Spis treści

1 Arytmetyka komputerowa

1.1 Omów reprezentację zmiennoprzecinkową

F - zbiór liczb zmiennoprzecinkowych (floating-point)
β - podstawa systemu (np 2 dla systemu dwójkowego)
t - dokładność
L, U - dolny i górny zakres wykładnika
d_i - liczby całkowite takie, że 0 <= d_i <= β - 1 dla i = 1, .., t
e - cecha taka, że L <= e <= U

$$x \in F <=> x = \pm\left(\frac{d_1}{\beta} + ... + \frac{d_t}{\beta^t}\right) \cdot \beta^e = \pm \sum_{i=1}^{t} \frac{d_i}{\beta^i} \cdot \beta^e$$

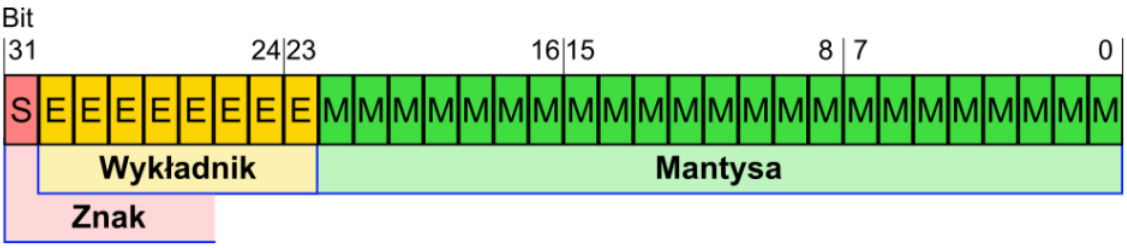
System F jest unormowany gdy nie istnieje taki x, dla którego $d_1 = 0$

Standaryzacja - just in case
W celu ujednolicenia zasad operacji na liczbach zmiennoprzecinkowych na różnych platformach sprzętowych, opracowano standard IEEE 754, w oparciu o który realizuje się obecnie wszystkie implementacje sprzętowe liczb zmiennoprzecinkowych. Definiuje on dwie klasy liczb:

- pojedynczej precyzji (ang. single, single precision [bity: 1-znak, 8-wykładnik, 23-bity]),
- podwójnej precyzji (ang. double, double precision) [bity: 1-znak, 11-wykładnik, 52-mantysa].

Mantysa jest znormalizowana, tj. należy do przedziału [1, β). Jeżeli mantysa jest stała, a e zmienia się, wówczas przesunięciu ulega przecinek

Przykład - liczba zmiennoprzecinkowa pojedynczej precyzji (float32)



1.2 Wyprowadź wzór na liczbę elementów zbioru liczb zmiennoprzecinkowych

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

2 - znak liczby $\pm$
(β - 1) - liczba mozliwosci na pierwszym bicie mantysy (o jeden mniej, bo unormowanie – nie ma zera)
β^{t-1} - liczba możliwości na pozostałych t-1 bitach
(U - L + 1) - zakres wykładnika
1 - zero

1.3 Wyprowadź wzór na błąd względny reprezentacji zmiennoprzecinkowej

Oznaczenia:
$x = (1.a_1a_2...)_2 * \beta^m = q * \beta^m$, $1 \leq q < 2$ - dodatnia (dla uproszczenia) liczba rzeczywista
$x_- = (1.a_1a_2...a_t)_2 * \beta^m$ - obcięcie liczby x

$x_+ = [(1.a_1a_2...a_t)_2 + \beta^{-t}] * \beta^m$ - zaokrąglenie w górę liczby x
$fl(x)$ - reprezentacja zmiennoprzecinkowa x, ta z liczb x_- i x_+, która jest bliższa x

Obliczenia:
$x_+ - x_- = \beta^{m-t}$
Niezależnie od tego, czy bliższe jest x_-, czy x_+, to największy możliwy błąd to $\frac{1}{2}|x_+ - x_-|$, który występuje wtedy, kiedy x leży dokładnie pomiędzy x_+ i x_-. Wynika z tego:

$$|x - fl(x)| \leq \frac{1}{2}|x_+ - x_-| = \frac{1}{2}\beta^{m-t}$$

Błąd względny:

$$\left|\frac{x - fl(x)}{x}\right| \leq \frac{\frac{1}{2}\beta^{m-t}}{q * \beta^m} = \frac{1}{2q}\beta^{1-t} \leq \frac{1}{2}\beta^{1-t}$$

1.4 Wyjaśnij pojęcia: nadmiar, niedomiar, błędy obcięcia

Nadmiar i niedomiar - Wystepuje, gdy ilosc bitów potrzebna do reprezentacji cechy danej liczby jest za mała.

Nadmiar (overflow) - wystepuje jesli wartosc bezwzgledna liczby jest za duza, aby ja reprezentowac (nie mozna juz zwiekszac cechy, bo brakuje bitów)

Niedomiar (underflow) - wystepuje jesli wartosc bezwzgledna liczby jest za mała, aby ja reprezentowac (nie mozna juz zmniejszac cechy, bo brakuje bitów)

Błąd obcięcia - wynika z przybliżenia wartości matematycznie nieskończonej przez skończone przybliżenie i obcięcie reszty elementów. W metodach numerycznych używa się skończonej liczby operacji matematycznych, co powoduje powstawanie błędu metody, np. przy przybliżaniu nieskończonej sumy szeregu sumą skończonej liczby elementów, przybliżanie pochodnej przez iloraz różnicowy lub przybliżanie całki sumą skończoną

1.5 Podaj definicję i napisz, co określa maszynowe epsilon?

Definicja
Najmniejsza liczba zmiennoprzecinkowa, dla której zachodzi:

$$1 \bigoplus \epsilon > 1$$

$\bigoplus$ - dodawanie zmiennopozycyjne

Co określa
Maszynowe epsilone to wartość określająca precyzję obliczeń numerycznych wykonywanych na liczbach zmiennoprzecinkowych

1.6 Omów własności numerycznej reprezentacji liczb rzeczywistych i arytmetyki zmiennoprzecinkowej

1. **Skończona precyzja reprezentacji liczb** - każda liczba jest reprezentowana na skończonej liczbie bitów, więc liczby od samego początku są obciążone błędem reprezentacji ($fl(x) = x*(1+\epsilon_{maszynowe})$), więc algorytmy trzeba projektować tak, aby były stabilne i nie kumulowały błędów reprezentacji, żeby wyniki były poprawne.
2. **Nie można porównywać równości liczb bezpośrednio** - sprawdzanie równości liczb zmiennoprzecinkowych przez $==$ prawie nigdy nie da prawidłowego wyniku (bo mogą różnić się na samych ostatnich bitach przez same błędy reprezentacji i działań), więc w celu sprawdzenia równości trzeba sprawdzać, czy różnią się co najwyżej o niewielkie δ.

3. **Przesuwanie przecinka podczas operacji** - przecinek jest przesuwany w stronę większej z liczb, co powoduje utratę informacji o ostatnich bitach mniejszej z liczb, np. dla dużego A i małego C może wystąpić A-A+C=C i A+C-A=0; z tego też powodu algorytmy trzeba projektować tak, żeby np. najpierw dodawały małe liczby, a dopiero potem duże, tak, żeby wykonywać działania na liczbach o zbliżonym w miarę możliwości rzędzie wielkości.
4. **Zakres reprezentowanych liczb jest skończony** - nie da się reprezentować liczb nieskończenie dużych ani małych (w sensie modułu), więc można przepełnić zakres z góry lub z dołu (*overflow*) lub zejść zbyt blisko zera (*underflow*); żeby tego uniknąć, trzeba projektować algorytmy z właściwą kolejnością obliczeń i np. najpierw mnożyć małe liczby przez duże: dla dużych A i B oraz małego C (A*B)*C może dać overflow, ale A*(B*C) jest już bezpieczniejsze.
5. **Brak łączności i rozdzielności działań** - w przeciwieństwie do zwykłej arytmetyki działania są co prawda przemienne, ale nie łączne ani rozdzielne, więc nie można projektować algorytmów opierając się o te własności.

1.7 Podaj definicje i objaśnij na przykładach pojęcia: zadanie, algorytm, realizacja zmiennoprzecinkowa algorytmu

Zadanie - dla danych postaci $\vec{d} = (d_1, d_2, ..., d_n) \in R_d$ znaleźć wynik postaci $\vec{w} = (w_1, w_2, ..., w_n) \in R_w$, gdzie $\vec{w} = \varphi(\vec{d})$. R_d i R_w to skończenie wymiarowe, unormowane przestrzenie kartezjańskie, a φ to odwzorowanie ciągłe postaci $\varphi : D_0 \subset R_d \to R_w$.
Przykład: zadanie posortowania danego zbioru liczb.

Algorytm - sposób wyznaczenia wyniku zadania postaci $\vec{w} = \varphi(\vec{d})$, gdzie zadanie należy do klasy zadań $\{\varphi, D\}$, $d \in D \subset D_0$ i rozwiązanie zadania jest dokładne (w zwykłej arytmetyce).
Przykład: algorytm quicksort pozwalający wyznaczyć posortowany zbiór liczb na podstawie danego.

Realizacja zmiennoprzecinkowa algorytmu - sposób realizacji algorytmu A w postaci $fl(A(\vec{d}))$, gdzie d, x... zastępuje się $rd(d)$, $rd(x)$... (gdzie rd oznacza liczbę maszynową, komputerową reprezentację rzeczywistej liczby) oraz zwykłą arytmetykę zastępuje arytmetyka zmiennoprzecinkowa. Oczekuje się, że dane i wyniki będą prezentowane z minimalnym błędem, tzn. błąd względny danych i wyniku będą co najwyżej rzędu $k * \beta^{1-t}$ (β - podstawa systemu, k - niewielka stała, ≈ 10, t - dokładność reprezentacji, $1-t$ - względna dokładność).
Przykład: algorytm quicksort operujący na liczbach zmiennoprzecinkowych z użyciem arytmetyki zmiennoprzecinkowej (ma wpływ na jego działanie np. przy porównywaniu liczb).

1.8 Podaj definicje i objaśnij na przykładach pojęcia: uwarunkowanie zadania, poprawność numeryczna algorytmu, stabilność numeryczna algorytmu

Uwarunkowanie zadania - czułość zadania na zaburzenie danych wejściowych. Charakteryzuje je wskaźnik uwarunkowania zadania, który mówi, jak niewielkie zaburzenie danych wejściowych wpłynie na zaburzenie wyników. Zadanie jest źle uwarunkowane, jeśli niewielka zmiana danych wejściowych powoduje dużą zmianę danych wyjściowych. Jest powodowane użyciem arytmetyki zmiennoprzecinkowej i zastąpieniem danych d_i przez $rd(d_i) = d_i * (1 + \epsilon_i)$.
Można je opisać jakościowo (rysunek dwóch przecinajacych się prostych na płaszczyźnie - pod im mniejszym kątem się przecinają, tym gorzej uwarunkowane jest zadanie) lub ilościowo (wzór $\left|\frac{x*f'(x)}{f(x)}\right|$).

Przykład: zadanie wyznaczenia iloczynu skalarnego wektorów $\vec{x} * \vec{y} = \sum_{i=1}^{n} x_i * y_i \neq 0$:

$$x_i \to x_i * (1 + \alpha_i) \qquad y_i \to y_i * (1 + \beta_i)$$

$$\underbrace{\left|\frac{f(x+\alpha,y+\beta)-f(x,y)}{f(x,y)}\right|}_{\text{błąd względny}} = \left|\frac{\sum_{i=1}^{n}[x_i*(1+\alpha_i)]*[y_i*(1+\beta_i)]-\sum_{i=1}^{n}x_i*y_i}{\sum_{i=1}^{n}x_i*y_i}\right| \approx \left|\frac{\sum_{i=1}^{n}x_i*y_i*(\alpha_i+\beta_i)}{\sum_{i=1}^{n}x_i*y_i}\right| \leq$$

$$\leq max|\alpha_i+\beta_i| * \underbrace{\frac{\sum_{i=1}^{n}|x_i*y_i|}{|\sum_{i=1}^{n}x_i*y_i|}}_{cond(\vec{x}*\vec{y})}$$

Powyżej $cond(\vec{x}*\vec{y})$ oznacza współczynnik uwarunkowania zadania, czyli część wyniku powstałego po oszacowaniu błędu względnego niezależną od błędu reprezentacji (zależną tylko od danych wejściowych x i y). Gdy wszystkie x_i, y_i są tego samego znaku, to $cond(\vec{x}*\vec{y})=1$.

Przykład 2: uwarunkowanie zadania dla $f(x)=\sqrt{x}$.

$$cond(f(x)) = \left|\frac{x*f'(x)}{f(x)}\right| = \left|\frac{x*\frac{1}{2\sqrt{x}}}{\sqrt{x}}\right| = \frac{1}{2}$$

To zadanie jest dobrze uwarunkowane, bo ma niewielki, stały współczynnik uwarunkowania.

Przykład 3: uwarunkowanie zadania dla $f(x)=\frac{1}{1-x}$.

$$cond(f(x)) = \left|\frac{x*f'(x)}{f(x)}\right| = \left|\frac{x*\frac{1}{(1-x)^2}}{\frac{1}{1-x}}\right| = \frac{x}{1-x}$$

To zadanie może być źle uwarunkowane dla x bliskich 1, np. $x=(1+10^{-6}) \rightarrow cond \approx 10^6$.

Przykład 4: uwarunkowanie zadania dla $(x-2)^2=10^{-6}$. Dla $x_{1,2}=2\mp 10^{-3}$, ale zmiana stałej o zaledwie 10^{-6} powoduje zmianę $x_{1,2}$ o aż 10^{-3}, z czego wynika, że zadanie jest źle uwarunkowane.

Poprawność numeryczna algorytmu - algorytm nazywamy numerycznie poprawnym, gdy dla nieco zaburzonych danych (na poziomie reprezentacji, rzędu błędu reprezentacji) dają tylko nieco zaburzone rozwiązania, są to algorytmy numeryczne najwyższej jakości. Algorytm jest numerycznie poprawny w klasie zadań $\{\varphi, D\}$, jeżeli istnieją takie stałe (wskaźniki kumulacji algorytmu) K_d, K_w, że:

- $\forall \vec{d} \in D$ - każdy zestaw poprawnych danych należy do D,
- dla każdej dostatecznie silnej arytmetyki β^{1-t} (β - podstawa systemu liczbowego, t - dokładność reprezentacji, $1-t$ - względna dokładność) $\exists \tilde{d} \in D_0$ (zestaw zaburzonych danych) takie, że:

$$\left\|\frac{\vec{d}-\tilde{d}}{\vec{d}}\right\| \leq \delta_d * K_d$$

$$\left\|\frac{\varphi(\tilde{d})-fl(A(\vec{d}))}{\varphi(\tilde{d})}\right\| \leq \delta_w * K_w$$

$\varphi(\tilde{d})$ to dokładne rozwiązanie dla danych zaburzonych. Wzory to zależności dla danych i wyniku. Z pierwszego wynika, że błąd względny między danymi zaburzonymi a oryginalnymi jest co najwyżej równy iloczynowi maksymalnego błędu reprezentacji danych (δ_d) oraz wskaźnika kumulacji algorytmu dla danych K_d. Drugi wzór jest analogiczny, ale dla wyniku. Wskaźniki kumulacji K_d, K_w:

- mają być prawdziwe dla dowolonych danych z klasy zadań $\{\varphi, D\}$,

- im mniejsze, tym lepszy algorytm (bo tym dokładniejszy jest wtedy algorytm przy danych zaburzonych).

Przykład 1: algorytm sumacyjny Kahana, bo jego błąd wynosi $O(1)$ i jest na poziomie reprezentacji.

Przykład 2: poprawność numeryczna algorytmu wyznaczającego iloczyn skalarny wektorów $\vec{x} * \vec{y} = \sum_{i=1}^{n} x_i * y_i$.

Reprezentacja zmiennoprzecinkowa danych:
$a_i \to \hat{a_i} = rd(a_i) = a_i * (1 + \alpha_i)$
$b_i \to \hat{b_i} = rd(b_i) = b_i * (1 + \beta_i)$

Działania w arytmetyce zmiennoprzecinkowej:
Dla $i = 1$: $fl(A(\vec{a}, \vec{b})) = \hat{a_1} * \hat{b_1} * (1 + \epsilon_1)$
Dla $i = 2$: $fl(A(\vec{a}, \vec{b})) = [\hat{a_1} * \hat{b_1} * (1 + \epsilon_1) + \hat{a_2} * \hat{b_2} * (1 + \epsilon_2)] * (1 + \delta_2)$
Dla $i = 3$: $fl(A(\vec{a}, \vec{b})) = \{[\hat{a_1} * \hat{b_1} * (1 + \epsilon_1) + \hat{a_2} * \hat{b_2} * (1 + \epsilon_2)] * (1 + \delta_2) + \hat{a_3} * \hat{b_3} * (1 + \epsilon_3)\} * (1 + \delta_3)$ Ogólnie:

$$fl(A(\vec{a}, \vec{b})) = \sum_{i=1}^{n} \underbrace{a_i * (1 + \alpha_i)}_{\tilde{a_i}} * \underbrace{b_i * (1 + \beta_i) * (1 + \epsilon_i) * \prod_{j=i}^{n} (1 + \delta_j)}_{\tilde{b_i}}$$

Interpretacja wyniku:

- dla dokładnego wyniku $K_w = 0$
- dla zaburzonych danych:

$$\left\| \frac{\vec{a} - \tilde{a}}{\vec{a}} \right\| \leq \beta^{1-t} \quad (K_{d1} = 1)$$

$$\left\| \frac{\vec{b} - \tilde{b}}{\vec{b}} \right\| \leq (n + 1) * \beta^{1-t} \quad (K_{d2} = n + 1)$$

- jeśli pominięto by błędy reprezentacji danych, to K_d byłoby mniejsze o 1

Jako że wskaźniki kumulacji są rzędu liczby działań, to algorytm jest użyteczny (jest też więc oczywiście poprawny numerycznie).

Stabilność numeryczna algorytmu - algorytm nazywamy numerycznie stabilnym, gdy mały błąd na dowolnym etapie przenosi się dalej z malejącą amplitudą, czyli jakość wyniku poprawia się z każdym kolejnym etapem obliczeń, czyli:

$$\epsilon^{n+1} = g * \epsilon^n \to \text{stabilna, bo } \left|\epsilon^{n+1}\right| \leq \left|\epsilon^n\right|,\ g \in (0, 1) \text{ - współczynnik wzmocnienia}$$

Algorytm nazywany numerycznie stabilnym w klasie $\{\varphi, D\}$, jeżeli istnieje mała stała K taka, że $\forall \vec{d} \in D$ oraz dla każdej dostatecznie silnej arytmetyki zachodzi:

$$\|\varphi(\vec{d}) - fl(A(\vec{d}))\| \leq K * P(\vec{d}, \varphi), \quad P(\vec{d}, \varphi) = \delta_w \|\vec{w}\| + \max_{\hat{d}} \|\varphi(d) - \varphi(\hat{d})\|$$

Powyżej P to optymalny poziom błędu rozwiązania $\varphi(\vec{d})$ w arytmetyce fl.
Aby algorytm mógł być numerycznie poprawny, musi być najpierw numerycznie stabilny (jest to minimalny wymóg).
Przykład: algorytm obliczania całki $E_n = \int_0^1 x^n * e^{x-1} dx, \quad n = 1, 2, ...$
Po całkowaniu przez części

$$E_n = \begin{cases} E_1 = \frac{1}{e}, & n = 1 \\ 1 - n * E_{n-1}, & n = 2, 3, ... \end{cases}$$

W tej wersji algorytm w części rekurencyjnej jest niestabilny, bo każde mnożenie zwiększa błąd n razy. Można go jednak poprawić, przekształcając ją do postaci:

$$E_{n-1} = \frac{1 - E_n}{n}$$

Dzięki dzieleniu błąd w każdym kroku zmniejsza się n - otrzymujemy algorytm stabilny.

1.9 Wyznacz wskaźnik uwarunkowania podanego zadania

Niezależnie od zadania korzysta się z metod podanych w przykładach do definicji wskaźnika uwarunkowania. Wskazówki:

- zadanie sprowadza się do rozdzielenia części z samymi x, y i z samymi α, β i tym samym do znalezienia $cond(x, y)$
- zaczyna się zwykle od użycia któregoś ze wzorów: na błąd względny ($\left|\frac{f^* - f}{f}\right|$), na współczynnik uwarunkowania dla funkcji ($\left|\frac{x * f'}{f}\right|$)
- aby dostać wartość przybliżoną (z błędem), trzeba za każdy x podstawić $x * (1 + \alpha)$, za każdy y podstawić $y * (1 + \beta)$ itd.
- trzeba używać operacji $\leq$ i $\approx$ i dowolnych poprawnych przekształceń matematycznych

1.10 Wyznacz wskaźnik kumulacji podanego algorytmu

Niezależnie od zadania korzysta się z metod podanych w przykładzie do definicji wskaźnika kumulacji. Wskazówki:

- dla danych oryginalnych x i zaburzonych $\hat{x}$ trzeba obliczyć wyniki dokładne i fl
- δ to zwykle po prostu precyzja arytmetyki, czyli β^{1-t}
- mając błędy względne dla danych i wyników oraz δ, wystarczy podstawić do wzoru i znaleźć współczynniki K_d i K_w

2 Interpolacja

2.1 Wyprowadź wzór na interpolację metodą Lagrange'a

Szukamy wielomianu postaci $\sum_{k=0}^{n} y_k * l_k(x)$, gdzie l_k to wielomiany zależne od danych węzłów interpolacji $x_0, x_1, x_2, ..., x_n$,ale nie od wartości $y_0, y_1, y_2, ..., y_n$.
Gdyby jedna z tych wartości y_i była równa 1, a pozostałe by znikały, to można by użyć do wyrażenia wartości delty Kroneckera:

$$\delta_{ij} = p_n(x_j) = \sum_{k=0}^{n} y_k * l_k(x_j) = \sum_{k=0}^{n} \delta_{ki} * l_k(x_j) = l_i(x_j) \quad (0 \leq j \leq n)$$

Wielomian l_i o powyższej własności łatwo znaleźć, bo wystarczy, że jego zerami będą wszystkie węzły poza x_i (ten, dla którego y_i jest niezerowe). Oznaczając jakąś stałą przez c można zapisać, że:

$$l_i(x) = c * (x - x_0) * (x - x_1) * ... * (x - x_{i-1}) * (x - x_{i+1}) * ... * (x - x_n)$$

Wartość stałej c wynika z nałożonego warunku $l_i(x_i) = 1$, a więc:

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \quad (0 \leq i \leq n)$$

Z powyższego wynika ostatecznie wzór na wielomian interpolacyjny Lagrange'a:

$$\sum_{i=0}^{n} y_i * \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

2.2 Wyprowadź wzór na błąd interpolacji metodą Lagrange'a

Zakładamy, że funkcja interpolowana $f(x)$ jest $n+1$ razy różniczkowalna na wybranym zamkniętym przedziale $[a, b]$, a interpoluje ją wielomian $p(x)$ (stopnia co najwyżej n) na $n + 1$ punktach $x_0, x_1, x_2, ..., x_n$.
Oznaczamy poszukiwany błąd przez $R(x) = f(x) - p(x)$ (dla $f(x) = p(x)$ w węzłach interpolacji błąd wynosi oczywiście 0). Wprowadzamy także funkcje pomocnicze:

$$w(x) = \prod_{i=0}^{n}(x - x_i) \qquad w(t) = \prod_{i=0}^{n}(t - x_i)$$

$$Y(t) = R(t) - \lambda * w(t)$$

W powyższej funkcji λ to taka liczba rzeczywista, dla której $Y(x) = 0$ (gdzie to x jest ustalone), zatem:

$$\lambda = \frac{R(x)}{w(x)}$$

$$Y(t) = R(t) - \frac{R(x)}{w(x)} * w(t)$$

Skoro węzły interpolacji x_i to pierwiastki $R(t)$ i $w(t)$, to $Y(x) = Y(x_i) = 0$. czyli Y ma przynajmniej $n + 2$ pierwiastków (ustalony x oraz $x_0, x_1, x_2, ..., x_n$) i jest klasy $C^{n+1}[a, b]$.
Z twierdzenia Rolle'a wynika, że razie $Y'(t)$ ma przynajmniej $n+1$ pierwiastków w (a, b), Y'' ma przynajmniej n pierwiastków w (a, b) itd., a $Y^{(n+1)}$ ma co najmniej jeden pierwiastek w tym przedziale. Oznaczmy go przez ξ:

$$Y^{(n+1)}(\xi) = 0$$

Dodatkowo:

$$Y^{(n+1)}(t) = R^{(n+1)}(t) - \lambda * w^{(n+1)}(t) = f^{(n+1)}(t) - p^{(n+1)}(t) - \lambda * (n + 1)!$$

Jako że $p(x)$ jest stopnia co najwyżej n, to $p^{(n+1)} = 0$ (można też od razu podstawić pierwiastek ξ):

$$Y^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \frac{R(x)}{w(x)} * (n+1)! = 0$$

W powyższego wynika ostatecznie wzór na błąd interpolacji metodą Lagrange'a:

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

2.3 Ilorazy różnicowe: podaj definicję i objaśnij ich związek z pochodnymi

Iloraz różnicowy to wielkość opisująca przyrost funkcji na danym przedziale $[a, b]$ postaci $\frac{f(b)-f(a)}{b-a}$. Jest to wartość średnia funkcji f na tym przedziale (z tw. o wartości średniej).
Przyjmując przedział długości h, otrzymuje się $\frac{f(x+h)-f(x)}{h}$, co przy $h \to 0$ daje definicję pochodnej:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Można także zdefiniować ilorazy różnicowe wyższych rzędów, przydatne w analizie numerycznej np. do obliczenia wielomianu interpolacyjnego Newtona metodą tablicy ilorazów różnicowych (algorytm Neville'a). Korzysta się wtedy z definicji rekurencyjnej dla danej tablicy zawierającej x_i i y_i:

$$\begin{cases} f[x_i] = y_i \\ f[x_i, x_{i+1}, x_{i+2}, ..., x_{i+j}] = \frac{f[x_{i+1}, x_{i+2}, ..., x_{i+j}] - f[x_i, x_{i+1}, x_{i+2}, ..., x_{i+j-1}]}{x_{i+j} - x_i} \end{cases}$$

Korzystając ponownie z tw. o wartości średniej można pokazać związek między powyższą definicją ilorazów różnicowych a pochodną funkcji:

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

2.4 Uzasadnij użyteczność użycia ilorazów różnicowych w interpolacji

- można łatwo skonstruować tablicę do obliczania ilorazów
- dla dobranych węzłów istnieje dokładnie jeden wielomian interpolacyjny
- łatwość dodania kolejnego węzła poprzez rozszerzenie tablicy

2.5 Przeprowadź porównanie: interpolacja Lagrange'a, a interpolacja Newtona

- Do obliczania wielomianu Lagrange'a wykorzystuje się bezpośrednio wzór z jego definicji czyli:

  $$P_n(x) = \sum_{k=0}^{n} f(x_k) \prod_{i=0, i \neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

  Natomiast w przypadku interpolacji Newtona obliczanie wielomianu przebiega z wykorzystaniem tablicy ilorazów różnicowych. Na podstawie przekątnej tej tablicy budujemy wielomian w postaci Newtona jako:

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

- Do wyliczenia wartości wielomianu w interpolacji Lagrange'a możemy wyznaczyć i zapisać sobie współczynniki:

  $$a_k = f(x_k) \prod_{i=0, i \neq k}^{n} \frac{1}{(x_k - x_i)}, \quad k = 0, 1, ..., n$$

i następnie korzystać ze wzoru:

$$P_n(x) = \sum_{k=0}^{n} a_k \prod_{i=0, i\neq k}^{n} (x - x_i)$$

W przypadku interpolacji Newtona jest prościej, bo wartości na przekątnej tablicy ilorazów różnicowych wyznaczają nam współczynniki postaci Newtona i aby obliczyć na ich podstawie wartość w danym punkcie możemy skorzystać z algorytmu Hornera dla postaci Newtona czyli:

$$W_n = b_n$$

$$W_i = W_{i+1}(x - x_i) + b_i$$

$$W(x) = W_0$$

gdzie:

$$b_n = f[x_0, x_1, ..., x_n]$$

- Jeśli chcemy dodać nowy węzeł to w przypadku interpolacji Lagrange'a zmieni nam się wzór i musimy go policzyć od nowa. Dla interpolacji Newtona wystarczy dodać nowy wiersz w tablicy ilorazów różnicowych i wziąć wartość z przekątnej jako kolejny współczynnik wielomianu w postaci Newtona.

