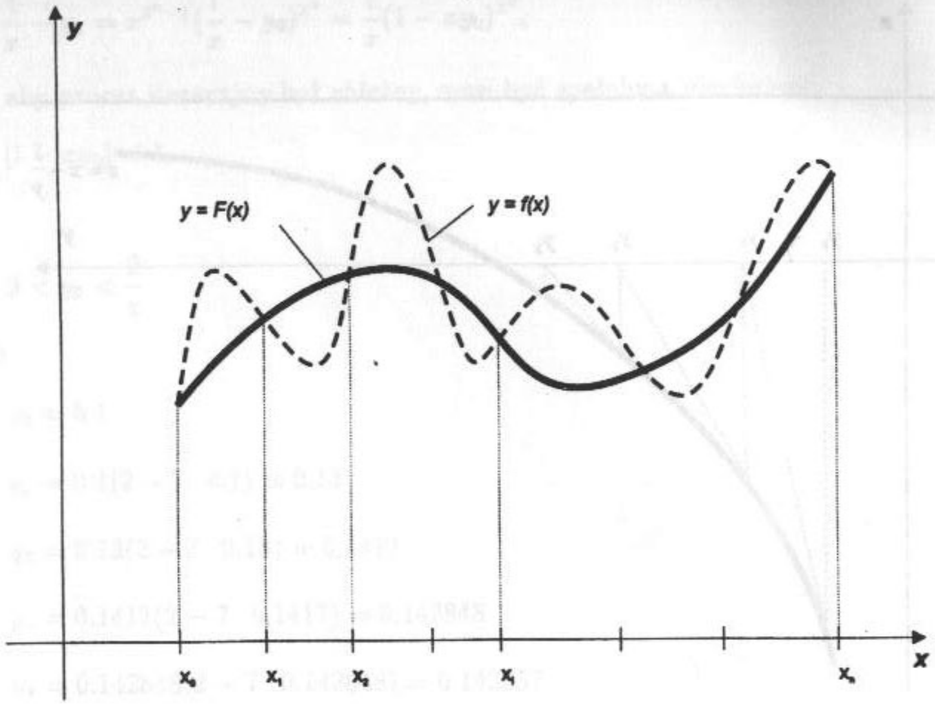
Twierdzenie

Istnieje dokładnie jeden wielomian interpolacyjny $W_n(x)$ stopnia co najwyżej n, $(n \geq 0)$, który w punktach $x_0, x_1, \cdots, x_n$ przyjmuje wartości $y_0, y_1, \cdots, y_n$.

Dowód: Szukany wielomian ma postać

$$W_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0. \tag{3.2}$$

Korzystając z warunku (3.1) otrzymujemy układ $n + 1$ równań liniowych z $n + 1$



Rys. 1.3. Graficzne sformułowanie zagadnienia interpolacji

niewiadomymi współczynnikami a_i, $i = 0, 1, \ldots, n$:

$$\begin{array}{ccccccccc} a_0 & + & a_1x_0 & + & a_2x_0^2 & + & \cdots & + & a_nx_0^n & = & f(x_0) \\ a_0 & + & a_1x_1 & + & a_2x_1^2 & + & \cdots & + & a_nx_1^n & = & f(x_1) \\ & & & & \vdots & & & & \vdots & & \\ a_0 & + & a_1x_i & + & a_2x_i^2 & + & \cdots & + & a_nx_i^n & = & f(x_i) \\ & & & & \vdots & & & & \vdots & & \\ a_0 & + & a_1x_n & + & a_2x_n^2 & + & \cdots & + & a_nx_n^n & = & f(x_n) \end{array} \tag{3.3}$$

Wyznacznik macierzy współczynników tego układu równań jest wyznacznikiem Vandermonde'a

$$\begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_i & x_i^2 & \cdots & x_i^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{vmatrix} \neq 0.$$

Wyznacznik Vandermonde'a jest różny od zera, jeśli $x_i \neq x_j$, $i \neq j$ $(i, j = 0, 1, \ldots, n)$, czyli układ równań (3.3) ma dokładnie jedno rozwiązanie.

powered by Misio]

1.3.2. Wzór interpolacyjny Lagrange'a

Wielomian interpolacyjny można również uzyskać bez czasochłonnego rozwiązywania układu równań (3.3). Skonstruujemy w tym celu tak zwane wielomiany pomocnicze $W_n^j(x)$, $j = 0, 1, \ldots, n$, stopnia co najwyżej n takie, że

$$W_n^j(x_i) = \delta_{ij} = \begin{cases} 0 & \text{gdy} \quad i \neq j \\ 1 & \text{gdy} \quad i = j. \end{cases} \tag{3.4}$$

Oznacza to, że wielomian $W_n^j(x)$ jest tożsamościowo równy zeru w punktach $x_0, x_1, \cdots, x_{j-1}, x_{j+1}, \cdots, x_n$, a w punkcie x_j równa się jedności. Stąd

$$W_n^j(x) = c_j(x - x_0)(x - x_1)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n). \tag{3.5}$$

Ponieważ $W_n^j(x_j) = 1$, więc

$$1 = c_j(x_j - x_0)(x_j - x_1)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\ldots(x_j - x_n). \tag{3.6}$$

Stąd po wyeliminowaniu stałej c_j ze wzorów (3.5) i (3.6) otrzymujemy

$$W_n^j(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n)}{(x_j - x_0)(x_j - x_1)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)}. \tag{3.7}$$

Dokonując sumowania tak skonstruowanych wielomianów $W_n^j(x)$, $j = 0, 1, \ldots, n$

$$W_n(x) = \sum_{j=0}^{n} W_n^j(x) y_j =$$

$$= \sum_{j=0}^{n} \frac{(x - x_0)(x - x_1)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n)}{(x_j - x_0)(x_j - x_1)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)} \cdot y_j \tag{3.8}$$

otrzymujemy wielomian stopnia co najwyżej n, spełniający wszystkie warunki wymagane przy interpolacji. Wzór (3.8) nosi nazwę wielomianu interpolacyjnego Lagrange'a. Przyjmując oznaczenie

$$\omega_{n+1}(x) = (x - x_0)(x - x_1)\cdots(x - x_n) \tag{3.9}$$

wzór (3.8) można zapisać następująco

$$W_n(x) = \sum_{j=0}^{n} \frac{\omega_{n+1}(x)}{(x - x_j)\omega_{n+1}^{(1)}(x_j)} \cdot y_j, \tag{3.10}$$

gdzie $\omega_{n+1}^{(1)}(x_j)$ jest wartością pochodnej wielomianu $\omega_{n+1}(x)$ w danym punkcie x_j.

Prowadząc obliczenia na komputerze najwygodniej jest skorzystać ze wzoru interpolacyjnego Lagrange'a w postaci

$$W_n(x) = \omega_{n+1}(x) \sum_{j=0}^{n} \frac{y_j}{(x - x_j) \prod\limits_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i)}. \tag{3.11}$$

Przybliżając funkcję $f(x)$ wielomianem interpolacyjnym $W_n(x)$, możemy obliczać jej wartości z dokładnością

$$R_n(x) = f(x) - W_n(x). \tag{3.12}$$

Można wykazać, że wartość popełnianego błędu metody przy interpolacji wyraża się zależnością

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x), \tag{3.13}$$

gdzie $\xi \in [x_0, x_n]$ jest punktem pośrednim.

W praktyce, ponieważ nie znamy wartości ξ, dla oszacowania modułu błędu metody posługujemy się wzorem

$$|R_n(x)| \leq \frac{M_{n+1}}{(n+1)!}|\omega_{n+1}(x)|, \tag{3.14}$$

gdzie

$$M_{n+1} = \max_{x_0 \leq x \leq x_n} |f^{(n+1)}(x)|. \tag{3.15}$$

Wzory (3.13)-(3.15) można stosować jedynie wówczas, gdy znana jest postać analityczna funkcji $f(x)$. Na dokładność obliczeń oprócz błędu metody oraz błędów zaokrągleń istotny wpływ może wywierać również błąd początkowy, spowodowany zbyt małą dokładnością określenia wartości funkcji w węzłach interpolacji.

Przykład 1

Znaleźć wielomian interpolacyjny, który w punktach $(1, 2, 3, 4)$ przyjmuje wartości $(3, 1, -1, 2)$.

$$\begin{aligned}
W_3(x) &= 3 \cdot \frac{(x-2)(x-3(x-4)}{(1-2)(1-3)(1-4)} + 1 \cdot \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} + \\
&- 1 \cdot \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 2 \cdot \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} = \\
&= -\frac{1}{2}(x^3 - 9x^2 + 26x - 24) + \frac{1}{2}(x^3 - 8x^2 + 19x - 12) + \\
&+ \frac{1}{2}(x^3 - 6x^2 + 11x - 6) + \frac{1}{3}(x^3 - 7x^2 + 14x - 8) = \\
&= \frac{5}{6}x^3 - 5x^2 + \frac{43}{6}x
\end{aligned}$$

Przykład 2

Oszacować dokładność, z jaką można obliczyć wartość $\ln 5.5$ przy użyciu wielomianu interpolacyjnego Lagrange'a stopnia 3, jeśli znane są wartości funkcji w następujących punktach $(4; 4.5; 5; 6)$.

powered by Misiol

Rozwiązanie

$$\begin{aligned}
f(x) &= \ln x, \\
f^{(4)}(x) &= -\frac{6}{x^4} \\
M_4 &= \max_{4 \leq x \leq 6} \left|\frac{6}{x^4}\right| = 0.024 \\
|R_4(x)| &\leq \frac{0.024}{4!}|1.5 \cdot 1 \cdot 0.5 \cdot (-0.5)| \approx 4 \cdot 10^{-4}
\end{aligned}$$

Odp. Obliczenia można wykonać z dokładnością do $4 \cdot 10^{-4}$.

Przykład 3

Obliczyć, z jaką dokładnością można wyznaczyć $\sqrt{115}$ za pomocą wzoru interpolacyjnego Lagrange'a dla funkcji $f(x) = \sqrt{x}$, jeżeli przyjmuje się następujące węzły interpolacji $x_0 = 100, x_1 = 121, x_2 = 144$.

Do oszacowania modułu błędu interpolacji wykorzystuje się wzór (1.13). Stąd

$$|R_2(\sqrt{x})| \leq \frac{1}{(2+1)!} \max_{x_0 \leq \xi \leq x_2} |(\sqrt{x})^{(3)}||(x-100)(x-121)(x-144)|.$$

W celu wykorzystania powyższego oszacowania wyznacza się pochodne odpowiednio wysokiego rzędu:

$$(\sqrt{x})^{(1)} = \frac{1}{2}x^{-\frac{1}{2}}, \quad (\sqrt{x})^{(2)} = -\frac{1}{4}x^{-\frac{3}{2}}, \quad (\sqrt{x})^{(3)} = -\frac{3}{8}x^{-\frac{5}{2}}.$$

Stąd

$$\max_{x_0 \leq \xi \leq x_2} \left|\frac{3}{8}\xi^{-\frac{5}{2}}\right| = \frac{3}{8}\frac{1}{\sqrt{100^5}} = \frac{3}{8} \cdot 10^{-5}.$$

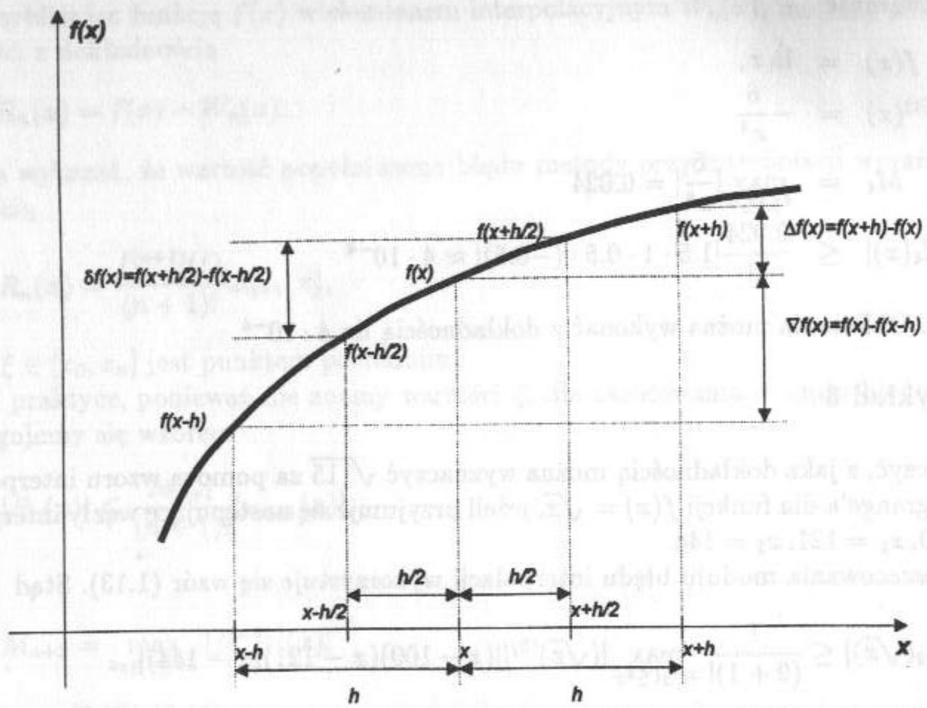
Czyli

$$\begin{aligned}
|R_2(\sqrt{115}| &\leq \frac{1}{6} \cdot \frac{3}{8} \cdot 10^{-5}|(115-100)(115-121)(115-144)| = \\
&= \frac{1}{16}10^{-5} \cdot 5 \cdot 6 \cdot 29 \approx 1.6 \cdot 10^{-3}.
\end{aligned}$$

1.3.3. Operatory różnicowe

Do rozwiązywania zagadnień numerycznych (np. interpolacji) wykorzystywane są operatory różnicowe: Rozpatrzmy następujące operatory różnicowe:

1. Operator różnicy zwykłej $\Delta f(x) \stackrel{\text{df}}{=} f(x+h) - f(x)$,
2. Operator różnicy wstecznej $\nabla f(x) \stackrel{\text{df}}{=} f(x) - f(x-h)$,
3. Operator różnicy centralnej (symetrycznej) $\delta f(x) \stackrel{\text{df}}{=} f(x+\frac{h}{2}) - f(x-\frac{h}{2})$.



Rys. 1.4. Graficzna interpretacja operatorów różnicowych

Interpretację graficzną operatorów różnicowych przedstawia rys. 1.4. Wymienione powyżej operatory są operatorami liniowymi, tzn.:

$$\Delta\left(\sum_{k=1}^{k=m}(\alpha_k f_k(x))\right) = \sum_{k=1}^{k=m}\alpha_k\,\Delta\, f_k(x) \text{ dla } \alpha_k \in R,\ k=1,2,\ldots m, \tag{3.16}$$

$$\nabla\left(\sum_{k=1}^{k=m}(\alpha_k f_k(x))\right) = \sum_{k=1}^{k=m}\alpha_k\,\nabla\, f_k(x) \text{ dla } \alpha_k \in R,\ k=1,2,\ldots m, \tag{3.17}$$

$$\delta\left(\sum_{k=1}^{k=m}(\alpha_k f_k(x))\right) = \sum_{k=1}^{k=m}\alpha_k\,\delta f_k(x) \text{ dla } \alpha_k \in R,\ k=1,2,\ldots m. \tag{3.18}$$

Ponadto są to operatory przemienne, tzn.:

$$\nabla(\Delta f(x)) = \Delta(\nabla f(x)), \tag{3.19}$$

$$\delta(\nabla f(x)) = \nabla(\delta f(x)), \tag{3.20}$$

$$\Delta(\delta f(x)) = \delta(\Delta f(x)). \tag{3.21}$$

Potęgowanie operatorów różnicowych zdefiniowane jest w następujący sposób:

$$\Delta^k f(x) \stackrel{\text{df}}{=} \Delta(\Delta^{k-1} f(x)),\ k=1,2,3\ldots, \tag{3.22}$$

$$\nabla^k f(x) \stackrel{\text{df}}{=} \nabla(\nabla^{k-1} f(x)),\ k=1,2,3\ldots, \tag{3.23}$$

$$\delta^k f(x) \stackrel{\text{df}}{=} \delta(\delta^{k-1} f(x)),\ k=1,2,3\ldots. \tag{3.24}$$

powered by Misiol

Pomiędzy poszczególnymi operatorami różnicowymi zachodzą między innymi następujące związki:

$$(\Delta + I) = (I - \nabla)^{-1}, \tag{3.25}$$

gdzie I-operator identycznościowy,

$$\delta^2 = \Delta\nabla = \nabla\Delta, \tag{3.26}$$

$$\delta = \Delta(I+\Delta)^{-0.5}, \tag{3.27}$$

$$\delta = \nabla(I-\nabla)^{-0.5}, \tag{3.28}$$

$$\Delta = (I-\nabla)^{-1} - I, \tag{3.29}$$

$$\nabla = I - (I+\Delta)^{-1}. \tag{3.30}$$

Przykłady

1. $\Delta(\Delta f(x)) = \Delta(f(x+h) - f(x)) = f(x+2h) - f(x+h) - [f(x+h) - f(x)] = f(x+2h) - 2f(x+h) + f(x)$

2. $\nabla(\Delta f(x)) = \nabla(f(x+h) - f(x)) = f(x+2h) - f(x) - [f(x) - f(x-h)] = f(x+h) - 2f(x) + f(x-h)$

3. $\delta(\nabla f(x)) = \delta(f(x) - f(x-h)) = f(x+\frac{h}{2}) - f(x-\frac{h}{2}) - [f(x-\frac{h}{2}) - f(x-\frac{3}{2}h)] = f(x+\frac{h}{2}) - 2f(x-\frac{h}{2}) + f(x-\frac{3}{2}h)$

W wielu zadaniach numerycznych zachodzi potrzeba obliczania wartości operatorów różnic zwykłych, wstecznych lub centralnych (symetrycznych). Różnice zwykłe oblicza się stosując następujący schemat:

x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$	$\Delta^4 f(x)$
x_0					
		$\Delta f(x_0)$			
x_0+h	$f(x_0+h)$		$\Delta^2 f(x_0)$		
		$\Delta f(x_0+h)$		$\Delta^3 f(x_0)$	
x_0+2h	$f(x_0+2h)$		$\Delta^2 f(x_0+h)$		$\Delta^4 f(x_0)$
		$\Delta f(x_0+2h)$		$\Delta^3 f(x_0+h)$	
x_0+3h	$f(x_0+3h)$		$\Delta^2 f(x_0+2h)$		
		$\Delta f(x_0+3h)$			
x_0+4h	$f(x_0+4h)$				

(3.31)

Przy obliczaniu różnic wstecznych posługujemy się następującym schematem:

x	$f(x)$	$\nabla f(x)$	$\nabla^2 f(x)$	$\nabla^3 f(x)$	$\nabla^4 f(x)$
x_0-4h	$f(x_0-4h)$				
		$\nabla f(x_0-3h)$			
x_0-3h	$f(x_0-3h)$		$\nabla^2 f(x_0-2h)$		
		$\nabla f(x_0-2h)$		$\nabla^3 f(x_0-h)$	
x_0-2h	$f(x_0-2h)$		$\nabla^2 f(x_0-h)$		$\nabla^4 f(x_0)$
		$\nabla f(x_0-h)$		$\nabla^3 f(x_0)$	
x_0-h	$f(x_0-h)$		$\nabla^2 f(x_0)$		
		$\nabla f(x_0)$			
x_0	$f(x_0)$				

(3.32)

Do wyznaczania różnic centralnych (symetrycznych) wykorzystuje się następujący schemat obliczeń:

x	$f(x)$	$\delta f(x)$	$\delta^2 f(x)$	$\delta^3 f(x)$	$\delta^4 f(x)$
x_0-2h	$f(x_0-2h)$				
		$\delta f(x_0-\frac{3}{2}h)$			
x_0-h	$f(x_0-h)$		$\delta^2 f(x_0-h)$		
		$\delta f(x_0-\frac{h}{2})$		$\delta^3 f(x_0-\frac{h}{2})$	
x_0	$f(x_0)$		$\delta^2 f(x_0)$		$\delta^4 f(x_0)$
		$\delta f(x_0+\frac{h}{2})$		$\delta^3 f(x_0+\frac{h}{2})$	
x_0+h	$f(x_0+h)$		$\delta^2 f(x_0+h)$		
		$\delta f(x_0+\frac{3}{2}h)$			
x_0+2h	$f(x_0+2h)$				

(3.33)

1.3.4. Wzory interpolacyjne Newtona

Wzory interpolacyjne Newtona są stosowane do rozwiązywania tego samego zagadnienia co wzór Lagrange'a, przy dodatkowym wymaganiu nałożonym na węzły interpolacyjne, aby były one rozmieszczone w jednakowych odstępach od siebie, tj.

$$x_i = x_0 + i \cdot h, \quad i = 0, 1, \ldots, n, \quad h = const. \tag{3.34}$$

Należy zatem znaleźć wielomian interpolacyjny $W_n(x)$ stopnia co najwyżej n, który w węzłach interpolacji x_i $(i = 0, 1, 2, \ldots, n)$ przyjmuje wartości

$$W_n(x_i) = f(x_i), \quad i = 0, 1, \ldots, n. \tag{3.35}$$

Warunek ten jest równoważny warunkowi

$$\Delta^m W_n(x_0) = \Delta^m y_0, \quad m = 0, 1, \ldots, n, \tag{3.36}$$

gdzie Δ jest operatorem obliczania różnic skończonych, zwykłych.

Rozważmy następującą postać wielomianu interpolacyjnego

$$\begin{aligned} W_n(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + \\ &+ a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}), \end{aligned} \tag{3.37}$$

spełniającego warunki (3.35). Współczynniki a_i, $i = 0, 1, \ldots, n$ można wyznaczyć w sposób następujący.

Przyjmując $x = x_0$, otrzymujemy

$$W_n(x_0) = y_0 = a_0. \tag{3.38}$$

W celu wyznaczenia współczynnika a_i tworzymy różnicę skończoną rzędu pierwszego

$$W_n(x) = a_1 h + 2h a_2(x - x_0) + \cdots + n h a_n(x - x_0)(x - x_1) \cdots (x - x_{n-2}). \tag{3.39}$$

Podstawiając $x = x_0$ i korzystając ze wzoru (3.36), mamy

$$W_n(x_0) = y_0 = a_1 h, \quad \text{stąd} \quad a_1 = \frac{y_0}{1!h}. \tag{3.40}$$

powered by Misiol

Analogicznie obliczamy pozostałe współczynniki a_i według wzoru

$$a_i = \frac{\Delta^i y_0}{i! h^i}, \quad i = 0, 1, \ldots, n, \tag{3.41}$$

przyjmując, że $0! = 1$ oraz $\Delta^0 y = y$. Podstawiając wyznaczone wartości współczynników do wzoru (3.37) otrzymujemy

$$\begin{aligned} W_n(x) &= y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2! h^2}(x - x_0)(x - x_1) + \cdots + \\ &+ \frac{\Delta^n y_0}{n! h^n}(x - x_0)(x - x_1) \cdots (x - x_{n-1}). \end{aligned} \tag{3.42}$$

Wzór ten nosi nazwę pierwszego wzoru interpolacyjnego Newtona. W praktycznych zastosowaniach wygodnie jest posługiwać się wielomianem interpolacyjnym Newtona, wprowadzając nową zmienną znormalizowaną

$$t = \frac{x - x_0}{h}, \quad t \in [0, n]. \tag{3.43}$$

Wtedy wzór (3.42) można przekształcić do postaci

$$\begin{aligned} W_n(x) &= W_n(x_0 + th) = y_0 + t \Delta y_0 + \frac{t(t-1)}{2!} \Delta^2 y_0 + \cdots + \\ &+ \frac{t(t-1)(t-2) \cdots (t-n+1)}{n!} \Delta^n y_0. \end{aligned} \tag{3.44}$$

Posługując się trójkątną tablicą różnic przy wyznaczaniu wielomianu do wzoru (3.44) podstawiamy różnice „schodzące" w dół po odpowiedniej przekątnej. Dlatego też wygodnie jest stosować ten wzór do interpolacji funkcji w pobliżu wartości początkowej x_0.

Przy interpolacji w pobliżu końca przedziału stosuje się inny wzór, który odpowiada wielomianowi interpolacyjnemu o postaci

$$\begin{aligned} W_n(x) &= a_0 + a_1(x - x_n) + a_2(x - x_n)(x - x_{n-1}) + \cdots + \\ &+ a_n(x - x_n)(x - x_{n-1}) \cdots (x - x_1). \end{aligned} \tag{3.45}$$

Współczynniki a_i, $i = 0, 1, \ldots, n$ można wyznaczyć tak samo jak poprzednio ze wzoru

$$a_i = \frac{\Delta^i y_{n-1}}{i! h^i}, \quad i = 0, 1, \ldots, n, \tag{3.46}$$

który jest równoważny – po zastąpieniu różnic zwykłych różnicami wstecznymi – wzorowi

$$a_i = \frac{\nabla^i y_0}{i! h}, \quad i = 0, 1, \ldots, n, \tag{3.47}$$

gdzie ∇ jest operatorem obliczania różnic wstecznych. Podstawiając współczynniki a_i wyznaczone ze wzoru (3.46) do (3.45), otrzymujemy

$$\begin{aligned} W_n(x) &= y_n + \frac{\Delta y_{n-1}}{h}(x - x_n) + \frac{\Delta^2 y_{n-2}}{2! h^2}(x - x_n)(x - x_{n-1}) + \cdots + \\ &+ \frac{\Delta^n y_0}{n! h^n}(x - x_n)(x - x_{n-1}) \cdots (x - x_1). \end{aligned} \tag{3.48}$$

Wzór ten nosi nazwę drugiego wzoru interpolacyjnego Newtona. Dla celów praktycznych wzór ten wygodnie jest przekształcić wprowadzając zmienną znormalizowaną

$$t = \frac{x - x_n}{h}, \quad t \in [-n, 0] \tag{3.49}$$

do postaci

$$\begin{aligned} W_n(x_n + th) &= y_n + q \triangle y_{n-1} + \frac{t(t+1)}{2!} \triangle^2 y_{n-2} + \ldots + \\ &+ \frac{t(t+1)(t+2)\ldots(t+n-1)}{n!} \triangle^n y_0. \end{aligned} \tag{3.50}$$

Wyznaczając podobnie współczynniki a_i za pomocą różnic wstecznych i wprowadzając zmienną znormalizowaną uzyskuje się inną postać drugiego wzoru interpolacyjnego Newtona

$$\begin{aligned} W_x(x_0 + th) &= y_0 + t \nabla y_0 + \frac{t(t+1)}{2!} \nabla^2 y_0 + \cdots + \\ &+ \frac{t(t+1)(t+2)\cdots(t+n-1)}{n!} \nabla^n y_0. \end{aligned} \tag{3.51}$$

Można wykazać, że błąd metody dla pierwszego wzoru Newtona ma postać

$$R_n(x_0 + th) = h^{n+1} \frac{t(t-1)\cdots(t-n)}{(n+1)!} f^{(n+1)}(\xi), \tag{3.52}$$

natomiast dla drugiego

$$R_n(x_0 + th) = h^{n+1} \frac{t(t+1)\cdots(t+n)}{(n+1)!} f^{(n+1)}(\xi), \tag{3.53}$$

przy czym ξ jest punktem pośrednim w przedziale $[x_0, x_n]$. Ponieważ wartości ξ nie znamy, w praktyce posługujemy się następującymi oszacowaniami modułów błędów interpolacji

$$|R_n(x_0 + th)| \le \max_{\xi \in [x_0, x_n]} \left| f^{(n+1)}(\xi) \frac{h^{n+1}}{(n+1)!} t(t-1)\cdots(q-n) \right| \tag{3.54}$$

dla pierwszego wzoru Newtona oraz

$$|R_n(x_0 + th)| \le \max_{\xi \in [x_0, x_n]} \left| f^{(n+1)}(\xi) \frac{h^{n+1}}{(n+1)!} t(t+1)\cdots(q+n) \right| \tag{3.55}$$

dla drugiego wzoru.

Posługując się w praktyce wzorami Newtona obcinamy wyrazy zawierające różnice, które w granicach przyjętej dokładności można uważać za stałe. Przyjmując więc, że $\triangle^{n+1} y$ są dla funkcji $y = f(x)$ prawie stałe, a h jest dostatecznie małe oraz uwzględniając, że

$$f^{(n+1)}(x) = \lim_{h \to 0} \frac{\triangle^{n+1} y}{h^{n+1}} \tag{3.56}$$

możemy w przybliżeniu zapisać

$$f^{(n+1)}(\xi) \approx \frac{\triangle^{n+1} y_0}{h^{n+1}}. \tag{3.57}$$

powered by Misiol

Wtedy błąd pierwszego wzoru interpolacyjnego w przybliżeniu jest równy

$$R_n(x_0 + th) \approx \frac{t(t-1)\ldots(q-n)}{(n+1)!} \triangle^{n+1} y_0. \tag{3.58}$$

Podobnie dla wzoru drugiego błąd wynosi wtedy

$$R_n(x_0 + th) \approx \frac{t(t+1)\ldots(t+n)}{(n+1)!} \nabla^{n+1} y_0. \tag{3.59}$$

Wzory te pozwalają określić błąd interpolacji, gdy nie jest znana postać analityczna funkcji $f(x)$ lub gdy jej postać jest skomplikowana.

Wzory interpolacyjne Newtona z operatorami różnicy zwykłej i różnicy wstecznej można również uzyskać wykorzystując rozwinięcia w szeregi potęgowe. Niech $Ef(x) \stackrel{df}{=} f(x+h)$. Zauważmy, że

$$f(x_0 + th) = E^t f(x_0). \tag{3.60}$$

Zatem podstawiając $E = (I + \triangle)$ otrzymujemy:

$$\begin{aligned} f(x_0 + th) &= E^t f(x) = (I + \triangle)^t f(x_0) = \\ &= f(x_0) + \binom{t}{1} \triangle f(x_0) + \binom{t}{2} \triangle^2 f(x_0) + \cdots + \end{aligned} \tag{3.61}$$

$$+ \binom{t}{n} \triangle^n f(x_0) + R_n(x_0 + th), \tag{3.62}$$

gdzie

$$\binom{t}{k} \stackrel{df}{=} \frac{t(t-1)(t-2)\cdots(t-k+1)}{k!}, \quad t \in R, k \in N$$

jest tzw. uogólnionym symbolem Newtona. Stąd wielomian interpolacyjny Newtona z operatorem różnicy zwykłej $\triangle$ wyraża się następującym wzorem

$$W_n(x_0 + th) = \sum_{k=0}^{k=n} \binom{t}{k} \triangle^k f(x_0), \tag{3.63}$$

natomiast błąd interpolacji

$$R_n(x_0 + th) = \frac{h^{n+1}}{(n+1)!} f^{(n+1)}(\xi) t(t-1)(t-2)\ldots(t-n), \tag{3.64}$$

gdzie $\xi \in [x_0, x_n]$ jest punktem pośrednim. Jeżeli natomiast zastosujemy podstawienie $E = (I - \nabla)^{-1}$, to otrzymamy wzór interpolacyjny Newtona z operatorem różnicy wstecznej ∇

$$W_n(x_0 + th) = f(x_0) + t \nabla f(x_0) + \frac{t(t+1)}{2!} \nabla^2 f(x_0) + \tag{3.65}$$

$$+ \frac{t(t+1)(t+2)}{3!} \nabla^3 f(x_0) + \cdots + \tag{3.66}$$

$$+ \frac{t(t+1)(t+2)\cdots(t+n)}{n!} \nabla^n f(x_0).$$

Wówczas błąd interpolacji dany jest wzorem

$$R_n(x_0+th)=\frac{h^{n+1}}{(n+1)!}f^{(n+1)}(\xi)t(t+1)(t+2)\cdots(t+n), \tag{3.67}$$

gdzie $\xi\in[x_{-n},x_0]$ jest punktem pośrednim.

Przykład 4

Znaleźć wielomian interpolacyjny Newtona dla funkcji $f(x)$ przyjmującej w punktach $(1;1.5;2;2.5)$ wartości $(2;2.5;3.5;4.0)$.

Tworzymy tablicę różnic.

x_i	$f(x_i)$	$\Delta f(x_i)$	$\Delta^2 f(x_i)$	$\Delta^3 f(x_i)$
1.0	2.0			
		0.5		
1.5	2.5		0.5	
		1.0		−1.0
2.0	3.5		−0.5	
		0.5		
2.5	4.0			

Podstawiając obliczone wartości różnic do wzoru (3.42), otrzymujemy:

$$\begin{aligned}W_3(x) &= 2+\frac{0.5}{0.5}(x-1)+\frac{0.5}{2!\cdot 0.5^2}(x-1)(x-1.5)+\\ &+\frac{-1.0}{3!\cdot 0.5^3}(x-1)(x-1.5)(x-2)=\\ &=-\frac{4}{3}x^3+7x^2-10\frac{1}{6}x+6\frac{1}{2}.\end{aligned}$$

Przykład 5

Dana jest tablica wartości funkcji $y=e^x$.

x_i	2.0	2.1	2.2	2.3
$f(x_i)$	7.3891	8.1662	9.0250	9.9742

Obliczyć wartość funkcji w punkcie $x=2.25$, korzystając z drugiego wzoru interpolacyjnego Newtona i oszacować błąd interpolacji.

x_i	$f(x_i)$	$\nabla f(x_i)$	$\nabla^2 f(x_i)$	$\nabla^3 f(x_i)$
2.0	7.3891			
		0.7771		
2.1	8.1662		0.0817	
		0.8588		0.0087
2.2	9.0250		0.0904	
		0.9492		
2.3	9.9742			

powered by Misiol

$$t=\frac{2.25-2.3}{0.1}=-0.5$$

Podstawiając obliczone wartości różnic oraz t do wzoru (3.51), otrzymujemy:

$$\begin{aligned}W_3(2.25) &= 9.9742-0.5\cdot 0.9492+\frac{-0.5(-0.5+1)}{2!}0.0904+\\ &+\frac{-0.5(-0.5+1)(-0.5+2)}{3!}0.0087=\\ &=9.9742-0.4746-0.00113-0.0005=9.4878.\end{aligned}$$

Błąd interpolacji określamy na podstawie zależności (3.36)

$$|R_3(2.25)|\le\max_{\xi\in[2.0;2.3]}\left|e^{\xi}\cdot\frac{0.1^4}{4!}(-0.5)\cdot(-0.5)\cdot 1.5\cdot 2.5\right|\approx 4x10^{-5}.$$

1.3.5. Optymalny dobór węzłów interpolacji

Jak widać ze wzoru (3.15), wielkość błędu interpolacji zależy od $f^{(n+1)}(\xi)$, która jest uzależniona od własności funkcji i nie mamy na nią wpływu, oraz od stopnia wielomianu i od wartości $\omega_{n+1}(x)$, która zależy od wyboru węzłów interpolacji. Możemy tak wybrać węzły x_i, $(i=0,1,\ldots,n)$, aby wartość

$$\max_{x\in[x_0,x_n]}|(x-x_0)(x-x_1)\cdots(x-x_n)| \tag{3.68}$$

była jak najmniejsza. Zagadnienie to sformułowane zostało przez Czebyszewa i nazywa się zagadnieniem znajdowania wielomianu algebraicznego najlepiej przybliżającego zero na zadanym przedziale. Rozwiązaniem tego zadania jest wielomian Czebyszewa „zawężony" do przedziału $[-1,1]$ o postaci

$$T_k(z)=cos(k\cdot\arccos z),\quad k=0,1,2,\ldots,;\ z\in[0,1]. \tag{3.69}$$

Wielomian (3.69) stopnia $n+1$ ma, w przedziale $[-1,+1]$, $n+1$ różnych pierwiastków w punktach

$$z_m=\cos\left(\frac{2m+1}{2n+1}\pi\right),\quad m=0,1,\ldots,n. \tag{3.70}$$

Wartości węzłów Czebyszewa w przedziale $[a,b]$ uzyskujemy przez transformację

$$x_m=\frac{1}{2}[(b-a)z_m+(a+b)],\quad m=0,1,\ldots,n. \tag{3.71}$$

Węzły te są zagęszczone przy końcach przedziału.

Można wykazać, że dla węzłów Czebyszewa oszacowanie błędu interpolacji jest następujące

$$R_n(x)\le\frac{M_{n+1}}{(n+1)!}\cdot\frac{(b-a)^{n+1}}{2^{2n+1}}, \tag{3.72}$$

gdzie M_{n+1} jest takie, jak we wzorze (3.15).

Z optymalnego doboru węzłów interpolacji korzystamy w przypadku planowania eksperymentów, w których uzyskujemy tablice wartości funkcji. Wielomian interpolacyjny wyznaczony w ten sposób daje najmniejszą wartość maksymalną oszacowania błędu.

1.3.6. Wzory interpolacyjne z operatorami różnicy symetrycznej

Wyprowadzone powyżej wzory interpolacyjne Newtona oparte na operatorach różnic zwykłych i różnic wstecznych wykorzystują jedynie wartości funkcji $f(x)$ dla argumentów leżących z jednej (prawej lub lewej) strony punktu x_0. Istnieją również wzory interpolacyjne wykorzystujące wartość funkcji $f(x)$ dla argumentów leżących po lewej i prawej stronie punktu x_0. Wzory te oparte są na operatorze różnicy centralnej (symetrycznej) δ. Najczęściej stosowanym wzorem należącym do tej kategorii jest wzór interpolacyjny Stirlinga.

Niech $x_{-n}, x_{-n+1}, \ldots, x_{-i}, \ldots, x_{-1}, x_0, x_1, \ldots, x_i, \ldots, x_{n-1}, x_n$ będą równoodległymi węzłami interpolacji, tzn. $x_{i+1} - x_i = h$ dla $i = -n, -n+1, \ldots, -1, 0, 1, \ldots, n-1$. Dla $x \in [-x_n, x_n]$ określmy zmienną znormalizowaną $t = \frac{x-x_0}{h}$, zatem $t \in [-n, n]$. Wielomian interpolacyjny Stirlinga ma postać

$$\begin{aligned} W_{2n}(x_0+th) &= f(x_0) + \frac{1}{2}\binom{t}{1}\left\{\delta f(x_0 - \frac{h}{2}) + \delta f(x_0 + \frac{h}{2})\right\} + \\ &+ \frac{1}{2}t\binom{t}{1}\delta^2 f(x_0) + \cdots + \frac{1}{2}\binom{t+n-1}{2n-1}\left\{\delta^{2n-1} f(x_0 - \frac{h}{2}) + \right. \\ &\left. + \delta^{2n-1} f(x_0 + \frac{h}{2})\right\} + \frac{t}{2n}\binom{t+n-1}{2n-1}\delta^{2n} f(x_0). \end{aligned} \tag{3.73}$$

Można wykazać, że błąd interpolacji $R_{2n}(x_0+th)$ dla wzoru interpolacyjnego Stirlinga jest określony wzorem

$$R_{2n}(x_0+th) = \frac{h^{2n+1}}{(2n+1)!} f^{(2n+1)}(\xi) t(t^2-1^2)(t^2-2^2)\cdots(t^2-n^2), \tag{3.74}$$

gdzie ξ jest punktem pośrednim w przedziale $[-x_n, x_n]$. Ponieważ nie znamy wartości ξ, więc w praktyce posługujemy się następującym oszacowaniem modułu błędu interpolacji

$$|R_{2n}(x_0+th)| \le \max_{x_{-n} \le \xi \le x_n} \left| f^{(2n+1)}(\xi) \frac{h^{2n+1}}{(2n+1)!} t(t^2-1^2)(t^2-2^2)\cdots(t^2-n^2) \right|. \tag{3.75}$$

Oprócz wzoru interpolacyjnego Stirlinga stosuje się również następujące wzory interpolacyjne oparte na operatorach różnicy symetrycznej (centralnej): wzór interpolacyjny Bessela, Everetta i Gaussa.

Wzór interpolacyjny Bessela

Następująca zależność wyraża wzór interpolacyjny Bessela

$$\begin{aligned} W_{2n+1}(x_0+th) &= \frac{1}{2}[f(x_0) + f(x_0+h)] + t\delta f(x_0 + \frac{h}{2}) + \\ &+ \frac{1}{2}\binom{t}{2}[\delta^2 f(x_0) + \delta^2 f(x_0+h) + \frac{t-\frac{1}{2}}{3}\binom{t}{2}\delta^3 f(x_0 + \frac{h}{2}) + \cdots \\ &\cdots + \frac{1}{2}\binom{t+n-1}{2n}[\delta^{2n} f(x_0) + \delta^{2n} f(x_0+h)] + \\ &+ \frac{t-\frac{1}{2}}{2n+1}\binom{t+n-1}{2n}\delta^{2n+1} f(x_0 + \frac{h}{2}). \end{aligned} \tag{3.76}$$

powered by Mistol

Reszta R_{2n+1} określona jest wzorem

$$\begin{aligned} R_{2n+1}(x_0+th) &= \frac{h^{2n+2}}{(2n+2)!} f^{(2n+2)}(\xi) t(t^2-1^2)(t^2-2^2)\cdots \\ &\cdots [t^2-(n-1)^2](t-n), \end{aligned} \tag{3.77}$$

gdzie ξ jest punktem pośrednim.

Wzór interpolacyjny Everetta

Następująca zależność wyraża wzór interpolacyjny Everetta

$$\begin{aligned} W_{2n+2}(x_0+th) &= (1-t)f(x_0) + \binom{2-t}{3}\delta^2 f(x_0) + \binom{3-t}{5}\delta^4 f(x_0) + \\ &+ tf(x_0+h) + \binom{t+1}{3}\delta^2 f(x_0+h) + \\ &+ \binom{t+2}{5}\delta^4 f(x_0+h) + \cdots \\ &\cdots + \binom{n+1-t}{2n+1}\delta^{2n} f(x_0) + \binom{t+n}{2n+1}\delta^{2n} f(x_0+h). \end{aligned} \tag{3.78}$$

Reszta R_{2n+2} określona jest wzorem

$$\begin{aligned} R_{2n+2}(x_0+th) &= \frac{h^{2n+2}}{(2n+2)!} f^{(2n+2)}(\xi) t(t^2-1^2)(t^2-2^2)\cdots \\ &\cdots (t^2-n^2)(t-n-1), \end{aligned} \tag{3.79}$$

gdzie ξ jest punktem pośrednim.

Wzór interpolacyjny Gaussa

Następująca zależność wyraża wzór interpolacyjny Gaussa

$$\begin{aligned} W_{2n}(x_0+th) &= f(x_0) + \binom{t}{1}\delta f(x_0 + \frac{h}{2}) + \\ &+ \binom{t}{2}\delta^2 f(x_0) + \binom{t+1}{3}\delta^3 f(x_0 + \frac{h}{2}) + \cdots \\ &\cdots + \binom{t+n-1}{2n-1}\delta^{2n-1} f(x_0 + \frac{h}{2}) + \binom{t+n-1}{2n}\delta^{2n} f(x_0). \end{aligned} \tag{3.80}$$

Reszta R_{2n} określona jest wzorem

$$R_{2n}(x_0+th) = \frac{h^{2n+1}}{(2n+1)!} f^{(2n+1)}(\xi) t(t^2-1^2)(t^2-2^2)\cdots(t^2-n^2), \tag{3.81}$$

gdzie ξ jest punktem pośrednim.

1.3.7. Przykład

Wyznaczyć wielomian interpolacyjny dla funkcji $f(x)$ danej następującą tabelą:

i	0	1	2	3	4
x_i	1	1.5	2	2.5	3
$f(x_i)$	1	3	4	2	0

(3.82)

Mamy: $n = 4$, węzły interpolacji równoodległe, $h = 0.5$. Wyznaczymy wielomian interpolacyjny wykorzystując różne wzory interpolacyjne.

1. Wielomian interpolacyjny Lagrange'a

Wielomiany pomocnicze $w_i(x)$, $i = 0, 1, 2, 3, 4$ mają postać następującą:

$$w_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)} = \frac{(x-1.5)(x-2)(x-2.5)(x-3)}{1-1.5)(1-2)(1-2.5)(1-3)},$$

$$w_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)} = \frac{(x-1)(x-2)(x-2.5)(x-3)}{1.5-1)(1.5-2)(1.5-2.5)(1.5-3)},$$

$$w_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)} = \frac{(x-1)(x-1.5)(x-2.5)(x-3)}{2-1)(2-1.5)(2-2.5)(2-3)},$$

$$w_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_1-x_4)} = \frac{(x-1)(x-1.5)(x-2)(x-3)}{2.5-1)(2.5-1.5)(2.5-2)(1.5-3)},$$

$$w_4(x) = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_3)} = \frac{(x-1)(x-1.5)(x-2)(x-2.5)}{3-1)(3-1.5)(3-2)(3-2.5)}.$$

Stąd

$$W_4(x) = \sum_{i=0}^{i=4} w_i(x) f(x_i) = w_0(x) + 3w_1(x) + 4w_2(x) + 2w_3(x) = \frac{10}{3}x^4 - 26x^3 + \frac{415}{6}x^2 - \frac{145}{2}x + 27.$$

powered by Misiol

2. Wielomian interpolacyjny Newtona

Ponieważ węzły interpolacji są równoodległe, więc do wyznaczenia wielomianu interpolacyjnego można posłużyć się wzorem interpolacyjnym Newtona, przyjmując $x_0 = 1$. Stąd:

$$W_4(1+0.5t) = \frac{t(t-1)(t-2)(t-3)(t-4)}{4!}\sum_{i=0}^{i=4}\frac{(-1)^{4-i}\binom{4}{i}}{(t-i)} f(x_i) =$$
$$= \frac{t(t-1)(t-2)(t-3)(t-4)}{24}\Big[\frac{1}{t}\binom{4}{0} f(x_0) -$$
$$-\frac{1}{t-1}\binom{4}{1} f(x_1) + \frac{1}{t-2}\binom{4}{2} f(x_2) +$$
$$+\frac{1}{t-3}\binom{4}{3} f(x_3) + \frac{1}{t-4}\binom{4}{4} f(x_4)\Big] =$$
$$= \frac{t(t-1)(t-2)(t-3)(t-4)}{24}\Big[\frac{1}{t} - \frac{4\cdot 3}{t-1} + \frac{6\cdot 4}{t-2} - \frac{1}{t-3}\cdot 2 + \frac{1}{t-4}\cdot 0\Big] =$$
$$= \frac{t(t-1)(t-2)(t-3)(t-4)}{24}\Big[\frac{1}{t} - \frac{12}{t-1} + \frac{24}{t-2} - \frac{2}{t-3}\Big]\Big|_{t=\frac{x-1}{0.5}} =$$
$$= \frac{12}{3}x^4 - 26x^3 + \frac{415}{6}x^2 - \frac{145}{2}x + 27.$$

3. Wielomian interpolacyjny Newtona z operatorem różnicy zwykłej Δ

Najpierw wyznaczymy odpowiednie operatory różnicowe, stosując schematy różnicowe podane w podrozdziale 1.3.3.

x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$	$\Delta^4 f(x)$
1	1				
		2			
1.5	3		-1		
		1		-2	
2	4		-3		5
		-2		3	
2.5	2		0		
		-2			
3.0	0				

Na podstawie schematu obliczeń uzyskuje się następujące równości:

$$W_4(1+0.5t) = f(x_0) + \binom{t}{1}\Delta f(x_0) + \binom{t}{2}\Delta^2 f(x_0) +$$
$$+\binom{t}{3}\Delta^3 f(x_0) + \binom{t}{4}\Delta^4 f(x_0) =$$
$$= f(1) + t\,\Delta f(1) + \frac{t(t-1)}{2!}\Delta^2 f(1) +$$

$$+\frac{t(t-1)(t-2)}{3!}\Delta^3 f(1)+\frac{t(t-1)(t-2)(t-3)}{4!}\Delta^4 f(1)=$$
$$=1+2t-\frac{1}{2}t(t-1)-\frac{1}{3}t(t-1)(t-2)+\frac{5}{24}t(t-1)(t-2)(t-3)|_{t=\frac{x-1}{0.5}}=$$
$$=\frac{10}{3}x^4-26x^3+\frac{415}{6}x^2-\frac{145}{2}x+27.$$

4. Wielomian interpolacyjny Newtona z operatorem różnicy wstecznej ∇

W tym przypadku wybieramy $x_0=3$. Zatem:

$$W_4(x_0+th)=W_4(3+0.5t)=f(x_0)+t\nabla f(x_0)+\frac{t(t+1)}{2!}\nabla^2 f(x_0)+$$
$$+\frac{t(t+1)(t+2)}{3!}\nabla^3 f(x_0)+\frac{t(t+1)(t+2)(t+3)}{4!}\nabla^4 f(x_0)=$$
$$=-2t+\frac{1}{2}t(t+1)(t+2)+\frac{5}{24}t(t+1)(t+2)(t+3)|_{t=\frac{x-3}{0.5}}=$$
$$\frac{10}{3}x^4-26x^3+\frac{415}{6}x^2-\frac{145}{2}x+27.$$

5. Wielomian interpolacyjny Stirlinga

W tym przypadku wybieramy punkt $x_0=2$.

$$W_4(2+0.5t)=f(x_0)+\frac{1}{2}\binom{t}{1}[\delta f(x_0-0.5h)+\delta f(x_0+0.5h)]+$$
$$+\frac{1}{2}t\binom{t}{1}\delta^2 f(x_0)=4+\frac{1}{2}t(1-2)+\frac{1}{2}t^2(-3)+$$
$$+\frac{1}{2}\binom{t+1}{3}(-2+3)+\frac{1}{4}t\binom{t+4}{3}5=$$
$$=4-0.5t-1.5t^2-\frac{1}{12}(t+1)t(t-1)+\frac{5}{24}t^2(t+1)(t-1)|_{t=\frac{x-2}{0.5}}=$$
$$=\frac{10}{3}x^4-26x^3+\frac{415}{6}x^2-\frac{145}{2}x+27$$

Wniosek 1

Przedstawiony przykład ilustruje generalny wniosek o tym, że przy zadanych węzłach interpolacji wielomian interpolacyjny dla danej funkcji $f(x)$, wyznaczony przy użyciu różnych wzorów interpolacyjnych, jest zawsze taki sam. Należy jednak wyraźnie zaznaczyć, że wielomiany interpolacyjne zmiennej znormalizowanej t są w każdym przypadku inne. Dopiero po podstawieniu $t=\frac{x-x_0}{h}$ uzyskuje się wielomian $W_n(x)$. Podstawienie odwrotne zależy do punktu startowego x_0.

powered by Misiol

Wniosek 2

Przy stosowaniu do tworzenia wzorów interpolacyjnych różnych operatorów różnicowych należy odpowiednio wybrać punkt startowy x_0. Wybór punktu startowego x_0 powinien gwarantować możliwość wyznaczenia na podstawie właściwego schematu różnicowego potęg odpowiednich operatorów różnicowych.

Wniosek 3

Oszacowanie modułu błędu interpolacji jest możliwe jedynie w sytuacji, gdy funkcja $f(x)$ jest zadana w postaci wzoru. W tym przypadku oszacowanie to zależy nie tylko od przebiegu funkcji $f(x)$ ale również od rozmieszczenia węzłów interpolacji.

1.3.8. Uwagi końcowe

Należy pamiętać, że podnoszenie stopnia wielomianu interpolacyjnego nie zawsze prowadzi do lepszego przybliżenia funkcji interpolowanej. W niektórych przypadkach (np. dla funkcji $y=|x|$, lub $y=1/(1+x^2)$) wielomiany wysokiego rzędu przybliżają funkcje znacznie gorzej niż wielomiany niskiego rzędu. Wynika to ze złego uwarunkowania zadania interpolacji dla wielomianów wysokiego stopnia, rozpiętych na równoodległych węzłach. Właściwość ta nosi nazwę zjawiska Rungego.

Oszacowanie błędu interpolacji jest praktycznie dość uciążliwe ze względu na konieczność oszacowania wartości $f^{(n+1)}(\xi)$. Dla równoodległych węzłów, jeśli różnice odpowiednich rzędów nie zmieniają wartości w ramach przyjętej dokładności, możemy zastąpić obliczanie pochodnej obliczeniem odpowiedniej różnicy. Umożliwia to oszacowanie błędu również w przypadku, gdy funkcja interpolowana dana jest w postaci tablicy wartości.

Wzory Newtona mają dodatkową zaletę w postaci korzystniejszej złożoności obliczeniowej algorytmu. Dla wielomianu Lagrange'a do obliczenia pojedynczej wartości należy wykonać $\frac{3}{2}n^2+\frac{13}{2}n+3$ działań (w tym n^2+4n+2 mnożeń i dzieleń), natomiast dla wzoru Newtona $\frac{3}{2}+\frac{9}{2}n$ (w tym $\frac{1}{2}n^2+\frac{3}{2}n$ mnożeń i dzieleń), przy czym n oznacza stopień wielomianu interpolacyjnego.

