

POLITECHNIKA OPOLSKA

Wydział Elektrotechniki, Automatyki i Informatyki



Autoreferat rozprawy doktorskiej

Mgr inż. Artur Pala

**Równoległe realizacje wybranych algorytmów
optymalizacji dynamicznej w technologii
masowo równoległej oraz badania ich
efektywności**

Promotor:

dr hab. inż. Jan Sadecki

Opole 2022

Spis treści:

1. Streszczenie	3
2. Wprowadzenie do tematyki badań	4
3. Sformułowanie celu, tezy oraz zakresu pracy	7
4. Masowo równoległa organizacja obliczeń w technologii Nvidia CUDA	11
5. Masowo równoległe realizacje klasycznych algorytmów rozwiązywania układów równań różniczkowych zwyczajnych w technologii Nvidia CUDA	13
6. Masowo równoległy algorytm oparty na założeniach koncepcji Spekulacyjnej dedykowany dla technologii Nvidia CUDA	16
7. Masowo równoległe realizacje algorytmów rozwiązywania układów równań różniczkowych zwyczajnych oparte na koncepcji równoległego rozwiązywania równań oraz na koncepcji Spekulacyjnej.....	21
8. Rozpatrywane w pracy problemy obliczeniowe	24
8.1 Jednowymiarowy problem optymalizacji -Metoda Gradientu Prostego	24
8.2 Diagramy bifurkacyjne dla układów łuku elektrycznego.....	25
9. Uwagi końcowe i wnioski	35
9.1 Wkład własny autora pracy:.....	36
9.2 Kierunki dalszych badań	37
10. Bibliografia – wybrane pozycje	38

1. Streszczenie

Rozprawa poświęcona została tematyce badań w zakresie obliczeń masowo równoległych z zastosowaniem do wybranych problemów optymalizacji dynamicznej. W ramach pracy wykorzystano masowo równoległą platformę obliczeniową Nvidia CUDA (ang. Compute Unified Device Architecture). Sformułowano i rozwiązano przykładowy problem optymalizacji dynamicznej. Opracowano również szereg algorytmów masowo równoległych mających zastosowanie dla szerszej, niż docelowa, grupy problemów. W oparciu o wybrane istniejące koncepcje zrównoleglenia obliczeń zaprojektowano oraz przebadano dedykowane algorytmy dla platformy CUDA. W toku prowadzonych badań opracowano również algorytm bazujący na autorskim pomysle dotyczącym połączenia wcześniej wykorzystanych koncepcji organizacji obliczeń równoległych. Efektem takiego zabiegu powstał algorytm pozwalający znacznie lepiej wykorzystać potencjał masowo równoległej architektury CUDA w ściśle określonych przypadkach. W pracy zawarto szczegółowe opisy przeprowadzonych badań, przedstawiono wyniki w formie licznych tabel oraz wykresów. Ponadto w ramach rozprawy opracowano oraz przebadano algorytmy dedykowane dla CUDA, służące obliczaniu diagramów bifurkacji dla przykładowego modelu łuku elektrycznego. Co ważne, w ramach pracy obliczane były nie jedno, lecz dwu oraz trójwymiarowe diagramy bifurkacji o dużym wymiarze. Opracowane algorytmy CUDA porównywano z analogicznymi algorytmami sekwencyjnymi. Ponadto dla niektórych problemów rozwiązywanych w ramach niniejszej rozprawy dokonano również porównania opracowanych algorytmów masowo równoległych do algorytmów równoległych pracujących na wielordzeniowych CPU z wykorzystaniem technologii OpenMP.

Abstract:

The dissertation is devoted to research in the field of massively parallel computing with application in dynamic optimization. As part of the work, a massively parallel computing platform Nvidia CUDA (Compute Unified Device Architecture) was used. An exemplary problem of dynamic optimization was formulated and solved. A number of massively parallel algorithms have also been developed that are applicable to a wider than target group of problems. Based on selected existing concepts of parallelization of computations, dedicated algorithms for the CUDA platform were designed and researched. In the course of the research, an algorithm based on a proprietary solution was also developed, combining the previously used concepts of the parallel organization of calculations. An algorithm was developed that allows much better use of the potential of massively parallel CUDA architecture in strictly defined cases. The dissertation contains detailed descriptions of the conducted research, and presents the results in the form of numerous tables and charts. In addition, as part of the dissertation, algorithms for calculating bifurcation diagrams for an exemplary electric arc model, dedicated to CUDA, were developed and research. Importantly, as part of the work, not one, but two and three-dimensional bifurcation diagrams of large dimensions were calculated. The developed CUDA algorithms were compared with analogous sequential algorithms. In addition, for some of the problems solved in this dissertation, a comparison of the developed massively parallel algorithms to parallel algorithms working on multi-core CPUs with the use of OpenMP technology was also made.

2. Wprowadzenie do tematyki badań

Rozwój współczesnych technologii w obszarze Automatyki oraz Informatyki Przemysłowej, stwarza realną możliwość automatyzacji wielu procesów. Przebieg tych procesów bez udziału człowieka, lub z ograniczonym jego udziałem, prowadzi z kolei do potrzeby określenia pewnych kryteriów jakości, które muszą zostać spełnione aby osiągnięty efekt był satysfakcjonujący. Istnieje zatem uzasadniona potrzeba optymalizacji przebiegu wspomnianych procesów.

Pojęcie optymalizacji na przestrzeni ostatnich lat mocno się upowszechniło i ma dobrze ugruntowaną pozycję w wielu dziedzinach życia, nauki oraz techniki takich jak: projektowanie produktów, procesy produkcyjne, finanse, przemysł chemiczny, lotnictwo, motoryzacja, informatyka, automatyka oraz wiele innych dziedzin. Przykładem działań optymalizacyjnych może być maksymalizacja zysków przy jednoczesnym ograniczeniu kosztów, lub maksymalizacja wydajności przy jednoczesnym ograniczeniu zużycia paliwa.

W ujęciu formalnym pojęcie optymalizacji może być rozumiane jako "...postępowanie polegające na wyborze elementu z danego zbioru w oparciu o relacje ustalające pewien porządek w tym zbiorze", przy założeniu, że elementami wspomnianego zbioru są rozwiązania konkretnych problemów [1].

Natomiast w ujęciu bardziej ogólnym pojęcie optymalizacji odnosi się do procesu poszukiwania najlepszych rozwiązań napotkanego problemu przy jednoczesnym uwzględnieniu jego specyfiki. Rozwiązania te z kolei przekładać się będą na podejmowanie szeregu konkretnych decyzji. W praktyce za rozwiązanie najlepsze z matematycznego punktu widzenia uważa się minimum lub maksimum przyjętego wskaźnika jakości [2].

Pojęcie wskaźnika jakości jest jednym z podstawowych pojęć teorii optymalizacji. Wynika to z faktu, iż w przypadku gdy przedmiotem naszych rozważań będzie pewien konkretny proces, którego przebieg chcemy poddać działaniu optymalizacji, a którego efekt końcowy może być zmierzony przy wykorzystaniu pewnej funkcji skalarnej zwanej funkcjonałem kosztu lub funkcją celu, to odpowiednim postępowaniem jest taki dobór parametrów tego procesu, który będzie tę funkcję ekstremalizował (minimalizował/maksymalizował). Zatem w przypadku funkcjonału kosztu działaniem optymalizacyjnym będzie dążenie do minimalizacji bądź maksymalizacji jego wartości. Przykładem praktycznym zadania optymalizacji obrazującym rolę wskaźnika jakości może być podróżowanie z punktu A do punktu B, w przypadku którego ponoszony jest pewien koszt związany z podróżą, który należy zminimalizować. Jednakże w innym ujęciu problemem może być szybkość podróżowania, która z kolei wymagała będzie maksymalizacji [3].

Podstawowego podziału zadań optymalizacji możemy dokonać już w oparciu o naturę problemów jakich one dotyczą. Będzie to zatem podział na zadania optymalizacji statycznej oraz dynamicznej [1].

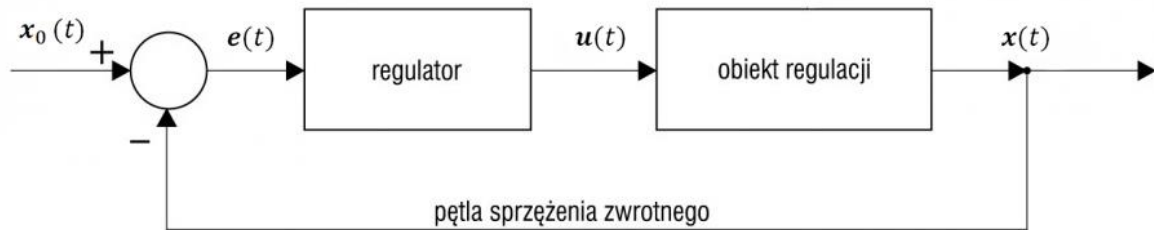
Optymalizacja statyczna w ogólnym ujęciu polega na poszukiwaniu rozwiązania optymalnego z punktu widzenia funkcji celu. W praktyce jest to poszukiwanie ekstremum funkcji, a więc wartości minimalnej lub maksymalnej w zależności od rodzaju zadania. Poszukiwanie ekstremum globalnego takiej funkcji mimo jego istnienia nie zawsze musi być skuteczne i często kończy się znalezieniem ekstremum lokalnego. Zatem skuteczność większości stosowanych algorytmów w dużym stopniu zależy od właściwego doboru punktu startowego. W obrębie optymalizacji statycznej wyróżnia się dwa podstawowe zagadnienia: programowanie liniowe oraz nieliniowe. W przypadku programowania liniowego wartością poszukiwaną będzie ekstremum liniowej funkcji celu, gdzie ograniczenia będą również funkcjami liniowymi. W przypadku programowania nieliniowego sytuacja jest odmienna gdyż tutaj funkcja celu, dla której szukamy ekstremum może mieć dowolną postać podobnie jak ograniczenia [1] [2].

Optymalizacja dynamiczna w automatyce „...polega na poszukiwaniu takiego sposobu zmian decyzji w danym przedziale czasu, który zapewni ekstremum pewnego wskaźnika jakości zależącego od przebiegu zmian tych decyzji ...” [2]. Zatem oddziaływanie odpowiedniego ciągu decyzji w danym przedziale czasu na określony proces fizyczny jest sterowaniem optymalnym. W odróżnieniu od optymalizacji statycznej, optymalizacja dynamiczna polega na poszukiwaniu funkcji sterowania a nie konkretnego punktu. Wskaźnikiem jakości jest w tym wypadku zazwyczaj funkcjonal wyrażony w postaci całki.

Pojęcie optymalizacji dynamicznej jest ściśle powiązane z pojęciem sterowania, które z kolei jest kluczowym problemem automatyki. W ogólnym ujęciu Automatyka jest nauką zajmującą się planowaniem oraz realizowaniem takiej strategii postępowania, która przeprowadzi obiekt sterowany z jednego stanu do innego, często przy spełnieniu dodatkowo określonych kryteriów jakościowych. Dlatego też zadania optymalizacji dynamicznej często określa się mianem zadań sterowania optymalnego [1] [2] [4].

Sterowanie może dotyczyć pojedynczych obiektów, układów obiektów lub całych systemów złożonych z obiektów sterowania. Poszukiwanie właściwego sterowania jest w Automatyce realizowane w oparciu o metody matematyczne modelowania statyki i dynamiki procesów, oraz o metody optymalizacji wybranych wskaźników jakości, reprezentujących najistotniejsze cechy z punktu widzenia procesu którym chcemy sterować. Dopiero po znalezieniu najlepszego sterowania możliwa jest jego implementacja, która najczęściej realizowana jest w oparciu o układ sprzężenia zwrotnego. Idea sprzężenia zwrotnego umożliwia prowadzenie ciągłej kontroli zachowania się procesu sterowanego. Sterowanie w układzie zamkniętym nazywamy regulacją automatyczną [1] [2] [4].

Rozpatrzmy zatem układ regulacji automatycznej złożony z regulatora i obiektu regulacji (Rys. 2.1)



Rys. 2.1. Schemat blokowy układu regulacji automatycznej (na podstawie [5])

Na przedstawionym schemacie porównanie wielkości regulowanej $x(t)$ z jej wartością zadaną $x_0(t)$ dokonywane jest w węzle sumacyjnym. Różnica $e(t) = x_0(t) - x(t)$ nazywana jest uchybem regulacji. Układ regulacji pracuje dobrze wówczas gdy mimo oddziaływania zakłóceń $z_1(t), \dots, z_r(t)$ uchyb regulacji $e(t)$ jest bliski wartości zerowej.

W ramach niniejszej rozprawy rozpatrywane są przypadki gdy obiekt regulacji opisany jest układem równań różniczkowych zwyczajnych (ang. Ordinary Differential Equations - ODE) opisanych wzorem (2.1) [4].

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \quad (2.1)$$

gdzie:

\mathbf{f} – funkcja wektorowa,

\mathbf{x} – wektor zmiennych stanu o składowych $[x_1, \dots, x_n] \mathbf{T}$,

\mathbf{u} – wektor sterowania $[u_1, \dots, u_m] \mathbf{T}$,

t – zmienna czasowa (t_0, t_n) .

Na wektor sterowania \mathbf{u} , zazwyczaj nałożone są dodatkowe warunki zwane ograniczeniami. W przypadku ogólnym zakładamy, że wielkości sterujące spełniają ograniczenia o postaci danej wzorem (2.2) [4].

$$g_i(u_1, u_2, \dots, u_m) \leq 0 \quad (i = 1, 2, \dots, r) \quad (2.2)$$

Wszystkie wielkości sterujące spełniające ograniczenia nazywamy dopuszczalnymi i zapisujemy w postaci danej wzorem (2.3). Tworzą one tak zwany zbiór sterowań dopuszczalnych.

$$\mathbf{u} \in D_u \quad (2.3)$$

gdzie:

D_u – zbiór sterowań dopuszczalnych

3. Sformułowanie celu, tezy oraz zakresu pracy

Efektywność równoległych implementacji algorytmów obliczeniowych w ogólności, w tym również algorytmów optymalizacji dynamicznej w znacznym stopniu zależy od struktury konkretnego algorytmu obliczeniowego, to znaczy jego tzw. „podatności” na zrównoleglenie, jak również od samej architektury sprzętu obliczeniowego, na którym algorytmy te mają być implementowane.

