Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Arkadiusz Słowik
Nr albumu: 417835

Metoda elementów skończonych dla stacjonarnego opływu przeszkody

Praca licencjacka
na kierunku MATEMATYKA

Praca wykonana pod kierunkiem
dr. Piotra Krzyżanowskiego
Instytut Matematyki Stosowanej i Mechaniki

Warszawa, sierpień 2022

Streszczenie

W pracy numerycznie rozwiązano stacjonarne równanie Stokesa w przypadku opływu zestawu prostokątnych przeszkód w prostokątnym kanale, wykorzystując w tym celu metodę elementów skończonych (MES) . Do konstrukcji MES wykorzystano klasyczny element Taylora–Hooda na siatce trójkątnej, implementując go następnie w języku Python 3, z wykorzystaniem jedynie podstawowych bibliotek numerycznych. Wykonane eksperymenty pokazują, że w rozważanym przypadku rząd schematu jest liniowy, a koszt rozwiązania jest superliniowy i niekwadratowy.

Słowa kluczowe

metoda elementów skończonych, równania Stokesa, implementacja

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.1 Matematyka

Klasyfikacja tematyczna

Numerical analysis
65-04 Software, source code
65Z05 Applications to the sciences
Fluid mechanics
76D07 Stokes and related (Oseen, etc.) flows

Tytuł pracy w języku angielskim

Finite element method for stationary flow around obstacle

Spis treści

Wprowadzenie

W pracy rozważać będziemy stacjonarny przepływ lepkiego i nieściśliwego płynu w prostokątnym obszarze z przeszkodami $\Omega \subset R^2$, który jest opisywany przez układ równań Stokesa:

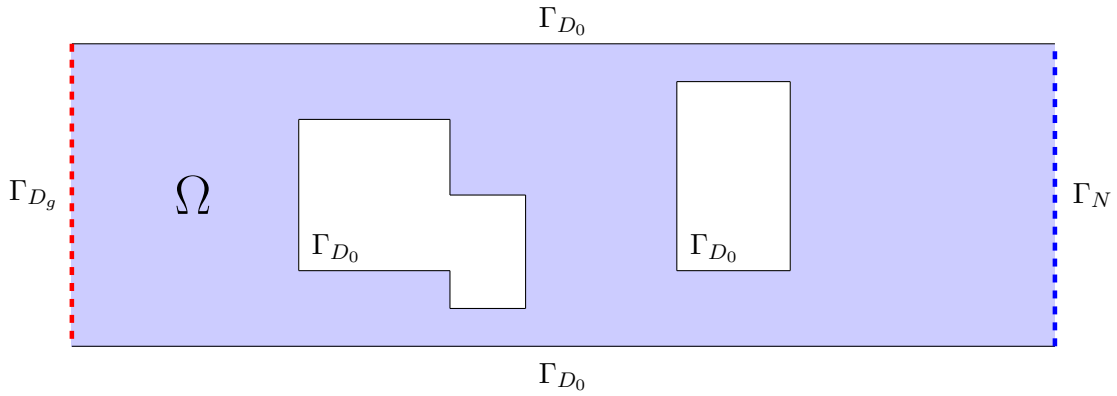
$$\nabla p - \nu \Delta u = f \qquad \text{w } \Omega, \tag{1a}$$

$$\nabla \cdot u = 0 \qquad \text{w } \Omega, \tag{1b}$$

$$u = g_D \qquad \text{na } \Gamma_D, \tag{1c}$$

$$\nu n \cdot \nabla u - pn = 0 \qquad \text{na } \Gamma_N. \tag{1d}$$

Danymi zadania są: lepkość $\nu > 0$, siła zewnętrzna $f : \Omega \to R$ oraz $g_D : \Gamma_D \to R$ funkcja określająca prędkość na części brzegu $\Gamma_D \subset \partial\Omega$, gdzie $\partial\Omega = \Gamma_D \cup \Gamma_N$.



Rysunek 0: Przykładowy obszar z przeszkodami Ω, z widocznym brzegiem. Γ_N to niebieska część brzegu - odpływ. Γ_{D_g} to czerwony fragment - dopływ. Pozostałą część brzegu $\partial\Omega$, w kolorze czarnym, stanowi Γ_{D_0}.

Natomiast szukane to funkcja skalarna ciśnienia $p(x)$ oraz pole wektorowe prędkości $u(x) = [u_1(x), u_2(x)]$, określone na $\bar{\Omega}$.

W równaniach użyliśmy notacji:

$\Delta u(x) = [\frac{\partial^2 u_1(x)}{\partial x_1^2} + \frac{\partial^2 u_1(x)}{\partial x_2^2}, \frac{\partial^2 u_2(x)}{\partial x_1^2} + \frac{\partial^2 u_2(x)}{\partial x_2^2}]$,

$\nabla \cdot u(x) = \frac{\partial u_1(x)}{\partial x_1} + \frac{\partial u_2(x)}{\partial x_2}$,

$\nabla u(x) = \begin{bmatrix} \frac{\partial u_1(x)}{\partial x_1} & \frac{\partial u_1(x)}{\partial x_2} \\ \frac{\partial u_2(x)}{\partial x_1} & \frac{\partial u_2(x)}{\partial x_2} \end{bmatrix}$,

n - zewnętrzny wektor normalny do $\partial\Omega$.

W rozważaniach brzeg omawianego obszaru z przeszkodami $\partial\Omega$ rozdzielamy na dwa rozłączne zbiory Γ_N i Γ_D, widoczne na rys. 0, gdzie Γ_N jest prawą ścianą kanału, Γ_{D_g} jest lewą

ścianą i kładziemy $\Gamma_{D_0} = \partial\Omega \setminus (\Gamma_N \cup \Gamma_{D_g})$, przy czym zakładać będziemy, że funkcja g_D jest stale równa 0 na Γ_{D_0}.

W pracy zrealizujemy konstrukcję MES dla (1) przy powyższych ograniczeniach i przeprowadzimy pełną implementację MES oraz zbadamy jakość solvera.

W rozdziale pierwszym wyprowadzimy sformułowanie wariacyjne dla omawianego problemu oraz przedstawimy twierdzenie o aproksymacji. W rozdziale drugim omówimy sposób implementacji MES dla tego zadania, zwracając uwagę na wykorzystanie tablic numpy i macierzy rzadkich. W rozdziale trzecim przeprowadzimy eksperymenty numeryczne, w których zbadamy podstawowe własności algorytmu oraz implementacji. W ostatnim rozdziale przedstawimy podsumowanie.

Rozdział 1

Konstrukcja MES dla równania Stokesa

W tym rozdziale przeprowadzimy, w oparciu o [1], konstrukcję MES dla (1). Na początku doprowadzimy nasze równania do postaci wariacyjnej, a następnie wprowadzimy jej dyskretyzację za pomocą elementów skończonych. Pod koniec rozdziału przytoczymy twierdzenie o błędzie aproksymacji rozwiązania.

1.1. Sformułowanie wariacyjne

Przy założeniu, że funkcje u i p są dostatecznie gładkie, pomnóżmy (1a) przez funkcje testową $v \in V_0$ oraz (1b) przez $q \in Q$, gdzie
$V_0 = \{v \in [C^\infty(\Omega)]^2 : v|_{\Gamma_D} = 0\}$,
$Q = \{q \in C^\infty(\Omega)\}$.
Następnie oba równania całkujemy obustronnie. Z (1a) dostajemy tożsamość

$$\int_\Omega \nabla p \cdot v \, d\Omega - \nu \int_\Omega \Delta u \cdot v \, d\Omega = \int_\Omega f \cdot v \, d\Omega \qquad \forall v \in V_0, \tag{2a}$$

gdzie $\cdot$ oznacza iloczyn skalarny.
Zgodnie z [2] korzystamy z twierdzenia Greena dla dywergencji

$$\int_\Omega b\nabla \cdot a \, d\Omega = -\int_\Omega \nabla b \cdot a \, d\Omega + \int_{\partial\Omega} bn \cdot a \, d\gamma, \tag{3a}$$

gdzie $a : R^2 \to R^2$ i $b : R \to R$, tż. $a \in C^1(R^2)$ i $b \in C^1(R)$. Regularność jest spełniona dla $b = p$ i $a = v$, stąd przekształcamy (2a) do postaci

$$\begin{aligned} -\int_\Omega p\nabla \cdot v \, d\Omega + \int_{\partial\Omega} pn \cdot v \, d\gamma - \nu \int_\Omega \Delta u \cdot v \, d\Omega = \\ = \int_\Omega f \cdot v \, d\Omega \qquad \forall v \in V_0. \end{aligned} \tag{4a}$$

Na mocy tożsamości

$$\nabla \cdot (v\nabla u) = v \cdot \Delta u + \nabla v : \nabla u,$$

gdzie $\nabla v : \nabla u = \sum_{i=1}^2 \frac{\partial v}{\partial x_i} \frac{\partial u}{\partial x_i}$, $v \in C^1(R^2)$ oraz $u \in C^2(R^2)$, podstawiamy $b = 1$ i $a = v\nabla u$ w (3a), spełniając założenia dostatecznej regularności i otrzymujemy

$$-\nu \int_\Omega \Delta u \cdot v \, d\Omega = \nu \int_\Omega \nabla u : \nabla v \, d\Omega - \nu \int_{\partial\Omega} n \nabla u \cdot v \, d\gamma,$$

skąd (4a) jest równoważna

$$\begin{aligned} \nu \int_\Omega \nabla u : \nabla v \, d\Omega - \int_\Omega p \nabla \cdot v \, d\Omega - \int_\Omega f \cdot v \, d\Omega = \\ = \int_{\partial\Omega} (\nu n \nabla u - pn) \cdot v \, d\gamma \qquad \forall v \in V_0. \end{aligned} \tag{5a}$$

