Sprawozdanie
Metody numeryczne 2
Temat 2, Zadanie nr 12

Mateusz Śliwakowski, F4

28/10/2018

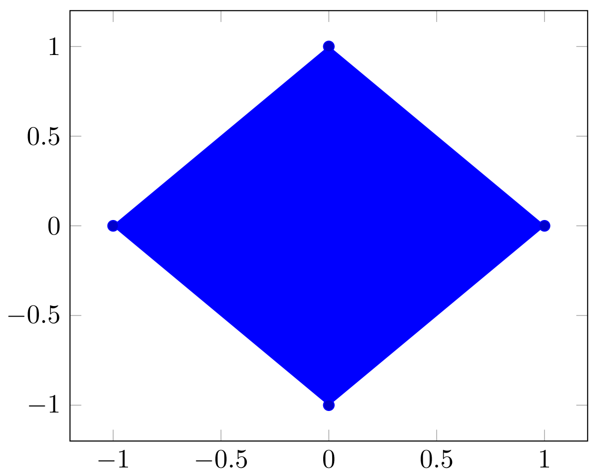
1 Treść zadania

Interpolacja funkcjami kwadratowymi na obszarze $D : |x| + |y| \leq 1$ podzielonym na $4n^2$ trójkątów przystających. Tablicowanie funkcji, przybliżenia i błędu w środkach ciężkości trójkątów. Obliczenie błędu maksymalnego w tych punktach.

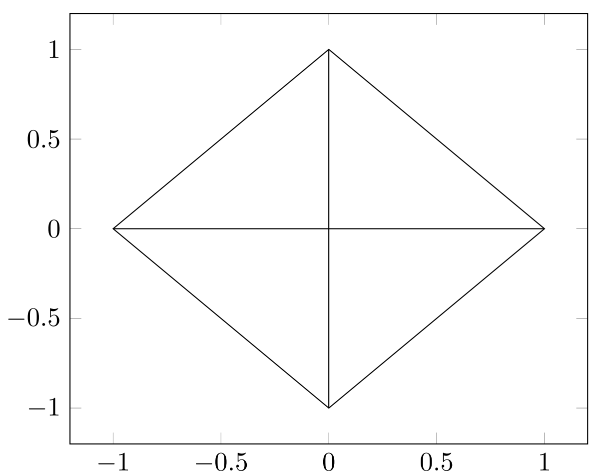
2 Opis metody

2.1 Podział obszaru

Na początku zobaczmy jak wygląda obszar $D : |x| + |y| \leq 1$.



Podzielimy go na 4 trójkąty prostokątne:



Każdy z trójkątów podzielimy na n^2 trójkątów przystających w następujący sposób:

1. Przyprostokątne dzielimy na n odcinków równej długości.
2. Tworzymy odcinki łączące punkty podziału z przeciwprostokątną równoległe do osi odpowiednio OX oraz OY.
3. W każdym z powstałych kwadratów prowadzimy jedną przekątną.

Tym sposobem w trójkącie otrzymujemy $1 + 3 + \cdots + 2n - 1 = n^2$ trójkątów, zatem podział jest poprawny.

2.2 Interpolacja funkcjami kwadratowymi

Niech f - interpolowana funkcja dwóch zmiennych. Będziemy przybliżać tę funkcję za pomocą funkcji kwadratowej postaci $w(x, y) = a_0 + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2$, gdzie dla $i = 0, \ldots, 5$ $a_i \in \mathbb{R}$.

Aby wyznaczyć współczynniki funkcji interpolacyjnej w musimy skorzystać z wartości funkcji f w 6 punktach. Weźmy zatem wierzchołki trójkąta (oznaczmy je P_0, P_1, P_2) oraz środki boków (oznaczmy P_{01}, P_{12}, P_{20}). Przyjmijmy ponadto, że $P_i := (x_i, y_i)$. Utwórzmy zatem układ równań:

$$\begin{cases} w(x_0, y_0) = f(x_0, y_0) \\ w(x_1, y_1) = f(x_1, y_1) \\ w(x_2, y_2) = f(x_2, y_2) \\ w(x_{01}, y_{01}) = f(x_{01}, y_{01}) \\ w(x_{12}, y_{12}) = f(x_{12}, y_{12}) \\ w(x_{20}, y_{20}) = f(x_{20}, y_{20}) \end{cases}$$

Rozwiązując powyższy URL otrzymamy współczynniki szukanej funkcji w.

3 Implementacja metody

```
function [B, err] =  squareInterpolation(fun, n)
```

Parametry wejściowe:

- fun - Uchwyt do interpolowanej funkcji dwóch zmiennych,
- n - liczba naturalna, parametr zadania.

