Kartkówka

Maciej Grzybacz

16 maja 2024

Numeryczna reprezentacja liczb rzeczywistych i arytmetyka zmiennoprzecinkowa

Numeryczna reprezentacja liczb rzeczywistych w komputerach wykorzystuje system zmiennoprzecinkowy, który jest określony przez kilka parametrów:

- β: podstawa,
- t: dokładność (liczba cyfr znaczących w mantysie),
- L, U: zakres wykładnika.

Liczba zmiennoprzecinkowa $x \in F$ jest reprezentowana w postaci:

$$x = \pm \left(d_1\beta^{-1} + d_2\beta^{-2} + \ldots + d_t\beta^{-t}\right)\beta^e$$

gdzie d_i to cyfry mantysy, a e to wykładnik. System ten jest unormowany, gdy $d_1 \neq 0$ dla $x \neq 0$.

Właściwości reprezentacji zmiennoprzecinkowej

1. **Skończoność**: Zbiór liczb zmiennoprzecinkowych F jest skończony, co oznacza, że nie może reprezentować wszystkich liczb rzeczywistych. Liczba elementów F jest dana wzorem:

$$2 \cdot (\beta - 1) \cdot \beta^{t-1} \cdot (U - L + 1) + 1$$

2. **Nierównomierne rozłożenie**: Elementy F nie są równomiernie rozłożone na osi liczbowej. Każdy element reprezentuje przedział liczb rzeczywistych, a względna dokładność reprezentacji jest szacowana jako β^{1-t}.

3. **Błędy zaokrąglenia**: Ze względu na skończoną liczbę cyfr mantysy, liczby zmiennoprzecinkowe są przybliżeniami rzeczywistych wartości, co wprowadza błędy zaokrąglenia. Błąd ten można oszacować jako 2^{-t}.

Problemy numeryczne w arytmetyce zmiennoprzecinkowej

1. **Nadmiar i niedomiar**: Nadmiar (overflow) występuje, gdy wartość bezwzględna liczby jest za duża, aby ją reprezentować. Niedomiar (underflow) występuje, gdy wartość jest za mała.

2. **Błędy obcięcia**: Występują, gdy nieskończone ciągi obliczeń muszą być zakończone po skończonej liczbie kroków, np. ograniczenie szeregu nieskończonego do skończonej liczby składników.

3. **Porównania liczb**: Porównywanie liczb zmiennoprzecinkowych wymaga sprawdzania, czy błąd między wartościami jest mniejszy od zadanego ϵ, zamiast bezpośredniego porównywania wartości.

Standaryzacja i operacje

Reprezentacja zmiennoprzecinkowa została ustandaryzowana w standardzie IEEE 754, który definiuje dokładne zasady reprezentacji i operacji na liczbach zmiennoprzecinkowych. Operacje arytmetyczne na tych liczbach muszą być implementowane tak, jakby działanie było wykonywane dokładnie, a tylko wynik był reprezentowany w zbiorze liczb maszynowych.

Implementacja w algorytmach numerycznych

1. **Poprawność numeryczna**: Algorytmy numeryczne muszą być zaprojektowane tak, aby były numerycznie poprawne, co oznacza, że wynik powinien być bliski rzeczywistego rozwiązania problemu przy niewielkim błędzie względnym. Osiąga się to poprzez minimalizację błędów zaokrągleń i stabilność numeryczną.

2. **Stabilność numeryczna**: Algorytmy stabilne numerycznie gwarantują, że małe błędy na dowolnym etapie obliczeń nie będą narastały w kolejnych krokach. Stabilność jest minimalnym wymogiem dla algorytmu, podczas gdy poprawność numeryczna jest najwyższym standardem.

3. **Uwarunkowanie zadania**: Jest to czułość rozwiązania zadania na zaburzenia danych wejściowych. Zadanie jest dobrze uwarunkowane, jeśli niewielkie zmiany w danych powodują proporcjonalnie niewielkie zmiany w rozwiązaniu. Algorytmy dla zadań źle uwarunkowanych muszą być szczególnie starannie projektowane, aby zminimalizować wpływ tych zaburzeń.

W opracowywaniu algorytmów numerycznych konieczne jest uwzględnienie powyższych własności reprezentacji numerycznej i arytmetyki zmiennoprzecinkowej, aby zapewnić dokładność, stabilność i efektywność obliczeń.

Własności numeryczne operacji zmiennoprzecinkowych

Operacje arytmetyczne na reprezentacjach liczb rzeczywistych muszą być implementowane tak, jakby działanie było wykonywane dokładnie, a tylko wynik był reprezentowany w zbiorze liczb maszynowych:

$$x \oplus y = \mathrm{fl}(x + y) \quad \text{dla} \quad x + y \,\text{z zakresu}\, F$$

Problemy związane z operacjami zmiennoprzecinkowymi:

1. **Gęstość elementów**: Ze względu na ograniczoną precyzję, dodawanie dwóch liczb może prowadzić do utraty dokładności. Przykładowo, jeśli dodajemy liczby bardzo odległe od siebie, ta mniejsza może "zniknąć", mimo że przed dodawaniem była reprezentowalna:

$$a + b = 2^{c_a} \cdot \left(m_a + m_b \cdot 2^{-(c_a - c_b)} \right)$$

2. **Overflow i underflow**: Przy mnożeniu liczb zmiennoprzecinkowych rzadko zdarza się, że wynik również jest liczbą zmiennoprzecinkową, ponieważ:

$$x \cdot y \quad \text{ma} \quad 2t \quad \text{lub} \quad 2t - 1 \quad \text{cyfr znaczących}$$

   Overflow jest bardziej prawdopodobny, gdy wynik jest za duży, aby go reprezentować, a underflow, gdy wynik jest za mały.

3. **Nieprzemienność operacji**: Operacje zmiennoprzecinkowe, takie jak dodawanie ($\oplus$) i mnożenie ($\odot$), są przemienne, ale nie są łączne ani rozdzielne:

$$(a \oplus b) \oplus c \neq a \oplus (b \oplus c)$$
$$a \odot (b \oplus c) \neq (a \odot b) \oplus (a \odot c)$$

Zaokrąglenia

Operacje arytmetyczne są implementowane tak, aby ich wyniki były jak najbliższe dokładnym wynikom. Zaokrąglony wynik można wyrazić jako:

$$\mathrm{fl}(a * b) = (a * b) \cdot (1 + \epsilon)$$

gdzie $* = +, -, \cdot, /$ oraz ϵ jest małym błędem zaokrąglenia zależnym od dokładności reprezentacji.

Maszynowe ϵ

Maszynowe ϵ to najmniejsza liczba zmiennoprzecinkowa, dla której:

$$1 \oplus \epsilon > 1$$

Wartość maszynowego ϵ określa precyzję obliczeń numerycznych wykonywanych na liczbach zmiennoprzecinkowych. Przykładowo, dla podwójnej precyzji (double precision), $\epsilon \approx 2.22 \times 10^{-16}$.

Problemy przybliżeń

Przybliżenia w operacjach zmiennoprzecinkowych mogą prowadzić do znaczących błędów w algorytmach numerycznych. Przykład:

$$\mathrm{fl}(A(\vec{a},\vec{b})) = [a_1 \cdot b_1 \cdot (1+\epsilon_1) + a_2 \cdot b_2 \cdot (1+\epsilon_2)] \cdot (1+\delta_2)$$

Stabilność numeryczna

Algorytmy numerycznie poprawne są najwyższej jakości, jednak udowodnienie ich numerycznej poprawności jest często trudne i wymaga znalezienia wskaźników kumulacji niezależnych od danych. Stabilność numeryczna jest minimalnym wymogiem dla algorytmu i oznacza, że małe błędy w danych wejściowych nie prowadzą do znacznych błędów w wynikach końcowych.

1. **Catastrophic cancellation**: Odejmowanie bliskich liczb może prowadzić do utraty dokładności, np.:
   $e^{-5.5} = 1 - 5.5 + 15.125 - 27.730 + 38.129 - 41.942 + 38.446 - 30.208 + 20.768 - 12.692 + 6.9803 - 3.4902 + 1.5997 + \ldots \approx 0.00408677$

2. **Zmiana algorytmu**: W celu poprawy stabilności numerycznej można zmieniać algorytmy, np.:

$$e^{-5.5} = \frac{1}{e^{5.5}} = \frac{1}{1 + 5.5 + 15.125 + \ldots} \approx 0.0040865$$

Podsumowując, operacje zmiennoprzecinkowe mają wiele specyficznych właściwości numerycznych, które muszą być uwzględniane przy projektowaniu algorytmów numerycznych, aby zapewnić ich poprawność i stabilność.

Zadanie, algorytm i realizacja zmiennoprzecinkowa algorytmu

Definicje

Zadanie

Zadanie dla danych $\vec{d} = (d_1, d_2, \ldots, d_n) \in \mathbb{R}^d$ polega na znalezieniu wyniku $\vec{w} = (w_1, w_2, \ldots, w_m) \in \mathbb{R}^w$ takiego, że $\vec{w} = \varphi(\vec{d})$, gdzie $\mathbb{R}^d$ i $\mathbb{R}^w$ są skończenie wymiarowymi, unormowanymi przestrzeniami kartezjańskimi, a $\varphi : D_0 \subset \mathbb{R}^d \to \mathbb{R}^w$ jest odwzorowaniem ciągłym.

Algorytm

Algorytm A w klasie zadań $\{\varphi, D\}$ jest sposobem wyznaczenia wyniku $\vec{w} = \varphi(\vec{d})$ dla $\vec{d} \in D \subset D_0$, z dokładną realizacją działań, tj. w zwykłej arytmetyce.

Realizacja zmiennoprzecinkowa algorytmu

Realizacja algorytmu w arytmetyce zmiennoprzecinkowej oznacza zastąpienie działań arytmetycznych na danych rzeczywistych odpowiednikami w arytmetyce zmiennoprzecinkowej. Oczekujemy, że zarówno dane $\vec{d}$, jak i wyniki $\vec{w}$ będą reprezentowane z małymi błędami:

$$||\vec{d} - rd(\vec{d})|| \leq \epsilon_d ||\vec{d}||$$

$$||\vec{w} - rd(\vec{w})|| \leq \epsilon_w ||\vec{w}||$$

gdzie ϵ_d, ϵ_w są małe i wynoszą około $10 \cdot \beta^{1-t}$.

Przykłady

Przykład zadania

Rozważmy zadanie obliczenia wartości funkcji $\varphi(x) = \sin(x)$ dla $x \in \mathbb{R}$. Tutaj $\vec{d} = x$, $\vec{w} = \sin(x)$, $\mathbb{R}^d = \mathbb{R}$ oraz $\mathbb{R}^w = \mathbb{R}$.

Przykład algorytmu

Algorytm obliczania $\sin(x)$ może być implementowany poprzez rozwinięcie funkcji w szereg Taylora:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

Algorytm A oblicza $\sin(x)$ dla $x \in \mathbb{R}$, wykonując skończoną liczbę operacji dodawania, odejmowania, mnożenia i dzielenia.

Przykład realizacji zmiennoprzecinkowej algorytmu

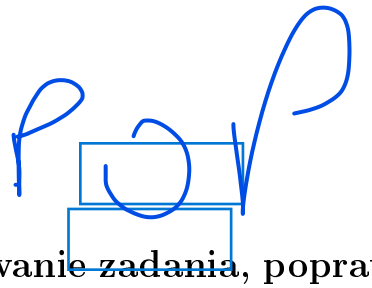
W realizacji zmiennoprzecinkowej algorytmu powyższe operacje arytmetyczne są wykonywane w arytmetyce zmiennoprzecinkowej:

$$\mathrm{fl}(x - \frac{x^3}{3!}) + \mathrm{fl}(\frac{x^5}{5!}) - \mathrm{fl}(\frac{x^7}{7!}) + \dots$$

gdzie $\mathrm{fl}(x)$ oznacza zmiennoprzecinkową reprezentację liczby x. Przy każdej operacji może dochodzić do błędów zaokrąglenia, co powoduje, że wynik jest tylko przybliżeniem rzeczywistej wartości $\sin(x)$.

