Na prawach rękopisu

Politechnika Wrocławska
Wydział Mechaniczno-Energetyczny

Raport serii PREPRINTY nr __/2015

Modelowanie ewolucji trójwymiarowych struktur wirowych w cieczy lepkiej metodami cząstek wirowych z wykorzystaniem obliczeń równoległych

Andrzej Kosior

Słowa kluczowe: metoda cząstek wirowych
rekonekcja struktur wirowych
obliczenia równoległe
karty graficzne

Praca doktorska

Promotorzy: dr hab. inż. Henryk Kudela, prof. PWr,
prof. dr hab. inż. Zbigniew Huzar

WROCŁAW, 2015

Pragną podziękować moim promotorom panu Profesorowi Henrykowi Kudeli i panu Profesorowi Zbigniewowi Huzarowi, bez pomocy których ta praca by nie powstała. Gorąco dziękuję także mojej rodzinie za wytrwałość.

Streszczenie

Wzajemne oddziaływania pola prędkości i wirowości powodują, że wirowość w przepływie ma tendencję do koncentracji i układania się w spójne (koherentne) struktury przypominające rurki. Dwa podstawowe procesy jakim poddawane są te struktury to oddziaływanie ze sztywną ścianą oraz oddziaływanie w innymi strukturami. W wyniku tego drugiego procesu może dojść do zjawiska zmiany topologii linii wirowych inaczej nazywanym także rekonekcją struktur wirowych. Jest ono bardzo powszechne w przyrodzie aczkolwiek nie jest szeroko opisane w literaturze. Uważa się, że ma ono decydujące znaczenie na ewolucję płynu.

W niniejszej pracy do badań numerycznych ewolucji i interakcji struktur wirowych została wybrana trójwymiarowa metoda cząstek wirowych. W metodzie tej używane są cząstki przenoszące informację o polu wirowości. W używanym algorytmie są one inicjalizowane zgodnie z początkowym (zadanym) polem wirowości na strukturalnej siatce numerycznej rozpiętej na obszarze obliczeniowym. Na podstawie znajomości pola wirowości można wyznaczyć pole prędkości rozwiązując równania Poissona na tzw. potencjał wektorowy. Do symulacji przepływu lepkiego został wykorzystany algorytm dekompozycji lepkościowej. Polega on na rozbiciu każdego kroku czasowego metody na dwa podkroki. W pierwszym rozwiązywane jest zagadnienie przepływu nielepkiego, w którym na podstawie twierdzenia Helmholtza cząstki niosące informację o wirowości unoszone są przez przepływ tak samo jak cząstki materialne. Następnie informacja o wirowości jest interpolowana z powrotem na węzły siatki numerycznej. W drugim podkroku symulowany jest wpływ lepkości poprzez rozwiązanie równania dyfuzji na siatce numerycznej.

Rozwiązywanie równań ruchu cieczy dla zagadnień trójwymiarowych, niezależnie od użytej metody, wiąże się z długimi czasami obliczeń. Przyrost mocy obliczeniowej komputerów, związany ze zwiększeniem częstotliwości zegara taktującego pracę procesora, w ostatnich latach zdecydowanie się zmniejszył. Dlatego w niniejszej pracy do przyspieszenia obliczeń zostały wykorzystane obliczenia równoległe na kartach graficznych. Metoda cząstek wirowych doskonale nadaje się do wykorzystania w obliczeniach równoległych. Najbardziej czasochłonna część – ruch cząstek i interpolacja wirowości na węzły – ma charakter lokalny i jest niezależna od pozostałych cząstek. Dzięki temu możliwe jest proste mapowanie jednej cząstki na jeden proces równoległy a tym samym efektywne przyspieszenie obliczeń. W celu zwiększenia dostępnej pamięci RAM a tym samym możliwości stosowania gęstszych siatek numerycznych i liczby cząstek wirowych wykorzystane została opracowana implementacja metody cząstek wirowych wykorzystująca do obliczeń wiele kart graficznych. Zastosowano programowanie hybrydowe MPI-OpenMP. Pozwoliło to na przeniesienie obliczeń na klastry obliczeniowe i wykorzystanie dowolnej dostępnej w systemie liczby kart graficznych niezależnie od konfiguracji sprzętowej. Dzięki temu można było połączyć przyspieszenie obliczeń (dla jednej karty graficznej uzyskano ok. 50-krotne skrócenie czasu obliczeniowego względem pojedynczego rdzenia procesora) oraz duży obszar obliczeniowy (dzięki połączeniu pamięci RAM kilku kart).

W pracy zostało zbadane zagadnienie rekonekcji struktur wirowych. Opisany został przebieg tego zjawiska w zależności od różnych geometrii początkowych rurek wirowych. Zbadany został wpływ liczby Reynoldsa na przebieg tego procesu. Przedstawione zostało zjawisko czołowego zderzenia dwóch pierścieni wirowych.

Abstract

Mutual interactions of the velocity and the vorticity fields cause the vorticity to concentrate in coherent structures resembling tubes (hence called vortex tubes). Those structures undergo two basic processes: interaction with a solid wall and interaction with other structures. During that second process a phenomena called as a reconnection of the vortex lines may occur. It is very common in the nature but is still not well explained in the literature. It is believed that it plays important role in the fluid motion.

In this thesis a three-dimensional vortex particle method was used to study the evolution of vortex structures. In this method particles that carry information about vorticity field are used. In the algorithm used the particles are initiated according to the initial distribution of vorticity field on a numerical grid. The velocity field is calculated from the so called vector potential obtained from the solution of the the Poisson equation. To simulate the viscous flow the so called viscous splitting algorithm was used. It involves splitting each time step into two substeps. In first the incompressible fluid flow equation is solved. In the second substep the viscous effects are modelled by the solution of the diffusion equation. In the algorithm used particles are remeshed onto the nodes of the numerical grid in every time step.

Solving the three-dimensional fluid flow equations is a time-consuming process independently on the method used. In the recent years the computational power of a single core of a processor has stopped rising. This is the reason why multiprocessor architectures have to be used to shorten the computational times. Surprisingly, graphics processing units (GPUs), built mainly for video games, can be used for scientific calculations. Vortex particle method suits very well for parallel computations. Its most time-consuming part - movement of the particles and interpolation of the vorticity to the numerical grid nodes has a local character and is independent on other particles/nodes. Due to this fact a single particle/node computations may be mapped onto a single computational thread and thus calculations may be run in parallel. In order to increase the amount of the available RAM memory and thus the use of denser meshes a multiGPU version of the VIC method was implemented. For the communication between the cards a hybrid MPI-OpenMP programming was used. This allowed for the implementation of the VIC method to be executed on clusters of computers with different configurations of nodes and number of GPUs. The parallel implementation of the VIC method allowed for nearly 50 times faster computations comparing to a single core.

In the thesis a reconection of the vortex tubes phenomena was investigated. A detailed description of this process was given depending on a different initial configuration of the vortex tubes. An influence of the Reynolds number on the occurrence of the reconnection process was investigated. A head-on collision of two vortex rings was presented.

Spis treści

Wykaz ważniejszych oznaczeń

$\mathbf{a}$	–	wektor współrzędnych cząstki wirowej w zmiennych Lagrange'a
A	–	macierz współczynników
$\mathbf{A}$	–	potencjał wektorowy
$\mathbf{b}$	–	wektor prawych stron układu równań
$\mathcal{D}$	–	przestrzeń trójwymiarowa
$\mathbf{e}$	–	wektor błędów
E	–	enstropia
$d\mathbf{l}$	–	element linii materialnej
$\mathbf{f}$	–	wektor sił masowych
h	–	krok siatki numerycznej
K	–	energia
L	–	impuls kątowy
p	–	ciśnienie
P	–	impuls liniowy
r_0	–	promień rdzenia struktury wirowej
$\mathbf{r}$	–	wektor residuów
R_0	–	promień pierścienia wirowego
$\mathcal{S}$	–	brzeg przestrzeni trójwymiarowej $\mathcal{D}$
Re	–	liczba Reynoldsa
t	–	czas
$\mathbf{u}$	–	wektor prędkości płynu
x, y	–	współrzędne kartezjańskie
$\mathbf{x}$	–	wektor niewiadomych
$\boldsymbol{\alpha}_p$	–	wektor intensywności cząstki wirowej
Γ	–	cyrkulacja
Δt	–	krok czasowy
ν	–	kinematyczny współczynnik lepkości płynu
φ	–	jądro interpolacyjne
ρ	–	gęstość płynu
$\boldsymbol{\omega}$	–	wirowość

Rozdział 1

Wprowadzenie

1.1. Wprowadzenie – przedmiot rozprawy

Badanie dynamiki wirowości jest jedną z podstawowych metod do objaśnienia i kontrolowania ruchu płynów. W przepływach rzeczywistych pola prędkości i wirowości wzajemnie na siebie oddziałują, powodując koncentrację wirowości w pewnych obszarach i tworzenie przez nią spójnych (koherentnych) struktur. Struktury te z racji swojego kształtu nazywane są rurkami lub pierścieniami wirowymi. To one decydują o kształcie i charakterze przepływu. Obrazy rzeczywistych przepływów turbulentnego uzyskane w czasie eksperymentów pokazują, że składa się on z wielu małych struktur wirowych oddziałujących ze sobą.

Pierścienie wirowe są dobrze znanymi i popularnymi obiektami w mechanice płynów. Badaczy przyciągają różnorodne zjawiska w których uczestniczą te struktury takie, jak gra wirów czy rekonekcja oraz ich zdolność do utrzymywania się w przepływie. Powstawanie pierścieni wirowych można zaobserwować w wielu eksperymentach związanych z warstwą przyścienną czy opływem ciał. Świetnie nadają się do prostych symulacji mających na celu zbadanie poszczególnych zjawisk składających się na bardziej skomplikowane przepływy. Nawet proste geometrie początkowe struktur wirowych mogą w czasie ewolucji prowadzić do bardzo skomplikowanego pola prędkości.

W pracy został podjęty temat zbadania wzajemnych interakcji struktur wirowych.

Badania struktur wirowych doczekały się bogatej literatury. Duża grupa publikacji została poświęcona pierścieniom wirowym i ich zachowaniu. Najczęściej podejmowane tematy to prędkości przemieszczania się pierścienia [102, 24, 91], jego stabilność [104] oraz wzajemne oddziaływanie pierścieni, takie jak np. tzw. gra wirów [106]. Podjęte zostały także próby numerycznego modelowania tego zjawiska. Przykładowe badania związane z prędkością przemieszczania się pierścienia

wirowego można znaleźć w [39], a jedną z pierwszych udanych prób odtworzenia zjawiska gry wirów w [94].

Dużo wysiłku zostało włożone w próby zrozumienia tzw. zjawiska rekonekcji struktur wirowych. Zjawisko to odpowiada za zmianę topologii linii wirowych w przepływie (rozumianej jako przecięcia się ich patrząc z pewnego ustalonego kierunku). Pierwsze próby zrozumienia tego zjawiska zostały opisane w [93]. W badaniach występuje wiele konfiguracji początkowych, jak na przykład: dwie antyrównoległe rurki wirowe [40, 75, 37, 44, 72, 25, 36, 100, 99], dwie prostopadłe rurki wirowe [107, 11], dwa pierścienie wirowe [108, 45] oraz inne [42, 78, 46].

Bogactwo literatury pokazuje, że jest to popularne zagadnienie. Mimo dużego wysiłku badawczego zjawisko to nie jest do końca zrozumiane, a jego powszechność w przepływach naturalnych mobilizuje do dalszej pracy związanej z lepszym poznaniem mechanizmów nim rządzących. Dodatkową motywacją może być fakt, że uważa się, że rekonekcje struktur wirowych mogą być podstawowym mechanizmem w tworzeniu się przepływów turbulentnych. Bardzo proste geometrie początkowe struktur wirowych po pewnym czasie ewolucji, w którym miało miejsce kilka takich zjawisk mogą prowadzić do powstawania przepływu o widmie charakterystycznym dla przepływu turbulentnego [41].

Szczególnie ciekawy jest eksperyment związany z rekonekcją struktur wirowych przedstawiony w pracy [69]. Przedstawione jest zderzenie czołowe dwóch pierścieni w wyniku którego została otrzymana seria mniejszych pierścieni. Eksperyment ten wzbudził ogromne zainteresowanie i poruszenie w środowisku związanym z mechaniką płynów. Autorowi do momentu pisania tej pracy (ponad 20 lat od publikacji) nie jest znana udana próba numerycznego powtórzenia tego eksperymentu. Taka próba została podjęta w niniejszej pracy.

Głównym źródłem wiedzy na temat interakcji pomiędzy strukturami wirowymi jest wizualizacja przepływu. Niestety w przepływie lepkim markery używane do wizualizacji (takiej jak np. barwnik) nie przemieszczają się zgodnie z polem wirowości. Budzi to wątpliwości czy metoda ta jest w stanie poprawnie odtworzyć wszystkie cechy przepływu, szczególnie przy długich czasach badania [74, 45]. W ostatnich latach często wykorzystywane są w badaniach metody optyczne takie jak PIV czy LDA. Są to jednak wtórne metody wyznaczania pola wirowości obarczone błędem zarówno pomiarowym, jak i wykorzystywanej metody różniczkowania. Na tym tle symulacje numeryczne wydają się być odpowiednim podejściem do badania wzajemnych oddziaływań pomiędzy strukturami wirowymi. Biorąc to pod uwagę, szczególne miejsce zajmują metody cząstek wirowych. W metodach tych elementami obliczeniowymi są cząstki niosące informację o wirowości. Dzięki badaniu ewolucji tych cząstek w przepływie możliwe jest odtworzenie zachowania się pola wirowości, a co za tym idzie również pola prędkości. Wyróżniane są dwie główne grupy metod wirowych:

- metody bezpośrednie, w których prędkość cząstki obliczana jest na podsta-

wie prawa Biota–Savarta poprzez sumowanie wkładów (indukcji) wszystkich pozostałych cząstek w przepływie na prędkość danej cząstki,

- metody łączące w sobie podejścia Eulerowskie i Lagrangowskie, takie jak metoda "Wir–w–komórce" (*ang.* Vortex–in–Cell, VIC), w których pole prędkości jest wyznaczane poprzez rozwiązanie równania Poissona na potencjał wektorowy na siatce numerycznej.

Metody wirowe zastosowane do symulacji przepływów nieściśliwych mają trzy podstawowe cechy. Po pierwsze, równania Naviera-Stokesa lub Eulera są przedstawione w sformułowaniu wirowym (w zmiennych wirowość-prędkość) i przez to dyskretyzacja jest dokonywana na polu wirowości zamiast na polu prędkości. Po drugie, do obliczeń wprowadza się Lagrangowskie cząstki wirowe unoszone przez przepływ. I po trzecie, wykorzystując definicję wirowości w postaci $\boldsymbol{\omega} = \nabla \times \mathbf{u}$, pole prędkości $\mathbf{u}$ wyznacza się z rozkładu wirowości. Jest to tak zwane prawo Biota-Savarta, dzięki któremu możliwe jest wyznaczenie pola prędkości tylko poprzez śledzenie elementów wirowych.

Przedstawione rozwiązanie ma swoje wady i zalety. Opisywanie przepływu poprzez wirowość jest pożądane głównie ze względu na jej zwarty obraz, szczególnie w skomplikowanych i niestacjonarnych przepływach. Spójne obszary wirowości przemieszczające się w postaci struktur wirowych są skoncentrowane przeważnie w małym obszarze mają decydujący wpływ na obraz pola prędkości w całym przepływie. Dzięki temu możliwe jest stosowanie mniejszych obszarów obliczeniowych, a co za tym idzie skrócenie czasu obliczeniowego. Drugą zaletą jest brak pola ciśnienia w otrzymanych równaniach. Wyznaczane jest ono tylko w przypadku, kiedy obliczane są siły w przepływie.

Metody wirowe w obecnej postaci zrodziły się w latach 70 poprzedniego stulecia. Zaangażowani w ten proces byli m.in. A. Chorin [15] i A. Leonard [65, 20]. Duże zainteresowanie w latach 80 skupiło się na aspektach matematycznych metody cząstek wirowych, takich jak zbieżność [7, 8, 16]. W późniejszych latach rozwijane były głównie metody dokładnego odwzorowania wpływu lepkości oraz warunków brzegowych na ściankach sztywnych oraz zmniejszenie kosztu obliczeniowego tak, aby metody te mogły być stosowane do symulacji z dużą dokładnością przepływów z wysoką liczbą Reynoldsa. Obszerną recenzję rozwoju metod wirowych i ich zastosowań można znaleźć w [83, 100]. Szczególnie ważną pozycją dla tej dziedziny jest książka poświęcona wyłącznie temu tematowi [17]. Zawiera ona wiele praktycznych uwag oraz implementacji metody.

Bezpośrednie metody wirowe mają, z teoretycznego punktu widzenia, wiele zalet. Wywodzą się bezpośrednio z równań Naviera-Stokesa, są bezsiatkowe i pozwalają na dokładne spełnienie warunku w nieskończoności (dalekiego pola). Pomimo szybkiego rozwoju metody wirowe nie weszły do głównego nurtu CFD. Często uważane są one za mało dokładne próby modelowania przepływów niestacjonarnych o dużej złożoności, takich z jakimi nie radzą sobie standardowe

metody CFD. Cztery główne przyczyny stanęły na drodze metod wirowych do zaistnienia w głównym nurcie CFD:

i) złożoność obliczeniowa procesu wyznaczania prędkości z użyciem prawa Biota-Savarta, analogicznie do zagadnienia n-ciał, jest rzędu drugiego ($\mathcal{O}(N^2)$),

ii) problemy z dodawaniem wpływu efektów lepkich do sformułowania Lagrangowskiego, dyfuzja wydaje się być łatwiej liczona w metodach używających siatki obliczeniowej,

iii) koncentracja cząstek w czasie ewolucji powodująca błąd związany z utratą dokładności dyskretyzacji,

iv) wprowadzenie do obliczeń często arbitralnych wielkości takich jak np. promień obcięcia.

Czas obliczeniowy związany z wyznaczeniem pola prędkości można skrócić na co najmniej dwa sposoby. Pierwszym jest zastosowanie Metody Wielopolowej (ang. Fast Multipole Method) [30], która wywodzi się z rozwiązywania zagadnień astronomicznych. Drugim jest tzw. metoda Wir-w-komórce [16, 17, 18], która do obliczeń wykorzystuje pomocniczą siatkę obliczeniową. Zastosowanie tej metody może skrócić czas obliczeń nawet 1000-krotnie [18], jednak wprowadza ona błąd związany z interpolacją wartości wirowości z cząstek na siatkę. Duży wysiłek w ostatnich latach został włożony w badania nad symulacją lepkości. Doprowadziło to do powstania wielu schematów numerycznych, takich jak Metoda Przypadkowego Błądzenia [15] czy PSE (*ang.* Particle Strength Exchange). Błędy związane z zaburzeniem rozkładu cząstek zostały zniwelowane poprzez zastosowanie tzw. "remeshingu", który po pewnej liczbie kroków czasowych tworzy nowy rozkład cząstek wykorzystując do tego pomocniczą siatkę kartezjańską i jądra interpolacyjne wysokich rzędów. Pozwala to na symulacje długich czasów przepływów jednak budzi kontrowersje wprowadzając siatkę do metod bezsiatkowych oraz pewien błąd interpolacji.

W niniejszej pracy do badań numerycznych została wybrana trójwymiarowa metoda cząstek wirowych typu "Wir-w-komórce". W metodzie tej używane cząstki przenoszące informację o polu wirowości są inicjalizowane zgodnie z początkowym (zadanym) polem wirowości na strukturalnej siatce numerycznej rozpiętej na obszarze obliczeniowym. Na podstawie znajomości pola wirowości pole prędkości wyznaczane jest poprzez rozwiązanie równań Poissona na tzw. potencjał wektorowy na siatce numerycznej. Do symulacji przepływu lepkiego korzysta się z algorytm dekompozycji lepkościowej. Polega on na rozbiciu każdego kroku czasowego metody na dwa podkroki. W pierwszym rozwiązywane jest zagadnienie przepływu nielepkiego, w którym na podstawie twierdzenia Helmholtza cząstki

niosące informację o wirowości unoszone są przez przepływ tak samo jak cząstki materialne. Następnie informacja o wirowości jest interpolowana z powrotem na węzły siatki numerycznej. W drugim podkroku symuluje się wpływ lepkości poprzez rozwiązanie równania dyfuzji na siatce numerycznej. Aby zapobiec skupianiu się cząstek w obszarach o dużym gradiencie prędkości stosowany jest „remeshing" w każdym kroku czasowym. Jednak liczba kroków czasowych po których dokonuje się „remeshingu" jest dość arbitralna. Przedstawione podejście pozwala na skrócenie czasu obliczeniowego związanego z wyznaczeniem pola prędkości, uniknięcie interpolacji nowych wartości wirowości z powrotem na stare położenia cząstek oraz dokładne rozwiązanie równania dyfuzji.

Rozwiązywanie równań ruchu cieczy dla zagadnień trójwymiarowych, niezależnie od użytej metody, wiąże się z długimi czasami obliczeń. Przyrost mocy obliczeniowej komputerów, związany ze zwiększeniem częstotliwości zegara taktującego pracę procesora w ostatnich latach zdecydowanie się zmniejszył. Dlatego aby przyspieszyć obliczenia należy stosować wieloprocesorowe środowiska obliczeniowe i odpowiednie dla nich algorytmy. Niespodziewanie okazało się, że budowane do gier komputerowych karty graficzne, posiadające niekiedy kilkaset procesorów strumieniowych, można wykorzystać do obliczeń naukowych.

Środowisko wieloprocesorowe kart graficznych zostało wybrane, ponieważ dobrze pasowały charakterystykami do rozwiązywanego zadania. Najdłuższy czas w wersji sekwencyjnej programu do obliczeń metodą cząstek wirowych typu "Wir w komórce" zajmowało przesunięcie (translacja) cząstek oraz dystrybucja wirowości z cząstek na węzły siatki. Obydwa te zadania są niezależne i mogą być wykonywane równolegle dla wszystkich cząstek. Karty graficzne są budowane w architekturze SIMD (*ang.* Single Instruction Multiple Data). Oznacza to, że wszystkie procesory strumieniowe muszą wykonywać tą samą operację w tym samym czasie (wiele strumieni danych jest przetwarzanych przez jeden strumień rozkazów). Rzeczywisty opis działania karty graficznej odbiega od tego modelu, ale jest on wystarczający na potrzeby tego wprowadzenia.

W pracy do obliczeń wykorzystane zostały karty graficzne firmy NVIDIA. Spowodowane było to udostępnieniem przez producenta języka programowania "C for CUDA", będącego rozszerzeniem popularnego języka C. Pozwala on na proste rozpoczęcie programowania na kartach graficznych, jak również efektywne wykorzystanie ich potencjału obliczeniowego. Możliwe jest wykorzystanie części już istniejącego kodu napisanego w języku C. Na nowo muszą zostać przepisane wyłącznie operacje wykonywane na karcie graficznej i zarządzanie transferami pamięci. W pierwszym etapie powstawania kodu dostępna była jedynie dokumentacja producenta oprogramowania i kart [3]. W kolejnych latach zaczęła pojawiać się coraz większa literatura z tego zakresu [47, 89].

Architektura karty graficznej doskonale nadaje się do zrównoleglenia operacji rachunkowych związanych z pojedynczą cząstką wirową. W metodzie VIC wystę-

pują jednak także operacje związane z rozwiązywaniem układu równań liniowych. Algorytmy wykorzystywane w wersji programu realizującego metodę cząstek wirowych działającej na procesorze, jak np. metoda szybkiego rozwiązywania równania Poissona (*ang.* Fast Poisson Solver) [95] mają strukturę sekwencyjną i nie da się ich efektywnie zrównoleglić. Dlatego w pracy zostały zaimplementowane dwie metody rozwiązywania układów równań liniowych: metoda wielosiatkowa (*ang.* Multigrid method) i metoda gradientów sprzężonych (*ang.* conjugate gradient method).

W przeciwieństwie do metody FPS [95], która jest metodą dokładną, opisane poniżej metody są metodami iteracyjnymi [12]. Oznacza to, że rozwiązanie otrzymywane jest w procesie wyliczania kolejnych, coraz lepszych, przybliżeń. Jakość przybliżenia jest mierzona jako różnica pomiędzy prawą a lewą stroną równania z nowym rozwiązaniem. Proces iteracji zatrzymywany jest po osiągnięciu założonej dokładności lub po zadanej liczbie iteracji.

W obecnej pracy do rozwiązania równania Poissona wykorzystana została metoda wielosiatkowa. Wykorzystuje ona właściwość grupy metody do rozwiązywania układów równań liniowych ogólnie nazywanych metodami wygładzającymi (*ang.* smoothers). Należą do niej takie metody jak metoda Jacobiego czy metoda Gaussa-Seidla. Charakteryzują się one tym, że szybko (w kilku iteracjach) wygaszane są składowe błędu o wysokiej częstotliwości. Niwelacja wolnozmiennych składowych zabiera dużo więcej iteracji. Rozrzedzenie siatki sprawia, że pewne wysokie częstotliwości staną się nieosiągalne, a cześć do tej pory niskich częstotliwości stanie się, relatywnie, wysokimi. Wykonując obliczenia na grupie coraz rzadszych siatek możliwe jest dużo efektywniejsza redukcja wszystkich częstotliwości składowych błędu. Metoda ta doczekała się bogatej literatury [103, 82, 97, 13, 98, 12]. Szczególny nacisk jest kładziony na dobór interpolacji przy przechodzeniu pomiędzy siatkami [56], jak i schematy zapewniające wyższy rząd [80, 31, 101], czy pozwalające stosować nierównomierną siatkę [23]. Wysoka wydajność metody wielosiatkowej spowodowała, że powstała jej wersja do wykorzystania z dowolnymi macierzami. W metodzie tej, nazywanej algebraiczną metodą wielosiatkową (*ang.* Algebraic Multigrid method), do niższych poziomów wybierane są pewne wiersze macierzy na podstawie liczby i wielkości występujących w nich współczynników [13, 98, 32, 9].

Wykorzystanie prostych metod iteracyjnych, takich jak metoda Jacobiego czy Gaussa-Seidla oraz wysoka wydajność sprawiły, że metoda wielosiatkowa stała się dobrym kandydatem do wykorzystania w obliczeniach równoległych na kartach graficznych. Obliczenia w metodzie Jacobiego są niezależne dla każdego węzła siatki i w prosty sposób można je zrównoleglić. Trudniejsze jest wykorzystanie metody Gaussa-Seidla. Tutaj wprowadza się tzw. kolorowanie węzłów [98, 32] i wykonuje obliczenia w kilku podkrokach. Dla siatek strukturalnych powstaje wtedy tzw. czerwono-czarna metoda Gaussa-Seidla (*ang.* Red-Black Gauss-Seidel

method). W ciągu ostatnich lat powstało i zostało opisanych w literaturze wiele implementacji metody wielosiatkowej zarówno na jednej [28, 10], jak i wielu kartach graficznych [27] oraz w wersji AMG [9]. Wykorzystywana w obecnej pracy implementacja cechuje się tym, że jest stworzona wyłącznie do rozwiązywania równania Poissona. Dzięki wykorzystaniu informacji o siatce zamiast tworzenia pełnej macierzy układu równań obliczenia prowadzone są bardziej efektywnie.

Metoda gradientów sprzężonych wykorzystuje fakt, że rozwiązanie układu równań algebraicznych liniowych dla dodatnio określonej macierzy jest jednocześnie minimum formy kwadratowej tego równania [92]. Metoda ta z wieloma różnymi warunkowaniami wstępnymi (*ang.* preconditioner) jest z powodzeniem wykorzystywana w różnych zastosowaniach i również doczekała się wielu implementacji na karcie graficznej [10].

Minusem prowadzenia obliczeń na karcie graficznej jest mała ilość dostępnej pamięci RAM. Przykładowo dla wykorzystywanej w pracy karty NVIDIA GTX 480 było to 1.5GB. Pozwalało to przeprowadzić symulacje na siatce numerycznej o rozmiarze $128 \times 128 \times 128$ węzłów. Jest to wielkość wystarczająca do badania wielu zagadnień inżynierskich, niestety jest to zbyt mała liczba do obliczeń niektórych interesujących zjawisk. Wyjściem z tej sytuacji było stworzenie implementacji metody cząstek wirowych wykorzystującej do obliczeń wiele kart graficznych. Do komunikacji pomiędzy kartami została wykorzystana biblioteka MPI oraz OpenMP. Pozwoliło to na przeniesienie obliczeń na klastry obliczeniowe i wykorzystanie dowolnej dostępnej w systemie liczby kart graficznych niezależnie od konfiguracji sprzętowej. Dzięki temu możliwe były symulacje numeryczne w wykorzystaniem $256 \times 256 \times 256$ (klaster zakładowy) i $512 \times 512 \times 512$ (klaster PL-GRID) węzłów. Są to już zadowalające gęstości siatki numerycznej.

Dla implementacji metody cząstek wirowych na pojedynczej karcie graficznej (NVIDIA GeForce GTX 480) otrzymano niemal 50 krotne przyspieszenie czasu obliczeń względem pojedynczego rdzenia procesora (Intel i7 960). Dla metody wykorzystującej wiele kart graficznych do obliczeń otrzymano ok. 90% efektywność skalowania słabego przy wykorzystaniu 8 GPU. Szczegóły tej implementacji opisane są w dalszej części niniejszej pracy.

Rozdział 2

Cel, tezy i zakres pracy

Cel pracy

1. Celem pracy jest opracowanie i implementacja metody cząstek wirowych typu "Wir w komórce" do rozwiązywania nieściśliwych, trójwymiarowych przepływów płynów lepkich wykorzystującej obliczenia równoległe, pozwalającej na badanie ewolucji oraz wzajemnych oddziaływań struktur wirowych.

2. Celem pracy jest budowa środowiska obliczeniowego opartego o ogólnie dostępne karty graficzne.

Tezy pracy

- Metoda cząstek wirowych umożliwia modelowanie ewolucji wirowości pozwala na symulację rzeczywistych zjawisk hydrodynamicznych i bardzo dobrze nadaje się do zastosowania w obliczeniach równoległych.

- Możliwe jest zbudowanie środowiska do obliczeń równoległych z ogólnodostępnych kart graficznych pozwalające znacząco przyspieszyć obliczenia.

Zakres pracy

Zakres pracy obejmował:

- Sformułowanie równań ruchu płynu w ujęciu wirowości i prędkości dla nielepkiego i lepkiego przepływu trójwymiarowego.

- Przygotowanie i przetestowanie algorytmu numerycznego metody wirowej typu "Wir w komórce" w oparciu o sformułowane równania na pojedynczym procesorze (CPU) a następnie na kartach graficznych.
- Sprawdzenie poprawności metody poprzez odtworzenie dynamiki pojedynczego pierścienia oraz porównanie jego prędkości translacji z formułą analityczną.
- Opracowanie implementacji metody "Wir w komórce" wykorzystującej do obliczeń wiele kart graficznych. Zastosowanie hybrydowego MPI-OpenMP do komunikacji pomiędzy kartami graficznymi. Napisanie i przetestowanie programów rozwiązujących równanie Poissona metodą wielosiatkową oraz równanie dyfuzji metodą gradientów sprzężonych. Wyznaczenie otrzymanego przyspieszenia oraz skalowania.
- Sprawdzenie poprawności implementacji.
- Odtworzenie zjawisk rekonekcji struktur wirowych, tzw. "Gry wirów" oraz zjawiska czołowego zderzenia pierścieni wirowych w przepływie lepkim w różnych konfiguracjach - porównanie wyników z literaturą. Badania wymienionych zjawisk na gęstszych siatkach. Prowadzenie obliczeń na superkomputerach zgromadzonych w programie PL-Grid.

Rozdział 3

Metoda cząstek wirowych

3.1. Sformułowanie równań Naviera-Stokesa w ujęciu wirowości

Niech $\mathbf{u}(\mathbf{x},t)$ będzie polem prędkości lepkiego, nieściśliwego płynu a $p(\mathbf{x},t)$ jego polem ciśnienia. Równania ruchu płynu nieściśliwego w trójwymiarowej przestrzeni $\mathcal{D}$ opisuje układ równań wyrażający prawa zachowania pędu i masy [6, 17, 84]:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\Delta\mathbf{u} + \mathbf{f}, \tag{3.1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{3.2}$$

gdzie ρ jest gęstością płynu (przyjęto, że $\rho = \text{const.}$) a ν – kinematycznym współczynnikiem lepkości.

W dalszych rozważaniach założono, że siły masowe są równe zero ($\mathbf{f}(\mathbf{x},t) = 0$). Równania ruchu należy uzupełnić o warunek początkowy $\mathbf{u}|_{t=0} = \mathbf{u}_0(\mathbf{x})$ oraz warunek brzegowy $\mathbf{u}|_{\mathcal{S}} = \mathbf{u}(\mathbf{x}_{\mathcal{S}},t)$, gdzie $\mathcal{S}$ - powierzchnia ograniczająca obszar $\mathcal{D}$. Po zastosowaniu podanych uproszczeń równania (3.1) i (3.2) można zapisać w postaci:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\Delta\mathbf{u}, \tag{3.3}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{3.4}$$

$$\mathbf{u}|_{t=0} = \mathbf{u}_0(\mathbf{x}), \quad \mathbf{u}|_{\mathcal{S}} = \mathbf{u}(\mathbf{x}_{\mathcal{S}},t). \tag{3.5}$$

Dla przypadku przepływu płynu nielepkiego ($\nu = 0$) podane równania upraszczają się dalej do równań Eulera:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p, \tag{3.6}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{3.7}$$

Dalej zostanie wykorzystana następująca tożsamość wektorowa:

$$\mathbf{v} \times (\nabla \times \mathbf{v}) + (\mathbf{v} \cdot \nabla)\,\mathbf{v} = \nabla\left(\frac{1}{2}\mathbf{v}^2\right), \tag{3.8}$$

prawdziwa dla każdego wektora $\mathbf{v}$. Wstawiając ją do równania (3.3) oraz definiując wektor wirowości jako

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}, \tag{3.9}$$

otrzymamy:

$$\frac{\partial \mathbf{u}}{\partial t} + \boldsymbol{\omega} \times \mathbf{u} = -\nabla\left(\frac{1}{\rho}p + \frac{\mathbf{u}^2}{2}\right) + \nu\Delta\mathbf{u}. \tag{3.10}$$

Następnie biorąc rotację równania (3.10) otrzymujemy równanie ewolucji wirowości w postaci

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \times (\boldsymbol{\omega} \times \mathbf{u}) = \nu\Delta\boldsymbol{\omega}. \tag{3.11}$$

Wykorzystując zależność (prawdziwą dla dowolnych wektorów $\mathbf{b}$ i $\mathbf{c}$)

$$\nabla \times (\mathbf{b} \times \mathbf{c}) = (\nabla \cdot \mathbf{c})\mathbf{b} + (\mathbf{c} \cdot \nabla)\mathbf{b} - (\nabla \cdot \mathbf{b})\mathbf{c} - (\mathbf{b} \cdot \nabla)\mathbf{c}, \tag{3.12}$$

oraz, że $\nabla \cdot \boldsymbol{\omega} = 0$ a płyn jest nieściśliwy $\nabla \cdot \mathbf{u} = 0$ możemy zapisać równanie (3.11) w postaci

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu\Delta\boldsymbol{\omega}. \tag{3.13}$$

Dla płynu nielepkiego równanie to będzie miało postać

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u}, \tag{3.14}$$

gdzie $D(\cdot)/Dt = \partial(\cdot)/\partial t + (\mathbf{u} \cdot \nabla)(\cdot)$ oznacza pochodną substancjalną.

3.2. Twierdzenie Helmholtza

Twierdzenie 3.1. *Wirowe twierdzenia Helmholtza*

Dla ruchu wirowego płynu nielepkiego i nieściśliwego danego równaniem (3.14) można udowodnić następujące twierdzenia oryginalnie podane przez Helmholtza [71]:

(I) Intensywność rurki wirowej jest stała wzdłuż rurki.

(II) Linie wirowe unoszone są tak jak linie materialne.

(III) Intensywność rurki wirowej nie zmienia się w czasie.

W metodach cząstek wirowych wykorzystywany jest fakt, że dla płynu nielepkiego i nieściśliwego linie wirowe unoszone są tak, jak linie materialne. Jest to treścią drugiego prawa Helmholtza (twierdzenie 3.1). Linią materialną będziemy nazywać linię łączącą dwie cząstki płynu znajdujące się dostatecznie blisko siebie. Ewolucja elementu $d\mathbf{l} = (dl_1, dl_2, dl_3)$ tej linii określona jest równaniem

$$\frac{Dd\mathbf{l}}{Dt} = (d\mathbf{l} \cdot \nabla)\mathbf{u}. \tag{3.15}$$

Z równań (3.14) i (3.15) widać, że jeżeli wybrany element płynu zawierał na początku fragment linii wirowej (linia styczna w każdym swoim punkcie do pola wirowości) to niezależnie od jego przemieszczenia i deformacji w trakcie przepływu fragment linii wirowej będzie zawsze składał się z tych samych cząstek płynu. Zatem cząstki wirowe będą przemieszczały się tak jak cząstki materialne [105], zgodnie z chwilowym polem prędkości. Ruch cząstek wirowych można zatem opisać poprzez nieskończony układ równań różniczkowych:

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p, t), \qquad \mathbf{x}(0, \mathbf{a}) = \mathbf{a}, \tag{3.16}$$

gdzie $\mathbf{a} = (a_1, a_2, a_3)$ oznacza współrzędne cząstki materialnej nazywane zmiennymi Lagrange'a. Rozwiązanie układu równań (3.16) określa odwzorowania $\Phi(\cdot, t) : \mathbb{R}^3 \to \mathbb{R}^3$; $\mathbf{q} \to \Phi(\mathbf{a}, t) = \boldsymbol{x} \in \mathbb{R}^3$, które jest wzajemnie jednoznaczne. Z założenia nieściśliwości ruchu płynu wynika, że $det(\nabla_a \mathbf{\Phi}(\mathbf{a}, t)) = 1$.

3.3. Cząstki wirowe

Dla przypadków dwuwymiarowych prawa strona równania (3.14) jest równa zero (brak rozciągania linii wirowych) i równanie ruchu przekształca się do prostej postaci $\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = 0$, gdzie $\frac{\mathrm{D}(\cdot)}{\mathrm{D}t}$ oznacza pochodną substancjalną. Wynika z niego, że wirowość unoszona przez płyn nie ulega zmianie.

Ciągłe pole wirowości zastępowane jest dyskretnym rozkładem cząstek wirowych którym przypisane są porcje wirowości, które będą nazywane intensywnością cząstki. Dla trzech wymiarów wirowość posiada trzy składowe. Wprowadzona do obliczeń cząstka jest wektorem o trzech składowych a więc, są one określone przez wektor położenia $\mathbf{x}_p$ oraz wektor intensywności $\boldsymbol{\alpha}_p$. Inicjalizacja cząstek może odbywać się w różny sposób. W metodzie typu „Wir w komórce" wykorzystywana jest w tym celu pomocnicza siatka numeryczna o stałym kroku h we wszystkich kierunkach. Zdyskretyzowane pole wirowości wyraża się jako suma wirowości cząstek wirowych w następujący sposób [17, 64]

$$\boldsymbol{\omega}(\mathbf{x}, t) \approx \boldsymbol{\omega}^h(\mathbf{x}, t) = \sum_{i=1}^{N} \boldsymbol{\alpha}_p(\mathbf{x}, t)\delta(\mathbf{x} - \mathbf{x}_p(t)), \tag{3.17}$$

gdzie $\boldsymbol{\alpha}_p$ oznacza cząstkę wektorową $\boldsymbol{\alpha}_p = (\alpha_{p(1)}, \alpha_{p(2)}, \alpha_{p(3)})$ w położeniu $\mathbf{x}_p = (x_{p(1)}, x_{p(2)}, x_{p(3)})$. Intensywność cząstki można wyznaczyć z pola wirowości w następujący sposób:

$$\alpha_{p(i)}(\mathbf{x}_p) = \int_{V_p} \omega_i(x_1, x_2, x_3)\, \mathrm{d}\mathbf{x} \approx h^3 \omega_i(\mathbf{x}_p), \tag{3.18}$$

gdzie V_p jest objętością komórki z indeksem p.

3.4. Wyznaczanie pola prędkości

Mając na uwadze fakt, że $\nabla \cdot \mathbf{u} = 0$ oraz, że dywergencja rotacji z dowolnego pola wektorowego jest równa 0 $\nabla \cdot (\nabla \times \mathbf{A}) \equiv 0$, można założyć, że [33, 34]:

$$\mathbf{u} = \nabla \times \mathbf{A}. \tag{3.19}$$

Wykorzystując definicję wektora wirowości i podstawiając do niej równanie (3.19) otrzymujemy:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} = \nabla \times \nabla \times \mathbf{A} = \nabla(\nabla \cdot \mathbf{A}) - \Delta \mathbf{A}, \tag{3.20}$$

gdzie wektor $\mathbf{A}$ nazywany jest potencjałem wektorowym.

Niżej przytoczone zostanie twierdzenie gwarantujące istnienia potencjału wektorowego $\mathbf{A}$ w ograniczonym obszarze [26].

Twierdzenie 3.2. *Niech obszar przepływu będzie oznaczony przez V i niech będzie regularny lub nieskończony. Niech S będzie brzegiem V $\left(\int_S = \sum_{i=1}^{N} \oint_{S_i}\right)$. Średni strumień przez każdy z brzegów musi być równy zeru $\left(\oint_{S_i} \mathbf{n} \cdot \mathbf{u}\, da = 0, i = 1, ..., N\right)$.*

Założenia

- *Niech $y^3 = f(y^1, y^2)$ będzie standardową reprezentacją regularnego elementu powierzchni S. Zdefiniujmy "ściankę" jako region S taki, że trzecia pochodna $f(y_1, y_2)$ spełnia warunek Höldera.*
- *S musi się składać ze skończonej liczby "ścianek", a każda z nich musi być otoczona przez skończoną liczbę regularnych łuków.*
- *Niech prędkość $\mathbf{u}$ ma ciągłe pochodne drugiego rzędu w V, ciągłe jednostronne pochodne normalne do S na S i ciągłe pierwsze pochodne w stosunku do współrzędnych S.*
- *Przepływ jest nieściśliwy $\nabla \cdot \mathbf{u} = 0$. Jeżeli obszar jest nieskończony, $\mathbf{u}$ musi być regularne w nieskończoności.*

Jeżeli wektor $\mathbf{u}$ *spełnia powyższe założenia w obszarze* V *to istnieje potencjał wektorowy* $\mathbf{A}$ *taki, że:*

$$\mathbf{u} = \nabla \times \mathbf{A} \qquad i \qquad \nabla \cdot \mathbf{A} = 0.$$

Dowód twierdzenia 3.2 można znaleźć w pracy [26]

Stosując twierdzenie 3.2 do równania (3.20) otrzymujemy

$$\boldsymbol{\omega} = -\Delta \mathbf{A}. \tag{3.21}$$

Rozpatrzmy nieściśliwe pole prędkości $\mathbf{u}(\mathbf{x})$ określone przez zwarte (ograniczone i ciągłe) pole wirowości $\boldsymbol{\omega}(\mathbf{x})$, dla którego wirowość znika poza ograniczonym obszarem $\mathcal{D}$:

$$\boldsymbol{\omega}(\mathbf{x}) \begin{cases} \neq 0 & \text{dla } \mathbf{x} \in \mathcal{D}, \\ = 0 & \text{dla } \mathbf{x} \notin \mathcal{D}. \end{cases} \tag{3.22}$$

W obszarze nieograniczonym rozwiązanie równania (3.21) można przedstawić w postaci:

$$\mathbf{A}(\mathbf{x}, t) = \frac{1}{4\pi} \int_D \frac{\boldsymbol{\omega}(\mathbf{y}, t)}{|\mathbf{x} - \mathbf{y}|} \, \mathrm{d}^3\mathbf{y}, \tag{3.23}$$

gdzie $\mathbf{y}$ oznacza wektor wodzący dla całkowania (przy ustalonym t) po obszarze $\mathcal{D}$.

Prędkość można zatem wyznaczyć biorąc rotację potencjału wektorowego:

$$\mathbf{u}(\mathbf{x}, t) = \nabla \times \mathbf{A} = \frac{1}{4\pi} \nabla \times \int_D \frac{\boldsymbol{\omega}(\mathbf{y}, t)}{|\mathbf{x} - \mathbf{y}|} \, \mathrm{d}^3\mathbf{y} \, . \tag{3.24}$$

Wykorzystując tożsamość

$$\nabla \times \left(\frac{\mathbf{a}}{|\mathbf{x}|} \right) = \mathbf{a} \times \left(\frac{\mathbf{x}}{|\mathbf{x}|^3} \right), \tag{3.25}$$

gdzie $\mathbf{a}$ jest stałym wektorem, otrzymujemy

$$\mathbf{u}(\mathbf{x}, t) = -\frac{1}{4\pi} \int_D \frac{(\mathbf{x} - \mathbf{y}) \times \boldsymbol{\omega}(\mathbf{y}, t)}{|\mathbf{x} - \mathbf{y}|^3} \, \mathrm{d}^3\mathbf{y} \, . \tag{3.26}$$

Wzór ten pozwala na wyznaczenie pola prędkości $\mathbf{u}$ kiedy znane jest pole wirowości $\boldsymbol{\omega}$. Zależność ta nazywana jest *prawem Biota-Savarta* i pierwotnie była podana dla wyznaczenie pola magnetycznego na podstawie znajomości stałego prądu elektrycznego.