1.4. Różniczkowanie numeryczne

1.4.1. Sformułowanie problemu

Wzory różniczkowania numerycznego znajdują zastosowanie, gdy trzeba wyznaczyć pochodne odpowiedniego rzędu funkcji $f(x)$, która określona jest tablicą lub ma skomplikowaną postać analityczną. Podstawowy sposób konstrukcji wzorów różniczkowania numerycznego polega na różniczkowaniu wzorów interpolacyjnych. Zastępujemy więc daną funkcję $f(x)$ wielomianem interpolacyjnym $W_n(x)$, w interesującym nas przedziale $[a,b]$, a następnie przyjmujemy

$$f^{(k)}(x)=W_n^{(k)}(x) \quad \text{dla} \quad a\le x\le b, \tag{4.1}$$

gdzie k jest rzędem pochodnej. Jeżeli błąd interpolacji $R_n(x)$ określony jest wzorem

$$R_n = f(x) - W_n(x), \tag{4.2}$$

to błąd pochodnej $r_n^{(k)}(x)$ wynosi

$$r_n^{(k)}(x) = f^{(k)}(x) - W_n^{(k)}(x). \tag{4.3}$$

Należy podkreślić, że różniczkowanie numeryczne jest na ogół operacją mniej dokładną niż interpolacja. W szczególności bowiem błąd różniczkowania w węzłach interpolacji może być różny od zera, chociaż błąd interpolacji jest równy zero. Z każdego wielomianu interpolacyjnego można wyprowadzić wzór różniczkowania numerycznego. Ogólnie mówiąc wzory te klasyfikuje się według wykorzystywanych różnic : zwykłych, wstecznych, centralnych.

1.4.2. Wzór Taylora

Rozwinięcie funkcji analitycznej $f(x)$ w otoczeniu punktu x w szereg Taylora można wyrazić w postaci

$$f(x+h) = f(x) + \frac{h}{1!}f^{(1)}(x) + \frac{h^2}{2!}f^{(2)}(x) + \cdots = \sum_{k=0}^{\infty}\frac{h^k}{k!}f^{(k)}(x). \tag{4.4}$$

Zdefiniujemy operator różniczkowania:

$$\begin{aligned} Df(x) &\stackrel{\mathrm{df}}{=} f^{(1)}(x), \\ D^k f(x) &= D(D^{k-1}f(x)) = f^{(k)}(x). \end{aligned} \tag{4.5}$$

Zależność (4.4) można teraz zapisać jako

$$f(x+h) = (1 + \frac{hD}{1!} + \frac{h^2D^2}{2!} + \cdots)f(x) = e^{hD}f(x). \tag{4.6}$$

Z definicji operatora różnicy zwykłej Δ, dla którego

$$\Delta f(x) = f(x+h) - f(x) \tag{4.7}$$

obliczamy

$$f(x+h) = (1+\Delta)f(x). \tag{4.8}$$

Z porównania zależności (4.6) oraz (4.8) uzyskujemy wzór na równość operatorów

$$(1+\Delta) = e^{hD}. \tag{4.9}$$

Logarytmując obustronnie (4.9), otrzymamy

$$hD = ln(1+\Delta) \tag{4.10}$$

oraz

$$D = \frac{1}{h}ln(1+\Delta). \tag{4.11}$$

powered by Misiol

Podnosząc obustronnie (4.11) do potęgi k-tej, uzyskamy

$$D^k = \frac{1}{h^k}[ln(1+\Delta)]^k. \tag{4.12}$$

Rozwinięcie $ln(1+\Delta) = \Delta - \frac{\Delta^2}{2} + \frac{\Delta^3}{3} - \frac{\Delta^4}{4} + \ldots$ w szereg potęgowy prowadzi do wzoru

$$D^k = \frac{1}{h^k}(\Delta - \frac{\Delta^2}{2} + \frac{\Delta^3}{3} - \frac{\Delta^4}{4} + \cdots)^k. \tag{4.13}$$

Z (4.13) można wyprowadzić wzory na dowolne pochodne funkcji $f(x)$ wyrażone za pomocą różnic zwykłych. Przykładowo:

$k = 1$

$$f^{(1)}(x) = \frac{1}{h}\left[\Delta f(x) - \frac{1}{2}\Delta^2 f(x) + \frac{1}{3}\Delta^3 f(x) - \frac{1}{4}\Delta^4 f(x) + \cdots\right], \tag{4.14}$$

$k = 2$

$$f^{(2)}(x) = \frac{1}{h^2}\left[\Delta^2 f(x) - \Delta^3 f(x) + \frac{11}{12}\Delta^4 f(x) - \frac{10}{12}\Delta^5 f(x) + \cdots\right], \tag{4.15}$$

$k = 3$

$$f^{(3)}(x) = \frac{1}{h^3}\left[\Delta^3 f(x) - \frac{3}{2}\Delta^4 f(x) + \frac{7}{4}\Delta^5 f(x) - \frac{45}{24}\Delta^6 f(x) + \cdots\right]. \tag{4.16}$$

W zależnościach tych wykorzystuje się wartości wzięte z tablicy różnic zwykłych (tabl.4.1).

Wzory (4.14) - (4.16) wyrazimy teraz za pomocą różnic wstecznych. Wychodząc z definicji operatora różnicy wstecznej

$$\nabla f(x) = f(x) - f(x-h) \tag{4.17}$$

oraz ze wzoru (4.7) dochodzimy do zależności

$$(1+\Delta) = (1-\nabla)^{-1}. \tag{4.18}$$

Wstawiając powyższe do (4.12), otrzymujemy

$$D^k = \frac{1}{h^k}[ln(1-\nabla)^{-1}]^k = \frac{(-1)^k}{h^k}[ln(1-\nabla)]^k. \tag{4.19}$$

Rozwinięcie $ln(1-\nabla) = \nabla + \frac{\nabla^2}{2} + \frac{\nabla^3}{3} + \frac{\nabla^4}{4} + \cdots$ w szereg potęgowy daje ostatecznie

$$D^k = \frac{1}{h^k}(\nabla + \frac{\nabla^2}{2} + \frac{\nabla^3}{3} + \frac{\nabla^4}{4} + \cdots)^k. \tag{4.20}$$

Z (4.20) można wyprowadzić wzory na dowolne pochodne funkcji $f(x)$ wyrażone za pomocą różnic wstecznych. Przykładowo:

$k = 1$

$$f^{(1)}(x) = \frac{1}{h}\left[\nabla f(x) + \frac{1}{2}\nabla^2 f(x) + \frac{1}{3}\nabla^3 f(x) + \cdots\right], \tag{4.21}$$

$k = 2$

$$f^{(2)}(x) = \frac{1}{h^2}\left[\nabla^2 f(x) + \nabla^3 f(x) + \frac{11}{12}\nabla^4 f(x) + \cdots\right], \tag{4.22}$$

$k = 3$

$$f^{(3)}(x) = \frac{1}{h^3}\left[\nabla^3 f(x) + \frac{3}{2}\nabla^4 f(x) + \frac{7}{4}\nabla^5 f(x) + \frac{45}{24}\nabla^6 f(x) + \cdots\right]. \tag{4.23}$$

W zależnościach tych wykorzystuje się wartości wzięte z tablicy różnic wstecznych (tabl. 4.2).

Błąd różniczkowania numerycznego wyprowadzimy z reszty szeregu Taylora. Wykorzystując zapis szeregu w postaci (4.4), resztę wyrażamy wzorem

$$R_n(h) = \frac{f^{(n+1)}(x+\theta h)}{(n+1)!}h^{n+1} \qquad 0 < \theta < 1, \tag{4.24}$$

który po zróżniczkowaniu przyjmuje postać

$$\frac{dR_n(h)}{dh} = f^{(n+1)}(x+\theta h)\frac{h^n}{n!}. \tag{4.25}$$

Dla małych wartości h i przy założeniu, że dla funkcji $f(x)$ różnice $\Delta^{n+1}f(x)$ są prawie stałe, można zapisać

$$f^{(n+1)}(x+\theta h) \approx \frac{\Delta^{n+1}f(x)}{h^{n+1}}. \tag{4.26}$$

Wstawiając (4.26) do (4.25), otrzymamy

$$R_n^{(1)}(h) \approx \frac{\Delta^{n+1}f(x)}{n!\,h}. \tag{4.27}$$

Tablica różnic zwykłych — Tablica 4.1

x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$	$\Delta^4 f(x)$
x_0	$f(x_0)$				
		$\Delta f(x_0)$			
x_0+h	$f(x_0+h)$		$\Delta^2 f(x_0)$		
		$\Delta f(x_0+h)$		$\Delta^3 f(x_0)$	
x_0+2h	$f(x_0+2h)$		$\Delta^2 f(x_0+h)$		$\Delta^4 f(x_0)$
		$\Delta f(x_0+2h)$		$\Delta^3 f(x_0+h)$	
x_0+3h	$f(x_0+3h)$		$\Delta^2 f(x_0+2h)$		
		$\Delta f(x_0+3h)$			
x_0+4h	$f(x_0+4h)$				

powered by Misiol

Tablica różnic wstecznych — Tablica 4.2

x	$f(x)$	$\nabla f(x)$	$\nabla^2 f(x)$	$\nabla^3 f(x)$	$\nabla^4 f(x)$
x_0-4h	$f(x_0-4h)$				
		$\nabla f(x_0-3h)$			
x_0-3h	$f(x_0-3h)$		$\nabla^2 f(x_0-2h)$		
		$\nabla f(x_0-2h)$		$\nabla^3 f(x_0-h)$	
x_0-2h	$f(x_0-2h)$		$\nabla^2 f(x_0-h)$		$\nabla^4 f(x_0)$
		$\nabla f(x_0-h)$		$\nabla^3 f(x_0)$	
x_0-h	$f(x_0-h)$		$\nabla^2 f(x_0)$		
		$\nabla f(x_0)$			
x_0	$f(x_0)$				

Przykład 4.1

Dla funkcji $f(x) = ln(x)$ obliczyć wartość dokładną i przybliżoną pierwszej pochodnej w punkcie $x_0 = 1$; przy czym x zmienia się od $x = 0.6$ do $x = 1$, z krokiem $h = 0.1$.

Rozwiązanie

Tworzymy tablicę różnic wstecznych.

x	$f(x)$	$\nabla f(x)$	$\nabla^2 f(x)$	$\nabla^3 f(x)$	$\nabla^4 f(x)$
0.6	−0.51083				
		0.15415			
0.7	−0.35667		−0.02062		
		0.13353		0.00487	
0.8	−0.22314		−0.01575		−0.00154
		0.11778		0.00333	
0.9	−0.10536		−0.01242		
		0.10536			
1	0				

Według (4.21) obliczamy

$$f^{(1)}(x) = \frac{1}{0.1}\left[0.10536 + \frac{1}{2}(-0.01242) + \frac{1}{3}0.00333. + \frac{1}{4}(-0.00154)\right] = 0.9988.$$

Dokładna wartość pochodnej $ln(x)^{(1)}|_{x=x_0} = 1$. Błąd bezwzględny wyznaczenia pochodnej wynosi więc

$$r^{(1)} = |1 - 0.9988| = 0.0012.$$

1.4.3. Wzór Stirlinga

Wyprowadzone w poprzednim punkcie wzory różniczkowania numerycznego funkcji $f(x)$ w punkcie $x = x_0$ mają tę wadę, że wykorzystuje się w nich jedynie wartości funkcji

$f(x)$ dla argumentów leżących z jednej strony x_0. Wady tej nie posiadają wzory wykorzystujące wartości funkcji $f(x)$ po prawej i lewej stronie punktu $x = x_0$. Są to wzory symetryczne, oparte na różnicach centralnych. Jednym z nich jest wzór Stirlinga.

Niech będzie dany zbiór punktów $x_{-n}, \cdots, x_{-1}, x_0, \cdots, x_n$, rozmieszczonych w jednakowych odstępach $x_{i+1} - x_i = h$. Rozpatrzmy funkcję $f(x)$ i jej wartości $f(x_i)$, $i = 0, \pm 1, \ldots, \pm n$. Określimy zmienną znormalizowaną $t = (x - x_0)/h$; $t \in [0, n]$.

Wielomian interpolacyjny Stirlinga ma następującą postać

$$\begin{aligned} W_n(x) &= f(x_0) + t\frac{\delta f(x_0 - \frac{1}{2}h) + \delta f(x_0 + \frac{1}{2}h)}{2} + \frac{t^2}{2}\delta^2 f(x_0) + \frac{t(t^2-1)}{3!} + \\ &+ \frac{\delta^3 f(x_0 - \frac{1}{2}h) + \delta^3 f(x_0 + \frac{1}{2}h)}{2} + \frac{t^2(t^2-1)}{4!}\delta^4 f(x_0) + \\ &+ \frac{t(t^2-1)(t^2-2^2)}{5!}\frac{\delta^5 f(x_0 - \frac{1}{2}h) + \delta^5 f(x_0 + \frac{1}{2}h)}{2} + \\ &+ \frac{t^2(t^2-1)(t^2-2^2)}{6!}\delta^6 f(x_0) + \cdots. \end{aligned} \tag{4.28}$$

W celu znalezienia pochodnej funkcji $f(x)$ zastępujemy ją najpierw wielomianem interpolacyjnym (4.28). Przyjmujemy więc

$$f(x) \cong W_n(x). \tag{4.29}$$

Wstawiając następnie (4.28) do (4.29) i różniczkując to ostatnie obustronnie (uwzględniamy, że $dt/dx = 1/h$), otrzymamy wzór Stirlinga różniczkowania numerycznego

$$\begin{aligned} f^{(1)}(x) &= \frac{1}{h}[\frac{\delta f(x - \frac{1}{2}h) + \delta f(x + \frac{1}{2}h)}{2} + t\delta^2 f(x_0) + \frac{3t^2-1}{6} \cdot \\ &\cdot (\frac{\delta^3 f(x - \frac{1}{2}h) + \delta^3 f(x + \frac{1}{2}h)}{2} + \frac{2t^3 - t}{12}\delta^4 f(x_0) + \\ &+ \frac{5t^4 - 15t^2 + 4}{120}(\frac{\delta^5 f(x - \frac{1}{2}h) + \delta^5 f(x + \frac{1}{2}h)}{2}) + \\ &+ \frac{3t^5 - 10t^3 + 4t}{360}\delta^6 f(x_0) + \cdots]. \end{aligned} \tag{4.30}$$

Zróżniczkowanie wzoru (4.30) i podstawienie $dt/dx = 1/h$ prowadzi do wzoru na drugą pochodną

$$\begin{aligned} f^{(2)}(x) &= \frac{1}{h^2}[\delta^2 f(x_0) + t(\frac{\delta^3 f(x - \frac{1}{2}h) + \delta^3 f(x + \frac{1}{2}h)}{2}) + \\ &+ \frac{6t^2-1}{12}\delta^4 f(x_0) + \frac{2t^3 - 3t}{12}(\frac{\delta^5 f(x - \frac{1}{2}h) + \delta^5 f(x + \frac{1}{2}h)}{2}) + \\ &+ \frac{15t^4 - 30t^2 + 4}{360}\delta^6 f(x_0) + \cdots]. \end{aligned} \tag{4.31}$$

Wzory (4.30) i (4.31) upraszczają się, jeżeli pierwszą i drugą pochodną obliczamy w punkcie $x = x_0$ (wtedy $t = 0$)

$$\begin{aligned} f^{(1)}(x) &= \frac{1}{h}[(\frac{\delta f(x - \frac{1}{2}h) + \delta f(x + \frac{1}{2}h)}{2}) - \end{aligned}$$

powered by Misiol

$$\begin{aligned} &- \frac{1}{6}(\frac{\delta^3 f(x - \frac{1}{2}h) + \delta^3 f(x + \frac{1}{2}h)}{2}) + \\ &+ \frac{1}{30}(\frac{\delta^5 f(x - \frac{1}{2}h) + \delta^5 f(x + \frac{1}{2}h)}{2}) + \cdots] \end{aligned} \tag{4.32}$$

oraz

$$f^{(2)}(x) = \frac{1}{h^2}[\delta^2 f(x_0) + \frac{1}{12}\delta^4 f(x_0) + \frac{1}{90}\delta^6 f(x_0) + \cdots]. \tag{4.33}$$

We wzorze Stirlinga różniczkowania numerycznego wykorzystuje się tablicę różnic centralnych (tabl.4.3).

Jeżeli maksymalny rząd różnicy w tablicy różnic centralnych wynosi nh oraz $x \in [x_0 - nh; x_0 + nh]$, to błąd $R_n(x)$ interpolacji wielomianu Stirlinga wynosi

$$R_n(x) = \frac{h^{2n+1} f^{(2n+1)}(\xi)}{(2n+1)!} t(t^2-1)(t^2-2^2)\cdots(t^2-n^2), \tag{4.34}$$

gdzie:

$$t = (x - x_0)/h,$$

$$\xi \in [x_0 - nh; x_0 + nh].$$

Tablica różnic centralnych — Tablica 4.3

x	$f(x)$	$\delta f(x)$	$\delta^2 f(x)$	$\delta^3 f(x)$	$\delta^4 f(x)$
$x_0 - 2h$	$f(x_0 - 2h)$				
		$\delta f(x_0 - \frac{3}{2}h)$			
$x_0 - h$	$f(x_0 - h)$		$\delta^2 f(x_0 - h)$		
		$\delta f(x_0 - \frac{1}{2}h)$		$\delta^3 f(x_0 - \frac{1}{2}h)$	
x_0	$f(x_0)$		$\delta^2 f(x_0)$		$\delta^4 f(x_0)$
		$\delta f(x_0 + \frac{1}{2}h)$		$\delta^3 f(x_0 + \frac{1}{2}h)$	
$x_0 + h$	$f(x_0 + h)$		$\delta^2 f(x_0 + h)$		
		$\delta f(x_0 + \frac{3}{2}h)$			
$x_0 + 2h$	$f(x_0 + 2h)$				

W celu obliczenia błędu pierwszej pochodnej różniczkujemy obustronnie (4.34) i po uwzględnieniu, że $dt/dx = 1/h$, otrzymujemy

$$\begin{aligned} R_n^{(1)}(x) &= \frac{h^{2n}}{(2n+1)!}[f^{(2n+1)}(\xi)\frac{d}{dt}(t(t^2-1)(t^2-2^2)\cdots(t^2-n^2)) + \\ &+ t(t^2-1)(t^2-2^2)\cdots(t^2-n^2)\frac{d}{dt}f^{(2n+1)}(\xi)]. \end{aligned} \tag{4.35}$$

Wzór (4.35) upraszcza się, jeżeli błąd liczyć w punkcie $x = x_0$ (tzn. $t = 0$). Wtedy po uwzględnieniu, że

$$\frac{d}{dt}[t(t^2-1)(t^2-2^2)\cdots(t^2-n^2)]\Big|_{t=0} = (-1)^n (n!)^2 \tag{4.36}$$

dochodzimy do następującej postaci błędu

$$R_n^{(1)}(x_0) = \frac{h^{2n}}{2n+1}(-1)^n(n!)^2 f^{(2n+1)}(\xi). \tag{4.37}$$

Zakładając, że dla małych h zachodzi

$$f^{(2n+1)}(\xi) \cong \frac{\delta^{2n+1} f(x_0 - \frac{1}{2}h) + \delta^{2n+1} f(x_0 + \frac{1}{2}h)}{2} \tag{4.38}$$

otrzymujemy ostatecznie

$$R_n^{(1)}(x_0) \cong \frac{(-1)^n (n!)^2}{h \cdot (2n+1)!} \cdot [\frac{\delta^{2n+1} f(x_0 - \frac{1}{2}h) + \delta^{2n+1} f(x_0 + \frac{1}{2}h)}{2}]. \tag{4.39}$$

Przykład 4.2

Dla funkcji $f(x) = ln(x)$ obliczyć wartość dokładną i przybliżoną pierwszej pochodnej w punkcie $x_0 = 1$, przy czym x zmienia się od $x = 0.8$ do $x = 1.2$ z krokiem $h = 0.1$.

Rozwiązanie

Tworzymy tablicę różnic centralnych.

x	$f(x)$	$\delta f(x)$	$\delta^2 f(x)$	$\delta^3 f(x)$	$\delta^4 f(x)$
0.8	−0.22314				
		0.11778			
0.9	−0.10536		−0.01242		
		0.10536		0.00237	
1	0		−0.01005		0.00062
		0.095310		0.00175	
1.1	0.09531		−0.00830		
		0.08701			
1.2	0.18232				

Według (4.32) obliczamy

$$f^{(1)}(x_0) = \frac{1}{0.1}[0.10033 - \frac{1}{6}0.00206] = 0.99992.$$

Ponieważ dokładna wartość pochodnej $ln(x)^{(1)}|_{x=1} = 1$, błąd obliczeń wynosi

$$R^{(1)}(x_0) = |1 - 0.99992| = 0.00008.$$

W porównaniu z przykładem 4.1 jest to wynik znacznie dokładniejszy.

powered by Misiol

1.4.4. Podsumowanie

Przedstawione wzory różniczkowania numerycznego różnią się zastosowaniem różnic zwykłych, wstecznych, centralnych. Te ostatnie wzory są najdokładniejsze, wykorzystują bowiem informacje o przebiegu funkcji po obu stronach punktu, w którym obliczamy pochodną. Obliczanie pochodnej na końcach przedziału zmienności argumentu funkcji zmusza do wykorzystania wzorów opartych na różnicach zwykłych lub wstecznych.

Może się wydawać, że obliczanie wartości pochodnych nie jest trudniejsze od obliczania wartości wielomianu interpolacyjnego. Dokładniejsza analiza wykazuje jednak, że błędy zaokrągleń, często nieistotne we wzorach interpolacyjnych, mogą mieć decydujący wpływ na różniczkowanie numeryczne. Ze wzorów (4.27) i (4.39) prawdziwych dla małych wartości h wynika, że błąd jest wtedy odwrotnie proporcjonalny do h. Błąd rośnie więc, gdy h maleje. Zwiększenie wartości h powoduje jednak zwiększenie błędu metody. Narzuca się zatem zadanie znalezienia optymalnej wartości h we wzorach różniczkowania numerycznego z węzłami interpolacji równoodległymi.

1.5. Całkowanie numeryczne

1.5.1. Wprowadzenie

W ogólnym przypadku, gdy funkcja $f(x)$ jest ciągła w przedziale $[a, b]$ i znana jest funkcja pierwotna $F(x)$, tj. $F(x)' = f(x)$, to wówczas słuszna jest zależność

$$\int_a^b f(x)dx = F(b) - F(a). \tag{5.1}$$

Jednak w wielu przypadkach wyznaczanie funkcji pierwotnej $F(x)$, a zatem obliczanie całki (5.1) jest bardzo trudne lub praktycznie niemożliwe. Taki problem powstaje wówczas, gdy nie można wyrazić funkcji pierwotnej za pomocą funkcji elementarnych lub wyrażenie takie jest bardzo skomplikowane. Ponadto, gdy funkcja podcałkowa jest dana tabelarycznie, pojęcie funkcji pierwotnej traci sens. Do obliczania całek oznaczonych stosuje się wówczas metody przybliżone, a w szczególności metody numeryczne. Metody te sprowadzają się do obliczania wartości całki jako kombinacji liniowej wartości funkcji podcałkowej w wybranych punktach. Tak więc zastępuje się daną funkcję $f(x)$ w rozpatrywanym przedziale domkniętym $[a, b]$ funkcją interpolującą lub aproksymującą $\varphi(x)$ o możliwie prostej postaci i przyjmuje się przybliżoną równość

$$\int_a^b f(x)dx \approx \int_a^b \varphi(x)dx. \tag{5.2}$$

Jeśli przyjąć, że $\varphi(x)$ jest wielomianem interpolacyjnym $W_n(x)$, otrzymuje się wzór na wartość całki, zwany często kwadraturą

$$\int_a^b f(x)dx = \sum_{i=0}^{i=n} H_i f(x_i) + R_n(f), \tag{5.3}$$

gdzie: x_i-węzły interpolacji, $x_i \in [a, b]$, H_i - stałe, $R_n(f)$ - błąd całkowania dla funkcji podcałkowej f.

Zadanie numerycznego całkowania polega na dobraniu stałych H_i oraz węzłów interpolacji x_i tak, aby przybliżony wzór całkowania dawał odpowiednią dokładność. Istnieje wiele różnych sposobów określania wartości stałych H_i oraz węzłów x_i. W dalszej części rozdziału omówione zostaną kwadratury Newtona-Cotesa i Czebyszewa.

1.5.2. Metoda Newtona-Cotesa

W metodzie Newtona-Cotesa w celu określenia wartości x_i oraz H_i zakłada się, że węzły interpolacji x_i są od siebie równoodległe o krok $h = (b-a)/n$. Tak więc w celu obliczenia całki postaci $\int_a^b f(x)dx$ dzielimy przedział $[a,b]$ na n równych części, gdzie:

$$x_0 = a, \quad x_i = x_0 + i \cdot h \quad (i = 1, 2, \ldots n-1), \quad x_n = b. \tag{5.4}$$

Przyjmując z kolei jako funkcję interpolującą wielomian Newtona $W_n(x)$, można wyznaczyć wartości współczynników H_i

$$H_i = \frac{(b-a)}{n} \frac{(-1)^{n-1}}{i!(n-1)!} \int_0^n \frac{t^{[n+1]}}{t-i} dt, \quad i = 0, 1, \ldots, n, \tag{5.5}$$

przy czym:

$$t = \frac{x - x_0}{h}, \quad dt = \frac{dx}{h} \tag{5.6}$$

oraz

$$t^{[n+1]} = t(t-1)\cdots(t-n), \tag{5.7}$$

gdzie: t – zmienna znormalizowana, $t \in [0, n]$.

Wartość całki wyznacza się ze wzoru noszącego nazwę wzoru Newtona-Cotesa

$$\int_a^b f(x)dx = (b-a)\sum_{i=1}^{n} C_{in} f(x_i), \tag{5.8}$$

przy czym

$$C_{in} = \frac{H_i}{b-a}, \tag{5.9}$$

gdzie: C_{in}, $i = 0, 1, \ldots, n$ są współczynnikami Newtona-Cotesa, zależnymi jedynie od indeksu i, natomiast niezależnymi od funkcji podcałkowej $f(x)$.

Wzór trapezów

Przyjmując $n = 1$ we wzorze (5.5) i uwzględniając (5.9), mamy:

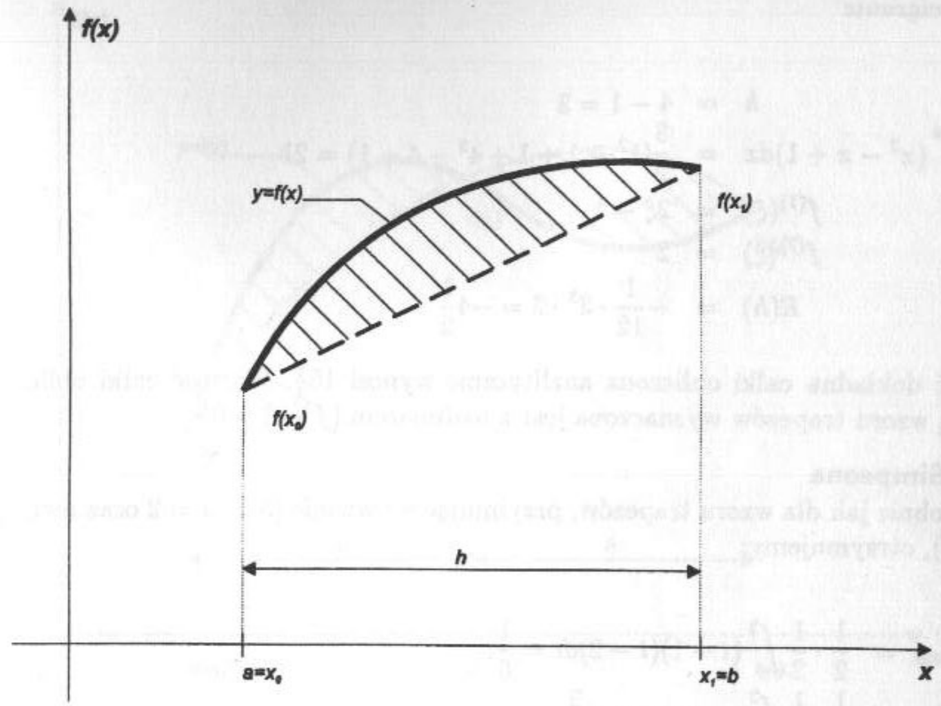
$$C_{10} = -\int_0^1 \frac{t(t-1)}{t} dt = \frac{1}{2},$$

$$C_{11} = -\int_0^1 t dt = \frac{1}{2}.$$

Stąd

$$\int_{x_0}^{x_1} f(x)dx = (b-a)[\frac{1}{2}f(x_0) + \frac{1}{2}f(x_1)] = \frac{h}{2}[f(x_0) + f(x_1)]. \tag{5.10}$$

powered by Misiol



Rys. 1.5. Geometryczna interpretacja wzoru trapezów

Otrzymaliśmy tzw. wzór trapezów na przybliżone obliczanie całki oznaczonej. Geometrycznie wzór ten otrzymuje się przez zastąpienie krzywej podcałkowej odcinkiem prostej przechodzącej przez dwa punkty wyznaczone przez $f(x_0)$ oraz $f(x_1)$ (rys. 1.5).

Błąd całkowania wzoru (5.10) wynosi

$$R = \int_{x_0}^{x_1} f(x)dx - \frac{1}{2}h[f(x_0) + f(x_1)]. \tag{5.11}$$

Rozpatrując R jako funkcję h przy założeniu, że $f(x) \in C^{(2)}[a,b]$, można zapisać

$$R(h) = \int_{x_0}^{x_0+h} f(x)dx - \frac{1}{2}h[f(x_0) + f(x_0+h)]. \tag{5.12}$$

Po przekształceniach otrzymamy

$$R(h) = -\frac{1}{12}h^3 f^{(2)}(\xi), \tag{5.13}$$

gdzie: $\xi \in (x_0, x_1)$.

Z otrzymanych zależności wynika, że dla $f^{(2)}(x) > 0$ wzór (5.10) daje wartość całki z nadmiarem, a dla $f^{(2)}(x) < 0$ z niedomiarem.

Przykład

Korzystając ze wzoru trapezów, obliczyć całkę z funkcji $x^2 - x + 1$ w przedziale $[1, 4]$.

Rozwiązanie

$$
\begin{aligned}
h &= 4-1=3\\
\int_1^4 (x^2-x+1)dx &= \frac{3}{2}(1^2-1+1+4^2-4+1)=21\\
f^{(1)}(\xi) &= 2\xi-1\\
f^{(2)}(\xi) &= 2\\
R(h) &= -\frac{1}{12}\cdot 3^3\cdot 2=-4\frac{1}{2}
\end{aligned}
$$

Wartość dokładna całki obliczona analitycznie wynosi $16\frac{1}{2}$. Wartość całki obliczona za pomocą wzoru trapezów wyznaczona jest z nadmiarem ($f^2(x)>0$).

Wzór Simpsona

Podobnie jak dla wzoru trapezów, przyjmując we wzorze (5.5) $n=2$ oraz uwzględniając (5.9), otrzymujemy:

$$C_{02} = \frac{1}{2}\cdot\frac{1}{2}\int_0^2 (t-1)(t-2)dt = \frac{1}{6}, \tag{5.14}$$

$$C_{12} = \frac{1}{2}\cdot\frac{1}{1}\int_0^2 t(t-2)dt = \frac{2}{3}, \tag{5.15}$$

$$C_{22} = \frac{1}{2}\cdot\frac{1}{2}\int_0^2 t(t-1)dt = \frac{1}{6}. \tag{5.16}$$

$$\tag{5.17}$$

Przyjmując, że $x_2-x_0=2h$, mamy

$$\int_{x_0}^{x_2} f(x)dx = \frac{1}{3}h[f(x_0)+4f(x_1)+f(x_2)]. \tag{5.18}$$

Otrzymany wzór nosi nazwę wzoru Simpsona. Geometrycznie wzór ten otrzymuje się przez zastąpienie krzywej podcałkowej parabolą wyznaczoną trzema punktami $f(x_0), f(x_1), f(x_2)$ (rys. 1.6).

Błąd wzoru (5.18) wyznaczamy z zależności

$$R = \int_{x_0}^{x_2} f(x)dx - \frac{1}{3}h[f(x_0)+4f(x_1)+f(x_2)]. \tag{5.19}$$

Rozpatrując R jako funkcję $h(h\geq 0)$ przy założeniu, że $f(x)\in C^{(4)}[a,b]$ oraz ustalając punkt środkowy x_1, można zapisać

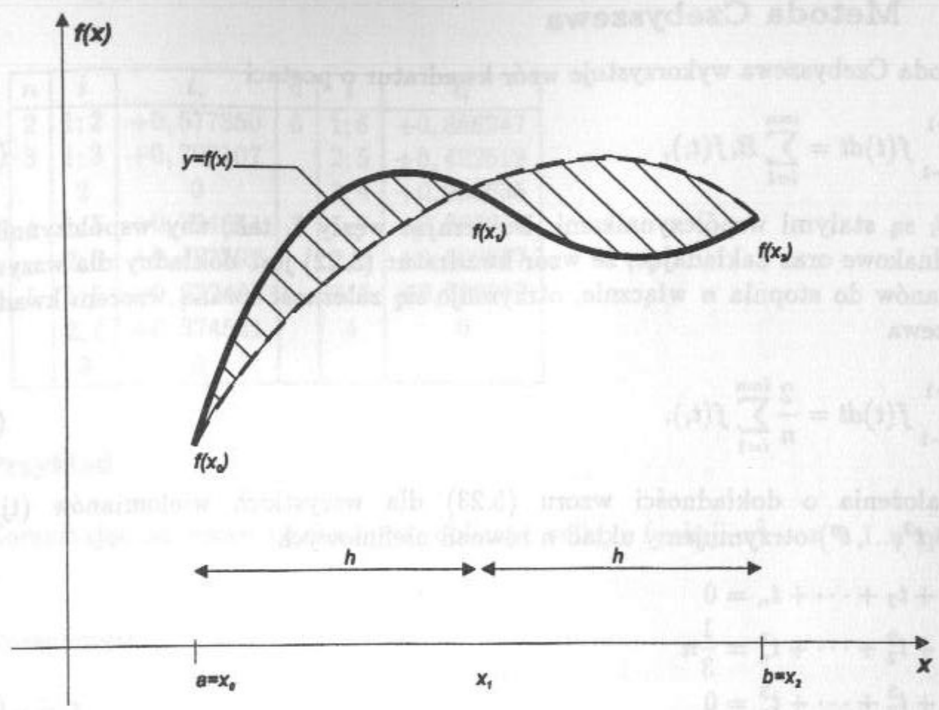
$$R(h) = \int_{x_1-h}^{x_1+h} f(x)dx - \frac{1}{3}h[f(x_0)+4f(x_1)+f(x_2)], \tag{5.20}$$

a po przekształceniach otrzymamy

$$R(h) = -\frac{1}{90}h^5 f^{(4)}(\xi), \tag{5.21}$$

gdzie: $\xi\in(x_0,x_2)$.

powered by Misiol



Rys. 1.6. Geometryczna interpretacja wzoru Simpsona

Można przyjąć, że wzór Simpsona jest dokładny dla wielomianów stopnia drugiego i trzeciego przy stosunkowo małej liczbie węzłów interpolacji.

Przykład

Korzystając ze wzoru Simpsona obliczyć całkę z funkcji x^2-x+1 w przedziale $[1,4]$.

Rozwiązanie

$$
\begin{aligned}
&x_0=1\\
&x_1=2\frac{1}{2}\\
&x_2=4\\
&h=\frac{1}{2}(x_2-x_0)=\frac{3}{2}\\
&\int_2^4 (x^2-x+1)dx=\frac{1}{3}\cdot\frac{3}{2}[1^2-1+1+4(2,5^2-2,5+1)+4^2-4+1]=16\frac{1}{2}\\
&f^{(4)}(\xi)=0\\
&R(h)=-\frac{1}{90}h^5 f^{(4)}(\xi)=0
\end{aligned}
$$

Wzór Simpsona daje w tym przypadku dokładną wartość całki (wielomian drugiego stopnia).

1.5.3. Metoda Czebyszewa

Metoda Czebyszewa wykorzystuje wzór kwadratur o postaci

$$\int_{-1}^{1} f(t)dt = \sum_{i=1}^{i=n} B_i f(t_i), \tag{5.22}$$

gdzie B_i są stałymi współczynnikami. Dobierając węzły t_i tak, aby współczynniki B_i były jednakowe oraz zakładając, że wzór kwadratur (5.22) jest dokładny dla wszystkich wielomianów do stopnia n włącznie, otrzymuje się zależność zwaną wzorem kwadratur Czebyszewa

$$\int_{-1}^{1} f(t)dt = \frac{2}{n}\sum_{i=1}^{i=n} f(t_i). \tag{5.23}$$

Z założenia o dokładności wzoru (5.23) dla wszystkich wielomianów (tj. dla $f(t) = t, t^2, ..., t^n$) otrzymujemy układ n równań nieliniowych:

$$\begin{aligned}
&t_1 + t_2 + \cdots + t_n = 0\\
&t_1^2 + t_2^2 + \cdots + t_n^2 = \frac{1}{3}n\\
&t_1^3 + t_2^3 + \cdots + t_n^3 = 0\\
&\vdots\\
&t_1^i + t_2^i + \cdots + t_n^i = \int_{-1}^{+1} t^i dt\\
&\vdots\\
&t_1^n + t_2^n + \cdots + t_n^n = \frac{n[1-(-1)^{n+1}]}{2(n+1)}
\end{aligned} \tag{5.24}$$

Z tego układu wyznaczamy niewiadome węzły $t_i, i = 1, 2, ..., n$. Wykazano jednak (Bernstein), że dla $n = 8$ oraz $n \geq 10$ układ (5.24) nie ma rozwiązań rzeczywistych, co jest podstawową wadą metody Czebyszewa.

Aby wykorzystać wzór kwadratur Czebyszewa do obliczania całki postaci $\int_a^b f(x)dx$, należy zmienić granice całkowania stosując podstawienie

$$x = \frac{1}{2}(b+a) + \frac{1}{2}(b-a)t. \tag{5.25}$$

Zatem

$$\int_a^b f(x)dx = \frac{(b-a)}{2}\int_{-1}^{+1} f\left[\frac{1}{2}(b+a) + \frac{1}{2}(b-a)t\right] dt. \tag{5.26}$$

Wzór Czebyszewa przyjmuje wówczas postać

$$\int_a^b f(x)dx = \frac{b-a}{n}\sum_{i=1}^{i=n} f(x_i), \tag{5.27}$$

gdzie: $x_i = \frac{1}{2}(b+a) + \frac{1}{2}(b-a)t_i$. Poniżej podano wartości pierwiastków t_i układu (5.24) dla $n = 2, 3, ..., 7$.

powered by Misiol

n	i	t_i	n	i	t_i
2	1; 2	+0,577350	6	1; 6	+0,866247
3	1; 3	+0,707107		2; 5	+0,422519
	2	0		3; 4	+0,266635
4	1; 4	+0,794654	7	1; 7	+0,883862
	2; 3	+0,187592		2; 6	+0,529657
5	1; 5	+0,832498		3; 5	+0,323912
	2; 4	+0,374541		4	0
	3	0			

Przykład

Korzystając ze wzoru Czebyszewa, obliczyć całkę z funkcji $x^2 - x + 1$ w przedziale [1,4].

Rozwiązanie

$n = 2$

$$x_1 = \frac{1}{2}(4+1) + \frac{1}{2}(4-1)\cdot(-0,58) = 1.63$$

$$x_0 = \frac{1}{2}(4+1) + \frac{1}{2}(4-1)\cdot 0,58 = 3.37$$

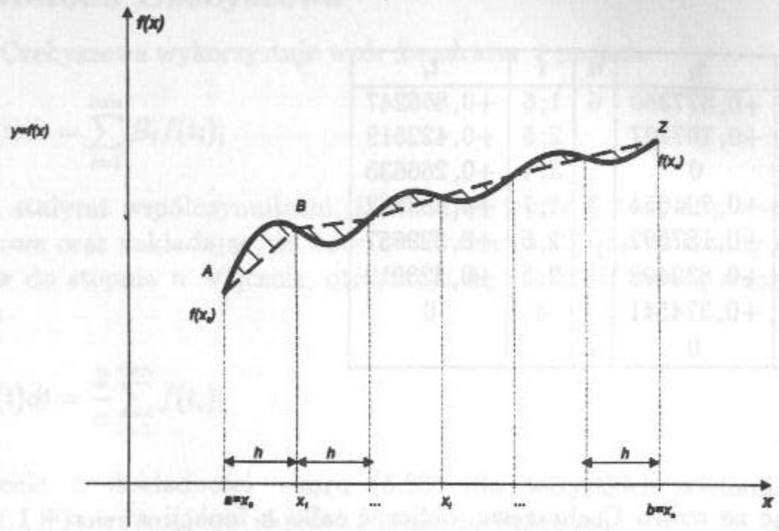
$$\int_1^4 (x^2 - x + 1)dx = \frac{4-1}{2}(1.63^2 - 1.63 + 1 + 3.37^2 - 3.37 + 1) = 16.53$$

1.5.4. Kwadratury złożone

Dokładność kwadratury zależy od liczby węzłów interpolacji. Ze względu na trudność oszacowania błędu dla większych wartości n stosuje się w praktyce kwadratury złożone. W tym przypadku przedział całkowania $[a, b]$ dzieli się na m równych podprzedziałów i w każdym podprzedziale stosuje się ustaloną kwadraturę o małym n, a otrzymane wyniki cząstkowe dodaje się. Najczęściej stosuje się szczególne przypadki kwadratury Newtona-Cotesa: wzór trapezów i wzór Simpsona.

Kwadratura złożona trapezów

W interpretacji geometrycznej kwadratura złożona trapezów polega na zastąpieniu krzywej przedstawiającej funkcję podcałkową $f(x)$ za pomocą linii łamanej (rys. 1.7). W celu obliczenia całki oznaczonej z funkcji $f(x)$ w przedziale $[a, b]$ dzieli się ten przedział na m równych podprzedziałów, każdy o długości $h = (a-b)/m$. W każdym podprzedziale $[x_i, x_{i+1}], x_i = a + i\cdot h,\ \ i = 0, 1, \ldots, m-1$ wyznacza się wartość całki korzystając ze wzoru trapezów (5.10). Zwróćmy uwagę, że wzór ten określa pole trapezu o bokach $[x_i, x_{i+1}]$ i $[F, G]$. W przedziale $[x_i, x_{i+1}]$ funkcja $f(x)$ jest interpolowana prostą. Wartość interesującej



Rys. 1.7. Geometryczna interpretacja kwadratury złożonej trapezów

nas całki jest więc w tym przypadku przybliżona polem S pod krzywą łamaną AB...Z (rys. 1.7)

$$S = h[\frac{f(x_0)+f(x_1)}{2} + \frac{f(x_1)+f(x_2)}{2} + \cdots + \frac{f(x_{+n-1})+f(x_n)}{2}]. \tag{5.28}$$

Stąd

$$\int_a^b f(x)dx \approx \frac{x_n - x_0}{2n}[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1} + f(x_n)], \tag{5.29}$$

gdzie : $x_0 = a$, $x_n = b$. Jeżeli $f(x) \in C^{(2)}[a,b]$, to ogólny błąd metody wyraża się wzorem

$$R(f) = -\frac{1}{12}nh^3 \cdot f^{(2)}(\xi), \quad \xi \in [a,b]. \tag{5.30}$$

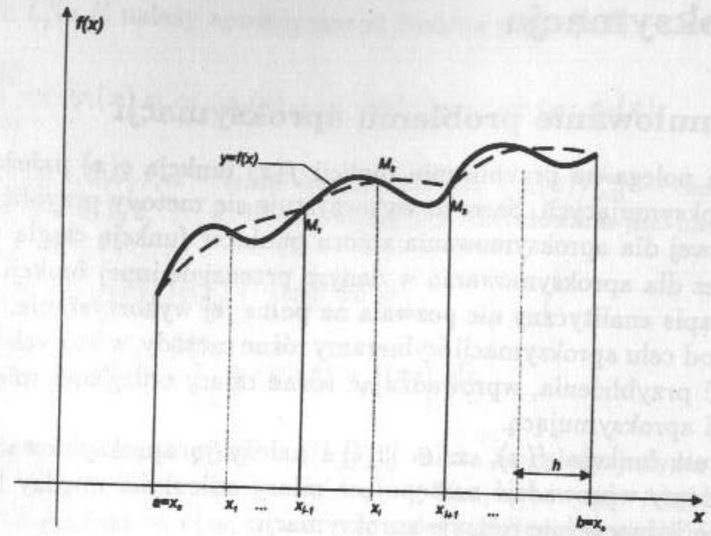
Ponieważ punkt ξ nie jest znany, w praktyce posługujemy się oszacowaniem modułu błędu całkowania

$$|R(f)| \leq \frac{nh^3}{12} \max_{\xi\in[a,b]} |f^{(2)}(\xi)|. \tag{5.31}$$

Kwadratura złożona Simpsona

Zastosowanie wzoru Simpsona można interpretować jako zastąpienie krzywej przedstawiającej funkcję podcałkową $f(x)$ łukami parabol (rys. 1.8). Aby obliczyć całkę z funkcji $f(x)$ w przedziale $[a,b]$, dzieli się ten przedział na $2n$ równych części o długości $h = (b-a)/2n$. W każdym z podwojonych przedziałów: $[x_0, x_2], [x_2, x_4], \cdots, [x_{2n-2}, x_{2n}]$ zastępuje się krzywą $f(x)$ parabolą przechodzącą przez trzy punkty: $M_1[x_{i-1}, f(x_{i-1})], M_2[x_i, f(x_i)], M_3[x_{i+1}, f(x_{i+1})]$, $i = 0, 1, \ldots, 2n$, co jest

powered by Misiol



Rys. 1.8. Geometryczna interpretacja kwadratury złożonej Simpsona

równoważne zastosowaniu w przedziale $[x_{i-1}, x_{i+1}]$ kwadratury Newtona-Cotesa dla $m = 2$. Po przekształceniach otrzymamy dla kwadratury złożonej Simpsona wzór o postaci

$$\begin{aligned}\int_{x_0}^{x_n} f(x)dx &= \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots \\ &+ 2f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_{2n})].\end{aligned} \tag{5.32}$$

Jeżeli $f(x) \in C^{(4)}[a,b]$, to ogólny błąd metody dany jest wzorem

$$R(f) = -\frac{1}{90}nh^5 \cdot f^{(4)}(\xi), \quad \xi \in [a,b]. \tag{5.33}$$

W praktyce, ponieważ punkt pośredni ξ nie jest znany, posługujemy się następującym oszacowaniem modułu błędu całkowania

$$|R(f)| = -\frac{nh^5}{90} \max_{\xi\in[a,b]} |f^{(4)}(\xi)|. \tag{5.34}$$

1.5.5. Podsumowanie

Omówione metody nie są jedynymi metodami obliczania całek oznaczonych. Istnieją jeszcze np.: metoda kwadratur Gaussa, wykorzystująca własności wielomianów Legendre'a, metoda Richardsona, oparta na ekstrapolacji, Eulera-Maclaurina i inne. Większość z tych metod jest opracowana na maszyny cyfrowe i znajduje się w bibliotekach programów ośrodków obliczeniowych.

1.6. Aproksymacja

1.6.1. Sformułowanie problemu aproksymacji

Aproksymacja polega na przybliżeniu funkcji $f(x)$ funkcją $q(x)$ należącą do pewnej klasy funkcji aproksymujących. Szeroko wykorzystuje się metody przybliżania funkcji w technice pomiarowej dla aproksymowania zbioru punktów funkcją ciągłą (aproksymacja punktowa) lub też dla aproksymowania w danym przedziale innej funkcji ciągłej, której skomplikowany zapis analityczny nie pozwala na pełne jej wykorzystanie.

W zależności od celu aproksymacji wybieramy różne metody, w których różnie możemy definiować jakość przybliżenia, wprowadzając różne miary odległości między funkcjami aproksymowaną i aproksymującą.

Jeżeli dana jest funkcja $f(x)$, $x \in [a,b]$ i należy ją aproksymować funkcją $q(x)$, $x \in [a,b]$, to możemy wprowadzić następujące miary odległości między funkcjami $f(x)$ i $q(x)$, określające jednocześnie rodzaje aproksymacji:

- Miara jednostajna (aproksymacja jednostajna)

$$\mu(f,q) = \max_{x\in[a,b]} |f(x) - q(x)|.$$

- Miara średniokwadratowa z wagą $p(x) > 0$ (aproksymacja średniokwadratowa)

$$\delta_p(f,q) = \sqrt{\int_a^b p(x)\cdot[f(x)-q(x)]^2 dx}.$$

- Jeżeli funkcja aproksymowana dana jest wyłącznie w dyskretnych punktach x_i, $i = 0,1,2,..,n$, to można wprowadzić następującą miarę odległości między funkcjami,nazywaną miarą punktową (aproksymacją punktową)

$$S(f,q) = \sqrt{\sum_{i=0}^{i=n}[f(x_i)-q(x_i)]^2}.$$

Zadaniem aproksymacji będzie zatem znalezienie takiej funkcji $q_{opt}(x)$ z pewnej klasy funkcji aproksymujących, która będzie minimalizować jedną z przyjętych miar odległości.

Funkcja $q(x)$ może być zwykłym wielomianem uogólnionym, składającym się z funkcji ortogonalnych, funkcją trygonometryczną itp.

1.6.2. Aproksymacja średniokwadratowa

Aproksymacja wielomianami uogólnionymi

Niech $L_p^2[a,b]$ będzie przestrzenią funkcji całkowalnych z kwadratem w przedziale $[a,b]$ z wagą $p(x) > 0$. Oznacza to, że dla funkcji $f(x) \in L_p^2[a,b]$, istnieje całka

$$\int_a^b p(x)\cdot f^2(x)dx < \infty.$$

powered by Misiol

Funkcję $f(x) \in L_p^2[a,b]$ należy aproksymować funkcją $q(x) : [a,b] \to R$ o postaci

$$q(x) = \sum_{i=0}^{i=n} c_i\cdot\varphi_i(x) = c_0\cdot\varphi_0(x) + c_1\cdot\varphi_1(x) + \cdots + c_n\cdot\varphi_n(x), \tag{6.1}$$

gdzie: c_i, $i = 0,1,2,\ldots,n$ - stałe współczynniki, $\{\varphi_i(x)\}_{i=0}^{i=n}$ - dany układ funkcji, $\varphi_i(x) \in L_p^2[a,b]$, $i = 0,1,2,\ldots,n$. Miara odległości zdefiniowana jest następująco

$$\begin{aligned}\delta_p^2(f,q) &= \int_a^b p(x)\cdot[q(x)-f(x)]^2 dx = \\ &= \int_a^b p(x)\cdot\left[\sum_{i=0}^{i=n} c_i\cdot\varphi_i(x) - f(x)\right]^2 dx.\end{aligned} \tag{6.2}$$

Przy zadanym układzie funkcji $\{\varphi_i(x)\}_{i=0}^{i=n}$ miara ta zależy jedynie od współczynników c_i, $i = 0,1,2,\ldots,n$, to znaczy: $\delta_p(f,q) = \delta(c_0,c_1,\ldots,c_n)$. Aby znaleźć minimum funkcji wypukłej $\delta_p(f,q) = \delta(c_0,c_1,\ldots,c_n)$, należy obliczyć wszystkie pochodne cząstkowe $\frac{\partial\delta}{\partial c_i}$, $(i = 0,1,2,\ldots,n)$, a następnie przyrównać je do zera. Otrzymamy zatem układ $(n+1)$ równań liniowych o $(n+1)$ niewiadomych $c_0,c_1,\ldots,c_n$ o postaci

$$\frac{1}{2}\frac{\partial\delta}{\partial c_j} = \int_a^b p(x)\cdot\left[\sum_{i=0}^{i=n} c_i\cdot\varphi_i(x) - f(x)\right]\cdot\varphi_j(x)dx = 0 \quad (j = 0,1,\ldots,n). \tag{6.3}$$

Po prostych przekształceniach układ równań (6.3) można zapisać następująco

$$\sum_{i=0}^{i=n} c_i\cdot\int_a^b p(x)\cdot\varphi_i(x)\cdot\varphi_j(x)dx = \int_a^b p(x)f(x)\cdot\varphi_j(x)0 \quad (j = 0,1,\ldots,n). \tag{6.4}$$

Jeżeli funkcje $\varphi_0(x),\varphi_1(x),\ldots,\varphi_n(x)$ są liniowo niezależne w przedziale $[a,b]$, to układ (6.4) ma jedno rozwiązanie, które odpowiada najmniejszemu odchyleniu $\delta_p(f,q)$.

Przykład 1

Znaleźć najlepszą aproksymację średniokwadratową z wagą $p(x) = 1$ dla funkcji $f(x) = \sqrt[3]{x}$ w przedziale $[0,1]$. Funkcję aproksymującą przyjąć w postaci wielomianu stopnia pierwszego postaci $q(x) = c_0 + c_1x$.

Rozwiązanie

Mamy tutaj $\varphi_0(x) = 1$ oraz $\varphi_1(x) = x$. Korzystając z (6.4), obliczamy kolejno:

$$\begin{aligned}\int_0^1 p(x)\cdot\varphi_0(x)\cdot\varphi_0(x)dx &= \int_0^1 1dx = 1,\\ \int_0^1 p(x)\cdot\varphi_0(x)\cdot\varphi_1(x)dx &= \int_0^1 xdx = \frac{1}{2},\\ \int_0^1 p(x)\cdot\varphi_1(x)\cdot\varphi_1(x)dx &= \int_0^1 x^2dx = \frac{1}{3},\end{aligned}$$

$$\int_0^1 p(x)\cdot f(x)\cdot\varphi_0(x)dx = \int_0^1 \sqrt[3]{x}dx = \frac{3}{4},$$
$$\int_0^1 p(x)\cdot f(x)\cdot\varphi_1(x)dx = \int_0^1 x\sqrt[3]{x}dx = \frac{3}{7}.$$

Układ równań (6.4) wygląda więc następująco:

$$c_0 + \frac{1}{2}c_1 = \frac{3}{4},$$
$$\frac{1}{2}c_0 + \frac{1}{3}c_1 = \frac{3}{7},$$

stąd $c_0 = \frac{3}{7}; c_1 = \frac{9}{14}$ zatem $q(x) = \frac{3}{7} + \frac{9}{14}x$. W przypadku gdy stopień wielomianu aproksymującego jest duży, powyższa metoda jest uciążliwa i wtedy funkcje $\varphi_i(x)$, $(i = 0, 1, \ldots, n)$ zastępuje się funkcjami ortogonalnymi.

