



Politechnika Warszawska

Wydział Mechaniczny Energetyki i Lotnictwa

Implementacja metody velocity correction do rozwiązania zagadnienia Stokesa

Przedmiot: Obliczenia inżynierskie w chmurze

Kajusz Zieliński

Numer albumu 313739

Prowadzący przedmiot: dr inż. Mateusz Źbikowski

Data oddania: 16 stycznia 2026

Dokument przedstawia raport z projektu zaliczeniowego, w tym omówienie analizowanego zagadnienia, wybranej metody rozwiązania problemu, implementacji numerycznej i analizę uzyskanych wyników.

WARSZAWA 2026

Spis treści

1 Wstęp i sformułowanie problemu	1
1.1 Opis zagadnienia	1
2 Omówienie kodu	2
2.1 Budowa siatki elementów skończonych	2
2.2 Budowa macierzy MES	3
2.3 Rozwiążanie analityczne	3
2.4 Implementacja velocity correction	4
3 Wyniki	4
3.1 Manufactured solution	5
3.2 Wyniki dla $(q,r)=(2,1)$ - Matlab	7
3.3 Wyniki dla $(q,r)=(2,1)$ - Python	13
4 Podsumowanie i wnioski	20

1 Wstęp i sformułowanie problemu

W ramach projektu zostały wykonane symulacje z zastosowaniem metody velocity correction wykorzystując do tego kod w języku programowania Python z zastosowaniem obliczeń w maszynie wirtualnej. Wyniki porównano z wynikami otrzymanymi w ramach zmodyfikowanego projektu polegającego na opracowaniu implementacji metody *pressure correction* w oprogramowaniu Matlab. Celem projektu było zastosowanie możliwości obliczeń w chmurze, porównanie jakości wyników między dwoma językami programowania oraz potwierdzenie poprawności zastosowania metody *velocity correction*. Dodatkowo sprawdzono sposób wykonywania symulacji w celu przedstawienia korzyści wynikających z przeprowadzania obliczeń w maszynie wirtualnej. W ramach sprawozdania z projektu przedstawiono skrócony do minimum opis rozwiązywanego zagadnienia oraz omówienie kodu, a następnie zaprezentowane zostały wyniki porównujące dwa podejścia do implementacji kodu. Kod w języku Python został napisany z wykorzystaniem biblioteki NumPy co pozwoliło na proste przekonwertowanie kodu napisanego w programie Matlab.

1.1 Opis zagadnienia

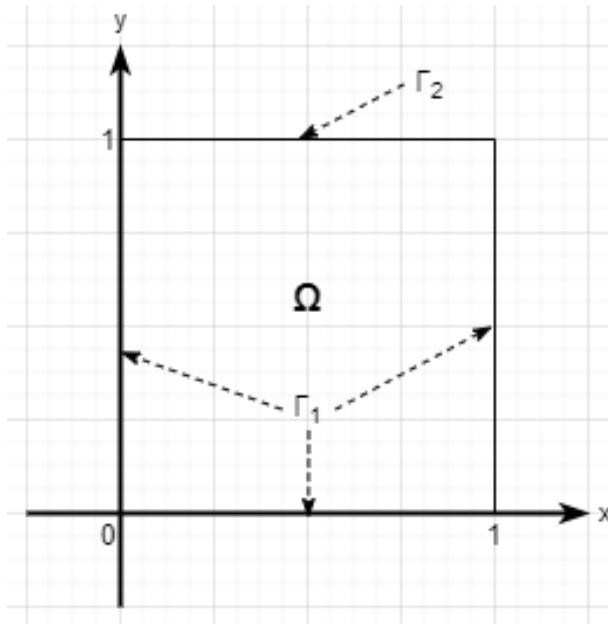
Zagadnienie dotyczy dwuwymiarowego niestacjonarnego ruchu płynu nieściśliwego w obszarze $\Omega = [0, 1] \times [0, 1]$, o brzegu Γ . Przyjmując, że prędkość płynu jest na tyle niska lub lepkość lepkość płynu jest na tyle duża [1], że człon lepki przeważa nad członem konwekcyjnym, formułuje się równanie Stokesa opisujące ruch cieczy:

$$\begin{cases} \partial_t \vec{u} = -\nabla p + \nu \Delta \vec{u} + \vec{f} & \text{w } \Omega \\ \nabla \cdot \vec{u} = 0 & \text{w } \Omega \end{cases} \quad (1)$$

Gdzie \vec{u} opisuje pole prędkości płynu, ν to lepkość kinematyczna płynu, p to pole ciśnienia dzielonego przez gęstość płynu, \vec{f} to pole sił masowych. Brzeg obszaru Ω można podzielić na dwie części, zgodnie z rysunkiem 1.1. W obszarze Ω zadane są następujące warunki brzegowe i początkowe:

$$\begin{cases} \vec{u}|_{\Gamma_1} = \vec{0} \\ \vec{u}|_{\Gamma_2} = U(x, t) \vec{e}_x \\ \vec{u}|_{t=0} = \vec{0} \end{cases} \quad (2)$$

Funkcja $U(x, t)$ jest funkcją opisującą ruch elastycznej pokrywy wywołującej ruch płynu, spełniającej następujące warunki: $U(x, 0) = 0$ i $U(0, t) = U(1, t) = 0$. Domenę obliczeniową przedstawiono na rysunku 1.1



Rysunek 1.1. Domena obliczeniowa z zaznaczonym podziałem brzegu

Problem testowy jest sformułowany w taki sposób, by istniało rozwiązanie analityczne, które posłuży do weryfikacji rozwiązania numerycznego i określenia rzędu metody numerycznej. W budowie rozwiązania zastosowano metodę *manufactured solution*.

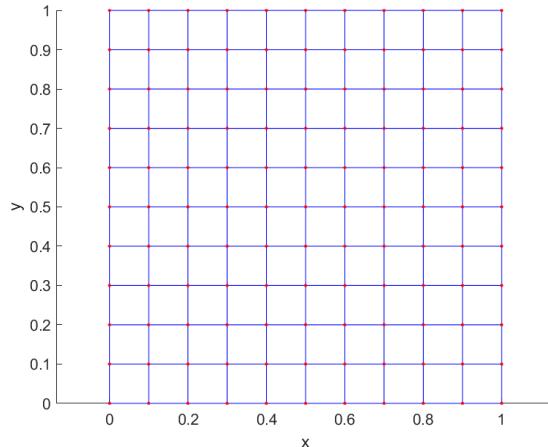
2 Omówienie kodu

W celu uruchomienia programu użytkownik wprowadza z klawiatury rozmiary domeny w postaci dwóch wektorów dwuelementowych, definiujących granice domeny wzdłuż osi x i y oraz liczbę elementów siatki wzdłuż jednego boku. Dla tych parametrów wywoływana jest funkcja budująca siatkę i macierze sztywności, masowe i gradientu wykorzystywane w obliczeniach. W dalszym kroku użytkownik podaje czas trwania symulacji, wielkość kroku czasowego i lepkość kinetyczną płynu. Korzystając z tych wartości i zbudowanych macierzy wykonywany jest algorytm metody *velocity correction* i obliczane są pola ciśnienia i prędkości na podstawie rozwiązania analitycznego.

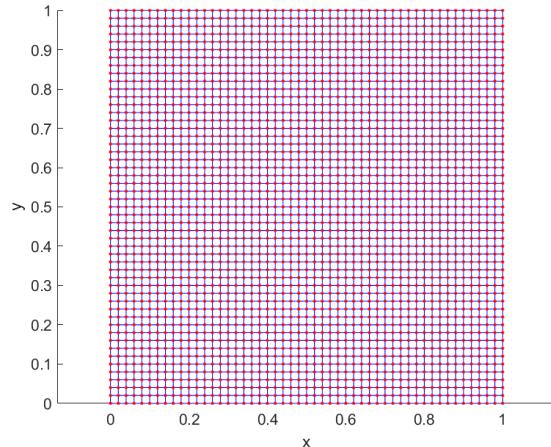
2.1 Budowa siatki elementów skończonych

Funkcja przyjmuje jako argumenty zadane z klawiatury wymiary domeny obliczeniowej i liczbę podziałów na krawędziach domeny. Zakłada się jednakową liczbę podziału wzdłuż każdego z boków kwadratu. Następnie generuje współrzędne węzłów siatki za pomocą gotowej funkcji *linspace()*, które przechowywane są w macierzy *coords*, rozmiarów liczba węzłów x 2. Oddziennie przechowywane są numery węzłów siatki. W pętli *for* numery węzłów wpisywane są we właściwe miejsca macierzy *mesh_matrix*. W macierzy *mesh_XY* zapisywane są współrzędne odpowiadające danemu węźlowi. Wewnątrz elementu węzły numerowane są przeciwnie do ruchu wskazówek zegara zaczynając od dolnego lewego węzła. Globalnie węzły

i elementy numerowane są poziomo od dolnego lewego węzła/elementu. Przykłady siatek wygenerowanych przy pomocy tej funkcji przedstawiono na 2.1. Na niebiesko zaznaczono krawędzie elementów, na czerwono węzły siatki.