2.6 Przeprowadź porównanie: interpolacja Newtona a interpolacja Hermite'a

- Interpolacja Lagrange'a korzysta tylko z wartości funkcji $y_i = f(x_i)$ dla danej tablicy $n+1$ węzłów, a interpolacja Hermite'a zakłada dodatkowo wykorzystanie danych wartości pochodnych w punktach (przy czym jeśli dane jest $f^{(j)}(x_i)$, to muszą być też dane $f'(x_i), ..., f^{(j-1)}(x_i)$). W interpolacji Hermite'a mamy dane $k+1$ węzłów $x_0, x_1, ..., x_k$. To ile pochodnych jest danych dla węzła x_i określa tak zwana krotność węzła m_i i zachodzi:

$$\sum_{i=0}^{k} m_i = n + 1$$

- Gdy wszystkie $m_i = 1$, to interpolacja Hermite'a działa tak jak interpolacja Newtona.
- Tworzenie tablicy ilorazów różnicowych w metodzie Hermite'a przebiega tak samo jak w metodzie Newtona, z tą różnicą, że tam gdzie nie można utworzyć ilorazu, to wykorzystujemy informację o pochodnej $f^{(j)}(x_i)$.

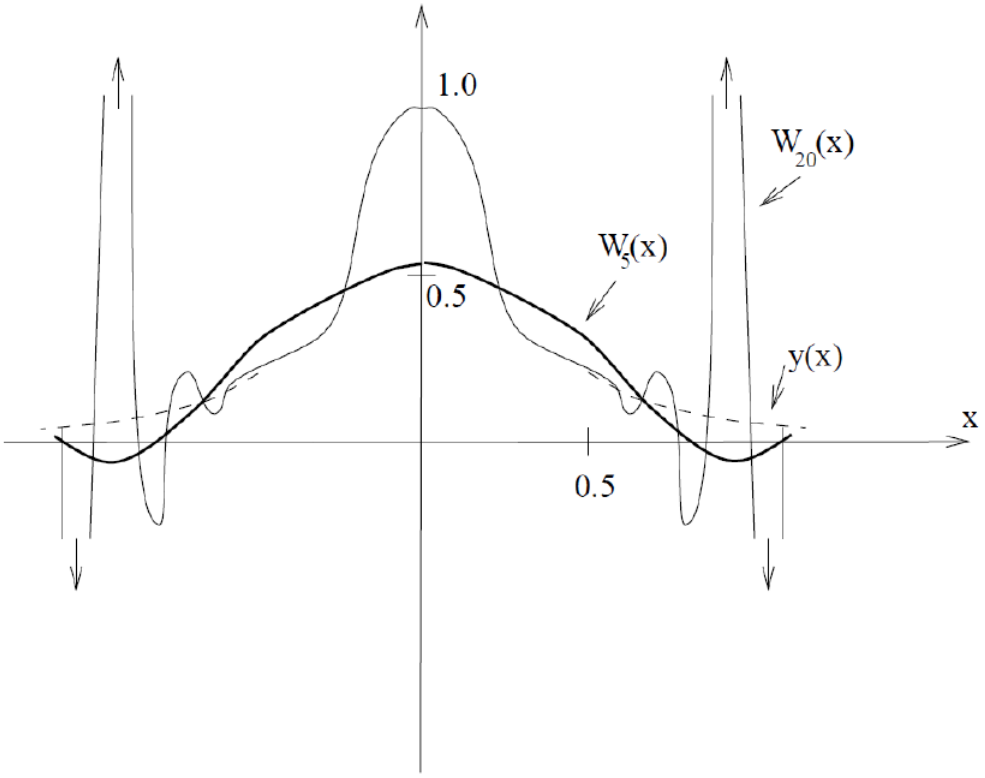
2.7 Objaśnij efekt Rungego: jak się objawia, co jest jego przyczyną, jak można zapobiegać

Efekt Rungego polega na obniżanie bniżeniu jakości interpolacji pomimo zwiększania liczby węzłów, tzn. początkowo wraz ze wzrostem liczby węzłów jakość interpolacji rośnie (błąd maleje), ale potem pogarsza się (szczególnie na końcach przedziału i dla wielomianów interpolujących wysokich stopni).
Przyczyny:

- interpolacja wielomianowa z równoczesnym nałożeniem warunku równoodległości węzłów,
- szybki wzrost wartości wyższych pochodnych.

Zapobieganie:

- wykorzystywać interpolację funkcjami sklejanymi (spline'ami),
- wybierać węzły gęściej przy granicach przedziału zamiast równoodlegle, np. poprzez wykorzystanie węzłów Czebyszewa (zera wielomianów Czebyszewa),
- zaczynać od interpolacji liniowej, a potem zwiększać liczbę punktów i stopień wielomianu dla ustabilizowania kluczowych miejsc.

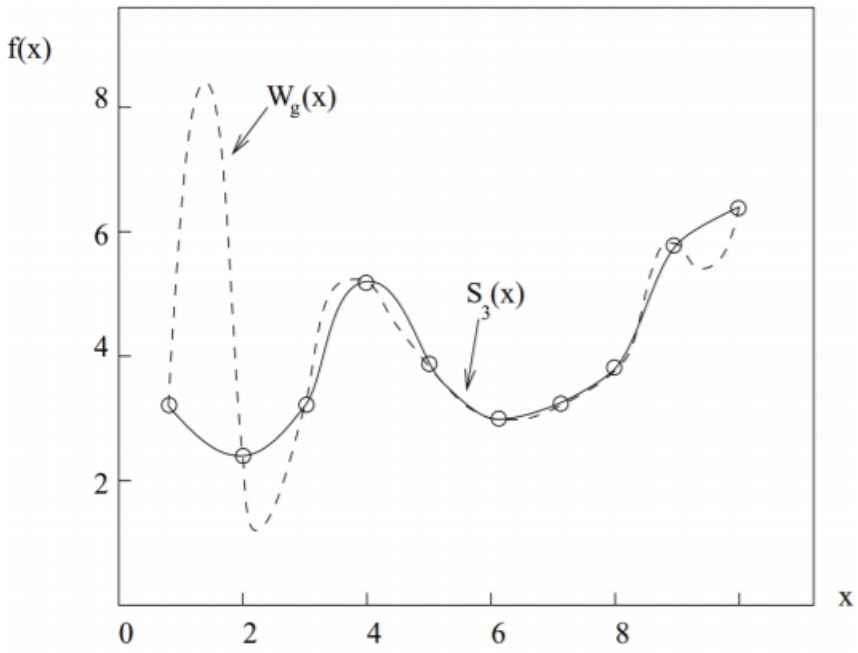


3 Spline

3.1 Podaj definicję funkcji sklejanej, objaśnij ją na odpowiednim rysunku, omów przydatność tych funkcji

Funkcja sklejana - (spline function, splajn) funkcja określona na przedziale $[a, b]$ (podzielonym na $n-1$ podprzedziałów węzłami: $a = x_1 < x_2 < ... < x_i < ... < x_n = b$) tak, że dla funkcji sklejanej stopnia m ($m \geq 1$) zachodzi:

- dla każdego przedziału $[x_i, x_{i+1}]$ funkcja jest wielomianem stopnia co najwyżej m
- pochodne funkcji do $m-1$ włącznie są ciągłe na przedziale $[a, b]$ (czyli funkcja jest $C^{m-1}([a, b])$)



Rysunek 1: W - wielomian interpolujący f(x), S - splajn interpolujący f(x)

Objaśnienie splajnów na podstawie rysunku:

- splajny mają różne rodzaje w zależności od tego, w jaki sposób konstruuje się wielomiany, np. splajn 0 i 1 stopnia, splajn kwadratowy (2 stopnia), splajn sześcienny (3 stopnia, cubic), naturalny splajn sześcienny
- stopień splajnu jest niezależny od liczby węzłów interpolacji
- dla każdego przedziału definiuje się inny wielomian według wzoru splajnu, ale wszystkie zachowują jego warunki
- zwykle używa się splajnów sześciennych (wymagają obliczenia współczynników dla podprzedziałów przez układ równań), w szczególności naturalnego splajnu sześciennego (natural cubic spline), gdyż jest on najgładszą funkcją interpolującą $\left(\int_a^b [S''(x)]^2 dx \leq \int_a^b [f''(x)]^2 dx\right)$
- można zastosować różne warunki brzegowe, żeby kontrolować zachowanie i kształt funkcji przy brzegach, np. $S''(x_1) = S''(x_n) = 0$ daje natural cubic spline, a $S'(x_1) = y'_1, S'(x_n) = y'_n$ daje complete spline.

Przydatność splajnów:

- nie występuje efekt Rungego jak przy interpolacji wielomianowej, splajny są bezpieczne dla węzłów równoodległych
- łatwo konstruuje się splajny sześcienne - wystarczy rozwiązać układ równań, który można sprowadzić do symetrycznej, trójprzekątniowej macierzy
- w praktyce splajny sześcienne są wystarczające do interpolacji
- dobre do obliczania pochodnych i całek (bo mają proste wzory)
- można z ich pomocą wygładzać funkcje i powierzchnie

- B-splajny (basic splines) wykorzystuje się w grafice komputerowej

3.2 Porównaj interpolację wielomianami i funkcjami sklejanymi

- W interpolacji wielomianowej cały przedział przybliżamy jedną funkcją, a korzystając ze splajnów dla każdego przedziału definiujemy inny wielomian zgodnie z zasadmi dla funkcji sklejanych
- Wielomian interpolacyjny jest jednoznaczny (jest tylko jeden wielomian o zadanych węzłach i wartościach), a splajnów jest wiele dzięki warunkom brzegowym (stopnie swobody), dzięki czemu poprzez ich dobór ma się wpływ na zachowanie funkcji przy końcach przedziału
- Dla wielomianów zwiększanie dokładności polega na zwiększaniu jego stopnia (ale może wystąpić efekt Rungego), a dla splajnów wystarczają zwykle splajny sześcienne
- W przypadku interpolacji wielomianowej dla węzłów równoodległych występuje efekt Rungego (zmniejszenie precyzji interpolacji wraz ze wzrostem liczby węzłów), natomiast przy interpolacji splajnami nie występuje (sposób rozmieszczenia węzłów nie ma znaczenia)
- Interpolacja wielomianowa wymaga obliczenia współrzędnych wielomianu np. za pomocą tablicy ilorazów różnicowych lub wzoru, może też wymagać znajomości wartości pochodnych w węzłach interpolacji (interpolacja Hermite'a), natomiast interpolacja splajnami wymaga obliczenia współrzędnych wielomianów i np. dla naturalnych splajnów sześciennych wymaga rozwiązywania układów równań, zwykle z symetrycznymi macierzami trójdiagonalnymi

3.3 Podaj i omów warunki brzegowe stosowane przy wyznaczaniu sześciennych funkcji sklejanych

Warunki brzegowe dla splajnów sześciennych:

- wykorzystując funkcje C_1 i C_n, przy czym stałe C_1''' i C_n''' mogą być wyznaczone bez znajomości $C_1(x)$ i $C_n(x)$:

$$C_1(x) \text{ - f. sześcienna przez pierwsze 4 punkty}$$
$$C_n(x) \text{ - f. sześcienna przez ostatnie 4 punkty}$$
$$S'''(x_1) = C_1''' \qquad S'''(x_n) = C_n'''$$

- natural cubic spline (free boundary):

$$S''(x_1) = S''(x_n) = 0$$

  - na końcach przedziału druga pochodna zeruje się (punkty przegięcia)
  - funkcja na końcu zamienia się w linię prostą
  - daje najgładszą funkcję interpolującą, tzn. dla $f \in C^2[a,b]$ i $a = x_1 < x_2 < ... < x_n = b$:

$$\int_a^b \left[S''(x)\right]^2 dx \leq \int_a^b \left[f''(x)\right]^2 dx$$

- complete spline (clamped boundary):

$$S'(x_1) = f'(x_1) \qquad S'(x_n) = f'(x_n)$$

  - trzeba znać wartość pierwszej pochodnej na końcach przedziału
  - pozwala dokładniej oddać kształt funkcji na końcach

- $S''(x_1) = y_1'' \qquad S''(x_n) = y_n''$

- not-a-knot condition - $S'''(x)$ ciągła w x_2 i x_{n-1}, wymusza ciągłość trzeciej pochodnej na końcach przedziału
- periodic cubic spline:

$$S(x_1) = S(x_n) \qquad S'(x_1) = S'(x_n) \qquad S''(x_1) = S''(x_n)$$

3.4 Wyprowadź wzór na kubiczne funkcje sklejane (3-stopnia)

Szukamy takiej funkcji S, która w danych węzłach t_i ma dane wartości y_i oraz w każdym przedziale $[t_i, t_{i+1})$ jest wielomianem stopnia co najwyżej 3 ($1 \leq i \leq n-1$).
Warunek $S_{i-1}(t_i) = y_i = S_i(t_i)$, czyli wartości w węzłach sąsiednich wielomianów zapewnia ciągłość funkcji S i daje nam $2n$ warunków (po 2 na każdy wielomian za jego brzegi).
Warunek ciągłości pierwszej pochodnej S', czyli $S'_{i-1}(t_i) = S'_i(t_i)$ daje $n-1$ warunków, tyle samo daje warunek ciągłości drugiej pochodnej.
Wielomiany mają razem $4n$ współczynników, powyższe warunki dają $4n-2$ warunków. Brakujące 2 to stopnie swobody, które można wyzyskać na różne sposoby.
Wprowadźmy oznaczenia i skorzystajmy z liniowości funkcji S''_i

$$z_i = S''_i(t_i) \qquad z_{i+1} = S''_i(t_{i+1}) \qquad h_i = t_{i+1} - t_i$$

$$S''_i(x) = \frac{z_i}{h_i}(t_{i+1} - x) + \frac{z_{i+1}}{h_i}(x - t_i)$$

Po scałkowaniu dwukrotnie drugiej pochodnej otrzymujemy wzór na $S_i(x)$, ale z nieznanymi współczynnikami:

$$S_i(x) = \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \frac{z_{i+1}}{6h_i}(x - t_i)^3 + C_i(x - t_i) + D_i(t_{i+1} - x)$$

Stałe C_i i D_i otrzymuje się, wykorzystując warunki interpolacyjne $S_i(t_i) = y_i$, $S_i(t_{i+1}) = y_{i+1}$. Wynika z nich, że:

$$S_i(x) = \frac{z_i}{6h_i}(t_{i+1} - x)^3 + \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \left(\frac{y_{i+1}}{h_i} - \frac{z_{i+1}h_i}{6}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{z_i h_i}{6}\right)(t_{i+1} - x)$$

Należy jeszcze wyznaczyć $z_i = S''_i(t_i)$, co można zrobić korzystając z warunków $S'_{i-1}(t_i) = S'_i(t_i)$. Różniczkujemy najpierw powyższy wielomian, a potem podstawiamy $x = t_i$, otrzymując prawą stronę tej równości:

$$S'_i(t_i) = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i - \frac{y_i}{h_i} + \frac{y_{i+1}}{h_i}$$

Aby otrzymać lewą stronę równości, wystarczy w obliczonej pochodnej podstawić zamienić wskaźnik i na $i-1$ i podstawić $x = t_i$:

$$S'_{i-1}(t_i) = \frac{h_{i-1}}{3}z_i + \frac{h_{i-1}}{6}z_{i-1} - \frac{y_{i-1}}{h_{i-1}} + \frac{y_i}{h_{i-1}}$$

Przyrównując do siebie powyższe wyrażenia otrzymujemy:

$$h_{i-1}z_{i-1} + 2(h_{i-1} + h_i)z_i + h_i z_{i+1} = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1})$$

Jest to układ $n-1$ równań z $n+1$ niewiadomymi $z_0, z_1, z_2, ..., z_n$. Wybór konkretnych z_0 i z_n daje rodzaj funkcji sklejanej. Po przyjęciu tych warunków brzegowych (np. dla naturalnej funkcji sześciennej $x_0 = x_n = 0$) trzeba rozwiązać układ równań.

3.5 Podaj definicje B-splines. Omów ich przydatność

$B_{j,k}(x)$ = B-spline rzędu k:

- $D_{j,0} = \begin{cases} 1 & , x_j \leq x \leq x_{j+1} \\ 0 & , \text{poza przedziałem} \end{cases}$
- wyższe rzędy (k $>$ 0) - rekurencyjnie:
  $B_{j,k}(x) = \frac{x-x_j}{x_{j+k}-x_j} B_{j,k-1}(x) + \frac{x_{j+k+1}-x}{x_{j+k+1}-x_{j+1}} B_{j+1,k-1}(x)$
- np. $B_{j,1} = \Psi_j$ (patrz interpolacja liniową funkcją sklejaną)

Reprezentacja funkcji sklejanej stopnia k:

$$S(x) = \sum_j p_j B_{j,k}(x)$$

gdzie p_j - współczynniki (w grafice komputerowej są to tzw. zadane punkty kontrolne)

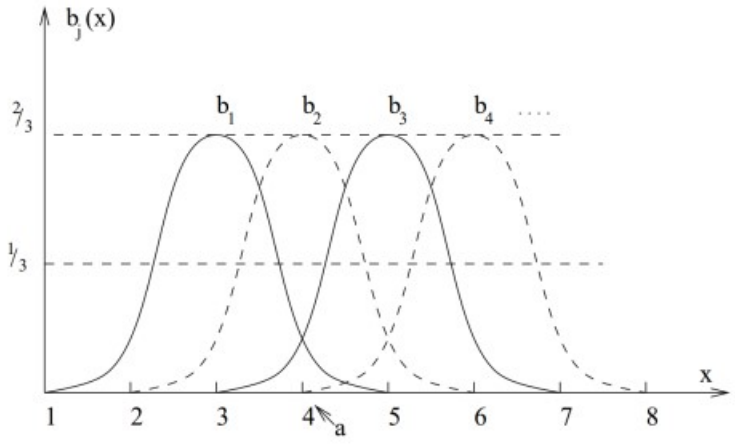
Funkcje B-sklejane:

- stosowane w grafice komputerowej do modelowania figur o skomplikowanych kształtach
- bazują na fakcie, że funkcje sklejane można wyrazić za pomocą kombinacji liniowej funkcji bazowych
- takie funkcje bazowe nazywamy funkcjami B-sklejanymi (B-splines)
- dla danego zestawu węzłów interpolacji - funkcje bazowe łatwo wyliczyć rekurencyjnie
- algorytmy o dobrych własnościach numerycznych

Własności:

- $B_{j,k}(x) > 0, x \in [x_j, x_{j+k+1}]$
- $B_{j,k}(x) = 0, x \notin [x_j, x_{j+k+1}]$
- w przedziale $[x_j, x_{j+1}]$ istotne jest tylko k + 1 funkcji:
  $B_{j-k,k}(x)...B_{j,k}(x) \neq 0$
- normalizacja: $\sum_j B_{j,k}(x) = \sum_{j=l-k}^{l} B_{j,k}(x) = 1$
  dla $x_l \leq x \leq x_{l+1}$

Funkcje B-sklejane stopnia 3



$$\underbrace{B_{j,3}(x)}_{b_j(x)} = \begin{cases} \frac{1}{6}z^3 & , z = \frac{x-x_j}{d} \quad x \in [x_j, x_{j+1}] \\ \frac{1}{6}[1+3(1+z(1-z))z] & , z = \frac{x-x_{j+1}}{d} \quad x \in [x_{j+1}, x_{j+2}] \\ \frac{1}{6}[1+3(1+z(1-z))(1-z)] & , z = \frac{x-x_{j+2}}{d} \quad x \in [x_{j+2}, x_{j+3}] \\ \frac{1}{6}(1-z)^3 & , z = \frac{x-x_{j+3}}{d} \quad x \in [x_{j+3}, x_{j+4}] \\ 0 & , x \notin [x_j, x_{j+4}] \end{cases}$$

Rysunek 2: Jakiś przykład z wykładu, ale może się przydać XDD

4 Aproksymacja

4.1 Podaj definicje i porównaj aproksymację średniokwadratową i jednostajną, wyjaśnij kiedy każda z nich jest użyteczna

Aproksymacja średniokwadratowa - sposób przybliżania wartości funkcji, który minimalizuje błąd przybliżenia na przedziale $[a, b]$. Dane jest $n+1$ węzłów $(x_i, y_i = F(x_i))$, $i = 0, 1, ..., n$ oraz $m+1$ funkcji bazowych $\varphi_i(x)$, $i = 0, 1, ..., m$, a szuka się wielomianu uogólnionego postaci $\sum_{j=0}^{m} a_j \varphi_j(x)$, czyli szuka się współczynników a_j takich, że poniższa norma osiąga minimum (suma dla przypadku dyskretnego, całka dla ciągłego):

$$min \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \varphi_j(x_i) \right]^2 \qquad min \int_a^b w(x)[F(x) - f(x)]^2 dx$$

$w(x)$ to funkcja wagowa wskazująca na istotność błędu w poszczególnych węzłach. Nazwa aproksymacji bierze się z postaci powyższej normy.

Aproksymacja jednostajna - sposób przybliżania wartości funkcji, w którym dla danej funkcji aproksymowanej $F(x)$ na danym przedziale $[a, b]$ szuka się funkcji aproksymującej $f(x)$ tak, aby minimalizować

największy błąd między nimi (minimalizuje się normę Czebyszewa):

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

Z twierdzeń Weierstrassa, Borela i Czebyszewa wynika kolejno, że każdą funkcję ciągłą na przedziale $[a, b]$ można aproksymować jednostajnie wielomianami, istnieje wielomian będący najlepszym takim przybliżeniem i jest tylko jeden. Porównanie:

- metoda średniokwadratowa minimalizuje średni błąd (dla wszystkich punktów), a metoda jednostajna minimalizuje błąd w punkcie, gdzie jest on największy
- metoda średniokwadratowa jest łatwiejsza w implementacji (wystarczy rozwiązać układ równań normalnych), a jednostajna (choć trudniejsza obliczeniowo) jest zwykle dokładniejsza (dla tego samego stopnia wielomianu)
- metoda średniokwadratowa pozwala narzucić wagi poszczególnych węzłów i wybraną klasę funkcji
- metoda średniokwadratowa pozwala wygładzać funkcje i powierzchnie
- metoda jednostajna pozwala wyznaczyć górne ograniczenie na błąd (wyklucza duży odchył funkcji na małym odcinku)
- stosując aproksymację średniokwadratową i wielomiany ortogonalne można pozbyć się efektu Rungego i słabego uwarunkowania, a stosując aproksymację jednostajną nie ma oscylacji na końcach przedziału
- jeśli węzły mają skoki wartości, to można użyć aproksymacji średniokwadratowej dla uznania ich za zaburzenia danych i nie wzięcia ich pod uwagę, lub aproksymacji jednostajnej, jeśli skoki są istotne i trzeba je dokładniej przybliżać
- aproksymacja średniokwadratowa pozwala przybliżać rozwiązania niedookreślonych układów równań (takich, gdzie równań jest mniej niż niewiadomych)

4.2 Wyprowadź wzór na aproksymację średniokwadratową wielomianem uogólnionym

Dane:

- $n + 1$ punktów (x_i, y_i), gdzie $i = 0, 1, 2, ..., n$ oraz $y_i = f(x_i)$
- układ funkcji bazowych $\varphi_j(x)$, $j = 0, 1, 2, ..., m$

Szukamy wielomianu uogólnionego, czyli funkcji postaci:

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

Trzeba wyznaczyć współczynniki a_j. Określa się je tak, aby różnica między funkcją aproksymującą a aproksymowaną była jak najmniejsza na normie średniokwadratowej:

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

W powyższym wzorze $w(x)$ to funkcja wagowa ($w(x) \geq 0$), zwykle $w(x) = 1$ lub $w(x_i) \approx \frac{1}{[f(x_i)error]^2}$.
Aby wyznaczyć a_j, trzeba teraz zminimalizować:

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

Należy skorzystać z warunku $\frac{\partial H}{\partial a_k} = 0, \quad k = 0, 1, 2, ..., m$. Takich warunków jest $m + 1$, a zatem otrzymuje się układ $m + 1$ równań z $m + 1$ niewiadomymi a_k (układ normalny):

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

4.3 Wyprowadź wzór na aproksymację średniokwadratową jednomianami

Dane:

- $n + 1$ punktów (x_i, y_i), gdzie $i = 0, 1, 2, ..., n$ oraz $y_i = f(x_i)$
- układ funkcji bazowych $\varphi_j(x) = x^j$, $j = 0, 1, 2, ..., m$

Szukamy funkcji postaci:

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

Trzeba wyznaczyć współczynniki a_j. Określa się je tak, aby różnica między funkcją aproksymującą a aproksymowaną była jak najmniejsza w normie średniokwadratowej. Rozważamy zbiór dyskretny x_i, $i = 0, 1, 2, ..., n$, na którym określona jest $f(x_i)$ i szukamy:

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

Tworzymy tzw. układ normalny równań, z których otrzymamy współczynniki a_i:

$$\sum_{i=0}^{n} w(x_i) \left[f(x_i) - \sum_{j=0}^{m} a_j x_i^j \right] x_i^k = 0 \qquad k = 0, 1, 2, ..., m$$

$$\sum_{i=0}^{n} w(x_i) x_i^k \sum_{j=0}^{m} a_j x_i^j = \sum_{i=0}^{n} w(x_i) f(x_i) x_i^k \qquad k = 0, 1, 2, ..., m$$

$$\sum_{j=0}^{m} \underbrace{\left(\sum_{i=0}^{n} w(x_i) x_i^{j+k} \right)}_{G_{k,j}} a_j = \underbrace{\sum_{i=0}^{n} w(x_i) f(x_i) x_i^k}_{B_k}$$

Powyższy układ równań można zapisać w postaci macierzowej, jak poniżej. Wystarczy go rozwiązać i otrzymuje się współczynniki a_i.

$$\underbrace{\begin{bmatrix} \sum w_i & \sum w_i x_i & \sum w_i x_i^2 & \dots & \sum w_i x_i^m \\ \sum w_i x_i & \sum w_i x_i^2 & \sum w_i x_i^3 & \dots & \sum w_i x_i^{m+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum w_i x_i^m & \sum w_i x_i^{m+1} & \sum w_i x_i^{m+2} & \dots & \sum w_i x_i^{2m} \end{bmatrix}}_{G_{k,j}} * \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}}_{A_k} = \underbrace{\begin{bmatrix} \sum w_i f_i \\ \sum w_i f_i x_i \\ \vdots \\ \sum w_i f_i x_i^m \end{bmatrix}}_{B_k}$$

$$G * A = B$$

Powyższy układ równań ma jedno rozwiązanie, jeśli $x_0, x_1, x_2, ..., x_n$ są różne i $m \leq n$.