Wykorzystanie wielomianów ortogonalnych

Daną funkcję $f(x) \in L_p^2[a,b]$ należy aproksymować z wagą $p(x)$ funkcją $q(x)$ o postaci

$$q(x) = \sum_{i=0}^{i=n} c_i\varphi_i(x) = c_0\cdot\varphi_0(x) + c_1\cdot\varphi_1(x) + \cdots + c_n\cdot\varphi_n(x), \tag{6.5}$$

gdzie: $\varphi_i(x) \in L_p^2[a,b]$, $i = 0, 1, \ldots, n$ są funkcjami tworzącymi układ ortogonalny w $L_p^2[a,b]$, tzn.

$$\int_a^b p(x)\cdot\varphi_i(x)\cdot\varphi_j(x)dx = \begin{cases} 0 & \text{dla } i \neq j \\ const & \text{dla } i = j. \end{cases} \tag{6.6}$$

Mamy więc do czynienia z układem ortogonalnym funkcji w przedziale $[a,b]$. W przypadku, gdy całka (6.6) dla $i = j$ jest równa jedności, wówczas mówimy o układzie ortonormalnym funkcji. Zatem każdy układ ortonormalny jest układem ortogonalnym. Miara odległości pomiędzy funkcjami jest zdefiniowana wzorem (6.2).

Stosując identyczne rozumowanie jak przy aproksymacji wielomianami uogólnionymi oraz wykorzystując wypukłość funkcji $\delta_p(f,q)$ oraz własności funkcji ortogonalnych, dochodzimy do wzoru na współczynniki c_i

$$c_i = \frac{1}{\lambda_i}\int_a^b p(x)\varphi_i(x)f(x)dx, \qquad i = 0, 1, \ldots, n, \tag{6.7}$$

gdzie

$$\lambda_i = \int_a^b p(x)\varphi_i^2(x)dx \qquad i = 0, 1, \ldots, n. \tag{6.8}$$

Współczynniki c_i, $i = 0, 1, 2, \ldots, n$, noszą nazwę współczynników Fouriera funkcji $f(x)$ względem danego układu ortogonalnego $\left\{\varphi_i(x)\right\}_{i=0}^{i=n}$.

Układ ortogonalny ze współczynnikami Fouriera minimalizuje odchylenie średniokwadratowe (6.2). Przekształcając wzór (6.2) dla układów funkcji ortogonalnych możemy napisać wzór na minimalną wartość kwadratu odchylenia średniokwadratowego

$$\delta_p^2(f,q) = \int_a^b p(x)f^2(x)dx - \sum_{i=0}^{i=n} c_i^2\cdot\lambda_i. \tag{6.9}$$

powered by Misiol

Można udowodnić, że wielomian uogólniony ze współczynnikami Fouriera danej funkcji ma najmniejsze odchylenie kwadratowe od tej funkcji w porównaniu ze wszystkimi innymi wielomianami uogólnionymi tego samego stopnia.

Przykłady funkcji ortogonalnych i ortonormalnych

- układ ortonormalny w przedziale $[-\pi,\pi]$ z wagą $p(x) = 1$

$$\frac{1}{\sqrt{2\pi}}, \quad \frac{1}{\sqrt{\pi}}cos(x), \quad \frac{1}{\sqrt{\pi}}sin(x), \ldots, \frac{1}{\sqrt{\pi}}cos(nx), \quad \frac{1}{\sqrt{\pi}}sin(nx)$$

- układ ortonormalny w przedziale $[0,\pi]$ z wagą $p(x) = 1$

$$\sqrt{\frac{2}{\pi}}sin(nx), \quad n = 1, 2, \ldots$$

- układ ortonormalny w przedziale $[0,\pi]$ z wagą $p(x) = 1$

$$\frac{1}{\sqrt{\pi}}, \quad \sqrt{\frac{2}{\pi}}cos(nx), \quad n = 1, 2, \ldots$$

- wielomiany Czebyszewa, przedział $(-1,1)$, $p(x) = \frac{1}{\sqrt{1-x^2}}$

$$T_0(x) = 1, \ T_1(x) = x, \ T_2(x) = 2x^2, \ T_3(x) = 4x^3 - 3x, \ldots$$

ogólnie: $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$ dla $n = 1, 2, 3, \ldots$

- wielomiany Legendre'a, przedział $[-1,1]$, $p(x) = 1$

$$P_0(x) = 1, \ P_1(x) = x, \ P_2(x) = \frac{1}{2}(3x^2 - 1), \ P_3(x) = \frac{1}{2}(5x^3 - 3x), \ldots$$

ogólnie: $(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$ dla $n = 1, 2, 3, \ldots$

- wielomiany Hermitte'a, przedział $(-\infty,\infty)$, $p(x) = e^{-x^2}$

$$H_0(x) = 1, \ H_1(x) = 2x, \ H_2(x) = 4x^2 - 2 \ H_3(x) = 8x^3 - 12x, \ldots$$

ogólnie: $H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)$ dla $n = 1, 2, 3, \ldots$

- wielomiany Laguerre'a, przedział $[0,\infty)$, $p(x) = e^{-x}$

$$L_0(x) = 1, \ L_1(x) = 1 - x, \ L_2(x) = \frac{1}{2}(2 - 4x + x^2), \ldots \ L_3(x) = 8x^3 - 12x, \ldots$$

ogólnie: $m(n+1)L_{n+1}(x) = (1 + 2n - x)L_n(x) - nL_{n-1}(x)$ dla $n = 1, 2, 3, \ldots$

Przykład 2

Znaleźć najlepszą aproksymację średniokwadratową z wagą $p(x) = 1$ funkcji $f(x) = e^x$ w przedziale $[-1, 1]$. Funkcję aproksymującą przyjąć postaci wielomianów Legendre'a drugiego stopnia.

Rozwiązanie

Mamy tutaj: $p(x) = 1$, $\varphi_0(x) = 1$, $\varphi_1(x) = x$, $\varphi_2(x) = \frac{1}{2}(3x^2 - 1)$. Korzystając z (6.7) i (6.8) obliczamy kolejno:

$$\begin{aligned}
\lambda_0 &= \int_{-1}^{1} \varphi_0^2(x) \cdot dx = \int_{-1}^{1} 1^2 dx = 2, \\
\lambda_1 &= \int_{-1}^{1} \varphi_1^2(x) \cdot dx = \int_{-1}^{1} x^2 dx = \frac{2}{3}, \\
\lambda_2 &= \int_{-1}^{1} \varphi_2^2(x) \cdot dx = \int_{-1}^{1} \frac{1}{2}[3x^2 - 1] dx = \frac{2}{5}, \\
c_0 &= \frac{1}{2}\int_{-1}^{1} e^x dx = 1.175, \\
c_1 &= \frac{3}{2}\int_{-1}^{1} x e^x dx = 1.104, \\
c_2 &= \frac{5}{2}\int_{-1}^{1} \frac{1}{2}(3x^2 - 1) \cdot e^x dx = 0.375.
\end{aligned}$$

Funkcja aproksymująca wygląda więc następująco:

$$\begin{aligned}
q(x) &= 1.175 + 1.104x + 0.375(\frac{1}{2}(3x^2 - 1)), \\
q(x) &= 0.996 + 1.104x + 0.530x^2.
\end{aligned}$$

1.6.3. Aproksymacja jednostajna

W przypadku, gdy od aproksymacji wymaga się, aby w całym przedziale $[a, b]$ odchylenie funkcji aproksymowanej $f(x)$ od wielomianu aproksymującego $q(x)$ było mniejsze od zadanej wielkości, musimy korzystać z aproksymacji jednostajnej. Problem sprowadza się do minimalizacji miary jednostajnej danej wzorem

$$\mu(f, q) = \max_{x \in [a,b]} |f(x) - q(x)|. \tag{6.10}$$

Można podać trzy twierdzenia dotyczące aproksymacji jednostajnej sformułowane przez Weierstrasse'a:

Twierdzenie 1

Jeżeli funkcja $f(x)$ jest ciągła w przedziale $[a, b]$, to dla dowolnie małej liczby dodatniej ε istnieje wielomian $W_m(x)$ dostatecznie wysokiego stopnia $m = m(\varepsilon)$, którego

powered by Misiol

bezwzględne odchylenie od danej funkcji $f(x)$ w przedziale $[a, b]$ jest mniejsze od ε, tj. dla wszystkich punktów $x \in [a, b]$ spełniona jest nierówność

$$|f(x) - Q_m(x)| < \varepsilon. \tag{6.11}$$

Twierdzenie 2

Jeżeli funkcja $f(x)$ jest funkcją ciągłą i okresową o okresie 2π, to dla każdego $\varepsilon > 0$ istnieje wielomian trygonometryczny

$$Q_m(x) = a_0 + \sum_{k=1}^{k=m} (a_k \cdot cos(kx) + b_k \cdot sin(kx)), \qquad m = m(\varepsilon),$$

spełniający dla wszystkich x nierówność

$$|f(x) - Q_m(x)| \leq \varepsilon. \tag{6.12}$$

Niech $p_0(x), p_1(x), \ldots, p_i(x), \ldots, p_m(x)$ będzie liniowo niezależnym układem funkcji ciągłych w przedziale $[a, b]$. Ponadto niech

$$Q_m(x) = \sum_{i=0}^{i=m} a_i p_i(x)$$

będzie poszukiwanym wielomianem aproksymacji jednostajnej o współczynnikach $a_0, a_1, \ldots, a_i, \ldots, a_m$. Warunki, które całkowicie charakteryzują wielomian aproksymacji jednostajnej $Q_m(x)$, są określone w następującym twierdzeniu sformułowanym przez Czebyszewa.

Twierdzenie 3

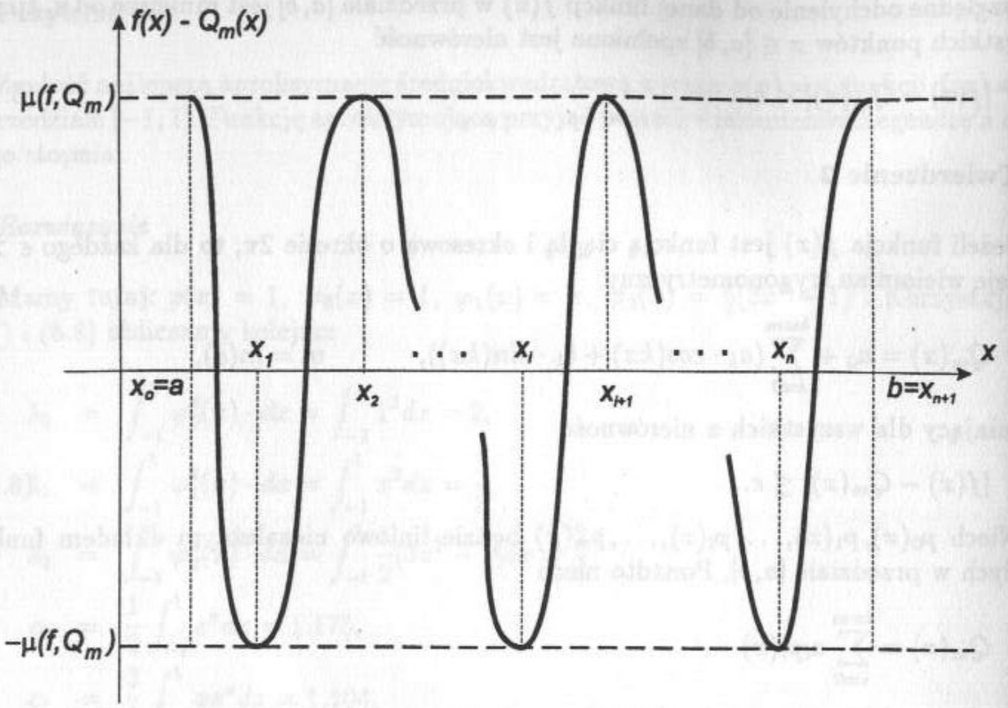
Wielomian $Q_m(x)$ jest wielomianem najlepiej przybliżającym funkcję ciągłą $f(x)$ w sensie metryki jednostajnej $\mu(f, Q_m)$ wtedy i tylko wtedy, gdy różnica $f(x) - Q_m(x)$ przyjmuje w co najmniej $m + 2$ punktach $x_0 < x_1 < \ldots < x_i < \ldots < x_{n+1}$ przedziału $[a, b]$ swoje ekstremalne wartości równe $\pm\mu(f, Q_m)$ z kolejno zmieniającymi się znakami (rys. 1.9).

Warunki tzw. równomiernego falowania sformułowane w twierdzeniu Czebyszewa o alternansie pozwalają w sposób jednoznaczny określić współczynniki $a_0, a_1, \ldots, a_i, \ldots, a_m$ w wielomianie aproksymującym $Q_m(x)$. Praktyczne wyznaczenie wielomianu aproksymującego $Q_m(x)$ jest jednak bardzo uciążliwe, gdyż równania, z których wyznacza się współczynniki $a_0, a_1, \ldots, a_i, \ldots, a_m$, stanowią układ równań nieliniowych.

Przykładem praktycznego zastosowania twierdzenia o alternansie jest twierdzenie 4 rozwiązujące zagadnienie aproksymacji jednostajnej z użyciem wielomianów Czebyszewa.

Metody znajdowania optymalnego wielomianu $Q_m(x)$ minimalizującego (6.11) są dość złożone, można jednak udowodnić, że wielomiany ortogonalne Czebyszewa przybliżają dobrze funkcję zarówno w sensie miary średniokwadratowej, jak i jednostajnej.

Wykorzystując wielomiany Czebyszewa otrzymuje się rozwiązanie zagadnienia aproksymacji jednostajnej dla funkcji $f(x) = x^n$.



Rys. 1.9. Interpretacja twierdzenia 3

Twierdzenie 4

Dla funkcji $f(x) = x^n$ $(n \geq 2)$ optymalnym wielomianem aproksymującym stopnia $(n-2)$ w przedziale $[-1,1]$ jest wielomian

$$W_{n-2}(x) = x^n - \frac{1}{2^{n-1}} T_n(x), \tag{6.13}$$

gdzie $T_n(x)$ jest wielomianem ortogonalnym Czebyszewa stopnia n. Ponadto odległość $\mu(x^n, W_{n-2}) = \frac{1}{2^{n-1}}$ jest przyjmowana w 2^{n-1} punktach przedziału $[-1,1]$.

Twierdzenie 4 rozwiązuje zagadnienie aproksymacji dla bardzo szczególnego przypadku, a mianowicie jedynie dla wybranych funkcji $f(x) = x^n$.

Ponieważ wielomian Czebyszewa ma postać

$$T_n(x) = 2^{n-1}x^n + a_{n-2}x^{n-2} + \cdots,$$

więc

$$W_{N-2}(x) = x^n - \frac{1}{2^{n-1}}(2^{n-1}x^n + a_{n-2}x^{n-2} + \cdots) = -\frac{1}{2^{n-1}}(a_{n-2}x^{n-2} + \cdots)$$

jest wielomianem stopnia $(n-2)$.

powered by Misiol

Przykład

Niech $f(x) = x^3$. Wówczas

$$W_{n-2}(x) = W_1(x) = x^3 - \frac{1}{4}T_3(x) = x^3 - \frac{1}{4}(4x^3 - 3x) = \frac{3}{4}x.$$

Ponadto odległość

$$\mu(x^3, W_1) = \mu(x^3, \frac{3}{4}x) = \max_{-1 \leq x \leq 1} \left| x^3 - \frac{3}{4}x \right| = \frac{1}{4}$$

jest przyjmowana w następujących 4 punktach przedziału $[-1,1]$: $x_1 = -1$, $x_2 = -0.5$, $x_3 = 0.5$, $x_4 = 1$.

1.6.4. Aproksymacja punktowa

Metoda wielomianów ortogonalnych

Metoda wielomianów ortogonalnych pozwala na aproksymację funkcji $f(x)$ zmierzonej doświadczalnie w $n+1$ równo oddalonych punktach. Dany jest zatem zbiór punktów $x_i, y_i = f(x_i)$, $i = 0,1,2,\ldots,n$. Wprowadzając nową zmienną znormalizowaną $t = (x - x_0)/h$, gdzie h jest odległością pomiędzy punktami x_i, przekształcamy punkty x_i do postaci $t_i = 0,1,2,\ldots,n$.

Poszukujemy rozwiązania zadania aproksymacji w postaci

$$Q_m(x) = \sum_{k=0}^{k=m} b_k \cdot P_{k,n}(t) \quad dla \quad n > m, \tag{6.14}$$

gdzie: b_k, $(k = 0,1,2,\ldots,m)$ – stałe współczynniki, $P_{k,n}(t)$, $(k = 0,1,\ldots,n)$ – wielomiany ortogonalne na zbiorze punktów $t_i = 0,1,2,\ldots,n$.

Wielomiany ortogonalne spełniają zależność

$$\sum_{i=0}^{i=n} P_{k,n}(t_i) \cdot P_{j,n}(t_i) = \begin{cases} 0 & \text{dla} \quad k \neq j \\ const & \text{dla} \quad k = j. \end{cases}$$

Optymalny wielomian $Q_m(t)$ powinien minimalizować przyjętą miarę odległości w postaci

$$S(f,Q) = \sum_{i=0}^{i=n} [f(t_i) - Q_m(t_i)]^2. \tag{6.15}$$

Ponieważ miara ta jest funkcją wypukłą względem współczynników b_k, więc można wykazać, że dla optymalnego wielomianu $Q_m(t)$ wzory na współczynniki b_k są następujące

$$b_k = \frac{c_k}{s_k} \quad (k = 0,1,2,\ldots,m), \tag{6.16}$$

gdzie:

$$c_k = \sum_{i=0}^{i=n} y_i \cdot P_{k,n}(t) \quad (k = 0,1,\ldots,m), \tag{6.17}$$

$$s_k = \sum_{i=0}^{i=n} P_{k,n}^2(t_i) \quad (k = 0,1,\ldots,m). \tag{6.18}$$

Przykładowe wielomiany ortogonalne mają następującą postać:

$$
\begin{aligned}
P_{0,n} &= 1,\\
P_{1,n} &= 1-2\frac{t}{n},\\
P_{2,n} &= 1-6\frac{t}{n}+6\frac{t(t-1)}{n(n-1)},\\
P_{3,n} &= 1-12\frac{t}{n}+30\frac{t(t-1)}{n(n-1)}-20\frac{t(t-1)(t-2)}{n(n-1)(n-2)},\\
&\vdots
\end{aligned}
$$

Przykład 3

Znaleźć najlepszą aproksymację średniokwadratową dla funkcji $f(x)$ mierzonej w 5 punktach:

x	0.0	0.5	1.0	1.5	2.0
y	0.1	0.3	1.1	2.2	4.2

Wykorzystać wielomiany ortogonalne stopnia drugiego.

Rozwiązanie

$n=4, \quad m=2$

Wprowadzamy nową zmienną

$$t=\frac{x-x_0}{h}=\frac{x}{0.5}=2x.$$

Korzystając ze wzorów (6.16), (6.17), (6.18) oraz wykorzystując wielomiany ortogonalne, możemy wyniki przedstawić w tabelce.

x	t	y	$P_{0,4}(t)$	$P_{1,4}(t)$	$P_{2,4}(t)$
0.0	0	0.1	1	1	1
0.5	1	0.3	1	0.5	-0.5
1.0	2	1.1	1	0	-1
1.5	3	2.2	1	-0.5	-0.5
2.0	4	4.2	1	-1	1
		c_i	7.9	-5.05	1.95
		s_i	5.0	2.5	3.5
		b_i	1.58	-2.02	0.557

Stąd

$$Q_2(t)=1.58P_{0,4}(t)-2.02P_{1,4}(t)+0.557P_{2,4}(t).$$

Podstawiając $t=2x$, otrzymujemy

$$Q_2(x)=0.1-0.2x+1.1x^2.$$

powered by Misiol

Metoda najmniejszych kwadratów

Metoda najmniejszych kwadratów jest najczęściej wykorzystywaną metodą aproksymacji ze względu na jej prostotę dla małych stopni wielomianu aproksymującego.

Funkcję $f(x)$ podaną w $(n+1)$ punktach dyskretnych x_i, $(i=0,1,\ldots,n)$ należy aproksymować wielomianem potęgowym o postaci

$$Q_m(x)=\sum_{i=0}^{i=m} a_i\cdot x^i, \tag{6.19}$$

gdzie m - stopień wielomianu aproksymującego $(m<n)$.

Minimalizujemy miarę odległości (6.15), która jest funkcją wypukłą. Znajdując pochodne cząstkowe wyrażenia (6.15) względem wszystkich niewiadomych współczynników a_i $(i=0,1,\ldots,m)$ oraz przyrównując je do zera, otrzymamy układ $(m+1)$ równań liniowych z $(m+1)$ niewiadomymi $a_0,a_1,\ldots,a_m$. Jeżeli oznaczymy

$$S_k=\sum_{i=0}^{i=n} x_i^k, \qquad k=0,1,\ldots,2m \tag{6.20}$$

oraz

$$T_k=\sum_{i=0}^{i=n} x_i^k\cdot y_i, \qquad k=0,1,\ldots,m, \tag{6.21}$$

to układ równań można zapisać następująco:

$$
\begin{aligned}
&a_0\cdot S_0+a_1\cdot S_1+\cdots+a_m\cdot S_m=T_0\\
&a_0\cdot S_1+a_1\cdot S_2+\cdots+a_m\cdot S_{m+1}=T_1\\
&a_0\cdot S_2+a_1\cdot S_3+\cdots+a_m\cdot S_{m+2}=T_2\\
&\vdots\\
&a_0\cdot S_k+a_1\cdot S_{k+1}+\cdots+a_m\cdot S_{m+k}=T_k\\
&\vdots\\
&a_0\cdot S_m+a_1\cdot S_{m+1}+\cdots+a_m\cdot S_{2m}=T_m.
\end{aligned} \tag{6.22}
$$

Jeżeli wszystkie punkty x_i, $(i=0,1,\ldots,n)$ są różne, to układ równań (6.22) ma jedyne rozwiązanie, a wielomian $Q_m(x)$ z tak obliczonymi współczynnikami minimalizuje przyjętą miarę odległości. W przypadku, gdy $n=m$, otrzymany wielomian aproksymujący jest jednocześnie wielomianem interpolacyjnym.

Przykład 4

Wykorzystując metodę najmniejszych kwadratów znaleźć najlepszą aproksymację pierwszego stopnia dla funkcji danej w czterech punktach:

x	1	2	4	7
y	1.6	2.0	2.5	3.5

Rozwiązanie

$n=3, \quad m=1$

Wyniki przedstawimy w tabelce.

x^0	x^1	x^2	y	$x * y$
1	1	1	1.6	1.6
1	2	4	2.0	4.0
1	4	16	2.5	10.0
1	7	49	3.5	24.5
$S_0 = 4$	$S_1 = 14$	$S_2 = 70$	$T_0 = 9.6$	$T_1 = 40.1$

Stąd układ równań ma postać:

$$\begin{aligned} 4 \cdot a_0 + 14 \cdot a_1 &= 9.6 \\ 14 \cdot a_0 + 70 \cdot a_1 &= 40.1 \end{aligned}$$

i daje rozwiązanie: $a_0 = 1.3$, $a_1 = 0.3$. Wielomian aproksymujący ma postać

$$Q_1(x) = 1.3 + 0.3x.$$

1.7. Metody rozwiązywania układów równań liniowych

1.7.1. Wprowadzenie

W układzie równań liniowych

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{1n}x_n & = & b_2 \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{i1}x_1 & + & a_{i2}x_2 & + & \cdots & + & a_{in}x_n & = & b_i \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & b_m \end{array} \tag{7.1}$$

wielkości $x_1, x_2, \cdots, x_n$ są niewiadomymi, elementy a_{ik} $(i = 1, 2, \cdots, m;\ k = 1, 2, \cdots, n)$ – współczynnikami przy niewiadomych, zaś b_i $(i = 1, 2, \ldots, m)$ – wyrazami wolnymi układu. W symbolice macierzowej układ równań (7.1) ma postać

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{7.2}$$

gdzie: macierz $\boldsymbol{A}$ jest macierzą współczynników układu (macierz główna) $\boldsymbol{A} = [a_{ik}]_{mn}$, wektor $\boldsymbol{b}$ – kolumną wyrazów wolnych $\boldsymbol{b}^T = (b_1, b_2, \cdots, b_m)$, wektor $\boldsymbol{x}$ – kolumną wyrazów niewiadomych $\boldsymbol{x}^T = (x_1, x_2, \cdots, x_n)$, przy czym symbol T oznacza transponowanie.

Zakładamy, że a_{ik} oraz b_i są liczbami rzeczywistymi.

Uporządkowany układ liczb rzeczywistych $x_1, x_2, \cdots, x_n$ spełniających tożsamościowo układ (7.1) nazywamy rozwiązaniem tego układu, a poszczególne liczby $x_1, x_2, \cdots, x_n$ – pierwiastkami tego układu.

powered by Misiol

Układ równań liniowych nazywamy:

- *zgodnym*, gdy ma przynajmniej jedno rozwiązanie, przy czym może on być układem:
  - — *oznaczonym*, gdy ma dokładnie jedno rozwiązanie,
  - — *nieoznaczonym*, gdy ma nieskończenie wiele rozwiązań;
- *sprzecznym*, gdy nie ma rozwiązań.

Rozważania ograniczymy do najczęściej spotykanych przy rozwiązywaniu zagadnień technicznych układów równań – układów niejednorodnych, tj. takich, że

$$\sum_{i=1}^{m} b_i^2 > 0. \tag{7.3}$$

Warunki istnienia rozwiązania układu niejednorodnego (7.1) podaje twierdzenie Kroneckera-Capelliego.

Układ równań (7.1)jest rozwiązywalny wtedy i tylko wtedy, gdy

$$\text{rząd}(\boldsymbol{A}) = \text{rząd}(\boldsymbol{B}) = r,$$

gdzie $\boldsymbol{B}$ - macierz rozszerzona powstała z macierzy $\boldsymbol{A}$ przez dołączenie do niej wektora $\boldsymbol{b}$ jako $n + 1$ kolumny, przy czym:

- istnieje dokładnie jedno rozwiązanie, gdy $r = n$;
- układ ma nieskończenie wiele rozwiązań zależnych od $e = n - r$ parametrów, gdy $r < n$;
- układ jest sprzeczny, gdy rząd $(\boldsymbol{A})$ < rząd $(\boldsymbol{B})$.

Wspólny rząd macierzy $\boldsymbol{A}$ i $\boldsymbol{B}$ nazywamy rzędem układu. Obliczenie pierwiastków układu równań liniowych niejednorodnych wymaga:

1. Wyznaczenia rzędu układu r oraz odpowiadającego mu minora D macierzy $\boldsymbol{A}$, różnego od zera.
2. Wyodrębnienia równań, których współczynniki wchodzą do minora D i odrzucenia pozostałych równań. W wyodrębnionych równaniach r niewiadomych stojących przy współczynnikach tworzących minor D pozostawiamy po lewej stronie r równań, a pozostałe $n-r$ niewiadomych traktujemy jako parametry i umieszczamy je po prawej stronie równań.
3. Rozwiązania otrzymanego układu r równań z n niewiadomymi. Niewiadome te są liniowymi funkcjami $n - r$ parametrów. Otrzymane rozwiązanie nazywamy rozwiązaniem ogólnym.
4. Przyjęcia za parametry dowolnych wartości liczbowych – uzyskujemy w ten sposób tzw. rozwiązanie szczególne układu.

Powyższe postępowanie wymaga znajomości metod rozwiązywania układów równań liniowych rzędu n. Metody takie przedstawiono w następnych częściach rozdziału, dzieląc je na metody dokładne i metody przybliżone.

1.7.2. Metody dokładne

Metodami dokładnymi będziemy nazywać takie metody rozwiązywania układów równań liniowych, które pozwalają znaleźć dokładne rozwiązanie w wyniku skończonej liczby elementarnych działań arytmetycznych, a liczba działań zależy tylko od algorytmu metody i rzędu układu.

Wzory Cramera

Dany jest układ Cramera, tj. układ n równań z n niewiadomymi. W zapisie macierzowym ma on postać

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \tag{7.4}$$

Macierz współczynników $\boldsymbol{A}$ jest macierzą nieosobliwą, tzn. $\det \boldsymbol{A} \neq 0$. Istnieje więc macierz odwrotna $\boldsymbol{A}^{-1}$. W celu znalezienia rozwiązania układu mnożymy obie strony równania (7.4) lewostronnie przez macierz $\boldsymbol{A}^{-1}$ i otrzymujemy

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}. \tag{7.5}$$

Macierz odwrotną $\boldsymbol{A}^{-1}$ obliczamy ze wzoru

$$\boldsymbol{A}^{-1} = \frac{\boldsymbol{A}^D}{\det \boldsymbol{A}}, \tag{7.6}$$

gdzie $\boldsymbol{A}^D$ jest transponowaną macierzą dopełnień algebraicznych, czyli tzw. macierzą dołączoną o postaci

$$\boldsymbol{A}^D = \begin{bmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1k} & A_{2k} & \cdots & A_{nk} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{bmatrix}, \tag{7.7}$$

gdzie elementy A_{ij} są dopełnieniami algebraicznymi elementów a_{ij}, tzn. wartościami wyznaczników stopnia $n-1$ powstałych z wyznacznika $\det \boldsymbol{A}$ przez skreślenie i-tego wiersza i j-tej kolumny pomnożonymi przez $(-1)^{i+j}$. Otrzymujemy więc

$$\boldsymbol{x} = \frac{1}{\det \boldsymbol{A}} \boldsymbol{A}^D \boldsymbol{b}. \tag{7.8}$$

Zauważmy, że w iloczynie

$$\boldsymbol{A}^D\boldsymbol{b} = \begin{bmatrix} A_{11}b_1 + A_{21}b_2 + \cdots + A_{n1}b_n \\ A_{12}b_1 + A_{22}b_2 + \cdots + A_{n2}b_n \\ \vdots \\ A_{1k}b_1 + A_{2k}b_2 + \cdots + A_{nk}b_n \\ \vdots \\ A_{1n}b_1 + A_{2n}b_2 + \cdots + A_{nn}b_n \end{bmatrix} \tag{7.9}$$

powered by Misiol

dowolny k-ty wiersz wektora jest równy wartości wyznacznika D_k powstałego z wyznacznika $\det \boldsymbol{A}$ przez zastąpienie w nim k-tej kolumny kolumną wyrazów wolnych

$$D_k = \sum_{j=1}^{n} A_{jk} b_k = \det \begin{bmatrix} a_{11} & \cdots & a_{1,k-1} & b_1 & a_{1,k+1} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2,k-1} & b_2 & a_{2,k+1} & \cdots & a_{2n} \\ \vdots & & \vdots & \vdots & & \vdots & \\ a_{i1} & \cdots & a_{i,k-1} & b_n & a_{i,k+1} & \cdots & a_{in} \\ \vdots & & \vdots & \vdots & & \vdots & \\ a_{n1} & \cdots & a_{n,k-1} & b_n & a_{n,k+1} & \cdots & a_{nn} \end{bmatrix}. \tag{7.10}$$

Ostatecznie równość (7.5) przyjmie postać

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ \vdots \\ x_n \end{bmatrix} = \frac{1}{\det \boldsymbol{A}} \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_k \\ \vdots \\ D_n \end{bmatrix}. \tag{7.11}$$

Z tego równania bezpośrednio wynikają wzory Cramera

$$x_k = \frac{D_k}{\det \boldsymbol{A}}, \quad k = 1, 2, \cdots, n. \tag{7.12}$$

Metoda eliminacji - Gaussa

Rozwiązywanie układu równań liniowych

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & a_{1,n+1} \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & + & \cdots & + & a_{2n}x_n & = & a_{2,n+1} \\ \vdots & & \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & a_{n3}x_3 & + & \cdots & + & a_{nn}x_n & = & a_{n,n+1} \end{array} \tag{7.13}$$

metodą Gaussa polega na kolejnym rugowaniu niewiadomych, prowadzącym do utworzenia układu pomocniczego o trójkątnej macierzy współczynników przy niewiadomych, a więc układu o postaci

$$\begin{array}{ccccccccccc} x_1 & + & b_{12}x_2 & + & b_{13}x_3 & + & \cdots & + & b_{1n}x_n & = & b_{1,n+1} \\ & & x_2 & + & b_{23}x_3 & + & \cdots & + & b_{2n}x_n & = & b_{2,n+1} \\ & & & & & & \ddots & & \vdots & & \vdots \\ & & & & & & & & x_n & = & b_{n,n+1}. \end{array} \tag{7.14}$$

Wyznaczanie współczynników b_{ik} układu (7.14) nazywamy postępowaniem prostym, a znajdowanie niewiadomych $x_1, x_2, \cdots, x_n$ tego układu – postępowaniem odwrotnym. Niewiadome znajdujemy ze wzoru

$$x_k = b_{k,n+1} - b_{k,k+1}x_{k+1} - \cdots - b_{nk}x_n, \quad k = n-1, n-2, \cdots, 1. \tag{7.15}$$

Współczynniki pomocniczego układu (7.14) możemy obliczyć jednym z dwóch podanych niżej sposobów.

I. Schemat jedynego dzielenia

Załóżmy, że w układzie (7.13) $a_{11} \neq 0$ i podzielmy obie strony pierwszego równania tego układu przez współczynnik a_{11}, zwany elementem kierującym. Otrzymujemy

$$x_1 + b_{12}x_2 + b_{13}x_3 + \cdots + b_{1n}x_n = b_{1,n+1}, \tag{7.16}$$

przy czym

$$b_{ik} = \frac{a_{1k}}{a_{11}}, \quad k = 2, 3, \cdots, n+1. \tag{7.17}$$

Równanie (7.16) mnożymy kolejno przez $a_{21}, a_{31}, \cdots, a_{1n}$ i tak powstałe równania odejmujemy odpowiednio od drugiego, trzeciego,...,n-tego równania układu (7.13). Otrzymujemy następujący układ $n-1$ równań

$$\begin{array}{ccccccccc} a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & + & \cdots & + & a_{2n}^{(1)}x_n & = & a_{2,n+1}^{(1)} \\ a_{32}^{(1)}x_2 & + & a_{33}^{(1)}x_3 & + & \cdots & + & a_{3n}^{(1)}x_n & = & a_{3,n+1}^{(1)} \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{n2}^{(1)}x_2 & + & a_{n3}^{(1)}x_3 & + & \cdots & + & a_{nn}^{(1)}x_n & = & a_{n,n+1}^{(1)}, \end{array} \tag{7.18}$$

gdzie

$$a_{ik}^{(1)} = a_{ik} - a_{i1}b_{1k}, \quad i, k \geq 2. \tag{7.19}$$

W drugim kroku eliminacji postępujemy analogicznie z układem (7.18), dzieląc obie strony pierwszego równania tego układu przez $a_{22}^{(1)} \neq 0$. Otrzymujemy równanie

$$x_2 + b_{23}x_3 + b_{24}x_4 + \cdots + b_{2n}x_n = b_{2,n+1}, \tag{7.20}$$

gdzie

$$b_{2k} = \frac{a_{2k}^{(1)}}{a_{22}^{(1)}}, \quad k \geq 3. \tag{7.21}$$

Następnie mnożąc obie strony równania (7.20) kolejno przez $a_{32}^{(1)}, a_{42}^{(1)}, \cdots, a_{n2}^{(1)}$ i odejmując tak powstałe równania od odpowiednich równań układu (7.18), otrzymujemy

$$\begin{array}{ccccccccc} a_{33}^{(2)}x_3 & + & a_{34}^{(2)}x_4 & + & \cdots & + & a_{3n}^{(2)}x_n & = & a_{3,n+1}^{(2)} \\ a_{43}^{(2)}x_3 & + & a_{44}^{(2)}x_4 & + & \cdots & + & a_{4n}^{(2)}x_n & = & a_{4,n+1}^{(2)} \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ a_{n3}^{(2)}x_3 & + & a_{n4}^{(2)}x_4 & + & \cdots & + & a_{nn}^{(2)}x_n & = & a_{n,n+1}^{(2)}, \end{array} \tag{7.22}$$

gdzie

$$a_{ik}^{(2)} = a_{ik}^{(1)} - a_{i2}^{(1)}b_{2k}, \quad i, k \geq 3. \tag{7.23}$$

W kolejnym s-tym kroku eliminacji otrzymujemy równanie

$$x_s + b_{s,s+1}x_{s+1} + \ldots + b_{sn}x_n = b_{s,n+1} \tag{7.24}$$

powered by Misiol

oraz układ równań

$$\begin{array}{ccccccc} a_{s+1,s+1}^{(s)}x_{s+1} & + & \cdots & + & a_{s+1,n}^{(s)}x_n & = & a_{s+1,n+1}^{(s)} \\ \vdots & & \ddots & & \vdots & & \vdots \\ a_{n,s+1}^{(s)}x_{s+1} & + & \cdots & + & a_{nn}^{(s)}x_n & = & a_{n,n+1}^{(s)}, \end{array} \tag{7.25}$$

gdzie:

$$b_{sk} = \frac{a_{sk}^{(s-1)}}{a_{ss}^{(s-1)}}, \quad k \geq s+1, \tag{7.26}$$

$$a_{ik}^{(s)} = a_{ik}^{(s-1)} - a_{is}^{(s-1)}b_{sk}, \quad i, k \geq s+1. \tag{7.27}$$

Po $n-1$ krokach eliminacji otrzymujemy układ postaci (7.14).

II. Schemat elementów podstawowych

W schemacie elementów podstawowych kolejność eliminowanych zmiennych uzależniona jest od wartości bezwzględnej współczynników przy niewiadomych. W macierzy rozszerzonej $\boldsymbol{B}$ układu (7.13) wybieramy największy co do wartości bezwzględnej współczynnik a_{ik} $(i, k \leq n)$, nazywany elementem podstawowym. Wiersz i-ty, w którym leży element a_{ik}, nazywa się wierszem podstawowym. Każdy pozostały j-ty wiersz przekształcamy dodając do odpowiednich elementów tego wiersza elementy wiersza podstawowego pomnożone przez czynniki

$$m_i = -\frac{a_{jk}}{a_{ik}}.$$

Otrzymujemy macierz, której k-ta kolumna oprócz elementu a_{ik} zawiera same zera. Usuwamy tę kolumnę oraz i-ty wiersz i otrzymujemy macierz $\boldsymbol{B}^{(1)}$ o wymiarach $(n-1) \times n$. Analogicznie z macierzy $\boldsymbol{B}^{(1)}$ obliczamy macierz $\boldsymbol{B}^{(2)}$ itd. aż do macierzy $\boldsymbol{B}^{(n-1)}$, która jest macierzą o jednym wierszu (podstawowym) i dwóch kolumnach. Z każdej z tych macierzy $(\boldsymbol{B}, \boldsymbol{B}^{(1)}, \ldots, \boldsymbol{B}^{(n-1)})$ wypisujemy wiersze podstawowe, otrzymując trójkątny układ równań, z którego podobnie jak z (7.14) znajdujemy niewiadome $x_1, x_2, \ldots, x_n$, pamiętając, że mogą wystąpić współczynniki układu trójkątnego $b_{ii} = 1$. Wtedy wzór (7.15) zastąpimy wzorem

$$x_k = \frac{1}{b_{kk}} b_{k,n+1} - b_{k,k+1}x_{k+1} - \ldots - b_{kn}x_n, \quad k = n-1, n-2, \ldots, 1. \tag{7.28}$$

Przyjęcie elementów podstawowych jako elementów macierzy współczynników o największej wartości bezwzględnej ma na celu zmniejszenie wpływu błędu zaokrągleń na wynik rozwiązania.

Metoda pierwiastków kwadratowych – Banachiewicza

Rozwiązanie układu równań liniowych można uprościć w przypadku, gdy macierz główna układu jest macierzą symetryczną. Wynika to z faktu, że każdą nieosobliwą macierz symetryczną można przedstawić w postaci iloczynu dwóch macierzy trójkątnych transponowanych względem siebie, tj.

$$\boldsymbol{A} = \boldsymbol{T}\boldsymbol{T}^T, \tag{7.29}$$

gdzie:

$$T = \begin{bmatrix} t_{11} & 0 & \cdots & 0 \\ t_{21} & t_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \\ t_{n1} & t_{n2} & \cdots & t_{nn} \end{bmatrix}, \quad T^T = \begin{bmatrix} t_{11} & t_{21} & \cdots & t_{n1} \\ 0 & t_{22} & \cdots & t_{n2} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & t_{nn} \end{bmatrix}.$$

Mnożąc macierz T przez T^T oraz porównując elementy o tych samych wskaźnikach w macierzy A i w obliczonym iloczynie otrzymujemy równania:

$$\begin{aligned} t_{i1}t_{j1} + t_{i2}t_{j2} + \ldots + t_{ii}t_{ji} &= a_{ij}, \quad i < j, \\ t_{i1}^2 + t_{i2}^2 + \ldots + t_{ii}^2 &= a_{ii}, \quad i = j, \quad j = 1, 2, \ldots, n. \end{aligned} \tag{7.30}$$

Stąd można wyznaczyć elementy t_{ij} macierzy T:

$$\begin{aligned} t_{11} &= a_{11}, \\ t_{i1} &= \frac{a_{1i}}{t_{11}} = \frac{a_{i1}}{t_{11}}, \\ t_{ii} &= a_{ii} - \sum_{k=1}^{i-1} t_{ik}^2, \quad i > 1, \\ t_{ji} &= \begin{cases} \dfrac{a_{ij} - \sum\limits_{k=1}^{i-1} t_{ik}t_{kj}}{t_{ii}}, & i < j \\ 0 & i > j. \end{cases} \end{aligned} \tag{7.31}$$

Zauważmy, że jeśli wystąpi $t_{ii}^2 \leq 0$, to wszystkie elementy i-tej kolumny T będą urojone. Nie wpływa to na tok obliczeń. Potrzeba jednak, by wszystkie $t_{ii} \neq 0$, a wtedy $\det A \neq 0$ oraz układ równań liniowych

$$Ax = b \tag{7.32}$$

ma jedno i tylko jedno rozwiązanie. Aby je obliczyć, uwzględnimy najpierw (7.29) otrzymując

$$TT^T x = b. \tag{7.33}$$

Stosując do układu (7.33) nieosobliwe podstawienie

$$T^T x = y \tag{7.34}$$

mamy

$$Ty = b \tag{7.35}$$

lub w postaci rozwiniętej

$$\begin{aligned} t_{11}y_1 & & &= b_1 \\ t_{21}y_1 &+ t_{22}y_2 & &= b_2 \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1}y_1 &+ t_{n2}y_2 &+ \cdots + t_{nn}y_n &= b_n. \end{aligned} \tag{7.36}$$

powered by Misiol

Stąd:

$$\begin{aligned} y_1 &= \frac{b_1}{t_{11}}, \\ y_i &= \frac{b_i - \sum\limits_{j=1}^{i-1} t_{ij}y_i}{t_{ii}}, \quad i > 1. \end{aligned} \tag{7.37}$$

Pisząc równanie (7.34) w postaci rozwiniętej, otrzymujemy

$$\begin{aligned} t_{11}x_1 &+ t_{21}x_2 &+ \cdots &+ t_{n1}x_n &= y_1 \\ & t_{22}x_2 &+ \cdots &+ t_{n2}x_n &= y_2 \\ & & \ddots & \vdots & \vdots \\ & & & t_{nn}x_n &= y_n. \end{aligned} \tag{7.38}$$

Znając wcześniej obliczone wartości $y_1, y_2, \cdots, y_n$, łatwo znajdujemy wartości pierwiastków $x_1, x_2, \cdots, x_n$:

$$\begin{aligned} x_n &= \frac{y_n}{t_{nn}}, \\ x_i &= \frac{y_i - \sum\limits_{k=i+1}^{n} t_{ki}x_i}{t_{ii}}, \quad i < n. \end{aligned} \tag{7.39}$$

W metodzie Banachiewicza obliczamy więc kolejno elementy t_{ij} według wzorów (7.31), niewiadome pomocnicze y_i według wzorów (7.37), a następnie pierwiastki x_i według wzorów (7.39).

Metoda Choleskiego

Podobnie jak w metodzie Banachiewicza tu również zastępujemy macierz główną A układu równań iloczynem dwóch macierzy trójkątnych

$$A = BC, \tag{7.40}$$

gdzie:

$$B = \begin{bmatrix} b_{11} & 0 & \cdots & 0 \\ b_{21} & b_{22} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}, \quad C = \begin{bmatrix} 1 & c_{12} & \cdots & c_{1n} \\ 0 & 1 & \cdots & c_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Mnożąc macierze B i C, a następnie przyrównując elementy otrzymanego iloczynu do odpowiednich elementów macierzy A, znajdujemy podane niżej zależności rekurencyjne w celu wyznaczenia niewiadomych elementów macierzy B oraz C:

$$b_{i1} = a_{i1}, \quad b_{ij} = a_{ij} - \sum_{k=1}^{j-1} b_{ik}c_{kj}, \quad 1 < j \leq i, \tag{7.41}$$

$$c_{1j} = \frac{a_{1j}}{b_{11}}, \quad c_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} b_{ik}c_{kj}}{b_{ii}}, \quad 1 < i \le j. \tag{7.42}$$

Znając wartości macierzy $\boldsymbol{B}$ i $\boldsymbol{C}$ zastępujemy nimi macierz $\boldsymbol{A}$ w układzie równań

$$\boldsymbol{Ax} = \boldsymbol{b}, \tag{7.43}$$

otrzymując

$$\boldsymbol{BCx} = \boldsymbol{b}. \tag{7.44}$$

Układ (7.44) można zastąpić teraz dwoma równoważnymi układami:

$$\boldsymbol{Cx} = \boldsymbol{y}, \tag{7.45}$$

$$\boldsymbol{By} = \boldsymbol{b}. \tag{7.46}$$

Układy (7.45) i (7.46) są układami o macierzach trójkątnych, łatwymi do rozwiązania. Można w tym celu wyprowadzić następujące wzory:

$$y_1 = \frac{a_{1,n+1}}{b_{11}}, \qquad y_i = \frac{a_{i,n+1} - \sum_{k=1}^{i-1} b_{ik}y_k}{b_{ii}}, \quad i = 2, 3, \ldots, n, \tag{7.47}$$

$$x_n = y_n, \qquad x_i = y_i - \sum_{k=i+1}^{n} c_{ik}x_k, \quad i = n-1, n-2, \ldots, 1. \tag{7.48}$$

1.7.3. Metody przybliżone

Metody przybliżone (iteracyjne) umożliwiają obliczenie przybliżonych wartości pierwiastków układu równań liniowych za pomocą kolejnych aproksymacji. W tym celu przyjmujemy pewien, na ogół dowolny wektor $x^{(0)}$ jako rozwiązanie początkowe i według określonego schematu tworzymy kolejno ciąg wektorów $x^{(1)}, x^{(2)}, \ldots, x^{(n)}, \ldots$ tak, aby wektor $x^{(n+1)}$ lepiej aproksymował rozwiązanie układu od wektora $x^{(n)}$. Wynika stąd, że w metodach iteracyjnych liczba wykonanych działań zależy od wymaganej dokładności rozwiązania.

Metoda iteracji prostej

W metodzie iteracji prostej przekształcamy układ równań liniowych

$$\begin{array}{ccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{1n}x_n & = & b_2 \\
\vdots & & \vdots & & \ddots & & \vdots & & \vdots \\
a_{i1}x_1 & + & a_{i2}x_2 & + & \cdots & + & a_{in}x_n & = & b_i \\
\vdots & & \vdots & & \ddots & & \vdots & & \vdots \\
a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & b_m
\end{array} \tag{7.49}$$

powered by Misiol

rozwiązując każde i-te równanie względem niewiadomej x_i przy założeniu, że $a_{ii} \neq 0$. Otrzymujemy układ równoważny

$$\begin{array}{ccccccccc}
x_1 & = & g_1 & + & h_{12}x_2 & + & \cdots & + & h_{1n}x_n \\
x_2 & = & g_2 & + & h_{21}x_1 & + & \cdots & + & h_{2n}x_n \\
\vdots & & \vdots & & \vdots & & \ddots & & \vdots \\
x_i & = & g_i & + & h_{i1}x_1 & + & \cdots & + & h_{in}x_n \\
\vdots & & \vdots & & \vdots & & \ddots & & \vdots \\
x_n & = & g_n & + & h_{n1}x_1 & + & \cdots & + & h_{n,n-1}x_{n-1},
\end{array} \tag{7.50}$$

gdzie

$$g_i = \frac{b_i}{a_{ii}}, \quad h_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}} & dla \quad i \neq j, \\ 0 & dla \quad i = j, \quad i, j = 1, 2, \ldots, n. \end{cases} \tag{7.51}$$

W zapisie macierzowym przekształcony układ równań ma postać

$$\boldsymbol{x} = \boldsymbol{g} + \boldsymbol{Hx}. \tag{7.52}$$

Równanie to rozwiązujemy metodą kolejnych przybliżeń. Za przybliżenie początkowe $\boldsymbol{x}^{(0)}$ przyjmujemy najczęściej kolumnę wyrazów wolnych, tzn. $\boldsymbol{x}^{(0)} = \boldsymbol{g}$. Otrzymujemy przybliżenie pierwsze

$$\boldsymbol{x}^{(1)} = \boldsymbol{g} + \boldsymbol{Hx}^{(0)},$$

następnie przybliżenie drugie, jako

$$\boldsymbol{x}^{(2)} = \boldsymbol{g} + \boldsymbol{Hx}^{(1)}.$$

Ogólnie, dowolne $(k+1)$-sze przybliżenie obliczamy według wzoru

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{g} + \boldsymbol{Hx}^{(k)}, \quad k = 0, 1, 2, \ldots. \tag{7.53}$$

W postaci skalarnej powyższy wzór przyjmie postać

$$x_i^{(0)} = g_i$$

$$x^{(k+1)} = g_i + \sum_{j=1}^{n} h_{ij}x_j^{(k)}, \quad i = 1, 2, \ldots, \quad k = 0, 1, 2, \ldots \tag{7.54}$$

Jeżeli proces przybliżeń jest zbieżny, to

$$\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}, \tag{7.55}$$

gdzie $\boldsymbol{x}$ jest jedynym (dokładnym) rozwiązaniem układu.

Warunkiem dostatecznym zbieżności metody iteracji prostej jest, aby dowolna z norm macierzy $\boldsymbol{H}$ była mniejsza od jedności, tzn.:

$$||\boldsymbol{H}||_I = \max_i \sum_{j=1}^{n} |h_{ij}| \le \mu < 1, \tag{7.56}$$

$$||\boldsymbol{H}||_{II} = \max_j \sum_{i=1}^{n} |h_{ij}| \leq \vartheta < 1, \tag{7.57}$$

$$||\boldsymbol{H}||_{III} = \sum_{i=1}^{n}\sum_{j=1}^{n} |h_{ij}|^2 \leq \tau < 1. \tag{7.58}$$

Powyższe warunki zbieżności możemy odnieść również do układu równań postaci $\boldsymbol{Ax} = \boldsymbol{b}$, wstawiając we wzorach (7.56)–(7.58) zamiast h_{ij} wartości

$$h_{ij} = \begin{cases} \dfrac{a_{ij}}{a_{ii}} & \text{dla} \quad i \neq j, \\ 0 & \text{dla} \quad i = j. \end{cases} \tag{7.59}$$

Metoda Seidela

Metodę Seidela można uważać za pewną modyfikację metody iteracji prostej. Tu również korzystamy z układu równań przekształconego do postaci $\boldsymbol{x} = \boldsymbol{g} + \boldsymbol{Hx}$. Jednak $(k+1)$-sze przybliżenie kolejnej zmiennej $x_i^{(k+1)}$ obliczamy za pomocą przybliżeń zmiennych wcześniej wyznaczonych w aktualnej, $(k+1)$-szej iteracji $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$ oraz pozostałych zmiennych w poprzedniej, k-tej iteracji $x_1^{(k)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)}$.

Wybieramy więc w dowolny sposób przybliżenie początkowe $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$. Zakładając, że znane są przybliżenia $x_i^{(k)}$ $(i = 1, 2, \ldots, n)$, obliczamy przybliżenia $x_i^{(k+1)}$ według wzorów Seidela:

$$x_1^{(k+1)} = g_1 + \sum_{j=1}^{n} h_{1j} x_j^{(k)}, \tag{7.60}$$

$$x_2^{(k+1)} = g_2 + \sum_{j=2}^{n} h_{2j} x_j^{(k)} \quad \text{itd.} \tag{7.61}$$

Ogólnie

$$x_i^{(k+1)} = g_i + \sum_{j=1}^{i-1} h_{ij} x_j^{(k+1)} + \sum_{j=i}^{n} h_{ij} x_j^{(k)}, \quad i = 1, 2, \ldots, n. \tag{7.62}$$

Również w metodzie Seidela obowiązują warunki dostateczne zbieżności (7.56)–(7.58). Zazwyczaj proces Seidela jest szybciej zbieżny niż proces iteracji prostej; może być nawet zbieżny w przypadku, gdy proces iteracji prostej jest rozbieżny. Istnieją też sytuacje odwrotne, które można określić za pomocą warunków koniecznych i wystarczających zbieżności. W praktyce warunki te są jednak rzadko stosowane ze względu na dużą złożoność obliczeniową.

