Arkadiusz GARDECKI

Politechnika Opolska, Instytut Układów Elektromechanicznych i Elektroniki Przemysłowej

Analiza kosztów obliczeń wybranych wariantów zrównoleglenia algorytmu rozwiązywania równań różniczkowych w obliczeniach obserwatora stanu

Streszczenie. *W artykule przedstawiono porównanie wydajności obliczeniowej procedury BGKODE_DSP służącej do rozwiązywania układów równań różniczkowych zwyczajnych (ODE) w obliczeniach równoległych na przykładzie obliczeń obserwatora stanu. Modyfikacje sposobu realizacji obliczeń w celu minimalizacji liczby punktów synchronizacji przyczyniły się do wzrostu jej wydajności obliczeniowej.*

Abstract. *This paper presents a performance analysis the BGKODE_DSP routine used to solve systems of ordinary differential equations (ODE) in the calculation of parallel computation on the example of a state observer. Modifications to the calculation method of execution in order to minimize the number of synchronization points contributed to the growth of its computational efficiency.*(***Analysis of the calculations cost of selected variants ODE algorithm parallelization during state observer calculations***).

Słowa kluczowe: obserwator stanu, metody numeryczne, obliczenia równoległe.
Keywords: state observer, numerical method, parallel computations.

doi:10.12915/pe.2014.02.52

Wstęp

Pomimo nieustannego wzrostu mocy obliczeniowych procesorów używanych w układach sterowania wciąż istotnym ograniczeniem używania bardziej skomplikowanych pod względem obliczeniowym algorytmów jest wymaganie dużej mocy obliczeniowej układów realizujących te zadania. Przykładem takiego algorytmu może być algorytm rozwiązywania układów równań różniczkowych zwyczajnych (ODE) typu predyktor – korektor (PECE) [1, 2]. Opisana w pracy [3] procedura BGKODE_DSP wykazała zalety związane z większą dokładnością w porównaniu do często stosowanych metod typu Rungego-Kutty (R-K) przy niewielkim i akceptowalnym zwiększeniu kosztów obliczeniowych. Wraz z upowszechnieniem się procesorów wielordzeniowych, zarówno w komputerach używanych do badań symulacyjnych jak i w procesorach sygnałowych wykorzystywanych do obliczeń w czasie rzeczywistym, naturalnym kierunkiem rozwoju tego typu metod jest przystosowanie ich do obliczeń równoległych.

Stosowanie algorytmu typu predykator-korektor do obliczeń układów równań różniczkowych jest szczególnie opłacalne jeśli chcemy otrzymać wyniki o dużej dokładności, porównywalnej z metodami R-K czwartego rzędu przy blisko dwukrotnie mniejszym koszcie obliczeniowym związanym z wywoływaniem prawej strony układu równań różniczkowych (DIF). Zrównoleglenie tych obliczeń jest uzasadnione przede wszystkim jeśli obliczanie funkcji w DIF jest kosztowne. Ma to miejsce jeśli liczba równań układu jest duża (np. w analizie dużych układów energoelektronicznych) lub równania są złożone obliczeniowo. Opłacalność ta rośnie także ze wzrostem przedziału całkowania lub jeśli układ równań różniczkowych musi być wielokrotnie całkowany (np. przy estymacji parametrów układu lub w zadaniach optymalizacji).

Sekwencyjny algorytm BGKODE_DSP

Procedura BGKODE_DSP wywodzi się z pełnego zespołu procedur BGKODE [2] realizującego zmodyfikowany algorytm Krogh'a [1]. Prowadzone w szerokim zakresie badania zespołu tych procedur wskazują na jej zalety, szczególnie w obliczeniach symulacyjnych. Ważniejsze zalety to: duża dokładność, duży obszar stabilność, efektywny algorytm zmiany długości kroku całkowania oraz rzędu aproksymacji, możliwość rozwiązywania układów sztywnostabilnych, możliwość rozwiązywania układów równań wyższych rzędów w sposób bezpośredni. Celem opracowana procedura BGKODE_DSP było przystosowanie algorytmów z pełnego zespołu procedur BGKODE do obliczeń w układach czasu rzeczywistego [3, 4]. Najważniejszą modyfikacją, z punktu widzenia czasu realizacji obliczeń, było dostosowanie jej do obliczeń ze stałym krokiem całkowania. Wynika to z charakteru realizowanych technik sterowania, które w cykliczny sposób dokonują aktualizacji informacji o napędzie i na tej podstawie wystawiają sterowanie. Założenie niezmienności kroku całkowania i ograniczenie rzędu aproksymacji upraszcza algorytm BGKODE i prowadzi do poniższej postaci:

$$
\begin{aligned}
P:\ & \boldsymbol{p}_{n+1} = \boldsymbol{y}_n + h\sum_{i=1}^{k} g_i \boldsymbol{\Phi}_i(n) \\
& \boldsymbol{\Phi}^e_{k+1}(n+1) = 0 \\
& \boldsymbol{\Phi}^e_i(n+1) = \boldsymbol{\Phi}^e_{i+1}(n+1) + \boldsymbol{\Phi}_i(n),\ i = k,\ldots,1 \\
E:\ & \boldsymbol{f}^p_{n+1} = \boldsymbol{f}(t_{n+1}, \boldsymbol{p}_{n+1}) \\
C:\ & \boldsymbol{y}_{n+1} = \boldsymbol{p}_{n+1} + h\ g_{k+1}(\boldsymbol{f}^p_{n+1} - \boldsymbol{\Phi}^e_1(n+1)) \\
E:\ & \boldsymbol{f}_{n+1} = \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}) \\
& \boldsymbol{\Phi}_{k+1}(n+1) = \boldsymbol{f}_{n+1} - \boldsymbol{\Phi}^e_1(n+1) \\
& \boldsymbol{\Phi}_i(n+1) = \boldsymbol{\Phi}^e_{i+1}(n+1) + \boldsymbol{\Phi}_{k+1}(n+1),\ i = k,\ldots,1
\end{aligned}
\tag{1}
$$

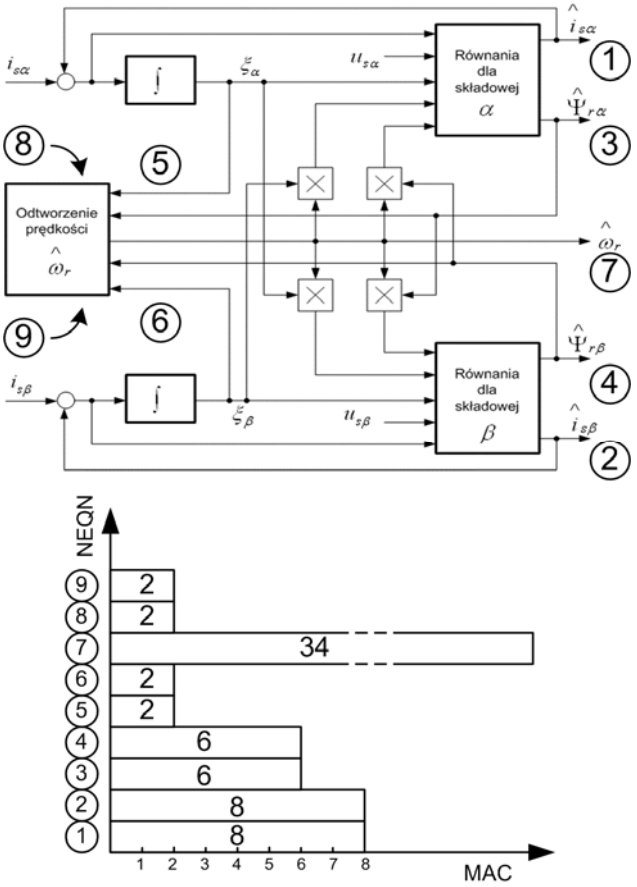
gdzie: P - oznacza predykcję, C: oznacza korekcję a E: oznacza obliczenie prawej strony układu równań różniczkowych (DIF). Występujące we wzorze (1) zmodyfikowane różnice dzielone [1, 2] $\boldsymbol{\Phi}_i$ oraz $\boldsymbol{\Phi}_i^e$ przechowują informacje z poprzednich kroków całkowania dzięki temu zapewniają dobrą reprezentację wielomianu, który interpoluje funkcję *w(t)* w równoodległych punktach [1]. Współczynniki g zależą od rzędu aproksymacji k i długości kroku całkowania. Na podstawie wielu testów ograniczono zmiany rzędu aproksymacji tylko do wybranego poziomu (k=2) [3, 4]. Wydaje się, że jest to dobrym kompromisem pomiędzy uzyskiwaną dokładnością a kosztem obliczeń. Z powodu braku możliwości powtórzenia kroku całkowania w układach realizujących obliczenia w czasie rzeczywistym zrezygnowano z kontroli błędu po etapie predykcji (jest on jednak dostępny w celach diagnostycznych lub na etapie testowania aplikacji).