Uwagi

- **Precyzja**: Realizacja zmiennoprzecinkowa wprowadza błędy zaokrąglenia, których wielkość zależy od precyzji reprezentacji (np. single precision, double precision).
- **Stabilność numeryczna**: Algorytmy muszą być projektowane tak, aby minimalizować wpływ błędów zaokrąglenia i zapewnić stabilność numeryczną, co oznacza, że małe błędy na wejściu powodują małe błędy na wyjściu.
- **Uwarunkowanie zadania**: Czułość zadania na zaburzenia danych wejściowych. Zadanie jest dobrze uwarunkowane, jeśli niewielkie zmiany w danych powodują niewielkie zmiany w wyniku.



Uwarunkowanie zadania, poprawność numeryczna algorytmu, stabilność numeryczna algorytmu

Uwarunkowanie zadania

Definicja

Uwarunkowanie zadania to czułość rozwiązania na zaburzenia danych wejściowych. Wskaźnikiem uwarunkowania zadania jest współczynnik określający, jak względny błąd danych wpływa na względny błąd rozwiązania. Zadanie nazywamy źle uwarunkowanym, jeśli niewielkie względne zmiany danych powodują duże względne zmiany wyniku.

Przykład

Rozważmy iloczyn skalarny wektorów $\vec{x}$ i $\vec{y}$:

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i$$

Jeśli dane x_i i y_i są zaburzone odpowiednio o α_i i β_i, to względny błąd wyniku można oszacować jako:

$$\left| \frac{\sum_{i=1}^{n} x_i(1+\alpha_i)y_i(1+\beta_i) - \sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i y_i} \right| \approx \max |\alpha_i + \beta_i|$$

Jeśli wszystkie x_i i y_i mają ten sam znak, to $cond(\vec{x} \cdot \vec{y}) = 1$.

Poprawność numeryczna algorytmu

Definicja

Algorytm numerycznie poprawny to algorytm, który dla każdej dostatecznie silnej arytmetyki, dla danych $\vec{d} \in D$, gwarantuje istnienie takich zaburzonych danych $\vec{d'}$, że:

$$\|\vec{d} - \vec{d'}\| \leq \epsilon_d \|\vec{d}\|$$

oraz

$$\|\varphi(\vec{d'}) - fl(A(\vec{d}))\| \leq \epsilon_w \|\varphi(\vec{d'})\|$$

gdzie ϵ_d i ϵ_w są małymi wartościami.

Przykład

Iloczyn skalarny wektorów $\vec{a}$ i $\vec{b}$:

$$A(\vec{a}, \vec{b}) = \sum_{i=1}^{n} a_i b_i$$

Realizacja zmiennoprzecinkowa tego algorytmu:

$$fl(A(\vec{a},\vec{b})) = \sum_{i=1}^{n} a_i(1+\alpha_i)b_i(1+\beta_i)(1+\epsilon_i)$$

Dla pewnych α_i, β_i, ϵ_i wyniki będą zaburzeniami dokładnych wartości, co oznacza, że algorytm jest numerycznie poprawny.

Stabilność numeryczna algorytmu

Definicja

Algorytm jest numerycznie stabilny, jeśli małe błędy na dowolnym etapie obliczeń prowadzą do małych błędów w końcowym wyniku. Formalnie, algorytm A nazywamy numerycznie stabilnym w klasie zadań $\{\varphi, D\}$, jeśli istnieje stała K taka, że dla każdego $\vec{d} \in D$:

$$\|\varphi(\vec{d}) - fl(A(\vec{d}))\| \leq K \cdot P(\vec{d},\varphi)$$

gdzie $P(\vec{d},\varphi)$ to optymalny poziom błędu rozwiązania.

Przykład

Rozważmy funkcję e^x rozwiniętą w szereg Taylora:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

Obliczenia dla $x = -5.5$ mogą prowadzić do błędów wynikających z zaokrągleń (catastrophic cancellation), ale zmieniając algorytm na:

$$e^{-5.5} = \frac{1}{e^{5.5}} = \frac{1}{1 + 5.5 + 15.125 + \dots}$$

możemy uzyskać bardziej stabilne numerycznie wyniki.

Ilorazy różnicowe i ich związek z pochodnymi

Ilorazy różnicowe

Definicja

Ilorazy różnicowe są narzędziem stosowanym w numerycznych metodach interpolacyjnych do wyznaczania wartości wielomianów interpolacyjnych. Dla funkcji f i zbioru punktów $x_0, x_1, \ldots, x_k$, k-ty iloraz różnicowy jest zdefiniowany rekurencyjnie:

$$f[x_i] = f(x_i)$$

$$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

Przykład

Rozważmy funkcję f w punktach x_0, x_1, x_2:

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$f[x_1, x_2] = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

Zastosowanie ilorazów różnicowych

Ilorazy różnicowe są podstawą interpolacyjnego wzoru Newtona:

$$P_n(x) = f[x_0] + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \ldots + (x - x_0)(x - x_1)\ldots(x - x_{n-1})f[x_0, x_1, \ldots, x_n]$$

Przykład

Dla trzech punktów x_0, x_1, x_2 wielomian interpolacyjny ma postać:

$$P_2(x) = f[x_0] + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2]$$

Związek z pochodnymi

Ilorazy różnicowe są związane z pochodnymi funkcji poprzez uogólnione twierdzenie o wartości średniej. Założenia:

$$f \in C^n[a, b]$$

$$x_0, x_1, \ldots, x_n \in [a, b]$$

Teza:

$$\exists \eta \in (a, b) \quad f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\eta)}{n!}$$

Przykład

Dla funkcji f i punktów x_0, x_1, x_2:

$$f[x_0, x_1, x_2] = \frac{f^{(2)}(\eta)}{2!} \quad \text{dla} \quad \eta \in (x_0, x_2)$$

Podsumowując, ilorazy różnicowe są użytecznym narzędziem do budowy wielomianów interpolacyjnych i mają bezpośredni związek z pochodnymi funkcji, co umożliwia lepsze zrozumienie właściwości funkcji na podstawie jej wartości w kilku punktach.

Porównanie interpolacji Lagrange'a i Hermite'a

Interpolacja jest metodą aproksymacji funkcji przy użyciu wielomianów przechodzących przez dane punkty. Dwie popularne metody interpolacji to interpolacja Lagrange'a i interpolacja Hermite'a. Poniżej przedstawiono definicje i porównanie obu metod.

Interpolacja Lagrange'a

Definicja

Interpolacja Lagrange'a polega na znalezieniu wielomianu $P_n(x)$ stopnia n, który przechodzi przez $n+1$ zadanych punktów $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$. Wielomian interpolacyjny Lagrange'a można zapisać w postaci:

$$P_n(x) = \sum_{k=0}^{n} y_k L_k(x)$$

gdzie $L_k(x)$ to wielomiany bazowe Lagrange'a:

$$L_k(x) = \prod_{\substack{0 \le j \le n \\ j \neq k}} \frac{x - x_j}{x_k - x_j}$$

Cechy

- **Łatwość utworzenia wzoru**: Wzór interpolacyjny Lagrange'a jest prosty do utworzenia.
- **Ponowne obliczanie**: Dodanie nowego węzła interpolacji wymaga ponownego wyliczenia wielomianu od początku.
- **Obliczanie wartości**: Obliczanie wartości wielomianu w konkretnym punkcie wymaga ponownego obliczenia liczników dla $L_k(x)$.
- **Błąd interpolacji**: Błąd interpolacji można oszacować za pomocą formuły:

  $$f(x) = P_n(x) + \frac{f^{(n+1)}(\eta)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

  gdzie $\eta \in (a, b)$.

Interpolacja Hermite'a

Definicja

Interpolacja Hermite'a rozszerza interpolację Lagrange'a poprzez uwzględnienie wartości pochodnych funkcji w węzłach interpolacji. Dla $k+1$ różnych węzłów

$x_0, x_1, \ldots, x_k$ i ich krotności $m_0, m_1, \ldots, m_k$ (gdzie $\sum_{i=0}^{k} m_i = n+1$), wielomian Hermite'a $H_n(x)$ stopnia $\leq n$ spełnia:

$$H_n^{(j)}(x_i) = f^{(j)}(x_i) \quad \text{dla} \quad i = 0, 1, \ldots, k \quad \text{oraz} \quad j = 0, 1, \ldots, m_i$$

Cechy

- **Uwzględnienie pochodnych**: Interpolacja Hermite'a uwzględnia wartości pochodnych w węzłach, co pozwala na dokładniejsze aproksymacje.
- **Krotność węzłów**: Krotność węzłów określa, ile pochodnych ma być równych w danym węźle. Gdy $m_i = 1$, interpolacja Hermite'a sprowadza się do interpolacji Lagrange'a.
- **Tablica ilorazów różnicowych**: Współczynniki wielomianu Hermite'a można znaleźć, tworząc tablicę ilorazów różnicowych, podobnie jak w metodzie Newtona.

Porównanie interpolacji Lagrange'a i Hermite'a

- **Złożoność obliczeniowa**: Interpolacja Hermite'a jest bardziej złożona obliczeniowo niż interpolacja Lagrange'a, ponieważ wymaga uwzględnienia pochodnych i krotności węzłów.
- **Dokładność**: Interpolacja Hermite'a jest zazwyczaj bardziej dokładna niż interpolacja Lagrange'a, szczególnie w przypadkach, gdy dostępne są informacje o pochodnych funkcji.
- **Elastyczność**: Interpolacja Hermite'a oferuje większą elastyczność w modelowaniu funkcji, ponieważ można kontrolować zarówno wartości funkcji, jak i jej pochodne w węzłach.
- **Stosowalność**: Interpolacja Lagrange'a jest prostsza i szybsza do zastosowania, gdy dostępne są tylko wartości funkcji, podczas gdy interpolacja Hermite'a jest preferowana, gdy dostępne są również pochodne.

Podsumowując, wybór między interpolacją Lagrange'a a interpolacją Hermite'a zależy od dostępnych danych i wymagań dotyczących dokładności oraz elastyczności aproksymacji funkcji .

Efekt Rungego

Definicja

Efekt Rungego występuje podczas interpolacji wielomianowej, gdy liczba węzłów interpolacji jest zwiększana, prowadząc do znacznych oscylacji wielomianu interpolacyjnego w pobliżu brzegów przedziału interpolacji. Problem ten został opisany przez Carla Rungego w 1901 roku.

Przyczyny

Efekt Rungego jest szczególnie wyraźny, gdy:

- Interpolujemy funkcję wielomianami wysokiego stopnia.
- Węzły interpolacyjne są równoodległe.

Przykład

Rozważmy funkcję:

$$f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, 1]$$

Interpolacja tej funkcji wielomianami wysokiego stopnia w równoodległych węzłach prowadzi do znacznych oscylacji wielomianu interpolacyjnego na brzegach przedziału $[0.726, 1]$ oraz $[-1, -0.726]$, co skutkuje zmniejszeniem dokładności interpolacji na tych odcinkach.

Rady praktyczne

Aby zmniejszyć efekt Rungego, można:

- Rozpoczynać od interpolacji liniowej i stopniowo zwiększać liczbę węzłów oraz stopień wielomianu.
- Stosować interpolację funkcjami sklejanymi.
- Używać specjalnych węzłów interpolacyjnych, takich jak węzły Czebyszewa, które są gęściej rozmieszczone na brzegach przedziału interpolacji.

Funkcje sklejane

Definicja

Funkcję $s(x)$ określoną na przedziale $[a, b]$ nazywamy funkcją sklejaną stopnia $m \geq 1$, jeżeli:

- $s(x)$ jest wielomianem stopnia $\leq m$ na każdym podprzedziale $[x_i, x_{i+1}]$.
- $s(x) \in C^{m-1}[a, b]$.
- Δ_n jest podziałem $[a, b]$ na $n-1$ podprzedziałów przez węzły: $a = x_1 < x_2 < \ldots < x_i < \ldots < x_n = b$.

Zastosowanie funkcji sklejanych

Funkcje sklejane są przydatne, gdy:

- Potrzebna jest dokładna i stabilna interpolacja danych.
- Interpolacja ma być gładka, a funkcja ciągła wraz ze swoimi pochodnymi do rzędu $m-1$.