4.4 Wyprowadź wzór na aproksymację średniokwadratowa wielomianami ortogonalnymi

Dane:

- $n+1$ punktów (x_i, y_i), gdzie $i = 0, 1, 2, ..., n$ oraz $y_i = f(x_i)$
- układ funkcji bazowych $\varphi_j(x)$, $j = 0, 1, 2, ..., m$ będących wielomianami ortogonalnymi (np. wielomianami Czebyszewa, wielomianami Legendre'a - zwłaszcza dla $w(x_i) = 1$)

Szukamy funkcji postaci:

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

Trzeba wyznaczyć współczynniki a_j. Określa się je tak, aby różnica między funkcją aproksymującą a aproksymowaną była jak najmniejsza na normie średniokwadratowej. Rozważamy zbiór dyskretny x_i, $i = 0, 1, 2, ..., n$, na którym określona jest $f(x_i)$ i szukamy:

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

Tworzymy tzw. układ normalny równań, z których otrzymamy współczynniki a_i:

$$\sum_{i=0}^{n} w(x_i) \left[f(x_i) - \sum_{j=0}^{m} a_j \varphi_j(x_i) \right] \varphi_k(x_i) = 0 \qquad k = 0, 1, 2, ..., m$$

$$\sum_{i=0}^{n} w(x_i) f(x_i) \varphi_k(x_i) = \sum_{i=0}^{n} w(x_i) \varphi_k(x_i) \sum_{j=0}^{m} a_j \varphi_j(x_i)$$

$$\sum_{i=0}^{n} w(x_i) f(x_i) \varphi_k(x_i) = \sum_{j=0}^{m} a_j \sum_{i=0}^{n} w(x_i) \varphi_k(x_i) \varphi_j(x_i)$$

$$\sum_{i=0}^{n} w(x_i) f(x_i) \varphi_k(x_i) = a_k \sum_{i=0}^{n} w(x_i) \varphi_k^2(x_i)$$

Powyższy układ równań z własności wielomianów ortogonalnych ma macierz diagonalną (bo jest równy 0 dla $j \neq k$ i > 0 dla $j = k$).

Przedstaw podstawowe własności wielomianów Legendre'a oraz ich typowe zastosowania.

Własności:

- są określone wzorem:

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n \quad x \in [-1, 1] \quad n = 0, 1, 2, ...$$

- spełniają wzór rekurencyjny:

$$P_0 = 1$$

$$P_{n+1}(x) = \frac{2n+1}{(n+1)x} P_n(x) - \frac{n}{(n+1)x} P_{n-1}(x) \quad n = 1, 2, ...$$

- spełniają równania różniczkowe postaci:

$$\frac{d}{dx} \left[(1 - x^2) \frac{dP_n(x)}{dx} \right] + n(n+1) P_n(x) = 0$$

- dla dowolnego n $|P_n(x)| \leq 1$ (przy spełnieniu warunku $x \in [-1, 1]$)
- symetria: $P_n(-x) = (-1)^n * P_n(x)$, wynika też z tego, że jeśli x_k jest zerem wielomianu, to $-x_k$ też
- są ortogonalne i:
$$\int_{-1}^{1} P_n(x) * P_m(x)dx = \begin{cases} 0, & m \neq n \\ \frac{2}{2n+1}, & m = n \end{cases}$$
- $\int_{-1}^{1} P_n(x)dx = 0$ dla $n \geq 1$
- wszystkie n zer wielomianu to zera rzeczywiste, parami różne i leżące w przedziale $(-1, 1)$, przy czym przy podziale przedziału $[-1, 1]$ na $n + 1$ przedziałów każdy z nich będzie zawierał dokładnie jedno zero P_{n+1}
- jeśli $P_n(1) = 1$, to wielomian Legendre'a jest w postaci znormalizowanej

Zastosowania:

- zera wielomianów Legendre'a (dla postaci znormalizowanej, $P_n(1) = 1$) są wykorzystywane jako węzły kwadratur Gaussa na przedziale $[-1, 1]$
- są wykorzystywane w fizyce, m. in. przy problemach o symetrii sferycznej czy przy harmonikach sferycznych

4.5 Opisz podstawowe metody aproksymacji jednostajnej

Metoda szeregów potęgowych - polega na aproksymacji jednostajnej funkcji za pomocą obliczania sum częściowych szeregu Taylora:
$$W_n(x) = \sum_{k=0}^{n} \frac{F^{(k)}(x_0)}{k!}(x - x_0)^k$$
Błąd takiej aproksymacji jest równy reszcie Lagrange'a:
$$F(x) - W_n(x) = \frac{F^{(n+1)}(\eta)}{(n+1)!}(x - x_0)^{n+1}, \eta \in [a, b]$$

Aproksymacja Padé - aproksymacja za pomocą funkcji wymiernej $r(x)$ stopnia $N = n+m$ postaci:
$$r(x) = \frac{p(x)}{q(x)} = \frac{p_0 + p_1x + p_2x^2 + ... + p_nx^n}{q_0 + q_1x + q_2x^2 + ... + q_mx^m}$$
Funkcja $r(x)$ jest nieredukowalna (p i q są względnie pierwsze), a wykorzystuje się ją do minimalizacji $f(x) - r(x)$, czyli $f(x) - r(x) = \frac{f(x)*q(x)-p(x)}{q(x)}$. Technika aproksymacji sprowadza się do rozwiązania układu równań liniowych w celu znalezienia współczynników a_i:
$$f(x) - r(x) = \frac{\sum\limits_{i=0}^{\infty} a_ix^i * \sum\limits_{i=0}^{m} q_ix^i - \sum\limits_{i=0}^{n} p_ix^i}{q(x)}$$
W metodzie tej należy tak dobrać $p_0, p_1, ..., p_n$ oraz $q_0, q_1, ..., q_n$, aby $f^{(k)}(0) - r^{(k)}(0) = 0$ dla $k = 0, 1, ..., N$. Oznacza to, że licznik nie ma wyrazów stopnia $\leq N$. Jeśli ten warunek jest spełniony, to współczynniki $\sum\limits_{i=0}^{k} a_iq_{k-i} - p_k$ są równe 0, a uzyskanie funkcji aproksymującej sprowadza się do do rozwiązania układu równań liniowych:
$$\sum_{i=0}^{k} a_iq_{k-i} - p_k = 0 \qquad k = 0, 1, ..., N$$
Aproksymacja Padé często daje lepszą aproksymację niż skończone sumy przybliżające szereg Taylora i może też działać tam, gdzie szereg Taylora nie jest zbieżny.

Aproksymacja wielomianami Czebyszewa - wykorzystuje sumy częściowe postaci $f(x) \approx \sum_{j=0}^{n} c_j T_j(x)$, gdzie $T_i(x)$ to wielomiany Czebyszewa. Baza wielomianów $T_i(x)$ jest wygodna, bo jest równomierna w $[-1, 1]$ (a każdy przedział da się przeskalować do tego), maksima i minima są dość równomiernie rozłożone na tym przedziale i mają porównywalne wartości. Stałe c_j oblicza się z warunku ortogonalności dla przypadku ciągłego:

$$c_0 = \frac{1}{\pi} \int_{-1}^{1} \frac{F(x)T_i(x)}{\sqrt{1-x^2}} dx$$

$$c_i = \frac{2}{\pi} \int_{-1}^{1} \frac{F(x)T_i(x)}{\sqrt{1-x^2}} dx, \quad i = 1, 2, ..., n$$

Aproksymacja wielomianami Czebyszewa wykorzystuje też często algorytm Clenshaw'a, który pozwala łatwo i szybko obliczać te kombinacje liniowe.

4.6 Podaj definicje i porównaj aproksymację średniokwadratową i jednostajną, wyjaśnij kiedy każda z nich jest użyteczna.

Aproksymacja średniokwadratowa - sposób przybliżania wartości funkcji, który minimalizuje błąd przybliżenia na przedziale $[a, b]$. Dane jest $n+1$ węzłów $(x_i, y_i = F(x_i))$, $i = 0, 1, ..., n$ oraz $m+1$ funkcji bazowych $\varphi_i(x)$, $i = 0, 1, ..., m$, a szuka się wielomianu uogólnionego postaci $\sum_{j=0}^{m} a_j \varphi_j(x)$, czyli szuka się współczynników a_j takich, że poniższa norma osiąga minimum (suma dla przypadku dyskretnego, całka dla ciągłego):

$$min \sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j \varphi_j(x_i) \right]^2 \qquad min \int_a^b w(x)[F(x) - f(x)]^2 dx$$

$w(x)$ to funkcja wagowa wskazująca na istotność błędu w poszczególnych węzłach. Nazwa aproksymacji bierze się z postaci powyższej normy.

Aproksymacja jednostajna - sposób przybliżania wartości funkcji, w którym dla danej funkcji aproksymowanej $F(x)$ na danym przedziale $[a, b]$ szuka się funkcji aproksymującej $f(x)$ tak, aby minimalizować największy błąd między nimi (minimalizuje się normę Czebyszewa):

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

Z twierdzeń Weierstrassa, Borela i Czebyszewa wynika kolejno, że każdą funkcję ciągłą na przedziale $[a, b]$ można aproksymować jednostajnie wielomianami, istnieje wielomian będący najlepszym takim przybliżeniem i jest tylko jeden. Porównanie:

- metoda średniokwadratowa minimalizuje średni błąd (dla wszystkich punktów), a metoda jednostajna minimalizuje błąd w punkcie, gdzie jest on największy
- metoda średniokwadratowa jest łatwiejsza w implementacji (wystarczy rozwiązać układ równań normalnych), a jednostajna (choć trudniejsza obliczeniowo) jest zwykle dokładniejsza (dla tego samego stopnia wielomianu)
- metoda średniokwadratowa pozwala narzucić wagi poszczególnych węzłów i wybraną klasę funkcji
- metoda średniokwadratowa pozwala wygładzać funkcje i powierzchnie
- metoda jednostajna pozwala wyznaczyć górne ograniczenie na błąd (wyklucza duży odchył funkcji na małym odcinku)

- stosując aproksymację średniokwadratową i wielomiany ortogonalne można pozbyć się efektu Rungego i słabego uwarunkowania, a stosując aproksymację jednostajną nie ma oscylacji na końcach przedziału
- jeśli węzły mają skoki wartości, to można użyć aproksymacji średniokwadratowej dla uznania ich za zaburzenia danych i nie wzięcia ich pod uwagę, lub aproksymacji jednostajnej, jeśli skoki są istotne i trzeba je dokładniej przybliżać
- aproksymacja średniokwadratowa pozwala przybliżać rozwiązania niedookreślonych układów równań (takich, gdzie równań jest mniej niż niewiadomych)

4.7 Wyprowadź wzór na wyznaczanie aproksymacji Pade

Szukamy funkcji wymiernej $r(x)$ aproksymującej funkcję $f(x)$. gdzie funkcja $r(x)$ stopnia $N = n + m$ ma postać:

$$r(x) = \frac{p(x)}{q(x)} = \frac{p_0 + p_1 x + p_2 x^2 + ... + p_n x^n}{q_0 + q_1 x + q_2 x^2 + ... + q_m x^m}$$

W powyższym wzorze p i q są względnie pierwsze, więc funkcja jest nieredukowalna. Celem aproksymacji jest minimalizacja $||f(x) - r(x)||$, przy czym.

$$f(x) - r(x) = f(x) - \frac{p(x)}{q(x)} = \frac{f(x) * q(x) - p(x)}{q(x)}$$

W powyższym wzorze $f(x)$ można wyrazić za pomocą szeregu Maclaurina, a wielomiany $q(x)$ i $p(x)$ zapisać jako sumy:

$$f(x) = \sum_{i=0}^{\infty} a_i x^i \qquad q(x) = \sum_{i=0}^{m} q_i x^i \qquad p(x) = \sum_{i=0}^{n} p_i x^i$$

$$f(x) - r(x) = \frac{\sum\limits_{i=0}^{\infty} a_i x^i * \sum\limits_{i=0}^{m} q_i x^i - \sum\limits_{i=0}^{n} p_i x^i}{q(x)}$$

Należy dobrać $p_0, p_1, ..., p_n$ oraz $q_0, q_1, ..., q_m$ tak, aby $f^{(k)}(0) - r^{(k)}(0) = 0$ dla $k = 0, 1, ..., N$. Oznacza to, że licznik nie powinien mieć wyrazów stopnia $\leq N$:

$$(a_0 + a_1 x + a_2 x^2 + ...)(1 + q_1 x + q_2 x^2 + ... + q_m x^m) - (p_0 + p_1 x + p_2 x^2 + ... + p_n x^n)$$

$$p_{n+1} = p_{n+2} = ... = p_{n+m} = p_N = 0$$

$$q_{m+1} = q_{m+2} = ... = q_{m+n} = q_N = 0$$

Przy powyższym warunku współczynniki $\sum\limits_{i=0}^{k} a_i q_{k-i} - p_k$ są równe 0. Wynika z tego, że znalezienie $p(x)$ i $q(x)$ sprowadza się do rozwiązania układu równań liniowych:

$$\sum_{i=0}^{k} a_i q_{k-i} - p_k = 0 \qquad k = 0, 1, ..., N$$

4.8 Przedstaw podstawowe własności wielomianów Czebyszewa, objaśnij do czego mogą być uzyteczne

Własności:

- mogą być reprezentowane trygonometrycznie:

$$T_n(x) = cos[n * arccos(x)] \quad x \in [-1, 1] \quad n = 0, 1, 2, ...$$

- mogą być reprezentowane jako relacja rekurencyjna:

$$T_0(x) = 1 \quad T_1(x) = x \quad T_n(x) = 2nT_{n-1}(x) - T_{n-2}(x), \ n \geq 2$$

- mają czynnik wiodący w reprezentacji rekurencyjnej (czynnik przy najwyższej potędze x) równy 2^{n-1} dla $n \geq 1$
- symetria: $T_n(-x) = (-1)^n * T_n(x) \longrightarrow$ jeżeli x jest pierwiastkiem, to $-x$ też
- wielomiany stopnia parzystego są funkcjami parzystymi, a stopnia nieparzystego funkcjami nieparzystymi
- wszystkie ich ekstrema mają wartości -1 (minima) lub 1 (maksima)
- miejsca zerowe to węzły Czebyszewa, przy czym wielomian $T_n(x)$ ma w $[-1, 1]$ n miejsc zerowych postaci:

$$x_k = cos\left(\frac{2k+1}{n} * \frac{\pi}{2}\right) \quad k = 0, 1, 2, ..., n-1$$

- są ortogonalne zarówno dla przypadku ciągłego (całka), jak i dyskretnego (suma):

$$\int_{-1}^{1} \frac{T_i(x) * T_j(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0, & i \neq j \\ \frac{\pi}{2}, & i = j \neq 0 \\ \pi, & i = j = 0 \end{cases}$$

$$\sum_{k=0}^{m} T_i(x_k) * T_j(x_k) = \begin{cases} 0, & i \neq j \\ \frac{m+1}{2}, & i = j \neq 0 \\ m+1, & i = j = 0 \end{cases}$$

- mają własność minimaksu, tzn. ze wszystkich wielomianów stopnia n ($n \geq 1$) z czynnikiem wiodącym (czynnik przy zmiennej o najwyższej potędze) równym 1 to wielomian $\frac{1}{2^{n-1}} T_n(x)$ ma na $[-1, 1]$ najmniejszą normę maksymalną spośród wszystkich wielomianów stopnia n:

$$||W_n||_\infty = \max_{x \in [a,b]} |W_n|$$

Zastosowania:

- dobór węzłów do interpolacji - wybór zer wielomianów Czebyszewa (węzłów Czebyszewa) na węzły interpolacji zamiast węzłów równoodległych uodparnia na efekt Rungego, zwłaszcza, że zera zagęszczają się przy końcach przedziału
- tworzy się z nich wielomian interpolujący Czebyszewa (szereg Czebyszewa) postaci $W_n(x) = \sum_{j=0}^{n} c_j T_j(x)$, gdzie współczynniki c_j oblicza się z własności ortogonalności dla przypadku dyskretnego, wielomianem tym aproksymuje się jednostajnie funkcje
- najlepiej ze wszystkich wielomianów przybliża się nimi zero na danym przedziale (przy czym każdy przedział da się znormalizować do wymaganego $[-1, 1]$)

4.9 Udowodnij własność minimaksu wielomianów Czebyszewa

Teza: dla wielomianu $p(x)$ stopnia n o czynniku wiodącym 1:

1. $||p||_{[-1,1]} \geq \frac{1}{2^{n-1}}$
2. znak równości jest osiągnięty dla $p(x) = \frac{1}{2^{n-1}} T_n(x)$

Dowód nie wprost:
Załóżmy, że $|p(x)| \leq \frac{1}{2^{n-1}}$. Oznaczmy też przez $q(x)$ wielomian stopnia n o czynniku wiodącym 1 taki, że:

$$q(x) = \frac{1}{2^{n-1}} * T_n(x)$$

Jeśli weźmiemy $x_i = cos\left(i * \frac{\pi}{n}\right)$, to:

$$(-1)^i p(x_i) \leq |p(x_i)| < \frac{1}{2^{n-1}} = (-1)^i q(x_i)$$

$$(-1)^i p(x_i) < (-1)^i q(x_i)$$

$$(-1)^i [q(x_i) - p(x_i)] > 0$$

Z powyższego wynika, że na przedziale $[-1, 1]$ różnica $p - q$ zmienia znak n razy, więc musi mieć tam n zer. Różnica $p - q$ jest jednak stopnia mniejszego od n, bo zarówno $p(x)$, jak i $q(x)$ były stopnia n i miały współczynnik przy najwyższej potędze równy 1, więc mamy sprzeczność i teza musi być prawdziwa.

4.10 Omów zastosowanie wielomianów Czebyszewa do interpolacji

Zastosowania:

- dobór węzłów do interpolacji - wybór zer wielomianów Czebyszewa (węzłów Czebyszewa) na węzły interpolacji zamiast węzłów równoodległych uodparnia na efekt Rungego, zwłaszcza, że zera zagęszczają się przy końcach przedziału
- tworzy się z nich wielomian interpolujący Czebyszewa (szereg Czebyszewa) postaci

4.11 Omów zastosowania wielomianów Czebyszewa do aproksymacji

wykorzystuje sumy częściowe postaci $f(x) \approx \sum_{j=0}^{n} c_j T_j(x)$, gdzie $T_i(x)$ to wielomiany Czebyszewa. Baza wielomianów $T_i(x)$ jest wygodna, bo jest równomierna w $[-1, 1]$ (a każdy przedział da się przeskalować do tego), maksima i minima są dość równomiernie rozłożone na tym przedziale i mają porównywalne wartości. Stałe c_j oblicza się z warunku ortogonalności dla przypadku ciągłego:

$$c_0 = \frac{1}{\pi} \int_{-1}^{1} \frac{F(x)T_i(x)}{\sqrt{1-x^2}} dx$$

$$c_i = \frac{2}{\pi} \int_{-1}^{1} \frac{F(x)T_i(x)}{\sqrt{1-x^2}} dx, \quad i = 1, 2, ..., n$$

Aproksymacja wielomianami Czebyszewa wykorzystuje też często algorytm Clenshaw'a, który pozwala łatwo i szybko obliczać te kombinacje liniowe.

4.12 Przedstaw algorytm Clenshawa i wyjaśnij, kiedy warto go stosować

Elegancki i efektywny sposób sumowania wyrazów spełniajacych pewien wzór rekurencyjny:

$$f(x) = \sum_{k=0}^{N} c_k F_k$$

przy czym c_k znane oraz F(x) - spełnia wzór rekurencyjny

$$F_{n+1}(x) = \alpha(n, x) \cdot F_n(x) + \beta(n, x) \cdot F_{n-1}(x)$$

gdzie: $\alpha(n, x), \beta(n, x)$ to pewne funkcje

Przykładem takiej sumy jest suma stosowana przy wyznaczaniu współczynników dla interpolacji Czebyszewa.

Pomocnicze zmienne:

$$y_{N+2} = y_{N+1} = 0$$

Wzór na c_k:

$$y_k(x) = \alpha(k,x) \cdot y_{k+1} + \beta(k+1,x) \cdot y_{k+2} + c_k, \quad k = N, .., 1$$
$$c_k = y_k - \alpha(k,x) \cdot y_{k+1} - \beta(k+1,x) \cdot y_{k+2}$$

Podstawiając do wzoru wyżej i po przekształceniu (zerowanie środkowych składników, podstawienie c_0):

$$f(x) = \sum_{k=N}^{0} [y_k - \alpha(k,x) \cdot y_{k+1} - \beta(k+1,x) \cdot y_{k+2}] \cdot F_k(x) = \beta(1,x) \cdot y_2(x) \cdot F_0(x) + y_1(x) \cdot F_1(x) + c_0 \cdot F_0(x)$$

Algorytm warto stosować gdy F_k jest zbyt skomplikowane, ale α i β są proste

Przykład:

$$f(x) = \sum_{k=0}^{N} a_k x^k$$
$$F_0 = 1, \quad F_{n+1} = x^{n+1} = x \cdot F_n(x)$$

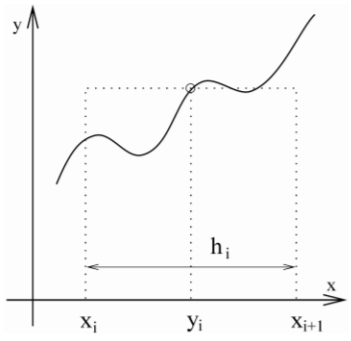
Porównując z wzorem rekurencyjnym na F_{n+1}, dostajemy:

$$\alpha(n,x) = x, \quad \beta(n,x) = 0$$
$$y_k(x) = \alpha(k,x) \cdot y_{k+1} + \beta(k+1,x) \cdot y_{k+2} + c_k = x \cdot y_{k+1} + c_k$$
$$f(x) = y_1 \cdot x + c_0 \cdot 1 = y_0$$

5 Kwadratury

5.1 Wyprowadź wzór na kwadratury elementarne trapezów i prostokątów (z błędami) korzystając ze wzoru Taylora

Wzór prostokątów:



Szukamy całki $\int_{x_i}^{x_{i+1}} f(x_i)dx \approx f(y_i)h_i$. Wykorzystując wzór Taylora:

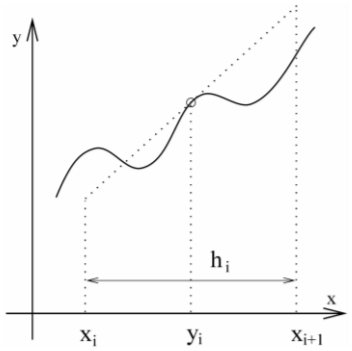
$$y_i = \frac{x_i + x_{i+1}}{2} \qquad h_i = x_{i+1} - x_i$$

$$f(x) = f(y_i) + \sum_{p=0}^{\infty} \frac{f^{(p)}(y_i)}{p!}(x - y_i)^p$$

$$\int_{x_i}^{x_{i+1}} f(x)dx = \underbrace{f(y_i)h_i}_{\text{kwadratura}} \underbrace{+\frac{1}{24}h_i^3 f''(y_i) + \frac{1}{1920}h_i^5 f^{(4)}(y_i)...}_{\text{błąd}}$$

Jak widać ze wzoru, kwadratura prostokątów ma błąd $O(h^3)$. Ma ona stopień dokładności 1, jest dokładna dla funkcji liniowej.

Wzór trapezów:



Szukamy całki $\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{1}{2}[f(x_i) + f(x_{i+1}] * h_i$

Wykorzystując dwukrotnie wzór Taylora:

$$f(x_i) = f(y_i) - \frac{1}{2}h_i f'(y_i) + ...$$

$$f(x_{i+1}) = f(y_i) + \frac{1}{2}h_i f'(y_i) + ...$$

$$\int_{x_i}^{x_{i+1}} f(x)dx = \underbrace{\frac{1}{2}[f(x_i) + f(x_{i+1})]h_i}_{\text{kwadratura}} \underbrace{-\frac{1}{12}h_i^3 f''(y_i) - \frac{1}{480}h_i^5 f^{(4)}(y_i) + ...}_{\text{błąd}}$$

Jak widać ze wzoru, kwadratura trapezów ma błąd $O(h^3)$. Ma ona stopień dokładności 1, jest dokładna dla funkcji liniowej.

5.2 Wyprowadź wzór na kwadraturę złożoną Simpsona (wraz ze wzorem na jej błąd)

Wzór Simpsona:

$$\int_a^b f(x)dx \approx \frac{1}{6}(b-a)\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Aby uzyskać wzór na metodę złożoną, przyjmujemy $x_i = a + i*h$, $i \in [0, 2h]$, $h = \frac{b-a}{2n}$ (dzieląc dany przedział $[a, b]$ na n podprzedziałów równej długości).

$$\int_a^b f(x)dx = \sum_{i=1}^{n} \int_{x_{2i-2}}^{x_{2i}} f(x)dx = \sum_{i=1}^{n} \frac{1}{6} * 2h * [f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] =$$

$$= \frac{h}{3} * \sum_{i=1}^{n} [f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] = \frac{h}{3} * [(f(x_0) + 4f(x_1) + f(x_2)) + (f(x_2) + 4f(x_3) + f(x_4)) + ...] =$$

$$= \frac{h}{3}\left[f(x_0) + 4\sum_{i=1}^{n} f(x_{2i-1}) + 2\sum_{i=1}^{n-1} f(x_{2i}) + f(x_{2n})\right]$$

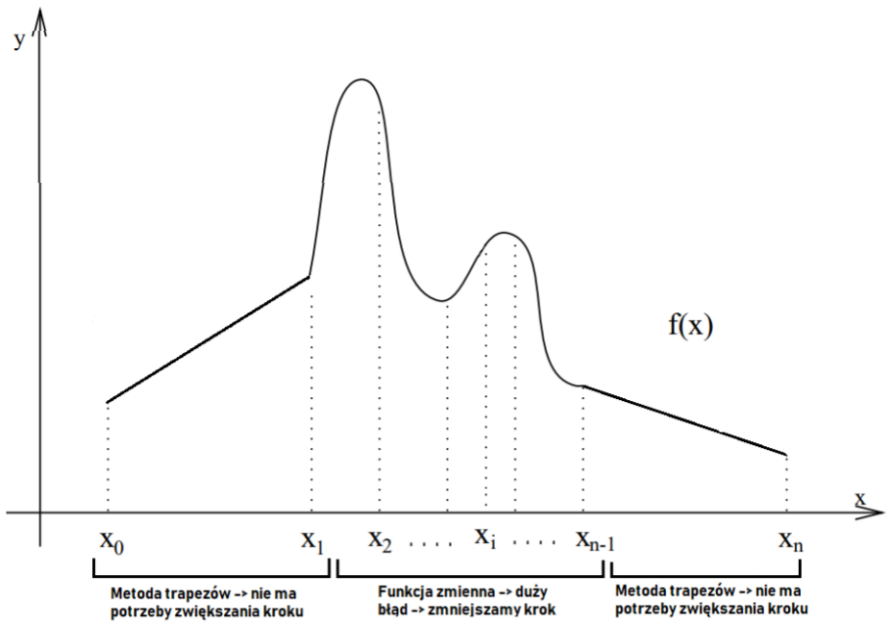
Powyższy wzór to już wzór na złożoną kwadraturę Simpsona. Błąd to suma błędów każdego z n podprzedziałów. Wychodząc od błędu podstawowej metody Simpsona (gdzie $\epsilon \in (a, b)$):

$$-\frac{1}{2880}\left(\frac{b-a}{n}\right)^5 f^{(4)}(\epsilon) * n = -\frac{1}{2880n^4}(b-a)^5 f^{(4)}(\epsilon)$$

5.3 Przedstaw i objaśnij algorytm całkowania adaptacyjnego (rozpocznij od dobrego rysunku)

Wzory złożone kwadratur posiadają węzły równoodległe, co ignoruje fakt, że funkcje mogą mieć w obszarze całkowania różny przedziałami przebieg, jak np. funkcje oscylacyjne.