Dla $|\mathbf{x}| \to \infty$ oraz $|\mathbf{x} - \mathbf{y}| \sim |\mathbf{x}|$ prędkość dąży do zera, $\mathbf{u} \to 0$, co można pokazać następująco [38]:

$$\mathbf{u}(\mathbf{x}) = -\frac{1}{4\pi|\mathbf{x}|^3} \mathbf{x} \times \left(\int \boldsymbol{\omega}(\mathbf{y}) \, \mathrm{d}^3\mathbf{y} + \mathcal{O}(|\mathbf{x}|^{-1}) \right) = O(|\mathbf{x}|^{-3}). \tag{3.27}$$

Prawo Biota-Savarta może posłużyć do wyznaczenia prędkości indukowanej przez nić wirową. Nicią wirową będziemy nazywali pewną idealizację rurki wirowej, której przekrój poprzeczny dąży do zera i jednocześnie strumień wirowości dąży do nieskończoności tak, aby w granicy otrzymać skończoną wartość cyrkulacji Γ.

Pole prędkości indukowane przez nić wirową otrzymujemy wprost ze wzoru (3.26) stosując podstawienie:

$$\boldsymbol{\omega}\,\mathrm{d}\upsilon = \Gamma\,\mathrm{d}\mathbf{s}, \tag{3.28}$$

gdzie $\mathrm{d}\mathbf{s}$ jest elementem długości łuku nici wirowej. Po podstawieniu otrzymamy wzór:

$$\mathbf{u}(\mathbf{x},t) = -\frac{\Gamma}{4\pi}\int_{\mathcal{C}} \frac{(\mathbf{x}-\mathbf{y})\times \mathrm{d}\mathbf{s}}{|\mathbf{x}-\mathbf{y}|^3}. \tag{3.29}$$

Wzór (3.29) jest podstawowym wzorem używanym do obliczania pola prędkości w bezpośrednich metodach wirowych [17].

Rozwiązanie równania (3.21) w obszarze nieograniczonym przyjmuje postać:

$$\mathbf{A}(\mathbf{x},t) = -\int_{\mathcal{D}} G(\mathbf{x}-\mathbf{x}')\boldsymbol{\omega}(\mathbf{x}',t)\,\mathrm{d}\upsilon. \tag{3.30}$$

Funkcja Greena dla równania Poissona (przy założeniach, że w nieskończoności $\boldsymbol{\omega}\to 0$ i $\mathbf{u}\to 0$) ma postać

$$G(\mathbf{r}) = \begin{cases} \frac{1}{2\pi}\ln(r) & \text{dla dwóch wymiarów,} \\ -\frac{1}{4\pi r} & \text{dla trzech wymiarów,} \end{cases} \tag{3.31}$$

gdzie $r = |\mathbf{r}| = |\mathbf{x}-\mathbf{y}|$.

Równania (3.30) są podstawowymi formułami dla bezpośrednich metod cząstek wirowych [70]. W obliczeniach tą metodą należy zastosować regularyzację usuwając osobliwość jądra G [83, 73, 70]. Wadą bezpośrednich metod wirowych jest to, że są one bardzo kosztowne obliczeniowo. Ze względu na globalny charakter oddziaływań cząstek wirowych (każda cząstka ma wpływ na prędkość indukowaną na wszystkich pozostałych cząstkach) obliczenie prędkości w jednym punkcie przestrzeni wymaga $\mathcal{O}(N)$ operacji (gdzie N oznacza liczbę cząstek). Powoduje to, że złożoność obliczeniowa tego zagadnienia wynosi $\mathcal{O}(N^2)$. Ta sytuacja jest analogiczna do problemu obliczania sił grawitacyjnych oddziałujących na N mas, który znany jest pod nazwą *„N-body problem”*.

Problem ten znalazł dwa powszechnie stosowane rozwiązania. Pierwszym jest wykorzystanie algorytmów znanych z astrofizyki takich jak np. Szybkie Metody Wielopolowe (*ang.* FMM - Fast Multipole Method) [30]. Metody te pozwoliły na obniżenie złożoności obliczeniowej zagadnienia wyznaczania prędkości do $\mathcal{O}(N\log N)$, a nawet $\mathcal{O}(N)$. Drugim sposobem jest zastosowanie metody siatkowo – cząstkowej tzn. wprowadzenie pomocniczej siatki numerycznej i rozwiązywaniem na niej równania Poissona. Tak jest to robione m.in. w metodzie „Wir

w komórce". Szybkie bezpośrednie algorytmy rozwiązywania równania Poissona można wykorzystać dla obszarów prostokątnych i równań dla których można zastosować metodę rozdzielonych zmiennych. Koszt obliczeń jest proporcjonalny do $\mathcal{O}(N\log(N))$, gdzie N oznacza liczbę oczek siatki numerycznej. Takie podejście pozwala nawet na $\sim$ 1000-krotne przyspieszenie obliczeń (przy liczbie cząstek $n \sim 10^5$) względem metod bezpośrednich [18].

3.5. Niezmienniki ruchu

Dla nieściśliwego i nielepkiego płynu, którego równania ruchu mają postać (3.14), (3.7) możemy zdefiniować wielkości, które będą miały stałą wartość przez cały czas trwania ruchu. Wielkości te nazywane są niezmiennikami ruchu [38]. Są one istotne przy testowaniu programu obliczeniowego.

3.5.1. Całkowita wirowość

Wartość całkowitej wirowości w przepływie trójwymiarowym w obszarze $\mathcal{D}$ można przedstawić następująco

$$\int_{\mathcal{D}} \boldsymbol{\omega}\,\mathrm{d}v. \tag{3.32}$$

Wzór ten można przekształcić do postaci

$$\int_{\mathcal{D}} \omega_i\,\mathrm{d}v = \int_{\mathcal{D}} \nabla\cdot(x_i\boldsymbol{\omega})\,\mathrm{d}v = \int_{\mathcal{S}} x_i\boldsymbol{\omega}\cdot\mathbf{n}\,\mathrm{d}s. \tag{3.33}$$

gdzie $\mathcal{S}$ jest brzegiem obszaru $\mathcal{D}$ z wektorem normalnym $\mathbf{n}$. Wartość tej całki jest zero jeżeli na całym brzegu $\boldsymbol{\omega}\cdot\mathbf{n} = 0$. Przy spełnieniu tego warunku wartość całkowitej wirowości pozostaje niezmienna

$$\int_{\mathcal{D}} \omega_i\,\mathrm{d}v = 0. \tag{3.34}$$

3.5.2. Całkowita cyrkulacja

Dla przepływów dwuwymiarowych cyrkulacja wyraża się wzorem

$$\Gamma = \int_{\mathcal{S}} \omega\,\mathrm{d}s. \tag{3.35}$$

Zmianę całkowitej cyrkulacji w czasie wyraża zależność:

$$\frac{d\Gamma}{dt} = \int_{\mathcal{D}} \frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t}\,\mathrm{d}v = \int_{\mathcal{D}} (\boldsymbol{\omega}\cdot\nabla\mathbf{u})\,\mathrm{d}v. \tag{3.36}$$

Stała wartość Γ dla przepływów dwuwymiarowych ($\boldsymbol{\omega} = [0, 0, \omega]$) wynika wprost z równania transportu wirowości (3.14). W tym wypadku człon $(\boldsymbol{\omega} \cdot \nabla)\mathbf{u}$ jest zawsze równa zeru.

Biorąc pochodną cyrkulacji po brzegu $\mathcal{S}$ i wykorzystując przekształcenie [71]

$$d\mathbf{x} = \mathbf{F} \cdot d\xi \qquad \text{lub} \qquad dx_i = F_{iA} d\xi_A, \tag{3.37}$$

można pokazać, że

$$\begin{aligned} \frac{d\Gamma}{dt} &= \frac{d}{dt} \int_{\mathrm{S}} u_i dx_i \\ &= \frac{d}{dt} \int_{\mathrm{S}_0} u_i F_{iA} d\xi_A \\ &= \int_{\mathrm{S}_0} \frac{\mathrm{D}}{\mathrm{D}t} (u_i F_{iA}) d\xi_A \\ &= \int_{\mathrm{S}_0} \left(\frac{\mathrm{D}u_i}{\mathrm{D}t} F_{iA} + u_i \frac{\partial u_i}{\partial x_j} F_{jA} \right) d\xi_A, \end{aligned} \tag{3.38}$$

gdzie wykorzystane zostało równanie $\dot{F}_{iA} = \frac{\partial u_i}{\partial x_j} \frac{\partial x_j}{\partial d\xi_A} = \frac{\partial u_i}{\partial x_j} F_{jA}$. Zamieniając w ostatnim członie indeksy i i j możemy ponownie wykorzystać przekształcenie $dx_i = F_{iA} d\xi_A$ i napisać

$$\frac{d\Gamma}{dt} = \int_{\mathrm{S}} \left(\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} + \nabla \frac{\mathbf{u}^2}{2} \right) \cdot d\mathbf{x}. \tag{3.39}$$

Wstawiając równanie pędu (3.6) otrzymujemy

$$\frac{d\Gamma}{dt} = \int_{\mathrm{S}} \left(\nabla \frac{\mathbf{u}^2}{2} - \frac{1}{\rho} \nabla p \right) \cdot d\mathbf{x}. \tag{3.40}$$

Można to dalej przekształcić do

$$\frac{d\Gamma}{dt} = \int_{\mathrm{S}} \nabla \left(\frac{\mathbf{u}^2}{2} - \frac{p}{\rho} \right) \cdot d\mathbf{x}. \tag{3.41}$$

Jeżeli $\mathcal{D}$ jest otwartą przestrzenią z granicą $\mathcal{S}$ w chili t i wektorem normalnym $\mathbf{n}$ to możemy wykorzystać twierdzenie Stokesa i zapisać równanie numer w postaci

$$\frac{d\Gamma}{dt} = \int_{\mathcal{D}} \left[\nabla \times \nabla \left(\frac{\mathbf{u}^2}{2} - \frac{p}{\rho} \right) \right] \cdot \mathbf{n} da. \tag{3.42}$$

Wykorzystując tożsamość $\nabla \times \nabla \phi = 0$ prawdziwą dla każdego pola ϕ można pokazać, że $\frac{d\Gamma}{dt} = 0$.

Dla przepływów lepkich całkowita cyrkulacja pozostaje niezmiennikiem tylko w przypadku braku w przepływie obszarów, w których zachodzi proces generowania wirowości np. ścianek sztywnych.

3.5.3. Enstropia

Enstropię E pola wirowości w trzech wymiarach definiuje się jako:

$$E = \frac{1}{2}\int_{\mathcal{D}} \boldsymbol{\omega}\cdot\boldsymbol{\omega}\,\mathrm{d}\upsilon. \tag{3.43}$$

Zmianę enstropii w czasie wyraża zależność:

$$\frac{dE}{dt} = \int_{\mathcal{D}} \boldsymbol{\omega}\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t}\,\mathrm{d}\upsilon = \int_{\mathcal{D}} \boldsymbol{\omega}\cdot((\boldsymbol{\omega}\cdot\nabla)\mathbf{u})\,\mathrm{d}\upsilon. \tag{3.44}$$

Dla dwuwymiarowego przepływu nielepkiego $\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = 0$ a co za tym idzie wartość enstropii jest stała w czasie. W trójwymiarowych przepływach nielepkich natomiast, człon związany z rozciąganiem linii wirowych $((\boldsymbol{\omega}\cdot\nabla)\mathbf{u})$ może powodować wzrost wartości enstropii w przepływie.

3.5.4. Energia kinetyczna

Całkowita energia wyraża się wzorem

$$K = \frac{1}{2}\int_{\mathcal{D}} \mathbf{u}\cdot\mathbf{u}\,\mathrm{d}\upsilon. \tag{3.45}$$

Dla płynu nielepkiego i nieściśliwego ($\nabla\cdot\mathbf{u} = 0$, $\rho = \text{const.}$), którego pole wirowości jest ograniczone (3.22) można wykazać, że $K = \text{const}$. Należy wyznaczyć pochodną równania (3.45) i wstawić do niego równanie pędu (3.6) [71]

$$\frac{dK}{dt} = \int_{\mathcal{D}} \mathbf{u}\cdot\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t}\,\mathrm{d}\upsilon = \int_{\mathcal{D}} \mathbf{u}\cdot\left(-\frac{1}{\rho}\nabla p\right)\mathrm{d}\upsilon. \tag{3.46}$$

Wykorzystując fakt nieściśliwości pola prędkości można napisać

$$\frac{dK}{dt} = \int_{\mathcal{D}} \nabla\cdot\left[\left(-\frac{p}{\rho}\right)\mathbf{u}\right]\mathrm{d}\upsilon = \int_{\mathcal{S}} \left(-\frac{p}{\rho}\right)(\mathbf{u}\cdot\mathbf{n})\,\mathrm{d}s, \tag{3.47}$$

gdzie $\mathbf{n}$ jest wektorem normalnym brzegu $\mathcal{S}$. Wynik ten wskazuje, że energia kinetyczna jest niezmiennikiem przepływu jeżeli $\mathbf{u}\cdot\mathbf{n} = 0$ na wszystkich brzegach S lub w przepływach zewnętrznych w których prędkość dla przepływów trójwymiarowych maleje szybciej niż r^{-2} (3.27), a dwuwymiarowych szybciej niż r^{-1}.

Wzór na energię kinetyczną (3.45) można przekształcić do postaci zawierającej wirowość i potencjał wektorowy. Wykorzystując regułę łańcuchową oraz tożsamość $\frac{\partial x_i}{\partial x_j} = \delta_{ij}$ można pokazać, że [71]:

$$\frac{\partial}{\partial x_j}(x_i u_i u_j) = u_i u_i + x_i u_j \frac{\partial u_i}{\partial x_j}, \tag{3.48}$$

$$\frac{1}{2}\frac{\partial}{\partial x_i}(x_i u_j u_j) = \frac{3}{2}u_j u_j + x_i u_j \frac{\partial u_j}{\partial x_i}. \tag{3.49}$$

Odejmując (3.48) od (3.49) otrzymujemy

$$\frac{1}{2}u_j u_j = -x_i u_j \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) + \frac{1}{2}\frac{\partial}{\partial x_i}(x_i u_j u_j) - \frac{\partial}{\partial x_j}(x_i u_j u_i). \tag{3.50}$$

Całkując (3.50) po obszarze $\mathcal{D}$ dwa ostanie człony mogą zostać zamienione na całki po brzegu $\mathcal{S}$, które będę znikały gdy $\mathcal{S} \to \infty$ dla obszaru nieograniczonego. Pierwszy człon równania (3.50) może zostać wyrażony poprzez tensor wirowości W_{ij} jako

$$x_i u_j \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) = x_i u_j W_{ij}. \tag{3.51}$$

Wykorzystując równanie $W_{ij} = \frac{1}{2}\varepsilon_{ijk}\omega_k$ łączące tensor wirowości i wektor wirowości możemy równanie (3.51) zapisać w postaci

$$x_i u_j \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) = \varepsilon_{ijk} x_i \omega_k u_j. \tag{3.52}$$

Wstawiając uzyskane wartości do równania (3.45) otrzymujemy

$$K = \frac{1}{2}\int_{\mathcal{D}} \mathbf{u} \cdot (\mathbf{x} \times \boldsymbol{\omega})\, \mathrm{d}v. \tag{3.53}$$

Wzór ten można dalej przekształcić używając tożsamości wektorowej [71]

$$\nabla \cdot (\mathbf{b} \times \mathbf{c}) = \mathbf{c} \cdot (\nabla \times \mathbf{b}) - \mathbf{b} \cdot (\nabla \times \mathbf{c}), \tag{3.54}$$

oraz podstawiając $\mathbf{b} = \mathbf{u}$ oraz $\mathbf{c} = \mathbf{A}$. Otrzymamy wtedy

$$\nabla \cdot (\mathbf{u} \times \mathbf{A}) = \mathbf{A} \cdot \boldsymbol{\omega} - \mathbf{u} \cdot \mathbf{u}. \tag{3.55}$$

Całkując (3.55) po obszarze $\mathcal{D}$ możemy przekształcić lewą stronę do całki po brzegu obszaru $\mathcal{S}$, która znika gdy $\mathcal{S} \to \infty$. Wzór na energię kinetyczną można zatem przedstawić w postaci

$$K = \frac{1}{2}\int_{\mathcal{D}} \mathbf{A} \cdot \boldsymbol{\omega}\, \mathrm{d}v. \tag{3.56}$$

3.5.5. Impuls liniowy

Impuls liniowy wyraża się wzorem

$$\mathbf{P} = \frac{1}{2}\int_D \mathbf{x}\times\boldsymbol{\omega}\,\mathrm{d}\upsilon. \tag{3.57}$$

Aby pokazać, że $\mathbf{P}$ jest niezmiennikiem przepływu można obliczyć pochodną $\mathbf{P}$ po czasie [71]

$$\frac{d\mathbf{P}}{dt} = \frac{1}{2}\int_{\mathcal{D}} \mathbf{u}\times\boldsymbol{\omega} + \mathbf{x}\times\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t}\,\mathrm{d}\upsilon. \tag{3.58}$$

Wstawiając równanie transportu wirowości (3.14) do (3.58) możemy zapisać

$$\begin{aligned}\frac{dP_i}{dt} &= \frac{1}{2}\int_{\mathcal{D}}\left(\varepsilon_{ijk}u_j\omega_k + \varepsilon_{ijk}x_j\omega_l\frac{\partial u_k}{\partial x_l}\right)\mathrm{d}\upsilon\\ &= \frac{1}{2}\int_{\mathcal{D}}\left[2\varepsilon_{ijk}u_j\omega_k + \frac{\partial}{\partial x_l}(\varepsilon_{ijk}x_j\omega_l u_k)\right]\mathrm{d}\upsilon\\ &= \int_{\mathcal{D}}\varepsilon_{ijk}u_j\omega_k\,\mathrm{d}\upsilon + \frac{1}{2}\int_{\mathcal{S}} n_l\varepsilon_{ijk}x_j\omega_l u_k\,\mathrm{d}s,\end{aligned} \tag{3.59}$$

gdzie $\mathcal{S}$ jest brzegiem $\mathcal{D}$. Kiedy $\mathcal{S}$ dąży do nieskończoności to druga całka znika. Wstawiając definicję wirowości [numer] do całki objętościowej otrzymamy

$$\begin{aligned}\frac{dP_i}{dt} &= \int_{\mathcal{D}}\varepsilon_{ijk}\varepsilon_{klm}u_j\frac{\partial u_m}{\partial x_l}\,\mathrm{d}\upsilon\\ &= \int_{\mathcal{D}}(\delta_{il}\delta_{jm} - \delta_{im}\delta_{jl})u_j\frac{\partial u_m}{\partial x_l}\,\mathrm{d}\upsilon\\ &= \int_{\mathcal{D}}\left[\frac{\partial}{\partial x_i}\left(\frac{1}{2}u_ju_j\right) - \frac{\partial}{\partial x_j}(u_ju_i)\right]\mathrm{d}\upsilon,\end{aligned} \tag{3.60}$$

gdzie w drugim członie wykorzystany został fakt nieściśliwości płynu. Oba człony można przekształcić do całek powierzchniowych po brzegu obszaru, które będą znikały gdy będzie on dążył do nieskończoności.

3.6. Redystrybucja intensywności cząstek na węzły siatki

W metodach cząstek wirowych nośniki wirowości mają tendencję do gromadzenia się w obszarach, w których są wysokie gradienty prędkości. Może to prowadzić do niedokładności obliczeniowych spowodowanych zbytnim zbliżeniem się ich do siebie. Aby rozwiązać ten problem, w niektórych rozwiązaniach, wprowadza się redystrybucje cząstek na regularne pozycje (np. na węzły siatki obliczeniowej - *ang.* remeshing) przy jednoczesnym przeliczeniu wartości wektora intensywności

w nowych położeniach cząstek. W praktyce zabieg ten wykonywany jest co kilka lub kilkanaście kroków czasowych.

Wykorzystywana jest w tym celu interpolacja:

$$\omega_j = \sum_p \tilde{\Gamma}_{p_n} \varphi \left(\frac{\mathbf{x}_j - \tilde{\mathbf{x}}_p}{h} \right) \frac{1}{h^3}, \tag{3.61}$$

gdzie j jest indeksem węzła siatki, a p indeksem cząstki.

Jądro interpolacyjne powinno mieć następujące właściwości:

$$\sum_p \varphi \left(\frac{\mathbf{x} - \mathbf{x}_p}{h} \right) \equiv 1, \tag{3.62}$$

oraz odpowiednie własności momentów:

$$\sum_p \mathbf{x}^\alpha \varphi \left(\frac{\mathbf{x} - \mathbf{x}_p}{h} \right) = 0. \tag{3.63}$$

Do wyznaczenia różnicy pomiędzy starym (zaburzonym) a nowym (regularnym) rozkładem cząstek,

$$\sum_p \tilde{\Gamma}_{p_n} \delta(\mathbf{x} - \tilde{\mathbf{x}}_p) - \sum_p \Gamma_{p_n} \delta(\mathbf{x} - \mathbf{x}_p), \tag{3.64}$$

można przemnożyć powyższą funkcję przez funkcję testową $\phi \in C^\infty$ i otrzymujemy [17]:

$$E = \sum_p \tilde{\Gamma}_{p_n} \phi(\tilde{\mathbf{x}}_p) - \sum_p \Gamma_{p_n} \phi(\mathbf{x}_p), \tag{3.65}$$

gdzie $\tilde{\Gamma}_{p_n}$ i $\tilde{\mathbf{x}}_p$ są wartościami ze starego rozkładu.

Używając (3.61) można zapisać:

$$E = \sum_p \tilde{\Gamma}_{p_n} \left[\phi(\tilde{\mathbf{x}}_p) - \sum_p \phi(\mathbf{x}_j) \varphi \left(\frac{\mathbf{x}_j - \tilde{\mathbf{x}}_p}{h} \right) \right]. \tag{3.66}$$

Miarą błędu wynikającego z redystrybucji intensywności cząstki jest różnica

$$f(\mathbf{x}) = \phi(\mathbf{x}) - \sum_j \phi(\mathbf{x}_j) \varphi \left(\frac{\mathbf{x}_j - \mathbf{x}}{h} \right). \tag{3.67}$$

Mając na uwadze równanie (3.62) równanie (3.67) możemy zapisać jako

$$f(\mathbf{x}) = \sum_j [\phi(\mathbf{x}) - \phi(\mathbf{x}_j)] \varphi \left(\frac{\mathbf{x}_j - \mathbf{x}}{h} \right). \tag{3.68}$$

Rozwijając ϕ w szereg Taylora otrzymamy:

$$f(\mathbf{x}) = \sum_\alpha \sum_j [(\mathbf{x}_j - \mathbf{x}) \cdot \nabla \phi]^\alpha \varphi \left(\frac{\mathbf{x} - \mathbf{x}_j}{h} \right). \tag{3.69}$$

Na tej podstawie możemy stwierdzić, że jeżeli φ spełnia zależność:

$$E = \sum_q (\mathbf{x} - \mathbf{x}_q)^\alpha \varphi\left(\frac{\mathbf{x} - \mathbf{x_q}}{h}\right) \qquad 1 \leqslant |\alpha| \leqslant m-1, \tag{3.70}$$

to wtedy

$$f(\mathbf{x}) = O\left(h^m\right), \tag{3.71}$$

a procedura przemieszczenia cząstek na węzły siatki numerycznej będzie rzędu m.

3.6.1. Schematy interpolacyjne w procesie redystrybucji

Powszechne podejście do „remeshingu" (zwanego także redystrybucją) polega na inicjowaniu nowych cząstek wirowych na regularnych pozycjach (głównie na węzłach pomocniczej siatki numerycznej) oraz wyznaczaniu ich wektora intensywności poprzez interpolację starych wartości na nowe położenia cząstek, wykorzystując przy tym jądra interpolacyjne wysokich rzędów. Dwu- i trójwymiarowe schematy budowane są poprzez iloczyn jednowymiarowych jąder. Powszechnie stosowane jądra należą do dwóch rodzin nazywanych Λ i M. Ich wzory zostały podane poniżej

$$\Lambda_0(x) = \begin{cases} 1 & \text{dla } 0 \leqslant x \leqslant \frac{1}{2}, \\ 0 & \text{dla } x > \frac{1}{2}. \end{cases} \tag{3.72}$$

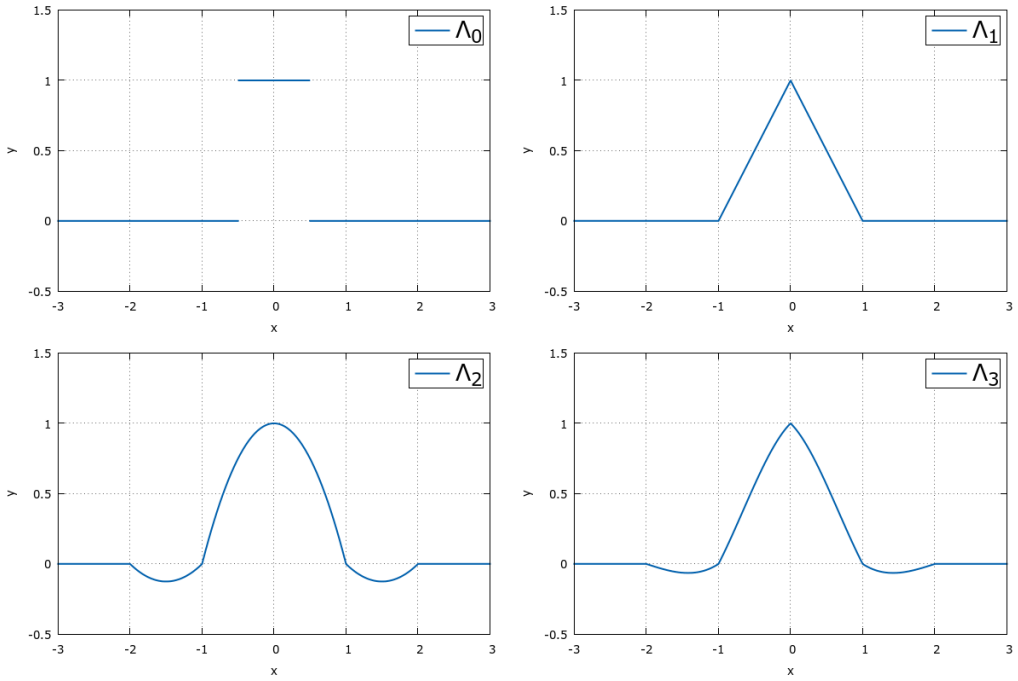
$$\Lambda_1(x) = \begin{cases} 1-x & \text{dla } 0 \leqslant x \leqslant 1, \\ 0 & \text{dla } x > 1. \end{cases} \tag{3.73}$$

$$\Lambda_2(x) = \begin{cases} 1-x^2 & \text{dla } 0 \leqslant x \leqslant \frac{1}{2}, \\ \frac{1}{2}(1-x)(2-x) & \text{dla } \frac{1}{2} \leqslant x \leqslant \frac{3}{2}, \\ 0 & \text{dla } x > \frac{3}{2}. \end{cases} \tag{3.74}$$

$$\Lambda_3(x) = \begin{cases} \frac{1}{2}(1-x^2)(2-x) & \text{dla } 0 \leqslant x \leqslant 1, \\ \frac{1}{6}(1-x)(2-x)(3-x) & \text{dla } 1 \leqslant x \leqslant 2, \\ 0 & \text{dla } x > 2. \end{cases} \tag{3.75}$$

$$M_1(x) \equiv \Lambda_0(x) = \begin{cases} 1 & \text{dla } 0 \leqslant x \leqslant \frac{1}{2}, \\ 0 & \text{dla } x > \frac{1}{2}. \end{cases} \tag{3.76}$$

$$M_2(x) \equiv \Lambda_1(x) = \begin{cases} 1-x & \text{dla } 0 \leqslant x \leqslant 1, \\ 0 & \text{dla } x > 1. \end{cases} \tag{3.77}$$



Rysunek 3.1: Rodzina jąder interopolacyjnych Λ.

$$M_3(x) = \begin{cases} \frac{1}{2}\left(\frac{3}{2} - x\right)^2 - \frac{3}{2}\left(x + \frac{1}{2}\right)^2 & \text{dla } 0 \leqslant x \leqslant \frac{1}{2}, \\ \frac{1}{2}\left(\frac{3}{2} - x\right)^2 & \text{dla } \frac{1}{2} \leqslant x \leqslant \frac{3}{2}, \\ 0 & \text{dla } x > \frac{3}{2}. \end{cases} \tag{3.78}$$

$$M_4(x) = \begin{cases} \frac{1}{6}(2 - x)^3 - \frac{4}{6}(1 - x)^3 & \text{dla } 0 \leqslant x \leqslant 1, \\ \frac{1}{6}(2 - x)^3 & \text{dla } 1 \leqslant x \leqslant 2, \\ 0 & \text{dla } x > 2. \end{cases} \tag{3.79}$$

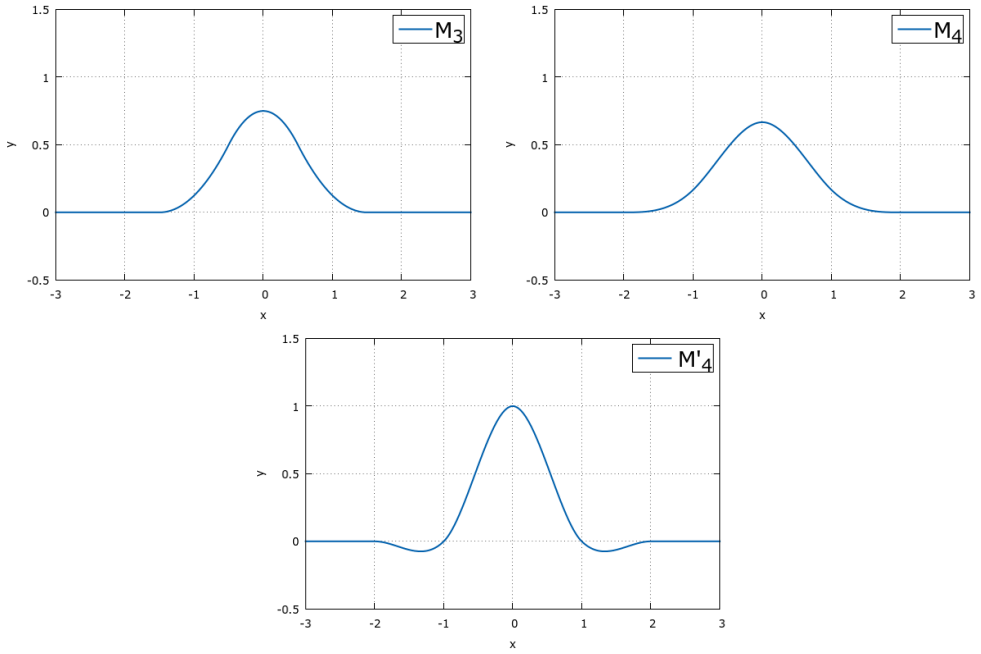
$$M_4'(x) = \begin{cases} 1 - \frac{5}{2}x^2 + \frac{3}{2}x^3 & \text{dla } 0 \leqslant x \leqslant 1, \\ \frac{1}{2}(1 - x)(2 - x)^2 & \text{dla } 1 \leqslant x \leqslant 2, \\ 0 & \text{dla } x > 2. \end{cases} \tag{3.80}$$

Używając jąder interpolacyjnych przedstawionych powyżej nowe cząstki otrzymują wkład od wirowości z i-tej starej cząstki Γ_i na nowe położenie $(\tilde{x}_j, \tilde{y}_j, \tilde{z}_j)$ według wzoru

$$\omega_j = \tilde{\Gamma}_p \varphi\left(\frac{x_j - \tilde{x}_p}{h}\right) \varphi\left(\frac{y_j - \tilde{y}_p}{h}\right) \varphi\left(\frac{z_j - \tilde{z}_p}{h}\right) \frac{1}{h^3}, \tag{3.81}$$

gdzie φ jest jednowymiarowym jądrem interpolacyjnym.

Jądra interpolacyjne rodziny M są wyznaczone z funkcji sklejanych. Charakteryzują się większą regularnością niż rodzina Λ. Jądro M_3 jest pierwszym



Rysunek 3.2: Rodzina jąder interopolacyjnych M.

z centralnych funkcji B-sklejanych, który ma ciągłe pierwsze pochodne. Jądro M_4 jest już klasy C^2. Obydwie te funkcje są rzędu drugiego ale jako funkcje B-sklejane są w stanie dokładnie odtworzyć wyłącznie funkcje liniowe. Poprawione jądro M_4' przedstawione w [77] jest wyższego rzędu (trzeciego). Zastosowanie go do metody VIC pozwala na zachowanie trzech momentów wirowości (całkowitą cyrkulację, impuls liniowy i impuls kątowy). Jądro M_4' przedstawione jest na rys. 3.2. Widoczny jest fakt zmiany znaku funkcji. Jądro M_4' jest używane w wypadkach kiedy potrzebna jest wysoka dokładność. Zaletą jest to, że jest gładkie (klasy C^1), a w [19] zostało pokazane, że jest bardziej dokładne niż Λ_2, które jest tego samego rzędu. Można powiedzieć, że jest to najpopularniejszy schemat interpolacyjny używany w metodach typu „Wiru w komórce". Jest także używany w SPH (*ang.* Smooth Particle Hydrodynamics).

3.7. Metody symulacji lepkości dla metod cząstek wirowych

Kolejnym krokiem w kierunku odtworzenia rzeczywistego przepływu jest wprowadzenie do algorytmu członów modelujących lepkość płynu. Wykorzystywane są różne schematy. Historycznie pierwszym była Metoda Przypadkowego Błądzenia [15]. Jest ona szybka ale niezbyt dokładna i bardzo nieefektywna dla przepływów trójwymiarowych. Wykorzystywane są także inne metody jak PSE (*ang.* Particle Strength Exchange). Metody te krytycznie zależą od zachowania pewnych relacji pomiędzy tzw. promieniem obcięcia osobliwości, którego wielkość dobiera się

arbitralnie co może być poważnym problemem, gdyż elementy Lagrange'a mogą zachowywać się chaotycznie ze względu na konwekcję. Metody te są zatem zależne od stosowania remeshingu z interpolacją jądrami wysokich rzędów.

Równania ruchu dla płynu lepkiego i nieściśliwego zawierają w sobie dwa zjawiska: konwekcję (człony bezwładnościowe) oraz dyfuzję. Wprowadzenie do rozwiązania siatki numerycznej wprowadza pewną lepkość numeryczną. Ponieważ wartość współczynnika lepkości jest mała, może się okazać, rząd lepkości wynikającej z dyskretyzacji równań, może być właśnie rzędu rzeczywistej lepkości płynu. Dodatkowo człony bezwładnościowe mają dużą wartość i przesłaniają człon dyfuzyjny. Może to prowadzić do zmiany fizyki badanego numerycznie zjawiska i powoduje, że rozwiązanie takiego równania stanowi poważne wyzwanie.

Jednym ze sposobów radzenia sobie z tego typu problemami jest dekompozycja lepkościowa (ang. operator splitting). Główną ideą takiego podejścia jest to, że całkowity operator ewolucji w czasie jest zapisany jako suma prostszych operatorów ewolucyjnych. Inaczej mówiąc, równanie jest rozbijane na mniejsze równania, dla których istnieją efektywne schematy obliczeniowe. Metoda numeryczna rozwiązywania tego równania jest wtedy formowana poprzez zestawienie ze sobą tych schematów.

Załóżmy przykładowo, że zadane jest zagadnienie początkowe opisujący ewolucję zjawiska

$$\frac{dU}{dt} + A(U) = 0 \quad U(0) = U_0. \tag{3.82}$$

Niech S_t będzie operatorem rozwiązującym powyższe zagadnienie czyli $U(t) = S_t U_0$. Przyjmijmy, że operator A może być rozłożony na sumę dwóch elementarnych operatorów: $A = A_1 + A_2$. Dla każdego z operatorów A_j można otrzymać rozwiązanie

$$U_j(t) = S_t^j U_0 = e^{-tA_j} U_0 \quad j = 1, 2. \tag{3.83}$$

Przybliżone rozwiązanie zagadnienia całkowitego otrzymuje się jako

$$U(\Delta t n) \approx [S^2_{\Delta t} S^1_{\Delta t}]^n = (e^{-\Delta t A_2} e^{-\Delta t A_1})^n U_0. \tag{3.84}$$

Wzór (3.84) nosi nazwę wzoru Lie-Trottera-Kato [35].

Mimo, że metoda dekompozycji operatorów wprowadza pewien dodatkowy błąd to ma kilka zalet. Pozwala ona na uzyskanie prostej do implementacji i bardziej efektywnej metody numerycznej. Dzięki niej można połączyć wyspecjalizowane metody numeryczne rozwinięte w celu efektywnego rozwiązania danej klasy zagadnień ewolucyjnych. W ten sposób można wybierać ze zbioru wydajnych i dobrze przetestowanych metod numerycznych dla podstawowych operatorów, które można ze sobą połączyć w celu rozwiązania skomplikowanych problemów, a także, w miarę potrzeb, na łatwą zamianę schematu numerycznego dla każdego z operatorów. Co więcej wykorzystanie takiego algorytmu pozwala czasami na zmniejszenie zapotrzebowania na pamięć operacyjną, zwiększenie przedziału

stabilności metody, a nawet stworzenie metody stabilnej bezwarunkowo. Dla zagadnień wielowymiarowych może to być czasami jedyne możliwe do zastosowania rozwiązanie. Ostatnią zaletą jest możliwość (oraz łatwość) dodawania kolejnych poziomów złożoności do modelu numerycznego ponieważ każdy nowy, kolejny wyraz (model) będzie niezależnym modułem numerycznym.

Przykładem zastosowania metody dekompozycji operatora może być popularna, metoda naprzemiennych kierunków (*ang.* ADI - Alternating Direction Implicit), w której wielowymiarowe zagadnienie jest redukowane do sekwencji zagadnień jednowymiarowych.

Rozbijanie równań ruchu na części lepkie i nielepkie w metodach krokowych jest wykorzystywane w wielu przypadkach. W [17] pomysł ten nawiązuje do podziału dokonanego przez Prandtla w 1904 r. na efekty lepkie i nielepkie. Metoda dekompozycji lepkościowej zwana także metodą kroków cząstkowych (*ang.* fractional step method) jest szczególnym przypadkiem techniki nazywanej dekompozycją operatorów i została wprowadzona do metod wirowych wraz z metodą przypadkowego błądzenia przez Chorina w [15]. Schemat ten składa się z dwóch podkroków, w których efekty konwekcji i dyfuzji są rozpatrywane jeden po drugim.

Schemat dwukrokowy ma dokładność drugiego rzędu dla danego kroku czasowego oraz pierwszego rzędu dla całej symulacji (możliwe jest podniesienie rzędu metody poprzez zastosowanie metody Stranga [66]). Aby pokazać rząd aproksymacji wynikający z dekompozycji operatorów zastosowany zostanie on do liniowego równania konwekcji-dyfuzji dla funkcji skalarnej W.

$$\frac{\partial W}{\partial t} + \mathbf{c} \cdot \nabla W = \nu \Delta W, \tag{3.85}$$

z warunkiem początkowym $W_0(\mathbf{x})$. Używając zapisu operatorowego, gdzie $\mathbf{c}\cdot\nabla \rightarrow A$ i $\nu\Delta \rightarrow B$, powyższe równanie może zostać zapisane jako $\frac{dW}{dt} = AW + BW$, którego całka ma postać

$$w(t) = W_0 e^{(A+B)t}. \tag{3.86}$$

Biorąc pod uwagę kroki czasowe o długości δt rozwiązanie w kroku czasowym $n\delta t$ jest wykorzystywane jako warunek początkowy przy rozwiązywaniu zagadnienia w kolejnym kroku czasowym $W^{n+1} = e^{(A+B)\delta t} W^n$. Metoda kroków cząstkowych wyraża się wtedy następująco:

- Pod-krok 1: Konwekcja

$$\frac{dW}{dt} = AW \Rightarrow W^{n+\frac{1}{2}} = e^{A\delta t} W^n. \tag{3.87}$$

- Pod-krok 2: Dyfuzja

$$\frac{dW}{dt} = BW \Rightarrow W^{n+1} = e^{B\delta t} W^{n+\frac{1}{2}}. \tag{3.88}$$

- Rozwiązanie przybliżone

$$W^{n+1} = e^{B\delta t} e^{A\delta t} W^n. \tag{3.89}$$

Dla małej wartości δt operator w wyrażeniu powyżej można rozłożyć w szereg Taylora

$$W^{n+1} = \left(I + \delta t B + \frac{\delta t^2}{2} B^2 + ... \right) \left(I + \delta t A + \frac{\delta t^2}{2} A^2 + ... \right). \tag{3.90}$$

Porównanie z rozkładem w szereg Taylora dokładnego operatora $e^{(A+B)\delta t}$ pokazuje, że są one równe tylko dla operatorów komutujących. W przypadku ogólnym jednak operatory A i B nie komutują ze sobą a tym samym popełniany błąd jest rzędu $\mathcal{O}(\delta t^2)$ w każdym kroku czasowym. Dzieląc przedział czasowy $t \in (0, T)$ na $n = (T-0)/\delta t$ równych warstw, błąd popełniany w czasie $t_n = n\delta t$ jest rzędu $\mathcal{O}(\delta t)$. Z tego powodu metoda dekompozycji lepkościowej jest pierwszego rzędu po czasie.

Metoda kroków cząstkowych dla równań Naviera - Stokesa wyraża się poprzez $\boldsymbol{\omega}^{n+1} = H^\nu(\delta t) E(\delta t) \boldsymbol{\omega}^n$, gdzie $E(t)\boldsymbol{\omega}_0$ i $H^\nu(t)\boldsymbol{\omega}_0$ są odpowiednio rozwiązaniami w czasie t równań Eulera i dyfuzji z warunkiem początkowym $\boldsymbol{\omega}_0$. Metoda składa się z przemieszczenia cząstek wirowych zgodnie z równaniem Eulera w pierwszym podkroku oraz wzięcia pod uwagę dyfuzji w drugim podkroku. Dla dwuwymiarowego przepływu lepkiego mamy

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \Delta \omega. \tag{3.91}$$

- Pod-krok 1: Konwekcja

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p), \tag{3.92}$$

$$\frac{d\alpha_p}{dt} = 0. \tag{3.93}$$

- Pod-krok 2: Dyfuzja

$$\frac{d\omega_p}{dt} = \nu \Delta \omega. \tag{3.94}$$

Dla przepływu trójwymiarowego dochodzi do równania Naviera-Stokesa człon związany z rozciąganiem linii wirowych. W tym wypadku algorytm wygląda następująco:

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu \Delta \omega. \tag{3.95}$$

- Pod-krok 1: Konwekcja

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p), \tag{3.96}$$

$$\frac{d\alpha_p}{dt} = [\nabla \mathbf{u}(\mathbf{x}_p, t)] \cdot \alpha_p. \tag{3.97}$$

- Pod-krok 2: Dyfuzja

$$\frac{d\omega_p}{dt} = \nu \Delta \omega. \tag{3.98}$$

3.8. Algorytm numeryczny metody wirowej typu „Wir w komórce”

Poniżej zostanie przedstawiony algorytm numeryczny, wedle którego realizowane były obliczenia metodą typu ”Wir w komórce”

Obszar obliczeniowy został pokryty siatką obliczeniową o równomiernym kroku h w każdym kierunku. Na każdym węźle siatki inicjowana była cząstka wirowa o intensywności

$$\alpha_p = \int_{dxdydy} \boldsymbol{\omega}(x, y, z, t_0) dxdydz \approx \boldsymbol{\omega}(x_n, y_n, z_n, t_0) \cdot h^3, \tag{3.99}$$

gdzie x_n, y_n, z_n były współrzędnymi węzła siatki.

W niniejszej pracy początkowe lokalizacje cząstek wybierane były na węzłach siatki kartezjańskiej. Alternatywą może być inicjalizacja pseudo-losowa [17] w której cząstki są inicjalizowane w losowych lokalizacjach wewnątrz komórek siatki.

W każdym kroku czasowym wykonywane są następujące obliczenia:

1. Wyznaczany był potencjał wektorowy $\mathbf{A}$ rozwiązując równanie Poissona (3.21) dla trzech składowych wirowości.

$$\Delta A_i = -\omega_i, \qquad i = 1, 2, 3. \tag{3.100}$$

   Na podstawie potencjału wektorowego wyznaczane było pole prędkości ze wzoru

$$\mathbf{u} = \nabla \times \mathbf{A}. \tag{3.101}$$

2. Cząstki z węzłów siatki przemieszcza się zgodnie z lokalnym polem prędkości. Do rozwiązania równania (3.96) wykorzystana została metoda Rungego-Kutty czwartego rzędu. Prędkość z węzłów na aktualne położenie cząstki była interpolowana za pomocą jądra M_4' (3.80).

