dr inż. Bogumiła Mrozek
Instytut Informatyki, Wydział Fizyki, Matematyki i Informatyki
Politechnika Krakowska

OBLICZENIA RÓWNOLEGŁE W MATLAB-ie

MATLAB jest językiem wysokiego poziomu do obliczeń technicznych oraz interaktywnym środowiskiem przeznaczonym do projektowania algorytmów, analizy i wizualizacji danych oraz obliczeń numerycznych. Do MATLAB-a wbudowano operacje na wektorach, macierzach i tablicach, które tworzą matematyczną podstawę do obliczeń naukowych i technicznych. Pozwala to na szybsze tworzenie i wykorzystywanie algorytmów obliczeniowych – niż przy użyciu tradycyjnych języków (C, Fortran), gdyż przy realizacji zadań na niskim poziomie nie ma potrzeby deklarowania zmiennych, ich typów i adresów. Obliczenia równoległe pozwalają realizować na komputerach wielordzeniowych, wieloprocesorowych i klastrach, zadania intensywne numerycznie i z dużą ilością danych.

W artykule opisano możliwości wykonywania obliczeń równoległych w środowisku MATLAB v. 7.11 (R2010b) z wykorzystaniem jego bibliotek Parallel Computing Toolbox v.5.0 oraz MATLAB Distributed Computing Server v.5.0.

PARALLEL COMPUTING WITH MATLAB

MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. The MATLAB language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables faster development and execution of algorithms than with traditional languages (C, FORTRAN) because it do not needs to perform low-level administrative tasks, such as declaring variables, specifying data types, and allocating memory. Parallel computing lets solve computationally and data-intensive problems using multicore processors, GPUs, and computer clusters.

In this paper, the application of the parallel computing in MATLAB v. 7.11 (R2010b) environments has been described with using Parallel Computing Toolbox v.5.0 and MATLAB Distributed Computing Server Version 5.0.

1. WPROWADZENIE

MATLAB znajduje zastosowanie w różnych dziedzinach – od matematyki, poprzez projektowanie, optymalizację i weryfikację układów mechatronicznych aż po nauki biologiczne. Dzięki wydajnym algorytmom obliczeniowym i mechanizmom analizy wyników, umożliwia szybkie i efektywne przeprowadzanie złożonych obliczeń oraz prezentację wyników w postaci grafiki dwu- i trójwymiarowej z wykorzystaniem animacji i generowania dźwięków. Umożliwia też generowanie kodu *OpenGL* dla akceleratorów sprzętowych.

MATLAB jest obiektowo zorientowanym językiem programowania wysokiego poziomu. Jego operatory i funkcje stosuje się do obliczeń numerycznych, w tym na macierzach i liczbach zespolonych. Posiada interakcyjne, przyjazne użytkownikowi środowisko pracy.

Otwarta architektura, rozszerzalność i możliwość wyboru platformy sprzętowej, pozwoliła na szybki rozwój tego środowiska. Istotny wkład ma tu wiele firm (ponad 300) współpracujących z producentem MATLAB-a. Dostarczają one produkty oparte na MATLAB-ie lub interfejsy łączące MATLAB z ich własnymi produktami.

Profesjonalna biblioteka matematyczna i graficzna jest oparta na optymalizowanych pod kątem operacji blokowo-macierzowych bibliotekach: LAPACK (*Linear Algebra Package*), BLAS (*Basic Linear Algebra Subroutines*) i FFTW (*Fast Fourier Transform In the West*). Stanowi ona bazę dla wszystkich elementów środowiska MATLAB. Wbudowano ją częściowo do jądra MATLAB-a, a w części ma ona postać plików zewnętrznych [4, 5, 9].

Pierwsze biblioteki tworzące tzw. *MATLAB równoległy* (ang. *parallel MATLAB*) powstały w latach 2004/05. Biblioteki te są rozwijane i unowocześniane, z powodów jak poniżej [3]:

- MATLAB przekształcił się z pakietu do prostych obliczeń z zakresu algebry macierzowej ("Matrix Laboratory") w dojrzałe środowisko do obliczeń technicznych, które obsługuje projekty dużej skali zawierające zacznie więcej algorytmów numerycznych niż te z zakresu algebry liniowej.
- Dzisiejsze mikroprocesory mają od 2 do 6 rdzeni (można oczekiwać, że w przyszłości będzie ich nawet więcej), ponadto nowoczesne komputery mają skomplikowaną hierarchiczną strukturę pamięci.
- Większość użytkowników pakietu MATLAB ma teraz dostęp do klastrów i komputerów pracujących w sieci, bądź posiada wielordzeniowe komputery osobiste z możliwością wykonywania obliczeń równoległych.