Ważnym zadaniem podczas przygotowywania algorytmu do obliczeń równoległych jest podział zadań, które mogą być realizowane niezależnie oraz punktów wymiany wyników otrzymywanych z tych zadań. W algorytmie procedury BGKODE_DSP potencjał zrównoleglenia posiadają obliczenia sumy zależne od rzędu aproksymacji k i fakt, że wszystkie równania są obliczane osobno, czyli zależność od liczby równań różniczkowych (NEQN) zawartych w DIF. Ograniczeniem natomiast są punkty synchronizacji występujące po istotnych częściach algorytmu: P - predykcji, C - korekcji i E - obliczeniu prawej strony układu równań różniczkowych DIF oraz obliczeniu zmodyfikowanych różnic dzielonych $\mathbf{\Phi}_i(n+1)$ i $\mathbf{\Phi}_i^{\,e}(n+1)$.

W publikacji [5] przedstawiono kilka przykładowych wariantów zrównoleglenia obliczeń w przypadku różnej liczby procesorów (1, 2, 4 i 6) oraz założonej struktury układu równań w DIF a także ich koszt obliczeniowy wyrażony poprzez liczbę mnożeń z akumulacją (ang. MAC). Na rzeczywisty czas wykonywania procedur wpływa wiele czynników, takich jak architektura procesora, sposób synchronizacji procesów i wymiany danych, dostęp do pamięci oraz taktowanie procesora. Dlatego zdaniem autora uniwersalnym wskaźnikiem umożliwiającym porównanie kosztów obliczeń jest liczba operacji MAC, a nie czas wykonywania obliczeń. Niemożna jednak do końca pominąć tych dodatkowych czynników ponieważ w wielu systemach wieloprocesorowych (wielordzeniowych) właśnie przyjęta organizacja obliczeń może istotnie wpłynąć na wydajność algorytmu PECE o typie równoległości drobnoziarnistej. Oznacza to, że komunikacja następuje wielokrotnie w ciągu krótkiego przedziału czasu i wymaga wielu punktów synchronizacji. Może to powodować, iż pomimo zrównoleglenia obliczeń rzeczywisty czas ich wykonywania wcale nie ulegnie skrócenia, a nawet w szczególnych okolicznościach może się zwiększyć. Najczęstszym powodem tego typu zachowania są koszty tworzenia procesów i wątków, migracja wątków i sposób dostępu do pamięci w trakcie obliczeń. Analiza tych przykładów skłoniła autora do kolejnej próby poprawy wydajności tego algorytmu, tym razem poprzez zmniejszenie liczby punktów synchronizacji.

Układ równań obserwatora prędkości kątowej wirnika o równaniach zależnych

Złożoność rozwiązywanego układu równań różniczkowych zawartych w DIF wpływa na wydajność obliczeniową zrównoleglonego programu. W złożonych zagadnieniach zazwyczaj jest to dosyć kosztowna część obliczeń oraz dodatkowo w algorytmie PECE jest dwukrotnie wykonywana w jednym kroku całkowania. Równania w takim układzie najczęściej są zależne (w sensie powiązania z rozwiązaniami z innych równań układu). W przypadku gdy są niezależne liczba punktów wymiany podczas zrównoleglenia algorytmu BGKODE_DSP spada do zera (rys.3a). Ale jest to skrajny przypadek w którym możemy stworzyć NEQN niezależnych procesów i zmaksymalizować korzyści ze zrównoleglenia obliczeń. Przypadek ten jest jednak w praktyce rzadki. W prezentowanych badaniach użyto równań obserwatora prędkości kątowej wirnika opisanego szczegółowo w pracy [6]. Obserwator ten odtwarza składowe wektora strumienia wirnika i wektora prądu stojana. Układ sześciu równań różniczkowych opisujących ten obserwator i jego struktura przedstawiona jest na rysunku 1. Układ tych równań służy obliczeniu prędkości kątowej silnika asynchronicznego co poza strukturą logiczną dodatkowo dodaje trzy równania do układu równań różniczkowych. W tym bloku także są pobierane informacje o sygnałach pomiarowych (lub dane z modelu silnika w przypadku badań symulacyjnych) wykorzystywanych w obliczeniach. Powiązania pomiędzy równaniami powodują, że blok tych obliczeń wymaga ustalenia przed jego wykonaniem punktu wymiany danych.



