ZESZYTY ENERGETYCZNE
TOM V. *Nowoczesne metody pomiarowe i modelowanie numeryczne w energetyce cieplnej*
2018, s. 25–40

Realizacja metody cząstek wirowych w środowisku wieloprocesorowym z użyciem schematów różnicowych wysokiego rzędu

Dominik Błoński

Politechnika Wrocławska, Wydział Mechaniczno-Energetyczny
Katedra Technologii Energetycznych, Turbin i Modelowania Procesów Cieplno-Przepływowych
E-mail: dominik.blonski@pwr.edu.pl

REKOMENDACJA: *Henryk Kudela*

STRESZCZENIE

W pracy przedstawiono algorytm rozwiązywania równań ruchu płynu z wykorzystaniem kompaktowej, czwartorzędowej metody „Vortex in Cell". W pierwszej częsci zawarto sposób budowania układów równań liniowych z wykorzystaniem wysokowydajnej biblioteki *hypre*. Przedstawiono kolejne kroki algorytmu wraz z badaniami dokładności i wydajności obliczeniowej poszczególnych schematów różnicowych. Ostatecznie przedstawiono badanie dokładności działania metody na przykładzie zagadnienia Taylora–Greena ze znanym rozwązaniem dokładnym i porównano otrzymane wyniki z wartościami dokładnymi.

SŁOWA KLUCZOWE: *metoda VIC, metoda różnic skończonych,schematy kompaktowe, obliczenia równoległe.*

1. WPROWADZENIE

Możliwość badania zmian zachowania się płynu poprzez śledzenie ewolucji pola wirowego jest bardzo atrakcyjną i pożądaną metodą badawczą. W związku z powyższym faktem w metodach cząstek wirowych równania ruchu płynu wyrażane są w formie transportu wirowości Helmholtza

$$\frac{\partial \omega}{\partial t} + (\nabla \omega) \cdot \mathbf{u} = \nu \Delta \omega, \tag{1}$$

$$\Delta \psi = -\omega, \qquad \mathbf{u} = (u, v) = (\partial_y \psi, -\partial_x \psi). \tag{2}$$

Powyższe równania można z powodzeniem rozwiązać przy użyciu metody VIC, która łączy w sobie podejścia eulerowskie i lagrangowskie i jest zaliczana do metod wirowych. Do obliczeń wprowadza się lagrangowskie cząstki wirowe unoszone przez przepływ. Pole prędkości $\mathbf{u}$ służące po rozwiązania Eulera wyznacza się z rozkładu wirowości.

W algorytmie metody VIC pierwszym etapem jest wyznaczenie wartości potencjału wektorowego z rozkładu wirowości poprzez rozwiązanie równania Poissona. W przypadku dwuwymiarowym równanie Poissona sprowadza się do wyznaczenia funkcji prądu ψ. Gdy rozkład funkcji prądu jest już znany, może zostać określona prędkość w poszczególnych węzłach. Następnie posługując się metodą dekompozycji lepkościowej równanie Helmholtza rozdzielane jest na nielepki krok konwekcyjny i uwzględniający lepkość płynu krok dyfuzyjny. W pierwszym kroku równanie konwekcji zostaje zastąpione równaniem ruchu cząstek transportujących wirowość

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = 0 \longrightarrow \frac{dx}{dt} = \mathbf{u}. \tag{3}$$

Zgodnie z trzecim twierdzeniem Helmholtza linie wirowe przemieszczają się zgodnie z ruchem płynu [1],[2]. Dlatego też cząstki wirowe mogą być traktowane są jako cząstki materialne unoszone w polu prędkości. W każdym kroku czasowym cząstki transportują pewną wartość wirowości, która jest brana z węzła siatki obliczeniowej. Po przemieszczeniu cząstki wartość wirowości musi zostać z powrotem zwrócona na siatkę. Procedura ta zwana jest redystrybucją i wykorzystuje jądra interpolacyjne wysokiego rzędu w celu równomierniej i dokładnej redystrybucji wirowości na sąsiadujące węzły. W kolejnym kroku uwzględniana jest lepkość płynu poprzez rozwiązanie równania dyfuzji

$$\frac{d\omega}{dt} = \nu \Delta \omega. \tag{4}$$

Za wyborem przedstawionej powyżej metody przemawia również szybkość wykonywanych obliczeń. Wykorzystanie metody VIC połączonej z równoległymi obliczeniami na wielu procesorach może z powodzeniem być stosowana do niestacjonarnych obliczeń numerycznych związanych z burzliwymi przepływami płynów i jest w stanie wielokrotnie skrócić czas oczekiwania na wyniki przy jednoczesnym zachowaniu wysokiej dokładności otrzymanych wyników. W niniejszym artykule zostaną przedstawione efekty prac nad wysokowydajnym solverem bazującym na metodzie VIC. Wykorzystuje się kompaktowe schematay czwartego rzędu po czasie i przestrzeni. Zdyskretyzowane równania rozwiązano przy użyciu biblioteki *hypre*, której sposób działania jest przedstawiony w następnym rozdziale.

2. Działanie biblioteki hypre

Do rozwiązywania algebraicznych układów równań z wykorzystaniem wielu procesorów skorzystano z biblioteki *hypre*, która wykorzystuje protokół MPI do wymiany informacji pomiędzy poszczególnymi procesami. Do rozwiązywania układów równań w bibliotece wykorzystuje się typowe metody iteracyjne z użyciem prekondycjonowania. Obejmuje to metody prekondycjonowania dla systemów niesymetrycznych, takich jak GMRES i metod dla macierzy symetrycznych, takich jak metoda gradientu sprzężonego. Możliwe jest również rozwiązywanie układów równań liniowych z wykorzystaniem metod wielosiatkowych.

Wszystkie obliczenia prowadzone są z wykorzystaniem interfejsu dla kartezjańskiej siatki strukturalnej, czyli takiej w której odstępy pomiędzy poszczególnymi węzłami siatki obliczeniowej są jednakowe.

Istnieje jednak możliwość wykonywania obliczeń da siatek pół strukturalnych, oraz niestrukturalnych. Przez siatkę pół strukturalną rozumie się taką, w której pewne obszary siatki obliczeniowej, nie są w pełni strukturalne. Charakteryzują się np. układem blokowym lub siatkami sklejanymi z różnych fragmentów siatek strukturalnych. Taki interfejs może skutecznie zostać połączony z adaptacyjnym zagęszczaniem obszaru obliczeniowego (AMR – Adaptive Mesh Refinement).