Przykład

Najczęściej stosowaną funkcją sklejaną jest funkcja sześcienna (cubic spline):

$$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \quad \text{dla} \quad x \in [x_i, x_{i+1}]$$

Funkcje te są często stosowane w praktyce ze względu na ich dobrą dokładność i łatwość wyznaczania wartości interpolowanych oraz pochodnych.

Zalety funkcji sklejanych

- Dokładniejsza i bezpieczniejsza interpolacja niż w przypadku interpolacji wielomianowej wysokiego stopnia.
- Funkcje sklejane sześcienne ($s^3(x)$) są często wystarczające w praktyce.
- Przydatne dla węzłów równoodległych i pozwalają na łatwe wyznaczanie wartości pochodnych oraz całek funkcji.

Podsumowując, funkcje sklejane są użytecznym narzędziem w numerycznych metodach interpolacyjnych, zapewniającym dokładność i stabilność obliczeń, szczególnie w przypadkach, gdy interpolacja wielomianowa prowadzi do problemów takich jak efekt Rungego.

Porównanie interpolacji wielomianowej i funkcjami sklejanymi

Interpolacja jest metodą aproksymacji funkcji, w której nowa funkcja przechodzi przez zadane punkty. Dwie popularne metody interpolacji to interpolacja wielomianowa i interpolacja funkcjami sklejanymi.

Interpolacja wielomianowa

Interpolacja wielomianowa polega na znalezieniu wielomianu $P_n(x)$ stopnia n, który przechodzi przez $n+1$ zadanych punktów. Przykładem jest interpolacja Lagrange'a:

$$P_n(x) = \sum_{k=0}^{n} y_k L_k(x)$$

gdzie $L_k(x)$ to wielomiany bazowe Lagrange'a:

$$L_k(x) = \prod_{\substack{0 \le j \le n \\ j \neq k}} \frac{x - x_j}{x_k - x_j}$$

Interpolacja funkcjami sklejanymi

Interpolacja funkcjami sklejanymi polega na użyciu kawałków wielomianów, które są łączone w sposób ciągły. Najczęściej stosowaną jest interpolacja sześcienna (cubic spline). Funkcja sklejana $s(x)$ jest zdefiniowana na przedziale $[a, b]$ i spełnia następujące warunki:

- $s(x)$ jest wielomianem stopnia ≤ 3 na każdym podprzedziale $[x_i, x_{i+1}]$.
- $s(x) \in C^2[a, b]$, co oznacza, że funkcja jest ciągła wraz ze swoimi pierwszymi i drugimi pochodnymi.

Porównanie

- **Złożoność obliczeniowa**: Interpolacja wielomianowa wymaga rozwiązania układu równań o wyższej złożoności, natomiast interpolacja funkcjami sklejanymi wymaga jedynie rozwiązywania układów równań liniowych.
- **Stabilność**: Interpolacja funkcjami sklejanymi jest bardziej stabilna numerycznie, zwłaszcza przy dużej liczbie węzłów. Interpolacja wielomianowa może prowadzić do efektu Rungego.
- **Dokładność**: Interpolacja funkcjami sklejanymi często daje bardziej dokładne wyniki, zwłaszcza gdy funkcja interpolowana jest gładka.
- **Elastyczność**: Funkcje sklejane pozwalają na łatwe dodawanie nowych węzłów bez konieczności ponownego przeliczania całego wielomianu.

Warunki brzegowe stosowane przy wyznaczaniu sześciennych funkcji sklejanych

Przy konstrukcji sześciennych funkcji sklejanych (cubic splines) stosuje się różne warunki brzegowe, aby zapewnić jednoznaczność rozwiązania układu równań. Poniżej omówiono najczęściej stosowane warunki brzegowe.

Naturalne warunki brzegowe (natural cubic splines)

Naturalne warunki brzegowe zakładają, że drugie pochodne funkcji sklejanego wielomianu na krańcach przedziału są równe zero:

$$s''(x_1) = 0 \quad \text{i} \quad s''(x_n) = 0$$

Warunki brzegowe typu clamped (clamped boundary conditions)

Warunki typu clamped zakładają, że pierwsze pochodne funkcji sklejanego wielomianu na krańcach przedziału są równe znanym wartościom:

$$s'(x_1) = f'(x_1) \quad \text{i} \quad s'(x_n) = f'(x_n)$$

Warunki brzegowe typu not-a-knot

Warunki typu not-a-knot zakładają, że trzecie pochodne funkcji sklejanego wielomianu są ciągłe na drugim i przedostatnim węźle:

$$s'''(x_2) = s'''(x_3) \quad \text{i} \quad s'''(x_{n-1}) = s'''(x_n)$$

Warunki brzegowe dla funkcji periodycznych

Dla funkcji periodycznych warunki brzegowe zakładają, że wartości funkcji oraz jej pochodnych na końcach przedziału są równe:

$$s(x_1) = s(x_n) \quad \text{i} \quad s'(x_1) = s'(x_n) \quad \text{i} \quad s''(x_1) = s''(x_n)$$

Warunki brzegowe są kluczowe dla poprawnej konstrukcji funkcji sklejanych, gdyż zapewniają ciągłość i gładkość interpolowanej funkcji oraz jednoznaczność rozwiązania.

Porównanie aproksymacji średniokwadratowej i jednostajnej

Aproksymacja średniokwadratowa

Aproksymacja średniokwadratowa polega na minimalizacji sumy kwadratów różnic między wartościami funkcji aproksymowanej $F(x)$ a wartościami funkcji aproksymującej $f(x)$. Szukamy wielomianu $f(x) = \sum_{i=0}^{n} a_i \phi_i(x)$, który minimalizuje wyrażenie:

$$\int_a^b w(x)[F(x) - f(x)]^2 \, dx$$

gdzie $w(x)$ jest funkcją wagową.

Aproksymacja jednostajna

Aproksymacja jednostajna dąży do minimalizacji maksymalnej odchyłki między $F(x)$ a $f(x)$ na przedziale $[a, b]$. Szukamy wielomianu $f(x)$, który minimalizuje wyrażenie:

$$\|F(x) - f(x)\| = \sup_{x \in [a,b]} |F(x) - f(x)|$$

Porównanie

- **Zastosowanie**: Aproksymacja średniokwadratowa jest użyteczna w sytuacjach, gdy istotne jest ogólne dopasowanie funkcji na całym przedziale, np. w analizie regresji. Aproksymacja jednostajna jest preferowana, gdy kluczowe jest maksymalne dopasowanie funkcji w każdym punkcie przedziału, np. w analizie błędów.
- **Stabilność**: Aproksymacja jednostajna może być mniej stabilna numerycznie niż średniokwadratowa, szczególnie przy wysokich stopniach wielomianu.
- **Obliczeniowość**: Aproksymacja średniokwadratowa często prowadzi do układów równań liniowych, które są łatwe do rozwiązania. Aproksymacja jednostajna wymaga rozwiązywania nierówności, co może być bardziej skomplikowane.

Metody aproksymacji jednostajnej

Szereg Taylora

Prosta metoda aproksymacji jednostajnej wykorzystująca pierwsze $n+1$ wyrazy szeregu Taylora funkcji $F(x)$ w punkcie x_0:

$$W_n(x) = \sum_{k=0}^{n} \frac{F^{(k)}(x_0)}{k!}(x - x_0)^k$$

Przybliżenia Padé

Aproksymacja za pomocą funkcji wymiernych $r(x) = \frac{p(x)}{q(x)}$, gdzie $p(x)$ i $q(x)$ są wielomianami stopnia odpowiednio n i m. Funkcje wymierne mogą lepiej aproksymować funkcje o skomplikowanych przebiegach niż zwykłe wielomiany.

Podstawowe własności wielomianów Czebyszewa oraz ich zastosowania

Wielomiany Czebyszewa

Wielomiany Czebyszewa $T_n(x)$ definiowane są rekurencyjnie:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \quad \text{dla} \quad n \geq 2$$

Własności

- **Ortogonalność**: Wielomiany Czebyszewa są ortogonalne względem funkcji wagowej $\frac{1}{\sqrt{1-x^2}}$ na przedziale $[-1, 1]$.
- **Symetria**: $T_n(-x) = (-1)^n T_n(x)$.
- **Zera**: Zera wielomianów Czebyszewa są rozmieszczone równomiernie w przedziale $[-1, 1]$ i stanowią węzły Czebyszewa.

Zastosowania

Wielomiany Czebyszewa są używane w aproksymacji funkcji, ponieważ minimalizują maksymalny błąd aproksymacji (aproksymacja minimaksowa). Są także stosowane w metodach numerycznych, takich jak kwadratury Czebyszewa.

Podstawowe własności wielomianów Legendre'a oraz ich zastosowania

Wielomiany Legendre'a

Wielomiany Legendre'a $P_n(x)$ definiowane są wzorem:

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} \left[(x^2 - 1)^n\right]$$

Własności

- **Ortogonalność**: Wielomiany Legendre'a są ortogonalne względem funkcji wagowej 1 na przedziale $[-1, 1]$.
- **Rekurencja**: Wielomiany Legendre'a spełniają rekurencję:

$$(n + 1)P_{n+1}(x) = (2n + 1)xP_n(x) - nP_{n-1}(x)$$

- **Symetria**: $P_n(-x) = (-1)^n P_n(x)$.

Zastosowania

Wielomiany Legendre'a są stosowane w kwadraturach Gaussa-Legendre'a do numerycznego całkowania. Są także wykorzystywane w fizyce, szczególnie w rozwiązywaniu równań różniczkowych w sferycznych układach współrzędnych.

Porównanie kwadratur Newtona-Cotesa i Gaussa

Kwadratury Newtona-Cotesa

Kwadratury Newtona-Cotesa polegają na aproksymacji całki funkcji $f(x)$ na przedziale $[a, b]$ za pomocą sumy wartości funkcji w równomiernie rozmieszczonych węzłach. Przykłady to reguła trapezów i reguła Simpsona.

Kwadratury Gaussa

Kwadratury Gaussa polegają na aproksymacji całki funkcji $f(x)$ za pomocą sumy wartości funkcji w specjalnie dobranych węzłach, które są zerami wielomianów ortogonalnych na danym przedziale.

Porównanie

- **Dokładność**: Kwadratury Gaussa są dokładniejsze niż kwadratury Newtona-Cotesa przy tej samej liczbie węzłów, ponieważ mają wyższy stopień dokładności.
- **Złożoność**: Kwadratury Newtona-Cotesa są prostsze do implementacji, ale mniej efektywne przy dużych przedziałach całkowania. Kwadratury Gaussa wymagają obliczenia miejsc zerowych wielomianów ortogonalnych, co jest bardziej złożone.
- **Zastosowanie**: Kwadratury Newtona-Cotesa są użyteczne dla prostych funkcji i małych przedziałów całkowania, podczas gdy kwadratury Gaussa są lepsze dla bardziej skomplikowanych funkcji i większych przedziałów.

Algorytm całkowania adaptacyjnego

Istota algorytmu

Całkowanie adaptacyjne polega na dostosowywaniu kroku całkowania w zależności od lokalnych właściwości funkcji podcałkowej. W obszarach, gdzie funkcja zmienia się szybko, krok całkowania jest zmniejszany, a tam, gdzie funkcja jest gładka, krok jest zwiększany.

Przykład procedury

1. Podział przedziału całkowania na mniejsze fragmenty.
2. Zastosowanie jednej z metod klasycznych (np. reguła Simpsona) na każdym fragmencie.
3. Ocena błędu przybliżenia na każdym fragmencie.
4. Dostosowanie kroków całkowania w zależności od ocenionego błędu, poprzez dalszy podział fragmentów z dużym błędem.

Zalety

- **Efektywność**: Algorytm adaptacyjny dostosowuje się do lokalnych właściwości funkcji, co pozwala na oszczędność obliczeń.
- **Dokładność**: Lepsze dostosowanie kroków całkowania do zmienności funkcji prowadzi do zwiększenia dokładności całkowania.