W obrębie algorytmów optymalizacji opartych na wykorzystaniu zasady maximum Pontriagina [1] [6], jednym z istotnych elementów składowych jest problem rozwiązywania układów równań stanu i równań sprzężonych. Badanie efektywności równoległych implementacji tego typu obliczeń jest przedmiotem niniejszej rozprawy doktorskiej. Jeżeli mamy do rozwiązania układy równań różniczkowych zwyczajnych o dużym wymiarze, wówczas zrównoleglenie tego typu obliczeń jest raczej oczywiste, jeżeli zamierza się wykorzystać nie więcej jednostek obliczeniowych niż wymiar układu. Ziarnistość takiego zadania można jeszcze zmniejszyć, wykorzystując znane w literaturze algorytmy równoległego obliczania wartości wyrażeń algebraicznych, przy czym ziarnistość tego podejścia jest również ograniczona, a dodatkowo dochodzą specyficzne wymagania odnośnie architektury sprzętu, która pozwoliłaby na osiągnięcie w takim podejściu zadowalających rezultatów w zakresie przyspieszenia obliczeń.

Problem staje się znacznie bardziej interesujący, jeżeli mamy do czynienia z niewielkim układem równań stanu i równań sprzężonych – w szczególnym przypadku może to być np. układ jednowymiarowy. W jaki sposób przyspieszyć obliczenia, których celem jest rozwiązanie jednego równania różniczkowego? Jeżeli dodatkowo postać funkcji występującej z prawej strony w równaniu różniczkowym nie pozwala na efektywne równoległe wyznaczanie jej wartości, pozostaje jedynie rozważenie zrównoleglenia obliczeń „względem czasu”. Pomysł ten nie jest nowy. W odniesieniu do równań różniczkowych został on podjęty w pracy [7]. W pracy tej opisano implementację takiego sposobu zrównoleglenia obliczeń przy wykorzystaniu systemu równoległego typu klastery, składającego się jedynie z siedmiu jednostek obliczeniowych, przy czym w większości realizowanych obliczeń wykorzystywanych było zaledwie 5 procesorów.

Kluczowym zadaniem niniejszej pracy doktorskiej jest zaprojektowanie, implementacja oraz przebadanie efektywności masowo równoległych realizacji algorytmów rozwiązywania równań różniczkowych zwyczajnych z zastosowanym zrównolegleniem obliczeń względem zmiennej czasu, przy czym implementacje te zaprojektowano i zrealizowano dla środowiska kart graficznych Nvidia CUDA (ang. Compute Unified Device Architecture) – dla procesorów GPU (ang. Graphics Processing Unit), gwarantujących znacznie większy poziom zrównoleglenia zadania (w kontekście liczby rdzeni/wątków), niż zostało to zrealizowane w ramach pracy [7].

Celem pracy doktorskiej jest:

- Opracowanie algorytmów numerycznego całkowania układów równań różniczkowych zwyczajnych, przy wykorzystaniu technologii Nvidia CUDA dla rozwiązywania złożonych zadań optymalizacji dynamicznej.
- Badanie efektywności opracowanych algorytmów ze zrównoleżeniem na poziomie układu równań oraz na poziomie pojedynczego równania.
- Zastosowanie badanych algorytmów do rozwiązania przykładowych, rozważanych w pracy problemów.

Tezę pracy można sformułować następująco:

Platforma Nvidia CUDA stwarza realne możliwości wyraźnej poprawy efektywności realizacji analizowanych w pracy metod i algorytmów optymalizacji dynamicznej poprzez znaczne skrócenie czasu całkowania układów równań różniczkowych zwyczajnych opisujących rozważane problemy dynamiczne.

Zakres pracy

Pracę rozpoczyna ogólne wprowadzenie czytelnika w tematykę realizowanych badań. Następnie sformułowany został w sposób ogólny problem optymalizacji dynamicznej oraz wprowadzone zostały niezbędne pojęcia podstawowe. Przedstawiono również przegląd metod i algorytmów rozwiązywania problemów optymalizacji dynamicznej. Opisano zarówno podstawowe metody analityczne o stosunkowo wąskim spektrum zastosowań jak również klasyczne metody numeryczne, które umożliwiają rozwiązywanie znacznie szerszej klasy problemów. W dalszej części przedstawiono przegląd klasycznych metod numerycznego rozwiązywania zagadnień początkowych dla równań różniczkowych zwyczajnych. Opisane zostały podstawowe metody jednokrokowe jak również metody wielokrokowe oraz metody typu predyktor-korektor. W ramach rozprawy opisano również wybrane zagadnienia dotyczące teorii obliczeń równoległych. Wprowadzone zostały podstawowe pojęcia, takie jak ziarnistość przetwarzania równoległego, komunikacja międzywątkowa czy przyspieszenie algorytmu równoległego. Przywołane i omówione zostały podstawowe prawa związane z obliczeniami równoległymi, takie jak Prawo Amdahla czy Prawo Gustafsona. Przedstawiono również podstawową klasyfikację komputerowych modeli przetwarzania danych. W części poświęconej technologii opisano historię rozwoju układów GPU oraz powstania idei GPGPU na przykładzie technologii Nvidia CUDA. Dalej zawarto opis specyfiki przetwarzania danych w obrębie technologii CUDA oraz wprowadzono niezbędne pojęcia podstawowe, zwracając

szczególną uwagę na kwestie mające istotny wpływ na projektowanie kodu dedykowanego dla tej platformy.

W zasadniczej części rozprawy sformułowano rozpatrywane problemy oraz przedstawiono sposób ich rozwiązywania. Prezentowane wyniki badań poprzedzone są zawsze szczegółowymi opisami realizacji obliczeń oraz niezbędnymi wprowadzeniami teoretycznymi dotyczącymi rozpatrywanych zagadnień. W pierwszej kolejności dokonano opisu zrealizowanych w ramach niniejszej rozprawy masowo równoległych realizacji klasycznych algorytmów rozwiązywania układów równań różniczkowych zwyczajnych w technologii Nvidia CUDA. Została tutaj przedstawiona idea na której bazują opracowane algorytmy jak również ogólny schemat działania algorytmów w ujęciu kilku etapów. Ponadto przedstawione i omówione zostały pomiary czasu pracy poszczególnych algorytmów w ujęciu kilku odmiennych przypadków. Dalsza część pracy stanowi teoretyczne wprowadzenie do koncepcji Spekulacyjnej jako propozycji równoległego rozwiązywania pojedynczych równań różniczkowych zwyczajnych lub niewielkich ich układów. Następnie przedstawiono opracowany w ramach niniejszej rozprawy algorytm dedykowany dla platformy Nvidia CUDA, który zaprojektowany został w oparciu o założenia koncepcji Spekulacyjnej. W tym przypadku również opisano schemat pracy algorytmu w ujęciu kilku etapów. Ponadto przedstawiono i omówiono pomiary czasu pracy algorytmu dla kilku przypadków. Ostatecznie sformułowano również opis algorytmu zrealizowanego na podstawie obu wykorzystanych wcześniej koncepcji. Powstała w ten sposób koncepcja Kombinowana. Choć sama idea łączenia różnych typów zrównoleglenia obliczeń w postaci jednego algorytmu nie jest nowa, to zaproponowana koncepcja Kombinowana w kontekście metody Spekulacyjnej oraz wykorzystanej technologii CUDA jest zupełnie nowym pomysłem autora niniejszej rozprawy. Koncepcja Kombinowana wprowadza co prawda w małym stopniu ograniczoną skalowalność na fizycznie odrębne GPU, jednak pozwala dużo lepiej wykorzystać dostępne rdzenie kiedy dysponujemy znaczną ich liczbą a jednocześnie **nie chcemy modyfikować parametrów zadania obliczeniowego !** Przedstawione zostały wyniki pomiarów czasu pracy algorytmu oraz płynące z nich wnioski.

Sformułowano został jednowymiarowy problem optymalizacji dynamicznej do rozwiązania którego zastosowana została metoda Gradientu Prostej w Przestrzeni Funkcyjnej Sterowań. Sformułowano również problem generowania diagramów Bifurkacyjnych dla układów łuku elektrycznego. Opisano sposób realizacji algorytmów dedykowanych dla CUDA rozwiązujących sformułowane problemy oraz przedstawiono wyniki obliczeń. W przypadku algorytmów obliczających diagramy bifurkacji przedstawiono również stosowne diagramy zarówno w formie 2D jak również w formie 3D – w postaci chmury bifurkacji oraz jej przekrojów.

Na końcu rozprawy znajduje się podsumowanie zrealizowanych w ramach pracy badań oraz sformułowanie wniosków końcowych. Spisano również w punktach wkład własny autora pracy oraz nakreślono kierunki dalszych badań.

Rozprawa zawiera również dwa dodatki:

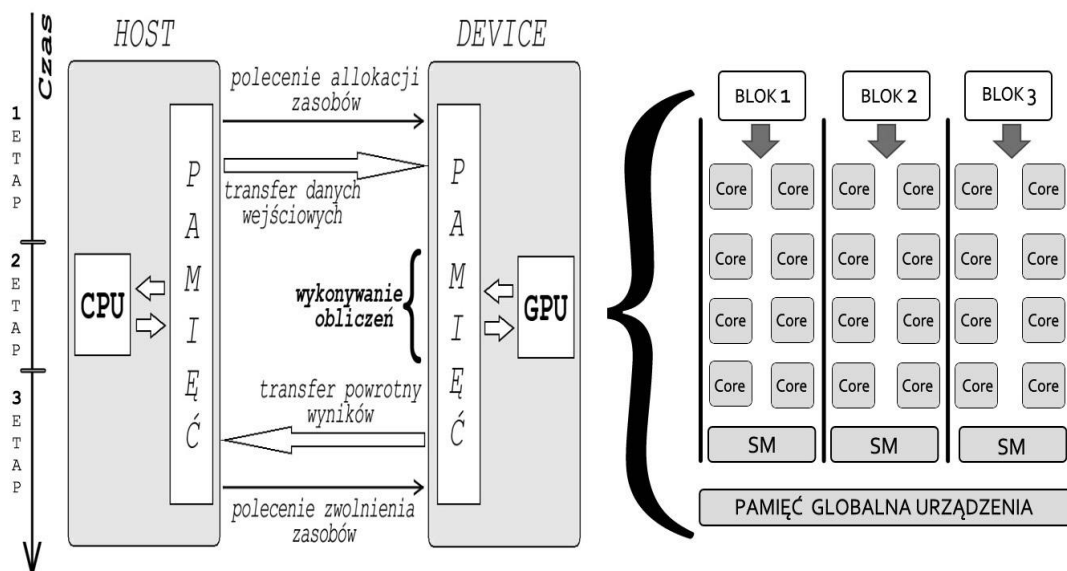
Dodatek A zawiera opis wybranych aspektów technicznych związanych z przeprowadzeniem obliczeń, które zostały zrealizowane w ramach pracy. Znajduje się tutaj opis środowiska badawczego oraz uruchomieniowego, które zostało zaprojektowane specjalnie na potrzeby niniejszej pracy. Przedstawione zostały wybrane cechy tego środowiska, jego funkcjonalności oraz w dużym uproszczeniu opisana została struktura jego budowy, pozwalająca czytelnikowi lepiej zrozumieć w jaki sposób prowadzono obliczenia i dokonywano szczegółowych pomiarów czasu pracy algorytmów.

Dodatek B zawiera natomiast opis trudności technicznych jakie napotkano podczas projektowania oraz implementacji algorytmów dedykowanych dla platformy Nvidia CUDA. Biorąc pod uwagę specyfikę przetwarzania danych oraz architekturę urządzeń CUDA istnieje szereg ograniczeń, które należy wziąć pod uwagę już na samym początku pracy nad algorytmem. Opisano między innymi ograniczenia związane z synchronizacją i komunikacją między wątkami oraz ograniczenia w dostępie do pamięci. Przedstawiono również problem efektywnego wykorzystania GPU na przykładzie opracowanych w ramach niniejszej rozprawy algorytmów. Analizę wyników wzbogacono wykorzystując narzędzie o nazwie CUDA Occupancy Calculator dostarczone przez Nvidię.

4. Masowo równoległa organizacja obliczeń w technologii Nvidia CUDA

Technologia Nvidia CUDA stanowi platformę obliczeń masowo równoległych, bazującą na wykorzystaniu układów GPU. Zgodnie z Taksonomią Flynna architekturę układów GPU należy zaklasyfikować jako SIMD (ang. Single Instruction, Multiple Data), ponieważ realizują one wykonywanie pojedynczej instrukcji na wielu danych [8] [9]. Nvidia jednak idzie o krok dalej wprowadzając własną unikalną nazwę architektury dla swoich układów. SIMT (ang. Single-Instruction, Multiple-Thread), bo o niej mowa, stanowi koncepcję bazującą na połączeniu architektury SIMD oraz wielowątkowości. Dzięki zastosowaniu technologii lekkich wątków w obrębie tej architektury, możliwe jest efektywne wykorzystanie bardzo dużej liczby rdzeni wykonawczych GPU. Wątki nazywane są lekkimi ponieważ przełączanie kontekstu pomiędzy nimi odbywa się bez opóźnień. Jest to możliwe ponieważ konteksty wszystkich aktywnych wątków przechowywane są w odpowiednio dużej liczbie rejestrów roboczych GPU.

Obliczenia na platformie CUDA realizowane są w oparciu o heterogeniczny model przetwarzania (Rys. 4.1). Oznacza to iż urządzenie CUDA zainstalowane jest w systemie hosta jako odrębny podsystem obliczeniowy ze swoimi odrębnymi zasobami.

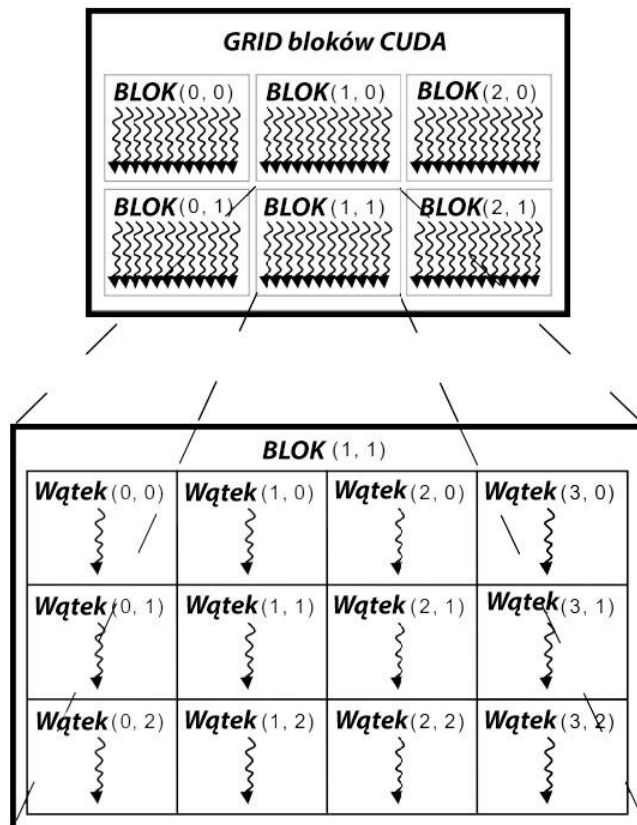


Rys. 4.1. Heterogeniczny model przetwarzania danych Nvidia CUDA (opracowanie własne na podstawie [10])

Wykonywanie obliczeń na urządzeniu CUDA zawsze wymaga sterowania ze strony hosta, oraz kopiowania danych pomiędzy hostem i urządzeniem CUDA. W typowym cyklu

przetwarzania danych wyróżnić można zatem trzy etapy (Rys. 4.1). Pierwszy etap stanowi przygotowanie do właściwych obliczeń i obejmuje alokację pamięci na urządzeniu CUDA, oraz transfer danych wejściowych do pamięci globalnej urządzenia. Drugi etap to wywołanie właściwego kodu wykonywanego przez GPU, tak zwanego kernela i wykonanie właściwych obliczeń. Trzeci etap natomiast odbywa się po zakończeniu pracy kernela i obejmuje transfer wyników obliczeń do pamięci hosta oraz zwalnianie pamięci urządzenia CUDA [10] [11] [12]. Przedstawione trzy etapy przetwarzania danych stanowią szkielet, na bazie którego zaprojektowano wszystkie algorytmy zrealizowane w toku przygotowania rozprawy.