Biblioteka *hypre* została wybrana do rozwiązywania algebraicznych układów równań liniowych głównie ze względu na swoją intuicyjność, obszerną dokumentację oraz ciągłe aktualizacje i wprowadzanie nowych funkcji przez autorów[5]. Warto również wspomnieć, że wykazana w licznych publikacjach [3],[4] skalowalność i prędkość obliczeń przemawia na jej korzyść. Przez skalowalność rozumie się zdolność do rozbudowy obszaru obliczeniowego poprzez rozdrobnienie siatki dodając kolejne wątki przy braku bądź akceptowalnym spadku wydajności. Innymi słowy wraz ze wzrostem liczby wątków o takim samym lokalnym problemie (np. 1 proces z siatką 8×8 i 4 procesy z siatką 8×8) do rozwiązania nie wzrośnie czas obliczeń. W następnym akapicie zostanie w syntetyczny sposób przedstawiona metoda przygotowania oraz prowadzenia obliczeń z wykorzystaniem wyżej omawianej biblioteki.

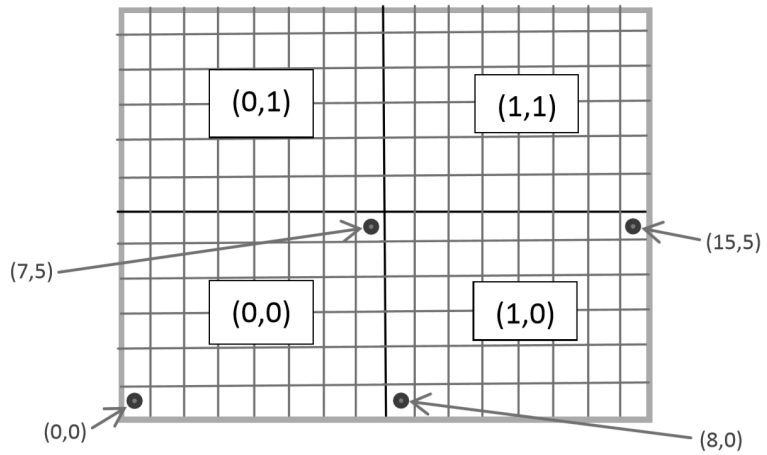
W celu rozwiązania układu równań algebraicznych w postaci $Ax = b$ musi zostać stworzony obszar obliczeniowy, zdefiniowany szablon schematu różnicowego, utworzenia macierzy A oraz jej modyfkacja dla zadanych warunków brzegowych, utworzenie wektora rozwiązania oraz prawej strony równania (wektor *b*). Po tych czynnościach wymagane jest wybranie algorytmu rozwiązującego układ równań (solvera) i określenie jego parametrów, a następnie uruchomienie go w celu otrzymania rozwiązania.

Pierwszym krokiem jaki należy wykonać to definicja obszaru obliczeniowego oraz określenie ilości wymiarów w których będą prowadzone obliczenia. Biblioteka numeruje odpowiednio każdy węzeł i procesor wg. układu kartezjańskiego (rys. 1). Kolorem czarnym zostały przedstawione numery procesorów, natomiast granatowym numery poszczególnych węzłów komórek. Są one w komórce dyskretyzowane schematami centralnymi. Wartości współrzędnych węzłów są wyliczane na podstawie rzędnej i odciętej danego procesora. Definiowanie obszaru obliczeniowego jak i wszelkie operacje na węzłach wykonuje się poprzez wskazanie współrzędnych dwóch skrajnych węzłów nazywanych odpowiednio ilower oraz iupper. Operacje na skrajnych, przeciwległych punktach dotyczą również modyfikacji poszczególnych rzędów, bądź kolumn siatki obliczeniowej (dla przypadku dwuwymiarowego). Wtedy tylko jedna ze współrzędnych nie ulega zmianie.

Przykład definicji obszaru obliczeniowego zgodnego z rys.1 jest następujący:

```
ilower[0] = pi*x;
ilower[1] = pj*y;
iupper[0] = ilower[0] + x-1;
iupper[1] = ilower[1] + y-1;
HYPRE_StructGridCreate(MPI_COMM_WORLD, ndim, &grid);
HYPRE_StructGridSetExtents(grid, ilower, iupper);
HYPRE_StructGridAssemble(grid);}
```

gdzie: x – ilość węzłów dla jednego procesora w kierunku osi x (np. tak jak na rys. 1, $x = 8$), y – ilość węzłów dla jednego procesora w kierunku osi y ($y = 6$), *pi, pj* – współrzędne procesora *ndim* – liczba wymiarów.



Rys. 1: Podział węzłów i procesorów

Po definicji obszaru obliczeniowego należy przystąpić do utworzenia macierzy A oraz wektorów x, b. Macierz A w metodzie ViC ulega zmianie w zależności od zagadnienia, które jest rozwiązywane (równanie Poissona, równanie dyfuzji).

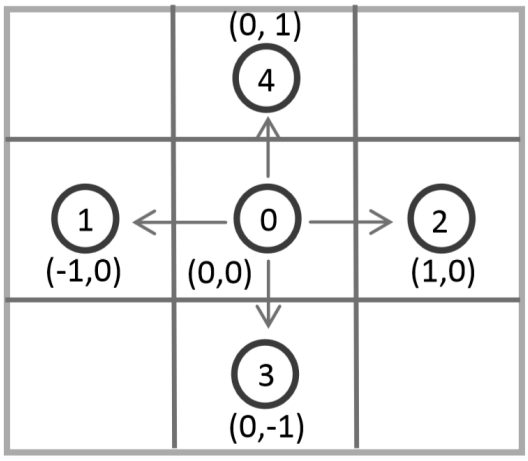
Każde rozwiązane w algorymie VIC równanie wymaga zastosowania innego schematu różnicowego. Współczynniki macierzy A dla danego zagadnienia nie ulegającą zmianie w kolejnych krokach czasowych. Współczynniki tych macierzy należy zdefiniować w pierwszej kolejności. W macierzy A przechowywane są jedynie niezerowe wartości odpowiadające wartościom współczynników przyległych punktów. Dla przykładu zaprezentowano sposób tworzenia macierzy A równania Laplace'a. Dla dwóch wymiarów równanie Laplace'a zdyskretyzowane przy użyciu schematu centralnego i założeniu, że $h = \Delta x = \Delta y$ przyjmuje postać:

$$(\nabla^2 f)_i, j = (-4f_{i,j} + f_{i+1,j} f_{i-1,j} + f_{i,j+1} f_{i,j-1}) \frac{1}{h^2} \tag{5}$$

Każdemu przesunięciu przypisany zostaje numer i jeśli przyjąć kolejność (i, j) , $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$ współczynniki dla biblioteki wyniosą odpowiednio -4,1,1,1,1. Schemat został graficznie przedstawiony na rys. 2.

