**Arkadiusz GARDECKI**

Politechnika Opolska, Instytut Układów Elektromechanicznych i Elektroniki Przemysłowej

# Obliczeniowa opłacalność zrównoleglenia drobnoziarnistego algorytmu numerycznego całkowania typu PECE

***Streszczenie.*** *W artykule omówiono problemy związane z przystosowaniem metody rozwiązywania układów równań różniczkowych zwyczajnych typu predyktor-korektor (PECE) do obliczeń w układach równoległych. Zastosowanie tego typu algorytmów może być obliczeniowo opłacalne, szczególnie gdy obliczanie funkcji prawej strony równania różniczkowego jest kosztowne. Jednakże obliczenia równoległe z wieloma punktami synchronizacji mogą powodować wydłużenie czasu obliczeń w porównaniu do obliczeń sekwencyjnych.*

***Abstract****. This paper presents a performance analysis predictor-corrector (PECE) numerical integration method in the parallel computation calculations. The use of parallel algorithms for performing calculations in the analysis of initial value problems can be computationally viable, especially if the right hand side of the calculation function of the differential equation is time expensive. However, the calculations in parallel with a number of synchronization points may take a long computation time in comparison to the sequential calculation. (**Computational cost-effectiveness of parallelization granularity numerical integration PECE algorithm**).*

**Słowa kluczowe**: numeryczne całkowanie, obliczenia równoległe.
**Keywords**: numerical integration, parallel computations.

doi:10.12915/pe.2014.11.20

## Wstęp

W artykule przedstawione zostaną wyniki porównania wydajności równoległych obliczeń procedury numerycznego rozwiązywania układów równań różniczkowych zwyczajnych typu predyktor-korektor (PECE) w zależności od złożoności obliczeniowej równań oraz typu powiązań pomiędzy równaniami. Procedura BGKODE_DSP [1] realizuje szczególny przypadek algorytmu Krogh'a [2]. Została ona specjalnie opracowana do zastosowań w układach czasu rzeczywistego i realizuje obliczenia ze stałą długością kroku. Jej zrównoleglona wersja jest obiektem prezentowanych badań. Proces poszukiwania wzrostu wydajności obliczeń tego typu algorytmu w przypadku obliczeń równoległych nie jest łatwy z powodu charakteru zarówno metody predyktor-korektor jak i samej realizacji obliczeń równoległych. Algorytm predyktor-korektor typu PECE wymaga dwukrotnego określenia prawej strony układu równań różniczkowych (DIF) w każdym kroku. W przypadku braku powiązań pomiędzy równaniami możliwe jest zrównoleglenie obliczeń bez punktów synchronizacji w pojedynczym kroku. Jeśli równania są ze sobą powiązane, co jest najczęstszym przypadkiem, w pojedynczym kroku całkowania występują dwa punkty synchronizacji co czyni go tzw. algorytmem drobnoziarnistym. Oznacza to, że zrównoleglone zadania są stosunkowo krótkie i koszty tworzenia i komunikacji równoległych wątków mogą zdecydować o opłacalności takiej operacji. Określenie teoretyczne czasów komunikacji i synchronizacji wątków jest w praktyce bardzo trudne, zależą one bowiem od wielu czynników i możliwe jest jedynie ich zgrubne szacowanie. Wydaje się, że jedyną skuteczną metodą ich poznani jest pomiar czasów wykonania poszczególnych zadań na wybranej platformie sprzętowej.

Użycie procedury realizującej algorytm typu predyktor-korektor do obliczeń układów równań różniczkowych jest szczególnie opłacalne jeśli chcemy otrzymać wyniki o dużej dokładności, porównywalnej z metodami Runge-Kutta (R-K) czwartego rzędu przy około dwukrotnie mniejszym koszcie obliczeniowym związanym z wywoływaniem prawej strony układu równań różniczkowych (DIF) [3]. Zrównoleglenie tych obliczeń jest uzasadnione wzrostem efektywności obliczeniowej przede wszystkim jeśli obliczanie funkcji w DIF jest kosztowne. Ma to miejsce jeśli liczba równań układu jest duża lub równania są złożone obliczeniowo.