Metoda relaksacji

W metodzie relaksacji poszukuje się rozwiązania układu równań liniowych (7.49) zerując iteracyjnie wartości elementów wektora błędów rozwiązania przybliżonego $\boldsymbol{x}_0$, tzw. residuum $\boldsymbol{r}$, zdefiniowanego następująco

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax}_0. \tag{7.63}$$

powered by Misiol

W tym celu układ równań przekształcamy, przenosząc wyrazy wolne na lewą stronę i dzieląc pierwsze równanie przez $(-a_{11})$, drugie przez $(-a_{22})$ itd. Otrzymujemy układ przygotowany do relaksacji

$$\begin{array}{ccccccccccc} -x_1 & + & c_{12}x_2 & + & \cdots & + & c_{1n}x_n & + & d_1 & = & 0 \\ c_{21}x_1 & - & x_2 & + & \cdots & + & c_{2n}x_n & + & d_2 & = & 0 \\ \vdots & & & & \ddots & & & & & & \vdots \\ c_{n1}x_1 & + & c_{n2}x_2 & + & \cdots & - & x_n & + & d_n & = & 0, \end{array} \tag{7.64}$$

gdzie:

$$c_{ij} = -\frac{a_{ij}}{a_{ii}}, \quad i \neq j,$$

$$d_i = \frac{b_i}{a_{ii}}.$$

Wybieramy w dowolny sposób przybliżenie początkowe $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$ pierwiastków układu (7.64) starając się, by odpowiadały one choć w pewnej mierze szukanym niewiadomym $x_1, x_2, \ldots, x_n$. Podstawiając przybliżenia początkowe do układu (7.64) obliczamy residua:

$$r_1^{(0)} = d_1 - x_1^{(0)} + \sum_{j=2}^{n} c_{1j} x_j^{(0)}, \tag{7.65}$$

$$r_2^{(0)} = d_2 - x_2^{(0)} + \sum_{j=1, j\neq 2}^{n} c_{2j} x_j^{(0)}, \tag{7.66}$$

$$\vdots$$

$$r_i^{(0)} = d_i - x_i^{(0)} + \sum_{\substack{j=1 \\ j\neq i}}^{n-1} c_{ij} x_j^{(0)}, \tag{7.67}$$

$$\vdots$$

$$r_n^{(0)} = d_n - x_n^{(0)} + \sum_{j=1}^{n-1} c_{nj} x_j^{(0)}. \tag{7.68}$$

(7.69)

Metoda relaksacji polega na zerowaniu w każdym kroku obliczeniowym największego co do wartości bezwzględnej residuum. W tym celu zmieniamy odpowiadający temu residuum $r_s^{(k+1)}$ składnik przybliżenia $r_s^{(k)}$ przez dodanie przyrostu $x_s^{(k)} = r_s^{(k)}$. Wtedy $r_s^{(k+1)} = 0$, a pozostałe $r_i^{(k+1)} = r_i^{(k)} + c_{is} x_s^{(k)}$. Postępowanie należy zakończyć w momencie, gdy w granicach zadanej dokładności wszystkie residua ostatniego przekształconego układu będą równe zeru. Wartości pierwiastków układu równań obliczamy dodając do przybliżeń początkowych $x_i^{(0)}$, odpowiadające im wartości wyznaczonych w kolejnych krokach przyrostów $x_i^{(k)}$, $(k = 0, 1, 2, \ldots)$.

Istnieje analogia warunków zbieżności procesu relaksacyjnego do warunków zbieżności poprzednich metod iteracyjnych.

1.7.4. Błędy rozwiązań układów równań liniowych

Źródła błędów

Istnieją cztery podstawowe źródła błędów w numerycznie obliczonym rozwiązaniu układu równań liniowych.

- Błąd spowodowany niedokładnymi wartościami współczynników układu równań (błąd początkowy). Przypadek ten zachodzi wówczas, gdy współczynniki układu są wynikiem rozpatrywania konkretnych zagadnień fizycznych i mogą być zniekształcone ze względu na nieuniknione błędy obserwacji.

- Błąd wyznaczania rozwiązania, spowodowany sposobem działania metody (błąd metody), gdy np. wyznaczamy ciąg wektorów $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots$ zbieżny do rozwiązania, lecz po skończonym czasie obliczeń przyjmujemy pewien wektor $\boldsymbol{x}^{(N)}$ jako przybliżone rozwiązanie.

- Błąd powstający na skutek konieczności zaokrąglania liczb w procesie obliczeniowym (błąd zaokrągleń). Błąd tego rodzaju ma istotny wpływ na dokładność rozwiązania, przede wszystkim wtedy, gdy na pewnym etapie obliczeniowym zachodzi konieczność odejmowania dwóch bliskich siebie wartości.

- Słabe uwarunkowanie macierzy. Macierzą słabo uwarunkowaną nazywa się macierz o wyznaczniku bliskim zeru. Macierz ta ma tę własność, że nawet małe błędy względne elementów tej macierzy prowadzą do dużych błędów względnych w rozwiązaniu układu równań. Ilustruje to podany niżej przykład. Układ równań

$$\begin{aligned} 2x + 6y &= 8 \\ 2x + 6.00001y &= 8.00001 \end{aligned}$$

ma rozwiązanie $x = 1$ i $y = 1$, natomiast układ równań

$$\begin{aligned} 2x + 6y &= 8 \\ 2x + 5.99999y &= 8.00002 \end{aligned}$$

ma rozwiązanie $x = 10$ i $y = -2$. Łatwo można sprawdzić, że wyznaczniki odpowiednich macierzy są bliskie zeru.

Pojęcie macierzy słabo uwarunkowanej wiąże się zatem ściśle z małą wartością bezwzględną wyznacznika macierzy $\boldsymbol{A}$ oraz z wielkością zmian elementów macierzy odwrotnej $\boldsymbol{A}^{-1}$ pod wpływem zmian elementów macierzy $\boldsymbol{A}$. A. Jako miarę słabego uwarunkowania macierzy $\boldsymbol{A}$ proponuje się N - warunkującą liczbę macierzy $\boldsymbol{A}$ daną wzorem

$$N = \frac{1}{n} N(\boldsymbol{A}) \cdot N(\boldsymbol{A}^{-1}), \tag{7.70}$$

gdzie

$$N(\boldsymbol{A}) = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2}. \tag{7.71}$$

Liczba warunkująca N jest zawsze nie mniejsza od jedności. Dla macierzy ortogonalnych, najlepiej uwarunkowanych $N = 1$. Duża wartość N (rzędu dziesiątek i więcej)

powered by Misiol

oznacza, że macierz jest słabo uwarunkowana. Zauważmy, że gdy dowolny wiersz macierzy $\boldsymbol{A}$ pomnożymy lub podzielimy przez dużą liczbę, to wartość liczby warunkującej zmienia się. Oznacza to, że układ o słabo uwarunkowanej macierzy można przez odpowiednie przekształcenie sprowadzić do układu o lepiej uwarunkowanej macierzy. Z drugiej jednak strony przez niewłaściwie dobrane przekształcenie można układ dobrze uwarunkowany sprowadzić do układu źle uwarunkowanego.

Oszacowanie błędu rozwiązań

Przy szacowaniu błędu rozwiązania układu równań liniowych będziemy posługiwać się jedną z norm wektora zdefiniowanych następująco:

$$||\boldsymbol{x}||_I = \max_i |x_i|, \tag{7.72}$$

$$||\boldsymbol{x}||_{II} = \sum_{i=1}^{n} |x_i|, \tag{7.73}$$

$$||\boldsymbol{x}||_{III} = \sqrt{\sum_{i=1}^{n} |x_i|^2}. \tag{7.74}$$

Dla macierzy używać będziemy norm podanych wcześniej wzorami (7.56)–(7.58). Ze względu na obszerność zagadnienia ograniczymy się do przytoczenia wyników końcowych analizy błędów.

Wpływ błędu początkowego układu równań $\boldsymbol{Ax} = \boldsymbol{b}$ można rozpatrywać niezależnie od rodzaju zastosowanej metody. Oznaczmy przez $\boldsymbol{x}_0$ obliczone rozwiązanie, $\boldsymbol{x}$ - dokładne rozwiązanie, $\boldsymbol{r}$ - wektor reszt. Możemy więc zapisać

$$\boldsymbol{Ax}_0 = \boldsymbol{b} + \boldsymbol{r}. \tag{7.75}$$

Należy odpowiedzieć na pytanie czy $\boldsymbol{x}_0$ możemy uznać za dobre przybliżenie, jeżeli wektor reszt $\boldsymbol{r}$ ma małe wartości składowych. W tym celu odejmiemy stronami równania (7.75) i (7.2), otrzymując

$$\boldsymbol{Ax}_0 - \boldsymbol{Ax} = \boldsymbol{r}, \quad \text{stąd} \quad \boldsymbol{x}_0 - \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{r},$$

a więc

$$||\boldsymbol{x}_0 - \boldsymbol{x}|| = ||\boldsymbol{A}^{-1}\boldsymbol{r}|| \tag{7.76}$$

lub inaczej

$$||\boldsymbol{x}_0 - \boldsymbol{x}|| < ||\boldsymbol{A}^{-1}|| \cdot ||\boldsymbol{r}||. \tag{7.77}$$

Zatem, jeżeli macierz jest słabo uwarunkowana, to norma $||\boldsymbol{A}^{-1}||$ może być tak duża, że nawet przy małej wartości normy $||\boldsymbol{r}||$ prawa strona nierówności jest duża. Oznacza to, że rozwiązanie $\boldsymbol{x}_0$ może się znacznie różnić od rozwiązania dokładnego $\boldsymbol{x}$.

Przyjmijmy teraz, że wektor wyrazów wolnych $\boldsymbol{b}$ jest obarczony błędami $\delta\boldsymbol{b}$, tzn.

$$\boldsymbol{b}_0 = \boldsymbol{b} + \delta\boldsymbol{b}.$$

Wskutek tego rozwiązanie układu będzie obarczone błędem $\delta\boldsymbol{x}$, tj.

$$\boldsymbol{x}_0 = \boldsymbol{x} + \delta\boldsymbol{x}.$$

Wtedy błąd względny rozwiązania określa zależność

$$\frac{||\delta \boldsymbol{x}||}{||\boldsymbol{x}||} \leq \frac{||\boldsymbol{A}^{-1}|| \cdot ||\boldsymbol{A}|| \cdot ||\delta \boldsymbol{b}||}{||\boldsymbol{b}||}. \tag{7.78}$$

W przypadku gdy elementy macierzy $\boldsymbol{A}$ podlegają małym zmianom przedstawionym w postaci macierzy $\delta \boldsymbol{A}$

$$\delta \boldsymbol{A} = \begin{bmatrix} \delta a_{11} & \delta a_{12} & \cdots & \delta a_{1n} \\ \delta a_{21} & \delta a_{22} & \cdots & \delta a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta a_{k1} & \delta a_{k2} & \cdots & \delta a_{kn} \\ \vdots & \vdots & \ddots & \vdots \\ \delta a_{n1} & \delta a_{n2} & \cdots & \delta a_{nn} \end{bmatrix},$$

a wyrazy wolne układu są dokładne, to oszacowanie błędu względnego ma postać

$$\frac{||\delta \boldsymbol{x}||}{||\boldsymbol{x}||} \leq \frac{||\boldsymbol{E}|| \cdot ||\boldsymbol{A}^{-1}|| \cdot ||\boldsymbol{A}|| \cdot ||\delta \boldsymbol{A}||}{1 - ||\boldsymbol{A}^{-1}|| \cdot ||\delta \boldsymbol{A}||}, \tag{7.79}$$

gdzie $\boldsymbol{E}$ – macierz jednostkowa.

W problemie z niedokładnościami dotyczącymi zarówno elementów macierzy $\boldsymbol{A}$, jak i wektora $\boldsymbol{b}$ do oszacowania błędu rozwiązania korzysta się ze wzoru

$$||\delta \boldsymbol{x}||_{II} \leq \frac{\varepsilon\sigma(1 + ||\boldsymbol{x}||_{II})}{1 - \varepsilon\sigma}, \tag{7.80}$$

gdzie:

$$\varepsilon = \max_j\{\delta b_j, \max_i \delta a_{ij}\}, \tag{7.81}$$

$$\sigma = \sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_{ij}|, \tag{7.82}$$

przy czym α_{ij} są elementami macierzy $\boldsymbol{A}^{-1}$.

Błędy metody powstające przy rozwiązywaniu układu równań liniowych dotyczą metod iteracyjnych. Dla metody iteracji prostej posługujemy się zależnością

$$||\boldsymbol{x} - \boldsymbol{x}^{(k)}|| \leq \frac{||\boldsymbol{H}||}{1 - ||\boldsymbol{H}||} ||\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}||, \tag{7.83}$$

przy założeniu, że $||\boldsymbol{H}|| < 1$. Jeżeli przyjmiemy $\boldsymbol{x}_0 = \boldsymbol{g}$, to wzór (7.83) będzie miał postać

$$||\boldsymbol{x} - \boldsymbol{x}^{(k)}|| \leq \frac{||\boldsymbol{H}||^{k+1}}{1 - ||\boldsymbol{H}||} ||\boldsymbol{g}||, \tag{7.84}$$

pozwalającą na określenie liczby kroków iteracyjnych koniecznych do uzyskania przyjętej z góry dokładności (błędu przybliżenia).

W metodzie Seidela, przy warunku, że $||\boldsymbol{H}||_I < 1$, do oszacowania błędu rozwiązania posługujemy się wzorem

$$||\boldsymbol{x} - \boldsymbol{x}^{(k)}||_I \leq \frac{\mu'}{1 - \mu'} ||\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}||_I, \tag{7.85}$$

powered by Misiol

gdzie

$$\mu' = \max_i \frac{\sum_{j=1}^{n} |h_{ij}|}{1 - \sum_{j=1}^{i-1} |h_{ij}|} < 1. \tag{7.86}$$

Liczbę kroków iteracyjnych koniecznych do uzyskania założonego z góry przybliżenia możemy wyznaczyć z zależności

$$||\boldsymbol{x} - \boldsymbol{x}^{(k)}||_I \leq \frac{(\mu')^k}{1 - \mu'} ||\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}||_I. \tag{7.87}$$

Ostatni typ błędów, błąd zaokrągleń, jest ściśle związany z ciągiem działań arytmetycznych, charakterystycznym dla każdej z omawianych metod.

1.7.5. Uwagi

Dokonując wyboru metody rozwiązania układu równań liniowych należy wziąć pod uwagę rozmiar problemu, rozmiar pamięci operacyjnej, czas dostępu do maszyny cyfrowej, pożądaną dokładność obliczeń, własności macierzy głównej układu, takie jak: uwarunkowanie, rzadkość (znaczna liczba współczynników o wartości zero) oraz złożoność obliczeniową metod.

Często przy oszacowaniu błędów przybliżeń konieczna jest znajomość macierzy odwrotnej. Można ją wyznaczyć posługując się wielokrotnie algorytmem rozwiązywania układu równań, przyjmując za wektor wyrazów wolnych kolejne kolumny macierzy jednostkowej. Wtedy macierz utworzona z kolumn kolejnych rozwiązań jest macierzą odwrotną do macierzy głównej układu równań.

W zakończeniu należy przypomnieć o możliwości poprawy efektywności omawianych metod. Każdą z metod dokładnych możemy uzupełnić procedurą iteracyjnego poprawiania otrzymanego rozwiązania. Polega to na wyznaczeniu wektora różnic (7.63) i ponownym rozwiązaniu układu równań, w którym kolumnę wyrazów wolnych zastąpiono kolumną różnic. Otrzymane w ten sposób poprawki $\boldsymbol{x}$ dodajemy do wcześniej otrzymanego rozwiązania $\boldsymbol{x}_0$. Nowe wartości pierwiastków możemy ponownie korygować w sposób wyżej opisany aż do momentu, gdy wartość wektora różnic będzie dostatecznie mała.

Posługując się metodami iteracyjnymi spotykamy wymagania dotyczące zbieżności procesu obliczeniowego. Zbieżność możemy poprawić dokonując za pomocą kombinacji liniowych równań takich przekształceń, w wyniku których powstaje układ o wystarczająco dużych wartościach bezwzględnych elementów macierzy współczynników leżących na przekątnej głównej.

1.8. Wyznaczanie wartości własnych i wektorów własnych macierzy

1.8.1. Wprowadzenie

Zagadnienie znajdowania wartości własnych i wektorów własnych jest przedstawione następująco: dla danej macierzy kwadratowej A stopnia n wyznaczyć te parametry λ, przy których układ jednorodnych równań liniowych o n niewiadomych $\boldsymbol{x}^T = (x_1, x_2, \cdots, x_n)$

$$\boldsymbol{A}\boldsymbol{x} = \lambda \cdot \boldsymbol{x} \tag{8.1}$$

ma rozwiązanie niezerowe (symbol „T" oznacza transpozycję). Równanie (8.1) można zapisać w postaci

$$(\boldsymbol{A} - \lambda \cdot \boldsymbol{I})\boldsymbol{x} = \boldsymbol{o}, \tag{8.2}$$

gdzie $\boldsymbol{I}$ jest $n \times n$ wymiarową macierzą jednostkową.

Macierz współczynników w równaniu (8.2)

$$\boldsymbol{A} - \lambda \cdot \boldsymbol{I} = \begin{bmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{bmatrix} \tag{8.3}$$

nazywa się macierzą charakterystyczną macierzy $\boldsymbol{A}$, a wyznacznik

$$D(\lambda) = det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}) \tag{8.4}$$

nazywa się wyznacznikiem charakterystycznym. Przyrównując ten wyznacznik do zera, tzn. pisząc

$$D(\lambda) = 0 \tag{8.5}$$

otrzymujemy równanie charakterystyczne macierzy $\boldsymbol{A}$.

Pierwiastki równania charakterystycznego nazywane są wartościami własnymi macierzy $\boldsymbol{A}$. Aby obliczyć wartości własne, należy najpierw obliczyć wyznacznik (8.4). Rozwijając ten wyznacznik, otrzymujemy wielomian charakterystyczny stopnia n, stąd równanie charakterystyczne przyjmuje postać

$$\lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \cdots + p_{n-1}\lambda + p_n = 0. \tag{8.6}$$

Zbiór wszystkich wartości własnych macierzy $\boldsymbol{A}$: $\lambda_1, \lambda_2, \cdots, \lambda_n$ nazywa się widmem (lub spektrum) macierzy $\boldsymbol{A}$. Wartości własne mogą być rzeczywiste lub zespolone, pojedyncze lub wielokrotne.

Wektor niezerowy $x^{(i)} \in R^n$, spełniający związek

$$\boldsymbol{A}\boldsymbol{x}^{(i)} = \lambda_i \boldsymbol{x}^{(i)}, \tag{8.7}$$

gdzie λ_i jest wartością własną, nazywamy wektorem własnym macierzy $\boldsymbol{A}$ odpowiadającym wartości własnej λ_i. Każdy wektor własny macierzy $\boldsymbol{A}$ jest obliczany z dokładnością do stałego czynnika. Jeżeli $\boldsymbol{x}^{(i)}$ jest rozwiązaniem równania (8.7), to jest nim również wektor $k \cdot \boldsymbol{x}^{(i)}$ (gdzie k jest dowolną liczbą różną od zera). Zazwyczaj wygodnie jest obrać stały czynnik k tak, aby zachodził jeden z następujących przypadków:

powered by Misiol

- suma kwadratów składowych wektora $\boldsymbol{x}^{(i)}$ była równa jedności,
- pierwsza składowa wektora $\boldsymbol{x}^{(i)}$ była równa jedności,
- największa składowa wektora $\boldsymbol{x}^{(i)}$ była równa jedności,
- suma modułów składowych wektora $\boldsymbol{x}^{(i)}$ była równa jedności.

Wektor $\boldsymbol{x}^{(i)}$ spełniający jeden z wymienionych warunków nazywany jest wektorem własnym znormalizowanym (unormowanym).

Zagadnienie wyznaczania wektorów własnych odpowiadających obliczonym wcześniej wartościom własnym sprowadza się do rozwiązania układu jednorodnych równań liniowych. Bardziej złożone są metody obliczania wartości własnych. Można wyróżnić dwie grupy metod:

- sprowadzenie wyznacznika do postaci wielomianu stopnia n, a następnie rozwiązanie równania charakterystycznego (8.6) za pomocą jednej ze znanych metod,
- przybliżone wyznaczenie pierwiastków równania charakterystycznego metodą iteracji, bez uprzedniego rozwijania wyznacznika charakterystycznego.

Przedstawione w niniejszym punkcie metody odnoszą się do macierzy o współczynnikach rzeczywistych.

1.8.2. Metoda Kryłowa

W pierwszym etapie obliczeń metodą Kryłowa poszukuje się wartości współczynników $p_1, p_2, \cdots, p_n$ równania charakterystycznego (8.6). Umożliwiają one obliczenie wartości własnych. Po znalezieniu wartości własnych można w drugim etapie obliczeń wyznaczyć wektory własne, korzystając z odpowiednich wzorów rekurencyjnych.

Wyznaczanie współczynników równania charakterystycznego

Przy wyznaczaniu współczynników równania charakterystycznego skorzystamy z twierdzenia Cayleya-Hamiltona: każda macierz kwadratowa $\boldsymbol{A}$ jest pierwiastkiem swego równania charakterystycznego. Możemy więc zapisać, że

$$D(\boldsymbol{A}) = 0$$

lub

$$\boldsymbol{A}^n + p_1\boldsymbol{A}^{n-1} + p_2\boldsymbol{A}^{n-2} + \cdots + p_{n-1}\boldsymbol{A} + p_n = \boldsymbol{o}. \tag{8.8}$$

Obierzmy dowolny niezerowy wektor $\boldsymbol{y}_0 \in R^n$

$$\boldsymbol{y}_0^T = (y_{01}, y_{02}, \cdots, y_{0n}).$$

Mnożąc prawostronnie równanie (8.8) przez wektor y_0, otrzymujemy

$$\boldsymbol{A}^n y_0 + p_1\boldsymbol{A}^{n-1}y_0 + p_2\boldsymbol{A}^{n-2}y_0 + \cdots + p_{n-1}\boldsymbol{A}y_0 + p_n y_0 = \boldsymbol{o}. \tag{8.9}$$

Po wprowadzeniu oznaczeń:

$$\begin{aligned} \boldsymbol{y}_1 &= \boldsymbol{A}\boldsymbol{y}_0, \\ \boldsymbol{y}_2 &= \boldsymbol{A}^2\boldsymbol{y}_0, \\ &\vdots \\ \boldsymbol{y}_k &= \boldsymbol{A}^k\boldsymbol{y}_0, \\ &\vdots \\ \boldsymbol{y}_n &= \boldsymbol{A}^n\boldsymbol{y}_0, \end{aligned} \tag{8.10}$$

równanie (8.9) ma postać następującą

$$\boldsymbol{y}_n + p_1\boldsymbol{y}_{n-1} + p_2\boldsymbol{y}_{n-2} + \cdots + p_{n-1}\boldsymbol{y}_1 + p_n\boldsymbol{y}_0 = \boldsymbol{o}. \tag{8.11}$$

Równanie wektorowe (8.11) jest równoważne układowi n równań liniowych o niewiadomych $p_1, p_2, \cdots, p_n$

$$\begin{bmatrix} y_{n-1,1} & y_{n-2,1} & \cdots & y_{01} \\ y_{n-1,2} & y_{n-2,2} & \cdots & y_{02} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n-1,k} & y_{n-2,k} & \cdots & y_{0k} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n-1,n} & y_{n-2,n} & \cdots & y_{0n} \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = - \begin{bmatrix} y_{n1} \\ y_{n2} \\ \vdots \\ y_{nn} \end{bmatrix}. \tag{8.12}$$

Zauważmy, że aby obliczyć składowe y_{ij} wektorów $\boldsymbol{y}_i$ $(i = 1, 2, \ldots, n)$, wygodniej jest skorzystać nie bezpośrednio z zależności (8.10), lecz z zależności rekurencyjnych:

$$\begin{aligned} \boldsymbol{y}_1 &= \boldsymbol{A}\boldsymbol{y}_0, \\ \boldsymbol{y}_2 &= \boldsymbol{A}\boldsymbol{y}_1, \\ &\vdots \\ \boldsymbol{y}_k &= \boldsymbol{A}\boldsymbol{y}_{k-1}, \\ &\vdots \\ \boldsymbol{y}_n &= \boldsymbol{A}\boldsymbol{y}_{n-1}. \end{aligned} \tag{8.13}$$

Oczywiste jest, że:

$$y_{11} = \sum_{k=1}^{n} a_{1k} \cdot y_{0k},$$

$$y_{12} = \sum_{k=1}^{n} a_{2k} \cdot y_{0k} \quad \text{itd.}$$

Stąd ogólny wzór

$$y_{ij} = \sum_{k=1}^{n} a_{ij} \cdot y_{i-1,k} \quad i, j = 1, 2, \ldots, n. \tag{8.14}$$

powered by Misiel

Przykład 1

Znaleźć współczynniki równania charakterystycznego macierzy

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}.$$

Rozwiązanie

Przyjmując jako wektor początkowy np. $\boldsymbol{y}_0^T = (1, 0, 0)$, otrzymujemy

$\boldsymbol{y}_0$	$\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{y}_0$	$\boldsymbol{y}_2 = \boldsymbol{A}\boldsymbol{y}_1$	$\boldsymbol{y}_3 = \boldsymbol{A}\boldsymbol{y}_2$
1	1	3	19
0	1	6	45
0	1	10	81

Układ równań liniowych do wyznaczenia współczynników równania charakterystycznego, zgodnie z (8.12), ma postać

$$\begin{aligned} 3p_1 + p_2 + p_3 &= -19 \\ 6p_1 + p_2 &= -45 \\ 10p_1 + p_2 &= -81. \end{aligned}$$

Po rozwiązaniu powyższego układu otrzymujemy: $p_1 = -9$, $p_2 = 9$, $p_3 = -1$. Równanie charakterystyczne macierzy $\boldsymbol{A}$ ma więc postać

$$\lambda^3 - 9\lambda^2 + 9\lambda - 1 = 0.$$

Jeżeli wyznacznik główny układu równań liniowych (8.12) jest różny od zera (można wykazać, że wtedy wektory $\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n-1}$ są liniowo niezależne), to układ ma tylko jedno rozwiązanie, a pierwiastki układu są współczynnikami równania charakterystycznego macierzy $\boldsymbol{A}$. Problem komplikuje się, gdy układ (8.12) ma więcej niż jedno rozwiązanie. W tym przypadku należy wybrać inny wektor $\boldsymbol{y}_0$.

Wyznaczanie wektorów własnych

Załóżmy, że znane są współczynniki $p_1, p_2, \cdots, p_n$ równania charakterystycznego macierzy $\boldsymbol{A}$ oraz że pierwiastki tego równania $\lambda_1, \lambda_2, \cdots, \lambda_n$ są wszystkie różne, to znaczy są pojedyncze. Należy obliczyć wektory własne $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}$, odpowiadające kolejnym wartościom własnym.

Niech $\boldsymbol{y}_0, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_{n-1}$ będą liniowo niezależnymi wektorami, z których korzystaliśmy przy obliczaniu współczynników $p_1, p_2, \cdots, p_n$. Przedstawmy wektor $\boldsymbol{x}^{(i)}$ jako kombinację liniową wektorów $\boldsymbol{y}_0, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_n$

$$\boldsymbol{x}^{(i)} = g_1\boldsymbol{y}_0 + g_2\boldsymbol{y}_1 + g_3\boldsymbol{y}_2 + \cdots + g_n\boldsymbol{y}_{n-1}, \tag{8.15}$$

gdzie $g_1, g_2, \cdots, g_n$ są współczynnikami liczbowymi. Uwzględniając zależności (8.10) w (8.15), otrzymujemy

$$\boldsymbol{x}^{(i)} = g_1\boldsymbol{y}_0 + g_2\boldsymbol{A}\boldsymbol{y}_0 + g_3\boldsymbol{A}^2\boldsymbol{y}_0 + \cdots + g_n\boldsymbol{A}^{n-1}\boldsymbol{y}_0. \tag{8.16}$$

Następnie w równaniu (8.6) wektor $\boldsymbol{x}^{(i)}$ zastąpmy wyrażeniem podanym w (8.16)

$$\begin{aligned} g_1\boldsymbol{A}\boldsymbol{y}_0 + g_2\boldsymbol{A}^2\boldsymbol{y}_0 + g_3\boldsymbol{A}^3\boldsymbol{y}_0 + \cdots + g_{n-1}\boldsymbol{A}^{n-1}\boldsymbol{y}_0 + \boldsymbol{y}_n\boldsymbol{A}^n\boldsymbol{y}_0 = \\ = \lambda_i(g_1\boldsymbol{y}_0 + g_2\boldsymbol{A}\boldsymbol{y}_0 + g_3\boldsymbol{A}^2\boldsymbol{y}_0 + \cdots + g_n\boldsymbol{A}^{n-1}\boldsymbol{y}_0). \end{aligned} \tag{8.17}$$

Z równania (8.9) mamy

$$\boldsymbol{A}^n\boldsymbol{y}_0 = -(p_n\boldsymbol{y}_0 + p_{n-1}\boldsymbol{A}\boldsymbol{y}_0 + p_{n-2}\boldsymbol{A}^2\boldsymbol{y}_0 + \cdots + p_1\boldsymbol{A}^{n-1}\boldsymbol{y}_0). \tag{8.18}$$

Po podstawieniu do (8.17) otrzymujemy

$$\begin{aligned} g_1\boldsymbol{A}\boldsymbol{y}_0 + g_2\boldsymbol{A}^2\boldsymbol{y}_0 + g_3\boldsymbol{A}^3\boldsymbol{y}_0 + \cdots + g_{n-1}\boldsymbol{A}^{n-1}\boldsymbol{y}_0 + \\ -g_n(p_n\boldsymbol{y}_0 + p_{n-1}\boldsymbol{A}\boldsymbol{y}_0 + p_{n-2}\boldsymbol{A}^2\boldsymbol{y}_0 + \cdots + p_1\boldsymbol{A}^{n-1}\boldsymbol{y}_0) + \\ -\lambda_i(g_1\boldsymbol{y}_0 + g_2\boldsymbol{A}\boldsymbol{y}_0 + g_3\boldsymbol{A}^2\boldsymbol{y}_0 + \cdots + g_n\boldsymbol{A}^{n-1}\boldsymbol{y}_0) = 0. \end{aligned} \tag{8.19}$$

Stąd

$$\begin{aligned} (\lambda_i g_1 + p_n g_n)\boldsymbol{y}_0 + (\lambda_i g_2 + p_{n-1}g_n - g_1)\boldsymbol{A}\boldsymbol{y}_0 + \\ +(\lambda_i g_3 + p_{n-2}g_n - g_2)\boldsymbol{A}^2\boldsymbol{y}_0 + \cdots + (\lambda_i g_n + p_1 g_n - g_{n-1})\boldsymbol{A}^{n-1}\boldsymbol{y}_0 = 0. \end{aligned} \tag{8.20}$$

Ponieważ wektory $\boldsymbol{y}_0, \boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{y}_0, \cdots, \boldsymbol{y}_{n-1} = \boldsymbol{A}^{n-1}\boldsymbol{y}_0$ są liniowo niezależne, więc ze związku (8.20) otrzymujemy:

$$\begin{aligned} \lambda_i g_1 + p_n g_n &= 0 \\ \lambda_i g_2 + p_{n-1}g_n - g_1 &= 0 \\ \lambda_i g_3 + p_{n-2}g_n - g_2 &= 0 \\ &\vdots \\ \lambda_i g_{k+1} + p_{n-k}g_n - g_k &= 0 \\ &\vdots \\ \lambda_i g_n + p_1 g_n - g_{n-1} &= 0 \end{aligned} \tag{8.21}$$

lub

$$\begin{aligned} g_{n-1} &= \lambda_i g_n + p_1 g_n \\ g_{n-2} &= \lambda_i g_{n-1} + p_2 g_n \\ &\vdots \\ g_k &= \lambda_i g_{k+1} + p_{n-k} g_n \\ &\vdots \\ g_2 &= \lambda_i g_3 + p_{n-2} g_n \\ g_1 &= \lambda_i g_2 + p_{n-1} g_n. \end{aligned} \tag{8.22}$$

W równaniu (8.22) nie uwzględniliśmy pierwszego równania z układu (8.21), gdyż wynika ono z pozostałych równań oraz z (8.6). Ponieważ wektor własny jest wyznaczony z dokładnością do stałego współczynnika, to przyjmując w zależnościach (8.22) $y_n = k_i \neq 0$, otrzymujemy współczynniki $g_1, g_2, \cdots, g_{n-1}$ wyrażone za pomocą k_i. Po podstawieniu tych

powered by Misiol

współczynników do wzoru (8.15) otrzymujemy składowe wektora $\boldsymbol{x}^{(i)}$, odpowiadające wartości własnej λ_i.

Przykład 2

Wyznaczyć wektory własne macierzy $\boldsymbol{A}$ z przykładu 1.

Rozwiązanie

Pierwiastkami równania charakterystycznego macierzy $\boldsymbol{A}$ są:

$$\lambda_1 = 0.127017, \quad \lambda_2 = 1.00000, \quad \lambda_3 = 7.872983.$$

Przyjmując w równaniach (8.22) $g_3 = 1$ dla wartości własnej λ_1, współczynniki g_1 i g_2 obliczamy z równań

$$\begin{aligned} g_2 &= \lambda_1 + p_1 \\ g_1 &= \lambda_1 g_2 + p_2. \end{aligned}$$

Po podstawieniu $\lambda_1 = 0.127017$ oraz $p_1 = -9$, $p_2 = 9$ otrzymujemy $g_2 = -8.872983$, $g_1 = 7.872983$. Podstawiając te wartości do wzoru (8.15) oraz korzystając z obliczonych w przykładzie 1 wektorów $\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2$, znajdujemy składowe wektora $\boldsymbol{x}^{(1)}$. Wartości obliczonych składowych wektorów własnych macierzy podano w poniższej tablicy.

$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
2.000000	-4.000000	1.999997
-2.872983	-2.000000	4.872983
1.127017	2.000000	8.872983

Po normalizacji otrzymujemy

$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
-0.696140	1.000000	0.225403
1.000000	0.500000	0.549193
-0.392281	-0.500000	1.000000

1.8.3. Iteracyjne wyznaczanie wartości własnej o największym module

Załóżmy, że liczba rzeczywista λ_1 jest największą co do modułu wartością własną macierzy $\boldsymbol{A}$

$$|\lambda_1| > \lambda_2 \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \tag{8.23}$$

oraz że macierz ma n liniowo niezależnych wektorów własnych $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}$, tworzących bazę w przestrzeni R^n. Zatem dowolny wektor $\boldsymbol{y}$ można przedstawić jako kombinację liniową wektorów bazy

$$\boldsymbol{y} = c_1\boldsymbol{x}^{(1)} + c_2\boldsymbol{x}^{(2)} + \cdots + c_n\boldsymbol{x}^{(n)} = \sum_{i=1}^{n} c_i\boldsymbol{x}^{(i)}, \tag{8.24}$$

gdzie c_i są stałymi takimi, że

$$\sum_{i=1}^{n} c_i^2 > 0.$$

Mnożąc lewostronnie zależność (8.24) przez macierz $\boldsymbol{A}$, otrzymujemy

$$\boldsymbol{A} \cdot \boldsymbol{y} = \sum_{i=1}^{n} c_i \boldsymbol{A}\boldsymbol{x}^{(i)}. \tag{8.25}$$

Uwzględniając, że wektory $\boldsymbol{x}^{(i)}$ są wektorami własnymi macierzy $\boldsymbol{A}$, tj. $\boldsymbol{A}\boldsymbol{x}^{(i)} = \lambda_i \boldsymbol{x}^{(i)}$, napiszemy

$$\boldsymbol{y}^{(1)} = \boldsymbol{A} \cdot \boldsymbol{y} = \sum_{i=1}^{n} c_i \lambda_i \boldsymbol{x}^{(i)}. \tag{8.26}$$

Wektory $\boldsymbol{y}^{(1)}$ nazywamy pierwszą iteracją wektora $\boldsymbol{y}$. Dalsze iteracje wektora $\boldsymbol{y}$ otrzymujemy w wyniku kolejnych mnożeń lewostronnych związku (8.26) przez macierz $\boldsymbol{A}$:

$$\boldsymbol{y}^{(2)} = \boldsymbol{A}^2 \cdot \boldsymbol{y} = \sum_{i=1}^{n} c_i \lambda_i^2 \boldsymbol{x}^{(i)}, \tag{8.27}$$

$$\boldsymbol{y}^{(3)} = \boldsymbol{A}^3 \cdot \boldsymbol{y} = \sum_{i=1}^{n} c_i \lambda_i^3 \boldsymbol{x}^{(i)}, \tag{8.28}$$

$$\vdots$$

$$\boldsymbol{y}^{(k)} = \boldsymbol{A}^k \cdot \boldsymbol{y} = \sum_{i=1}^{n} c_i \lambda_i^k \boldsymbol{x}^{(i)}. \tag{8.29}$$

Ponieważ

$$\boldsymbol{y}^{(k)} = \begin{bmatrix} y_1^{(k)} \\ y_2^{(k)} \\ \vdots \\ y_j^{(k)} \\ \vdots \\ y_n^{(k)} \end{bmatrix}, \quad \boldsymbol{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_j^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}, \tag{8.30}$$

to po uwzględnieniu tej postaci wektorów w (8.29) otrzymujemy

$$\begin{bmatrix} y_1^{(k)} \\ y_2^{(k)} \\ \vdots \\ y_j^{(k)} \\ \vdots \\ y_n^{(k)} \end{bmatrix} = \sum_{i=1}^{n} c_i \lambda_i^k \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_j^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}. \tag{8.31}$$

Z zależności (8.31) dla j-tej składowej mamy

$$y_j^{(k)} = \sum_{i=1}^{n} c_i x_j^{(i)} \lambda_i^k. \tag{8.32}$$

powered by Misiol

Analogicznie otrzymujemy

$$y_j^{(k+1)} = \sum_{i=1}^{n} c_i x_j^{(i)} \lambda_i^{k+1}. \tag{8.33}$$

Dzieląc (8.33) przez (8.32) dostajemy

$$\frac{y_j^{(k+1)}}{y_j^{(k)}} = \frac{c_1 x_j^{(1)} \lambda_1^{k+1} + c_2 x_j^{(2)} \lambda_2^{k+1} + \cdots + c_n x_j^{(n)} \lambda_n^{k+1}}{c_1 x_j^{(1)} \lambda_1^{k} + c_2 x_j^{(2)} \lambda_2^{k} + \cdots + c_n x_j^{(n)} \lambda_n^{k}}. \tag{8.34}$$

Zakładając, że $c_i \neq 0$ oraz $x_j^{(1)} \neq 0$, przekształcimy zależność (8.34) w następujący sposób

$$\frac{y_j^{(k+1)}}{y_j^{(k)}} = \lambda_1 \frac{1 + \frac{c_2 x_j^{(2)}}{c_1 x_j^{(1)}} (\frac{\lambda_2}{\lambda_1})^{k+1} + \cdots + \frac{c_n x_j^{(n)}}{c_1 x_j^{(1)}} (\frac{\lambda_n}{\lambda_1})^{k+1}}{1 + \frac{c_2 x_j^{(2)}}{c_1 x_j^{(1)}} (\frac{\lambda_2}{\lambda_1})^{k} + \cdots + \frac{c_n x_j^{(n)}}{c_1 x_j^{(1)}} (\frac{\lambda_n}{\lambda_1})^{k}}. \tag{8.35}$$

Przechodząc we wzorze (8.35) do granicy przy $k \to \infty$ i uwzględniając, że na podstawie nierówności (8.23)

$$(\frac{\lambda_2}{\lambda_1})^k \to 0, \cdots, (\frac{\lambda_n}{\lambda_1})^k \to 0, \tag{8.36}$$

otrzymujemy

$$\lim_{k \to \infty} \frac{y_j^{(k+1)}}{y_j^{(k)}} = \lambda_1. \tag{8.37}$$

Widzimy więc, że jeżeli k jest dostatecznie duże, to

$$\lambda_1 \approx \frac{y_j^{(k+1)}}{y_j^{(k)}}, \quad j = 1, 2, \ldots, n \tag{8.38}$$

lub

$$\lambda_1 \approx \frac{(\boldsymbol{A}^{k+1}\boldsymbol{y})_j}{(\boldsymbol{A}^k \boldsymbol{y})_j}, \quad j = 1, 2, \ldots, n. \tag{8.39}$$

Ponieważ dla różnych j wyrażenia (8.38) lub (8.39) mogą się nie różnić między sobą, to za wartość λ_1 przyjmujemy średnią arytmetyczną wszystkich ilorazów typu (8.38) lub (8.39). W przypadku otrzymania „skaczących" wartości ilorazów należy przyjąć inny wektor początkowy $\boldsymbol{y}$. Aby uniknąć zbyt wielu liczb w kolejnych iteracjach, wygodnie jest pomnożyć wektor $\boldsymbol{y}^{(k)}$ przez pewien czynnik liczbowy λ_k, przyjmując w (8.38) $\tilde{\boldsymbol{y}}^{(k)} = \alpha_k \boldsymbol{A}^k \boldsymbol{y}$ oraz $\tilde{\boldsymbol{y}}^{(k+1)} = \boldsymbol{A}\tilde{\boldsymbol{y}}^{(k)}$.

W podobny sposób wyznaczymy iteracyjnie przybliżony wektor własny $\boldsymbol{x}^{(1)}$ macierzy $\boldsymbol{A}$, odpowiadający wartości własnej λ_1. Z zależności (8.29) mamy

$$\boldsymbol{A}^k \boldsymbol{y} = c_1 \lambda_1^k \boldsymbol{x}^{(1)} + \sum_{i=2}^{n} c_i \lambda_i^k \boldsymbol{x}^{(i)}. \tag{8.40}$$

Stąd

$$\boldsymbol{A}^k\boldsymbol{y} = c_1\lambda_1^k\left[\boldsymbol{x}^{(1)} + \sum_{i=2}^{n}\frac{c_i}{c_1}\left(\frac{\lambda_i}{\lambda_1}\right)^k\cdot\boldsymbol{x}^{(i)}\right]. \tag{8.41}$$

Ponieważ dalej prawdziwe są zależności (8.36), to

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{A}^k\boldsymbol{y} \approx c_1\lambda_1^k\boldsymbol{x}^{(1)}. \tag{8.42}$$

Oznacza to, że wektor $\boldsymbol{y}^{(k+1)}$ jest w przybliżeniu proporcjonalny do wektora własnego $\boldsymbol{x}^{(1)}$ odpowiadającego wartości własnej λ_1.

Przedstawiona metoda jest poprawna także w przypadku, gdy największa co do modułu wartość własna λ_1 macierzy $\boldsymbol{A}$ będzie r-krotnym pierwiastkiem równania charakterystycznego, tzn. $\lambda_1 = \lambda_2 = \cdots = \lambda_r$ oraz gdy $\lambda_1 = -\lambda_2$.

Przykład 3

Wyznaczyć największą co do modułu wartość własną macierzy

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}$$

oraz odpowiadający jej wektor własny.

Rozwiązanie

Przyjmujemy wektor początkowy $\boldsymbol{y}^{(0)^T} = (1,0,0)$ i korzystając ze wzoru (8.13) budujemy tablicę.

$y^{(0)}$	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	$y^{(4)}$	$y^{(5)}$	$y^{(6)}$	$y^{(7)}$	$y^{(8)}$
1	1	3	19	145	1137	8947	70435	554529
0	1	6	45	352	2769	21798	171613	1351104
0	1	10	81	640	5041	39690	213481	2460160

Przerywając obliczenia po 8-mej iteracji otrzymujemy:

$$\begin{aligned}
\frac{y_1^{(8)}}{y_1^{(7)}} &= \frac{554529}{70435} = 7.87291,\\
\frac{y_2^{(8)}}{y_2^{(7)}} &= \frac{1351104}{171613} = 7.87297,\\
\frac{y_3^{(8)}}{y_3^{(7)}} &= \frac{2460160}{312481} = 7.87299.
\end{aligned}$$

Stąd

$$\lambda_1 = \frac{1}{3}(7.8729 + 7.87297 + 7.87299) \approx 7.87296.$$

powered by Misiol

Wektor własny macierzy $\boldsymbol{A}$ odpowiadający wartości własnej λ_1 ma postać

$$\boldsymbol{x}^{(1)} \approx \boldsymbol{y}^{(8)} = \begin{bmatrix} 554529 \\ 1351104 \\ 2460160 \end{bmatrix} \quad \text{lub} \quad \boldsymbol{x}^{(1)} \approx \begin{bmatrix} 0.22540 \\ 0.54919 \\ 1.00000 \end{bmatrix}.$$

1.8.4. Metoda iteracyjna wyznaczania wartości własnych i wektorów własnych macierzy symetrycznej dodatnio określonej

Macierz symetryczna dodatnio określona posiada następujące właściwości:

- pierwiastki równania charakterystycznego są rzeczywiste i dodatnie,
- wektory własne spełniają warunki ortogonalności

$$\sum_{i=1}^{n} x_i^{(j)}\cdot x_i^{(k)} = 0 \quad \text{dla} \quad j \neq k. \tag{8.43}$$

Przedstawimy równanie (8.7) w postaci układu równań liniowych, na podstawie którego można wyznaczyć wektor własny $\boldsymbol{x}^{(1)}$

$$\begin{aligned}
(a_{11}-\lambda_1)x_1^{(1)} + a_{12}x_2^{(1)} + \cdots + a_{1n}x_n^{(1)} &= 0\\
a_{21}x_1^{(1)} + (a_{22}-\lambda_1)x_2^{(1)} + \cdots + a_{2n}x_n^{(1)} &= 0\\
&\vdots\\
a_{n1}x_1^{(1)} + a_{n2}x_2^{(1)} + \cdots + (a_{nn}-\lambda_1)x_n^{(1)} &= 0.
\end{aligned} \tag{8.44}$$

Przekształcając układ do postaci

$$\begin{aligned}
x_1^{(1)} &= \frac{1}{\lambda_1}(a_{11}x_1^{(1)} + a_{12}x_2^{(1)} + \cdots + a_{1n}x_n^{(1)})\\
x_2^{(1)} &= \frac{1}{\lambda_1}(a_{21}x_1^{(1)} + a_{22}x_2^{(1)} + \cdots + a_{2n}x_n^{(1)})\\
&\vdots\\
x_{n-1}^{(1)} &= \frac{1}{\lambda_1}(a_{n-1,1}x_1^{(1)} + a_{n-1,2}x_2^{(1)} + \cdots + a_{n-1,n}x_n^{(1)})\\
\lambda_1 &= \frac{1}{x_n^{(1)}}(a_{n1}x_1^{(1)} + a_{n2}x_2^{(1)} + \cdots + a_{nn}x_n^{(1)})
\end{aligned} \tag{8.45}$$

i przyjmując, że $x_n^{(1)} = 1$ (jest to dopuszczalne, ponieważ wektory własne są wyznaczane z dokładnością do czynnika stałego), otrzymujemy układ n równań z n niewiadomymi, który najwygodniej jest rozwiązać metodą iteracyjną (lub Seidela), przyjmując dogodne wartości początkowe $x_i^{(1,0)}$ oraz $\lambda_1^{(0)}$ dla układu:

$$\begin{aligned}
x_i^{(1,k+1)} &= \frac{1}{\lambda_1^{(k)}}\sum_{j=1}^{n-1} a_{ij}x_j^{(1,k)} + a_{in}, \qquad i = 1,2,\ldots,n-1,\\
\lambda_1^{(k+1)} &= \sum_{j=1}^{n-1} a_{nj}x_j^{(1,k+1)} + a_{nn}, \qquad k = 0,1,2,\ldots.
\end{aligned} \tag{8.46}$$

W ten sposób znajdujemy przybliżone wartości pierwiastka λ_1 równania charakterystycznego oraz odpowiadającego mu wektora własnego $\boldsymbol{x}^{(1)}$

$$\lambda_1 \approx \lambda_1^{(k)} \qquad \boldsymbol{x}^{(1)} \approx \begin{bmatrix} x_1^{(1,k)} \\ x_2^{(1,k)} \\ \vdots \\ x_{n-1}^{(1,k)} \\ 1 \end{bmatrix}. \tag{8.47}$$

Przy wyznaczaniu drugiej wartości własnej λ_2 i odpowiadającego jej wektora własnego $\boldsymbol{x}^{(2)}$ korzystamy z układu równań wynikającego z (8.7)

$$\sum_{j=1}^{n} a_{ij}x_j^{(2)} - \lambda_2 x_i^{(2)} = 0, \qquad i = 1, 2, \ldots, n \tag{8.48}$$

oraz z warunku ortogonalności

$$\sum_{j=1}^{n} x_j^{(1)} \cdot x_j^{(2)} = 0. \tag{8.49}$$

Wyznaczając z (8.49) jedną z niewiadomych $\boldsymbol{x}_j^{(2)}$, na przykład $\boldsymbol{x}_n^{(2)}$, i podstawiając do (8.48) otrzymujemy układ równań równoważny układowi (8.48):

$$\begin{aligned} x_i^2 &= \frac{1}{\lambda_2}\sum_{j=1}^{n-1} a_{ij}^{(2)} \cdot x_j^{(2)}, \qquad i = 1, 2, \ldots, n-2, \\ \lambda_2 &= \frac{1}{x_{n-1}^{(2)}}\sum_{j=1}^{n-1} a_{n-1,j}^{(2)} x_j^{(2)}. \end{aligned} \tag{8.50}$$

Układ (8.50) rozwiązujemy metodą iteracyjną, przyjmując $\boldsymbol{x}_{n-1}^{(2)} = 1$. Otrzymujemy drugą wartość własną λ_2 i wektor własny $\boldsymbol{x}^{(2)}$, przy czym n-tą składową wektora obliczamy z warunku ortogonalności.

Analogicznie wyznacza się pozostałe wartości własne λ_i macierzy $\boldsymbol{A}$ oraz odpowiadające im wektory własne $\boldsymbol{x}^{(i)}$, każdorazowo korzystając z równania (8.7) i warunków ortogonalności wektora $\boldsymbol{x}^{(i)}$ w stosunku do wektorów $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(i-1)}$, obliczonych wcześniej.