2. OBLICZENIA WYSOKIEJ WYDAJNOŚCI W MATLAB-ie

Poszerzeniem możliwości pakietu (języka) MATLAB w zakresie paradygmatu programowania równoległego (w porównaniu z wersją podstawową sekwencyjną) są biblioteki dodatkowe:

- ***Parallel Computing Toolbox*** pozwala rozwiązywać zadania złożone obliczeniowo i problemy intensywnego przetwarzania danych z zastosowaniem wielordzeniowych procesorów, procesorów graficznych GPUs (ang. *graphics processing units*) i klastrów komputerowych. Konstrukcje wysokiego poziomu takie, jak: równoległe pętle *for*, specjalne typy tablic i zrównoleglone algorytmy numeryczne umożliwiają zrównoleglenie aplikacji MATLAB-a bez stosowania elementów języków programowania równoległego jak np. CUDA lub MPI [5],[7].
- ***MATLAB Distributed Computing Server*** pozwala użytkownikom rozwiązywać zadania złożone obliczeniowo oraz problemy intensywnego przetwarzania danych poprzez wykonywanie aplikacji środowiska MATLAB/Simulink na komputerach klastrowych (ang. *computer cluster*) [5, 8].

2.1. Obliczenia równoległe

Obliczenia równoległe to taki sposób wykonywania obliczeń komputerowych, w którym wiele instrukcji jest wykonywanych jednocześnie. Taka forma przetwarzania danych była stosowana przez wiele lat, głównie przy korzystaniu z superkomputerów. Zyskała ona szczególne zainteresowanie w ostatnich latach, z uwagi na fizyczne ograniczenia uniemożliwiające dalsze zwiększanie częstotliwości taktowania procesorów. Obliczenia równoległe stały się dominującym wzorcem w architekturze komputerowej, głównie za sprawą upowszechnienia procesorów wielordzeniowych.

Jedną z najwcześniejszych klasyfikacji systemów dla równoległych (i sekwencyjnych) komputerów i programów stworzył M. J. Flynn. Zaklasyfikował on programy i komputery według tego, czy dany program lub komputer korzysta z jednego czy z wielu zbiorów instrukcji oraz czy te instrukcje korzystają z jednego, czy z wielu zbiorów danych.

W obrębie podziału, M. J. Flynn wyróżnił cztery klasy:

- **SISD** (ang. S*ingleIinstruction-Single-Data*) − równoważna przetwarzaniu całkowicie sekwencyjnemu,
- **SIMD** (ang. *Single-Instruction-Multiple-Data*) − te same operacje wykonuje się na różnych zbiorach danych,
- **MISD** (ang. *Multiple-Instruction-Single-Data*) − różne operacje wykonuje się na tym samym zbiorze danych,
- **MIMD** (ang. *Multiple-Instruction-Multiple-Data*) − różne operacje wykonywane są na różnych zbiorach danych. Jest to najczęstszy przypadek w przetwarzaniu równoległym.

Ze względu na skalę można wyróżnić obliczenia równoległe: na poziomie bitów, instrukcji, danych oraz zadań.

Ze względu na poziom, na którym sprzęt wspomaga operacje równoległe, można wyróżnić:

- Komputery symetryczne wieloprocesorowe (zawierają kilka identycznych, równorzędnych procesorów). ***SMP*** (ang. *Symmetric Multiprocessing*, przetwarzanie symetryczne), w tym jednoprocesorowe wielordzeniowe (zawierają jeden procesor wielordzeniowy). Jest to architektura komputerowa, która pozwala na znaczne zwiększenie mocy obliczeniowej systemu komputerowego przez wykorzystanie dwóch lub więcej procesorów do jednoczesnego wykonywania zadań.
  W *architekturze SMP* procesory te współdzielą zasoby pamięci oraz wejścia/wyjścia poprzez magistralę. Ważną rzeczą jest obsługa wielowątkowości przez system operacyjny i wykorzystanie wielowątkowości w programach komputerowych − upraszcza to możliwość "podzielenia" procesu dla kilku procesorów.
- Systemy składające się z wielu komputerów:
  - *Klastry komputerowe* (ang. *cluster*) − grupa połączonych jednostek komputerowych, które współpracują ze sobą w celu udostępnienia zintegrowanego środowiska pracy. Komputery wchodzące w skład klastra są nazywane węzłami (ang. *node*).
  - *Systemy MPP* (ang. *Massively Parallel Processors*) komputery masowo równoległe – rodzaj architektury komputerowej, której zadaniem jest umożliwienie przetwarzania współbieżnego (jednoczesnego) na wielu procesorach. Alternatywną architekturą do MPP (przetwarzanie równoległe) jest SMP(przetwarzanie symetryczne).
  - *Grid komputerowy*. Grid (ang. *grid – przetwarzanie sieciowe*) to system, który integruje i zarządza zasobami, które są pod kontrolą różnych domen (od instytucji po system operacyjny) i są połączone sieciami komputerową. System ten udostępnia standardowych, otwartych protokołów i interfejsów ogólnego przeznaczenia (odkrywania i dostępu do zasobów, autoryzacji, uwierzytelniania) oraz dostarcza usług odpowiedniej jakości.