Metoda Newtona-Raphsona rozwiązywania równań nieliniowych (wraz z zabezpieczeniami)

Definicja

Metoda Newtona-Raphsona jest iteracyjną metodą służącą do znajdowania pierwiastków funkcji nieliniowych $f(x) = 0$. Iteracyjny wzór metody Newtona-Raphsona jest dany przez:

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

Zbieżność

Metoda jest zbieżna kwadratowo, co oznacza, że jeśli x_i jest wystarczająco blisko rzeczywistego pierwiastka α, to:

$$|x_{i+1} - \alpha| \approx C|x_i - \alpha|^2$$

gdzie C jest stałą.

Zabezpieczenia

Aby zwiększyć niezawodność metody Newtona-Raphsona, stosuje się różne zabezpieczenia:

- **Wybór przedziału startowego**: Upewnienie się, że początkowe przybliżenie x_0 znajduje się w przedziale, gdzie funkcja zmienia znak.
- **Kontrola liczby iteracji**: Ustalona maksymalna liczba iteracji, aby uniknąć nieskończonych cykli.
- **Zabezpieczenie przed dzieleniem przez zero**: Sprawdzanie wartości $f'(x_i)$ i unikanie wartości bliskich zeru.
- **Metoda bisekcji**: Użycie metody bisekcji jako pomocniczej, gdy iteracja Newtona-Raphsona przestaje zbiegać.

Interpolacyjne metody znajdowania rozwiązań równań nieliniowych

Interpolacyjne metody rozwiązywania równań nieliniowych opierają się na aproksymacji funkcji za pomocą wielomianów lub innych funkcji.

Metoda siecznych

Metoda siecznych jest uproszczoną wersją metody Newtona-Raphsona, w której pochodna $f'(x)$ jest aproksymowana różnicą ilorazową. Iteracyjny wzór jest dany przez:

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$

Metoda Brent'a

Metoda Brent'a łączy zalety metod bisekcji, siecznych i Newtona-Raphsona, oferując niezawodność i szybkie zbieżności.

Metoda Mullera

Metoda Mullera używa interpolacji kwadratowej do znajdowania pierwiastków. Korzysta z trzech punktów do aproksymacji funkcji za pomocą parabolicznej krzywej.

Algorytm rozwiązywania układów równań liniowych metodą eliminacji Gaussa

Kroki algorytmu

1. **Eliminacja**: Przekształć macierz A do postaci trójkątnej górnej poprzez dodawanie do siebie odpowiednich wielokrotności wierszy. 2. **Podstawianie wsteczne**: Rozwiąż układ równań dla wektora x, zaczynając od ostatniego wiersza.

WTF

Szczegóły algorytmu

1. Dla każdego wiersza k od 1 do $n-1$:
   (a) Znajdź maksymalny element w kolumnie k (pivoting).
   (b) Zamień wiersze, aby maksymalny element znalazł się na przekątnej.
   (c) Wyzeruj elementy poniżej pivotu przez dodawanie odpowiednich wielokrotności pivotu do wierszy poniżej.
2. Rozwiąż powstały układ równań trójkątnych górnych przez podstawianie wsteczne.

Porównanie algorytmów faktoryzacji LU Doolittle'a, Crout'a i Choleskiego

Faktoryzacja LU Doolittle'a

Metoda Doolittle'a faktoryzuje macierz A jako $A = LU$, gdzie L jest macierzą dolną trójkątną z jedynkami na diagonali, a U jest macierzą górną trójkątną.

Faktoryzacja Crout'a

Metoda Crout'a faktoryzuje macierz A jako $A = LU$, gdzie L jest macierzą dolną trójkątną, a U jest macierzą górną trójkątną z jedynkami na diagonali.

Faktoryzacja Choleskiego

Metoda Choleskiego jest specjalnym przypadkiem faktoryzacji LU dla macierzy symetrycznych i dodatnio określonych, faktoryzując macierz jako $A = LL^T$, gdzie L jest macierzą dolną trójkątną.

Przewaga algorytmów faktoryzacji LU nad metodą eliminacji Gaussa

Faktoryzacja LU oferuje kilka zalet w porównaniu z klasyczną eliminacją Gaussa:

- **Efektywność**: Raz przefaktoryzowana macierz może być użyta do rozwiązywania wielu układów równań z różnymi wektorami prawych stron.
- **Stabilność**: Faktoryzacja LU z częściowym wyborem elementu wiodącego poprawia numeryczną stabilność obliczeń.
- **Oszczędność pamięci**: Faktoryzacja LU przechowuje macierz w spakowanej postaci, co zmniejsza wymagania pamięciowe.

Porównanie metod iteracyjnych Jacobiego, S-R, SOR, Czebyszewa

Metoda Jacobiego

Każdy element rozwiązania jest aktualizowany na podstawie wartości z poprzedniej iteracji. Iteracyjny wzór jest dany przez:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right)$$

Metoda Gaussa-Seidla (S-R)

Podobna do metody Jacobiego, ale każdy element rozwiązania jest aktualizowany natychmiastowo, co prowadzi do szybszej zbieżności:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right)$$

Metoda SOR (Successive Over-Relaxation)

Ulepszona wersja metody Gaussa-Seidla, w której każda iteracja jest modyfikowana przez parametr relaksacji ω:

$$x_i^{(k+1)} = (1-\omega) x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right)$$

Metoda Czebyszewa

Metoda iteracyjna oparta na wielomianach Czebyszewa, która jest szczególnie efektywna dla dużych układów równań o specjalnych własnościach spektralnych. Wykorzystuje ortogonalność wielomianów Czebyszewa do przyspieszenia zbieżności.

Znaczenie szukania elementu wiodącego w metodach rozwiązywania układów równań liniowych

Szukanie elementu wiodącego (pivoting) w metodach rozwiązywania układów równań liniowych jest kluczowym krokiem, ponieważ:

- **Stabilność numeryczna**: Wybór największego elementu jako elementu wiodącego minimalizuje błędy zaokrągleń i zwiększa stabilność obliczeń.
- **Unikanie dzielenia przez zero**: Gwarantuje, że dzielnik w eliminacji Gaussa nie będzie zerem ani nie będzie bliski zeru, co mogłoby prowadzić do dużych błędów numerycznych.
- **Poprawa kondycjonowania**: Poprawia kondycjonowanie macierzy, co jest istotne dla dokładności rozwiązania.

Błąd względny reprezentacji zmiennoprzecinkowej

Definicja

Błąd względny reprezentacji zmiennoprzecinkowej jest miarą dokładności, z jaką liczba rzeczywista x jest reprezentowana w arytmetyce zmiennoprzecinkowej. Reprezentacja zmiennoprzecinkowa liczby x jest oznaczana jako $\mathrm{fl}(x)$.

Wyprowadzenie wzoru

Dla liczby x reprezentowanej w systemie zmiennoprzecinkowym:

$$x = s \cdot 2^c \cdot m$$

gdzie s to znak, c to cecha, a m to mantysa.

Mantysa m może być przedstawiona jako:

$$m = \sum_{i=1}^{t} e_i \cdot 2^{-i}$$

gdzie e_i są bitami mantysy.

Jeśli liczba jest bliżej do swojego zaokrąglenia w dół, to pomijamy resztę bitów i reprezentacja jest równa:

$$\mathrm{fl}(x)^- = \pm \sum_{i=1}^{t} d_i \beta^{-i} \cdot \beta^e$$

Najdalsza "końcówka"takiej liczby to:

$$m = \sum_{i=1}^{t} e_i \cdot 2^{-i} + 0 \cdot 2^{-(t+1)} + \sum_{i=t+2}^{\infty} 1 \cdot 2^{-i} = 2^{-(t+1)}$$

W przypadku zaokrąglenia w górę:

$$\mathrm{fl}(x)^{+} = \pm \left(\sum_{i=1}^{t} d_i \beta^{-i} + \frac{1}{\beta^t} \right) \cdot \beta^e$$

Błąd względny można oszacować:

$$\left| \frac{m - \mathrm{fl}(m)}{m} \right| \leq 2^{-t}$$

Wskaźnik uwarunkowania zadania

Definicja

Wskaźnik uwarunkowania zadania, oznaczany jako $\text{cond}(\varphi(x))$, określa czułość rozwiązania zadania na błędy wejściowe. Wskaźnik ten mierzy, jak względne błędy danych wejściowych wpływają na względne błędy wyniku.

Wyprowadzenie wzoru

Jeśli dane wejściowe x są zaburzone o niewielki błąd względny ϵ, to względny błąd wyniku nie przekroczy $\epsilon \cdot \text{cond}(\varphi(x))$.

Przykład wyznaczenia wskaźnika uwarunkowania dla iloczynu skalarnego wektorów $\vec{x} \cdot \vec{y}$:

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i \neq 0$$

Dla zaburzonych danych $x_i \to x_i(1+\alpha_i)$ i $y_i \to y_i(1+\beta_i)$:

$$\sum_{i=1}^{n} x_i(1+\alpha_i)y_i(1+\beta_i) = \sum_{i=1}^{n} x_i y_i(1+\alpha_i+\beta_i+\alpha_i\beta_i)$$

Zakładając, że błędy α_i i β_i są małe, wyraz $\alpha_i\beta_i$ można pominąć, co daje:

$$\sum_{i=1}^{n} x_i(1+\alpha_i)y_i(1+\beta_i) \approx \sum_{i=1}^{n} x_i y_i(1+\alpha_i+\beta_i)$$

Różnica względna między zaburzonym a niezaburzonym iloczynem skalarnym wynosi:

$$\left|\frac{\sum_{i=1}^{n} x_i y_i(1+\alpha_i+\beta_i) - \sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i y_i}\right| = \left|\frac{\sum_{i=1}^{n} x_i y_i(\alpha_i+\beta_i)}{\sum_{i=1}^{n} x_i y_i}\right| = \left|\frac{\sum_{i=1}^{n} x_i y_i(\alpha_i+\beta_i)}{\sum_{i=1}^{n} x_i y_i}\right|$$

Wskaźnik uwarunkowania dla iloczynu skalarnego można więc wyrazić jako:

$$\text{cond}(\vec{x} \cdot \vec{y}) = \frac{\sum_{i=1}^{n} |x_i y_i|}{|\sum_{i=1}^{n} x_i y_i|}$$

Gdy wszystkie x_i i y_i mają ten sam znak, to $\text{cond}(\vec{x} \cdot \vec{y}) = 1$.

Interpolacja metodą Lagrange'a

Wielomian interpolacyjny Lagrange'a jest używany do znalezienia wielomianu, który przechodzi przez zadane punkty $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$. Ogólna postać tego wielomianu to:

$$P_n(x) = \sum_{k=0}^{n} y_k \cdot L_k(x)$$

gdzie $L_k(x)$ to k-te bazowe wielomiany Lagrange'a zdefiniowane jako:

$$L_k(x) = \prod_{\substack{0 \leq j \leq n \\ j \neq k}} \frac{x - x_j}{x_k - x_j}$$

Wielomiany te spełniają warunek:

$$L_k(x_i) = \begin{cases} 1 & \text{gdy } i = k \\ 0 & \text{gdy } i \neq k \end{cases}$$

co oznacza, że każdy wielomian $L_k(x)$ jest równy 1 w punkcie x_k i 0 w pozostałych węzłach x_i dla $i \neq k$.

Aby zilustrować konstrukcję, rozważmy funkcję $f(x)$ przechodzącą przez węzły $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$. Wielomian interpolacyjny Lagrange'a $P_n(x)$ można wyrazić jako:

$$P_n(x) = \sum_{k=0}^{n} f(x_k) \cdot \prod_{\substack{0 \leq j \leq n \\ j \neq k}} \frac{x - x_j}{x_k - x_j}$$

Przykład: Jeśli mamy trzy punkty $(x_0, y_0), (x_1, y_1), (x_2, y_2)$, to wielomian interpolacyjny stopnia drugiego $P_2(x)$ jest dany przez:

$$P_2(x) = y_0 \cdot \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \cdot \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \cdot \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Tak skonstruowany wielomian przechodzi przez wszystkie zadane punkty $(x_0, y_0), (x_1, y_1), (x_2, y_2)$.

Wzór ogólny na wielomian interpolacyjny Lagrange'a $P_n(x)$ jest więc:

$$P_n(x) = \sum_{k=0}^{n} f(x_k) \cdot \prod_{\substack{0 \leq j \leq n \\ j \neq k}} \frac{x - x_j}{x_k - x_j}$$

Interpolacja metodą Lagrange'a ma swoje zalety i wady: - Zalety: Łatwość utworzenia wzoru wielomianu interpolacyjnego. - Wady: Przy dodawaniu nowego węzła konieczność ponownego wyliczenia całego wielomianu oraz konieczność powtórnego obliczenia licznika dla $L_k(x)$ przy obliczaniu wartości w konkretnym punkcie.