3. Zmiana intensywności cząstek (na skutek rozciągania linii wirowych) była realizowana zgodnie ze wzorem (3.97) również metodą Rungego-Kutty czwartego rzędu. Wartości pochodnych prędkości pod operatorem gradientu były obliczane za pomocą metody różnic skończonych drugiego rzędu. Prędkość z węzłów siatki na aktualne położenia cząstek była interpolowana

za pomocą jądra M_4' (3.80). W tym wypadku lepszym sposobem obliczania prawej strony równania (3.97) jest użycie transpozycji gradientu prędkości

$$\frac{d\alpha_p}{dt} = [\nabla \mathbf{u}(\mathbf{x}_p, t)]^T \cdot \alpha_p. \tag{3.102}$$

Schemat ten dawał najlepsze wyniki jeżeli chodzi o zachowanie energii kinetycznej. W opracowaniach [17, 81] autorzy wskazują, że obydwie postaci członów źródłowych dają te same wyniki jeżeli spełniony jest warunek $\nabla \cdot \boldsymbol{\omega} \equiv 0$. Niestety obliczenia numeryczne wprowadzają pewną niewielką niezerową źródłowość pola wirowości. W tym wypadku w pracach [81, 85] postulowane jest, że najlepszy jest schemat transponowany.

4. Po zakończonym kroku przemieszczania cząstek ich intensywność była redystrybuowana z powrotem na węzły siatki. W niniejszej pracy było to wykonywane w każdym kroku czasowym a nie po arbitralnie przyjętej liczbie kroków czasowych. Zastosowanie takiego rozwiązania pozwoliło na rozwiązywanie równania dyfuzji na siatce, a co za tym idzie wykorzystanie dokładnych schematów numerycznych dopasowanych do tego równania. Dodatkowo pozwoliło to na zredukowanie algorytmu o jedną operację interpolacji (nowej intensywności na stare położenia cząstek).

5. Ostatnim krokiem było rozwiązanie równania dyfuzji na siatce numerycznej. Równanie dyfuzji w niniejszej pracy było rozwiązywane za pomocą metody różnic skończonych. To podejście częściowo ogranicza zyski związane z wykorzystaniem metod Lagrange'owskich, ponieważ wprowadza do obliczeń siatkę numeryczną oraz dyfuzję numeryczną. Dzięki temu jednak możliwe jest dokładne rozwiązanie równania dyfuzji oraz przyspieszenie obliczeń.

Rozdział 4

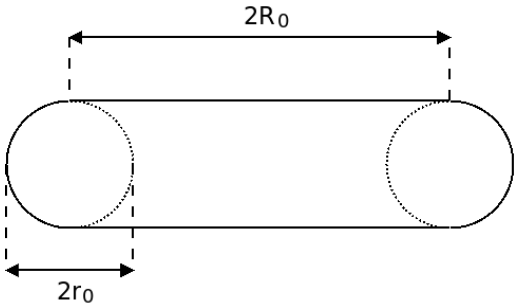
Badania numeryczne metodą typu Wir-w-komórce na pojedynczym procesorze

Przedstawiony w poprzednim rozdziale algorytm realizujący obliczenia metodą cząstek wirowych typu „Wir w komórce" został zaimplementowany w języku programowania Fortran 90.

Obszar obliczeniowy został opisany strukturalną siatką numeryczną. Na każdym węźle siatki została umieszczona cząstka wirowa. Początkowa intensywność cząstki była obliczana zgodnie ze wzorem (3.18). Następnie rozwiązywany był układ równań ruchu cząstki (3.96), (3.97). Prędkość na węzłach siatki była wyznaczona z równania Poissona (3.21). Do jego rozwiązania została użyta biblioteka Fishpack [95, 96], która wykorzystuje metodę szybkiego rozwiązywania równania Poissona (*ang.* Fast Poisson Solver - FPS) drugiego rzędu. Równania ruchu zostały rozwiązane metodą Rungego-Kutty czwartego rzędu. Wartości prędkości cząstki były interpolowane z węzłów siatki za pomocą jądra (3.80). Po każdym kroku czasowym wykonywana była redystrybucja wirowości na węzły siatki przy pomocy jądra interpolacyjnego (3.80). To kończyło krok czasowy przy symulacji przepływu nielepkiego. Przedstawione poniżej wyniki odnoszą się do płynu nielepkiego.

4.1. Badanie numeryczne prędkości przemieszczania się pierścienia wirowego

Aby sprawdzić poprawność implementacji metody został wykonany test, w którym badana była prędkość przemieszczania się pierścienia wirowego w czasie.



Rysunek 4.1: Geometria pierścienia wirowego

Pierścień wirowy wraz z oznaczeniami przedstawiony jest na rys. 4.1. Dla pierścienia wirowego o cienkim rdzeniu ($r_0/R_0 << 1$) i stałym rozkładzie wirowości wewnątrz rdzenia prędkość translacji można wyznaczyć ze wzoru Kelvina [29]:

$$U = \frac{\Gamma}{4\pi R_0}\left[\ln\left(\frac{8R_0}{r_0}\right) - \frac{1}{4} + O\left(\frac{r_0}{R_0}\right)\right]. \tag{4.1}$$

Istnieje także wzór na prędkość pierścienia wirowego z nieruchomym płynem wewnątrz (wirowość znajduje się jedynie na obwodzie rdzenia) nazywany wzorem Hicksa [88]:

$$U = \frac{\Gamma}{4\pi R_0}\left[\ln\left(\frac{8R_0}{r_0}\right) - \frac{1}{2} + O\left(\frac{r_0}{R_0}\right)\right]. \tag{4.2}$$

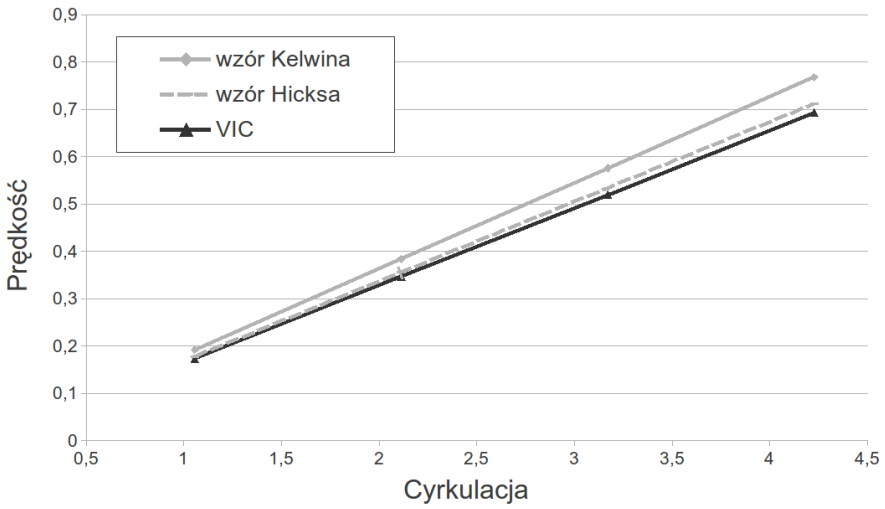
W celu porównania wyników symulacji z wartościami teoretycznymi oraz sprawdzenia poprawności kodu, przeprowadzone zostały badania numeryczne, w których promień pierścienia i promień w chwili początkowej wynosiły $R_0 = 1.5$, $r_0 = 0.3$, a zmieniano początkową wartość cyrkulacji w przedziale $\Gamma \in [1.06, 4.23]$. Obszar obliczeniowy miał wymiary $2\pi \times 2\pi \times 2\pi$ i był podzielony siatką numeryczną o $128 \times 128 \times 128$ węzłach. We wszystkich kierunkach były zadane okresowe warunki brzegowe. Warunek początkowy był realizowany poprzez nadanie odpowiedniej wartości wektora intensywności wszystkim węzłom siatki numerycznej, które spełniały równanie

$$r_p = \left(\sqrt{x^2 + y^2} - R_0\right)^2 + z^2 < r_0^2, \tag{4.3}$$

czyli leżały wewnątrz pierścienia wirowego.

Równania (3.96) i (3.97) były rozwiązywane w każdym kroku czasowym metodą Rungego-Kutty czwartego rzędu z krokiem czasowym $\triangle t = 0.01$. Wyniki symulacji widoczne są na rys. 4.2. Przemieszczenie pierścienia wirowego dla największej wartości cyrkulacji ($\Gamma = 4.23$) przedstawiono na rys. 4.3 i rys. 4.4.

Na rys. 4.2 widać dobrą zbieżność pomiędzy wynikami teoretycznymi i numerycznymi. Pewna rozbieżność pomiędzy wynikami numerycznymi a teoretycznymi może wynikać z faktu, że formuła (4.1) została wyprowadzona dla pierścienia



Rysunek 4.2: Porównanie teoretycznej i numerycznej prędkości przemieszczania się pierścienia wirowego.

wirowego o stałym rozkładzie wirowości w jądrze. Podczas przemieszczenia się pierścienia w symulacji numerycznej rozkład wirowości w jądrze zmienia się w czasie i nie jest stały (rys. 4.5). Warto zauważyć, że otrzymane wyniki są bliższe wartościom wynikającym ze wzoru Hicksa (4.2) wyprowadzonemu dla ruchu pierścienia wirowego z nieruchomym płynem w jądrze. W trakcie ewolucji małe porcje wirowości ułożyły się w podłużne struktury ciągnięte za pierścieniem. Jest to dokładnie widoczne na rys. 4.4 gdzie pokazano początkową i końcową pozycję wiru bez odcinania małych wartości wirowości. Takie zaburzenia otrzymywane były również przez innych autorów, na przykład w [5].

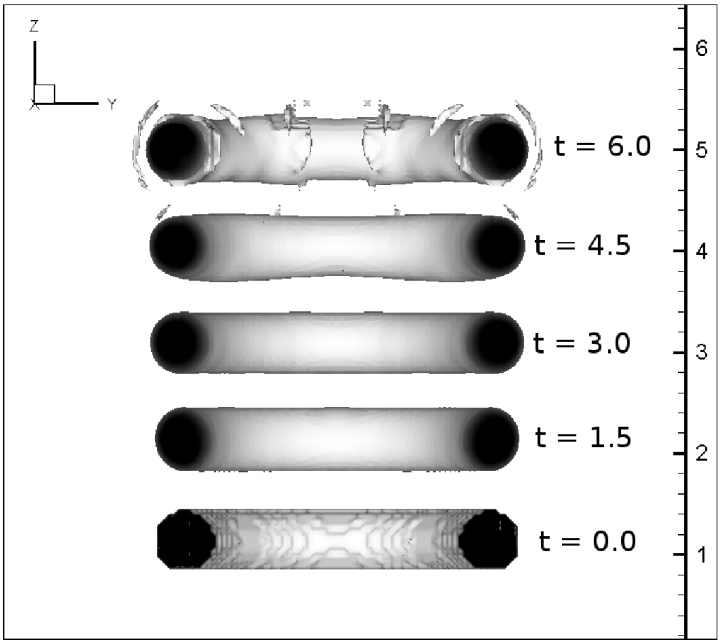
W czasie obliczeń monitorowane były niezmienniki ruchu przedstawione w sekcji 3.5 w tym energia kinetyczna, która wyznaczana była z dwóch różnych wzorów (3.45) i (3.56).

Na końcu obliczeń (czyli po 750 kroków czasowych) energia kinetyczna wyznaczona z obu równań zmieniła się o mniej niż 2%. Wartości całek z dywergencji potencjału wektorowego $\mathbf{A}$ (warunek potrzebny przy wyprowadzeniu równania (3.21)), dywergencji wirowości ($\boldsymbol{\omega}$) oraz dywergencji prędkości ($\mathbf{u}$) po całym obszarze obliczeniowym były mniejsze 10^{-10} w ciągu całych obliczeń.

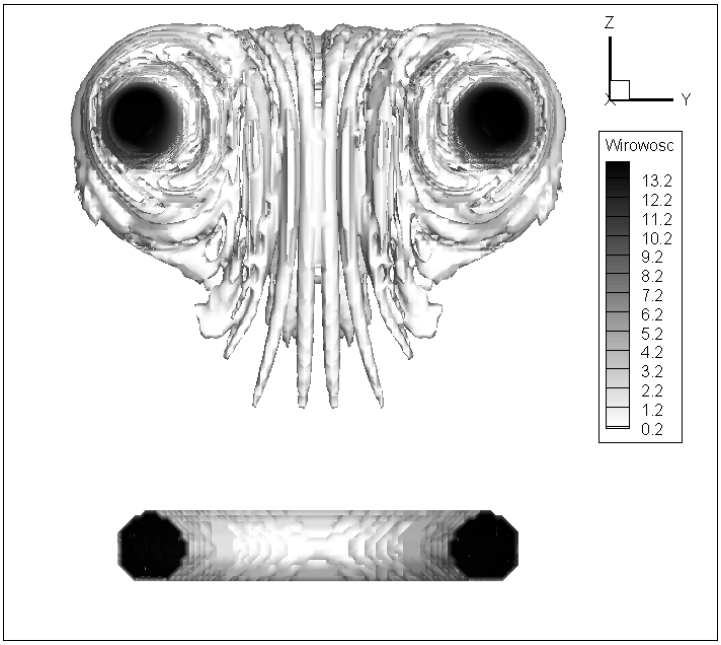
Wyniki powyższych badań zostały opublikowane w artykule [52].

4.2. Porównanie ewolucji pierścieni wirowych z różnymi rozkładami wirowości w rdzeniu

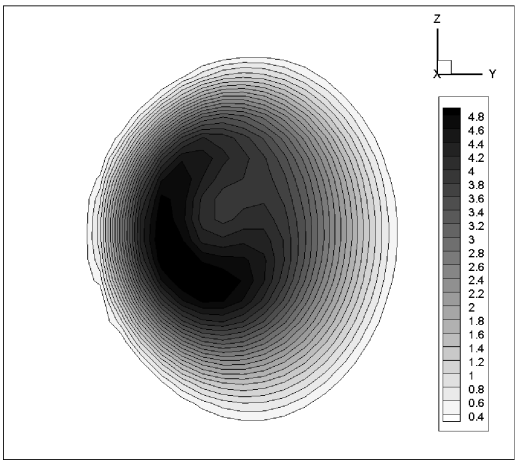
Niestabilność pierścienia wirowego przedstawionego w poprzednim punkcie wynika z zadanego rozkładu wirowości w rdzeniu. Uzasadnionym wydaje się zatem



Rysunek 4.3: Ewolucja pierścienia wirowego w czasie dla przypadku $\Gamma = 4.23$. Sekwencja izopowierzchni wirowości dla $|\omega| \in [6, 14]$ w różnych chwilach czasowych t



Rysunek 4.4: Ewolucja pierścienia wirowego w czasie dla przypadku $\Gamma = 4.23$. Izopowierzchnie wirowości dla $|\omega| \in [0.2, 14]$ oraz początkowej i końcowej chwili czasu $t = 6.0$.



Rysunek 4.5: Izolinie wirowości w rdzeniu pierścienia wirowego w chwili czasu $t = 6.0$ dla przypadku $\Gamma = 1.06$. Początkowa wirowość w rdzeniu miała wartość $|\omega| = 3.75$.

porównanie zachowania się pierścienia z rozkładem stałym z pierścieniem z zadanym rozkładem normalnym Gaussa wewnątrz rdzenia.

Przeprowadzony został test porównawczy dla tych dwóch przypadków. Wielkości promienia pierścienia i promienia w chwili początkowej wynosiły $R_0 = 1.5$, $r_0 = 0.3$, a zmieniana była początkowa wartość cyrkulacji w zakresie $\Gamma \in [0.23, 0.94]$. Obszar obliczeniowy tak jak poprzednio miał rozmiar $2\pi \times 2\pi \times 2\pi$. Siatka numeryczna miała $257 \times 257 \times 257$ węzłów. Przyjęte zostały okresowe warunki brzegowe w kierunkach osi x, y i z. Warunek początkowy realizowany był poprzez nadanie każdemu węzłowi siatki spełniającemu warunek (4.3) niezerowych wartości wektora intensywności. Zbadano dwa różne początkowe rozkłady wirowości w rdzeniu pierścienia:

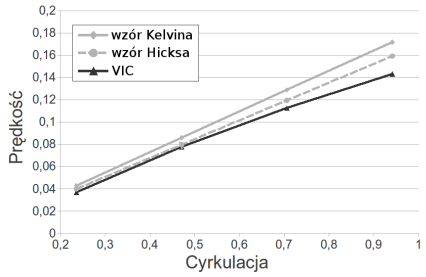
1. rozkład stały,

2. rozkład normalny Gaussa zadany wzorem:

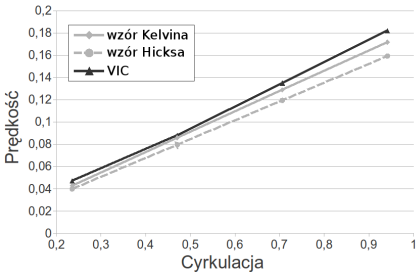
$$\omega_p = c\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{r_p^2}{2\sigma^2}}, \tag{4.4}$$

gdzie c był znanym zmiennym współczynnikiem, a $\sigma = r_0/3$. Początkowa liczba cząstek, którymi przybliżano zadaną geometrię pierścienia wirowego w obu przypadkach wynosiła 179780. Krok czasowy wynosił $\Delta t = 0.01$.

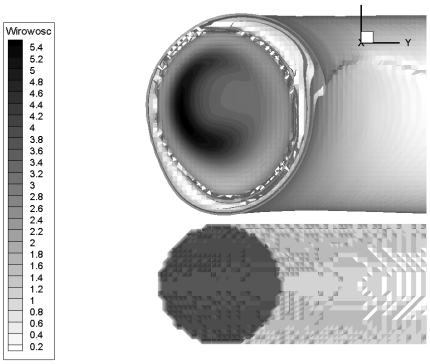
Na rys. 4.6 została pokazana prędkość przemieszczania się pierścienia wirowego z stałą wirowością w rdzeniu, natomiast na rys. 4.7 została pokazana prędkość przemieszczania się pierścienia wirowego z rozkładem normalnym w rdzeniu.



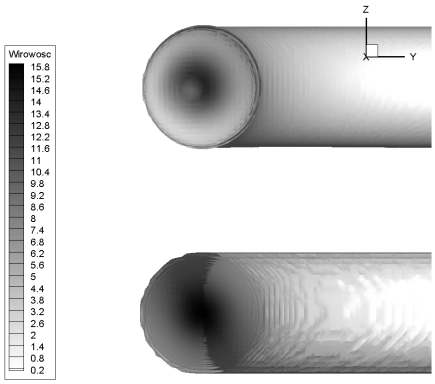
Rysunek 4.6: Porównanie teoretycznych i numerycznych wartości prędkości przemieszczania się pierścienia wirowego dla stałego rozkładu wirowości w rdzeniu.



Rysunek 4.7: Porównanie teoretycznych i numerycznych wartości prędkości przemieszczania się pierścienia wirowego dla normalnego rozkładu wirowości w rdzeniu.



Rysunek 4.8: Ewolucja w czasie pierścienie wirowego ze stałym rozkładem wirowości w rdzeniu ($\Gamma = 0.94$); $t = 0.0$ (dół); $t = 6.0$ (góra)



Rysunek 4.9: Ewolucja w czasie pierścienie wirowego ze normalnym rozkładem wirowości w rdzeniu ($\Gamma = 0.94$); $t = 0.0$ (dół); $t = 6.0$ (góra)

Prędkość jest w tym wypadku większa (krzywa $V(t)$ leży nad krzywymi wynikającymi ze wzorów Kelvina i Hicksa). Wyniki te pokazują, że rozkład wirowości we wnętrzu rdzenia istotny wpływ na ruch pierścienia.

Na rys. 4.8 i rys. 4.9 widoczna jest ewolucja w czasie pierścieni wirowych o tej samej cyrkulacji ale różnego początkowego rozkładu wirowości. Na rys. 4.8 widać, że dla pierścienia wirowego ze stałym rozkładem początkowym, rozkład ten znacząco zmienia się w trakcie ewolucji. Dodatkowo na obwodzie pojawiają się pewne wyciągające się struktury tak jak to miało miejsce w poprzednim przykładzie. Dla rozkładu normalnego dystrybucja wirowości w rdzeniu na końcu obliczeń jest bardziej gładka oraz, w przybliżeniu, dalej przypomina rozkład normalny. Struktura pierścienia wirowego z normalnym rozkładem wirowości w rdzeniu jest zdecydowanie mniej zaburzona w stosunku do pierścienia z rozkładem

stałym. Dodatkowo, jak widać na rys. 4.5 w czasie ewolucji pierścienia z początkowym stałym rozkładem wirowości wykształca się wyraźne maksimum wirowości a na granicy pierścienia wartości dążą do zera. Można stąd wyciągnąć wniosek, że początkowy rozkład wirowości we wnętrzu rdzenia ma wpływ na stabilność ruchu pierścienia wirowego.

W czasie obliczeń monitorowane były niezmienniki ruchu przedstawione w sekcji 3.5 w tym energia kinetyczna, która wyznaczana była z dwóch różnych wzorów (3.45) i (3.56).

Na końcu obliczeń (czyli po 600 kroków czasowych) energia kinetyczna obliczona z obu wzorów spadła o mniej niż 2% wartości początkowej. Wartości całek z dywergencji potencjału wektorowego $\mathbf{A}$ (warunek potrzebny przy wyprowadzeniu równania (3.21)), wirowości ($\boldsymbol{\omega}$) oraz prędkości ($\mathbf{u}$) po całym obszarze obliczeniowym były mniejsze 10^{-10} w ciągu całych obliczeń.

Wyniki powyższych badań zostały opublikowane w artykule autorskim [59].

Rozdział 5

Obliczenia równoległe na karcie graficznej

5.1. Uwagi wstępne do obliczeń równoległych

Rozwiązywanie trójwymiarowych równań Naviera-Stokesa dowolną metodą numeryczną jest czasochłonne, szczególnie dla dużych liczb Reynoldsa. Nie inaczej jest w metodzie cząstek wirowych przedstawionej w poprzednich rozdziałach. W ostatnich latach moc obliczeniowa pojedynczego rdzenia procesora przestała rosnąć. Głównym kierunkiem rozwoju stało się zwiększanie liczby rdzeni procesora a co za tym idzie wykorzystanie obliczeń równoległych. W niniejszej pracy postanowiono dokonać przeniesienia obliczeń numerycznych związanych z metodą cząstek wirowych typu „Wir w komórce" na architektury równoległe. Ponieważ duża część obliczeń odnosi się do pojedynczego elementu (jak cząstka wirowa albo węzeł siatki) do obliczeń może być efektywnie zastosowana architektura SIMD (*ang.* Single Instruction Multiple Data). Cechą charakterystyczną tej architektury jest fakt, że wszystkie jednostki obliczeniowe wykonują jednocześnie to samo polecenie. Obliczenia związane z pojedynczą cząstką wirową (takie jak interpolacja czy przemieszczanie cząstek) nie są zależne od położenia pozostałych cząstek i dlatego mogą odbywać się równolegle. Ponieważ w obszarze obliczeniowym może znajdować się bardzo dużo cząstek (nawet kilka milionów) to wykorzystanie obliczeń równoległych jest bardzo pożądane. Opisany w rozdziale 3 algorytm numeryczny dla metody VIC ze względu na możliwość przeniesienia obliczeń na architektury równoległe można podzielić na dwie części:

- część, dla której możliwe jest bezpośrednie zrównoleglenie kodu (ruch cząstek, wyznaczanie prędkości w węzłach siatki),
- oraz część, dla której zrównoleglenie obliczeń wymaga zmiany stosowane-

go algorytmu i przystosowanie go do obliczeń równoległych (rozwiązanie równania Poissona i równania dyfuzji).

W pracy do obliczeń równoległych zostało wykorzystane środowisko wieloprocesorowe kart graficznych (GPU - *ang.* Graphics Processing Unit). Charakteryzuje się ono m.in. bardzo dobrym stosunkiem mocy obliczeniowej do ceny oraz wysoką mocą obliczeniową. Niestety wymagane jest inne podejście do programowania niż przy użyciu obliczeń sekwencyjnych. Należy starannie zapoznać się ze strukturą i budową tej architektury, aby móc w pełni wykorzystać jej moc obliczeniową.

Do obliczeń w pracy wybrane zostały karty graficzne firmy NVIDIA wyposażone w technologię CUDA (*ang.* Compute Unified Device Architecture). Głównym powodem była łatwość przejścia do programowania równoległego przy wykorzystaniu języka programowania „C for CUDA" – stanowiącego rozwinięcie popularnego języka programowania C.

5.2. Uwagi o architekturze CUDA

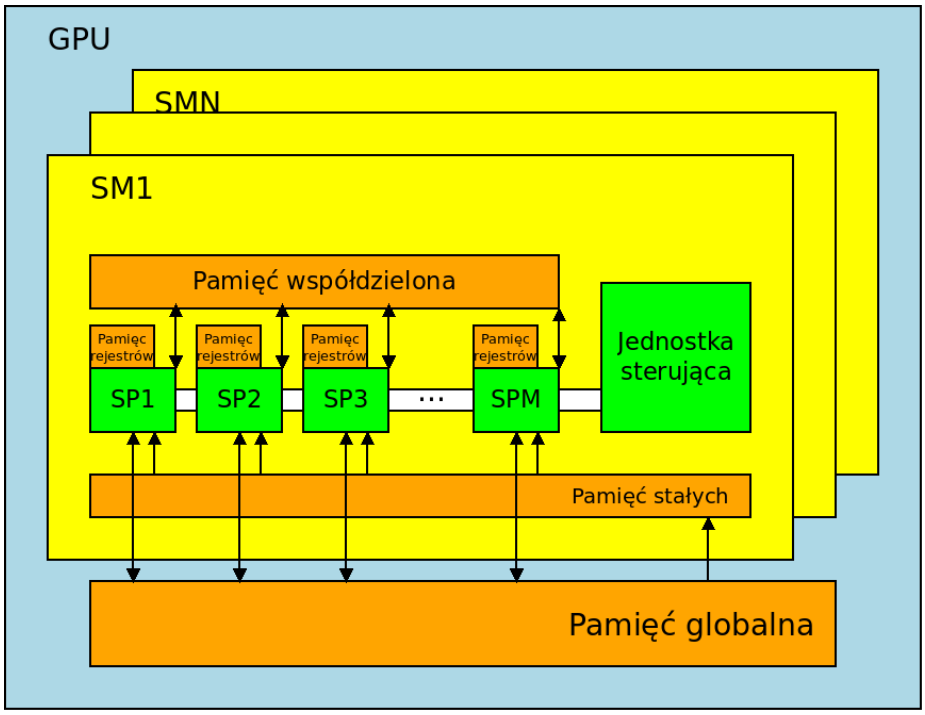
Dla zapoznania z programowaniem na kartach graficznych (GPU), zostanie dokonany przegląd z zakresu struktury sprzętu oraz interfejsu programowania (API - *ang.* Application Programming Interface).

Firma NVIDIA jest zdecydowanym liderem jeżeli chodzi o sprzedaż kart graficznych ogólnego zastosowania (*ang.* GPGPU - General Purpose GPU). W ofercie tej firmy można znaleźć m.in. karty o nazwie Tesla przeznaczone specjalnie do obliczeń numerycznych, jak również technologię CUDA API. Wsparcie przez firmę NVIDIA i rozwój społeczności programistów używających CUDA sprawiają, że jest ona liderem liczby prowadzonych projektów związanych z obliczeniami numerycznymi na kartach graficznych.

Do programowania kart graficznych firmy NVIDIA można wykorzystać interfejs API oraz rozszerzenie języka C/C++ o nazwie „C for CUDA", które daje dostęp do pamięci niskiego poziomu i procesów obliczeniowych na GPU. Kody CUDA są kompilowane przez dedykowany kompilator o nazwie `nvcc`.

Model programowania jest ściśle związany z fizyczną budową kart graficznych. Na rynku dostępnych jest kilka różnych modeli i generacji kart GPU. Poznanie budowy konkretnego modelu na którym będą prowadzone obliczenia pozwala na właściwy dobór parametrów dla wydajniejszej paralelizacji programu programu. Kod napisany w języku „C for CUDA" może być uruchomiony na każdej karcie obsługującej technologię CUDA.

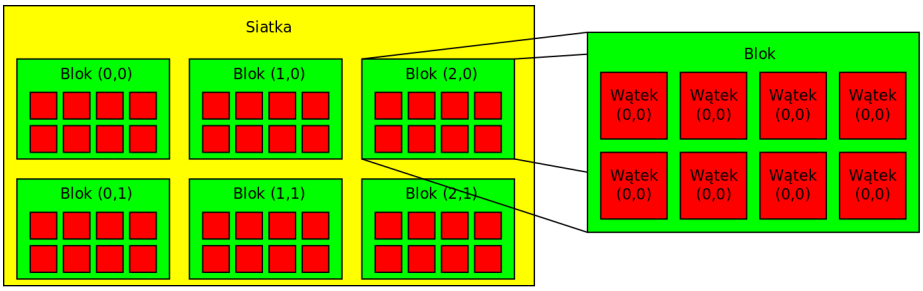
Karta graficzna jest oddzielnym podzespołem komputerowym podłączonym do komputera przez magistralę PCIe (*ang.* Peripheral Component Interconnect Express). W opracowaniu [3] komputer, do którego jest podłączona karta graficzna nazywany jest gospodarzem (*ang.* host). Uproszczony schemat budowy karty graficznej przedstawiony jest na rys. 5.1. Karta graficzna z architekturą CUDA



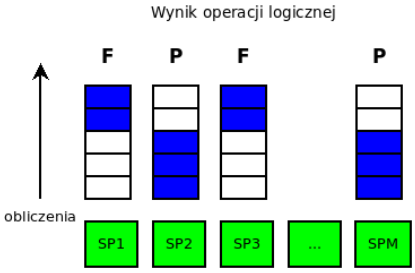
Rysunek 5.1: Budowa karty graficznej w architekturze CUDA.

jest zbudowana wokół wielowątkowych multiprocesorów strumieniowych (SM - *ang.* Streaming Multiprocessors). W różnych kartach graficznych znajduje się różną ich liczbę. Każdy z nich składa się z od 8 do 192 (w zależności od generacji) tak zwanych procesorów strumieniowych (SP - *ang.* Streaming Processors).

Kod, który będzie wykonany na karcie graficznej umieszczony jest w specjalnych funkcjach zwanych jądrami (lub z angielskiego kernelami). Zapisane w nich operacje są realizowane przez tak zwane wątki (*ang.* threads). Wątek to zbiór instrukcji wykonywanych przez pojedynczy procesor strumieniowy. Wszystkie wątki muszą wykonywać tą samą instrukcję w tym samym czasie, ale może się to odbywać na różnych zbiorach danych. Obliczenia wykonywane przez dany wątek są niezależne i równoległe do pozostałych. Aby lepiej odwzorować wykonywane zadanie wątki mogą być ustawione w dwu lub trójwymiarowe siatki (*ang.* grid). Siatki są dzielone na bloki (*ang.* block), to znaczy grupy wątków, które będą wykonywane na tym samym multiprocesorze. Synchronizacja i komunikacja pomiędzy tymi wątkami jest możliwa, ale dla uzyskania najlepszej wydajności powinna być ograniczona do minimum. Hierarchiczna struktura siatki, bloków i wątków przedstawiona jest na rys. 5.2. Ponieważ w wielu wypadkach bloków jest więcej niż dostępnych multiprocesorów, to będą one wykonywane sekwencyjnie w ramach dostępnych zasobów. Wprowadza się pojęcie czasu życia bloku. Jest to czas przez jaki wykonywane są obliczenia na jego wątkach na muliproce-



Rysunek 5.2: Podział logiczny na Siatkę i Bloki wątków w czasie wykonywania programu na GPU.



Rysunek 5.3: Sposób wykonywania instrukcji warunkowych przez multiprocesor.

sorze [3]. Podział wątków na grupy pozwala na lepszą wydajność obliczeń oraz uruchamianie programu na różnych modelach kart graficznych bez konieczności dokonywania zmian w kodzie.

Ważną konsekwencją istnienia tylko jednej jednostki sterującej dla multiprocesora jest sposób wykonywania przez kartę graficzną instrukcji warunkowych. Jeżeli w kodzie jądra znajduje się taka instrukcja to wykonanie jej jest kolejkowane tzn. najpierw zostaną wykonane obliczenia dla jednej gałęzi (np. gdy warunek jest prawdziwy) a dopiero potem dla drugiej gałęzi. Przedstawione jest to na rys. 5.3. Może to powodować wolniejsze działanie programu i wskazane jest, w miarę możliwości, omijanie instrukcji warunkowych w kodzie jądra.

Ważnym elementem opracowywania programów na architektury kart graficznych jest odpowiednie zarządzanie dostępami procesów do pamięci. W architekturze CUDA możemy wyróżnić następujące rodzaje pamięci [3]:

- pamięć urządzenia (pamięć globalna - *ang.* device memory),
- pamięć podręczną tekstur (*ang.* texture memory),
- pamięć podręczną stałych (*ang.* constant memory),
- pamięć współdzielona (*ang.* shared memory),
- pamięć rejestrów (*ang.* register memory).

Pamięć globalna GPU jest typu DRAM (*ang.* Dynamic Random Access Memory). Jej wielkość w zależności od typu karty graficznej może się wahać od 512 MB do nawet 12 GB. Jest ona dostępna dla wszystkich procesorów strumieniowych, wszystkich multiprocesorów. Niestety dostęp do tej pamięci jest bardzo powolny, szacuje się, że wymaga ok. 400 cykli zegara [3]. Przez ten czas dany wątek nie może wykonywać żadnych innych poleceń. Częste odwoływanie się przez wątek do pamięci globalnej może spowodować spadek wydajności programu. Jeżeli każda wartość zapisana w pamięci jest wykorzystywana tylko raz w programie, to procesu obliczeniowego nie można przyspieszyć. Jednak takie sytuacje są rzadkie. Częściej zdarza się, że zmienna używana jest w programie wielokrotnie. W takim wypadku korzystne jest wykorzystanie pamięci współdzielonej. Dostęp do tej pamięci trwa zaledwie kilka cykli zegara [3]. Wadą tego rozwiązania jest to, że dane z pamięci współdzielonej widoczne są jedynie dla wątków danego bloku, wykonywanych na danym mulitprocesorze. Mimo tej niedogodności liczba kosztownych dostępów do pamięci urządzenia może zostać znacząco zredukowana. W nowszych generacjach GPU pojawiła się pamięć podręczna drugiego poziomu (*ang.* L2 cache) znajdująca się pomiędzy DRAM a SM. Pozwala to w niektórych przypadkach na przyspieszenie dostępów do pamięci globalnej. Szybka pamięć znajdująca się na każdym SM może służyć jako pamięć podręczna pierwszego poziomu (*ang.* L1 cache) lub być przeznaczoną na pamięć współdzieloną przez wszystkie SP (*ang.* shared memory). Sposób jej podziału może być określony w czasie kompilacji programu.

Typowy program na GPU kopiuje dane z pamięci RAM komputera-gospodarza do RAMu karty, wykonuje obliczenia w oparciu o te dane, a po ukończeniu, przenosi wyniki z powrotem do RAMu gospodarza. Taki transfer danych powoduje duże opóźnienia w obliczeniach. Wskazane jest więc, wykonywanie jak największej ilości obliczeń bez transferu danych.

Pierwsze jednostki arytmetyczno-logiczne (ALU) z SP nie były w pełni zgodne ze standardami IEEE (*ang.* Institute of Electrical and Electronics Engineers), co oznaczało, że wyniki obliczeń na GPU mogły się nieznacznie różnić od wykonywanych na CPU. Arytmetyka podwójnej precyzji również nie była wspierana w pierwszych modelach. Aktualnie obliczenia prowadzone na GPU są w pełni zgodne z normami IEEE-754 oraz mogą być prowadzone w podwójnej precyzji. W niektórych modelach (głównie serii Tesla) dostępna jest pamięć RAM wyposażona w system kodowania korekcyjnego ECC (*ang.* Error Corecting Code). W tym rozwiązaniu przesyłane są nadmiarowe dane kontrolne, umożliwiające korygowanie pewnych błędów pamięci.

Najnowsza generacja procesorów graficznych NVIDIA nosi nazwę Kepler. W poprzedniej generacji (Fermi) oraz w obecnej wprowadzono wiele poprawek mających duży wpływ na czas obliczeń numerycznych. Podstawowymi ulepszeniami są:

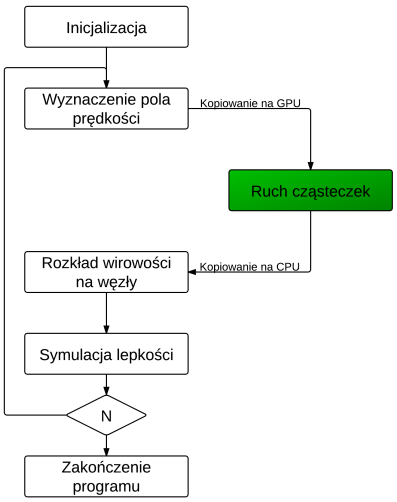
- Dodanie pamięci podręcznej L1 i L2, które pomaga zredukować opóźnienia pamięci.
- Wprowadzenie korekcji błędów pamięci ECC (dostępne w modelach serii Tesla).
- Wzrost liczby rdzeni (SP) na multiprocesorze (SM) do 192, wzrost całkowitej liczby rdzeni do 2880.
- Pełna zgodność operacji zmiennoprzecinkowych ze standardem IEEE-754.
- Ilość pamięci współdzielonej na SM zwiększona do 48kB.
- Liczba wątków obliczeniowych na blok wzrosła do 1024.

Rozdział 6

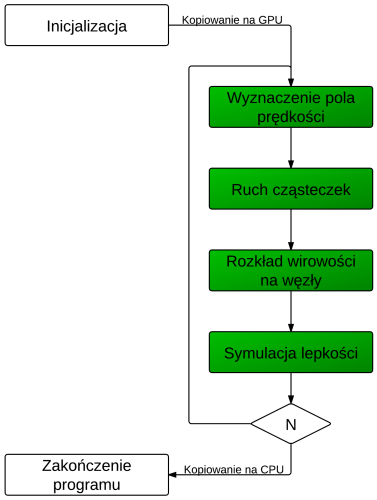
Implementacja algorytmu Wir-w-komórce na kartach graficznych

Zrównoleglenie kodu odpowiedzialnego za przemieszczanie cząstek oraz redystrybucję wektora wirowości z cząstek na węzły zostało dokonane poprzez przyporządkowanie obliczeń związanych z pojedynczą cząstką (węzłem siatki) do pojedynczego wątku wykonywanego na karcie graficznej. Tą część kodu można łatwo zrealizować na karcie graficznej. Zadaniem niebanalnym była natomiast implementacja metody do rozwiązania równania Poissona oraz równania dyfuzji. Można napisać program w taki sposób, aby jedynie obliczenia związane z cząstkami i węzłami siatki numerycznej były wykonywane na GPU. Następnie dane musiałyby być przesyłane do pamięci RAM gospodarza a równania Poissona i dyfuzji rozwiązywane były by na CPU. Schemat takich obliczeń przedstawiony jest na rys. 6.1. Niestety transfer danych pomiędzy CPU a GPU zabiera sporo czasu, a więc takie rozwiązanie jest nieefektywne. Lepszym rozwiązaniem jest przeniesienie całości obliczeń związanych z pętlą czasową na GPU (rys. 6.2). W takim przypadku dane są inicjalizowane na CPU i przesyłane jednokrotnie na GPU. Transfer w drugą stronę występuje tylko w momencie zapisu pośrednich wyników do pliku. Aby to było możliwe niezbędne jest zaimplementowanie na GPU programów rozwiązujących równania Poissona i równania dyfuzji.

Do rozwiązania równania Poissona w implementacji wykonującej obliczenia na CPU wykorzystywana była metoda szybkiego rozwiązywania równania Poissona (FPS - *ang.* Fast Poisson Solver) [95, 96]. Jest to metoda dokładna w tym sensie, że powstały w wyniku dyskretyzacji równania Poissona układ równań algebraicznych rozwiązywany jest dokładnie, a nie iteracyjnie. Niestety obliczenia wykonywane w tej metodzie są sekwencyjne i nie jest ona w stanie w pełni wyko-



Rysunek 6.1: Schemat metody VIC z przeniesieniem jednej operacji na GPU



Rysunek 6.2: Schemat metody VIC z przeniesieniem wszystkich obliczeń na GPU

rzystać potencjału architektury równoległej. Potrzebny był zatem inny algorytm. Do obliczeń wybrana została metoda wielosiatkowa (*ang.* Multigrid). Opis metody zamieszczono w rozdziale 6.2.3.

W implementacji na CPU równanie dyfuzji było rozwiązywane w sposób jawny. Aby poprawić stabilność metody do implementacji na GPU została wybrana metoda niejawna. W pierwszych testach do rozwiązywania powstałego układu równań algebraicznych wykorzystywano metodę wielosiatkową. Niestety, dla równania dyfuzji okazała się ona nieefektywna. Dlatego też zaimplementowana została metoda gradientów sprzężonych (CG - *ang.* Conjugent Gradient). Opis metody zamieszczono się w rozdziale 6.4.1

6.1. Operacje związane z ruchem cząstki i redystrybucją wirowości

Najprostszą do przeniesienia na kartę graficzną operacją w algorytmie VIC było przemieszczanie cząstek wirowych w zadanym polu prędkości. Jak wynika z przedstawionego opisu metody VIC, cząstki wirowe w płynie nielepkim (lub po zastosowaniu dekompozycji lepkościowej) przemieszczają się tak jak cząstki materialne płynu.

6.1.1. Implementacja ruchu cząstek na GPU

Pierwszą przeniesioną na kartę metodą przemieszczania cząstek na GPU była metoda Eulera. Zostanie ona teraz pokrótce przedstawiona. Jako język programowania zostało wybrane rozszerzenie języka C dla kart graficznych „C for CUDA".

Dla programu napisanego dla procesora sekwencyjnego obliczenia ruchu cząstek startujących z węzłów trójwymiarowej siatki strukturalnej można zapisać następująco:

```
for (int i=0; i < mesh_NX; i++)
    for (int j=0; j < mesh_NY; j++)
        for (int k=0; k < mesh_NZ; k++)
        {
            index = i + j*mesh_NX + k*mesh_NX*mesh_NY;
            particle_x[index] += DT * particle_velocity_x[index];
            particle_y[index] += DT * particle_velocity_y[index];
            particle_z[index] += DT * particle_velocity_z[index];
        }
```

gdzie `particle_x` jest wektorem zawierającym położenie cząstek a `particle_velocity_x` wektorem zawierającym ich prędkości.

Kod, który ma być wykonywany na karcie graficznej powinien znajdować się w tzw. jądrach (kernelach) tj. osobnych funkcjach poprzedzonych słowem kluczowym `__global__`. Funkcje te zawsze zwracają typ `void`. Należy pamiętać, że z punktu widzenia programisty kod jądra wykonywany jest przez wszystkie jednostki obliczeniowe (procesory strumieniowe) jednocześnie i należy go napisać tak, aby do każdego procesora trafiła odpowiednia porcja danych. Pojedynczy proces obliczeniowy przetwarzający kod kernela nazywany jest wątkiem. Ponieważ można zadać liczbę uruchamianych wątków wykonujących dane jądro, wystarczy więc zapisać w nim zawartość pętli przedstawionej powyżej:

```
__global__ void particle_movement(int NX, int NY, int NZ,
                                  double *d_particle_velocity_x,
                                  double *d_particle_velocity_y,
                                  double *d_particle_velocity_z,
                                  double *d_particle_x,
                                  double *d_particle_y,
                                  double *d_particle_z)
{
    int idx = blockIdx.x*blockDim.x+threadIdx.x;
    int idy = blockIdx.y*blockDim.y+threadIdx.y;
    int idz = blockIdx.z*blockDim.z+threadIdx.z;

    if( idx<NX && idy<NY && idz<NZ)
    {
        int index = idx + idy*NX + idz*NX*NY;
        d_particle_x[index] += DT * d_particle_velocity_x[index];
        d_particle_y[index] += DT * d_particle_velocity_y[index];
```

```
        d_particle_z[index] += DT * d_particle_velocity_z[index];
    }
    __syncthreads();
}
```

Wątki mogą być ustawiane w jedno-, dwu- lub trójwymiarową siatkę aby lepiej dopasować się do przetwarzanych danych. W powyższym przykładzie obliczenia prowadzone są na trójwymiarowej siatce strukturalnej i dlatego wątki także zostały ułożone w trójwymiarową strukturę. Szczegóły ustawienia i uruchomienia wątków podane są w dalszej części tego rozdziału. Siatkę wątków można podzielić na bloki. Każdy blok oraz każdy wątek w bloku ma swój unikalny numer dzięki czemu można określić pozycję danego wątku w siatce, a tym samym przyporządkować do niego odpowiednią porcję danych. Pozycja wątku w bloku zapisana jest w zmiennej `threadIdx`, która ma trzy elementy, dla każdego z kierunków (`x`,`y`,`z`), a pozycja bloku w siatce zapisana jest w zmiennej `blockIdx`. Wielkość bloku zapisana jest w zmiennej `blockDim`. Ponieważ wielkość struktury wątków jest wielokrotnością wielkości bloku to może się zdarzyć, że będzie ona większa niż siatka węzłów numerycznych w których należy prowadzić obliczenia. Zatem aby zapobiec dostępom poza zadany obszar pamięci badany jest warunek `if(idx<NX && idy<NY && idz<NZ)` sprawdzający czy dany wątek nie wychodzi poza siatkę numeryczną. Dalej obliczenia są prowadzone tak, jak wewnątrz pętli dla procesora sekwencyjnego. Aby wyróżnić obliczenia prowadzone na danych znajdujących się w pamięci karty graficznej nazwy tych danych oznaczone są przez przedrostek `d_`.