Projektując kernel CUDA programista może dokładnie określić sposób adresowania wątków wykorzystując każdy z trzech dostępnych wymiarów (x,y,z). Adres wątku w obrębie bloku określany jest strukturą o nazwie **threadIdx** z polami x,y,z. Przykład dwuwymiarowego adresowania wątków w bloku oraz adresowania bloków przedstawiono na rysunku 4.2.



Rys. 4.2. Model adresowania wątków oraz bloków Nvidia CUDA. Zarówno w przypadku wątków jak i bloków zobrazowano jedynie 2 z 3 dostępnych wymiarów adresowania [10] [13].

5. Masowo równoległe realizacje klasycznych algorytmów rozwiązywania układów równań różniczkowych zwyczajnych w technologii Nvidia CUDA

W ramach opracowanych algorytmów masowo równoległych prezentowanych w rozprawie zaimplementowane zostały trzy dobrze znane klasyczne, sekwencyjne, metody numeryczne rozwiązywania ODE: metoda Eulera [14] [15], metoda Rungego- Kuty II rzędu [15] [16], oraz metoda Rungego-Kuty IV rzędu [15] [17]. Metoda Eulera choć wysoce nieefektywna i obciążona dużym błędem całkowania, jest tutaj ważnym elementem odniesienia ukazującym wpływ algorytmu o stosunkowo małej złożoności obliczeniowej na efektywność wykorzystania platformy CUDA w stosunku do prezentowanych algorytmów o większej złożoności. Prezentowane w niniejszym rozdziale algorytmy bazują na wykorzystaniu metod stałokrokowych dzięki czemu możliwe jest ukazanie w prosty sposób wpływu złożoności obliczeń na efektywność wykorzystania technologii Nvidia CUDA. Algorytmy prezentowane w niniejszym rozdziale wykorzystują ideę bazującą na dekompozycji układu równań na pojedyncze równania bądź ich grupy (Rys. 5.1). Ostateczny sposób dystrybucji równań pomiędzy rdzenie CUDA uzależniony jest od liczby równań w stosunku do liczby dostępnych rdzeni. Równania obliczane są równoległe lecz niezbędna jest komunikacja międzywątkowa w każdym kroku całkowania. Ze względu na wymaganą komunikację istnieją dla tej koncepcji poważne ograniczenia w zakresie skalowalności obliczeń na wiele odrębnych GPU.

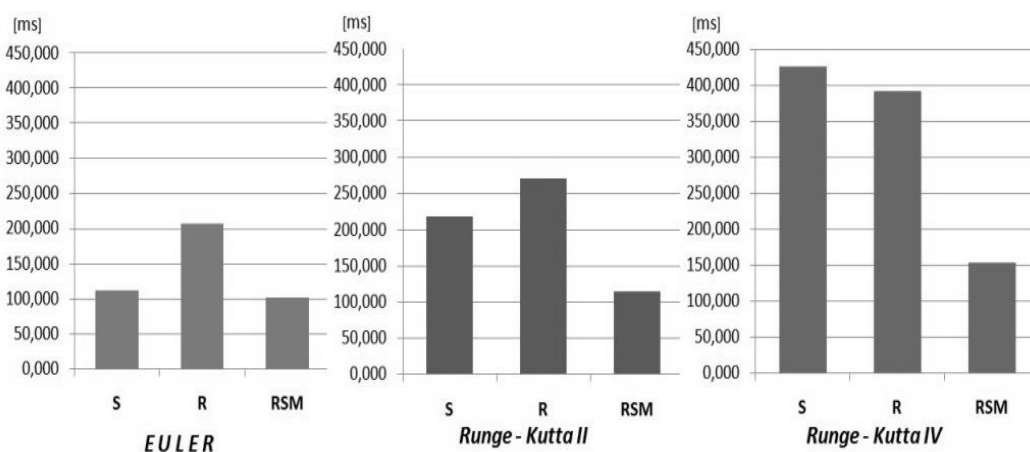
$$\begin{array}{l} \text{Procesor 1} \leftarrow \dot{x}_1 = f_1(x_1, x_2, \dots, x_N, t) \\ \text{Procesor 2} \leftarrow \dot{x}_2 = f_2(x_1, x_2, \dots, x_N, t) \\ \vdots \quad \dots \\ \text{Procesor } N \leftarrow \dot{x}_N = f_N(x_1, x_2, \dots, x_N, t) \end{array}$$

Rys. 5.1. Dystrybucja równań układu pomiędzy procesory / rdzenie CUDA (opracowanie własne na podstawie [7])

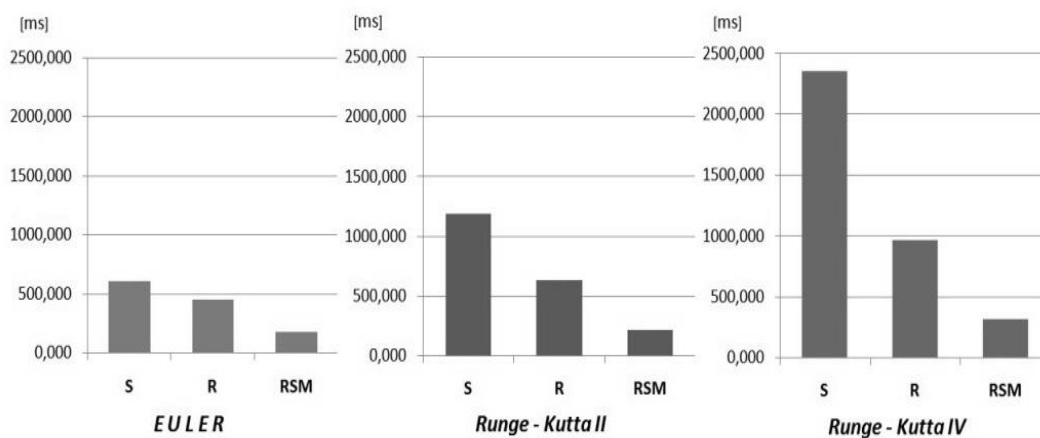
Sposób organizacji obliczeń leżący u podstaw omawianej koncepcji klasyfikuje ją do zastosowania w sytuacji kiedy liczba równań w układzie jest dostatecznie duża aby dokonać efektywnej dystrybucji równań pomiędzy rdzenie CUDA. Warto wspomnieć, iż możliwe jest również osiągnięcie równoległości na poziomie realizacji samej metody numerycznej.

Takie podejście oferuje jednak zbyt mały stopień zrównoleglenia aby mogło być brane pod uwagę w kontekście technologii masowo równoległej jaką jest CUDA.

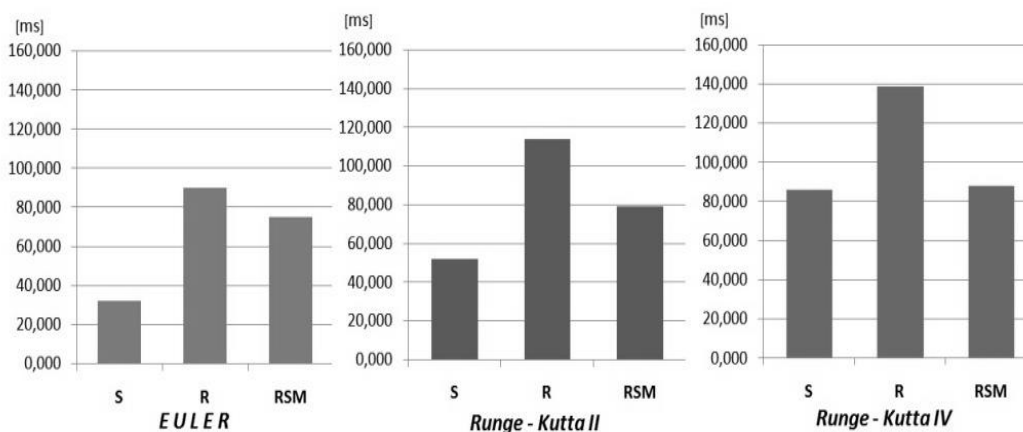
Na rysunkach 5.2, 5.3 oraz 5.4 zestawiono czasy realizacji opracowanych algorytmów dla kilku wybranych przypadków. Uwzględniono czasy realizacji algorytmów sekwencyjnych (S), równoległych (R) oraz równoległych wykorzystujących szybką pamięć współdzieloną (RSM). Dokonano również podziału algorytmów w zależności od zastosowanej metody numerycznej (Euler, Runge-Kutta II, Runge-Kutta IV). Dla przypadku pierwszego (Rys. 5.2) liczba równań w układzie ODE wynosiła 120. Dla przypadku drugiego (Rys.4.3) liczba równań w układzie wynosiła 300. Natomiast dla przypadku trzeciego (Rys. 4.4) liczba równań w układzie wynosiła zaledwie 50. Urządzenie obliczeniowe CUDA dysponowało liczbą rdzeni wynoszącą 768. Rozwiązywane równania generowane były w sposób losowy przy pomocy odpowiedniego modułu opracowanego środowiska badawczego.



Rys. 5.2. Zestawienie czasów podanych w milisekundach dla układu 120 równań



Rys. 5.3. Zestawienie czasów podanych w milisekundach dla układu 300 równań



Rys. 5.4. Zestawienie czasów podanych w milisekundach dla układu 50 równań

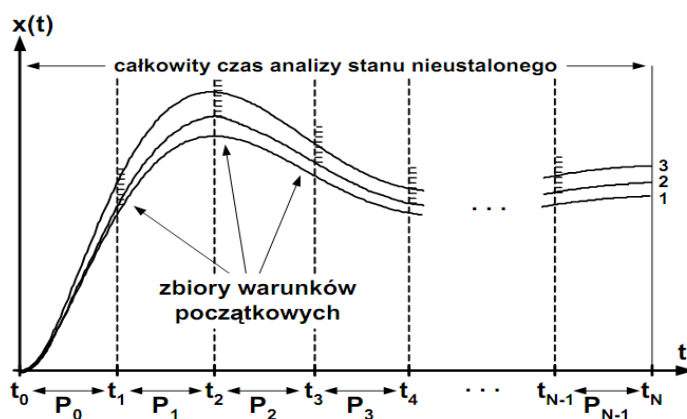
Przypadek trzeci (Rys. 5.4) jest negatywnym przykładem wykorzystania technologii Nvidia CUDA. Nie uzyskano tutaj przyspieszenia dla żadnego z prezentowanych algorytmów masowo równoległych a winą za taki stan rzeczy należy obarczyć zbyt mały stopień zrównoleglenia problemu. Inaczej rzecz ujmując zbyt mały wymiar rozwiązywanego zadania (zaledwie 50 równań) uniemożliwił efektywne obciążenie GPU wystarczającą liczbą wątków.

Opracowana grupa algorytmów masowo równoległych, rozwiązywania układów ODE pozwoliła na osiągnięcie realnego wzrostu wydajności obliczeń względem analogicznych algorytmów sekwencyjnych. Na szczególną uwagę zasługuje tutaj dobre wykorzystanie pamięci współdzielonej, która pozwoliła osiągnąć jeszcze lepsze wyniki. Należy jednak zauważyć, że aby osiągnąć zadowalające przyspieszenie obliczeń muszą być spełnione określone kryteria. Wymagany jest między innymi odpowiednio duży wymiar zadania obliczeniowego, zastosowanie metody numerycznej o odpowiednio dużej złożoności obliczeniowej oraz wynikający z tego faktu recykling danych w obrębie urządzenia obliczeniowego CUDA.

6. Masowo równoległy algorytm oparty na założeniach koncepcji Spekulacyjnej dedykowany dla technologii Nvidia CUDA

Technologia Nvidia CUDA oferuje bardzo dużą liczbę rdzeni wykonawczych w stosunku z klasycznymi układami CPU. Powyższy fakt pozwala zakwalifikować CUDA jako środowisko wykonawcze dla wszelkich algorytmów wykorzystujących koncepcję Spekulacyjną rozwiązywania ODE. Koncepcja ta zakłada bowiem dekompozycję przedziału całkowania na szereg podprzedziałów stanowiących odrębne zadania, z których każde może być rozwiązywane niezależnie, na osobnym rdzeniu, jedną ze znanych klasycznych metod sekwencyjnych. Podejście takie jest jednak dosyć kosztowne obliczeniowo ponieważ wymaga między innymi wyznaczenia wartości szukanych funkcji na początku każdego podprzedziału przed przystąpieniem do właściwego rozwiązywania układu równań.

O dużym koszcie obliczeniowym omawianej koncepcji dodatkowo świadczy sposób w jaki konstruowane jest rozwiązanie końcowe. Powstaje ono na zasadzie wyboru z każdego podprzedziału tylko jednego rozwiązania cząstkowego na zasadzie dopasowania podprzedziałów. Pozostałe rozwiązania cząstkowe są usuwane z pamięci jako nieprzydatne.



Rys. 6.1. Dekompozycja problemu względem czasu [7]

Prezentowana idea była ostatnio rozwijana w pracy [7] pod nazwą Metody Spekulacyjnej, w której zastosowano między innymi metodę Rungego-Kutty IV rzędu [15] [17] [18].

