Metoda elementów skończonych

Jakub Świerczyński

23 stycznia 2025

1 Wstęp

Celem projektu jest numeryczna analiza przewodzenia ciepła w próbce. Rozwiązanie problemu jest oparte na Metodzie Elementów Skończonych (MES) zaimplementowanej w języku Python. Kody źródłowe składają się z kilku plików odpowiedzialnych m.in. za:

- wczytywanie danych wejściowych (`GlobalData.py`, `Node.py`, `Element.py`),
- definiowanie funkcji kształtu i punktów całkowania (`UniversalElement.py`),
- obliczanie macierzy lokalnych (`H`, `HBC`, `C`) i wektora wymuszeń (`P`) dla każdego elementu,
- składanie (agregację) macierzy lokalnych i wektorów w macierze globalne (`Solution.py`)

W dalszych sekcjach zostanie omówione, jak działają najważniejsze elementy kodu, jak przebiega całkowanie numeryczne oraz w jaki sposób liczone są poszczególne macierze i wektory. Na końcu zostaną przedstawione wyniki i porównanie z testami:

(i) `4x4`,

(ii) `31x31`,

(iii) `4x4mix`

2 Opis głównych etapów obliczeń

2.1 Funkcje kształtu i całkowanie numeryczne

W celu aproksymacji rozkładu temperatury $T(x, y)$ w każdym elemencie skończonym (element czworokątny 4-węzłowy) stosowane są **funkcje kształtu**:

$$N_1(\xi,\eta) = \tfrac{1}{4}(1-\xi)(1-\eta), \quad N_2(\xi,\eta) = \tfrac{1}{4}(1+\xi)(1-\eta),$$
$$N_3(\xi,\eta) = \tfrac{1}{4}(1+\xi)(1+\eta), \quad N_4(\xi,\eta) = \tfrac{1}{4}(1-\xi)(1+\eta).$$

Kod generuje te funkcje (oraz ich pochodne względem ξ, η) w pliku `UniversalElement.py`. Następnie, aby obliczyć całki w formułach na macierze lokalne, wykorzystujemy **kwadraturę Gaussa** o zadanej liczbie punktów całkowania:

$$\int_{-1}^{1} f(\xi)\, d\xi \approx \sum_{i=1}^{n_{pc}} w_i\, f(\xi_i),$$

gdzie n_{pc} to liczba punktów całkowania, zaś w_i – odpowiednie wagi. W kodzie parametry te (pozycje punktów ξ_i, η_i oraz ich wagi w_i) są zdefiniowane także w `UniversalElement.py`.

2.2 Jakobian i transformacja do układu globalnego

Przy obliczeniach pochodnych funkcji kształtu w układzie (x, y) konieczne jest użycie **Jakobianu** przekształcenia między lokalnym układem (ξ, η) a globalnym (x, y). Macierz Jacobiego wyliczana jest w pliku `Element.py` (metoda `calculate_jacobian()`) na podstawie:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}.$$

Następnie wyznacznik $\det(J)$ pozwala wyliczyć skalę przekształcenia przy całkowaniu. Odwrócenie Jacobiego daje $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$ z pochodnych po ξ, η.

2.3 Macierz H (przewodzenia) i macierz HBC (warunek brzegowy konwekcji)

Macierz H: Macierz sztywności przewodzenia, oznaczana tutaj przez **H**, wyliczana jest wg wzoru:

$$\mathbf{H}^{(e)} = \int_{\Omega_e} k \, \nabla N^T \cdot \nabla N \; d\Omega,$$

gdzie k to współczynnik przewodzenia ciepła, ∇N – wektor pochodnych funkcji kształtu w układzie globalnym. W kodzie odpowiada za to metoda `calculate_matrix_h()` (plik `Element.py`), która sumuje wkład z każdego punktu całkowania (również uwzględniając $\det(J)$ oraz wagi kwadratury).

Macierz HBC (konwekcja): Dla warunku konwekcji (Newtona) na brzegu rozważamy lokalną macierz:

$$\mathbf{HBC}^{(e)} = \int_{\Gamma_e} \alpha \, [N]^T [N] \, d\Gamma,$$

gdzie α to współczynnik konwekcji, a $[N]$ – wektor funkcji kształtu na danym brzegu. W kodzie jest to metoda `calculate_matrix_Hbc()` w `Element.py`. Wykorzystuje się tam m.in. tablicę `N_bc` (funkcje kształtu na krawędziach) oraz wagi do całkowania brzegowego.