Błąd interpolacji metodą Lagrange'a

Błąd interpolacji metodą Lagrange'a mierzy różnicę pomiędzy wartością rzeczywistej funkcji $f(x)$ a wartością wielomianu interpolacyjnego $P_n(x)$. Aby oszacować ten błąd, możemy skorzystać z twierdzenia Rolle'a i uzyskać następujący wzór:

$$E_n(x) = f(x) - P_n(x)$$

W przypadku, gdy f jest funkcją $(n+1)$-krotnie różniczkowalną na przedziale $[a, b]$, błąd interpolacji w punkcie x można wyrazić wzorem:

$$E_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

gdzie ξ jest pewnym punktem w przedziale (a, b), zależnym od x.

Wyprowadzenie wzoru

Funkcja pomocnicza

Zdefiniujmy funkcję pomocniczą $g(t)$ jako różnicę pomiędzy funkcją $f(t)$ a wielomianem interpolacyjnym $P_n(t)$, skorygowaną o wyraz dodatkowy, który ma za zadanie anulowanie wartości $g(t)$ w dodatkowym punkcie $\tilde{x}$:

$$g(t) = f(t) - P_n(t) - (f(\tilde{x}) - P_n(\tilde{x})) \prod_{i=0}^{n} \frac{t - x_i}{\tilde{x} - x_i}$$

Zastosowanie twierdzenia Rolle'a

Funkcja $g(t)$ ma $n + 2$ miejsc zerowych: $x_0, x_1, \ldots, x_n$ oraz $\tilde{x}$, dlatego na mocy twierdzenia Rolle'a, istnieje punkt $\eta \in (a, b)$, taki że pochodna $(n + 1)$-szego rzędu $g(t)$ w punkcie η wynosi zero:

$$g^{(n+1)}(\eta) = 0$$

Obliczanie pochodnych

Obliczając $(n + 1)$-szą pochodną $g(t)$, uzyskujemy:

$$f^{(n+1)}(\eta) - P_n^{(n+1)}(\eta) - (f(\tilde{x}) - P_n(\tilde{x})) \frac{d^{n+1}}{dt^{n+1}} \left(\prod_{i=0}^{n} \frac{t - x_i}{\tilde{x} - x_i} \right) \Bigg|_{t=\eta} = 0$$

Ponieważ $P_n(t)$ jest wielomianem interpolacyjnym stopnia n, jego pochodna $(n + 1)$-szego rzędu wynosi zero:

$$P_n^{(n+1)}(\eta) = 0$$

Wyrażenie końcowe

Wyznaczając pochodną $(n+1)$-szego rzędu iloczynu, otrzymujemy:

$$\frac{d^{n+1}}{dt^{n+1}}\left(\prod_{i=0}^{n}\frac{t-x_i}{\tilde{x}-x_i}\right)\Bigg|_{t=\eta}=\frac{(n+1)!}{\prod_{i=0}^{n}(\tilde{x}-x_i)}$$

Podstawiając do równania:

$$f^{(n+1)}(\eta)-(f(\tilde{x})-P_n(\tilde{x}))\frac{(n+1)!}{\prod_{i=0}^{n}(\tilde{x}-x_i)}=0$$

Ostatecznie, wyrażenie na błąd interpolacji przyjmuje postać:

$$f(x)=P_n(x)+\frac{f^{(n+1)}(\eta)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)$$

gdzie η jest pewnym punktem z przedziału $[a,b]$.

$$E_n(x)=\frac{f^{(n+1)}(\eta)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)$$

$$\boxed{E_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)}$$

Sześcienne funkcje sklejane

Sześcienne funkcje sklejane, znane również jako sześcienne splajny, są często używane do konstrukcji gładkich krzywych przechodzących przez zadane punkty. Sześcienny splajn na przedziale $[x_i, x_{i+1}]$ jest trzeciego stopnia wielomianem, który zapewnia ciągłość pierwszej i drugiej pochodnej na całym przedziale interpolacji.

Definicja splajnów sześciennych

Dla danego zestawu punktów $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, splajn sześcienny $s(x)$ jest zdefiniowany w każdym podprzedziale $[x_i, x_{i+1}]$ jako:

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

gdzie a_i, b_i, c_i, d_i są współczynnikami określonymi dla każdego przedziału $[x_i, x_{i+1}]$.

Warunki ciągłości

Aby zapewnić gładkość splajnu, funkcja $s(x)$ musi spełniać następujące warunki ciągłości: 1. $s_i(x_i) = y_i$ 2. $s_i(x_{i+1}) = y_{i+1}$ 3. $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$ 4. $s_i''(x_{i+1}) = s_{i+1}''(x_{i+1})$

Wyprowadzenie wzoru

Splajn sześcienny jest trzeciego stopnia wielomianem, a jego druga pochodna jest funkcją liniową. Możemy więc zapisać drugą pochodną splajnu na przedziale $[x_i, x_{i+1}]$ jako:

$$s_i''(x) = \frac{s_i''(x_i)(x_{i+1} - x) + s_i''(x_{i+1})(x - x_i)}{h_i}$$

gdzie $h_i = x_{i+1} - x_i$.
Integrując dwukrotnie, otrzymujemy wyrażenie na $s_i(x)$:

$$s_i(x) = \frac{s_i''(x_i)}{6h_i}(x_{i+1} - x)^3 + \frac{s_i''(x_{i+1})}{6h_i}(x - x_i)^3 + A_i(x - x_i) + B_i$$

Stosując warunki brzegowe $s_i(x_i) = y_i$ i $s_i(x_{i+1}) = y_{i+1}$, możemy wyznaczyć stałe A_i i B_i:

$$A_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}(s_i''(x_{i+1}) - s_i''(x_i))$$

$$B_i = y_i$$

Stąd mamy pełne wyrażenie na splajn sześcienny na przedziale $[x_i, x_{i+1}]$:

$$s_i(x) = \frac{s_i''(x_i)}{6h_i}(x_{i+1}-x)^3 + \frac{s_i''(x_{i+1})}{6h_i}(x-x_i)^3 + \left(\frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{6}(s_i''(x_{i+1}) - s_i''(x_i))\right)(x-x_i) + y_i$$

Wyznaczenie drugich pochodnych

Aby wyznaczyć drugie pochodne $s_i''(x_i)$, tworzymy układ równań wynikający z warunku ciągłości pierwszej pochodnej:

$$h_{i-1}s_{i-1}''(x_i) + 2(h_{i-1}+h_i)s_i''(x_i) + h_i s_i''(x_{i+1}) = 6\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right)$$

dla $i = 1, 2, \ldots, n-1$.

Ten układ równań można rozwiązać metodą eliminacji Gaussa lub inną odpowiednią metodą numeryczną.

Warunki brzegowe

Sposób określenia dodatkowych warunków brzegowych decyduje o rodzaju splajnu sześciennego. Najczęściej stosowane są dwa rodzaje warunków brzegowych:

1. Natural spline (naturalne splajny): $s''(x_1) = s''(x_n) = 0$ 2. Clamped spline (splajny z warunkami brzegowymi na pochodnych): $s'(x_1) = f'(x_1)$ i $s'(x_n) = f'(x_n)$

Ostateczny wzór

Ostateczny wzór na splajn sześcienny na przedziale $[x_i, x_{i+1}]$ przyjmuje postać:

$$s_i(x) = \frac{s_i''(x_i)}{6h_i}(x_{i+1}-x)^3 + \frac{s_i''(x_{i+1})}{6h_i}(x-x_i)^3 + \left(\frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{6}(s_i''(x_{i+1}) - s_i''(x_i))\right)(x-x_i) + y_i$$

Kwadratowe funkcje sklejane

Kwadratowe funkcje sklejane, znane również jako kwadratowe splajny, są używane do konstrukcji gładkich krzywych, które przechodzą przez zadane punkty. Kwadratowy splajn na przedziale $[x_i, x_{i+1}]$ jest drugiego stopnia wielomianem, który zapewnia ciągłość funkcji i jej pierwszej pochodnej na całym przedziale interpolacji.

Definicja splajnów kwadratowych

Dla danego zestawu punktów $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, splajn kwadratowy $s(x)$ jest zdefiniowany w każdym podprzedziale $[x_i, x_{i+1}]$ jako:

$$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2$$

gdzie a_i, b_i, c_i są współczynnikami określonymi dla każdego przedziału $[x_i, x_{i+1}]$.

Warunki ciągłości

Aby zapewnić gładkość splajnu, funkcja $s(x)$ musi spełniać następujące warunki ciągłości: 1. $s_i(x_i) = y_i$ 2. $s_i(x_{i+1}) = y_{i+1}$ 3. $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$

Wyprowadzenie wzoru

Splajn kwadratowy jest drugiego stopnia wielomianem. Wyznaczając współczynniki a_i, b_i, c_i dla każdego przedziału $[x_i, x_{i+1}]$, możemy zapisać:

$$s_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2$$

Stosując warunki brzegowe $s_i(x_{i+1}) = y_{i+1}$ oraz warunki ciągłości $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$, możemy wyznaczyć wartości współczynników b_i i c_i:

$$s_i(x_{i+1}) = y_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 = y_{i+1}$$

$$s_i'(x) = b_i + 2c_i(x - x_i)$$

Stosując warunki ciągłości pochodnych, mamy:

$$b_i + 2c_i(x_{i+1} - x_i) = b_{i+1}$$

Podstawiając $h_i = x_{i+1} - x_i$, możemy wyrazić b_i i c_i jako:

$$b_i = \frac{y_{i+1} - y_i}{h_i} - c_i h_i$$

$$c_i = \frac{b_{i+1} - b_i}{2h_i}$$

Warunki brzegowe

Sposób określenia dodatkowych warunków brzegowych decyduje o rodzaju splajnu kwadratowego. Najczęściej stosowane są dwa rodzaje warunków brzegowych:

1. Natural spline (naturalne splajny): $s''(x_1) = s''(x_n) = 0$ 2. Clamped spline (splajny z warunkami brzegowymi na pochodnych): $s'(x_1) = f'(x_1)$ i $s'(x_n) = f'(x_n)$

Ostateczny wzór

Ostateczny wzór na splajn kwadratowy na przedziale $[x_i, x_{i+1}]$ przyjmuje postać:

$$s_i(x) = y_i + \left(\frac{y_{i+1} - y_i}{h_i} - c_i h_i \right) (x - x_i) + c_i (x - x_i)^2$$

gdzie c_i jest wyznaczane na podstawie warunków ciągłości pochodnych.

Aproksymacja średniokwadratowa wielomianem uogólnionym

Aproksymacja średniokwadratowa (metoda najmniejszych kwadratów) jest techniką stosowaną do przybliżenia funkcji poprzez minimalizację sumy kwadratów odchyleń między wartościami funkcji aproksymowanej a wartościami funkcji aproksymującej. W tej sekcji przedstawimy wyprowadzenie wzoru dla aproksymacji średniokwadratowej przy użyciu wielomianu uogólnionego.

Postawienie problemu

Dane są: - Funkcja aproksymowana $F(x)$ zadana w punktach $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$. - Układ funkcji bazowych $\{\phi_j(x)\}$, gdzie $j = 0, 1, \ldots, m$.

Szukamy wielomianu uogólnionego w postaci:

$$f(x) = \sum_{j=0}^{m} a_j \phi_j(x)$$

współczynniki $\{a_j\}_{j=0}^{m}$ znajdujemy z warunku:

$$\min \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \phi_j(x_i) \right]^2$$

gdzie $w(x_i)$ jest funkcją wagową.