Tak przygotowane jądro należy uruchomić z wnętrza programu głównego. Aby można było to zrobić należy najpierw skopiować wszystkie potrzebne dane na kartę graficzną. Wykonywane jest to w dwóch etapach. Najpierw należy zaalokować potrzebną przestrzeń w pamięci GPU,

```
double *d_particle_velocity_x,
       *d_particle_velocity_y,
       *d_particle_velocity_z;

    cudaMalloc( (void **) &d_particle_velocity_x,
                sizeof(double)*(meshNX*meshNY*meshNZ) );
    cudaMalloc( (void **) &d_particle_velocity_y,
                sizeof(double)*(meshNX*meshNY*meshNZ) );
    cudaMalloc( (void **) &d_particle_velocity_z,
                sizeof(double)*(meshNX*meshNY*meshNZ) );

        double *d_particle_x, *d_particle_y, *d_particle_z;

    cudaMalloc( (void **) &d_particle_x,
                sizeof(double)*(meshNX*meshNY*meshNZ) );
    cudaMalloc( (void **) &d_particle_y,
                sizeof(double)*(meshNX*meshNY*meshNZ) );
    cudaMalloc( (void **) &d_particle_z,
```

```
                sizeof(double)*(meshNX*meshNY*meshNZ) );
```

a następnie przesłać dane przygotowane na CPU

```
cudaMemcpy(d_particle_velocity_x, x_zero,
           sizeof(double)*(meshNX*meshNY*meshNZ),
           cudaMemcpyHostToDevice);
cudaMemcpy(d_particle_velocity_y, x_zero,
           sizeof(double)*(meshNX*meshNY*meshNZ),
           cudaMemcpyHostToDevice);
cudaMemcpy(d_particle_velocity_z, x_zero,
           sizeof(double)*(meshNX*meshNY*meshNZ),
           cudaMemcpyHostToDevice);

cudaMemcpy(d_particle_x, particle_x,
           sizeof(double)*(meshNX*meshNY*meshNZ),
           cudaMemcpyHostToDevice);
cudaMemcpy(d_particle_y, particle_y,
           sizeof(double)*(meshNX*meshNY*meshNZ),
           cudaMemcpyHostToDevice);
cudaMemcpy(d_particle_z, particle_z,
           sizeof(double)*(meshNX*meshNY*meshNZ),
           cudaMemcpyHostToDevice);
```

Uruchomienie obliczeń wykonywane jest poleceniem

```
particle_movement<<<dimGrid, dimBlock>>>(meshNX, meshNY, meshNZ,
                                         d_particle_velocity_x,
                                         d_particle_velocity_y,
                                         d_particle_velocity_z,
                                         d_particle_x,
                                         d_particle_y,
                                         d_particle_z);
```

Wywołanie to różni się od wywołania zwykłej funkcji napisanej w języku C przez dodanie w potrójnych nawiasach parametrów uruchomienia. Zmienne `dimGrid` i `dimBlock` określają odpowiednio liczbę bloków w siatce oraz liczbę wątków w bloku i muszą zostać zdefiniowane wcześniej w kodzie. Przykładowo:

```
dim3 dimBlock;
dim3 dimGrid;

int block_size_x = 8;
int block_size_y = 8;
int block_size_z = 8;

dimBlock.x = block_size_x;
dimBlock.y = block_size_y;
dimBlock.z = block_size_z;

dimGrid.x = meshNX / block_size_x;
dimGrid.y = meshNY / block_size_y;
```

```
dimGrid.z = meshNZ / block_size_z;

if (meshNX % block_size_x !=0 ) dimGrid.x+=1;
if (meshNY % block_size_y !=0 ) dimGrid.y+=1;
if (meshNZ % block_size_z !=0 ) dimGrid.y+=1;
```

Taki zapis oznacza przygotowanie trójwymiarowej struktury wątków obliczeniowych, której bloki będą miały wymiar (`dimBlock`) $8 \times 8 \times 8$. Liczba bloków w każdym kierunku siatki wątków (`dimGrid`) jest wyznaczana jako iloraz rozmiaru siatki i liczby bloków. Jeżeli rozmiar siatki numerycznej nie jest wielokrotnością rozmiaru bloku to pozostają wątki, które nie należą do żadnego bloku. Dlatego należy w danym kierunku dodać jeszcze jeden blok.

Do kompilacji kodu napisanego w języku „C for CUDA" używany jest specjalny kompilator o nazwie `nvcc` dostępny do pobrania za darmo ze strony `http://www.nvidia.com/cuda`. Najprostszy program kompiluje się poleceniem [3]

```
nvcc czastki.cu -o czastki
```

W wyniku kompilacji powstaje plik wykonywalny, który po uruchomieniu przeprowadzi obliczenia numeryczne na karcie graficznej.

6.2. Program do rozwiązywania równania Poissona

Równanie Poissona ma postać:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} + \frac{\partial^2 \psi}{\partial z^2} = -f(x, y, z), \tag{6.1}$$

gdzie $\psi(x, y, z)$ jest nieznanym polem skalarnym, a $f(x, y, z)$ funkcją prawej strony.

Siedmiopunktowa dyskretyzacja równania Poissona na siatce numerycznej (ih_x, jh_y, kh_z), zakładając dodatkowo równe kroki siatki we wszystkich kierunkach ($h_x = h_y = h_z = h$), będzie miała ostateczną postać:

$$\psi_{i,j,k+1} + \psi_{i,j+1,k} + \psi_{i+1,j,k} - 6\psi_{i,j,k} + \psi_{i-1,j,k} + \psi_{i,j-1,k} + \psi_{i,j,k-1} = f_{i,j,k}h^2, \tag{6.2}$$

gdzie indeks i odnosi się do osi x ($x_i = ih$), $i = 0, 1, 2, ..., N_x$, indeks j do osi y ($y_j = jh$), $j = 0, 1, 2, ..., N_y$ a indeks k do osi z ($z_k = kh$), $k = 0, 1, 2, ..., N_z$. Otrzymywany układ równań algebraicznych dla wartości w węzłach siatki numerycznej będzie rozwiązywany metodami iteracyjnymi. Zapis macierzowy tego układu jest postaci

$$A\psi = f, \tag{6.3}$$

gdzie A jest macierzą rzadką, siedmioprzekątniową zawierającą współczynniki dyskretyzacji.

6.2.1. Metody iteracyjne

Układ równań algebraicznych postały po dyskretyzacji równania Poissona może być rozwiązywany metodami dokładnymi (takimi jak FPS) lub iteracyjnymi. Obliczenia w metodach dokładnych są sekwencyjne i nie są one w stanie w pełni wykorzystać możliwości jakie daje karta graficzna. Dlatego do pracy została użyta metoda wielosiatkowa. Łączy ona zarówno dobrą zbieżność obliczeń jak i możliwość efektywnego zrównoleglenia. W schemacie metody wielosiatkowej wykorzystywane są prostsze metody iteracyjne takie, jak metoda Jacobiego czy metoda Gaussa-Seidla.

Metoda Jacobiego

W metodzie Jacobiego proces iteracyjny przebiega następująco [12]:

$$\psi_{i,j,k}^{(n)} = \frac{1}{6}\left(f_{i,j,k}h^2 + \psi_{i,j,k+1}^{(n-1)} + \psi_{i,j+1,k}^{(n-1)} + \psi_{i+1,j,k}^{(n-1)} + \psi_{i-1,j,k}^{(n-1)} + \psi_{i,j-1,k}^{(n-1)} + \psi_{i,j,k-1}^{(n-1)}\right), \tag{6.4}$$

gdzie n jest numerem iteracji. Zdefiniujmy błąd rozwiązania jako:

$$e_{i,j,k}^{(n)} = u_{i,j,k} - \psi_{i,j,k}^{(n)}, \tag{6.5}$$

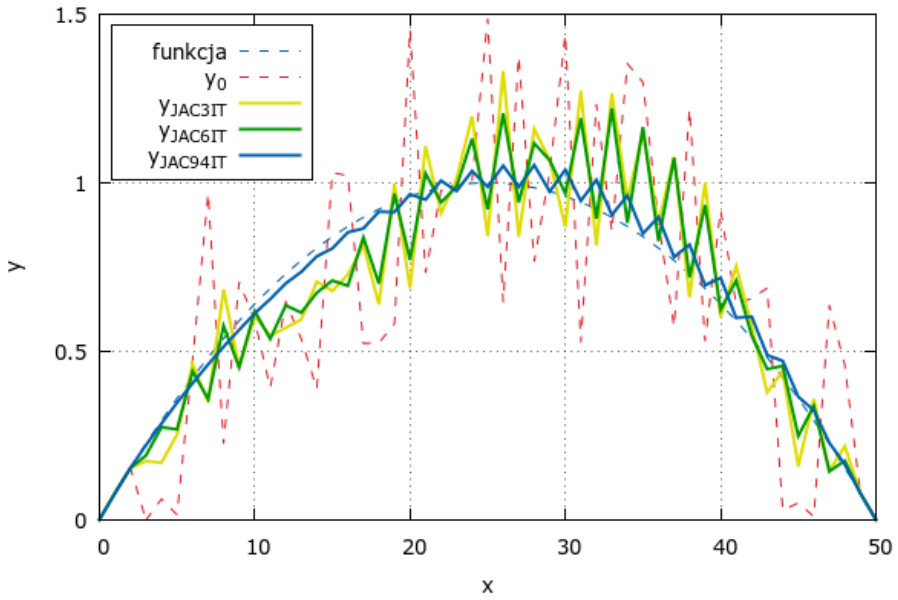
gdzie u jest rozwiązaniem dokładnym. Możemy przekształcić wzór (6.4) do postaci:

$$e_{i,j,k}^{(n)} = \frac{1}{6}\left(e_{i,j,k+1}^{(n-1)} + e_{i,j+1,k}^{(n-1)} + e_{i+1,j,k}^{(n-1)} + e_{i-1,j,k}^{(n-1)} + e_{i,j-1,k}^{(n-1)} + e_{i,j,k-1}^{(n-1)}\right). \tag{6.6}$$

Widać zatem, że błąd w każdej kolejnej iteracji jest uśrednieniem wartości błędów w węzłach sąsiednich z poprzedniej iteracji. Skutkiem tego następować będzie wygładzanie błędu. Stąd właśnie inna nazwa tych metod, używana w kontekście metody wielosiatkowej, metody wygładzające błąd (*ang.* smoothers). Przykład iteracji metodą Jacobiego dla jednowymiarowej funkcji znajduje się na rys. 6.3.

Metoda Gaussa-Seidela z czerwono-czarnym numerowaniem węzłów

Kolejną metodą iteracyjną do rozwiązywania układów równań liniowych jest metoda Gaussa-Seidla [12, 97]. W algorytmie sekwencyjnym obliczenia dla węzłów siatki mogą być wykonywane w kolejności leksykograficznej przedstawionej na rys. 6.4. Można zauważyć, że dla wybranego węzła istnieją już nowe wartości w węzłach o niższych numerach. Dla przykładu przy wyznaczaniu nowej wartości w węźle 13 korzystamy z wartości niewiadomych z węzłów 12 i 8, dla których istnieją już nowe wartości. Można wykorzystać ten fakt i przepisać równanie (6.4) w postaci:



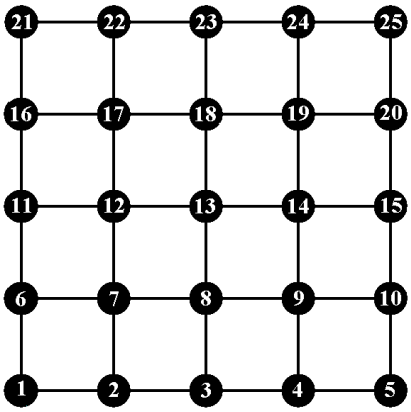
Rysunek 6.3: Wygładzanie metodą Jacobiego dla jednowymiarowego równania Poissona. Niebieską linią przerywaną zaznaczone jest rozwiązanie, czerwoną linią przerywaną losowe przybliżenie początkowe. Liniami ciągłymi zaznaczone są rozwiązania otrzymane po 3 iteracjach (żółty), 6 iteracjach (zielony) i 94 iteracjach (niebieski).

$$\psi_{i,j,k}^{(n)} = \frac{1}{6}\left(f_{i,j,k}h^2 + \psi_{i,j,k+1}^{(n-1)} + \psi_{i,j+1,k}^{(n-1)} + \psi_{i+1,j,k}^{(n-1)} + \psi_{i-1,j,k}^{(n)} + \psi_{i,j-1,k}^{(n)} + \psi_{i,j,k-1}^{(n)}\right). \tag{6.7}$$

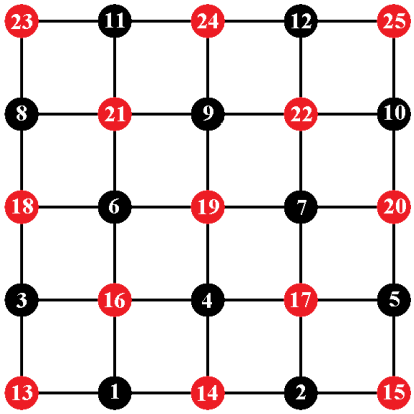
Dzięki temu metoda Gaussa-Seidla wymaga mniejszej liczby iteracji do osiągnięcia zadanej zbieżności niż wcześniej przedstawiona metoda Jacobiego. Niestety w postaci (6.7) nie da się jej wykorzystać w obliczeniach równoległych ponieważ obliczenia dla poszczególnych węzłów siatki (zmiennych) nie będą niezależne.

Dla siatek strukturalnych możliwe jest rozbicie obliczeń na dwa etapy. Siatka obliczeniowa zostaje podzielona na dwie części tak, jak to jest przedstawione na rys. 6.5. W pierwszym etapie obliczenia prowadzone są jedynie dla węzłów przestawionych na rysunku kolorem czarnym. Jak widać, do obliczeń są potrzebne jedynie wartości niewiadomych z czerwonych węzłów. Dzięki temu wszystkie obliczenia są niezależne i można je wykonać równolegle. W drugim etapie obliczenia prowadzone są tylko dla węzłów oznaczonych czerwonym kolorem. Równanie (6.7) możemy przepisać do postaci:

$$\psi_{i,j,k}^{B(n)} = \frac{1}{6}\left(f_{i,j,k}h^2 + \psi_{i,j,k+1}^{R(n-1)} + \psi_{i,j+1,k}^{R(n-1)} + \psi_{i+1,j,k}^{R(n-1)} + \psi_{i-1,j,k}^{R(n-1)} + \psi_{i,j-1,k}^{R(n-1)} + \psi_{i,j,k-1}^{R(n-1)}\right), \tag{6.8}$$



Rysunek 6.4: Leksykograficzna numeracja węzłów.



Rysunek 6.5: Czerwono-czarna numeracja węzłów.

$$\psi_{i,j,k}^{R(n)} = \frac{1}{6}\left(f_{i,j,k}h^2 + \psi_{i,j,k+1}^{B(n)} + \psi_{i,j+1,k}^{B(n)} + \psi_{i+1,j,k}^{B(n)} + \psi_{i-1,j,k}^{B(n)} + \psi_{i,j-1,k}^{B(n)} + \psi_{i,j,k-1}^{B(n)}\right). \tag{6.9}$$

gdzie indeks R odnosi się do węzłów zaznaczonych na czerwono, natomiast B do węzłów „czarnych".

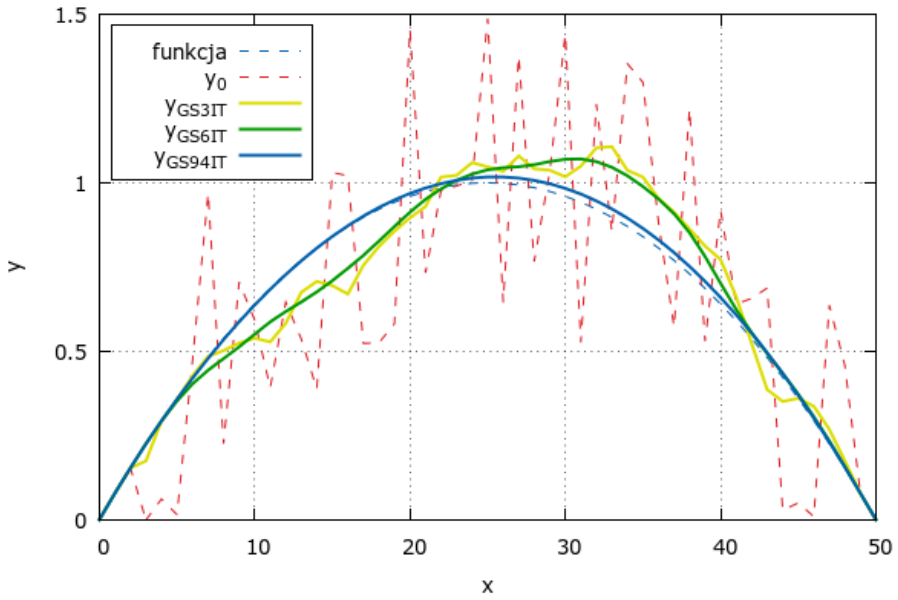
Przykład iteracji metodą Gaussa-Seidla dla jednowymiarowej funkcji znajduje się na rys. 6.6. Już po pierwszych 3 iteracjach rozwiązanie jest znacząco mniej nieregularne (poszarpane) niż przybliżenie początkowe. Porównując wyniki do metody Jacobiego (rys. 6.3) widać szybszą zbieżność metody Gaussa-Seidla oraz mocniejszy efekt wygładzania. Zależność liczby potrzebnych iteracji do osiągnięcia zadanej dokładności obliczanej jako największy co do wartości bezwzględnej element wektora residuów

$$max|r_i| \qquad i = 1, 2, ..., N, \tag{6.10}$$

gdzie N jest liczbą węzłów dla przypadków przedstawionych na rys. 6.3 i rys. 6.6 przedstawiona jest w Tabeli 6.1.

Tabela 6.1: Liczba iteracji metodami Jacobiego i Gaussa-Seidla potrzebna do osiągnięcia zadanej dokładności.

Dokładność	Metoda Jacobiego	Metoda Gaussa-Seidla
10^{-1}	94	3
10^{-2}	936	6
10^{-3}	1996	26
10^{-4}	2996	145
10^{-5}	4026	439



Rysunek 6.6: Wygładzanie metodą Gaussa-Seidla dla jednowymiarowego równania Poissona. Niebieską linią przerywaną zaznaczone jest rozwiązanie, czerwoną linią przerywaną losowe przybliżenie początkowe. Liniami ciągłymi zaznaczone są rozwiązania otrzymane po 3 iteracjach (żółty), 6 iteracjach (zielony) i 94 iteracjach (niebieski).

SOR

Wprowadźmy oznaczenie dla wyników otrzymanych po dwóch krokach metody Gaussa-Seidla z czerwono-czarną numeracją węzłów:

$$g^{(n)}_{i,j,k} = \psi^{(n)}_{i,j,k}. \tag{6.11}$$

Aby poprawić własności wygładzające podanych metod możemy wprowadzić dodatkowo współczynnik relaksacyjny ω. Wynik jednej iteracji ma wtedy postać

$$\psi^{(n)}_{i,j,k} = \psi^{(n-1)}_{i,j,k} + \omega \left[g^{(n)}_{i,j,k} - \psi^{(n-1)}_{i,j,k} \right]. \tag{6.12}$$

Właściwie dobrana wartość ω może poprawić własności wygładzające metod iteracynych [98]. Zachowanie błędu charakteryzują wartości i funkcje własne macierzy (6.3). W celu lepszego zobrazowania własności wygładzających algorytmu iteracyjnego rozważania zostaną podane dla przypadku jednowymiarowego, $-\frac{d^2\psi}{dx^2} = 0$ w przedziale $(0, 1)$ z podziałem jednorodnym na N punktów. Dla tego przypadku funkcje i wartości własne mają postać [98]:

$$w_{l,i} = sin(l\pi x_i), \quad l = 1, ..., N-1, \tag{6.13}$$

$$\lambda_l(\omega) = 1 - \omega sin^2(l\pi h). \tag{6.14}$$

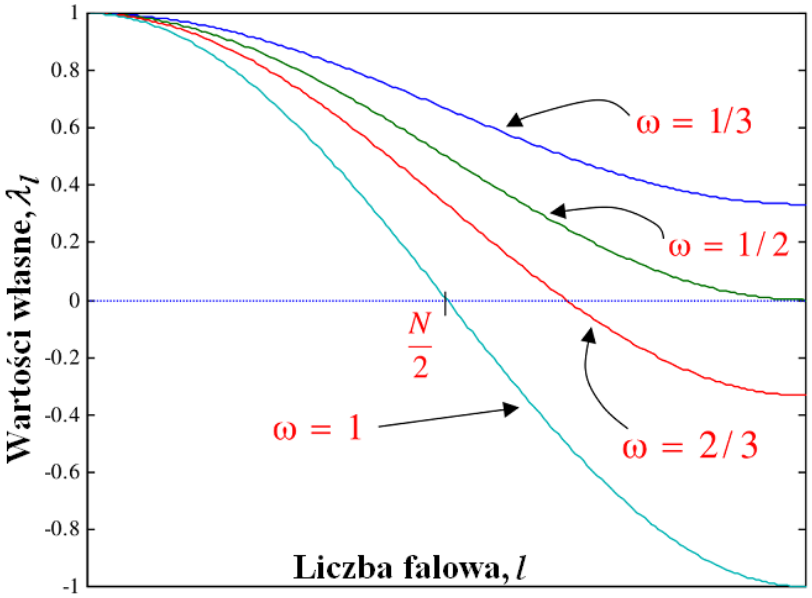
Błąd rozwiązania można przedstawić w postaci szeregu Fouriera przy pomocy wektorów własnych [98]:

$$e = \sum_{l=1}^{N-1} \alpha_l w_l. \tag{6.15}$$

Po jednej iteracji metodą Jacobiego bądź Gaussa-Seidla błąd będzie miał postać [98]:

$$\bar{e} = \sum_{l=1}^{N-1} \lambda_l \alpha_l w_l. \tag{6.16}$$

Zatem wartości własne możemy traktować jak współczynnik tłumienia. Na rysunku 6.7 przedstawione są wartości własne w zależności od liczby falowej l sparametryzowane wartościami ω. Widać, że w zależności od wybranego współczynnika relaksacji ω otrzymujemy różne wartości własne. Widać także, że we wszystkich przypadkach najmocniej tłumione będą składowe szybkozmienne (duże wartości l), a najsłabiej – wolnozmienne (małe wartości l)



Rysunek 6.7: Wartości własne w funkcji liczby falowej [1]

6.2.2. Metoda dwusiatkowa

Podstawowa idea metody wielosiatkowej wynika z obserwacji, że dla metody Jacobiego czy Gaussa-Seidla szybkozmienne składowe błędu są szybciej redukowa-

ne niż składowe wolnozmienne. Należy zauważyć, że jeżeli zastosujemy rzadszą siatkę, to na niej szybciej będą niwelowane składowe błędu o niższych częstotliwościach niż to miało miejsce na siatce gęstej. Rozważmy zatem zagadnienie brzegowe dla dyskretnego równania eliptycznego w obszarze Ω_h postaci [98]:

$$L_h u_h = f_h, \qquad (\Omega_h). \tag{6.17}$$

gdzie u_h jest nieznanym rozwiązaniem tego układu równań. Wprowadźmy wektor błędu $(e_h^{(n)})$ i wektor residuów $r_h^{(n)}$ dla pewnego przybliżenia rozwiązania $u_h^{(n)}$

$$e_h^{(n)} = u_h - u_h^{(n)}, \tag{6.18}$$

$$r_h^{(n)} = f_h - L_h u_h^{(n)}. \tag{6.19}$$

Wstawiając (6.18) i (6.19) do (6.17) otrzymujemy:

$$L_h e_h^{(n)} = r_h^{(n)}. \tag{6.20}$$

To równanie można rozwiązać a następnie wyznaczony błąd $e_h^{(n)}$ dodać do rozwiązania równania (6.17). W metodzie dwusiatkowej do znalezienia poprawki wykorzystywana jest rzadsza siatka (rys. 6.8). Schemat obliczeń jest następujący [98]:

1. Najpierw należy wykonać pewną zadaną liczbę iteracji ν_1 wybraną metodą wygładzającą błąd (Jacobi, Gauss-Seidel, SOR). Otrzymany w ten sposób wektor rozwiązania można zapisać następująco

   $$\hat{u}_h^{(n)} = \Phi^{\nu_1}\left(u_h^{(n)}, L_h, f_h\right), \tag{6.21}$$

   gdzie Φ^{ν_1} oznacza wykonanie ν_1 iteracji metodą wygładzającą błąd.

2. Obliczane są residua ze wzoru (6.19)

   $$\hat{r}_h^{(n)} = f_h - L_h \hat{u}_h^{(n)}. \tag{6.22}$$

3. Wartości wektora residuów rzutowane są na rzadszą siatkę za pomocą operatora I_h^H:

   $$\hat{r}_H^{(n)} = I_h^H \hat{r}_h^{(n)}. \tag{6.23}$$

4. Na rzadszej siatce dokładnie rozwiązywane jest równanie (6.20)

   $$L_H \hat{e}_H^{(n)} = r_H^{(n)}. \tag{6.24}$$

5. Otrzymane rozwiązanie interpolowane jest na gęstą siatkę za pomocą operatora I_H^h

$$\hat{e}_h^{(n)} = I_H^h \hat{e}_H^{(n)}. \tag{6.25}$$

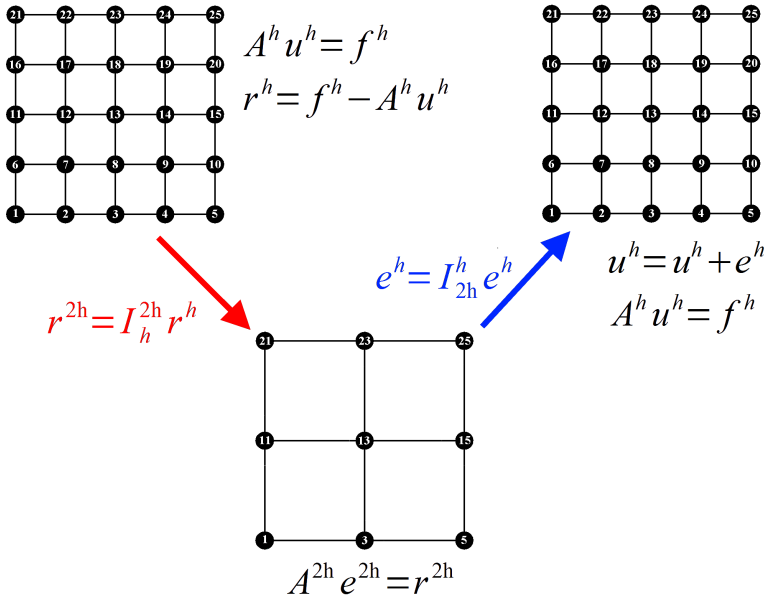
I dodawane jest jako poprawka do wektora niewiadomych $\hat{u}_h^{(n)}$

$$\bar{u}_h^{(n)} = \hat{u}_h^{(n)} + \hat{e}_h^{(n)}. \tag{6.26}$$

6. Na koniec wykonywane jest ponowne wygładzanie błędu wybraną metodą

$$u_h^{(n+1)} = \Phi^{\nu_2}\left(\bar{u}_h^{(n)}, L_h, f_h\right). \tag{6.27}$$

W ten sposób kończy się pojedyncza iteracja metody dwusiatkowej. Dokładne rozwiązanie na rzadszej siatce pozwoliło na wyeliminowanie składowych błędu, które są wolno zbieżne przy wykorzystaniu metod wygładzających na siatce gęstej.



Rysunek 6.8: Idea metody dwusiatkowej

6.2.3. Metoda wielosiatkowa (Multigrid method)

Metoda wielosiatkowa stanowi rozwinięcie metody dwusiatkowej. Zamiast rozwiązywać dokładnie zagadnienia na błąd na siatce rzadszej $e_h^{(n)}$ wywołuje się rekurencyjnie schemat wygładzania wektora niewiadomych i przejścia na coraz

rzadsze siatki. Powtarzane jest to do momentu osiągnięcia najrzadszej, założonej siatki, na której otrzymane zagadnienie rozwiązywane jest dokładnie.

Cykl V

Schemat cyklu V przedstawiony jest na rys. 6.9. Obliczenia rozpoczynają się z poziomu najgęstszej siatki. Na tym poziomie wykonuje się zadaną liczbę iteracji wybraną metodą relaksacyjną (Jacobi, Gauss-Seidel, SOR). Następnie wyliczany jest wektor residuów. Wektor ten posłuży za wektor prawych stron w rozwiązywaniu zagadnienia na rzadszej siatce. Zostaje on przeniesiony na węzły rzadszej siatki. Przykładowymi metodami używanymi w tym etapie są [98]

- metoda wstrzyknięcia (*ang.* injection), polegająca na przypisaniu węzłom rzadszej siatki wartości w odpowiadającym im węzłom gęstszej siatki,
- metoda pełnego ważenia (FW - *ang.* full-weighting), przypisująca węzłom siatki rzadkiej średnią ważoną z wartości w węzłach odpowiadających i sąsiednich na siatce gęstej z wagami odpowiednio

$$I_h^{2h} = [1/4 \quad 1/2 \quad 1/4], \tag{6.28}$$

dla przypadku dwuwymiarowego operator ten ma postać

$$I_h^{2h} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \tag{6.29}$$

a dla trójwymiarowego:

$$I_h^{2h} = \frac{1}{64}\begin{bmatrix} 1 & 2 & 1 & & & & & & \\ 2 & 4 & 2 & & & & & & \\ 1 & 2 & 1 & & & & & & \\ & & & 2 & 4 & 2 & & & \\ & & & 4 & 8 & 4 & & & \\ & & & 2 & 4 & 2 & & & \\ & & & & & & 1 & 2 & 1 \\ & & & & & & 2 & 4 & 2 \\ & & & & & & 1 & 2 & 1 \end{bmatrix}. \tag{6.30}$$

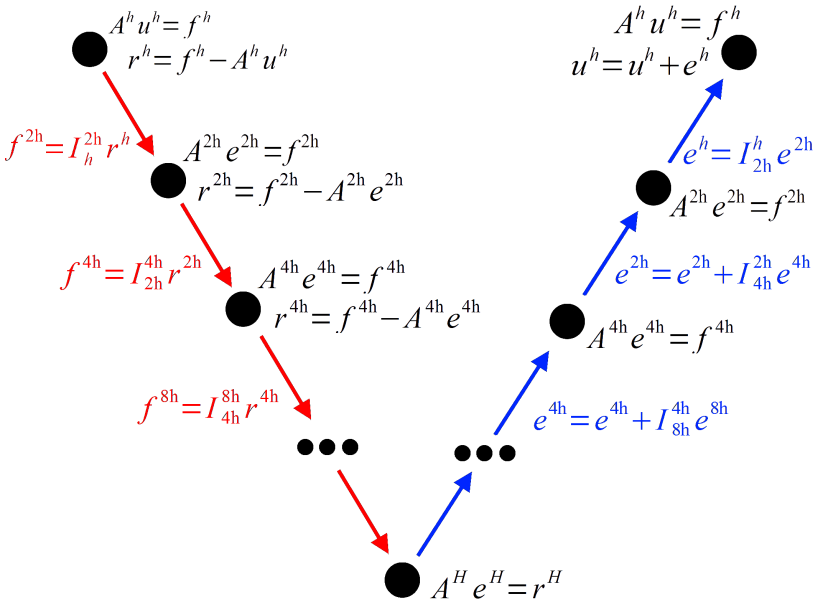
- metoda połówkowego ważenia (HW - *ang.* half-weighting), dla której dwuwymiarowa postać operatora przedstawia się następująco:

$$I_h^{2h} = \frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{6.31}$$

a trójwymiarowa następująco:

$$I_h^{2h} = \frac{1}{12}\begin{bmatrix} & & & & & & & & \\ & 1 & & & & & & & \\ & & & 1 & & & & & \\ & & & & 6 & & & & \\ & & & & 1 & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & 1 & \\ & & & & & & & & \end{bmatrix}. \tag{6.32}$$

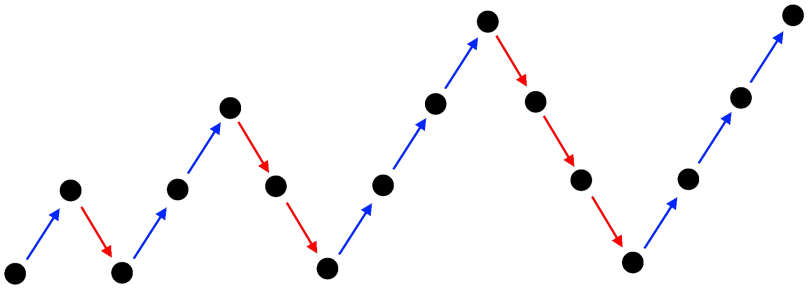
Na niższym poziomie za rozwiązanie początkowe przyjmuje się wektor zerowy. Ponownie stosuje się zadaną liczbę iteracji wybraną metodą iteracyjną, wylicza residua i przechodzi poziom niżej. Schemat ten powtarzany jest do osiągnięcia najniższego zadanego poziomu, gdzie otrzymane zagadnienie rozwiązywane jest dokładnie dowolną metodą. Otrzymane wyniki interpolowane są na gęstszą siatkę wyższego poziomu i jako poprawka rozwiązania dodawane są do aktualnego wektora niewiadomych na danym poziomie. Na tym poziomie ponownie wykonywana jest zadana liczba iteracji wybraną metodą relaksacyjną. Otrzymane rozwiązanie interpolowane jest na wyższy poziom i dodawane jako poprawka. Proces ten powtarzany jest do momentu dojścia do najwyższego poziomu (oznaczonego jako H na rys. 6.9). W ten sposób kończy się jeden cykl V. Proces ten powtarzany jest iteracyjnie do uzyskania zadanej dokładności rozwiązania.



Rysunek 6.9: Schemat cyklu V dla metody wielosiatkowej

Cykl FMG (Full multigrid method) - Pełna metoda wielosiatkowa

Największym problemem wykonywania obliczeń cyklem V jest brak odpowiedniego przybliżenia początkowego. Ten problem zostaje rozwiązany w cyklu FMG, czyli tzw. pełnej metodzie wielosiatkowej. Obliczenia rozpoczynane są na najniższym poziomie od dokładnego rozwiązania zagadnienia na najrzadszej siatce. Wynik ten jest interpolowany na gęstszą siatkę i rozpoczynany jest cykl V dla tego poziomu. Otrzymany wynik ponownie jest interpolowany na gęstszą siatkę i rozpoczynany jest cykl V. Proces ten jest powtarzany do osiągnięcia najgęstszej siatki. Osiągnięte w ten sposób rozwiązanie stanowi przybliżenie początkowe do pełnego cyklu V. Jedna iteracja cyklu FMG daje dokładność obliczeń na poziomie błędu dyskretyzacji zagadnienia [98]. W większości zastosowań cykl FMG jest najbardziej efektywną formą metody wielosiatkowej [98]. Cykl FMG został przedstawiony na rys. 6.10



Rysunek 6.10: Schemat pełnej metody wielosiatkowej

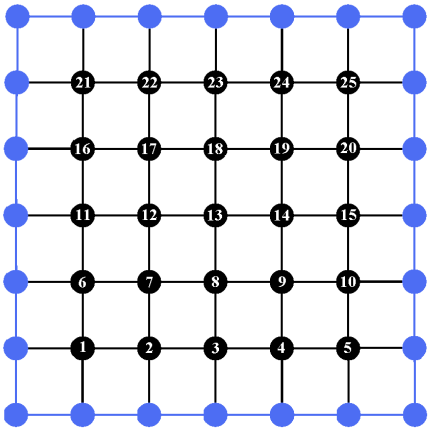
6.2.4. Badania numeryczne implementacji metod iteracyjnych na karcie graficznej

Metoda Jacobiego

Zadaniem testowym było trójwymiarowe równanie Poissona, ze znanym rozwiązaniem:

$$\psi(x,y,z) = 100xyz(x-1)(y-1)(z-1), \quad x,y,z \in [0,1] \tag{6.33}$$

Testowane były różne rodzaje pamięci (pamięć współdzielona i pamięć tekstur) oraz rozszerzona siatka numeryczna (rys. 6.11) pozwalająca na wyeliminowanie z kodu jądra instrukcji warunkowych. Pozwala to na przyspieszenie obliczeń ale jednocześnie zwiększa ilości potrzebnej pamięci. Parametry siatki i otrzymanych przyspieszeń względem obliczeń na CPU są przedstawione w Tabeli 6.2. W każdym teście wykonano 100 iteracji metodą Jacobiego. Obliczenia były prowadzone na: CPU (Intel Core 2 Quad Q9550), TESLA GPU (NVIDIA TESLA S1070) i FERMI GPU (NVIDIA GFX 480).



Rysunek 6.11: Powiększona siatka numeryczna pozwalająca na eliminację z kodu jądra instrukcji warunkowych (opis w sekcji 5.2) poprzez umieszczenie wartości brzegowych

Tabela 6.2: Osiągnięte przyspieszenie dla metody Jacobiego.

Rozmiar siatki	TESLA pamięć współdzielona	TESLA pamięć tekstur	TESLA bez instrukcji warunkowych	FERMI bez instrukcji warunkowych
32x32x32	4.05	6.61	6.94	12.32
64x64x64	17.71	26.32	31.26	52.82
128x128x128	24.78	29.95	43.67	58.89
256x256x256	25.47	24.59	41.92	60.96

Przedstawione wyniki pokazują, że przyspieszenie obliczeń zależy zarówno od wykorzystanego typu pamięci, jak i wielkości obszaru obliczeniowego. Uzyskane przyspieszenia na poziomie $50 - 60$ są potwierdzeniem dużych możliwości kart graficznych i uzasadniają dalszą pracę podjętą przy implementacji metody VIC na GPU.

Testy metody wraz z uzyskanymi przyspieszeniami zostały opublikowane w artykułach autorskich [58, 52, 51, 54].

Metoda Gaussa-Seidla z czerwono-czarną numeracją węzłów

Rozwiązanie równania Poissona (6.33) posłużyła również do przeprowadzenia testu dla metody Gaussa-Seidla z czerwono-czarną numeracją węzłów. Wykorzystano w nim siatkę taką, aby nie wykonywać operacji warunkowych na GPU (rys. 6.11). Parametry siatki oraz otrzymane wartości przyspieszeń podane są w Tabeli 6.3. W każdym teście wykonano 100 iteracji metodą Gaussa-Seidla. Obliczenia były prowadzone na: CPU (Intel Core 2 Quad Q9550), TESLA GPU (NVIDIA

TESLA S1070) i FERMI GPU (NVIDIA GFX 480).

Tabela 6.3: Osiągnięte przyspieszenia dla metody Gaussa-Seidla z czerwono-czarną numeracją węzłów.

Rozmiar siatki	TESLA bez instrukcji warunkowych	FERMI bez instrukcji warunkowych
32x32x32	11.17	45.06
64x64x64	26.11	63.01
128x128x128	35.95	88.62
256x256x256	31.22	89.47

Uzyskano przyspieszenia na poziome $80 - 90$ razy. Test potwierdził wnioski z wcześniejszych badań.

Testy metody wraz z uzyskanymi przyspieszeniami zostały opublikowane w artykułach autorskich [58, 52, 51, 54].

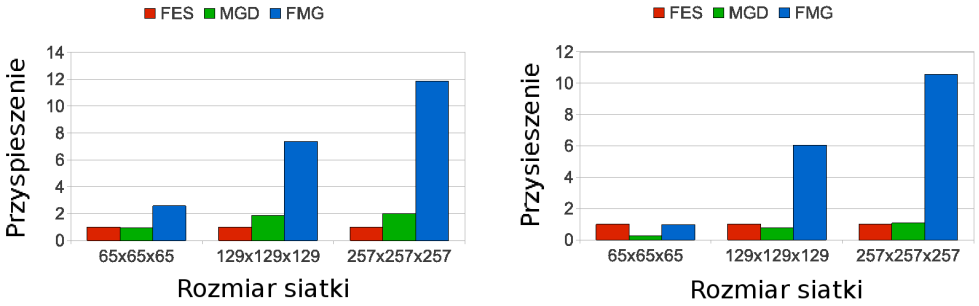
Metoda wielosiatkowa

Testy zostały przeprowadzone dla trójwymiarowego równania Poissona, ze znanym rozwiązaniem:

$$\psi(x,y,z) = sin(2\pi x) \cdot sin(2\pi y) \cdot sin(2\pi z), \qquad x,y,z \in [0,1]. \tag{6.34}$$

Z dwoma rodzajami warunków brzegowych. Badane były warunki brzegowe typu Dirichleta oraz okresowe. Przygotowane siatki miały $2^n + 1$ węzłów w każdym kierunku, gdzie n było numerem poziomu. Jako metoda wygładzania została wykorzystana metoda SOR Gaussa-Seidla z czerwono-czarnym numerowaniem węzłów oraz $\omega = 1.15$ [98]. Przeniesienie wartości na niższe poziomy siatki było realizowane przy pomocy metody „pełnego ważenia" (6.28). Rozwiązanie na najniższym poziomie (siatka o liczbie węzłów $3 \times 3 \times 3$) było otrzymywane iteracyjnie metodą Gauss-Seidla. Maksymalna wartość residuów dla rozwiązania na tym poziomie siatki została przyjęta jako 10^{-4}. Przeniesienie wartości na siatkę gęstszą było realizowane za pomocą interpolacji liniowej.

Dla obydwu cykli przetestowane zostały różne konfiguracje liczby iteracji wygładzania błędu. Dla cyklu V do testów wybrana została konfiguracja z trzema iteracjami metodą RBGS przy przechodzeniu na rzadszą siatkę ($\nu_1 = 3$) i czterema iteracjami metodą RBGS przy przechodzeniu na gęstszą siatkę ($\nu_2 = 4$). Konfiguracja taka będzie oznaczana jako V(3,4). Zatrzymywanie obliczeń następowało w momencie gdy maksymalna wartość residuów dla najgęstszej siatki nie przekraczała wartości 10^{-8}. W cyklu FMG wykonywana była jedna iteracja z konfiguracją FMG(8,0). Wyniki otrzymanych przyspieszeń względem metody



Rysunek 6.12: Przyspieszenie metody wielosiatkowej (MGD) i pełnej metody wielosiatkowej (FMG) względem szybkiej metody rozwiązywania równania Poissona (FES) dla warunków brzegowych typu Dirichleta (lewa strona) i okresowych warunków brzegowych (prawa strona). Warunkiem zatrzymania obliczeń było dla metody wielosiatkowej uzyskanie maksymalnej wartości residuów dla najgęstszej siatki mniejszej niż 10^{-8}. W cyklu FMG wykonywana była jedna iteracja.

Fast Elliptic Solver z biblioteki Fishpack [96] działającej na jednym rdzeniu znajdują się na rys. 6.12. W Tabeli 6.4 i Tabeli 6.5 umieszczone zostały informacje o osiągniętych dokładnościach poszczególnych metod. Obliczenia były prowadzone na: CPU (Intel Core i7 960) oraz GPU (NVIDIA GTX 480).

Tabela 6.4: Błąd numeryczny otrzymany dla różnych metod dla przypadku z warunkami brzegowymi typu Dirichleta.

Rozmiar siatki	FES(CPU)	MGD(GPU)	FMG(GPU)
65×65×65	8.0358e-4	8.0358e-4	8.1833e-4
129×129×129	2.0082e-4	2.0082e-4	2.0374e-4
257×257×257	5.0201e-5	5.0201e-5	5.0757e-5

Tabela 6.5: Błąd numeryczny otrzymany dla różnych metod dla przypadku z okresowymi warunkami brzegowymi.

Rozmiar siatki	FES(CPU)	MGD(GPU)	FMG(GPU)
65×65×65	8.0358e-4	8.0358e-4	8.2968e-4
129×129×129	2.0082e-4	2.0082e-4	2.0546e-4
257×257×257	5.0201e-5	5.0201e-5	5.0835e-5

Skróty użyte w Tabeli 6.4 i Tabeli 6.5 oraz rys. 6.12: FES - Fast Elliptic Solver, MGD - metoda wielosiatkowa, FMG - pełna metoda wielosiatkowa.