(a) 10 podziałów wzdłuż krawędzi



(b) 50 podziałów wzdłuż krawędzi

Rysunek 2.1. Przykłady siatek elementów skończonych wygenerowanych funkcją *make_mesh*

2.2 Budowa macierzy MES

Macierze sztywności, masowa i gradientu budowane są w oddzielnych funkcjach, w ten sam sposób. Dla każdej z nich obliczono analitycznie wartości elementów macierzy dla pojedynczego elementu. Następnie wewnątrz pętli *for* dla każdego elementu siatki identyfikowany jest globalny numer węzła odpowiadający lokalnej numeracji i w to miejsce dodawany jest element lokalnej macierzy do globalnej macierzy rzadkiej, przy pomocy funkcji *sparse()*. Wartości elementów lokalnych macierzy przedstawiono w równaniach (3) i (4)

$$K = \frac{1}{6} \begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix} \quad M = \frac{h^2}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} \quad (3)$$

$$R_x = \frac{h}{12} \begin{bmatrix} -2 & 2 & 1 & -1 \\ 2 & -2 & -1 & 1 \\ 1 & -1 & -2 & 2 \\ -1 & 1 & 2 & -2 \end{bmatrix} \quad R_y = \frac{h}{12} \begin{bmatrix} -2 & -1 & 1 & 2 \\ -1 & -2 & 2 & 1 \\ 1 & 2 & -2 & -1 \\ 2 & 1 & -1 & -2 \end{bmatrix} \quad (4)$$

2.3 Rozwiązanie analityczne

Funkcja *exact_solution* wyznacza wartości wektora sił masowych \vec{f} w węzłach siatki dla zadanego *manufactured solution*. Jako argumenty wejściowe przyjmuje macierz współrzędnych węzłów, czas trwania symulacji, krok czasowy i lepkość kinetyczną płynu,

odczytane z klawiatury. Pochodne po czasie i przestrzeni zostały obliczone ręcznie i wprowadzone do kodu. W pętli *for* dla każdego kroku czasowego obliczane są wartości składowych prędkości, pole ciśnienia oraz składowe wektora sił masowych, które są zwracane przez program. Wyniki zostały przedstawione w następnym rozdziale.

2.4 Implementacja velocity correction

Cała metoda zaimplementowana jest w jednej funkcji. Jako dane wejściowe przyjmuje globalne macierze MES, macierz współrzędnych węzłów siatki, liczba węzłów siatki, wielkość kroku czasowego, czas trwania symulacji i lepkość kinematyczną. Jako pierwsza wywoływana jest funkcja *exact_solution* w celu obliczenia składowych wektora sił masowych oraz inicjalizowane są zmienne. Wartości ciśnienia i prędkości rozwiązanej numerycznie inicjalizowane są wartościami rozwiązania dokładnego w pierwszym kroku czasowym. Wewnątrz pętli *for* wykonywane są wszystkie kroki metody. Najpierw aktualizowane są wartości prędkości i pola sił masowych z poprzednich kroków czasowych i budowana jest prawa strona równania dla pomocniczego pola ϕ w pierwszym kroku wewnątrz funkcji *rhs_phi* w kroku $k+1$. Rozwiązanie równania pola ϕ wymaga nałożenia warunku brzegowego Neumanna. Zaimplementowany jest poprzez modyfikację macierzy K , zgodnie z równaniem (5) przedstawionym poniżej.

$$\begin{bmatrix} K & B \\ B^T & 0 \end{bmatrix} = \begin{bmatrix} \phi^{k+1} \\ x \end{bmatrix} = \begin{bmatrix} F^{k+1} \\ 0 \end{bmatrix} \quad (5)$$

Gdzie B to wektor jedynek, F^{k+1} to prawa strona równania obliczona dla kroku $k+1$. Równanie rozwiązywane jest za pomocą wbudowanej funkcji *mldivide()*. Następnie aktualizowane jest pole ciśnienia w kroku czasowym $k+1$ zgodnie z równaniem kroku drugiego. Ostatnim krokiem jest rozwiązanie równania dla pola \tilde{u}^{k+1} . W tym celu budowany jest wektor prawej strony równania w funkcji *rhs_u_wave*, nakładany jest warunek Dirichleta poprzez wpisanie wartości z wektora prędkości obliczonej analitycznie w odpowiednich wierszach w wektorze \tilde{u} i zastąpienie wyrazów w danym wierszu zerami i jedynką na diagonali. Układ równań rozwiązywany jest za pomocą funkcji *mldivide()*. Funkcja zwraca wartości pola ϕ , ciśnienia i prędkości \tilde{u} .

3 Wyniki

Wyniki zostały uzyskane za pomocą zaimplementowanego programu, rozwiązującego niestacjonarne zagadnienie Stokesa przy użyciu metody *velocity correction* dla wariantu $q = 2$, $r = 1$. Celem symulacji numerycznych była weryfikacja poprawności implementacji oraz porównanie sprawności zastosowanej metody względem rozwiązania analitycznego. Dla zdefiniowanego problemu testowego z rozwiązaniem analitycznym (manufactured solution) przeprowadzono serię obliczeń dla różnych parametrów dyskretyzacji przestrzennej oraz

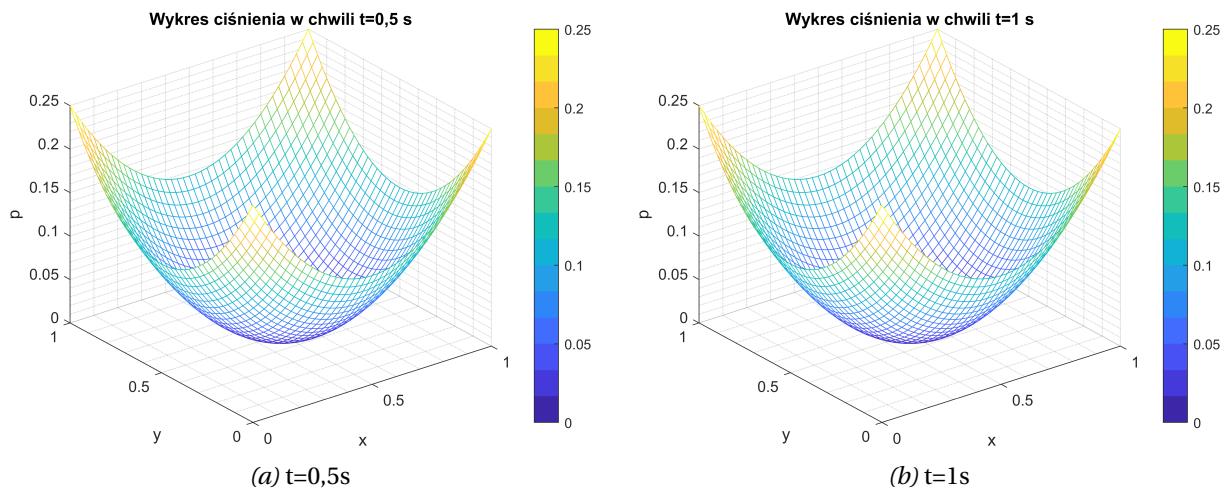
czasowej. Otrzymane wyniki numeryczne zostały porównane z odpowiadającymi im wartościami analitycznymi. Na tej podstawie obliczono błędy rozwiązania w funkcji czasu. Błąd rozwiązania numerycznego prezentowany w dalszej części tego rozdziału zdefiniowany jest jako maksimum wartości bezwzględnej błędu bezwzględnego dla pola ciśnienia i modułu prędkości w domenie obliczeniowej, matematyczne sformułowanie przedstawiono w równaniach (6) i (7).

$$\delta_p = \max (\| p_{num} - p_{man} \|)_{\Omega} \quad (6)$$

$$\delta_u = \max (\| \vec{u} - \vec{u}_{man} \|)_{\Omega} \quad (7)$$

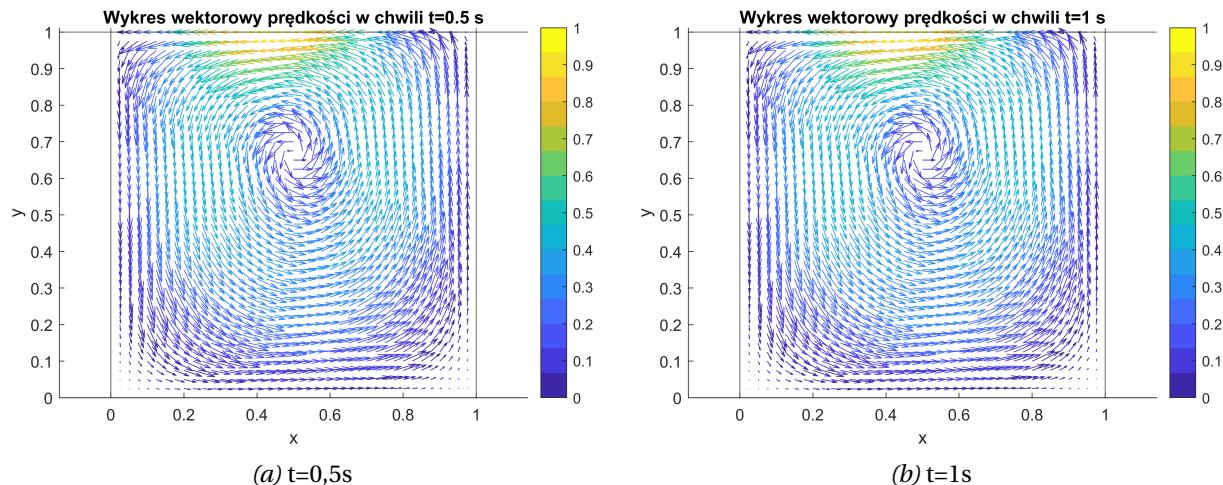
3.1 Manufactured solution

Na wykresach 3.1 i 3.2 przedstawiono pole ciśnienia i pole wektorowe prędkości dla dwóch chwil czasu równych 0,5 s oraz 1 s dla lepkości kinematycznej równej $0,5 \text{ m}^2 \text{ s}^{-1}$ i przyjętej funkcji $A(t) = \sin(t)$. Uzyskane pole ciśnienia zgodnie z założeniami jest stacjonarne i ma kształt paraboloidy, osiągając maximum równe $p_{max} = 0,5v$ w rogach obszaru. Pole ciśnienia w kodzie obliczane jest prawidłowo.