Formułowanie funkcji celu

Funkcję celu możemy zapisać jako:

$$H(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \phi_j(x_i) \right]^2$$

Aby znaleźć minimalną wartość H, bierzemy pochodne cząstkowe względem każdego a_k i przyrównujemy je do zera:

$$\frac{\partial H}{\partial a_k} = 0, \quad k = 0, 1, \ldots, m$$

Wyprowadzenie układu równań normalnych

Pochodne cząstkowe funkcji celu H względem a_k wynoszą:

$$\frac{\partial H}{\partial a_k} = -2 \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \phi_j(x_i) \right] \phi_k(x_i)$$

Przyrównując do zera, otrzymujemy:

$$\sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \phi_j(x_i) \right] \phi_k(x_i) = 0, \quad k = 0, 1, \ldots, m$$

Rozwijając to równanie, mamy:

$$\sum_{i=0}^{n} w(x_i) F(x_i) \phi_k(x_i) = \sum_{i=0}^{n} w(x_i) \phi_k(x_i) \sum_{j=0}^{m} a_j \phi_j(x_i)$$

Zamieniamy sumy:

$$\sum_{i=0}^{n} w(x_i) F(x_i) \phi_k(x_i) = \sum_{j=0}^{m} a_j \sum_{i=0}^{n} w(x_i) \phi_j(x_i) \phi_k(x_i)$$

Ostatecznie otrzymujemy układ równań liniowych:

$$\sum_{j=0}^{m} a_j \sum_{i=0}^{n} w(x_i) \phi_j(x_i) \phi_k(x_i) = \sum_{i=0}^{n} w(x_i) F(x_i) \phi_k(x_i), \quad k = 0, 1, \ldots, m$$

Rozwiązanie układu równań

Otrzymany układ równań normalnych można zapisać w postaci macierzowej:

$$\mathbf{Ga} = \mathbf{b}$$

gdzie:

$$\mathbf{G}_{kj} = \sum_{i=0}^{n} w(x_i) \phi_j(x_i) \phi_k(x_i)$$

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}$$

$$\mathbf{b}_k = \sum_{i=0}^{n} w(x_i) F(x_i) \phi_k(x_i)$$

Macierz $\mathbf{G}$ jest macierzą Gramianą i jeśli funkcje bazowe $\phi_j(x)$ są liniowo niezależne, to macierz $\mathbf{G}$ jest odwracalna, co zapewnia unikalność rozwiązania układu równań.

Aproksymacja średniokwadratowa jednomianami

Aproksymacja średniokwadratowa jednomianami jest szczególnym przypadkiem metody najmniejszych kwadratów, gdzie funkcje bazowe są jednomianami. Oznacza to, że do aproksymacji używamy wielomianów w postaci x^j.

Postawienie problemu

Dane są: - Funkcja aproksymowana $F(x)$ zadana na zbiorze dyskretnym (x_i, y_i) dla $i = 0, 1, \ldots, n$. - Funkcja aproksymująca $f(x)$ w postaci wielomianu:

$$f(x) = \sum_{j=0}^{m} a_j x^j$$

Celem jest znalezienie współczynników a_j, które minimalizują sumę kwadratów odchyleń pomiędzy wartościami funkcji aproksymowanej a wartościami funkcji aproksymującej:

$$\min \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j x_i^j \right]^2$$

gdzie $w(x_i)$ jest funkcją wagową.

Formułowanie funkcji celu

Funkcję celu można zapisać jako:

$$H(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j x_i^j \right]^2$$

Aby znaleźć minimalną wartość H, bierzemy pochodne cząstkowe względem każdego a_k i przyrównujemy je do zera:

$$\frac{\partial H}{\partial a_k} = 0, \quad k = 0, 1, \ldots, m$$

Wyprowadzenie układu równań normalnych

Pochodne cząstkowe funkcji celu H względem a_k wynoszą:

$$\frac{\partial H}{\partial a_k} = -2 \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j x_i^j \right] x_i^k$$

Przyrównując do zera, otrzymujemy:

$$\sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j x_i^j \right] x_i^k = 0, \quad k = 0, 1, \ldots, m$$

Rozwijając to równanie, mamy:

$$\sum_{i=0}^{n} w(x_i) F(x_i) x_i^k = \sum_{i=0}^{n} w(x_i) x_i^k \sum_{j=0}^{m} a_j x_i^j$$

Zamieniając kolejność sumowania, uzyskujemy:

$$\sum_{i=0}^{n} w(x_i) F(x_i) x_i^k = \sum_{j=0}^{m} a_j \sum_{i=0}^{n} w(x_i) x_i^{j+k}$$

Ostatecznie otrzymujemy układ równań liniowych:

$$\sum_{j=0}^{m} a_j \sum_{i=0}^{n} w(x_i) x_i^{j+k} = \sum_{i=0}^{n} w(x_i) F(x_i) x_i^k, \quad k = 0, 1, \ldots, m$$

Rozwiązanie układu równań

Otrzymany układ równań normalnych można zapisać w postaci macierzowej:

$$\mathbf{Ga} = \mathbf{b}$$

gdzie:

$$\mathbf{G}_{kj} = \sum_{i=0}^{n} w(x_i) x_i^{j+k}$$

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}$$

$$\mathbf{b}_k = \sum_{i=0}^{n} w(x_i) F(x_i) x_i^k$$

Macierz $\mathbf{G}$ jest macierzą Gramianą i jeśli funkcje bazowe x^j są liniowo niezależne, to macierz $\mathbf{G}$ jest odwracalna, co zapewnia unikalność rozwiązania układu równań.

Aproksymacja średniokwadratowa wielomianami ortogonalnymi

Aproksymacja średniokwadratowa (metoda najmniejszych kwadratów) wielomianami ortogonalnymi jest techniką polegającą na minimalizowaniu sumy kwadratów odchyleń między wartościami funkcji aproksymowanej a wartościami funkcji aproksymującej, przy użyciu wielomianów ortogonalnych.

Postawienie problemu

Dane są: - Funkcja aproksymowana $F(x)$ zadana na zbiorze dyskretnym (x_i, y_i) dla $i = 0, 1, \ldots, n$. - Układ funkcji bazowych $\{\phi_j(x)\}$, gdzie $j = 0, 1, \ldots, m$, przy czym funkcje te są ortogonalne.

Szukamy wielomianu ortogonalnego w postaci:

$$f(x) = \sum_{j=0}^{m} a_j \phi_j(x)$$

Współczynniki $\{a_j\}_{j=0}^{m}$ znajdujemy z warunku:

$$\min \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \phi_j(x_i) \right]^2$$

gdzie $w(x_i)$ jest funkcją wagową.

Formułowanie funkcji celu

Funkcję celu można zapisać jako:

$$H(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \phi_j(x_i) \right]^2$$

Aby znaleźć minimalną wartość H, bierzemy pochodne cząstkowe względem każdego a_k i przyrównujemy je do zera:

$$\frac{\partial H}{\partial a_k} = 0, \quad k = 0, 1, \ldots, m$$

Wyprowadzenie układu równań normalnych

Pochodne cząstkowe funkcji celu H względem a_k wynoszą:

$$\frac{\partial H}{\partial a_k} = -2 \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \phi_j(x_i) \right] \phi_k(x_i)$$

Przyrównując do zera, otrzymujemy:

$$\sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \phi_j(x_i) \right] \phi_k(x_i) = 0, \quad k = 0, 1, \ldots, m$$

Rozwijając to równanie, mamy:

$$\sum_{i=0}^{n} w(x_i) F(x_i) \phi_k(x_i) = \sum_{i=0}^{n} w(x_i) \phi_k(x_i) \sum_{j=0}^{m} a_j \phi_j(x_i)$$

Zamieniając kolejność sumowania, uzyskujemy:

$$\sum_{i=0}^{n} w(x_i) F(x_i) \phi_k(x_i) = \sum_{j=0}^{m} a_j \sum_{i=0}^{n} w(x_i) \phi_j(x_i) \phi_k(x_i)$$

Ostatecznie otrzymujemy układ równań liniowych:

$$\sum_{j=0}^{m} a_j \sum_{i=0}^{n} w(x_i) \phi_j(x_i) \phi_k(x_i) = \sum_{i=0}^{n} w(x_i) F(x_i) \phi_k(x_i), \quad k = 0, 1, \ldots, m$$

Rozwiązanie układu równań

Otrzymany układ równań normalnych można zapisać w postaci macierzowej:

$$\mathbf{G}\mathbf{a} = \mathbf{b}$$

gdzie:

$$\mathbf{G}_{kj} = \sum_{i=0}^{n} w(x_i) \phi_j(x_i) \phi_k(x_i)$$

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}$$

$$\mathbf{b}_k = \sum_{i=0}^{n} w(x_i) F(x_i) \phi_k(x_i)$$

Macierz $\mathbf{G}$ jest macierzą Gramianą i jeśli funkcje bazowe $\phi_j(x)$ są ortogonalne, to macierz $\mathbf{G}$ jest diagonalna, co upraszcza rozwiązanie układu równań.

Przykłady wielomianów ortogonalnych

Najczęściej stosowane wielomiany ortogonalne to: - Wielomiany Czebyszewa $T_j(x)$ - Wielomiany Legendre'a $P_n(x)$ (szczególnie dla $w(x_i) = 1$)

Własność minimaksu wielomianów Czebyszewa

Wielomiany Czebyszewa $T_n(x)$ odgrywają kluczową rolę w teorii aproksymacji, szczególnie ze względu na ich własność minimaksu. Oznacza to, że spośród wszystkich wielomianów stopnia n z czynnikiem wiodącym równym 1, wielomian Czebyszewa $T_n(x)$ ma najmniejszą możliwą normę maksymalną na przedziale $[-1, 1]$.

Twierdzenie o normie wielomianu $T_n(x)$

Ze wszystkich wielomianów stopnia n z czynnikiem wiodącym równym 1, najmniejszą normę maksymalną na przedziale $[-1, 1]$:

$$\|W_n\|_\infty = \max_{x\in[-1,1]} |W_n(x)|$$

ma wielomian $2^{1-n}T_n(x)$. Norma ta wynosi 2^{1-n}.

Dowód

Założenie dowodu niewprost Załóżmy, że istnieje wielomian $p_n(x)$ stopnia n o współczynniku wiodącym równym 1, taki że:

$$\forall x \in [-1, 1] \quad |p_n(x)| < 2^{1-n}$$

Punkty ekstremalne Wielomiany Czebyszewa mają punkty ekstremalne w miejscach:

$$x'_k = \cos\left(\frac{k\pi}{n}\right), \quad k = 0, 1, \ldots, n$$

dla których $|T_n(x'_k)| = 1$. Oznacza to, że dla każdego x'_k:

$$p_n(x'_0) < 2^{1-n}\cdot T_n(x'_0), p_n(x'_1) > 2^{1-n}\cdot T_n(x'_1), p_n(x'_2) < 2^{1-n}\cdot T_n(x'_2), \vdots p_n(x'_n) > 2^{1-n}\cdot T_n(x'_n)$$

Zmiana znaku Wielomian $[p_n(x) - 2^{1-n}T_n(x)]$ powinien zmieniać znak w każdym z przedziałów (x'_{k+1}, x'_k), $k = n-1, n-2, \ldots, 1, 0$. To oznacza, że powinien być wielomianem stopnia n na przedziale $[-1, 1]$.

Sprzeczność Jednakże $p_n(x)$ i $2^{1-n}T_n(x)$ mają ten sam współczynnik wiodący, zatem ich różnica jest wielomianem stopnia $n-1$. To prowadzi do sprzeczności, ponieważ wielomian stopnia $n-1$ nie może zmieniać znaku n razy na przedziale $[-1, 1]$.

Wniosek

Zatem założenie, że istnieje wielomian $p_n(x)$ o mniejszej normie maksymalnej niż $2^{1-n}T_n(x)$, jest fałszywe. W związku z tym, ze wszystkich wielomianów stopnia n z czynnikiem wiodącym równym 1, wielomian Czebyszewa $T_n(x)$ ma najmniejszą normę maksymalną na przedziale $[-1, 1]$.

Wyznaczanie aproksymacji Padé

Aproksymacja Padé jest metodą aproksymacji funkcji za pomocą funkcji wymiernej. W przeciwieństwie do szeregu Taylora, który aproksymuje funkcję wielomianem, aproksymacja Padé stosuje stosunek dwóch wielomianów. Dzięki temu aproksymacja Padé często lepiej oddaje zachowanie funkcji, szczególnie w przypadku funkcji zbieżnych wolniej lub mających osobliwości.