Teoretycznie rzecz biorąc niektóre modele współczesnych wieloprocessorowych urządzeń obliczeniowych CUDA posiadają znacznie większą liczbę rdzeni wykonawczych, niż jest to potrzebne w przypadku zastosowania koncepcji Spekulacyjnej i pojedynczego równania. Pozostawia to zatem pewną rezerwę dla ewentualnego zwiększenia liczby spekulatywnych układów równań jeśli zajdzie taka potrzeba w celu zwiększenia dokładności rozwiązania problemu. Jednocześnie zmiana taka nie powinna pociągnąć za sobą wzrostu

czasu realizacji algorytmu. Problemem natomiast może okazać się chęć nadmiernego zwiększenia liczby przyjętych podprzedziałów całkowania, ponieważ powoduje to jednocześnie przesunięcie balansu pracy w kierunku hosta i w efekcie może znacząco wydłużyć czas realizacji algorytmu jako całości. Ponadto większa liczba podprzedziałów to również konieczność transferu większej liczby warunków początkowych na styku każdego z nich do urządzenia CUDA.

Kwestia odpowiedniego doboru parametrów startowych dla algorytmu opartego na koncepcji Spekulacyjnej jest na tyle istotna w kontekście algorytmu dedykowanego dla technologii CUDA, iż należy poświęcić jej nieco więcej uwagi. Poniżej przedstawiono ten problem w ujęciu 5 przypadków obrazując w jaki sposób poszczególne wartości parametrów startowych wpływają na uzyskane rozwiązanie końcowe dla koncepcji Spekulacyjnej bazującej na zastosowaniu metody Rungego Kutty IV rzędu. W poniższym przykładzie rozwiązywane jest typowe równanie różniczkowe jednorodne dane wzorem 6.1.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) \quad (6.1)$$

gdzie:

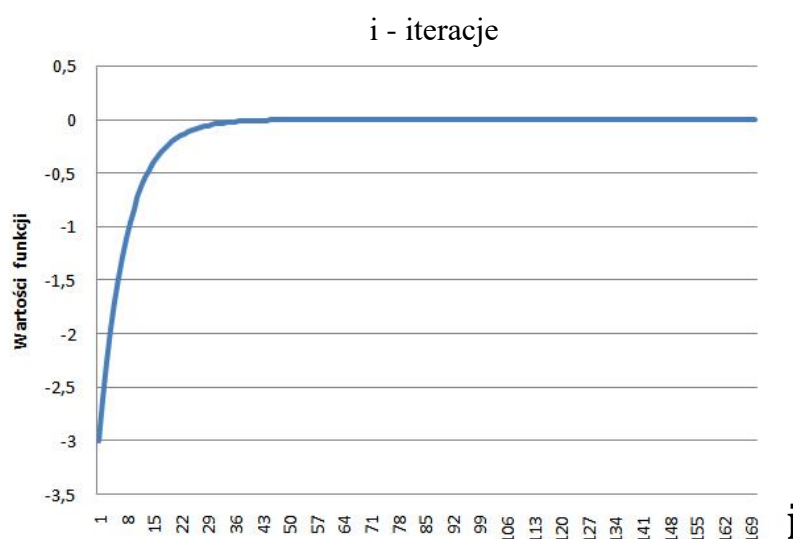
A – macierz współczynników,

\mathbf{x} – wektor zmiennych stanu,

t – zmienna niezależna.

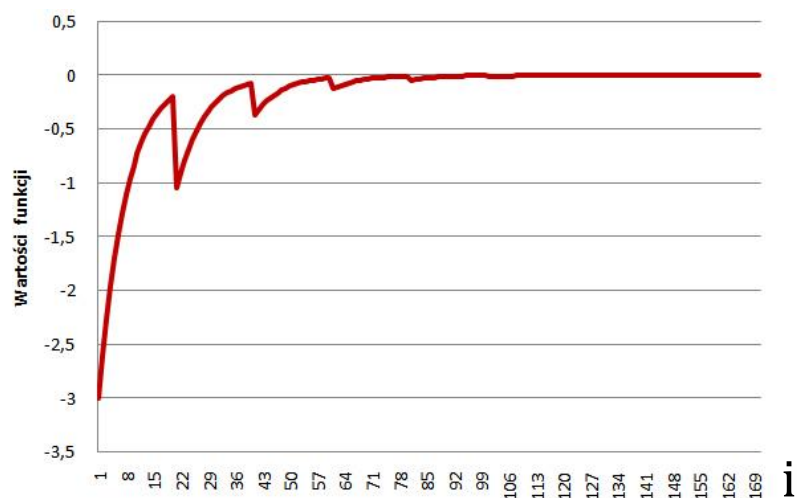
- Przypadek 1:

- liczba kroków całkowania 400



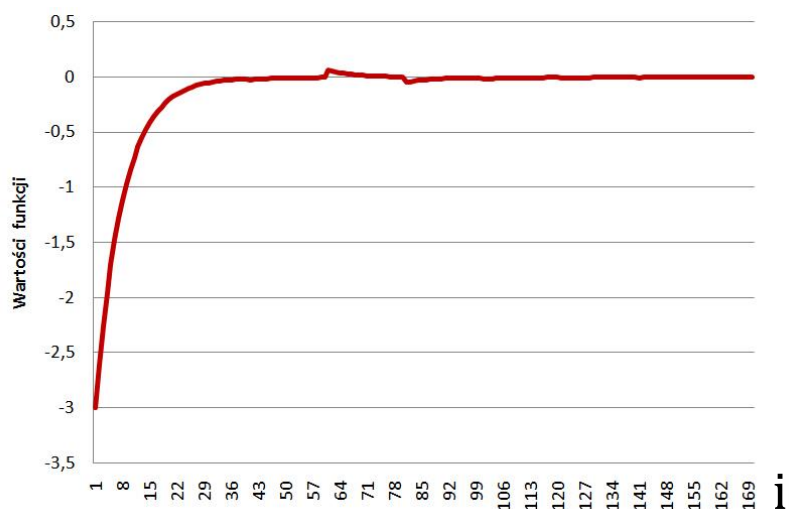
Rys. 6.2. Rozwiązanie sekwencyjne uzyskane metodą Rungego Kutty IV

- Przypadek 2:
 - liczba kroków całkowania 400
 - liczba podprzedziałów 20
 - liczba spekulatywnych rozwiązań 1



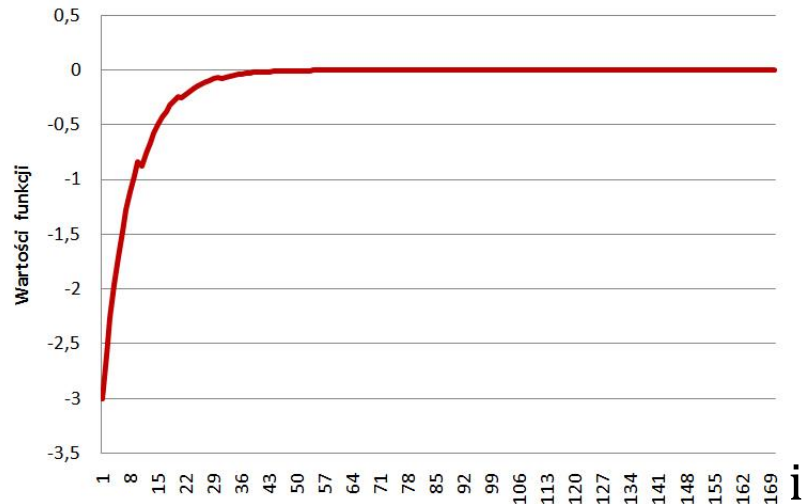
Rys. 6.3. Rozwiązanie uzyskane na podstawie algorytmu równoległego metodą Spekulacyjną przy wykorzystaniu metody Rungego Kutty IV. Uwidocznione duże amplitudy wartości funkcji na łączeniach podprzedziałów.

- Przypadek 3:
 - liczba kroków całkowania 400
 - liczba podprzedziałów 20
 - liczba spekulatywnych rozwiązań 500



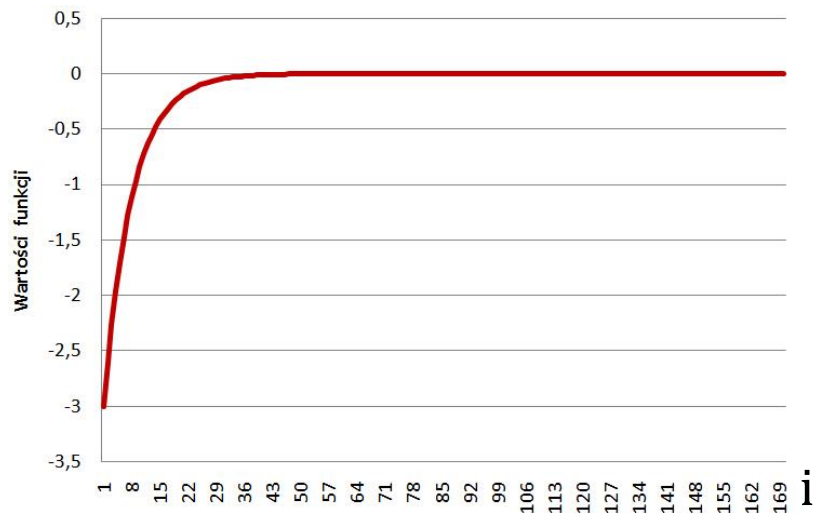
Rys. 6.4. Dzięki zwiększeniu liczby rozwiązań spekulacyjnych uzyskano znaczące zmniejszenie amplitudy wartości funkcji na łączeniach podprzedziałów w stosunku do rys. 6.3

- Przypadek 4:
 - liczba kroków całkowania 400
 - liczba podprzedziałów 40
 - liczba spekulatywnych rozwiązań 1



Rys. 6.5. Dzięki zwiększeniu liczby podprzedziałów uzyskano znaczące zmniejszenie amplitudy wartości funkcji na łączeniach podprzedziałów w stosunku do rys. 6.3

- Przypadek 5:
 - liczba kroków całkowania 400
 - liczba podprzedziałów 40
 - liczba spekulatywnych rozwiązań 500



Rys. 6.6. Zastosowano jednoczesne zwiększenia liczby podprzedziałów oraz liczby rozwiązań spekulatywnych w stosunku do rys. 6.3

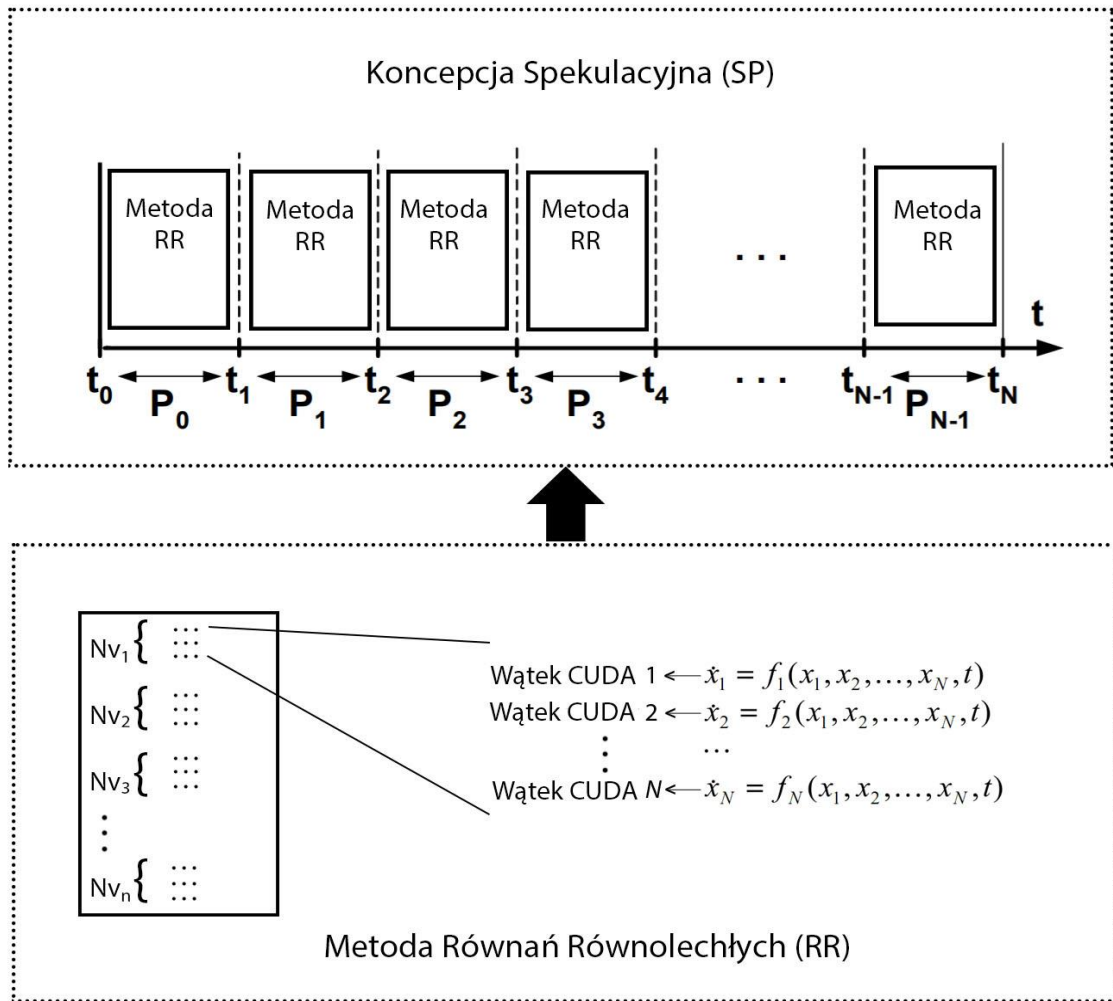
Analizując przedstawione przypadki zauważyć należy, iż znaczącą poprawę zbieżności rozwiązania spekulacyjnego do rozwiązania sekwencyjnego uzyskano już w przypadku 3 za sprawą zwiększenia liczby rozwiązań spekulatywnych. Podobnie w przypadku 4 uzyskano poprawę rozwiązania zwiększając liczbę przyjętych podprzedziałów. Najlepszy efekt uzyskano jednak zwiększając oba te parametry jednocześnie w przypadku 5. W oparciu o zaprezentowane wykresy rozwiązań dla przedstawionych przypadków uwidoczniono, iż algorytm jest znacznie bardziej wrażliwy na zmianę liczby podprzedziałów niż na zmianę liczby spekulatywnych rozwiązań. Jak wcześniej wspomniano ze względu na dużą liczbę rdzeni CUDA teoretycznie można podnosić liczbę rozwiązań spekulatywnych do pewnego stopnia bezkosztowo. W praktyce jednak dużą rolę odgrywa również przyjęty horyzont czasowy oraz związana z nim liczba kroków całkowania.

Chcąc zapewnić odpowiednio duży stopień zrównoleglenia w przypadku dużej liczby kroków całkowania dążyć będziemy również do zwiększenia liczby przyjętych podprzedziałów. Finalnie zatem każdy problem numeryczny który ma być rozwiązywany przy użyciu tego algorytmu musi zostać poddany analizie a parametry startowe muszą zostać dobrane w pewnym stopniu empirycznie.