Do prowadzenia obliczeń równoległych, oprócz sprzętu, konieczne są również odpowiednie algorytmy nazywane równoległymi. Są one trudniejsze w implementacji niż algorytmy sekwen-cyjne, ponieważ współbieżność wprowadza dodatkowe możliwości popełnienia błędu. Powstają tak że dodatkowe problemy w uzyskaniu wysokiej wydajności, powodowane koniecznością uwzględnienia mechanizmów komunikacji i synchronizacji obliczeń.

MATLAB obsługuje trzy rodzaje obliczeń równoległych [3]:

- **Równoległość wielowątkowa** (ang. *multithreaded parallelism*)**,** wątek (ang. *thread*) to pojedynczy proces wykonywany w wielozadaniowym systemie operacyjnym. W równoległościach wielowątkowych, jedna zmienna typu obiektowego MATLAB-a automatycz-

nie tworzy wiele jednoczesnych potoków instrukcji. W iele procesorów lub rdzeni, współużytkuje pamięć jednego komputera przy wykonywaniu tych potoków instrukcji.

- **Obliczenia rozproszone** (ang. *distributed computing*) W obliczeń rozproszonych, wiele zmiennych typu obiektowego MATLAB-a uruchamia wiele niezależnych obliczeń na oddzielnych komputerach, każdy z własną pamięcią. W większości przypadków jeden program jest uruchamiany wiele razy z różnymi parametrami lub z różnymi losowymi wartościami początkowymi.
- **Równoległości jawne** (ang. *explicit parallelism*) Przy zrównolegleniu wyraźnym (jawnym), kilka zmiennych typu obiektowego MATLAB-a działa na kilku procesorach lub komputerach, często z osobnymi pamięciami, i jednocześnie wykonuje pojedyncze polecenie MATLAB-a lub M-funkcji. Ten rodzaj równoległości opisują nowe konstrukcje programowania, które zawierają równoległe pętle i tablice rozproszone.

Te trzy rodzaje obliczeń równoległych mogą współistnieć. Na przykład, zadanie dotyczące obliczeń rozproszonych może wywoływać funkcje wielowątkowe na każdym komputerze, a następnie użyć rozproszonej tablicy, aby zebrać wyniki końcowe. Dla równoległości wielowątkowych, liczbę wątków można ustawić w panelu MATLAB Preferences. W pakiecie MATLAB zastosowano *Intel Math Kernel Library*, która zawiera wielowątkowe wersje BLAS (*Basic Linear Algebra Subroutines*). Dla argumentów wektorowych, biblioteka funkcji elementarnych MATLAB-a, która zawiera funkcje wykładnicze i trygonometryczne, jest wielowątkowa.

2.2. Przyjazne środowisko dla wykonywania obliczeń równoległych

Celem nadrzędnym firmy *MathWorks* jest rozszerzenie istotnych dla użytkowników zalet MATLAB-a na środowisko komputerów wielordzeniowych i klastrów, z uwzględnieniem takich cech jak: interaktywność i obsługa wielu platform [5]. Przyjęto następujące założenia:

- Użytkownicy powinni mieć możliwość wykonania dowolnego kodu MATLAB i modeli Simulinka na komputerach wieloprocesorowych (na klastrach) i z procesorami wielordzeniowymi. W dalszym ciągu jest to najważniejszy cel tego projektu.
- Użytkownicy powinni mieć możliwość korzystania z dobrze im znanej składni języka MATLAB dla wszystkich zadań związanych z pisaniem i wykonywaniem programów równoległych w MATLAB-ie.
- Użytkownicy powinni mieć dostęp do podstawowych konstrukcji języka, aby wyrazić równoległości. Konstrukcje te nie powinny zmieniać się znacząco w trybie pracy do tworzenia programów równoległych. Użytkownik nie powinien martwić się o architekturę lub konstrukcje dla systemów specyficznych, albo musieć radzić sobie z wątkowaniem, zarządzaniem danych lub synchronizacją.
- Programowalność (ang. p*rogrammability*) rozumiana jako łatwość wprowadzania zmian w oprogramowaniu, będzie zawsze atutem w porównaniu z innymi zadaniami. Użytkownicy powinni umieć tworzyć programy, które są poprawne, czytelne, łatwe do debugowania oraz łatwe w utrzymaniu.
- Język programowania powinien być całkowicie niezależny od alokacji zasobów. Ten sam program powinien działać poprawnie na jednym procesorze lub na kilkuset procesorach.