W kodzie obliczeniowym schemat zostanie zadeklarowany w następujący sposób:

```
HYPRE_StructStencil stencil;
int ndim=2;
int size=5;
int entry;
int offsets[][2]={{0,0},{-1,0},{1,0},{0,-1},{0,1}};
HYPRE_StructStencilCreate(ndim, size,&stencil);
for (entry = 0; entry < size; entry++)
HYPRE_StructStencilSetElement(stencil,entry,
offsets[entry]);
```



Rys. 2: Reprezentacja schematu 5-punktowego

Wykorzystując założony schemat, należy następnie wypełnić macierz A odpowiednimi współczynnikami. Do tego celu posługujemy się funkcją SetBoxValues(), która wprowadza odpowiednie wartości współczynników do wybranego boxa zdefiniowanego poprzez dwa skrajne węzły ilower oraz iupper. Na wstępie domyślnie wypełnia się cały obszar obliczeniowy właściwymi dla schematu współczynnikami. Kolejnym krokiem jest modyfikacja wartości współczynników wynikających z przyjętych warunków brzegowych. Fragment kodu odpowiedzialny za wypełnienie całej tablicy domyślnymi wartościami ma postać:

```
HYPRE_StructMatrix A;
double values[x*y];
int nentries = 5;
int stencil_indices[5] = {0,1,2,3,4};
int i;
HYPRE_StructMatrixCreate(MPI_COMM_WORLD,grid,stencil,&A);
HYPRE_StructMatrixInitialize(A);
for (i = 0; i < x*y; i+=nentries){
values[i] = -4.0;
values[i+1] = 1.0;
values[i+2] = 1.0;
values[i+3] = 1.0;
values[i+4] = 1.0;
}
HYPRE_StructMatrixSetBoxValues(A,ilower,iupper,
nentries,stencil_indices,values);
```

Zgodnie z tą procedurą, przy użyciu funkcji $SetBoxValues()$, odpowiednio modyfikujemy macierz A, tak aby odpowiadała warunkom brzegowym, nadpisując zmienną $values$ oraz wartości $ilower$ oraz $iupper$ dla interesującego nas fragmentu obszaru obliczeniowego. Analogicznie postępujemy przy wypełnianiu wektora b oraz x. Biblioteka pozwala swobodnie wykonywać operacje dodawania elementów do już istniejących wektorów oraz przemnażanie wektora przez macierze pomocnicze (które są tworzone w identyczny sposób jak zostało to przedstawione powyżej). Dokładniejszy opis stosowania biblioteki wraz z instrukcją użytkowania solverów został opisany obszerniej w dokumentacji użytkownika [4].

3. Rozwiązywanie równania Poissona

Pierwszym równaniem, jakie należy rozwiązać w metodzie Wiru w Komórce (VIC) jest równanie Poissona dla funkcji prądu i zadanej wirowości:

$$\Delta\psi = -\omega. \tag{6}$$

W poprzednich kodach obliczeniowych tworzonych i wykorzystywanych w Katedrze do dyskretyzacji funkcji prądu stosowano schemat centralny rzędu drugiego. Wraz z rozwojem możliwości obliczeniowych współczesnych komputerów zaistniała możliwość wprowadzenia dokładniejszych, choć bardziej kosztownych obliczeniowo schematów różnicowych. W kodzie została zaimplementowana kompaktowa metoda czwartego rzędu wykorzystująca 9-punktowy schemat różnicowy. Wyprowadzenie tego schematu zostało szczegółowo przedstawiono w [6],[7].

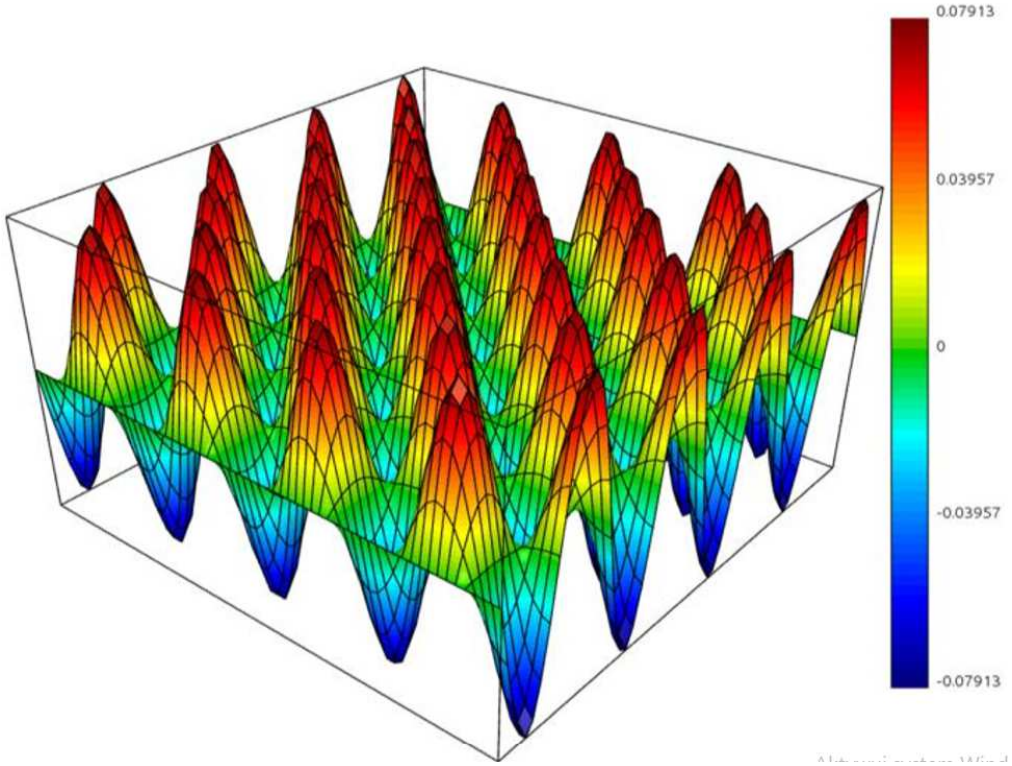
$$\begin{aligned} &a\psi_i, j + b(\psi_{i+1,j}\psi_{i-1,j}) + c(\psi_{i,j+1} + \psi_{i,j-1}) \\ &+d(\psi_{i+1,j+1} + \psi_{i-1,j-1} + \psi_{i+1,j-1} + \psi_{i-1,j+1}) \\ &= 0.5\Delta x^2(8\psi_{i,j} + \psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1}) \end{aligned} \tag{7}$$

gdzie: $a = -10(1+\gamma^2)$, $b = 5+\gamma^2$, $c = \gamma^2 - 1$, $d = (1+\gamma^2)$,$\gamma = \Delta x/\Delta y$

Powyżej przedstawiony sposób dyskretyzacji równania Poissona został przetestowany na funkcji analitycznej wraz z warunkiem Dirichleta na brzegach tak, aby możliwe było porównanie z wartością dokładną (rys. 3). Problem został przedstawiony przy użyciu następujących funkcji:

$$\begin{aligned} \omega(x,y) &= \frac{1}{-128\pi^2}(8\pi x)\sin(8\pi x) && (x,y) \in [0,1]\times[0,1] \\ \omega(0,y) &= \frac{1}{-128\pi^2}(8\pi x)\sin(8\pi x) && (y) \in [0,1] \\ \omega(1,y) &= \frac{1}{-128\pi^2}(8\pi x)\sin(8\pi x) && (y) \in [0,1] \\ \omega(x,0) &= \frac{1}{-128\pi^2}(8\pi x)\sin(8\pi x) && (x) \in [0,1] \\ \omega(x,1) &= \frac{1}{-128\pi^2}(8\pi x)\sin(8\pi x) && (x) \in [0,1] \end{aligned} \tag{8}$$