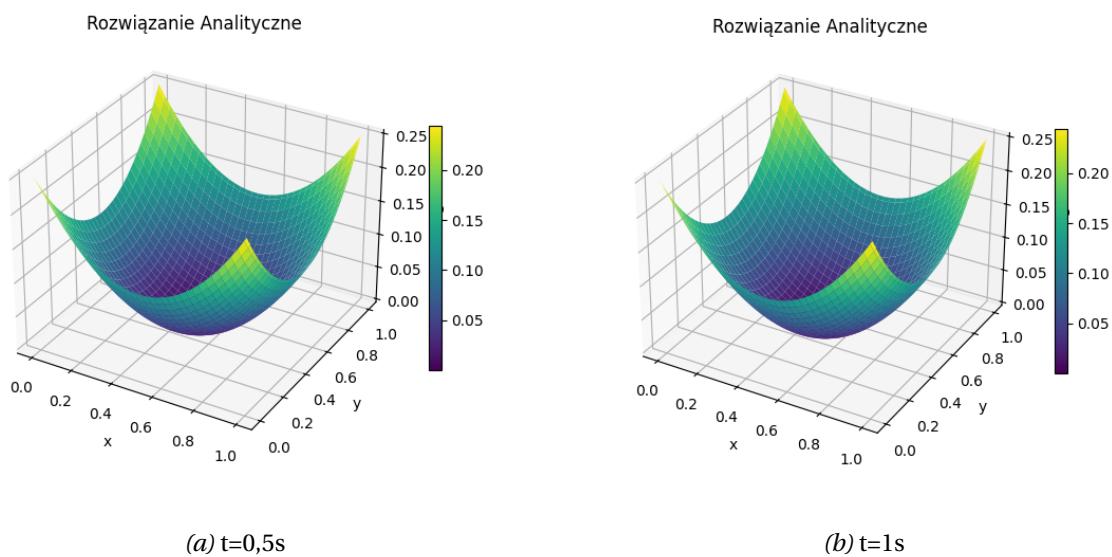


Rysunek 3.1. Wykres konturowy założonego pola ciśnienia

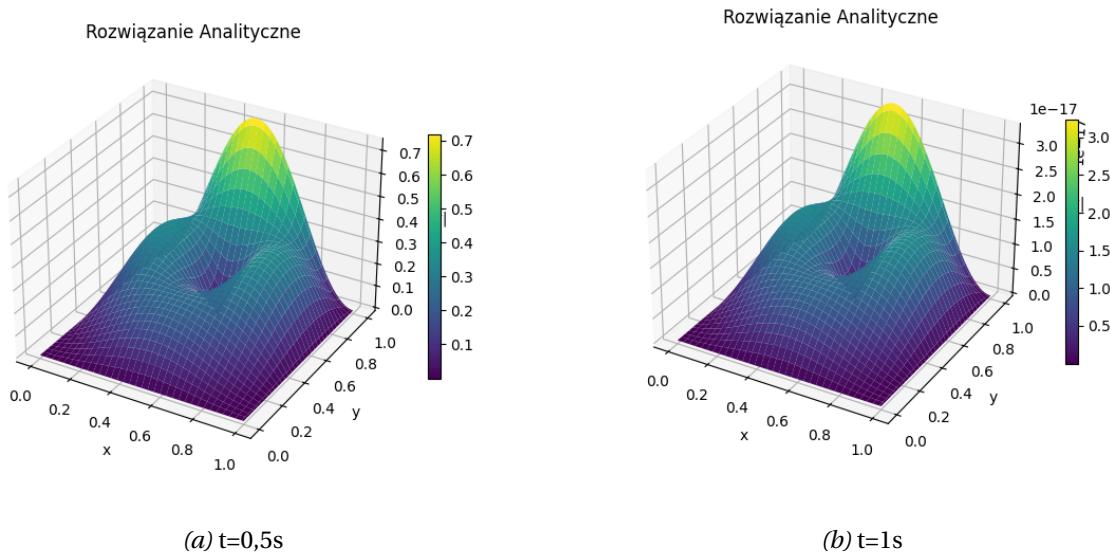
Wykres wektorowy pola prędkości również wskazuje, że pole prędkości obliczane jest przez program prawidłowo. Warunek Dirichleta jest spełniony, na górnej krawędzi prędkość ma wyłącznie składową x -ową, zgodną z zadanym polem, na pozostałych jest równa 0. Wewnątrz domeny powstaje wir generowany przez źródło prędkości na górnej krawędzi, którego centrum znajduje się mniej więcej w punkcie (0,65, 0,5).



Rysunek 3.2. Wykres wektorowy założonego pola prędkości



Rysunek 3.3. Wykres wektorowy założonego pola ciśnienia - python



Rysunek 3.4. Wykres wektorowy założonego pola prędkości - python

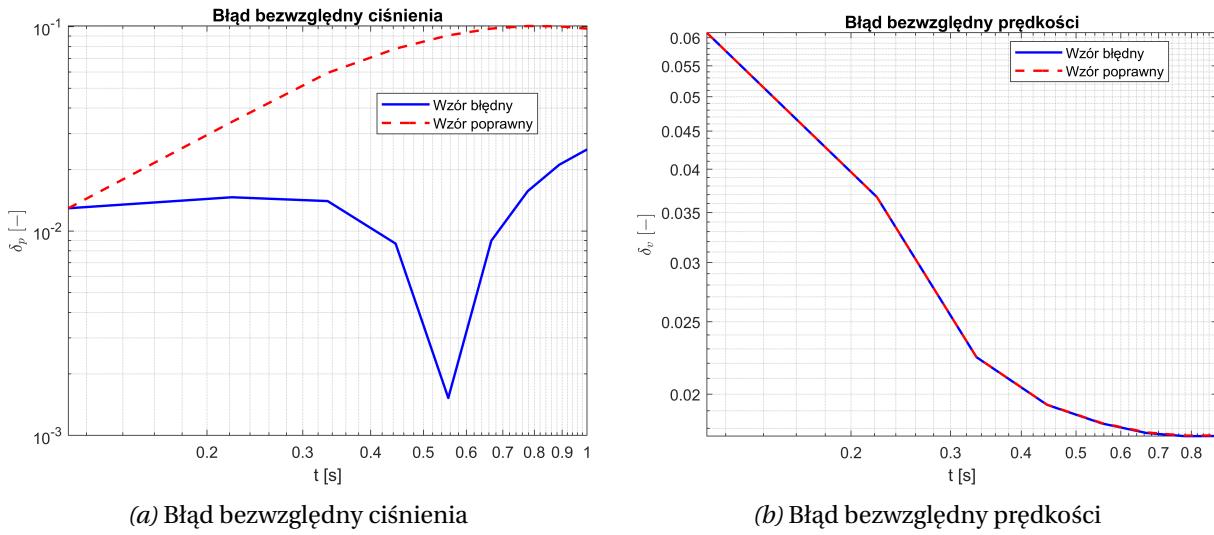
3.2 Wyniki dla $(q,r)=(2,1)$ - Matlab

Przeprowadzono serię symulacji dla takich samych warunków (liczba elementów wzdłuż krawędzi: 100, lepkość: 0,5, czas trwania 1), z jedynie zmiennym krokiem czasowym z zakresu [0,1, 0,05, 0,01, 0,005, 0,001, 0,0005, 0,0001], dalsze zmniejszenie kroku okazało się niemożliwe ze względu na ograniczenia komputera. Zauważono również, że schemat nie jest bezwarunkowo stabilny, przy małym kroku czasowym wymagana jest dostatecznie gęsta dyskretyzacja domeny obliczeniowej.

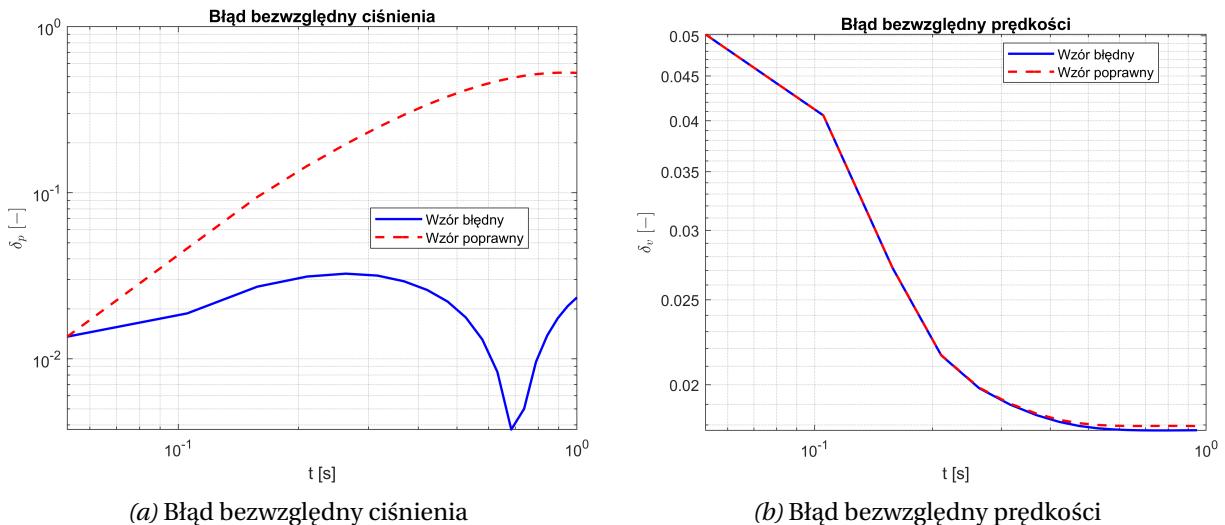
W trakcie symulacji w programie Matlab zaobserwowano, że rozwiązanie numeryczne uzyskane w programie jest bliższe rozwiązaniu analitycznemu, kiedy dokona się zmiany dyskretyzacji pochodnej po czasie w prawej stronie równania w kroku pierwszym, zastępując wyraz z równania (8) wyrazem (9), będącym sformułowaniem schematu wstecznego dla trzech kroków. Nie znaleziono żadnego wyjaśnienia, dlaczego zmiana wzoru na niezgodny z wyprowadzeniem analitycznym korzystnie wpływa na dokładność rozwiązania. Na wykresach od 3.5 do 3.11 przedstawiono porównanie błędu rozwiązania pola prędkości i pola ciśnienia dla obu dyskretyzacji w funkcji czasu symulacji.