Przykład 4

Wyznaczyć wartości własne λ_i oraz odpowiadające im wektory własne $\boldsymbol{x}^{(i)}$ macierzy $\boldsymbol{A}$ metodą iteracyjną

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}.$$

powered by Misiol

Rozwiązanie

Macierz $\boldsymbol{A}$ jest macierzą symetryczną. Ponadto:

$$\Delta_1 = 1 > 0, \qquad \Delta_2 = \begin{vmatrix} 1 & 1 \\ 1 & 2 \end{vmatrix} = 1 > 0, \qquad \Delta_3 = det\boldsymbol{A} = 1 > 0,$$

a więc macierz $\boldsymbol{A}$ jest dodatnio określona. Układ (8.45) dla i-tej wartości własnej ma postać

$$\begin{aligned} x_1^{(1)} &= \frac{1}{\lambda_1}(x_1^{(1)} + x_2^{(1)} + x_3^{(1)}) \\ x_2^{(1)} &= \frac{1}{\lambda_1}(x_1^{(1)} + 2x_2^{(1)} + 3x_3^{(1)}) \\ \lambda_1 &= \frac{1}{x^{(i)3}}(x_1^{(1)} + 3x_2^{(1)} + 6x_3^{(1)}). \end{aligned} \tag{8.51}$$

Przyjmując $i = 1$ oraz $\boldsymbol{x}_3^{(1)} = 1$ otrzymujemy

$$\begin{aligned} x_1^{(1)} &= \frac{1}{\lambda_1}(x_1^{(1)} + x_2^{(1)} + 1) \\ x_2^{(1)} &= \frac{1}{\lambda_1}(x_1^{(1)} + 2x_2^{(1)} + 3) \\ \lambda_1 &= x_1^{(1)} + 3x_2^{(1)} + 6. \end{aligned}$$

Powyższy układ rozwiązujemy metodą iteracji. Przyjmując wartości początkowe $\boldsymbol{x}_1^{(1,0)} = 1$ oraz $\boldsymbol{x}_2^{(1,0)} = 1$ z ostatniego równania obliczamy $\lambda_1^{(0)} = 10$. Wyniki obliczeń przedstawia tablica.

k	$\boldsymbol{x}_1^{(1,k)}$	$\boldsymbol{x}_2^{(1,k)}$	$\boldsymbol{x}_3^{(1,k)}$	$\lambda_1^{(k)}$
0	1	1	1	10
1	0.3	0.6	1	8.1
2	0.234568	0.555556	1	7.901234
3	0.226562	0.550000	1	7.876562
4	0.225550	0.549296	1	7.873438
5	0.225422	0.549206	1	7.873041
6	0.225406	0.549195	1	7.872990
7	0.225404	0.549194	1	7.872984
8	0.225403	0.549193	1	7.872983

Przyjmujemy, że:

$$\lambda_1 \approx 7.872983, \qquad \boldsymbol{x}^{(1)} \approx \begin{bmatrix} 0.225403 \\ 0.549193 \\ 1 \end{bmatrix}.$$

Teraz przyjmiemy $i = 2$. Z warunku ortogonalności wektorów $\boldsymbol{x}^{(1)}$ oraz $\boldsymbol{x}^{(2)}$ mamy

$$0.225403x_1^{(2)} + 0.549193x_2^{(2)} + x_3^{(2)} = 0,$$

stąd

$$x_3^{(2)} = -0.225403x_1^{(2)} - 0.549193x_2^{(2)}.$$

Podstawiając to wyrażenie do układu (8.51) i przyjmując $x_2^{(2)} = 1$, otrzymujemy

$$x_1^{(2)} = \frac{1}{\lambda_2}(-0.774597x_1^{(2)} + 0.450807), \tag{8.52}$$

$$\lambda_2 = 0.323791x_1^{(2)} + 0.352421. \tag{8.53}$$

Obieramy $x_1^{(2,0)} = 1$, obliczamy $\lambda_x^{(0)} = 0.676212$ i rozwiązujemy układ (8.52) metodą iteracji. Wyniki obliczeń przedstawia tablica.

k	$x_1^{(2,k)}$	$x_2^{(2,k)}$	$\lambda_2^{(k)}$
0	1	1	0.676212
1	1.037566	1	0.688375
2	1.822410	1	0.942501
3	1.976062	1	0.992252
4	1.997031	1	0.999042
5	1.999618	1	0.999879
6	1.999946	1	0.999986
7	1.999988	1	0.999999

Przyjmujemy $\lambda_2 = 0.999999$, $x_1^{(2)} = 1.999988$, $x_2^{(2)} = 1$. Z warunku ortogonalności wektorów własnych otrzymujemy $x_3^{(2)} = -0.999996$; a po unormowaniu

$$x^{(2)} = \begin{bmatrix} 1 \\ 0.500003 \\ 0.500001 \end{bmatrix}.$$

Trzeci wektor własny $x^{(3)}$ wyznaczamy z dwóch warunków ortogonalności

$$0.225403x_1^{(3)} + 0.549193x_2^{(3)} + x_3^{(3)} = 0$$
$$x_1^{(3)} + 0.500003x_2^{(3)} + 0.500001x_2^{(3)} = 0.$$

Przyjmując $x_1^{(3)} = 1$, otrzymujemy $x_2^{(3)} = -1.436486$, $x_3^{(3)} = 0.563506$. Wartość własną λ_3 obliczamy z ostatniego równania układu (8.51) $\lambda_3 = 0.127012$. Wektor własny $x^{(3)}$ po unormowaniu jest następujący

$$x^{(3)} = \begin{bmatrix} -0.696143 \\ 1 \\ -0.392280 \end{bmatrix}.$$

powered by Misiol

1.9. Przybliżone rozwiązywanie układów równań nieliniowych

1.9.1. Wprowadzenie

Przedmiotem rozważań w niniejszym punkcie są niektóre metody przybliżonego rozwiązywania równań nieliniowych algebraicznych i przestępnych. W metodach tych korzysta się ze wzorów rekurencyjnych, określających sposób obliczania kolejnych wyrażeń ciągu przybliżeń miejsc zerowych funkcji $f(x)$, tzn.

$$f(x) = 0. \tag{9.1}$$

Zakłada się że równanie (9.1) ma wyłącznie pierwiastki odosobnione, tzn. zlokalizowane wewnątrz wąskich przedziałów (a, b), które zawierają jeden i tylko jeden pierwiastek równania (9.1). Do tej lokalizacji wykorzystuje się twierdzenie Bolzano-Cauchy'ego:

- Jeżeli funkcja $f(x)$ jest ciągła w przedziale domkniętym $[a, b]$ i $f(a) \cdot f(b) < 0$, to wewnątrz tego przedziału znajduje się co najmniej jeden pierwiastek równania $f(x) = 0$, tj. istnieje co najmniej jedna liczba $\xi \in (a, b)$, dla której $f(\xi) = 0$.
- Jeżeli w przedziale otwartym (a, b) istnieje pochodna $f^{(1)}(x)$ i nie zmienia znaku, to znaczy $sgn f^{(1)}(x) = const$ dla $x \in (a, b)$, to przedział (a, b) jest przedziałem izolacji pierwiastka ξ.

W praktyce pierwiastki równania lokalizuje się najczęściej za pomocą wykresu lub tabeli wartości funkcji $y = f(x)$, szacując w przybliżeniu położenie miejsc zerowych. Przedstawione tu algorytmy przybliżonego rozwiązania równań oparte są na założeniu, że dokładne rozwiązanie równania (9.1) zostało uprzednio zlokalizowane, tzn.

$$\xi \in (a, b). \tag{9.2}$$

Sformułowane wyżej twierdzenie Bolzano-Cauchy'ego jest podstawą przybliżonych metod rozwiązywania nieliniowych równań algebraicznych i przestępnych, między innymi metod połowienia przedziałów, siecznych i stycznych (Newtona).

W opisie każdej z metod oprócz jej algorytmu i interpretacji geometrycznej zamieszczono również oszacowanie dokładności końcowego przybliżenia pierwiastka oraz ocenę szybkości zbieżności procesu iteracyjnego metody. Ponadto działanie każdej z metod zostało zilustrowane przykładem.

1.9.2. Metoda połowienia przedziałów

Załóżmy, że dane jest równanie $f(x) = 0$, przy czym funkcja $f(x)$ jest ciągła i monotoniczna w przedziale izolacji $[a, b]$ pierwiastka oraz zachodzi nierówność $f(a) \cdot f(b) < 0$. Algorytm metody jest następujący: pierwsze przybliżenie pierwiastka otrzymuje się przez podział przedziału $[a, b]$ na połowy. Jeżeli $f(x_1) = 0$, gdzie $x_1 = \frac{1}{2}(a + b)$, to $\xi = x_1$ jest pierwiastkiem równania. Jeżeli natomiast $f(x_1) \neq 0$, to w celu wyznaczenia kolejnego przybliżenia wybiera się ten z przedziałów $[a, x_1]$ i $[x_1, b]$, dla którego $f(x_1) \cdot f(a) < 0$ lub

$f(x_1) \cdot f(b) < 0$. Oznaczmy granice nowego przedziału przez a_1, b_1. Kolejne przybliżenie pierwiastka otrzymuje się dzieląc kolejno zawężone przedziały na połowy i ponownie rozpatrując wartość i znaki funkcji $f(x)$ na ich końcach. Ponieważ kolejne przybliżenia x_n znajdują się w przedziałach izolacji

$$[a_n, b_n] \subset \cdots \subset [a_2, b_2] \subset [a_1, b_1], \tag{9.3}$$

więc metoda jest zbieżna. W każdym kroku przedział izolacji zmniejsza się o połowę. Wynika stąd proste oszacowanie dokładności n-tego przybliżenia:

$$f(a_n) \cdot f(b_n) < 0, \quad n = 1, 2, \ldots,$$

$$b_n - a_n = \frac{1}{2^n}(b - a), \quad n = 1, 2, \ldots.$$

Przyjmujemy

$$\xi = \frac{1}{2}(b_n + a_n), \quad \xi \in [a_n, b_n],$$

zatem

$$|x_n - \xi| \leq \frac{1}{2^{n+1}}(b - a).$$

Geometryczną interpretację metody połowienia przedziałów przedstawiono na rys. 1.10. Dużą zaletą omówionej metody jest jej prostota. Metoda jest zbieżna liniowo, a więc wolno. Dlatego należy jej używać jedynie do otrzymania dobrego przybliżenia początkowego pierwiastka.

Przykład

Znaleźć z dokładnością do $\varepsilon = 0.1$ pierwiastek równania

$$f(x) = \operatorname{tg}^3 x - 4 \operatorname{tg} x = 0 \quad \text{w przedziale} \quad [-0,5; 0,2].$$

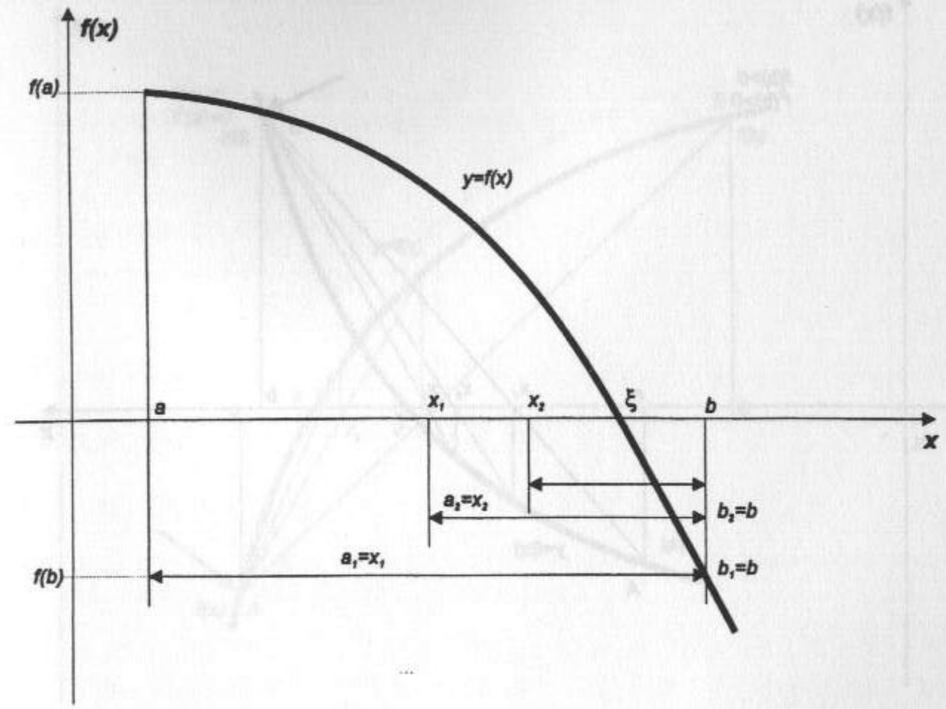
Rozwiązanie

$a = -0.5$
$b = 0.2$
$f(-0.5) = 2.0221679$
$f(0.2) = -0.80251051$
$f(x_1 = \frac{a+b}{2} = -0.15) = 0.60108866 \quad [-0.5; 0.2]$
$f(x_2 = 0.025) = -0.100052 \quad [-0.15; 0.2]$
$f(x_3 = -0.0625) = 0.25008093 \quad [-0.15; 0.025]$
$f(x_4 = -0.01875) = 0.075002196 \quad [-0.0625; 0.025]$

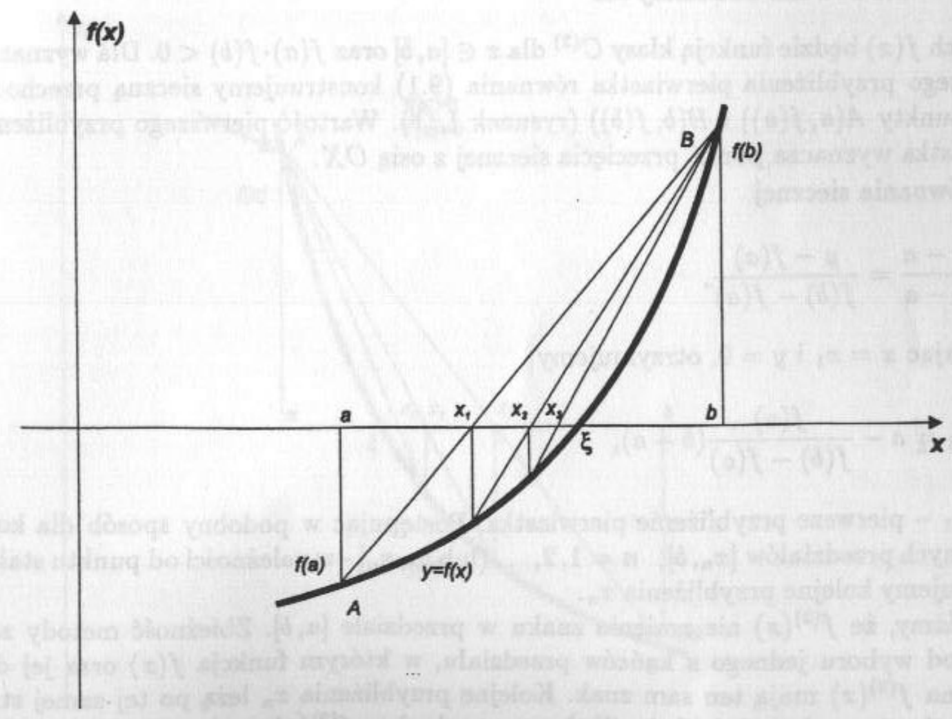
Szukanym przybliżeniem pierwiastka jest $x_4 = -0.001875$, gdyż

$$|f(x_4)| = |f(-0.01875)| = |0.0075002196| < \varepsilon = 0.1.$$

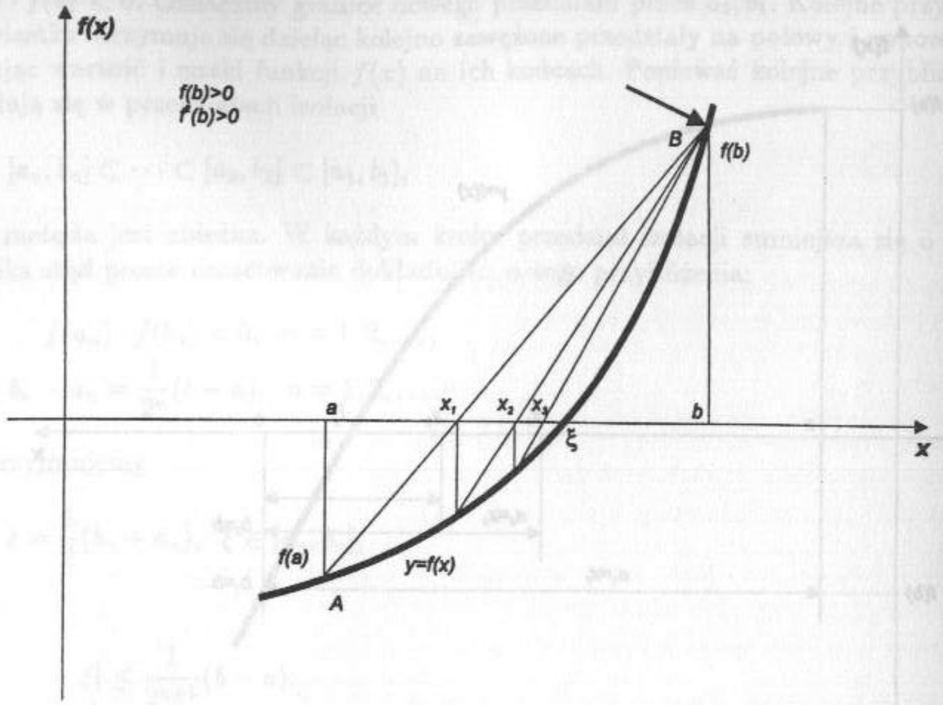
powered by Misiol



Rys. 1.10. Interpretacja geometryczna metody połowienia przedziału



Rys. 1.11. Interpretacja geometryczna metody siecznych



Rys. 1.12. Dobór punktu stałego w metodzie siecznych – przypadek 1

1.9.3. Metoda siecznych

Niech $f(x)$ będzie funkcją klasy $C^{(2)}$ dla $x \in [a,b]$ oraz $f(a)\cdot f(b) < 0$. Dla wyznaczenia pierwszego przybliżenia pierwiastka równania (9.1) konstruujemy sieczną przechodzącą przez punkty $A(a, f(a))$ i $B(b, f(b))$ (rysunek 1.11). Wartość pierwszego przybliżenia x_1 pierwiastka wyznacza punkt przecięcia siecznej z osią OX.

Z równania siecznej

$$\frac{x-a}{b-a} = \frac{y-f(a)}{f(b)-f(a)},$$

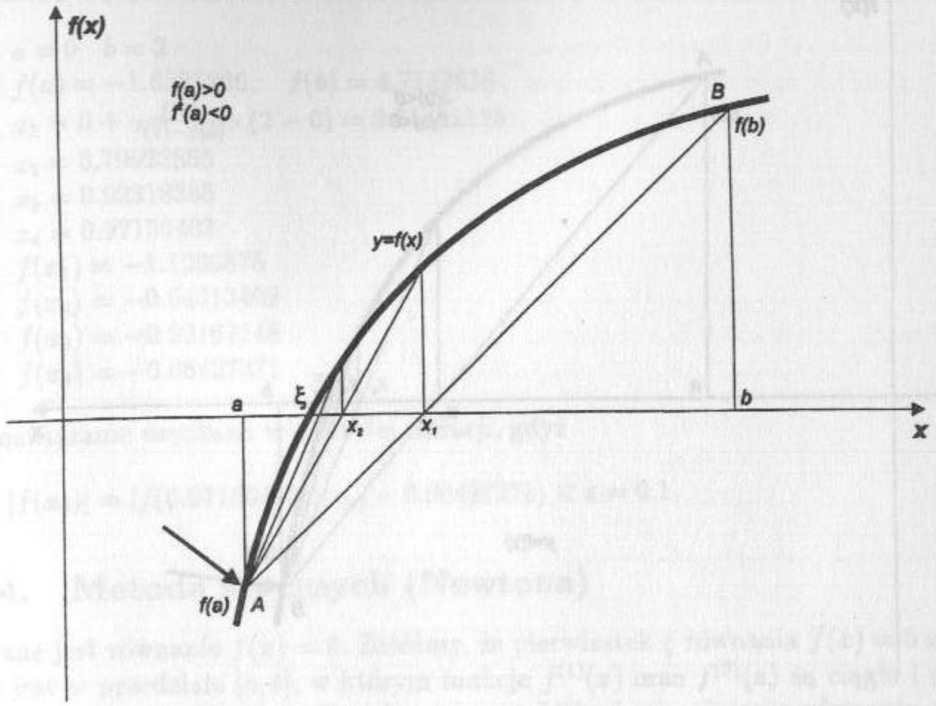
wstawiając $x = x_1$ i $y = 0$, otrzymujemy

$$x_1 = a - \frac{f(a)}{f(b)-f(a)}(b-a),$$

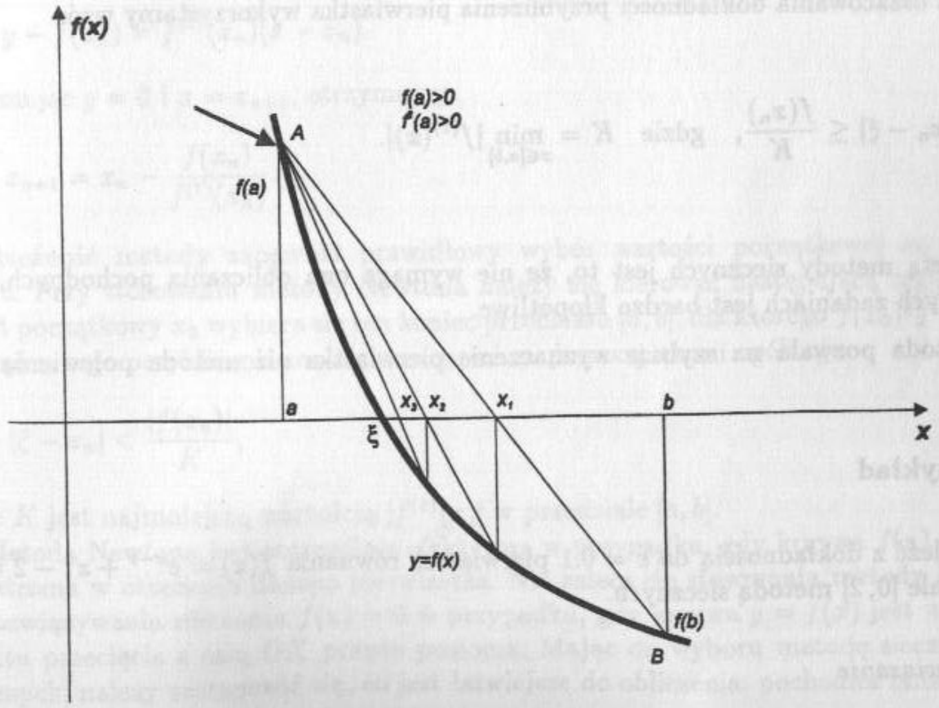
gdzie x_1 – pierwsze przybliżenie pierwiastka. Postępując w podobny sposób dla kolejno zawężanych przedziałów $[x_n, b]$, $n = 1, 2, \ldots$ (lub $[a, x_n]$ - w zależności od punktu stałego), otrzymujemy kolejne przybliżenia x_n.

Załóżmy, że $f^{(2)}(x)$ nie zmienia znaku w przedziale $[a,b]$. Zbieżność metody zależy wtedy od wyboru jednego z końców przedziału, w którym funkcja $f(x)$ oraz jej druga pochodna $f^{(2)}(x)$ mają ten sam znak. Kolejne przybliżenia x_n leżą po tej samej stronie pierwiastka ξ, po której funkcja $f(x)$ oraz pochodna $f^{(2)}(x)$ mają różne znaki. Dobór punktu stałego przedstawiono na rysunkach 1.12 – 1.15.

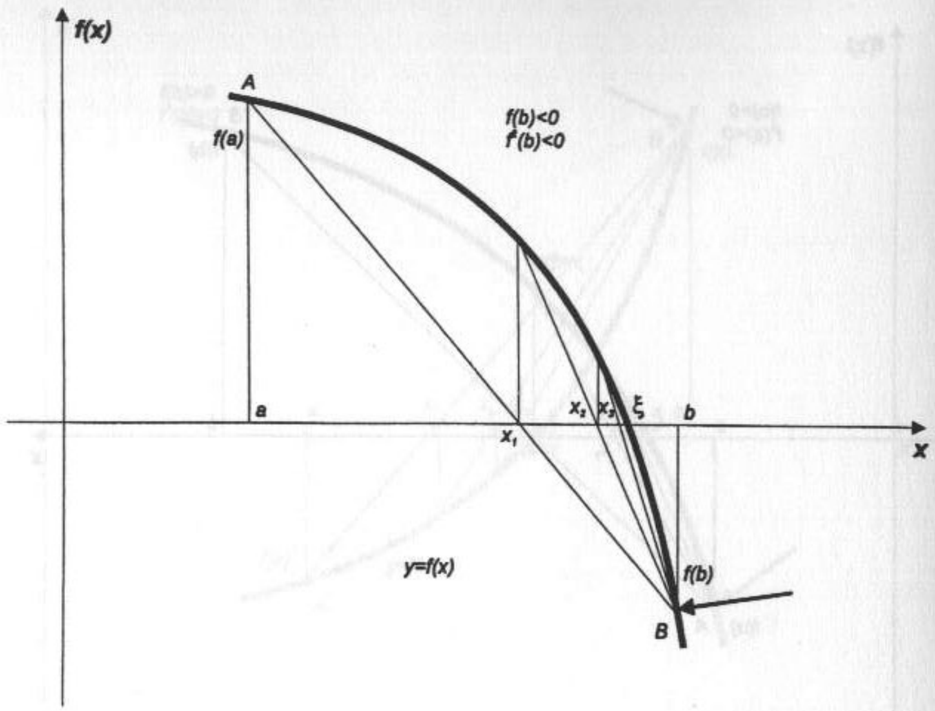
powered by Misiol



Rys. 1.13. Dobór punktu stałego w metodzie siecznych – przypadek 2



Rys. 1.14. Dobór punktu stałego w metodzie siecznych – przypadek 3



Rys. 1.15. Dobór punktu stałego w metodzie siecznych – przypadek 4

Dla oszacowania dokładności przybliżenia pierwiastka wykorzystamy wzór

$$|x_n - \xi| \leq \frac{f(x_n)}{K}, \quad \text{gdzie} \quad K = \min_{x\in[a,b]} |f^{(1)}(x)|.$$

Zaletą metody siecznych jest to, że nie wymaga ona obliczania pochodnych, co w niektórych zadaniach jest bardzo kłopotliwe.

Metoda pozwala na szybsze wyznaczenie pierwiastka niż metoda połowienia przedziału.

Przykład

Znaleźć z dokładnością do $\varepsilon = 0.1$ pierwiastek równania $f(x) = e^{x-1} + x^2 - 2 = 0$ w przedziale $[0, 2]$ metodą siecznych.

Rozwiązanie

Punktem stałym jest $b = 2$ ponieważ w tym punkcie funkcja $f(x)$ oraz jej druga

powered by Misiol

pochodna $f^{(2)}(x)$ mają ten sam znak.

$$\begin{aligned}
&a = 0 \quad b = 2\\
&f(a) = -1.6321206, \quad f(b) = 4.7182818\\
&x_1 = 0 + \frac{f(0)}{f(2)-f(0)} \cdot (2-0) = 0.51402115\\
&x_2 = 0.79922865\\
&x_3 = 0.92318385\\
&x_4 = 0.97150403\\
&f(x_1) = -1.1206875\\
&f(x_2) = -0.54313409\\
&f(x_3) = -0.22167148\\
&f(x_4) = -0.08427371
\end{aligned}$$

Rozwiązanie uzyskano w czwartej iteracji, gdyż

$$|f(x_4)| = |f(0.97150403)| = |-0.08427371| < \varepsilon = 0.1.$$

1.9.4. Metoda stycznych (Newtona)

Dane jest równanie $f(x) = 0$. Załóżmy, że pierwiastek ξ równania $f(x) = 0$ zlokalizowany jest w przedziale $[a, b]$, w którym funkcje $f^{(1)}(x)$ oraz $f^{(2)}(x)$ są ciągłe i nie zmieniają znaku. Omawiana metoda polega na przybliżeniu rozwiązania równania $f(x) = 0$ wyrazami ciągu miejsc zerowych stycznych do $f(x)$ (rysunek 1.16). Równanie stycznej w punkcie $B_n(x_n, f(x_n))$, $n = 0, 1, 2, \ldots$ ma postać

$$y - f(x_n) = f^{(1)}(x_n)(x - x_n).$$

Przyjmując $y = 0$ i $x = x_{n+1}$, otrzymamy

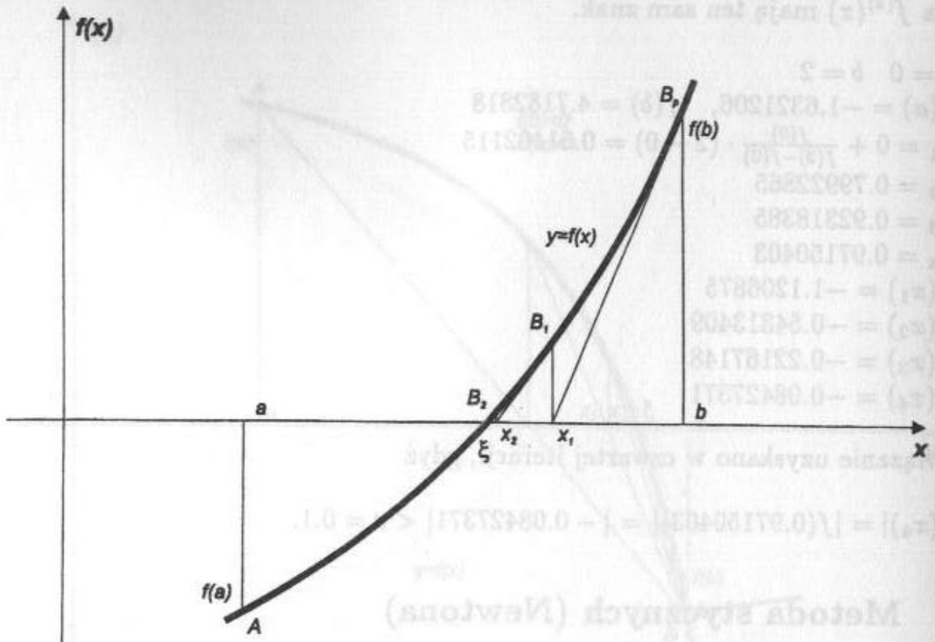
$$x_{n+1} = x_n - \frac{f(x_n)}{f^{(1)}(x_n)}.$$

Zbieżność metody zapewnia prawidłowy wybór wartości początkowej $x_0 = a$ lub $x_0 = b$. Przy stosowaniu metody Newtona należy się kierować następującą regułą: jako punkt początkowy x_0 wybiera się ten koniec przedziału $[a, b]$, dla którego $f(x_0) \cdot f^{(2)}(x_0) > 0$. Błąd n-tego przybliżenia można oszacować za pomocą wzoru ogólnego

$$|\xi - x_n| < \frac{|f(x_n)|}{K},$$

gdzie K jest najmniejszą wartością $|f^{(1)}(x)|$ w przedziale $[a, b]$.

Metoda Newtona jest szczególnie efektywna w przypadku, gdy krzywa $f(x)$ jest bardzo stroma w otoczeniu danego pierwiastka. Nie zaleca się stosowania metody Newtona do rozwiązywania równania $f(x) = 0$ w przypadku, gdy krzywa $y = f(x)$ jest w pobliżu punktu przecięcia z osią OX prawie pozioma. Mając do wyboru metodę siecznych lub stycznych, należy zastanowić się, co jest łatwiejsze do obliczenia: pochodna funkcji y czy też sama funkcja y.



Rys. 1.16. Interpretacja geometryczna metody stycznych (Newtona)

Przykład 1

Za pomocą metody stycznych znaleźć z dokładnością do $\varepsilon = 0.001$ pierwiastek równania $f(x) = 2^{x^2-5x+10} - 64 = 0$ w zadanym przedziale $[0; 1,25]$.

Rozwiązanie

Początkowym punktem jest punkt $a = 0$, w którym $f(a) \cdot f^{(2)}(a) > 0$.

$$x_1 = a - \frac{f(a)}{f^{(1)}(a)} = 0 - \frac{f(0)}{f^{(1)}(0)} = 0.27050532 \qquad f(x_1) = 357.86761$$
$$x_2 = 0.5446864 \qquad f(x_2) = 126.36209$$
$$x_3 = 0.78985117 \qquad f(x_3) = 38.154506$$
$$x_4 = 0.94739411 \qquad f(x_4) = 7.5353811$$
$$x_5 = 0.99633456 \qquad f(x_5) = 0.49028212$$
$$x_6 = 0.9999816 \qquad f(x_6) = 0.0024489462$$
$$x_7 = 1 \qquad f(x_7) = 0.0000000894$$

Szukanym pierwiastkiem jest $x_7 = 1$, gdyż

$$|f(x_7)| = |0.0000000894| < \varepsilon = 0.001.$$

powered by Misiol

Przykład 2

Porównać metody: połowienia przedziałów, siecznych i stycznych przy obliczaniu pierwiastków równania $f(x) = 10^x - 5x^4 - 20 = 0$ w zadanym przedziale $[0.3; 3.1]$ z dokładnością do $\varepsilon_1 = 0.1$, $\varepsilon_2 = 0.001$ *i* $\varepsilon_3 = 0.000001$ dla $n_{max} = 50$ iteracji.

Rozwiązanie

Stosując kolejne metody otrzymano następujące rozwiązania:

	$\varepsilon_1 = 0.1$		$\varepsilon_2 = 0.001$		$\varepsilon_3 = 0.000001$	
Metoda	n	x_n	n	x_n	n	x_n
połowienia przedziału	9	2.00078125	15	2.00001221	27	2.00000000
siecznych		rozbieżny		rozbieżny		rozbieżny
stycznych	6	2.00023519	7	2.00000011	8	2.00000000

Jak wynika z przedstawionych rozwiązań, najszybciej zbieżna jest metoda stycznych, a wynika to ze stromego kształtu funkcji $f(x)$ w otoczeniu pierwiastka. Metoda stycznych jest szczególnie efektywna przy rozwiązywaniu takich przykładów. Metoda połowienia przedziału w miarę zwiększania dokładności jest coraz wolniej zbieżna, co potwierdza jej przydatność do otrzymywania dobrego przybliżenia początkowego pierwiastka, lecz w większej liczbie kroków. Z kolei metoda siecznych okazała się dla omawianego przykładu rozbieżna, co spowodowane jest dużym nachyleniem krzywej. Nachylenie konstruowanych kolejno siecznych zmieniało się nieznacznie i do osiągnięcia żądanych dokładności metoda siecznych wymagała większej liczby iteracji niż $n_{max} = 50$.

1.9.5. Metoda Bernoulliego

Zastosowanie opisanych powyżej metod do obliczania pierwiastków rzeczywistych i zespolonych wielomianów wysokich stopni danych równaniem

$$W(x) = 0$$

jest możliwe po dokonaniu pewnych modyfikacji. Zrozumiałe jest więc poszukiwanie metod specjalnie dostosowanych do rozwiązywania tego typu zadań. Są to metody zawsze zbieżne ze względu na to, że nie ma na ogół dobrej informacji o lokalizacji wszystkich lub niektórych pierwiastków. Do metod tych należy metoda Bernoulliego.

Załóżmy, że dane jest równanie algebraiczne

$$W_n(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n = 0, \quad a_0 \neq 0, \tag{9.4}$$

którego pierwiastki $x_1, x_2, \cdots, x_n$ są parami różne. Znając współczynniki a_k, $k = 0, 1, \ldots, n$ można ułożyć tzw. równanie różnicowe

$$a_0y_{n+i} + a_1y_{n+i-1} + \cdots + a_ny_i = 0, \quad i = 0, 1, 2, \ldots, \tag{9.5}$$

które wiąże w sposób rekurencyjny kolejne $n+1$ wyrazów ciągu $y_0, y_1, \ldots, y_i$, spełniającego równanie różnicowe (9.5). Aby zbudować rozwiązanie y_i, wystarczy zadać n wartości

początkowych $y_0, y_1, \ldots, y_{n-1}$. Pozostałe wyrazy $y_{n+1}, y_{n+2}, \ldots$ można wyznaczyć w sposób rekurencyjny z równania (9.5). Można udowodnić, że jeśli równanie (9.4) ma jeden i tylko jeden największy co do modułu pierwiastek x_j, to stosunek dwóch kolejnych wyrazów y_{i+1} i y_i rozwiązania równania różnicowego (9.5) dąży na ogół do granicy równej x_j, tj.

$$\lim_{i\to\infty} \frac{y_{i+1}}{y_i} = x_j. \tag{9.6}$$

W celu określenia wartości przybliżonej największego co do modułu pierwiastka równania (9.4) - także wielokrotnego - można korzystać ze wzoru $x_j = \frac{y_{i+1}}{y_i}$, gdzie j jest dostatecznie duże. Obliczenia rozpoczyna się od dowolnych wartości początkowych dla $y_0, y_1, \ldots, y_{n-1}$ (zazwyczaj przyjmuje się $y_0 = y_1 = \cdots = y_{n-2} = 0$, $y_{n-1} = 1$), po czym korzystając ze wzoru

$$y_{n+1} = -\frac{1}{a_0}(a_n y_i + a_{n-1} y_{i+1} + \ldots + a_1 y_{n+i-1}), \quad i = 0, 1, 2, \ldots$$

oblicza się ciąg liczb $y_n, y_{n+1}, y_{n+2}, \ldots$ oraz ilorazy $y_n/y_{n-1}, y_{n+1}/y_n, y_{n+2}/y_{n+1}, \ldots$. Jeżeli stosunek y_{n+i}/y_{n+i-1} dąży przy rosnącym i do pewnej liczby, wówczas liczbę tę przyjmuje się za wartość największego co do modułu pierwiastka równania (9.4).

Metodę Bernoulliego można stosować przy szukaniu pierwiastka o największym lub najmniejszym module (wtedy trzeba rozpatrzyć wielomian $(W(\frac{1}{z})$, gdzie $x = \frac{1}{z})$. Po obliczeniu pierwiastka i wyrugowaniu go z równania (9.1) za pomocą dzielenia przez dwumian metodę Bernoulliego można zastosować do obliczenia pierwiastka o następnym co do wartości module. W przypadku pierwiastków zespolonych równania (9.4) stosunek y_{i+1}/y_i oscyluje, nie dążąc do granicy i wzór (9.6) nie jest wówczas prawdziwy. Wzór (9.6) nie jest również prawdziwy w przypadku, gdy równanie (9.4) ma kilka pierwiastków (w tym zespolonych) o tym samym największym module. Metoda Bernoulliego jest na ogół wolno zbieżna i należy jej używać razem z inną szybciej zbieżną metodą.

Przykład 1

Metodą Bernoulliego znaleźć największy co do modułu pierwiastek x_1 równania

$$x^5 + 5x^4 - 5 = 0.$$

Rozwiązanie

Równanie różnicowe odpowiadające danemu równaniu algebraicznemu ma postać

$$y_{i+5} = 5(y_i - y_{i+4}) \quad i = 0, 1, 2, \ldots. \tag{9.7}$$

Przyjmijmy $y_0 = 0$, $y_1 = 0$, $y_2 = 0$, $y_3 = 0$, $y_4 = 1$. Obliczamy kolejne wartości y_i przy

powered by Misiol

$i \geq 5$ wg wzoru rekurencyjnego (9.7). Znalezione wartości podane są w tabelce.

i	y_i	y_i/y_{i-1}
1	−5	−5
2	25	−5
3	−125	−5
4	625	−5
5	−3120	−4.992
6	15575	−4.992
7	−77750	−4.928
8	388125	−4.99196
9	−1937500	−4.991948

$$x_1 = \frac{y_9}{y_8} = \frac{-1937500}{388125} = -4.991948$$

Przykład 2

Metodą Bernoulliego znaleźć największe co do modułu pierwiastki wielomianów

$$x^3 - 11x^2 + 8x + 20 = 0 \tag{9.8}$$

oraz

$$x^3 - x^2 - 4x + 4 = 0. \tag{9.9}$$

Ocenić zbieżność metody dla każdego przykładu.

Rozwiązanie

Pierwiastkami wielomianu (9.8) są liczby: +10, −1, +2. W zamieszczonej poniżej tabelce podano kolejne przybliżenia największego co do modułu pierwiastka wielomianu obliczanego metodą Bernoulliego.

i	x_i
1	11.0000000
2	10.2727273
3	10.0442478
4	10.0096916
5	10.0018484
6	10.0003784
7	10.0000748
8	10.0000150
9	10.0000030
10	10.0000006
11	10.0000001
12	10.0000000
13	10.0000000

W przykładzie uzyskano szybką zbieżność (w dwunastej iteracji), wynikającą z tego, że moduł największego pierwiastka był dużo większy od pozostałych pierwiastków.

Rozwiązaniami równania (9.9) są pierwiastki: +2, −2, 1. Stosując metodę Bernoulliego otrzymano następujące wyniki w kolejnych iteracjach:

i	x_i
1	1.00000000
11	1.00000000
21	1.00000000
31	1.00000000
41	1.00000000
51	1.00000000

W obliczeniach uzyskano rozwiązanie rozbieżne, co jest uzasadnione tym, że metoda Bernoulliego nie jest prawdziwa dla wielomianów posiadających kilka pierwiastków o tym samym największym module.

1.10. Przybliżone rozwiązywanie równań różniczkowych zwyczajnych

1.10.1. Wprowadzenie

Równania różniczkowe są szeroko stosowane w różnych dziedzinach nauki i techniki. Ze względu na ich rozpowszechnienie istnieje dużo metod rozwiązywania równań różniczkowych.

Metody numeryczne rozwiązywania równań różniczkowych zwyczajnych można podzielić na kilka grup, np. jednokrokowe, wielokrokowe, różnicowe, typu Rungego-Kutty, interpolacyjne, ekstrapolacyjne itd.
W niniejszym rozdziale zostaną przedstawione następujące metody przybliżonego rozwiązywania równań różniczkowych: metoda Eulera, metoda Rungego-Kutty czwartego rzędu, metoda szeregów Taylora oraz metoda Picarda.

1.10.2. Sformułowanie zadania przybliżonego rozwiązywania równań różniczkowych zwyczajnych

Najprostszym równaniem różniczkowym zwyczajnym jest równanie różniczkowe pierwszego rzędu

$$y^{(1)}(x) = f(x, y(x)) \tag{10.1}$$

z warunkiem początkowym $y(x_0) = y_0$.
Zakładamy, że

- funkcja $f(x, y(x))$ jest określona i ciągła w obszarze $x_0 \le x \le b$, $-\infty < y < \infty$, gdzie x_0 i b są skończone;

powered by Mi Siol

- istnieje stała $L > 0$ taka, że dla każdego $x \in [x_0, b]$ i dowolnych liczb $y, \tilde{y}$ zachodzi nierówność

$$|f(x, y) - f(x, \tilde{y}| \le L|y - \tilde{y}|. \tag{10.2}$$

Warunek (10.2) nazywamy warunkiem Lipschitza.

Przy powyższych założeniach można udowodnić, że w przedziale $[x_0, b]$ istnieje dokładnie jedna funkcja ciągła i różniczkowalna $y(x)$, spełniająca równanie (10.1) z danym warunkiem początkowym.

Celem przybliżonego rozwiązywania równania (10.1) jest obliczenie wartości funkcji $y(x)$ dla ciągu wartości $x = x_i$, $i = 1, 2, \ldots$, czyli znajdowanie punktów na krzywej całkowej $y(x)$.

Przy konstrukcji metod przybliżonego rozwiązywania numerycznego równań różniczkowych zwyczajnych ważną rolę odgrywają następujące rozważania:

1. Jak duży błąd powstaje w każdym etapie obliczeń (błąd metody i zaokrąglenia) i jak ten błąd wpływa na wyniki następnych etapów? Jest to przypadek, w którym obserwujemy przenoszenie się błędu powstającego w jednym etapie obliczeń na wyniki obliczeń wykonywanych w dalszych etapach. To zjawisko, które występuje w wielu dziedzinach metod numerycznych, rozważa się na ogół pod hasłem „stabilność". Metodami stabilnymi nazywamy takie, w których błąd powstający w jednym etapie obliczeń nie ma tendencji wzrostu w dalszych etapach.

2. Z powstawaniem i przenoszeniem się błędu związane jest zagadnienie możliwości oszacowania błędu w ustalonym miejscu obliczeń jako funkcji obliczonych wyników.

3. Jak rozpocząć obliczenia? W równaniu (10.1) jest zawarty warunek początkowy $x = x_0$. Wiele metod różnicowych wymaga jednak wartości y w kilku punktach do obliczenia wartości w następnym punkcie. Dlatego często są potrzebne dodatkowe metody w celu rozpoczęcia obliczeń.

4. Szybkość metody. Przy rozwiązywaniu dużych układów równań różniczkowych czas potrzebny na wykonanie obliczeń (nawet w najszybszych maszynach cyfrowych) może być znaczny. Od szybkości obliczeń będzie zależeć ocena rozpatrywanych metod, gdyż żadna sensowna dyskusja nie może pominąć takich względów praktycznych.

1.10.3. Metoda Eulera

Podstawowe zależności metody

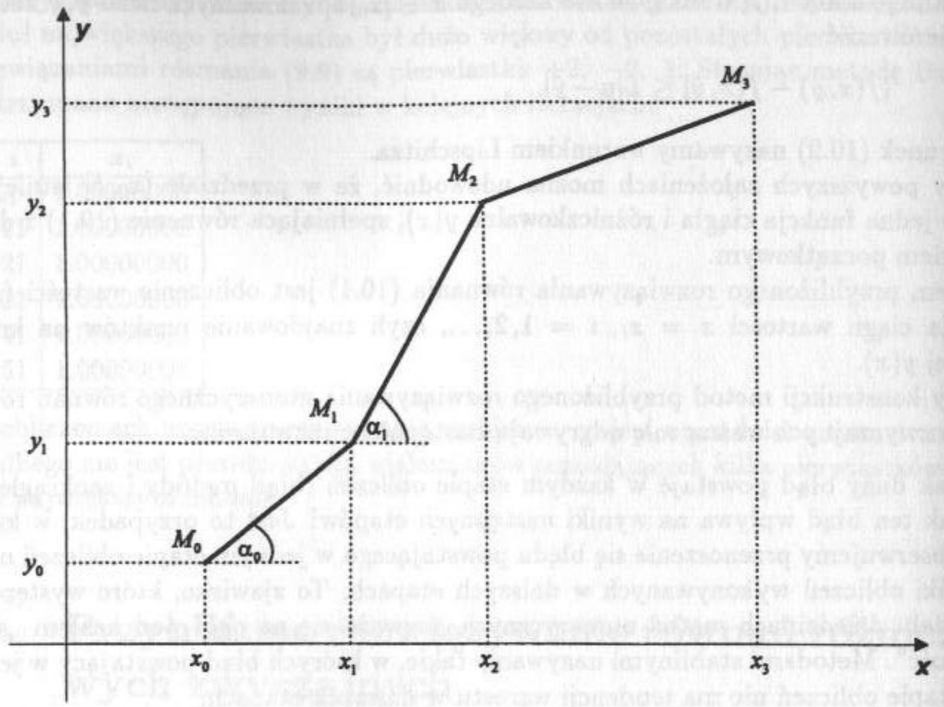
Niech będzie dane równanie różniczkowe zwyczajne wyrażone zależnością (10.1), z warunkiem początkowym $y(x_0) = y_0$.

Metoda Eulera polega na zastąpieniu krzywej całkowej $y = y(x)$ przechodzącej przez punkt $M_0(x_0, y_0)$, odpowiadający warunkom początkowym, łamaną $M_0M_1M_2\ldots$ o wierzchołkach $M_i(x_i, y_i)$, $i = 0, 1, 2, \ldots$, składającą się z odcinków prostych (rysunek 1.17).

Punkt rozpoczęcia i-tego odcinka prostej określony jest punktem osiągnięcia przez $(i-1)$-szy odcinek prostej odciętej

$$x_i = x_0 + ih, \tag{10.3}$$

gdzie h – stały krok obliczeń.



Rys. 1.17. Interpretacja graficzna metody Eulera

Punkt M_0 rozpoczęcia pierwszego odcinka prostej jest określony warunkiem początkowym (x_0, y_0). Odcinki M_iM_{i+1}, $i = 0, 1, 2, \ldots$ łamanej mają współczynnik kątowy wyrażony następującą zależnością

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i) = y_i^{(1)} = y^{(1)}(x_i). \tag{10.4}$$

Z powyższej zależności wynika wzór na tangens kąta nachylenia i-tego odcinka prostej. Zatem:

$$tg\alpha_0 = f(x_0, y_0),$$
$$tg\alpha_1 = f(x_1, y_1).$$

Ogólnie

$$tg\alpha_i = f(x_i, y_i).$$

Ze wzoru (10.4) wynika również, że wartości y_i można znaleźć z następujących wzorów:

$$y_{i+1} = y_i + \Delta y_i, \tag{10.5}$$
$$\Delta y_i = h \cdot f(x_i, y_i), \quad i = 0, 1, 2, \ldots. \tag{10.6}$$

powered by Misiol

Przykład

Rozwiązać równanie różniczkowe

$$y^{(1)}(x) = 2xy(x)$$

z warunkiem początkowym $y(0) = 1$; długość kroku $h = 0.1$.

Rozwiązanie

i	x	y	$f(x,y) = 2xy$	$\Delta y = h \cdot f(x,y)$
0	0	1	0	0
1	0.1	1	0.2	0.02
2	0.2	1.02	0.408	0.0408
3	0.3	1.0608	0.63648	0.06365
4	0.4	1.12445	0.89956	0.08996
5	0.5	1.21441	1.121441	1.12144
6	0.6	1.33585		

Po sześciu krokach uzyskano wynik 1.33585.

Ocena metody Eulera

Metoda Eulera jest najprostszą metodą numeryczną rozwiązywania równań różniczkowych, ma jednak następujące wady:

- małą dokładność,
- kolejne nakładanie się błędów.

W literaturze można znaleźć dowody, że jeśli prawa strona $f(x, y(x))$ równania (10.1) jest ciągła, to ciąg łamanych Eulera jest jednostajnie zbieżny do szukanej krzywej całkowej $y = y(x)$ w dostatecznie małym przedziale $[x_0, x_0 + h]$ dla $h \to 0$. W praktyce jednak zmniejszenie kroku całkowania h powoduje zwiększenie liczby kroków, a tym samym narastanie błędów zaokrągleń (rysunek 1.18).

1.10.4. Metoda Rungego-Kutty

Podstawowe wzory

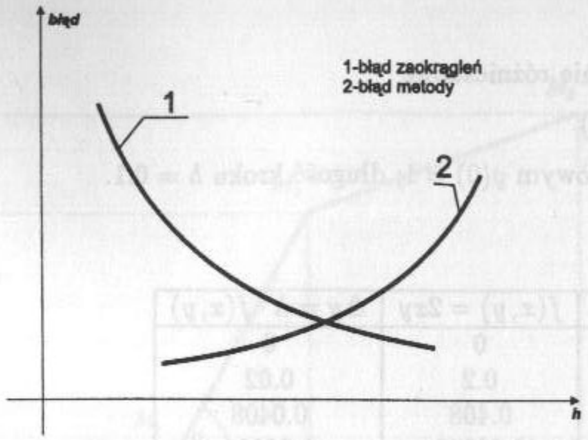
Metoda Rungego-Kutty jest metodą jednokrokową charakteryzującą się tym, że przy obliczaniu wartości funkcji w kolejnym punkcie $x_i = x_0 + ih$ bierze się pod uwagę również punkty wewnątrz i-tego kroku. W zależności od tego, ile tych punktów bierze się pod uwagę, mówi się o rzędzie metody.

W niniejszym rozdziale wykorzystywana będzie metoda Rungego-Kutty czwartego rzędu.

Niech dane będzie równanie różniczkowe pierwszego rzędu

$$y^{(1)}(x) = f(x, y(x)) \tag{10.7}$$

z warunkiem początkowym $y(x_0) = y_0$.
Wybierzmy krok całkowania h i wprowadźmy następujące oznaczenia $x_i = x_0 + ih$, $y_i = y(x_i)$, $i = 0, 1, 2, \ldots$).



Rys. 1.18. Błąd metody i zaokrąglenia w metodzie Eulera w zależności od kroku

Zgodnie z metodą Rungego-Kutty czwartego rzędu kolejne wartości y_i szukanej krzywej całkowej $y = y(x)$ znajdujemy ze wzoru

$$y_{i+1} = y_i + \Delta y_i, \tag{10.8}$$

gdzie

$$\Delta y_i = \frac{1}{6}(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}), \qquad i = 0, 1, 2, \ldots \tag{10.9}$$

oraz:

$$k_1^{(i)} = h \cdot f(x_i, y_i), \tag{10.10}$$

$$k_2^{(i)} = h \cdot f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1^{(i)}), \tag{10.11}$$

$$k_3^{(i)} = h \cdot f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2^{(i)}), \tag{10.12}$$

$$k_4^{(i)} = h \cdot f(x_i + h, y_i + k_3^{(i)}). \tag{10.13}$$

Wartości $k_n^{(i)}$, $n = 1, 2, 3, 4)$, są różne w każdym kroku i. W celu ułatwienia korzystania z metody Rungego-Kutty można podać następujący schemat rozwiązywania równania różniczkowego:

i	x	y	$k = h \cdot f(x, y)$	Δy
0	x_0	y_0	$k_1^{(0)}$	$k_1^{(0)}$
	$x_0 + \frac{1}{2}h$	$y_0 + \frac{1}{2}k_1^{(0)}$	$k_2^{(0)}$	$2k_2^{(0)}$
	$x_0 + \frac{1}{2}h$	$y_0 + \frac{1}{2}k_2^{(0)}$	$k_3^{(0)}$	$2k_3^{(0)}$
	$x_0 + h$	$y_0 + k_3^{(0)}$	$k_4^{(0)}$	$k_4^{(0)}$
				$\frac{1}{6}\sum = \Delta y_0$
1	itd. jak wyżej ze zmianą indeksu z „0" na „1"			

powered by Misiol

Przykład

Rozwiązać równanie

$$y^{(1)}(x) = 2xy(x)$$

z warunkiem początkowym $y(0) = 1$; długość kroku $h = 0.1$.

Rozwiązanie

i	x	y	$k = h \cdot f(x, y)$	Δy
0	0	1	0	0
	0.05	1	0.01	0.02
	0.05	1.005	0.01005	0.0201
	0.1	1.01005	0.020201	0.020201
				$\frac{1}{6}\sum = \Delta y = 0.01005$
1	0.1	1.1005	0.02201	0.02201
	0.15	1.01101	0.03033	0.06066
	0.15	1.01517	0.03046	0.06091
	0.2	1.01523	0.04061	0.04061
				$\frac{1}{6}\sum = \Delta y = 0.02737$
2 itd.	0.2	1.04058		

Równanie to można łatwo rozwiązać analitycznie. Przy podanych warunkach początkowych spełnia je krzywa całkowa o postaci

$$y(x) = e^{x^2}.$$

Użycie omawianych powyżej metod przybliżonych do rozwiązywania tego równania pozwala na proste ich porównanie (zgodnie z rozważaniami zawartymi w punkcie 10.2). Ponadto znajomość rozwiązania dokładnego daje możliwość oceny metody Eulera i Rungego-Kutty poprzez porównanie wyników tych rozwiązań na jednym wykresie.

Ocena metody Rungego-Kutty

Metoda Rungego-Kutty jest bardzo dokładna i mimo swej uciążliwości jest jedną z najczęściej stosowanych metod numerycznego rozwiązywania równań różniczkowych.