Rys.1. Struktura obserwatora prędkości kątowej silnika z zaznaczonymi numerami równań oraz oszacowana liczba operacji MAC dla układu równań różniczkowych (1÷6) oraz pozostałych równań obserwatora (7÷9)

Na rysunku 1 można zauważyć, że znaczny koszt obliczeniowy związany jest z wyliczeniem prędkości kątowej (równanie 7) na podstawie rozwiązań układu równań różniczkowych. We wzorze tym występuje dzielenie i pierwiastkowanie, które to operacje są zwykle dosyć kosztowne obliczeniowo.

Układy do obliczeń równoległych i oprogramowanie wspomagające te obliczenia

Dostępnych jest coraz więcej układów umożliwiających obliczenia równoległe. Różnią się one liczbą możliwych do realizacji niezależnych procesów, sposobami dostępu do pamięci czy nawet możliwościami dostosowania do wymagań użytkownika (np. układy FPGA). Migracja kodu napisanego do wykonywania sekwencyjnego na układy realizujące obliczenia równoległe nie jest trywialnym zagadnieniem. Powstało wiele programów wspomagających podział zadań pomiędzy procesy i wątki realizowane równolegle. Chyba najbardziej obecnie popularne standardy zakładające pamięć współdzieloną to POSIX Threads i OpenMP. Dzięki nim możemy w przejrzysty sposób wyodrębnić obszary podlegające zrównolegleniu, sposób współdzielenia pamięci przez zmienne czy też punkty synchronizacji. Korzystając z OpenMP oraz komputera składającego się z dwóch 4 rdzeniowych procesorów Intel Xeon E5430 przetestowano wiele wariantów zrównoleglenia

i opcji obliczeń realizowanych przez procedurę BGKODE_DSP opisanego w rozdziale 3 układu obserwatora prędkości kątowej. Podczas próby określenia prawie optymalnych wartości istotnych dla wydajności obliczeniowej, rozumianej jako możliwie równe obciążenie procesorów i zrównoleglenie niezależnych operacji MAC, wykorzystano metodę polegającą na optymalizacji poszczególnych zasobów procesora osobno, przy niezmienianiu pozostałych czynników. Taki sposób postępowania zazwyczaj daje poprawne wyniki choć nie zawsze zapewnia to pełną optymalizację.

Organizacja obliczeń równoległych w algorytmie BGKODE_DSP

Rozpoczęcie obliczeń w przypadku metody k-krokowej (wykorzystującej informację z poprzednich kroków całkowania) wymaga znajomości przybliżeń początkowych z poprzednich kroków całkowania $y_1, ..., y_{k-1}$. W celu zapewnienia możliwości samodzielnego wystartowania procedury BGKODE_DSP realizowana jest następująca strategia: w pierwszym kroku obliczeń stosuje się metodę jednokrokową, w kolejnym kroku metodę dwukrokową i tak dalej. Stosowanie zmiennej krokowości (a co za tym idzie zmiennego rzędu aproksymacji) jest naturalnym procesem w trakcie obliczeń pełnego zespołu procedur BGKODE [2]. W analizowanym przypadku zrezygnowano z możliwości zmian zarówno długości kroku jak i rzędu aproksymacji k poza fazą startu. Inkrementacja rzędu aproksymacji k i związane z tym magazynowanie informacji w wektorze zmodyfikowanych różnic dzielonych wykonywane jest tylko dodatkowo w pierwszym kroku. Operacji tej nie opłaca się zrównoleglić ponieważ nie jest ona kosztowna obliczeniowo a wykonywanie jej na różnych jednostkach mogłoby ostatecznie spowolnić algorytm. Możemy ją w takim razie przypisać wątkowi głównemu.