## Oprogramowanie wspomagające projektowanie obliczeń równoległych

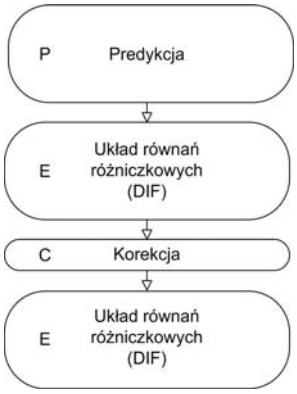
Powstało wiele narzędzi umożliwiających migrację kodu sekwencyjnego na układy realizujące obliczenia równoległe. Mechanizmy te wspomagają między innymi podział zadań pomiędzy procesy i wątki realizowane równolegle, dostęp do pamięci, punkty synchronizacji oraz inne wpływające na efektywność wykonywania operacji równoległych parametry. Do popularnych standardów wspierających obliczenia równoległe zaliczamy POSIX Threads i OpenMP [4]. W prezentowanych badaniach używano OpenMP (Open Multi-Processing) w implementacji dla C/C++ który zawiera zbiór funkcji bibliotecznych, dyrektyw kompilatora i zmiennych środowiskowych umożliwiających zrównoleglenie programów wykonywanych na komputerach o współdzielonej pamięci. Umożliwiają one między innymi w przejrzysty sposób wyodrębnienie z kodu obszarów podlegające zrównolegleniu, określenie sposobu współdzielenia pamięci przez zmienne w wątkach czy określenie punktów synchronizacji wątków. W celu minimalizacji kosztów migrowania wątków między procesorami i kosztów przełączania kontekstu ustawiano koligację procesorów na wątki w celu przyporządkowania ich do wybranych rdzeni procesora.

Wykorzystując mechanizm zrównoleglenia OpenMP porównano efektywność zrównoleglenia w przypadku obliczeń na 8 rdzeniowym procesorze Intel Xeon E5430 w zależności od liczby wykonywanych wątków (używanych rdzeni procesora) oraz kosztów obliczeniowych w funkcji określającej prawą stronę układu równań różniczkowych (DIF).

Testowano wiele możliwych wariantów zrównoleglenia procedury BGKODE_DSP realizującej algorytm typu PECE. Do analizy wydajności zrównoleglonych programów oraz próby ich optymalizacji ze względu na całkowity czas obliczeń wykorzystywano tzw. profilery. Programy te umożliwiają śledzenie podziału zadań na wątki, czas ich wykonania, określenie kosztów tworzenia, zarządzania i kończenia wątków, analizę dostępu do pamięci i wiele innych aspektów. W czasie badań wykorzystano *Intel Vtune Amplifier XE 2013* z *Paraller Studio XE* [5], *AMD CodeAnalyst* [6]. Ich działanie w przypadku analizy zadań drobnoziarnistych nie zawsze daje oczekiwane rezultaty, są one naturalnie przystosowane do bardziej masowych obliczeń.

**Obliczenia równoległe w algorytmie typu PECE**

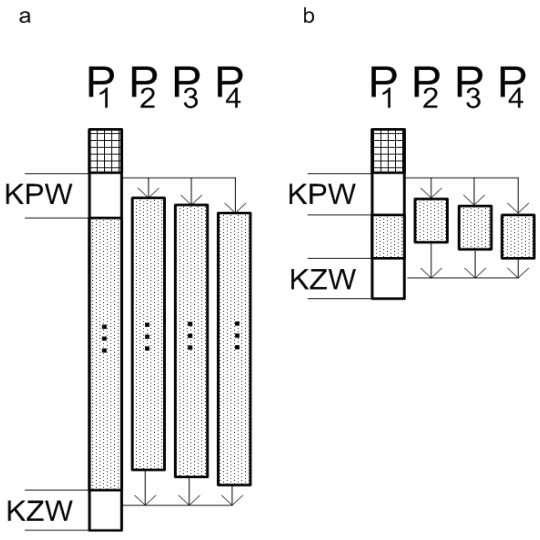
Algorytm numerycznego całkowania typu predyktor-korektor zastosowany w procedurze BGKODE_DSP został szczegółowo opisany w [1]. Na rysunku 1 pokazany jest schemat blokowy najważniejszych z punktu widzenia operacji zrównoleglenia członów tego algorytmu.




Rys.1. Schemat organizacji obliczeń w algorytmie typu PECE z dwoma wywołaniami funkcji DIF (BGKODE_DSP)

Algorytm PECE należy do algorytmów o typie równoległości drobnoziarnistej (wymagających wielu punktów synchronizacji) z powodu obliczeń realizowanych w DIF. W większości przypadków obliczenia te zawierają powiązania pomiędzy równaniami co wymusza użycie tzw. barier w celu zapewnienia poprawności obliczeń (wyjątkiem jest przypadek braku powiązań pomiędzy równaniami w DIF). Częste synchronizacje mało kosztownych zadań mogą powodować, iż pomimo zrównoleglenia obliczeń rzeczywisty czas ich wykonywania wcale nie ulegnie skrócenia, a nawet może się zwiększyć względem czasu obliczeń sekwencyjnych. Powodem tego efektu są koszty tworzenia procesów i wątków, migracja wątków i sposób dostępu do pamięci w trakcie obliczeń.