Całkowanie adaptacyjne (adaptacyjny wzór Simpsona) dla $\int_a^b f(x)dx$ dzieli przedzial $[a, b]$ na fragmenty, dla których klasyczne metody całkowania dają już dostatecznie dobre wyniki. Dla dużych przedziałów o małej zmienności funkcji daje od razu wynik, miejsca o dużej zmienności dzieli na krótsze przedziały i wykonuje tam gęściej obliczenia. Jest ono równie precyzyjne co klasyczne całkowanie dla "dobrych"funkcji, ale dla bardziej problematycznych funkcji daje dużo precyzyjniejsze wyniki.



Rysunek 3: Działanie całkowania adaptacyjnego

Zakładając, że $f \in C^4([a,b])$, to można skorzystać ze wzoru Simpsona dla $h = \frac{b-a}{2}$ i zapisać:

$$\int_a^b f(x)dx = \underbrace{\frac{h}{3}[f(a) + 4f(a+h) + f(b)]}_{S(a,b)} - \frac{h^5}{90}f^{(4)}(\mu) \quad \mu \in (a,b)$$

Warunek, z którego korzysta całkowanie adaptacyjne dla obliczenia całki z precyzją ϵ to:

$$\left|S(a,b) - S\left(a, \frac{a+b}{2}\right) - S\left(\frac{a+b}{2}, b\right)\right| < 15\epsilon \quad (*)$$

Algorytm całkowania adaptacyjnego wygląda następująco:

1. Gdy spełniony jest warunek $(*)$, to $S\left(a, \frac{a+b}{2}\right) + S\left(\frac{a+b}{2}, b\right)$ przybliża $\int_a^b f(x)dx$ z dokładnością ϵ i można przerwać.
2. Gdy $(*)$ nie jest spełnione, to oceniamy błąd w $\left[a, \frac{a+b}{2}\right]$ i $\left[\frac{a+b}{2}, b\right]$, w każdym z nich $\epsilon' = \frac{\epsilon}{2}$
3. Połowimy podprzedziały i wyznaczamy S
4. Ten podprzedział (możliwe, że oba), na którym odpowiednik $(*)$ nie jest spełniony znów połowimy, nie zmieniając gotowych przedziałów.

5.4 Przedstaw podstawowe własności wielomianów Legendre'a oraz ich typowe zastosowania

Własności:

- są określone wzorem:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n \quad x \in [-1,1] \quad n = 0,1,2,...$$

- spełniają wzór rekurencyjny:

$$P_0 = 1$$

$$P_{n+1}(x) = \frac{2n+1}{(n+1)x}P_n(x) - \frac{n}{(n+1)x}P_{n-1}(x) \quad n = 1, 2, ...$$

- spełniają równania różniczkowe postaci:

$$\frac{d}{dx}\left[(1-x^2)\frac{dP_n(x)}{dx}\right] + n(n+1)P_n(x) = 0$$

- dla dowolnego n $|P_n(x)| \leq 1$ (przy spełnieniu warunku $x \in [-1, 1]$)
- symetria: $P_n(-x) = (-1)^n * P_n(x)$, wynika też z tego, że jeśli x_k jest zerem wielomianu, to $-x_k$ też
- są ortogonalne i:

$$\int_{-1}^{1} P_n(x) * P_m(x)dx = \begin{cases} 0, & m \neq n \\ \frac{2}{2n+1}, & m = n \end{cases}$$

- $\int_{-1}^{1} P_n(x)dx = 0$ dla $n \geq 1$
- wszystkie n zer wielomianu to zera rzeczywiste, parami różne i leżące w przedziale $(-1, 1)$, przy czym przy podziale przedziału $[-1, 1]$ na $n + 1$ przedziałów każdy z nich będzie zawierał dokładnie jedno zero P_{n+1}
- jeśli $P_n(1) = 1$, to wielomian Legendre'a jest w postaci znormalizowanej

Zastosowania:

- zera wielomianów Legendre'a (dla postaci znormalizowanej, $P_n(1) = 1$) są wykorzystywane jako węzły kwadratur Gaussa na przedziale $[-1, 1]$
- są wykorzystywane w fizyce, m. in. przy problemach o symetrii sferycznej czy przy harmonikach sferycznych

5.5 Porównaj kwadratury Newtona-Cotesa i Gaussa; wyjaśnij różnice między nimi

1. **Węzły:** Kwadratury Newtona-Cotesa wykorzystują węzły równoodległe, a kwadratury Gaussa węzły będące zerami wielomianów ortogonalnych wybranego typu, np. wielomianów Czebyszewa
2. **Wagi**: dla formuły $\int_a^b f(x)w(x) \approx \sum_{i=0}^{n} a_i f(x_i)$ kwadratury Newtona-Cotesa mają $w(x) = 1$, a a_i to liczby Cotesa. Dla kwadratur Gaussa $w(x)$ i a_i zależą od wybranego typu wielomianów ortogonalnych.
3. **Przedział całkowania**: dla kwadratur Newtona-Cotesa przedział całkowania nie wpływa na sposób obliczeń. Dla kwadratur Gaussa natomiast różne typy pozwalają całkować na różnych obszarach, np. dla kwadratury Gaussa-Legendre'a lub Gaussa-Czebyszewa przedział całkowania to $[-1, 1]$, a jeśli nasz przedział całkowania jest inny, to trzeba go przekształcić do $[-1, 1]$ (i potem odpowiednio w drugą stronę wynik).
4. **Typy otwarte i zamknięte:** kwadratury Newtona-Cotesa dzielą się na otwarte i zamknięte (odpowiednio niewykorzystujące i wykorzystujące punkty na brzegach przedziału całkowania), dla kwadratur Gaussa nie wprowadza się takiego podziału.
5. **Typy elementarne i złożone:** kwadratury Newtona-Cotesa dzieli się na elementarne (jeden wzór dla całego przedziału naraz, niska dokładność dla dużych przedziałów) i złożone (podział na podprzedziały i stosowanie tam wzorów elementarnych, lepsza dokładność), dla kwadratur Guassa nie wprowadza się takiego podziału.

6. **Dokładność:** dla kwadratur Newtona-Cotesa dokładność zależy od parzystości n (danej liczby węzłów) i tego, czy kwadratura jest otwarta, czy zamknięta. Kwadratura Gaussa dla n punktów ma stopień dokładności $2n-1$.

5.6 Omów zasadę tworzenia kwadratur Gaussa, podaj potrzebne twierdzenia

Kwadratura Gaussa opiera się na twierdzeniu:
Jeżeli $q(x)$ jest wielomianem ortogonalnym stopnia $n+1$ o zerach w $x_0, x_1, ..., x_n$ takim, że:

$$\int_a^b x^k q(x)dx = 0 \quad 0 \leq k \leq n$$

To wtedy wzór:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i) = \sum_{i=0}^{n} f(x_i) * \int_a^b l_i(x)dx \quad (*)$$

Dla x_i jako węzłów da dokładne wyniki dla wszystkich wielomianów stopnia do $2n+1$, a węzły te będą leżeć w przedziale (a, b).

Węzły kwadratury wyprowadza się w dowodzie części powyższego twierdzenia.
Weźmy f jako dowolny wielomian stopnia co najwyżej $2n+1$. Dzieląc przez $q(x)$ otrzymujemy iloraz i resztę stopnia co najwyżej n:

$$f(x) = p * q + r$$

Z założenia $\int_a^b q(x) * p(x)dx = 0$. Ponadto jako że x_i to pierwiastki $q(x)$, to:

$$f(x_i) = p(x_i) * q(x_i) + r(x_i) = 0 + r(x_i) = r(x_i)$$

Skoro $r(x)$ ma stopień co najwyżej n, to wzór $(*)$ pozwala obliczać $\int_a^b r(x)dx$ dokładnie. A zatem:

$$\int_a^b f(x)dx = \int_a^b p(x)q(x)dx + \int_a^b r(x)dx = \int_a^b r(x)dx = \sum_{i=0}^{n} A_i r(x_i) = \sum_{i=0}^{n} A_i f(x_i)$$

Z powyższego wynika, że węzłami kwadratury Gaussa są takie x_i, że są one zerami wykorzystanego wyżej wielomianu ortogonalnego $q(x)$, a ich wykorzystanie daje wzór dokładny aż dla wielomianów $2n+1$ stopnia.

5.7 Omów zasadę wyznaczania wag w kwadraturach Gaussa,

Mając dane punkty $x_1, x_2, ..., x_n$ można wyznaczyć wagi z układu równań:

$$\begin{bmatrix} \varphi_0(x_1) & \varphi_0(x_2) & \dots & \varphi_0(x_n) \\ \varphi_1(x_1) & \varphi_1(x_2) & \dots & \varphi_1(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{n-1}(x_1) & \varphi_{n-1}(x_2) & \dots & \varphi_{n-1}(x_n) \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \int_a^b w(x)\varphi_0(x)dx \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Zera w macierzy po prawej biorą się stąd, że pod całkami są tam funkcje $\varphi_1(x)$, $\varphi_2(x)$ itd., które są z własności funkcji $\varphi(x)$ ortogonalne względem stałej $\varphi_0(x)$, więc ich iloczyn się zeruje.
Wagi uzyskane z powyższego układu są dokładne dla wielomianów ortogonalnych stopnia co najwyżej $n-1$.

5.8 Podaj i scharakteryzuj poznane dotąd przykłady użyteczności wielomianów ortogonalnych w obliczeniach numerycznych

1. **Węzły interpolacji** - jeśli jako węzły interpolacji wybierze się miejsca zerowe ortogonalnych wielomianów Czebyszewa, to nie wystąpi efekt Rungego.
2. **Baza aproksymacji** - jako bazę wielomianów układu normalnego wybiera się wielomiany ortogonalne Czebyszewa lub Legendre'a.
3. **Całkowanie numeryczne** - do obliczenia optymalnego rozłożenia węzłów w kwadraturze Gaussa korzysta się z wielomianu ortogonalnego n-tego stopnia. Wykorzystuje się je także w kwadraturze Clenshawa-Curtisa.
4. **Metody iteracyjne niestacjonarne** - metody iteracyjne niestacjonarne rozwiązywania układów równań liniowych to takie, w których macierz iteracji ulega modyfikacji po każdym kroku. Metoda Czebyszewa jest oparta na idei wielomianów ortogonalnych, pozwala unikąć obliczania iloczynów skalarnych macierzy. Metody takie są szybciej zbieżne od metod stacjonarnych.

5.9 Opisz w jaki sposób można wykorzystać metodę divide and conquer (dziel i rządź) w algorytmach całkowania numerycznego

Zrównoleglenie obliczeń - algorytmy numeryczne można techniką *divide-and-conquer* łatwo zrównoleglić, gdyż ich istotą jest niezależne wykonywanie podproblemów, co pozwala wykorzystać możliwości współczesnych maszyn wielordzeniowych. Pozwala to łatwo znacznie przyspieszyć np. obliczanie FFT (równolegle parzyste i nieparzyste wyrazy).

5.10 Przedstaw przykłady wykorzystania twierdzeń z analizy matematycznej do tworzenia/analizy algorytmów numerycznych

1. **Wykorzystanie naturalnych splajnów sześciennych** - splajny to funkcje złożone kawałkami z wielomianów. Dzięki wykorzystaniu własności wielomianów (funkcje klasy C^∞ - najgładsze funkcje) i nałożeniu surowych warunków ciągłości (ciągłość wielomianów, 1 i 2 pochodnej dla splajnu 3 stopnia) powstają bardzo gładkie, dobrze interpolujące funkcje. Dodatkowo nakładając warunek $S''(x_1) = S''(x_n) = 0$ otrzymujemy *natural cubic spline* - zgodnie z twierdzeniem najgładszą funkcję interpolującą $\left(\int_a^b [S''(x)]^2 dx \leq \int_a^b [f''(x)]^2 dx\right)$.
2. **Aproksymacja jednostajna wielomianami Czebyszewa** - n-ty wielomian interpolujący Czebyszewa ma postać $W_n(x) = \sum_{j=0}^{n} c_j T_j(x)$. Współczynniki c_j oblicza się korzystając z ortogonalności wielomianów Czebyszewa. Same wielomiany mają wiele ciekawych własności: równomierne rozłożenie zer w przedziale $[-1, 1]$, $n + 2$ ekstremów na tym przedziale, a także oscylację wartości w przedziale $[-1, 1]$. Dzięki temu mają także właśność minimaksu: ze wszystkich wielomianów n-tego stopnia z czynnikiem wiodącym 1 to wielomian $\frac{1}{2^{n-1}} T_n(x)$ ($T_n(x)$ - wielomian Czebyszewa) ma najmniejszą normę maksimum na przedziale $[-1, 1]$. Te matematyczne własności sprawiają, że ta aproksymacja ma wszędzie na przedziale $[-1, 1]$ równomiernie rozłożony bardzo niewielki błąd.
3. **Rozwiązywanie równań nieliniowych metodą siecznych** - wykorzystując przybliżenie ze wzoru Taylora otrzymuje się styczną do funkcji, która wymaga jednak obliczenia pochodnej, co bywa bardzo trudne i kosztowne. Z definicji pochodnej można otrzymać jednak przybliżenie pochodnej różnicą skończoną, której obliczenie jest znacznie łatwiejsze. Zbieżność tej metody jest prawie tak szybka jak metody Newtona-Raphsona (1,6 vs 2), a przy tym unika się obliczania pochodnej, nie trzeba też sprawdzać, czy funkcja ma przeciwne znaki na końcach przedziału.
4. **Obliczanie wyznacznika macierzy korzystając z rozkładu LU** - wykorzystując częściowe poszukiwanie piwotu każdą kwadratową macierz nieosobliwą da się rozłożyć do iloczynu macierzy trójkątnych L i U, do tego *in situ*. Z twierdzenia Cauchy'ego $det(A) = det(L * U) = det(L) * det(U)$, a z własności

macierzy trójkątnej wyznacznik jest równy iloczynowi elementów na przekątnej - jako że L lub U ma same jedynki na przekątnej, to $det(A)$ jest równy $det(L)$ lub $det(U)$, którakolwiek macierz nie ma jedynek na przekątnej. Pozwala to na obliczenie wyznacznika macierzy A w czasie $O(\frac{2}{3}n^3)$.

6 Równania nieliniowe

6.1 Podaj i udowodnij twierdzenie o zbieżności procesu iteracyjnego

Twierdzenie o zbieżności procesu iteracyjnego $x_{i+1} = h(x_i)$ - jeżeli funkcja $x_{i+1} = h(x_i)$ ma pierwiastek α i na przedziale $I = [\alpha - a, \alpha + a]$ zachodzi $|h'(x)| \leq L < 1$ (gdzie L to stała), to $\forall x_o \in I$ zachodzi:

1. $x_i \in I$, $i = 1, 2, ...$
2. $\lim\limits_{i\to\infty} x_i = \alpha$
3. α to jedyny pierwiastek $x_{i+1} = h(x_i)$ na przedziale I

Dowód:

1. Załóżmy, że $x_{i-1} \in I$. Wtedy, korzystając z definicji funkcji $x_{i+1} = h(x_i)$:

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

   Korzystając z twierdzenia o wartości średniej istnieje taki punkt $\eta_{i-1} \in I$, że:

   $$\frac{h(x_{i-1}) - h(\alpha)}{x_{i-1} - \alpha} = h'(\eta_{i-1}) \longrightarrow (x_{i-1} - \alpha) * h'(\eta_{i-1})$$

   Z powyższych oraz warunku $|h'(x)| \leq L$ wynika, że:

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

   A w takim razie $x_i \in I$.

2. Powyższego dowodu własności 1. można użyć wielokrotnie:

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

   Z warunku $L < 1$ wynika, że $\lim\limits_{i\to\infty} L^i = 0$, a zatem $\lim\limits_{i\to\infty} |x_i - a| = 0$.

3. Załóżmy, że poza α istnieje w I jeszcze drugi pierwiastek β. W takim razie (korzystając z definicji $x_{i+1} = h(x_i)$ oraz twierdzenia o wartości średniej, analogicznie jak wyżej):

   $$\alpha - \beta = h(\alpha) - h(\beta) = (\alpha - \beta) * h'(\eta)$$

   $$|\alpha - \beta| = |\alpha - \beta| * |h'(\eta)|$$

   Jako że $|h'(\eta)| < 1$, to otrzymujemy sprzeczność ($|\alpha - \beta| < |\alpha - \beta|$), a zatem teza o istnieniu tylko pierwiastka α musi być prawdziwa.

6.2 Wyjaśnij pojęcie rzędu zbieżności procedury iteracyjnej

Krótko: jak szybko zbiega nasza metoda

Rząd zbieżności procedury iteracyjnej $x_i = \phi(x_{i-1})$

1. $\{x_i\}$ - zbieżność do α
2. "szybkość" zbieżności $|\frac{\epsilon_i}{\epsilon_{i-1}}| = ?$

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

rozwinięcie względem α

$$\phi(x_{i-1}) = \phi(\alpha) + \sum_{j=1}^{p-1} \frac{1}{j!} \underbrace{(x_{i-1} - \alpha)^j}_{\epsilon_{i-1}} \cdot \phi^{(j)}(\alpha) + \frac{1}{p!}(x_{i-1} - \alpha)^p \cdot \phi^{(p)}(\eta_{i-1})$$

$\eta_{i-1} \in (x_{i-1}, \alpha)$

$$\epsilon_i = \sum_{j=1}^{p-1} \frac{1}{j!} \epsilon_{i-1}^j \cdot \phi^{(j)}(\alpha) + \frac{\epsilon_{i-1}^p}{p!} \phi^{(p)}(\eta_{i-1})$$

Rysunek 4: Slajd 1

Rząd zbieżności procedury iteracyjnej $x_i = \phi(x_{i-1})$

Jeżeli: $\phi'(\alpha) = \phi''(\alpha) = \ldots = \phi^{(p-1)}(\alpha) = 0, \quad \phi^{(p)}(\alpha) \neq 0$

to: $\epsilon_i = \frac{\epsilon_{i-1}^p}{p!} \phi^{(p)}(\eta_{i-1})$

$\lim_{i \to \infty} |\frac{\epsilon_i}{\epsilon_{i-1}^p}| = \frac{1}{p!} |\phi^{(p)}(\alpha)|$

$|\frac{\epsilon_i}{\epsilon_{i-1}^p}|$ - *p-ty rząd zbieżności (p-th order of convergence)*

$\frac{1}{p!} |\phi^{(p)}(\alpha)|$ - stała asymptotyczna błędu (asymptotic error constant)

p = 1 - linear
2 - quadratic
3 - cubic

Rysunek 5: Slajd 2

Można konstruować metody iteracyjne $x_i = \phi(x_{i-1})$ dowolnego rzędu p - lecz: konieczne wyliczanie 0, 1, ..., $p-1$ pochodnych: np. Richmond's:

$$x_i = x_{i-1} - \frac{2 \cdot f(x_{i-1}) \cdot f'(x_{i-1})}{2 \cdot (f'(x_{i-1}))^2 - f(x_{i-1}) f''(x_{i-1})}$$

Rysunek 6: Slajd 3

6.3 Scharakteryzuj metody iteracyjne w obliczeniach numerycznych, podaj: ogólny algorytm, potrzebne twierdzenia, kiedy są przydatne

Metody iteracyjne - metody (algorytmy), których idea opiera się na ulepszaniu rozwiązania wraz z kolejnymi iteracjami (powtórzeniami procesu przybliżania).

Ogólny algorytm:

```
x = x0  // initial guess
max_iterations  // required number of iterations, usually more = greater precision
i = 0
while i < max_iterations:
  x = F(x)  // improve solution
  i += 1
```

Do wykorzystywania metod iteracyjnych do konkretnych problemów potrzebne są twierdzenia zapewniające warunki zbieżności. Metody można stosować tylko dla problemów, dla których te warunki są spełnione, czyli iteracje będą dawać coraz lepsze rozwiązania. Są to np. twierdzenie o zbieżności procesu iteracyjnego $Ax = b$ dla układów równań liniowych czy twierdzenie o zbieżności procesu iteracyjnego $x = h(x)$ dla równań nieliniowych.
Złożoność obliczeniowa zależy liniowo od liczby operacji (ich złożoność zależy od konkretnego problemu), dlatego można dobrać ilość operacji w zależności od dostępnego czasu i wymaganej precyzji.
Metody iteracyjne są szczególnie przydatne, kiedy możemy zadowolić się przybliżonym rozwiązaniem lub chcemy mieć kontrolę nad złożonością czasową problemu i związaną z tym precyzją rozwiązania. Ponadto metody te mogą mieć dodatkowe właściwości zależne od konkretnego problemu, np. w układach równań liniowych nie naruszają one struktur macierzy rzadkich.

6.4 Wyprowadź wzór na metodę Newtona-Raphsona i jej rząd zbieżności

Wzór:
Mając daną funkcję $f(x)$ i szukając pierwiastka α ($f(\alpha) = 0$) wykorzystujemy przybliżenia α postaci x_i. Oznaczając $\alpha = x_i + h$ oraz biorąc początek szeregu Taylora otrzymujemy wzór na styczną do funkcji:

$$f(\alpha) = 0 = f(x_i + h)$$

$$f(x_i + h) = f(x_i) + h * f'(x_i) + \underbrace{\dots}_{\text{pomijamy}}$$

$$h = -\frac{f(x_i)}{f'(x_i)}$$

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

Otrzymuje się tym samym wzór iteracyjny na $x = h(x)$ postaci $h(x) = x - \frac{f(x)}{f'(x)}$.

Rząd zbieżności:
Wykorzystując twierdzenie Taylora dla punktu x_i bliskiego α ($f''(x)$ jest ciągła) otrzymujemy przybliżenie funkcji $f(x)$ (gdzie $\xi_i \in (\alpha, x_i)$):

$$f(\alpha) = f(x_i) + f'(x_i) * (\alpha - x_i) + \frac{f''(\xi_i)}{2!}(\alpha - x_i)^2$$

Wykorzystując fakt, że α jest pierwiastkiem, otrzymujemy:

$$0 = f(\alpha) = f(x_i) + f'(x_i) * (\alpha - x_i) + \frac{f''(\xi_i)}{2}(\alpha - x_i)^2$$

Dzieląc powyższe równanie przez $f'(x_i)$ (łatwo sprawdzić, że jest różna od 0) i przekształcając otrzymujemy:

$$\frac{f(x_i)}{f'(x_i)} + (\alpha - x_i) = -\frac{f''(\xi_i)}{2 * f'(x_i)} * (\alpha - x_i)^2$$

Podstawiając do tego równania $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ otrzymujemy:

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

Powyżej otrzymaliśmy ϵ_{i+1} i ϵ_i, po przekształceniach:

$$|\epsilon_{i+1}| = \frac{|f''(\xi_i)|}{2|f'(x_i)|} * \epsilon_i^2$$

Z powyższego wynika, że rząd zbieżności metody Newtona-Raphona to 2 ($\epsilon_{i+1} \approx \epsilon_i^2$).

6.5 Przedstaw metodę Aitkena - do czego służy i kiedy się ją stosuje

Przyspieszanie zbieżności - metoda Aitkena

Stosowana aby polepszyć zbieżność dla metody o zbieżności liniowej

x_i, x_{i+1}, x_{i+2} - kolejne iteracje

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

$$x_{i+2} - \alpha = (x_{i+1} - \alpha) \cdot \phi'(\eta_{i+1})$$

$$x_{i+1} - \alpha = (x_i - \alpha) \cdot \underbrace{\phi'(\eta_i)}_{(*)}$$

Założenie: $(*)$ - nieznane, ale dla dwóch kolejnych iteracji blisko rozwiązania $\phi'(\eta_i) \approx \phi'(\eta_{i+1})$ ($\phi'(\eta_i)$ dąży do stałej asymptotycznej błędu)

Rysunek 7: Slajd 1

Służy do poprawienia punkt x^*_{i+2} (możemy zacząć dopiero od drugiego punktu, kosztem obliczeniowym ale nie trzeba znać pochodnej) (?)

6.6 Scharakteryzuj interpolacyjne metody znajdowania rozwiązań równań nieliniowych

Regula falsi - (łac. fałszywa linia prosta, fałszywa reguła) sposób znajdowania pierwiastka równania nieliniowego bazujący na fałszywym twierdzeniu, że na pewnym przedziale funkcja jest liniowa. Wykorzystuje on interpolowanie funkcji poprzez prostą, która zawsze przechodzi przez jeden punkt (*fixed-point*, wynika z warunku 3). Wymagania:

1. W przedziale $[a, b]$ znajduje się dokładnie 1 pojedynczy pierwiastek
2. $f(a) * f(b) < 0$
3. f' i f'' istnieją na tym przedziale i mają na nim stałe znaki

Algorytm ten przeprowadza cięciwy (na początku przez punkty $(a, f(a))$ i $(b, f(b))$), biorąc punkt przecięcia cięciwy z osią OX jako przybliżenie pierwiastka. Pierwsze przybliżenie pierwiastka ma zatem postać:

$$x_1 = \frac{a * f(b) - b * f(a)}{f(b) - f(a)}$$

$$\frac{x_{i+2}-\alpha}{x_{i+1}-\alpha} = \frac{x_{i+1}-\alpha}{x_i-\alpha}$$

$$\alpha = \frac{x_i \cdot x_{i+2} - x_{i+1}^2}{x_{i+2} - 2 \cdot x_{i+1} + x_i} = x_i - \frac{(x_{i+1}-x_i)^2}{x_{i+2} - 2 \cdot x_{i+1} + x_i}$$

Na tej podstawie mamy wzór na nowe, lepsze przybliżenie x^*_{i+2}

$$\boxed{x^*_{i+2} = x_i - \frac{(x_{i+1}-x_i)^2}{x_{i+2}-2\cdot x_{i+1}+x_i} \rightarrow (*)}$$

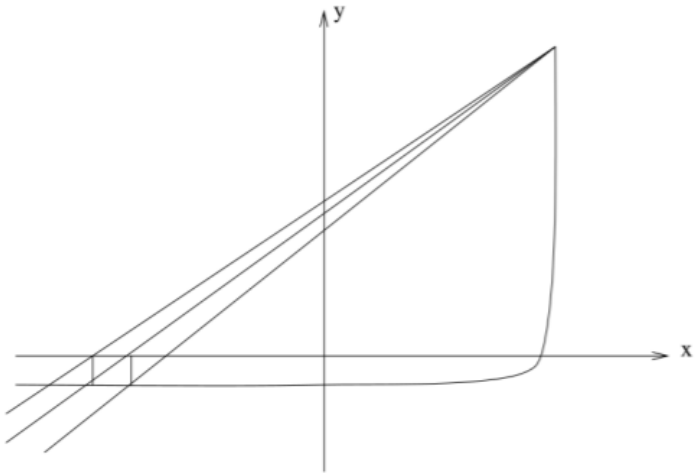
$(*)$ - zbieżność kwadratowa bez obliczania 2-gich pochodnych

Rysunek 8: Slajd 2

Następnie bierze się wyznaczony punkt (w pierwszym przybliżeniu $(x_1, f(x_1))$), jeden z punktów początkowych (*fixed-point*, ten z wartości o przeciwnym znaku, więc zawsze jest wybierany ten sam) i powtarza aż do uzyskania wymaganej dokładności. Wzór ma postać (gdzie jeden z punktów jest z warunku 3 taki sam i jest jednym z punktów początkowych, *fixed-point*):

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

Metoda ta ma zbieżność liniową. Łatwo znaleźć szczególnie trudne przypadki:



Rysunek 9: Dla powyższej funkcji proste bardzo długo będą zbiegać do pierwiastka.

