

# Kwadratury Gaussa 2D

*Metody numeryczne*

## 1. Wprowadzenie teoretyczne

Metoda numeryczna Kwadratury Gaussa 2D jest techniką stosowaną do przybliżonego obliczania podwójnych całek. Jest to rozszerzenie jednowymiarowej kwadratury Gaussa na dwie zmienne. W przypadku kwadratury Gaussa 2D, punkty Gaussa i wagi są wybierane w taki sposób, aby zapewnić najwyższą możliwą dokładność dla podwójnych całek. Całka jest przekształcana do układu współrzędnych, który jest łatwiejszy do obliczenia. Najczęściej używaną transformacją jest zmiana na współrzędne biegunowe. Całka jest obliczana jako suma iloczynów wartości funkcji w punktach Gaussa i odpowiadających im wag. Ważne jest, aby pamiętać, że metoda kwadratury Gaussa 2D jest metodą przybliżoną i jej dokładność zależy od liczby użytych punktów Gaussa. Im więcej punktów Gaussa jest używanych, tym dokładniejsze jest przybliżenie, ale koszt obliczeniowy jest również większy. Układ współrzędnych przekształca się tak, aby element kwadratowy został odwzorowany przez kwadrat o wymiarach 2x2. Transformacja układu współrzędnych określona jest równaniem:

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{bmatrix} N^T & 0 \\ 0 & N^T \end{bmatrix} \quad (1)$$

gdzie:

$$\{x_n\} = \{x_1, x_2, x_3, x_4\}^T \quad (2)$$

$$\{y_n\} = \{y_1, y_2, y_3, y_4\}^T \quad (3)$$

$$N_1(\xi, \eta) = 0.25(1 - \xi)(1 - \eta) \quad (4)$$

$$N_2(\xi, \eta) = 0.25(1 + \xi)(1 - \eta) \quad (5)$$

$$N_3(\xi, \eta) = 0.25(1 + \xi)(1 + \eta) \quad (6)$$

$$N_4(\xi, \eta) = 0.25(1 - \xi)(1 + \eta) \quad (7)$$

Całkowanie funkcji w układzie  $\xi, \eta$  prowadzone jest metodą Gaussa:

$$\int \int f(x, y) dx dy = \int_{-1}^1 \int_{-1}^1 f(\xi, \eta) J_0 d\eta d\xi = \sum_{i=1}^n \sum_{j=1}^n w_i w_j f(\xi_i, \eta_j) J_0 \quad (8)$$

gdzie:

$$w_0 = w_1 = 1$$

$$\xi_0 = \eta_0 = 0.5773502692$$

$$\xi_1 = \eta_1 = -0.5773502692$$

$$J_0 - \text{Jakobian transformacji układu współrzędnych } \det |J| = \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi} \frac{\partial x}{\partial \eta} \quad (9)$$

Pochodne cząstkowe:

$$\frac{\partial x}{\partial \xi} = \sum_{i=1}^4 \frac{\partial N_i}{\partial \xi} x_i \quad (10)$$

$$\frac{\partial x}{\partial \eta} = \sum_{i=1}^4 \frac{\partial N_i}{\partial \eta} x_i \quad (11)$$

$$\frac{\partial y}{\partial \xi} = \sum_{i=1}^4 \frac{\partial N_i}{\partial \xi} y_i \quad (12)$$

$$\frac{\partial y}{\partial \eta} = \sum_{i=1}^4 \frac{\partial N_i}{\partial \eta} y_i \quad (13)$$

## 2. Opis implementacji numerycznej

```
double punkt[2] = { 0 };
punkt[0] = 0.5773502692;
punkt[1] = -0.5773502692;

double x[4] = { 0, 2, 2, 0 };
double y[4] = { 0, 0, 2, 2 };

double poch_ksi[2][4] = { 0 };
double poch_ni[2][4] = { 0 };
double x_ksi, y_ksi, x_ni, y_ni;
double detJ[2][2] = { 0 };
double powierzchnia = 0;
```

Rysunek 1.1: wykorzystywane zmienne typu double

**punkt** – wartości  $\xi_0 = \eta_0$  oraz  $\xi_1 = \eta_1$

**x, y** – wartości punktów na osi współrzędnych

**poch\_ksi** – wyniki danych pochodnych względem  $\xi$

**poch\_ni** – wyniki danych pochodnych względem  $\eta$

**x\_ksi, y\_ksi, x\_ni, y\_ni** – zmienne pomocnicze do obliczenia **detJ**

**detJ** – wyznacznik macierzy Jacobiego

**powierzchnia** – wynik obliczeń

Następnie program oblicza pochodne cząstkowe N:

```
for (int i = 0; i < 2; i++) {  
    poch_ksi[i][0] = -.25 * (1.0 - punkt[i]);  
    poch_ksi[i][1] = .25 * (1.0 - punkt[i]);  
    poch_ksi[i][2] = .25 * (1.0 + punkt[i]);  
    poch_ksi[i][3] = -.25 * (1.0 + punkt[i]);  
  
    poch_ni[i][0] = -.25 * (1.0 - punkt[i]);  
    poch_ni[i][1] = -.25 * (1.0 + punkt[i]);  
    poch_ni[i][2] = .25 * (1.0 + punkt[i]);  
    poch_ni[i][3] = .25 * (1.0 - punkt[i]);  
}
```