$$\frac{1}{2\Delta t} \left(7\tilde{u}^k - 5\tilde{u}^{k-1} + \tilde{u}^{k-2} \right) \quad (8)$$

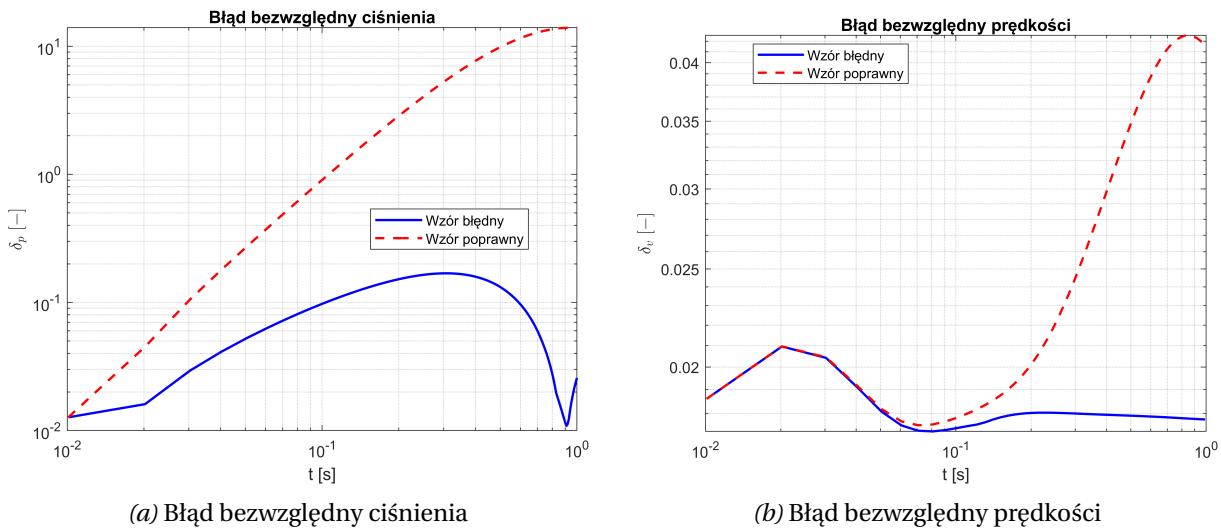
$$\frac{1}{2\Delta t} \left(3\tilde{u}^k - 4\tilde{u}^{k-1} + \tilde{u}^{k-2} \right) \quad (9)$$



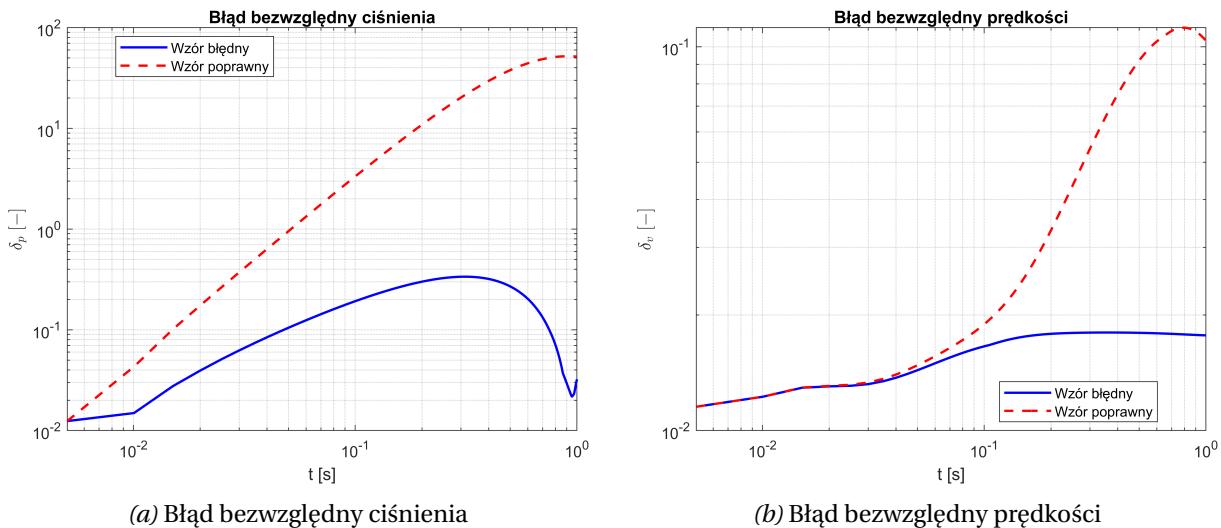
Rysunek 3.5. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,1 s



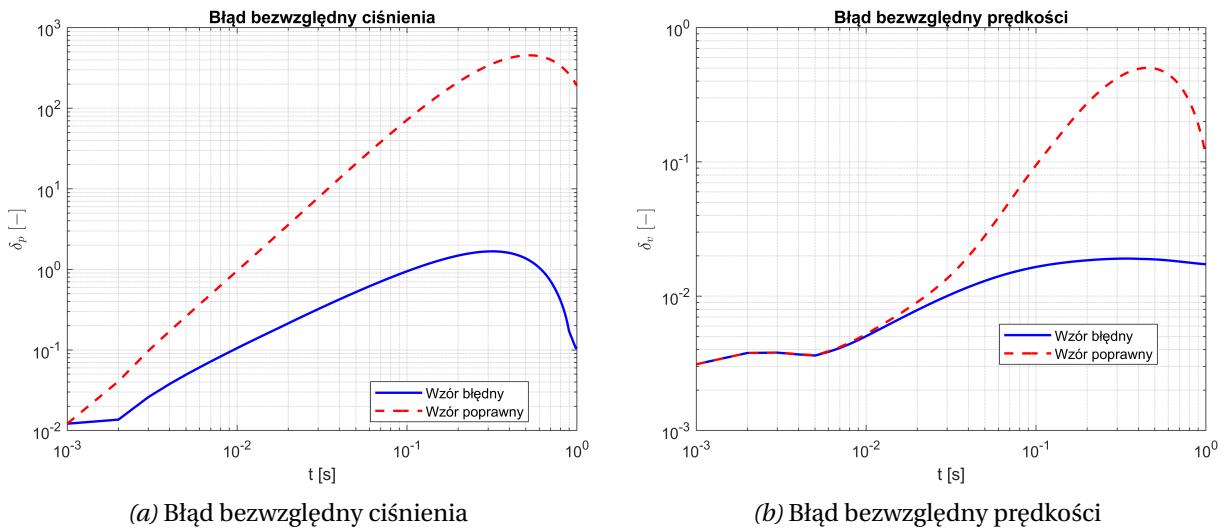
Rysunek 3.6. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,05 s



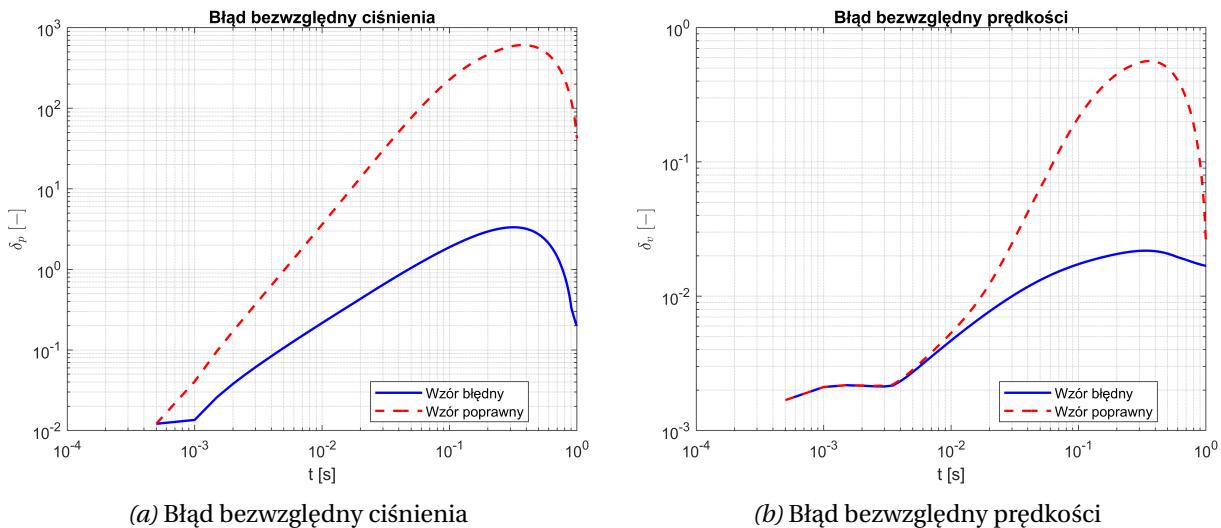
Rysunek 3.7. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,01 s



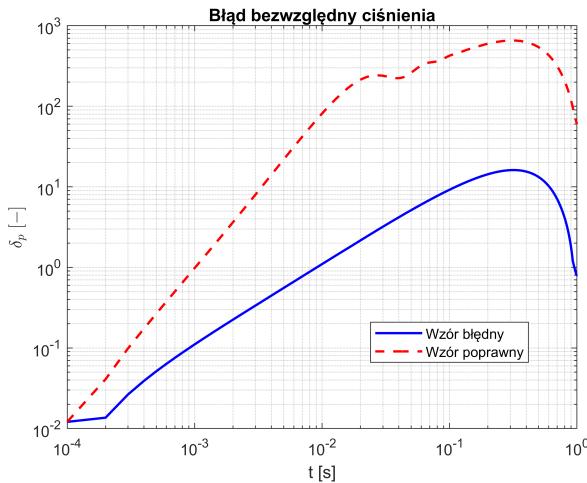
Rysunek 3.8. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,005 s