Metoda Illinois - ulepszenie regula falsi. Jeżeli nowa wartość y, czyli $f(x_{i+1})$ ma ten sam znak co $f(x_i)$, to y mnoży się przez $\alpha = \frac{1}{2}$. Jeżeli ma przeciwny znak, to działa jak zwykła regula falsi. Zwiększa to zbieżność do nadliniowej (ok. 1,4).

Metoda Pegasus - ulepszenie regula falsi. Jeżeli nowa wartość y, czyli $f(x_{i+1})$ ma ten sam znak co $f(x_i)$, to y mnoży się przez:

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

Jeżeli ma przeciwny znak, to działa jak zwykła regula falsi. Zwiększa to zbieżność do nadliniowej, większej niż metody Illinois (ok. 1,7).

Metoda siecznych - modyfikacja metody Newtona-Raphsona, która dzięki przybliżeniu pochodnej poprzez różnicę skończoną pozwala uniknąć obliczania pochodnej. Nie wymaga także badania znaków $f_0 * f_1$. Wzór na kolejne przybliżenia pierwiastka przyjmuje postać:

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

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

Metoda ta ma rząd zbieżności ok. 1.6, więc co prawda jest gorsza od metody Newtona-Raphsona, lecz pozwala uniknąć liczenia pochodnej (które może być kosztowne), a także nie wymaga sprawdzania znaku funkcji w przeciwieństwie do regula falsi i ma lepszą zbieżność od niej (choć już nie od metody Pegasus).

6.7 Opisz sposoby wykorzystania metody divide and conquer w algorytmach numerycznych

1. **Redukcja złożoności obliczeniowej** - podejście *divide-and-conquer* pozwoliło w prosty sposób zmniejszyć złożoność np. obliczania transformaty Fouriera (algorytm Cooleya-Tukeya FFT liczący dwie o połowę mniejsze transformaty rekurencyjnie, $O(nlog_2(n))$ zamiast $O(n^2)$ algorytmu klasycznego) czy mnożenia macierzy (algorytm Strassena mnożenia macierzy dzielący macierze na podmacierze $\frac{n}{2} \times \frac{n}{2}$, złożoność $O(n^{2,8})$ zamiast $O(n^3)$ algorytmu klasycznego).
2. **Zwiększenie precyzji obliczeń** - poprzez podział wymaganych obliczeń na mniejsze i połączenie wyników zgodnie z techniką *divide-and-conquer* można zwiększyć precyzję poprzez lokalizację problematycznych fragmentów obliczeń i wykonywanie ich tam dokładniej, np. w algorytmie całkowania adaptacyjnego (można nawet narzucić precyzję całkowania ϵ) lub sumowaniu rekurencyjnym liczb (*pairwise summation*, domyślne sumowanie np. w języku do obliczeń numerycznych Julia).
3. **Zrównoleglenie obliczeń** - algorytmy numeryczne można techniką *divide-and-conquer* łatwo zrównoleglić, gdyż ich istotą jest niezależne wykonywanie podproblemów, co pozwala wykorzystać możliwości współczesnych maszyn wielordzeniowych. Pozwala to łatwo znacznie przyspieszyć np. obliczanie FFT (równolegle parzyste i nieparzyste wyrazy).
4. **Lepsze wykorzystanie pamięci** - podproblemy małych rozmiarów mogą być rozwiązywane w całości w cache'u, bez czasochłonnych operacji wejścia-wyjścia na dysku. Dzięki temu proste operacje, których w podejściu *divide-and-conquer* i problemach numerycznych jest najwięcej, są wykonywane bardzo szybko, np. mnożenie wektorów.

6.8 Objaśnij na czym polega deflacja i kiedy jest stosowana

Deflacja - obniżenie stopnia wielomianu po znalezieniu jego pierwiastka; bardzo użyteczna część algorytmu wyznaczania pierwiastków wielomianów. Do deflacji można użyć algorytmu Hornera.
$P(x)$ – wielomian,
r – znaleziony pierwiastek wielomianu P.
Realizujemy faktoryzację:

$$P(x) = (x-r)Q(x)$$

- Zmniejszenie złożoności obliczeniowej (Q – niższego stopnia niż P)
- Uniknięcie pomyłki – powrotu do pierwiastka już znalezionego.

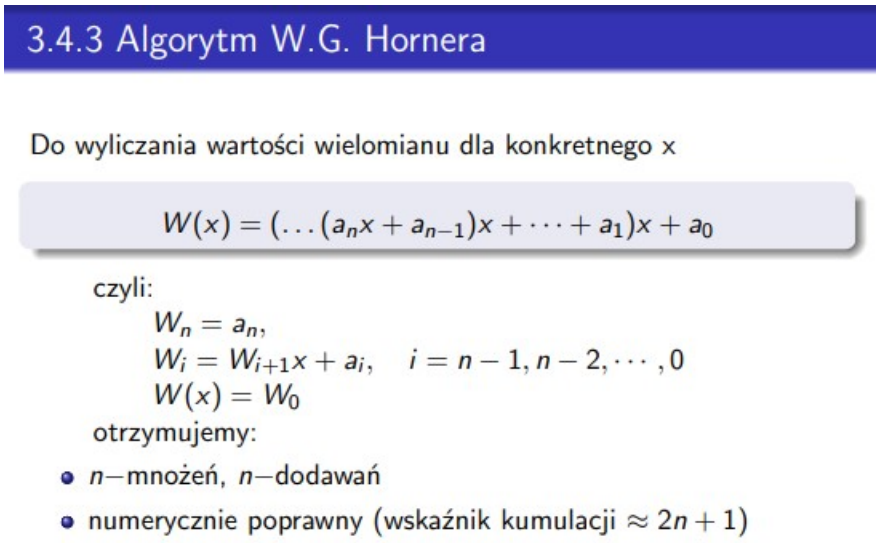
Stosowanie deflacji musi być ostrożne!
Powód: – pierwiastki są wyznaczane ze skończoną dokładnością (kilka deflacji pociąga kumulację błędu).

- Forward deflation – nowe współczynniki Q(x) obliczane od najwyższych potęg x . . . Stabilna, gdy zaczynamy od pierwiastków o najmniejszej wartości bezwzględnej,
- Backward deflation – współczynniki Q(x) wyznaczane poczynając od wyrazu wolnego. Stabilna, gdy zaczynamy od pierwiastków o największej wartości bezwzględnej.

Podejście minimalizujące błędy

- Kolejne pierwiastki uzyskane w procesie deflacji są próbnymi (tentative),
- Polishing (re-solving) – wygładzanie – z użyciem pełnego P(x) np. metodą Newtona-Raphsona,
- Niebezpieczeństwo – zlanie się dwóch pierwiastków w jeden

6.9 Omów zastosowanie metody Hornera dla wyznaczania pierwiastków wielomianów



Rysunek 10: Slajd 1

- jest numerycznie poprawny i ma dobry wskaźnik kumulacji
- ma mało mnożeń, mało dodawań
- można stosować algorytm Hornera do dzielenia wielomianu przez czynnik liniowy $(x - \lambda)$
- do obliczania wartości wielomianu w postaci Newtona najlepiej użyć wariantu schematu Hornera
- używany przy deflacji (???)
- używany przy wygładzaniu pierwiastków rzeczywistych metodą Newtona-Raphsona

6.10 Przedstaw metodę Bairstowa wyznaczania pierwiastków wielomianów

Metoda Bairstowa znajduje pierwiastki zespolone wielomianów o współczynnikach rzeczywistych. Wykorzystuje ona własność, że dla wielomianu $P(z) = \sum_{k=0}^{n} a_k z^k$ jeżeli w jest jego pierwiastkiem, to $\overline{w}$ też jest. Iloczyn $(z - w) * (z - \overline{w})$ to czynnik kwadratowy, gdzie $(z - w) * (z - \overline{w}) = z^2 - pz - q$.

W metodzie szuka się czynników kwadratowych zgodnie ze wzorem rekurencyjnym:

$$\sum_{k=0}^{n} a_k z^k = (\sum_{k=2}^{n} b_k z^{k-2}) * (z^2 - pz - q) + b_1(z-p) + b_0$$

Z powyższego wzoru wynika, że:

$$a_k = b_k - pb_{k+1} - qb_{k+2}$$

Otrzymujemy także układ równań:

$$b_0(p,q) = 0$$

$$b_1(p,q) = 0$$

Do rozwiązania powyższego układu używamy metody Newtona-Raphsona dla układów równań nieliniowych. Załóżmy zatem, że przybliżeniem rozwiązania tego układu jest para (p,q), a szukamy lepszego przybliżenia postaci $(p+\delta p, q+\delta q)$. Potrzeba do tego pochodnych cząstkowych:

$$c_k = \frac{\partial b_k}{\partial p} \qquad d_k = \frac{\partial b_{k-1}}{\partial q}$$

Równanie dla a_k różniczkujemy stronami ze względu na p i obliczamy c_k:

$$a_k = b_k - pb_{k+1} - qb_{k+2}$$

$$0 = \frac{\partial b_k}{\partial p} - b_{k+1} - p * \frac{\partial k_{k+1}}{\partial p} - q * \frac{\partial b_{k+2}}{\partial p}$$

$$c_k = b_{k+1} + p * c_{k+1} + q * c_{k+2}$$

Analogiczną operację wykonujemy dla d_k, ale różniczkując ze względu na q.
Łącząc te wyniki, otrzymujemy:

$$\underbrace{\begin{bmatrix} c_0 & c_1 \\ c_1 & c_2 \end{bmatrix}}_{macierz\ biego} * \underbrace{\begin{bmatrix} \delta p \\ \delta q \end{bmatrix}}_{wektor\ poprawek} = - \begin{bmatrix} b_0(p,q) \\ b_1(p,q) \end{bmatrix}$$

Rozwiązaniem powyższego układu jest para:

$$\delta p = \frac{c_1 b_1 - c_2 b_0}{c_0 c_2 - c_1^2}$$

$$\delta q = \frac{c_1 b_0 - c_0 b_1}{c_0 c_2 - c_1^2}$$

Jako nowe, lepsze przybliżenie pierwiastka przyjmujemy $(p+\delta p, q+\delta q)$.

6.11 Przedstaw metodę Laguerra wyznaczania pierwiastków wielomianów

Metoda ta oparta jest o poniższe związki między wielomianem, jego pierwiastkami i pochodnymi:

1. $P_n(x) = (x-x_1) * (x-x_2) * ... * (x-x_n)$
2. $ln|P_n(x)| = ln|x-x_1| + ln|x-x_2| + ... + ln|x-x_n|$
3. $\frac{d}{dx} ln|P_n(x)| = \frac{1}{x-x_1} + \frac{1}{x-x_2} + ... + \frac{1}{x-x_n} = \frac{P_n'(x)}{P_n(x)} = G$
4. $-\frac{d^2}{dx^2} ln|P_n(x)| = \frac{1}{(x-x_1)^2} + \frac{1}{(x-x_2)^2} + ... + \frac{1}{(x-x_n)^2} = \left[\frac{P_n'(x)}{P_n(x)}\right]^2 - \frac{P_n''(x)}{P_n(x)} = H$

W oparciu o powyższe czynimy założenia:

1. x - bieżące (na początku odgadnięte) położenie pierwiastka

2. $a = x - x_1$ - odległość od pierwiastka
3. $b = x - x_i$, $i = 2, 3, ...n$ - odległość od pozostałych pierwiastków

Z powyższych założeń i związków wynika:

- ze związku 3.: $\frac{1}{a} + \frac{n-1}{b} = G$
- ze związku 4.: $\frac{1}{a^2} + \frac{n-1}{b^2} = H$

Z powyższych równań można już obliczyć a:

$$a = \frac{n}{G \pm \sqrt{(n-1) * (nH - G^2)}}$$

W powyższym wzorze znak wybieramy tak, aby mianownik miał większy moduł (bo wtedy dostaniemy mniejsze a - odległość od pierwiastka). Wzór działa także dla zespolonych a.
Metoda Laguerre'a wykorzystuje powyższy wzór, zaczynając od pewnego początkowego przybliżenia x, a następnie oblicza G, H oraz a i wykonuje $x_{i+1} = x_i - a$, aż zostanie osiągnięty warunek stopu (np. dostateczna precyzja). Jest ona zbieżna dla każdego $P(x)$ o współczynnikach rzeczywistych (niezależnie od wyboru początkowego x), ale wymaga obliczenia P, P' oraz P'' w każdym kroku.

6.12 Omów technikę Maehly'ego wygładzania pierwiastków wielomianów

Pro. Maehly'ego – technika wygładzania pierwiastków

Wykorzystujemy znane pierwiastki bez stosowania deflacji.
Równocześnie zapobiega zlewaniu się w jeden dwóch różnych pierwiastków (na etapie wygładzania)

Zredukowany wielomian

$$P_j(x) \equiv \frac{P(x)}{(x-x_1)\dots(x-x_j)}$$

Pochodna

$$P_j'(x) = \frac{P'(x)}{(x-x_1)\dots(x-x_j)} - \frac{P(x)}{(x-x_1)\dots(x-x_j)} \sum_{i=1}^{j} \frac{1}{x-x_i}$$

Rysunek 11: Slajd 1

Wykorzystujemy znane pierwiastki $(x_1, ..., x_j)$ do znalezienia kolejnych.
Pojedynczy krok metody Newtona-Raphsona można zapisać używając zredukowanego wielomianu:

$$x_{k+1} = x_k - \frac{P_j(x_k)}{P_j'(x_k)}$$

Zastępujemy zredukowany wielomian początkowym wielomianem, ale uwzględniamy informację o już znalezionych pierwiastkach:

$$x_{k+1} = x_k - \frac{P(x_k)}{P'(x_k) - P(x_k) \cdot \sum_{i=1}^{j} \frac{1}{(x_k - x_i)}}$$

Rysunek 12: Slajd 2

Dodatkowo nie narażamy się na propagację błędów.

6.13 Scharakteryzuj trudności występujące w rozwiązywaniu układów równań nieliniowych

1. Trzeba wybrać pewne początkowe przybliżenie pierwiastka $\vec{x_0}$, przy czym zły wybór może spowodować wolną zbieżność metody lub nawet sprawić, że w ogóle nie będzie zbieżna.
2. Liczba zer nie jest zawczasu znana, trzeba określić warunki końca, a ponadto bardzo ciężko jest znaleźć wszystkie miejsca zerowe.
3. Metody nie są niezawodne, mogą nie być zbieżne (w szczególności dla liczb zespolonych istnieją całe obszary niezbieżności). Można stosować tłumione wersje (najpierw poszukiwanie odpowiedniego obszaru, a potem szukanie pierwiastka), ale może to być nieskuteczne.

4. Mają dużą złożoność obliczeniową, w każdej iteracji trzeba na nowo obliczać macierze (jakobian i jego odwrotność), które mogą mieć też dużą złożoność pamięciową.

5. Szybko zbieżne metody wymagają formalnego różniczkowania, co może być trudne albo nawet niemożliwe dla niektórych funkcji.

7 Bezpośrednie metody rozwiązywania układów równań liniowych

7.1 Przedstaw algorytm rozwiązywania układów równań liniowych metodą eliminacji Gaussa

Algorytm eliminacji Gaussa polega zredukowaniu macierzy do postaci macierzy górnotrójkątnej za pomocą operacji elementarnych (mnożenie wiersza przez $\lambda \neq 0$, dodawanie wierszy, zamianę wierszy), a następnie obliczeniu poszukiwanych wartości od końca (*backward substitution*).

```
for i := i to n-1 do
    p := largest number != 0 in i-th column
    if no p then no unique solution exists, STOP
    if p != i then swap rows Ep <-> Ei
    for j := i+1 to n do
      Ej := Ej-a[j][i]/a[i][i] * Ei
if a[n][n]=0 then no unique solution exists, STOP
```

Wybór jak największego p bierze się stąd, że później dzielenie przez $p = a_{ii}$ spowoduje wtedy możliwie mały błąd i nie spowoduje overflowu (mógłby wystąpić przy dzieleniu przez małe elementy). Formalnie wystarczyłoby jakiekolwiek $p \neq 0$, żeby metoda działała, ale taki *partial pivoting* zwiększa też stabilność numeryczną algorytmu.
Element w ostatnim wierszu po eliminacji można łatwo obliczyć (tylko a'_{nn} jest niezerowe), a następnie obliczyć wartości dla $i = n-1, n-2, ..., 1$:

$$x_n = \frac{b'_n}{a'_{nn}} \qquad x_i = \frac{1}{a_{ii}} * \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right)$$

Metoda ta ma złożoność $O(n^3)$.

7.2 Wyznacz złożoność obliczeniową metody Gaussa rozwiązywania układów równań liniowych

Pseudokod pierwszego etapu, eliminacji:

```
for i := i to n-1 do
    p := largest number != 0 in i-th column
    if no p then no unique solution exists, STOP
    if p != i then swap rows Ep <-> Ei
    for j := i+1 to n do
      Ej := Ej-a[j][i]/a[i][i] * Ei
if a[n][n]=0 then no unique solution exists, STOP
```

W każdym obiegu pętli (których jest $n-1$ przy macierzy o n x n) wykonuje się operacje:

1. Wybór piwotu (największego elementu w kolumnie): dla i-tej kolumny wymaga sprawdzenia $n-i$ elementów.

2. Zamiana wierszy: zwykle wymaga tylko przepięcia wskaźników (nawet w językach wyższego poziomu tak działa "pod spodem") i jest w czasie $O(1)$.

3. Eliminacja: dla każdego wiersza poniżej dla danej kolumny i każdej kolejnej odjąć obecny wiersz pomnożony przez $\frac{a_{ji}}{a_{ii}}$, co daje zera pod główną przekątną. Dla i-tego wiersza jest to $(n-i)*(n-i+1)$ operacji.

Z powyższego wynika, że część eliminacyjna ma złożoność:

$$\sum_{i=1}^{n-1}((n-i)+(n-i)*(n-i+1)) = O(n^3)$$

W części *backward substitution* wykorzystuje się wzór dla $i = n-1, n-2, ..., 1$:

$$x_n = \frac{b'_n}{a'_{nn}} \qquad x_i = \frac{1}{a_{ii}} * \left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j \right)$$

Wprost ze wzoru wynika jego złożoność $O(n^2)$ (bo trzeba $n-1$ razy wykonać $n-i$ operacji).
Sumarycznie więc złożoność metody Gaussa to $O(n^3)$.

7.3 Wyjaśnij dlaczego istotnym krokiem każdej metody rozwiązywania układów równań liniowych jest szukanie elementu wiodącego (głównego), a następnie opisz gdzie i jak go się poszukuje

Element wiodący (główny, piwot) - element macierzy wybierany do przeprowadzania głównych operacji w algorytmach algebry liniowej, np. eliminacji Gaussa czy metodzie simpleksów. Czasami jest on niezbędny dla działania algorytmu, a czasami zwiększa stabilność numeryczną.
W przypadku rozwiązywania układów równań liniowych (eliminacja Gaussa, rozkład LU) piwotem jest element, przez który dzielimy odejmowane elementy wierszy. W związku z tym musi on być niezerowy, aby algorytm działał (w tym celu może być konieczne najpierw znalezienie go i przesunięcie na pozycję na diagonali), a im większa jest jego wartość bezwzględna, tym bardziej algorytm jest stabilny (bo to przez niego dzielimy).
Istnieją trzy główne metody jego poszukiwania:

1. Częściowe poszukiwanie (*partial pivoting*) - k-ty piwot to element z k-tej kolumny (jej części poniżej k-tego wiersza z nim włącznie) o największej wartości bezwzględnej. Zamienia się miejscami tylko wiersze.

2. Pełne poszukiwanie (*full pivoting*) - na k-ty piwot wybiera się element o największej wartości bezwzględnej z podmacierzy od k-tego wiersza włącznie w dół i od k-tej kolumny włącznie w prawo. Wymaga zamian wierszy i kolumn, a więc np. trzymania macierzy permutacji (żeby pamiętać kolejność niewiadomych). Daje lepszą stabilność numeryczną od częściowego poszukiwania, ale wymaga przeszukania aż k^2 elementów.

3. Skalowane poszukiwanie (*scaled pivoting*) - jako piwot wybierany jest element największy w stosunku do reszty elementów w swoim wierszu (relatywnie największy w stosunku do największego elementu w wierszu). Pozwala to uniknąć błędów ze względu na różne rzędy wielkości liczb w wierszu, więc używa się tej metody, gdy mamy macierz z elementami o bardzo różnych rzędach wielkości.

4. Dla metod iteracyjnych - głównie dla macierzy rzadkich (dużo zer, tam te metody są najefektywniejsze), przestawia się wiersze macierzy tak, aby na diagonali nie było zer, a elementy niezerowe o jak największym module.

7.4 Objaśnij jaki jest cel i na czym polega wyważanie macierzy

Wyważanie macierzy (skalowanie, równoważenie) to takie przekształcenie macierzy, żeby maksymalne elementy w każdej kolumnie i każdym wierszu były tego samego rzędu.
Wyważanie macierzy nie jest jednoznaczne - może istnieć wiele zrównoważonych postaci macierzy o różnych właściwościach, np. jedno może być poprawne, a drugie dać macierz osobliwą na maszynie o małej liczbie

cyfr znaczących.
Operacja ta poprawia często wskaźnik uwarunkowania macierzy w danej normie, czyniąc ją bardziej stabilną numerycznie. Ma to zastosowanie np. w metodach iteracyjnych algebry liniowej, gdzie pozwala zmniejszyć błędy zaokrąglania w obliczaniu iloczynów wektorów macierzy. W połączeniu z przestawianiem wierszy (permutowaniem macierzy) można też tak wyważyć macierz, aby nie trzeba było stosować pivotingu (poszukiwania elementu wiodącego) w eliminacji Gaussa.

7.5 Opisz i porównaj algorytmy faktoryzacji LU Doolittle'a, Crout'a i Choleskiego

Faktoryzacja LU polega na zapisaniu macierzy A jako iloczynu macierzy dolnotrójkątnej L i górnotrójkątnej U. W praktyce obie te macierze przechowuje się w jednej, gdyż niezależnie od przyjętej metody nigdy nie trzeba pamiętać 2 różnych elementów na diagonali. Dla $k = 1, 2, ..., n-1$ w k-tym kroku wykonujemy operację:

$$a_{kk} = \sum_{s=1}^{k-1} l_{ks} v_{sk} + l_{kk} v_{kk} \quad (*)$$

W powyższej operacji oblicza się l_{kk} lub v_{kk}, z której następnie można obliczyć (dla $j = k+1, k+2, ..., n-1$):

$$a_{kj} = \sum_{s=1}^{k-1} l_{ks} v_{sj} + l_{kk} v_{kj} \quad \text{k-ty wiersz}$$

$$a_{ik} = \sum_{s=1}^{k-1} l_{is} v_{sk} + l_{ik} v_{kk} \quad \text{k-ta kolumna}$$

W operacji $(*)$ trzeba znać l_{kk} lub v_{kk}, aby wyliczyć drugą. W zależności od wybranego algorytmu faktoryzacji otrzymuje się to w różny sposób:

- **faktoryzacja Doolitle'a** ma $l_{ii} = 1$, $i \in [0, n-1]$. Daje ona rozkład $LDU = L(DU)$ (gdzie D to macierz z wartościami na diagonali).
- **faktoryzacja Crouta** ma $u_{ii} = 1$, $i \in [0, n-1]$. Daje ona rozkład $LDU = (LD)U$.
- **faktoryzacja Cholesky'ego** ma $l_{ii} = u_{ii}$, $i \in [0, n-1]$. Daje ona rozkład $(LD^{\frac{1}{2}})(D^{\frac{1}{2}}U) = LL^T$. Faktoryzację tę można zastosować tylko dla macierzy rzeczywistej, symetrycznej i dodatnio określonej, ale algorytm ten jest szybszy od pozostałych.

7.6 Objaśnij na czym polega przewaga algorytmów faktoryzacji LU nad metodą eliminacji Gaussa

Główną przewagą faktoryzacji LU jest oddzielenie faz faktoryzacji (obliczenia macierzy L i U) oraz rozwiązania układu równań $Ax = B$. Dzięki temu:

1. Można zrównoleglić obliczenia przy rozwiązywaniu układu równań wielokrotnie dla różnych macierzy B.
2. Koszt obliczeń dla rozwiązywania jednego układu równań jest taki sam, a jest niższy, kiedy układ równań rozwiązuje się wielokrotnie dla różnych macierzy B: dla LU wynosi $O(n^3 + k * n^2)$, a dla eliminacji Gaussa $O(k * n^3)$.

7.7 Wyjaśnij na czym polega przydatność metod blokowych do rozwiązywania układów równań liniowych

1. Stanowią kompromis między prostotą metod iteracyjnych a efektywnością metod dokładnych.
2. Zapisanie macierzy rzadkiej przekątniowej jako macierzy blokowej przekątniowej nie zaburza struktury macierzy rzadkiej.

3. W równaniach różniczkowych cząstkowych często pojawiają się przekątniowe macierze rzadkie, które można zapisać jako macierze przekątniowe blokowe i zastosować dla nich szybkie metody bezpośrednie.
4. W metodach iteracyjnych (np. Jacobiego, S-R) korzystamy z łatwości odwracania macierzy diagonalnej D, a za tę macierz można przyjąć inną macierz łatwą do odwracania (np. trójdiagonalną), co pozwala przy poprawnym wyborze macierzy uzyskać szybką zbieżność i jednocześnie zachować iteracyjny charakter metody.

8 Iteracyjne metody rozwiązywania układów równań

8.1 Wyjaśnij kiedy warto używać iteracyjnych metod rozwiązywania układów równań liniowych

- dla specyficznych układów równań, które powstają w wyniku rozwiązywanie równań różniczkowych
- kiedy nie potrzebujemy dokładnego rozwiązanie i wystarczy jego przybliżenie
- gdy układy są nieliniowe
- Dla macierzy rzadkich metody iteracyjne zachowują ich strukturę (bezpośrednie ją zmieniają), dzięki czemu można zapamiętywać macierze rzadkie w efektywnych strukturach i zmniejszać koszty pamięciowe.

8.2 Podaj i udowodnij twierdzenie o zbieżności procesu iteracyjnego rozwiązywania Ax = b

Twierdzenie o zbieżności procesu iteracyjnego rozwiązywania Ax=b - ciąg rozwiązań iteracyjnych postaci $x^{(t+1)} = M * x^{(t)} + W$, gdzie $A = B + R$, $M = I - B^{-1}A$ i $W = B^{-1}b$ z dowolnym wektorem startowym $x^{(0)}$ jest zbieżny do jedynego granicznego $x^{(\infty)}$ wtedy i tylko wtedy, gdy promień spektralny macierzy iteracji M jest mniejszy od 1, tzn. $\rho(M) < 1$, $\rho = max\{|\lambda_i|\}$ (gdzie λ_i to i-ta wartość własna macierzy M).