Aby sprawdzić poprawne działanie programu, jak i używanych bibliotek, został wyznaczony rząd aproksymacji, który jest zdefiniowany jako $\alpha = \epsilon_h/\epsilon_{\frac{h}{2}}$, gdzie ϵ odpowiada wartości błędu. Obliczenia były prowadzone w podwójnej precyzji przy wykorzystaniu 4 wątków, gdzie każdy wątek rozwiązywał problem na kwadracie składający się z $N \times N$ węzłów. Za kryterium zbieżności solvera układu równań liniowych przyjęto wartość residuum wynoszącą 10^{-9}. W badaniach była mierzona norma maksimum ($L_{\max} = \max|u_{\text{dokładna}} - u_{\text{obliczona}}|$), której wartość w zależności od gęstości siatki została przedstawione w tabeli 1. Na jej podstawie został wyznaczony rząd aproksymacji. Można zauważyć, że rząd wynosi 4 praktycznie w całym badanym zakresie. Oznacza to, że przy podwójnym zagęszczeniu siatki błąd dyskretyzacji równania różniczkowego maleje aż 16-krotnie.



Rys. 3: Prezentacja graficzna rozwiązania równania Poissona

Tabela 1: Wyznaczenie rzędu aproksymacji równania Poissona

L. wątków	N	El. na wątek	El. globalnie	$L_{\max}$	Rząd aproksymacji
4	8	64	256	1,26E-2	-
4	16	256	1024	9,37E-4	3,75
4	32	1024	4096	5,73E-05	4,03
4	64	4096	16384	3,45E-06	4,05
4	128	16384	65536	2,30E-07	3,91
4	256	65536	262144	1,34E-08	4,10
4	512	262144	1048576	8,37E-10	4,00
4	1024	1048576	4194304	5,25E-11	3,99

W technice obliczeniowej z użyciem wielu procesorów istotną cechą algorytmów służących do rozwiązywania równań ruchu płynu są skalowalność oraz złożoność obliczeniowa metod rozwiązywania układów równań liniowych. W zawiązku z tym celowe staje się przebadanie kodu obliczeniowego pod tym kątem.

Przeprowadzony został eksperyment numeryczny polegający na pomiarze czasów rozwiązywania równania Poissona zmieniając przy tym ilość wykorzystywanych wątków przy zachowaniu jednakowej liczby węzłów dla pojedynczego wątku lub zmianie ilości węzłów przy zachowaniu jednakowej liczby procesorów. Obliczenia wykonano na jednostce wyposażonej w 2 procesory Xeon E5-2670 (premiera Q1'2012) dysponującej łącznie 16 rdzeniami fizycznymi, w którym każdy ma 2 wątki. Komputer wyposażony był w 32GB pamięci pracującej w układzie dual-channel (po dwie kości pamięci na procesor). Wykorzystano do niego 4 różne solvery w najkorzystniejszej konfiguracji parametrów. Zostały one zamieszczone w tabeli 2.

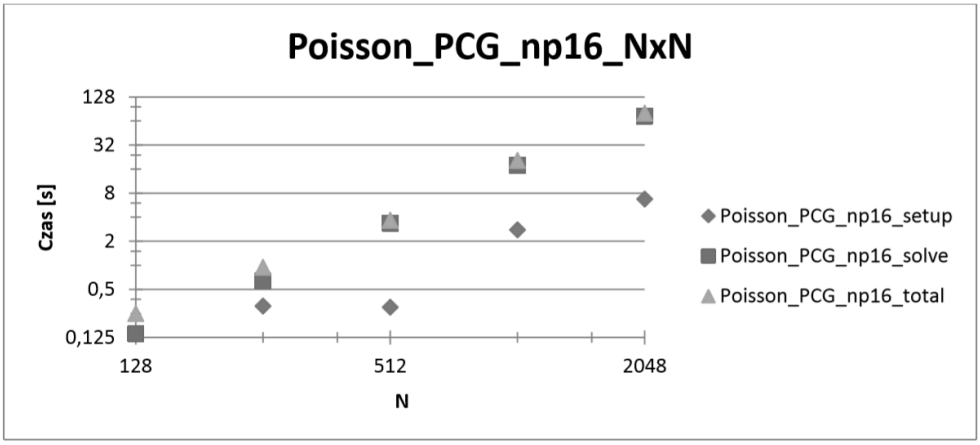
Eksperyment podzielono na dwie części. W pierwszej mierzony był czas obliczeń wykorzystując 16 wątków każdorazowo dwukrotnie zwiększając liczbę węzłów

ściany kwadratu (ilość węzłów rosła czterokrotnie). Badany był zakres od 128×128 do 2048×2048 dla pojedynczego wątku. W drugiej części liczba węzłów kwadratu dla pojedynczego wątku była stała i wynosiła 256×256. Zmianie ulegała natomiast ilość wątków które brały udział w obliczeniach. Przebadano skalowalność problemu w zakresie od 1 do 32 wątków, każdorazowo podwajając globalną wielkość problemu. Mierzony czas został rozbity na fazę przygotowania/prekondycjonowania oraz czas samego rozwiązania układu równań liniowych. Wykresy przedstawione zostały w skali logarytmicznej o podstawie 4.

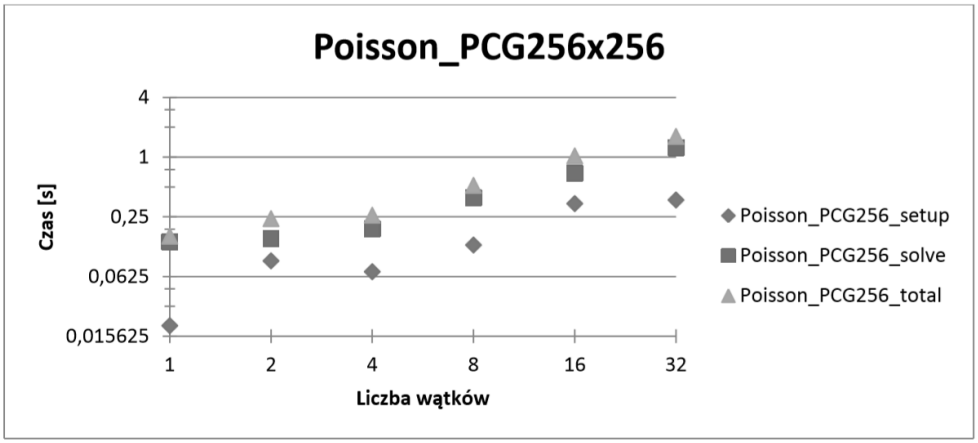
Powyższe obliczenia wykazały że złożoność obliczeniowa każdego z wybranych solverów jest rzędu $O(N)$.