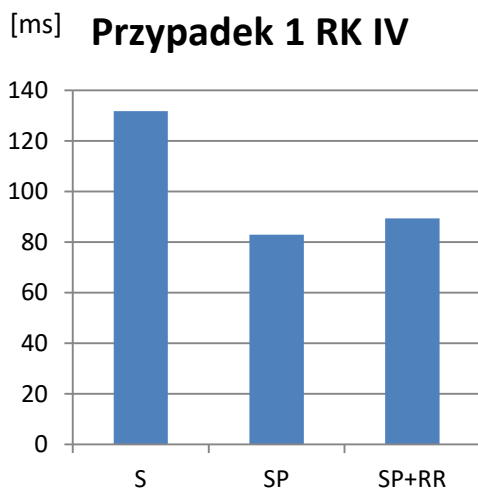
7. Masowo równoległe realizacje algorytmów rozwiązywania układów równań różniczkowych zwyczajnych oparte na koncepcji równoległego rozwiązywania równań oraz na koncepcji Spekulacyjnej

Trzeci opracowany w ramach rozprawy algorytm rozwiązywania ODE jest w zasadzie pewną odmianą algorytmu drugiego, opartego na wykorzystaniu koncepcji Spekulacyjnej. Różnica pomiędzy nimi polega na innej realizacji pracy samego kernela obliczeniowego CUDA. Mimo tego, że użyto liczby pojedynczej w odniesieniu do algorytmu, właściwie jest to grupa algorytmów uwzględniająca, tak jak to miało miejsce w pozostałych przypadkach, warianty dla poszczególnych metod numerycznych wykorzystywanych w ramach niniejszej rozprawy. Zauważono, iż w przypadku gdy mamy do rozwiązania więcej niż jedno równanie, wówczas możemy uzyskać większy stopień zrównoleglenia obliczeń stosując wewnątrz każdego z podprzedziałów koncepcję opartą na równoległym rozwiązywaniu równań. Trzeci algorytm bazuje zatem na dwóch koncepcjach jednocześnie, koncepcji równań równoległych oraz koncepcji Spekulacyjnej. Jest to zatem algorytm Kombinowany, który globalnie posiada cechy i ograniczenia algorytmu bazującego na koncepcji Spekulacyjnej. Lokalnie w podprzedziałach zachowuje się natomiast zupełnie jak typowy algorytm oparty na dekompozycji układu równań na pojedyncze równania. Schemat ideowy realizacji algorytmu kombinowanego przedstawia rysunek 7.1. Algorytm ten umożliwi zatem zwiększenie stopnia ziarnistości obliczeń gdy jest to wskazane w celu lepszego dopasowania problemu do architektury CUDA i zagospodarowania wszystkich dostępnych jednostek wykonawczych.

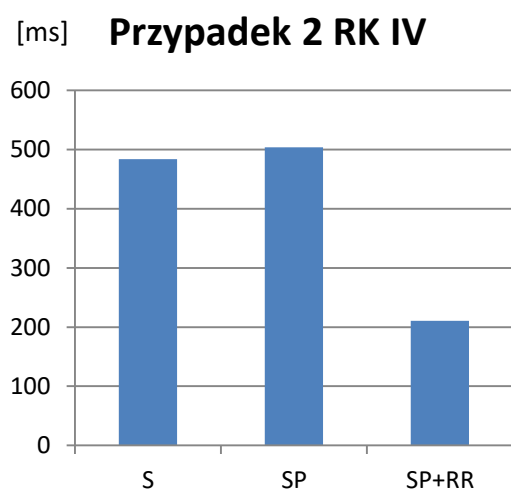
Poniżej przedstawiono trzy wybrane przypadki analizy czasu pracy opracowanych algorytmów, podczas gdy rozwiązywane było pojedyncze równanie lub układ równań ODE generowany losowo. Czas realizacji algorytmu kombinowanego przedstawiono na wykresach w kontekście pozostałych algorytmów. Rysunek 7.2 przedstawia pesymistyczny wariant wykorzystania algorytmu kombinowanego (SP+RR) względem algorytmu SP (Spekulacyjnego). W tym przypadku algorytm kombinowany SP+RR jest co prawda szybszy od sekwencyjnego S, jednak wykonuje się dłużej od SP. Sytuacja taka wynika z faktu, iż rozwiązywane jest tylko jedno równanie i nie można wykorzystać koncepcji równań równoległych wewnątrz podprzedziałów całkowania. Dla wszystkich prezentowanych przypadków zastosowano algorytmy masowo równoległe CUDA bazujące na wykorzystaniu metody Rungego-Kutty IV rzędu.



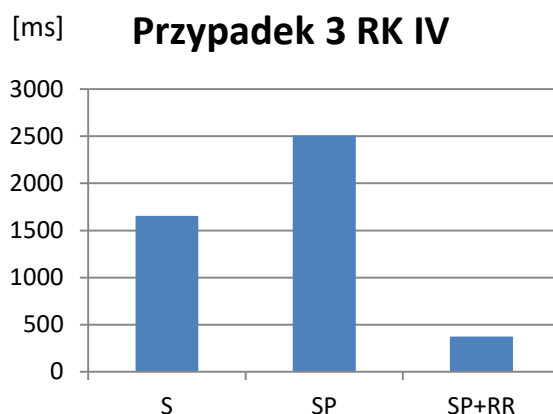
Rys. 7.1. Schemat idealowy realizacji algorytmu w oparciu o autorską koncepcję kombinowaną. Poprzez N_v oznaczono kolejne spekulatywne układy równań.



Rys. 7.2. Zestawienie czasu pracy badanych algorytmów dla przypadku 1 (1 równanie)



Rys. 7.3. Zestawienie czasu pracy badanych algorytmów dla przypadku 2 (3 równania)



Rys. 7.4. Zestawienie czasu pracy badanych algorytmów dla przypadku 3 (6 równań)

Ciekawe zestawienie stanowi rysunek 7.4, który obrazuje wyraźne skrócenie czasu realizacji algorytmu kombinowanego (SP+RR) względem pozostałych algorytmów. Skrócenie czasu dla algorytmu kombinowanego wynika z faktu zwiększenia stopnia ziarnistości obliczeń przy jednoczesnym zachowaniu tych samych wartości głównych parametrów startowych dla algorytmu SP+RR względem SP. Ponieważ wartości parametrów startowych nie uległy zmianie dokładność rozwiązania również została zachowana. Na szczególną uwagę podczas analizy rysunku 7.3 zasługuje jednak fakt iż algorytm SP pracował dłużej niż algorytm sekwencyjny S. Winę za taki stan rzeczy ponosi zbyt małe uzyskane przyspieszenie równoległej części algorytmu SP, które zostało całkowicie zniwelowane przez koszt wykonania części sekwencyjnej odpowiedzialnej za przygotowanie zbiorów warunków początkowych oraz składanie rozwiązania końcowego.

Przypadek 3 przedstawiony na rysunku 7.4 podkreśla niekorzystny dobór parametrów startowych algorytmu SP dla zadania zawierającego aż 6 równań w układzie. Jednocześnie przy zachowaniu tych samych parametrów startowych kombinowana wersja algorytmu SP+RR zyskuje zdecydowaną przewagę nad dwoma pozostałymi.

Podsumowując wyniki wszystkich algorytmów zauważyć należy, iż w przypadku 2 i 3 algorytm SP+RR pozwolił na znacznie większe zrównoleglenie obliczeń niż algorytm SP co czyni go znacznie szybszym nie tylko od algorytmu SP, ale również od S. W przypadku pierwszym natomiast najszybszy okazał się algorytm SP ponieważ rozwiązywane było tylko jedno równanie.

8. Rozpatrywane w pracy problemy obliczeniowe

8.1 Jednowymiarowy problem optymalizacji -Metoda Gradientu Prostego

W ramach niniejszej rozprawy rozpatrywany był jednowymiarowy problem optymalizacji dynamicznej sformułowany poniżej. W celu rozwiązania postawionego zadania optymalizacji opracowano i zastosowano algorytm masowo równoległy dedykowany dla platformy obliczeniowej Nvidia CUDA. Algorytm opracowany został w oparciu o wykorzystanie Metody Gradientu w Przestrzeni Funkcyjnej Sterowań a zatem bazuje również na wykorzystaniu Zasady Maksimum Pontriagina. Rozpatrywany przykład skonstruowany został w oparciu o analogiczny problem rozpatrywany w pracy [1].

Obiekt regulacji będący przedmiotem zadania optymalizacji opisany został równaniem 8.1

$$\frac{d}{dt}(x) = u, \quad x(0) = 8, \quad t = \langle 0, 10 \rangle \quad (8.1)$$

Wskaźnik jakości dla bieżącego zadania optymalizacji wyrażony został wzorem (8.2) W rozważanym przykładzie należy wyznaczyć takie sterowanie $u(t)$, które, przy spełnieniu równań stanu i równań sprzężonych gwarantuje uzyskanie minimalnej wartości przyjętego wskaźnika jakości.

$$Q\{x, u\} = 2,5[(x(10) - 2)^2] + \int_{t_0}^{t_k} (x^2 + u^2)dt \quad (8.2)$$

Mamy zatem do czynienia z problemem Bolzy. Podstawowe warianty wskaźnika jakości dla zadań optymalizacji dynamicznej opisane zostały w rozdziale 2 niniejszej rozprawy.

Hamiltonian zadania optymalizacji wyrażony został wzorem (8.3).

$$H(\Psi, x, u, t) = -(x^2, u^2) + \Psi u \quad (8.3)$$

Korzystając z Hamiltonianu zapisujemy równanie sprzężone w postaci (7.4)

$$\dot{\Psi} = -2x \quad (8.4)$$

oraz gradient Hamiltonianu w postaci (7.5)

$$\gamma = -\frac{\partial H}{\partial u} = 2u - \Psi \quad (8.5)$$

Warunki transversalności natomiast zostały zdefiniowane dla bieżącego zadania w postaci (8.6)

$$\Psi(t_k) = 10 - 5x(t_k) \quad (8.6)$$

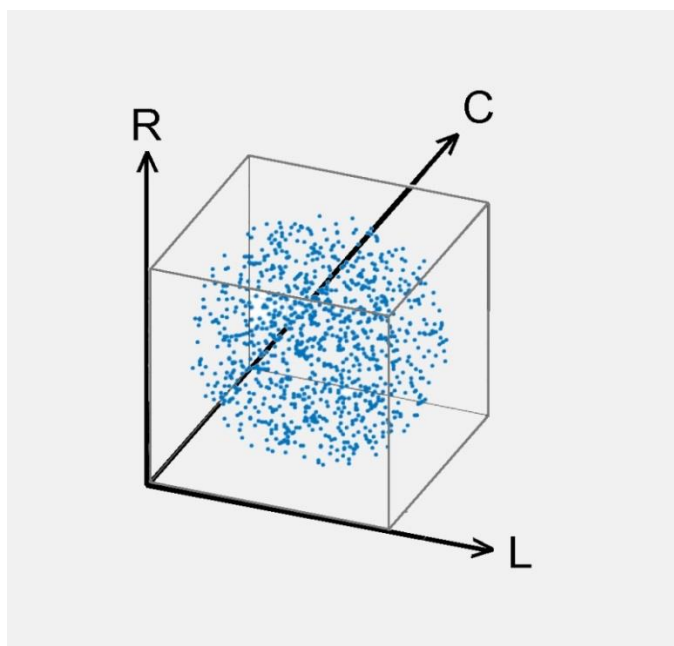
W celu dokonania pomiarów zrealizowane zostało oprogramowanie rozwiązujące przedstawiony problem optymalizacji dynamicznej. W oprogramowaniu zaimplementowano Metodę Gradientu Prostego w Przestrzeni Funkcyjnej Sterowań, która wprowadza konieczność wielokrotnego całkowania równań różniczkowych zwyczajnych (ang. ODE – Ordinary Differential Equations). W celu skrócenia czasu rozwiązywania równań ODE względem klasycznych metod sekwencyjnych, zaprojektowano oraz zaimplementowano w oprogramowaniu testowym algorytm dedykowany dla platformy Nvidia CUDA, który oparty został na Spekulacyjnej koncepcji całkowania tego typu równań. W celu oceny wydajności algorytmu równoległego rozpatrywany problem optymalizacji dynamicznej rozwiązywany był zarówno przy wykorzystaniu metod sekwencyjnych, jak również przy pomocy opracowanego algorytmu równoległego na platformie Nvidia CUDA. W oprogramowaniu stanowiącym środowisko badawcze zaimplementowano również szczegółowy pomiar czasu pracy porównywanych algorytmów. Ze względu na ograniczenia wynikające z fizycznej budowy układu GPU, optymalne obciążenie multiprocessorów strumieniowych CUDA w przypadku zastosowania koncepcji Spekulacyjnej gdy rozwiązywane jest tylko jedno równanie nie jest zadaniem trywialnym i wymaga dobrej znajomości architektury danego GPU.

8.2 Diagramy bifurkacyjne dla układów łuku elektrycznego

Diagramy bifurkacyjne rozpatrywane w niniejszej pracy rozumiane są jako zmiany w reakcjach oscylacyjnych zmiennych parametrów danego modelu łuku elektrycznego.

Najczęściej prezentowane w literaturze diagramy bifurkacyjne przyjmują postać jedno lub zdecydowanie rzadziej dwu-parametryczną [19] [20] [21]. Zauważyć należy, iż w niniejszej pracy obliczane są diagramy bifurkacyjne dla dwóch a nawet trzech zmieniających się jednocześnie parametrów.

Zatem jako trójwymiarowy diagram bifurkacyjny rozumiana jest chmura różnokolorowych punktów w przestrzeni parametrów RLC (Rys. 8.1). Analogicznie w przypadku diagramów dwuwymiarowych punkty odwzorowane są na płaszczyźnie dla wybranej pary parametrów: RL, RC lub LC.



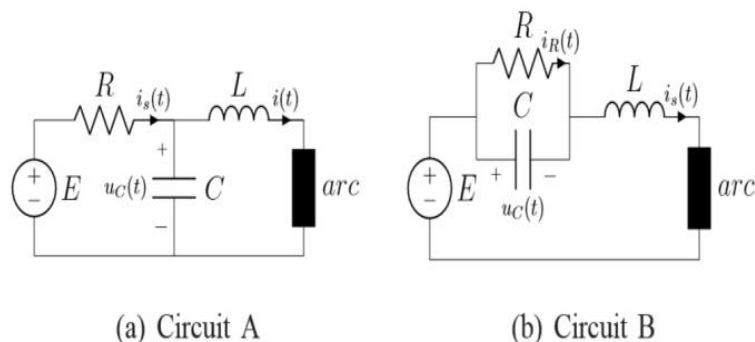
Rys 8.1. Chmura punktów w przestrzeni RLC

Aby uzyskać diagramy o wysokiej szczegółowości/rozdzielczości przyjęte zostały odpowiednie wymiary zadań obliczeniowych. Dla diagramów dwuparametrycznych przyjęto rozdzielczość 1000 x 1000 punktów. Dla diagramów przestrzennych gdzie zmieniały się jednocześnie wszystkie trzy parametry, ustalono poprzez analogię rozdzielczość 1000 x 1000 x 1000 punktów.