Ponieważ zachodzi (1d) oraz $v \in V_0$ to (5a) możemy zapisać jako

$$\nu \int_\Omega \nabla u : \nabla v \, d\Omega - \int_\Omega p \nabla \cdot v \, d\Omega = \int_\Omega f \cdot v \, d\Omega \qquad \forall v \in V_0. \tag{6a}$$

Z kolei po przemnożeniu (1b) przez q oraz obustronnym scałkowaniu, otrzymujemy tożsamość

$$\int_\Omega \nabla \cdot u q \, d\Omega = 0 \qquad \forall q \in Q. \tag{6b}$$

Na tej podstawie, por. [1, rozdz. 12.2.2] oraz [2, rozdz. 16.1], możemy zdefiniować wariacyjną postać (1) dla znacznie szerszych przestrzeni funkcyjnych, ponieważ formalnie (6) ma sens również, gdy $V_0 = \{v \in [H^1(\Omega)]^2 : v|_{\Gamma_D} = 0\}$, $V = \{u \in [H^1(\Omega)]^2 : u|_{\Gamma_D} = g_D\}$ oraz $Q = \{q \in L^2(\Omega)\}$.

Definicja 1 (Sformułowanie wariacyjne (1)) *Znajdź $(u, p) \in V \times Q$, tż.*

$$\nu \int_\Omega \nabla u : \nabla v \, d\Omega - \int_\Omega p \nabla \cdot v \, d\Omega = \int_\Omega f \cdot v \, d\Omega \qquad \forall v \in V_0, \tag{7a}$$

$$\int_\Omega \nabla \cdot u q \, d\Omega = 0 \qquad \forall q \in Q, \tag{7b}$$

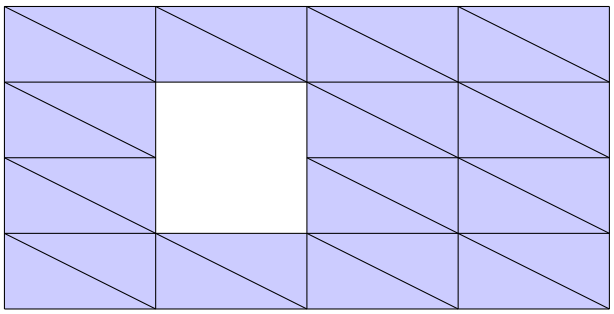
gdzie
$V_0 = \{v \in [H^1(\Omega)]^2 : v|_{\Gamma_D} = 0\}$,
$V = \{u \in [H^1(\Omega)]^2 : u|_{\Gamma_D} = g_D\}$,
$Q = \{q \in L^2(\Omega)\}$.

1.2. Dyskretyzacja MES

Aby wyznaczyć przybliżone rozwiązanie (7) zastąpimy je pewnym zadaniem postawionym w przestrzeni skończenie wymiarowej. Triangulacją $\mathcal{K}_h$ obszaru Ω nazwiemy zbiór trójkątów prostokątnych K, tż. $\Omega = \cup_{K \in \mathcal{K}_h} K$, przy czym przecięcie dowolnych dwóch trójkątów jest krawędzią, wierzchołkiem lub jest puste. Parametrem określającym gęstość siatki jest $h = \max_{\{K \in \mathcal{K}_h\}} h_K$. Dla ułatwienia implementacji, ograniczymy się do sytuacji, gdy obszar Ω możemy pokryć jednorodną siatką prostokątną. Wówczas $\mathcal{K}_h$ konstruujemy poprzez podział prostokątów wzdłuż ustalonych przekątnych zob. rys. 1.1.

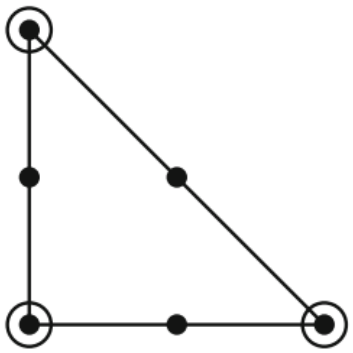
Na każdym z trójkątów określamy przestrzeń funkcji liniowych
$P_1(K) = \{v : v = c_0 + c_1 x_1 + c_2 x_2, (x_1, x_2) \in K, c_0, c_1, c_2 \in R\}$
oraz przestrzeń funkcji kwadratowych

$P_2(K) = \{v : v = c_0 + c_1 x_1 + c_2 x_2 + c_3 x_1^2 + c_4 x_1 x_2 + c_5 x_2^2,$
$(\mathrm{x}_1, x_2) \in K, c_0, c_1, c_2, c_3, c_4, c_5 \in R\}$.



Rysunek 1.1: Przykład siatki dla obszaru z przeszkodą.

Następnie definiujemy V_h i Q_h, przestrzenie funkcji ciągłych na $\mathcal{K}_h$ tż.
$V_h = \{u_h \in [C^0(\Omega)]^2 : u_h|_K \in [P_2(K)]^2\}$,
$Q_h = \{p_h \in C^0(\Omega) : p_h|_K \in P_1(K)\}$.
Taki wybór przestrzeni nosi nazwę elementu Taylora–Hooda [1, rozdz. 12.2.7.1]. Warto w tym miejscu zaznaczyć, że wybór $V_h = \{u_h \in [C^0(\Omega)]^2 : u_h|_K \in [P_1(K)]^2\}$, przy Q_h jak powyżej, prowadziłby do problemów numerycznych [3, Finite Elements for the Stokes Problem, s.53].



Rysunek 1.2: Węzły prędkości • i ciśnienia ∘ dla elementu Taylora–Hooda. Źródło: [1].

Na koniec definiujemy przybliżone zagadnienie wariacyjne dla (7): znajdź $(u_h, p_h) \in V_{h_g} \times Q_h$, tż.

$$\nu \int_\Omega \nabla u_h : \nabla v \, d\Omega - \int_\Omega p_h \nabla \cdot v \, d\Omega = \int_\Omega f \cdot v \, d\Omega \qquad \forall v \in V_{h_0}, \tag{8a}$$

$$\int_\Omega \nabla \cdot u_h q \, d\Omega = 0 \qquad \forall q \in Q_h, \tag{8b}$$

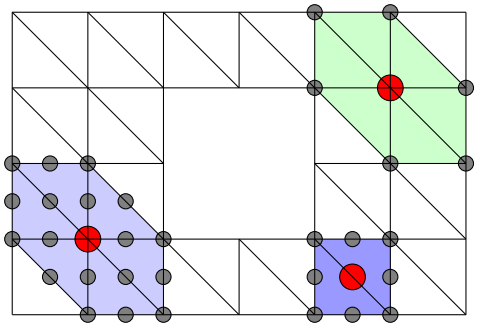
gdzie $V_{h_g} = \{u_h \in V_h : u_h|_{\Gamma_D} = g_D\}$ oraz $V_{h_0} = \{v \in V_h : v|_{\Gamma_D} = 0\}$.
W poniższych rozważaniach będziemy odwoływać się do bazy przestrzeni V_h, danej jako zbiór funkcji kawałkami wielomianowych 2 stopnia $\{\Phi_i\}_{i=1}^{n_u}$, gdzie $\Phi_i \in V_h$ tż.

$$\Phi_i(N_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

oraz bazy przestrzeni Q_h, czyli zbioru funkcji kawałkami liniowych $\{\phi_k\}_{k=1}^{n_p}$, gdzie $\phi_k \in Q_h$ tż.

$$\phi_k(M_l) = \begin{cases} 1 & k = l \\ 0 & k \neq l \end{cases},$$

gdzie $\{N_j\}_{j=1}^{n_u}$ to zbiór węzłów prędkości, a $\{M_l\}_{l=1}^{n_p}$ to zbiór węzłów ciśnienia zob. rys. 1.2. Na poniższym rysunku zaznaczono nośniki przykładowych funkcji z powyższych zbiorów.



Rysunek 1.3: Przykłady nośników funkcji bazowych przestrzeni V_h i Q_h. Kolorami niebieskimi zaznaczono nośniki funkcji z V_h, na zielono z Q_h. W czerwonych węzłach funkcje są równe 1, w pozostałych są równe 0.