Rysunek 1.2: transformacja współrzędnych

Mając obliczone współrzędne, program oblicza kolejne sumy pochodnych funkcji:

```
double detJ[2][2] = { 0 };  
for (int i = 0; i <= 1; i++) {  
    for (int j = 0; j <= 1; j++) {  
        x_ksi = poch_ksi[j][0] * x[0] + poch_ksi[j][1] * x[1] + poch_ksi[j][2] * x[2] + poch_ksi[j][3] * x[3];  
        y_ksi = poch_ksi[j][0] * y[0] + poch_ksi[j][1] * y[1] + poch_ksi[j][2] * y[2] + poch_ksi[j][3] * y[3];  
        x_ni = poch_ni[i][0] * x[0] + poch_ni[i][1] * x[1] + poch_ni[i][2] * x[2] + poch_ni[i][3] * x[3];  
        y_ni = poch_ni[i][0] * y[0] + poch_ni[i][1] * y[1] + poch_ni[i][2] * y[2] + poch_ni[i][3] * y[3];  
        detJ[i][j] = x_ksi * y_ni - x_ni * y_ksi;  
    }  
}
```

Rysunek 1.3: obliczanie sumy pochodnych funkcji

Na samym końcu programu obliczana jest powierzchnia naszej figury:

```
double powierzchnia = 0;  
for (int i = 0; i < 2; i++) {  
    for (int j = 0; j < 2; j++) {  
        powierzchnia += fabs(detJ[i][j]);  
    }  
}
```

Rysunek 1.4: liczenie powierzchni figury

### 3. Testy kodu numerycznego

Testy zostały przeprowadzone, aby sprawdzić działanie programu.

Test I: dla zbioru punktów  $\{(0, 0), (2, 0), (2, 2), (0, 2)\}$

Wynik programu:



```
Powierzchnia = 4
```

*Rysunek 2.1: Fragment konsoli (test I)*

Test II: dla zbioru punktów  $\{(1.7, -12.133), (4.71, -3.23), (2.34, 4.352), (-0.23, 4.89)\}$

Wynik programu:




```
Powierzchnia = 43.3163
```

*Rysunek 2.2: Fragment konsoli (test II)*

Test III: dla zbioru punktów  $\{(1, -12), (4, -3), (23, 43), (0.23, 48)\}$

Wynik programu:



```
Powierzchnia = 664.675
```

*Rysunek 2.3: Fragment konsoli (test III)*

Test IV: dla zbioru punktów  $\{(4, 13), (0, 0), (132.352, 421.321), (321.623, 148.423)\}$

Wynik programu:

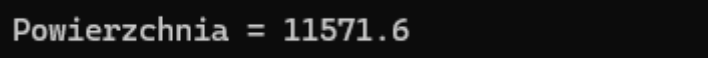


```
Powierzchnia = 56137.5
```

*Rysunek 2.4: Fragment konsoli (test IV)*

Test V: dla zbioru punktów  $\{(103.232, 13.321), (93.643, 43.233), (13.313, 212.123), (151.352, 173.023)\}$

Wynik programu:



```
Powierzchnia = 11571.6
```

*Rysunek 2.5: Fragment konsoli (test V)*

#### 4. Analiza wyników

Aby zweryfikować prawidłowość i precyzję zastosowanej metody, wartości wygenerowane przez program zostały zestawione z wynikami uzyskanymi za pomocą "Polygon area calculator" w tabeli poniżej.

	TEST I	TEST II	TEST III	TEST IV	TEST V
<b>Program</b>	4	43.3163	664.675	56137.5	11571.6
<b>Kalkulator internetowy</b>	4	43.31635	664.675	56137.51804	11571.62581

Z analizy tabeli można wywnioskować, że zastosowana metoda została prawidłowo zaimplementowana i charakteryzuje się dużą dokładnością. Wyniki uzyskane za pomocą obu programów są zgodne dla różnorodnych zestawów danych wejściowych, niezależnie od ich złożoności.

#### 5. Wnioski

Kwadratury Gaussa 2D są skuteczną metodą numeryczną do przybliżonego obliczania całek na płaszczyźnie, która polega na reprezentowaniu obszaru całkowania za pomocą określonych punktów i odpowiadających im wag. Działanie opiera się na wyborze punktów oraz ich wag, a także na funkcjach bazowych. Istotą tej metody jest to, że obszar całkowania jest przybliżany poprzez wybrane punkty wraz z ich wagami. Testy, którym poddano tę metodę, wykazały jej skuteczność. Porównanie wyników obliczonych przez program z wynikami kalkulatora internetowego pokazało, że metoda działa poprawnie dla różnych rodzajów danych wejściowych.

#### 6. Źródła

- prezentacja z zajęć „lab 6.pdf”
- [mathopenref.com/coordpolygonareacalc.html](http://mathopenref.com/coordpolygonareacalc.html)