Tabela 2: Konfiguracja wykorzystanych solverów

Solver	Prekondycjonowanie
PCG–preconditioned conjugated gradient	cykl SMG – V(1,1)
SMG–semicoarsening multigrid – V(1,1) cycle	brak
GMRES–generalized minimal residual	PFMG – V(1,1) cycle
PFMG–parallel semicoarsening multigrid	brak



Rys. 4: Wyniki rozwiązania solvera PCG – prekondycjonowanego gradientu sprzężonego przy stałej liczby procesów ze zmianą lokalnego rozmiaru problemu



Rys. 5: Wyniki rozwiązania solvera PCG – prekondycjonowanego gradientu sprzężonego przy stałym rozmiarze problemu lokalnego ze zmianą liczby procesów

Każdy z nich rozwiązuje układ równań liniowych w zadowalającym czasie, jednak solver GMRES okazał się być najszybszym.

Równanie Poissona składające się z 67 108 864 niewiadomych podzielone zostało na 16 procesów zostało rozwiązane w czasie 36,91 sekund. Okazało się czas rozwiązywania był dwukrotnie krótszy od solvera prekondycjonowanych gradientów sprzężonych (PCG), który na rozwiązanie tego problem potrzebował aż 80,75 s (rys. 4). Czasy rozwiązania układu równań wykorzystujących wielosiatkowe metody SMG i PFMG wynosiły odpowiednio 57,79 oraz 51,23 s.

Z powyższego testu wynika, że w celu prowadzenia wydajnych obliczeń należy przede wszystkich dobrać optymalny solver. Ilość sposobów konfiguracji oferowanych przez bibliotekę *hypre* solverów przekracza liczbę 100, więc w przypadku przeprowadzenia bardziej dokładnych badań, otrzymany czas obliczeń może zostać jeszcze bardziej skrócony. Patrząc pod kątem skalowalności solverów widoczne jest wydłużenie się czasu obliczeń wraz z dokładaniem kolejnych wątków, (rys. 5). Jest to związane z koniecznością wymiany większej ilości informacji pomiędzy poszczególnymi procesami. Powyższe wyniki dotyczące skalowania należy traktować orientacyjnie, ponieważ maksymalna liczba procesów wynosząca 32 nie jest zbyt reprezentatywną do ilości, jaką potrafią obsłużyć współczesne klastry obliczeniowe. W [3] badano skalowalność biblioteki *hypre* przy użyciu centrum obliczeniowego pracującego na 100 000 wątkach. Rezultaty dla małej ilości wątków wyglądały analogicznie. Doskonała skalowalność była możliwa do zaobserwowania przy wykorzystaniu powyżej 500 wątków. Problemy tam rozwiązywane składały maksymalnie się z 12 miliardów elementów, co jest wynikiem imponującym.

4. Wyznaczanie prędkości z funkcji prądu

Kolejnym krokiem niezbędnym do wykonania w algorytmie metody VIC jest wyznaczenie prędkości z funkcji prądu

$$\begin{aligned} u &= \partial_y \psi, \\ v &= -\partial_x \psi. \end{aligned} \tag{9}$$

Podobnie jak w przypadku równania Poissona i w tym przypadku do dyskretyzacji wykorzystano kompaktową metodę czwartego rzędu. Aby osiągnąć wyższy rząd, posłużono się interpolacją Hermite'a, w której pochodna została zainterpolowana pod postacią schematu różnicowego pochodnych. Zauważalną wadą tej metody jest konieczność znajomości wartości pochodnej na brzegu bądź też przyjęcie warunku periodyczności. Dokładniejszy opis i wyprowadzenia zostały zawarte w [8]. Gotowe do użycia w bibliotece formy do wyznaczenia składowych prędkości u i v na podstawie funkcji prądu mają postać:

$$\begin{aligned} 4\psi_{y,(i,j)} + \psi_{y,(i,j+1)} + \psi_{y,(i,j-1)} &= \frac{3}{h}(\psi_{i,j+1} + \psi_{i,j-1}) \\ 4\psi_{x,(i,j)} + \psi_{x,(i+1,j)} + \psi_{x,(i-1,j)} &= \frac{-3}{h}(\psi_{i+1,j} + \psi_{i-1,j}) \end{aligned} \tag{10}$$

Przeprowadzone zostały testy sprawdzające rząd metody oraz poprawność działania programu analogicznie do równania Poissona.

Sprawdzenia dokonano dla następujących warunków obliczeniowych

$$\begin{aligned}
&\psi(x,y) = e^{2x} + 6\sin(2\pi x), && (x,y) \in [0,1] \times [0,1] \\
&\psi_x(0,y) = 2(e^{2x} + 6\sin(2\pi x)) && (y) \in [0,1] \\
&\psi_x(1,y) = 2(e^{2x} + 6\sin(2\pi x)) && (y) \in [0,1] \\
&\psi_x(x,0) = 2(e^{2x} + 6\sin(2\pi x)) && (x) \in [0,1] \\
&\psi_x(x,1) = 2(e^{2x} + 6\sin(2\pi x)) && (x) \in [0,1]
\end{aligned} \tag{11}$$

Otrzymane wyniki przedstawione w tabeli 3 potwierdzają zakładaną dokładność aproksymacji.

Tabela 3: Rząd aproksymacji wyznaczania prędkości z funkcji prąd

L. wątków	N	El. na wątek	El. globalnie	$L_{\max}$	Rząd aproksymacji
4	2	4	16	1,95307	-
4	4	16	64	1,32 E-1	3,88
4	8	64	256	6,43E-03	4,36
4	16	256	1024	3,5E-04	4,19
4	32	1024	4096	2,1E-05	4,09
4	64	4096	16384	1,3E-06	4,05
4	128	16384	65536	7,7E-08	4,02

5. KONWEKCJA

Cząstki transportujące wirowość poruszają się jak cząstki materialne, gdzie nie działa na nie lepkość płynu. Są one unoszone w polu prędkości zgodnie z równaniem

$$\frac{dx}{dt} = u \tag{12}$$

Równanie to jest rozwiązywane przy pomocy metody Rungego–Kutty 4 rzędu z podkrokową interpolacją prędkości względem położenia cząstek o następującym algorytmie rozwiązywania

$$\begin{aligned}
k_1 &= \Delta t u(t_0, x_0), \\
k_2 &= \frac{\Delta t}{2} u(t_0, x_0 + \frac{k_1}{2}), \\
k_3 &= \frac{\Delta t}{2} u(t_0, x_0 + \frac{k_2}{2}), \\
k_4 &= \Delta t u(t_0, x_0 + k_3), \\
x_1 &= x_0 + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}.
\end{aligned} \tag{13}$$

Dla podkroków k_2, k_3, k_4 konieczne jest każdorazowo określenie wektora prędkości w jakiej się znajduje. Do tego celu stosowana jest interpolacja tej wartości na podstawie 4 najbliżej położonych węzłów. W związku z powyższym, w implementacji wieloprocesorowej, może wystąpić sytuacja, że cząstka zostanie uniesiona w obszar

leżący pomiędzy sąsiadującymi procesami. W tym celu obszar obliczeniowy każdego procesora został powiększony o jeden rząd w każdym kierunku, a następnie uzupełniony wartościami wektora prędkości swoich sąsiadów przy użyciu protokołu MPI. Każdy procesor wysyła swoje wartości z brzegu ograniczającego jego obszar obliczeniowy komendą MPI_Send, a następnie odbiera brakujące wartości dzięki komendzie MPI_Recv.

