Journal of Theoretical and Applied Mechanics, Sofia, 2013, vol. 43, No. 2, pp. 77–82

OPTIMIZATION OF THE CODE OF THE NUMERICAL MAGNETOSHEATH-MAGNETOSPHERE MODEL

P. Dobreva
Institute of Mechanics, Bulgarian Academy of Sciences,
Acad. G. Bonchev St., Bl. 4, 1113 Sofia, Bulgaria,
e-mail: polya2006@yahoo.com

[Received 29 November 2012. Accepted 03 June 2013]

Abstract. The proposed three dimensional model contains two earlier developed 3D regional numerical models: a grid-characteristic model of the magnetosheath and a finite element model of the magnetosphere. The model output is the distribution of gas-dynamic parameters in the magnetosheath and of magnetic field inside the magnetosphere. The efforts are focused on the modernization of the existing software, written in Fortran, using several techniques for parallel programming such as OpenMP extensions. After analyzing the numerical performance of the model a possible scenario for the code optimization is shown. First results with the improved variant of the model are presented.
Key words: Magnetosheath, magnetosphere, parallel algorithms.

1. Introduction

The multi-core processor technologies made it possible to significantly reduce the computational time in numerical calculations. Here, the efforts are focused on adaptation of the existing magnetosheath-magnetosphere model, so that it can be used in space weather forecasting. The model describes the Earth's environment, including both the magnetosphere and the magnetosheath areas, but using the simpler gas-dynamic, instead of magneto-hydrodynamic (MHD) approach. The magnetospheric part is based on finite element method (FEM), which is the main-time consuming part of the model. Nowadays, the possibilities of parallel computation are widely applied in FEM problems. Parallel computation of microwave fields in 3D domain on Cluster of workstations

[*]The work is supported by Bulgarian National Council "Scientific Research" under Project DMU-03/88.

is performed by [7]. In calculation of waves – [5] significant reduction of computational time was achieved. These are just two of many examples, showing that parallel algorithms can be applied in FEM problems with a great success.

The goal in the present work is to improve the performance of model and to make it usable for Space weather predictions in near real time. In order to do this, it is necessary first to optimize the code of the program, so that it could be implemented in multicore computer platforms. Here, the first step in the improvement of the code is presented, consisting of optimizing of one of the main time-consuming routines.

2. Brief description of the model

The model magnetosheath-magnetosphere is created in the Institute of Mechanics, Bulgarian Academy of Sciences [4]. In the area of magnetosheath, the Euler's equations of a compressible ideal gas are used for a description of the shocked solar wind. At every point of the boundaries, the Rankine-Hugoniot relationships need to be satisfied, that are reduced to the condition of pressure balance at the magnetopause points. The shapes and locations of the two boundary surfaces – bow shock and magnetopause, are also determined as a part of the solution.

In the area of magnetosphere, the model describes the magnetic field distribution. The Chapman-Ferraro currents, that are the subject of determination, are calculated by FEM. The total magnetic field is a sum of the field of Chapman-Ferraro currents, the dipole field and the fields of: ring, Birkeland and tail currents. The contribution of the "extraterrestrial sources" is calculated using one of the variants of the Tsyganenko model – T01 [6]. Input parameters are some solar wind and geomagnetic characteristics. In comparison of model results with experimental data – [2], [4] a good resemblance was received.

3. Analysis of the model implementation

The code of the model, performing the calculations, is written in Fortran programming language. The entire package contains about 17 180 operators (lines), separated into 302 procedure subprograms and 47 functions. To determine which part of the program makes sense to be optimized, it is necessary first to analyze the implementation of the numerical algorithm.

At the beginning of execution, some appropriate initial forms and initial distribution of the parameters are chosen. Then, an iterative algorithm starts, which includes consecutive execution of the two basic programs: *magnetosheath* and *magnetosphere*. The *magnetosheath* subroutine calculates the distribution of the gasdynamic parameters at the magnetosheath points and determines the new position of the boundaries. The *magnetosphere* program computes the magnetospheric field and hence the magnetic pressure at the magnetopause. The *magnetosheath* and *magnetosphere* programs are invoked consecutively until the boundary conditions are satisfied. It usually takes about 3000-5000 iterations to reach convergence. The *magnetosheath* subroutine takes about 2% of the total central processing unit (CPU) time. The structure of *magnetosphere* subprogram will be considered in details, because it is the main-time consuming part, taking 98% of the total CPU time. It is known, that FEM requires significant computational resources, so this CPU time distribution is an expectable result.

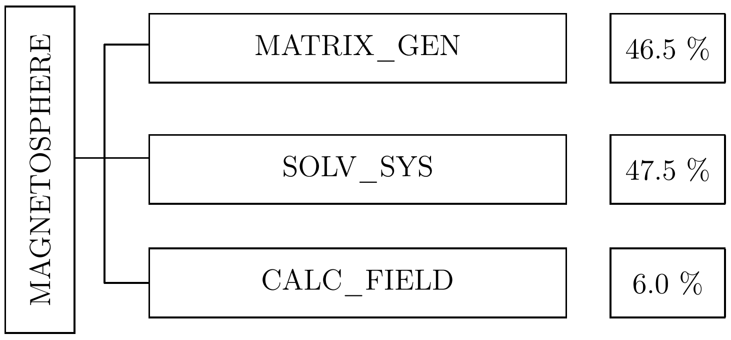

Fig. 1. Distribution of computational times between the major subroutines of module *magnetosphere*.