2.4 Wektor P (warunek brzegowy z T_∞)

W warunku konwekcji pojawia się też człon αT_∞, co generuje w **wektorze prawych stron** (**P**) składnik:

$$\mathbf{P}^{(e)} = \int_{\Gamma_e} \alpha \, T_\infty [N]^T \, d\Gamma.$$

W kodzie oblicza go metoda `calculate_vector_P()` (plik `Element.py`). Zwraca wektor 4-elementowy (dla 4-węzłowego elementu), który następnie jest składany w wektor globalny.

2.5 Macierz C (pojemność cieplna) i schemat niestacjonarny

Dla zagadnienia niestacjonarnego występuje macierz $\mathbf{C}$, która reprezentuje pojemność cieplną. Jej wzór (lokalny) to:

$$\mathbf{C}^{(e)} = \int_{\Omega_e} \rho\, c\, [N]^T [N]\; d\Omega,$$

gdzie ρ – gęstość, c – ciepło właściwe. W kodzie obliczana jest przez `calculate_matrix_C()`. Przy rozwiązywaniu układu równań w kolejnych krokach czasowych t_{n+1} stosuje się schemat typu:

$$\big(\mathbf{H} + \frac{1}{\Delta t}\mathbf{C}\big)\mathbf{T}^{(n+1)} = \frac{1}{\Delta t}\mathbf{C}\,\mathbf{T}^{(n)} + \mathbf{P}.$$

Implementacja pętli czasowej znajduje się w `Solution.py` (metoda `solve()`).

3 Agregacja (składanie) macierzy i wektorów lokalnych

Po obliczeniu dla każdego elementu macierzy lokalnych `H`, `Hbc`, `C` oraz wektora `P`, następuje **agregacja** do struktur globalnych. W pliku `Solution.py` (metoda `agregate()`) widać:

```
for element in self.elements:
    for i in range(len(element.node_ids)):
        global_i = element.node_ids[i] - 1
        for j in range(len(element.node_ids)):
            global_j = element.node_ids[j] - 1
            self.matrix_h_global[global_i, global_j] += element.H[i][j]
            self.matrix_h_global[global_i, global_j] += element.Hbc[i][j]
            self.matrix_c_global[global_i, global_j] += element.C[i][j]
        self.vector_p_global[global_i] += element.P[i]
```

Każdy element zna swoje 4 węzły (w naszym przypadku czworokąt), a `node_ids` pozwala wstawić lokalne wartości do odpowiednich miejsc w macierzy/wektorze globalnym.

4 Porównanie wyników dla różnych siatek

Testy wykonano dla kilku przypadków:

- **Siatka 4x4** – jest to siatka z 9 elementami (każdy 4-węzłowy), w prostokątnym obszarze.
- **Siatka 31x31** – gęstsza siatka, pozwala uzyskać dokładniejszy wynik.
- **Siatka 4x4mix** – wariant z pewną modyfikacją geometrii lub węzłów brzegowych (np. zróżnicowane rozmieszczenie węzłów).

Dodatkowo wykonano:

- **Test niestacjonarny** – zgodnie z załączonym plikiem `11. Test niestacjonarny.pdf`, weryfikując rozkład temperatur w czasie.
- **Test stacjonarny** – długotrwałe symulacje prowadzą do wyrównania temperatury próbki z temperaturą otoczenia (co potwierdza zgodność z teorią).

5 Porównanie wyników dla różnych siatek

Testy wykonano dla kilku przypadków:

- **Siatka 4x4** – jest to siatka z 9 elementami (każdy 4-węzłowy), w prostokątnym obszarze.
- **Siatka 31x31** – gęstsza siatka, pozwala uzyskać dokładniejszy wynik.
- **Siatka 4x4mix** – wariant z pewną modyfikacją geometrii lub węzłów brzegowych (np. zróżnicowane rozmieszczenie węzłów).

Dodatkowo wykonano:

- **Test niestacjonarny** – zgodnie z załączonym plikiem `11. Test niestacjonarny.pdf`, weryfikując rozkład temperatur w czasie.
- **Test stacjonarny** – długotrwałe symulacje prowadzą do wyrównania temperatury próbki z temperaturą otoczenia (co potwierdza zgodność z teorią).