Poniżej przedstawiono operację mnożenia z akumulacją (ang. MAC). Jest to podstawowa operacja wykonywana w wielu architekturach procesorów w jednym cyklu obliczeniowym, np. w procesorach sygnałowych:

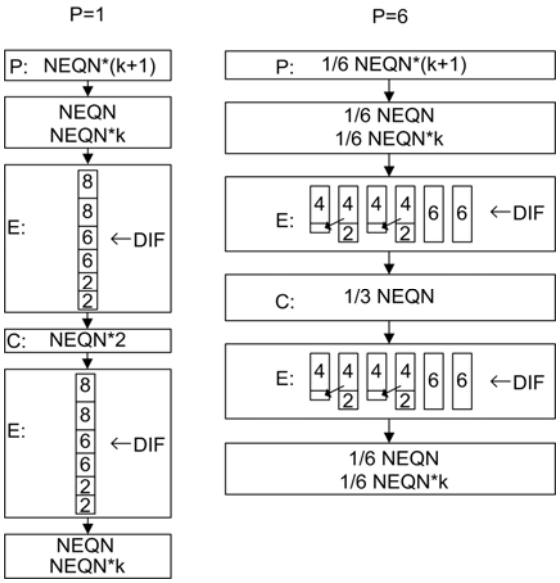
$$x \leftarrow x + y \times z \tag{2}$$

i odpowiada ona przykładowemu fragmentowi kodu procedury BGKODE_DSP:

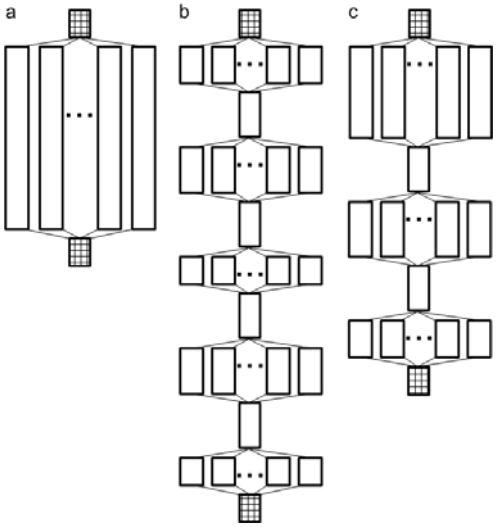
```
p[l] = p[l] + temp2 * phi[l,i]
```
(3)

Jako wskaźnik szybkości działania algorytmu przyjęto liczbę tych operacji wykonywanych przez program sekwencyjny oraz wersje zrównoleglone. Rzeczywisty czas wykonywania procedur najczęściej jest różny dla odmiennych architektur procesora, przez które sposób synchronizacji procesów, zarządzanie informacjami z części kodu i programu, dostęp do pamięci oraz taktowanie procesora są danymi trudnymi do określenia z góry i nawet producenci układów procesorowych najczęściej ich nie podają. Między innymi dlatego przyjęto, iż w miarę uniwersalnym wskaźnikiem szybkości działania algorytmu będzie liczba operacji MAC, a nie rzeczywisty czas wykonywania obliczeń na konkretnym typie procesora.

Poniżej przestawiono wybrane warianty organizacji obliczeń dla następujących wartości współczynników występujących w algorytmie BGKODE_DSP oraz określających złożoność obliczeń: liczba równań układu równań różniczkowych NEQN = 6, organizacja i złożoność równań w DIF jak na rysunku 1, rząd wielomianu aproksymującego k=2. Przyjęte założenia odpowiadają aplikacji tego algorytmu do obliczeń obserwatora prędkości kątowej silnika asynchronicznego [6]. Na rysunku 2 porównano dwa przypadki organizacji obliczeń pod względem liczby operacji MAC. Pierwszy przedstawia obliczenia sekwencyjne, odpowiednio o liczbie procesorów P = 1 drugi natomiast przedstawia przypadek obliczeń realizowanych na 6 procesorach (liczba procesorów równa liczbie równań różniczkowych rozwiązywanych w DIF). W bloku DIF uwzględniono tylko równania różniczkowe (1÷6).