Dowód:
Oznaczmy wektor błędu w iteracji t jako:

$$\epsilon^{(t)} = x^t - x$$

Powyżej x to rozwiązanie dokładne). Na początku dla $x^{(0)}$ mamy $\epsilon^{(0)}$. Kiedy macierz iteracji M zmienia się w procesie iteracji, to:

$$\epsilon^{(t)} = M^t * \epsilon^{(0)}$$

Aby proces był zbieżny, to dla skalara $||\epsilon^{(t)}||$ musi zachodzić:

$$||\epsilon^{(t+1)}|| < ||\epsilon^{(t)}||$$

Macierz iteracji M ma n różnych wartości własnych ρ_i i tyle samo różnych wektorów własnych s_i. Można zatem zapisać:

$$Ms_i = \rho_i s_i$$

Zapiszmy $\epsilon^{(t)}$ z użyciem powyższych oznaczeń (α_i - amplituda kierunku s_i):

$$\epsilon^{(t)} = M^t \sum_{i=1}^{n} \alpha_i s_i = \sum_{i=1}^{n} \alpha_i M^{t-1} \underbrace{Ms_i}_{\rho_i s_i} = \cdots = \sum_{i=1}^{n} \alpha_i \rho_i^t s_i$$

$$\epsilon^{(t)} = \sum_{i=1}^{n} \alpha_i \rho_i^t s_i$$

Weźmy teraz granicę tego błędu $\epsilon^{(t)}$ ($\rho_m = \max_i |\rho_i| = \rho$ - promień spektralny macierzy iteracji):

$$\lim_{t\to\infty} \epsilon^{(t)} = \alpha_m \rho_m^t s_m$$

Jak widać, granica ta istnieje tylko dla $\rho < 1$ (bo α_m i s_m to stałe), co jest warunkiem zbieżności i to kończy dowód.

8.3 Podaj po kolei wzory robocze i macierzowe dla metod iteracyjnych Jacobiego, S-R, SOR, Czebyszewa

Wprowadźmy oznaczenia dla macierzy: D - macierz diagonalna, L - macierz poddiagonalna, U - macierz naddiagonalna.
Rozważamy układ równań postaci $Ax = b$, gdzie $A = D + L + U$. Szukamy wektora rozwiązań za pomocą kolejnych iteracyjnych przybliżeń $x^{(n+1)}$. Można zatem zapisać:

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

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

Metoda Jacobiego:
Korzystamy wprost z definicji dla $Dx^{(t+1)}$:

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

$$x^{(t+1)} = D^{-1} * \left[b - (L + U)x^{(t)}\right]$$

Powyższy wzór to wzór macierzowy dla metody Jacobiego. Aby otrzymać wzór roboczy, wystarczy zapisać:

$$x^{(t+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(t)} \right] \quad a_{ii} \neq 0, \forall i \in 1, 2, ..., n$$

Na powyższym wzorze roboczym dla metody Jacobiego widać, że macierz diagonalna D nie może mieć żadnych zer na diagonali.

Metoda S-R:
Zapiszmy:

$$A = \underbrace{D + L}_{B} + U = B + U$$

Dzięki takiemu pogrupowaniu wyrazów można zapisać:

$$M = I - B^{-1}A = I - B^{-1}(B + U) = I - I - B^{-1}U = -B^{-1}U$$

$$x^{(t+1)} = Mx^{(t)} + W$$

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

Po pomnożeniu powyższego wyrażenia stronami przez $B = D + L$ otrzymujemy:

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

$$Dx^{(t+1)} = -Lx^{(t+1)} - Ux^{(t)} + b$$

$$x^{(t+1)} = D^{-1} * \left[b - Lx^{(t+1)} - Ux^{(t)}\right]$$

Powyższy wzór to wzór macierzowy metody S-R. Aby otrzymać wzór roboczy, wystarczy zapisać:

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

W powyższym wzorze pierwsza suma jest znana, bo otrzymuje się ją z rozwiązania poprzednich równań w bieżącej $t + 1$ iteracji.

Metoda SOR:
Wzór roboczy metody S-R można zapisać jako:

$$x_i^{(t+1)} = x_i^{(t)} + \underbrace{\frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(t+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(t)}\right]}_{r_i^{(t)} - \text{ poprawka do starego rozwiązania } x_i^{(t)}}$$

Ideą metody SOR jest przyspieszenie zbieżności wykorzystując pewną stałą ω:

$$x_i^{(t+1)} = x_i^{(t)} + \omega r_i^{(t)}$$

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

Po uproszczeniu powyższego wzoru można go zapisać roboczo i macierzowo:

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

$$x_i^{(t+1)} = \underbrace{(D+\omega L)^{-1}[D-\omega(D+U)]}_{M}x^{(t)} + \underbrace{\omega(D+\omega L)^{-1}b}_{W=B^{-1}b}$$

Metoda Czebyszewa:
Przyjmujemy $B^{-1} = W(A)$, czyli taki wielomian macierzowy, że $M = 1 - W(A) * A$. Taki $W(A)$ (gdzie $A = L + D + U$), że $\min \max_U |1 - W(U) * U|$ to wielomian Czebyszewa.
Obliczenia wykonuje się najpierw dla węzłów parzystych, potem dla nieparzystych (odpowiednio t całkowite i t ułamkowe). Różnice względem SOR to tylko postać B^{-1} i zmienna ω, więc poza tym wzory są takie jak dla SOR (ρ - promień spektralny macierzy iteracji M):

$$\omega^{(0)} = 1 \qquad \omega^{\frac{1}{2}} = \frac{1}{1 - \frac{1}{2}\rho^2}$$

$$\omega^{(t+\frac{1}{2})} = \frac{1}{1 - \frac{1}{4}\rho^2\omega^{(t)}} \quad t = \frac{1}{2}, 1, \frac{3}{2}, ...$$

$$\omega^{(\infty)} = \omega_{opt}$$

8.4 Porównaj metody iteracyjne Jacobiego, GS, SOR, Czebyszewa

Metoda Jacobiego - wynika wprost z definicji iteracyjnego przybliżania $x^{(n+1)}$ i korzysta ze wzoru:

$$A = D + L + U \qquad Ax = b$$

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

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

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

- wolno zbieżna
- nie wykorzystywana w praktyce - ma znaczenie dydaktyczne
- nie wykorzystuje całej dostępnej w danym kroku informacji
- zbieżna dla silnie diagonalnie dominujących macierzy A

Metoda S-R - ulepszenie metody Jacobiego, wykorzystuje zapis $A = (L+D)+U$ jako $B = L+D$, korzysta ze wzoru:

$$x^{(t+1)} = D^{-1} * [-Lx^{(t+1)} - Ux^{(t)} + b]$$

Powyżej $x^{(t+1)}$ jest znane, bo obliczone wcześniej w danej iteracji.

- wykorzystuje informacje z wcześniejszych obliczeń z danej iteracji, stąd wzrost efektywności
- elemety diagonali muszą być różne od 0
- zbieżna dla macierzy A silnie diagonalnie dominujących wierszowo lub kolumnowo i dla macierzy symetrycznych i dodatnio określonych

Metoda SOR - ulepszenie metody S-R, wykorzystuje poprawkę rozwiązania $r_i^{(t)}$:

$$x_i^{(t+1)} = x_i^{(t)} + \omega r_i^{(t)}$$

Powyżej ω to pewna liczba (wskaźnik relaksacji), przy czym dla ważnych praktycznie klas macierzy znane są optymalne wartości ω.

- przyspiesza zbieżność metody S-R poprzez wykorzystanie poprawki
- metoda S-R to szczególny przypadek metody SOR dla $\omega = 1$

Metoda Czebyszewa - ulepszenie metody SOR, w którym $B^{-1} = W(A)$ ($W(A)$ - wielomian macierzowy, $A = L+D+U$) i $min \max_U |1-W(U)*U|$ - wykorzystuje się wielomiany Czebyszewa. Istotą metody jest także użycie zmiennych wartości ω zgodnie ze wzorem (ρ - promień spektralny), w którym obliczenia przeprowadza się najpierw dla węzłów parzystych (t całkowite), a potem dla nieparzystych (t ułamkowe):

$$\rho = \rho(M)$$

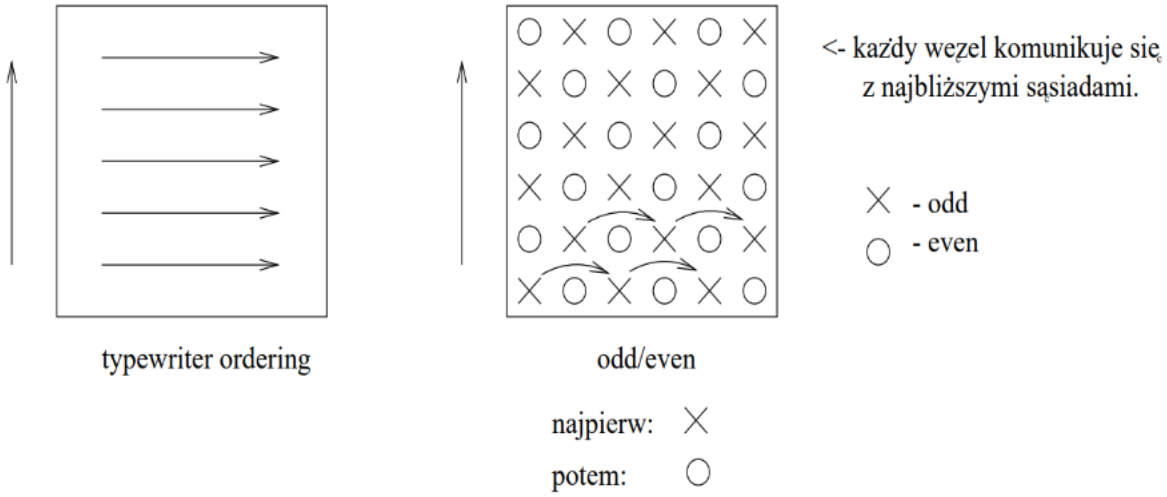
$$\omega^{(0)} = 1$$

$$\omega^{(\frac{1}{2})} = \frac{1}{1 - \frac{1}{2}\rho^2}$$

$$\omega^{(t+\frac{1}{2})} = \frac{1}{1 - \frac{1}{4}\rho^2\omega^{(t)}} \quad t = \frac{1}{2}, 1, \frac{3}{2}, ...$$

$$\omega^{(\infty)} = \omega_{opt}$$

8.5 Objaśnij różnice między przeglądaniem punktów siatki typu type writer oraz odd-even



Odd/even pozwala na zrównoleglanie pewnych procesów iteracyjnych

8.6 Podaj i scharakteryzuj 4 przykłady użyteczności wielomianów Czebyszewa w obliczeniach numerycznych

1. **Węzły interpolacyjne Czebyszewa** - jeżeli w interpolacji wielomianowej na węzły wybierze się zera wielomianów Czebyszewa, to nie wystąpi efekt Rungego.
2. **Baza aproksymacji** - wielomiany Czebyszewa można wybrać jako bazę aproksymacji układu normalnego z węzłami w zerach tych wielomianów, otrzymuje się wtedy aproksymację jednostajną Czebyszewa z własnością minimaksu (minimalnego maksymalnego błędu).
3. **Całkowanie numeryczne** - w kwadraturach Gaussa do obliczenia optymalnych węzłów wykorzystuje się wielomiany Czebyszewa. Wykorzystuje je także kwadratura Clenshawa-Curtisa.
4. **Iteracyjna niestacjonarna metoda Czebyszewa** - w rozwiązywaniu układów równań liniowych metody niestacjonarne są szybciej zbieżne, zmieniają one macierz iteracji. Jedną z takich metod jest metoda Czebyszewa wykorzystująca jego wielomiany, pozwala ona uniknąć obliczania iloczynów skalarnych macierzy.

8.7 Porównaj zasadę działania metod iteracyjnych do rozwiązywania równań nieliniowych i do rozwiązywania układów równań liniowych: ogólny algorytm, potrzebne twierdzenia, kiedy są przydatne

Metody iteracyjne - metody (algorytmy), których idea opiera się na ulepszaniu rozwiązania wraz z kolejnymi iteracjami (powtórzeniami procesu przybliżania).
Ogólny algorytm:

```
x = x0  // initial guess
max_iterations  // required number of iterations, usually more = greater precision
i = 0
while i < max_iterations:
  x = F(x)  // improve solution
  i += 1
```

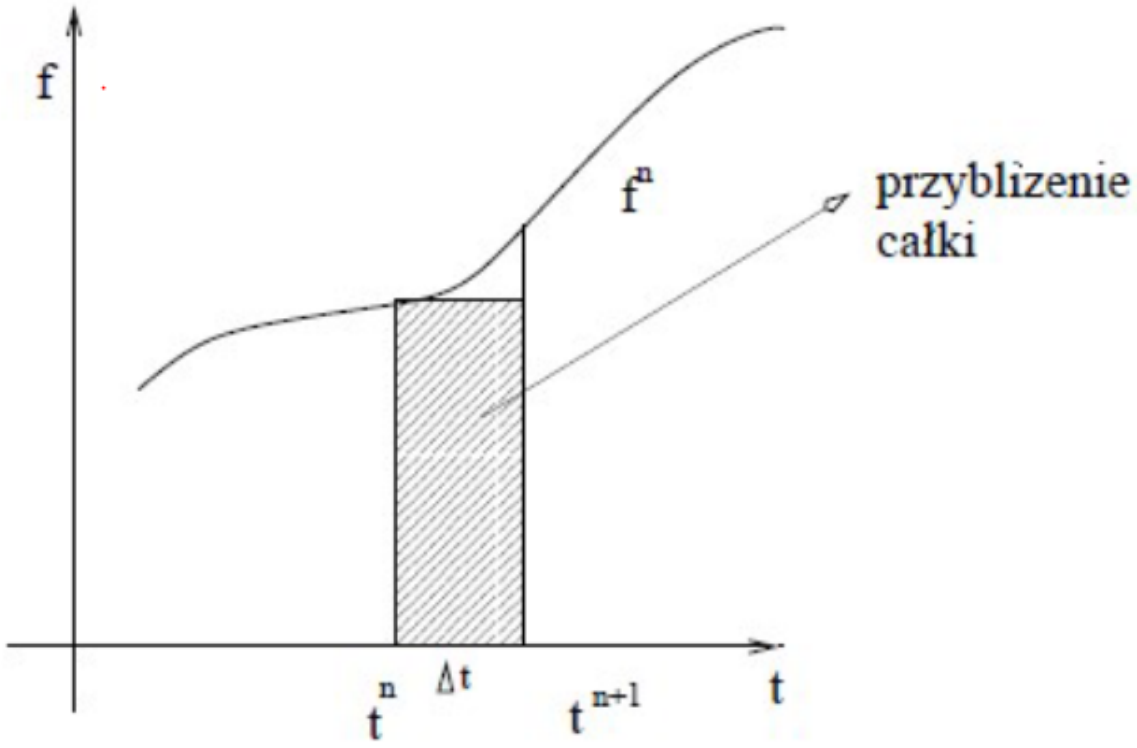
Do wykorzystywania metod iteracyjnych do konkretnych problemów potrzebne są twierdzenia zapewniające warunki zbieżności. Metody można stosować tylko dla problemów, dla których te warunki są spełnione, czyli iteracje będą dawać coraz lepsze rozwiązania. Są to np. twierdzenie o zbieżności procesu iteracyjnego $Ax = b$ dla układów równań liniowych czy twierdzenie o zbieżności procesu iteracyjnego $x = h(x)$ dla równań nieliniowych.
Złożoność obliczeniowa zależy liniowo od liczby operacji (ich złożoność zależy od konkretnego problemu), dlatego można dobrać ilość operacji w zależności od dostępnego czasu i wymaganej precyzji.
Metody iteracyjne są szczególnie przydatne, kiedy możemy zadowolić się przybliżonym rozwiązaniem lub chcemy mieć kontrolę nad złożonością czasową problemu i związaną z tym precyzją rozwiązania. Ponadto metody te mogą mieć dodatkowe właściwości zależne od konkretnego problemu, np. w układach równań liniowych nie naruszają one struktur macierzy rzadkich.

8.8 Porównaj rozwiązywanie układów równań liniowych metodami bezpośrednimi i iteracyjnymi

1. Dla macierzy pełnych metody bezpośrednie są szybsze od iteracyjnych.
2. Dla macierzy rzadkich metody iteracyjne zachowują ich strukturę (bezpośrednie ją zmieniają), dzięki czemu można zapamiętywać macierze rzadkie w efektywnych strukturach i zmniejszać koszty pamięciowe.
3. Dla niektórych przypadków metody iteracyjne mogą się cyklicznie zapętlić, nie dając rozwiązania, a metody bezpośrednie zawsze dadzą rozwiązanie.
4. Kiedy nie interesuje nas dokładne rozwiązanie (z dokładnością do błędów zmiennoprzecinkowych), to można użyć metod iteracyjnych do otrzymania przybliżonego rozwiązania i zwiększenia szybkości - z metodami bezpośrednimi nie da się tak zrobić.
5. Metody iteracyjne są zwykle stabilne (błędy zaokrągleń są wygaszane w trakcie obliczeń), podczas gdy metody bezpośrednie nie mają takich właściwości.

9 Równania różniczkowe zwyczajne

9.1 Omów metodę Eulera rozwiązywania równań różniczkowych zwyczajnych



$$u^{n+1} = u^n - f(u^n, t^n)\Delta t$$

$$\epsilon^{n+1} = \epsilon^n - \frac{\partial f}{\partial u}\Big|_n \Delta t \epsilon^n$$

- stabilna if $\frac{\partial f}{u} > 0$
- błąd $O(\Delta t)$ (pierwszego rzędu)
- jawna
- prosta
- efektywna

$$\epsilon^{n+1} = g * \epsilon^n$$

9.2 Omów sposób badania stabilności metod rozwiązywania równań różniczkowych zwyczajnych (ODE) na podstawie metody Eulera. Podaj przykłady

$$u^{n+1} + \epsilon^{n+1} = u^n + \epsilon^n - f(u^n + \epsilon^n, t^n)\Delta t(*)$$

e^n -> małe => rozwijamy Taylorem pomijając nieliniowe:

$$f(u^n + \epsilon^n, t^n) = f(u^n, t^n) + \frac{\partial f}{\partial u}\Big|_n \epsilon^n$$

$$u^{n+1} + \epsilon^{n+1} = u^n + e^n - [f(u^n, t^n) + \frac{\partial f}{\partial u}\Big|_n \epsilon^n] * \Delta t$$

$$\epsilon^{n+1} = \epsilon^n - \frac{\partial f}{\partial u}\Big|_n \epsilon^n * \Delta t$$

$$\frac{\epsilon^{n+1}}{\epsilon^n} = g = 1 - \frac{\partial f}{\partial u}\Big|_n \Delta t$$

WARUNEK STABILNOŚCI $|g| <= 1$

$$\frac{\partial f}{\partial u}\Delta t <= 2$$

$$\Delta t <= \frac{2}{\frac{\partial f}{\partial u}}$$

Przykłady

1. Rozkład promieniotwórczy

$$\frac{du}{dt} + \frac{u}{\tau} = 0, u(0) = 1$$

$$u = e^{-\frac{t}{\tau}}$$

stabilne gdy:

$$\Delta t <= \frac{2}{\frac{\partial f}{\partial u}\Big|_n}$$

$$\frac{\partial f}{\partial u}\Big|_n = \frac{1}{\tau}$$

$$\Delta t <= 2\tau$$

np. równanie oscylatora

$$\frac{d^2x}{dt^2} + \omega^2 x = 0 \qquad v = \frac{dx}{dt}$$

$$\begin{cases} \frac{dv}{dt} = \omega^2 x = 0 \\ \frac{dx}{dt} - v = 0 \qquad / \cdot (-i\omega) \end{cases}$$

Dodajemy stronami

2. Oscylator harmoniczny

$$\frac{dv}{dt} - i\omega\frac{dx}{dt} + \omega^2 x + i\omega v = 0$$

$$\frac{dv}{dt} - i\omega\frac{dx}{dt} - i^2\omega^2 x + i\omega v = 0$$

$$- \text{''} - + i\omega(v - i\omega x) = 0$$

Nowe wprowadzone $u = v - i\omega x$

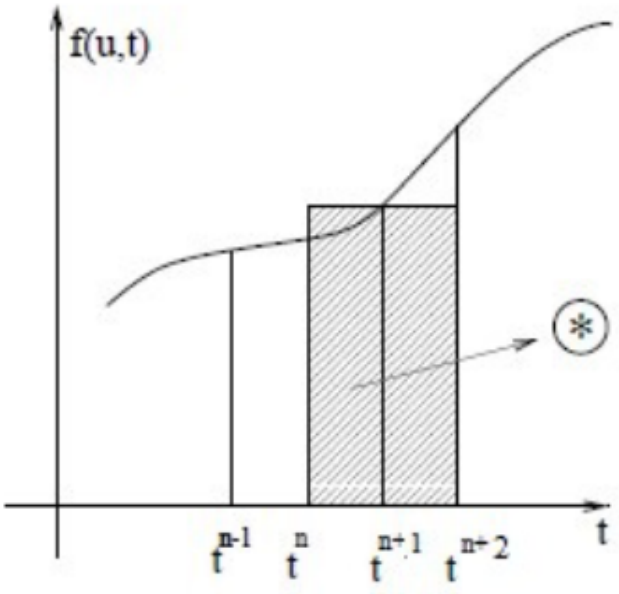
$$\frac{du}{dt} + i\omega u = 0 \qquad f(u) = i\omega u$$

$$g = 1 - \frac{\partial f}{\partial u}\Big|_n \Delta t \Rightarrow g = 1 - i\omega\Delta t$$

$$|g|^2 = 1 + \omega^2\Delta t^2 > 1$$

metoda niestabilna

9.3 Omów metodę skokową rozwiązywania równań różniczkowych zwyczajnych



Założenia:

- całkujemy na kroku czasowym o podwójnej długosci
- punkt czasowy w srodku takiego kroku uzywamy dla obliczenia całki metoda prostokatów
- metoda wycentrowana w czasie; dokładnosc drugiego rzedu $\epsilon = 0(\Delta t^2)$

Algorytm:

$$u^{n+1} := u^{n-1} - f(u^n, t^n) * 2 * \Delta t$$
$$u^{n+2} := u^n - f(u^{n+1}, t^{n+1}) * 2 * \Delta t(*)$$

Trudności:

- znamu u^0 = u(0), potrzebne u^1 = u(Δt). Od u^1 zależy całkowita dokładność
- punkt czasowy w srodku takiego kroku uzywamy zagadnienie nieliniowe -> zmienne Δt, niewycentrowane w czasie

- metoda wycentrowana w czasie ! dokładnosc drugiego rzedu $\epsilon = 0(\Delta \mathrm{t}^2)$

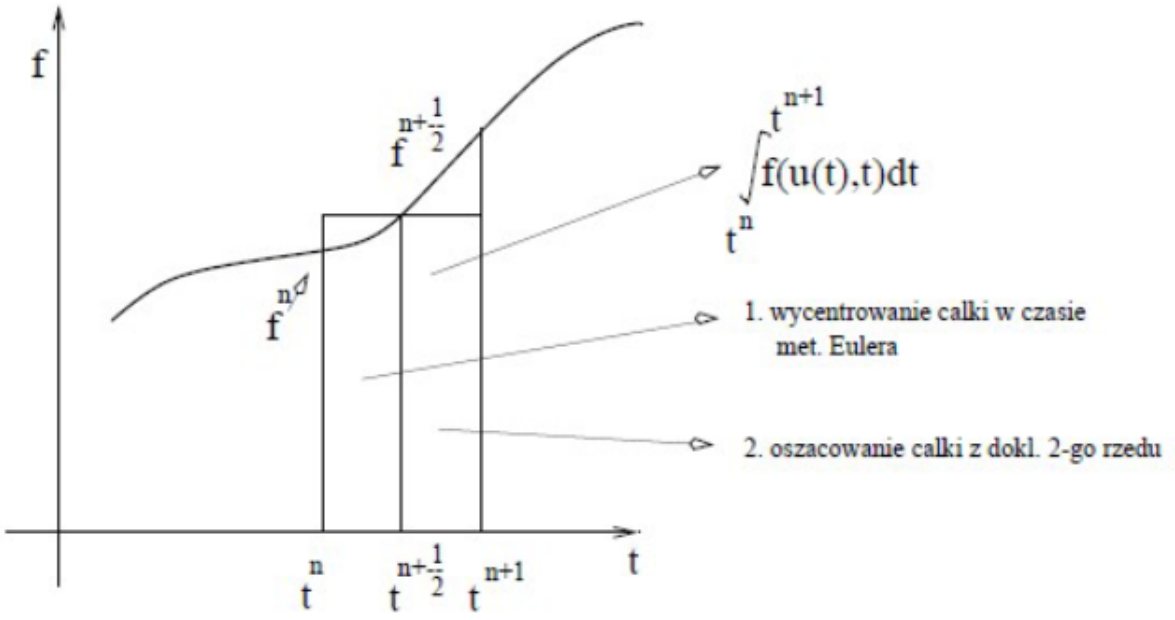
  Współczynnik wzmocnienia:

$$\epsilon^{n+1} = \epsilon^{n-1} - \frac{\partial f}{\partial u}\Big|_n * 2 * \Delta t * \epsilon^n | : \epsilon^{n-1}$$

$$\alpha = \frac{\partial f}{\partial u}\Big|_n * 2 * \Delta t; g = \frac{\epsilon^{n+1}}{\epsilon^n} \approx \frac{\epsilon^n}{\epsilon^{n-1}}; g^2 = \frac{\epsilon^{n+1}}{\epsilon^n} * \frac{\epsilon^n}{\epsilon^{n-1}}$$
$$g^2 = 1 - \alpha * 2g$$
$$g = -\alpha \pm \sqrt{\alpha^2 + 1}$$

  - if $\alpha \in \mathbf{R} \implies |g| > 1 \rightarrow$ niestabilna
  - if $\alpha \in \mathbf{C}$ & $\alpha = i\beta$ where $\beta \leqslant 1 \implies g = -i\beta \pm \sqrt{1 - \beta^2}; |g|^2 = g * g^* = 1$

9.4 Omów metodę ulepszoną Eulera rozwiązywania równań różniczkowych zwyczajnych



$u^{n+\frac{1}{2}}$- pomocnicze, nie zachowywane powyżej t^{n+1}

1. wyliczenie zmiennej u dla pośredniego $t^{n+\frac{1}{2}}$ metodą Eulera

$$u^{n+\frac{1}{2}} = u^n - f(u^n, t^n)\frac{\Delta t}{2}$$

2. Wylcizenie właściwej wartości u^{n+1} na podstawie $u^{n+\frac{1}{2}}$ z poprzedniego kroku:

$$u^{n+1} = u^n - f(u^{n+\frac{1}{2}}, t^{n+\frac{1}{2}}) * \Delta t$$

Stabilność:

$$\epsilon^{n+1} = \epsilon^n - \frac{\partial f}{\partial u}\Big|_n \Delta t[1 - \frac{\partial f}{\partial u}\Big|_n \frac{\Delta t}{2}] * \epsilon^n$$