5.1 Porównanie z wynikami testowymi

(1) **4x4:** Dla symulacji w czasie do 500 s z krokiem 50 s otrzymano poniższe wartości skrajnych temperatur (minimalnych i maksymalnych) w każdym kroku. W nawiasach podano dla porównania wartości oczekiwane (testowe):

Czas [s]	Otrzymane (min, max)	Oczekiwane (min, max)
50	110.03797, 365.81547	110.03798, 365.81547
100	168.83701, 502.59171	168.83702, 502.59171
150	242.80085, 587.37267	242.80086, 587.37267
200	318.61459, 649.38748	318.61459, 649.38748
250	391.25579, 700.06842	391.25579, 700.06842
300	459.03691, 744.06334	459.03690, 744.06334
350	521.58629, 783.38285	521.58627, 783.38285
400	579.03446, 818.99218	579.03444, 818.99219
450	631.68926, 851.43104	631.68924, 851.43104
500	679.90762, 881.05763	679.90759, 881.05763

Różnice między wartościami otrzymanymi a oczekiwanymi są bardzo małe (rzędu 10^{-5} do 10^{-4}). Pod względem jakościowym i ilościowym wyniki dobrze pokrywają się z danymi referencyjnymi.

(2) **31x31 (kwadrat):** W tym przypadku przeprowadzono obliczenia do 20 s z krokiem 1 s, co daje 20 kroków czasowych. W kolejnych sekundach odnotowano wartości minimalne oraz maksymalne temperatur w węzłach. Poniżej pokazano wybrane etapy:

Czas [s]	Otrzymane (min, max)	Oczekiwane (min, max)
1	100.00000, 149.55695	99.99970, 149.55663
5	100.00000, 226.68258	100.00150, 226.68374
10	100.00037, 276.70110	100.00370, 276.70464
15	100.00858, 312.45123	100.01392, 312.45687
20	100.06432, 341.08466	100.07184, 341.09201

Tutaj również różnice są niewielkie (głównie w trzecim–czwartym miejscu po przecinku). Niewielkie rozbieżności (rzędu 10^{-3}–10^{-4}) wynikają z zastosowanej dyskretyzacji w czasie oraz sposobu aproksymacji warunków brzegowych. Ogólnie jednak charakter zmian temperatur i wartości końcowe w pełni potwierdzają zgodność z wynikami testowymi.

(3) **4x4mix:** Przykładowe wartości otrzymane i oczekiwane (dane w krokach co 50 s do 500 s) pokazano w tabeli poniżej.

Czas [s]	Otrzymane (min, max)	Oczekiwane (min, max)
50	95.15907, 374.66827	95.15185, 374.68633
100	147.65590, 505.95426	147.64442, 505.96811
150	220.17811, 586.98942	220.16445, 586.99785
200	296.75087, 647.28011	296.73644, 647.28558
300	440.57401, 741.21565	440.56014, 741.21911
400	564.01392, 817.42051	564.00151, 817.39151
500	667.77643, 880.19230	667.76555, 880.16761

Jak widać, w przypadku tej siatki „mieszanej" różnice sięgają niekiedy 10^{-2}–10^{-3}, co jest zgodne z wcześniejszym opisem (zwiększona nieregularność geometrii wpływa na nieco większe błędy aproksymacji).

(4) **Test niestacjonarny (pdf)**: Przebieg zmian temperatury w czasie (dostarczony w `Test niestacjonarny.pdf`) został w pełni potwierdzony — wraz z postępem czasu poszczególne węzły wyrównują temperaturę, dążąc do warunków brzegowych z otoczenia (zgodnie z wartością `Tot`).

(5) **Test stacjonarny:** Przy dużych czasach symulacji (np. rzędu kilku tysięcy sekund), we wszystkich wariantach siatek temperatura w próbce stabilizuje się i dąży do wartości zadanej `Tot` = 1200 K. Potwierdza to poprawność podejścia oraz zgodność z teorią wymiany ciepła w stanie ustalonym.

6 Wnioski

- Implementacja kodu MES pozwoliła na poprawne wyznaczenie rozkładu temperatur i zachowanie zgodności z danymi testowymi w różnych wariantach siatek (4x4, 31x31, 4x4mix).
- Kluczowe elementy obliczeń obejmują m.in.: prawidłowe wyznaczenie macierzy Jacobiego, poprawną implementację funkcji kształtu oraz właściwe całkowanie w punktach Gaussa (zarówno w obszarze elementu, jak i na krawędziach).
- Przeprowadzono zarówno analizę niestacjonarną (z krokiem czasowym `SimulationStepTime`), jak i przypadek stacjonarny (duży czas symulacji). We wszystkich przypadkach otrzymano oczekiwane wyniki, a drobne różnice rzędu 10^{-5}–10^{-2} wynikają z dyskretyzacji oraz sposobu aproksymacji warunków brzegowych.