Rys.2. Schemat organizacji obliczeń pod względem liczby operacji MAC dla układu sekwencyjnego (P=1) oraz w przypadku układu z sześcioma procesorami (P=6)



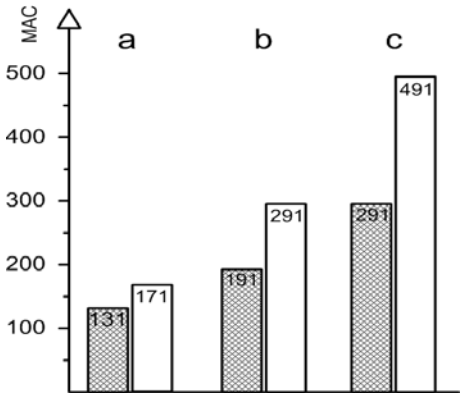
Rys.3. Symboliczny schemat organizacji obliczeń równoległych w BGKODE_DSP w przypadku: a - braku powiązań pomiędzy równaniami w DIF, b – powiązania pomiędzy równaniami w DIF i 4 punkty synchronizacji oraz c- powiązania pomiędzy równaniami w DIF i 2 punkty synchronizacji

Na rysunku 2 strzałkami pionowymi zaznaczono punkty synchronizacji. Jest ich stosunkowo dużo co utrudnia wykonywanie większych bloków. Największy pod tym względem potencjał przyspieszenia obliczeń posiada dwukrotnie wykonywany w jednym kroku całkowania blok obliczania DIF. Widać to wyraźnie w przypadku obliczeń przy użyciu sześciu procesorów. W blokach E: przedstawiono sposób ich obliczania z uwzględnieniem podziału na dostępne procesory. Strzałkami skośnymi zaznaczono operacje sumowania (pusty bloczek) wyników pochodzących z obliczeń na innej jednostce. Taki podział zadań wydaje się najbardziej korzystny z punktu widzenia obciążenia

procesorów. Strzałkami oznaczono przekazanie do wspólnego obszaru pamięci obliczonej wcześniej zmiennej.

Na rysunku 3 porównano sposób organizacji obliczeń pod względem liczby punktów synchronizacji dla trzech wariantów obliczeń. Przyjęto, że najlepiej ze względu na równe obciążenie procesorów i najkrótszy czas obliczeń przyjąć ich liczbę równa liczbie rozwiązywanych równań (NEQN) [5]. W pierwszym wariancie (rys.3a) obliczenia są realizowane w niezależnych wątkach na NEQN jednostkach obliczeniowych. Ten przypadek jest najwydajniejszy pod względem kosztów obliczeniowych związanych z synchronizacją obliczeń, ale możliwy jedynie w przypadku braku powiązań między równaniami w DIF. W praktyce jest to rzadki przypadek. Drugi wariant (rys.3b) pokazuje liczbę punktów synchronizacji w pierwszej wersji zrównoleglonego algorytmu realizowanego w procedurze BGKODE_DSP [5]. Posiadał on 4 punkty synchronizacji.

Połączenie bloku nadpisywania zmodyfikowanych różnic dzielonych po korekcji z blokiem predykcji oraz połączenie obliczania układu równań w DIF po predykcji z korekcją w jeden blok możliwy do obliczeń równoległych w NEQN niezależnych procesach umożliwiło zmniejszenie liczby punktów synchronizacji (LPS) do 2 (rys.3c). W przypadku równoległości drobnoziarnistej ma to istotne znaczenie na wydajność obliczeń równoległych. Próba optymalizacji najbardziej czasochłonnej części obliczeń – równań zawartych w DIF pod względem równomiernego podziału zadań na dostępne procesory wymaga wielu prób i trudno ten proces sformalizować.