Definicja aproksymacji Padé

Aproksymujemy funkcję $F(x)$ za pomocą funkcji wymiernej $r(x)$, której stopień licznika wynosi n, a mianownika m:

$$r(x) = \frac{p(x)}{q(x)} = \frac{p_0 + p_1 x + \ldots + p_n x^n}{q_0 + q_1 x + \ldots + q_m x^m}$$

Przyjmujemy, że $p(x)$ i $q(x)$ są względnie pierwsze (nie mają wspólnych dzielników) oraz, że $q_0 = 1$.

Technika aproksymacji Padé

Celem jest dobranie współczynników p_i i q_i tak, aby rozwinięcie w szereg Maclaurina funkcji $F(x)$ i $r(x)$ było jak najbardziej zgodne, czyli aby możliwie najwięcej pochodnych $F(x)$ i $r(x)$ w punkcie $x = 0$ było równych:

$$F^{(k)}(0) - r^{(k)}(0) = 0 \quad \text{dla} \quad k = 0, 1, \ldots, N$$

Wyprowadzenie układu równań

Zakładamy, że $F(x)$ można rozwinąć w szereg Maclaurina:

$$F(x) = \sum_{i=0}^{\infty} a_i x^i$$

Aproksymacja Padé zakłada, że różnica $F(x)$ i $r(x)$ jest minimalna:

$$F(x) - r(x) = \frac{F(x) \cdot q(x) - p(x)}{q(x)}$$

Aby współczynniki wielomianów $p(x)$ i $q(x)$ były zgodne, rozwinięcie $F(x) \cdot q(x) - p(x)$ powinno być równe zero dla jak największej liczby wyrazów. Przyjmując $q_0 = 1$:

$$F(x) \cdot q(x) - p(x) = \sum_{i=0}^{\infty} a_i x^i \cdot \sum_{j=0}^{m} q_j x^j - \sum_{k=0}^{n} p_k x^k$$

Rozwijamy powyższe wyrażenie do postaci:

$$F(x) \cdot q(x) = \sum_{i=0}^{\infty} a_i x^i \cdot \sum_{j=0}^{m} q_j x^j = \sum_{i=0}^{\infty} \left(\sum_{j=0}^{\min(i,m)} a_{i-j} q_j \right) x^i$$

$$p(x) = \sum_{k=0}^{n} p_k x^k$$

Równając współczynniki przy tych samych potęgach x, otrzymujemy układ równań liniowych, który pozwala znaleźć p_i i q_j.

Rozwiązanie układu równań

Dla $k = 0, 1, \ldots, N$, rozwiązujemy następujący układ równań:

$$\sum_{j=0}^{m} q_j a_{k-j} = p_k, \quad k = 0, 1, \ldots, n$$

Zakładając $q_0 = 1$, mamy:

$$a_k + \sum_{j=1}^{\min(k,m)} q_j a_{k-j} = p_k, \quad k = 0, 1, \ldots, n$$

dla $k > n$:

$$\sum_{j=1}^{m} q_j a_{k-j} = 0, \quad k = n+1, n+2, \ldots, n+m$$

Otrzymujemy układ $n + m + 1$ równań z $n + m + 1$ niewiadomymi p_i i q_i, który rozwiązujemy metodami numerycznymi.

Kwadratury elementarne: trapezów, prostokątów, Simpsona

Kwadratury elementarne są podstawowymi metodami numerycznego całkowania, które pozwalają na aproksymację wartości całki oznaczonej funkcji. W tej sekcji omówimy trzy popularne metody: prostokątów, trapezów i Simpsona.

Metoda prostokątów

Metoda prostokątów polega na aproksymacji obszaru pod krzywą przez prostokąty. Przyjmujemy, że wysokość każdego prostokąta jest równa wartości funkcji w wybranym punkcie przedziału.

Dla funkcji $f(x)$ na przedziale $[a, b]$ podzielonym na n podprzedziałów, całka jest przybliżona jako suma pól prostokątów:

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n-1} f(x_i)\Delta x$$

gdzie x_i to punkt w i-tym podprzedziale, a $\Delta x = \frac{b-a}{n}$.

Metoda trapezów

Metoda trapezów polega na aproksymacji obszaru pod krzywą przez trapezy. Dla funkcji $f(x)$ na przedziale $[a, b]$, całka jest przybliżona jako suma pól trapezów:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}[f(a) + f(b)]$$

Dla przedziału podzielonego na n równych części, formuła ta jest rozszerzona do postaci:

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{2}\left(f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right)$$

gdzie $\Delta x = \frac{b-a}{n}$.

Metoda Simpsona

Metoda Simpsona aproksymuje obszar pod krzywą za pomocą parabol. Dla funkcji $f(x)$ na przedziale $[a, b]$, całka jest przybliżona jako suma pól parabol przechodzących przez trzy kolejne punkty:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)]$$

Dla przedziału podzielonego na n równych części (gdzie n jest parzyste), formuła ta jest rozszerzona do postaci:

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}\left(f(x_0) + 4\sum_{\text{odd } i} f(x_i) + 2\sum_{\text{even } i} f(x_i) + f(x_n)\right)$$

gdzie $\Delta x = \frac{b-a}{n}$.

Kwadratura złożona Simpsona

Kwadratura złożona Simpsona to metoda numerycznego całkowania, która dzieli przedział całkowania na mniejsze podprzedziały i stosuje regułę Simpsona do każdego z nich. Jest to szczególny przypadek kwadratury Newtona-Cotesa, który zapewnia wyższą dokładność poprzez uwzględnienie więcej niż dwóch punktów w każdym podprzedziale.

Wyprowadzenie wzoru

Rozważmy przedział $[a, b]$, który dzielimy na n podprzedziałów, gdzie n jest parzystą liczbą. Wtedy:

$$h = \frac{b-a}{n}$$

Punkty podziału to:

$$x_i = a + i \cdot h, \quad i = 0, 1, \ldots, n$$

Stosując regułę Simpsona do każdego podprzedziału $[x_{2i}, x_{2i+2}]$:

$$\int_{x_{2i}}^{x_{2i+2}} f(x)\,dx \approx \frac{h}{3}\left[f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})\right]$$

Łącząc wyniki dla wszystkich podprzedziałów, otrzymujemy:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 4\sum_{i=1,\,\text{odd}}^{n-1} f(x_i) + 2\sum_{i=2,\,\text{even}}^{n-2} f(x_i) + f(x_n)\right]$$

Wzór na błąd kwadratury złożonej Simpsona

Błąd kwadratury złożonej Simpsona można wyprowadzić analizując resztę metody Simpsona dla każdego podprzedziału i sumując je. Dla funkcji $f(x)$, która ma ciągłą czwartą pochodną na $[a, b]$, błąd jest dany wzorem:

$$E_n = -\frac{b-a}{180}h^4 \cdot f^{(4)}(\xi)$$

gdzie ξ jest pewnym punktem w przedziale (a, b).
Podstawiając $h = \frac{b-a}{n}$, otrzymujemy:

$$E_n = -\frac{(b-a)}{180}\left(\frac{b-a}{n}\right)^4 \cdot f^{(4)}(\xi)$$

Uproszczając, mamy:

$$E_n = -\frac{(b-a)^5}{180n^4} \cdot f^{(4)}(\xi)$$

Węzły i wagi kwadratury Gaussa

Kwadratury Gaussa są metodą numerycznego całkowania, która wykorzystuje węzły i wagi dobrane tak, aby zapewnić maksymalną dokładność dla całek wielomianów do stopnia $2n-1$. W tej sekcji omówimy wyznaczanie węzłów i wag kwadratury Gaussa.

Węzły kwadratury Gaussa

Węzły x_i kwadratury Gaussa są zerami wielomianu ortogonalnego $\phi_n(x)$ względem danej funkcji ważącej $w(x)$ na przedziale $[a, b]$. Dla klasycznych kwadratur Gaussa, węzły są zerami wielomianów Czebyszewa, Legendre'a lub Hermite'a w zależności od wyboru funkcji ważącej $w(x)$:

$$\int_a^b w(x)\phi_n(x)\,dx = 0$$

gdzie $\phi_n(x)$ jest n-tym wielomianem ortogonalnym. Wielomiany te są definiowane na różnych przedziałach z różnymi funkcjami ważącymi: - Wielomiany Legendre'a: $w(x) = 1$, przedział $[-1, 1]$ - Wielomiany Czebyszewa: $w(x) = \frac{1}{\sqrt{1-x^2}}$, przedział $[-1, 1]$ - Wielomiany Hermite'a: $w(x) = e^{-x^2}$, przedział $(-\infty, \infty)$

Węzły są rozwiązaniami równania:

$$\phi_n(x) = 0$$

Wagi kwadratury Gaussa

Wagi a_i kwadratury Gaussa są obliczane tak, aby całka była dokładna dla wielomianów do stopnia $2n-1$. Wagi są związane z węzłami przez funkcję ważącą $w(x)$ i wielomiany Lagrange'a:

$$a_i = \int_a^b w(x)L_i(x)\,dx$$

gdzie $L_i(x)$ jest wielomianem Lagrange'a, który przyjmuje wartość 1 w węźle x_i i 0 w pozostałych węzłach. Dokładniej, wagi można wyrazić jako:

$$a_i = \int_a^b w(x) \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}\,dx$$

Z uwagi na ortogonalność wielomianów $\phi_n(x)$, wagi można również obliczyć jako:

$$a_i = \frac{\int_a^b w(x)\phi_n^2(x)\,dx}{\left(\phi_n'(x_i)\right)^2}$$

gdzie $\phi_n'(x_i)$ jest pochodną wielomianu $\phi_n(x)$ w punkcie x_i.

Przykłady wag i węzłów

1. **Kwadratura Gaussa-Legendre'a:** - Węzły są zerami wielomianów Legendre'a $P_n(x)$. - Wagi są obliczane z:

$$a_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2}$$

2. **Kwadratura Gaussa-Czebyszewa:** - Węzły są zerami wielomianów Czebyszewa $T_n(x)$. - Wagi są równe:

$$a_i = \frac{\pi}{n}$$

3. **Kwadratura Gaussa-Hermite'a:** - Węzły są zerami wielomianów Hermite'a $H_n(x)$. - Wagi są obliczane z:

$$a_i = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2}$$

Twierdzenie o zbieżności procesu iteracyjnego $x = \phi(x)$

Twierdzenie o zbieżności procesu iteracyjnego jest fundamentalnym narzędziem w analizie metod iteracyjnych stosowanych do rozwiązywania równań nieliniowych. Twierdzenie to określa warunki, pod którymi ciąg iteracyjny zbiega do rozwiązania równania $x = \phi(x)$.

Twierdzenie o zbieżności procesu iteracyjnego

Niech $x = \phi(x)$ ma pierwiastek α. Zakładamy, że: 1. Funkcja $\phi(x)$ jest ciągła na przedziale $I = [\alpha - a, \alpha + a]$. 2. Funkcja $\phi(x)$ spełnia warunek Lipschitza na I: $\exists L < 1$ takie, że $|\phi'(x)| \leq L$ dla każdego $x \in I$.

Wówczas dla dowolnego punktu startowego $x_0 \in I$: 1. Ciąg $\{x_i\}$ zdefiniowany przez $x_{i+1} = \phi(x_i)$ jest zawarty w I, tj. $x_i \in I$ dla $i = 1, 2, \ldots$. 2. Ciąg $\{x_i\}$ zbiega do α, tj. $\lim_{i\to\infty} x_i = \alpha$. 3. α jest jedynym pierwiastkiem równania $x = \phi(x)$ w przedziale I.

Dowód twierdzenia

1. Zawartość w I Zakładamy, że $x_{i-1} \in I$. Wówczas różnicę $x_i - \alpha$ można wyrazić jako:

$$x_i - \alpha = \phi(x_{i-1}) - \phi(\alpha)$$

Z twierdzenia o wartości średniej wynika, że istnieje $\eta_{i-1} \in I$ takie, że:

$$\phi'(\eta_{i-1}) = \frac{\phi(x_{i-1}) - \phi(\alpha)}{x_{i-1} - \alpha}$$

Stąd mamy:

$$x_i - \alpha = (x_{i-1} - \alpha) \cdot \phi'(\eta_{i-1})$$

Biorąc wartość bezwzględną, otrzymujemy:

$$|x_i - \alpha| = |x_{i-1} - \alpha| \cdot |\phi'(\eta_{i-1})| \leq |x_{i-1} - \alpha| \cdot L$$

Ponieważ $L < 1$ oraz $x_{i-1} \in I$, wynika z tego, że $x_i \in I$.