Zauważyć należy, iż rozwiązanie takiego problemu wymaga wielokrotnego rozwiązywania układu równań różniczkowych (ODE). W przypadku diagramów dwuwymiarowych jest to liczba rozwiązań (rzędu 10^6) a w przypadku diagramów trójwymiarowych (rzędu 10^9). Ponadto w każdej iteracji algorytmu oprócz rozwiązywania układu równań wymagane jest znalezienie maksimum lokalnych oraz identyfikacja okresu na ich podstawie.

W literaturze [22] zostały przedstawione typowe modele łuków elektrycznych. Prezentowane modele oparte są na różnych charakterystykach napięcia łuku (V-I) i służą do oszacowania energii w przypadku zwarcia łukowego. Tego rodzaju charakterystyki napięcia mogą być wykorzystywane w celu obliczania diagramów bifurkacyjnych. Literatura [23] i [24] zawiera przykłady obliczania diagramów bifurkacji, dla których zmiany wartości zachodzą powoli w odniesieniu do jednego parametru. Jak pokazano w literaturze, każda zmiana wartości takiego parametru może prowadzić do zmiany charakteru odpowiedzi układu ODE.

Typowe obwody łuku elektrycznego przedstawiono na rysunku 8.2. Diagramy bifurkacyjne uzyskane w ramach przygotowania niniejszej rozprawy, obliczane były dla takich właśnie obwodów, w których uwzględniono jednoczesne zmiany wartości parametrów.



Rys. 8.2. Dwa typowe obwody łuku elektrycznego - SCHEMAT ZASTĘPCZY

Poniżej przedstawiono układ równań (8.7) dla typowych obwodów łuku elektrycznego pokazanych na rysunku 8.2. Układ równań prezentowany po lewej stronie (8.7) opisuje obwód na Rys. 8.2(a). Po prawej stronie (8.7) znajduje się bezwymiarowy zapis tego układu równań, gdzie R, L, C są parametrami [23] [25] [26]. Bezwymiarowe zmienne wprowadzono w celu uogólnienia analizy problemu obliczeniowego.

$$\begin{aligned}
 \frac{di}{d\tau} &= \frac{1}{L} \left(u - \frac{U(i_\theta)}{i_\theta} i \right) & \frac{dx}{dt} &= \frac{1}{L} (y - xz^m) \\
 \frac{du}{d\tau} &= \frac{1}{RC} (E - u - Ri) & \rightarrow & \frac{dy}{dt} = \frac{1}{RC} (R + 1 - y - Rx) \\
 \frac{di_\theta^2}{d\tau} &= \frac{1}{\theta} (i^2 - i_\theta^2) & \frac{dz}{dz} &= x^2 - z
 \end{aligned}
 \tag{8.7}$$

gdzie:

$$x = \frac{i}{i_0}, y = \frac{u_c}{U_0}, z = i_\theta^2 / I_0^2,$$

i_θ - prąd łuku,

U_0, I_0 - stałe ze statycznej woltamperowej charakterystyki ($U(i) = U_0 \left(\frac{i}{i_0}\right)^n$) - gdzie $n < 0$,

$\theta = \tau/t$ - stała czasowa,

R, L, C - oznaczają rezystancję, indukcyjność oraz pojemność,

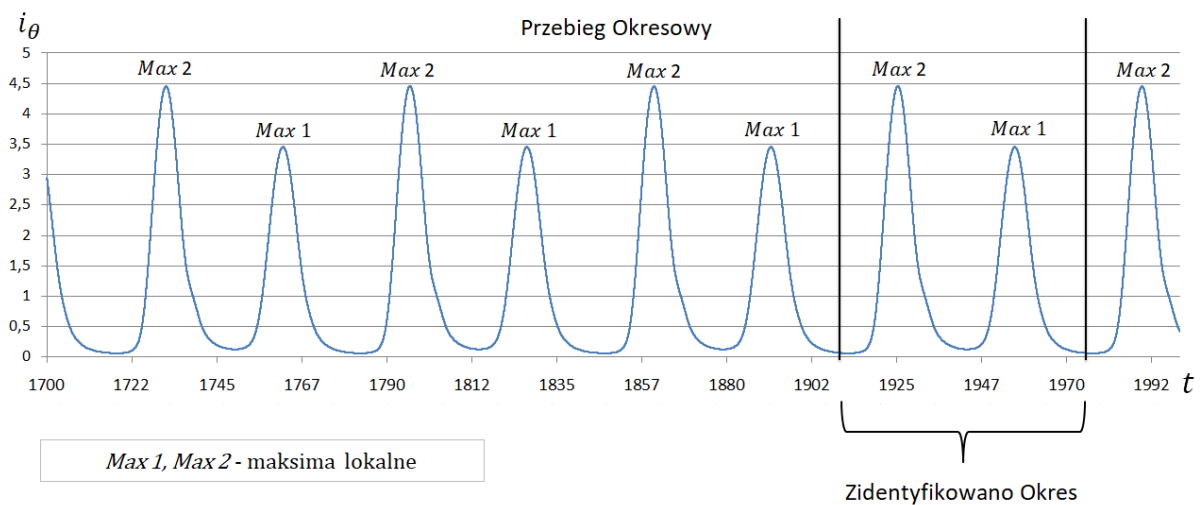
m - wynika z $-1 < m = \frac{n-1}{2} < 0$, zazwyczaj $m = -2/3$, jak wynika z $U(i_\theta) = i_\theta^n$, przy $n = -1/3$.

Możemy również rozważyć układ równań po prawej stronie (8.7), jako układ po lewej stronie gdy $U_0 = I_0 = \theta = 1$ oraz $E = RI_0 + U_0$. Jak pokazano w literaturze [23], uwzględniając odpowiednią zamianę zmiennych układ równań (8.7) opisuje również obwód prezentowany na Rys. 8.2 (b).

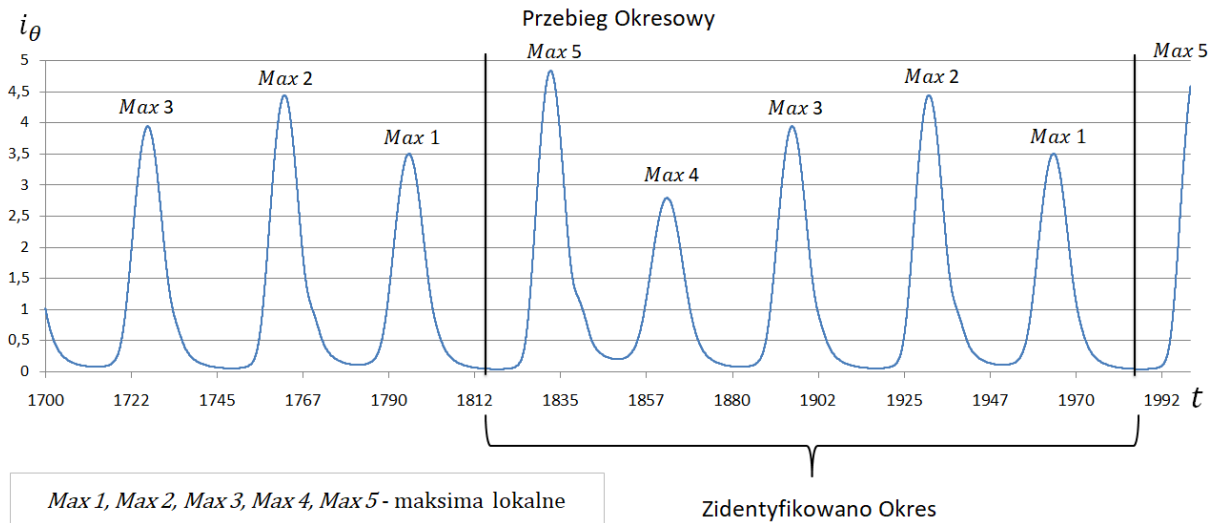
W praktycznych zastosowaniach bardzo częstym problemem okazuje się fakt, iż algorytmy w oparciu, o które obliczane są diagramy bifurkacyjne, a szczególnie diagramy o wysokiej rozdzielczości, bazują na wykorzystaniu dobrze znanych, klasycznych metod numerycznych. Z oczywistych powodów metody takie nie są dostosowane do nowoczesnego sprzętu obliczeniowego. Tego typu algorytmy stanowią zatem duże wyzwanie dla dzisiejszych technologii, nawet biorąc pod uwagę zastosowanie najnowszych procesorów wielordzeniowych typu CPU, których potencjał nie może zostać w pełni wykorzystany. Jak zostało wspomniane wcześniej taki algorytm dla dużych przypadków jest bardzo kosztowny obliczeniowo, a czas jego realizacji jest bardzo często nieakceptowalny. Rozwiązaniem tego problemu może być zaprojektowanie algorytmów dopasowanych do masowo równoległej architektury nowoczesnego sprzętu obliczeniowego o architekturze masowo równoległej.

Jak zostało wspomniane aby uzyskać diagram bifurkacji konieczne jest nie tylko rozwiązywanie układu ODE dla każdego z punktów diagramu ale wymagane jest znalezienie maksimów lokalnych oraz identyfikacja okresu na ich podstawie.

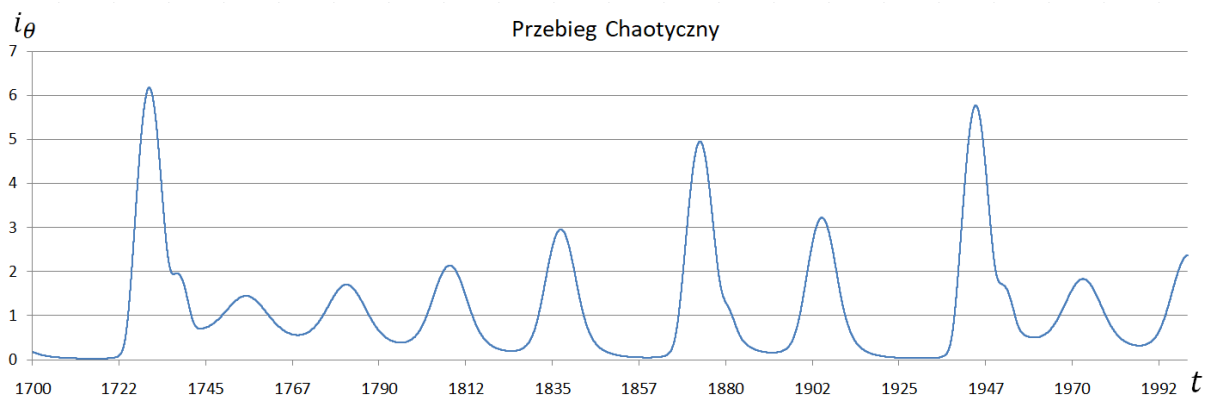
W celu zilustrowania problemu poszukiwania maksimów lokalnych oraz identyfikacji okresu, na rysunkach 8.3, 8.4 oraz 8.5 przedstawiono przykładowe przebiegi rozwiązań dla układu (8.7) Dla prezentowanych przebiegów przyjęto oznaczenia: i_θ – prąd łuku, t – czas analizy rozwiązania.



Rys. 8.3. Okresowy przebieg rozwiązania układu (8.7) dla parametrów $R=8$; $L=1$; $C=3,14$;



Rys. 8.4. Okresowy przebieg rozwiązania układu (8.7) dla parametrów $R=8$; $L= 1,16$; $C=3,14$;

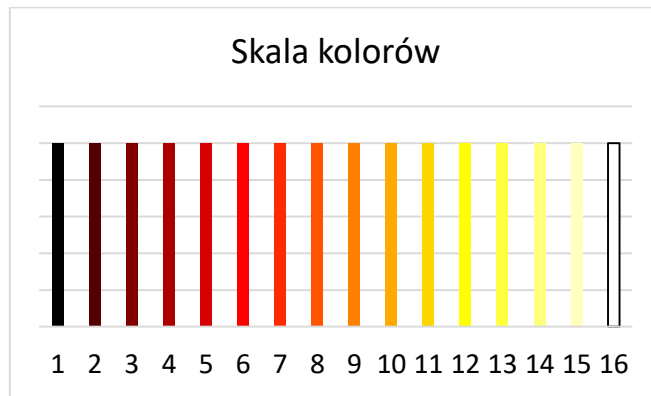


Rys. 8.5. Chaotyczny¹ przebieg rozwiązania układu (8.7) dla parametrów $R=8$; $L= 1$; $C=3,38$;

W ramach opracowanych algorytmów identyfikacja okresu odbywała się w zakresie maksimum od 1 do 16. Przyjęto, że jeżeli liczba maksimum będzie większa niż 16 to okres zostanie zidentyfikowany jako 16. Na wykresach każdy okres otrzymał inny kolor. Im „wyższy” okres tym jaśniejszy kolor został mu przypisany. Dla 16 był to kolor biały natomiast dla 1 przypisano kolor czarny.

Przykładową skalę kolorów prezentuje rysunek 8.6. Na podstawie obserwacji uzyskiwanych wyników wyznaczono wartość tolerancji błędu porównywania maksimum podczas identyfikacji okresu i wynosiła ona $tol=0,0005$.

¹ Chaos Deterministyczny – termin ten odnosi się do nieregularnych zachowań (zwanymi Bifurkacjami) układów deterministycznych w konsekwencji niewielkich zmian wartości parametrów takiego układu. Zachowanie się układu jest całkowicie nieprzewidywalne choć w pełni reprodukowalne dla konkretnych wartości parametrów ze względu na determinizm opisany równaniami [33] [34] [35] [36].