Przedstawione dokładności numeryczne pokazują zarówno poprawność implementacji metody na karcie graficznej, jak i jej przydatność do rozwiązania tego zagadnienia. Na największej siatce numerycznej dla cyklu V uzyskano ok. dwu-

krotne przyspieszenie obliczeń względem metody FES. Nie jest to znacząca wartość ale pozwala na przeniesienie całego algorytmu VIC na GPU i z tego powodu jest przydatna. Dużo lepiej w tym świetle wypada cykl FMG. Przyspieszenie na poziomie $10 - 12$ jest zadowalające i pozwala na znaczące przyspieszenie całego algorytmu w metodzie cząstek wirowych VIC na GPU.

Testy metody wraz z uzyskanymi przyspieszeniami zostały opublikowane w artykułach autorskich [49, 54].

6.3. Badania numeryczne implementacji metody VIC na GPU dla płynu nielepkiego

Po opracowaniu programu realizującego metodę wielosiatkową na GPU oraz zaimplementowania wszystkich elementów metody VIC cały kod mógł być uruchomiony na GPU bez potrzeby ciągłego przesyłania danych pomiędzy CPU a GPU według schematu pokazanego na rys. 6.2. Dane początkowe były przygotowywane na CPU i przesyłane na GPU. Od tego momentu wszystkie operacje były wykonywane na karcie graficznej, a transfer do pamięci RAM gospodarza był wykonywany jedynie w momencie zapisu do pliku.

Aby sprawdzić poprawność nowej implementacji wykonano obliczenia sprawdzające. Zadaniem testowym było wyznaczenie prędkości przemieszczania się pierścienia wirowego. Obliczenia były prowadzone na karcie graficznej. Otrzymane wyniki zostały porównane z wartościami teoretycznymi i wynikami z sekcji 4.2.

Dla cienkiego pierścienia wirowego o cyrkulacji Γ prędkość przemieszczania wyrażana jest przez wzór Kelvina (4.1) oraz wzór Hicksa (4.2).

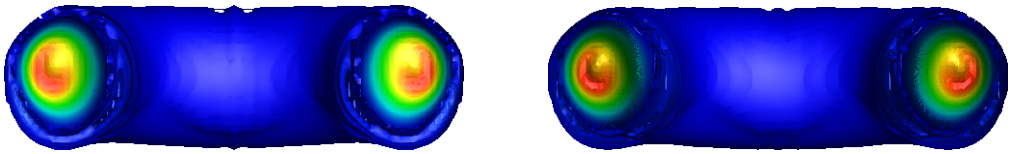
Zostały wykonane cztery testy numeryczne. Początkowe wymiary pierścienia wynosiły c, $R_0 = 1.5$, $r_0 = 0.3$ (rys. 4.1), a cyrkulacja pierścienia była zmieniana z zakresie $\Gamma \in [0.25, 1.00]$. Obszar obliczeniowy miał wymiary $2\pi \times 2\pi \times 2\pi$. Sitaka numeryczna miała $129 \times 129 \times 129$ węzłów. Na wszystkich brzegach zostały zadane okresowe warunki brzegowe.

Do rozwiązania równań ruchu cząstek wirowych wykorzystywana była metoda Rungego-Kutty czwartego rzędu z krokiem czasowym $\triangle t = 0.01$. Obliczenia były prowadzone na procesorze Intel Core i7 960 oraz karcie graficznej NVIDIA GTX 480, a ich wyniki przedstawione są na rys. 6.13 and rys. 6.14.

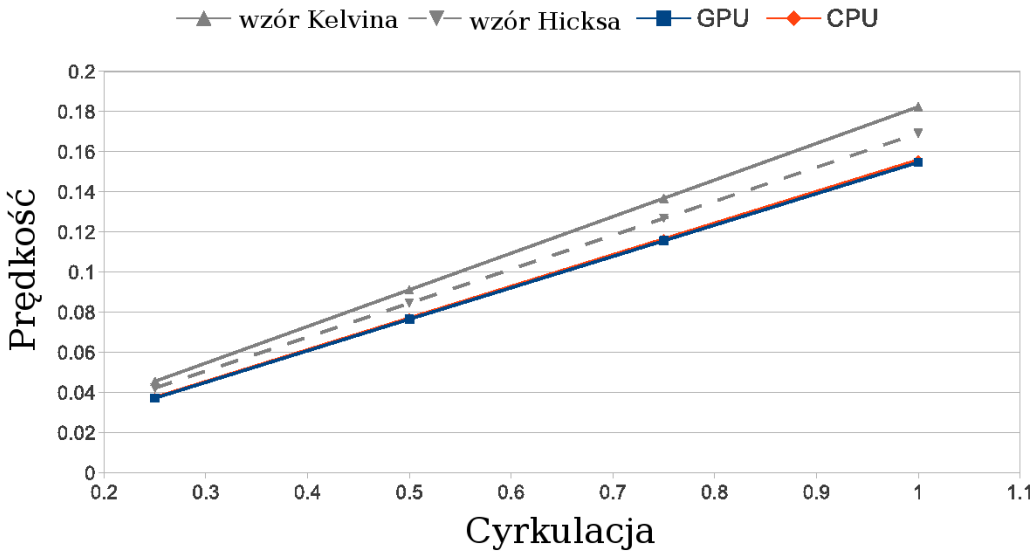
Jak widać na rys. 6.13 wyniki obliczeń na CPU i GPU są niemal identyczne. To pokazuje poprawność przeniesienia metody na kartę graficzną. Odchylenie wyników numerycznych od wartości teoretycznych zostało omówione w sekcji 4.2.

Czas obliczeń programu wykorzystującego kartę graficzną w obliczeniach był 46 razy krótszy niż programu działającego na procesorze. Jest to bardzo duże przyspieszenie.

Przedstawione wyniki zostały opublikowane w artykułach autorskich [50, 54].



Rysunek 6.13: Wyniki dla metody VIC po 600 krokach czasowych ($t = 6.0$) dla przypadku $\Gamma = 1.00$ i 129 węzłów numerycznych w każdym kierunku. Lewa strona - wyniki otrzymane na CPU, prawa - GPU



Rysunek 6.14: Prędkość pierścienia wirowego w funkcji cyrkulacji dla obliczeń prowadzonych na CPU i GPU

6.4. Rozwiązywanie równania dyfuzji

Ostatnim elementem algorytmu VIC przedstawionego w rozdziale 3 jest rozwiązanie równania dyfuzji symulującego zmianę wirowości cząstki spowodowaną lepkością płynu.

Równanie dyfuzji ma postać:

$$\frac{d\boldsymbol{\omega}}{dt} = \nu \Delta \boldsymbol{\omega}, \tag{6.35}$$

gdzie $\boldsymbol{\omega}$ jest wektorem wirowości a ν kinematycznym współczynnikiem lepkości płynu.

Dyskretyzacja przestrzenna tego zagadnienia, tak jak w poprzednim punkcie została wykonana przy użyciu siedmiopunktowego schematu różnic skończonych na siatce strukturalnej o równych krokach w każdym kierunku. Do dyskretyzacji czasowej równania (6.35) zastosowano schemat Cranka-Nicholsona postaci:

$$\frac{\omega_{i,j,k}^{n+1} - \omega_{i,j,k}^{n}}{\Delta t} = \Theta F(\omega^{n+1}) + (1-\Theta)F(\omega^{n+1}) \tag{6.36}$$

W zależności od parametru Θ otrzymuje się:

- $\Theta = 0$ - schemat jawny,
- $\Theta = 1$ - schemat niejawny,
- $\Theta = 1/2$ - klasyczny schemat jawno-niejawny.

Po zastosowaniu ostatniego z wymienionych schematów równanie (6.35) miało postać:

$$\begin{aligned}\frac{\omega_{i,j,k}^{n+1} - \omega_{i,j,k}^{n}}{\Delta t} = \frac{\nu}{2h^2} \Big(& \omega_{i-1,j,k}^{n+1} + \omega_{i,j-1,k}^{n+1} + \omega_{i,j,k-1}^{n+1} - 6\omega_{i,j,k}^{n+1} \\ & + \omega_{i+1,j,k}^{n+1} + \omega_{i,j+1,k}^{n+1} + \omega_{i,j,k+1}^{n+1} \\ & + \omega_{i-1,j,k}^{n} + \omega_{i,j-1,k}^{n} + \omega_{i,j,k-1}^{n} - 6\omega_{i,j,k}^{n} \\ & + \omega_{i+1,j,k}^{n} + \omega_{i,j+1,k}^{n} + \omega_{i,j,k+1}^{n} \Big) .\end{aligned} \tag{6.37}$$

W schemacie jawnym od razu wyznaczyć wszystkie wartości w nowym kroku czasowym. W schematach niejawnym i jawno-niejawny tworzone są układy równań algebraicznych liniowych, których rozwiązaniem są wartości zmiennych w nowym kroku czasowym. W różnych etapach pracy używane były wszystkie z wymienionych schematów, jednak w dalszej części opisane będą schematy niejawny oraz jawno-niejawny. Do rozwiązania otrzymanych układów równań wykorzystano metodę gradientów sprzężonych.

6.4.1. Metoda Gradientów Sprzężonych

Metoda gradientów sprzężonych wykorzystuje fakt, że rozwiązanie układu równań algebraicznych liniowych

$$A\mathbf{x} = \mathbf{b}, \tag{6.38}$$

gdy macierz A jest dodatnio określona

$$\mathbf{x}^T A\mathbf{x} = 0, \quad \forall x \in \mathbb{R}^n, \tag{6.39}$$

można sprowadzić do minimalizacji formy kwadratowej, która ma postać [92]:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x} + c. \tag{6.40}$$

Pochodna funkcji $f(\mathbf{x})$ (6.40) wyraża się wzorem:

$$f'(\mathbf{x}) = \frac{1}{2}A^T\mathbf{x} + \frac{1}{2}A\mathbf{x} - b. \tag{6.41}$$

Jeżeli macierz A jest dodatkowo macierzą symetryczną, to równanie (6.41) można zapisać jako:

$$f'(\mathbf{x}) = A\mathbf{x} - b. \tag{6.42}$$

Dla wartości pochodnej równej zero otrzymujemy równanie (6.38). Wartość zmiennej $\mathbf{x}$, dla której spełnione jest równanie (6.38) jest rozwiązaniem równania (6.42). Metoda gradientów sprzężonych jest metodą iteracyjną. Kolejne przybliżenia szukanego rozwiązania będą oznaczane indeksem i jako $\mathbf{x}_{(i)}$

Algorytm metody gradientów sprzężonych wyraża się następująco [92, 22]:

$$\mathbf{d}_{(0)} = \mathbf{r}_{(0)} = \mathbf{b} - A\mathbf{x}_{(0)}, \tag{6.43}$$

$$\alpha_{(i)} = \frac{\mathbf{r}^T_{(i)}\mathbf{r}_{(i)}}{\mathbf{d}^T_{(i)}A\mathbf{d}_{(i)}}, \tag{6.44}$$

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha_{(i)}\mathbf{d}_{(i)}, \tag{6.45}$$

$$\mathbf{r}_{(i+1)} = \mathbf{r}_{(i)} + \alpha_{(i)}A\mathbf{d}_{(i)}, \tag{6.46}$$

$$\beta_{(i+1)} = \frac{\mathbf{r}^T_{(i+1)}\mathbf{r}_{(i+1)}}{\mathbf{r}^T_{(i)}\mathbf{r}_{(i)}}, \tag{6.47}$$

$$\mathbf{d}_{(i+1)} = \mathbf{r}_{(i+1)} + \beta_{(i+1)}\mathbf{d}_{(i)}. \tag{6.48}$$

Proces iteracyjny zatrzymywany jest po zadanej liczbie iteracji lub gdy obliczane residua $\mathbf{r}_{(i+1)}$ spadną poniżej zadanej wartości.

6.4.2. Badania numeryczne implementacji metody gradientów sprzężonych na GPU

Równanie (6.35) zostało zdyskretyzowane przy pomocy centralnych różnic skończonych dla pochodnych po przestrzeni oraz schematu Crancka-Nicholsona (6.36) dla pochodnej po czasie. Zastosowany schemat dla członów przestrzennych jest drugiego rzędu. W badaniach został sprawdzony rząd zbieżności schematów: niejawnego ($\Theta = 1$ w (6.36)) oraz jawno-niejawnego ($\Theta = 1/2$ w (6.36)).

Zadaniem testowym było trójwymiarowe równanie dyfuzji (z $\nu \equiv 1$), którego rozwiązaniem była funkcja:

$$\psi\left(x, y, z\right) = e^{-3\nu t} sin(x) \cdot sin(y) \cdot sin(z), \qquad x, y, z \in [0, 2\pi] \tag{6.49}$$

Na brzegach obszaru obliczeniowego przyjęto okresowe warunki brzegowe. Obliczenia zostały przeprowadzone na siatkach numerycznych o różnej liczbie

węzłów, tak aby możliwe było zbadanie rzędu i zbieżności metod. Za rozwiązanie uważano wektor, dla którego maksymalna wartość residuów nie przekraczała 10^{-20}.

Całkowity błąd metody rozwiązywania równania dyfuzji jest sumą błędów po przestrzeni oraz po czasie $(O(\delta) + O(\tau))$. Ponieważ schemat „wstecz" jest pierwszego rzędu, a schemat różnic centralnych drugiego rzędu to błąd całkowity metody będzie miał postać $(O(h^2) + O(\Delta t))$. Na tej podstawie do testów została przyjęta zasada, że zmiana kroku czasowego będzie proporcjonalna do zmiany kroku przestrzennego w kwadracie $(h^2/\Delta t = const.)$. Wyniki otrzymane dla schematu niejawnego zostały przedstawione jako funkcja kroku przestrzennego w Tabeli 6.6 oraz na rys. 6.15.

Tabela 6.6: Zbieżność schematu „wstecz" rozwiązywania równania dyfuzji

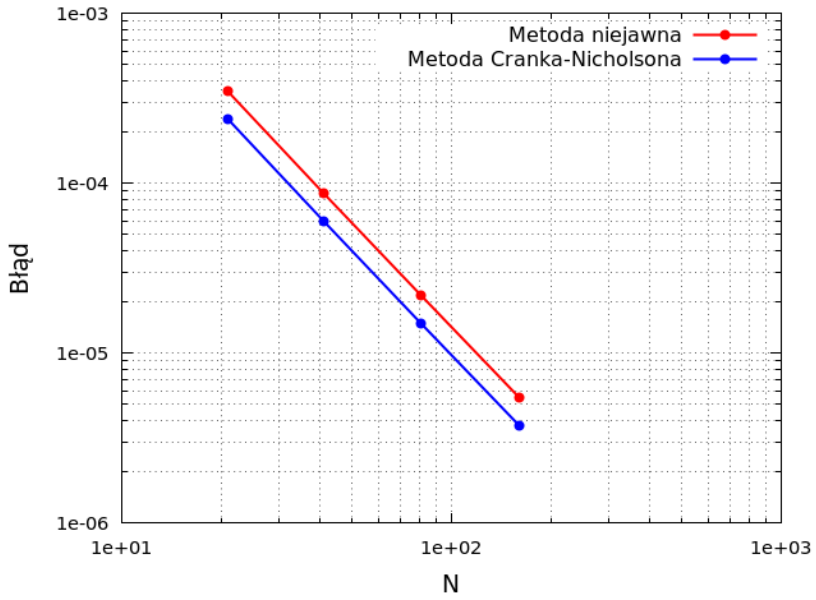
N	h	Δt	błąd $(t = 0.1)$	rząd
10	0.628319	0.1	1.16E-003	
20	0.314159	0.025	3.46E-004	1.75
40	0.15708	0.00625	8.70E-005	1.99
80	0.07854	0.0015625	2.18E-005	2.00
160	0.03927	0.000390625	5.45E-006	2.00

Zastosowany w drugim przypadku schemat jawno-niejawny jest drugiego rzędu po czasie. Całkowity błąd metody będzie miał zatem postać $(O(h^2)+O(\Delta t^2))$. Na tej podstawie do testów została przyjęta zasada, że zmiana kroku czasowego będzie proporcjonalna do zmiany kroku przestrzennego $(h/\Delta t = const.)$. Otrzymane wyniki zostały przedstawione w Tabeli 6.7 oraz na rys. 6.15.

Tabela 6.7: Zbieżność schematu Cranka-Nicholsona rozwiązywania równania dyfuzji

N	h	Δt	błąd $(t = 0.1)$	rząd
10	0.628319	0.1	8.12E-004	
20	0.314159	0.05	2.38E-004	1.77
40	0.15708	0.025	5.97E-005	2.00
80	0.07854	0.0125	1.49E-005	2.00
160	0.03927	0.00625	3.73E-006	2.00

W obu z przedstawionych przypadkach rząd metody wynosił ok. 2.00 co w tym wypadku było wartością oczekiwaną. Pozwala to wnioskować, że zastosowane schematy oraz implementacja metody są poprawne.



Rysunek 6.15: Zbieżność metody niejawnej i metody Crancka-Nicholsona użytych do rozwiązania równania dyfuzji zgodnie z opisanymi założeniami.

6.5. Obliczenia na wielu kartach graficznych

Implementacja algorytmu VIC wykorzystująca w obliczeniach pojedynczą kartę graficzną ma pewne ograniczenia. Z punktu widzenia obecnej pracy jednym z najważniejszych jest dostępna ilość pamięci RAM na pojedynczym GPU. W przedstawionej implementacji na jednej karcie można było pomieścić przypadek z siatką nie większą niż 128x128x128 węzłów (GTX480, dla GTX580 z 3GB było to już 224x224x224). Siatki takie są wystarczające w prostych zastosowaniach inżynieryjnych. Sposobem na pozbycie się ograniczenia związanego z pamięcią RAM jest wykorzystanie w obliczeniach wielu kart graficznych. Zostaje wtedy zwielokrotniona dostępna pamięć. Aby zapewnić poprawność obliczeń numerycznych pewien obszar danych (zwany w literaturze angielskiej jako „halo") z każdego procesu równoległego musi zostać przesłany do innego procesu. Wykorzystanie wielu procesów w obliczeniach powoduje to potrzebę napisania dodatkowego kodu mającego na celu rozdział danych pomiędzy karty oraz zapewnienie komunikacji pomiędzy nimi.

Ostateczna implementacja wykorzystująca wiele kart graficznych do obliczeń numerycznych była tworzona z myślą o wykorzystaniu jej na klastrach komputerowych tj. grupach komputerów, zwanych węzłami, działających jak serwery, połączonych ze sobą poprzez lokalną sieć LAN (*ang.* Local Area Network). Każdy z tych węzłów może pracować jako samodzielny komputer. W docelowej konfiguracji węzły mogą mieć więcej niż jedną kartę graficzną przeznaczoną do obliczeń

numerycznych (w centrach obliczeniowych można spotkać węzły komputerowe wyposażone nawet w 8 kart graficznych [4]).

Do stworzenia ostatecznej wersji programu do obliczeń metodą VIC wykorzystane zostało tzw. programowanie hybrydowe MPI-OpenMP (*ang.* hybrid MPI-OpenMP programming), zwane także programowaniem wielopoziomowym (*ang.* multilevel programming). W modelu tym wykorzystywane są dwa standardy do prowadzenia obliczeń równoległych. Wprowadźmy pojęcie procesu jako ciągu instrukcji wykonywanych przez procesor wraz z dostępem do pamięci lokalnej [2].

OpenMP (*ang.* Open Message Passing) jest standardem do programowania równoległego na systemach z pamięcią współdzieloną (tj. takich w których wszystkie procesy równoległe mogą mieć dostęp do obszarów pamięci zaalokowanych przez którykolwiek inny proces). Zmienne stworzone przez dany proces mogą być współdzielone lub prywatne.

W standardzie MPI (*ang.* Message Passing Interface) każdy proces tworzy własną kopię potrzebnych danych do której pozostałe procesy nie mają dostępu. Jednak wykorzystując odpowiednie instrukcje dane te mogą zostać przesłane pomiędzy procesami.

Podstawową zaletą programowania hybrydowego jest wykorzystanie standardu OpenMP do zapewnienia komunikacji pomiędzy procesami działającymi na jednym węźle oraz wykorzystaniu standardu MPI do wymiany informacji pomiędzy węzłami.

Do wymiany informacji pomiędzy kartami graficznymi działającymi na tym samym węźle została wykorzystana technologia GPUDirect 2.0. Pozwala ona na wykonywanie transferów danych pomiędzy GPU bez wykorzystania pamięci RAM gospodarza (hosta). Dzięki wykorzystaniu strumieni CUDA (*ang.* CUDA streams), transfer ten odbywał się bez blokowania pozostałych operacji (mówimy, że transfer jest asynchroniczny). Aby w efektywny sposób wykorzystać kopiowanie asynchroniczne należy napisać program tak, aby najpierw obliczenia były wykonywane na obszarze „halo". Następnie jednocześnie wysyłane są dane i wykonywane są obliczenia dla pozostałego obszaru. W przypadku bibliotek MPI i OpenMP wykonywanie tych operacji jest automatyczne. Aby użyć tego mechanizmu dostępnego w kartach z technologią CUDA należy jawnie stworzyć dwa strumienie CUDA i przypisać do jednego operację kopiowania danych a do drugiego pozostałe obliczenia.

Rozważania na temat wprowadzenia obliczeń równoległych na wielu kartach graficznych do metody VIC zostały przedstawione w artykułach autorskich [60, 61, 62, 63, 55]

Poniżej przedstawiony zostanie opis kolejnych etapów implementacji iteracyjnej metody Jacobiego do rozwiązywania układów równań liniowych.

6.5.1. Rozdział danych pomiędzy karty graficzne

Do obliczeń wykorzystywany jest sześcienny obszar obliczeniowy podzielony strukturalną siatką numeryczną. Przy dekompozycji obszaru obliczeniowego jednym z najważniejszych wskaźników, mówiących o jakości podziału, jest wielkość danych, która będzie musiała być wymieniana pomiędzy procesami, czyli wielkość obszaru „halo". Najbardziej efektywnym pod tym względem podziałem dla wybranego kształtu i dyskretyzacji obszaru obliczeniowego jest podział na mniejsze sześciany, taki sam w każdym kierunku (czyli np. 8 sześcianów, po 2 w każdym kierunku). Ze względu na trudność prezentacji na rys. 6.16 przedstawiony został przykład takiego podziału dla przypadku dwuwymiarowego. Dzięki temu otrzymany zostanie minimalny rozmiar obszaru „halo". Niestety przy takim podziale nie wszystkie przesyłane dane stanowią spójny obszar w pamięci urządzenia. Biblioteka MPI oferuje funkcję pozwalającą ominąć tę niedogodność, ale w momencie pisania niniejszej pracy nie istniała taka funkcja dla kopiowania danych na kartach graficznych. W tym wypadku należało by napisać funkcję, która sama składa potrzebne dane w jeden spójny obszar pamięci, przesyła je a następnie odpowiednio zapisuje w miejscu docelowym. Aby uniknąć tej komplikacji w prezentowanym rozwiązaniu będzie wykorzystywany podział tylko wzdłuż jednej osi (w tym wypadku osi z). Przykład takiego podziału jest pokazany na rys. 6.17. Dzięki temu dane wymieniane pomiędzy procesami będą zawsze zajmowały obszar spójny w pamięci urządzenia.

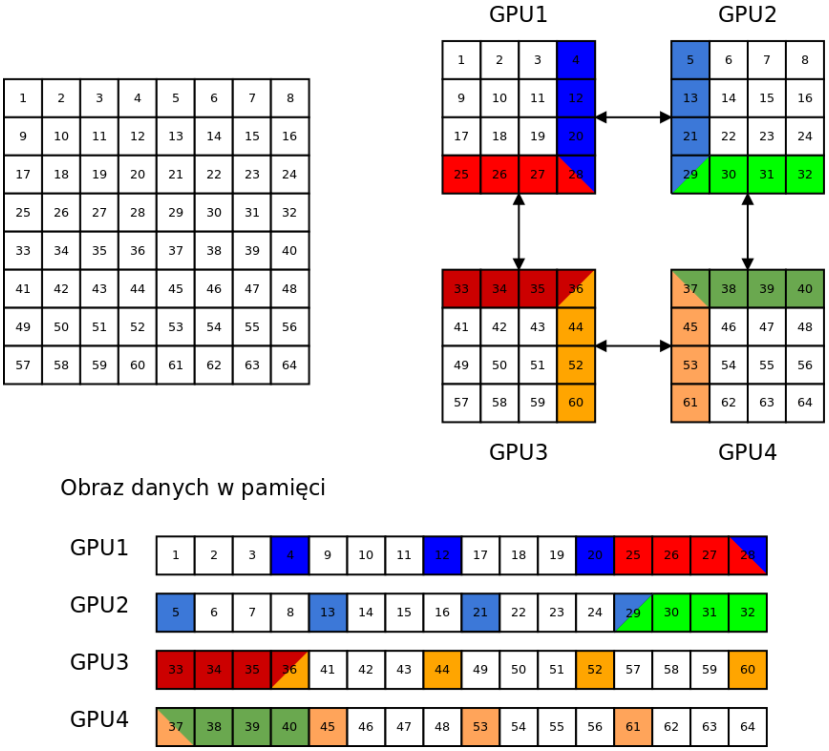
Kopiowanie blokujące i nieblokujące w standardzie MPI

Najpierw wykorzystany został standardu MPI do kopiowania danych pomiędzy kartami. W ten sposób będzie prowadzona wymiana danych pomiędzy węzłami klastra w ostatecznej implementacji. Program zostaje uruchomiony tak, że jeden proces MPI kontroluje obliczenia na jednej karcie graficznej.

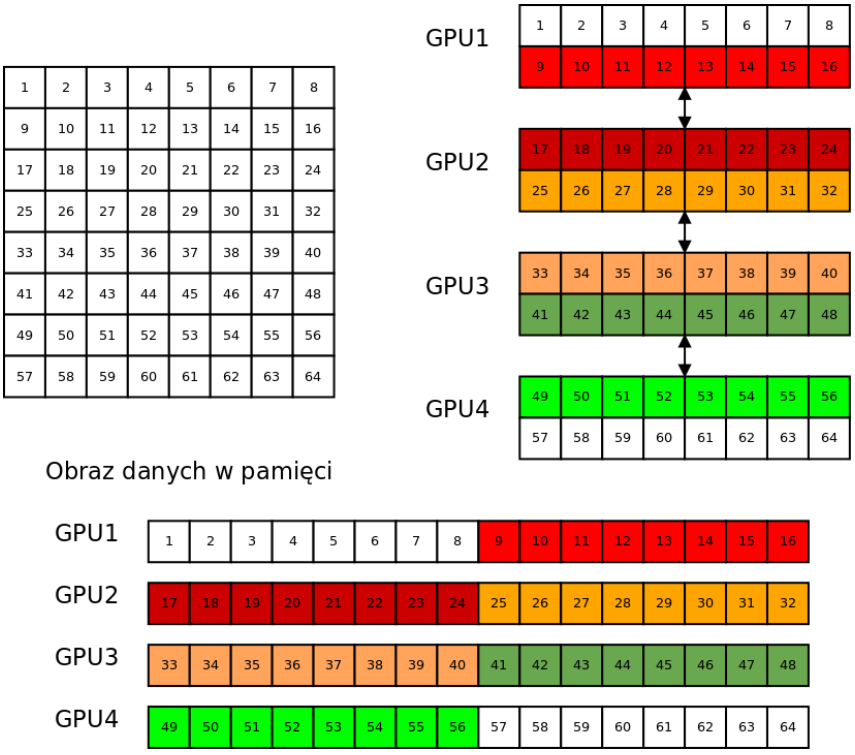
Wymiana danych pomiędzy kartami graficznymi wymaga wykonania trzech operacji:

- skopiowanie danych z pamięci karty graficznej do pamięci RAM gospodarza (hosta),
- przesłanie danych pomiędzy procesami,
- skopiowanie danych na docelową kartę graficzną.

Do przeprowadzenia drugiej z wymienionych operacji, czyli przesyłania danych pomiędzy procesami, wykorzystano bibliotekę MPI. Oferuje ona dwa podstawowe typy komunikacji pomiędzy równolegle działającymi procesami: blokujący i nieblokujący [2]. Kopiowanie blokujące zawiesza wykonanie pozostałego kodu do momentu ukończenia kopiowania danych. Dzięki temu obszar pamięci,



Rysunek 6.16: Dekompozycja siatki obliczeniowej na 4 części z najmniejszą liczbą danych przesyłanych pomiędzy równoległymi procesami.



Rysunek 6.17: Dekompozycja siatki obliczeniowej na 4 części z danymi przesyłanymi pomiędzy procesami będącymi ciągłym obszarem pamięci.

na którym odbywała się ta operacja jest od razu gotowy do użycia. Kopiowanie nieblokujące zleca wykonanie kopii danych i przechodzi do kolejnej linijki kodu. Pozwala ta na równoczesne wykonywanie obliczeń i kopiowanie danych. Aby skorzystać z obszaru pamięci, na którym wykonywana jest ta operacja, trzeba się upewnić wcześniej, że została ona ukończona. Wykorzystanie komunikacji nieblokującej pozwala na osiągnięcie krótszych czasów wykonywania programów. Przyspieszenie algorytmu S definiuje się następująco [21]:

$$S(p,n) = \frac{T(1,n)}{T(p,n)} \tag{6.50}$$

gdzie T oznacza złożoność czasową programu, p liczbę wykorzystanych procesów (kart graficznych), a n – rozmiar problemu.

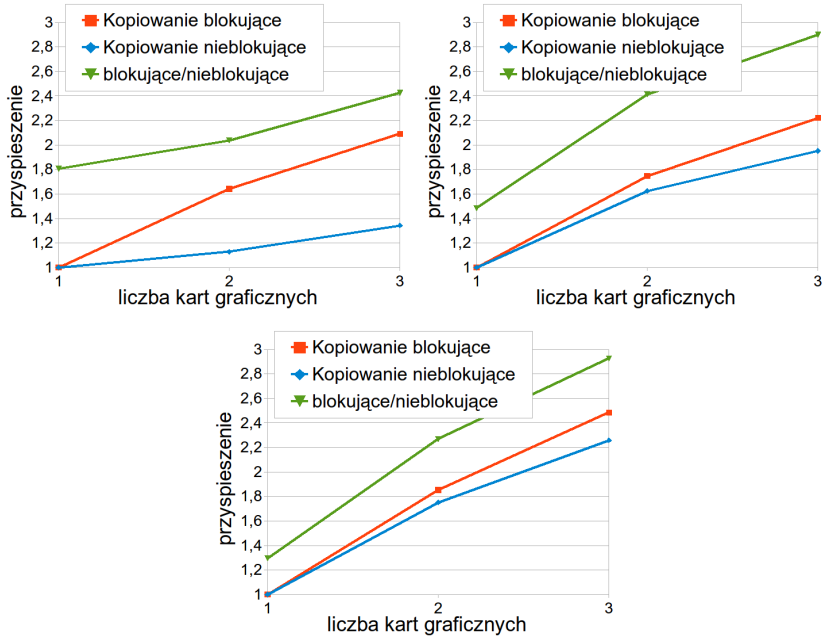
Początkowe testy zostały wykonane na komputerze wyposażonym w trzy karty graficzne: NVIDIA GeForce GTX 590, widziana w systemie jako dwa osobne urządzenia, i NVIDIA GeForce GTX 480

Najpierw przetestowano kopiowanie blokujące. Dekompozycja obszaru obliczeniowego została dokonana równomiernie wzdłuż osi z. Dla zadanej funkcji prawej strony równania Poissona (3.100) oraz zerowych warunków brzegowych zostało wykonanych 5000 iteracji metodą Jacobiego.

Ten sam test został powtórzony z wykorzystaniem kopiowanie nieblokującego. Porównanie przyspieszenia uzyskanego obydwoma metodami dla trzech różnych wielkości obszaru obliczeniowego przedstawione jest na rys. 6.18

Jak widać dla obszaru obliczeniowego $65 \times 65 \times 65$ przyspieszenie kopiowania nieblokującego (linia niebieska) jest niewielkie (tylko ok. 1.3x szybciej przy wykorzystaniu 3 kart). Wartość ta jest wyższa dla kopiowania blokującego (linia czerwona). Wyniki te mogą być mylące ponieważ kod wykorzystujący kopiowanie nieblokujące wykonuje się szybciej (linią zieloną przedstawiono przyspieszenie obliczeń dla kopiowania blokującego odniesione do czasu obliczeń kopiowania nieblokującego na jednej karcie graficznej - jak widać czas wykonania programu jest krótszy niż dla kopiowania blokującego). Wzrost liczby węzłów w obszarze obliczeniowym znacząco poprawia efektywność wykorzystania kart graficznych. Spowodowane jest to zwiększeniem czasu obliczeń w stosunku do czasu transferu danych.

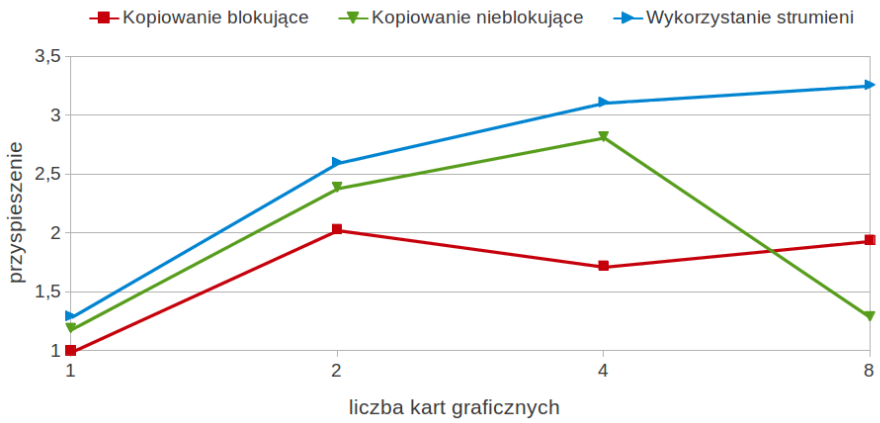
Biblioteka MPI, z której pochodzą przedstawione procedury, wykorzystywana jest tylko w drugim z wymienionych etapów. Pozostałe dwa dalej wymagają zatrzymania obliczeń na czas kopiowania danych. Problem ten widoczny staje się przy wykorzystaniu 8 kart graficznych. Na rys. 6.19 przedstawiono uzyskane przyspieszenie względem jednej karty graficznej z wykorzystaniem kopiowania blokującego. Dla programu wykorzystującego kopiowanie blokujące użycie więcej niż dwóch kart graficznych nie powoduje dalszego przyspieszenia obliczeń. Z kolei dla programu wykorzystującego asynchroniczny transfer danych ten problem widoczny jest przy wykorzystaniu ośmiu kart graficznych.



Rysunek 6.18: Przyspieszenie dla kopiowania blokującego i nielokującego dla obszarów obliczeniowych: lewy - $65 \times 65 \times 65$, środkowy - $129 \times 129 \times 129$, prawy - $257 \times 257 \times 257$

Wykorzystanie strumieni CUDA

Aby zaradzić spadkowi wydajności opisanemu w poprzednim punkcie należy wprowadzić kopiowanie asynchroniczne dla danych przesyłanych z i na kartę graficzną. Możliwe staje się to poprzez wykorzystanie tzw. strumieni CUDA. Jak już wiadomo strumieniami nazywane są ciągi instrukcji dla karty graficznej. Budowa karty pozwala na jednoczesne kopiowanie danych z i do pamięci globalnej oraz prowadzenie obliczeń pod warunkiem, że działania te będą wykonywanie przez dwa różne strumienie [3]. Do efektywnego wykorzystania strumieni należy napisać program tak, aby najpierw obliczenia zostały wykonane dla obszaru, który jest przesyłany. Następnie jeden strumień zajmuje się przesłaniem danych do pamięci RAM gospodarza a drugi prowadzi obliczenia dla pozostałej części obszaru. Tak samo jest wykonywane kopiowanie danych na kartę graficzną. Pozwala to na ukrycie czasu transferów pamięci w czasie obliczeń, a tym samych poprawę skalowania programu. Widoczne jest to na rys. 6.19. Kolorem niebieskim zaznaczono przyspieszenie dla programu wykorzystującego strumienie. Widać dalszy wzrost przyspieszenia przy wykorzystaniu 8 kart.



Rysunek 6.19: Przyspieszenie programu wykorzystującego wiele kart graficznych i różne metody kopiowania danych względem obliczeń na jednej karcie i kopiowania blokującego dla obszaru obliczeniowego 257x257x257

6.5.2. Skalowanie

Nieformalnie, skalowalność dotyczy możliwości zwiększania przyspieszeń lub, równoważnie, utrzymywania stałej efektywności wraz ze wzrostem liczby procesów użytych do obliczeń. Na skalowalność ma wpływ nie tylko budowa algorytmu równoległego, ale także czas komunikacji pomiędzy procesami oraz właściwości komputera równoległego, w którym algorytm jest wykonywany [21].

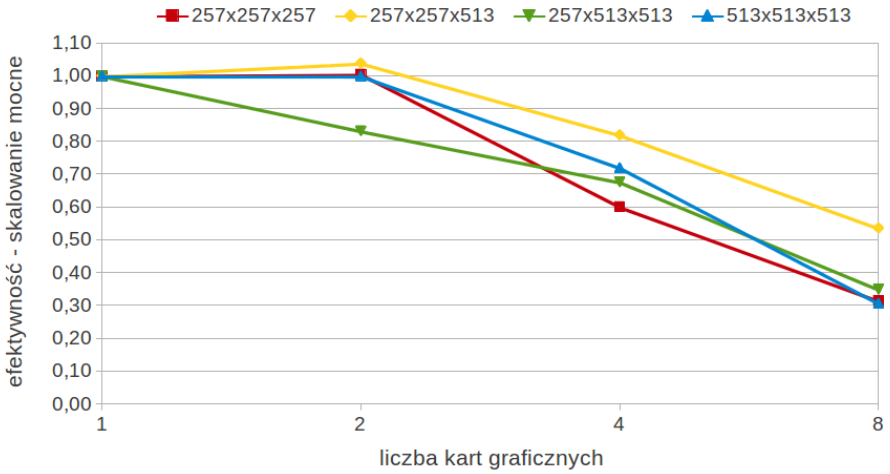
Zaprezentowane w tym punkcie charakterystyki powstały przy wykorzystaniu węzłów obliczeniowych superkomputera Zeus znajdującego się w Akademickim Centrum Komputerowym CYFRONET AGH. Możliwe to było dzięki programowi PL-Grid. Każdy węzeł obliczeniowy klastra składał się z dwóch procesorów Intel Xeon X5670, 70GB pamięci operacyjnej oraz z dwóch kart graficznych NVIDIA Tesla M2050. Obliczenia wykorzystywały maksymalnie cztery takie węzły. Wszystkie obliczenia były prowadzone na programie wykorzystującym kopiowanie nieblokujące i strumienie opisanym w poprzednim punkcie.

Skalowanie mocne

Skalowanie mocne przedstawia możliwość wykorzystania większej liczby kart graficznych (procesów) do obliczeń bez zmiany obszaru obliczeniowego. Stosuje się je dla programów, które charakteryzują się długim czasem obliczeń. Ta sama całkowita liczba węzłów siatki numerycznej dzielona jest na więcej podobszarów i rozsyłana pomiędzy karty graficzne. Efektywność w skalowaniu mocnym (E_m) definiuje się jako:

$$E_m(p,n) = \frac{T(1,n)}{pT(p,n)} \tag{6.51}$$

Wyniki dla badanego programu i różnych wielkości obszaru obliczeniowego przedstawione są na rys. 6.20



Rysunek 6.20: Efektywność programu wykorzystującego wiele kart graficznych i różne wielkości obszarów obliczeniowych

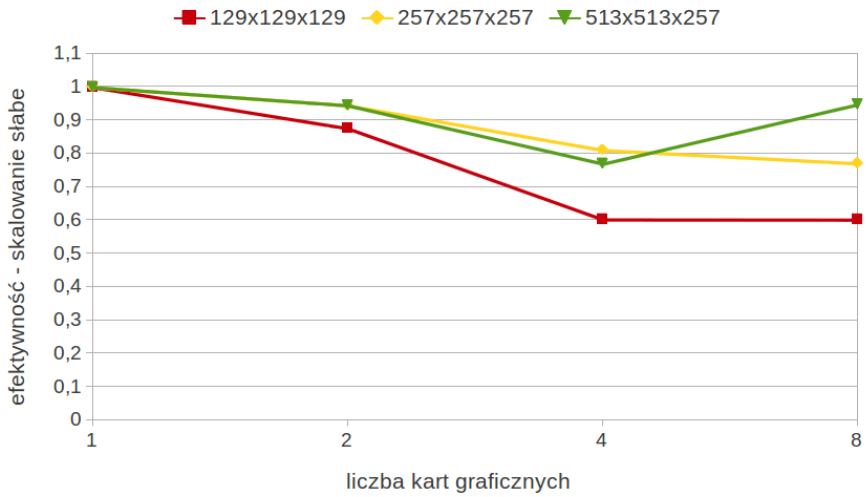
Najwyższa efektywność została uzyskana dla przypadku $257 \times 257 \times 513$. Wynika to z jego najlepszego stosunku liczby węzłów wewnętrznych do liczby przesyłanych danych (podział obszaru jest tylko w osi z). Efektywność pozostałych przypadków wynosi pomiędzy 0.3 a 0.4. Pokazuje to, że wykorzystanie coraz większej liczby kart graficznych bez zwiększania liczby węzłów pozwala jedynie na niewielkie przyspieszenie obliczeń.

Skalowanie słabe

Potrzeba wykorzystania w niniejszej pracy gęstych siatek obliczeń powoduje, że ważniejszym w tym wypadku wskaźnikiem jest tzw. skalowanie słabe. W skalowaniu słabym stała jest liczba węzłów obliczeniowych przypadających na jedną kartę graficzną (stały nakład pracy). Oddaje to sytuację, w której badany przypadek będzie zbyt duży, aby zmieścić się na jednej jednostce GPU. Każdej karcie przydzielamy taką samą liczbę węzłów obliczeniowych. Przy zwiększeniu liczby kart zwiększa się również obszar obliczeniowy. W idealnym przypadku, kiedy karty nie musiałyby się komunikować lub czas transferów danych byłby całkowicie ukryty, czas obliczeń dla wielu kart byłby taki sam jak dla jednej karty. Niestety kopiowanie pomiędzy procesami nie pozwala na to. Efektywność w skalowaniu słabym E_s definiuje się jako:

$$E_s(p, n) = \frac{T(1, n)}{T(p, n)} \tag{6.52}$$

Na rys. 6.21 przedstawiono efektywność algorytmu. Dla idealnego przypadku wykres byłby linią prostą na poziomie 1.



Rysunek 6.21: Efektywność programu wykorzystującego wiele kart graficznych i różne wielkości obszarów obliczeniowych

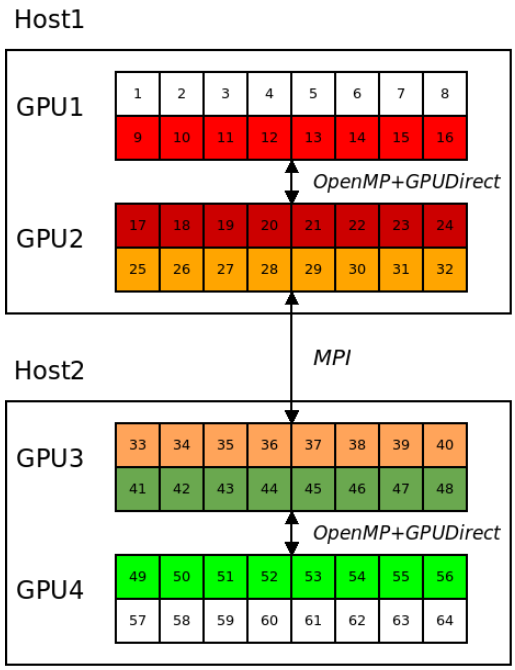
Na wszystkich wykresach widać wyraźne pogorszenie przy przejściu z dwóch do czterech GPU. Spowodowane jest to koniecznością wymiany danych pomiędzy węzłami przez sieć, która jest wolniejsza niż wewnętrzne połączenia w komputerze. Jak widać przypadek $129 \times 129 \times 129$ wypada najgorzej ponieważ najmniej obciąża kartę obliczeniowo. O wiele lepsze wyniki osiągają pozostałe dwa przypadki.

Wyniki skalowania słabego i mocnego zostały opublikowane w artykułach autorskich [60, 63, 55, 57].

6.6. Programowanie hybrydowe MPI-OpenMP z wykorzystaniem GPUDirect

Następnie stworzona została implementacja metody Jacobiego przy pomocy programowania hybrydowego. Na potrzeby tego rozdziału można przyjąć, że program do obliczeń metodą VIC z siatką podzieloną na cztery części tak jak na rys. 6.17 został uruchomiany na dwóch węzłach obliczeniowych z dwoma kartami graficznymi każdy. Oznacza to, że pomiędzy urządzeniami GPU1 i GPU2 oraz GPU3 i GPU4 do komunikacji będzie wykorzystywany standard OpenMP, a pomiędzy urządzeniami GPU2 i GPU3 - standard MPI. Widoczne jest to na rys. 6.6.