A diagram of execution of the *magnetosphere* subprogram is presented in Fig. 1. The times, needed for execution of every subprogram are given on the right. The *matrix_gen* subprogram sets the new boundary conditions, corresponding to the new position of the magnetopause, and generates the global matrix of the linear system. The *solv_ sys* subroutine solves the system of linear algebraic equations. The *calc_field* program calculates the magnetic field at a given arbitrary point, and in particular at the magnetosheath points, that coincide with the magnetopause. It is decided the first program, implemented in parallel, to be *matrix_gen* upon the analysis of the computational times.

4. Numerical results

The test results are performed on a computer architecture of Workstation type, equipped with Intel Xeon 5620 Quad-core processor. The station works at a clock frequency of 2.4 GHz. Having Hyper-threading enhancement, the Xeon Quad-core processor allows eight threads to run simultaneously. The total memory resource is 12 Gb; every core has 12Mb L1 memory cash. The development environment is Intel Visual Fortran Composer XE 2011, including OpenMp libraries that support shared-memory parallel programming [3]. The main problem in the case of multi-thread architectures is that memory bugs are hard to be differentiated and classified. To avoid main types of memory conflicts, such as Memory and Threading errors, Intel Inspector XE 2013 is used (also known as Memory Checker).

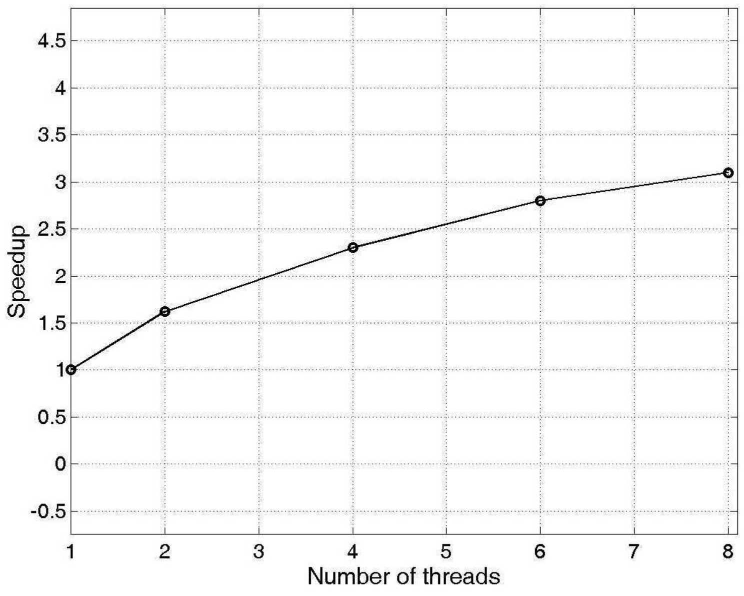

Fig. 2. An estimate of acceleration of the *matrix_gen* program.

The code of entire package was written for a sequential implementation and was running an a Desktop machine. First, the code was translated into the Intel programming environment [3] and was working in sequential mode on the Xeon Server. Then, one of the mainly used ways of parallelization, consisting in putting the OpenMP directives, is applied to the *matrix_gen* program. It is a large program, written for a sequential machine and containing 4-level nested loop. Parallelization, applied to each of this levels of that nested loop leads to bad performance. The efficiency could not be achieved without changing the existing code of the program. In order to run in parallel, the code was rewritten, as the nested loops were replaced by a single one. Then the OpenMP directives [1] were put to that loop. Every thread executes 1/p (p=8) part of the same loop. All the threads have access to the global shared memory.

Speedup is a characteristic of the acceleration, determined as the ratio T_1 / T_p, where T_1 is the time for execution on one processor, and T_p is the time on p processors (threads). Here, maximum number of threads is 8. Figure 2 shows the measured speedup of *matrix_gen* subroutine for a given number of threads. The speedup of *matrix_gen* of about 3x leads to an acceleration of 1.45x for the whole model. The overall time, needed for the execution of 1500 iterations, is now reduced to about 40–60 min., depending on the input conditions.

5. Conclusion

The first partial parallelization gives a promising result in the process of adaptation of the model for space weather purposes. The next step to be performed is to reduce the computational time to solve the system linear algebraic equations (*solv_sys* program). The overall performance can be further improved by the use of Message Passing Interface (MPI) directives on a Cluster with distributed memory resources.

REFERENCES

[1] Chapman, B., G. Jost, R. van der Pas. Using OpenMP: Portable Shared Memory Parallel Programming, England, London, The MIT Press, 2007.

[2] Dobreva, P. S., M. D. Kartalev, N. N. Shevyrev, G. N. Zastenker. Compassion of a New Magnetosphere-magnetosheath Model with Interball-1 Magnetosheath Plasma Measurements. *Planet. Space Sci.*, **53** (2005), 117–125.

[3] Intel Programmer's Reference, USA, Intel Corporation, 2003.

[4] Kartalev, M., P. Dobreva, E. Amata, M. Dryer, S. Savin. Some Tests of a New Magnetosheath Model via Comparison with Satellite Measurement. *J. Atmos. Sol. Terr. Phys.*, **70** (2008), 627–636.

[5] Kerepova, E. D., V. V. Shaidurov. Parallel Implementation of FEM for the Initial Bounday Value Problem. *Comp. Tech.,* **14** (2009), No. 6, 45–57.

[6] Tsyganenko, N. A. A Model of the Near Magnetosphere with a Dawn-dusk Asymmetry 1. Mathematical Structure. *J. Geophys. Res.*, **107** (2002), 1179–1193.

[7] Vollaire, C., L. Nicolas, A. Nicolas. Parallel Computing for the Finite Element Method. *Eur. Phys. J. Appl. Phys.,* **1** (1998), No. 3, 305–314.