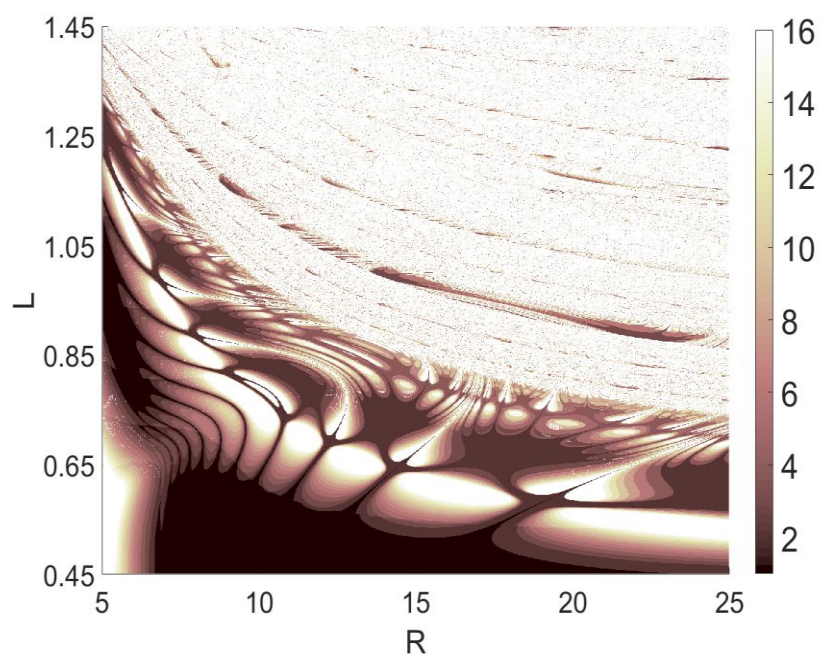


Rys. 8.6. Przykładowa skala kolorów wykorzystywana podczas generowania diagramów

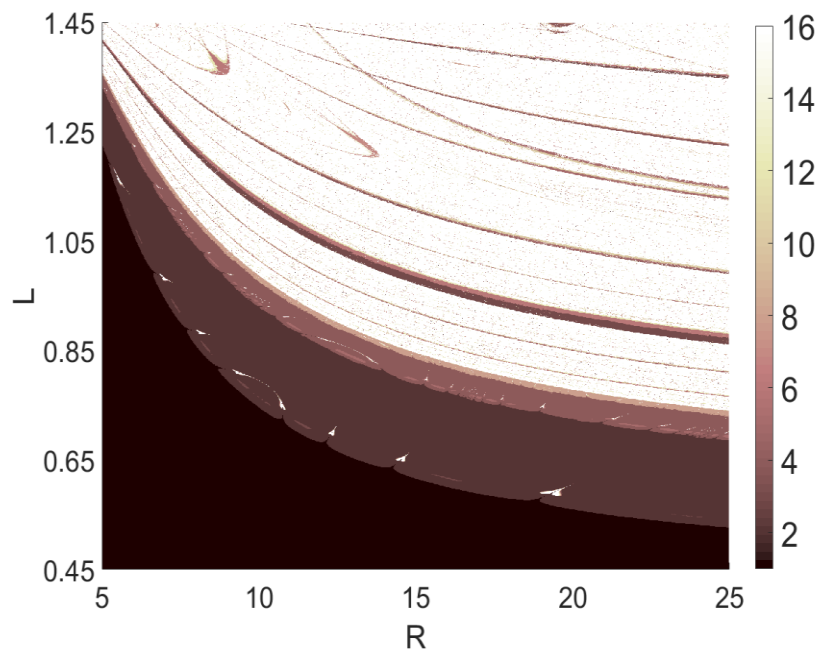
Do rozwiązywania układu równań (8.7) w opracowanych algorytmach dla diagramów 2D i 3D wykorzystano metodę Rungego-Kutty IV rzędu.

Poniżej przedstawione zostały wybrane diagramy bifurkacji uzyskane w ramach realizacji badań nad algorytmami masowo równoległymi dedykowanymi dla technologii CUDA.

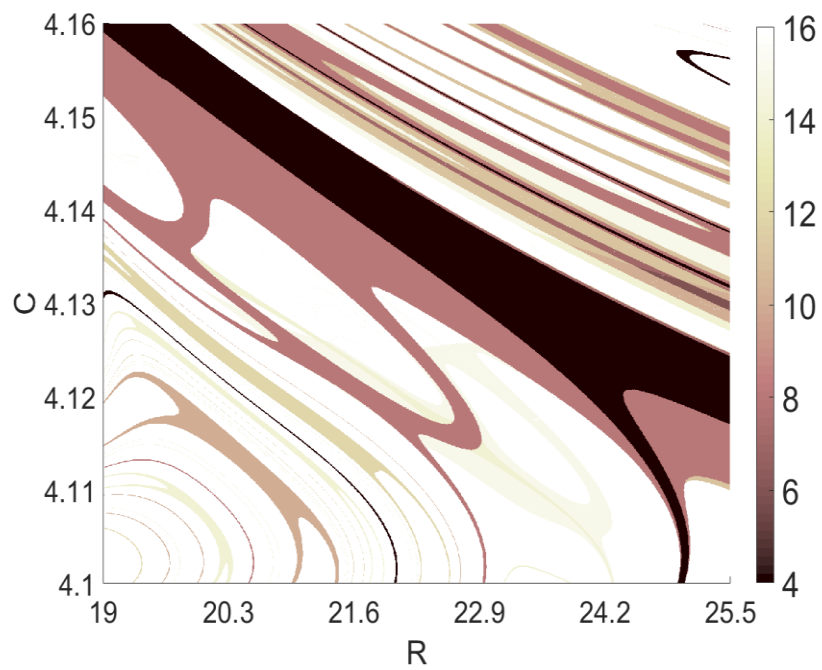
Zarówno diagramy 2D jak również 3D wykonane zostały przy wykorzystaniu środowiska do obliczeń naukowych Matlab [27] [28]. Przez N oznaczono liczbę kroków całkowania. Jako docelowe należy traktować diagramy dla których liczba kroków wynosiła 400 000, ponieważ dopiero wówczas rozwiązanie było ustabilizowane.



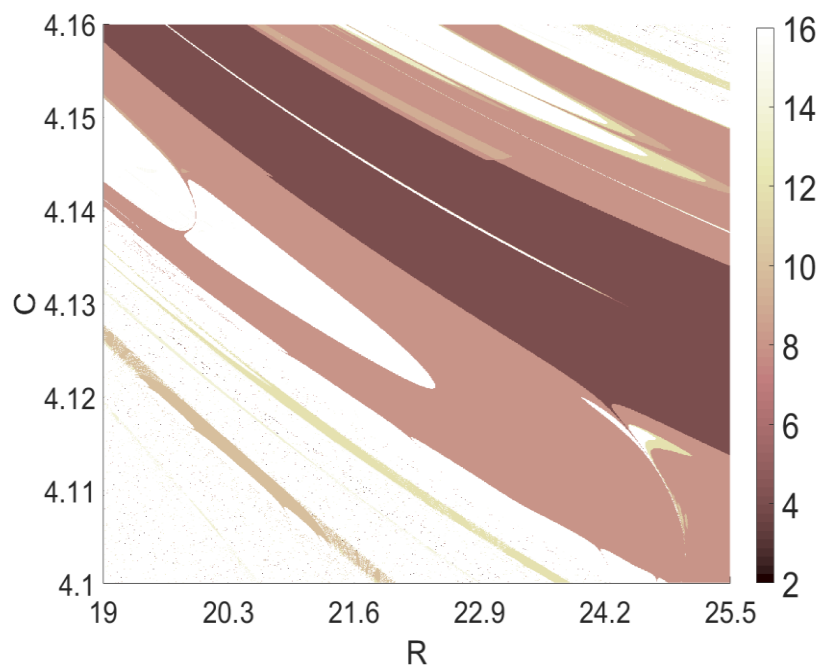
Rys. 8.7. Diagram dla zmiennych RL oraz stałej $C=3.14$ ($N=100,000$)



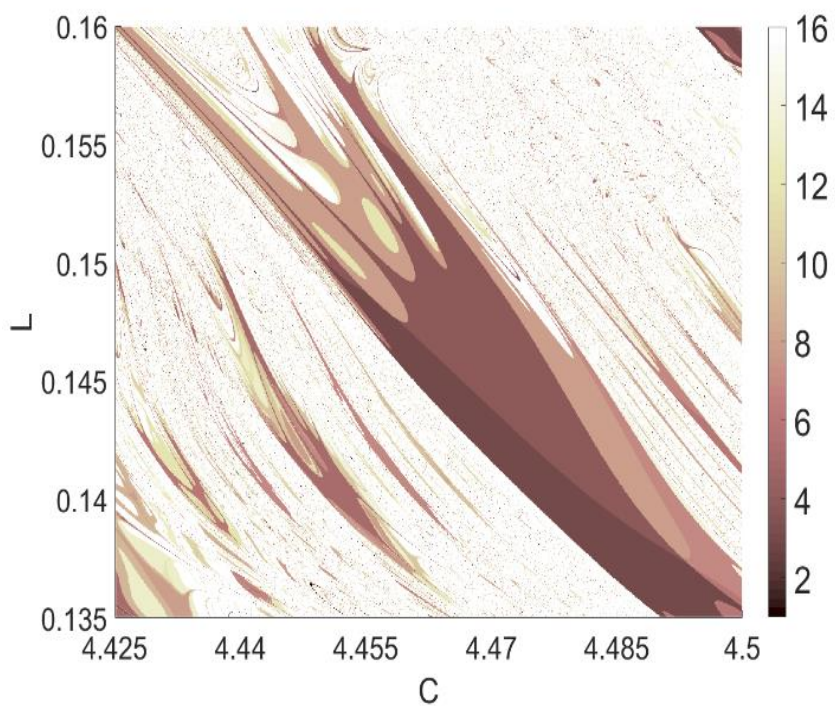
Rys. 8.8. Diagram dla zmiennych RL oraz stałej $C=3.14$ ($N=400,000$)



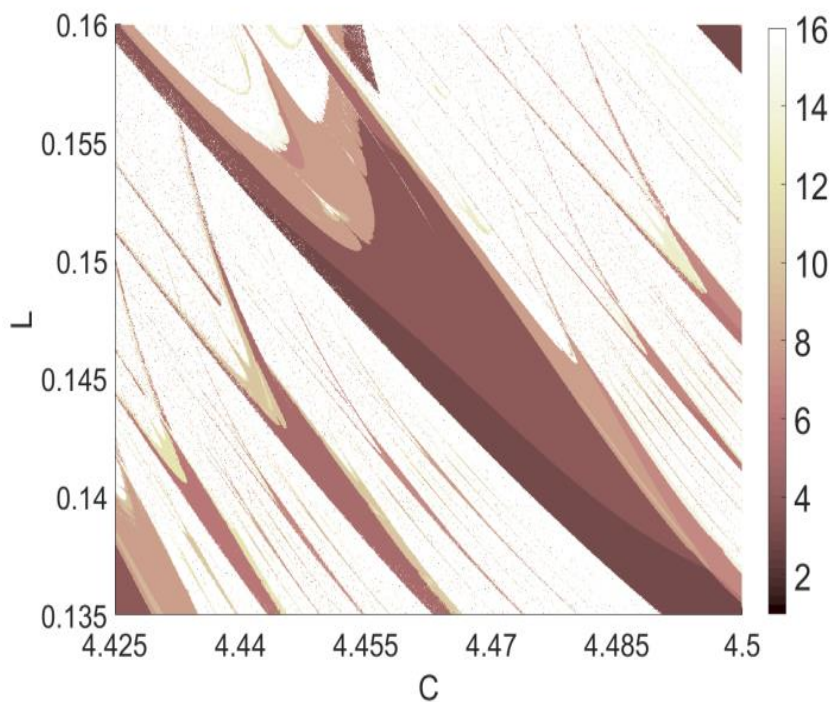
Rys. 8.9. Diagram dla zmiennych RC oraz stałej $L=0.2$ ($N=100,000$)



Rys. 8.10. Diagram dla zmiennych RC oraz stałej $L = 0.2$ ($N = 400,000$)

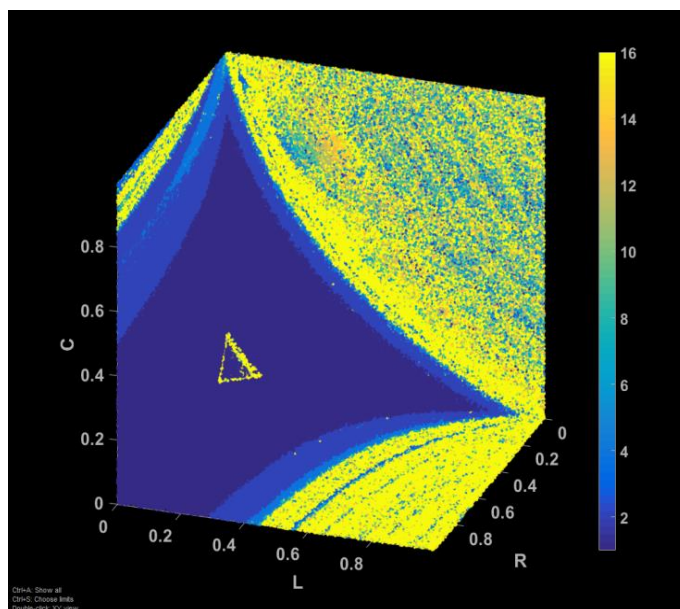


Rys. 8.11. Diagram dla zmiennych LC oraz stałej $R = 15.0$ ($N = 100,000$)

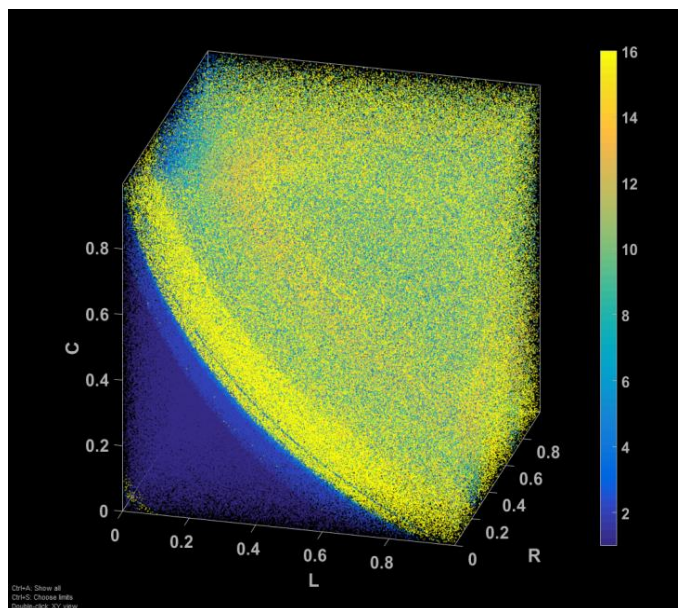


Rys. 8.12. Diagram dla zmiennych LC oraz stałej $R= 15.0$ ($N=400,000$)

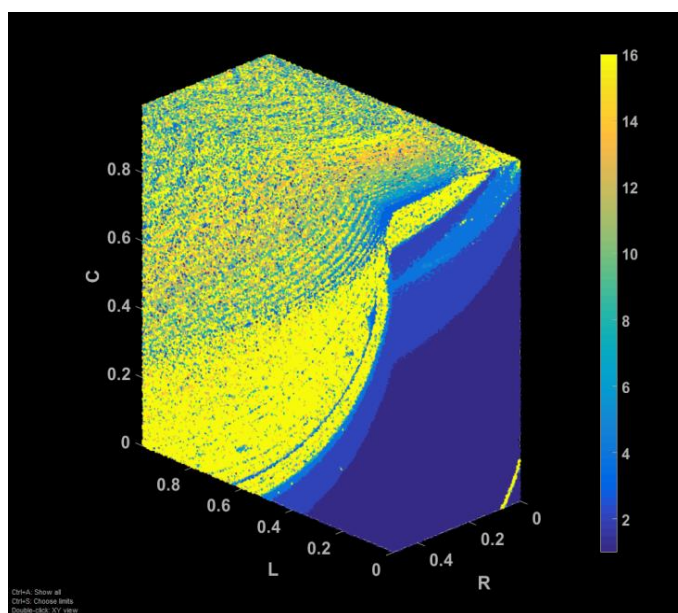
Trójwymiarowa wizualizacja chmury punktów w przestrzeni zrealizowana została za pomocą specjalnej biblioteki wspomagającej tego typu zadania. Wspomniana biblioteka nosi nazwę Point Cloud [29] [30] i dostępna jest jako rozszerzenie środowiska Matlab.