2.3.Zastosowane rozwiązania i terminologia

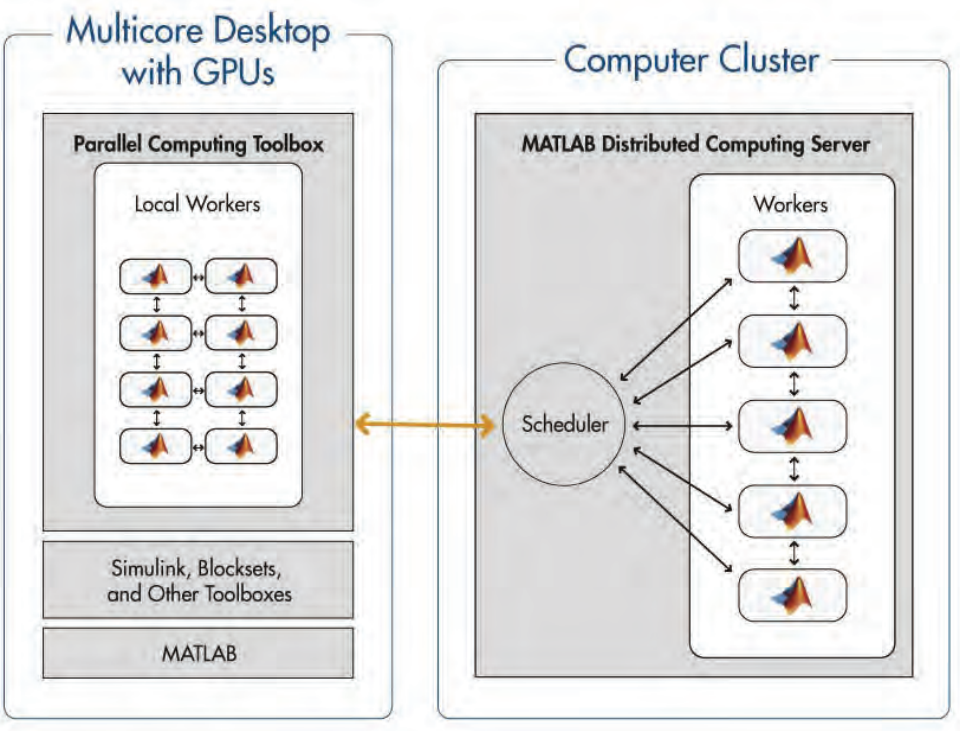
W narzędziach do obliczeń równoległych udostępnionych w MATLAB-ie wyróżniono elementy związane z infrastrukturą i składowe języka. Język wprowadza konstrukcje takie jak pętle równoległe, tablice rozproszone i funkcje przekazywania komunikatów.

Składową infrastruktury jest mechanizm funkcjonowania leżący poniżej konstrukcji języka obejmujący przetwarzanie danych i transfer kodu, zestawienie środowiska wykonawczego itp.

MATLAB Distributed Computing Server uruchamia kilka *worker* (ang. *worker*), które otrzymują zadania obliczeniowe (ang. *task*) od MATLAB-a klienta poprzez funkcje z *Parallel Computing Toolbox*. Termin *worker* jest używany jako ogólne pojęcie, które określa silnik procesów obliczeniowych MATLAB-a, uruchamiany na klastrze jako część *MATLAB Distributed Computing Server* (rys. 1).

Serwer ten obsługuje tryb operacji, w których funkcje *worker*'ów są całkowicie niezależne od siebie i nie wymagają żadnej konfiguracji infrastruktury komunikacji pomiędzy *worker*'ami. Infrastruktura komunikacji jest jednak potrzebna przy stosowaniu takich konstrukcji jak funkcje przekazywania komunikatów (ang. *message passing functions*) i tablic rozproszonych (ang. *distributed arrays*). Wtedy, te połączone *worker* są nazywane '*labs*'. Różnorodne funkcje przekazywania komunikatów przenoszą '*lab*' w ich nazwach, aby je rozróżnić.

Kompletną operację dużej skali do wykonania w MATLAB-ie, która składa się ze zbioru zadań (*task*), nazywano '*job*'. *MPI* (*Message Passing Interface*) oznacza sposoby komunikacji, pomiędzy '*labs*', gdy wykonywane są zadania (*task*) w tym samym '*job*'.



Rys. 1. Obliczenia równoległe z MATLAB-em. Można użyć *Parallel Computing Toolbox* do uruchamiania aplikacji na wielordzeniowych procesorach z ośmioma *worker* (dostępnych w tej bibliotece) lub na klastrze (z *MATLAB Distributed Computing Server*) [7]

3. ROZSZERZENIA JĘZYKA MATLAB O KONSTRUKCJE ZRÓWNOLEGLONE

Rozszerzeń języka MATLAB o zbiory zrównoleglonych struktur danych i zrównoleglonych konstrukcji dokonano tak, aby środowisko pracy było nadal przyjazne dla użytkowników [5]. Nowe konstrukcje i struktury są niezbędne, aby udostępnić użytkownikom język programowania niezależny od przydzielania zasobów i wcześniejszych wersji implementacji. Poszerzenie możliwości języka MATLAB wykonano tak, aby użytkownicy nie musieli wprowadzać większych zmian w już istniejących i poprawnie działających aplikacjach.