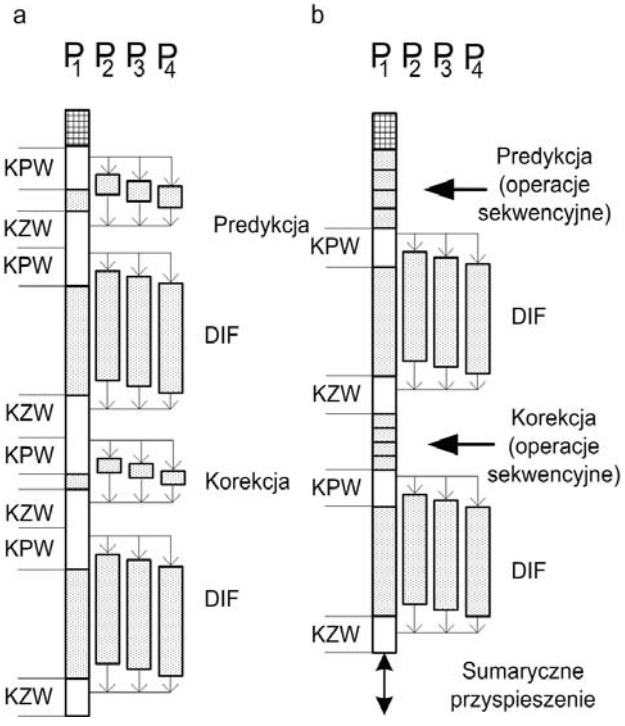
Poniżej przestawiono sposób organizacji obliczeń dla 4 procesorów w przypadku braku powiązań pomiędzy równaniami w układzie równań różniczkowych (DIF).




Rys.2. Symboliczny schemat organizacji obliczeń równoległych algorytmu PECE w przypadku braku powiązań pomiędzy równaniami w DIF: a – znaczący koszt obliczeń w DIF i b – mały koszt obliczeń w DIF. $P_1$ – procesor master, $P_2$,$P_3$, $P_4$ – procesory slave, KPW – koszt powołania wątków, KZW – koszt zamknięcia wątków, obszar zakratkowany – część sekwencyjna kodu.

W tym przypadku możliwe jest zrównoleglenie obliczań ze względu na niezależność działań dla poszczególnych równań układu bez punktów synchronizacji w pojedynczym kroku całkowania. Na rysunku 2a zobrazowano schematycznie podział zadań na cztery procesory. Koszty tworzenia wątków wykonywanych równolegle (slave) zwiększają sumaryczny czas obliczeń wątku głównego (master). W przypadku małego kosztu obliczeń układu równań w DIF (rys.2b) udział tego czasu w stosunku do czasu obliczeń zrównoleglonych wątków jest większy. Może to powodować brak oczekiwanego przyspieszenia zrównoleglonego programu. Istotnym jest tu obliczeniowy koszt powołania wątków i ich zamknięcia zależny w dużej mierze od warstwy sprzętowej, wydajności i architektury procesora.

Na rysunku 3 pokazano dwa przykładowe sposoby organizacji obliczeń w przypadku powiązań pomiędzy równaniami w DIF. W przypadku obliczeń drobnoziarnistych (rys.3a) efekt zrównoleglenia mało kosztownej obliczeniowo części może być przysłonięty czasem tworzenia wątków (KPW) i ich kończenia (KZW). Koszty te spowodowane są między innymi koniecznością przesyłania danych pomiędzy procesami (master-slave) oraz koniecznością synchronizacji procesów. Może okazać się, że rezygnacja ze zrównoleglenia części z pośród wszystkich możliwych bloków będzie korzystniejszym wyborem. Przypadek ten obrazuje rysunek 3b gdzie małe koszty obliczenia części predykcyjnej i korekcyjnej algorytmu względem kosztów tworzenia i zamykania wątków czynią zrównoleglenie tych bloków nieopłacalne. Korzystniej jest zostawić je do realizacji w wątku głównym. Mechanizm ten może powodować w niektórych przypadkach brak korzyści ze zrównoleglenia programu sekwencyjnego.