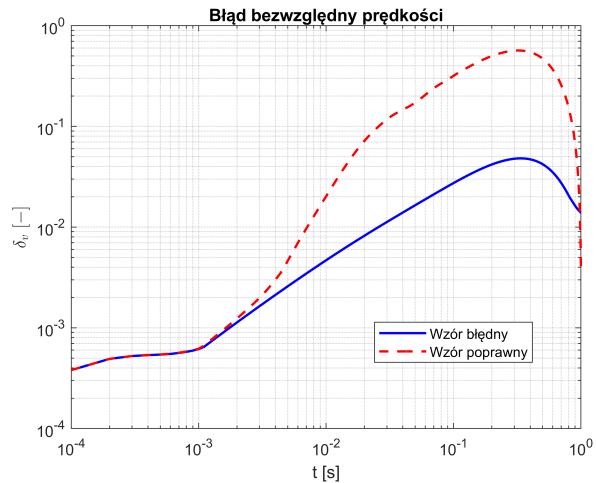
Rysunek 3.9. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,001 s



Rysunek 3.10. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,0005 s



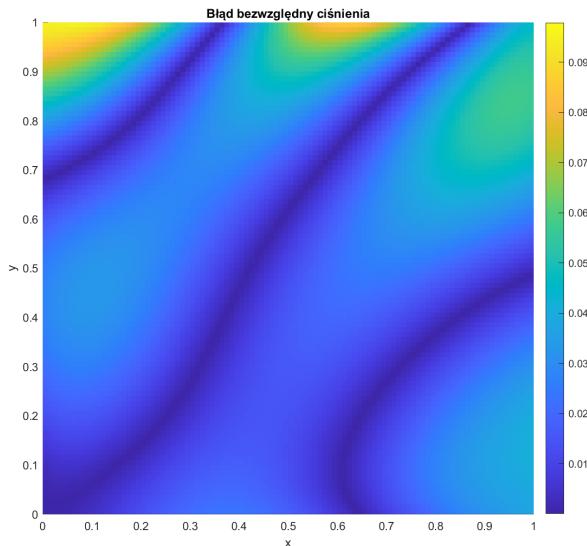
(a) Błąd bezwzględny ciśnienia



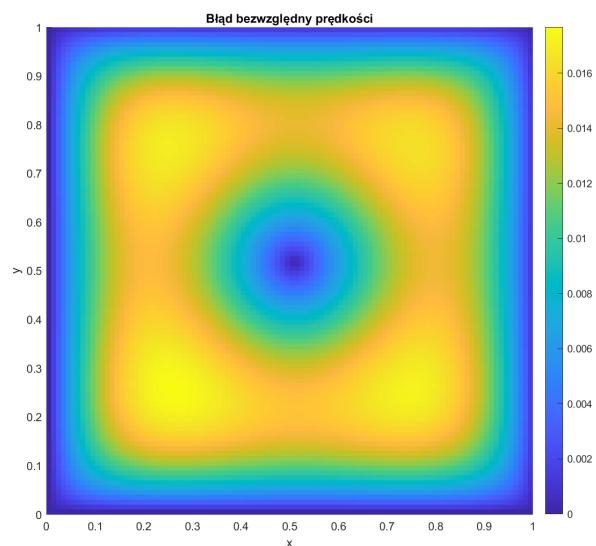
(b) Błąd bezwzględny prędkości

Rysunek 3.11. Wykres błędu rozwiązań ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,0001 s

Da dwóch pierwszych kroków czasowych rozwiązywanie pola prędkości dla obu sformułowań jest bardzo zbliżone do siebie, natomiast wraz z krótszym krokiem czasowym błąd rozwiązań z poprawnym wzorem zaczyna rosnąć, osiągając wartości o blisko rząd wielkości większe. Natomiast rozwiązywanie pola ciśnienia dla każdego z rozważonych kroków czasowych jest obarczone większym błędem dla sformułowania poprawnego. Błąd początkowo rośnie przez cały czas trwania symulacji, jednak z coraz mniejszym krokiem czasowym zaczyna gwałtownie maleć przy zbliżaniu się do końca trwania symulacji. W sformułowaniu błędym obserwuje się podobne zachowanie błędu dla pola ciśnienia i prędkości.