3.1. Funkcje przekazywania komunikatów

Istniejącą implementację MPI (ang. *Message Passing Interface*) udostępniono w bardzo uproszczonej i łatwej do stosowania formie, z zachowaniem bogactwa modelu programowania z przekazywaniem komunikatów.

Ładowanie, inicjalizacja, na koniec usunięcie z przydzielonego obszaru pamięci i na wreszcie zakończenie sesji były pracami standardowymi. Użytkownicy końcowi mogą bezpośrednio rozpocząć pracę korzystając z dostępnych funkcji przekazywania komunikatów niezależne od środowiska, w którym chcieli oni pracować – na przykład w środowisku równoległym interaktywnym lub środowisku procesu wsadowego.

Ponadto wprowadzono wystarczająco duże uogólnienia, aby użytkownicy mogli wymieniać dowolne typy danych MATLAB-a bez specjalnych przygotowań takich jak deklaracje typów danych (ang. *MPI_Datatype*). Funkcje przekazywania komunikatów w MATLAB-ie są funkcjami o wysokim poziomie abstrakcji zapisanym w standardzie MPI-2. Aktualnie język MATLAB-a nie obsługuje określonych przez użytkownika komunikatorów MPI.

Jednym z najbardziej istotnych wymagań dla tej biblioteki funkcji przekazywania komunikatów jest zdolność do nieznacznej zmiany konfiguracji dowolnego typu danych MATLAB-a – łącznie z tablicami numerycznymi dowolnej precyzji, tablicami zawierającymi struktury i tablicami komórkowymi (tablice MATLAB-a mogą zawierać dane dowolnego typu).

W normalnych warunkach użytkowania, biblioteka MPI oczekuje informacji o rozmiarze danych, które są faktycznie przekazywane. Użytkownik nie musi mieć tej informacji, dla dowolnej tablicy MATLAB-a. Aby rozwiązać ten problem zaprojektowano protokół, w którym można wysłać dwie wiadomości – pierwsza to bardzo krótki nagłówek komunikatu o znanym rozmiarze, wskazujący spodziewany typ danych w MATLAB-ie i drugi komunikat zawierający aktualny blok danych. Dla wystarczająco małych wielkości danych można po prostu skompresować całość danych do nagłówka komunikatu.

Dla nienumerycznych typów danych, informacja o typie danych MATLAB-a jest wstawiana wewnątrz nagłówka informacji (komunikatu) i przekształca nadchodzący potok danych binarnych na odbiorczy *'lab'*. Aktualny blok danych jest budowany przez szeregowanie tablic danych MATLAB-a, które mogą być rozszeregowane i odtworzone na odbiorczym *'lab'*. Dla typów danych MATLAB-a, które mogą być bezpośrednio odwzorowane na typ danych MPI, np. typ `double` MATLAB-a na MPI_DOUBLE itp., proces szeregowania i rozszeregowania jest pomijany i dane te są wysyłane bezpośrednio.

Protokół ten pozwala na detekcję błędów, detekcję blokady (zakleszczenia), usuwanie komunikatów i kolejek komunikatów, gdy pojawia się detekcja błędów komunikatów. Komunikaty są gotowe do wysłania tak szybko, jak to tylko możliwe bez żadnego zauważalnego (istotnego) wpływu użytkownika. Jednak, jeśli na przykład nie jest możliwe wysłanie komunikatu i potrzebne jest zablokowanie MATLAB-a, należy użyć dodatkowych cykli do obsłużenia błędów takich jak blokady.

Cykliczne blokady i niedopasowania komunikacyjne są powszechnie spotykanymi błędami w czasie wykonywania programów obliczeń równoległych, korzystających z przekazywania komunikatów. Dlatego zaimplementowano mechanizm detekcji dla obu typów błędów. Zaprojektowano i skonfigurowano dedykowany komunikator MPI dla każdego typu błędu. Działa on na początku wykonywania programu i nie ma wpływu na prawidłowe wykonanie algorytmu. Użytkownik może opcjonalnie włączyć mechanizmy wykrywania błędów dla de-

bagowania procesów wsadowych. Mechanizmy wykrywania błędów są automatycznie włączane w trakcie trwania sesji interaktywnych.

3.2. Dane rozproszone

Przekazywanie komunikatów jest najczęściej stosowaną metodą do tworzenia programów równoległych na komputerach klastrowych. Semantyka tablic globalnych zmniejsza złożoność programowania, pomijając szczegóły przesyłania komunikatów i pozwala użytkownikom pisać programy, które wyglądają jak pisane seryjnie.

W modelu PGAS (ang. *Partitioned Global Address Space*) dla programów SPMD (ang. *Single Program Multiple Data*), wiele wątków SPMD lub procesów wydziela część ich przestrzeni adresowej [5]. Ta wydzielona przestrzeń jest partycjonowana na części umiejscowione w każdym wątku lub procesie.