Rys. 8.13. Diagram przedstawia chmurę punktów w pełnym zakresie parametrów $R[5.0, 25.0]$, $L[0.16, 1.45]$, $C[3.14, 4.5]$. Pozostałe parametry: point size = 10, point limit = 5 000 000.



Rys. 8.14. Diagram przedstawia chmurę punktów w pełnym zakresie parametrów $R[5.0, 25.0]$, $L[0.16, 1.45]$, $C[3.14, 4.5]$. Pozostałe parametry: point size = 1, point limit = 1 000 000.



Rys. 8.15. Diagram przedstawia przekrój izometryczny chmury punktów w zakresie parametrów $R[2.5, 25.0]$, $L[0.16, 1.45]$, $C[3.14, 4.5]$. Pozostałe parametry: point size = 8, point limit = 1 000 000.

9. Uwagi końcowe i wnioski

Niewątpliwie teoretyczny potencjał urządzeń CUDA, ale i całej technologii na którą składa się również oprogramowanie mające na celu maksymalne ułatwienie programiście wykorzystania tak skomplikowanego sprzętu jak karta graficzna, daje powody by sądzić, iż wiele problemów obliczeniowych nie dających się dotychczas rozwiązać w rozsądnym czasie może nie stanowić już wyzwania. Dowodem tego są prezentowane w ramach niniejszej rozprawy wyniki badań. Wyzwaniem okazuje się jednak bardzo często samo projektowanie algorytmów, które będą w stanie efektywnie wykorzystać pełnię potencjału urządzeń CUDA. Przeprowadzone w ramach niniejszej pracy badania pokazały, iż efektywne wykorzystanie technologii CUDA jest możliwe tylko dla bardzo specyficznych problemów i algorytmów obliczeniowych, które charakteryzują się wielokrotnym wykorzystaniem tych samych danych, nie wymagają dużych transferów danych pomiędzy hostem a urządzeniem obliczeniowym, oraz nie wymagają zbyt wielu operacji synchronizacyjnych pomiędzy wątkami i dzięki temu dobrze skalują się na wiele multiprocessorów oraz wiele niezależnych od siebie fizycznych urządzeń CUDA. Aby zatem dążyć do optymalnego wykorzystania potencjału technologii CUDA należy poszukiwać problemów spełniających wszystkie przedstawione powyżej kryteria. Niestety jednak sytuacja w rzeczywistości wygląda zupełnie odwrotnie gdyż mając postawiony konkretny problem będziemy poszukiwać sposobu – algorytmu/technologii, która pozwoli go rozwiązać w jak najkrótszym czasie lub/i w znacznie większym wymiarze niż było to możliwe przed zrównolegleniem co często prowadzi do sięgania po bardziej złożone problemy. Skrócenie czasu realizacji dobrze znanych problemów obliczeniowych nie jest więc głównym celem zastosowania technologii takich jak CUDA. Głównym celem jest rozwiązywanie problemów, które wcześniej nie były możliwe do rozwiązania ze względu na brak dostatecznie wydajnej i odpowiednio elastycznej technologii.

Mając na uwadze przedstawione w ramach niniejszej rozprawy wyniki badań, w ramach których zrealizowano szereg wnikliwych analiz porównawczych opracowanych algorytmów, można stwierdzić, iż wszystkie sformułowane na początku pracy **cele zostały osiągnięte**.

W zdecydowanej większości rozpatrywanych w ramach niniejszej rozprawy przypadków osiągnięto znaczącą poprawę wydajności algorytmów masowo równoległych nie tylko w stosunku do algorytmów sekwencyjnych, ale co ważniejsze, również w stosunku do algorytmów równoległych realizowanych z udziałem technologii OpenMP na wielordzeniowych układach CPU. Należałoby zatem stwierdzić, iż sformułowana na początku pracy teza o następującym brzemieniu: „Platforma Nvidia CUDA stwarza realne możliwości wyraźnej poprawy efektywności realizacji analizowanych w pracy metod i algorytmów optymalizacji dynamicznej poprzez znaczne skrócenie czasu całkowania układów równań różniczkowych zwyczajnych opisujących rozważane problemy dynamiczne” – **została udowodniona**. Należy jednak pamiętać, że technologia CUDA nie gwarantuje przyspieszenia w każdym wypadku, a jedynie w sytuacji gdy zostaną spełnione dodatkowe wymagania stawiane algorytmom opisane w niniejszej rozprawie.

9.1 Wkład własny autora pracy:

- Opracowanie algorytmu oraz implementacji masowo równoległego całkowania układów ODE, dedykowanego dla platformy Nvidia CUDA w oparciu o koncepcję równoległego całkowania równań składowych układu.
- Opracowanie algorytmu oraz implementacji masowo równoległego całkowania zarówno pojedynczych równań jak również układów ODE, dedykowanego dla platformy Nvidia CUDA w oparciu o koncepcję równoległego podziału czasu całkowania, która znana jest również pod nazwą koncepcji Spekulacyjnej.
- Opracowanie algorytmu oraz implementacji masowo równoległego całkowania układów ODE, dedykowanego dla platformy Nvidia CUDA w oparciu o kombinację zarówno koncepcji Spekulacyjnej jak również koncepcji równoległego całkowania równań składowych układu. W kontekście metody Spekulacyjnej oraz zastosowanej technologii obliczeniowej powstała koncepcja Kombinowana, która jest **oryginalnym** osiągnięciem autora niniejszej rozprawy.
- Opracowanie algorytmu oraz implementacji masowo równoległej realizacji metody „Gradientu w przestrzeni funkcyjnej sterowań”, dedykowanej dla platformy Nvidia CUDA w oparciu o założenia koncepcji Spekulacyjnej.
- Opracowanie algorytmu oraz implementacji masowo równoległego obliczania diagramów bifurkacyjnych 2D jak również 3D dla typowych obwodów łuku elektrycznego. Oryginalny wkład autora pracy dotyczy algorytmów dedykowanych dla platformy Nvidia CUDA.
- Badania efektywności opracowanych algorytmów masowo równoległych oraz analiza porównawcza względem klasycznych algorytmów sekwencyjnych jak również innych algorytmów równoległych. Przeprowadzone badania pozwoliły na ocenę potencjału technologii Nvidia CUDA w kontekście rozwiązywanych w pracy problemów jak również całej klasy podobnych problemów bazujących na konieczności wielokrotnego całkowania układów ODE i/lub całkowania problemów od dużym wymiarze.

9.2 Kierunki dalszych badań

W ramach przygotowania niniejszej rozprawy rozważanych było wiele problemów badawczych zanim zakres pracy został ostatecznie sformułowany. W bieżącym punkcie wskazane zostały kierunki dalszych badań jako otwarte problemy wykraczające poza zakres niniejszej rozprawy.

Pozostając w kontekście problemów obliczeniowych rozpatrywanych w ramach niniejszej rozprawy, jak również w kontekście zastosowania technologii Nvidia CUDA, wskazać można, następujące otwarte problemy badawcze:

Przebadanie opracowanych algorytmów dedykowanych dla CUDA bazujących na koncepcji Spekulacyjnej w kontekście układów z rozwiązaniem o charakterze oscylacyjnym wraz z próbą implementacji algorytmu masowo równoległego ze zmiennym krokiem całkowania.

Przebadanie opracowanych algorytmów, a w szczególności algorytmów bazujących na równoległym podziale układu na poszczególne równania w kontekście problemów opisanych macierzami rzadkimi, jak również implementacje rozwiązań bazujących na założeniach koncepcji znanej jako Dekompozycja Epsilon, która to została dość obszernie przedstawiona w pracach Dragoslava Šiljaka [31] [32].

10. Bibliografia – wybrane pozycje

1. FINDEISEN W., SZYMANOWSKI J., WIERZBICKI A.: *Metody obliczeniowe optymalizacji*, Wydaw. Politechniki Warszawskiej 1973.
2. AMBORSKI K.: *Podstawy metod optymalizacji*, Oficyna wydawnicza Politechniki Warszawskiej 2009.
3. KRYSZEWSKI W.: *Podstawy Teorii Sterowania Optymalnego*. [<http://www-users.mat.umk.pl/~wkrysz/attachments/sterowanie.pdf>, Dostęp: 08.12.2018].
4. KACZOREK T.: *Teoria sterowania*, PWN Warszawa 1981.
5. KACZOREK T.: *Podstawy teorii sterowania*, WNT 2005.
6. PONTRYAGIN L. S., BOLTYANSKII V. G., GAMKRELIDZE R. V., MISHCHENKO E. F.: *The Mathematical Theory of Optimal Processes*. Wiley, New York, 1965.
7. FORENC J.: *Analiza spekulacyjna stanów nieustalonych w układach elektrycznych*, Rozprawa doktorska, Politechnika Białostocka 2006.
8. STALLINGS W.: *Organizacja i architektura systemu komputerowego*. Wydawnictwa Naukowo-Techniczne, Warszawa 2000.
9. DUNCAN R.: "A Survey of Parallel Computer Architectures", February 1990
<http://www.eng.ucy.ac.cy/theocharides/Courses/ECE656/Duncan90.pdf> [dostęp: 10.01.2021].
10. NVIDIA CORPORATION: *Nvidia CUDA C Programming Guide*, Nvidia Corporation 2018.
11. NVIDIA CORPORATION: *Cuda C Best Practices Guide*, Nvidia Corporation 2015.
12. KIRK D. B., HWU W. WEN-MEI: *Programming Massively Parallel Processors: A Hands-on Approach*, Morgan Kaufmann, 2010.
13. NVIDIA CORPORATION: *CUDA C++ Programming Guide*, Nvidia Corporation 2021.
14. STOER J., BULIRSCH R.: *Wstęp do metod numerycznych. Tom II*. PWN, Warszawa, 1980.
15. HAPRA S.C., CANALE R.P.: *Numerical Methods for Engineers*, The McGraw-Hill Companies 2011 .
16. MAJCHRZAK E., MOCHNACKI B.: *Metody numeryczne. Podstawy teoretyczne, aspekty praktyczne i algorytmy*, Wydawnictwo Politechniki Śląskiej, Gliwice, 2004.
17. CONTE S.D., DE BOOR C.: *Elementary Numerical Analysis. An Algorithmic Approach*. McGraw-Hill, 1980.

18. FORTUNA Z., MACUKOW B., WĄSOWSKI J.: *Metody numeryczne*, WNT, Warszawa, 2005.
19. ITOH Y., UENOHARA S., ADACHI M., MORIE T., AIHARA K.: *Reconstructing bifurcation diagrams only from time-series data generated by electronic circuits in discrete-time dynamical systems*, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, 20.
20. PALA A., MACHACZEK M.: *Parallel computing of two-parameter bifurcation diagrams of an electric arc model with chaotic dynamics using Nvidia CUDA and OpenMP technologies*, *Przegląd Elektrotechniczny* nr 03/2019.
21. AWREJCEWICZ J., PYRYEV J., KUDRA G., OLEJNIK P.: *Matematyczne i numeryczne metody analizy bifurkacji i dynamiki chaotycznej układów mechanicznych z tarciem i uderzeniami*, Politechnika Łódzka 2006.
22. AMMERMAN R. F., GAMMON T., SEN P. K. and NELSON J. P., "DC-Arc models and incident energy calculations," *IEEE Trans. Industry Appls.*, vol. 46, no. 5, pp. 1810–1819, Sept./Oct. 2010. .
23. SYDORETS V. N. and PENTEGOV I. V., *Deterministic Chaos in Nonlinear Circuits with Arcs*, Kiev, Ukraine: Svarka, 2013 (in Russian). .
24. PENTEGOVI I. V. and SYDORETS V. N., "Comparative analysis of models of dynamic welding arc," *The Paton Welding Journal*, vol. 12, pp. 45–48, 2015, doi: 10.15407/tpwj2015.12.09.
25. SYDORETS V.: „The Bifurcations and Chaotic Oscillations in Electric Circuits with Arc” In: MITKOWSKI W., KACPRZYK J.: *Studies in Computational Intelligence, Modelling Dynamics in Processes and Systems, Volume 180*, Springer 2009.
26. ROSOŁOWSKI E.: *Podstawy Modelowania Systemów*, Wrocław, marzec 2020, <http://www.rose.pwr.wroc.pl/PMS/PMS.pdf> [dostęp: 06.11.2021].
27. HUNT B. R., LIPSMAN R. L., ROSENBERG, J. M.: *Guide to MATLAB (R)*, Cambridge University Press, 2014.
28. RUDRA P.: *Matlab dla naukowców i inżynierów*, Wydawnictwo PWN 2020.
29. MathWorks Help Center, <https://www.mathworks.com/help/vision/ref/pointcloud.html#d122e92017> [dostęp: 04.02.2020].
30. Kurs Point Cloud, <https://www.geo.tuwien.ac.at/downloads/pg/pctools/pctools.html> [dostęp: 10.02.2020].
31. SEZER M. i ŠILJAK D. *Nested ε decompositions and clustering of complex systems*, *Automatica*, 22, pp. 321–331, 1986.

32. ZEČEVIĆ A. i ŠILJAK D. *Control of Complex Systems. Structural Constraints and Uncertainty*, London, Springer, 2010.
33. *Encyklopedia PWN – Bifurkacja*,
<https://encyklopedia.pwn.pl/haslo/bifurkacja;3877582.html> [dostęp: 28.10.2021].
34. SCHUSTER H.G.: *Chaos deterministyczny*, PWN Warszawa 1995.
35. TEMPCZYK M.: *Teoria chaosu dla odważnych*, PWN Warszawa Wydanie: 2021.
36. STEWART I.: *Czy Bóg gra w kości? Nowa matematyka chaosu*, PWN Warszawa 2001.