Ważną zaletą tej metody jest możliwość stosowania zmiennego kroku całkowania. Znając $y_i = y(x_i)$, obliczamy według wzorów Rungego-Kutty wartość $\overline{y}_{i+1} = y(x_i + h)$. Następnie powtarzamy obliczenia, ale z krokiem dwa razy mniejszym, tak że po dwukrotnym zastosowaniu wzorów Rungego-Kutty obliczamy ponownie wartość $\overline{\overline{y}}_{i+1} = y(x_i + h)$. Stąd błąd $\delta = (\overline{\overline{y}}_{i+1} - \overline{y}_{i+1})$. Obliczanie błędu umożliwia automatyczny wybór kroku h. W tym celu ustala się dwie liczby – ϵ_1 oraz ϵ_2 ($\epsilon_1 < \epsilon_2$), charakteryzujące dokładność obliczeń. Jeśli po przejściu z punktu x_i do x_{i+1} okaże się, że:

- $\delta < \epsilon_1$, to zwiększamy krok h dwukrotnie (krok $2h$),
- $\epsilon_1 < \delta < \epsilon_2$, to nie zmieniamy kroku (krok h),
- $\epsilon_2 < \delta$, to wracamy do punktu x_i i powtarzamy obliczenia z krokiem dwukrotnie mniejszym (krok $0.5h$).

Metoda Rungego-Kutty jest samostartowna. Ta cecha powoduje, że metoda ta jest idealnym uzupełnieniem innych metod przybliżonych przy obliczaniu wstępnych wartości. W takim przypadku trzeba zwracać uwagę przede wszystkim na błąd metody, a mniej na stabilność. Dlatego używa się tu metod Rungego-Kutty z najlepszym oszacowaniem błędu.

Wadami omawianej metody są: stosunkowo duży czas obliczeń oraz trudności w oszacowaniu błędu.

1.10.5. Rozwiązywanie układów równań różniczkowych

W zastosowaniach praktycznych często spotyka się układy równań różniczkowych zwyczajnych. Układ równań można zapisać w postaci wektorowej

$$\frac{d\boldsymbol{y}(\boldsymbol{x})}{d\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})), \tag{10.14}$$

gdzie:

$$\boldsymbol{y} = [y_1(\boldsymbol{x}), y_2(\boldsymbol{x}), \cdots, y_n(\boldsymbol{x})]^T \in R^n, \tag{10.15}$$

$$\boldsymbol{f} = [f_1, f_2, \cdots, f_n]^T \in R^n. \tag{10.16}$$

Przez rozwiązanie układu (10.14) rozumiemy dowolny układ funkcji $\boldsymbol{y}(\boldsymbol{x}) = [y_1(\boldsymbol{x}), y_2(\boldsymbol{x}), \cdots, y_n(\boldsymbol{x})]^T$, który po podstawieniu do równania (10.14) spełnia je tożsamościowo. Geometrycznie każde rozwiązanie jest krzywą całkową w przestrzeni $R \times R^n$.

Ponieważ układ równań różniczkowych ma nieskończenie wiele rozwiązań, to dla wydzielenia jednego konkretnego rozwiązania $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$ muszą być podane dodatkowe warunki. W najprostszym przypadku podaje się warunki początkowe w postaci

$$\boldsymbol{y}(\boldsymbol{x}_0) = \boldsymbol{y}_0 \in R^n. \tag{10.17}$$

Układ równań (10.14) z warunkami początkowymi (10.17) rozwiązuje się za pomocą wzorów (10.8) - (10.13) zapisanych wektorowo, np.

$$\boldsymbol{k}_1^{(i)} = h \cdot \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{y}_i) \in R^n. \tag{10.18}$$

Pozostałe wzory zapisuje się analogicznie.

1.10.6. Rozwiązywanie równań różniczkowych wyższych rzędów

Metodą Rungego-Kutty można rozwiązywać również równania różniczkowe zwyczajne wyższych rzędów. Równanie takie sprowadza się poprzez odpowiednie podstawienie do układu równań różniczkowych pierwszego rzędu i postępuje się tak jak w punkcie 10.5.

Rozważmy np. równanie różniczkowe rzędu drugiego. Ogólna postać takiego równania jest następująca

$$y^{(2)}(x) = f(x, y(x), y^{(1)}(x)) \tag{10.19}$$

powered by Misiol

z warunkami początkowymi: $y(x_0) = y_0$; $y^{(1)}(x_0) = y_0^{(1)}$.
Równanie to można zapisać jako układ dwóch równań różniczkowych pierwszego rzędu

$$\begin{cases} z^{(1)} = f(x, y(x), z(x)); & z(x_0) = y_0^{(1)} \\ y^{(1)}(x) = z(x); & y(x_0) = y_0. \end{cases} \tag{10.20}$$

Korzystając ze wzorów podanych w poprzednim punkcie, znajdujemy rozwiązanie równania (10.19).

1.10.7. Metoda szeregów Taylora

Niech będzie dane równanie różniczkowe zwyczajne n-tego rzędu o postaci

$$y^{(n)}(x) = f(x, y(x), y^{(1)}(x), \cdots, y^{(n-1)}(x)) \tag{10.21}$$

z warunkami początkowymi

$$y(x_0) = y_0, \quad y^{(1)}(x_0) = y_0^{(1)}, \ \cdots, \quad y^{(n-1)}(x_0) = y_0^{(n-1)}. \tag{10.22}$$

Przy założeniu, że funkcja $f : R^{n+1} \to R$ jest analityczna w pewnym obszarze $D \subset R^{n+1}$, poszukujemy rozwiązania $y(x)$ w postaci szeregu Taylora.

$$\begin{aligned} y(x) &= y(x_0) + \frac{y^{(1)}(x_0)}{1!}(x - x_0) + \frac{y^{(2)}(x_0)}{2!}(x - x_0)^2 + \cdots \\ &+ \cdots \frac{y^{(n)}(x_0)}{n!}(x - x_0)^n + \cdots \end{aligned} \tag{10.23}$$

Wartości $y(x_0), y^{(1)}(x_0), \cdots, y^{(n-1)}(x_0)$ są dane w postaci warunków początkowych, natomiast $y^{(n)}(x_0)$ oblicza się bezpośrednio z równania (10.21) podstawiając $x = x_0$. Wartości pochodnych $y^{(k)}(x_0), \cdots$, dla $k = n+1, n+2, \ldots$ otrzymuje się przez wielokrotne obustronne różniczkowanie równania (10.21) i podstawienie $x = x_0$.

Przykład

Niech będzie dane równanie różniczkowe

$$y^{(2)}(x) = \frac{xy^{(1)}(x) - y(x)}{x - 1} \tag{10.24}$$

z warunkami początkowymi:

$$y(x_0) = y(0) = 1, \quad y^{(1)}(x_0) = y^{(1)}(0) = 3. \tag{10.25}$$

Z równania różniczkowego bezpośrednio oblicza się

$$y^{(2)}(x_0) = y^{(2)}(0) = 1. \tag{10.26}$$

Różniczkując obustronnie równanie różniczkowe otrzymuje się kolejno:

$$y^{(3)}(x) = \frac{(x^2 - x)y^{(2)}(x) - xy^{(1)}(x) + y(x)}{(x - 1)^2}, \tag{10.27}$$

stąd $y^{(3)}(0) = 1$,

$$y^{(4)}(x) = \frac{x(x-1)^2 y^{(3)}(x) - (x^2-1)y^{(2)}(x) + 2xy^{(1)}(x) - 2y(x)}{(x-1)^3}, \tag{10.28}$$

zatem $y^{(4)}(0) = 1$. Na podstawie wzoru (10.23) przybliżone rozwiązanie równania różniczkowego ma więc postać

$$\begin{aligned} y(x) &\approx y(0) + y^{(1)}(0)x + \frac{1}{2}y^{(2)}(0)x^2 + \frac{1}{6}y^{(3)}(0)x^3 + \frac{1}{24}y^{(4)}(0)x^4 = \\ &+ 1 + 3x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4. \end{aligned} \tag{10.29}$$

Przykład

Niech będzie dane równanie różniczkowe

$$y^{(1)}(x) = [y(x)]^2 + 2x$$

z warunkiem początkowym $y(x_0) = y(0) = 0$. Z równania różniczkowego bezpośrednio oblicza się

$$y^{(1)}(x_0) = y^{(1)}(0) = y^2(0) = 0.$$

Różniczkując obustronnie równanie różniczkowe otrzymuje się kolejno

$$y^{(2)}(x) = 2y(x)y^{(1)}(x) + 2,$$

stąd: $y^{(2)}(0) = 2y(0)y^{(1)}(0) + 2 = 2$,

$$y^{(3)}(x) = 2[y^{(1)}(x)]^2 + 2y(x)y^{(2)}(x),$$

stąd: $y^{(3)}(0) = 2[y^{(1)}(0)]^2 + 2y(0)y^{(2)}(0) = 0$,

$$y^{(4)}(x) = 6y^{(1)}(x)y^{(2)}(x) + 2y(x)y^{(3)}(x),$$

stąd: $y^{(4)}(0) = 6y^{(1)}(0)y^{(2)}(0) + 2y(0)y^{(3)}(0) = 0$,

$$y^{(5)}(x) = 6[y^{(2)}(x)]^2 + 8y^{(1)}(x)y^{(3)}(x) + 2y(x)y^{(4)}(x),$$

stąd: $y^{(5)}(0) = 6[y^{(2)}(0)]^2 + 8y^{(1)}(0)y^{(3)}(0) + 2y(0)y^{(4)}(0) = 24$.

Zatem na podstawie wzoru (10.23) przybliżone rozwiązanie równania różniczkowego ma postać

$$\begin{aligned} y(x) &= y(0) + y^{(1)}(0)x + \frac{1}{2}y^{(2)}(0)x^2 + \frac{1}{6}y^{(3)}(0)x^3 + \frac{1}{24}y^{(4)}(0)x^4 + \\ &+ \frac{1}{120}y^{(5)}(0)x^5 = x^2 + \frac{1}{5}x^5. \end{aligned}$$

powered by Misiol

1.10.8. Metoda Picarda

Metoda Picarda umożliwia znajdowanie przybliżonych rozwiązań równania różniczkowego pierwszego rzędu

$$y^{(1)}(x) = f(x, y(x)) \tag{10.30}$$

z warunkiem początkowym $y(x_0) = y_0$ w postaci wzoru analitycznego. Wykorzystując wzór iteracyjny

$$y_{n+1}(x) = y_0 + \int_{x_0}^{x} f(t, y_n(t))dt \tag{10.31}$$

z początkową funkcją $y_0(x) = y_0 = const$ tworzy się ciąg kolejnych przybliżeń $y_1(x), y_2(x), \cdots, y_k(x), \cdots$ rozwiązania równania (10.24). Można dowieść, że ciąg funkcyjny $\{y_k(x)\}_{k=0}^{k=\infty}$ jest niemal jednostajnie zbieżny do rozwiązania $y(x)$ równania (10.24) z zadanym warunkiem początkowym. Wadą metody Picarda jest konieczność wykonywania operacji całkowania w każdym kroku iteracji.

Przykład

Niech będzie dane równanie różniczkowe

$$y^{(1)}(x) = x^2 - [y(x)]^2$$

z warunkiem początkowym $y(x_0) = y(0) = y_0 = 0$. Zatem $f(x, y(x)) = x^2 - [y(x)]^2$. Przyjmując funkcję początkową $y_0(x) = y(0) = 0$ na podstawie wzoru iteracyjnego (10.25) uzyskuje się kolejno:

$$\begin{aligned} y_1(x) &= y_0 + \int_{x_0}^{x} f(t, y_0(t))dt = \int_0^x [t^2 - y_0(t)^2]dt = \int_0^x t^2 dt = \frac{x^3}{3}, \\ y_2(x) &= y_0 + \int_{x_0}^{x} f(t, y_1(t))dt = \int_0^x [t^2 - y_1(t)^2]dt = \int_0^x [t^2 - \frac{t^6}{9}]dt = \frac{x^3}{3} - \frac{x^7}{63}, \\ y_3(x) &= y_0 + \int_{x_0}^{x} f(t, y_2(t))dt = \int_0^x [t^2 - y_2(t)^2]dt = \int_0^x [t^2 - (\frac{t^3}{3} - \frac{t^7}{63})^2]dt = \\ &= \frac{x^3}{3} - \frac{x^7}{63} + 2\frac{x^{11}}{2079} - \frac{x^{15}}{59535}. \end{aligned}$$

W przybliżeniu można zatem przyjąć, że rozwiązanie $y(x) \approx y_3(x) = \frac{x^3}{3} - \frac{x^7}{63} + 2\frac{x^{11}}{2079} - \frac{x^{15}}{59535}$.

Przykład

Niech będzie dane równanie różniczkowe

$$y^{(1)}(x) = e^x - [y(x)]^2$$

z warunkiem początkowym $y(x_0) = y(0) = 0$. Zatem

$$f(x, y(x)) = e^x - [y(x)]^2.$$

Przyjmując funkcję początkową $y_0(x) = y_0(0) = 0$ na podstawie wzoru iteracyjnego (10.25) uzyskuje się kolejno:

$$y_1(x) = y_0 + \int_{x_0}^{x} f(t, y_0(t))dt = \int_0^x e^t dt = e^x - 1,$$
$$y_2(x) = y_0 + \int_{x_0}^{x} f(t, y_1(t))dt = \int_0^x [e^t - (e^t - 1)^2]dt = 3e^x - \frac{1}{2}e^{2x} - \frac{5}{2}.$$

W przybliżeniu można zatem przyjąć, że rozwiązanie $y(x) \approx y_2(x) = 3e^x - \frac{1}{2}e^{2x} - \frac{5}{2}$.

1.11. Równania różniczkowe cząstkowe typu parabolicznego

1.11.1. Zagadnienie Dirichleta dla równania parabolicznego

Równaniem różniczkowym cząstkowym typu parabolicznego nazywa się równanie o postaci

$$\frac{\partial V(t,z)}{\partial t} - a^2 \cdot \frac{\partial^2 V(t,z)}{\partial z^2} = \eta(t,z), \qquad (t,z) \in \Omega' \subset R^2, \tag{11.1}$$

gdzie: a – stała , $V(t,z)$ – nieznana funkcja, $\eta(t,z)$ – zadana funkcja ciągła określona dla (t,z) z podzbioru Ω' dwuwymiarowej przestrzeni euklidesowej R^2. Przez podstawienie $z = x/a$ otrzymuje się równoważne prostsze równanie

$$\frac{\partial y(t,x)}{\partial t} - \frac{\partial^2 y(t,x)}{\partial x^2} = \xi(t,x), \qquad (t,x) \in \Omega \subset R^2, \tag{11.2}$$

gdzie: $y(t,x) = V(t, x/a)$ – nowa nieznana funkcja, $\xi(t,x) = \eta(t, x/a)$ – zadana funkcja ciągła określona na zbiorze Ω, powstałym ze zbioru Ω' poprzez przekształcenie liniowe. Będzie nas interesować przypadek, kiedy $\Omega = [0,1] \times [0,1]$. Aby móc jednoznacznie określić rozwiązanie równania (11.2) należy podać warunek początkowy

$$y(0,x) = y_0(x), \qquad x \in [0,1], \tag{11.3}$$

gdzie $y_0(x)$ – zadana funkcja ciągła na odcinku $[0,1]$ oraz warunki brzegowe:

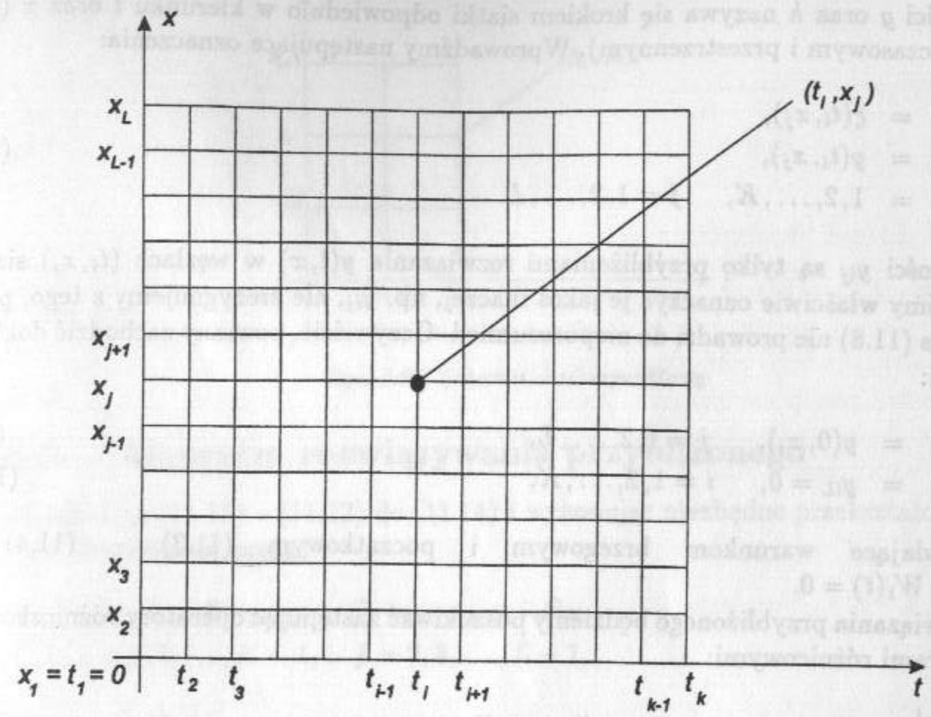
$$\begin{aligned} y(t,0) &= W_0(t), \qquad t \in [0,1], \\ y(t,1) &= W_1(t), \qquad t \in [0,1]. \end{aligned} \tag{11.4}$$

Oczywiście muszą zachodzić tzw. warunki zgodności postaci

$$y_0(0) = W_0(0) \text{ oraz } y_0(1) = W_1(0).$$

Będziemy dla uproszczenia przyjmować zerowe warunki brzegowe $W_0(t) = W_1(t) = 0$. Zbiór warunków (11.3) - (11.4) nazywa się zagadnieniem brzegowym typu Dirichleta dla

powered by Misiol



Rys. 1.19. Fragment siatki

równania (11.2). Zagadnienie Dirichleta dla równania (11.2) ma jednoznaczne rozwiązanie dane szeregiem nieskończonym

$$y(t,x) = 2\sum_{i=1}^{\infty}\left\{\sin(i\pi x)\cdot e^{-i^2\pi^2 t}\cdot\left[\int_0^1 \sin(i\pi u)y_0(u)du + \int_0^t\int_0^1 \xi(\tau,u)\sin(i\pi u)du e^{i^2\pi^2\tau}d\tau\right]\right\}. \tag{11.5}$$

Postać rozwiązania sprawia oczywiste trudności obliczeniowe, gdy chcemy znać wartość rozwiązania w określonym punkcie $(t_0, x_0) \in [0,1] \times [0,1]$. Stąd potrzeba wprowadzenia nieskomplikowanych, przybliżonych metod rozwiązywania zagadnienia Dirichleta.

1.11.2. Metoda siatek

Podstawą większości przybliżonych metod rozwiązywania zagadnień brzegowych dla równań różniczkowych cząstkowych jest dyskretyzacja czasu i przestrzeni (tzn. zmiennych t i x), co sprowadza się do konstrukcji odpowiedniej siatki pokrywającej obszar $\Omega \subset R^2$, (rysunek 1.19).

Węzłami siatki nazywa się punkty przestrzeni R^2 o współrzędnych (t_i, x_j) gdzie:

$$t_i = (i-1)h, \quad i = 1,2,\ldots,K, \quad h = 1/(K-1), \tag{11.6}$$

$$x_j = (j-1)g, \quad j = 1,2,\ldots,L, \quad g = 1/(L-1), \tag{11.7}$$

a wielkości g oraz h nazywa się krokiem siatki odpowiednio w kierunku t oraz x (bądź krokiem czasowym i przestrzennym). Wprowadźmy następujące oznaczenia:

$$\begin{aligned} \xi_{ij} &= \xi(t_i, x_j), \\ y_{ij} &= y(t_i, x_j), \\ i &= 1,2,\ldots,K, \quad j = 1,2,\ldots,L. \end{aligned} \tag{11.8}$$

Wartości y_{ij} są tylko przybliżeniami rozwiązania $y(t,x)$ w węzłach (t_i, x_j) siatki i powinniśmy właściwie oznaczyć je jakoś inaczej, np. $\tilde{y}_{ij}$, ale zrezygnujemy z tego, ponieważ zapis (11.8) nie prowadzi do nieporozumień. Oczywiście, powinny zachodzić dokładne równości:

$$y_{1j} = y(0, x_j), \quad j = 1,2,\ldots,L, \tag{11.9}$$

$$y_{i1} = y_{iL} = 0, \quad i = 1,2,\ldots,K, \tag{11.10}$$

odpowiadające warunkom brzegowym i początkowym (11.3) – (11.4) dla $W_0(t) = W_1(t) = 0$.

Rozwiązania przybliżonego będziemy poszukiwać zastępując operatory różniczkowania operatorami różnicowymi:

$$\left.\frac{\partial y}{\partial t}\right|_{t=t_{i+1}, x=x_j} \to \Delta_t(y_{i+1,j}) = \frac{y_{i+1,j} - y_{i,j}}{h}, \tag{11.11}$$

$$i = 1,2,\ldots,K-1; \quad j = 1,2,\ldots,L;$$

$$\begin{aligned} \left.\frac{\partial^2 y}{\partial x^2}\right|_{t=t_{i+1}, x=x_j} &\to \Delta_{xx}(y_{i+1,j}, \delta) = \frac{1}{g^2}[(1-\delta)(y_{i,j+1} - 2y_{i,j} + y_{i,j-1}) + \\ &\quad + \delta(y_{i+1,j+1} - 2y_{i+1,j} + y_{i+1,j-1}], \\ i &= 1,2,\ldots,K-1, \quad j = 2,3,\ldots,L-1, \quad \delta \in (0,1). \end{aligned} \tag{11.12}$$

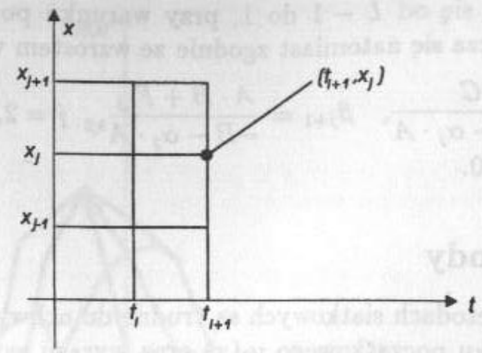
Operator $\Delta_t(\cdot)$ jest operatorem różnicy wstecznej podzielonym przez h (tzn. ilorazem różnicowym). Operator $\Delta_{xx}(\cdot, \delta)$ jest kombinacją liniową dwóch operatorów dwukrotnej różnicy centralnej podzielonej przez g^2. Taka postać $\Delta_{xx}(\cdot, \delta)$ zapewnia lepsze przybliżenie, a współczynnik δ dobiera się stosownie do postaci zagadnienia. Rysunek 1.20 przedstawia tzw. schemat sześciopunktowy, na którym opiera się sposób obliczania wartości operatora $\Delta_{xx}(\cdot, \delta)$ (tzn. do obliczenia wartości $\Delta_{xx}(y_{i+1,j}, \delta)$ używa się wartości w sześciu węzłach siatki). Oczywiście, operator $\Delta_t(\cdot)$ oparty jest na prostym schemacie dwupunktowym. Tak więc jako rozwiązanie przybliżone zagadnienia (11.2) - (11.4) przyjmiemy ciąg podwójny $\{y_{ij}\}$, dla którego zachodzi

$$\Delta_t(y_{i+1,j}) - \Delta_{xx}(y_{i+1,j}) = \xi_{ij}, \tag{11.13}$$

$$i = 1,2,\ldots,K-1; \quad j = 2,3,\ldots,L-1. \tag{11.14}$$

oraz spełniający warunki (11.9), (11.10).

powered by Misiol



Rys. 1.20. Schemat sześciopunktowy

1.11.3. Algorytm rozwiązywania przybliżonego

Wstawiając (11.11) – (11.12) do (11.14) i wykonując niezbędne przekształcenia otrzymujemy równoważną postać

$$A \cdot y_{i+1,j-1} + B \cdot y_{i+1,j} + C \cdot y_{i+1,j+1} = -F_{ij}, \tag{11.15}$$

$$i = 1,2,\ldots,K-1, \quad j = 2,3,\ldots,L-1, \tag{11.16}$$

gdzie:

$$A = -\delta/g^2; \quad B = 2\delta/g^2 + 1/h; \quad C = \delta/g^2, \tag{11.17}$$

$$F_{ij} = -\xi_{ij} - \frac{1-\delta}{g^2}(y_{i,j+1} + y_{i,j-1}) + [\frac{2(1-\delta)}{g^2} - \frac{1}{h}]y_{ij}. \tag{11.18}$$

Zauważmy, że jeżeli dla ustalonego wskaźnika i oraz danych wartości $y_{i,j}$, $j = 1,2,\ldots,L$ potrafimy z (11.15), (11.16) i warunków $y_{i+1,1} = y_{i+1,L} = 0$ wyznaczyć wartości $y_{i+1,j}$, $j = 1,2,\ldots,L$, to tym samym potrafimy wyznaczyć wartości $y_{i,j}$ dla wszystkich wskaźników i,j (ponieważ $y_{i,j}$; $j = 1,2,\ldots,L$ jest dane). Układ równań (11.15), (11.16) – dla ustalonego i – jest tzw. układem pasmowym. Nazwa ta wywodzi się z wyglądu postaci macierzowej (11.15), (11.16):

$$\begin{bmatrix} A & B & C & 0 & \cdots & & & \cdots & 0 \\ 0 & A & B & C & 0 & \cdots & & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & A & B & C & 0 & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & & \cdots & 0 & A & B & C & 0 \\ 0 & \cdots & & & \cdots & 0 & A & B & C \end{bmatrix} \cdot \begin{bmatrix} y_{i+1,1} \\ y_{i+1,2} \\ \vdots \\ y_{i+1,j} \\ \vdots \\ y_{i+1,L-1} \\ y_{i+1,L} \end{bmatrix} = - \begin{bmatrix} F_{i,2} \\ F_{i,3} \\ \vdots \\ F_{i,j-1} \\ \vdots \\ F_{i,L} F_{i,L-1} \end{bmatrix}, \tag{11.19}$$

gdzie macierz pasmowa ma $L-2$ wiersze i L kolumn. Układ (11.15) – (11.19) ma zawsze jedyne rozwiązanie, jeżeli założyć, że $y_{i+1,1}$ oraz $y_{i+1,L}$ są znane. Rozwiązanie to najłatwiej otrzymać następującą metodą rekurencyjną. Poszukujemy rozwiązania w postaci

$$y_{i+1,j} = \alpha_{j+1} \cdot y_{i+1,j+1} + \beta_{j+1}, \tag{11.20}$$

gdzie wskaźnik j zmienia się od $L-1$ do 1, przy warunku początkowym $y_{i+1,L} = 0$. Współczynniki α_j, β_j oblicza się natomiast zgodnie ze wzrostem wskaźnika j, tzn.:

$$\alpha_{j+1} = \frac{C}{-B-\alpha_j \cdot A}, \quad \beta_{j+1} = \frac{A \cdot B + F_{i,j}}{-B-\alpha_j \cdot A}, \quad j = 2,3,\ldots,L-1, \tag{11.21}$$

gdzie $\alpha_2 = \beta_2 = 0$.

1.11.4. Błąd metody

Błędy przybliżeń w metodach siatkowych są trudne do uchwycenia i zależą w dużym stopniu od postaci warunku początkowego $y_0(x)$ oraz wyrazu wolnego $\xi(t,x)$. Można co najwyżej udowodnić, że metoda jest stabilna, tzn., że błąd przybliżenia maleje wraz z krokiem siatki i ocenić szybkość tej zmiany. Ponadto, można dobrać współczynnik δ tak, aby metoda dawała jak najmniejszy błąd dla równania jednorodnego (tzn. $\xi(t,x) = 0$). Ta optymalna wartość δ wynosi

$$\delta = \frac{1}{2} - \frac{g^2}{12h}. \tag{11.22}$$

Ponieważ $0 < \delta < 1$, więc wzór (11.22) narzuca warunki na dopuszczalne kroki siatki

$$g^2 \leq \delta h. \tag{11.23}$$

W praktyce dla $\xi(t,x) = 0$ wystarczy wziąć $K = L \approx 50$. Dla $\xi(t,x) \neq 0$ krok czasowy powinien być 2–3 razy większy. Oczywiście, te wskazówki mają czysto orientacyjny charakter.

1.11.5. Przykład

Rozwiązanie dokładne

Wzór (11.5) umożliwia w pewnych przypadkach znalezienie postaci analitycznej $y(t,x)$. Dzieje się tak zwłaszcza wtedy, kiedy $\xi(t,x)$ ma postać

$$\xi(t,x) = \varphi(t) \cdot \psi(x)$$

i ponadto zachodzi:

$$\int_0^1 sin(i\pi u) y_0(u) du \neq 0, \qquad \int_0^1 sin(i\pi u)\psi(u) du \neq 0,$$

dla co najwyżej skończonej liczby wskaźników i. Wtedy suma nieskończona w (11.5) redukuje się do sumy skończonej.

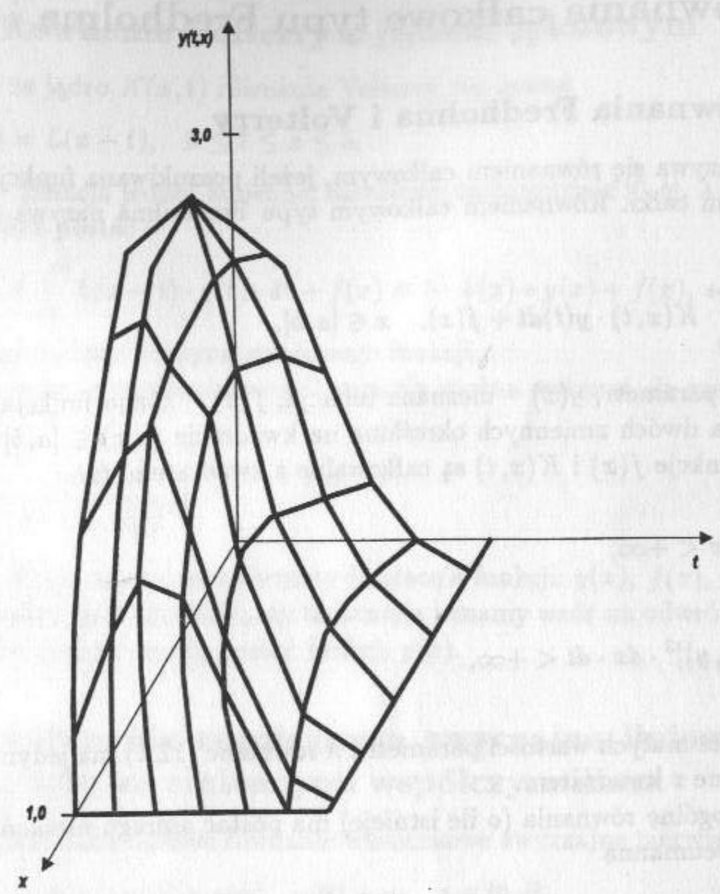
Weźmy np. $\xi(t,x) = 4\pi sin(\pi x) \cdot sin(at) \cdot exp(-\pi^2 \cdot t)$ oraz $y_0(x) = 0$, $a = 4\pi$. Wtedy:

$$\begin{aligned} y(t,x) &= 2\sum_{i=1}^{\infty}\{\sin(i\pi x) exp(i\pi^2 t) \cdot \int_0^t [\int_0^1 \sin(\pi u)\sin(i\pi u) du] \sin(4\pi\tau) d\tau\} = \\ &= 2\sin(\pi x) exp(Pi^2 t) \cdot \int_0^t [\int_0^1 \sin^2(\pi u) du] \sin(4\pi\tau) d\tau, \end{aligned}$$

ponieważ całka od 0 do 1 z funkcji $sin(\pi u) \cdot sin(i\pi u)$ równa jest zeru dla $i \neq 1$. Ostatecznie

$$y(t,x) = sin(\pi x) exp(-\pi^2 t)[1 - cos(4\pi t)].$$

powered by Misiol



Rys. 1.21. Wykres przestrzenny powierzchni y(t,x)

Rozwiązanie przybliżone

Równanie różniczkowe cząstkowe o postaci takiej samej jak w poprzednim punkcie rozwiązano metodą siatek, stosując: $h = 0.005$, $g = 0.02$. Takie wartości g oraz h spełniają warunek (11.23), ponieważ $g^2 = 0.0004 < 0.03 = 6h$. Wynika stąd wartość $\delta = 0.5 - g^2(12h = 0.5 - 0.0004)0.06 \approx 0.493$. Przestrzenny wykres rozwiązania numerycznego (oraz sposób postępowania przy konstrukcji takiego wykresu) przedstawiono na rysunku 1.21. Wykres przestrzenny pozwala na zorientowanie się w ogólnym kształcie powierzchni $y(t,x)$. Należy go rysować stosując oczywiście „rzadszą" siatkę od tej, która służyła do obliczeń, np. $\Delta t, \Delta x \approx 0.05 - 0.1$.

1.12. Równania całkowe typu Fredholma i Volterry

1.12.1. Równania Fredholma i Volterry

Równanie nazywa się równaniem całkowym, jeżeli poszukiwana funkcja występuje w nim pod znakiem całki. Równaniem całkowym typu Fredholma nazywa się równanie o postaci

$$y(x) = \lambda \int_a^b K(x,t) \cdot y(t)dt + f(x), \quad x \in [a,b], \tag{12.1}$$

gdzie: λ – stały parametr, $y(x)$ – nieznana funkcja, $f(x)$ – znana funkcja, wyraz wolny, $K(x,t)$ – funkcja dwóch zmiennych określona na kwadracie $(t,x) \in [a,b] \times [a,b]$ zwana jądrem. Jeżeli funkcje $f(x)$ i $K(x,t)$ są całkowalne z kwadratem, tzn.:

$$\int_a^b [f(x)]^2 dx < +\infty,$$

$$\int_a^b \int_a^b [K(x,y)]^2 \cdot dx \cdot dt < +\infty,$$

to dla dostatecznie małych wartości parametru λ równanie (12.1) ma jedyne rozwiązanie, również całkowalne z kwadratem.

Rozwiązanie ogólne równania (o ile istnieje) ma postać szeregu nieskończonego, zwanego szeregiem Neumanna

$$y(x) = f(x) + \sum_{i=1}^{i=\infty} f_i(x), \tag{12.2}$$

gdzie

$$f_i(x) = \int_a^b K_i(x,t) \cdot f(t) \cdot dt,$$

a funkcje $K_i(x,t)$ są zdefiniowane rekurencyjnie:

$$\begin{aligned} K_1(x,t) &= \lambda \cdot K(x,t), \\ K_{i+1}(x,t) &= \lambda \cdot \int_a^b K(x,\tau) \cdot K_i(\tau,t)d\tau, \quad i = 1,2,\ldots. \end{aligned} \tag{12.3}$$

Jeżeli jądro $K(x,t)$ ma własność $K(x,t) = 0$ dla $x < t$, to równanie (12.1) przybiera postać następującą

$$y(x) = \lambda \cdot \int_a^x K(x,t) \cdot y(t) \cdot dt + f(x), \quad x \in [a,b] \tag{12.4}$$

i zwane jest równaniem całkowym Volterry II rodzaju. Równania całkowe Volterry mają rozwiązania dla dowolnych wartości λ.

powered by Misiol

1.12.2. Równanie Volterry z jądrem splotowym

Załóżmy, że jądro $K(x,t)$ równania Volterry ma postać

$$K(x,t) = L(x-t), \quad 0 \le t \le x \le b, \tag{12.5}$$

gdzie: $L(x)$ – funkcja jednej zmiennej określona na przedziale $[0,b]$, λ – stała. Równanie (12.4) przybiera postać

$$y(x) = \lambda \int_0^x L(x-t) \cdot y(t) \cdot dt + f(x) \equiv \lambda \cdot L(x) \star y(x) + f(x), \tag{12.6}$$

gdzie $\star$ jest symbolem iloczynu splotowego funkcji.

Stosując metody przekształcenia Laplace'a można wykazać, że zachodzi następujący związek

$$\hat{Y}(s) = \frac{\hat{F}(s)}{1 - \lambda \cdot \hat{L}(s)}, \tag{12.7}$$

gdzie: $\hat{Y}(s)$, $\hat{F}(s)$, $\hat{L}(s)$ – transformaty Laplace'a funkcji: $y(x)$, $f(x)$, $L(x)$.

W przypadku, gdy transformaty te istnieją i znamy wzór na odwrócenie transformaty $\hat{Y}(s)$, możemy znaleźć jawną postać funkcji $y(x)$.

1.12.3. Równanie różniczkowe zwyczajne liniowe pierwszego rzędu ze zmiennymi współczynnikami

Niech będzie dane liniowe równanie różniczkowe zwyczajne pierwszego rzędu

$$y^{(1)}(x) = A(x) \cdot y(x) + g(x), \quad y(0) = y_0, \quad x \in [0,b]. \tag{12.8}$$

Jak wiadomo, w przypadku gdy $A(x), g(x)$ są całkowalne z modułem na przedziale $[0,b]$, tzn.

$$\int_0^b |A(x)| \cdot dx < +\infty, \quad \int_0^b |g(x)| \cdot dx < +\infty, \tag{12.9}$$

to równanie (12.8) ma rozwiązanie ogólne w sensie Caratheodory'ego (całkę ogólną) $y(x)$ o postaci

$$y(x) = y_0 e^{\int_0^x A(s) \cdot ds} + \int_0^x e^{\int_0^x A(u)du} f(s) \cdot ds. \tag{12.10}$$

Jeżeli więc potrafimy znaleźć postaci analityczne całek występujących we wzorze (12.10), to możemy znaleźć postać analityczną $y(x)$.

Z drugiej strony, całkując obustronnie równanie (12.8), otrzymujemy równanie całkowe o następującej postaci

$$y(x) = \int_0^x A(t) \cdot y(t)dt + y_0 + \int_0^x g(t)dt, \tag{12.11}$$

które jest równaniem całkowym Volterry z jądrem $K(x,t) = A(t)$ oraz wyrazem wolnym $f(x) = y_0 + \int_0^x g(t) \cdot dt$. Na odwrót, jeżeli równanie Volterry ma postać

$$y(x) = \int_0^x A(t) \cdot y(t) \cdot dt + f(x) \tag{12.12}$$

i funkcja $f(x)$ posiada pochodną, to jest ono równoważne równaniu różniczkowemu (12.8), jeżeli za y_0 przyjąć $f(0)$, a za $g(x)$

$$g(x) = \frac{d}{dx}[f(x)]. \tag{12.13}$$

Rozwiązanie równania Volterry typu (12.12) wyraża się więc wzorem (12.10). Odwrotnie, jeżeli nie potrafimy policzyć całek występujących w (12.10), możemy równanie różniczkowe (12.8) przedstawić w postaci (12.11) i rozwiązać metodą Neumanna.

1.12.4. Metoda Neumanna

Zazwyczaj nie można znaleźć jawnej postaci rozwiązania równań (12.1), (12.4) i trzeba użyć metody przybliżonej wyznaczania rozwiązania. Zauważmy na wstępie, że wzory (12.2) - (12.4) można zapisać alternatywnie w następującej postaci:

$$y(x) = \lim_{n\to\infty} y_n(x), \tag{12.14}$$

$$y_0(x) \equiv 0, \tag{12.15}$$

$$y_{n+1}(x) = f(x) + \int_a^b \lambda \cdot K(x,t) \cdot y_n(t) \cdot dt, \qquad n = 0,1,2,\ldots$$

(przyjęcie innej postaci funkcji $y_0(x)$ nie tylko nie przeszkadza, ale może przyspieszyć zbieżność ciągu $y_n(x)$ do rozwiązania $y(x)$). Metoda przybliżona, wykorzystująca wzory (12.2) - (12.4), nazywa się metodą Neumanna.

Z obliczeniowego punktu widzenia dla każdej konkretnie wybranej wartości x należy wykonać całkowanie w całym zakresie zmian t. Takie całkowanie względem t dla pojedynczej wartości $x = x_i$ nazywa się małym cyklem. Aby zwiększyć wskaźnik n , tzn. uzyskać kolejną przybliżoną funkcję $y_{n+1}(x)$, należy odpowiednio dużo razy wykonać mały cykl. Taka grupa małych cykli nazywa się dużym cyklem. Jeżeli metoda Neumanna ma być zaprogramowana, zmienne t oraz x należy zdyskretyzować i przyjąć odpowiednią regułę całkowania numerycznego. Jeżeli stosuje się całkowanie metodą Eulera, czyli metodą prostokątów, równanie (12.14) przyjmuje następującą postać

$$y_{n+1}(x_i) = f(x_i) + \lambda \cdot \sum_{j=1}^{N} K(x_i,t_j) \cdot y_n(t_j) \cdot \Delta t, \qquad i = 1,2,\ldots,N, \tag{12.16}$$

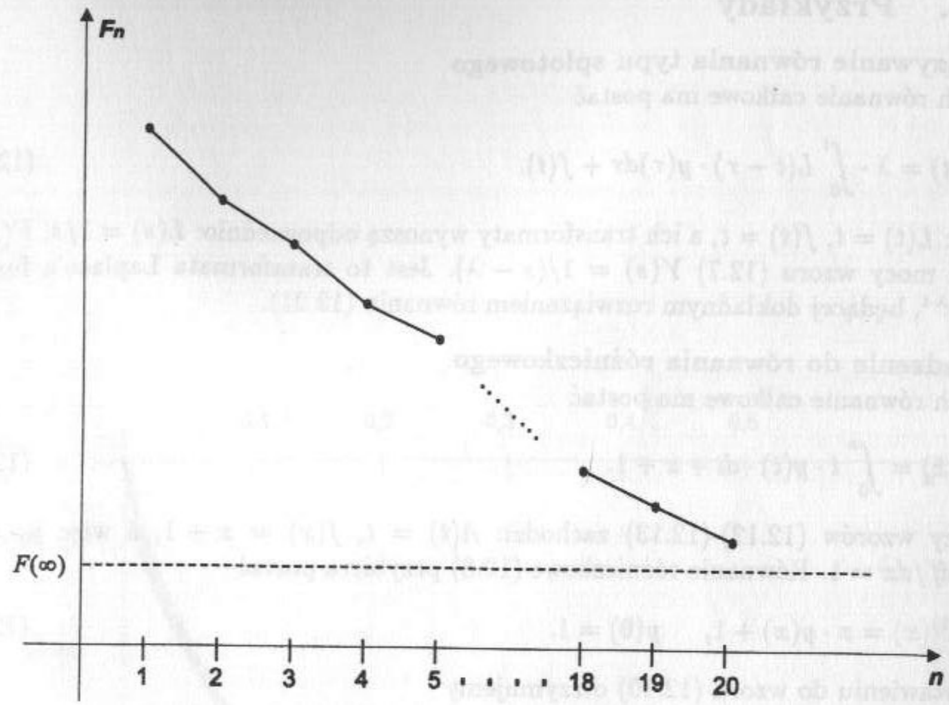
gdzie:

$$\begin{aligned} N &= (b-a)/\Delta t, \\ x_i &= i \cdot \Delta t, \quad i = 1,2,\ldots,N, \\ t_j &= j \cdot \Delta t, \quad j = 1,2,\ldots,N, \end{aligned} \tag{12.17}$$

a Δt jest przyjętym krokiem dyskretyzacji. W przypadku równania Volterry wzór (12.16) redukuje się do postaci następującej

$$y_{n+1}(x_i) = f(x_i) + \lambda \cdot \sum_{j=1}^{i} K(x_i,t_j) \cdot y_n(t_j) \cdot \Delta t, \qquad i = 1,2,\ldots,N. \tag{12.18}$$

powered by Misiol



Rys. 1.22. Przebieg wartości błędu średniokwadratowego w metodzie Neumanna

Teoretycznie do uzyskania ścisłego rozwiązania wymagana jest nieskończona liczba dużych cykli. Praktyczne doświadczenia wykazały, że dla odpowiednio regularnych funkcji K oraz f można uzyskać dokładność rzędu 1% po wykonaniu około 20 dużych cykli. Błąd rozwiązywania numerycznego w n-tym cyklu (tzw. błąd średniokwadratowy) definiuje się wzorem

$$F_n = \left\{ \sum_{i=1}^{N} [y_n(x_i) - y(x_i)]^2 \right\}^{1/2}; \quad n = 1,\ldots,N, \tag{12.19}$$

gdzie $y(x)$ jest rozwiązaniem dokładnym. Przykładowy wykres błędu średniokwadratowego jest przedstawiony na rysunku 1.22.

Wartość $F(\infty)$, do której zmierza błąd F_n przy $n \to \infty$, nie może być równa zeru, ponieważ w metodzie Neumanna rozwiązuje się właściwie układ równań algebraicznych postaci

$$y_i = f_i + \sum_{j=1}^{N} K_{ij} \cdot y_j, \qquad i = 1,2,\ldots,N, \tag{12.20}$$

a nie oryginalne równanie całkowe. Teoretycznie błąd F_∞ powinien maleć jak $(\Delta t)^2$. W praktyce, stosowanie wartości $\Delta t < (b-a)/100$ prowadzi do oscylacji wartości F_n dla dużych n (z powodu przypadkowej kumulacji błędów zaokrągleń). Ponadto czas trwania obliczeń w jednej iteracji dla równań Volterry jest zazwyczaj kilka razy krótszy od czasu obliczeń w jednej iteracji dla równań Fredholma.

1.12.5. Przykłady

Rozwiązywanie równania typu splotowego

Niech równanie całkowe ma postać

$$y(t) = \lambda \cdot \int_0^t L(t-\tau) \cdot y(\tau) d\tau + f(t). \tag{12.21}$$

Funkcje: $L(t) = t$, $f(t) = t$, a ich transformaty wynoszą odpowiednio: $L(s) = 1/s$, $F(s) = 1/s$. Na mocy wzoru (12.7) $Y(s) = 1/(s-\lambda)$. Jest to transformata Laplace'a funkcji $y(t) = e^{\lambda \cdot t}$, będącej dokładnym rozwiązaniem równania (12.21).

Sprowadzenie do równania różniczkowego

Niech równanie całkowe ma postać

$$y(x) = \int_0^x t \cdot y(t) \cdot dt + x + 1. \tag{12.22}$$

Na mocy wzorów (12.12)-(12.13) zachodzi: $A(t) = t$, $f(x) = x + 1$, a więc $y_0 = 1$, $g(x) = df/dx = 1$. Równanie różniczkowe (12.8) przybiera postać

$$y^{(1)}(x) = x \cdot y(x) + 1, \qquad y(0) = 1. \tag{12.23}$$

Po podstawieniu do wzoru (12.10) otrzymujemy

$$y(x) = e^{\frac{x^2}{2}} + \int_0^x e^{\left(\frac{x^2}{2} - \frac{s^2}{2}\right) ds} = e^{\frac{x^2}{2}} \cdot \left[1 + \int_0^x e^{\frac{-s^2}{2}} \cdot ds\right]. \tag{12.24}$$

Niestety, całka funkcji $e^{\frac{-s^2}{2}}$ nie wyraża się przez funkcje elementarne.

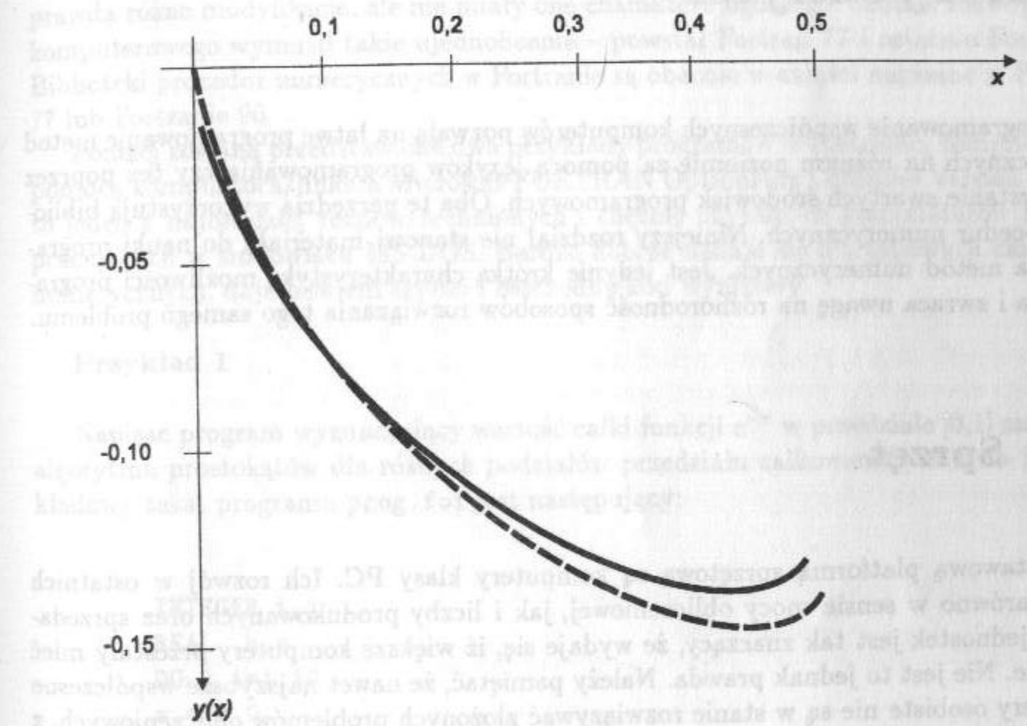
Przykład działania algorytmu Neumanna

Zaprogramowany na komputerze algorytm Neumanna posłużył do rozwiązania równania całkowego

$$y(x) = \int_{-1/2}^{+1/2} (x^2 + t^2) \cdot y(t) \cdot dt + x^3 - x/2, \tag{12.25}$$

którego dokładne rozwiązanie ma postać $y(x) = x^3 - x/2$. Przyjęto $\Delta t = 0.01$, $y_0(x) = 2x^2$. Rozwiązania dokładne i przybliżone dla $x \in [0, 1/2]$ porównane są na rysunku 1.23.

powered by Misiol



Rys. 1.23. Rozwiązanie przybliżone (linia ciągła) i dokładne (linia przerywana) równania

Rozdział 2

Narzędzia programowania metod numerycznych

Oprogramowanie współczesnych komputerów pozwala na łatwe programowanie metod numerycznych na różnym poziomie za pomocą języków programowania czy też poprzez wykorzystanie zwartych środowisk programowych. Oba te narzędzia wykorzystują biblioteki procedur numerycznych. Niniejszy rozdział nie stanowi materiału do nauki programowania metod numerycznych. Jest jedynie krótką charakterystyką możliwości programowania i zwraca uwagę na różnorodność sposobów rozwiązania tego samego problemu.

2.1. Sprzęt

Podstawową platformą sprzętową są komputery klasy PC. Ich rozwój w ostatnich latach zarówno w sensie mocy obliczeniowej, jak i liczby produkowanych oraz sprzedawanych jednostek jest tak znaczący, że wydaje się, iż większe komputery przestały mieć znaczenie. Nie jest to jednak prawda. Należy pamiętać, że nawet najszybsze współczesne komputery osobiste nie są w stanie rozwiązywać złożonych problemów obliczeniowych, z którymi duże komputery radzą sobie w rozsądnym czasie. Jednak ze względu na popularność komputerów personalnych poniższe punkty będą odwoływały się głównie do tej klasy sprzętu.

2.2. Języki programowania

Podstawowym narzędziem programowania są języki, z których najbardziej popularne dla programowania metod numerycznych to Fortran, Pascal i C. Szczególną rolę odgrywa Fortran, który daje na ogół najszybszy kod. Poniżej wymienione języki będą pokrótce scharakteryzowane.

powered by Misiol

2.2.1. Język Fortran

Język Fortran jest wciąż uważany za podstawowy język programowania algorytmów rozwiązujących zadania numeryczne. Opracowanie Fortranu jest głównie zasługą firmy IBM (wszystkie komputery produkowane przez tę firmę są wyposażone w kompilatory tego języka).

Fortran powstał na bazie doświadczeń z zakresu języków programowania z lat 1951-1956; pierwsze publikacje pochodzą z 1957r. Jest to więc jeden z najwcześniejszych języków wyższego rzędu. Pierwszą wersją był tzw. Fortran-SIT, następnie Fortran II, który wprowadził możliwość niezależnej kompilacji podprogramów. Pewną odmianą był opracowany później i wchłonięty przez język Algol – XTran. Opracowanie na początku lat 60-tych Fortranu IV na długi czas określiło zasadnicze cechy tego języka. Wprowadzano co prawda różne modyfikacje, ale nie miały one charakteru ogólnego. Jednak rozwój sprzętu komputerowego wymusił takie ujednolicenia – powstał Fortran 77 i ostatnio Fortran 90. Biblioteki procedur numerycznych w Fortranie są obecnie w całości napisane w Fortranie 77 lub Fortranie 90.

Poniżej zostaną przedstawione dwa przykłady programów w Fortranie, kompilowane za pomocą kompilatora/linkera Microsoft FORTRAN Optimizing Compiler Version 5.0. Jest to jeden z najbardziej rozpowszechnionych i chętnie używanych kompilatorów Fortranu, pracujących w środowisku MS-DOS. Bardzo dobrze nadaje się dla typowych zastosowań numerycznych, daje bowiem szybki i oszczędny kod wynikowy.