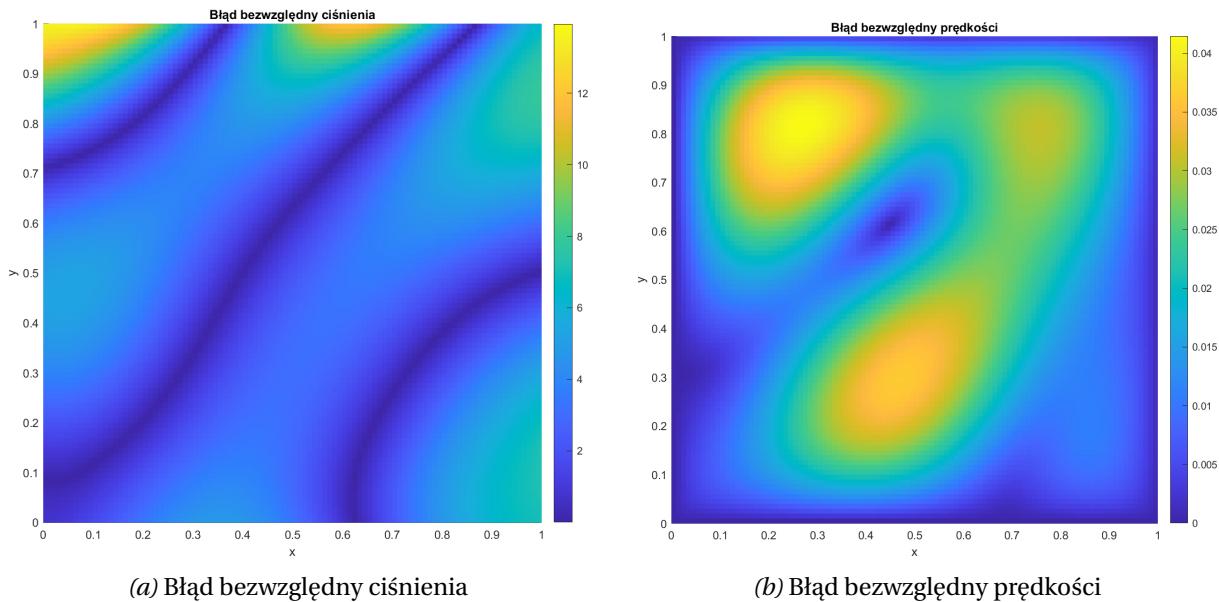


(a) Błąd bezwzględny ciśnienia

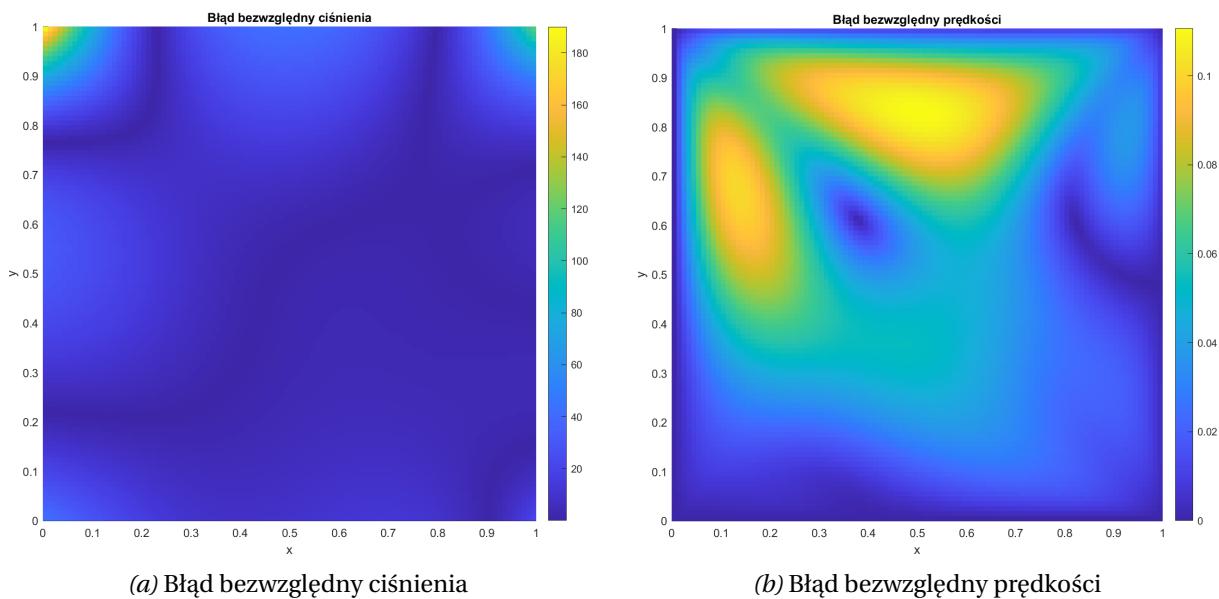


(b) Błąd bezwzględny prędkości

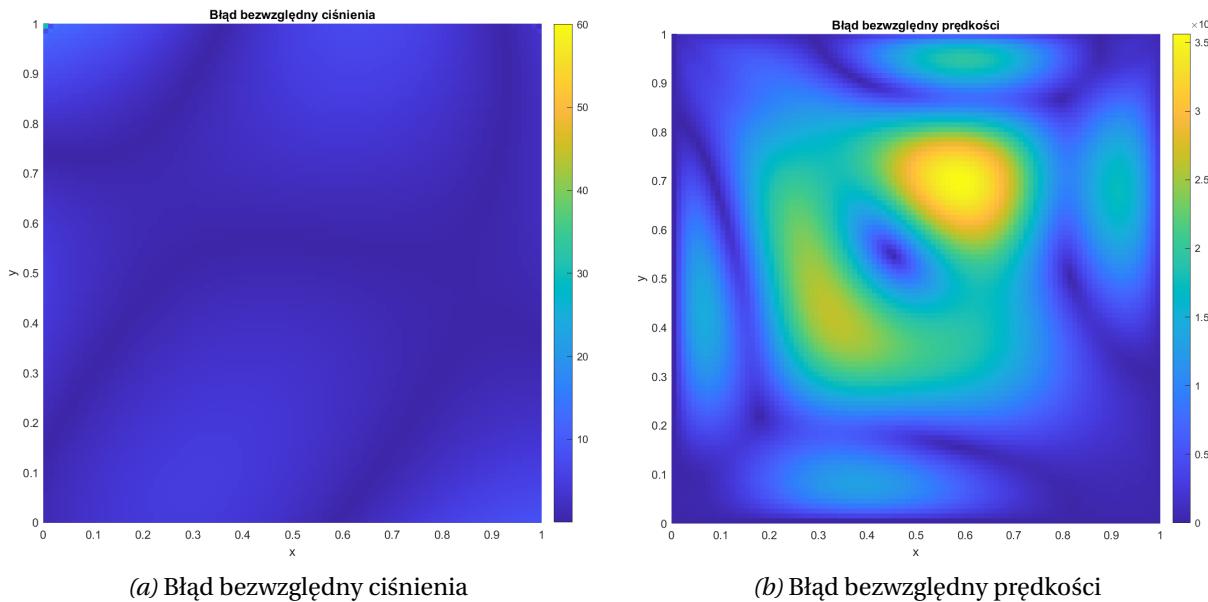
Rysunek 3.12. Wykres błędu rozwiązań ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,1 s dla sformułowania poprawnego



Rysunek 3.13. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,01 s dla sformułowania poprawnego



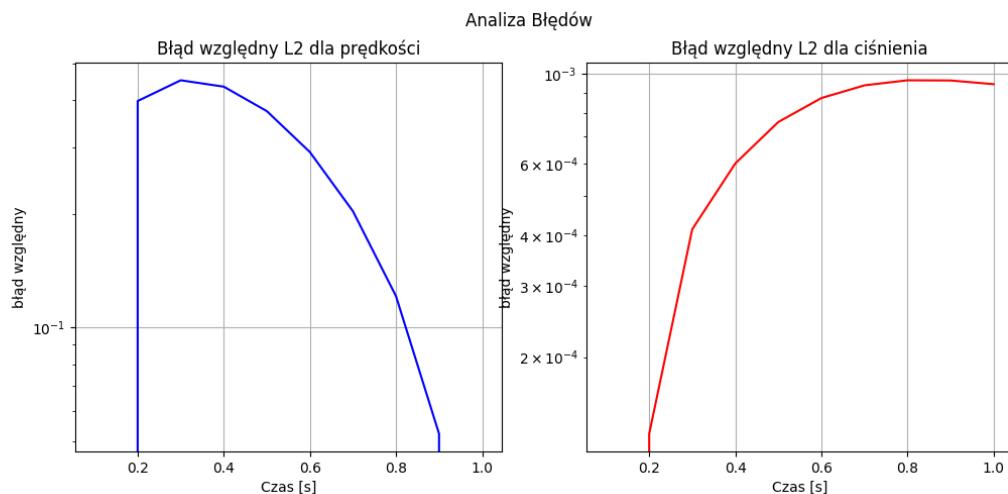
Rysunek 3.14. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,001 s dla sformułowania poprawnego



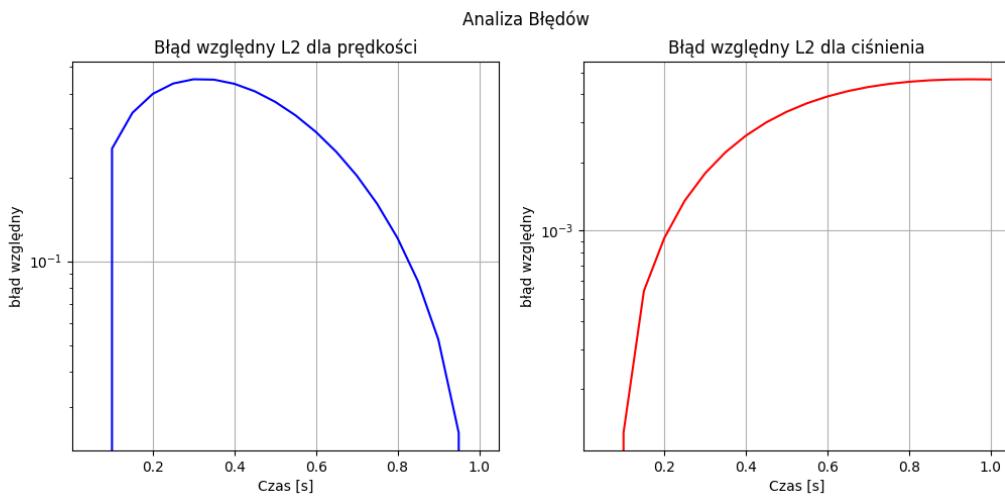
Rysunek 3.15. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,0001 s dla sformułowania poprawnego

3.3 Wyniki dla $(q,r)=(2,1)$ - Python

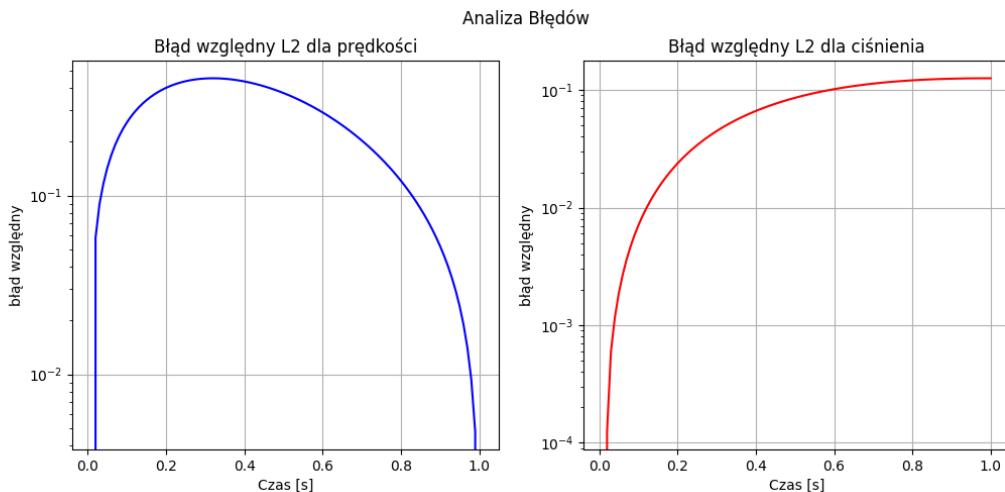
Przedstawione wyniki nie uwzględniają sformułowania błędnego ze względu na brak jasnego uzasadnienia tego podejścia i znaczące wydłużenie procesu otrzymywania pełnych wyników.



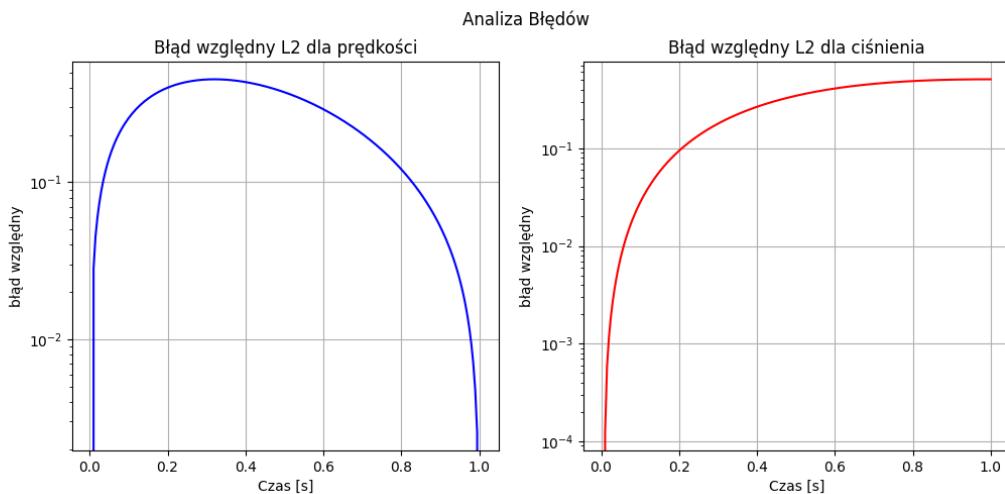
Rysunek 3.16. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,1 s