Zauważmy, że (8) jest równoważne

$$\nu \int_\Omega \nabla u_h : \nabla \Phi_i \, d\Omega - \int_\Omega p_h \nabla \cdot \Phi_i \, d\Omega = \int_\Omega f \cdot \Phi_i \, d\Omega \qquad \forall i \in \{1, ..., n_u\}, \tag{9a}$$

$$\int_\Omega \nabla \cdot u_h \phi_j \, d\Omega = 0 \qquad \forall j \in \{1, ..., n_p\}. \tag{9b}$$

Zapisując u_h i p_h jako kombinacje liniowe funkcji bazowych

$$u_h = [\sum_{i=1}^{n_u} \xi_i^{(1)} \Phi_i(x), \sum_{i=1}^{n_u} \xi_i^{(2)} \Phi_i(x)], \tag{10a}$$

$$p_h = \sum_{j=1}^{n_p} \gamma_j \phi_j(x), \tag{10b}$$

wystarczy znaleźć

$$\xi^{(1)} = \begin{bmatrix} \xi_1^{(1)} \\ \xi_2^{(1)} \\ \vdots \\ \xi_{n_u}^{(1)} \end{bmatrix}, \; \xi^{(2)} = \begin{bmatrix} \xi_1^{(2)} \\ \xi_2^{(2)} \\ \vdots \\ \xi_{n_u}^{(2)} \end{bmatrix} \tag{11a}$$

oraz

$$\gamma = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n_p} \end{bmatrix}, \tag{11b}$$

tż. tożsamość (9) jest spełniona.

Podstawiając (10) do (9) otrzymujemy układ $2n_u + n_p$ równań liniowych z $2n_u + n_p$ niewiadomymi

$$\left.\begin{aligned}
&\nu \sum_{k=1}^{n_u} \xi_k^{(1)} \underbrace{\int_\Omega \nabla\Phi_k : \nabla\Phi_i \, d\Omega}_{\mathrm{A}_{ki}} - \sum_{s=1}^{n_p} \gamma_s \underbrace{\int_\Omega \phi_s \frac{\partial \Phi_i}{\partial x_1} \, d\Omega}_{\mathrm{D}_{1si}} = \underbrace{\int_\Omega f \cdot \Phi_i \, d\Omega}_{\mathrm{b}_i}, \\
&\nu \sum_{k=1}^{n_u} \xi_k^{(2)} \underbrace{\int_\Omega \nabla\Phi_k : \nabla\Phi_i \, d\Omega}_{\mathrm{A}_{ki}} - \sum_{s=1}^{n_p} \gamma_s \underbrace{\int_\Omega \phi_s \frac{\partial \Phi_i}{\partial x_2} \, d\Omega}_{\mathrm{D}_{2si}} = \underbrace{\int_\Omega f \cdot \Phi_i \, d\Omega}_{\mathrm{b}_i} \\
&\qquad\qquad \forall i \in \{1, ..., n_u\},
\end{aligned}\right\} \quad (12a)$$

$$\sum_{k=1}^{n_u} \xi_k^{(1)} \underbrace{\int_\Omega \frac{\partial \Phi_k}{\partial x_1} \phi_j \, d\Omega}_{\mathrm{D}_{1jk}} + \sum_{k=1}^{n_u} \xi_k^{(2)} \underbrace{\int_\Omega \frac{\partial \Phi_k}{\partial x_2} \phi_j \, d\Omega}_{\mathrm{D}_{2jk}} = 0 \qquad \forall j \in \{1, ..., n_p\}. \quad (12b)$$

Oznaczając $A := [A_{ij}]$ (tzw. macierz sztywności), $D_1 := [D_{1ik}]$, $D_2 := [D_{2ik}]$ (podmacierze dywergencji), $b := [b_i]$ (wektor obciążeń), gdzie $i, j \in \{1, ..., n_u\}$ oraz $k \in \{1, ..., n_p\}$, układ (12) można równoważnie zapisać w postaci macierzowej

$$\begin{bmatrix} \nu A & 0 & -D_1 \\ 0 & \nu A & -D_2 \\ -D_1^T & -D_2^T & 0 \end{bmatrix} \begin{bmatrix} \xi^{(1)} \\ \xi^{(2)} \\ \gamma \end{bmatrix} = \begin{bmatrix} b \\ b \\ 0 \end{bmatrix} \quad (13)$$

Przytoczmy twierdzenie o aproksymacji dla układu równań Stokesa z warunkiem brzegowym Dirichleta na całym $\partial\Omega$ [4, rozdz. 2, tw. 4.3.].

Twierdzenie 1 *Niech $\Omega \subset R^2$ będzie ograniczonym prostokątem i niech (u, p) będzie rozwiązaniem (1) tż. $u \in [H^3(\Omega) \cap H_0^1(\Omega)]^2$, $p \in H^2(\Omega) \cup L_0^2(\Omega)$, gdzie $H_0^1 = \{v \in H^1 : v|_{\partial\Omega} = 0\}$ i $L_0^2 = \{w \in L^2 : \bar{w} = 0\}$ oraz $\partial\Omega = \Gamma_D$, $\Gamma_N = \emptyset$. Niech $\mathcal{K}_h$ będzie triangulacją obszaru Ω jak w rozdziale 1.2. oraz niech (u_h, p_h) będzie rozwiązaniem (8), wtedy*

$$||u - u_h||_{H^1} + ||p - p_h||_{L^2} \leq Ch^2(|u|_{H^3} + |p|_{H^2}),$$

gdzie dla $k \in \{1, 2, 3\}$ oznaczamy
$||u||_{H^k} = (\sum_{j=0}^k \int_\Omega |\nabla^j u_1|^2 \, d\Omega)^{1/2} + (\sum_{j=0}^k \int_\Omega |\nabla^j u_2|^2 \, d\Omega)^{1/2}$,
$||p||_{H^k} = (\sum_{j=0}^k \int_\Omega |\nabla^j p|^2 \, d\Omega)^{1/2}$,
$|u|_{H^k} = (\int_\Omega |\nabla^k u_1|^2 \, d\Omega)^{1/2} + (\int_\Omega |\nabla^k u_2|^2 \, d\Omega)^{1/2}$,
$|p|_{H^k} = (\int_\Omega |\nabla^k p|^2 \, d\Omega)^{1/2}$
oraz $(\nabla u_i)^2 = (\nabla u_i) \cdot (\nabla u_i)$,
$\nabla^k u_i = [\frac{\partial^k u_i}{\partial x_1^k}, \frac{\partial^k u_i}{\partial x_2^k}]$,
h - globalny rozmiar siatki i $C > 0$ niezależne od h, u i p.

W rozdziale trzecim sprawdzimy eksperymentalnie, czy poprzez pominięcie założeń dotyczących obszaru oraz przestrzeni rozwiązań, powyższe twierdzenie możemy stosować również w rozważanym w pracy przypadku.

Rozdział 2

Implementacja

2.1. Struktura projektu i biblioteki

Przedstawioną metodę aproksymacji rozwiązań (1) zaimplementowano w języku Python 3.9.7. Struktura projektu została oparta o skomentowane, interaktywne notebooki Jupyter [5]. Projekt można podzielić na 4 etapy:

- utworzenie triangulacji, siatki węzłów dla ciśnienia i siatki węzłów dla prędkości,
- konstrukcja układu równań (13),
- obliczenie rozwiązania (13),
- postprocessing (wizualizacja, obliczenie błędów rozwiązania).

Pierwsze dwa etapy wykorzystały koncepcje opisane w [1]. Pozostałe komponenty zrealizowano w oparciu o własne pomysły. Program został uzupełniony o testy własności macierzy masy i sztywności.

Kod jest samodzielną implementacją MES od podstaw, z wykorzystaniem bibliotek numerycznych ogólnego użytku:

- NumPy [6] - dostarcza narzędzia do obliczeń numerycznych opartych na wektorach i macierzach,
- SciPy [7] - wprowadza klasę macierzy rzadkich

oraz biblioteki Matplotlib [8], wspomagającej wizualizacje.

2.2. Uwagi implementacyjne

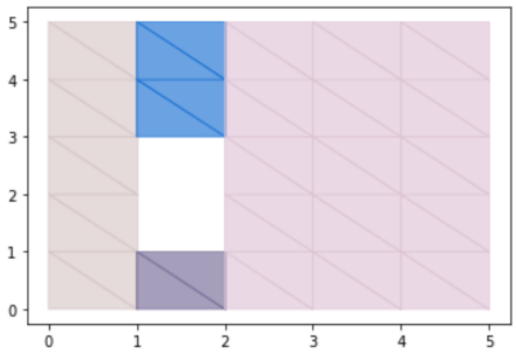
2.2.1. Tablice biblioteki NumPy

Biblioteka NumPy pozwala na użycie tablic wielowymiarowych numpy array. Jest to lepsze rozwiązanie od podstawowych list języka Python, gdyż znamy szacowaną liczbę elementów do przechowania. Zaletą numpy array jest szybkość dostępu do danych. Implementując elementy skończone wielokrotnie będziemy przeprowadzać działania arytmetyczne na macierzach (tablicach dwuwymiarowych) i wektorach (tablicach jednowymiarowych). Wykorzystując tablice numpy stronimy od wykorzystania pętli, zamiast tego operacje wykonujemy na wektorach.

2.2.2. Utworzenie triangulacji i siatek

W implementacji ograniczyliśmy sie do triangulacji jednorodnej, konstruowanej w określony sposób. Zakładając, że obszar z przeszkodami daje się podzielić na jednakowe prostokąty o danych wymiarach. Następnie prostokąty rozdzielamy na 2 trójkąty wzdłuż ustalonej

przekątnej. Trójkątne elementy siatki numerujemy wewnątrz wyznaczonych podobszarów, kolumnami od lewej do prawej i wierszami od dołu do góry.



Rysunek 2.1: Siatka z zaznaczonymi podobszarami.