W opisie zostaną wymienione tylko dodatkowe operacje pozwalające na uruchomienia równoległych procesów MPI-OpenMP. Zostanie pominięty, przedsta-



Rysunek 6.22: Konfiguracja uruchomienia testu używającego programowania hybrydowego MPI-OpenMP.

wiony wcześniej, opis obliczeń na karcie graficznej. Pierwszym etapem tworzenia kodu jest utworzenie procesów MPI.

```
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);

MPI_Cart_create(MPI_COMM_WORLD, ndims, dims, period, reorder,
                &grid_comm);
MPI_Comm_rank(grid_comm, &me);
```

Napstępnie tworzone są procesy OpenMP. Wszystkie operacje, które mają być wykonane przez te procesy muszą się znajdować w klamrach znajdujących się po wywołaniu.

```
#pragma omp parallel num_threads(num_th) private(...) shared(...)
{
    ...
}
```

O ile w inicjalizacji procesów MPI ich liczba jest zależna od komendy uruchamiającej program o tyle w OpenMP ta liczba musi być jawnie podana, w przykładzie jest to `num_th`, lub zadana jako zmienna systemowa `OMP_NUM_THREADS`. Po wywołaniu można podać listę zmiennych prywatnych (`private`) oraz współdzie-

lonych (`shared`). Zmienna, której nie będzie na żadnej z tych list jest domyślnie zmienną współdzieloną.

Kolejnym krokiem jest utworzenie dwóch strumieni CUDA do obliczeń na brzegu obszaru i we wnętrzu. Dodatkowo jest włączana możliwość transferu danych pomiędzy kartami bez konieczności przesyłania ich do pamięci hosta.

```
cudaStreamCreate(&(stream_bound));
cudaStreamCreate(&(stream_inter));

cudaDeviceEnablePeerAccess(gpunr[(tid+1)%num_th], 0);
```

Po takiej inicjalizacji można przejść do obliczeń numerycznych. Uruchamiana jest pętla iteracji metodą Jacobiego. W niej pierwszą operacją jest wykonanie obliczeń dla obszaru „halo", który będzie musiał być przesłany do innych kart graficznych. Następnie jednocześnie uruchamiane są obliczenia dla pozostałej części obszaru oraz rozpoczynany jest transfer danych z brzegu obszaru.

```
jacobi3D_boundaries_gpu(..., stream_bound);

cudaDeviceSynchronize();
#pragma omp barrier

jacobi3D_interior_gpu(..., stream_inter);

if (tid == 0)
{
    cudaMemcpyAsync(..., stream_bound);

}
else
{
    cudaMemcpyPeerAsync(..., stream_bound);
}

if (tid == num_th-1)
{
    cudaMemcpyAsync(..., stream_bound);
}
else
{
    cudaMemcpyPeerAsync(..., stream_bound);
}

cudaDeviceSynchronize();
#pragma omp barrier
```

Operacje na obszarze „halo" (obliczenia i kopiowanie) oraz operacje na wnętrzu obszaru są wykonywane przez różne strumienie CUDA (w przykładzie odpowiednio `stream_bound` i `stream_inter`). W zależności od położenia danego procesu dane są kopiowane albo do pamięci hosta w celu przesłania do innego węzła

obliczeniowego (`cudaMemcpyAsynch`) lub bezpośrednio do innej karty graficznej (`cudaMemcpyPeerAsynch`). Dane skopiowane do pamięci hosta muszą następnie zostać wysłane do innego węzła.

```
if (tid == 0)
{
    MPI_Cart_shift(grid_comm, 0, -1, &below1, &above1);
    MPI_Isend(..., above1, 1001, MPI_COMM_WORLD, &request1);
    MPI_Irecv(..., below1, 1001, MPI_COMM_WORLD, &request2);

    MPI_Cart_shift(grid_comm, 0, 1, &below2, &above2);
    MPI_Isend(..., above2, 999, MPI_COMM_WORLD, &request3);
    MPI_Irecv(..., below2, 999, MPI_COMM_WORLD, &request4);
}
```

Ponieważ kopiowanie pomiędzy procesami MPI jest asynchroniczne należy jawnie zadać warunek sprawdzający ukończenie tej operacji przed wysłaniem tych danych na GPU. Transfer danych na kartę odbywa się ponownie przy pomocy strumienia CUDA `stream_bound`.

```
if(tid == 0)
{
    MPI_Wait(&request2, &status1);
    MPI_Wait(&request4, &status2);
}

cudaDeviceSynchronize();
#pragma omp barrier

if (tid == 0)          cudaMemcpyAsync(..., stream_bound);
if (tid == num_th-1) cudaMemcpyAsync(..., stream_bound);

cudaDeviceSynchronize();
#pragma omp barrier
```

Kompilacja tak napisanego kodu odbywa się w dwóch etapach. Ponieważ zawiera on kod zarówno CUDA jak i MPI-OpenMP to najpierw kompilowany jest sam kod CUDA:

```
nvcc -cuda -arch=sm_20 -lgomp -Xcompiler -fopenmp gpu_jacobi_async.cu
apjac3D.cu -I/usr/lib/openmpi/include
```

W wyniku tego powstają pliki z rozszerzeniem `.cu.cpp.ii`. Te pliki należy skompilować kompilatorem MPI z dołączoną flagą OpenMP

```
mpicxx -fopenmp -c gpu_jacobi_async.cu.cpp.ii apjac3D.cu.cpp.ii
-I/usr/local/cuda/include
mpicxx -fopenmp apjac3D.cu.cpp.o gpu_jacobi_async.cu.cpp.o
-L/usr/local/cuda/lib64 -lcudart -o apjac3D_gpu_mpiomp
```

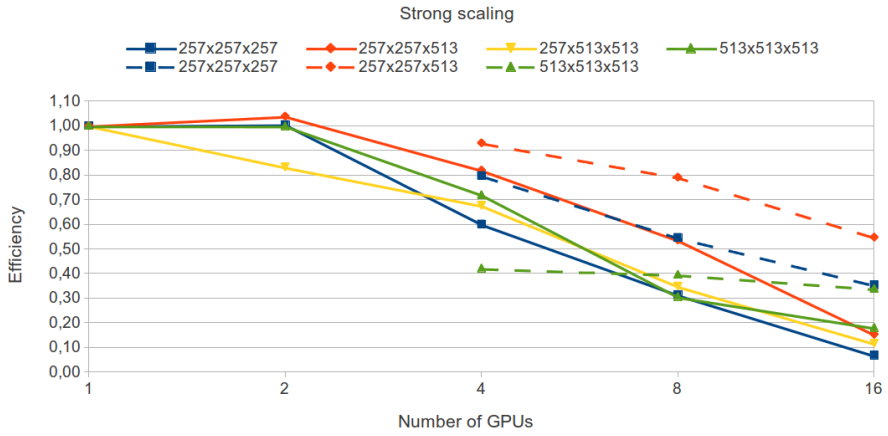
Uruchomienie programu odbywa się tak jak uruchamia się programy MPI.

```
mpiexec -hostfile hostfile -pernode apjac3D_gpu_mpiomp
```

Należy jednak pamiętać, że powinien zostać uruchomiony dokładnie jeden proces MPI na każdym węźle obliczeniowym. Służy do tego flaga `-pernode`. Lista nazw węzłów obliczeniowych znajduje się w pliku `hostfile`.

6.6.1. Skalowanie

Wyniki skalowania mocnego uzyskanego dla programu wykorzystującego programowanie hybrydowe MPI-OpenMP są przedstawione na rys. 6.23. Widoczna jest znacząca poprawa efektywności mocnego skalowania.



Rysunek 6.23: Efektywność programu wykorzystującego wiele kart graficznych i różne wielkości obszarów obliczeniowych wykorzystującego do komunikacji standard MPI (linia ciągła) i programowanie hybrydowe MPI-OpenMP (linia przerywana).

Testy wykazały, że implementacja metody VIC używająca programowania hybrydowego MPI-OpenMP do obliczeń na wielu kartach graficznych pozwoliła na powiększenie rozmiaru siatki a tym samym liczby cząstek w badanych przypadkach oraz na przyspieszenie obliczeń. Dzięki temu możliwe było przebadanie przedstawionych wcześniej przypadków z większą dokładnością a także dodanie nowych eksperymentów do badań.

Zwiększenie liczby kart graficznych dostępnych w obliczeniach numerycznych można wykorzystać na dwa sposoby. W pierwszym można podzielić zadany obszar obliczeniowy pomiędzy więcej kart a tym samym wykorzystać więcej jednostek obliczeniowych i skrócić czas obliczeń. Sytuacja ta odpowiada skalowaniu mocnemu przedstawionemu w punkcie 6.5.2. W niniejszej pracy ważniejsze jednak było drugie zastosowanie, a mianowicie wykorzystanie większej ilości pamięci RAM

(odpowiadające skalowaniu słabemu na wielu kartach graficznych, opisanemu w punkcie 6.5.2). Pozwala to na zagęszczenie siatki obliczeniowej lub zwiększenie obszaru obliczeniowego. Dzięki temu możliwe jest zwiększenie dokładności obliczeń.

Rozdział 7

Badania numeryczne ewolucji struktur wirowych

Program rozwiązujący równania Naviera–Stokesa metodą typu „Wir w komórce” wykorzystujący do obliczeń wiele kart graficznych umożliwił rozwiązanie szeregu problemów pozwalających na lepsze poznanie zjawisk towarzyszących wzajemnym oddziaływaniom struktur wirowych w płynie lepkim.

Dzięki znaczącemu przyspieszeniu obliczeń można było przeprowadzić więcej testów wprowadzających, co pozwoliło sprawdzić różne konfiguracje warunków początkowych oraz zbadać różne parametry przepływu.

Do testów numerycznych zostały wybrane dobrze udokumentowane w literaturze przypadki rekonekcji struktur wirowych. Na ich podstawie można było ocenić poprawność implementacji metody na karcie graficznej.

7.1. Zjawisko Gry wirów

Jako pierwszy, wybrano eksperyment pokazujący tzw. grę wirów (*ang.* vortex game, leapfrogging), dla którego wyniki eksperymentalne opublikowano w [68], a numeryczne m.in. w [64]. W zjawisku tym dwa pierścienie wirowe, w przedstawionym przypadku o takiej samej cyrkulacji, oddziałują ze sobą. Początkowa konfiguracja przedstawiona jest na rys. 7.1. Prędkość indukowana przez przedni pierścień powoduje zmniejszanie się pierścienia tylnego i przyspieszenie jego ruchu. Z kolei prędkość indukowana przez tylny pierścień powoduje powiększenie średnicy pierścienia przedniego i jego spowolnienie. Powoduje to przejście pierścienia tylnego przez środek przedniego. Następnie role się zamieniają i proces może się powtórzyć. Gra wirów zależy w dużym stopniu od warunków początkowych pierścieni, ich cyrkulacji i wzajemnego położenia. Przy źle dobranych warunkach początkowych może nie zachodzić.

W przedstawionym przykładzie rozkład wirowości w przekroju pierścieni miał postać:

$$\omega(r) = \omega_0 e^{-\frac{r^2}{r_0^2}} \tag{7.1}$$

Cyrkulacja dla tego przypadku wynosiła $\Gamma = 1.0$. Wykorzystując wzór

$$\Gamma = \int_0^{r_0} \omega(r) 2\pi r \; dr \tag{7.2}$$

oraz parametry geometryczne $r_0 = 0.15$ i $R_0 = 1.5$ można wyliczyć wartość $\omega_0 = 22.3804$. Początkowa odległość pomiędzy pierścieniami wynosiła $h = 0.9$.

Obszar obliczeniowy miał rozmiar $[20 \times 20 \times 20]$, a rozpięta na nim siatka numeryczna miała gęstość $256 \times 256 \times 256$ węzłów. We wszystkich kierunkach były zadane okresowe warunki brzegowe.

Przepływ nielepki

Najpierw przeprowadzono obliczenia dla płynu nielepkiego. Ponieważ nie dysponowano wynikami eksperymentalnymi dla takiego przypadku, poprawność uzyskanych wyników będzie się opierała o spełnienie twierdzenia Helmholtza 3.1, w myśl którego w przepływie nielepkim linie wirowe nie mogą się ani rozerwać ani połączyć. Pierścienie nie mogą się zatem połączyć ze sobą.

Wyniki obliczeń przedstawione zostały na rys. 7.1.

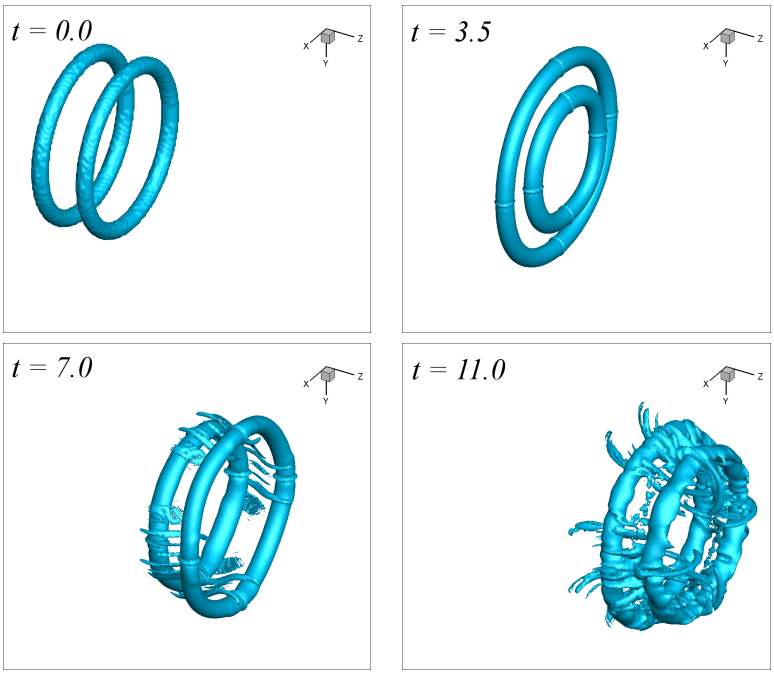
Jak widać w trakcie symulacji doszło do dwóch następujących po sobie przejść pierścieni. Widać zaburzenie kształtów pierścieni, które jednak nie prowadzi do połączenia się ich ze sobą. Podobne zaburzenia były widoczne w testach z przemieszczającym się pojedynczym pierścieniem przedstawione w rozdziale 4. Otrzymane wyniki wskazują na poprawności kodu numerycznego.

Przepływ lepki, $Re_\Gamma = 1000$

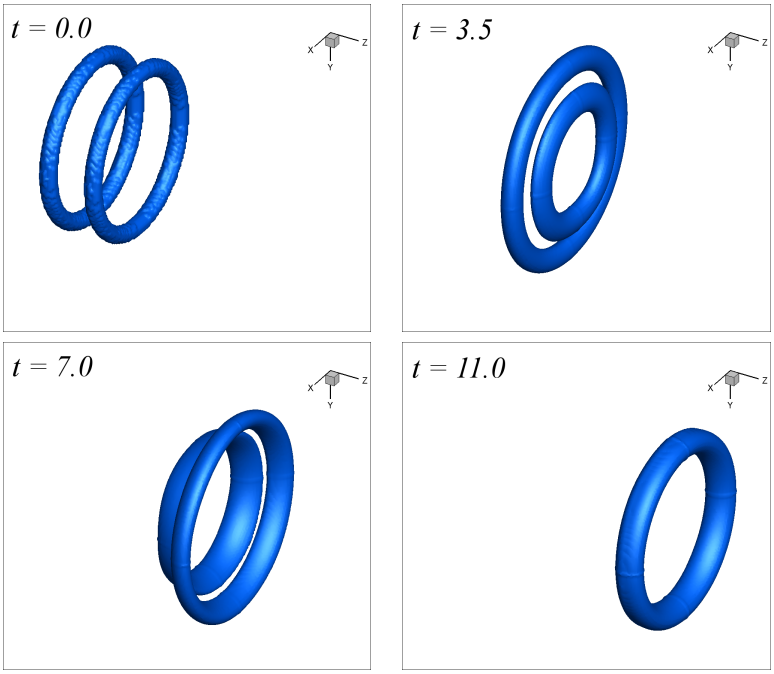
Przypadek ten pozwolił w pełni sprawdzić poprawność implementacji metody VIC dla płynu lepkiego. Do obliczeń posłużył ten sam przykład (pierścienie miały takie same parametry jak w poprzednim przedstawionym przypadku) tylko z zadaną lepkością (rozwiązywane jest równanie dyfuzji). Współczynnik lepkości w tym wypadku wynosił $\nu = 0.001$. Liczba Reynoldsa zdefiniowana względem cyrkulacji pierścienia wynosiła:

$$Re_\Gamma = \frac{\Gamma}{\nu} = 1000. \tag{7.3}$$

Wyniki obliczeń widoczne są na rys. 7.2. W tym przypadku pierścienie wykonały jedno przejście a następnie połączyły się ze sobą tworząc jedną strukturę. Lepkości płynu powoduje wygładzanie powierzchni tworzących pierścień co jest zgodne z rolą jaką przypisuje się jej działaniu. Jest to wynik zbliżony do



Rysunek 7.1: Gra wirów w przepływie nielepkim. Izopowierzchnia $|\omega| = 0.2\omega_0$.



Rysunek 7.2: Gra wirów w przepływie lepkim. Izopowierzchnia $|\omega| = 0.15\omega_0$.

rezultatów eksperymentów opisanych w [68]. Lepkość płynu umożliwia również połączenie się obu pierścieni.

Wyniki numeryczne prezentowane były również m.in. w [64]. W tym wypadku wyniki symulacji przedstawione są za pomocą cząstek wirowych. Widoczne na nich jest dwukrotne przejście pierścieni przez siebie. Należy jednak zwrócić uwagę na fakt, że w płynie lepkim ewolucja w czasie markerów (np. cząstek barwnika) jest różna od ewolucji elementów wirowości. Cząstki pasywne nie przemieszczają się tak jak elementy wirowości. Dokładny opis tego zjawiska przedstawiony jest w kolejnym rozdziale. Zaprezentowane wyniki przemawiają za poprawnością obliczeń numerycznych.

Wyniki dla zjawiska gry wirów zostały opublikowane w artykułach autorskich [53, 61, 63, 55, 57].

7.2. Zmiana topologii linii wirowych, izopowierzchni wirowości i pasywnych markerów

Głównym źródłem wiedzy na temat wzajemnej interakcji struktur wirowych jest wizualizacja. Niestety uważa się, że nie jest ona w stanie właściwie oddać ewolucji struktur wirowych, zwłaszcza przy długim czasie obserwacji [74, 43]. Bezpośredni pomiar wirowości nie jest łatwy, w badaniach eksperymentalnych do śledzenia struktur wirowych wykorzystywane są pasywne markery takie jak dym czy barwnik. Interpretacja wyników otrzymanych takimi metodami wymaga ostrożności ponieważ pasywne markery ewoluują inaczej niż wirowość [43, 44]. Obecnie do wyznaczania pola wirowości stosowane są metody optyczne takie, jak np. PIV (*ang.* Particle Image Velocimetry), w których na podstawie przemieszczenia się pasywnych markerów wyznaczane jest komputerowo pole prędkości, a na jego podstawie, numerycznie, pole wirowości. Metody te jednak są obarczone błędem zarówno pomiarowym, jak i wykorzystywanej metody różniczkowania.

Rekonekcją wirowości będziemy nazywali zmianę topologii linii wirowych.

Topologia w odniesieniu do mechaniki płynów zajmuje się badaniem takich własności struktur wirowych w przepływach nielepkich, które pozwalają scharakteryzować przepływ tylko na podstawie geometrii (ułożenia) i wzajemnego położenia tych struktur oraz nie ulegają zmianie w czasie ewolucji płynu pomimo zmiany lub zaburzenia struktur [86]. Zgodnie z tw. Helmholtza 3.1 własności takie jak liczba pierścieni czy rurek wirowych w przepływie jest topologicznym niezmiennikiem przepływu. W topologicznej mechanice płynów określa się wielkości takie jak połączone (*ang.* linked vortices) i splątane wiry (*ang.* knotted vortices). Są to również topologiczne niezmienniki. Oznacza to, że jeżeli struktury były złączone lub splątane będą ewoluowały w czasie, zmieniając swój kształt ale zachowując rodzaj splątania lub połączenia [86]. Własności topologiczne płynów są zatem niezmiennikami przepływu a własności fizyczne wyrażone poprzez

własności topologiczne także są niezmienne w czasie ruchu płynu. Dobrym przykładem takiej własności jest skrętność wprowadzona przez Moffata w [76].

$$H = \int \mathbf{u} \cdot \boldsymbol{\omega} dV \tag{7.4}$$

który dla dwóch włókien wirowych o siłach κ_1 i κ_2 wynosi $\alpha\kappa_1\kappa_2$, gdzie α jest liczbą całkowitą zależną od stopnia połączenia tych włókien: $\alpha = 0$ jeżeli nie są one połączone oraz $\alpha \pm 1$ jeżeli są jednokrotnie połączone. Ponieważ α jest topologicznym niezmiennikiem to również skrętność jest niezmiennikiem przepływu. Każda odosobniona porcja wirowości (*ang.* patch of vorticity), otoczona przez powierzchnię materiału lub linie wirowe posiada niezmienną skrętność [14].

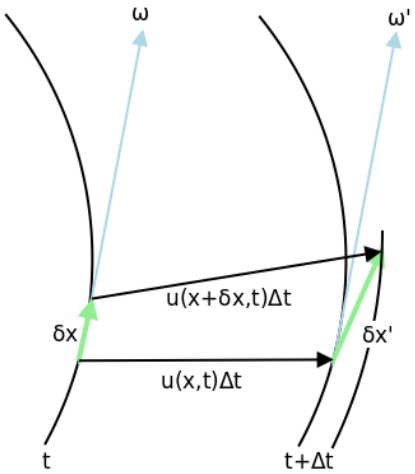
W płynie idealnym nie ma efektów lepkościowych co oznacza, że struktury nie mogą dyfundować lub swobodnie zanikać. Zmiana w układzie struktur spowodowana rekonekcją nie może zachodzić bez efektów lepkościowych. Topologia może się zmieniać kiedy pojawiają się efekty lepkościowe.

W większości badań eksperymentalnych zjawisko rekonekcji jest rozumiane jako zmiana topologii pasywnych markerów, które w płynie lepkim zachowują się inaczej niż pole wirowości. W symulacjach numerycznych natomiast rekonekcja jest opisywana poprzez izopowierchnie modułu pola wirowości. Obie te metody mogą jednak nie wskazywać jednoznacznie na zmianę topologii linii wirowych [44].

Można zatem wyróżnić trzy typy zmiany topologii [44]: skalarów (*ang.* scalar reconnection), wirów (*ang.* vortex reconnection) i linii wirowych (*ang.* vorticity reconnection). Pierwsza odnosi się do zmiany topologii pasywnych markerów, druga do zmiany topologii izopowierzchni modułu wirowości, a ostatnia do zmiany topologii linii wirowych. W przepływie nielepkim ostatnia z wymienionych rekonekcji jest niemożliwa z powodu twierdzeń Kelvina-Helmholtza 3.1. Jednakże pozostałe dwie mogą zachodzić [44].

Aby sprawdzić czy dana linia wirowa zmieniła swoją topologię należy śledzić jej położenie w czasie. W przepływie nielepkim jest to możliwe ponieważ elementy wirowe i elementy płynu poruszają się po tych samych trajektoriach. Wystarczy zatem śledzić elementy płynu (pasywne markery). Jednakże w płynie nielepkim, zgodnie z twierdzeniem Helmholtza 3.1, niemożliwa jest zmiana topologii linii wirowych, a tym samym ich rekonekcja. W płynie lepkim zaś dochodzi do rozróżnienia ruchu elementów płynu i elementów wirowości. Para elementów płynu znajdująca się na tej samej linii wirowej w pewnej chwili czasu nie musi się znaleźć na tej samej linii wirowej w innej chwili czasu.

Można przedstawić różnicę w ruchu elementów płynu i elementów wirowości w płynie lepkim. Weźmy dwa elementy płynu leżące na tej samej linii wirowej, w odległości $\delta\mathbf{x}$ od siebie w chwili czasu t. Wynika z tego, że $\delta\mathbf{x}||\boldsymbol{\omega}(\mathbf{x}, t)$ (rys. 7.2). Te dwa elementy płynu są unoszone przez przepływ, a ich nowe położenie $\delta\mathbf{x}'$ po czasie Δt można zapisać jako:



Rysunek 7.3: Różnica w przemieszczeniach pomiędzy elementami płynu a wirowością w przepływie lepkim.

$$\delta\mathbf{x}' = \delta\mathbf{x} + [\mathbf{u}(\mathbf{x}+\delta\mathbf{x},t) - \mathbf{u}(\mathbf{x},t)\Delta t = \delta\mathbf{x} + (\delta\mathbf{x}\cdot\nabla)\mathbf{u}(\mathbf{x},t)\Delta t. \tag{7.5}$$

Z kolei wirowość w chwili $t+\Delta t$ wyraża się następująco:

$$\begin{aligned}\boldsymbol{\omega}' &\equiv \boldsymbol{\omega}[\mathbf{x}+\mathbf{u}(\mathbf{x},t)\Delta t, t+\Delta t] \\ &= \boldsymbol{\omega}(\mathbf{x},t) + [\mathbf{u}(\mathbf{x},t)\cdot\nabla]\boldsymbol{\omega}(\mathbf{x},t)\Delta t + \frac{\partial\boldsymbol{\omega}(\mathbf{x},t)}{\partial t}\Delta t \\ &= \boldsymbol{\omega}(\mathbf{x},t) + [\boldsymbol{\omega}(\mathbf{x},t)\cdot\nabla]\mathbf{u}(\mathbf{x},t)\Delta t + \nu\nabla^2\boldsymbol{\omega}(\mathbf{x},t)\Delta t,\end{aligned} \tag{7.6}$$

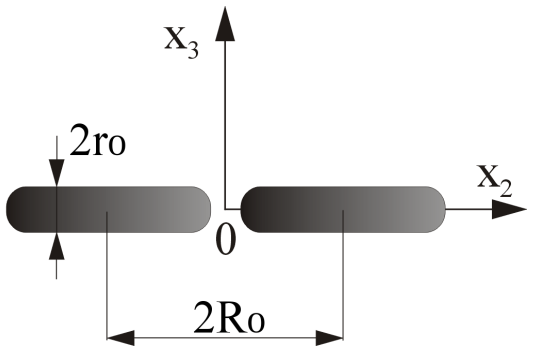
gdzie użyte zostało równanie (3.13). Równania (7.5) i (7.6) pokazują, że dla przepływu nielepkiego ewolucja w czasie wartości $\delta\mathbf{x}$ i $\boldsymbol{\omega}$ jest taka sama jeżeli tylko w pewnej chwili czasu $\delta\mathbf{x}||\boldsymbol{\omega}$. Zależność ta nie zachodzi w przepływie lepkim ($\nu > 0$). Składowa równoległa do wirowości członu lepkościowego $\nu(\nabla^2\boldsymbol{\omega})_{||}$ reprezentuje różnicę w prędkości rozciągania (*ang.* streching rate) linii wirowych i linii prądu. Składowa prostopadła $\nu(\nabla^2\boldsymbol{\omega})_{\perp}$ natomiast reprezentuje prędkość odchylenia (*ang.* rate of deviation) pomiędzy tymi dwoma rodzajami linii. Z tego powodu w momencie, kiedy składowa prostopadła jest różna od zera linie wirowe nie pokrywają się liniami prądu cząstek płynu.

Rekonekcja struktur wirowych jest tematem popularnym i ważnym w mechanice płynów. Zajmowało się nią wielu autorów. Przykładowymi zagadnieniami mogą być rekonekcja identycznych, prostopadłych rurek wirowych przesuniętych względem siebie [11, 107], rekonekcja dwóch antyrównoległych rurek wirowych [40, 72, 75, 36, 99], rekonekcja dwóch pierścieni wirowych ustawionych obok siebie [43, 45] czy zderzenie czołowe dwóch pierścieni wirowych [69, 68].

Przypadki te zostaną przedstawione w kolejnych sekcjach tego rozdziału. Na podstawie tych przykładów zostanie opisany proces rekonekcji z podziałem na etapy. Dla zrozumienia podstawowych mechanizmów wzajemnych oddziaływań pomiędzy strukturami wirowymi najlepiej jest badać możliwie najprostsze przypadki, w których one występują. Dlatego do testów zostały wybrane proste i dobrze udokumentowane w literaturze przypadki rekonekcji.

7.2.1. Rekonekcja dwóch pierścieni wirowych ($Re_\Gamma = 730$)

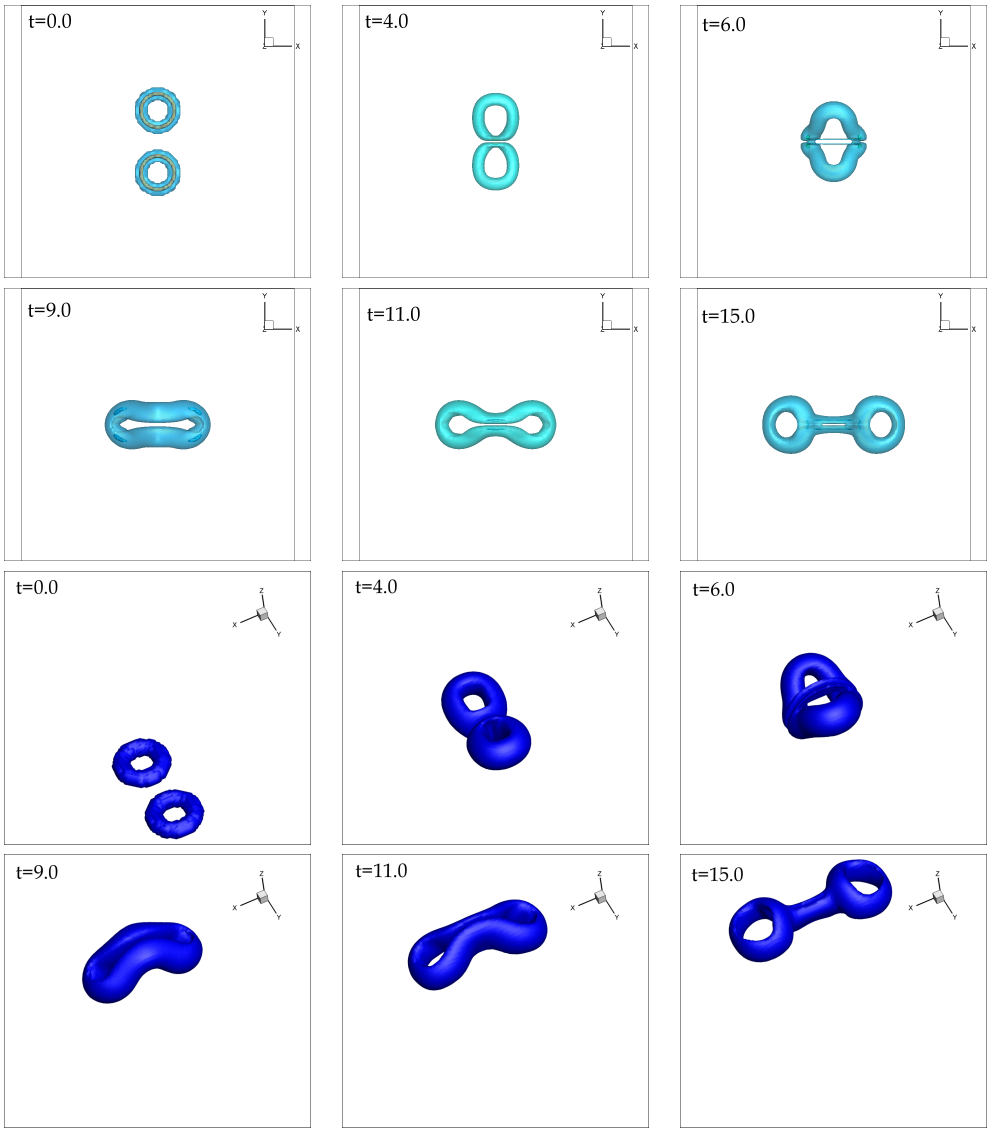
Odkrycie uporządkowanych koherentnych struktur wirowych rozpoczęło systematyczne badania nad rekonekcją struktur wirowych. Specjalny nacisk został położony na budowę topologiczną obszarów o wysokiej, skoncentrowanej wirowości. Jednym z najprostszych i najbardziej fundamentalnych eksperymentów była rekonekcja dwóch pierścieni wirowych o takich samych cyrkulacjach ustawionych początkowo obok siebie [79, 90]. W eksperymentach, w których wizualizowano przepływ przy pomocy dymu lub barwnika, zaobserwowano dwie następujące po sobie rekonekcje [67]. Dwa pierścienie wyrzucane z dysz położonych obok siebie łączą się w jedną wydłużoną strukturę wirową, aby po skomplikowanym trójwymiarowym ruchu rozłączyć się na dwa pierścienie wirowe ułożone prostopadle do pierwotnych pierścieni.



Rysunek 7.4: Początkowe położenie dwóch pierścieni wirowych widzianych z boku.

Początkowe położenie dwóch identycznych pierścieni wirowych jest widoczne na rys. 7.4. Dane geometryczne pochodzą z pracy [45]. Odległość pomiędzy środkami pierścieni wynosiła $D = 3.65$. Promień pierścienia wynosił $R_0 = 0.982$, a promień rdzenia $r_0 = 0.393$. Wewnątrz rdzenia założono normalny rozkład wirowości zgodny ze wzorem (7.7) gdzie $\omega_0 = 23.8$. Zgodnie ze wzorami (7.8) i (7.9) wartości cyrkulacji i liczby Reynoldsa wynosiły odpowiednio $\Gamma = 7.3$ i $Re_\Gamma = 730$. Obszar obliczeniowy miał rozmiar $[4\pi \times 4\pi \times 4\pi]$. Liczba węzłów w kierunku x, y i z wynosiła $N = 128$. Użyto kroku czasowego $\Delta t = 0.01$.

Wyniki przestawione są na rys. 7.5. Pierścienie poruszają się pionowo (oś z) na skutek samoindukcji pola prędkości. Wzajemna indukcja powoduje, że zaczy-

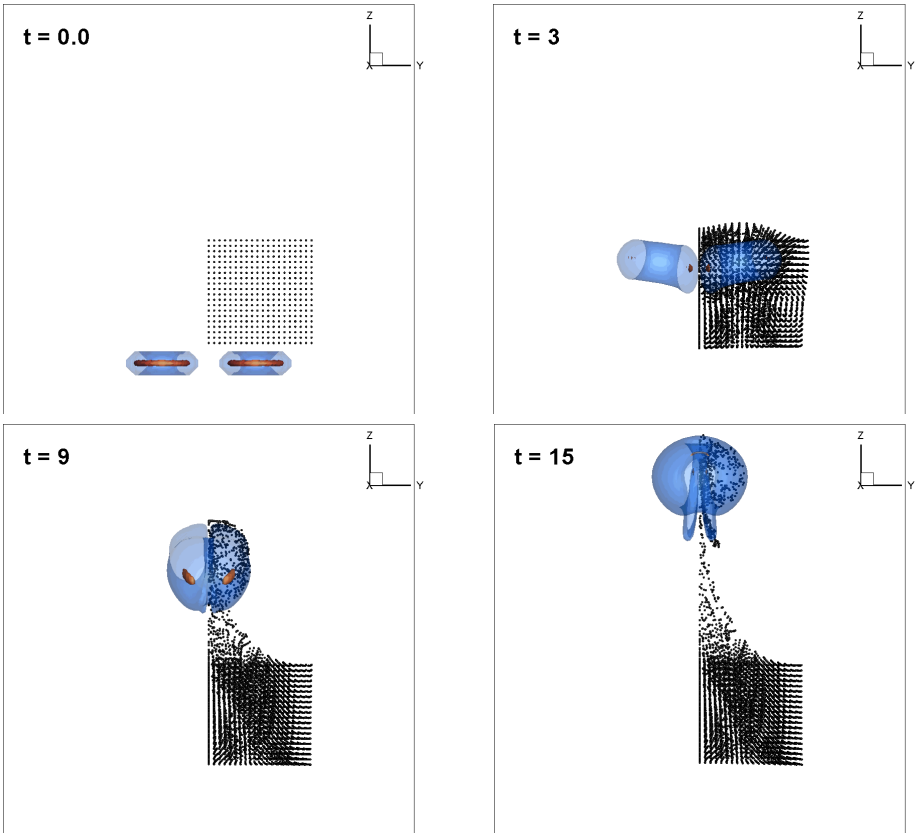


Rysunek 7.5: Widok z góry (na górze) oraz pod kątem (na dole) izopowierzchni wirowości $|\omega|$. Widoczne poziomy to 10% and 90% chwilowej wartości ω_{max} dla przypadku rekonekcji dwóch pierścieni wirowych.

nają się przekręcać w stronę płaszczyzny (x, y). Do zderzenia dochodzi w chwili $t = 4.0$. Rurki wirowe są w konfiguracji równoległej z przeciwnie skierowanymi wektorami prędkości (APV - *ang.* Anti-Parallel Vortices) w miejscu kontaktu i linie wirowe przeciwnych znaków są znoszone przez efekty lepkościowe (*ang.* viscous cross-diffusion). W tym samym czasie linie wirowe wewnątrz pierścieni są łączone i tworzą jeden duży pierścień wirowy. Ponieważ znoszenie wirowości nie zostaje dokończone pozostają widoczne nici ($t = 6.0$). Samoindukcja nowego pier-

ścienia powoduje jego rozciąganie w osi x i kurczenie wzdłuż osi y. Krzywizna na końcach pierścienia robi się większa i indukuje większą prędkość co powoduje, że te części pierścienia zaczynają się poruszać szybciej. Środkowa część staje się coraz cieńsza i w chwili $t = 11.0$ dwie części pierścienia wchodzą ze sobą w kontakt, rozpoczynając drugą rekonekcję. Tworzone są dwa pierścienie wirowe ułożone prostopadle do osi pierwotnych pierścieni. Pierścienie zaczynają oddalać się od siebie pozostawiając ślad nazywany nogami (*ang.* legs).

Aby sprawdzić jak proces rekonekcji wpływa na płyn w otoczeniu miejsca zachodzenia tego zjawiska, w przepływie umieszczono grupę pasywnych markerów. Proces mieszania przedstawiony jest na rys. 7.6. Widać, że poruszone zostały tylko markery będące na drodze pierścieni wirowych. Część z nich została uniesiona i wymieszana. Duża część markerów mimo znacznej bliskości obszaru rekonekcji pozostała nieporuszona.



Rysunek 7.6: Ruch pasywnych markerów wywołany polem prędkości wywołanym rekonekcją dwóch pierścieni wirowych. Widok z boku.

Pomimo bliskości procesu rekonekcji jedynie niewielka liczba markerów została uniesiona wraz ze strukturami wirowymi. Większość cząstek pasywnych pozostało w swoich początkowych położeniach. Tylko cząstki z najbliższego oto-

czenia procesu rekonekcji, które zostały zabrane przez pierścienie doświadczyły intensywnego mieszania.

7.2.2. Rekonekcja identycznych, prostopadłych rurek wirowych przesuniętych względem siebie ($Re_\Gamma = 1403$)

Jeden z pierwszych modeli zakładał, że rekonekcja ma miejsce w wyniku znoszenia się wirowości o przeciwnych znakach w miejscu spotkania dwóch zachodzących na siebie pod wpływem wzajemnej indukcji lub przepływu zewnętrznego rurek wirowych. W pracy [87] Saffman podał model, w którym opisuje rozłączanie i łączenie się linii wirowych w momencie spotkania się rurek wirowych o równych co do wartości i przeciwnie skierowanych wirowościach. W modelu tym efekty lepkościowe znoszą wirowość w miejscu zetknięcia się rurek wirowych. Osłabienie siły odśrodkowej w rdzeniu wirowym prowadzi do lokalnego wzrostu ciśnienia, które przyspiesza płyn w rdzeniu w kierunku osiowym i unosi wirowość z dala od miejsca kontaktu co powoduje widoczne połączenie. W rzeczywistości ma jednak miejsce bardziej skomplikowane przełączanie rurek wirowych, w czasie którego linie wirowe zostają skręcone i splątane.

W trakcie różnych badań odkryto, że są pewne wspólne mechanizmy w procesie rekonekcji dwóch rurek wirowych, które mają taką samą cyrkulację, niezależnie od początkowego położenia. Typowy scenariusz zdarzeń jest następujący. Zbliżające się do siebie rurki wirowe są unoszone przez prędkość wytworzoną przez samoindukcję jak i indukcję drugiej struktury. Mają one tendencję do równoległego ułożenia w przestrzeni z przeciwnie skierowanymi wektorami wirowości (APV). Efekt ten został bardzo dobrze opisany w [93]. W miarę zbliżania się struktur ich rdzenie wirowe spłaszczają się i deformują do tzw. struktury głowa-ogon (*ang.* head-tail) [44]. Następnie znoszenie się przeciwnie skierowanej wirowości w strefie kontaktu (*ang.* cancelling) rozpoczyna proces rekonekcji. Unoszone przez skomplikowane trójwymiarowe pole prędkości linie wirowe są przełączane (*ang.* cross-linking) lub mostkowane (*ang.* bridging). Powstający w tym procesie wektor wirowości jest prostopadły do początkowego kierunku rurek wirowych. Intensywność nowej wirowości jest wzmacniana przez rozciąganie i w pewnym momencie przewyższa intensywność pierwotnych rurek. Powoduje to zmianę globalnej topologii rurek wirowych. Opisany proces jest dobrze widoczny w przypadku rekonekcji dwóch prostopadłych rurek wirowych. Ten przypadek został opisany w [107] jako prototypowa geometria procesu rekonekcji. Osie rurek są do siebie prostopadłe i odsunięte względem siebie. Położenie początkowe rurek jest widoczne w ramce $t = 0.0$ na rys. 7.7. Do walidacji wyników posłużyły dane zamieszczone w [107]. W pracy tej wykorzystano metodę spektralną, bardzo dokładną metodę rozwiązywania równań ruchu.

Początkowy rozkład wirowości zadany był wzorem:

$$\omega(r) = \omega_0 e^{-\frac{r^2}{r_0^2}}. \tag{7.7}$$

Wartości cyrkulacji i liczby Reynoldsa wyliczane były ze wzorów odpowiednio

$$\Gamma = \int_0^{r_0} \omega(r) 2\pi r \; dr, \tag{7.8}$$

$$Re_\Gamma = \frac{\Gamma}{\nu}. \tag{7.9}$$

Dla badanego przypadku użyte zostały następujące wartości: $\omega_0 = 20$, $r_0 = 3^{-1/2}$, co w wyniku daje $\Gamma = 14.73$, $Re_\Gamma = 1403$. Obszar obliczeniowy był sześcianem o wymiarach $[2\pi \times 2\pi \times 2\pi]$ i siatką $h_x = h_y = h_z = 2\pi/N$, gdzie $N = 256$. Krok czasowy wynosił $\Delta t = 0.001$.

Na rys. 7.7 przedstawiona jest rekonekcja dwóch prostopadłych rurek wirowych przesuniętych względem siebie. W lewej kolumnie przedstawione są izopowierzchnie modułu wirowości dla $|\omega| = 12$, a w prawej linie wirowe. Wyniki bardzo dobrze odpowiadają wynikom przedstawionym na rys. 3 zamieszczonym w pracy [107], które otrzymano używając metody spektralnej. Świadczy to o poprawności implementacji algorytmu i dobrej dokładności metody VIC.