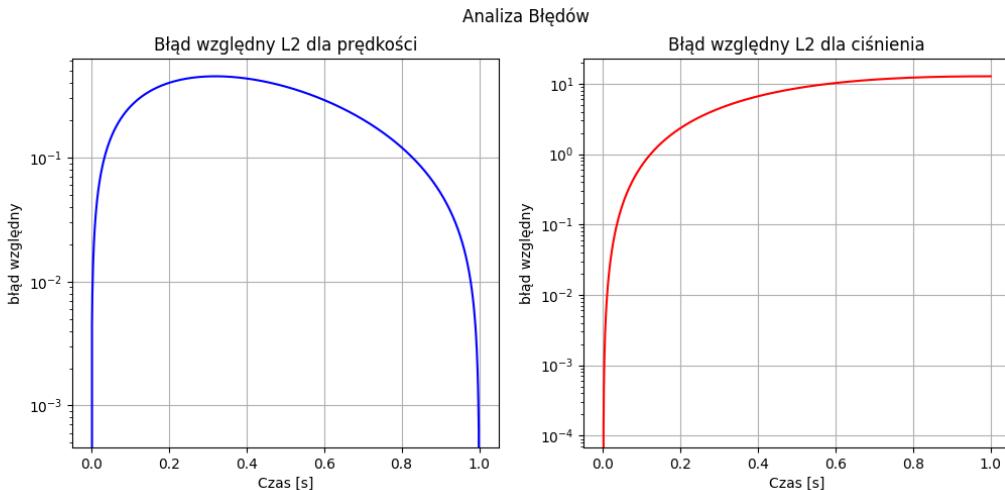
Rysunek 3.17. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,05 s



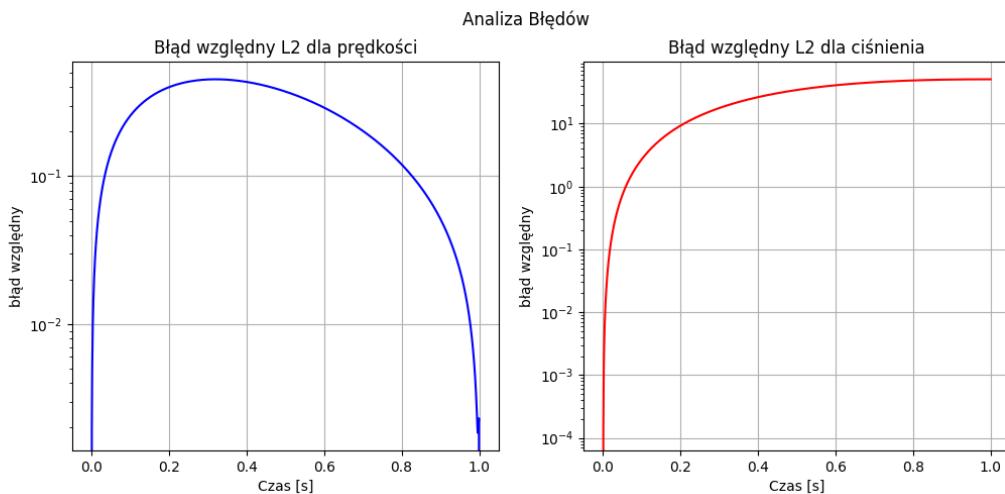
Rysunek 3.18. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,01 s



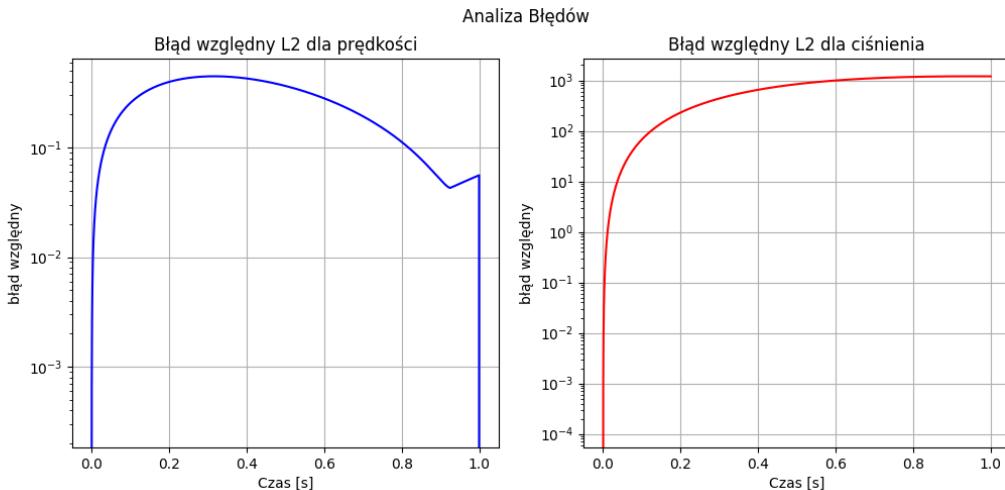
Rysunek 3.19. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,005 s



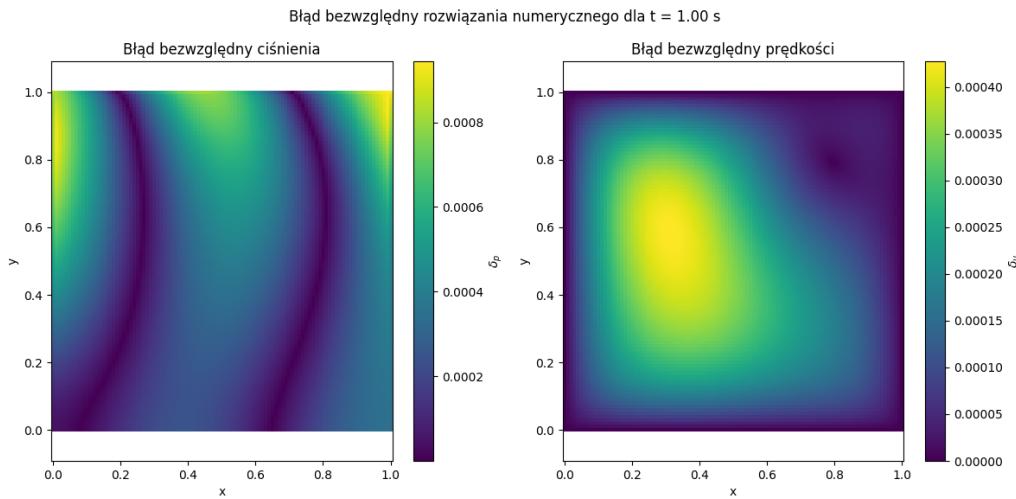
Rysunek 3.20. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,001 s



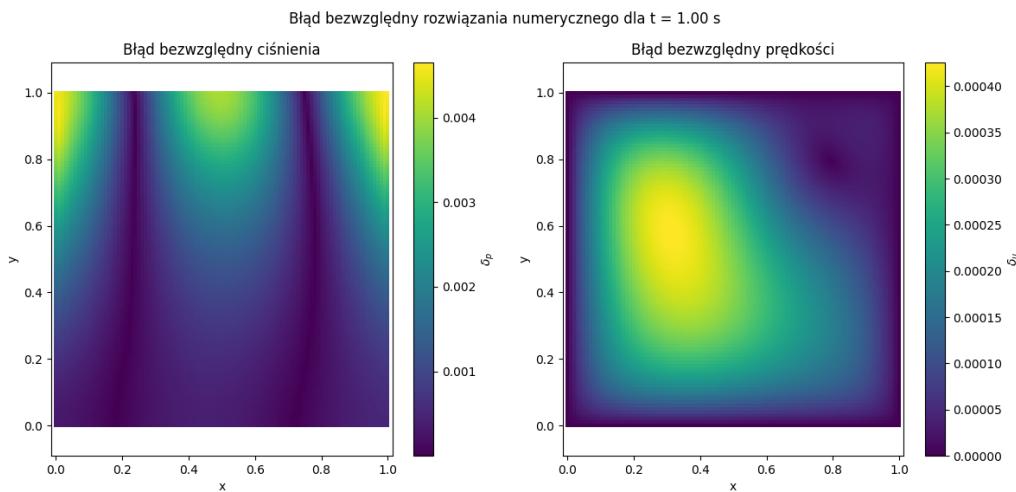
Rysunek 3.21. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,0005 s



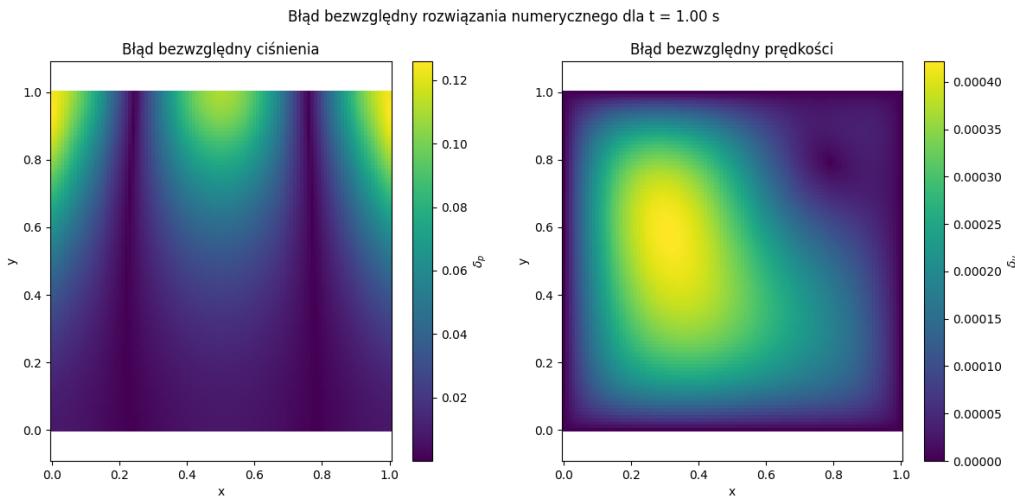
Rysunek 3.22. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,0001 s