Podobszary w algorytmie są wyznaczane na podstawie rzutów wierzchołków obszaru i przeszkód na oś poziomą, a także na podstawie takich informacji jak np. czy przeszkoda jest jedyną w danej kolumnie, czy dotyka ścian kanału itp. Przykład utworzonej siatki z widocznymi podobszarami widzimy na rys. 2.1. Podobnie numeracja wierzchołków kolejnych elementów, powinna odbywać się przy z góry ustalonej konwencji, ułatwiającej implementację.

Na triangulacji tworzymy siatki węzłów w oparciu o [1, rozdz. 3.1.2]. Wykorzystujemy koncepcję macierzy punktów i połączeń. Te struktury przechowują dane o współrzędnych węzłów i informacje o krawędziach między nimi. Do utworzenia siatki dla węzłów prędkości wykorzystano ciąg technologiczny opisany w [1, rozdz. 8.2], pozwalający na utworzenie dodatkowych węzłów na siatce dla węzłów ciśnienia.

2.2.3. Konstrukcja układu równań

Macierze układu równań (13) konstruujemy odwołując się do schematu opisanego w [1, rozdz. 4.2.5, 4.6.1, 8.2.3], wykorzystując utworzone macierze punktów i połączeń. Wymiary macierzy rosną wraz ze wzrostem liczby wierzchołków i stają się zbyt duże, aby pamięć naszego komputera była w stanie przechować tablice. Jednak sposób ich konstrukcji jest oparty o funkcje bazowe przestrzeni V_h i Q_h. Nośniki tych funkcji są ograniczone do kilku lokalnych elementów siatki, patrz rys. 1.3. Ponieważ konstrukcja macierzy układu równań (13) jest oparta o całki z iloczynów par tych funkcji, a ich wspólny nośnik stanowi co najwyżej 6 elementów siatki, to przeważająca część liczb w macierzach jest zerowa.

Liczba niezerowych elementów macierzy sztywności szacuje się z góry przez $6 \cdot 6 \cdot n$, a podmacierzy dywergencji przez $3 \cdot 6 \cdot n$, gdzie n to liczba wszystkich trójkątów siatki. Oszacowania wynikają z liczby par funkcji bazowych, których wspólny nośnik zawiera pojedynczy element siatki.

Zgodnie z oszacowaniami macierze są rzadkie. W odróżnieniu od [1] tworzymy macierz w formacie csr, zaimplementowanym w bibliotece SciPy, którego atutem jest konstrukcja oparta o iterację. Zaprezentowane podejście wpływa na znaczne zredukowanie ilości potrzebnej pamięci.

Przy założeniu, że szukany wektor prędkości jest dwuwymiarowy, liczba niezerowych elementów, macierzy lewej strony układu równań (13), szacuje się przez $2 \cdot nnz(A) + 2 \cdot nnz(D_1) + 2 \cdot nnz(D_2) = 2 \cdot 6^2 \cdot n + 4 \cdot 3 \cdot 6 \cdot n = 144 \cdot n \ll (2 \cdot n_u + n_p)^2$, gdzie nnz(P) oznacza liczbę niezerowych elementów macierzy P. Informacja o wymiarach macierzy sztywności i podmacierzy

dywergencji, a w konsekwencji także o wymiarach macierzy lewej strony układu równań (13) pozwala na wykorzystanie koncepcji prealokacji, co obniża koszt algorytmu przez porzucenie konieczności przydzielania i zwalniania pamięci dla macierzy o zwiększających się wymiarach.

2.2.4. Obliczenie rozwiązania

Macierz lewej strony początkowo tworzymy w formacie lil, który wspomaga bezpośrednie przypisywanie bloków do jego struktury. Następnie format lil konwertujemy do csr, na którym operacje arytmetyczne są szybkie. Układ równań z macierzą formatu csr rozwiązujemy solverem spsolve modułu scipy.sparse.linalg. Pozwala to na porzucenie konieczności wykonywania operacji na elementach zerowych, dzięki czemu koszt jest niższy niż $O(n^3)$, co potwierdzą eksperymenty w rozdziale 3.

2.2.5. Postprocessing

W ostatnim etapie implementacji wizualizujemy rozwiązania wykorzystując bibliotekę Matplotlib. W oparciu o macierze reprezentacji obszaru, przechowujące wartości rozwiązania na węzłach, wizualizujemy obszar Ω na mapach ciepła. Poprawność wizualizacji gwarantuje zgodność koordynatów macierzy punktów z indeksami macierzy reprezentacji obszaru. Ponieważ prędkość jest wektorem, rozwiązanie wizualizujemy w postaci mapy ciepła normy euklidesowej wektora oraz pola wektorowego.

Rozdział 3

Eksperymenty numeryczne

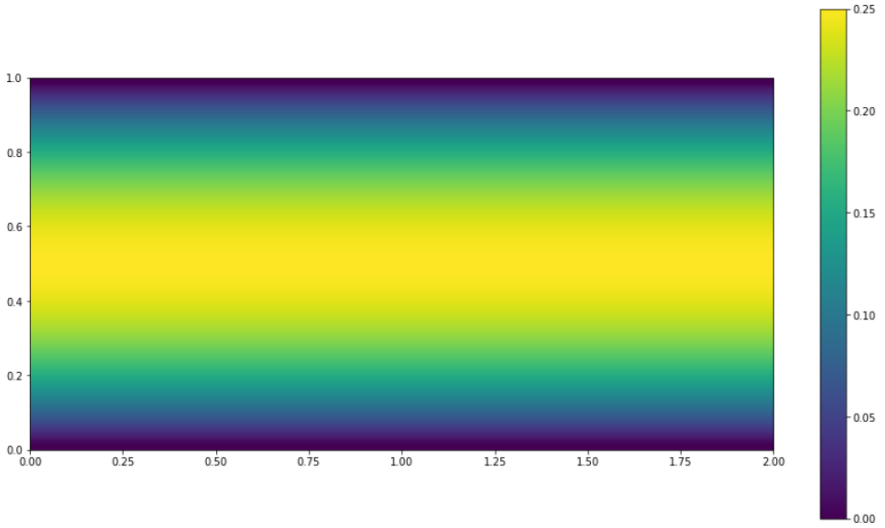
Celem niniejszego rozdziału jest przeprowadzenie numerycznych testów opracowanej implementacji. Zrealizujemy test kodu, porównując otrzymany wynik dla przepływu Poiseuille'a obszaru bez przeszkody, ze znanym rozwiązaniem dokładnym. Następnie przeprowadzimy test algorytmu dla opływu przeszkody, którego celem będzie sprawdzenie własności teoretycznych zaprogramowanego solvera, w oparciu o błędy rozwiązań dla siatek różnej gęstości. Na końcu zwizualizujemy przykładowe opływy grupy przeszkód.

3.1. Przepływ paraboliczny w prostokącie bez przeszkód

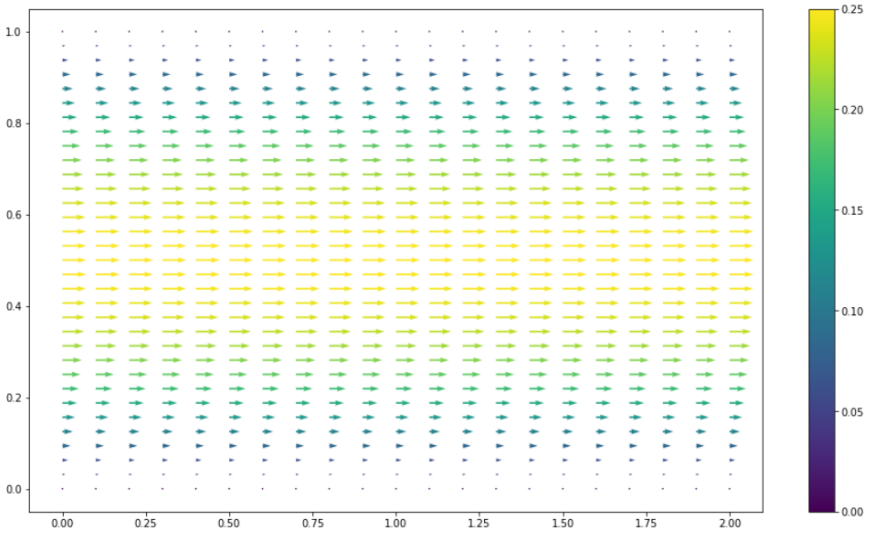
Rozważmy sytuację, gdy $\Omega = [0, 2] \times [0, 1]$, $f = 0$ oraz $u_g = x_2(2 - x_2)$ (tzw. przepływ Poiseuille'a [9, rozdz. 2.1]). Wtedy rozwiązanie (1) jest postaci $u(x_1.x_2) = [-(x_2 - a)(x_2 - b), 0]$, $p(x_1.x_2) = -2x_1 + 4$, co łatwo sprawdzić bezpośrednimi rachunkami. W szczególności p jest niezależne od x_2 oraz u jest niezależne od x_1.

Przyjmijmy $\nu = 1$. Tak postawiony problem rozwiązujemy solverem.