Programy wykorzystują lokalność. Oznacza to, że obliczenia wykonane przez wątek lub proces zależą tylko od wartości przechowywanych w bliskich adresach pamięci. Rozproszone tablice MATLAB-a są implementacją modelu PGAS. Tablice te mogą być konstruowane przez konkatencję części fragmentów tablic o podobnych rozmiarach w *worker*'ach, generowanie dużych macierzy w postaci tablic rozproszonych lub poprzez użycie specjalnych konstruktorów, które są przeciążoną postacią ich licznika części sekwencyjnych. Poniżej zamieszczono przykłady trzech sposobów tworzenia takich tablic rozproszonych [5]:

```
% Distributing an existing array "A" which
% is the same on all labs
dA = distributed(A, 'convert');
% Joining pieces on workers
localB = labindex() * rand(10000, 10000);
dB = distributed(localB);
% Using constructor functions
dC = rand(10000, 10000, distributor());
```

Przykład pierwszy zawiera replikowaną macierz `A`, która jest identyczna we wszystkich '*labs*'. Tablica rozproszona `dA` ma ten sam rozmiar, co `A`, ale lokalne fragmenty do każdego '*lab*' przechowują tylko podzbiory danych zawarte w `A`.

W drugim przykładzie `localB` jest wariantem macierzy zawierającej różne wartości, lecz o tych samych rozmiarach na różnych *'labs'*. Syntaktyczny zapis wywołania jest przykładem konkatenacji tablic wzdłuż kolumn. W ten sposób `dB` ma identyczną liczbę wierszy jak `localB`, lecz liczbę kolumn ma równą *numlabs* razy liczba kolumn macierzy `localB`. Dodatkowe argumenty mogą być dostarczone przy rozdzielaniu danych tj. tworzeniu danych rozproszonych. W tym przypadku '*labs*' potrzebują tylko wymiany meta-danych, aby utworzyć tablicę rozproszoną `dB`.

W przykładzie trzecim użyto przeciążonej funkcji `rand` z obiektem *distribution*, jako dodatkowy argument, celem utworzenia losowo rozproszonej tablicy `dC` (`distributor()` jest domyślnym konstruktorem obiektu *distribution*). Dodatkowe argumenty w konstruktorze obiektu *distribution* mogą być użyte do określenia własnej dystrybucji danych.

Tablice rozproszone są zaimplementowane jako warstwa biblioteczna na szczycie infrastruktury MPI. Użytkownik otrzymuje udział w widoku pamięci w środowisku wykonawczym. Nie wprowadzono żadnych założeń dotyczących tego środowiska za wyjątkiem tego, że jest wymagane, aby infrastruktura MPI była zainstalowana i zainicjowana dla tych tablic, które mogą być równoległymi strukturami danych.

Jedną z dużych zalet tablic rozproszonych jest to, iż współpracują one z dowolnymi typami danych MATLAB-a, takimi jak: macierze pojedynczej i podwójnej precyzji, tablice komórkowe, struktury oraz macierze rzadkie. Użytkownik może łatwo użyć dowolnego typu danych. Z implementacją tablic rozproszonych jest także dostarczany interfejs użytkownika, który jest bardzo podobny do interfejsu pMATLAB z MIT Lincoln Laboratory.

Tablice rozproszone są zaimplementowane jako obiekty MATLAB-a w ten sposób, że każdy '*lab*' przechowuje fragment tablicy. Każdy fragment takiej tablicy, oprócz danych przechowuje też informacje o typach dystrybucji danych, lokalnych wskaźnikach, globalnych rozmiarach tablic, blokadach, liczbie procesów *worker* i inne. Tablice rozproszone obsługują dwa rodzaje dystrybucji danych — dystrybucję jednowymiarową i dwuwymiarowy blok cyklicznych dystrybucji. Użytkownicy mają dostęp do różnych parametrów po określeniu dystrybucji danych dla ich rozproszonych tablic.

W tablicach rozproszonych MATLAB-a użytkownicy mogą zmienić dystrybucje i wykonać ponowną redystrybucję danych. Użytkownicy mogą nawet tworzyć tzw. dystrybucje dynamiczne, na przykład odczytując dowolne fragmenty danych z pliku i łącząc indywidualne fragmenty tak, aby utworzyć rozproszone tablice. Zmianę dystrybucji danych tak że można uzyskać poprzez użycie niektórych operacji i funkcji matematycznych.

W tablicach rozproszonych MATLAB-a, postawiono na programowalność, tj. łatwość wprowadzania zmian w oprogramowaniu kosztem przejrzystości wykonywanych procesów. Nie ma różnic syntaktycznych w odwoływaniu się do elementów regularnych tablic MATLAB-a i jego tablic rozproszonych. Zatem, jeśli w przykładzie prezentowanym powyżej `dA` jest tablicą rozproszoną, to `dA(4,3)` oznacza dostęp użytkownika do elementu `(4,3)` całej rozproszonej tablicy `dA`, a nie tylko do lokalnego fragmentu `dA` z właściwego *worker*'a. Takie podejście może być niekiedy karane kosztami komunikacji.