Parametry wyjściowe:

- B - tablica zawierająca: współrzędne x oraz y środków ciężkości trójkątów, wartości funkcji interpolowanej w tych punktach, wartości funkcji interpolacyjnej w tych punktach, błąd przybliżenia.
- err - maksymalny błąd interpolacji.

Na początku musimy znaleźć wierzchołki trójkątów wymaganych podczas interpolacji. W tym celu za pomocą funkcji *meshgrid* dzielimy obszar na siatkę równoodległych punktów na obszarze $[-1,1] \times [-1,1]$. Następnie wyznaczamy podział lewego górnego trójkąta i zapisujemy go w wektorze pomocniczym. Podział ten odbijamy wzdłuż osi OY, a następnie OX otrzymując wszystkie wymagane punkty.

Potem następuje właściwa interpolacja. Dla każdego z trójkątów:

- Zapisujemy do tablicy wynikowej współrzędne środka ciężkości danego trójkąta.
- Zapisujemy wartość funkcji interpolowanej w środku ciężkości.
- Wyznaczamy współczynniki funkcji interpolacyjnej.
- Tablicujemy wartość funkcji interpolacyjnej oraz błąd interpolacji.

Na koniec wyznaczamy maksymalny błąd.

Dla uproszczenia kodu, użyta została prosta funkcja $initializeAFromT$, która konwertuje współrzędne iteracyjne (z zakresu $[1, n]$) na współrzędne na płaszczyźnie.

4 Przykłady i wnioski

4.1 Przykłady

Wszystkie przedstawione przykłady zdefiniowane są w skrypcie $testInterpolation.m$. Najpierw przetestujemy funkcję interpolacyjną pod kątem maksymalnego błędu przybliżenia. W ramkach przedstawiona jest część wyjścia Matlaba po uruchomieniu skryptu $testInterpolation$.

- Rozpoczniemy od przykładu, gdzie nasz program powinien działać praktycznie bezbłędnie, czyli wywołamy funkcję $squareInterpolation$ dla funkcji kwadratowej dwóch zmiennych.

  ===============TEST CASE===============
  $@(x, y)\ \ 2 - 23 * x + 30 * y + 4 * x * y - 2.5 * x * x + 9 * y * y$

  n = 1: maxError: 0
  n = 10: maxError: 1.4211e-14
  n = 100: maxError: 1.7764e-14
  n = 200: maxError: 1.7764e-14

  Zgodnie z oczekiwaniami otrzymujemy dokładność porównywalną z dokładnością maszynową.

- Zobaczmy jak program zachowa się dla funkcji złożonej z funkcji trygonometrycznych.

```
===============TEST CASE================
@(x,y) sin(x) + 3 * cos(y) - 4 * sin(x * y) + 8 * cos(x * x * y)

n = 1: maxError: 0.039506
n = 10: maxError: 0.00026801
n = 100: maxError: 3.1845e-07
n = 200: maxError: 4.0173e-08
```

  Program zachowuje się poprawnie - wraz ze wzrostem liczby trójkątów, których używamy do interpolacji rośnie dokładność.

- Dla wielomianu wyższego stopnia:

```
===============TEST CASE================
@(x,y) @4 * x^9 + 3 * x^3 + 8 * x - 4 * y^8 + 3 * y^2;

n = 1: maxError: 0.97998
n = 10: maxError: 0.00906
n = 100: maxError: 1.2157e-05
n = 200: maxError: 1.5444e-06
```

  Również osiągamy dobrą dokładność dla wysokich n, lecz już nie tak wysoką jak dla funkcji trygonometrycznej.

- Na koniec sprawdźmy jak program zadziała dla funkcji, która w zadanym przedziale ma punkty, dla których nie jest określona.

```
===============TEST CASE================
@(x,y) tan(2 * x + 3 * y)

n = 1: maxError: 17.2849
n = 10: maxError: 216.6608
n = 100: maxError: 599.0144
n = 200: maxError: 770.5785
```

  W tym przypadku nie jesteśmy w stanie przeprowadzić poprawnej interpolacji - dla wszystkich wartości n maksymalny błąd osiąga wysokie wartości. Dzieje się tak dlatego, że w otoczeniu punktów, gdzie funkcja tangens nie jest określona jej pochodna osiąga bardzo wysokie wartości - nie jest zatem możliwa dokładna interpolacja.

4.2 Doświadczalne wyznaczenie współczynnika zbieżności

Sprawdźmy teraz jaki jest współczynnik zbieżności naszej metody. W każdej iteracji będziemy zwiększać parametr n dwa razy i wyświetlać iloraz kolejnych błędów.

```
5.7240
7.6428
7.3948
7.7738
7.9081
7.9693
7.9837
7.9916
```

Jak widać $n \to 8$ zatem współczynnik zbieżności może wynosić 3 (bo $2^3 = 8$).