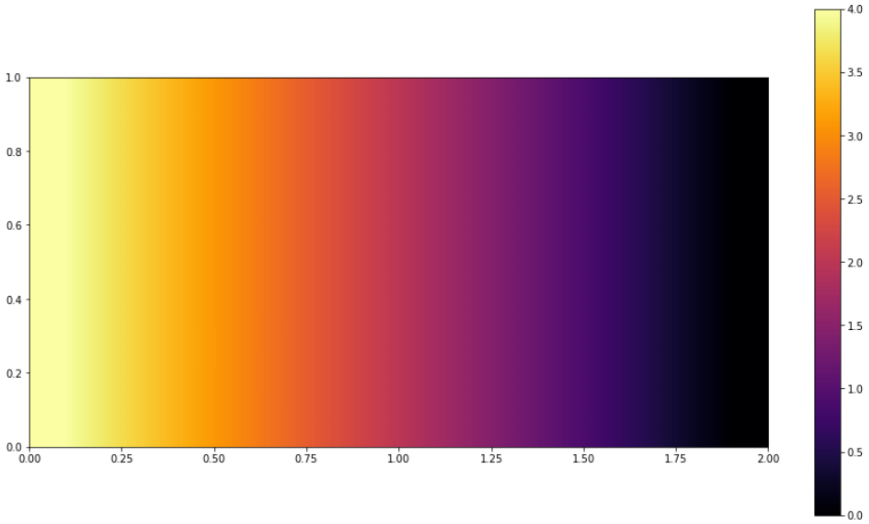
Poniższe rysunki przedstawiają wizualizacje wyników dla $\nu = 1$ i $h = 0.21$.



Rysunek 3.1: Norma euklidesowa wektora prędkości.



Rysunek 3.2: Pole wektorowe prędkości.



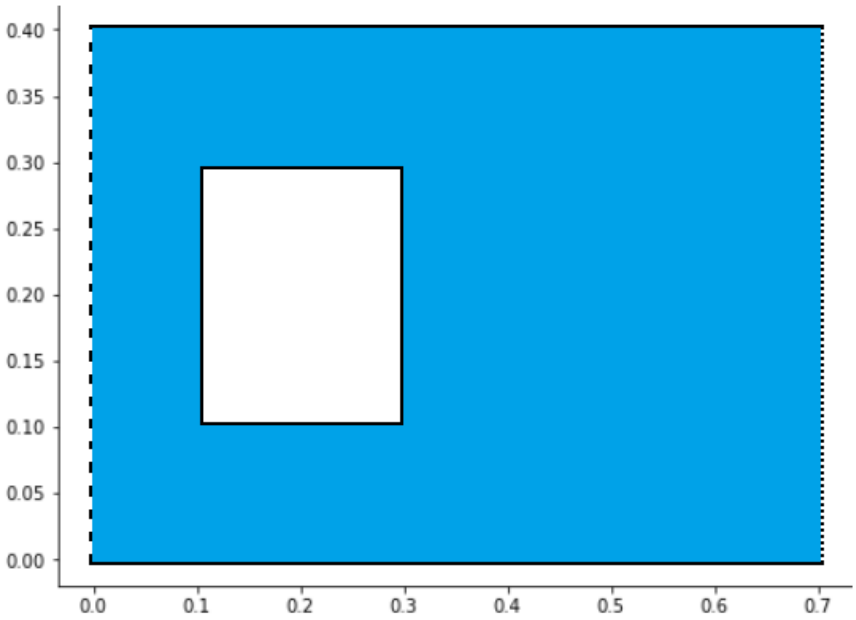
Rysunek 3.3: Ciśnienie.

Znając dokładne rozwiązanie (1) możemy obliczyć normę supremum błędu prędkości i ciśnienia liczoną w węzłach siatki:

h	$\|p - p_h\|_\infty$	$\|u - u_h\|_\infty$
1.41	$5.33{\cdot}10^{-16}$	$9.33{\cdot}10^{-15}$
0.71	$2.05 \cdot 10^{-15}$	$7.55 \cdot 10^{-15}$
0.42	$3.07 \cdot 10^{-15}$	$1.69 \cdot 10^{-14}$
0.28	$5.68 \cdot 10^{-15}$	$4.71 \cdot 10^{-14}$
0.21	$1.07 \cdot 10^{-14}$	$1.07 \cdot 10^{-13}$
0.17	$3.90 \cdot 10^{-14}$	$9.38 \cdot 10^{-13}$

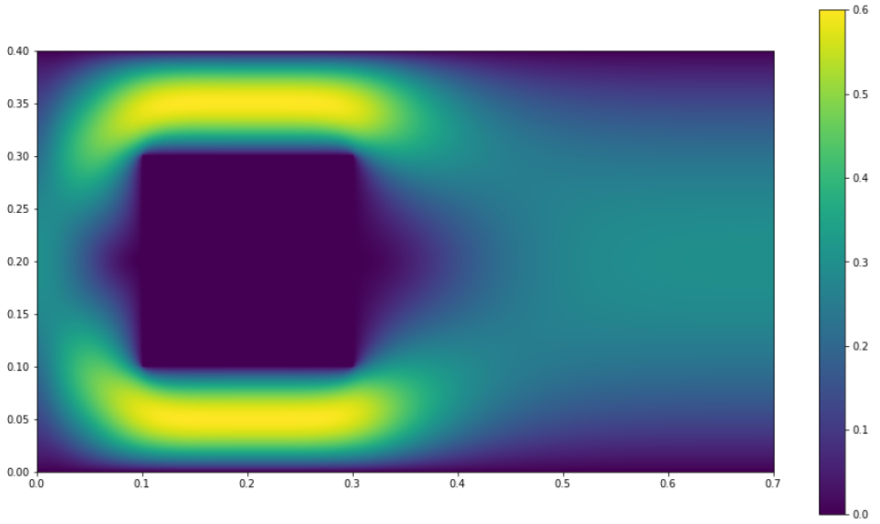
Zauważmy, że wraz ze zmniejszaniem wartości h, błędy w omawianej normie rosną. Ponadto uzyskane błędy rozwiązania są na poziomie wyższym, niż poziom precyzji arytmetyki podwójnej (10^{-16}), w której rozwiązywaliśmy postawiony problem. Wnioskujemy, że na uzyskane błędy w dużym stopniu miały wpływ błędy zaokrągleń.

3.2. Opływ przeszkody

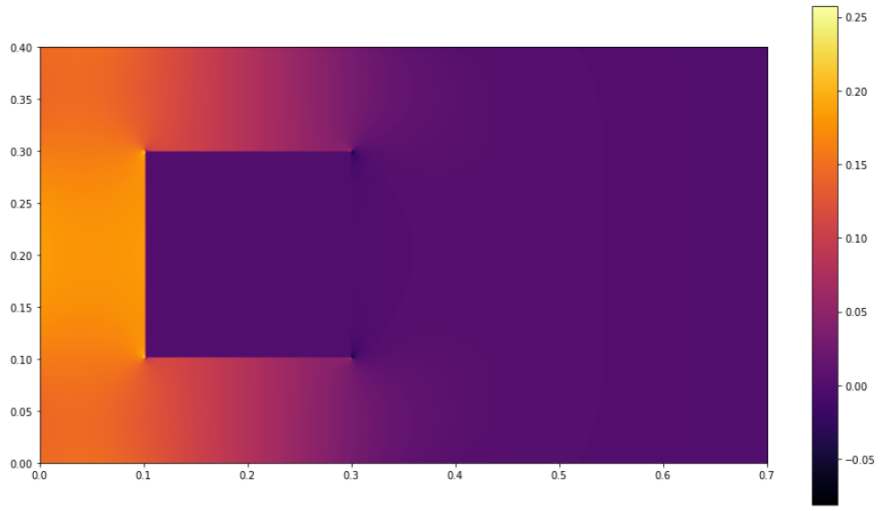


Rysunek 3.4: Obszar z przeszkodą.

Przeprowadźmy eksperyment analogiczny z symulacją zrealizowaną w [1, rozdz. 12.3.2.1] (tzw. test porównawczy DFG). Zaimplementowany generator siatek jest ograniczony do grupy przeszkód prostokątnych, dlatego zamiast przeszkody w kształcie koła, rozważymy kwadrat. Rozważmy opływ przeszkody dla $g_D(x_1, x_2) = 1.2(x_2)(0.4 - x_2)/0.4^2$, gdzie $\Omega = ([0, 0.7] \times [0, 0.4]) \setminus ([0.1, 0.3] \times [0.1, 0.3])$ jak na rys. 3.4. Aby przepływ był laminarny, lepkość w teście wynosi $\nu = 0.001$. Zwizualizujmy rozwiązanie dla $h = 0.002$:



Rysunek 3.5: Norma euklidesowa wektora prędkości.



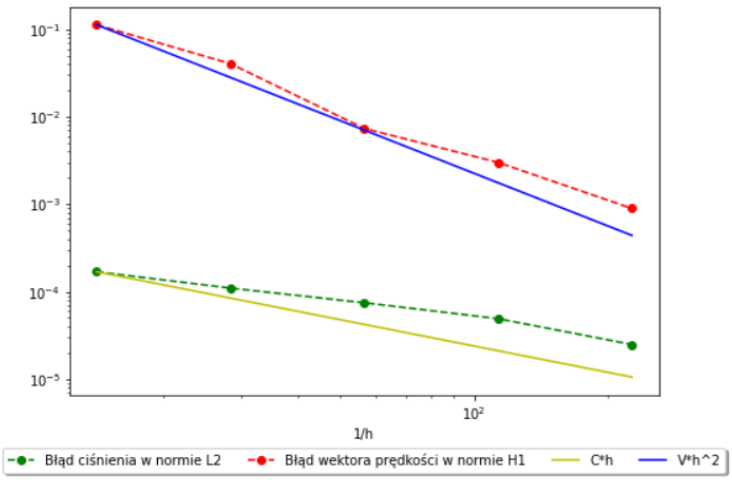
Rysunek 3.6: Ciśnienie.