6. Redystrybucja

Znając nowe położenie cząstek niosących ze sobą wirowość zabraną z węzłów należy ją z powrotem zwrócić na siatkę. Dlatego w każdej iteracji metody VIC należy przeprowadzić redystrybucję wirowości. Wykorzystywane jest do tego jądro interpolacyjne wysokiego rzędu. Wartość nowej wirowości w stałym węźle jest otrzymywana jako suma wkładu wirowości starych cząstek pomnożonych przez jądra jednowymiarowe. W programie zostało zaimplementowane jądro Z_2 sklejane z 3 funkcji. Opisane jest następującymi równaniami

$$Z_2(x) = \begin{cases} 1 - \frac{5}{2}x^2 + \frac{3}{2}x^3 & \text{dla } 0 \leq x \leq 1 \\ \frac{1}{2}(x-1)(x-2)^2 & \text{dla } 1 < x \leq 2 \\ 0 & \text{dla } x > 2 \end{cases}$$

Aby otrzymać nową wartość wirowości w węźle posługujemy się równaniem:

$$\omega(x,y) = \sum_p \Gamma_p \varphi(\frac{x_j - x_p}{h}) \varphi(\frac{y_j - y_p}{h}) \frac{1}{h^2} \tag{14}$$

gdzie: Γ_p – wirowość starej cząstki, φ – jednowymiarowe jądro interpolacyjne, x_j, y_j – współrzędne węzła, x_p, y_p – położenie cząstki.

Implementacja wieloprocesorowa wymaga, tak jak w przypadku konwekcji cząstek, rozszerzenia obszaru obliczeniowego dla każdego procesora. W tym przypadku powiększa się obszar obliczeniowy aż o dwie warstwy w każdym kierunku. Wkłady do wirowości są przekazywane poprzez komendy MPI_Send oraz MPI_Recv.

7. Równanie dyfuzji

Po zakończeniu kroku nielepkiego, jakim jest konwekcja oraz redystrybucji wirowości na węzły, można przystąpić do rozwiązania równania symulującego lepkość

$$\frac{d\omega}{dt} = \nu \Delta \omega. \tag{15}$$

Do dyskretyzacji czasowej równania dyfuzji wykorzystany jest schemat jawno–niejawny Cranka–Nicholsona. Pozwala on na osiągnięcie drugiego rzędu dyskretyzacji po czasie. Dyskretyzacja po przestrzeni opiera się na czwartorzędowej aproksymacji Pade'go. Czwarty rząd dokładności po czasie jest możliwy do osiągnięcia poprzez ekstrapolację Richardsona połączoną z metodą ADI [9]. W tym opracowaniu zdecydowano

się nie implementować tego rozwiązania. Równanie rozwiązywane jest w dwóch krokach, co wymaga dwukrotnego rozwiązania układu równań z macierzą tridiagonalną zamiast jednego dziewięcioprzekątniowego. Zastosowany schemat różnicowy przedstawia się następująco:

$$\begin{aligned} a\omega^*_{i,j} + b(\omega^*_{i+1,j} + \omega^*_{i-1,j}) =& c\omega^n_{i,j} + d(\omega^n_{i+1,j} + \omega^n_{i-1,j} + \omega^n_{i,j+1} + \omega^n_{i,j-1}) + \\ & e(\omega^n_{i+1,j+1} + \omega^n_{i-1,j-1} + \omega^n_{i-1,j+1} + \omega^n_{i-1,j-1}), \end{aligned} \tag{16}$$

$$a\omega^{n+1}_{i,j} + b(\omega^{n+1}_{i,j+1} + \omega^{n+1}_{i,j-1}) = \omega^*_{i,j}, \tag{17}$$

gdzie: $r = \frac{\nu \triangle t}{2h^2}$, $a = 2r + \frac{5}{6}$,$b = -r + \frac{1}{12}$,$c = (-2r + \frac{5}{6})^2$,$d = (-2r + \frac{5}{6})(r + \frac{1}{12})$, $e = (r + \frac{1}{12})$.

Analogicznie jak w pozostałych równaniach przeprowadzono test sprawdzający dokładność dyskretyzacji.Po wykonaniu odpowiednio dużej liczby kroków czasowych liczony był błąd maksymalny rozwiązania względem rozwiązania dokładnego. Posłużono się następującą funkcją:

$$\begin{aligned} &\omega_0(x,y) = e^{-8t\pi^2\nu}\sin(2\pi x)\sin(2\pi y), && (x,y) \in [0,1]\times[0,1] \\ &\omega(0,y) = \sin(2\pi x)\sin(2\pi y) && (y) \in [0,1] \\ &\omega(1,y) = \sin(2\pi x)\sin(2\pi y) && (y) \in [0,1] \\ &\omega(x,0) = \sin(2\pi x)\sin(2\pi y) && (x) \in [0,1] \\ &\omega(x,1) = \sin(2\pi x)\sin(2\pi y) && (x) \in [0,1] \end{aligned} \tag{18}$$

Wyniki zamieszczone w tabeli 4 potwierdzają oczekiwany 4 rząd dokładności aproksymacji po przestrzeni. Badanie dokładności po czasie nie jest konieczne, gdyż wielokrotnie wykazano, że metoda Cranka–Nicholsona jest metodą drugiego rzędu dokładności.