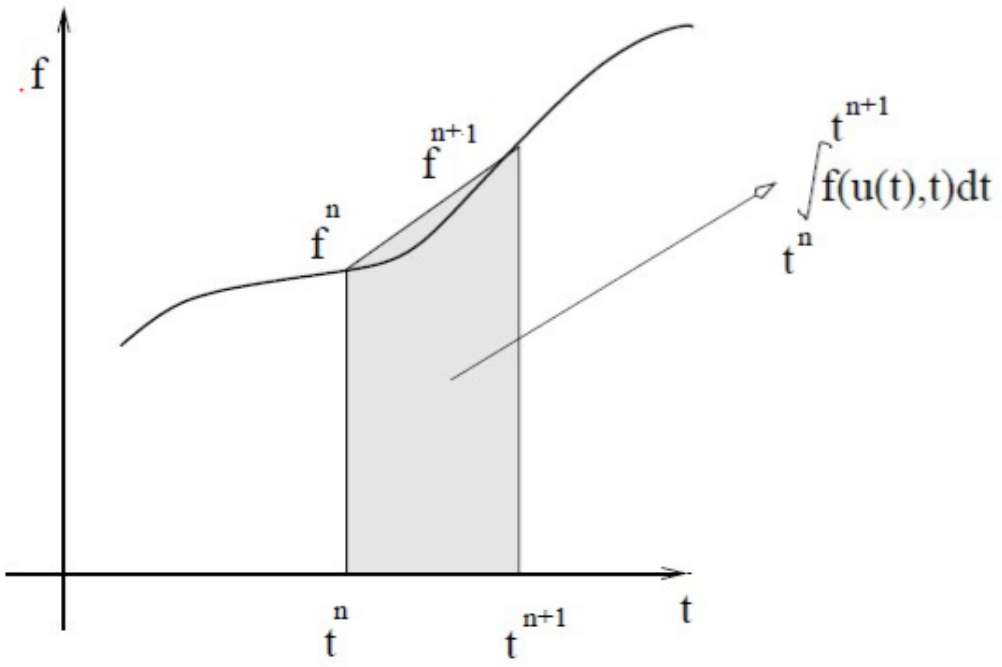
$$\alpha = \frac{\partial f}{\partial u}\Big|_n \Delta t$$

$$\epsilon^{n+1} = \epsilon^n - \alpha * [1 - \frac{\alpha}{2}] * \epsilon^n$$

$$g = 1 - \alpha + \frac{1}{2}\alpha^2$$

- stabilna dla $\alpha \in R$, jeśli $\Delta t \leqslant \frac{2}{\frac{\partial f}{\partial u}\Big|_n}$
- może być niestabilna if $\alpha \in \mathbf{C}$
- jawna

9.5 Omów niejawną metodę drugiego rzędu rozwiązywania ODE. Porównaj z metodami jawnymi: Eulera i ulepszonego Eulera



- u^{n-1} - uwikłane (nieznany argument funkcji $f(u^n, t^n)$)
- gdy f bardziej skomplikowana to w kazdym kroku czasowym moze wymagac rozwiazania równania

$$u^{n+1} = u^n - [f(u^n, t^n) + f(u^{n+1}, t^{n+1})] * \frac{\Delta t}{2}$$

Stabilność:

$$g = 1 - \frac{\partial f}{\partial u}\Big|_n * \frac{\Delta t}{2} - \frac{\partial f}{\partial u}\Big|_{n+1} * \frac{\Delta t}{2} * g$$

$$g = \frac{1 - \frac{\partial f}{\partial u}\Big|_n * \frac{\Delta t}{2}}{\frac{\partial f}{\partial u}\Big|_{n+1} * \frac{\Delta t}{2}}$$

- (+) metoda jest bezwzględnie stabilna (ważne w zagadnieniach nieliniowych)
- (+) metoda 2-go rzędu
- (-) trzeba rozwiązywać równanie algebraiczna na u^{n+1} lub stosować wzór iteracyjny

9.6 Przedstaw zasadę konstruowania metod Rungego Kutty. Podaj związki z metodą Eulera oraz ulepszonego Eulera

$$(***) = \begin{cases} u_{i+1} = u_i + hF(t_i, u_i, h), & i = 0, 1, 2, ..., n-1 \\ F(t,u,h) = c_1 k_1(t,u,h) + c_2 k_2(t,u,h) + ... + c_r k_r(t,u,h) & \\ k_1(t,u,h) = f(t,u) & \\ k_j(t,u,h) = f(t + ha_j, u + h\sum_{s=1}^{j-1} b_{js} k_s(t,u,h)), & j = 2, ..., r \\ a_j = \sum_{s=1}^{j-1} b_{js} & \end{cases} \quad (1)$$

a_j, b_{js}, c_k - stałe rzeczywiste, dobrane tak, alby:

$$\phi(h) = F(t,u,h) - F_T(t,u,h)$$

zawierały jedynie potęgi h możliwie wysokiego rzędu

Euler jeśli:

- r = 1
- $c_2 = c_3 = a_2 = a_3 = b_{32} = 0$
- $c_1 = 1$

Ulepszony Euler jeśli:

- r = 2
- $c_1 = c_3 = b_{32} = 0$
- $c_2 = 1$
- $a_2 = \frac{1}{2}$

10 Liczby losowe i całkowanie Monte Carlo

10.1 Podaj przykłady i opisz działanie generatorów liczb równomiernych

$$x_{n+1} = f(x_n, x_{n+1}, ..., x_{n-k+1})(modM)$$

Założenia:

- $Z_m = 0, 1, ..., M-1$
- Dziedzina $D(f) = Z_M^{\otimes k}$
- Przeciwdziedzina $D^{-1}(f) = Z_M$

Takie generatory są okresowe tzn.:

$$\exists N, r : \forall n \geq N | x_n = x_{n+jr} \text{ j = 1,2,..}$$

Przykład: Generator Fibonacciego

$$x_{n+1} = (x_n + x_{n-1})(modM)$$

- okres $\leqslant M^2$
- prosty
- (-) korelacje w ciągach generowanych liczb

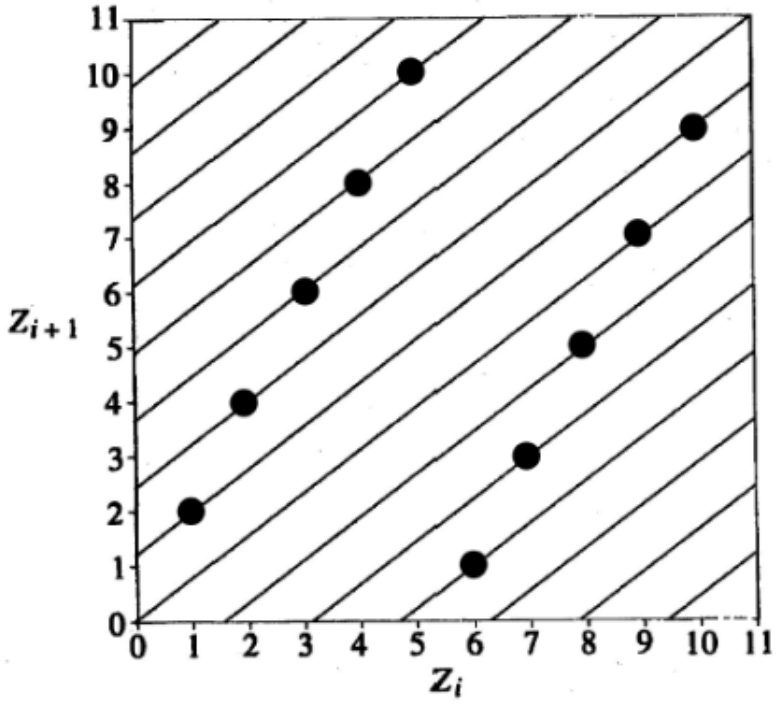
10.2 Omów wady i zalety generatorów liniowych kongruentnych

Zalety:

- mało obliczeń
- proste to skonstruowania
- szybkość

Wady:

- korelacje w sekwencji
- niższe bity są "mniej losowe niż wyższe"
- szybkość



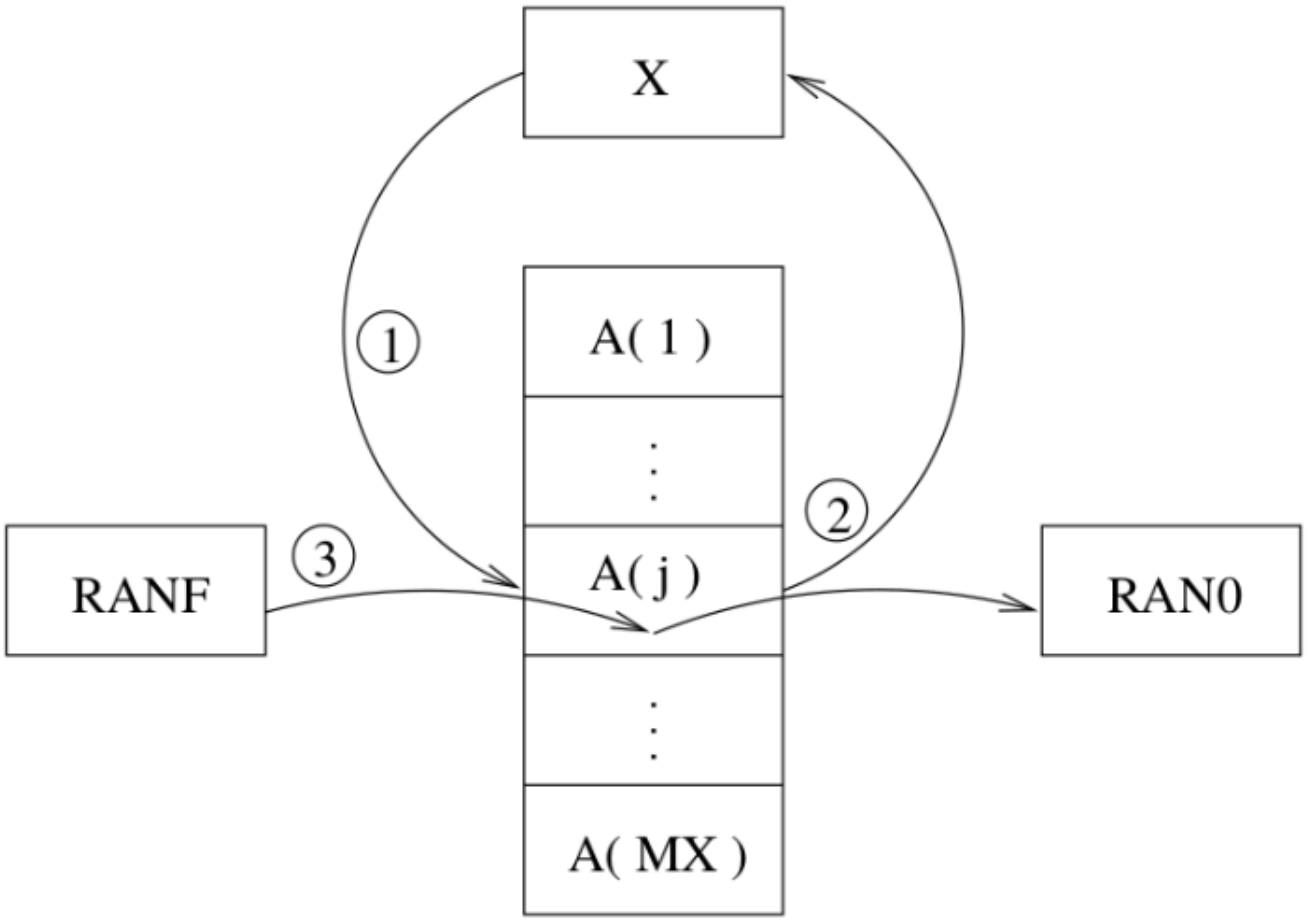
10.3 Omów wybrany sposób ulepszania jakości generatorów liczb pseudolosowych

Dobieranie odpowiednich parametrów generatora:

- a:
  - $a(mod 8) = 5$
  - $\frac{m}{100} < a < m - \sqrt{m}$
  - brak wzoru w zapisie binarnym
- c:
  - nieparzyste
  - $\frac{c}{m} \approx \frac{1}{2} - \frac{\sqrt{3}}{6}$

- c:
  - $m = 2^t$, t-liczba dostępnych bitów

Losowe potasowanie:



Rysunek 14.1: Idea ulepszonego gen. liczb losowych

0 Wypełniamy tablicę A i zmienną x liczbami losowymi

1 x to indeks j wskazujący element w A

2 A[j] wstawiamy w miejsce x i zwracamy jako liczbę losową

3 z generatora RANF losujemy brakującą liczbę w miejsce A[j]

10.4 Omów metodę odwróconej dystrybuanty: do czego służy, jak ją stosować, wady, zalety

Służą do tworzenia zmiennych losowych o konkretnych rozkładach

$$U \approx uniform(0,1); X = F^{-1}(U) \text{ma rozkład o dystrybuancie } F$$

$$P(X < x) = P(F^{-1}(U) < x) = P(U < x) = F(x)$$

zamieniamy na współżędne sferyczne:

Wady:

- w ogólności odwracanie jest trudne, wymaga kosztownych obliczeń
- w wielu przypadkach odwrócenie jest niemożliwe albo daje funkcję nieelementarną

Zalety:

- Dokładne
- W niektórych przypadkach bardzo proste
- potrzeba tylko jednej liczby z U do generowania

10.5 Omów metodę Boxa-Mullera: do czego służy, jak ją stosować i dlaczego

Służy do generowanie zmiennych z rozkładu normalnego:

Let $x_1, x_2 \in (-\infty, \infty)$

$$p(x_1, x_2) = e^{\frac{-x_1^2}{2}} * e^{\frac{-x_2^2}{2}} = e^{-\frac{x_1^2+x_1^2}{2}}$$

$$\begin{cases} r^2 = x_1^2 + x_2^2 \\ x_1 = rsin(\phi) \\ x_2 = rcos(\phi) \end{cases} \tag{2}$$

$$p(x, y)dxdy = p(r, \phi)rd\phi dr$$

$$e^{-\frac{r^2}{2}} rd(z = \frac{r^2}{2})$$

$$dz = rdr$$

$$e^{-z} d\phi dz \rightarrow \text{wykładniczy}$$

$$F^{-1}(w) = -\ln(1 - w) \rightarrow \text{odwrotna dystrybuanta}$$

$$\frac{r^2}{2} = z \implies r = \sqrt{2z}$$

Wady:

- potrzeba dwóch liczb do generowania
- kosztowne obliczeniowo

Zalety:

- generujemy od razu dwie liczby
- generowanie ϕ jest proste, wyciągamy je z U

10.6 Opisz dlaczego możemy wyznaczać całki metodami Monte Carlo

Całkowanie Monte Carlo jest zawsze możliwe, gdyż obliczanie całki zawsze można przedstawić jako zagadnienie obliczania wartości oczekiwanej pewnej zmiennej losowej ciągłej.

10.7 Opisz całkowanie Monte Carlo metodami: orzeł-reszka, podstawowa, średniej ważonej

1. **Metoda orzeł-reszka:**
   a) Najprostsza i najmniej skuteczna z metod Monte Carlo
   b) Szukamy całki zawierającej się w kwadracie $[0,1] \times [0,1]$:
   $$I = \int_0^1 f(x)dx \quad 0 \leq f(x) \leq 1$$
   c) Mamy zmienną losową (X, Y) o rozkładzie równomiernym na kwadracie - prawdopodobieństwo, że (X, Y) przyjmie wartość mniejszą od $f(x)$ jest równe polu pod wykresem funkcji, a więc jej całce
   d) Generujemy N punktów z rozkładem równomiernym, M z nich trafia w pole pod funkcją, oszacowanie całki to $\frac{M}{N}$:
   $$I \approx \frac{M}{N}$$
   e) Odchylenie majele wraz ze wzrostem wylosowanych liczb jak $\frac{1}{\sqrt{N}}$ - prosta metoda na zwiększenie precyzji.
   f) Wariancja:
   $$Var(I) = \frac{1}{n}(I - I^2)$$
2. **Metoda podstawowa:**
   a) Problem znalezienia całki z $f(x)$ przedstawia się jako problem znalezienia wartości oczekiwanej zmiennej losowej ($g(x)$ - gęstość zmiennej losowej):
   $$I = \int_a^b f(x)dx = \int_a^b \frac{f(x)}{g(x)} * g(x)dx = \int_a^b h(x)g(x)dx = E\{Y\}$$
   b) Dla uproszczenia I można bez straty ogólności przyjąć, że liczymy całkę od 0 do 1
   c) Generujemy N wartości x_i o rozkładzie równomiernym z (a, b), dla każdej obliczamy $f(x_i)$. Wartość oczekiwana to średnia arytmetyczna, całka to $f(X)$ pomnożona przez długość przedziału (przy uproszczeniu do $(0, 1)$ niepotrzebne):
   $$E\{f(x)\} = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$
   $$I \approx E\{f(x)\} * (b - a)$$
   d) Wariancja metody orzeł-reszka nigdy nie będzie mniejsza niż metody podstawowej, która wynosi:
   $$Var(I) = \frac{1}{N}\left[\int_a^b f^2(x)dx - I^2\right]$$
3. **Metoda średniej ważonej:**
   a) Idea - zmniejszenie wariancji samej funkcji $f(x)$.

b) Podobna do metody podstawowej, ale losuje się punkty dla rozkładu $g(x)$ podobnego kształtem do $f(x)$, $x \in [0,1]$ takigo, że:

1) $g(x) > 0$, $x \in [0,1]$

2) $\int_0^1 g(x)dx = 1$

3) $\frac{f(x)}{g(x)}$, $x \in (0,1)$ - iloraz taki musi być bardzo gładki, znacznie gładszy niż $f(x)$

4) $g(x)$ dana prostym analitycznym wzorem

5) Najlepiej, żeby dystrybuanta $G(x) = \int_0^x g(x)dx$ była odwracalna (a więc i ściśle rosnąca).

c) Działanie: losuje się $y_i \in (0,1)$ z rozkładem równomiernym, rozwiązuje się $G(x_i) = y_i$ dla znalezienia x_i, przybliża się wartość całki jako:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

d) Metoda ta daje lepsze wyniki od metody podstawowej, bo zmniejsza wariancję funkcji.

e) Problemy: mało funkcji jest odwracalnych analitycznie, ciężko tę metodę uogólnić na wiele wymiarów, jest niestabilna, bo iloraz $\frac{f(x)}{g(x)}$ może dla bardzo małych $g(x)$ dać ogromną wariancję.

10.8 Porównaj całkowanie numeryczne (Newtona-Cotesa, Gaussa) i całkowanie metodami Monte Carlo

1. **Determinizm** - całkowanie kwadraturami jest w pełni deterministyczne, podczas gdy metody Monte Carlo opierają się na niedeterministycznych liczbach pseudolosowych.
2. **Zależność od dostępnych generatorów liczb pseudolosowych** - dla kwadratur nie ma znaczenia, czy generatory są dostępne i jakiej są jakości, podczas gdy jakość metod Monte Carlo opiera się na dostępności wysokiej jakości generatorów.
3. **Złożoność obliczeniowa** - szczególnie dla całek wielowymiarowych o trudnych granicach lub bardzo dużych obszarów całkowania metody Monte Carlo są znacznie mniej wymagające obliczeniowo.
4. **Precyzja obliczeń** - dla małej liczby wymiarów (najlepiej 1) i funkcji dającej się precyzyjnie przybliżyć wielomianem kwadratury są precyzyjniejsze, w szczególności kwadratura Gaussa ma stopień dokładności $2n-1$ dla n punktów.

11 Metoda simulated annealing

11.1 Opisz i objaśnij metodę simulated annealing

Symulowane wyżarzanie (simulated annealing) - algorytm optymalizacji kombinatorycznej pozwalający znaleźć przybliżone rozwiązania problemów dyskretnych. Jest on oparty o analogie do fizyki statystycznej i metalurgii.
Ważnym elementem algorytmu jest akceptowanie gorszych rozwiązań z pewnym prawdopodobieństwem (lepsze są akceptowane zawsze), co pozwala uciec z minimów lokalnych. Prawdopodobieństwo zaakceptowania gorszego rozwiązania zależy od temperatury (ilości wykonanych kroków algorytmu): na początku jest wysokie (działanie podobne do losowego wędrowania po przestrzeni przeszukiwań), potem coraz niższe, aż na koniec gorsze rozwiązania prawie nigdy nie są akceptowane (działanie podobne do metody gradientu). Trzeba też przy tym zawsze pamiętać najlepsze rozwiązanie.

Jest to metaheurystyka (rodzina heurystyk), więc wiele elementów algorytmu można zmieniać i dostosowywać do aktualnego problemu:

1. Funkcja spadku temperatury (*temperature cooldown schedule*) - często potęgowa, np. $T_{i+1} = 0.9 * T_i$, generuje temperatury $T_0, T_1, ..., T_n$
2. Liczba iteracji algorytmu - musi on działać odpowiednio długo, aby mógł się schłodzić
3. Sposób generowania kolejnej konfiguracji (*neighbor state*) - zależny głównie od konkretnego problemu
4. Funkcja kosztu (energii) - minimalizowana funkcja, to według niej liczy się zmianę energii w nowym stanie ΔE
5. Prawdopodobieństwo zaakceptowania gorszej konfiguracji - zwykle akceptuje się gorsze rozwiązanie, gdy wylosowana liczba $R \in (0, 1)$ jest większa od rozkładu Boltzmanna $e^{-\frac{\Delta E}{k*T}}$ (schemat Metropolisa) - funkcja taka musi zapewniać malejące prawdopodobieństwo akceptacji wraz ze spadkiem temperatury i zależność od różnicy energii ΔE
6. Podgrzewanie układu - jeżeli np. przez określoną liczbę ostatnich iteracji nie znaleziono lepszego rozwiązania, to można zwiększyć temperaturę, żeby pomóc algorytmowi uciec z nieoptymalnego minimum lokalnego.

Zastosowania metody:

- problem komiwojażera
- optymalizacja rozłożenia układów elektronicznych na chipach
- optymalizacja rozłożenia przewodów w układach VLSI
- algorytm Metropolisa optymalizacji kodu komputerowego

Pseudokod:

```
i = 1
T = temperature(1)
C = random_state()  // generate random first configuration
best_C = C
until i == number_of_iterations:
  new_C = neighbor_state(C)  // generate neighbor state, new solution

  delta_E = energy(new_C) - energy(C)  // change in energy for new state

  if delta_E < 0:  // new_C has lower energy -> is better
    C = new_C
    if energy(new_C) < energy(best_C):  // remembering best solution
      best_C = new_C
  else:
    R = random(0,1)
    if exp(-delta_E/T) > R:  // accepting worse solution with some probability
      C = new_C

  i += 1
  T = temperature(i)
```

12 FFT

12.1 Objaśnij przydatność transformat Fouriera, podaj ich główne rodzaje

Zastosowanie transformat Fouriera:

a) metody spektralne:
   - fizyka, chemia: badanie właściwości atomów, cząsteczek, itp.
   - na podstawie widma promieniowania elektromagnetycznego

b) algorytmy numeryczne:
   - równania różniczkowe
   - analiza: badanie jakości algorytmów (np. dla MES)

c) cyfrowe przetwarzanie sygnału:
   - badanie składowych harmonicznych
   - filtracja obrazów i dźwięku
   - kompresja

Rodzaje:

a) transformata Fouriera (Fourier Transform)

b) szereg Fouriera (Fourier Series) - stosowany w transformacie odwrotnej

c) transformata dyskretna (Discrete-time Fourier Transform), może być też dwuwymiarowa

d) skończnona transformata Fouriera (finite FT)

12.2 Objaśnij, na czym polega interpolacja trygonometryczna, kiedy ją warto stosować, jaki jest jej związek z dyskretną transformatą Fouriera

- Wielomiany algebraiczne nie są dobre do opisu zjawisk okresowych
- Rozwiązanie: interpolacja wielomianami opartymi o funkcje trygonometryczne

funkcje okresowe o okresie L spełniają zależność g(y + L) = g(y) funkcje trygonometryczne: okresem jest 2π, po przeskalowaniu:

$$x = \frac{2\pi}{L} * y, f(x) = g(\frac{x * L}{2\pi})$$

okresem g(y) jest L, okresem f(x) jest 2π

$$f(x + 2\pi) = g(\frac{(x + 2\pi) * L}{2\pi}) = g(\frac{x * L}{2\pi} + L) = g(\frac{x * L}{2\pi} = f(x)$$

szukamy wielomianu trygonometrycznego

$$t_{n-1}(x) = \sum_{j=0}^{n-1} c_j * (e^{ix})^j = \sum_{j=0}^{n-1} c_j * e^{ijx}$$

gdzie (przypomnienie - wzór Eulera)

$$e^{ix} = cos(x) + isin(x)$$

który w n punktach $x_k \in (0, 2\pi]$ przyjmuje te same wartości, co interpolowana funkcja
$t_{n-1}(x_k) = f(x_k), k = 0, 1, ..., n - 1$ W praktyce - ważny przypadek szczególny:
n węzłów równoodległych, $x_k = \frac{2\pi}{n} * k, k = 0, 1, ..., n - 1$
funkcje $e^{ijx}, j = 0, 1, ..., n - 1$ tworzą układ ortogonalny w sensie iloczynu skalarnego zdefiniowanego:

$$\langle f|g\rangle = \sum_{k=0}^{n-1} f(x_k) * g^*(x_k), x_k = \frac{2\pi}{n} * k, k = 0, 1, ..., n - 1$$

A dokładniej:

$$\langle e^{ijx}|e^{ilx}\rangle = \sum_{k=0}^{n-1} e^{ijx_k} * e^{-ilx_k} = n * \delta_{j,l} = \begin{cases} 0, j \neq l \\ n, j = l \end{cases}$$

Dla

$$x_k = \frac{2\pi}{n} * k, k = 0, 1, ..., n-1$$

Gdzie $\delta_{j,l}$ to delta Kroneckera

$$\delta_{j,l} = \begin{cases} 0, j \neq l \\ 1, j = l \end{cases}$$

Jakie powinny być współczynniki wielomianu interpolacyjnego c_j?

$$c_j = \frac{1}{n} \sum_{k=0}^{n-1} f(x_k) * e^{-ij*x_k}, k = 0, 1, ..., n-1$$

Dwa etapy obliczeń:
analiza Fouriera:
dla danych liczb zespolonych $f(x_k), k = 0, 1, ..., n-1$ szukamy $c_j, j = 0, 1, ..., n-1$
synteza Fouriera:
mając liczby $c_j, j = 0, 1, ..., n-1$ szukamy

$$f(x) = \sum_{j=0}^{n-1} c_j * e^{ij\frac{2\pi}{n}*k}, k = 0, 1, ..., n-1$$

12.3 Opisz własności funkcji stosowanych w interpolacji trygonometrycznej - w szczególności ortogonalność i jak z niej korzystamy

Dzięki ortogonalności można uprościć obliczenia, elementy w sumie = 0 poza elementami o tych samych indeksach.
delta Kroneckera
okresowość funkcji
Więcej w poprzednim punkcie.