Ewolucja może zostać podzielona na trzy zachodzące na siebie etapy:

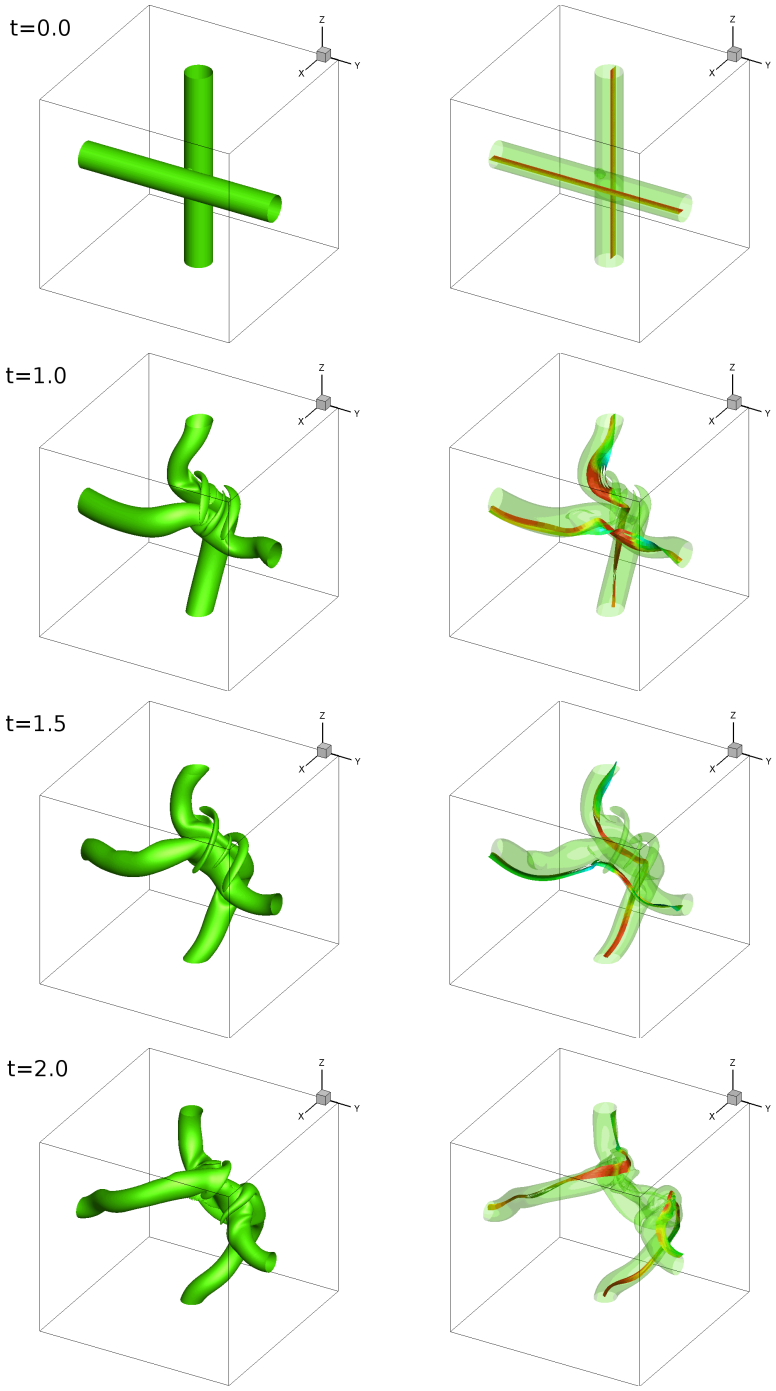
1. $0 \leqslant 1.0$: Konwekcja pierwotnych rurek wirowych do konfiguracji APV. Wytworzenie się wtórnych wirów spinkowych (*ang.* hairpin vortex),

2. $1.0 < t < 2.0$: Powstawanie i intensyfikacja mostów, ruch obszaru kontaktu w dół i w lewo. Jednocześnie splątanie i wiązanie słabszych wirów spinkowych. Spłaszczenie strefy kontaktu.

3. $2.0 < t < 3.0$: Gwałtowna dyssypacja lub osłabienie pierwotnej poruszającej się w dół pary rurek wirowych. Jednoczenie ruch do góry i na zewnątrz oraz skręcenie wirów wtórnych.

W momencie $t = 1.0$ widoczne jest wygięcie rurek wirowych wynikające z wyindukowanego pola prędkości. Jednocześnie pojawiają się wtórne stryktury palczaste będące niemal prostopadłe do osi pierwotnych rurek. W momencie $t = 2.0$ widać spłaszczenie strefy kontaktu. W $t = 3.0$ widzimy wtórne struktury wirowe połączone nicią (*ang.* thread) pozostałą po rekonekcji.

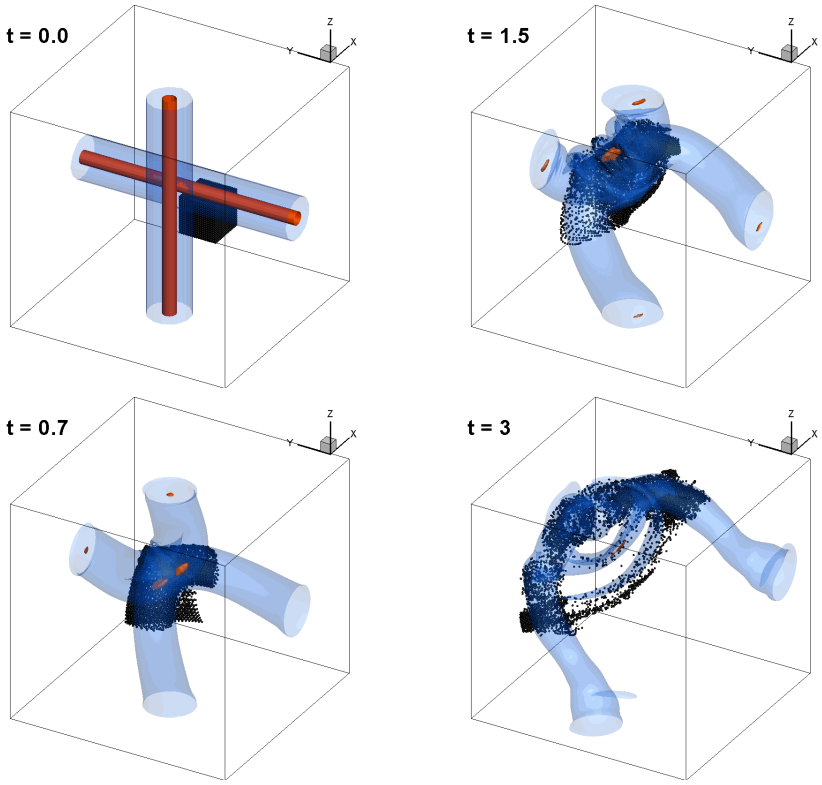
Aby pokazać, że w tym przypadku zachodzi nie tylko rekonekcja izopowierzchnii wirowości ale również rekonekcja linii wirowych na prawej stronie rys. 7.7 zostały wykreślone linie wirowe. Punktami początkowymi tych linii były dwa, leżące na brzegu obszaru, odcinki o wymiarze 0.2 prostopadłe do siebie i zgodne z kierunkami osi współrzędnych, których punktem przecięcia był punkt maksymalnej wartości modułu wirowości na danym brzegu. Linie widoczne na rys. 7.7 zostały



Rysunek 7.7: Obraz izopowierzchni $|\omega| = 0.6\omega_0$ (po lewej) oraz linii wirowych (po prawej) ewolucji dwóch identycznych, prostopadłych rurek wirowych przesuniętych względem siebie. $Re_\Gamma = 1403$

rozpoczęte na lewym i dolnym brzegu obszaru. Na obrazkach są one wyświetlane za pomocą wstęg pokolorowanych wartością wirowości. Dzięki takiemu ułożeniu dobrze widoczne jest zarówno położenie, jak i skręcenie linii wirowych.

Na pierwszym obrazku widać początkowy układ rurek wirowych. W czasie obliczeń zaczynają się one skręcać i przechodzić w położenie antyrównoległe. Na ostatnim obrazku widać zmianę topologii wyświetlanych linii wirowych. Linie, które początkowo przebiegały z dolnego brzegu obszaru obliczeniowego do górnego oraz z lewego do prawego teraz przebiegają z lewej strony do góry oraz z dolnego brzegu do prawego. W obszarze, w którym nastąpiła rekonekcja, pozostała pewna porcja wirowości (tzw. nici *ang.* thread) co świadczy o tym, że nie wszystkie linie wirowe zmieniły swoją topologię. Zgodnie z przytoczonym wcześniej opisem śledzenie linii wirowych jest najpewniejszym sposobem na stwierdzenie zajścia procesu rekonekcji i jest ono możliwe tylko dzięki wykorzystaniu symulacji numerycznych.



Rysunek 7.8: Ruch pasywnych markerów unoszonych przez pole prędkości wytworzone w czasie rekonekcji dwóch rurek wirowych. Początkowo markery ułożone są w kształcie sześcianu ze środkiem w miejscu kontaktu rurek. Na obrazku widoczna 1/8 wszystkich markerów.

Na rys. 7.8 przedstawiony został ruch pasywnych markerów unoszonych przez

pole prędkości wytworzone w czasie rekonekcji dwóch rurek wirowych. Ponieważ proces rekonekcji zachodzi w tym przykładzie bardzo gwałtownie to szybko widoczna jest utrata symetrii ułożenia markerów względem siebie oraz intensywne mieszczanie tych cząstek. Pomimo, że część pasywnych markerów znajdowała się w chwili początkowej poza obszarem rurek wirowych, to w trakcie procesu rekonekcji zostały wciągnięte w obszar rekonekcji. Pokazuje to m.in., że w płynie lepkim cząstki wirowe nie poruszają się tak, jak cząstki materialne płynu [43].

Wyniki przedstawiające symulację zjawiska rekonekcji dwóch prostopadłych rurek wirowych zostały opublikowane w artykułach autorskich [61, 63, 57].

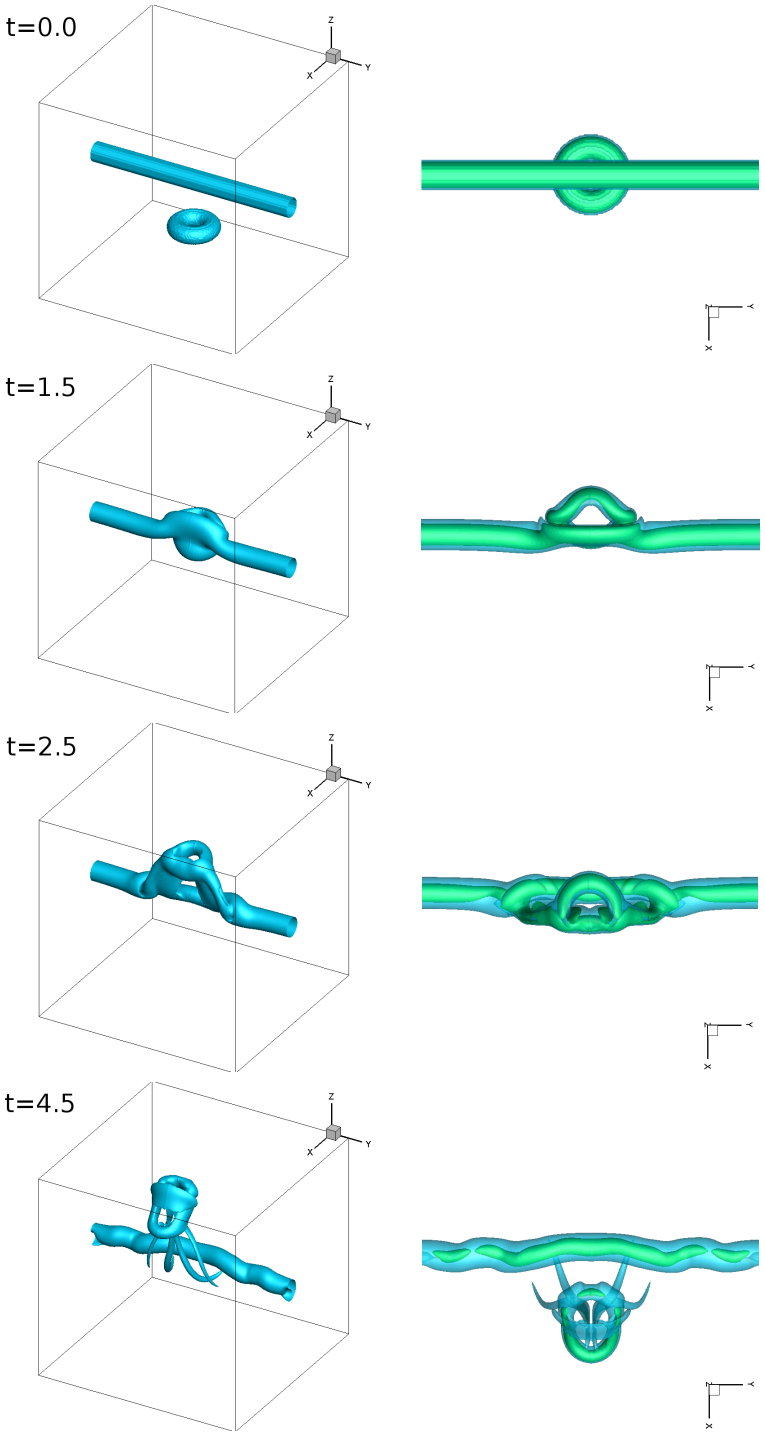
7.2.3. Zderzenie rurki wirowej i pierścienia

Ciekawym przypadkiem przy omawianiu zmiany topologii linii wirowych jest zderzenie się ze sobą pierścienia wirowego i rurki wirowej. Wyniki symulacji dla dwóch różnych liczb Reynoldsa przedstawione są na rys. 7.9 i rys. 7.10.

W obu wypadkach rurka wirowa i pierścień wirowy miały takie same promienie rdzeni $r_0 = 0.5$ oraz taki sam promień pierścienia $R_0 = 1.5$. Odległość pomiędzy pierścieniem a rurką wynosiła $d = 3.0$. W pierwszym przypadku liczba Reynoldsa wynosiła $Re_\Gamma = 1000$ ($|\omega_0| = 20$) a w drugim $Re_\Gamma = 2000$ ($|\omega_0| = 40$). Obszar obliczeniowy miał wymiary $4\pi \times 4\pi \times 4\pi$ oraz $256 \times 256 \times 256$ węzłów. Na wszystkich brzegach zadane były okresowe warunki brzegowe.

Na rys. 7.9 widoczny jest proces rekonekcji dla przypadku $Re_\Gamma = 1000$ widziany pod kątem (lewa strona) oraz z góry (prawa strona). Pierścień wirowy został tak umieszczony aby pod wpływem samoindukcji zbliżał się do rurki. W trakcie zbliżania się pierścień zaczyna być obracany wokół osi rurki wirowej z powodu wytworzonego przez nią pola prędkości. Widoczne jest to szczególnie dobrze na widoku z góry. Do zderzenia tych struktur wirowych dochodzi zatem pod innym kątem niż zostały one ułożone na początku obliczeń. Pierścień wirowy przechodzi przez rurkę (trzeci obraz), a następnie oddala się od niej w kierunku innym niż początkowy (czwarty obraz) zostawiając za sobą porcje wirowości (nici wirowe *ang.* threads). W trakcie opisanego procesu rurka wirowa prawie nie zmieniła swojego położenia. Przejście pierścienia wywołało falowanie rurki. Widoczne pomiędzy strukturami po zakończonym procesie porcje wirowości mogą wskazywać, że miała miejsce zmiana topologii linii wirowych. Obserwując tylko obrazy izopowierzchni wirowości nie można wyciągnąć takich wniosków.

Na rys. 7.10 przedstawiona jest rekonekcja dla przypadku $Re_\Gamma = 2000$. Dane zostały przedstawione za pomocą izopowierzchnii wirowości (lewa strona) oraz linii wirowych (prawa strona). Punktami startowymi do wykreślenia linii wirowych były punkty maksymalnej wartości wirowości na lewej i dolnej granicy obszaru obliczeniowego. Dodatkowo punkty te były środkami dwóch odcinków o wymiarze 0.2 prostopadłych do siebie i zgodnych z kierunkami osi współrzędnych z których również były rozpoczynane linie wirowe. Linie wirowe są na obrazkach wyświe-



Rysunek 7.9: Obraz izopowierzchni $|\omega| = 0.25\omega_0$ oraz $|\omega| = 0.50\omega_0$ (tylko prawa strona) dla zderzenia rurki wirowej z pierścieniem wirowym. $Re_\Gamma = 1000$

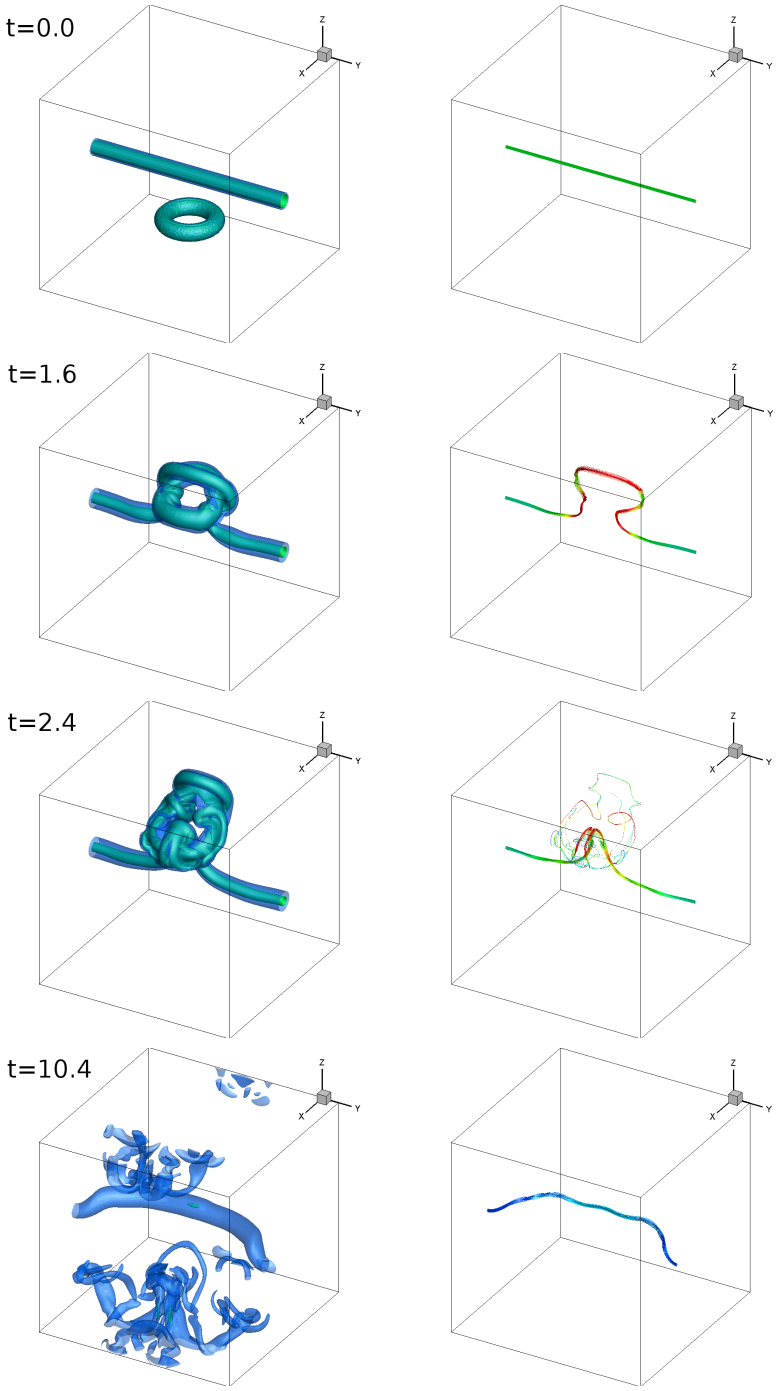
tlane za pomocą wstęg pokolorowanych wartością wirowości. Dzięki takiemu ułożeniu dobrze widoczne jest zarówno położenie, jak i skręcenie linii wirowych.

Pomimo większej liczby Reynoldsa przedstawiony proces przechodzi przez te same etapy co w poprzednim przypadku. Występuje zbliżenie się pierścienia spowodowane samoindukcją oraz obrót pierścienia spowodowany indukcją rurki. Sam proces przejścia pierścienia przez rurkę jest bardziej dynamiczny. Widać jak część rurki wirowej jest owijana wokół pierścienia (drugi i trzeci obraz). Proces ten jest także widoczny na obrazie linii wirowych, które są wyciągane do góry przez pierścień. Dynamika tego procesu powoduje, że w dalszym etapie pierścień wirowy jest rozrywany na mniejsze struktury wirowe. Na obrazie linii wirowych widać, że linie należące do rurki wirowej ponownie się połączyły ze sobą jednocześnie oddając pewną porcję wirowości pierścieniowi. Ponownie na ostatnim obrazie widać rurkę wirową tym razem mocniej zaburzoną, ale w kształcie zbliżonym do początkowego.

Pokazuje to, że w trakcie zderzenia pierścienia wirowego z rurką dochodzi do procesu zmiany topologii linii wirowych. Cześć wirowości znajdującej się początkowo w rurce wirowej zostaje oddana do pierścienia i razem z nim kontynuuje ruch. Po przejściu przez rurkę za pierścieniem pojawiają się smugi (nici) wirowe, obecne także w innych przedstawionych przypadkach, w których zaszła zmiana topologii linii wirowych.

7.2.4. Rekonekcja dwóch antyrównoległych rurek wirowych ($Re_\Gamma = 1003$)

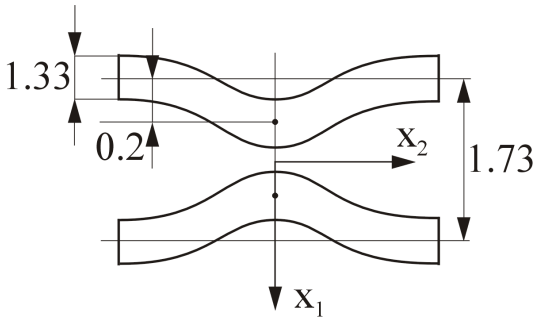
Bardziej szczegółowy opis samego procesu mostowania pojawia się w pracy [44]. W momencie kontaktu dwóch rurek wirowych ich linie wirowe będące najbliżej drugiej struktury znoszą się tworząc obszar interakcji. Jednocześnie ich pozostałe części łączą się ze swoimi odpowiednikami w przeciwnej rurce wirowej na końcach strefy interakcji. Małe porcje o wysokiej wirowości są wyrzucane z głównych rurek wirowych. W czasie ewolucji coraz bardziej się one wydłużają aż do momentu, połączenia się z porcjami wyrzuconymi z drugiej struktury wirowej, tworząc mosty (*ang.* bridges). Ich grubość rośnie w czasie co wskazuje na wzrost wartości wirowości wewnątrz nich. Proces ten postępuje do momentu kiedy grubość tych porcji jest porównywalna z grubością pierwotnych rurek. Wewnątrz mostów linie wirowe są wyciągane w postaci wirów spinkowych (*ang.* hairpin vortex). Pierwotne i wtórne linie wirowe są skręcane wokół siebie wewnątrz rdzenia wirowego a prędkość kątowa tego obrotu jest mniejsza w pobliżu miejsca interakcji z powodu spłaszczenia rdzenia wirowego, a większa na końcach strefy interakcji. W czasie rekonekcji strefa interakcji jest wyginana w kierunku prostopadłym do płaszczyzny początkowej rurek wirowych co sprawia, że znoszenie się wirowości jest coraz słabsze. Proces mostowania został zauważony w interakcji dwóch antyrównoległych rurek wirowych z zaburzeniem sinusoidalnym [74] i interakcji dwóch



Rysunek 7.10: Obraz izopowierzchni $|\omega| = 0.25\omega_0$ i $|\omega| = 0.50\omega_0$ (lewa strona) oraz linii wirowych (prawa strona) dla zderzenia rurki wirowej z pierścieniem wirowym. $Re_\Gamma = 2000$

pierścieni wirowych [45]. W tych przypadkach nie ma drastycznej zmiany kształtu rurek wirowych przed procesem rekonekcji. Bardziej intensywne procesy mostowania widoczne są w przypadku niesymetrycznych warunków początkowych, jak np. dla dwóch prostopadłych rurek wirowych [107, 42].

Proces ten jest dobrze widoczny w przypadku rekonekcji dwóch antyrównoległych rurek wirowych. Wymiary geometryczne do tego przypadku zaczerpnięto z pracy [74] i są przedstawione na rys. 7.11.

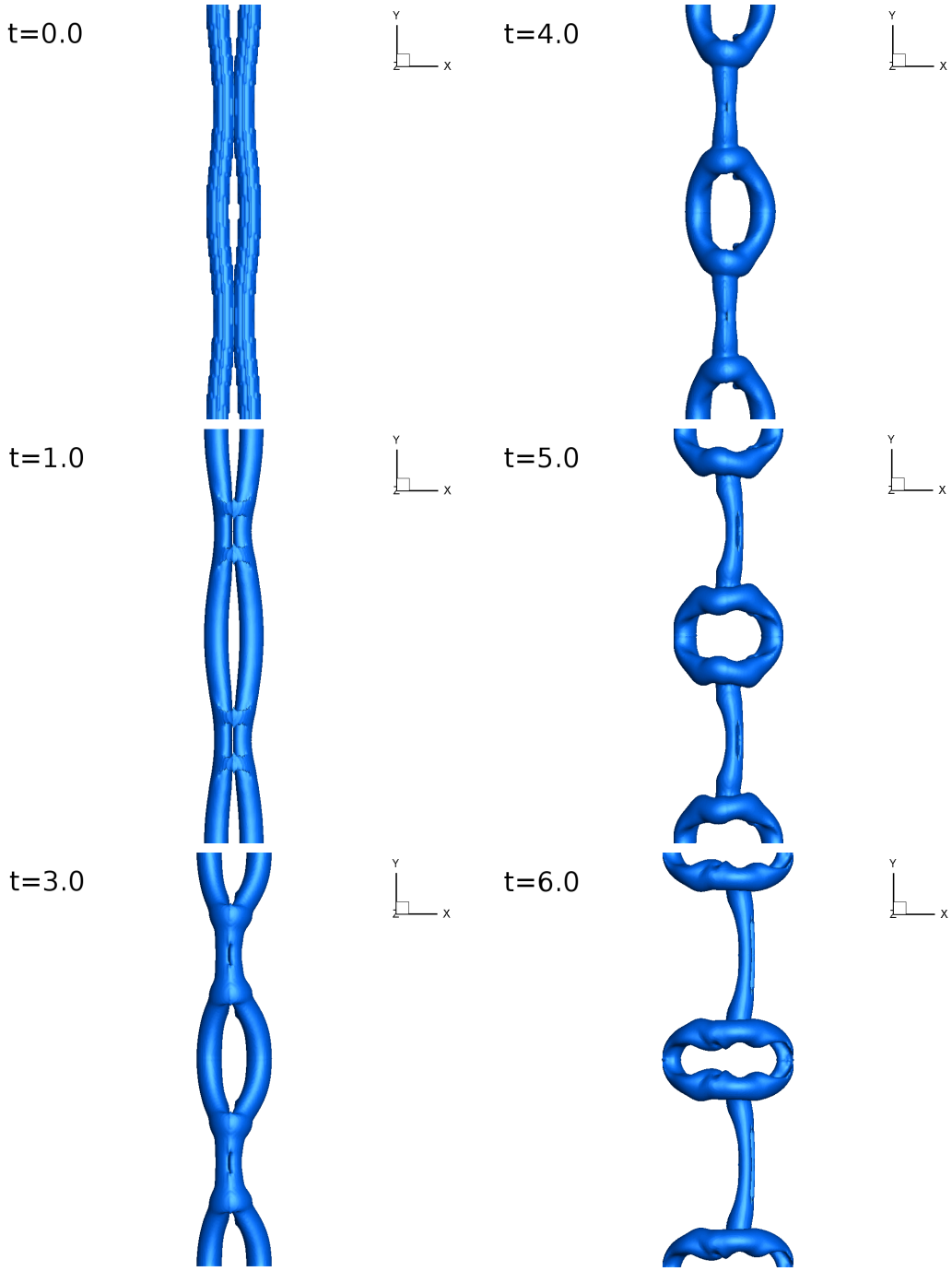


Rysunek 7.11: Warunki początkowe dla przypadku równoległych rurek wirowych z symetrycznym zaburzeniem.

Początkowy rozkład wirowości w rdzeniu opisany był wzorem (7.7), z wartościami $r_0 = 0.666$, $\omega_0 = 20.0$. Zgodnie ze wzorami (7.8) i (7.9) cyrkulacja i liczba Reynoldsa dla tego testu wynosiły odpowiednio: $\Gamma = 17.65$, $Re_\Gamma = 1003$. Obszar obliczeniowy stanowił sześcian o wymiarach $[8\pi \times 8\pi \times 8\pi]$ z okresowymi warunkami brzegowymi we wszystkich kierunkach. Liczba węzłów wynosiła $256 \times 256 \times 256$. Do obliczeń przyjęto krok czasowy $\Delta t = 0.01$. W obszarze zostały umieszczone rurki z dwoma sinusoidalnymi zaburzeniami, aby dokładnie oddać struktury tworzące się między nimi. Jakościowa zgodność z wynikami przedstawionymi w pracy [74] jest bardzo dobra.

Interakcja rurek wirowych przedstawiona na rys. 7.12 przechodzi przez klasyczne etapy rekonekcji wymienione przez innych autorów (np. [74], [44]). Rurki wirowe zbliżają się do siebie i zderzają się w $t = 1.0$. W miarę zbliżania się wirów do siebie ich rdzenie spłaszczają się, stają się coraz bardziej zdeformowane i przekształcają się w strukturę głowa - ogon (*ang.* head-tail). Ten etap charakteryzuje się aktywną dyfuzją (*ang.* cross-diffusion), której towarzyszy mostowanie pomiędzy dwoma strukturami wirowymi Wytworzone mosty pod wpływem wzajemnej indukcji zaczynają się poruszać w górę i oddalać od siebie. Z powodu tego ruchu efekty lepkościowe pomiędzy dwoma strukturami nie mają okazji do całkowitej anihilacji wirowości. Pozostałości wirowości, zwane nićmi (*ang.* threads), zaczynają się wyginać w górę z powodu prędkości indukowanej przez nowe struktury wirowe.

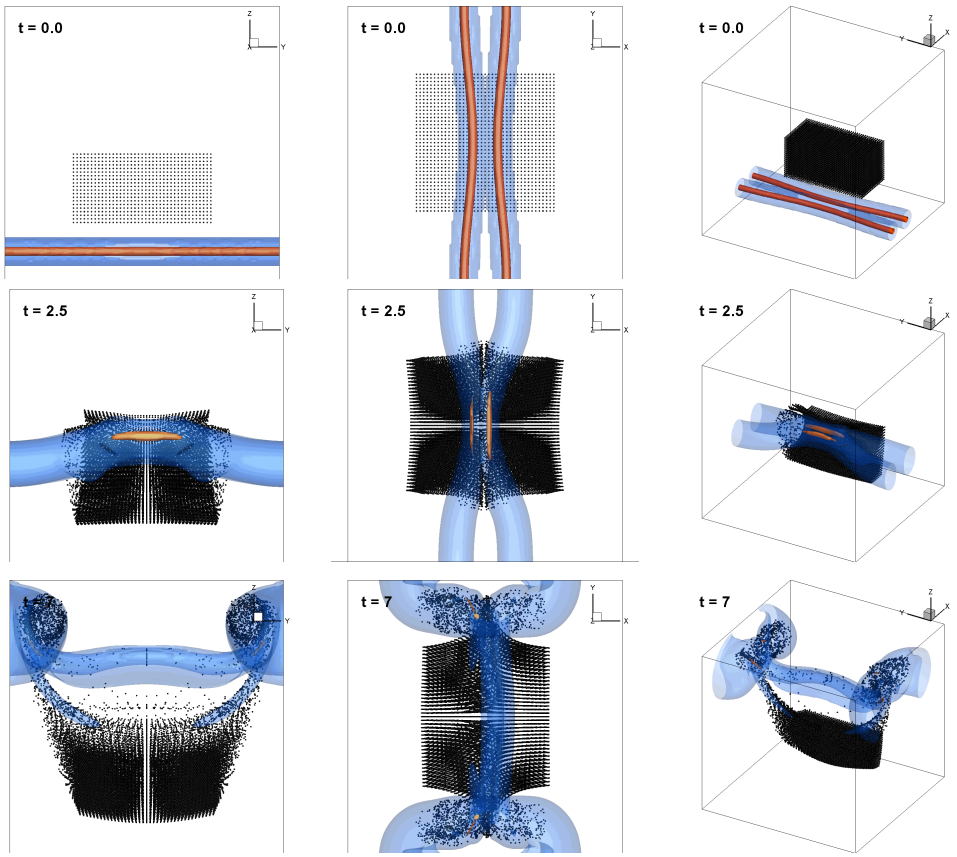
Mieszanie płynu otaczającego rurki wirowe biorące udział w procesie reko-



Rysunek 7.12: Obraz izopowierzchni $|\omega| = 0.15\omega_0$ dla ewolucji dwóch antyrównoległych rurek wirowych.

nekcji jest przedstawione za pomocą pasywnych markerów na rys. 7.13. Widać, że proces rekonekcji zintensyfikował proces mieszania w pewnym obszarze prze-

pływu. Markery leżące w środkowej części obszaru przepływu, w której doszło do rekonekcji, zostały uniesione przez przepływ i wymieszane. Z kolei cząstki pasywne leżące w obszarze przez który nie przechodziły rurki wirowe pozostały praktycznie nieruszone. Widać ograniczony zasięg działania procesu rekonekcji.



Rysunek 7.13: Ruch pasywnych markerów wywołany polem prędkości powstałym w czasie rekonekcji dwóch rurek wirowych widziany pod różnymi kątami. Dla lepszej wizualizacji w prawej kolumnie widoczna jest tylko połowa pasywnych markerów.

Wyniki przedstawiające symulację zjawiska rekonekcji dwóch antyrównoległych rurek wirowych zostały opublikowane w artykułach autorskich [61, 63, 48, 57].

7.2.5. Wpływ liczby Reynoldsa na proces rekonekcji

W wielu pracach uznaje się, że struktury wirowe przypominające rurki ewoluują i oddziałują ze sobą przy wysokich liczbach Reynoldsa w trójwymiarowych przepływach turbulentnych. Można sobie wyobrazić, że większość obszaru przepływu jest wypełniona płynem o zerowej lub bardzo niskiej wirowości a jego właściwości

są zależne do struktur wirowych o stosunkowo małych średnicach [107]. Przełączanie się linii wirowych może być jednym z podstawowych procesów w ewolucji trójwymiarowych struktur wirowych i mechanice turbulencji [74].

W tym świetle uzasadnionym jest zbadanie jaki wpływ ma liczba Reynoldsa na przebieg procesu rekonekcji. Zostanie to pokazane na przykładzie dwóch równoległych rurek wirowych.

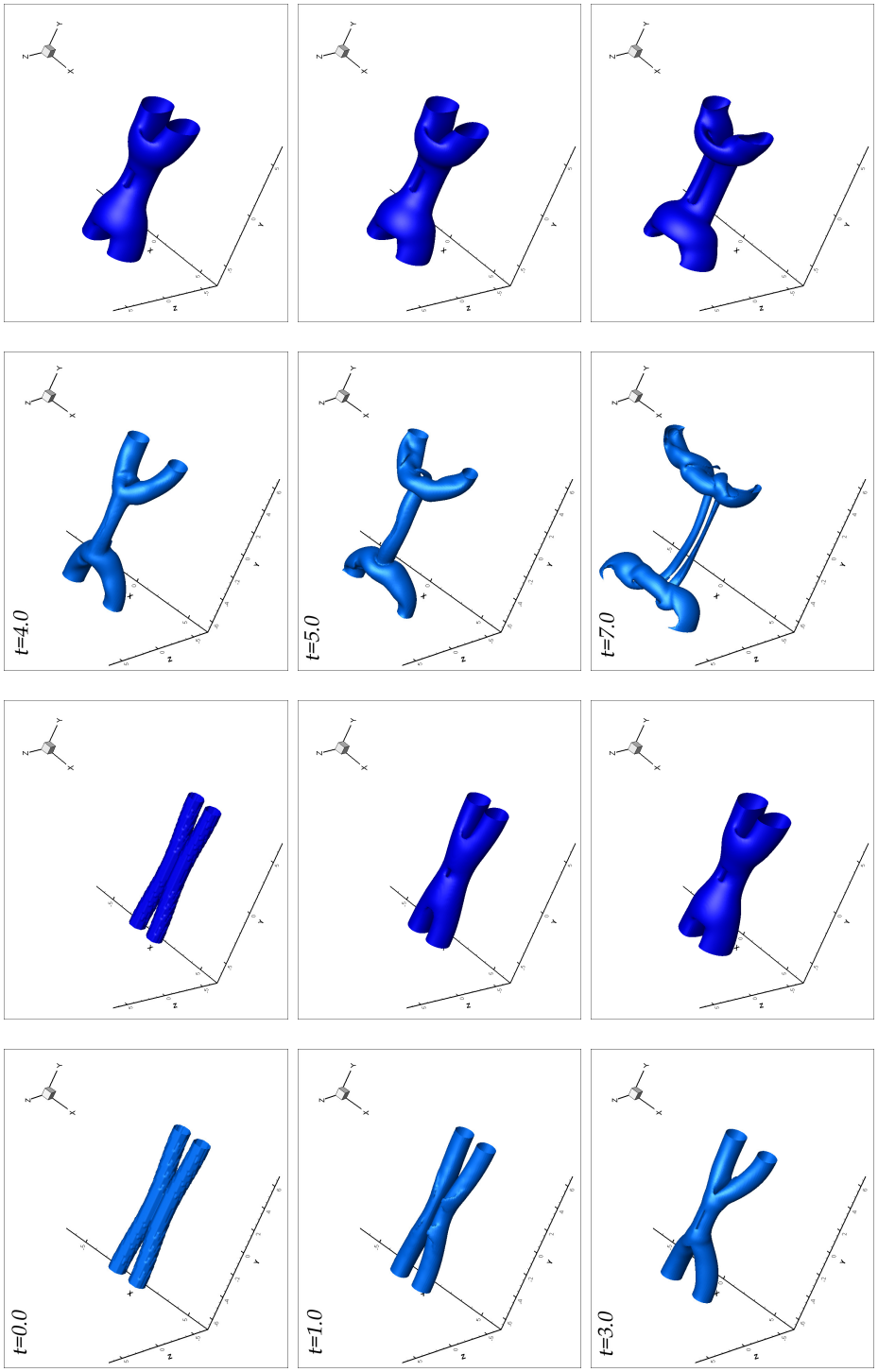
W tym przypadku badane rurki wirowe miały te same parametry geometryczne co rurki przedstawione w rozdziale 7.2.4. Zmiana liczby Reynoldsa była realizowana poprzez zmianę współczynnika lepkości. W pierwszym przypadku miał on wartość $\nu = 0.0176$ co dawało liczbę Reynoldsa $Re_\Gamma = 1003$, w drugim wypadku wartości te miały odpowiednio $\nu = 0.05$ i $Re_\Gamma = 353$. Otrzymane wyniki przedstawione są na rys. 7.14.

Przebieg procesu rekonekcji przedstawiony po lewej stronie rys. 7.14 jest zgodny z opisem w sekcji 7.2.4. Po prawej stronie tego rysunku widoczny jest obraz izolinii wirowości dla tej samej konfiguracji początkowej ale innej liczby Reynoldsa. Można z niego wnioskować, że nie doszło do pełnego procesu rekonekcji. W czasie ewolucji doszło do procesu mostowania i wytworzyły się połączenia pomiędzy początkowymi strukturami wirowymi. Jednak oddziaływania nie były na tyle silne, żeby nowe struktury mogły się rozdzielić. W dalszym etapie ewolucji struktury wirowe podlegają procesowi dyfuzji i powiększają swoje średnice nie zmieniając topologii linii wirowych.

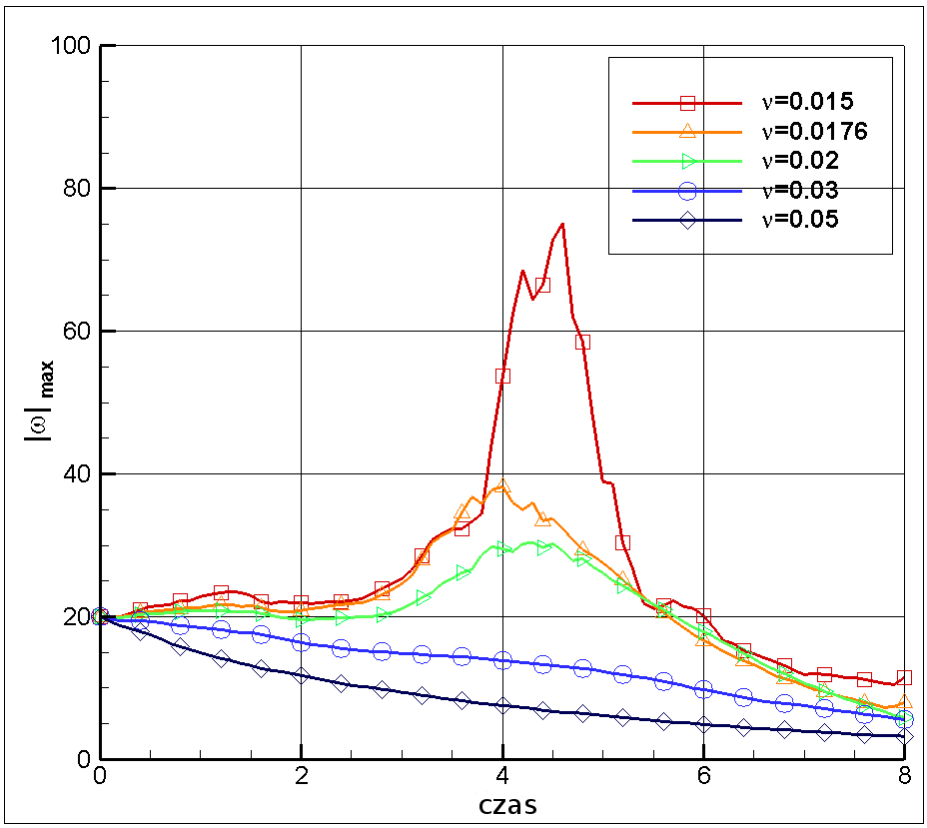
Badania te zostały powtórzone dla większej liczby różnych wartości współczynnika lepkości. Na rys. 7.15 został pokazany wykres zależności maksymalnej wartości wirowości w obszarze obliczeniowym w czasie w zależności od wartości współczynnika lepkości. Podobny wykres dla pojedynczej wartości współczynnika lepkości jest przedstawiony w pracy [45].

Wyniki pokazują, że dla przypadków w których zaszedł pełny proces rekonekcji pojawiło się maksimum na wykresie największej wartości wirowości w obszarze obliczeniowym. Wartość tego maksimum jest kilkukrotnie wyższa niż dla początkowej chwili obliczeń. Pokazuje to, że lokalnie proces rekonekcji ma charakter bardzo dynamiczny. Może to być także wskazówką, że w przepływ turbulentny jest tworzony przez struktury wirowe, które cięgle poddawane są procesom rekonekcji. Wedle wiedzy autora wynik ten jest pierwszym powiązaniem faktu występowania maksimum wirowości z przejściem pełnego procesu rekonekcji.

Wyniki przedstawiające wpływ liczby Reynoldsa na przebieg zjawiska rekonekcji zostały opublikowane w artykułach autorskich [61, 63, 57].



Rysunek 7.14: Przebieg procesu rekonekcji dwóch prostopadłych rurek wirowych dla różnych liczb Reynoldsa. Prawa strona: Współczynnik lepkości $\nu = 0.0176$ – doszło do procesu rekonekcji; Prawa strona: Współczynnik lepkości $\nu = 0.05$ – nie doszło do pełnego procesu rekonekcji.



Rysunek 7.15: Zależność maksymalnej wartości wirowości $|\omega|_{max}$ w czasie w zależności od współczynnika lepkości w procesie rekonekcji dwóch antyrównoległych rurek wirowych.

7.3. Zderzenie czołowe dwóch pierścieni wirowych

Bardzo ciekawym zagadnieniem jest zderzenie czołowe dwóch pierścieni wirowych. Wyniki eksperymentalne zostały zaprezentowany w *Nature* [69] a na stronie internetowej autora [67] można obejrzeć film. Eksperyment przedstawia dwa pierścienie wirowe, które w wyniku zderzenia czołowego rozpadają się na serię mniejszych pierścieni wirowych rozłożonych na obwodzie pierwotnych pierścieni. Po rekonekcji wtórne struktury zaczynają się promieniście oddalać od osi pierwotnych pierścieni. Zgodnie z wiedzą autora wyniku tego nie udało się powtórzyć numerycznie.