Tabela 4: Wyznaczanie rzędu aproksymacji po przestrzeni dla równania dyfuzji

N	Δt	L. kroków	$L_{\max}$	Rząd aproksymacji
4	0.0001	1000	0.0105	-
8	0.0001	1000	7.15E-04	3,87
16	0.0001	1000	3.22E-05	4,47
32	0.0001	1000	1.44E-06	4,48
64	0.0001	1000	6.80E-08	4,39

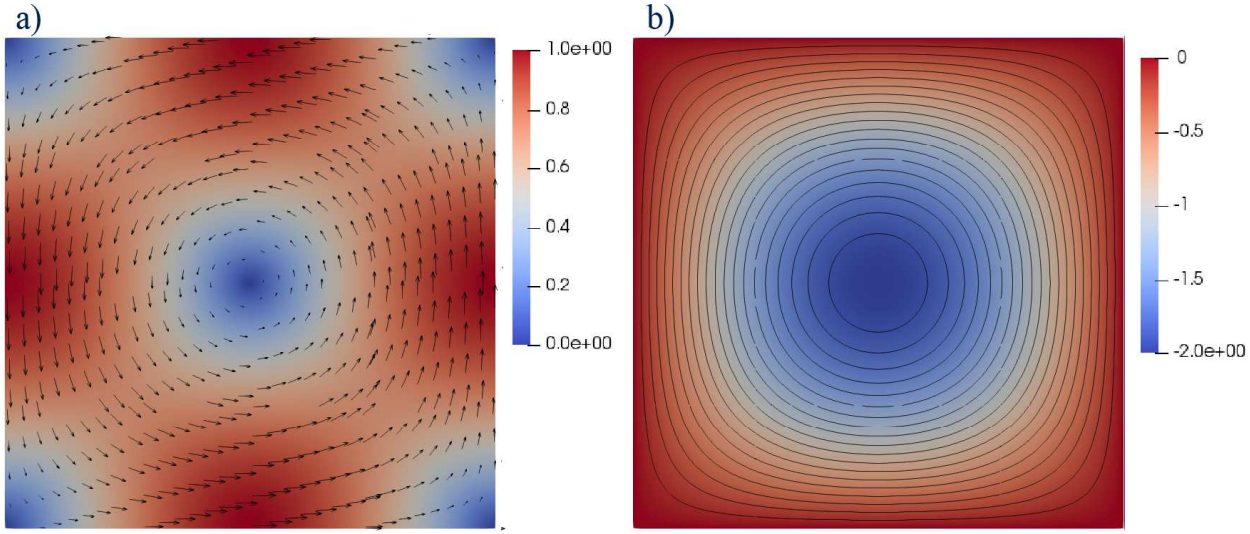
8. Zagadnienie testowe dla znanego pola wirowego

Ostatecznym testem który został wykonany w celu określenia dokładności algorytmu oraz stwierdzenia poprawności stworzonego silnika obliczeniowego jest zagadnienie ruchu płynu ze znanym polem wirowym. Posłużono się funkcjami na dwuwymiarowy wir Taylora–Greena. W omawianym przypadku funkcję zmodyfikowano tak,

aby otrzymać pojedynczą łatę wirową. Rozwiązanie dokładne opisane jest następującym zestawem funkcji

$$\begin{aligned} u(x,y,t) &= \cos(-\frac{\pi}{2}+x)\sin(-\frac{\pi}{2}+y)e^{-2t\nu}, \\ v(x,y,t) &= -\sin(-\frac{\pi}{2}+x)\cos(-\frac{\pi}{2}+y)e^{-2t\nu}, \\ \psi(x,y,t) &= -\cos(-\frac{\pi}{2}+x)\cos(-\frac{\pi}{2}+y)e^{-2t\nu}, \\ \omega(x,y,t) &= -2\cos(-\frac{\pi}{2}+x)\cos(-\frac{\pi}{2}+y)e^{-2t\nu}. \end{aligned} \tag{19}$$

Obliczenia były prowadzone na kwadracie w obszarze $\Omega = \{0{\leq}x{\leq}\pi, 0{\leq}y{\leq}\pi\}$. Jako początkowy warunek brzegowy przyjęto rozkład wirowości z wartości dokładnej dla czasu $t = 0$ s. Prędkość na brzegach obszaru obliczeniowego również aktualizowana była co krok czasowy zgodnie z równaniem (19). Wartość kinematycznego współczynnika lepkości była jednakowa we wszystkich próbach i wynosiła $\nu = 0.002$. Dla równania dyfuzji przyjęto, że wartość wirowości na brzegu będzie zgodna z wartością dokładną i będzie wynosić 0. Miarą błędu była norma maksimum po upływie czasu $t = 12$ s względem prędkości. Rozwiązanie dla chwili $t = 0$ s zostało przedstawione na rys. 6, natomiast rozkład błędu rozwiązania po czasie $t = 12$ s został przedstawiony na rys. 7. Można zaobserwować, że ze względu na fakt, że zagadnienie jest symetryczne w obu płaszczyznach błąd również charakteryzuje się symetrią. Największe jego wartości osiągane są w obszarze największej prędkości, co może wskazywać na utratę dokładność przy konwekcji cząstek i redystrybucji.



Rys. 6: a) rozkład prędkości wraz z zaznaczonymi wektorami; b) rozkład wirowości wraz z izoliniami funkcji prądu dla chwili $t = 0$ s (izolinie występują co 0.05)

Wyniki zamieszczone w tabelach 5 i 6 wskazują poprawność działania całego algorytmu. Zgodnie z oczekiwaniami uzyskano drugi rząd dokładności po przestrzeni oraz pierwszy po czasie.

Pomimo stosowania schematów różnicowych wysokiego rzędu, procedura redystrybucji jest nadal metodą rzędu drugiego. Skutkuje to obniżeniem globalnej dokład-

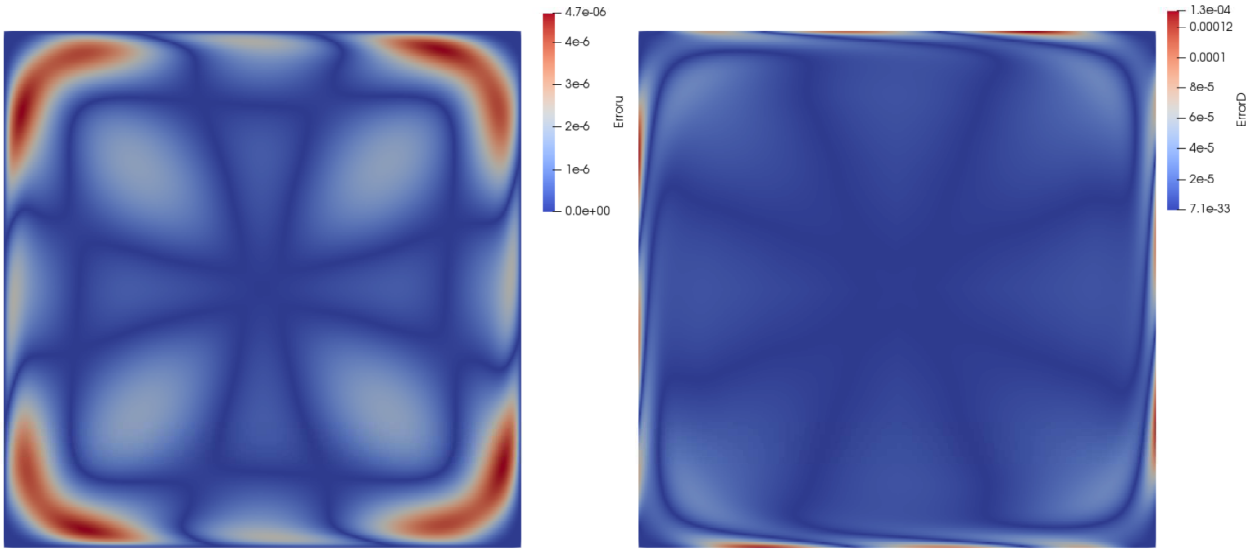
Tabela 5: Wyznaczanie rzędu dokładności metody VIC po przestrzeni i po czasie