Na podstawie uzyskanego rozwiązania, zbadajmy jakość uzyskanych aproksymacji. Za substytut dokładnego rozwiązania przyjmiemy rozwiązanie określone na najdrobniejszej siatce dla $h = 0.002$, o dostatecznie niskim koszcie rozwiązywania. Eksperyment nie zweryfikuje czy implementacja jest poprawna, natomiast jeśli jest, to pozwoli nam na oszacowanie rzędu zbieżności. Zauważmy, że $\lim_{h\to 0} \|p - p_h\|_Q = 0$ oraz $\lim_{h\to 0} \|u - u_h\|_V = 0$, gdzie (u,p) jest rozwiązaniem dokładnym (7). Na podstawie tej obserwacji zbadamy rząd zbieżności błędu rozwiązania w normach $\|.\|_Q$ i $\|.\|_V$, odpowiednio dla ciśnienia i prędkości. W tabeli poniżej znajduje się podsumowanie uzyskanych wyników.

h	n_p	n_u	$\|\widehat{p} - p_h\|_Q$	$\|\widehat{u} - u_h\|_V$
0.07	126	444	0.00017	0.11273
0.035	444	1656	0.00011	0.04050
0.018	1656	6384	0.00007	0.00736
0.009	6384	25056	0.00005	0.00301
0.004	25056	99264	0.00002	0.00090

Tablica 3.1: Zestawienie wyników, względem substytutu rozwiązania dokładnego $(\widehat{u}, \widehat{p})$ problemu wariacyjnego dla $h = 0.002$.

Na rys. 3.7 przedstawiono wykres norm błędów rozwiązań: ciśnienia w normie $\|.\|_{L^2}$ i wektora prędkości w normie $\|.\|_{H^1}$.



Rysunek 3.7: Porównanie norm błędów w skali podwójnie logarytmicznej. C,V stałe.

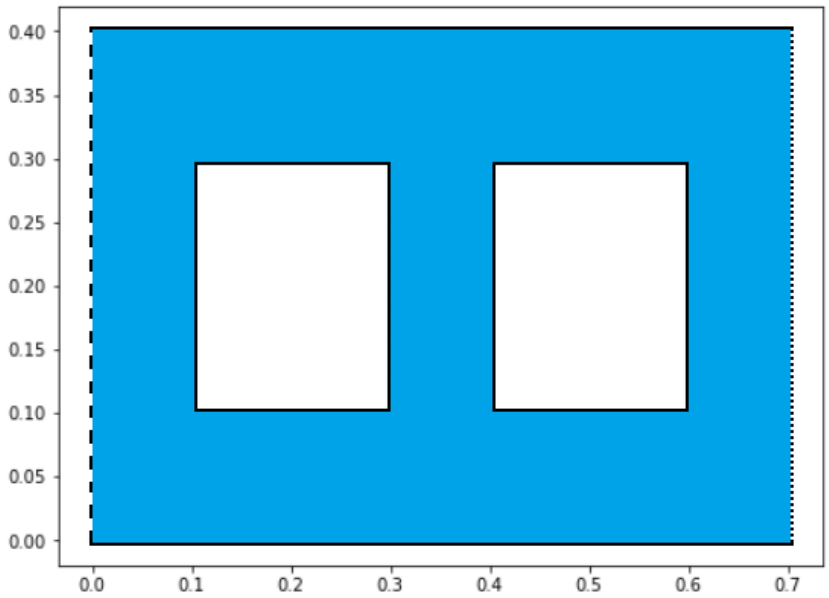
Powyższe wyniki sugerują, że błąd wektora prędkości w normie $\|.\|_{H^1}$ jest w przybliżeniu rzędu 2. Natomiast błąd ciśnienia w normie $\|.\|_{L^2}$ jest w przybliżeniu rzędu 1.

Tabela poniżej zawiera średnie rzędy zbieżności ciśnienia w normie $\|.\|_{L^2}$ i prędkości w normie $\|.\|_{H^1}$, obliczone na podstawie danych z wykresu, gdzie przez średni rząd będziemy rozumieć średnią arytmetyczną rzędów uzyskanych pomiędzy dwoma sąsiednimi węzłami, funkcji kawałkami liniowej, interpolującej funkcję danej normy błędu w węzłach.

Ciśnienie	Prędkość
0.69	1.77

Jeśli spełnione są założenia twierdzenia 1, to obie aproksymacje powinny być co najmniej rzędu 2. Na tej podstawie zakładamy, że założenia dotyczące obszaru i przestrzeni rozwiązań twierdzenia są kluczowe.

Uzyskane wyniki zestawmy z podobnym problemem dla opływu dwóch przeszkód rys. 3.8.



Rysunek 3.8: Obszar z dwiema przeszkodami.

h	n_p	n_u	$\|\widehat{p} - p_h\|_Q$	$\|\widehat{u} - u_h\|_V$
0.07	117	395	0,00032	0,13860
0.035	395	1431	0,00022	0,04949
0.018	1431	5423	0,00016	0,00897
0.009	5423	21087	0,00011	0,00367
0.004	21087	83135	0.00005	0,00109

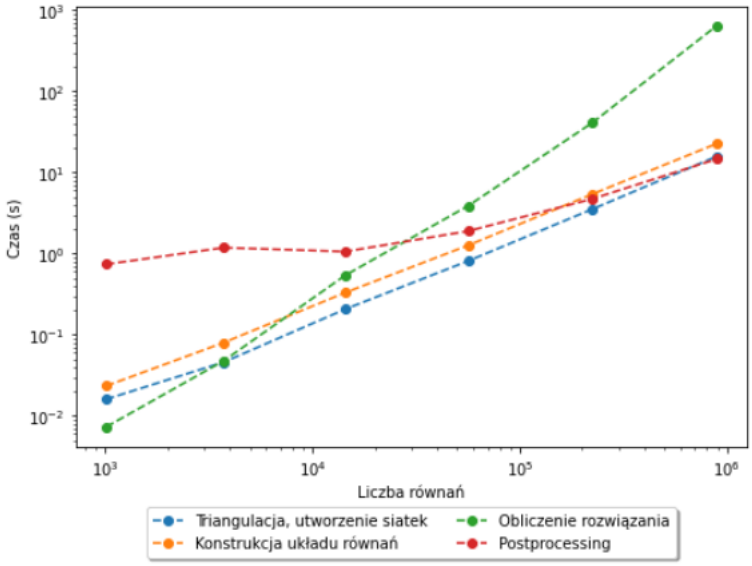
Tablica 3.2: Zestawienie wyników, względem substytutu rozwiązania dokładnego $(\widehat{u}, \widehat{p})$ problemu wariacyjnego dla dwóch przeszkód, $h = 0.002$.

Tabela poniżej zawiera średnie rzędy zbieżności ciśnienia w normie L^2 i prędkości w normie H^1, obliczone na podstawie danych z tabeli, dla opływu dwóch przeszkód.

Ciśnienie	Prędkość
0.64	1.74

Średnie rzędy zbieżności są bliskie uzyskanym w eksperymencie dla jednej przeszkody. Postulujemy, że rozmieszczenie przeszkód nie ma wpływu na uzyskiwane rzędy zbieżności.

Z powodu zbyt dużego kosztu rozwiązywania, w eksperymencie nie mogliśmy porównać otrzymanych rozwiązań do wyniku opartego na siatce gęstszej niż dla $h = 0.02$. Może to świadczyć o nieoptymalności naszej implementacji. Na rys. 3.9 zestawiono czasy obliczeń poszczególnych etapów naszego algorytmu w skali podwójnie logarytmicznej.



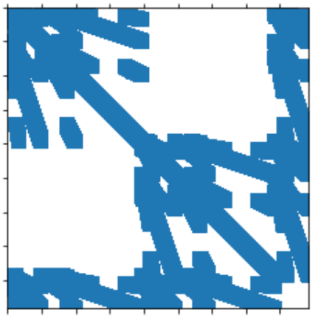
Rysunek 3.9: Porównanie czasu obliczeń poszczególnych etapów w skali podwójnie logarytmicznej.

Tabela poniżej zawiera średnie rzędy złożoności czasowej, dla kolejnych etapów, obliczone na podstawie danych z wykresu.