12.4 Na czym polega FFT - szybka transformata Fouriera: przedstaw algorytm, podaj złożoność obliczeniową, porównaj z algorytmem klasycznym

Algorytm FFT (algorytm Cooleya-Tukeya) - sposób szybkiego obliczania dyskretnej transformaty Fouriera (DFT).
Dane:
$f(x_k)$, $x_k = k * \frac{2\pi}{n}$, $k = 0, 1, ..., n-1$, $n = 2^m$
Algorytm ma wyznaczyć:

$$c_j = \frac{1}{n} \sum_{k=0}^{n-1} f(x_k) * e^{kj*\left(-i\frac{2\pi}{n}\right)} \quad j = 0, 1, ..., n-1$$

$$a_k = \frac{1}{n} * f(x_k) \qquad \omega = e^{-i\frac{2\pi}{n}}$$

$$c_j = \sum_{k=0}^{n-1} a_k \omega^{jk} \quad j = 0, 1, ..., n-1$$

Istotą FFT jest rozbicie sumy na indeksy parzyste i nieparzyste i wykorzystanie techniki *divide-and-conquer*. Oznaczmy $0 \leq k_1 \leq \frac{n}{2} - 1$:

$$c_j = \sum_{k_1=0}^{\frac{n}{2}-1} a_{2k_1}\omega^{2jk_1} + \sum_{k_1=0}^{\frac{n}{2}-1} a_{2k_1+1} * \omega^{2jk_1} * \omega^j$$

Jednocześnie $0 \leq j \leq n-1$. Dzieląc na 2 zbiory oznaczamy $j = \frac{n}{2} * I + j_1$, $0 \leq j_1 \leq \frac{n}{2} - 1$ (dzielenie przez $\frac{n}{2}$). Jednocześnie $\omega^{2jk_1} = \omega^{2j_1k_1}$, a więc:

Dla 0<=j<=n/2-1

$$c_j = \underbrace{\sum_{k_1=0}^{\frac{n}{2}-1} a_{2k_1} * \omega^{2j_1k_1}}_{\varphi(j_1)} + \omega^j \underbrace{\sum_{k_1=0}^{\frac{n}{2}-1} a_{2k_1+1} * \omega^{2j_1k_1}}_{\psi(j_1)} \quad 0 \leq j_1 \leq \frac{n}{2} - 1$$

Dla n/2 <= j <= n-1

$$c_j = \underbrace{\sum_{k_1=0}^{\frac{n}{2}-1} a_{2k_1} * \omega^{2j_1k_1}}_{\varphi(j_1)} - \omega^j \underbrace{\sum_{k_1=0}^{\frac{n}{2}-1} a_{2k_1+1} * \omega^{2j_1k_1}}_{\psi(j_1)} \quad 0 \leq j_1 \leq \frac{n}{2} - 1$$

W powyższym wzorze transformata dla n punktów została rozbita na sumę 2 tranformat w $\frac{n}{2}$ punktach każda. Dokonując dalszych takich podziałów, otrzymujemy rekurencyjny algorytm *divide-and-conquer*, co oznacza złożoność $O(nlog_2(n))$. Algorytm klasyczny mnoży macierz Fouriera F_n i dany wektor f, więc ma złożoność $O(n^2)$. W praktyce dla problemu rzędu $n = 10^6$ algorytm FFT zajmuje ok. 0,1 sekundy, a klasyczny ok. 1,5 godziny.
Pseudokod:

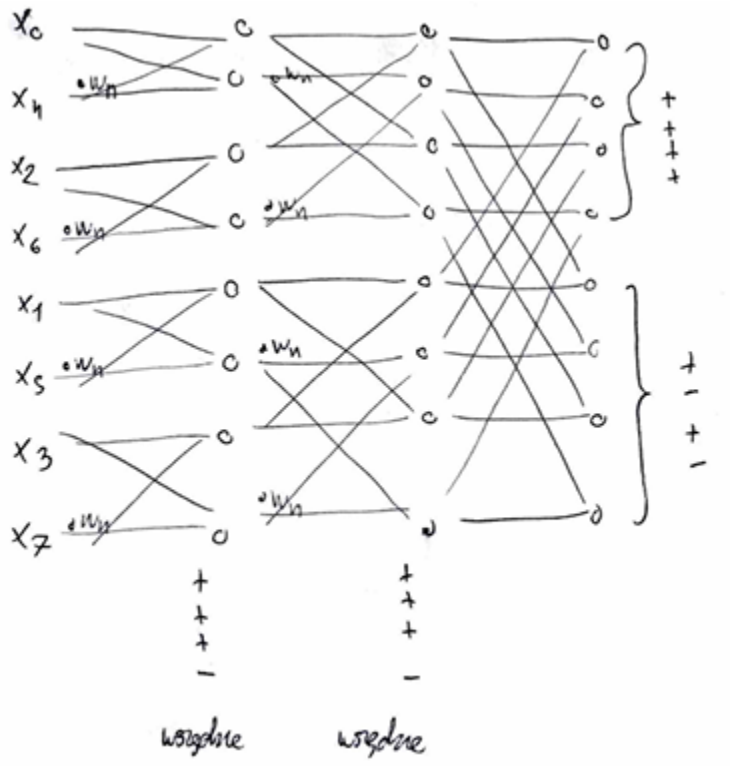
```
n = size(x)
if n == 1:
  return x
else:
  even, odd = split(x)  // split x for x_2n and x_(2n+1)
  even_FFT = FFT(even)
  odd_FFT = E(n/2) * FFT(odd)  // E - omega multiplier for odd elements

  return (even_FFT + odd_FFT) + (even_FFT - odd_FFT) // adding arrays
```

12.5 Pokaż jak działa algorytm FFT na przykładzie wyznaczania transformaty dla 8 punktów

Dane są wartości funkcji postaci $x = [x_0, x_1, ..., x_7]$. Szukamy n współczynników postaci:

$$c_j = \frac{1}{n} \sum_{k=0}^{n-1} x_k * e^{kj*\left(-i\frac{2\pi}{n}\right)} \quad j - 0, 1, ..., n-1$$

Punkty trzeba ułożyć w odpowiedniej kolejności, a następnie zastosować transformaty Fouriera rekurencyjnie dla 4 i 2 elementów i połączyć wyniki. Na poniższym rysunku dla czytelności zapisano transformaty w kolejności finalnego obliczania i łączenia wyników.



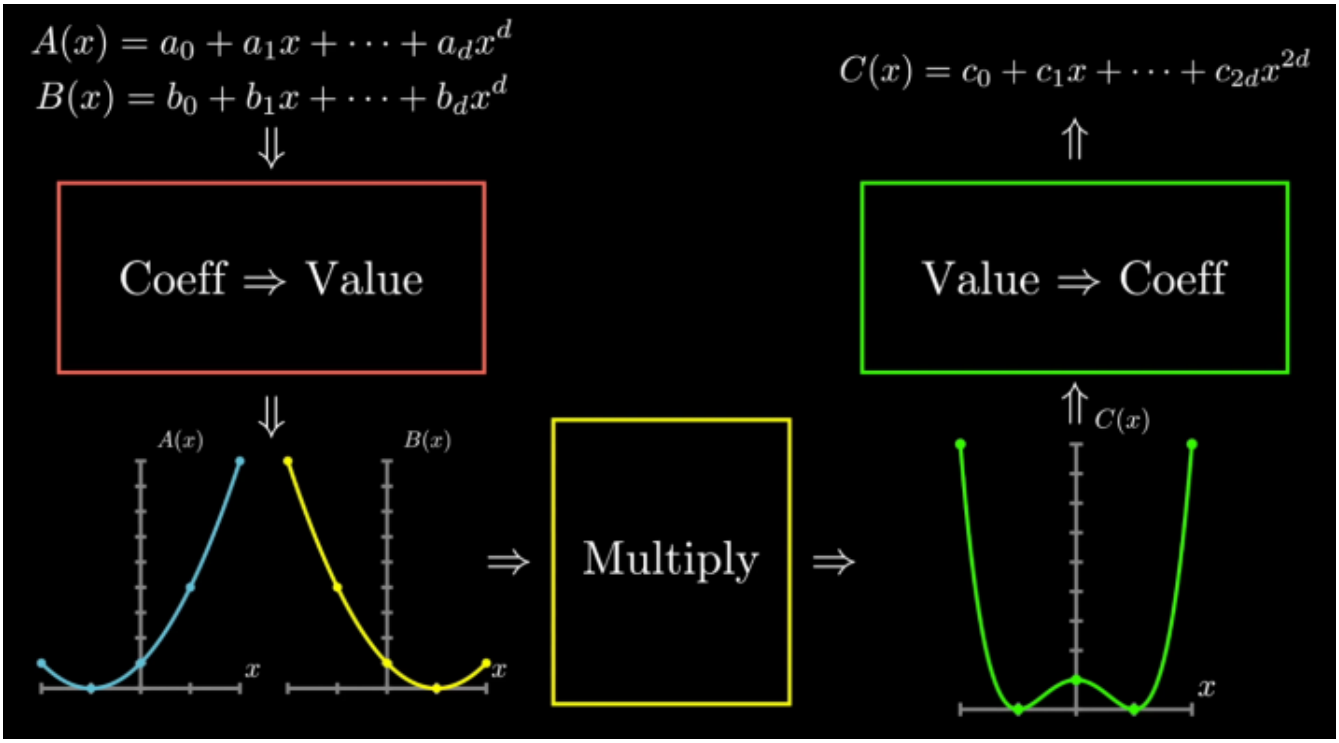
Rysunek 13: Transformata dla 8 punktów

12.6 Opis zasadę dziel i zwyciężaj stosowaną w projektowaniu algorytmów na przykładzie algorytmu FFT

Metoda "dziel i zwyciężaj"polega na rozbiciu problemu na prostsze podproblemy w sposób rekurencyjny tak długo, aż fragmenty staną się wystarczająco proste do bezpośredniego rozwiązania.
Rozwiązanie całego problemu otrzymuje się scalając rozwiązania podproblemów.

Algorytm FFT polega na podzieleniu transformaty wejściowej na dwie o połowę mniejsze transformaty rekurencyjnie.
Otrzymuje się w ten sposób algorytm o złożoności $O(nlog_2(n))$ zamiast $O(n^2)$ (algorytm klasyczny).

12.7 Opisz zastosowanie FFT do algorytmu szybkiego mnożenia wielomianów



Rysunek 14: Mnożenie wielomianów (Coeff $\implies$ Value (FT), Value $\implies$ Coeff (reverse FT))

- Dane: $P(x) = \sum_{i=0}^{n-1} a_i x^i$ $Q(x) = \sum_{i=0}^{n-1} b_i x^i$
- Jeśli $[a_i]$ i $[b_i]$ to wektory, to $[c_i] \implies \sum_{j=0}^{i} a_j b_{i-j}$ jest ich splotem
- $W(x) = P(x)Q(x) = \sum_{i=0}^{2n-2} c_i x^i$
- zamiast wyliczać współczynniki wprost: transformata → iloczyn → transformata JAKAŚ

13 Minimalizacja

13.1 Omów metody minimalizacji funkcji jednej zmiennej: przeglądanie siatki, metoda złotego podziału, metoda kwadratowej interpolacji, metoda prób i błędów

- **przeglądanie siatki**
  Realizacja:
  - Przegląd wszystkich elementów iloczynu kartezjańskiego podzbiorów parametrów.
  - Wybór wartości najmniejszej.

  Zalety:
  - Absolutna prostota, problem typu embarrassingly parallel.
  - Bezwzględna zbieżność.
  - Brak ”czułości” na szczegółowe zachowanie się F(x).

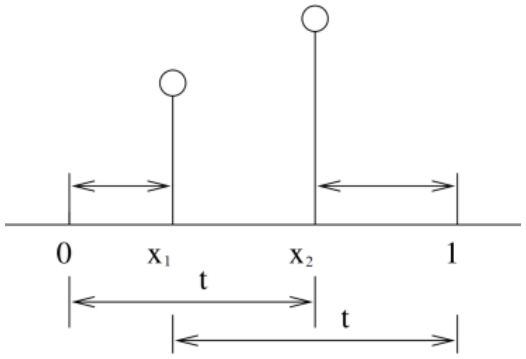
  Wady:

- Nie może być stosowana dla przedziału nieskończonego.
- Nieefektywna, nie "uczy się" własności funkcji.
- Problem z doborem rozmiaru siatki, tak aby nie zgubić szukanej wartości.

- **metoda złotego podziału**
  Realizacja:
  $t \in (0,1)$ - współczynnik redukcji po każdym etapie

  1. Obliczamy nową długość przedziałów $d = t(b-a)$
  2. $x_L = b - d, x_R = a + d$
  3. Jeżeli $f(x_L) > f(x_R) \implies x_0 \in [x_L, b], a := x_L$
     Jeżeli $f(x_L) < f(x_R) \implies x_0 \in [a, x_R], b := x_R$
  4. Procedurę powtarzamy, aż do osiągnięcia żądanej zbieżności.

  Dobieramy współczynnik t, aby w kolejnym kroku wykorzystać jedną z dwóch próbek: $f(x_L)$ lub $f(x_R)$.
  Dla $[0,1]$ – długość przedziału po pierwszym etapie: $d1 = t$
  Dla przykładowego rozmieszczenie punktów w kolejnym kroku rozwiązania szukamy w $(0, x_2)$:

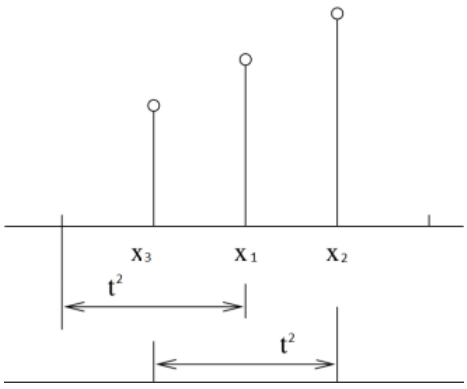


Rysunek 15: Krok 1

Długość przedziału po drugim etapie to t^2.
Jednocześnie długość ta jest wyznaczona przez pozycję z wnętrza przedziału na poprzednim etapie (tutaj pozycję x_1)

$$t^2 = 1 - t \implies \frac{\sqrt{5}-1}{2} \approx 0.616 \rightarrow \text{złoty podział}$$

Dla zadanej liczby kroków - optymalna.



Rysunek 16: Krok 2

- **metoda kwadratowej interpolacji**
  Założenia:
  - Wykres funkcji f(x) jest parabolą.

  Realizacja:

  1. Interpolujemy f(x) funkcją kwadratową w 3 punktach: x_1, x_2, x_3.
  2. Ekstremum f(x) to ekstremum paraboli przechodzącej przez x_1, x_2, x_3, znajduje się w punkcie x_4:

$$x_4 = -\frac{\frac{f_1*(x_2+x_3)}{(x_1-x_2)*(x_1-x_3)} + \frac{f_2*(x_1+x_3)}{(x_2-x_1)*(x_2-x_3)} + \frac{f_3*(x_1+x_2)}{(x_3-x_1)*(x_3-x_2)}}{2*[\frac{f_1}{(x_1-x_2)*(x_1-x_3)} + \frac{f_2}{(x_2-x_1)*(x_2-x_3)} + \frac{f_3}{(x_3-x_1)*(x_3-x_2)}]}$$

  3. x_4 - zastępuje jeden z x_1, x_2, x_3, wyznaczamy nowy x_4.
  4. Procedurę kończymy gdy wartość $f(x_4)$ jest bliska $f(x_3)$ z zadaną dokładnością.

  Problemy:

  1. Na każdym kroku x_1, x_2, x_3 mogą wyznaczać max, a nie min powodując rozbieżność.
  2. Gdy x_1, x_2, x_3 leżą prawie na prostej, otrzymujemy duży krok:
     - Trudności numeryczne
     - Rozbieżność
  3. Który z poprzednich punktów odrzucić?
  4. Możliwe oscylacje wokół minimum, zamiast zbieżności.

  Zastosowanie:
  W ostatniej fazie minimalizacji funkcji f.
  (Funkcje fizyczne są zwykle paraboliczne w pobliżu minimum.)

- **metoda prób i błędów**
  Założenia:
  Procedura składa się z dwóch części:
  - Iteracyjne zawężenie przedziału – podobne do grid search.
  - Kwadratowa interpolacja na otrzymanym przedziale.

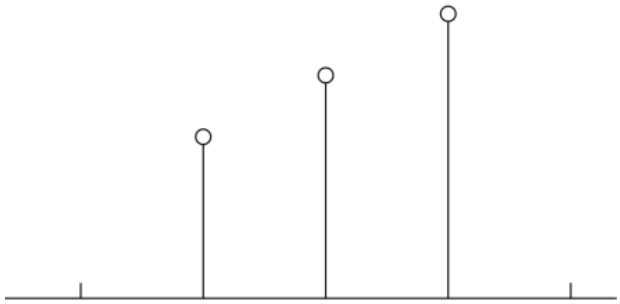
  x_0 - start point, d - initial step size

$$\text{Gdy } f(x_0 + d) < f(x_0) - \text{sukces:}$$

$$x_0 \to x_0 + d,$$
$$d \to \alpha * d, \alpha - \text{ expansion factor } (\alpha > 1)$$

$$\text{Gdy } f(x_0 + d) > f(x_0) - \text{niepowodzenie:}$$
$$d \to -\beta * d, \beta - \text{contraction factor } (\beta < 1)$$

α oraz β - ustalamy arbitralnie
Procedurę powtarzamy do zbieżności, tj. $|f(x_0 + d) - f(x_0)| < e$.
Minimum jest wstępnie zlokalizowane (bracketed), gdy po sukcesie - niepowodzenie, wtedy mamy trzy punkty typu:



Są one punktem startowym interpolacji kwadratowej.
Uniwersalna, efektywna metoda 1-D dla ogólnych funkcji.

13.2 Omów metody bezgradientowe minimalizacji funkcji: metoda Powella, metoda Simpleksów

- **metoda Powella**
  1. Wykonaj jeden pełny cykl minimalizacji wzgl. kolejno wszystkich parametrów (współrzędnych),
  2. Zmień osie układu współrzędnych - nowy układ ortogonalny: jedna z osi od punktu początkowego do końcowego w ostatnim cyklu minimalizacji,
  3. Wykonaj kolejny cykl w nowym układzie współrzędnych.

  Metoda mało efektywna dla dużego n

- **metoda Simpleksów** Simplex - najprostsza n-wymiarowa figura określona przez (n + 1) wierzchołków (vertex).
  Algorytm:

  1. Wybieramy (losowo, min 1-D) 3 punkty P_1, P_2, P_3 i wyznaczamy spośród nich: P_H - gdzie F największa (highest) P_L - gdzie F najmniejsza (lowest)
  2. Wyznaczamy "środek masy" wszystkich punktów z pominięciem P_H

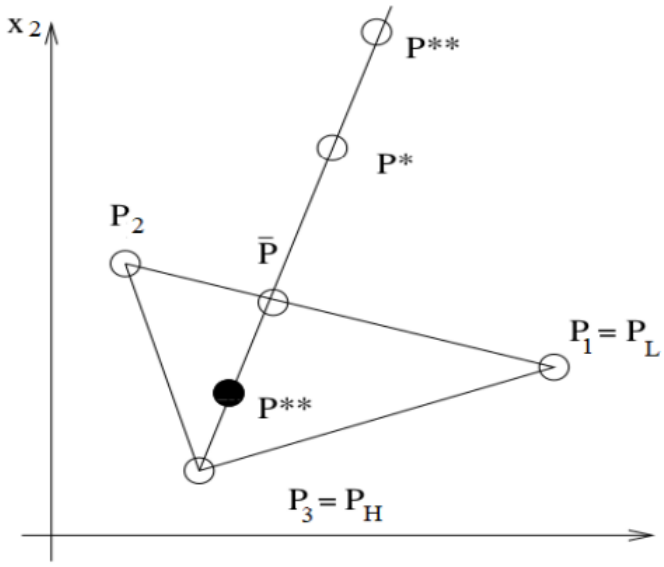
  $$\overline{P} = \frac{1}{n}[\sum_{i=1}^{n+1}(P_i - P_H)]$$

  3. Obliczamy odbicie P_H wzgl. $\overline{P}$: $P^* = \overline{P} + (\overline{P} - P_H)$, jeżeli:
     $F(P^*) < F(P_L) \implies$ nowy $P^{**} = \overline{P} + 2 * (\overline{P} - P_H)$
     $F(P^*) > F(P_L) \implies$ nowy $P^{**} = \overline{P} - 1/2 * (\overline{P} - P_H)$

4. Punkt P_H zastępujemy przez najlepszy z P^* i P^{**} Jeżeli żaden z nowych punktów nie jest lepszy od P_H, tworzymy simplex oparty o P_L w wymiarach 0.5 poprzednie.

Modyfikacje:

- Inne współczynniki $\neq 2$ oraz $\neq \frac{1}{2}$
- Interpolacja kwadratowa wzdłuż prostej $(P_H, \overline{P})$



Rysunek 17: Metoda Simpleksów

Uwaga:
Nowy punkt nie może być zbyt blisko $\overline{P}$, bo to grozi redukcją (bez powrotu) simpleksów w n do hiperpłaszczyzny. **Zalety:**

- nieczuła na płytkie minima (pochodzenia: zaokrąglenia, statystyka ...),
- mała ilość obliczeń funkcji F(X) w każdym kroku,
- największe możliwe kroki,
- rozsądny kierunek poszukiwań,
- bezpieczna i szybka daleko od minimum

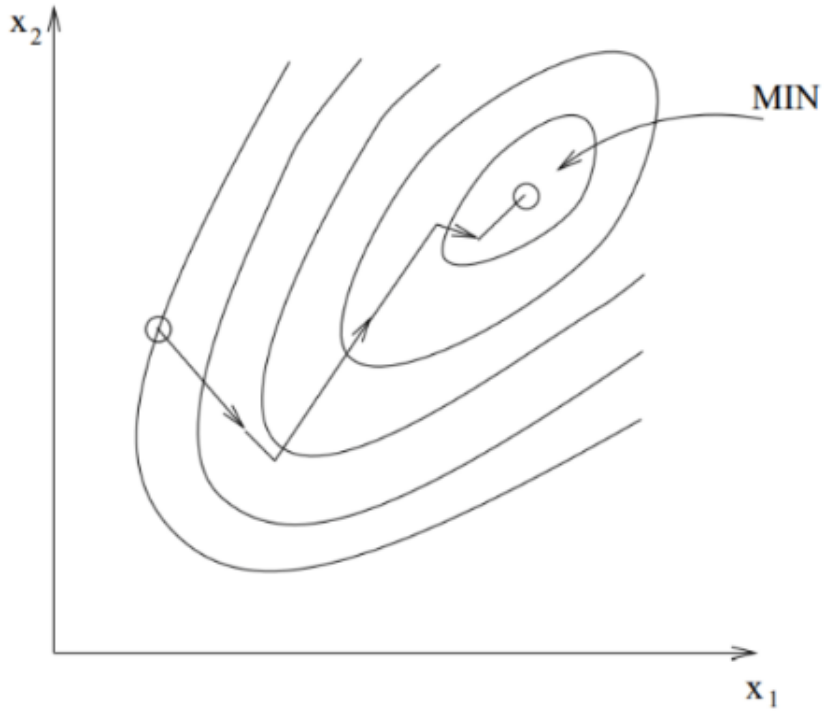
Zbieżność:

$$EDM = F(P_H) - F(P_L) < e$$

Gdzie EDM - estimated distance to minimum

13.3 Omów metody gradientowe minimalizacji funkcji: metoda prostego gradientu, metoda maksymalnego spadku, metoda Newtona, metoda gradientów sprzężonych

Metoda maksymalnego spadku



Rysunek 18: Metoda Maksymalnego Spadku

- podążanie w kierunku wyznaczonym przez $-\vec{g}$ ($g_i = \frac{\partial F}{\partial x_i}$)
- w tym kierunku funkcja maleje najszybciej (nierówność Cauchy'ego-Schwarza)
- seria minimalizacji 1-D wzdłuż kierunku największego spadku

$$\vec{x}_{k+1} = \vec{x}_k - \alpha\vec{g}|_{x_k}\alpha_k \text{ dobrane tak, ze:}$$

$$f(\vec{k} - \alpha_k\vec{g}|_{xk}) = min_{\alpha>0}(\vec{x}_k - \alpha_k\vec{g}|_{xk})$$

- iteracyjna, bo gradient nie jest stały
- $g_{k+1} \perp g_k$ metoda prowadzi do żygzakowania"

Metoda Newtona:

Ogólna funkcja kwadratowa jest określana przez:

- wartość
- pierwsze pochodne
- drugie pochodne

$F(\vec{x})$ rozwijamy w Taylora, pomijając dallsze wyrazy:

$$F(\vec{x}) = F(\vec{x}_0) + \vec{g}^T * (\vec{x} - \vec{x}_0) + \frac{1}{2}(\vec{x} - \vec{x}_0)^T * H * (\vec{x} - \vec{x}_0)$$

i szukamy minimum takiej funkcji kwadratowej Wzór na minimum:

$$\vec{x}_{i+1} = \vec{x} - H^{-1}|_{\vec{x}_i} * \vec{g}|_{\vec{x}_i} = \vec{x}_i - V * \vec{g}$$

Wady:

- moze byc niestabilna
- rozbieżna, gdy V nie jest dodatnio określona ($\neg\forall x | x^T V x > 0$)
- to samo co w interpolacji kwadratowej

Zalety:

- krok nie jest dowolny, lecz określony przez metodę
- kierunek != wartość gradientu, tylko brana pod uwagę korelacja parametrów (pamiętane w macierzy V)

Zastosowania:

- blisko minumum
- gdy funkcja jest określona formą kwadratową

Metoda Gradientów Sprzężonych:

Wektory $\vec{d_1}$ $\vec{d_2}$ są sprzężone ze względu na dodatnio określoną macierz, jeżeli:

$$\vec{d_i}^T A \vec{d_j} = 0 \quad \text{dla} \quad i \neq j$$

$$\text{gdy } A = I, \vec{d_i} \rightarrow \text{ ortogonalne}$$

(sprzężenie - uogólnienie ortogonalności) **n sprzężonych wektorów rozpina n-D przestrzeń**
A - nie określa jednoznacznie zbioru wektorów sprzężonych

Jak wyznaczyć kierunki sprzężone bez znajomości H?

Metoda wykorzystuje tylko 1-sze pochodne.
Jeśli $F(\vec{x})$ i $\vec{g}(\vec{x})$ wyznaczone w $\vec{x}_0$ i $\vec{x}_1$ z nich:

$$\vec{\Delta x} = \vec{x}_1 - \vec{x}_0$$
$$\vec{\Delta g} = \vec{g}_1 - \vec{g}_0$$

to dla $F(\vec{x})$ kwadratowej, z hesjanem H: $\vec{\Delta g} = H * \vec{\Delta x}$
zatem dowolny $\vec{d_1}$ sprzężony do $\vec{\Delta x}$ będzie $\perp \vec{\Delta g}$:

$$\vec{d_1}^T H \vec{\Delta x} = \vec{d_1}^T * \vec{\Delta g} = 0 (*)$$

Pierwszy kierunek: $\vec{d_0} = 0\vec{g_0}$
znajdujemy minimum $\vec{x}_1$ wzdłuż $\vec{d_0}$ według $\vec{x}_1 = \vec{x}_0 + \alpha \vec{d_0}$
a następnie gradient w tym punkcie (czyli $\vec{g}_1$)
Drugi kierunek tworzymy jako liniową kombinacją znanych kierunków: $\vec{d_1} = -\vec{g}_1 + b * \vec{d_0}$

Jak ustalić b?:

Warunek sprzężenia:

$$\vec{d_1}^T * H * \vec{d_0} = 0 \text{ czyli:}$$

$$\vec{d_1}^T * H * \frac{1}{\alpha}(\vec{x}_1 - \vec{x}_0) = 0$$

bo $\vec{d_1}^T * (\vec{g}_1 - \vec{g}_0) = 0$, czuli:

$$(-\vec{g}_1 + b * \vec{d_0})^T * H * \vec{d_0} = (-\vec{g}_1 + b * \vec{g}_0)^T * (-\vec{g}_1 - \vec{g}_0) = 0$$

$\vec{x}_1$ - jest znalezionym minimum wzdłuż $-\vec{g}_0$

$\implies$ kolejny $\vec{g}_1$, uzyskany w p.$\vec{x}_1$, więc jest $\perp$ do $\vec{g}_0 \implies \vec{g_1}^T * \vec{g}_0 = 0$

$\implies b = \frac{\vec{g_1}^T * \vec{g_1}}{\vec{g_0}^T * \vec{g_0}}$, czyli nowy sprzężony kierunek $\vec{d_1} = -\vec{g_1} + \frac{\vec{g_1}^T * \vec{g_1}}{\vec{g_0}^T * \vec{g_0}} * \vec{d_0}$

Ten proces kontynuujemy generując n kierunków wzajemnie sprzężonych do użycia dla kolejnych kroków minimalizacji

$$\vec{d_{i+1}} = -\vec{d_{i+1}} + \frac{\vec{g_{i+1}}^T * \vec{g_{i+1}}}{\vec{g_i}^T * \vec{g_i}} * \vec{d_i}$$