N	Δt	L. kroków	$L_{\max}$	Rząd aproksymacji
32	0.04	300	4.10E-03	-
64	0.02	600	7.52E-04	2.45
128	0.01	1200	1.57E-04	2.26
256	0.005	2400	3.50E-05	2.16
512	0.0025	4800	8.21E-06	2.09

Tabela 6: Wyznaczanie rzędu dokładności aproksymacji metody VIC po czasie

N	Δt	L. kroków	$L_{\max}$	Rząd aproksymacji
128	0.01	1200	1.57E-04	-
128	0.005	2400	8.51E-05	0.88
128	0.0025	4800	4.43E-05	0.94
128	0.00125	9600	2.26E-05	0.97
128	0.000625	19200	1.13E-05	1.00

ności całego programu. Ze względu na wysokie skomplikowanie i duży koszt obliczeniowy przy użyciu dotychczas stosowanych jąder interpolacyjnych ten krok nie został jeszcze w pełni udoskonalony. Przy redystrybucji rzędu czwartego wymagana jest znajomość wirowości na kroku czasowym wcześniejszym co najmniej w 36 najbliższych komórek w przypadku obliczeń dwuwymiarowych. Dla obliczeń w trzech wymiarach wynosi on aż 216. Sposobem na obejście tej trudności może być stosowanie konwekcji i redystrybucji w ujęciu jednowymiarowym, gdzie rozwiązywane są po kolei pojedyncze wymiary. Niski rząd dokładności po czasie jest spowodowany dekompozycją lepkościową, która prowadzi do obniżenia globalnego rzędu dokładności do poziomu 1.



Rys. 7: Rozkład błędu prędkości oraz wirowości dla chwili $t = 12$ przy $N = 512$ i $\Delta t = 0.0025$

W celu sprawdzenia poprawności algorytmu dla przypadku, gdy w obszarze obliczeniowym występują ścianki sztywne, których należy zrealizować warunek braku poślizgu (ang. *no slip wall condition*), przeprowadzono dodatkowy test. W tym przypadku na brzegach w równaniu dyfuzji, zamiast określonej wartości wirowości, należy wyliczyć taką wartość wirowości, aby składowa prędkości styczna do ściany była równa 0.

Do określenia tej wartości skorzystano z trzeciorzędowej formuły Briley'a, która wykorzystuje 3 najbliższe wartości funkcji prądu w kierunku normalnym do ściany w liczonym węźle oraz wartość prędkości z jaką porusza się ścianka:

$$\omega_{0,j} = \frac{-66u_{s\,0,j} + 108\psi_{1,j} - 27\psi_{2,j} + 4\psi_{3,j}}{18h^2}. \tag{20}$$

Pomimo zmiany warunku brzegowego dla wirowości na ścianie w równaniu dyfuzji rzędy aproksymacji dla całej metody pozostały niezmienione w stosunku do przypadku z wartością wirowości wziętą z rozwiązania dokładnego. Maksymalna wartość błędu w całym obszarze obliczeniowym jest natomiast o rząd wielkości większa. Maksymalna wartość błędu występowała na brzegu, gdzie generowana wartość wirowości była różna od wartości dokładnej (równej 0 dla wszystkich brzegów). Powyższy test potwierdził poprawność działania algorytmu dla przypadków przepływów ze ścianką sztywną. Wyniki dla tego testu zostały przedstawione w tabelach poniżej.

Tabela 7: Wyznaczanie rzędu dokładności metody VIC po przestrzeni i po czasie przy ściankach sztywnych

N	Δt	L. kroków	$L_{\max}$	Rząd aproksymacji
32	0.04	300	2.08E-03	-
64	0.02	600	3.65E-04	2.51
128	0.01	1200	8.20E-05	2.16
256	0.005	2400	1.94E-05	2.08
512	0.0025	4800	4.69E-06	2.05

Tabela 8: Wyznaczanie rzędu dokładności aproksymacji metody VIC po czasie przy ściankach sztywnych

N	Δt	L. kroków	$L_{\max}$	Rząd aproksymacji
128	0.01	1200	8.20E-05	-
128	0.005	2400	4.62E-05	0.83
128	0.0025	4800	2.44E-05	0.92
128	0.00125	9600	1.25E-05	0.96
128	0.000625	19200	6.35E-06	0.98

9. Podsumowanie

W pracy przedstawiona została implementacja metody cząstek wirowych typu VIC wykorzystująca metodę dekompozycji lepkościowej. Dokonano szczegółowych analiz dotyczących zachowania odpowiedniej dokładności oraz wydajności poszczególnych schematów różnicowych. Udowodniono możliwość rozwiązywania dużych problemów obliczeniowych przy zachowaniu krótkich czasów obliczeniowych poprzez wykorzystanie wysokowydajnej zrównoleglonej biblioteki *hypre*. Ostatecznie przedstawiono zagadnienie testowe ze znanym polem wirowym w którym wykazano poprawność działania solvera. Przedstawiono również metody i kierunki dalszej poprawy silnika obliczeniowego.

Literatura

[1] Kudela H., Malecha Z.M., *Investigation of unsteady vorticity layer eruption induced by vortex patch using vortex particles method*, Journal of Theoretical and Applied Mechanics,**45**, 485–800, 2007.

[2] Kudela H., *Matematyczne wprowadzenie do mechaniki płynów*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2016.

[3] Baker A.H., Falgout R.D., Kolev Tz.V., Yang U.M., *Scaling hypre's multigrid solvers to 100,000 cores*, High Performance Scientific Computing: Algorithms and Applications, 261–279, Springer, London, 2012.

[4] Baker A.H., Falgout R.D., Gamblin T., Kolev Tz.V., Schulz M., Yang U.M., *Scaling algebraic multigrid solvers: On the road to exascale*, Competence in High Performance Computing, 215–226, Springer, Berlin, 2010.

[5] Lawrence Livermore National Laboratory, *hypre user's manual.*, 2012.

[6] Wang Y., Zhang J., *Sixth order compact scheme combined with multigrid method and extrapolation technique for 2D poisson equation*, Journal of Computational Physics, **228**, 137–146, 2009.

[7] Zhang J., *Multigrid method and fourth order compact difference scheme for 2D Poisson equation with unequal meshsize discretization*, Journal of Computational Physics, **179**, 170–179, 2002.

[8] Fishelov D., Ben-Artzi M., Croisille J.–P, *Recent advances in the study of a fourth-order compact scheme for the one-dimensional biharmonic equation*, Journal of Scientic Computing, **53**, 55–79, 2012.

[9] Liao, W., Zhu, J. and Khaliq, A. Q.M., *An efficient high-order algorithm for solving systems of reaction-diffusion equations.*, Numerical Methods Partial Differential Eq., **18**, 340–354, 2002.