Sprawozdanie do zadania **P2.03**

Maurycy Borkowski

13.12.2020

Przedstawienie problemu

W zadaniu mamy przedstawić algorytm konstrukcji wielomianu w możliwie najniższego stopnia, który dla $\varepsilon > 0$ spełnia nierówność:

$$\int_0^1 x[f(x) - w(x)]^2 dx < \varepsilon \tag{1}$$

Szukamy więc wielomianu optymalnego w^* w sensie aproksymacji średnio kwadratowej dla funkcji wagowej $p(x) = x$. Najniższego stopnia żeby spełniony był warunek (1).

Mamy zdefiniowany iloczyn skalarny:

$$\langle f, g \rangle = \int_0^1 p(x)f(x)g(x)dx = \int_0^1 xf(x)g(x)dx \tag{2}$$

Szukany w^* ma spełniać:

$$\|f - w^*\|^2 = \langle f - w^*, f - w^* \rangle = \int_0^1 x[f(x) - w^*(x)]^2 dx < \varepsilon \tag{3}$$

Szukanie wielomianu optymalengo

Korzystamy z faktu (**M7.5**):

$$w_n^* \text{ jest optymalny } \iff \langle f - w_n^*, w_i \rangle = 0 \iff \langle f, w_i \rangle = \langle w_n^*, w_i \rangle \tag{4}$$

$w_n^* \in \Pi_n$ więc możemy go rozpisać jednoznacznie w bazie: $\{1, x, x^2, \ldots, x^n\}$:

$$w_n^* = \sum_{i=0}^{n} \alpha_i \cdot x^i \tag{5}$$

Z (4) i (5):

$$\langle f, w_i \rangle = \langle w_n^*, w_i \rangle = \langle \sum_{i=0}^{n} \alpha_i \cdot x^i, w_i \rangle \tag{6}$$

teraz z liniowości iloczynu skalarnego:

$$\langle f, w_i \rangle = \langle w_n^*, w_i \rangle = \sum_{i=0}^{n} \alpha_i \cdot \langle x^i, w_i \rangle \tag{7}$$

Z powyższych rozważań wynika, że aby wyznaczyć wielomian optymalny w_n^* wystarczy, że wyznaczymy $\alpha_0, \ldots, \alpha_n$ spełniające (7), rozpiszmy to w bardziej intuicyjnej postaci macierzowej:

$$\begin{bmatrix} \langle 1,1 \rangle & \langle 1,x \rangle & \ldots & \langle 1,x^n \rangle \\ \langle x,1 \rangle & \langle x,x \rangle & \ldots & \langle x,x^n \rangle \\ \vdots & \vdots & & \vdots \\ \langle x^n,1 \rangle & \langle x^n,x \rangle & \ldots & \langle x^n,x^n \rangle \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \langle f,1 \rangle \\ \langle f,x \rangle \\ \vdots \\ \langle f,x^n \rangle \end{bmatrix} \tag{8}$$

Pozostaje jeszcze kwestia uzupełnienia macierzy iloczynów skalarnych $n \times n$. Dobór bazy Π_n był nieprzypadkowy, $i,j-ty$ wyraz tej macierzy możemy obliczać w czasie stałym:

$$\langle x^i, x^j \rangle = \int_0^1 x \cdot x^i \cdot x^j dx = \int_0^1 x^{i+j+1} dx = \frac{x^{i+j+2}}{i+j+2}\Big|_0^1 = \frac{1}{i+j+2} \tag{9}$$

Podsumowując, aby znaleźć wielomian optymalny w_n^* wystarczy rozwiązać układ równań zadany (8). Będziemy go rozwiązywać za pomocą macierzy odwrotnej znalezionej za pomocą gotowych funkcji wbudowanych (nie poznawaliśmy jeszcze metod do rozwiązywania tego typu problemów).

Musimy rozwiązać jeszcze dwa podproblemy:

- Liczenie nietrywialnych iloczynów skalarnych, $\langle f, x^i \rangle$ z (8)
- Wybór optymalnego n

Obliczanie $\langle f, g \rangle$

Aby móc obliczać wyrazy wektora (prawa strona) z (8) typu $\langle f, x^i \rangle$ oraz żeby liczyć błąd (1) musimy być w stanie liczyć wartości całek.