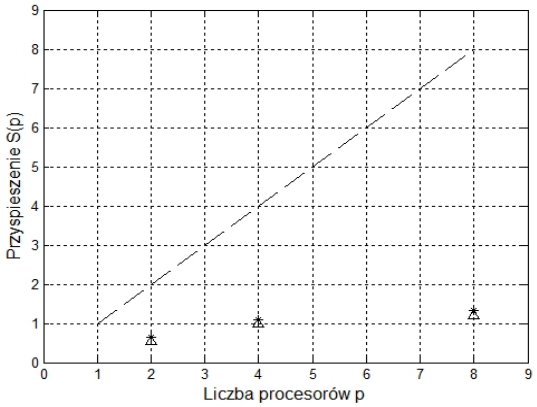



Rys.3. Symboliczny schemat organizacji obliczeń równoległych w BGKODE_DSP w przypadku powiązań pomiędzy równaniami w DIF: a – zrównoleglenie wszystkich sekcji, b – predykcja i korekcja liczone w wątku master w sposób sekwencyjny, zrównoleglenie jedynie sekcji obliczeń DIF. $P_1$ – procesor master, $P_2$,$P_3$, $P_4$ – procesory slave, KPW – koszt powołania wątków, KZW – koszt zamknięcia wątków, obszar zakratkowany – część sekwencyjna kodu.

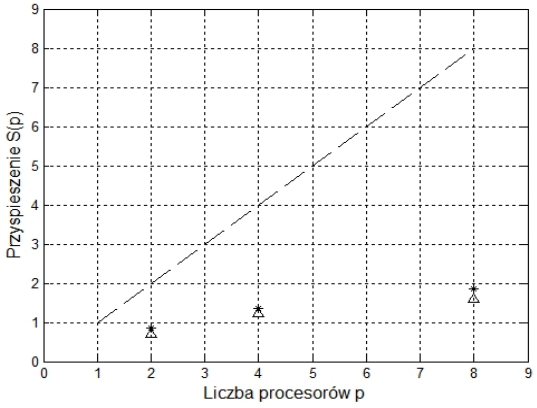
Jako miarę obliczeń równoległych przyjęto przyspieszenie obliczeń zdefiniowane jako stosunek czasu rozwiązania zadania sekwencyjnym algorytmem PECE na jednym procesorze ($T_S$) do czasu rozwiązania zadania algorytmem równoległym na p procesorach ($T_R(p)$):

$$S(p) = \frac{T_S}{T_R(p)}. \tag{1}$$

Koszt obliczeniowy funkcji DIF zdefiniowano przy pomocy liczby operacji mnożenia z akumulacją (ang. MAC). Poniżej na wykresach przedstawiono porównanie przyspieszenia zrównoleglonego algorytmu PECE dla 2, 4 i 8 procesorów oraz 8 równań w DIF o złożoności obliczeniowej 48 MAC (8 równań po 6 operacji MAC) (Rys.4) i 480 MAC (8 równań po 60 operacji MAC) (rys.5).




Rys.4. Przyśpieszenie obliczeń równoległego algorytmu PECE w zależności od liczby procesorów dla kosztu obliczeniowego w DIF=48 MAC. Linia kreskowana obrazuje idealne przyspieszenie, trójkątami zaznaczono wyniki przyspieszenia dla algorytmu gdzie zrównoleglono wszystkie sekcje, natomiast gwiazdkami zaznaczono wyniki przyspieszenia dla algorytmu gdzie predykcja i korekcja liczone są w wątku master w sposób sekwencyjny, zrównoleglono jedynie sekcje obliczeń DIF.




Rys.5. Przyśpieszenie obliczeń równoległego algorytmu PECE w zależności od liczby procesorów dla kosztu obliczeniowego w DIF=480 MAC. Linia kreskowana obrazuje idealne przyspieszenie, trójkątami zaznaczono wyniki przyspieszenia dla algorytmu gdzie zrównoleglono wszystkie sekcje, natomiast gwiazdkami zaznaczono wyniki przyspieszenia dla algorytmu gdzie predykcja i korekcja liczone są w wątku master w sposób sekwencyjny, zrównoleglono jedynie sekcje obliczeń DIF.