Etap	Średni rząd złożoności czasowej
Triangulacja, utworzenie siatek	1.01
Konstrukcja układu równań	1.01
Obliczenie rozwiązania	1.68
Postprocessing	0.44

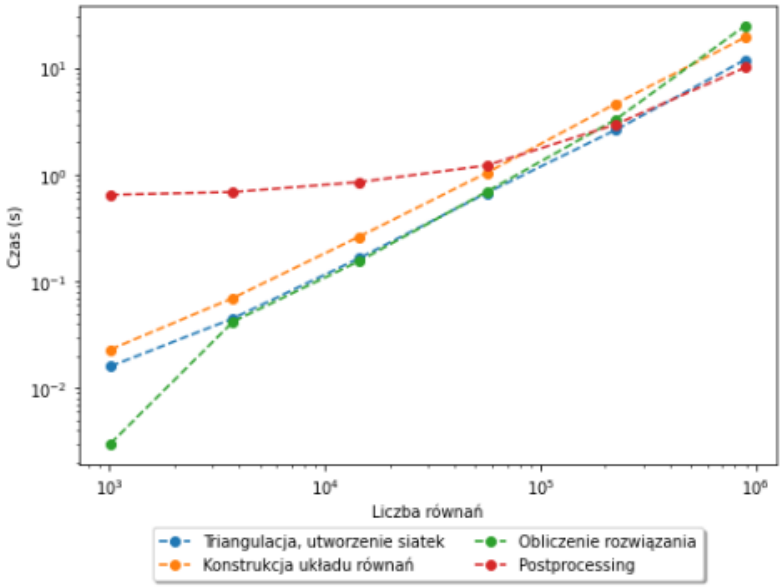
Czas rozwiązywania dwóch pierwszych etapów jest w przybliżeniu liniowy, zgodny z oczekiwaniami. Również czas ostatniego etapu jest korzystny, mniejszy niż liniowy.

Dla dostatecznie małych h, rozwiązanie (13) jest najbardziej kosztownym fragmentem algorytmu. Czas obliczeń tego etapu rośnie superliniowo, z wykładnikiem w przybliżeniu równym 1.68. Kluczowy jest fakt, że uzyskany rząd nie jest sześcienny, dzięki wykorzystaniu rzadkiej struktury macierzy. Rys. 3.10 obrazuje strukturę macierzy lewej strony układu równań (13). To macierz blokowo-taśmowa. Struktura tej macierzy nie jest dostatecznie prosta, aby solver mógł ją w pełni wykorzystać.



Rysunek 3.10: Struktura macierzy lewej strony układu równań (13) dla omawianego problemu opływu ($h = 0.002$, 889536 równań).

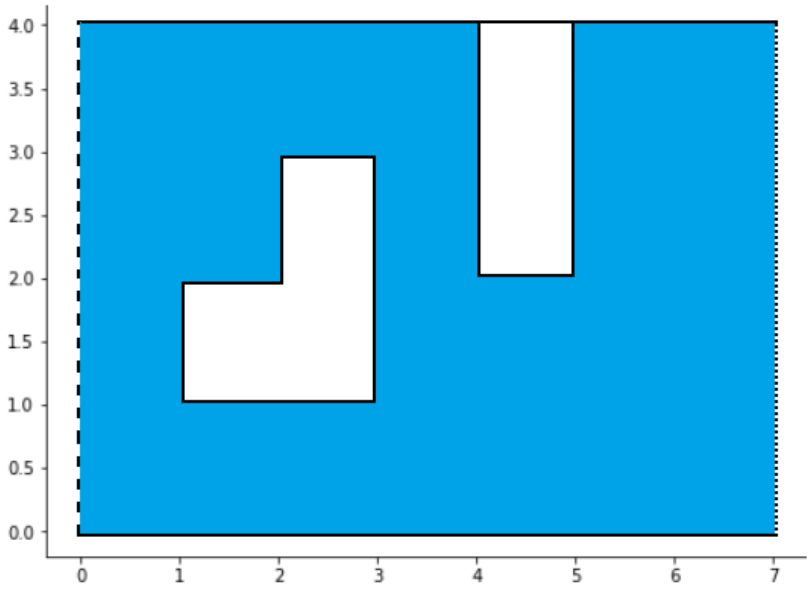
W celu zmniejszenia kosztu algorytmu, rozważono jedną z iteracyjnych metod przestrzeni Kryłowa - GMRES. Z powodu złego uwarunkowania macierzy lewej strony układu równań (13) przed użyciem GMRES do diagonali macierzy dodano niewielką stałą, w celu poprawy uwarunkowania macierzy. Metoda iteracyjna nie pozwoliła na uzyskanie dostatecznie dokładnego rozwiązania, przy zmniejszonym koszcie rozwiązywania. Warto jednak rozważyć inne metody poprawy uwarunkowania macierzy np. operatory ściskające. Natomiast wykorzystanie alternatywnego solvera bezpośredniego pypardiso.spsolve pakietu PyPardiso [10], pozwoliło na znaczne przyspieszenie algorytmu rys. 3.11, przy zachowaniu dokładności wyników. Czas obliczeń rozwiązania (13) rośnie niemal liniowo, z wykładnikiem w przybliżeniu równym 1.08.



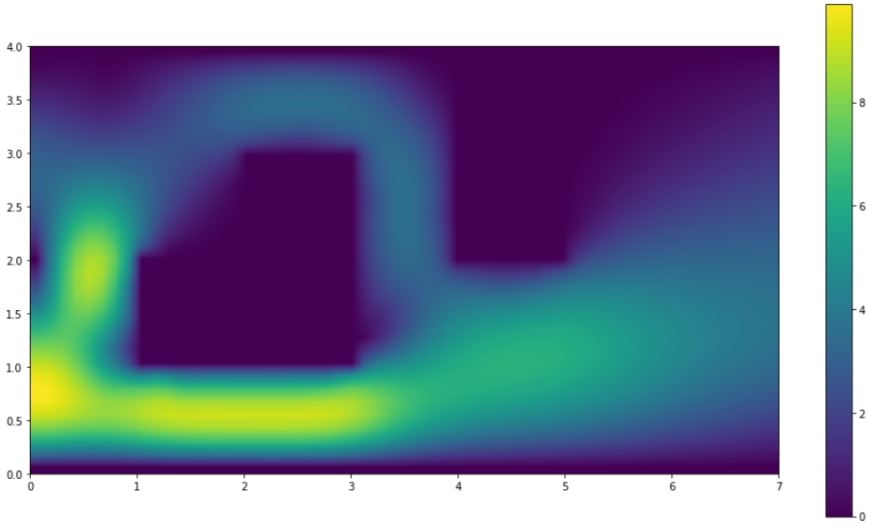
Rysunek 3.11: Przybliżony średni rząd złożoności czasowej poszczególnych etapów.

3.3. Inne przykłady

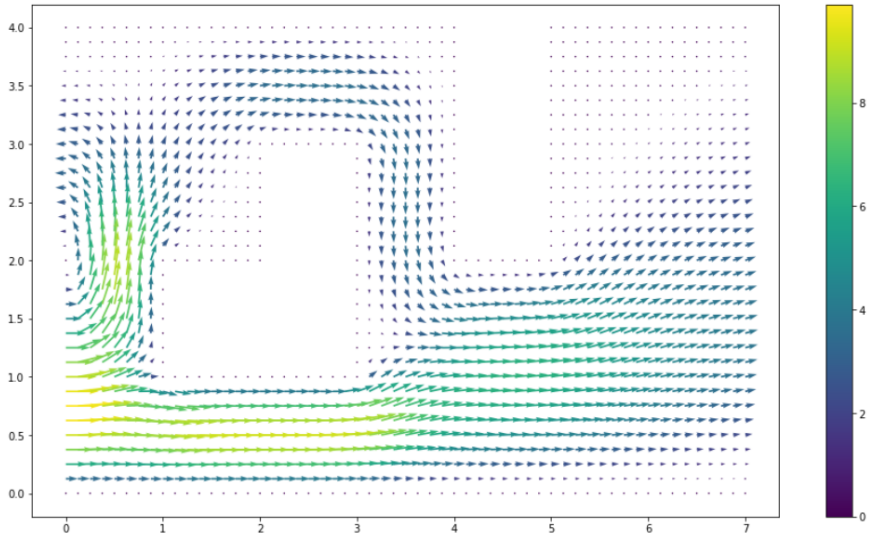
Program pozwala na rozważania opływu bardziej złożonych przeszkód. Poniżej przedstawimy wizualizację przykładowego opływu dla $g_D(x_1, x_2) = -(x_2)(x_2 - 4)^2(x_2 - 2)$, gdzie Ω jak na rys. 3.12, $\nu = 5$ i $h = 0.35$.



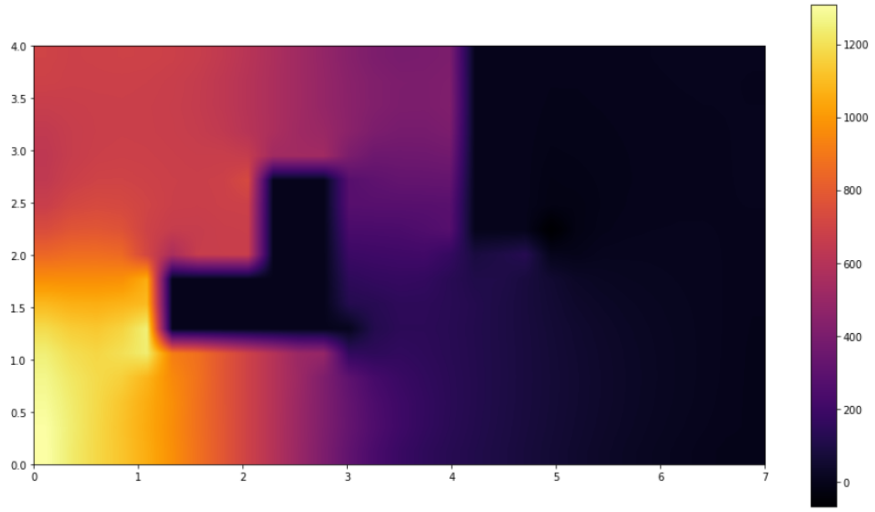
Rysunek 3.12: Obszar z przeszkodą przy ścianie kanału.