Przykład 1

Napisać program wyznaczający wartość całki funkcji e^{-x} w przedziale [0,1] za pomocą algorytmu prostokątów dla różnych podziałów przedziału całkowania: od 1 do 10. Przykładowy tekst programu `prog.for` jest następujący:

```
      INTEGER i,j
      REAL  h,s
      DO 2 i=1,10
      s = 0.0
      h=i
      h = 1/h
      DO 1 j=1,i
1     s = s + h*exp(-h*(j-1))
2     WRITE(*,3)s
3     FORMAT(F8.5)
      END
```

Uruchomienie kompilacji następuje przez zlecenie `fl prog.for`. W wyniku kompilacji i konsolidacji powstaje program `prog.exe`. Jego uruchomienie powoduje następujący wydruk na ekranie monitora:

```
1.00000
 .80326
```

```
.74331
.71442
.69743
.68625
.67834
.67245
.66788
.66425
```

Są to kolejne wartości całki dla odpowiednich wartości liczby podprzedziałów całkowania.

Należy zwrócić uwagę na charakterystyczną deklarację zmiennych w programie: INTEGER i REAL oraz na pętlę DO. Urządzenie o numerze logicznym '*' to dla operacji czytania klawiatura, a dla operacji pisania ekran monitora. Dzięki podstawieniu h=i typ całkowity (i) zmienia się na rzeczywisty (h), co umożliwia poprawne obliczenie ilorazu h/i.

Kolejny przykład jest bardziej złożony i ilustruje inne możliwości języka w zagadnieniach numerycznych.

Przykład 2

Napisać program realizujący algorytm iteracji prostej dla zadania rekurencyjnego rozwiązywania układu równań liniowych. Program powinien sprawdzać czy dany układ równań można rozwiązać rekurencyjnie (należy wykorzystać jeden z warunów wystarczających zbieżności). Ponadto program powinien badać sprzeczność układu (dla uproszczenia należy zbadać przypadek, w którym jedna z kolumn macierzy układu jest zerowa). Przykład programu rozwiązującego to zadanie jest następujący:

```
      INTEGER i,j,lo,l,n,k,r,sprawdz
      REAL z,z1,e,modul,modul1
      REAL x(10),xi(10),xo(10),g(10),b(10),blad(10),a(10,10),h(10,10)

      WRITE(*,1)
1     FORMAT(' Dane przykladowe (1) czy z klawiatury (2)')
      READ(*,*)i

      WRITE(*,2)
2     FORMAT(' Podaj dokladnosc :')
      READ(*,*)e
      WRITE(*,3)
3     FORMAT(' Podaj liczbe iteracji :')
      READ(*,*)lo

      IF(i.NE.1) GO TO 4
      CALL DANE(a,b,n)
      GO TO 8
```


powered by Mi3iol

```
4     CALL ODCZYT(a,b,n)

8     WRITE(*,5)
5     FORMAT(' Podaj przyblizone poczatkowe rozwiazanie ')
      DO 7 i=1,n
      WRITE(*,6)i
6     FORMAT(' Xo',I2,' =')
7     READ(*,*)xo(i)

      DO 15 i=1,n

      modul1=0.0
      DO 9 k=1,n
      modul=abs(a(k,i))
      IF(modul.LE.modul1) GO TO 9
      modul1=modul
      r=k
9     CONTINUE

      IF(a(r,i).NE.0.0) GO TO 11
      WRITE(*,10)
10    FORMAT(' Uklad sprzeczny ')
      PAUSE 'Nacisnij dowolny klawisz'
      STOP
11    CONTINUE

      g(i)=b(i)/a(i,i)
      DO 12 j=1,n
12    h(i,j)=-a(i,j)/a(i,i)
      h(i,i)=0.0
      z=0.0
      DO 13 j=1,n
13    z=z+abs(h(i,j))
      IF(abs(z).LT.1) GO TO 15
      WRITE(*,14)
14    FORMAT(' Uklad rozbiezny ')
      PAUSE 'Nacisnij dowolny klawisz'
      STOP
15    CONTINUE

      DO 16 i=1,n
      xi(i)=xo(i)
16    x(i)=xo(i)

      DO 24 l=1,lo
```

```
      DO 18 i=1,n
      z=0
      DO 17 j=1,n
17    z=z+h(i,j)*xi(j)
18    x(i)=g(i)+z

      z=0.0
      DO 20 i=1,n
      z1=z
      z=abs(x(i)-xi(i))
      blad(i)=z
      IF(z1.GE.z) GO TO 20
19    z1=z
20    CONTINUE

      sprawdz=0
      IF(abs(z1).GT.e) GO TO 22
      sprawdz=1
      CALL WYNIK(sprawdz,n,e,l,x,blad)
      PAUSE 'Nacisnij dowolny klawisz'
      STOP
22    CALL WYNIK(sprawdz,n,e,l,x,blad)
      PAUSE 'Nacisnij dowoly klawisz'

      IF(l.GE.lo) GO TO 24
      DO 23 i=1,n
23    xi(i)=x(i)
24    CONTINUE

      l=l-1
      CALL WYNIK_ZLY(n,l,blad,x,xi,e)

      END

      SUBROUTINE DANE(a,b,n)
      INTEGER n
      REAL b(10),a(10,10)
      a(1,1) =  6.1
      a(1,2) =  2.2
      a(1,3) =  1.2
      a(2,1) =  2.2
      a(2,2) =  5.5
      a(2,3) = -1.5
      a(3,1) =  1.2
      a(3,2) = -1.5
      a(3,3) =  7.2
```


powered by Misiol

```
      b(1)   =  16.55
      b(2)   =  10.55
      b(3)   =  16.8
      n = 3
      END

      SUBROUTINE ODCZYT(a,b,n)
      INTEGER n
      REAL b(10),a(10,10)

      WRITE(*,1)
1     FORMAT(' Podaj liczbe zmiennych :')
      READ(*,*)n
      WRITE(*,2)
2     FORMAT(' Podaj elementy macierzy A :')
      DO 4 i=1,n
      DO 4 j=1,n
      WRITE(*,3)i,j
3     FORMAT(' a(',I2,',',I2,')  :')
4     READ(*,*)a(i,j)
      WRITE(*,5)
5     FORMAT(' Podaj wektor wyrazow wolnych :')
      DO 7 i=1,n
      WRITE(*,6)i
6     FORMAT(' b(',I2,')  :')
7     READ(*,*)b(i)
      END

      SUBROUTINE WYNIK(sprawdz,n,e,l,x,blad)
      INTEGER n,sprawdz,l
      REAL blad(10),x(10),e

      IF(sprawdz.NE.1) GO TO 2
      WRITE(*,1)e,l
1     FORMAT(' Wynik koncowy przy dokladnosci e=',F12.8,'
     *                osiagnieto w ',I4,' iteracjach.')
2     WRITE(*,3)l
3     FORMAT(' Iteracja ',I4)
      DO 4 i=1,n
4     WRITE(*,5)i,x(i),blad(i)
5     FORMAT(' X',I2,'=',F12.4,'    Blad=',F12.4)
      END

      SUBROUTINE WYNIK_ZLY(n,l,blad,x,xi,e)
      INTEGER n,l
      REAL blad(10),x(10),xi(10),e
```

```
      WRITE(*,1)e,l
1     FORMAT(' Nie osiagnieto dokladnosci ',F12.8,' w ',I4,
     *       ' iteracjach')
      DO 2 i=1,n
2     WRITE(*,3)i,xi(i),i,x(i),blad(i)
3     FORMAT(' X(i-1)',I2,'=',F12.4,'  X',I2,'=',F12.4,'  Blad=',F12.8)
      PAUSE 'Nacisnij dowolny klawisz'
      END
```

Program wykorzystuje podprogramy: DANE – do ustalenia przykładowych wartości elementów macierzy a i wektora b, ODCZYT – do wprowadzenia tych wartości z klawiatury, WYNIK – do wyświetlenia wyników obliczeń, jeśli zakończyły się one prawidłowo oraz WYNIK_ZLY – do wyprowadzenia komunikatu o nieprawidłowym zakończeniu obliczeń.

Program rozpoczyna się od ustalenia danych wejściowych. Wykorzystuje się przykład danych na stałe wprowadzonych do programu lub pisząc dowolną liczbę różną od 1 użytkownik może wprowadzić dane z klawiatury. Wcześniej jednak trzeba dodatkowo podać założoną dokładność, liczbę iteracji oraz początkowe rozwiązanie układu równań.

Następnie program sprawdza, czy układ jest sprzeczny – zgodnie ze specyfikacją zadania. Dokonuje się to w pętli DO 15. Jeśli maksymalna co do modułu wartość w kolumnie jest równa zero (jest to sprawdzane pętlą DO 9 względem zmiennej k), to układ jest sprzeczny, co program sygnalizuje i kończy pracę. Jednocześnie, w pętli DO 15 dokonuje się przekształcenia układu równań do postaci $\boldsymbol{x} = \boldsymbol{g} + \boldsymbol{H}\boldsymbol{x}$ oraz sprawdza się jeden z warunków wystarczających zbieżności procesu iteracji, jak wyspecyfikowano w zadaniu.

Z kolei program przechodzi do wykonywania poszczególnych iteracji. Najpierw nadaje się zmiennym iterowanym wartości początkowe x0, a następnie w pętli DO 24 względem zmiennej l dokonuje się iteracji. Najpierw jest obliczona nowa wartość zmiennej iterowanej x (pętla DO 18 względem i). Następnie wyznacza się z1 – największy co do modułu przyrost różnicy pomiędzy rozwiązaniem w poprzedniej iteracji a rozwiązaniem aktualnym. Jeśli z1 jest mniejsze od założonej dokładności, to zostanie wyświetlony wynik działania ostatniej iteracji i obliczenia zostaną zakończone, jeśli nie – zostanie wyświetlony wynik ostatniej iteracji i obliczenia będą kontynuowane. O tym, jak ma zadziałać procedura wyświetlania wyniku, decyduje zmienna sprawdz. Aktualne rozwiązanie jest przepisywane do bufora xi. Jeśli założona dokładność nie zostanie osiągnięta w danej liczbie iteracji, program zakończy obliczenia wywołując procedurę WYNIK_ZLY.

Należy podkreślić, że program ma charakter przykładowy i że to samo zadanie można rozwiązać inaczej i efektywniej. W przykładzie chodzi jedynie o ilustrację sposobu programowania. Poniżej przedstawiono przykład działania programu dla danych zakodowanych wewnątrz programu.

```
Dane przykladowe (1) czy z klawiatury (2)
1
Podaj dokladnosc :
0.01
Podaj liczbe iteracji :
10
Podaj przyblizone poczatkowe rozwiazanie
```


powered by Misiol

```
Xo 1 =
1
Xo 2 =
2
Xo 3 =
1
Iteracja   1
X 1=      1.7951   Blad=       .7951
X 2=      1.7909   Blad=       .2091
X 3=      2.5833   Blad=      1.5833
Nacisnij dowolny klawisz

Iteracja   2
X 1=      1.5590   Blad=       .2361
X 2=      1.9047   Blad=       .1138
X 3=      2.4073   Blad=       .1761
Nacisnij dowolny klawisz

Iteracja   3
X 1=      1.5526   Blad=       .0064
X 2=      1.9511   Blad=       .0464
X 3=      2.4703   Blad=       .0630
Nacisnij dowolny klawisz

Iteracja   4
X 1=      1.5235   Blad=       .0291
X 2=      1.9709   Blad=       .0198
X 3=      2.4810   Blad=       .0107
Nacisnij dowolny klawisz

Iteracja   5
X 1=      1.5142   Blad=       .0092
X 2=      1.9854   Blad=       .0146
X 3=      2.4900   Blad=       .0090
Nacisnij dowolny klawisz

Wynik koncowy przy dokladnosci e=   .01000000
   osiagnieto w    6 iteracjach.
Iteracja   6
X 1=      1.5072   Blad=       .0070
X 2=      1.9916   Blad=       .0061
X 3=      2.4946   Blad=       .0046
```

```
Nacisnij dowolny klawisz

Stop - Program terminated
```

Dokładne rozwiązanie wynosi:

```
X1 = 1.5
X2 = 2
X3 = 2.5
```

W ostatnim czasie pojawiła się na rynku okienkowa wersja Fortranu, a mianowicie Microsoft Fortran Power-Station. Jej szczególną zaletą jest łatwość pisania, uruchamiania i sprawdzania poprawności działania programu napisanego w Fortranie.

2.2.2. Język Pascal

Pascal jest obecnie jednym z najpopularniejszych języków programowania. Od chwili powstania, tj. od 1971r. uchodzi za podstawowy język programowania strukturalnego metodą „top – down". Na szczególne miejsce Pascala wśród języków programowania miało wpływ upowszechnienie mikrokomputerów, dla których opracowano odpowiednie wersje tego języka. Nie ulega wątpliwości, że prymat wiedzie tu Turbo Pascal, wprowadzony na rynek przez amerykańską firmę Borland International Inc. w 1983r.

Turbo Pascal nie jest tylko językiem jako takim, jest to środowisko składające się z kompilatora, ekranowego edytora tekstowego oraz systemu do śledzenia programu i wyszukiwania błędów Całość połączona jest w jednolity interakcyjny system programowania. Pakiet, początkowo oferowany w stosunkowo ubogiej i prostej wersji, z czasem został rozbudowany również o odpowiednie możliwości wizualizacji, a także o techniki programowania obiektowego. Kolejne wersje Turbo Pascala udoskonalały kompilator, zwłaszcza pod względem szybkości kompilacji, rozbudowywały system informacji pomocniczych, udoskonalały fazę łączenia poprzez usuwanie zbędnych obiektów, co zasadniczo poprawiało kod wynikowy, poprawiały również zarządzanie strukturami nakładkowymi oraz rozbudowywały możliwości kontroli w czasie uruchamiania programu.

Z uwagi na to, że Turbo Pascal jest najczęściej wykorzystywanym narzędziem do programowania w przedstawionym w niniejszym skrypcie kursie metod numerycznych, poniżej przedstawione są krótko podstawowe elementy tego języka. Opis ma pomagać w wykonywaniu ćwiczeń laboratoryjnych i daleki jest od wyczerpania tematu. Na rynku księgarskim znajduje się wiele pozycji zawierających szczegółowy kurs Języka Turbo Pascal.

Przykład 3

Napisać program wyznaczający wartość całki funkcji e^{-x} w przedziale [0,1] za pomocą algorytmu prostokątów dla różnych podziałów przedziału całkowania: od 1 do 10. Przykład rozwiązania jest następujący:

powered by Misiol

```
PROGRAM Calkowanie_numeryczne_przyklad;

   uses crt;

   var
i,j : integer;
h,s : real;

BEGIN
  clrscr;
  for i := 1 to 10 do
    begin
      s := 0.0;
      h := 1/i;
      for j := 1 to i do
s := s + h*exp(-h*(j-1));
      writeln(s);
    end;
  readln;
END.
```

Pogram wykorzystuje moduł `crt` (`crt.tpu`), zawierający szereg użytecznych procedur i funkcji, dzięki którym program użytkownika może sterować ekranem, rozszerzonymi kodami znaków klawiatury, kolorem, okienkami i dźwiękiem. Na początku każdego programu należy zadeklarować zmienne w nim wykorzystywane. W powyższym przykładzie są wykorzystane zmienne całkowite `i` oraz `j`, które pełnią w nim rolę indeksów w pętlach. Program korzysta również ze zmiennych rzeczywistych: `h`, będącej krokiem całkowania oraz `s`, która służy do sumowania aproksymującego całkowanie.

Właściwy program rozpoczyna `BEGIN`. Procedura `clrscr` czyści ekran i jest częścią modułu `crt`. Obliczenia wykonywane są w dwóch pętlach `for`; w pętli zewnętrznej wyświetlane są kolejne wartości obliczonej aproksymacji całki. Procedurę `readln` wywołano jedynie po to, aby można było odczytać z ekranu wyniki obliczeń. Jeśli więc przedstawiony powyżej tekst zostanie napisany za pomocą edytora ekranowego Turbo Pascala i następnie zostanie wywołane zlecenie `Run` (za pomocą `Alt R` lub `Ctrl F9`), to na ekranie będą wyświetlone wyniki obliczeń:

```
1.0000000000E+00
8.0326532986E-01
7.4331614320E-01
7.1442449888E-01
6.9743827987E-01
6.8625983712E-01
6.7834669680E-01
6.7245095314E-01
6.6788856367E-01
6.6425326613E-01
```

Naciśnięcie dowolnego klawisza powoduje powrót do edytora ekranowego Turbo Pascala.

Format, w którym wyniki są przedstawione, można zmieniać. Pisząc przykładowo

```
writeln(s:8:5);
```

otrzymuje się wyniki zawierające 8 cyfr, w tym 5 po przecinku dziesiętnym:

```
1.00000
0.80327
0.74332
0.71442
0.69744
0.68626
0.67835
0.67245
0.66789
0.66425
```

Pomimo swojej prostoty przytoczony program stanowi typowy przykład programu dla obliczeń numerycznych w Turbo Pascalu. Poniżej zostanie przedstawiony inny przykład, zwierający rozwiązania programowe szczególnie użyteczne dla obliczeń numerycznych w Turbo Pascalu.

Przykład 4

Napisać program realizujący algorytm iteracji prostej dla zadania rekurencyjnego rozwiązywania układu równań liniowych. Program powinien sprawdzać, czy dany układ równań można rozwiązać rekurencyjnie wykorzystując w celu sprawdzenia wybrany warunek wystarczający zbieżności. Ponadto należy badać sprzeczność układu (dla uproszczenia powinno badać się przypadek, w którym jedna z kolumn macierzy układu jest zerowa). Przykład programu rozwiązującego to zadanie jest następujący:

```
PROGRAM Iteracja_prosta;

 uses crt;

 var
    zn : char;
    i,j,lo,l,n,k,r,sprawdz : integer;
    z,z1,e,modul,modul1 : real;
    x   : array[1..10] of real;
    xi  : array[1..10] of real;
    xo  : array[1..10] of real;
    g   : array[1..10] of real;
    b   : array[1..10] of real;
```


powered by Misiol

```
    blad: array[1..10] of real;
    a   : array[1..10,1..10] of real;
    h   : array[1..10,1..10] of real;

PROCEDURE dane;
begin
  a[1,1] :=  6.1;
  a[1,2] :=  2.2;
  a[1,3] :=  1.2;
  a[2,1] :=  2.2;
  a[2,2] :=  5.5;
  a[2,3] := -1.5;
  a[3,1] :=  1.2;
  a[3,2] := -1.5;
  a[3,3] :=  7.2;

  b[1]   := 16.55;
  b[2]   := 10.55;
  b[3]   :=  16.8;

  n := 3;

end;

PROCEDURE odczyt;
begin
  write('Podaj liczbe zmiennych :');readln(n);
  writeln('Podaj elementy macierzy A :');
  for i:=1 to n do
    for j:=1 to n do
      begin
        write('a[',i,',',j,']  :');
        readln(a[i,j]);
      end;
  writeln('Podaj wektor wyrazow wolnych :');
  for i:=1 to n do
    begin
      write('b[',i,']  :');
      readln(b[i]);
    end;
  clrscr;
end;

PROCEDURE wynik;
begin
  if sprawdz = 1 then
```

```
    begin
      writeln;
      writeln('Wynik koncowy przy dokladnosci e=',e,
                                ' osiagnieto w ',l,' iteracji.');
      writeln;
    end
  else
    writeln('Iteracja ',l);
    for i:=1 to n do writeln('X',i,'=',x[i],'    Blad=',blad[i]);
    if sprawdz = 1 then
      begin
        writeln;
        writeln('K - koniec');
      end;
end;

PROCEDURE wynik_zly;
begin
  writeln;
  writeln('Nie osiagnieto dokladnosci ',e,' w ',l,' iteracjach');
  writeln;
  for i:=1 to n do
    writeln(' X(i-1)',i,'=',xi[i],'  X',i,'=',x[i],'  Blad=',blad[i]);
  writeln;writeln('K - koniec');
  repeat
    zn:=readkey;
  until (zn='k') or (zn='K');
end;

BEGIN
  clrscr;
  writeln('Dane przykladowe (T) czy z klawiatury (N)');
  zn:=readkey;
  clrscr;
  write('Podaj dokladnosc :');readln(e);
  write('Podaj liczbe iteracji :');readln(lo);
  if (zn='t') or (zn='T') then dane
    else odczyt;
  writeln('Podaj przyblizone poczatkowe rozwiazanie  ');
  for i:=1 to n do
    begin
      write('Xo',i,' =');
      readln(xo[i]);
    end;
  writeln;
```


powered by Misiol

```
  i:=1;
  repeat
    modul1:=0;
    for k:=1 to n do
    begin
      modul:=abs(a[k,i]);
      if modul>modul1 then
      begin
        modul1:=modul;
        r:=k;
      end;
    end;
    if a[r,i]=0 then
    begin
      clrscr;
      writeln('Uklad sprzeczny ');
      readln;
      halt;
    end;

    g[i]:=b[i]/a[i,i];
    for j:=1 to n do h[i,j]:=-a[i,j]/a[i,i];
    h[i,i]:=0;
    z:=0;
    for j:=1 to n do z:=z+abs(h[i,j]);
    if abs(z)>=1 then
      begin
        clrscr;
        writeln('Uklad rozbiezny ');
        readln;
        halt;
      end;
    i:=i+1;
  until i>n;

  l:=1;
  for i:=1 to n do
    begin
      xi[i]:=xo[i];
      x[i]:=xo[i];
    end;

  repeat
    for i:=1 to n do
```

```
  begin
    z:=0;
    for j:=1 to n do z:=z+h[i,j]*xi[j];
    x[i]:=g[i]+z;
  end;
  z:=0;
  for i:=1 to n do
    begin
      z1:=z;
      z:=abs(x[i]-xi[i]);
      blad[i]:=z;
      if z1<z then z1:=z;
    end;
  sprawdz:=0;
  if abs(z1)<=e then
    begin
      sprawdz:=1;
      wynik;
      repeat
        zn:=readkey;
      until (zn='k') or (zn='K');
      halt;
    end;
  wynik;
  readln;
  if l<lo then
    for i:=1 to n do xi[i]:=x[i];
    l:=l+1;
 until l>lo ;
 l:=l-1;
 wynik_zly;
end.
```

Program wykorzystuje podprogramy (procedury): **dane** – do ustalenia przykładowych wartości elementów macierzy $\boldsymbol{A}$ i wektora $\boldsymbol{b}$, **odczyt** – do wprowadzenia tych wartości z klawiatury, **wynik** – do wyświetlenia wyników obliczeń, jeśli zakończyły się one prawidłowo oraz **wynik_zly** – do wyprowadzenia komunikatu o nieprawidłowym zakończeniu obliczeń.

Program rozpoczyna się od ustalenia danych wejściowych. Wykorzystuje się przykład na stałe umieszczony w programie (procedura **dane**). Pisząc T użytkownik może wprowadzić dane z klawiatury. Wcześniej jednak trzeba dodatkowo podać założoną dokładność, liczbę iteracji oraz początkowe rozwiązanie.

Następnie program sprawdza, czy układ jest sprzeczny – zgodnie ze specyfikacją zadania. Dokonuje się to w pętli **repeat** względem zmiennej i określającej kolumny macierzy $\boldsymbol{A}$. Jeśli maksymalna co do modułu wartość w kolumnie jest równa zero (jest to sprawdzane pętlą **for** względem zmiennej k), to układ jest sprzeczny, co program sygnalizuje, po czym kończy pracę. Jednocześnie dokonuje się przekształcenia układu równań do postaci $\boldsymbol{x} = \boldsymbol{g} + \boldsymbol{H}\boldsymbol{x}$ oraz sprawdza się jeden z warunków wystarczających zbieżności procesu iteracji, jak wyspecyfikowano w zadaniu.

powered by Misiol

Z kolei program przechodzi do wykonywania poszczególnych iteracji. Najpierw nadaje się zmiennym iterowanym wartości początkowe **x0**, a następnie w pętli **repeat** względem zmiennej l dokonuje się iteracji. Oblicza się nową wartość zmiennej iterowanej **x** (pierwsza pętla względem **i**). Następnie wyznacza się **z1**, największy co do modułu przyrost różnicy pomiędzy rozwiązaniem w poprzedniej iteracji a rozwiązaniem aktualnym. Jeśli **z1** jest mniejsze od założonej dokładności, to zostanie wyświetlony wynik działania ostatniej iteracji i obliczenia zostaną zakończone, jeśli nie, to wyświetla się wynik ostatniej iteracji i obliczenia są kontynuowane. O tym, jak ma zadziałać procedura wyświetlania wyniku, decyduje zmienna **sprawdz**. Aktualne rozwiązanie jest przepisywane do bufora **xi**. Jeśli założona dokładność nie będzie osiągnięta w danej liczbie iteracji, program zakończy obliczenia wywołując procedurę **wynik_zly**.

Należy podkreślić, że program ma charakter przykładowy i że to samo zadanie można rozwiązać inaczej i efektywniej. W prezentowanym przykładzie chodzi jedynie o ilustrację sposobu programowania. Oprócz struktury programu warto zwrócić uwagę na sposób wprowadzania danych oraz wykorzystania sterowania pracą programu z klawiatury (funkcja **readkey**).

Program, uruchomiony dla danych domyślnych i przykładowych parametrów, powoduje, że na ekranie monitora otrzymuje się następujący wydruk:

```
Dane przykladowe (T) czy z klawiatury (N)
Podaj dokladnosc :0.01
Podaj liczbe iteracji :10
Podaj przyblizone poczatkowe rozwiazanie
Xo1 =1
Xo2 =2
Xo3 =1

Iteracja 1
X1= 1.4016393443E+00   Blad= 4.0163934426E-01
X1= 2.3363636364E+00   Blad= 3.3636363636E-01
X1= 2.5833333333E+00   Blad= 4.1666666667E-01

Iteracja 2
X1= 1.3622950820E+00   Blad= 3.9344262295E-02
X1= 2.0620715350E+00   Blad= 2.7429210134E-01
X1= 2.5864692002E+00   Blad= 3.1358668675E-03

Iteracja 3
X1= 1.4606032103E+00   Blad= 9.8308128312E-02
X1= 2.0786644764E+00   Blad= 1.6592941338E-02
X1= 2.5358823895E+00   Blad= 5.0586810732E-02

Iteracja 4
X1= 1.4645703745E+00   Blad= 3.9671642535E-03
```

```
X1= 2.0255448221E+00   Blad= 5.3119654251E-02
X1= 2.5229545642E+00   Blad= 1.2927825272E-02

Iteracja 5
X1= 1.4862714614E+00   Blad= 2.1701086831E-02
X1= 2.0204321859E+00   Blad= 5.1126362305E-03
X1= 2.5112267755E+00   Blad= 1.1727788678E-02

Iteracja 6
X1= 1.4904224689E+00   Blad= 4.1510075607E-03
X1= 2.0085532633E+00   Blad= 1.1878922553E-02
X1= 2.5065447952E+00   Blad= 4.6819803538E-03

Wynik koncowy przy dokladnosci e=1.0000000000E-02 osiagnieto w 7 iteracji

X1= 1.4956277158E+00   Blad= 5.2052468927E-03
X1= 2.0056159566E+00   Blad= 2.9373067591E-03
X1= 2.5033781850E+00   Blad= 3.1666101240E-03

K - koniec
```

2.2.3. Język C

Język C został po raz pierwszy zdefiniowany w 1972r. w Bell Laboratories w USA. Był to efekt modyfikacji języków B i BCPL, opracowanych kilka lat wcześniej. Mimo pewnych zmian w trakcie użytkowania zasadniczo język C nie zmienił się. Najważniejszą zaletą tego języka jest prostota jego wyrażeń i bogaty zestaw operatorów. Dzięki temu język C jest znacznie bardziej zwięzły niż Fortran lub Pascal. Obecnie język C jest jednym z najszerzej stosowanych języków programowania komputerów, szczególnie gdy chodzi o oprogramowanie systemowe. Z powodzeniem można również stosować C do programowania obliczeń numerycznych. Firma Microsoft stosuje na przykład ten sam optymalizator kodu dla kompilatorów Fortranu i C. W efekcie otrzymuje się bardzo dobry kod wynikowy. Należy ponadto podkreślić, że w sprzedaży znajduje się wiele bardzo dobrych bibliotek numerycznych pisanych w języku C.

Podobnie jak w przypadku Fortranu i Pascala poniżej zostaną przedstawione dwa przykłady programów napisanych w C. Oba zostały przygotowane i uruchomione w środowisku Turbo C firmy Borland International.

Przykład 5

Napisać program wyznaczający wartość całki funkcji e^{-x} w przedziale [0,1] za pomocą algorytmu prostokątów dla różnych podziałów przedziału całkowania: od 1 do 10. Przykład rozwiązania jest następujący:

powered by Misiol

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

void _cdecl main(void)

{
  int i,j;
  double h,s;

  for (i=1;i<=10;i++)
    {
    s=0.0;
    h=1.0/ i;
    for (j=1; j<=i; j++)
      {
      s=s+h*exp(-h*((double)j-1));
      }
    printf("%8.5lf\n",s);
    }
}
```

Header `stdio.h` dołącza do programu standardowe funkcje wejścia-wyjścia. Natomiast header `math.c` dołącza funkcje matematyczne – w tym przypadku funkcję `exp`. Program działa identycznie jak poprzednio omówione.

Przykład 6

Napisać program realizujący algorytm iteracji prostej dla zadania rekurencyjnego rozwiązywania układu równań liniowych. Program powinien sprawdzać, czy dany układ równań można rozwiązać rekurencyjnie wykorzystując w celu sprawdzenia wybrany warunek wystarczający zbieżności. Ponadto należy badać sprzeczność układu (dla uproszczenia powinno badać się przypadek, w którym jedna z kolumn macierzy układu jest zerowa). Przykład programu rozwiązującego to zadanie jest następujący:

```
#include <stdio.h>
#include <process.h>
#include <conio.h>
#include <math.h>

typedef double T_X[10];
typedef double T_XI[10];
typedef double T_XO[10];
typedef double T_G[10];
typedef double T_B[10];
typedef double T_BLAD[10];
```

```
typedef double T_A[10][10];
typedef double T_H[10][10];

char zn;
int i,j,lo_1,l,n,k,r,sprawdz;
double z,z1,e,modul,modul1;
T_X x;
T_XI xi;
T_XO xo;
T_G g;
T_B b;
T_BLAD blad;
T_A a;
T_H h;

void dane(void);
void odczyt(void);
void wynik(void);
void wynik_zly(void);


void dane(void)
{
  a[0][0]=6.1;
  a[0][1]=2.2;
  a[0][2]=1.2;
  a[1][0]=2.2;
  a[1][1]=5.5;
  a[1][2]=-1.5;
  a[2][0]=1.2;
  a[2][1]=-1.5;
  a[2][2]=7.2;
  b[0]=16.55;
  b[1]=10.55;
  b[2]=16.8;
  n=3;
}

void odczyt(void)
{
  puts("Podaj liczbe zmiennych :");
  scanf("%d\n",&n);
  puts("Podaj elementy macierzy A :\n");
  for (i=1; i<=n; i++)
    {
```


powered by Misiol

```
    for (j=1; j<=n; j++)
      {
      printf("a[%d,%d]  :",i,j);
      scanf("%f\n",&(a[i-1][j-1]));
      }
    }
  puts("Podaj wektor wyrazow wolnych :\n");
  for (i=1; i<=n; i++)
    {
    printf("b[%d]  :",i);
    scanf("%f\n",&(b[i-1]));
    }
}


void wynik(void)
{
  if (sprawdz==1)
    {
    putch('\n');
    printf("Wynik koncowy przy dokladnosci e=
    %.9lE osiagnieto w %d iteracji.\n",e,l);
    putch('\n');
    }
  else
    {
    printf("Iteracja %d\n",l);
    }
  for (i=1; i<=n; i++)
    {
    printf("X%d= %.9lE    Blad= %.9lE\n",i,x[i-1],blad[i-1]);
    }
  if (sprawdz==1)
    {
    putch('\n');
    puts("K - koniec\n");
    }
}


void wynik_zly(void)
{
  putch('\n');
  printf("Nie osiagnieto dokladnosci  %.9lE w %d iteracjach\n",e,l);
  putch('\n');
  for (i=1; i<=n; i++)
```

```
    {
    cprintf(" X(i-1)%d= % .91E  X%d=
    % .91E  Blad= % .91E\n",i,xi[i-1],i,x[i-1],blad[i-1]);
    }
  putch('\n');
  puts("K - koniec\n");
  do
    zn = (char)getch();
  while (!((zn=='k') || (zn=='K')));
}

void _cdecl main(void)
{
  puts("Dane przykladowe (T) czy z klawiatury (N)\n");
  zn = (char)getch();
  puts("Podaj dokladnosc :");
  scanf("%f\n",&e);
  puts("Podaj liczbe iteracji :");
  scanf("%d\n",&lo_1);
  if ((zn=='t') || (zn=='T'))
    {
    dane();
    }
  else
    {
    odczyt();
    }
  puts("Podaj przyblizone poczatkowe rozwiazanie  \n");
  for (i=1; i<=n; i++)
    {
    printf("Xo%d =",i);
    scanf("%f\n",&(xo[i-1]));
    }
  putch('\n');
  i=1;
  do
    {
    modul1=0.0;
    for (k=1;k<=n;k++)
      {
      modul=fabs(a[k-1][i-1]);
      if (modul>modul1)
        {
        modul1=modul;
        r=k;
        }
```


powered by Misiol

```
      }
    if (a[r-1][i-1]==0.0)
      {
      puts("Uklad sprzeczny \n");
      scanf("\n");
      exit(0);
      }
    g[i-1]=b[i-1]/ a[i-1][i-1];
    for (j=1;j<=n;j++)
      {
      h[i-1][j-1]=-a[i-1][j-1]/ a[i-1][i-1];
      }
    h[i-1][i-1]=0.0;
    z=0.0;
    for (j=1;j<=n;j++)
      {
      z=z+fabs(h[i-1][j-1]);
      }
    if (fabs(z)>=1.0)
      {
      puts("Uklad rozbiezny \n");
      scanf("\n");
      exit(0);
      }
    i++;
    }
  while (i<=n);
  l=1;
  for (i=1;i<=n;i++)
    {
    xi[i-1]=xo[i-1];
    x[i-1]=xo[i-1];
    }
  do
    {
    for (i=1;i<=n;i++)
      {
      z=0.0;
      for (j=1;j<=n;j++)
        {
        z=z+h[i-1][j-1]*xi[j-1];
        }
      x[i-1]=g[i-1]+z;
      }
    z=0.0;
    for (i=1;i<=n;i++)
```

```
    {
    z1=z;
    z=fabs(x[i-1]-xi[i-1]);
    blad[i-1]=z;
    if (z1<z)
      {
      z1=z;
      }
    }
  sprawdz=0;
  if (fabs(z1)<=e)
    {
    sprawdz=1;
    wynik();
    do
      zn = (char)getch();
    while (!((zn=='k') || (zn=='K')));
    exit(0);
    }
  wynik();
  scanf("\n");
  if (l<lo_1)
    {
    for (i=1;i<=n;i++)
      {
      xi[i-1]=x[i-1];
      }
    }
    l++;
    }
  while (l<=lo_1);
  l--;
  wynik_zly();
}
```

Dołączono tu dodatkowy header `process`, dzięki któremu można wykorzystać w programie funkcję `exit`. Analizując program można zwrócić uwagę, jak działają w nim specyficzne dla języka C operatory, np. ++ i !. Program został skonstruowany analogicznie do poprzednich, szczególnie tak jak program napisany w Pascalu. Jego działanie jest również analogiczne.

Przytoczone przykłady można również kompilować i uruchamiać za pomocą kompilatora liniowego `cl` lub w środowisku Borland C++.

powered by Misiol

2.3. Biblioteki metod numerycznych

Szereg firm produkujących oprogramowanie oferuje komercyjne biblioteki numeryczne, przeznaczone do bardzo różnych celów, w szczególności realizujące algorytmy przedstawione w rozdziale pierwszym niniejszego skryptu. Wykorzystywanie podprogramów bibliotecznych ma następujące zalety:

- Często napisanie podprogramu realizującego wybrany algorytm jest czasochłonne i drogie, zwłaszcza ze względu na konieczność przeprowadzenia testów poprawności działania podprogramu.
- Właściwie skonstruowany podprogram może być wykorzystywany do różnych celów (w różnych programach).
- Producenci profesjonalnych bibliotek dokonują optymalizacji tekstu podprogramów pod względem szybkości działania i zajętości zasobów komputera.
- Na ogół podprogramy są napisane tak, by można było je wykorzystywać na różnych platformach sprzętowych.

Dalej zostaną krótko scharakteryzowane najbardziej znane i używane biblioteki numeryczne: NAG, Mumerical Recipes, IMSL i Harwell.

2.3.1. Biblioteki NAG

Amerykańskie konsorcjum Numerical Algorithmic Group, Inc. (NAG) od 25 lat oferuje oprogramowanie komputerów: biblioteki numeryczne, oprogramowanie własne (NAGware), programy do wizualizacji obliczeń symbolicznych i statystycznych. Wśród oprogramowania numerycznego znajdują się biblioteka Fortran 77, biblioteki FL90plus i fl90 dla Fortranu 90, biblioteka procedur dla obliczeń równoległych, biblioteka dla języka C i innych języków. Jako przykład zostanie pokrótce scharakteryzowana biblioteka numeryczna dla Fortranu.

Biblioteka NAG dla Fortranu zawiera około 1200 podprogramów napisanych w Fortranie 77. Dla całości napisano odpowiednią interakcyjną i hipertekstową dokumentację oraz wskazania do instalacji i użytkowania. Ponadto biblioteka zawiera łączniki do Fortranu 90, pliki nagłówków C, zapewniające możliwość wywoływania procedur napisanych w Fortranie z poziomu C lub C++ oraz sposoby wywoływania podprogramów biblioteki w środowisku Matlab i Mathematica.

Biblioteka NAG Fortran powstała po raz pierwszy w 1971r. Od tamtego czasu oferta NAG znacznie się powiększyła i, co najważniejsze, podprogramy zostały gruntownie przetestowane, dzięki czemu biblioteki numeryczne NAG są uważane za najlepiej „serwisowane". Biblioteka jest podzielona na rozdziały, które z kolei dzielą się na mniejsze gałęzie. Poniżej są przedstawione wybrane rozdziały wraz z ich gałęziami.

- Minimalizacja
  - z ograniczeniami i bez ograniczeń

— z różniczkowaniem i bez różniczkowania
— nieliniowe metody najmniejszych kwadratów
— programowanie liniowe i kwadratowe
— liniowe programowanie całkowitoliczbowe
— programowanie nieliniowe

- Równania różniczkowe zwyczajne
  — zagadnienie warunków początkowych
  — zagadnienia brzegowe
  — algebraiczne problemy różniczkowe
  — problem Struma–Liouville'a
- Cząstkowe równania różniczkowe
  — równania eliptyczne
  — równania Helmholtza
  — równania o parametrach zależnych od czasu
- Transformacje
  — szybka transformacja Fouriera
  — transformacje trygonometryczne
  — sploty
  — odwrotna transformacja Laplace'a
- Algebra liniowa
  — systemy liniowe
  — metoda najmniejszych kwadratów
  — dekompozycja singularna
  — wektory i wartości własne
- Równania nieliniowe
  — obliczanie pierwiastków równań
  — uogólnione zagadnienie pierwiastków
- Równania całkowe
  — równania Fredholma
  — splot Volterry
  — splot Volterry–Abela

powered by Misiol

- Interpolacja, ekstrapolacja i zagadnienia pokrewne
  — interpolacja i ekstrapolacja
  — aproksymacje wielomianowe
  — metody najmniejszych kwadratów, normy l_1 i l_∞
- Funkcje specjalne
  — funkcje Bessela, Hankela i Airy
  — funkcje Kelvina
  — funkcje Gamma
- Podstawowe narzędzia badań statystycznych
  — analiza i przedstawianie danych
  — funkcje dystrybucyjne
- Generatory liczb pseudolosowych
  — dystrybucje
  — ciągi czasowe
  — macierze
  — próbkowanie losowe
- Analiza korelacyjna i regresyjna
  — korelacja nieparametryczna
  — regresja liniowa
  — uogólnione modele liniowe
  — techniki odporne

Z powyższego widać, że niemal cały materiał teoretyczny przedstawiony w pierwszym rozdziale niniejszego skryptu może być zilustrowany podprogramami z omawianej biblioteki.

Biblioteka fortranowa NAG została napisana w dwóch wersjach: dla pojedynczej i podwójnej precyzji obliczeń. Poprawność działania procedur była testowana za pomocą specjalnie w tym celu napisanych programów sprawdzających. Niewątpliwą zaletą Biblioteki jest bardzo dobra dokumentacja.

2.3.2. Biblioteki Numerical Recipes

Numerical Recipes są bibliotekami numerycznymi napisanymi w Fortranie i w C, dostępnymi w postaci opracowań książkowych. Biblioteki te są bardzo popularne na europejskich uniwersytetach. Poniżej zostanie krótko scharakteryzowana biblioteka napisana w języku C i opublikowana przez Williama Pressa i innych w książce „Numerical Recipes in C. The art of Scientific Computing. Second Edition", Cambridge Press 1995. Książka

ta stanowi połączenie pracy specjalistów z Cambridge i Numerical Recipes Software. Niektóre podprogramy zostały zaczerpnięte z publikacji w Computers in Phisics (American Institute of Phisics) w latach 1988–1992.

Biblioteka zawiera ok. 300 podprogramów ułożonych w następujących grupach tematycznych:

- rozwiązywanie układów równań liniowych (metoda eliminacji Gaussa, dekompozycja macierzy układu równań, metody iteracyjne i szereg innych),
- interpolacja i ekstrapolacja (metody wielomianowe i metody funkcji wymiernych, obliczanie współczynników interpolacji i inne),
- całkowanie numeryczne (algorytmy podstawowe, metoda Romberga, całki niewłaściwe, kwadratury Gaussa i wielomiany ortogonalne),
- obliczanie wartości funkcji (szeregi i ich zbieżność, wielomiany i funkcje wymierne, arytmetyka zespolona, zależności rekurencyjne, równanie kwadratowe i trzeciego stopnia),
- różniczkowanie numeryczne i aproksymacje Czebyszewa,
- funkcje specjalne (funkcje Gamma i Beta, funkcje Bessela, całki Fresnela, sinusowe, cosinusowe, eliptyczne i Dowsona, funkcje eliptyczne i hipergeometryczne),
- generatory liczb losowych,
- sortowanie,
- równania nieliniowe (bieguny wielomianów, metody Riddersa, van Wijngardena–Dekkera–Brenta, Newtona–Raphsona i inne),
- metody obliczania ekstremów funkcji (złoty podział, interpolacja kwadratowa i metoda Brenta, metody gradientowe, metoda Powella, gradientu sprzężonego i zmiennej metryki, programowanie liniowe),
- wartości i wektory własne (transformacja Jacobiego, Givensa i Hauseholdera, formy Hassenberga, algorytm QR),
- analiza spektralna (szybka transformacja Fouriera, metody splotowe, estymacja gęstości widmowej mocy, filtracja cyfrowa i inne),
- statystyczne przetwarzanie danych (średnie i wariancje, korelacja i inne metody nieparametryczne),
- modelowanie (estymator najmniejszych kwadratów i największej wiarygodności, modele nieliniowe, odporna estymacja),
- całkowanie zwyczajnych równań różniczkowych (metody Rungego–Kutty, ekstrapolacja Richardsona, metoda Burlischa–Stoera i inne),
- równania całkowe (równania Fredholma drugiego rodzaju, równania Volterry, metody liniowej regularyzacji, metoda Backusa–Gilberta i inne),

powered by Misiol

- równania różniczkowe cząstkowe (zagadnienia warunków początkowych, zagadnienia warunków brzegowych).

Jak widać, metody numeryczne omawiane w pierwszym rozdziale niniejszego skryptu są reprezentowane w całości w bibliotece Numerical Recipes.

2.3.3. Biblioteki IMSL

Biblioteki numeryczne IMSL są licencjonowanym produktem amerykańskiej firmy Visual Numerics Inc. i zostały wprowadzone na rynek ponad 20 lat temu. Akronim IMSL jest skrótem angielskiego International Mathematical and Statistical Library. Visual Numerics szczyci się, że biblioteki IMSL zostały zainstalowane już w niemal 70 krajach świata. Liczba oferowanych w nich podprogramów wynosi ponad 200 tys.

Obecnie całość bibliotek IMSL pogrupowano następująco:

- IMSL Fortran 77 Numerical Libraries Ver. 3.0. Znajduje się tu ponad 900 podprogramów napisanych w Fortranie 77 z myślą o zastosowaniach w matematyce stosowanej, analizie i prezentacji danych statystycznych oraz w ekonomii. W sumie stanowi to ok. 200kB tekstu. Podprogramy dotyczą następujących dziedzin:
  - — systemy liniowe
  - — wartości i wektory własne
  - — interpolacja i aproksymacja
  - — całkowanie i różniczkowanie numeryczne
  - — równania różniczkowe
  - — transformacje
  - — równania nieliniowe
  - — optymalizacja
  - — operacje na macierzach i wektorach
  - — funkcje elementarne, trygonometryczne i hiperboliczne
  - — całki wykładnicze
  - — funkcje gamma, Bessela, Kelvina i Airy
  - — funkcje i całki eliptyczne
  - — zagadnienia probabilistyczne
  - — funkcje Mathieu
  - — podstawowa analiza statystyczna
  - — regresje, korelacje, analiza wariancji
  - — kategoryzacja i dyskretyzacja danych
  - — statystyka nieparametryczna
  - — generatory losowe

— zagadnienia predykcji
— inne programy użytkowe.

Jak widać, występują tu dwie części biblioteki: matematyczna i statystyczna. Wersja 3.0 zawiera bardzo dobrze opracowaną dokumentację *on-line*, w szczególności dotyczącą algebry liniowej, optymalizacji i równań liniowych. Podprogramy mogą pracować na aż 65 platformach sprzętowych.

- IMSL Fortran 90 MP Library Ver. 2.0. Jest nowa generacja biblioteki IMSL, wykorzystująca wszystkie udogodnienia, jakie wniósł Fortran 90, w szczególności w zakresie optymalizacji i przetwarzania wieloprocesorowego.
- IMSL C Numerical Libraries Ver. 2.0. Jest to biblioteka napisana dla użytkowników języka C. Tu również podzielono zawartość biblioteki na dwie części. Część matematyczna zawiera:

  — całkowanie i różniczkowanie
  — transformacje
  — równania różniczkowe
  — systemy liniowe
  — interpolację i aproksymację
  — wektory i wartości własne
  — optymalizację
  — funkcje specjalne
  — podstawowe operacje macierzowo-wektorowe
  — równania nieliniowe
  — programy użytkowe

  Część statystyczna zawiera:

  — podstawowe narzędzia statystyki
  — testy statystyczne
  — zagadnienia predykcji
  — analizę regresyjną i korelacyjną
  — generację liczb pseudolosowych
  — kategoryzację i dyskretyzację danych
  — inne programy użytkowe.

- IMSL Exponent Graphics. Jest to biblioteka procedur graficznych. Stanowi ważne uzupełnienie poprzednio wymienionych bibliotek, zwłaszcza że poprzez dołączone drajwery współpracuje z różnymi urządzeniami, np. CGM, PostScript lub HPGL. Dostarczany jest też dokładny opis przeznaczony do wykorzystania w okienkowych systemach operacyjnych.

powered by Misiol

2.3.4. Biblioteki Harwell

Biblioteka Harwell (Harwell Subroutine Library – HSL) została napisana w Fortranie 77 i Fortranie 90. Biblioteka zawiera podprogramy zgrupowane w rozdziałach oznaczonych literą stanowiącą zarazem pierwszą literę nazwy podprogramu:

- A – algebra
- D – równania różniczkowe
- E – wartości własne i wektory własne
- F – funkcje matematyczne
- G – problemy geometryczne
- I – funkcje całkowitoliczbowe
- K – sortowanie
- L – programowanie liniowe
- M – macierze i algebra liniowa
- N – równania nieliniowe
- O – przetwarzanie danych wejściowych i wyjściowych
- P – wielomiany i funkcje wymierne
- Q – całkowanie numeryczne
- S – statystyka
- T – interpolacja i ekstrapolacja
- V – optymalizacja i analiza nieliniowa
- Y – generator programów testujących
- Z – programy wspomagające

Jako przykład zostanie scharakteryzowany rozdział M – macierze i algebra liniowa. Rozdział zawiera 66 podstawowych podprogramów, które między innymi rozwiązują następujące zadania:

- rozwiązywanie układów równań liniowych (np. podprogram MA01 – eliminacja Gaussa, MA09 – dekompozycja Choleskiego, MA10 – dekompozycja Banachiewicza, MA11 – układy nadparametryzowane, MA12 – układy z macierzą Hessenberga, MA19 – rozwiązuje zadanie minimaksowe, MA20 – układy nadparametryzowane, metody programowania liniowego);

- odwracanie macierzy (np. MB01 – metoda eliminacji Gaussa, MB04 – odwracanie symetrycznej macierzy blokowej, MB10 – uogólnione odwracanie macierzy prostokątnej);
- faktoryzacja macierzy (np. MC06 – faktoryzacja Grama–Schmidta, MC11 – faktoryzacja dodatnio lub półdodatnio określonej macierzy trójkątnej);
- wektory i wartości własne (np. MC15 – równoważenie wartości elementów macierzy w celu obliczenia jej wartości własnych i wektorów własnych).

2.4. Pakiety programowe

Pakiety programowe do rozwiązywania zadań matematycznych, obejmujących również problemy numeryczne, są obecnie bardzo często wykorzystywane. Jest to związane z rozpowszechnieniem komputerów osobistych, dla których te pakiety pisane są przede wszystkim. Zaletą pakietów jest zintegrowanie środowiska programowego w taki sposób, że edycja, programowanie, kompilacja i głównie wizualizacja są przyjazne dla użytkownika. Pakiety są wyposażone w wiele funkcji wbudowanych, dzięki czemu programista jest zwolniony z konieczności każdorazowego pisania odpowiednich procedur i nie musi też sięgać po specjalistyczne biblioteki. Poniżej przedstawiono wybrane pakiety oraz ich krótką charakterystykę. Wybrane dwa pakiety zostały szerzej omówione w następnych podpunktach.

1. Matlab, producent The MathWorks Inc.
   - Platforma sprzętowa: MacIntosh, Windows, OS/2, Unix (SGI, SUN, HP,IBM).
   - Pakiet wykorzystuje własny język programowania.
   - Można pisać moduły w językach C lub Fortran.
   - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych; można przedstawiać przekroje i wykresy teksturowe.
   - Do pakietu można dokupić specjalistyczne oprogramowanie dla aplikacji pracujących w czasie rzeczywistym oraz dla obliczeń specjalistycznych.
2. Mathematica, producent Wolfram Research Inc.
   - Platforma sprzętowa: MacIntosh, Windows, OS/2, Unix (SGI, SUN, HP,IBM).
   - Pakiet posiada znaczną liczbę wbudowanych funkcji, w większości napisanych we własnym specjalistycznym języku.
   - Wykorzystuje własny język programowania oraz język makr.
   - Można pisać moduły w językach C lub Fortran.
   - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, rastrowych; można przedstawiać np.: przekroje, wykresy teksturowe itp. innych sposobów.

powered by Misiol

   - Programy przygotowane na jednej platformie mogą pracować na każdej innej.
3. Data Desk, producent Data Description Inc.
   - Platforma sprzętowa: MacIntosh.
   - Pakiet jest przeznaczony do obliczeń w czasie rzeczywistym, również do tworzenia wykresów i animacji.
   - Wykorzystuje własny język programowania oraz język makr.
   - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych poprzez interakcyjnie dołączane procedury graficzne.
4. Origin, producent Microcal Software Inc.
   - Platforma sprzętowa: Windows.
   - Pakiet wspomaga dynamiczną wymianę danych Microsoft (DDE) oraz łączenie i zanurzanie obiektów (OLE), dzięki czemu można integrować go z dowolnym pakietem Windows.
   - Wykorzystuje własny język programowania oraz język makr.
   - Można pisać moduły w językach C.
   - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych i słupkowych.
5. O-Matrix, producent Harmonics Software Inc.
   - Platforma sprzętowa: Windows, Windows NT.
   - Pakiet jest zorientowany na przetwarzanie macierzy. Posiada szereg wbudowanych funkcji do analizy numerycznej, wizualizacji, optymalizacji, przetwarzania sygnałów, aproksymacji, obliczeń statystycznych, rozwiązywania równań różniczkowych i analizy widmowej.
   - Wykorzystuje własny język programowania oraz język makr.
   - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, rastrowych i teksturowych.
6. Gauss-386i, producent Aptech Systems Inc.
   - Platforma sprzętowa: DOS, Unix (SGI, SUN, HP,IBM, DEC, DEC Alpha).
   - Pakiet posiada znaczną liczbę wbudowanych funkcji zawierających Linpack, Eispack i Blas. Można dokupić moduły opcjonalne. Posiada rozbudowany debager i system pomocy (help). Wykorzystuje popularne formaty graficzne, np. PostScript.
   - Wykorzystuje własny język programowania.
   - Można pisać moduły w językach C lub Fortran.
   - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, można przedstawiać również przekroje.

7. Scientist, producent Micromath Scientific Software Inc.
   - Platforma sprzętowa: Windows.
   - Pakiet jest przeznaczony głównie dla celów identyfikacji. Ponadto służy do rozwiązywania równań różniczkowych oraz różnych problemów interpolacyjnych.
   - Wykorzystuje własny język programowania.
   - Nie posiada własnych procedur do wizualizacji wyników.
8. MLAB, producent Civilized Software Inc.
   - Platforma sprzętowa: DOS, MacIntosh, NeXT, Unix (SGI, SUN, HP,IBM).
   - Pakiet jest przeznaczony dla celów identyfikacji, rozwiązywania zadań numerycznych i analizy statystycznej.
   - Wykorzystuje własny język programowania i język makr.
   - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych; można przedstawiać również przekroje.
9. Igor Pro, producent WaveMetrics Inc.
   - Platforma sprzętowa: MacIntosh.
   - Pakiet jest przeznaczony głównie dla celów identyfikacji. Jest wyposażony w narzędzia do analizy numerycznej, między innymi szybką transformatę Fouriera i algorytmy wygładzania.
   - Wykorzystuje własny język programowania oraz język makr.
   - Można pisać moduły w językach C lub Fortran.
   - Wizualizację wyników można przeprowadzać za pomocą wykresów dwuwymiarowych i wykresów konturowych.
10. CoVis, producent CoHort Software.
    - Platforma sprzętowa: DOS.
    - Pakiet posiada wbudowany arkusz kalkulacyjny pozwalający importować dane ASCII, binarne, z LOTUSa, EXCELa, QUATTRO i dBASE.
    - Wykorzystuje własny język makr.
    - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, rastrowych; można przedstawiać różnorakie przekroje.
11. MacPhase, producent Otter Solution.
    - Platforma sprzętowa: MacIntosh.

powered by Misiol

    - Pakiet posiada znaczną liczbę wbudowanych funkcji – od najprostszego dodawania macierzy po szybką transformatę Fouriera. Wszystkie operacje mogą być uaktywniane z poziomu menu lub za pomocą rozbudowanego języka makr. Pakiet akceptuje szereg formatów zbiorów: PICT, TIFF, binarny, tekstowy, MATLABa, HDF i FITS.
    - W pakiecie można również programować za pomocą własnego języka programowania.
    - Można pisać moduły w językach C lub Fortran.
    - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych i rastrowych.
12. Snap-Master General Analisis, producent Hem Data Corp.
    - Platforma sprzętowa: Windows.
    - Pakiet umożliwia operowanie złożonymi równaniami. Posiada wbudowane funkcje umożliwiające całkowanie, różniczkowanie, analizę statystyczną, cyfrowe filtrowanie i analizę korelacyjną
    - Własne moduły można pisać w specjalnym języku.
    - Pakiet nie posiada funkcji dla wizualizacji wyników.
13. PS1-Plot, producent Poly Software International Inc.
    - Platforma sprzętowa: Windows, Windows NT.
    - Pakiet posiada funkcje wykonujące interpolację, rozwiązujące zwyczajne równania różniczkowe, szybką transformację Fouriera, cyfrowe przetwarzanie sygnałów oraz testy statystyczne.
    - Własne moduły można pisać w specjalnym języku.
    - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, rastrowych; można przedstawiać przekroje, wykresy teksturowe i na szereg innych sposobów.
14. TK Solver, producent Universal Technical Systems Inc.
    - Platforma sprzętowa: MacIntosh, Windows, Unix (SGI, Sun, HP, IBM).
    - Pakiet posiada funkcje do rozwiązywania zadań numerycznych: różniczkowanie, całkowanie, równania różniczkowe, algebra macierzowa, optymalizacja i statystyka.
    - Własne moduły można pisać w specjalnym języku i języku makr.
    - Pakiet nie posiada funkcji do wizualizacji wyników.
15. Maple V, producent Waterloo Maple Inc.
    - Platforma sprzętowa: MacIntosh, Windows, Unix (SGI, Sun, HP, IBM, DEC, MIPS, Inergraph).

- Pakiet stanowi narzędzie programowe do rozwiązywania problemów matematycznych w sposób symboliczny i algebraiczny. Zawiera ponad 2500 wbudowanych funkcji bibliotecznych. Obejmują one zadania arytmetyczne, wielomiany, różniczkowanie, szeregi, całkowanie, równania różniczkowe, równania rekurencyjne i operacje macierzowe.
- Własne moduły można pisać w specjalnym języku.
- Można pisać moduły w językach C lub Fortran.
- Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, można przedstawiać różnorakie przekroje.

16. StatView, producent Abacus Concepts Inc.

- Platforma sprzętowa: MacIntosh.
- Pakiet jest użyteczny przy przetwarzaniu różnych typów zbiorów. Może odczytywać i zapisywać dane w formacie EXCELa.
- Własne moduły można pisać za pomocą języka makr.
- Pakiet nie posiada funkcji dla wizualizacji wyników.

17. Cornerstone, producent BBN Software Products Corp.

- Platforma sprzętowa: Windows, Windows NT, Unix (Sun, HP).
- Pakiet jest podobny do arkusza kalkulacyjnego i służy do analizy danych lub do prac w trybie klient-serwer. Posiada rozbudowany system pomocy. Dodatkowe moduły służą do analizy regresji, obliczeń statystycznych itp.
- Własne moduły można pisać w specjalnym języku lub języku makr.
- Można pisać również moduły w językach C lub C++.
- Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych i wykresów konturowych.

18. BMDP/Diamond, producent BMDP Statistical Software Inc.

- Platforma sprzętowa: Windows, OS/2, Unix (Sun, HP).
- Pakiet zawiera wiele możliwości wielowymiarowej prezentacji graficznej. Posiada dodatkowe funkcje i programy autonomiczne do obliczeń kalkulacyjnych (arkusze), do przetwarzania danych i obliczeń statystycznych.
- Pakiet oferuje możliwości animacji.
- Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych i innych.

19. Pirouette, producent Infometrix Inc.

- Platforma sprzętowa: Windows, Windows NT, Windows 95.

powered by Misiol

- Pakiet jest przeznaczony głównie do wizualizacji. Złożone dane można odpowiednio przetwarzać za pomocą algorytmów splotowych oraz siedmiu algorytmów klastrujących.
- Własne moduły można pisać w specjalnym języku lub języku makr.
- Można pisać również moduły w języku C.
- Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, dendrogramów i innych.

20. Metrica, producent Metrica Inc.

- Platforma sprzętowa: Unix (SGI, Sun, HP, IBM, DEC).
- Pakiet wykorzystuje szybką bazę relacyjną. Użytkownik ma dostęp do bazy danych oraz odpowiednich funkcji analizujących, zawierających algorytmy do obliczeń statystycznych, przetwarzania sygnałów oraz operacji graficznych. Pakiet może wykonywać obliczenia rozproszone.
- Własne moduły można pisać w specjalnym języku lub języku makr.
- Można pisać również moduły w języku C, Fortran lub HP-Basic.
- Pakiet oferuje procedury do animacji.
- Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, rastrowych, słupkowych, histogramowych, można wykonywać różne przekroje itp.

21. Minitab Release 10 Xtra, producent Minitab Inc.

- Platforma sprzętowa: MacIntosh, Windows, Windows NT, Unix (SGI, HP, IBM).
- Wersja Windows pakietu oferuje nowe możliwości graficznego opisu zagadnień statystycznych. Pakiet zawiera rozbudowany system pomocy. Dodatkowo wersja do MacIntosh wykorzystuje algorytmy do analizy szeregów oraz ciekawe procedury graficzne.
- Własne moduły można pisać w specjalnym języku lub języku makr.
- Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych; można też wykonywać różne przekroje.

22. SPSS, producent SPSS Inc.

- Platforma sprzętowa: Windows, RS/6000.
- Pakiet służy głównie do specjalistycznych obliczeń z dziedziny zarządzania i marketingu. Można go jednak wykorzystywać do rozwiązywania innych zadań numerycznych dzięki wielu dodatkowym modułom programowym.
- Dla pakietu nie można pisać własnych modułów.
- Nie posiada własnych procedur do wizualizacji wyników.

23. Design-Expert, producent Stat-Ease Inc.
    - Platforma sprzętowa: MacIntosh, Windows.
    - Pakiet pozwala na prowadzenie interakcyjnych doświadczeń symulacyjnych dla różnego rodzaju modeli. Zapewnia metody oszacowania wyników. Ponadto oferuje metody regresji liniowej i metody optymalizacji.
    - Własne moduły można pisać w specjalnym języku lub języku makr.
    - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych z możliwością obracania oraz wykresów konturowych.
24. S-Plus, producent StatSci – firma podlegające MathSoft Inc.
    - Platforma sprzętowa: Windows, Unix (SGI, Sun, HP, IBM).
    - Pakiet zawiera odporne algorytmy regresji liniowej, metody odpornej identyfikacji modeli autoregresyjnych, a ponadto algorytmy analizy korelacyjnej.
    - Własne moduły można pisać w specjalnym języku lub języku makr.
    - Można pisać również moduły w języku C lub Fortran.
    - Pakiet oferuje procedury do animacji.
    - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, rastrowych, słupkowych, histogramowych, perspektywowych i innych.
25. Sigma, producent Jandel Scientific.
    - Platforma sprzętowa: Windows, Windows NT, Windows 95.
    - Pakiet posiada rozbudowane możliwości graficzne. Zawiera również metody nieliniowej identyfikacji.
    - Własne moduły można pisać w specjalnym języku.
    - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, rastrowych i innych.
26. Macsyma, producent Macsyma Inc.
    - Platforma sprzętowa: Windows, Windows 95, Windows NT, Unix (SGI, Sun, HP, IBM).
    - Pakiet jest przeznaczony do obliczeń symbolicznych i numerycznych. Posługuje się formatem skoroszytu, posiada wygodne procedury graficzne, możliwości animacji i wbudowany edytor hipertekstu. Wbudowany zestaw funkcji obejmuje algebrę, trygonometrię, transformaty całkowe i rachunek wektorowo-macierzowy.
    - Własne moduły można pisać w specjalnym języku lub języku makr.
    - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych i innych.

powered by Misiol

27. Mathcad, producent MathSoft Inc.
    - Platforma sprzętowa: Windows, Windows 95, Windows NT,
    - Oprócz podstawowych procedur numerycznych pakiet zawiera szereg dodatkowych funkcji, w tym: transformacje welwetowe, procedury obliczeń symbolicznych, transformacje całkowe, procedury całkowania równań różniczkowych, dekompozycji macierzy, wyznaczanie wartości i wektorów własnych oraz procedury do obliczeń statystycznych. Wyniki obliczeń mogą być eksportowane do formatu Microsoft Word.
    - Własne moduły można pisać w specjalnym języku.
    - Można pisać również moduły w języku C lub C++.
    - Pakiet oferuje procedury do animacji.
    - Wizualizację wyników można przeprowadzać za pomocą wykresów dwu- lub trójwymiarowych, wykresów konturowych, przekrojowych itp.

2.4.1. Matlab

Pakiet Matlab jest zintegrowanym środowiskiem programowym amerykańskiej firmy MathWorks Co. Z chwilą pojawienia się systemu Windows pakiet został do niego przystosowany i uzupełniony o interface graficzny – Simulink, który pozwala operować obiektami graficznymi: łączyć je z sobą wskazując na przepływ danych (sygnałów), grupować je, maskować, nadawać wartości parametrom itd. Simulink jest przeznaczony do symulowania układów dynamicznych.

W zakresie programowania metod numerycznych wygodnie jest posługiwać się prostym, podobnym do C językiem Matlaba, w którym można pisać programy i podprogramy, wykorzystywać szereg wbudowanych funkcji wewnętrznych oraz stosować przyborniki (ang. toolboxes), czyli zestawy funkcji napisanych w języku Matlaba. Po uruchomieniu Matlaba otwiera się okno zleceń, w którym można wywoływać zlecenia bądź uruchamiać programy. Każdy program stanowi zbiór z rozszerzeniem *.m*. Jego uruchomienie następuje po podaniu nazwy w oknie zleceń. Praca w Matlabie ma więc charakter konwersacyjny, bowiem język Matlaba jest językiem konwersacyjnym. Dzięki temu można bez trudu programować proste algorytmy. Złożone obliczenia numeryczne, nawet przy wykorzystaniu szybkiego komputera, mogą okazać się zbyt czasochłonne. Należy jednak podkreślić, że nie stanowi problemu włączanie podprogramów pisanych w C lub Fortranie; służą do tego celu odpowiednie kompilatory (np. Cmex dla języka C). Po kompilacji uzyskuje się programy wykonywalne, które spełniają podobną rolę jak wbudowane funkcje Matlaba. Uzyskuje się w ten sposób 10–15-krotne przyśpieszenie obliczeń.

Przykład 7

Napisać program wyznaczający wartość całki funkcji e^{-x} w przedziale [0,1] za pomocą algorytmu prostokątów dla różnych podziałów przedziału całkowania: od 1 do 10. Przykład rozwiązania (tekst programu zapisany jako *prg.m*) jest następujący:

```
e = [];                    % wektor błędu
for k = 1:10
   h = 1/k; % h - krok całkowania
   s = 0; % wartość początkowa argumentu
   c = 0; % wartość początkowa całki
   for i = 1:k
      c = c + exp(-s) * h;
      s = s + h;
   end;
   e = [e;c];              % dopisanie nowej wartości do wektora błędu
end;
```

W oknie Matlaba wystarczy uruchomić program pisząc:

```
>> prg;
```

Znak „>>" wskazuje w oknie zleceń Matlaba gotowość przyjęcia kolejnej instrukcji. W roboczej przestrzeni Matlaba znajduje się wynik. Wystarczy napisać e, by Matlab zwrócił wartość zmiennej:

```
>> e

e =
    1.0000
    0.8033
    0.7433
    0.7144
    0.6974
    0.6863
    0.6783
    0.6725
    0.6679
    0.6643

>>
```

Można teraz wykorzystać inne funkcje Matlaba. Na przykład przebieg wartości całki można wykreślić w funkcji liczby podprzedziałów przedziału całkowania:

```
>> plot(e)
```

Wśród wielu przyborników firma Mathworks oferuje również przybornik numeryczny: NAG Foundation Toolbox, zawierający programy i procedury do optymalizacji, statystyki, równań różniczkowych i szeregu innych zadań numerycznych. Przybornik powstał dzięki współpracy pomiędzy Mathworks i NAG. Zawiera 240 funkcji zwykle w postaci programów wykonywalnych (tzw. zbiorów MEX), które mogą być wywoływane bezpośrednio podobnie jak wbudowane funkcje Matlaba. Oto wybrane przykłady zestawów funkcji przybornika:

powered by Misiol

- obliczanie zer wielomianów,
- rozwiązywanie równań różniczkowych zwyczajnych i cząstkowych,
- interpolacja i aproksymacja funkcji,
- rozwiązywanie równań liniowych,
- faktoryzacja macierzowa.

Jednym z interesujących przyborników jest przybornik do obliczeń symbolicznych – Symbolic Toolbox. Funkcją tego przybornika, która dokonuje całkowania symbolicznego jest int. Instrukcja w postaci int)S,'v' wyznacza całkę nieoznaczoną S względem v. Jeśli zostanie pominięte S, całka będzie obliczona dla wyrażenia v. Wywołanie int(S,'v',a,b) spowoduje symboliczne obliczenie całki oznaczonej w granicach a,b. W rozpatrywnym przykładzie otrzymuje się:

```
>> int('exp(-x)')

ans =

-exp(-x)

>> int('exp(-x)',0,1)

ans =

-exp(-1)+1

>>
```

2.4.2. Mathematica

Mathematica jest pakietem programowym wyprodukowanym w końcu lat osiemdziesiątych przez amerykańską firmę Wolfram Research Inc. Podobnie jak Matlab jest to pakiet pracujący w środowisku Windows. Jego uruchomienie powoduje pojawienie się okna zleceń, za pomocą którego można wykonywać zlecenia (instrukcje) lub uruchamiać programy, można również definiować własne funkcje. Mathematica ma wiele podobieństw do Matlaba, nie oferuje jednak narzędzi graficznych w rodzaju Simulinka, które bardzo ułatwiają "programowanie" układów do symulacji. Jeśli jednak chodzi o samą wizualizację graficzną wyników obliczeń – Mathematica posiada rozbudowane funkcje.

Ogromną zaletą pakietu jest elastyczne połączenie obliczeń numerycznych i symbolicznych. W tym względzie jest to bodaj najlepszy pakiet programowy dostępny na rynku i przeznaczony dla komputerów PC. Język programowania Mathematica jest językiem wysokiego poziomu, za pomocą którego można pisać dowolnie złożone programy. Jeśli np. języki typu Fortran oferują ok. 30 wbudowanych operacji matematycznych, Mathematica ma ich ponad 750 (dla jednej z pierwszych wersji – 2.1).

Pewną specyfiką pakietu Mathematica jest koncepcja zapisu instrukcji, funkcji i programów. Podstawową jednostką jest tzw. komórka, zaznaczana w oknie zleceń nawiasem po prawej stronie. Mathematica automatycznie numeruje zlecenia użytkownika, nadając im określenie *In*, gdy dotyczą wprowadzania danych lub *Out*, gdy dotyczy to odpowiedzi pakietu. Kolejne numery wskazują więc kolejność wykonywanej sekwencji. Komórki można edytować, przemieszczać, łączyć, rozłączać, drukować itd. Komórki tworzą tzw. notatnik. Może on być zapisany na trwałe i ponownie otwierany.

Za pomocą języka Mathematica można programować dowolne metody numeryczne. Trzeba podkreślić, że szereg algorytmów numerycznych jest już wbudowana w pakiecie.

Przykład 8

Napisać program wyznaczający wartość całki funkcji e^{-x} w przedziale [0,1] za pomocą algorytmu prostokątów dla różnych podziałów przedziału całkowania: od 1 do 10. Przykład rozwiązania (tekst programu napisany w oknie zleceń) jest następujący:

```
Do[
  Print[
      N[Sum[E^(-(1/k)i)(1/k),{i,0,k-1}]]],
  {k,1,10}];
```

```
1.
0.803265
0.743316
0.714424
0.697438
0.68626
0.678347
0.672451
0.667889
0.664253
```

Instrukcja `Do` wykonuje pętlę po indeksie `k` od `1` do `10`. Instrukcja `Print` powoduje, że po wykonaniu obliczeń wynik zostanie wydrukowany. Operator `N` powoduje, że obliczenia będą prowadzone numerycznie, a nie symbolicznie. Same zaś obliczenia przeprowadzone są dla konkretnej wartości `k` za pomocą funkcji wbudowanej `Sum`: liczba `E` (koniecznie dużą literą) oznacza e i jest podnoszona do potęgi $x = -i/k$, a wynik jest mnożony przez krok $1/k$. Tak obliczane wartości są sumowane. Należy zwrócić uwagę na domyślność operatora mnożenia, który zwykle występuje jako znak „*".

Ponieważ Mathematica pozwala prowadzić obliczenia symboliczne, można obecnie sprawdzić, że istotnie

$$inte^{-x} = -e^{-x}$$

powered by Misiol

pisząc w oknie zleceń:

```
Integrate[E^(-x),x]
```

w wyniku otrzymuje się

```
  -x
-E
```

Można również obliczać wartość całki oznaczonej: symbolicznie

```
Integrate[E^(-x),{x,0,1}]
```

otrzymując w wyniku:

```
     1
1 - -
    E
```

lub numerycznie pisząc

```
N[Integrate[E^(-x),{x,0,1}]]
```

i otrzymując

```
0.632121
```

Rozdział 3.

Ćwiczenia laboratoryjne

Niniejszy rozdział jest poświęcony ćwiczeniom laboratoryjnym z wybranych metod numerycznych. Zakres tematyczny ćwiczeń pokrywa i wyczerpuje materiał teoretyczny przedstawiony w rozdziale 1. Celem poszczególnych ćwiczeń jest ugruntowanie materiału teoretycznego przez zaprogramowanie na komputerze odpowiednich metod, przetestowanie poprawności działania oraz wykonanie obliczeń dla konkretnych przykładów. Metody programowania wybierane są dla konkretnego ćwiczenia przez prowadzącego i studentów, i obejmują sposoby przedstawione w rozdziale 2.

Opis każdego ćwiczenia obejmuje:

- cel ćwiczenia,
- zestaw pytań kontrolnych,
- tematy zadań do wykonania w trakcie ćwiczenia,
- uwagi dotyczące sprawozdania.

Prowadzący ćwiczenie decyduje o tym, które z zadań zostaną wykonane przez studentów. W szczególnych przypadkach formułuje inne zadania oraz dokonuje podziału sekcji na mniejsze grupy.

3.1. Błędy obliczeń numerycznych

3.1.1. Cel ćwiczenia

Celem ćwiczenia jest powtórzenie podstawowych wiadomości dotyczących teorii błędów oraz przeprowadzenie obliczeń analitycznych i numerycznych, wyznaczających wartości błędów dla wybranych problemów. W ćwiczeniu zwraca się szczególną uwagę na metodę różniczki zupełnej i metodę przedziałów wartości.

powered by Misiol

3.1.2. Zestaw pytań kontrolnych

1. Wyjaśnić następujące pojęcia związane z błędem:
   - liczba przybliżona,
   - nadmiar i niedomiar przybliżenia,
   - błąd liczby,
   - błąd bezwzględny i względny,
   - kres górny błędu i kres górny błędu względnego.
2. Wyjaśnić, co to jest błąd metody i omówić sposoby szacowania błędów metody (w tym resztę Peano i resztę Lagrange'a).
3. Wyjaśnić różnicę pomiędzy błędem obcięcia i błędem zaokrąglenia.
4. Omówić błędy początkowe i błędy działań arytmetycznych. Jak one są związane?
5. Przedstawić metodę różniczki zupełnej.
6. Sformułować zależności, za pomocą których wyznacza się przedziały wartości podstawowych działań arytmetycznych w zależności od przedziału wartości argumentów.
7. Omówić błędy działań arytmetycznych: błąd sumy, różnicy, iloczynu i ilorazu.
8. Wyjaśnić, co to jest problem odwrotny teorii błędów i jak się go rozwiązuje.

3.1.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Korzystając z rozwinięcia w szereg Taylora oraz szacując resztę według wzoru Lagrange'a obliczyć z dokładnością do 0.1, 0.01 i 0.001 następujące wartości:
   - $y = e^2$
   - $y = sin(1.5)$
   - $y = ln(3)$
2. Napisać program obliczający pole trójkąta dla boków danych z zadaną dokładnością. Program powinien wyznaczać błąd obliczenia pola powierzchni dwoma metodami: metodą różniczki zupełnej i metodą przedziałów wartości. Przeprowadzić obliczenia dla wybranych przykładów.
3. Analogicznie do punktu 2 napisać programy obliczające:
   - wysokość trójkąta i błąd jej obliczenia przy zadanych długościach boków i błędach tych długości,
   - promień koła wpisanego w trójkąt przy zadanych długościach boków i błędach tych długości,

- promień koła opisanego na trójkącie przy zadanych długościach boków i błędach tych długości,

4. Napisać program obliczający wymaganą dokładność bezwzględną promienia podstawy walca i jego wysokości, aby objętość walca była obliczana z zadaną dokładnością.
5. Napisać analogiczny do punktu 4 program, ale dla zadanej dokładności wyznaczania pola powierzchni walca.

3.1.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń analitycznych z punktu 1 w 3.1.3,
- wyniki obliczeń wszystkich napisanych w trakcie ćwiczenia programów,
- porównanie wyników obliczeń programów dla metody różniczki zupełnej i metody przedziałów wartości,
- omówienie wyników i wnioski.

3.2. Metody numerycznego obliczania wartości funkcji

3.2.1. Cel ćwiczenia

Celem ćwiczenia jest utrwalenie wiadomości na temat sposobów numerycznego wyznaczania wartości funkcji. Część ćwiczenia jest poświęcona zagadnieniom obliczania wartości wielomianu. Dokładność i szybkość obliczeń według schematu Hőrnera jest tu porównywana z obliczeniami według wzoru definicyjnego. Wyznacza się również przedział, w którym znajdują się rzeczywiste pierwiastki wielomianu. Ponadto w ćwiczeniu bada się metody rozwinięcia w szereg Taylora oraz iteracyjne wyznaczanie wartości funkcji.

3.2.2. Zestaw pytań kontrolnych

1. Sformułować iteracyjny wzór obliczania wartości wielomianu i uzasadnić jego poprawność.
2. Co to jest i jak jest sformułowany schemat Hőrnera?

powered by Misiol

3. Omówić własności numeryczne schematu Hőrnera: złożoność obliczeniową i błędy zaokrąglenia.
4. Omówić metodę wyznaczania za pomocą schematu Hőrnera przedziału, w którym znajduje się pierwiastek wielomianu.
5. Podać definicję funkcji analitycznej. Jak za jej pomocą można określić przybliżoną wartość funkcji i jak ocenić błąd tego przybliżenia?
6. Przedstawić i uzasadnić wzór na iteracyjne obliczanie wartości funkcji. Jakie warunki musi spełniać rozpatrywana funkcja, aby proces iteracyjnego obliczania jej wartości był zbieżny?

3.2.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Napisać program wyznaczający wartości wielomianu za pomocą schematu Hőrnera i za pomocą wzoru ogólnego. Program powinien:
   - porównać szybkości obliczeń obu sposobów wyznaczania wartości wielomianu,
   - wyznaczać dokładność obliczenia wartości w zależności od długości reprezentacji liczb w programie.

   Należy przeprowadzić obliczenia dla przykładów wielomianów podanych przez prowadzącego.
2. Napisać program wyznaczający przedział, w którym zawarte są pierwiastki rzeczywiste wielomianu. W programie należy wykorzystać schemat Hőrnera. Przeprowadzić obliczenia dla przykładów wielomianów podanych przez prowadzącego.
3. Napisać program wyznaczający wartość funkcji $y(x) = e^x$ dla zadanej wartości x, wykorzystujący rozwinięcie w szereg Taylora. Program powinien również szacować błąd obcięcia. Należy tak napisać program, aby można było wyznaczać dokładność obliczeń w funkcji liczby uwzględnionych elementów rozwinięcia.
4. Napisać program iteracyjnego obliczania wartości funkcji $y(x) = \frac{1}{x}$ dla zadanej wartości x i zadanej dokładności. Program powinien być tak napisany, aby można było wyznaczyć liczbę potrzebnych iteracji w funkcji przyjętej dokładności obliczeń.

3.2.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń wszystkich napisanych w trakcie ćwiczenia programów,
- porównanie wyników obliczeń (szybkość, dokładność, zbieżność),
- omówienie wyników i wnioski.

3.3. Metody interpolacji

3.3.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie podstawowych metod interpolacji funkcji oraz oszacowanie analityczne bądź numeryczne błędu interpolacji. Funkcja interpolowana jest znana, co umożliwia dokładniejsze szacowanie błędu interpolacji.

3.3.2. Zestaw pytań kontrolnych

1. Co to jest interpolacja funkcji i kiedy ją stosujemy?
2. Podać twierdzenie o interpolacji $n+1$-punktowej za pomocą wielomianu stopnia co najmniej n.
3. Podać wzór interpolacyjny Lagrange'a i jego modyfikacje dla obliczeń numerycznych.
4. Omówić błąd metody interpolacji funkcji.
5. Co oznacza liniowość i przemienność operatorów różnicowych? Omówić potęgowanie operatorów różnicowych i związki pomiędzy nimi.
6. Omówić pierwszy wzór interpolacyjny Newtona. Co przede wszystkim odróżnia go od wzoru Lagrange'a?
7. Przedstawić modyfikacje wielomianu interpolującego Newtona, a w szczególności drugi wzór interpolujący Newtona.
8. Omówić sposób oszacowania błędu metody przy interpolacji wzorami Newtona.
9. Przedstawić wzory interpolacyjne Newtona stosując rozwinięcia w szereg potęgowy.
10. Co to jest zadanie znajdowania wielomianu algebraicznego najlepiej przybliżającego zero na zadanym przedziale i jakie jest jego rozwiązanie?
11. Kiedy korzystamy z optymalnego wyboru węzłów interpolacji?
12. Omówić wzór interpolacyjny Stirlinga. Jak szacuje się jego błąd?
13. Omówić wzory interpolacyjne Bessela, Everetta i Gausa.
14. Co to jest zjawisko Rungego?

powered by Misiol

3.3.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Napisać program do wyznaczania wartości współczynników wielomianu interpolacyjnego Lagrange'a. Prowadzący podaje współrzędne węzłów oraz postać funkcji interpolowanej. Należy obliczyć odpowiadające wartości funkcji oraz oszacować analitycznie błąd interpolacji za pomocą wielomianu Lagrange'a. Obliczenia powtórzyć dla wszystkich podanych przez prowadzącego przykładów.
2. Napisać program służący do wyznaczania wartości współczynników wielomianu interpolacyjnego, określonego pierwszym wzorem Newtona. Należy porównać dokładność interpolacji z dokładnością uzyskaną w analogicznym zadaniu interpolacji, rozwiązanym za pomocą programu z pkt.1. W tym celu program powinien mieć możliwość wyznaczania wartości wielomianu interpolującego Lagrange'a o współczynnikach wprowadzanych jako dane.
3. Napisać program obliczający wartość funkcji w pobliżu końca przedziału. Program ma wykorzystywać pierwszy wzór interpolacyjny Newtona dla zmiennej znormalizowanej. Postać funkcji, przedział interpolacji, liczbę węzłów oraz wartość argumentu, dla którego wartość funkcji jest obliczana, podaje prowadzący. Dodatkowo należy analitycznie oszacować błąd wyznaczenia wartości funkcji.
4. Napisać analogiczny do pkt. 3 program obliczający wartość funkcji na końcu przedziału interpolacji wykorzystując drugi wzór interpolacyjny Newtona. Wykonać odpowiednie obliczenia analityczne do oszacowania błędu.
5. Napisać program wyznaczający współczynniki wielomianu interpolującego, wykorzystujący wskazany przez prowadzącego wzór interpolujący z operatorem różnicy symetrycznej.

3.3.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń wszystkich napisanych w trakcie ćwiczenia programów,
- porównanie wyników obliczeń pod względem dokładności,
- omówienie wyników i wnioski.

3.4. Metody różniczkowania numerycznego

3.4.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie metod numerycznego wyznaczania wartości pochodnej. Ćwiczenie pozwala porównać dokładność wyznaczania pochodnej metodą różnic zwykłych, wstecznych i centralnych.

3.4.2. Zestaw pytań kontrolnych

1. Sformułować problem różniczkowania numerycznego.
2. Podać wzór Taylora na rozwinięcie funkcji analitycznej.
3. Jak definiuje się operatory różniczkowania?
4. Wyjaśnić, jak oblicza się numerycznie k-tą pochodną korzystając z operatora różnic zwykłych i wstecznych. Jak w tych przypadkach można oszacować błąd różniczkowania?
5. W jaki sposób uzyskuje się wzory Stirlinga na kolejne aproksymacje pochodnej?
6. Omówić szacowanie błędu wzorów Stirlinga.
7. Jak powinny być wybierane metody numerycznego różniczkowania?

3.4.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Napisać program obliczający wartość pochodnej funkcji za pomocą przybliżenia różnicami zwykłymi. Program powinien również obliczać błąd wyznaczenia pochodnej. Prowadzący podaje przykłady funkcji (postacie analityczne), przedziały aproksymacji, wartość argumentu, dla którego obliczana jest pochodna oraz krok. Stanowi to zarazem dane dla programu.
2. Napisać program analogiczny do pkt. 1, ale dla aproksymacji różnicami wstecznymi.
3. Podobnie jak w obu poprzednich punktach napisać program dla aproksymacji wartości różniczki z wykorzystaniem wzoru Stirlinga.

3.4.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń wszystkich napisanych w trakcie ćwiczenia programów,

powered by Misiol

- porównanie wyników obliczeń pod względem dokładności; w szczególności należy porównać dokładność obliczenia pochodnej dla tej samej funkcji i tego samego argumentu ale dla różnych metod,
- omówienie wyników i wnioski.

3.5. Metody całkowania numerycznego

3.5.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie podstawowych metod całkowania numerycznego. W ćwiczeniu zakłada się, że funkcja podcałkowa jest znana. Pozwala to dokładnie badać błędy całkowania, a zarazem dyskutować dokładność metod całkowania oraz sposobów szacowania błędów całkowania.

3.5.2. Zestaw pytań kontrolnych

1. Wyjaśnić konieczność stosowania numerycznych metod całkowania.
2. Co to jest kwadratura?
3. Jak otrzymuje się wzór trapezów? Wyjaśnić interpretację geometryczną metody trapezów.
4. Omówić błąd metody trapezów.
5. Jak otrzymuje się wzór Simpsona? Wyjaśnić interpretację geometryczną metody trapezów.
6. Omówić błąd metody trapezów.
7. Wyjaśnić kwadraturę Czebyszewa?
8. Jaka jest najważniejsza wada metody Czebyszewa?
9. Co to jest kwadratura złożona?
10. Omówić błąd kwadratury złożonej trapezów.
11. Omówić błąd kwadratury złożonej Simpsona.

3.5.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Napisać program obliczający wartość całki oznaczonej, wykorzystujący metodę prostokątów. Prowadzący podaje przykłady funkcji podcałkowej, przedział całkowania, liczbę podziału przedziału całkowania. Program powinien wyznaczać oszacowanie

błędu całkowania. Obliczenia wykonane za pomocą programu powinno się uzupełnić obliczeniami analitycznymi w celu sprawdzenia dokładności obliczeń i szacowania błędu.

2. Napisać analogiczny program do pkt. 1, wykorzystujący jednak kwadraturę trapezów.

3. Napisać analogiczny program do pkt. 1 i 2, wykorzystujący kwadraturę Simpsona.

4. Napisać analogiczny program do punktów poprzednich, wykorzystujący całkowanie w podprzedziałach przedziału całkowania metodę Czebyszewa. Prowadzący podaje liczbę węzłów dla metody.

3.5.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń wszystkich napisanych w trakcie ćwiczenia programów dla wszystkich przykładów,
- porównanie wyników obliczeń pod względem dokładności, w szczególności należy porównać dokładność całkowania dla tych samych przykładów,
- omówienie wyników i wnioski.

3.6. Metody aproksymacji

3.6.1. Cel ćwiczenia

3.6.2. Zestaw pytań kontrolnych

1. Sformułować zadanie aproksymacji.
2. Podać przykłady miar wykorzystywanych dla celów aproksymacji.
3. Omówić aproksymację wielomianami uogólnionymi.
4. Co to jest układ ortogonalny funkcji?
5. Przedstawić przykłady funkcji ortogonalnych.
6. Podać twierdzenie Weierstrasse'a.
7. Omówić aproksymację jednostajną wielomianami Czebyszewa.

powered by Misiol

8. Przedstawić zagadnienie aproksymacji punktowej metodą wielomianów ortogonalnych.
9. Omówić metodę najmniejszych kwadratów.

3.6.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Dla zadanej przez prowadzącego funkcji aproksymowanej, funkcji ważącej, przedziału aproksymacji oraz funkcji aproksymującej w postaci wielomianu uogólnionego obliczyć współczynniki funkcji aproksymującej wykorzystując aproksymację średniokwadratową. W tym celu można wykorzystać programy całkowania numerycznego (z ćwiczenia Metody całkowania numerycznego), napisać własny program, wykorzystać odpowiednie funkcje w środowisku Matlab lub Mathematica, bądź przeprowadzić obliczenia analitycznie. Napisać program obliczający błąd aproksymacji w zadanym przedziale. W tym celu zaproponować miarę błędu i sposób obliczania.
2. Wykonać analogiczne do pkt. 1 zadanie, ale wykorzystać zadane przez prowadzącego wielomiany ortogonalne.
3. Napisać program obliczający aproksymację punktową metodą wielomianów ortogonalnych. Stabelaryzowane wartości funkcji aproksymowanej podaje prowadzący.
4. Napisać program obliczający aproksymację punktową metodą najmniejszych kwadratów. Stabelaryzowane wartości funkcji aproksymowanej podaje prowadzący.

3.6.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń wszystkich napisanych w trakcie ćwiczenia programów oraz obliczeń analitycznych,
- porównanie wyników obliczeń pod względem dokładności aproksymacji różnymi metodami,
- omówienie wyników i wnioski.

3.7. Metody rozwiązywania układów równań liniowych

3.7.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie podstawowych metod rozwiązywania układów równań liniowych: metod dokładnych i przybliżonych. Zadaniem studentów jest również zbadanie szybkości obliczeń w przypadku metod dokładnych oraz dokładności w zależności od liczby iteracji w przypadku metod przybliżonych.

3.7.2. Zestaw pytań kontrolnych

1. Wyjaśnić następujące określenia układu równań liniowych: zgodny, oznaczony, nieoznaczony, niejednorodny, jednorodny, sprzeczny.
2. Podać warunki istnienia rozwiązania układu równań liniowych.
3. Co to jest rząd macierzy?
4. Jak wyznacza się rozwiązania układu równań liniowych, gdy rząd macierzy układu jest mniejszy od wymiaru wektora wyrazów wolnych?
5. Podać wzory Cramera.
6. Jak definiuje się macierz odwrotną i jak się ją oblicza?
7. Jak można wyprowadzić wzory Cramera?
8. Wyjaśnić ideę metody eliminacji Gaussa.
9. Podać założenia dla schematu jedynego dzielenia.
10. Wyjaśnić schemat elementów podstawowych.
11. Przedstawić ideę metod Banachiewicza i Choleskiego.
12. Jak dokonuje się faktoryzacji macierzy układu na odpowiednie macierze trójkątne. Podać założenie dla metody Banachiewicza.
13. Wyjaśnić ideę metod iteracyjnych.
14. Jak wyznacza się kolejne rozwiązania w metodach iteracyjnych: iteracji prostej, metodzie Seidela i metodzie relaksacji?
15. Podać warunki zbieżności metod iteracyjnych.
16. Podać źródła błędów, ich interpretacje i oszacowania.

powered by Misiol

3.7.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Zaprogramować metodę eliminacji Gaussa wykorzystując jeden ze znanych schematów oraz jedną z metod wykorzystujących faktoryzację macierzy układu na iloczyn macierzy trójkątnych. Przy wyborze schematu eliminacji dla metody Gaussa oraz przy wyborze metod z faktoryzacją należy pamiętać o założeniach dla tych metod i tak dobierać rozwiązywane przykłady, aby założenia były spełnione. Dla wybranego przykładu układu równań należy wykonać doświadczenie, sprawdzające czas obliczeń obiema metodami. Doświadczenie można powtórzyć dla kilku przykładów układów równań.
2. Zaprogramować osobno każdą z trzech metod iteracyjnych: metodę iteracji prostej, metodę Seidela i metodę relaksacji. Dobrać przykłady układów równań tak, aby w każdym przypadku uzyskać zbieżność rozwiązania. Dla wybranych przykładów sprawdzić dokładność rozwiązania wykonując tę samą liczbę iteracji każdą metodą i rozpoczynając od tego samego punktu startowego.
3. Wykorzystując gotowe procedury zaprogramować rozwiązywanie układu równań przez odwracanie macierzy układu oraz przez stosowanie wzorów Cramera. Postępując podobnie jak w pierwszym zestawie zadań zbadać czas trwania obliczeń w obu przypadkach.

3.7.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń wykonanych za pomocą napisanych programów,
- porównanie czasów i dokładności obliczeń zgodnie ze specyfikacją zadań do wykonania; odpowiednie wnioski wynikające z tego porównania,
- dla przykładów układów rozwiązywanych metodami iteracyjnymi należy zamieścić sprawdzenie warunków zbieżności oraz oszacowanie liczby koniecznych iteracji dla osiągnięcia zadanej dokładności; oszacowanie takie powinno być uwzględnione w dyskusji dokładności obliczeń metodami iteracyjnymi z punktu poprzedniego.

3.8. Numeryczne obliczanie wartości i wektorów własnych macierzy

3.8.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie wybranych metod obliczania wartości własnych i wektorów własnych danej macierzy. W większości wybranych metod problem nume-

ryczny sprowadza się do rozwiązania układów równań liniowych. W ćwiczeniu proponuje się iteracyjny sposób rozwiązania układów równań metodą iteracji prostej.

3.8.2. Zestaw pytań kontrolnych

1. Co to jest macierz, wyznacznik i równanie charakterystyczne danej macierzy?
2. Jak definiuje się wartości własne i wektory własne macierzy?
3. Wyjaśnić dlaczego wektory własne wyznacza się z dokładnością do stałego czynnika.
4. Jak można znormalizować wektory własne?
5. W jaki sposób w metodzie Kryłowa wyznacza się współczynniki równania charakterystycznego i jaki warunek musi być spełniony, by można było stosować ten sposób?
6. Omówić sposób wyznaczania wektorów własnych w metodzie Kryłowa oraz podać założenia.
7. Podać założenia dla metody iteracyjnego wyznaczania największej co do modułu wartości własnej.
8. Omówić metodę wyznaczania największej co do modułu wartości własnej.
9. Jak można „wygładzić" proces zbieżności metody iteracyjnego wyznaczania największej co do modułu wartości własnej?
10. Jak iteracyjnie wyznacza się wektor własny odpowiadający największej co do modułu wartości własnej?
11. Jakie własności posiadają wartości własne i wektory własne macierzy symetrycznej i dodatnio określonej?
12. Podać definicję i sposób sprawdzania dodatniej określoności macierzy.
13. Omówić metodę wyznaczania wartości własnych i wektorów własnych macierzy symetrycznej.

3.8.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Napisać program wyznaczający współczynniki równania charakterystycznego danej macierzy. Wykorzystać metodę iteracji prostej do rozwiązywania układu równań liniowych (metoda iteracji prostej jest opisana w ćwiczeniu Rozwiązywanie układów równań liniowych). Sprawdzić działanie programu dla różnych wektorów y_0 wykonując za każdym razem tę samą liczbę iteracji.
2. Napisać program wyznaczający wektory własne dla danych współczynników równania charakterystycznego danej macierzy i znanych, różnych wartości własnych tej macierzy.

powered by Misiol

3. Napisać program wyznaczający największą co do modułu wartość własną danej macierzy. Zbadać, jak zachowuje się rozwiązanie pod względem zbieżności i dokonać odpowiednich modyfikacji programu w celu wygładzenia procesu zbiegania się rozwiązania.
4. Napisać program wyznaczający iteracyjnie wartości własne i wektory własne macierzy symetrycznej i dodatnio określonej. Wykorzystać metodę iteracji prostej dla rozwiązywania odpowiednich układów równań liniowych.

3.8.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń wykonanych za pomocą napisanych programów,
- porównanie wyników obliczeń z wartościami dokładnymi, otrzymanymi analitycznie (odpowiednie obliczenia mają być również zamieszczone w sprawozdaniu).

3.9. Metody przybliżonego rozwiązywania równań nieliniowych

3.9.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie wybranych metod przybliżonego rozwiązywania równań nieliniowych i przestępnych, a więc takich, dla których rozwiązanie analityczne jest bardo trudne lub nie istnieje. W metodach wykorzystuje się wzory rekurencyjne wynikające z prostych interpretacji geometrycznych.

3.9.2. Zestaw pytań kontrolnych

1. Podać twierdzenie Bolzano Cauchy'ego.
2. Co to jest przedział izolacji?
3. Podać warunki, jakie musi spełniać funkcja, aby można było zastosować metodę połowienia przedziałów i omówić tę metodę.
4. Podać warunki, jakie musi spełniać funkcja, aby można było zastosować metodę siecznych i omówić tę metodę.
5. Omówić metodę stycznych i podać dla niej założenia.

6. Omówić zagadnienia zbieżności metody połowienia przedziałów, metody siecznych i stycznych.
7. Jakiego typu zadanie rozwiązuje metoda Bernouliego?
8. Omówić metodę Bernouliego. Jakie są jej wady?

3.9.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Znaleźć metodą analityczną pierwiastek równania nieliniowego podanego przez prowadzącego i metodą graficzną określić przedział izolacji. Określić również punkt początkowy dla metod iteracyjnych.
2. Napisać program znajdujący miejsca zerowe dla rozpatrywanego równania realizujący metody: połowienia przedziałów, metody siecznych i stycznych oraz metody Bernouliego. Dla metod połowienia, siecznych i stycznych należy uwzględnić zadaną dokładność i określić liczbę iteracji. Dla metody Bernouliego należy przyjąć liczbę iteracji.
3. Zbadać czas obliczeń dla tego samego przykładu, tego samego punktu startowego oraz tej samej liczby iteracji. Która z metod daje dokładniejsze wyniki?
4. Dla metod: połowienia, siecznych i stycznych zbadać czy teoretyczna dokładność pokrywa się z uzyskaną doświadczalnie.

3.9.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń analitycznych przeprowadzonych w celu określenia pierwiastka i przedziału izolacji dla badanego równania, rysunki ilustrujące rozwiązanie,
- porównanie wyników obliczeń pod względem czasu obliczeń i dokładności.

3.10. Metody całkowania równań różniczkowych zwyczajnych

3.10.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie wybranych metod przybliżonego rozwiązywania równań różniczkowych zwyczajnych: metody Eulera i Rungego-Kutty. W ramach ćwiczenia bada się metody pod względem dokładności i złożoności obliczeniowej. Ponadto

powered by Misiol

dokonuje się porównania z metodami pozwalającymi znajdować analityczne przybliżenie funkcji, stanowiącej rozwiązanie równania: metodą szeregu Taylora i metodą Picarda.

3.10.2. Zestaw pytań kontrolnych

1. Sformułować warunki, jakie musi spełniać równanie różniczkowe pierwszego rzędu, aby posiadało dokładnie jedno rozwiązanie.
2. Przedstawić i uzasadnić podstawowe zależności metody Eulera.
3. Podać i omówić wady metody Eulera.
4. Omówić ideę metody Rungego-Kutty.
5. Przedstawić sposób uzmiennienia kroku całkowania w metodzie Rungego-Kutty.
6. Omówić uogólnienie metody Rungego-Kutty na układy równań różniczkowych pierwszego rzędu.
7. Jak należy stosować metodę Rungego-Kutty do rozwiązywania równań różniczkowych wyższych rzędów?
8. Jakiego rodzaju równania różniczkowe rozwiązuje się metodą szeregów Taylora?
9. Omówić metodę szeregów Taylora.
10. Jaki typ równań różniczkowych rozwiązuje się metodą Picarda?
11. Omówić metodę Pickarda i przedstawić jej zalety i wady.

3.10.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Napisać program dla metody Eulera. Dla podanego przez prowadzącego równania różniczkowego, warunku początkowego i przedziału całkowania znaleźć metodą Eulera krzywą podcałkową dla kilku wybranych wartości kroku całkowania. Rozwiązać analitycznie podane równanie i obliczyć błąd całkowania (np. jako sumę kwadratów błędów co krok całkowania).
2. Napisać program dla metody Rungego-Kutty. Dla podanego przez prowadzącego równania różniczkowego, warunku początkowego i przedziału całkowania znaleźć metodą Rungego-Kutty krzywą podcałkową dla kilku wybranych wartości kroku całkowania. Rozwiązać analitycznie podane równanie i obliczyć błąd całkowania (np. jako sumę kwadratów błędów co krok całkowania).
3. Uzupełnić program z pkt.2 uzmiennieniem kroku całkowania. Wykonać obliczenia dla tego samego przykładu jak w pkt.2 znajdując jednocześnie liczbę iteracji (kroków całkowania). Porównać dokładność całkowania i liczbę kroków z dokładnością i liczbą kroków uzyskaną w programie z pkt.2.

4. Wyznaczyć przybliżone rozwiązanie równania z poprzednich punktów za pomocą metody szeregu Taylora. Porównać dokładność tego rozwiązania z dokładnością rozwiązań uzyskanych metodą Eulera i Rungego-Kutty. Zbadać dokładność rozwiązania metodą szeregu Taylora w zależności od liczby uwzględnionych elementów rozkładu w szereg.

5. Wyznaczyć przybliżone rozwiązanie równania z poprzednich punktów za pomocą metody Picarda. Porównać dokładność tego rozwiązania z rozwiązaniami uzyskanymi metodą Eulera i Rungego-Kutty. Zbadać dokładność rozwiązania metodą Picarda w zależności od liczby iteracji.

3.10.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- specyfikacje zadań wykonywanych w trakcie ćwiczenia,
- teksty wszystkich napisanych programów,
- wyniki obliczeń programów dla wybranego przykładu równania różniczkowego, wyniki badań dokładności poszczególnych metod oraz ich nakład obliczeniowy,
- wyniki obliczeń analitycznych, tj. rozwiązanie analityczne wybranego równania różniczkowego, rozwiązania przybliżonego metodą szeregu Taylora i metodą Picarda,
- porównanie wyników obliczeń pod względem dokładności.

3.11. Metody całkowania równań różniczkowych cząstkowych typu parabolicznego

3.11.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie metody siatek do rozwiązywania równań cząstkowych typu parabolicznego. Za pomocą programu rozwiązywane są przykłady, dla których znane są rozwiązania analityczne. Pozwala to zbadać dokładność metody numerycznej w funkcji wybieranych parametrów.

3.11.2. Zestaw pytań kontrolnych

1. Zdefiniować równanie cząstkowe typu parabolicznego.
2. Wyjaśnić pojęcia: warunki początkowe, brzegowe i warunki zgodności.
3. Wyjaśnić ideę metody siatek.
4. Sformułować algorytm dla metody siatek.

powered by Misiol

5. Omówić problemy szacowania błędu metody siatek.
6. W jakim przypadku można poszukiwać analitycznego rozwiązania równania różniczkowego cząstkowego typu parabolicznego?

3.11.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Zaprogramować metodę siatek.
2. Znaleźć analityczne rozwiązanie równań podanych przez prowadzącego.
3. Obliczyć za pomocą programu rozwiązanie przybliżone dla wybranego zestawu parametrów metody siatek.
4. Obliczyć wskaźnik błędu (np. suma kwadratów różnic w węzłach siatki) rozwiązania przybliżonego w zależności od parametrów metody siatek.

3.11.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- tekst programu dla metody siatek,
- wyniki obliczeń programu dla wybranego przykładu równania dla wszystkich przypadków parametrów metody siatek,
- obliczenia analityczne rozwiązania wybranego równania,
- porównanie wyników obliczeń numerycznych pod względem dokładności.

3.12. Równania całkowe typu Fredholma i Volterry

3.12.1. Cel ćwiczenia

Celem ćwiczenia jest zaprogramowanie metody Neumanna, służącej do rozwiązywania równań całkowych typu Fredholma i Volterry. Rozwiązywane przykłady są tak dobrane, aby poprzez znalezienie ich rozwiązania analitycznego można było zbadać dokładność metody Neumanna w zależności od wybranych parametrów i funkcji początkowej.

3.12.2. Zestaw pytań kontrolnych

1. Zdefiniować równanie całkowe typu Fredholma.
2. Podać warunki istnienia jednego rozwiązania równania całkowego typu Fredholma.
3. Jaką postać posiada ogólne rozwiązanie równania całkowego typu Fredholma?

4. Kiedy równanie całkowe typu Fredholma sprowadza się do równania całkowego Volterry II rodzaju?
5. Kiedy i w jaki sposób można rozwiązać analitycznie równanie Volterry stosując transformatę Laplace'a?
6. Omówić związek równań całkowych Volterry z równaniami różniczkowymi zwyczajnymi liniowymi pierwszego rzędu ze zmiennymi współczynnikami.
7. Wyjaśnić ideę metody Neumanna.
8. Na czym polega różnica pomiędzy dużym i małym cyklem metody Neumanna?
9. Omówić zbieżność metody Neumanna.

3.12.3. Tematy zadań do wykonania w trakcie ćwiczenia

1. Zaprogramować metodę Neumanna. Jako dane wejściowe do programu należy wprowadzać stabelaryzowane wartości jądra, wyrazu wolnego i funkcji początkowej. Ponadto należy wprowadzić pozostałe parametry: stałą λ, przedział całkowania, krok całkowania dla małych cykli i liczbę dużych cykli.
2. Rozwiązać analitycznie równanie całkowe podane przez prowadzącego.
3. Dokonać tabelaryzacji odpowiednich funkcji równania z pkt.2 i znaleźć rozwiązanie numerycznie za pomocą napisanego programu. Obliczenia należy powtórzyć wybierając różną gęstość tabelaryzacji i różną liczbę dużych cykli. Obliczenia należy powtórzyć dla różnych funkcji początkowych.
4. Obliczyć dokładność wyznaczenia rozwiązania numerycznego porównując je z rozwiązaniem analitycznym.

3.12.4. Uwagi dotyczące sprawozdania

Sprawozdanie powinno zawierać:

- tekst programu dla metody Neumanna,
- wyniki obliczeń programu dla wybranego przykładu równania dla wszystkich przypadków wybranych parametrów,
- obliczenia analityczne rozwiązania wybranego równania,
- porównanie wyników obliczeń numerycznych pod względem dokładności.

powered by Misiol

Literatura

1. *Demidowicz B.P., Maron I.A.* Metody numeryczne. Cz. 1, PWN, Warszawa 1965.
2. *Demidowicz B.P., Maron I.A., Szuwałowa E.J.* Metody numeryczne. Cz. 2, PWN, Warszawa 1965.
3. *Dryja M., Jankowscy J. i M.* Przegląd metod i algorytmów numerycznych. WNT, Warszawa 1982.
4. *Fortuna Z., Macukow B., Wąsowski J.* Metody numeryczne. WNT. Warszawa 1983.
5. *Klamka J. i inni*, Laboratorium metod numerycznych. Skrypt Politechniki Śląskiej nr 1790, Gliwice 1993.
6. *Zaguskin W.L.* Przegląd metod numerycznych rozwiązywania równań. PWN, Warszawa 1965.
7. *Rumszyski L.Z.* Ćwiczenia praktyczne z metod numerycznych. PWN, Warszawa 1976.
8. *Sendow B.* Stare i nowe w metodach numerycznych. PWN, Warszawa 1976.
9. *Ralston A.* Wstęp do analizy numerycznej. PWN, Warszawa 1971.
10. *Piskorek A.* Równania całkowe. Elementy teorii i zastosowanie. PWN, Warszawa 1971.