2. Zbieżność ciągu Stosując powyższe nierówności wielokrotnie, otrzymujemy:

$$|x_i - \alpha| \leq L \cdot |x_{i-1} - \alpha| \leq L^2 \cdot |x_{i-2} - \alpha| \leq \ldots \leq L^i \cdot |x_0 - \alpha|$$

Ponieważ $L < 1$, więc $\lim_{i\to\infty} L^i = 0$, co implikuje, że:

$$\lim_{i\to\infty} |x_i - \alpha| = 0 \quad \text{czyli} \quad \lim_{i\to\infty} x_i = \alpha$$

3. Jedyność pierwiastka Zakładamy, że oprócz α istnieje jeszcze jeden pierwiastek β w I. Wówczas mamy:

$$\alpha - \beta = \phi(\alpha) - \phi(\beta) = (\alpha - \beta) \cdot \phi'(\eta)$$

gdzie $\eta \in I$. Biorąc wartość bezwzględną, otrzymujemy:

$$|\alpha - \beta| = |\alpha - \beta| \cdot |\phi'(\eta)| < |\alpha - \beta|$$

co prowadzi do sprzeczności, ponieważ $|\phi'(\eta)| < 1$.

Metoda Newtona-Raphsona i jej rząd zbieżności

Metoda Newtona-Raphsona jest jedną z najczęściej używanych metod numerycznych do znajdowania pierwiastków równań nieliniowych. Opiera się na iteracyjnym przybliżaniu pierwiastka równania $f(x) = 0$ przy użyciu pochodnych funkcji.

Wzór iteracyjny metody Newtona-Raphsona

Rozważmy równanie $f(x) = 0$. Wybieramy początkowe przybliżenie x_0 i używamy następującego wzoru iteracyjnego, aby znaleźć lepsze przybliżenie pierwiastka:

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

Iterację kontynuujemy, aż do osiągnięcia zadowalającej dokładności.

Wyprowadzenie wzoru

Niech x_i będzie aktualnym przybliżeniem pierwiastka α. Aby znaleźć x_{i+1}, rozwijamy $f(x)$ w szereg Taylora wokół x_i:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{f''(x_i)}{2}(x - x_i)^2 + \cdots$$

Ponieważ szukamy pierwiastka, czyli $f(x) = 0$, przybliżamy:

$$0 \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

Rozwiązując równanie dla x_{i+1}:

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

Warunki zbieżności

Metoda Newtona-Raphsona jest zbieżna, jeżeli: 1. Funkcja $f(x)$ jest różniczkowalna w otoczeniu pierwiastka α. 2. Pierwsza pochodna $f'(x)$ nie znika w otoczeniu α.

Dla x bliskiego α, $f'(\alpha) \neq 0$. Dodatkowo, aby zapewnić zbieżność, powinna istnieć otoczenie α, w którym $|\phi'(x)| < 1$, gdzie $\phi(x) = x - \frac{f(x)}{f'(x)}$.

Rząd zbieżności metody Newtona-Raphsona

Rząd zbieżności określa, jak szybko metoda zbliża się do pierwiastka. Metoda Newtona-Raphsona ma zbieżność kwadratową. Oznacza to, że błąd iteracji $\epsilon_i = x_i - \alpha$ w przybliżeniu jest proporcjonalny do kwadratu błędu poprzedniej iteracji ϵ_{i-1}:

$$\epsilon_{i+1} \approx C\epsilon_i^2$$

gdzie C jest stałą proporcjonalności.

Dowód zbieżności kwadratowej

Rozważmy funkcję iteracyjną $\phi(x) = x - \frac{f(x)}{f'(x)}$. Pochodne tej funkcji są:

$$\phi'(x) = 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2}$$

W punkcie α, gdzie $f(\alpha) = 0$, mamy:

$$\phi'(\alpha) = 0$$

Druga pochodna $\phi(x)$ wynosi:

$$\phi''(x) = \frac{f'(x)f''(x) - f(x)f'''(x)}{(f'(x))^3}$$

Ponieważ $f(\alpha) = 0$, w punkcie α:

$$\phi''(\alpha) = \frac{f''(\alpha)}{f'(\alpha)^2} \neq 0$$

Dlatego rozwinięcie w szereg Taylora dla błędu ϵ_i jest:

$$\epsilon_{i+1} = \phi(x_i) - \alpha \approx \frac{\phi''(\alpha)}{2}\epsilon_i^2$$

To dowodzi, że metoda Newtona-Raphsona ma zbieżność kwadratową:

$$\epsilon_{i+1} \approx C\epsilon_i^2$$

gdzie $C = \frac{\phi''(\alpha)}{2}$.

Metoda siecznych i jej rząd zbieżności

Metoda siecznych jest numeryczną metodą znajdowania pierwiastków równań nieliniowych, która nie wymaga obliczania pochodnych funkcji. Jest to iteracyjna metoda, która wykorzystuje styczne do funkcji w dwóch kolejnych przybliżeniach pierwiastka.

Wzór iteracyjny metody siecznych

Rozważmy równanie $f(x) = 0$. Wybieramy dwa początkowe przybliżenia x_0 i x_1. Wzór iteracyjny metody siecznych jest dany przez:

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Iterację kontynuujemy, aż do osiągnięcia zadowalającej dokładności.

Wyprowadzenie wzoru

Metoda siecznych wykorzystuje liniową interpolację do przybliżenia pierwiastka. Przybliżenie x_{i+1} jest punktem przecięcia osi x prostej przechodzącej przez punkty $(x_{i-1}, f(x_{i-1}))$ oraz $(x_i, f(x_i))$. Równanie prostej można zapisać jako:

$$y - f(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}(x - x_i)$$

Z warunku $y = 0$ (poszukujemy punktu przecięcia z osią x) otrzymujemy:

$$0 - f(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}(x_{i+1} - x_i)$$

Rozwiązując równanie dla x_{i+1}:

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Warunki zbieżności

Metoda siecznych jest zbieżna, jeżeli: 1. Funkcja $f(x)$ jest różniczkowalna w otoczeniu pierwiastka α. 2. Pierwsza pochodna $f'(x)$ nie zanika w otoczeniu α.

W porównaniu z metodą Newtona-Raphsona, metoda siecznych wymaga tylko obliczenia wartości funkcji, co może być korzystne w przypadkach, gdy obliczenie pochodnych jest trudne lub niemożliwe.

Rząd zbieżności metody siecznych

Rząd zbieżności metody siecznych wynosi około 1.62, co oznacza, że metoda ta jest nadliniowa, ale wolniejsza od metody Newtona-Raphsona, która ma zbieżność kwadratową.

Dowód rzędu zbieżności

Niech α będzie pierwiastkiem $f(x) = 0$, a $\epsilon_i = x_i - \alpha$ będzie błędem i-tej iteracji. Rozwijając $f(x)$ w szereg Taylora wokół α, mamy:

$$f(x_i) = f(\alpha + \epsilon_i) = f'(\alpha)\epsilon_i + \frac{f''(\xi_i)}{2}\epsilon_i^2 \quad \text{dla} \quad \xi_i \in (\alpha, x_i)$$

Dla x_{i-1}:

$$f(x_{i-1}) = f(\alpha + \epsilon_{i-1}) = f'(\alpha)\epsilon_{i-1} + \frac{f''(\xi_{i-1})}{2}\epsilon_{i-1}^2 \quad \text{dla} \quad \xi_{i-1} \in (\alpha, x_{i-1})$$

Podstawiając te wyrażenia do wzoru iteracyjnego metody siecznych, otrzymujemy:

$$x_{i+1} - \alpha = x_i - \alpha - \frac{(f'(\alpha)\epsilon_i + \frac{f''(\xi_i)}{2}\epsilon_i^2)(\epsilon_i - \epsilon_{i-1})}{f'(\alpha)(\epsilon_i - \epsilon_{i-1}) + \frac{f''(\xi_i)\epsilon_i^2 - f''(\xi_{i-1})\epsilon_{i-1}^2}{2}}$$

Uproszczając i zaniedbując wyrazy wyższego rzędu, otrzymujemy:

$$\epsilon_{i+1} \approx C\epsilon_i\epsilon_{i-1}$$

co wskazuje na nadliniowy rząd zbieżności.

Metody iteracyjne: Jacobiego, Gaussa-Seidla, SOR, Czebyszewa

W tej sekcji przedstawimy wzory robocze i zapisy macierzowe dla różnych metod iteracyjnych używanych do rozwiązywania układów równań liniowych $Ax = b$.

Metoda Jacobiego

Metoda Jacobiego jest jedną z najprostszych metod iteracyjnych. Opiera się na rozdzieleniu macierzy A na macierz diagonalną D, macierz dolną trójkątną L, i macierz górną trójkątną U:

$$A = D + L + U$$

Wyprowadzenie wzoru

Układ równań $Ax = b$ można zapisać jako:

$$(D + L + U)x = b$$

Rozbijając x na poprzednią iterację $x^{(t)}$ i nową iterację $x^{(t+1)}$, mamy:

$$Dx^{(t+1)} = b - (L + U)x^{(t)}$$

Rozwiązując to równanie względem $x^{(t+1)}$, otrzymujemy wzór iteracyjny metody Jacobiego:

$$x^{(t+1)} = D^{-1}(b - (L + U)x^{(t)})$$

Elementy $x^{(t+1)}$ są obliczane jako:

$$x_i^{(t+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(t)} \right), \quad i = 1, \ldots, n$$

Metoda Gaussa-Seidla

Metoda Gaussa-Seidla jest modyfikacją metody Jacobiego, która wykorzystuje już zaktualizowane wartości x w bieżącej iteracji:

$$A = D + L + U$$

Wyprowadzenie wzoru

Układ równań $Ax = b$ zapisujemy jako:

$$(D + L + U)x = b$$

Wykorzystując aktualizowane wartości x, mamy:

$$(D + L)x^{(t+1)} = b - Ux^{(t)}$$

Rozwiązując to równanie względem $x^{(t+1)}$, otrzymujemy wzór iteracyjny metody Gaussa-Seidla:

$$x^{(t+1)} = (D + L)^{-1}(b - Ux^{(t)})$$

Elementy $x^{(t+1)}$ są obliczane jako:

$$x_i^{(t+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(t+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(t)} \right), \quad i = 1, \ldots, n$$

Metoda SOR (Successive Over-Relaxation)

Metoda SOR jest rozszerzeniem metody Gaussa-Seidla z dodatkowym parametrem relaksacji ω w celu przyspieszenia zbieżności.

Wyprowadzenie wzoru

Rozważamy równanie iteracyjne:

$$x_i^{(t+1)} = (1 - \omega)x_i^{(t)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(t+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(t)} \right)$$

W zapisie macierzowym:

$$Dx^{(t+1)} = (1 - \omega)Dx^{(t)} + \omega[b - (Lx^{(t+1)} + Ux^{(t)})]$$

Metoda Czebyszewa

Metoda Czebyszewa stosuje zmieniające się parametry relaksacji ω zgodnie z wielomianami Czebyszewa, aby przyspieszyć zbieżność.

Wyprowadzenie wzoru

Stosujemy wzór roboczy SOR:

$$x_i^{(t+1)} = x_i^{(t)} + \omega \left(\frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(t+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(t)} \right) \right)$$

Parametry ω zmieniają się w każdym kroku zgodnie z zależnościami rekurencyjnymi opartymi na przeskalowanych wielomianach Czebyszewa:

$$\omega^{(0)} = 1$$

$$\omega^{\left(t+\frac{1}{2}\right)} = \frac{1}{1 - \frac{1}{4}\rho^2 \omega^{(t)}}$$

gdzie ρ jest promieniem spektralnym macierzy iteracji M_J.

Literatura

[1] M. Bubak, K. Rycerz, *Metody Obliczeniowe w Nauce i Technice*, AGH University of Science and Technology.