Rysunek 3.13: Norma euklidesowa wektora prędkości.

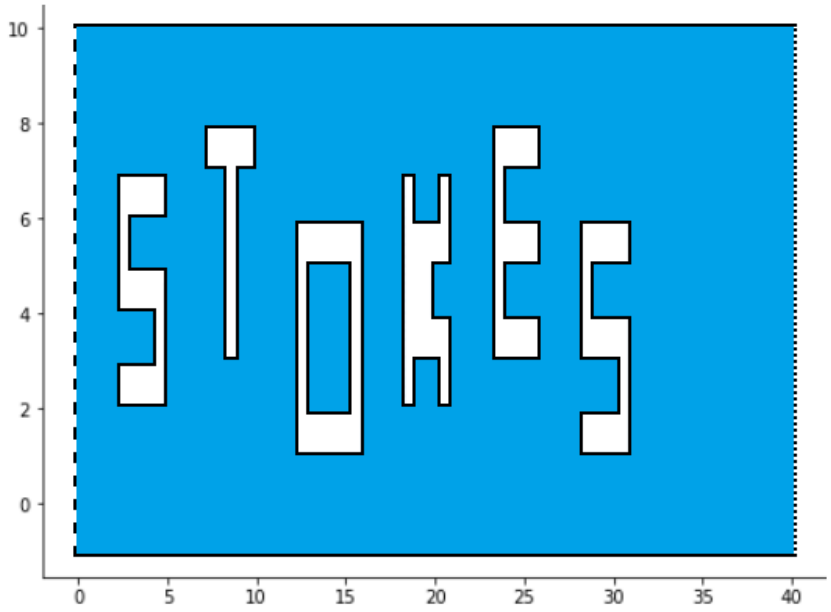


Rysunek 3.14: Pole wektorowe prędkości.

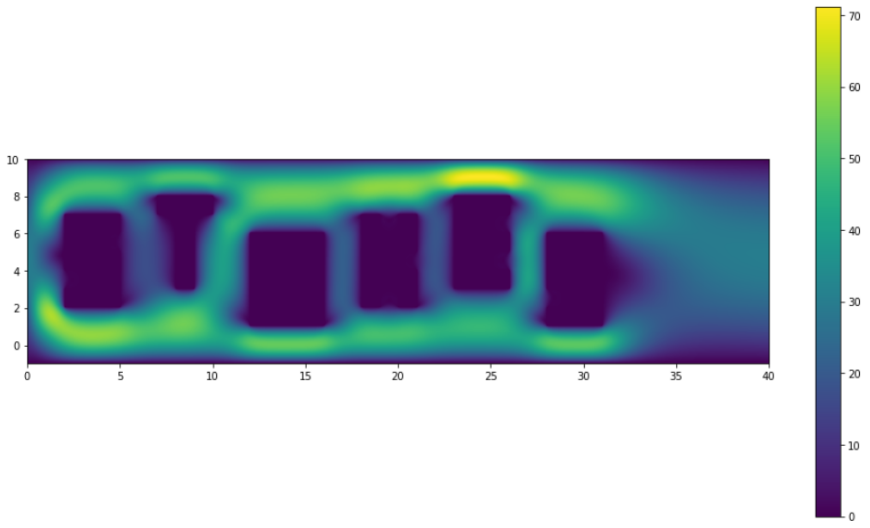


Rysunek 3.15: Ciśnienie.

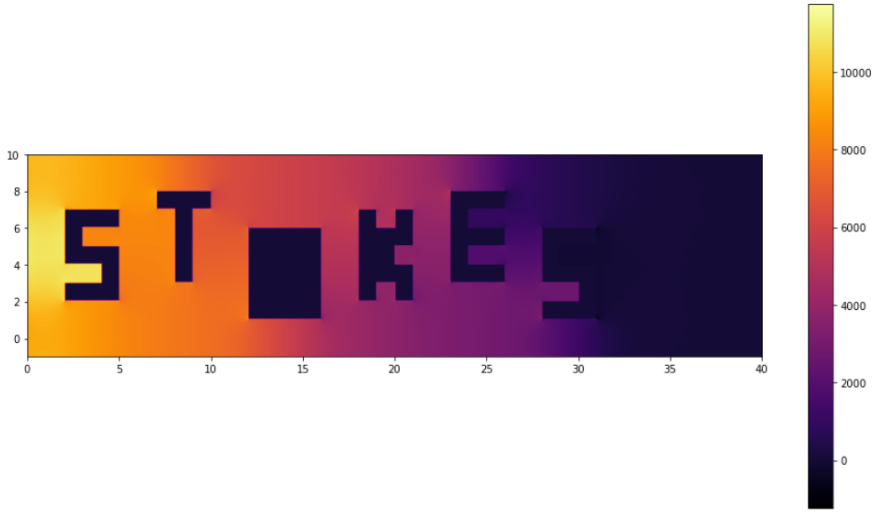
Zwizualizujmy również przypadek dla bardzo złożonej przeszkody. Niech $g_D(x_1, x_2) = (x_2 - 1)(10 - x_2)$, gdzie Ω jak na rys. 3.16, $\nu = 2$ i $h = 0.09$.



Rysunek 3.16: Obszar ze złożoną przeszkodą



Rysunek 3.17: Norma euklidesowa wektora prędkości.



Rysunek 3.18: Norma euklidesowa wektora prędkości.

Rozdział 4

Podsumowanie

W pracy przeprowadziliśmy konstrukcję MES dla stacjonarnego równania Stokesa opływu grupy przeszkód w prostokątnym obszarze oraz jej implementację od podstaw w języku Python 3.9.7, z wykorzystaniem jedynie bibliotek SciPy i Numpy. Kluczowym elementem przedstawionej implementacji okazało się wykorzystanie macierzy rzadkich, co pozwoliło na znaczne zaoszczędzenie pamięci i przyspieszenie algorytmu przy konstrukcji układu równań (13) oraz jego rozwiązywaniu.

Przeprowadziliśmy podstawową weryfikację przeprowadzonej implementacji, dokonując analizy błędów w normie supremum względem znanego rozwiązania dokładnego przepływu Poiseuille'a, uzyskując, zgodnie z oczekiwaniami, zgodność rozwiązania numerycznego i dokładnego na poziomie niewiele odbiegającym od precyzji arytmetyki. Ponadto, dla zadania z przeszkodami, przeprowadzono eksperymentalną i przybliżoną analizę rzędu błędów w normach przestrzeni L^2 i H^1. Wyniki nie zgodziły się z 1, stąd doszliśmy do wniosku, że pominięte przez nas założenia twierdzenia są kluczowe. Na koniec zinterpretowaliśmy uzyskane czasy obliczeń poszczególnych etapów w zależności od wielkości problemu, skąd wywnioskowaliśmy nieoptymalność algorytmu i zaproponowaliśmy wykorzystanie metody iteracyjnej lub solvera zaawansowanego pakietu numerycznego PyPardiso w celu zmniejszenia kosztu rozwiązania.

Największym wyzwaniem implementacji była optymalizacja kodu. Głównym problemem okazało się wykorzystywanie pętli w celu dostępu do konkretnych węzłów. Nieoptymalne rozwiązanie zostało zastąpione przez użycie tablic reprezentujących obszar, których liczby kodowały informacje o rodzaju warunku brzegowego, czy ponumerowaniu danego węzła. Duży wpływ na zmniejszenie kosztu algorytmu, miało także zastosowanie wektoryzacji i prealokacji oraz odpowiedni dobór formatów macierzy rzadkich.

Konstrukcja kodu jest modułowa i sparametryzowana z myślą o ewentualnych rozszerzeniach. Autor planuje udoskonalenie implementacji na przypadek 3D oraz równanie ewolucyjne Stokesa.

Bibliografia

[1] Mats G. Larson and Fredrik Bengzon. *The finite element method: theory, implementation, and applications*, volume 10 of *Texts in Computational Science and Engineering*. Springer, Heidelberg, 2013.

[2] Alfio Quarteroni. *Numerical models for differential problems*, volume 8 of *MS&A. Modeling, Simulation and Applications*. Springer, Milan, second edition, 2014. Translated from the fifth (2012) Italian edition by Silvia Quarteroni.

[3] Daniele Boffi, Franco Brezzi, Leszek F. Demkowicz, Ricardo G. Durán, Richard S. Falk, and Michel Fortin. *Mixed finite elements, compatibility conditions, and applications*, volume 1939 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin; Fondazione C.I.M.E., Florence, 2008. Lectures given at the C.I.M.E. Summer School held in Cetraro, June 26–July 1, 2006, Edited by Boffi and Lucia Gastaldi.

[4] Pierre-Arnaud Raviart Vivette Girault. *Finite Element Methods for Navier-Stokes Equations*. Springer Series in Computational Mathematics. Springer Berlin, Heidelberg.

[5] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. Jupyter notebooks - a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Scmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, Netherlands, 2016. IOS Press.

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[7] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[8] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[9] Per-Olov Åsén. Stability of plane Couette flow and pipe Poiseuille flow. 2007.

[10] O.Schenk and K.Gartner. Solving unsymmetric sparse systems of linear equations with PARDISO. J. of Future Generation Computer Systems, 20(3):475-487, 2004.