Na rysunkach 4 i 5 porównano wyniki przyspieszenia obliczeń równoległych w przypadku obliczeń algorytmu całkowicie równoległego (zrównoleglenie wszystkich sekcji) oraz algorytmu częściowo równoległego (predykcja i korekcja liczone są w wątku master w sposób sekwencyjny, zrównoleglone jedynie zostały sekcje obliczeń DIF). W przypadku pierwszego algorytmu i małego kosztu obliczeń w DIF opłacalność zrównoleglenia występuje w przypadku wykorzystania ośmiu rdzeni procesora (trójkąty na rys.4). W przypadku użycia drugiej wersji algorytmu opłacalność (choć niewielka) pojawia się już od przypadku wykorzystania czterech rdzeni (gwiazdki na rys.4). Bardziej kosztowne obliczenia w DIF (rys.5) zarówno w przypadku pierwszego algorytmu jak i drugiego powodują, że opłacalne staje się użycie czterech i ośmiu rdzeni. Dopiero zwiększenie zasobów (liczby wykorzystanych rdzeni procesora) powoduje opłacalność zrównoleglonych obliczeń, zarówno w przypadku pierwszego i drugiego porównywanego algorytmu, która jest i tak daleka od wartości przyśpieszenia (linia kreskowana) w przypadku obliczeń dobrze nadających się do zrównoleglenia (tzw. dekompozycja trywialna algorytmu i danych).

**Podsumowanie**

Skomplikowane i kosztowne obliczeniowo algorytmy predyktor-korektor typu PECE do rozwiązywania układów równań różniczkowych stosuje się jeśli zależy nam na większej dokładności rozwiązań, porównywalnej z wynikami uzyskiwanymi z metody R-K czwartego rzędu [1]. Dzięki tylko dwukrotnemu wywołanie funkcji określającej prawą stronę układu równań różniczkowych (DIF) następuje blisko dwukrotnie zmniejszenie kosztów obliczeniowym w stosunku do metody R-K czwartego rzędu. Zrównoleglenie tego typu algorytmu nie zawsze jest opłacalne z powodu tzw. drobnoziarnistości. Ta cecha spowodowana jest licznymi punktami synchronizacji algorytmu wynikającymi ze stosunkowo niedużych niezależnych bloków obliczeń podlegających zrównolegleniu. Jeśli koszty tworzenia i zarządzania wątkami są istotne w porównaniu z zyskiem ze zrównoleglenia obliczeń nie warto stosować tej metody poprawy wydajności obliczeń. Zależy to w dużej mierze od wydajności i architektury procesorów używanych w obliczeniach. Odpowiedź o opłacalność tych obliczeń może dać w zasadzie tylko próba optymalizacji [7] polegająca na precyzyjnym podziale zadań na poszczególne procesy. Szczególnie istotne są w tym przypadku powiązania poszczególnych równań w DIF umożliwiające możliwie najrówniejsze obciążenie procesów oraz właściwy dobór liczby wykorzystywanych procesorów i sposób dostępu do pamięci w celu wymiany danych.

LITERATURA


[1] Gardecki A., Macek-Kamińska K., Comparison of the influence of selected numerical integration algorithms on DTC induction motor drive, *Power electronics and electrical drives selected problems; Chapter: Electrical drives*, OWPW Wrocław, (2007)

[2] Krogh F.T., Changing stepsize in the integration of differential equations using modified divided differences, *Lecture Notes in Math.* 362, Springer-Verlag, Berlin-New York, (1974), 22-71

[3] Gardecki A., Macek-Kamińska K., Badania porównawcze wybranych procedur numerycznego rozwiązywania równań różniczkowych używanych w układach czasu rzeczywistego, *Przegląd Elektrotechniczny* R 84 (2008), nr. 11, 322-325

[4] Chapman B., Jost G., van der Pas R., Using OpenMP. Portable Shared Memory Parallel Programming, Cambridge, Mass: MIT Press, (2008)

[5] https://software.intel.com/en-us/vtuneampxe_2013_ug_win, (1.04.2014)

[6] Drongowski P. J., An introduction to analysis and optimization with AMD CodeAnalyst™ Performance Analyzer, Advanced Micro Devices, Inc., Boston Design Center (.pdf), (2008)

[7] Gardecki A., Zastosowanie techniki zrównoleglenia obliczeń do poprawy wydajności numerycznego algorytmu rozwiązywania równań różniczkowych, *Elektronika*, 12 (2012)



***Autor**: dr inż. Arkadiusz Gardecki, Politechnika Opolska, Instytut Układów Elektromechanicznych i Elektroniki Przemysłowej, ul. Prószkowska 76 bud. 1, 45-758 Opole, e-mail: a.gardecki@po.opole.pl.*