Tablice rozproszone mogą być używane z prawie wszystkimi 150 zrównoleglonymi wbudowanymi funkcjami MATLAB-a, które obejmują redukcję operacji, indeksowanie i operacje algebry liniowej takie jak rozkład LU. Dla ścisłych operacji algebry liniowej jest stosowana biblioteka ScaLAPACK, zawsze, gdy tylko jest potrzebna. Inne algorytmy, jak te dla macierzy rzadkich, są zaimplementowane w języku MATLAB.

MATLAB używa kilku procedur biblioteki ScaLAPACK dla zrównoleglonych funkcji algebry liniowej, które działają na tablicach rozproszonych. Funkcje tablic rozproszonych MATLAB-a uzyskują dostęp do procedur z ScaLAPACK poprzez kilka prostych procedur obsługi bramek (lub MEX-funkcje wywoływane z MATLAB-a).

Bezpośrednie użycie biblioteki ScaLAPACK stwarza dwa problemy. Pierwszy z nich to składnia wywołań, która dla ScaLAPACK pochodzi z języka FORTRAN i częściowo z C-MPI. Kolejnym problemem jest konieczność określenia optymalnej dystrybucji danych i wybór siatek procesorów dla konkretnych problemów (zadań). ScaLAPACK oczekuje od użytkowników zaprezentowania danych we właściwy sposób. Oczekuje też wskazań jak powinny być zorganizowane obliczenia.

Oznacza to, że ScaLAPACK musi wiedzieć, jaka jest sieć dostępnych procesorów. Zazwyczaj jest to niezbyt łatwe zadanie dla przeciętnego użytkownika. Dlatego dokonano takiego wyboru arbitralnie. Sprawdzono, że aktualna dystrybucja danych potwierdza prawidłowość tego wyboru siatki procesorów.

3.3. Pętla iteracyjna w tablicach rozproszonych *FOR-DRANGE*

Konstrukcja *for-drange* realizuje iteracje pętli *for* w formacie danych rozproszonych (ang. *distributed range* (*DRANGE*)) [5, 7]. Każdy *worker* działa na fragmencie zakresu, który mu przydzielono. Kolejność iteracji jest zawsze taka sama i uwzględniając pewną liczbę *worker*'ów oraz zakres iteracji, podział tego zakresu pozostaje taki sam przez wiele uruchomień.(wywołań). Składnia pętli *for-drange* jest następująca:

```
for variable = drange(colonop)
% loop body (zestaw instrukcji)
end
```

Wyrażenie `colonop` przyjmuje postać: `start:increment:finish` lub `start:finish`. Domyślna wartość kroku iteracji (ang. *increment*) jest równa 1. Wyrażenie `colonop` jest partycjonowane wewnątrz sąsiadujących segmentów `numlabs` na prawie równe długości (`numlabs` zwraca całkowitą liczbę '*labs*' aktualnie działających, na aktualnym '*job*'). Każdy segment staje się iteratorem dla konwencjonalnej pętli *for*, dla poszczególnych '*lab*'.

Konstrukcja *for-drange* wymaga, aby pętle iteracyjne były niezależne od siebie oraz aby nie wymieniały danych między '*labs*', gdy pętla jest wykonywana. Te dwa wymagania oznaczają, że w pętli *for-drange* można uzyskać dostęp tylko do części tablicy rozproszonej, która jest lokalna dla każdego *'labs'*, tzn. indeks elementu w tablicy może uzyskać dostęp tylko do lokalnego fragmentu tablicy rozproszonej.

3.4. Zrównoleglona pętla iteracyjna *PARFOR*

Pętla *parfor* jest zrównoleglonąą instrukcją iteracyjną i różni się znacząco od pętli *for-drange* [5, 7]. Jeśli są udostępnione dodatkowe zasoby obliczeniowe, to podstawowy silnik procesów obliczeniowych może użyć zrównoleglonego kodu dla szybszego uzyskania wyników. Jeśli te zasoby obliczeniowe nie są dostępne, to na komputerze z jednym procesorem pętla *parfor* działa jak tradycyjna pętla *for*. Istotnym wymaganiem dla poprawnego działania pętli jest, aby wykonywane iteracje były całkowicie od siebie niezależne.

Należy użyć funkcji `matlabpool`, aby zarezerwować liczbę *worker*'ów MATLAB-a do wykonania późniejszych pętli *parfor*. Zależnie od algorytmu szeregowania (ang. *scheduler*), *worker*'y mogą być uruchamiane zdalnie na klastrze lub mogą być uruchomione lokalnie na komputerze, na którym zainstalowano MATLAB−klient.

