Metoda Galerkina dla równania Poissona w jednym wymiarze

Jan Smółka
Wydział Informatyki, Elektroniki i Telekomunikacji, Akatemia Górniczo Hutnicza w Krakowie

Niniejszy dokument zawiera opis wykonania programu rozwiązującego jednowymiarowy problem Poissona z warunkami brzegowymi Dirichleta, wykonanego w języku programowania Julia jako rozwiązanie zadania obliczeniowego w ramach przedmiotu Równania Różniczkowe i Różnicowe.

1. Omówienie zagadnienia

1.1. *Problem równania Poissona*

Znaleźć funkcję jednej zmiennej spełniającą równanie z warunkami brzegowymi:

$$\begin{cases} \frac{d^2u}{dx^2} = 4\pi G\rho(x), & x \in \Omega = (a,b) \subset \mathbb{R} \\ u(a) = u_a \wedge u(b) = u_b, & u_a, u_b = const. \end{cases} \tag{1.1}$$

1.2. *Sformułowanie wariacyjne*

Od rozwiązania oczekujemy, by było elementem hilbertowskiej przestrzeni Sobolewa funkcji dwukrotnie różniczkowalnych w sensie słabym:

$$u \in W^{2,2}(\Omega) = H^2(\Omega) \tag{1.2}$$

Nad Ω wprowadzamy przestrzeń funkcji próbnych:

$$v \in H_0^\infty(\Omega) \tag{1.3}$$

Obie strony danego równania mnożymy przez funkcję próbną i całkujemy po Ω:

$$\int_a^b \frac{d^2u(x)}{dx^2} v(x)dx = 4\pi G \int_a^b \rho(x)v(x)dx \tag{1.4}$$

Po lewej stronie całkujemy przez części i korzystamy z faktu, że $v(\partial\Omega) = 0$:

$$\int_a^b \frac{du(x)}{dx}\frac{dv(x)}{dx}dx = -4\pi G \int_a^b \rho(x)v(x)dx \tag{1.5}$$

Od rozwiązania (1.1) oczekujemy, by miało niezerowe wartości na brzegu Ω, zgodnie ze schematem warunków Dirichleta. Połóżmy $u = w + \widetilde{u}$. Funkcję $\widetilde{u}(x)$ wybierzmy tak, by osiągała na $\partial\Omega$ takie wartości jak poszukiwane rozwiązanie:

$$\widetilde{u}(x) = (x-a)\frac{u_b - u_a}{b-a} + u_a \implies (\widetilde{u}(a) = u_a \wedge \widetilde{u}(b) = u_b) \tag{1.6}$$

Przekształcając (1.5) zgodnie z podstawieniem otrzymujemy:

$$\int_a^b \frac{dw(x)}{dx}\frac{dv(x)}{dx}dx = -4\pi G\int_a^b \rho(x)v(x)dx - \int_a^b \frac{d\widetilde{u}(x)}{dx}\frac{dv(x)}{dx}dx \tag{1.7}$$

Dalej, z postaci $\widetilde{u}$:

$$\int_a^b \frac{dw(x)}{dx}\frac{dv(x)}{dx}dx = -4\pi G\int_a^b \rho(x)v(x)dx - \frac{u_b-u_a}{b-a}\int_a^b \frac{dv(x)}{dx}dx \tag{1.8}$$

Dla nowego rozwiązania $w \in H_0^1(\Omega)$.

Widzimy, że:

$$a(w,v) = \int_a^b \frac{dw(x)}{dx}\frac{dv(x)}{dx}dx \tag{1.9}$$

jest symetryczną formą dwuliniową, zaś

$$\phi(v) = -4\pi G\int_a^b \rho(x)v(x)dx - \frac{u_b-u_a}{b-a}\int_a^b \frac{dv(x)}{dx}dx \tag{1.10}$$

jest formą liniową.

Otrzymany problem wariacyjny polega na znalezieniu funkcji $w(x)$ takiej, że:

$$\forall v \in H_0^\infty(\Omega) : a(w,v) = \phi(v) \tag{1.11}$$

Rozwiązaniem jest funkcja $u(x) = w(x) + \widetilde{u}(x)$.

1.3. *Metoda elementów skończonych*

Przeprowadzamy dyskretyzację zbioru Ω na n równoodległych punktów $(x_i)_{i=1,2,\ldots,n}$, zwanych *węzłami*. Stosujemy konwencję $x_1 = a$, $x_n = b$. Odległość między sąsiednimu punktami oznaczmy jako h.

Dla sformułowania (1.11) stosujemy metodę Galerkina, przyjmując za przestrzeń aproksymującą powłokę liniową zbioru n jednowymiarowych funkcji bazowych Lagrange'a:

$$\eta_i(x) = \begin{cases} \frac{1}{h}(x - x_{i-1}), & x \in [x_{i-1}, x_i] \\ -\frac{1}{h}(x - x_{i+1}), & x \in [x_i, x_{i+1}] \end{cases} \tag{1.12}$$

uwzględniając jednostronne określenie funkcji η_1 i η_n.