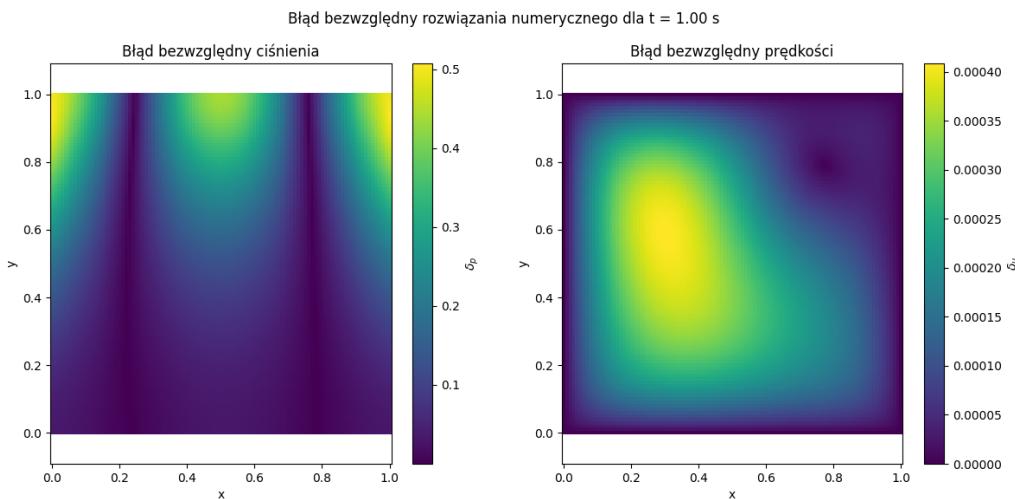
Rysunek 3.23. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,1 s



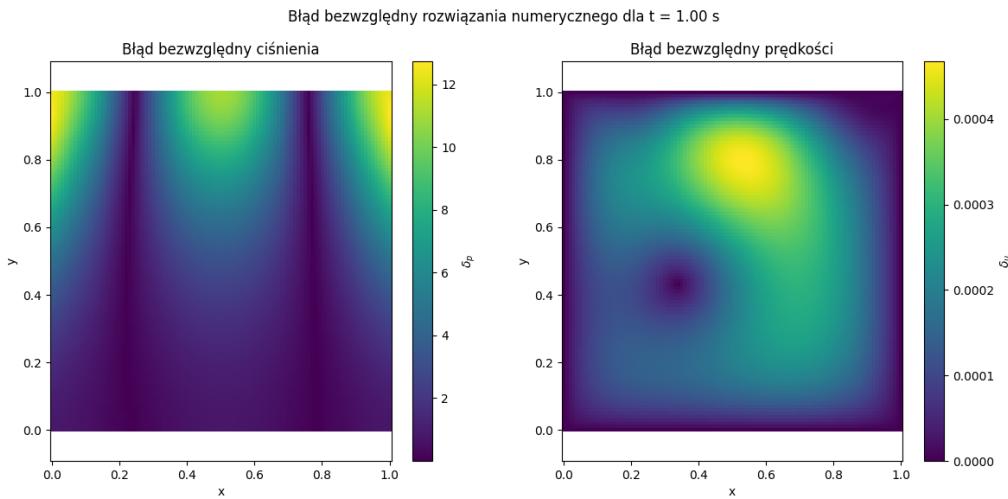
Rysunek 3.24. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,05 s



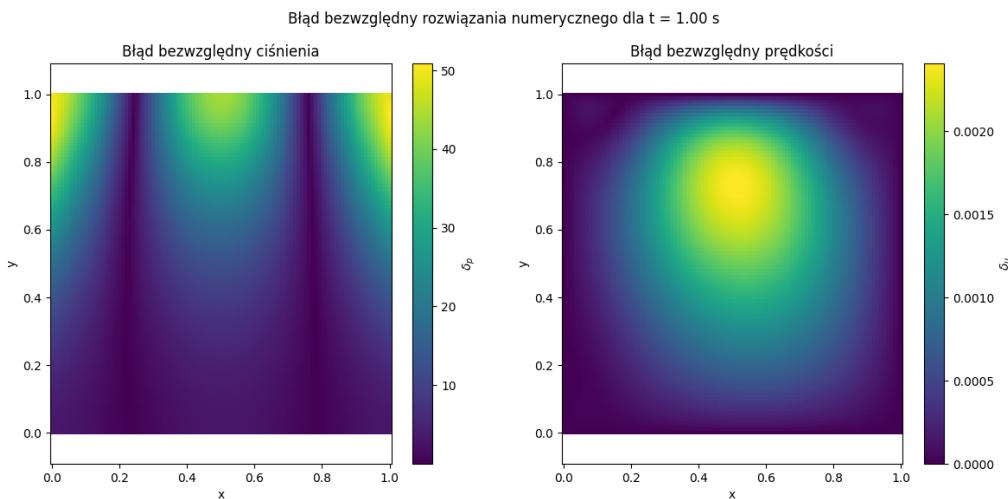
Rysunek 3.25. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,01 s



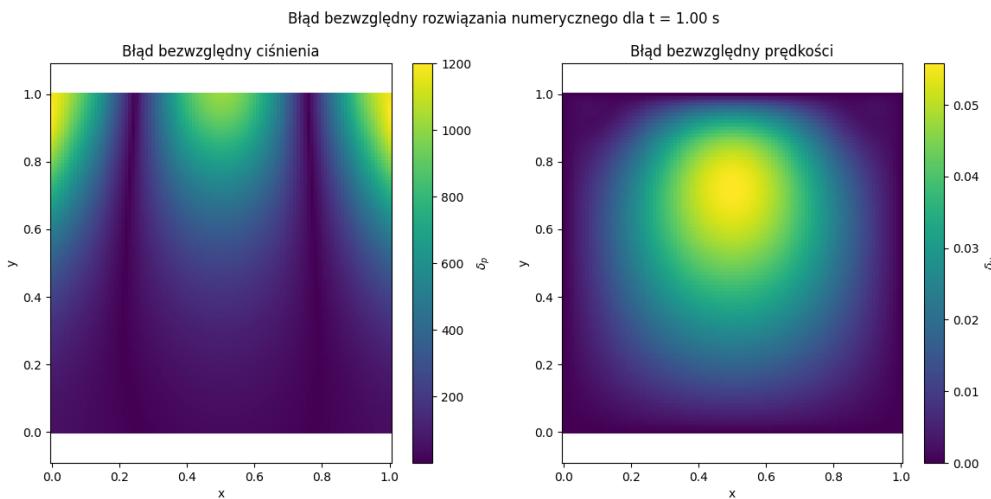
Rysunek 3.26. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,005 s



Rysunek 3.27. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,001 s



Rysunek 3.28. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,0005 s



Rysunek 3.29. Wykres błędu rozwiązania ciśnienia i prędkości w funkcji czasu trwania symulacji dla kroku czasowego 0,0001 s

4 Podsumowanie i wnioski

W ramach projektu stworzono program, który rozwiązuje numerycznie zagadnienie Stokesa dla niestacjonarnego przepływu płynu nieściśliwego w dwuwymiarowej domenie obliczeniowej. Do rozwiązania zastosowano algorytm *velocity correction* z dyskretyzacją domeny obliczeniowej elementami biliniowymi pierwszego rzędu. W ramach analizy wyników porównano wpływ rzędu metody, wielkości kroku czasowego i liczby elementów skończonych oraz zastosowanej funkcji czasu. Z wyników wysnuto następujące wnioski:

- Algorytmu nie udało się zaimplementować z pełnym sukcesem niezależnie od środowiska, uzyskiwane rozwiązanie numeryczne jest niedokładne, a schemat nie jest bezwarunkowo stabilny, wbrew [2]. Przyczyny błędu rozwiązania należy szukać w pierwszych dwóch krokach algorytmu, ponieważ \tilde{u} jest rozwiązywane z dobrą dokładnością, podstawiając analitycznie obliczone ciśnienie zamiast numerycznego, natomiast przy odwrotnym podstawieniu ciśnienie obliczane jest z dużym błędem.
- Czas trwania symulacji w przypadku wykonywania obliczeń w chmurze nie został zredukowany znacząco w porównaniu z lokalnie uruchomionym programem ze względu na niską wydajność parametrów wybranej maszyny (Standard B2ls v2). W celu poprawy należałoby wykorzystać maszynę wirtualną o większych możliwościach, szczególnie w zakresie pamięci RAM. Ze względu na naturę obliczeń w chmurze alternatywnym podejściem byłoby uruchomienie jednocześnie wielu symulacji na różnych maszynach wirtualnych.
- Wydajność obu środowisk (Matlab i Python) okazała się bardzo podobna z minimalną przewagą dla języka Python. W przypadku gdyby nie zastosowano biblioteki NumPy czas trwania symulacji byłby wyraźnie wolniejszy w porównaniu z Matlabem.

- Rozwiążanie pola ciśnienia jest wyznaczane z mniejszą dokładnością niż pola prędkości.
- Wraz ze zmniejszeniem kroku czasowego błąd bezwzględny ciśnienia rośnie, natomiast błąd bezwzględny prędkości początkowo maleje, dla kroku mniejszego od 0,005 s zaczyna rosnąć. Może być to wynik dominacji macierzy masowej nad macierzą sztywności w równaniu kroku trzeciego lub zaburzenie rozwiązania prędkości przez znaczący błąd rozwiązania ciśnienia.
- Dla siatek o większej liczbie elementów zaobserwowano opóźnienie propagacji błędów z obszarów brzegowych do wnętrza domeny obliczeniowej, co sugeruje, że część obserwowanej poprawy dokładności wynika z efektów numerycznych, a nie tylko z lepszej aproksymacji rozwiązania.

Bibliografia

- [1] C. Conca, F. Murat i O. Pironneau, „The Stokes and Navier-Stokes equations with boundary conditions involving the pressure”, *Japanese journal of mathematics. New series*, t. 20, nr. 2, s. 279–318, 1994. DOI: 10.4099/math1924.20.279.
- [2] S. Dong i J. Shen, „An unconditionally stable rotational velocity-correction scheme for incompressible flows”, *Journal of Computational Physics*, t. 229, nr. 19, s. 7013–7029, 2010, ISSN: 0021-9991. DOI: <https://doi.org/10.1016/j.jcp.2010.05.037>. adr.: <https://www.sciencedirect.com/science/article/pii/S0021999110002986>.