Rys.4. Koszty obliczeń równoległych (w jednym kroku) obserwatora stanu (rys.1) wyrażone liczbą operacji MAC dla schematu obliczeń z 2 punktami synchronizacji (kratkowane wypełnienie) oraz z 4 punktami synchronizacji (brak wypełnienia) dla przykładowych trzech wartości kosztów synchronizacji: a=20 MAC, b=50 MAC i c=100 MAC

Na rysunku 4 porównano przykładowe koszty obliczeniowe wyrażone w liczbie operacji MAC dla 2 i 4 punktów synchronizacji. Założono, że koszty towarzyszące tworzeniu wątków po synchronizacji są stałe i przyjęto ich trzy przykładowe wartości. W pierwszym przypadku reprezentują one małe koszty synchronizacji (20 operacji MAC), w drugim przypadku średnie koszty (50 operacji MAC) i w trzecim przypadku duże koszty (100 operacji MAC). W rzeczywistości wielkości te mogą być zmienne i zależą od architektury układu wykonującego obliczenia równoległe. Można je traktować jako wielkość charakterystyczną dla danego systemu obliczeniowego.

Podsumowanie

Zastosowanie algorytmu rozwiązywania układów równań różniczkowych typu predykator-korektor jest uzasadnione jeśli chcemy otrzymać wyniki o dużej dokładności, porównywalnej z metodami R-K czwartego rzędu [3, 4] przy blisko dwukrotnie mniejszym koszcie obliczeniowym. Zrównoleglenie tych obliczeń jest uzasadnione przede wszystkim jeśli obliczanie funkcji prawej strony równania różniczkowego DIF jest kosztowne. Mamy z tym do czynienia jeśli liczba równań układu jest duża. Przykładem może być rozbudowana analiza układów energoelektronicznych [7, 8] gdzie układ równań różniczkowych musi być wielokrotnie całkowany (np. przy estymacji parametrów układu lub w zadaniach optymalizacji).

W porównaniu do aplikacji sekwencyjnych w przypadku aplikacji wielowątkowych pojawiają się dodatkowe problemy. Kłopotliwe jest na przykład usuwaniem błędów w fazie testowania. Niektóre błędy są niepowtarzalne, ich wystąpienie jest spowodowane incydentalną konfiguracją wielu czynników mających wpływ na synchronizację procesorów i ich przydzielanie do aktualnych zasobów. Mimo to zalety z poprawnie działających programów przystosowanych do obliczeń równoległych przeważają nad tymi niedogodnościami.

Modyfikacja kolejności wykonywanych obliczeń i zgrupowanie ich w bloki w taki sposób aby zmniejszyć liczbę punktów synchronizacji umożliwiła poprawę wydajności procedury BGKODE_DSP w obliczeniach równoległych. Zmniejszenie liczby punktów synchronizacji i związanych z nimi kosztów tworzeniem procesów, wątków, ich migracją w przypadku obliczeń drobnoziarnistych wyraźnie skraca czas obliczeń. Poprawa ta zależy oczywiście od wielu czynników (architektura jednostek obliczeniowych, sposób zarządzania procesami i/lub watkami). Pomimo swoich zalet interfejs OpenMP wymaga od użytkownika wiele uwagi w celu dopasowania do warstwy sprzętowej np. w celu częściowej optymalizacji kosztów obliczeniowych. Podobnie dużo uwagi wymaga podziała zadań na procesory w bloku DIF w przypadku powiązań pomiędzy równaniami i różnym stopniu ich złożoności obliczeniowej.

LITERATURA

[1] 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

[2] Beniak R., Gardecki A., Procedura całkująca układy równań różniczkowych zwyczajnych z wykorzystaniem zmodyfikowanego algorytmu Krogha, *Prace X Konferencji Symulacja Procesów Dynamicznych*, Zakopane, (1998), 23-30

[3] 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)

[4] 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

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

[6] Krzemiński Z., Cyfrowe sterowanie maszynami asynchronicznymi, Wydawnictwo PG, (2001)

[7] Beniak R., Gardecki A., Analiza wielowariantowa napędu przekształtnikowego umożliwiająca ocenę sprawności i oddziaływania na środowisko, *Przegląd Elektrotechniczny* R 87(2010), nr. 2, 22-25

[8] Beniak R., Gardecki A., Computationally efficient method of simulation of an electric drive impact on the power grid, *Archives of Electrical Engineering* VOL. 62(1), (2013), 77-90

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