Do tego skorzystamy z Metody Romberga.

Polega ona na wyznaczeniu tablicy przybliżeń wartości całek, $R_{i,j}$ w następujący sposób:

$R_{0,i}$ jest wzorem z metody trapezów (z podziałem odcinka na 2^i części), po obliczeniu pierwszej kolumny tzw. tablicy Romberga, kolejne kolumny obliczane są rekurencyjnie, otrzymując coraz lepsze przybliżenie funkcji:

$$\begin{cases} R_{0,i} & : R_{2^i} = h_i \cdot \sum_{k=0}^{2^i-1} \left(\frac{f(x_k) + f(x_{k+1})}{2} \right) \\ R_{m,i} & : \frac{4^m \cdot R_{m-1,i+1} - R_{m-1,i}}{4^m - 1} \end{cases} \tag{10}$$

Można wykazać że:

$$R_{m,i} = I - c_m^* h_i^{2m+2} - \dots \quad (i \geqslant 0; m \geqslant 1)$$

Widzimy zatem, że błąd metody Romberga dla wyrazu $R_{n,m}$ jest rzędu $\mathcal{O}(h_n^{2m+2})$ gdzie $h = \frac{1}{2^n}(b-a)$.

Algorytm obliczający przybliżenia wartości całki za pomocą Metody Romberga:

---

input: a, b, M, f
$h \leftarrow a - b$
$R_{0,0} \leftarrow \frac{1}{2}(b-a)|f(a) - f(b)|$
for $n \leftarrow 1$ to M **do**
  $h \leftarrow \frac{h}{2}$
  $R_{n,0} \leftarrow \frac{1}{2} R_{n-1,0} + h \sum_{i=1} 2^{n-1} f(a + (2i-1)h)$
  for $m \leftarrow 1$ to n **do**
    $R_{n,m} \leftarrow R_{n,m-1} + (R_{n,m-1} - R_{n-1,m-1})/(4^m - 1)$
    print $n, m, R_{n,m}$
  end for
end for

Jest to algorytm pokazujący ideę tej metody, we właściwej implementacji usprawnimy go trochę. Zauważmy, że nie musimy trzymać w pamięci całej dwuwymiarowe tablicy $R[]$ wystarczy ostatni wiersz, obliczenia możemy przerwać przy zadowalającej nas dokładności (nie musimy liczyć aż do $R_{M,0}$).

Wybór n

Dla danego n możemy policzyć wielomian optymalny, w_n^* z rozwiązania (8) następnie stosując Metodę Romberga policzyć błąd (1). To już jesteśmy w stanie zrobić z powyższych wniosków i pomysłów. Teraz pytanie jak dobrać optymalne n to znaczy, najmniejsze spełniające (1)?
Oczywiście zwiększanie n (stopnia wielomianu w_n^*) nie zmniejsza dokładność tj. może zmniejszać $\|f - w_n^*\|$.

Zauważmy jeszcze, że jeżeli w_n^* spełnia (3) to dla dowolnego $k > n$ w_k^* również spełnia warunek (3). Zakres poszukiwań jest więc *uporządkowany* możemy wyszukać binarnie optymalnego n ($\mathcal{O}(\log_2 n)$). W przypadku naszego problemu zakres n nie jest duży, będziemy się po prostu iterować od $n = 0$ aż, w_n^* będzie spełniał (3), rozważmy więc algorytm:

---

input: ε, f
$n \leftarrow 0$
$error \leftarrow get\_error(n, f)$
while $error \geqslant \varepsilon$ **do**
  $error \leftarrow get\_error(n, f)$
  $n \leftarrow n + 1$
end while
return r

Obliczanie błędu (funkcja $get\_error(n, f)$) to po prostu obliczenie $\|f - w_n^*\|^2$ ($\mathcal{O}(n)$).

Implementacja

Cały algorytm będziemy implementowali w języku Julia, używając podwójnej precyzji, Float64.
Do rozwiązywania (8) skorzystamy z

$$A \cdot B = C \iff B = A^{-1} \cdot C$$

Dla odwracalnej A.

Macierz odwrotną macierzy iloczynów skalarnych $n \times n$ policzymy z funkcji wbudowanej języka Julia: **inv()**

Wyniki i interpretacja

Podsumowanie