Zarówno funkcje próbne, jak i poszukiwane rozwiązanie problemu wariacyjnego można przybliżyć w tak ustalonej bazie:

$$w(x) \approx \sum_{i=1}^n \alpha_i \eta_i(x) \tag{1.13}$$

$$v(x) \approx \sum_{i=1}^n \beta_i \eta_i(x) \tag{1.14}$$

Podstawiając (1.13) i (1.14) do (1.11) i korzystając z własności odpowiednich form otrzymujemy:

$$\sum_{i=1}^{n} \beta_j (\sum_{j=1}^{n} (\alpha_j a(\eta_i, \eta_j)) - \phi(\eta_i)) = 0 \tag{1.15}$$

W równoważnej postaci:

$$\forall i = 1, 2, ..., n : \sum_{j=1}^{n} \alpha_j a(\eta_i, \eta_j) = \phi(\eta_i) \tag{1.16}$$

Co można zapisać macierzowo:

$$\begin{bmatrix} a(\eta_1,\eta_1) & a(\eta_1,\eta_2) & \dots & a(\eta_1,\eta_n) \\ a(\eta_2,\eta_1) & a(\eta_2,\eta_2) & \dots & a(\eta_2,\eta_n) \\ \dots & \dots & \dots & \dots \\ a(\eta_n,\eta_1) & a(\eta_n,\eta_2) & \dots & a(\eta_n,\eta_n) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \phi(\eta_1) \\ \phi(\eta_2) \\ \dots \\ \phi(\eta_n) \end{bmatrix} \tag{1.17}$$

Obliczmy wartości formy a:

$$a(\eta_i, \eta_i) = \int_a^b \eta_i^2(x) dx = \frac{2}{h} \tag{1.18}$$

$$a(\eta_i, \eta_{i\pm1}) = \int_a^b \eta_i(x) \eta_{i\pm1} dx = -\frac{1}{h} \tag{1.19}$$

pozostałe elementy znikają, z własności wybranych funkcji bazowych.

Rozwiązanie układu równań (1.17) jest tożsame ze znalezieniem przybliżenia funkcji $u(x)$ stanowiącej rozwiązanie problemu wariacyjnego.

2. Implementacja

Dla wyprowadzonego w powyższej sekcji problemu wykonano implementację, domyślnie działającą dla parametrów danych w zadaniu:

$$\Omega = (0, 3) \tag{2.1}$$

$$\rho(x) = \begin{cases} 0, & x \in (0,1) \cup (2,3) \\ 1, & x \in (1,2) \end{cases} \tag{2.2}$$

$$u(0) = 5 \wedge u(3) = 4 \tag{2.3}$$

jednak możliwe jest działanie dla innych końców przedziału, warunków brzegowych i rozkładu masy.

2.1. *Struktury danych*

Kluczową strukturą zdefiniowaną w kodzie programu jest *Basis*. Przechowuje ona informacje o końcach przedziału, liczbie węzłów i odległości między nimi. Zawiera również tablicę węzłów i słownik funkcji bazowych, skojarzonych z kolejnymi indeksami węzłów.

2.2. *Całkowanie*

Program korzysta z funkcji *integral*() całkowania numerycznego metodą prostokątów. Parametr metody, *integral_steps*, zdefiniowany na początku skryptu, wskazuje liczbę podziałów przedziału całkowania.

2.3. *Generowanie równań*

Za generowanie równań Galerkina odpowiadają funkcje $lhs\_matrix()$ i $rhs\_vector()$. Pierwsza z nich tworzy macierz formy dwuliniowej (1.9), wstawiając wartości (1.18) i (1.19) w odpowiednie pozycje, resztę wypełniając zerami. Druga funkcja oblicza wartości formy liniowej (1.10) wykonując całkowanie numeryczne za pomocą funkcji *integral*().

2.4. *Rozwiązywanie układu równań*

Układ równań (1.17) jest rozwiązywany za pomocą standardowej funkcji języka Julia, bazującej na dekompozycji LU macierzy.

2.5. *Generowanie rozwiązania*

Rozwiązanie układu (1.17) daje wartości $w(x)$ w węzłach. Ostatecznym wynikiem działania programu jest obraz gęstszego zbioru przez odwzorowanie $\sum_{i=1}^{n} \alpha_i \eta_i(x) + \widetilde{u}(x)$.

3. Program

3.1. *Wymagania*

Do korzystania z programu konieczne jest posiadanie Julia REPL, dostępnego do pobrania z oficjalnej strony języka. Konieczna jest instalacja pakietu *Plots.jl*, umożliwiającego rysowanie wykresów:

```
julia> using Pkg
julia> Pkg.add("Plots")
```

3.2. *Uruchomienie*

W celu skorzystania z programu, należy uruchomić terminal Julii w katalogu zawierającym plik źródłowy *solver.jl*, zaimportowanie go do bieżącej sesji oraz wywołanie funkcji *main*(), podając jako argument liczbę węzłów (domyślna wartość argumentu wynosi 100).

```
$ julia
julia> include("solver.jl")
julia> main(100)
```

Po wykonaniu powyższych komend, po upłynięciu czasu proporcjonalnego do podanej liczby węzłów, na ekranie powinno pojawić się okno zawierające wykres otrzymanego rozwiązania. Należy zwrócić uwagę, że skala osi dobierana jest dynamicznie, zależnie od prezentowanych wartości.

Dla większej wygody warto rozważyć korzystanie z programu wewnątrz środowiska programistycznego zintegrowanego z procedurami generującymi wykresy, np. Visual Studio Code z wtyczką języka Julia.