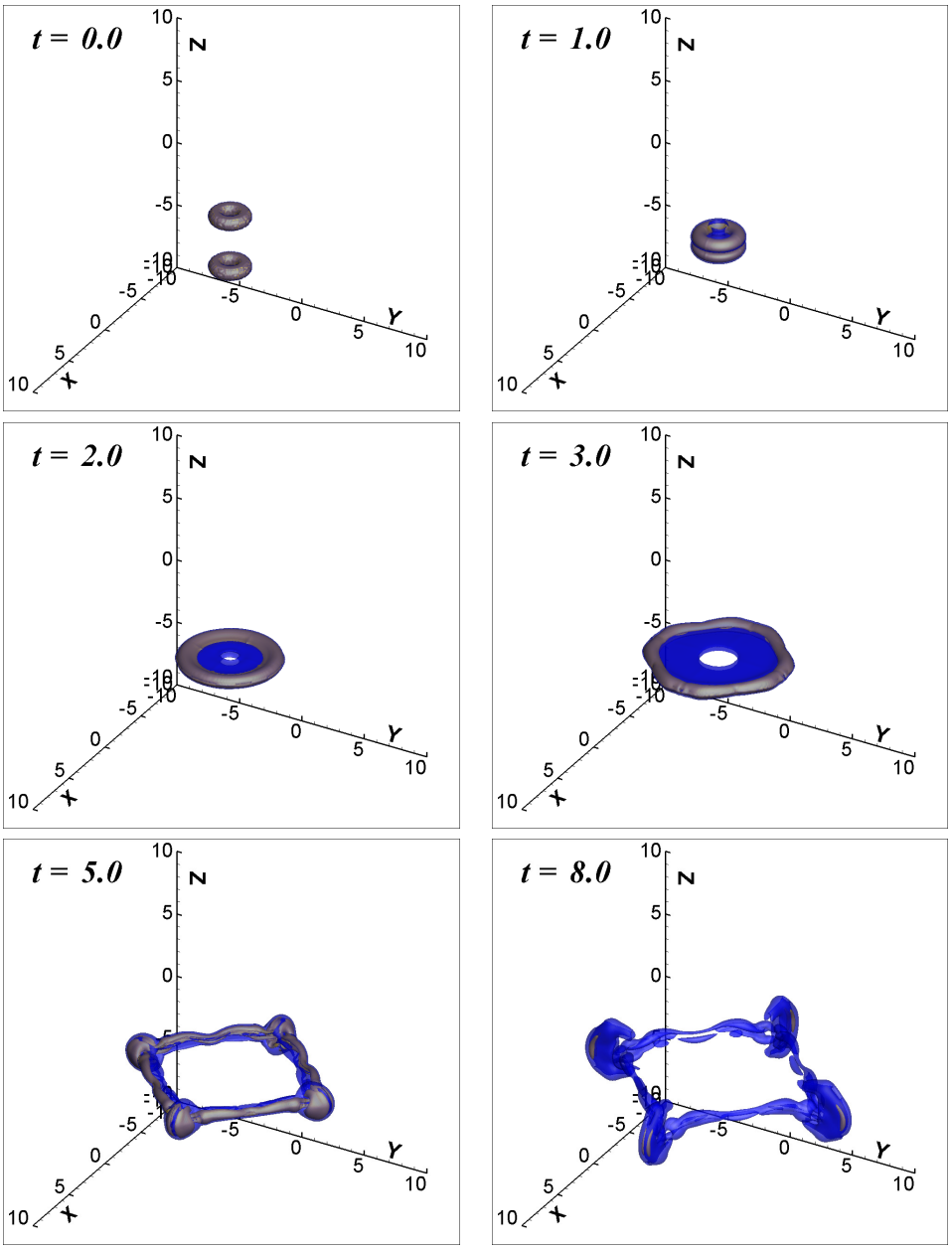
7.3.1. Obliczenia dla liczby Reynoldsa $Re_\Gamma = 1000$

W niniejszej pracy dokonano próby odtworzenia zjawiska rekonekcji powstającej w wyniku czołowego zderzenia dwóch pierścieni wirowych. W tym przypadku parametry we wzorach (7.7), (7.8) i (7.9) miały wartości $\omega_0 = 20$, $r_0 = 0.5$, $R_0 = 1.0$, $\nu = 0.01$, $Re_\Gamma = 1000$. Siatka numeryczna miała $256 \times 256 \times 256$ węzłów. Wyniki w formie izopowierzchni wirowości przedstawione są na rys. 7.16.

Niestety trudno jest dokonać dokładnego porównania eksperymentu Lima na podstawie porównania liczb Reynoldsa. W eksperymencie była bowiem ona wyliczana na podstawie prędkości tłoka i średnicy otworu służących do wytworzenia pierścienia wirowego. W obliczeniach numerycznych zdefiniowana jest w zależności od cyrkulacji pierścienia. Mogą to być znacznie różne liczby.

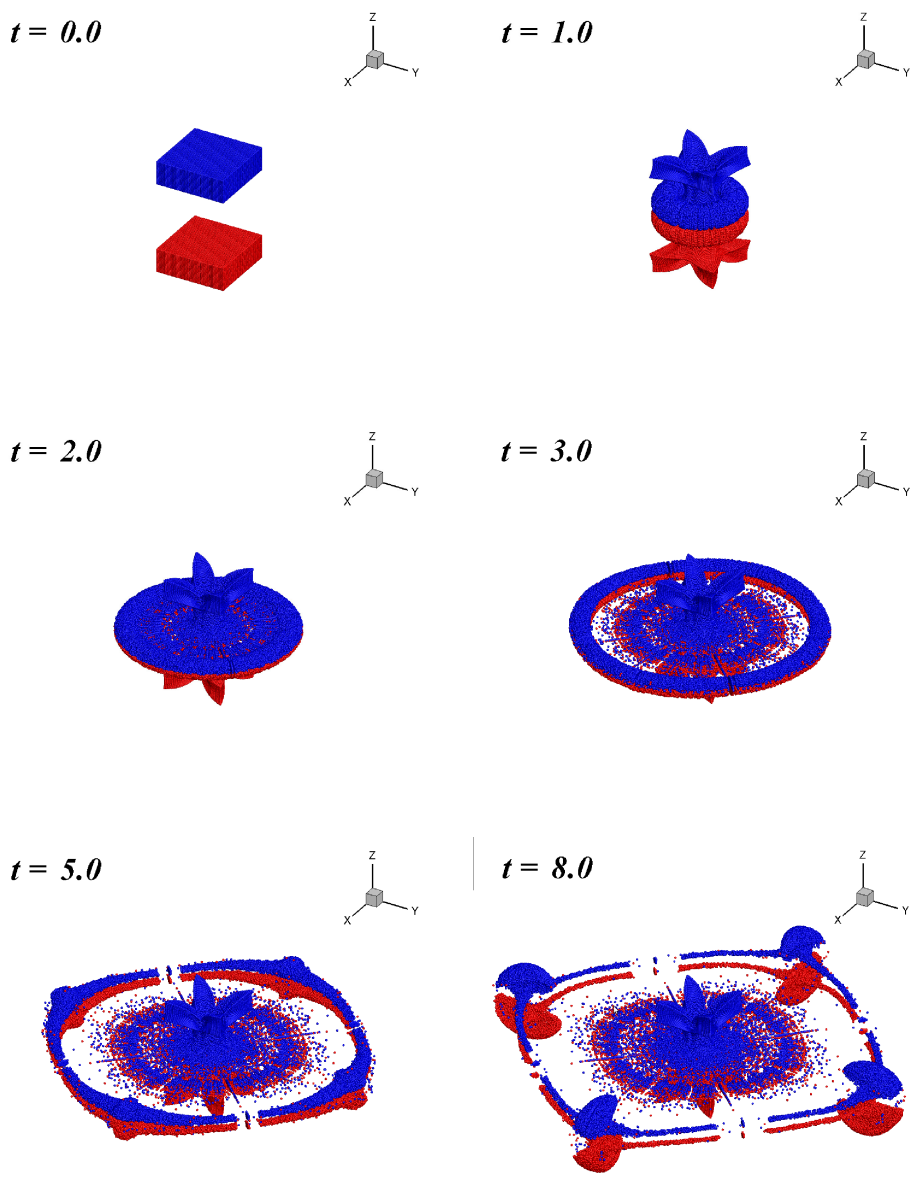
W czasie zbliżania się do siebie prędkości indukowane przez pierścienie powodują rozszerzanie się ich i spłaszczenie rdzenia (przekroju). W chwili $t = 2$ na rys. 7.16 widoczny jest specyficzny rozkład wirowości nazywany przez Lima membraną. Zgodnie z [69] w momencie, kiedy pierścienie osiągają wielkość ok. 4 początkowych średnic pojawia się asymetryczne zaburzenie w postaci fali. Powoduje ono powstanie na obwodzie każdego pierścienia czterech skupisk wirowości. Wyniki numeryczne pokrywają się z wynikami eksperymentu dla niskich liczb Reynoldsa. Oprócz liczby zgrubień główną różnicą pomiędzy otrzymanymi wynikami są pozostałości barwnika w środku zderzających się pierścieni widoczne na zdjęciach zarówno w [69] jak i na [67].

W poprzednich punktach zostało pokazane, że w przepływie lepkim wirowość nie jest unoszona przez cząstki płynu i dlatego trudno jest przeprowadzić wizualizację ewolucji pola wirowości. Z tego powodu lepszym sposobem porównania otrzymanych wyników numerycznych z eksperymentem jest umieszczenie w obszarze obliczeniowym pasywnych markerów unoszonych przez płyn. Zostały przeprowadzone symulacje, w których do przepływu wprowadzono pasywne markery symulujące zachowanie się cząstek barwnika używanego w eksperymentach. Na rys. 7.17 przedstawione są pokolorowane pasywne markery. Kolory odpowiadają początkowemu położeniu. W chwili początkowej pierścienie wirowe są całkowicie



Rysunek 7.16: Obraz izopowierzchni $|\omega| = 0.2\omega_0$ (czerwony) i $|\omega| = 0.05\omega_0$ (niebieski) czołowego zderzenia dwóch pierścieni wirowych . $Re_\Gamma = 1000$.

schowane w obszarze pokrytym markerami. Każdy z tych obszarów zawiera markery rozłożone na siatce posiadającej $100 \times 100 \times 50$ węzłów, czyli w symulacji bierze udział 10^6 markerów. Część markerów leżała poza obszarem posiadającym wirowość. Dla czasu $t = 1$ widoczny jest ogon utworzony z markerów znajdujących się początkowo poza pierścieniami wirowymi. W chwili $t = 2$ widoczna jest



Rysunek 7.17: Obraz pasywnych markerów dla przypadku czołowego zderzenia dwóch pierścieni wirowych. Kolory oznaczają początkowe położenie markerów. $Re_\Gamma = 1000$.

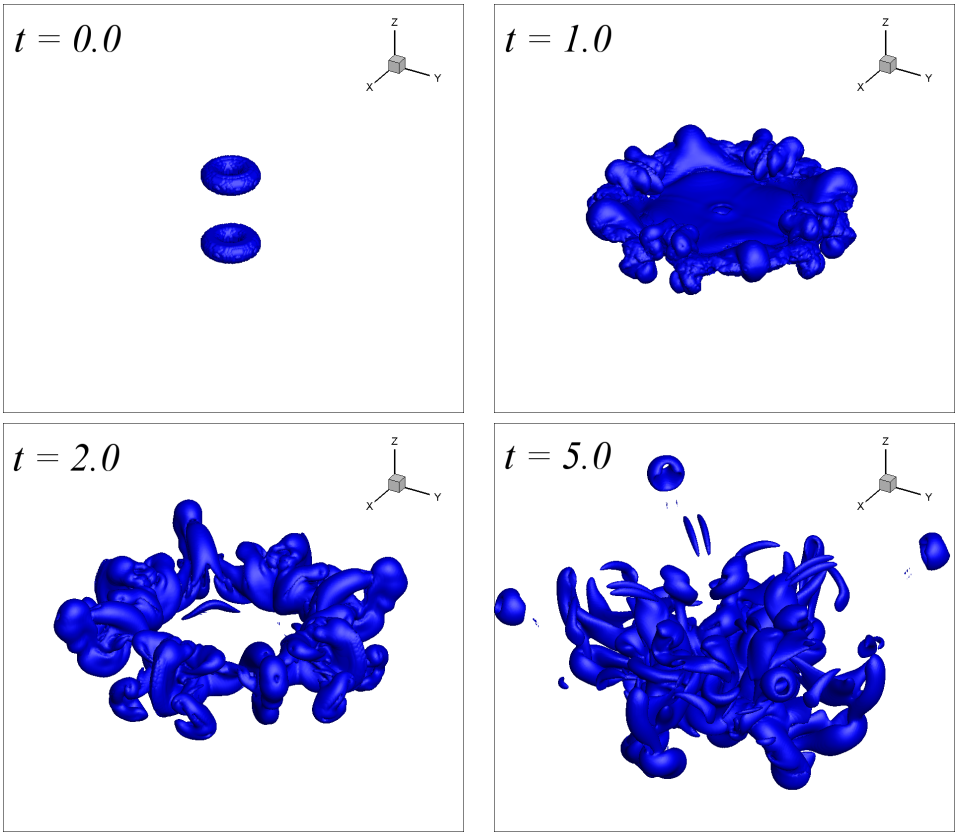
membrana. W chwili $t = 3$ widać, że tylko część z markerów znajdujących się wewnątrz pierścieni na początku ruchu została wraz z nimi przemieszczona. Druga część pozostała w centralnej części obszaru zderzenia. Obraz przedstawiony w chwili $t = 5$ bardzo przypomina wyniki otrzymane przez Lima umieszczone na

stronie internetowej tego autora [67]. W odróżnieniu od wizualizacji izopowierzchniami wirowości dobrze widoczne jest skupisko markerów w środkowej części obszaru zderzenia. Była to podstawowa różnica pokazana w punkcie 7.3 pomiędzy wynikami eksperymentalnymi a obliczeniami numerycznymi. Jasne jest teraz, że była ona spowodowana różnicą pomiędzy ewolucją skalarów a ewolucją wirowości. Z obrazu dla chwili $t = 8$ wynika, że proces rekonekcji skalarów w tym wypadku nie zaszedł.

Zastosowanie pasywnych markerów pozwoliło potwierdzić zgodność obliczeń numerycznych z eksperymentem. Nie było to wprost możliwe tylko przy wizualizacji izopowierzchniami wirowości.

7.3.2. Obliczenia dla liczby Reynoldsa $Re_\Gamma = 2000$

Została również przeprowadzona symulacja dla większej liczby Reynoldsa $Re_\Gamma = 2000$. Parametry pierścieni wirowych w tym przypadku były następujące: $\omega_0 = 80$, $a = 0.5$ $R = 1.0$, $\nu = 0.02$. Wyniki widoczne są na rys. 7.18.

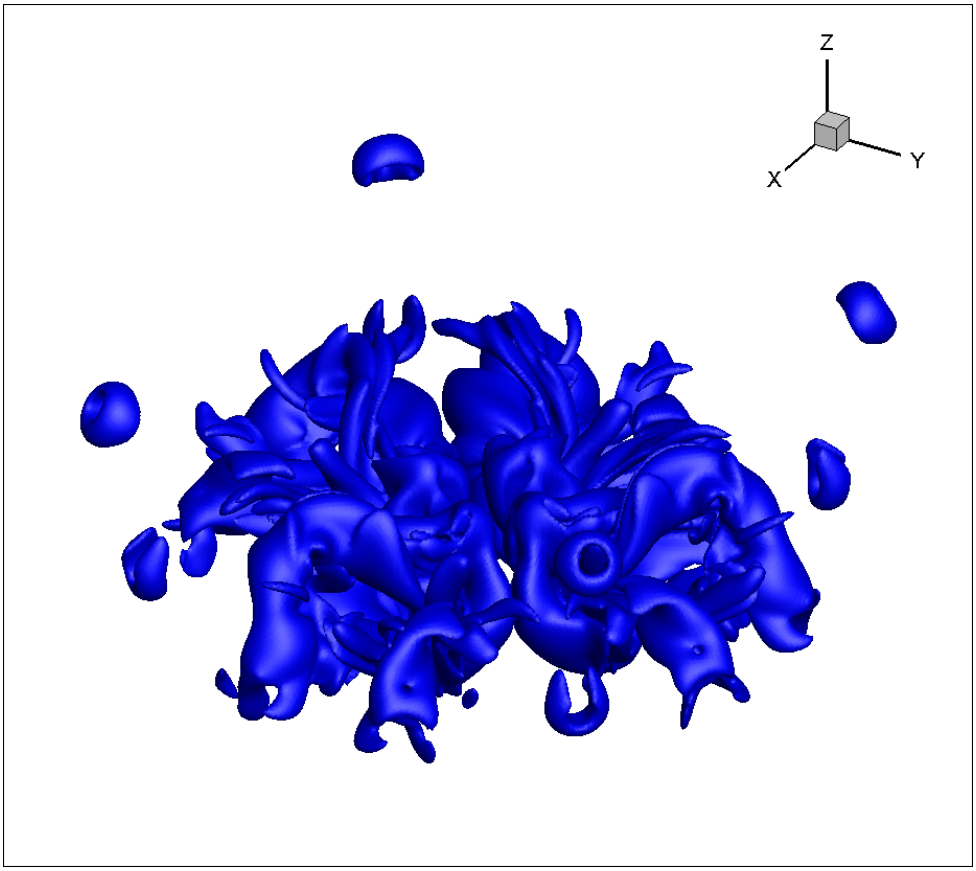


Rysunek 7.18: Obraz izopowierzchni wirowości $|\omega| = 0.025\omega_0$ dla przypadku zderzenia czołowego dwóch pierścieni wirowych. $Re_\Gamma = 2000$.

Wyniki wyglądają podobnie do przypadku z niższą liczbą Reynoldsa tylko do momentu pojawienia się membrany. Fala na obwodzie ma większą częstotliwość i powoduje mocniejsze odkształcenie struktur wirowych. Skutkiem tego jest widoczne na obrazie $t = 5$ oderwanie się czterech pierścieni wirowych, które oddalają się promieniście do pozostałości po pierwotnych pierścieniach. Może to dowodzić, że w tym wypadku miała miejsce rekonekcja struktur wirowych.

7.3.3. Obliczenia dla liczby Reynoldsa $Re_\Gamma = 2000$ z powiększoną siatką $512 \times 512 \times 512$ węzłów

Ten przypadek został przebadany na jeszcze gęstszej siatce $512 \times 512 \times 512$. Wymiary obszaru obliczeniowego pozostały bez zmian a krok siatki został zmniejszony dwukrotnie w każdym kierunku. Parametry początkowe pierścieni pozostały bez zmian. Na rys. 7.19 przedstawiona jest chwila $t = 5$.



Rysunek 7.19: Obraz izopowierzchni wirowości $|\omega| = 0.025\omega_0$ dla przypadku zderzenia czołowego dwóch pierścieni wirowych. $Re_\Gamma = 2000$. Obliczenia prowadzone na siatce $512x512x512$.

Na rysunku widać nie cztery, jak to miało miejsce w poprzednim punkcie, a osiem oddalających się pierścieni wirowych. Poza inną liczbą powstałych pierścieni wirowych, wyniki dobrze odpowiadają wynikom eksperymentów przedstawionych w [69] i [67]. Pokazuje to, że metoda VIC w przedstawionej implementacji jest w stanie odtworzyć rzeczywiste eksperymenty, nawet tak skomplikowanych zjawisk, jakim jest rekonekcja struktur wirowych przy czołowym zderzeniu dwóch pierścieni wirowych.

Wyniki przedstawiające zderzenie czołowe dwóch pierścieni wirowych zostały opublikowane w artykułach autorskich [62, 55, 48, 57].

Rozdział 8

Podsumowanie

Badając ewolucję i wzajemną interakcję prostych (podstawowych) struktur wirowych, takich jak rurki i pierścienie można poznać fascynujące zjawiska takie jak rekonekcja czy gra wirów. Skomplikowanie pola prędkości w przepływie jakie powstaje po tych zjawiskach prowadzi do wniosku, że mogą być one podstawą powstawania turbulencji [41].

Przedmiotem niniejszej rozprawy było opracowanie implementacji metody cząstek wirowych typu „Wir w komórce" (VIC) do rozwiązywania nieściśliwych, trójwymiarowych przepływów lepkich wykorzystującej w obliczeniach numerycznych wieloprocesorowe środowisko złożone z kart graficznych. Proponowana w pracy metoda oparta jest na aproksymacji ciągłego pola wirowości dyskretnym rozkładem cząstek wirowych, które niosą informację o trzech składowych wektora wirowości. Równania ruchu cieczy zostały sformułowane w ujęciu wirowości $\boldsymbol{\omega}$ i prędkości $\mathbf{u}$. W algorytmie metody wykorzystywany był także potencjał wektorowy $\mathbf{A}$.

W metodach wirowych cząstki mają tendencję do koncentrowania się w obszarach o wysokim gradiencie prędkości. Aby uniknąć błędów związanych ze zbytnim zbliżeniem się cząstek do siebie stosuje się tzw. redystrybucję, czyli przemieszczenie ich na regularne pozycje (np. węzły siatki). Proces ten powtarzany jest co zadaną (wyznaczoną metodą prób i błędów) liczbę kroków czasowych. Modyfikacją metody typu „Wir w komórce" zastosowaną w niniejszej pracy było wprowadzenie redystrybucji cząstek wirowych na węzły pomocniczej siatki numerycznej w każdym kroku czasowym metody.

Modelowanie nieściśliwych lepkich przepływów trójwymiarowych oparte było na algorytmie dekompozycji lepkościowej tzn. rozwiązanie otrzymywane było w dwóch krokach: najpierw rozwiązywano równania ruchu cieczy nielepkiej, a następnie uwzględniano w obliczeniach lepkość. Wprowadzenie redystrybucji cząstek wirowych na węzły siatki w każdym kroku czasowym pozwoliło na roz-

wiązywanie równania dyfuzji na siatce numerycznej dzięki czemu poprawiono dokładność symulacji efektów lepkościowych.

Poprawność implementacji metody sprawdzono badając dynamikę pojedynczego nielepkiego pierścienia wirowego. Porównując jego prędkość translacji z formułą analityczną, otrzymano dobrą zgodność wyników. Dla przepływów nielepkich modelowano również dynamikę układu dwóch pierścieni wirowych w różnych konfiguracjach początkowych. Odtworzono, dobrze znane w literaturze, zjawisko „gry wirów", uzyskując jeden pełny cykl przejścia pierścieni.

W pracy przedstawiono implementację metody VIC na pojedynczej karcie graficznej. Uzyskano ok. 50 krotne przyspieszenie obliczeń względem pojedynczego rdzenia procesora. Największa siatka jaka mieściła się na jednej karcie ($128 \times 128 \times 128$ węzłów) pozwoliła na przeprowadzenie badań nad rekonekcją prostych struktur wirowych. Otrzymane wyniki były zgodne z wynikami zarówno numerycznymi, jak i eksperymentalnymi innych autorów. Niestety siatka okazała się za mała przy próbie odtworzenia bardziej skomplikowanego zjawiska jakim było zderzenie się dwóch pierścieni wirowych.

Przeprowadzone w pracy badania numeryczne pozwoliły na głębsze poznanie procesu ewolucji i rekonekcji struktur wirowych. W pracy zostały pokazane zmiana topologii linii wirowych, wpływ liczby Reynoldsa na proces rekonekcji. Ważnym wnioskiem uzyskanym w czasie badań było pokazanie, że proces rekonekcji jest związany ze wzrostem maksymalnej wartości modułu wirowości w obszarze obliczeniowym. Zostały zidentyfikowane i pokazane wspólne etapy procesu rekonekcji dla większości przedstawionych przypadków.

Z przeprowadzonych badań wynika, że obliczenia na wielu kartach graficznych pozwalają uzyskać przyspieszenie względem pojedynczej karty. Wynik ten nie jest oczywisty z powodu konieczności przesyłania danych etapami. Dopiero wykorzystanie kopiowania asynchronicznego i strumieni pozwoliło otrzymać efektywne skalowanie czasu obliczeniowego dla większej liczby kart. Przedstawione skalowanie słabe pokazało, że mała pojemność pamięci kart graficznych nie musi być przeszkodą w wykonywaniu obliczeń z gęstymi siatkami. Odpowiednia dekompozycja danych pomiędzy wiele kart graficznych pozwoli na obejście tego problemu oraz dodatkowo może przyspieszyć obliczenia.

Zaprezentowane w niniejszej rozprawie wyniki numeryczne otrzymane za pomocą nowej implementacji metody VIC dla przepływów nielepkich i lepkich są dobrej jakościowo zgodności z wynikami eksperymentalnymi i numerycznymi dostępnymi w literaturze. Osiągnięte wyniki potwierdzają postawioną na początku pracy tezę, że w obliczeniach numerycznych możliwe jest zastąpienie pola wirowości dyskretnym rozkładem cząstek wektorowych będącymi nośnikami wirowości, a środowisko kart graficznych nadaje się do obliczeń naukowych i pozwala znacząco skrócić czas obliczeń. Cele pracy zostały zrealizowane.

Bibliografia

[1] A multigrid tutorial. www.math.ust.hk/~{}mawang/teaching/math532/mgtut.pdf. [cytowanie na str. 55, 126]

[2] Cyberinfrastructure tutor (ci-tutor). www.citutor.org, 2012. [cytowanie na str. 70, 71]

[3] Nvidia cuda c programming guide. www.nvidia.com/cuda, 2012. [cytowanie na str. 5, 40, 42, 43, 50, 74]

[4] Klaster obliczeniowy zeus. http://www.cyfronet.krakow.pl/komputery/13345,artykul,zeus.html, 2013. [cytowanie na str. 70]

[5] Okulov V. L. Alekseenko S. V., Kuibin P. A. *Theory of Concentrated Vortices*. Springer, 2007. [cytowanie na str. 33]

[6] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1970. [cytowanie na str. 11]

[7] J. T. Beal, A. Majda. Vortex methods i: Convergence in three dimensions. *Mathematics of Computation*, 39(159):1–27, 1982. [cytowanie na str. 3]

[8] J. T. Beal, A. Majda. Vortex methods ii: High-order accuracy in two and three dimensions. *Mathematics of Computation*, 39(159):29–52, 1982. [cytowanie na str. 3]

[9] N. Bell, S. Dalton, L. N. Olson. Exposing fine-grained parallelism in algebraic multigrid methods. 34(4), 2008. [cytowanie na str. 6, 7]

[10] J. Bolz, I. Farmer, E. Grinspun, P. Schro"der. Sparse matrix solvers on the gpu: Conjugate gradients and multigrid. Raport instytutowy, Caltech, 2004. [cytowanie na str. 7]

[11] O.N. Boratav, R.B. Pelz, N.J. Zabusky. Reconnection in orthogonally interacting vortex tubes: Direct numerical simulations and quantifications. *Phys. Fuids A*, 4(3):581–605, 1991. [cytowanie na str. 2, 88]

[12] B. Braide. *A Friendly Introduction to Numerical Analysis*. Pearson Prentice Hall, 2000. [cytowanie na str. 6, 51]

[13] W. L. Briggs, V. E. Henson, S. F. McCormick. *A Multigrid Tutorial*. SIAM, 2000. [cytowanie na str. 6]

[14] S. Childress. Topological fluid dynamics for fluid dynamicists. http://www.math.nyu.edu/faculty/childres/tfd.pdf. [cytowanie na str. 87]

[15] A. J. Chorin. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57:785–796, 1973. [cytowanie na str. 3, 4, 25, 27]

[16] G. H. Cottet. Convergence of a vortex-in-cell method for the two-dimensional euler equations. *Mathematics of Computation*, 49(180):407–425, 1987. [cytowanie na str. 3, 4]

[17] G.-H. Cottet, P. D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000. [cytowanie na str. 3, 4, 11, 13, 16, 22, 27, 29, 30]

[18] G.-H. Cottet, P. Poncet. Advances in direct numerical simulations of 3d wall-bounded flows by vortex-in-cell methods. *Journal of computational physics*, 193(1):136–158, 2004. [cytowanie na str. 4, 17]

[19] G.-H. Cottet, M.-L. Ould Salihi, M. El Hamraoui. Multi-purpose regridding in vortex methods. *ESAIM : Proceedings*, wolumen 7, strony 94–103. [cytowanie na str. 25]

[20] B. Couet, O. Buneman, A. Leonard. Simulation of the three-dimensional incompressible flows with a vortex-in-cell method. *Journal of Computational Physics*, 39(2):305–328, 1981. [cytowanie na str. 3]

[21] Z. Czech. *Wprowadzenie do obliczeń równoległych*. Wydawnictwo Naukowe PWN SA, Warszawa, 2010. [cytowanie na str. 73, 75]

[22] T.L. Freeman, C. Phillips. *Parallel Numerical Algorithms*. Prentice Hall, 1992. [cytowanie na str. 67]

[23] Y. Ge. Multigrid method and fourth-order compact difference discretization scheme with unequal meshsizes fof 3d poisson equation. *Journa of Computational Physics*, 229:6381–6391, 2010. [cytowanie na str. 6]

[24] M. Gharib, E. Rambod, K. Shariff. A universal time scale for vortex ring formation. *Journal of Fluid Mechanics*, 360:121–140, 1998. [cytowanie na str. 1]

[25] A. Giovannini, Y. Gagnon. Validation of a three-dimensional vortex particle method for fluid flows. *Applied Mathematics Research Express*, 2006:1–31, 2006. [cytowanie na str. 2]

[26] Hirasaki G.J. *A general formulation of the boundary conditions on the vector potential in three-dimensional hydrodynamics.* Praca doktorska, Rice University, 1967. [cytowanie na str. 14, 15]

[27] D. Göddeke, R. Strzodka, J. Mohd-Yusof, P. McCormick, H. Wobker, C. Becker, S. Turek. Using gpus to improve multigrid solver performance on a cluster. *International Journal of Computational Science and Engineering*, 4(1):36–55, 2008. [cytowanie na str. 7]

[28] N. Goodnight, G. Lewin, D. Luebke, K. Skadron. A multigrid solver for boundary value problems using programmmable graphics hardware. Raport instytutowy, University of Virginia, 2003. [cytowanie na str. 7]

[29] S. I. Green. *Fluid Vortices.* Springer, 1995. [cytowanie na str. 32]

[30] L. Greengard, V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987. [cytowanie na str. 4, 16]

[31] M.M. Gupta. A fourth-order poisson solver. *Journal of Computational Physics*, 55:166–172, 1984. [cytowanie na str. 6]

[32] V. Heuveline, D. Lukarski, N. Trost, J.-P. Weiss. Parallel smoothers for matrix-based multigrid methods on unstructured meshes using multicore cpus and gpus. Raport instytutowy 2011-09, Karlsruhe Institute of Technology, 2011. [cytowanie na str. 6]

[33] G.J. Hirasaki, J.D. Hellums. A general formulation of the boundary conditions on the vector potential in three-dimensional hydrodynamics. *Quart. Appl. Math*, 26:331–342, 1968. [cytowanie na str. 14]

[34] G.J. Hirasaki, J.D. Hellums. Boundary conditions on the vector and scalar potentials in viscous three-dimensional hydrodynamics. *Quart. Appl. Math*, 28:293–296, 1970. [cytowanie na str. 14]

[35] H. Holden, K. H. Karlsen, K.-A. Lie, W. H. Risebro. *Splitting Methods for Partial Differential Equations with Rough Solutions.* European Mathematical Society, 2010. [cytowanie na str. 26]

[36] F. Hussain, K. Duraisamy. Mechanics of viscous vortex reconnection. *Physics of fluids*, 23(2), 2011. [cytowanie na str. 2, 88]

[37] F. Hussain, M. V. Melander. New aspects of vortex dynamics: Melical waves, core dynamics, viscous helicity generation and interaction with turbulence. H.K. Moffat et al., redaktor, *Topological Aspects of the Dynamics of Fluids and Plasma.* Institue of Theoretical Physics - Santa Barbara, Kluwer Publ. [cytowanie na str. 2]

[38] T. Kambe. *Elementary Fluid Mechanics*. World Scientific Publishing, 2007. [cytowanie na str. 15, 17]

[39] F.B. Kaplanskii, Y.A. Rudi. Evolution of vortex ring. *Fluid Dynamics*, 36(1):16–25, 1999. [cytowanie na str. 2]

[40] R. M. Kerr, F. Hussain. Simulation of vortex reconnection. *Physica D*, 37:474–484, 1989. [cytowanie na str. 2, 88]

[41] Robert M. Kerr. Swirling, turbulent vortex rings formed from a chain reaction of reconnection events. *Physics of Fluids (1994-present)*, 25(6):–, 2013. [cytowanie na str. 2, 113]

[42] S. Kida, M. Takaoka. Reconnection of vortex tubes. *Fluid Dynamics Research*, 3:257–261, 1988. [cytowanie na str. 2, 100]

[43] S. Kida, M. Takaoka. Breakdown of frozen motion of vorticity field and vorticity reconnection. *J. of Physical Sociecty of Japan*, 60(7):2184–2196, July 1991. [cytowanie na str. 86, 88, 96]

[44] S. Kida, M. Takaoka. Vortex reconnection. *Annual Review of Fluid Mechanics*, 26:169–189, 1994. [cytowanie na str. 2, 86, 87, 92, 98, 100]

[45] S. Kida, M. Takaoka, F. Hussain. Collision of two vortex rings. *Journal of Fluid Mechanics*, 230:583–646, 1991. [cytowanie na str. 2, 88, 89, 100, 103]

[46] Y. Kimura. Motion of an elliptic vortex ring and particle transport. strony 119–124. [cytowanie na str. 2]

[47] D. B. Kirk, W. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers, 2010. [cytowanie na str. 5]

[48] A. Kosior. Metoda dekompozycji lepkościowej w obliczeniach numerycznych płynów lepkich na przykładzie zmiany topologii struktur wirowych. *Zeszyty Energetyczne*, 1:43–52, 2014. [cytowanie na str. 102, 111]

[49] A. Kosior, H. Kudela. Równoległe rozwiązywanie równania poissona metodą wielosiatkową na karcie graficznej. J. Szrek, redaktor, *Interdyscyplinarność badań naukowych 2011*, strony 169–174. Oficyna Wydawnicza Politechniki Wrocławskiej, 2011. [cytowanie na str. 64]

[50] A. Kosior, H. Kudela. Modelowanie dynamiki pierścienia wirowego metodą cząstek wirowych z wykorzystaniem obliczeń równoległych na kartach graficznych. *Modelowanie Inżynierskie*, 13(44):145–15, 2012. [cytowanie na str. 64]

[51] A. Kosior, H. Kudela. Parallel computations in engineering. K. Wójs, P. Szulc, T. Tietze, redaktorzy, *Aktualne kierunki rozwoju energetyki*, strony 17–24. Oficyna Wydawnicza Politechniki Wrocławskiej, 2012. [cytowanie na str. 61, 62]

[52] A. Kosior, H. Kudela. Vortex particle method and parallel computing. *Journal of Theoretical and Applied Mechanics*, 50(1):285–300, 2012. [cytowanie na str. 33, 61, 62]

[53] A. Kosior, H. Kudela. Modelowanie ewolucji pierścieni wirowych za pomocś metody cząstek wirowych z wykorzystaniem obliczeń równoległych na kartach graficznych. J. Szrek, redaktor, *Interdyscyplinarność badań naukowych 2013*, strony 282–287. Oficyna Wydawnicza Politechniki Wrocławskiej, 2013. [cytowanie na str. 86]

[54] A. Kosior, H. Kudela. Parallel computations on gpu in 3d using the vortex particle method. *Computers & Fluids*, 80:423–428, 2013. [cytowanie na str. 61, 62, 64]

[55] A. Kosior, H. Kudela. The 3d vortex particle method in parallel computations on many gpus. *Computers & Fluids*, 92:274–280, 2014. [cytowanie na str. 70, 77, 86, 111]

[56] J. Kouatchou, J. Zhang. Optimal injection operator and high order schemes for multigrid solution of 3d poisson equation. *International Journal of Computer Mathematics*, 75:173–190, 2000. [cytowanie na str. 6]

[57] H. Kudela, A. Kosior. Simulation of the vortex tube reconnection using graphics cards and parallel computations by vortex-in-cell method. [cytowanie na str. 77, 86, 96, 102, 103, 111]

[58] H. Kudela, A. Kosior. Zastosowanie obliczeń równoległych w metodzie cząstek wirowych. J. Szrek, redaktor, *Interdyscyplinarność badań naukowych 2010*, strony 193–196. Oficyna Wydawnicza Politechniki Wrocławskiej, 2010. [cytowanie na str. 61, 62]

[59] H. Kudela, A. Kosior. Modeling vortex rings dynamics with vortex in cell method. *Journal of Physics. Conference Series*, 318(6):062014, 2011. [cytowanie na str. 37]

[60] H. Kudela, A. Kosior. Rozwiązanie równania poissona z wykorzystaniem wielu kart graficznych. J. Szrek, redaktor, *Interdyscyplinarność badań naukowych 2011*, strony 263–268. Oficyna Wydawnicza Politechniki Wrocławskiej, 2012. [cytowanie na str. 70, 77]

[61] H. Kudela, A. Kosior. Parallel computation of vortex tube reconnection using a graphics card and the 3d vortex-in-cell method. *Procedia IUTAM, Topological Fluid Dynamics II*, 7:59–66, 2013. [cytowanie na str. 70, 86, 96, 102, 103]

[62] H. Kudela, A. Kosior. Numerical study of the vortex tube reconnection using vortex particle method on many graphics cards. *Journal of Physics. Conference Series*, 530(1):012021, 2014. [cytowanie na str. 70, 111]

[63] H. Kudela, A. Kosior. Vortex particle method in parallel computations on graphical processing units used in study of the evolution of vortex structures. *Fluid Dynamics Research*, 46(6):061414, 2014. [cytowanie na str. 70, 77, 86, 96, 102, 103]

[64] H. Kudela, P. Regucki. The vortex-in-cell method for the study of three-dimensional flows by vortex methods. K. Bajer, H. K. Moffatt, redaktorzy, *Tubes, Sheets and Singularities in Fluid Dynamics*, Fluid Mechanics and Its Applications vol. 71, strony 49–54. Kluwer Academic Publishers, 2002. [cytowanie na str. 13, 83, 86]

[65] A. Leonard. Vortex methods for flow simulation. *Journal of Computational Physics*, 37:289–335, 1980. [cytowanie na str. 3]

[66] R.J. LeVeque. Finite difference methods for ordinary and partial differential equations. 2010. [cytowanie na str. 27]

[67] T. T. Lim. http://serve.me.nus.edu.sg/limtt/. [cytowanie na str. 89, 106, 109, 111]

[68] T. T. Lim. A note on the leapfrogging between two coaxial vortex rings at low reynolds numbers. *Phys. Fluids*, 9:239–241, 1997. [cytowanie na str. 83, 86, 88]

[69] T. T. Lim, T. B. Nickels. Instability and reconnection in the head-on collision of two vortex rings. *Nature*, 357:225–227, 1992. [cytowanie na str. 2, 88, 106, 111]

[70] A. J. Majda, A. L. Bertozzi. *Vorticity and Incompressible Flow.* Cambridge University Press, 2002. [cytowanie na str. 16]

[71] J.S. Marshall. *Inviscid Incompressible Flow.* John Wiley & Sons, 2001. [cytowanie na str. 12, 18, 19, 20, 21]

[72] J.S. Marshall, P. Brancher, A. Giovannini. Interaction of unequal anti-parallel vortex tubes. *J. Fluid Mech*, 446:229–252, 2001. [cytowanie na str. 2, 88]

[73] E. Meiburg. *Fluid Vortices*, rozdzia/l Three-dimensional vortex dynamics simulations, strony 651–685. Fluid Mechanics and its applications. Kluwer Academic Publishers, 1995. [cytowanie na str. 16]

[74] V. M. Melander, F. Hussain. Cross-linking of two antiparallel vortex tubes. *Phys. Fluids A*, 1(4):633–636, April 1989. [cytowanie na str. 2, 86, 98, 100, 103]

[75] V. M. Melander, F. Hussain. Reconnection of two antiparallel vortex tubes: A new cascade mechanism. F. Durst, B. Launder, W.C. Reynolds, F.W. Schmidt, J.H. Whitelaw, redaktorzy, *Turbulent Shear Flows 7*, strony 9–16, 1991. [cytowanie na str. 2, 88]

[76] H. K. Moffatt. The degree of knottedness of tangled vortex lines. *Journal of Fluid Mechanics*, 35(1):117–129, 1969. [cytowanie na str. 87]

[77] J. J. Monaghan. Extrapolating b-splines for interpolation. *Journal of Computational Physics*, 60:253–262, 1985. [cytowanie na str. 25]

[78] P. Orlandi, G.F. Carnevale. Nonlinear amplification of vorticity in inviscid interaction of orthogonal lamb dipolws. *Physics of Fluids*, 19, 2007. [cytowanie na str. 2]

[79] Y. Oshima, S. Asaka. Interaction of two rings moving side by side. *Natural Science Report, Ochanomizu University*, 26(1):31–37, 1975. [cytowanie na str. 89]

[80] D. J. Panow. *Metody numeryczne rozwiązywania równań różniczkowych cząstkowych*. Warszawa, 1955. [cytowanie na str. 6]

[81] P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, M.S. Warren. Vortex methods for direct numerical simulations of three-dimensional bluff body flows: Application to the sphere at $re = 300$, 500, and 1000. *Journal of Computational Physics*, 178:427–463, 2000. [cytowanie na str. 30]

[82] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery. *Numerical Recepies in Fortran*. Cambridge University Press, 1992. [cytowanie na str. 6]

[83] E. G. Puckett. *Incompressible Computational Fluid Dynamics Trends and Advances*, rozdzia/l Vortex Methods: An Introduction and Survey of Selected Research Topics, strony 335–407. Cambridge University Press, 1993. [cytowanie na str. 3, 16]

[84] L. Quartapelle. *Numerical Solution of the Incompressible Navier-Stokes Equations*. Birkhäuser Verlag, 1993. [cytowanie na str. 11]

[85] P. Regucki. *Modelowanie trójwymiarowych przepływów wirowych metodami dyskretnych wirów.* Praca doktorska, Politechnika Wrocławska, 2003. [cytowanie na str. 30]

[86] R. L. Ricca, M. A. Berger. Topological ideas and fluid mechanics. *Physics Today*, 49(12):28–34, 1996. [cytowanie na str. 86]

[87] P. G. Saffman. A model of vortex reconnection. *Journal of Fluid Mechanics*, 212:395–402, 1990. [cytowanie na str. 92]

[88] P. G. Saffman. *Vortex Dynamics.* Cambridge University Press, 1992. [cytowanie na str. 32]

[89] J. Sanders, E. Kandrot. *CUDA by Example An Introduction to General-Purpose GPU Programming.* Addison-Wesley Professional, 2010. [cytowanie na str. 5]

[90] P. R. Schaltze. *An experimental study of fusion of vortex rings.* Praca doktorska, California Institue of Technology, 1987. [cytowanie na str. 89]

[91] C. Schram, M.L. Riethmuller. Vortex ring evolution in an impulsively started jet using digital particle image velocimetry and continuous wavelet analysis. *Measurement Science and Technology*, 12:1413–1421, 2001. [cytowanie na str. 1]

[92] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonixing pain, 1994. [cytowanie na str. 7, 66, 67]

[93] E. D. Siggia. Collapse and amplification of a vortex filament. *Physics of Fluids*, 28(3):794–805, 1984. [cytowanie na str. 2, 92]

[94] S. K. Stanaway, B. J. Cantwell. Navier-stokes simulations of axisymmetric vortex rings. *AIAA 26th Aerospace Sciences Meeting*, 1988. [cytowanie na str. 2]

[95] P. N. Swarztrauber. Fast poisson solvers. *Studies in Numerical Analysis*, 24:319–370, 1984. [cytowanie na str. 6, 31, 45]

[96] P. N. Swarztrauber, R. A. Sweet, J. C. Adams. Fishpack: Efficient fortran subprograms for the solution of elliptic partial differential equations, 1999. [cytowanie na str. 31, 45, 63]

[97] J. W. Thomas. *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations.* Springer-Verlag, 1999. [cytowanie na str. 6, 51]

[98] A. Schuller U. Trottenberg, C.W. Oosterlee. *Multigrid.* Academic Press, Londyn, 2001. [cytowanie na str. 6, 54, 55, 56, 58, 60, 62]

[99] W.M. van Rees, F. Hussain, Koumoutsakos P. Vortex tube reconnection at $re = 10^4$. *Physics of Fluids*, 24(7), 2012. [cytowanie na str. 2, 88]

[100] W.M. van Rees, A. Leonard, D.I. Pullin, Koumoutsakos P. A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high reynolds numbers. *Journal of Computational Physics*, 230:2794–2805, 2011. [cytowanie na str. 2, 3]

[101] J. Wang, W. Zhong, J. Zhang. A general meshsize fourth-order compact difference discretization scheme for 3d poisson equation. *Applied Mathematics and Computation*, 183:804–812, 2006. [cytowanie na str. 6]

[102] A. Weigand, M. Gharib. On the evolution of laminar vortex rings. *Experiments in Fluids*, 22:447–457, 1997. [cytowanie na str. 1]

[103] P. Wesseling. *An Introduction to multigrid methods*. John Wiley & Sons, 1992. [cytowanie na str. 6]

[104] S.E. Widnall, J.P. Sullivan. On the stability of vortex rings. *Proc. R. Soc. Lond. A.*, 332:335–353, 1973. [cytowanie na str. 1]

[105] J. Z. Wu, H. Y. Ma, M. D. Zhou. *Vorticity and Vortex Dynamics*. Springer, 2006. [cytowanie na str. 13]

[106] H. Yamada, T. Matsui. Mutual slip-through of a pair of vortex rings. *Phys. Fluids*, 22(7):1245–1249, 1978. [cytowanie na str. 1]

[107] N. J. Zabusky, M. V. Melander. Three-dimensional vortex tube reconnection: Morphology for orthogonally - offset tubes. *Physica D*, 37:555–562, 1989. [cytowanie na str. 2, 88, 92, 93, 100, 103]

[108] I. Zawadzki, H. Aref. Mixing during vortex ring collision. *Physics of Fluids A*, 3(5):1405–1410, 1989. [cytowanie na str. 2]

Spis rysunków

Spis tabel