Część zestawu instrukcji (ang. *loop body*) jest wykonywana przez klienta MATLAB-a (gdzie *parfor* jest przydzielona), a część jest wykonywana równolegle, na *worker*'ach MATLAB-a. Potrzebne dane, na których *parfor* pracuje, są wysyłane od klienta do *worker*'ów, gdzie jest wykonywana większość tych obliczeń. Wyniki obliczeń są przesyłane z powrotem do klienta i składane razem. Składnia pętli *parfor* jest następująca:

```
parfor (itr = m : n, [NumWorkers])
% loop body (zestaw instrukcji)
end
```

Argument `NumWorkers` wskazuje górne ograniczenie liczby *worker*'ów, MATLAB-a, które użytkownik chce zastosować przy wykonywaniu zestawu instrukcji w pętli.

Przykłady zastosowania funkcji `matlabpool` i pętli *parfor* dla algorytmów rozwiązywania równań różniczkowych zwyczajnych w MATLAB-ie zamieszczono w [4]. Opisane tam obliczenia równoległe wykonano na komputerze z procesorem wielordzeniowym. Przykłady użycia pętli *parfor* w obliczeniach macierzowych opisano w [2].

3.5. Zastosowania zrównoleglonych pętli *PARFOR i FOR-DRANGE*

Zarówno *parfor* jak i *for-drange* realizują zrównoleglone pętle, które funkcjonalnie różnią się od siebie znacząco. Największa różnica to taka, że *parfor* nie wymaga żadnych wcześniejszych informacji dotyczących obliczeń równoległych do ich wykonania w pozornie zwykłym programie MATLAB-a. Istnieje pojedyncza przestrzeń robocza przed i po wykonaniu pętli *parfor*, ponadto brak kontekstów równoległości.

Konstrukcja *for-drange* operuje wewnątrz kontekstów równoległości, które wymagają od użytkownika pewnej wiedzy o środowisku obliczeń równoległych. Wymagane jest zrozumienie, że przestrzenie robocze zwielokrotniają się (dla każdego '*lab*') i że nie ma ukrytego przekazywania zmiennych między tymi przestrzeniami roboczymi. Ponadto istnieje statyczna zależność 1:1 między '*labs*' i wykonanymi tam iteracjami.

Konstrukcja *for-drange* jest przydatna w pewnych kontekstach wykonywania obliczeń równoległych. Natomiast konstrukcja *parfor* jest zalecana wszędzie tam, gdzie jedynym powodem jest przyspieszenie wykonywania pętli iteracyjnych, z wykorzystaniem dodatkowych zasobów obliczeniowych (np. procesorów wielordzeniowych).

4. UWAGI KOŃCOWE

Pakiet MATLAB jest do śćpowszechnie stosowany do projektowania, optymalizacji oraz testowania układów automatycznej regulacji, w tym do dużych i skomplikowanych obiektów [1, 6]. W najnowszej wersji (R2010b), MATLAB udostępnia użytkownikom możliwości wykonywania obliczeń równoległych, o bardzo zróżnicowanym zakresie ich stosowania. Poszerza to znacząco zakresy aplikacji jego bibliotek *toolbox* i *blockset* w rozwiązywaniu technicznych problemów automatyzacji i robotyzacji.

W [6] opisano przykład zrównoleglenia procesu doboru parametrów układu sterowania dla konstrukcji lotniczych, z wykorzystaniem *Parallel Computing Toolbox* i *Simulink Design Optimization*. W [1] opisano użycie techniki obliczeń równoległych do analizy układu elektromechanicznego uruchamianego przez urządzenie mikroelektromechaniczne, z wykorzystaniem metody elementów skończonych.

BIBLIOGRAFIA

1. Hosagrahara V., Tamminana K., Sharma G., *Accelerating Finite Element Analysis in MATLAB with Parallel Computing*, The MathWorks Newsletters-MATLAB Digest, November 2010, Vol. 19, Nr 6.
2. Łuszczek P., *Enhancing Multicore System Performance Using Parallel Computing with MATLAB*, The MathWorks Newletters-MATLAB Digest, September 2008, Vol. 17, Nr 5
3. Moler C., *Parallel MATLAB: Multiple Processors and Multiple Cores*, The MathWorks News&Notes, June 2007.
4. Mrozek B., Mrozek Z. *MATLAB i Simulink. Poradnik użytkownika*, Wyd. III, Helion, 2010.
5. Sharma G., Martin J., *MATLAB: A Language for Parallel Computing*, International Journal of Parallel Programming (2009), Vol. 37, Nr 1, pp. 3−36.
6. Stothert A., Turevskiy A., *Improving Simulink Design Optimization Performance Using Parallel Computing*, The MathWorks Newsletters-MATLAB Digest, May 2009, Vol 8, Nr 3.
7. *Parallel Computing Toolbox User's Guide*, *Version* 5.0 (R2010b), The MathWorks, Inc.
8. *MATLAB Distributed Computing Server System Administrator's Guide, Version* 5.0 (R2010b), The MathWorks, Inc.
9. *MATLAB User Guides*, *Version 7.11* (R2010b), The MathWorks, Inc.