PARALLEL APPROACH TO THE SOLUTION OF STATIONARY REACTION-DIFFUSION PROBLEM

Daniela Bímová

Technical University of Liberec
Faculty of Science, Humanities and Education
Department of Mathematics and Didactics of Mathematics
Studentská 2, 461 17, Liberec, Czech Republic
daniela.bimova@tul.cz

Abstract

The paper is devoted to the parallel solution of the two-dimensional stationary reaction-diffusion problem. By the usage of parallel approach to the linear algebra representation we create the parallel algorithm for computing a numerical solution of the two-dimensional stationary reaction-diffusion problem. We compare calculation times of computing the approximate solution of the system of (linear) difference equations for different sizes of the system matrix by the numerical conjugate gradient method on 1, 2, 3, and 4 processors, respectively.

Keywords: Stationary reaction – diffusion problem, parallel linear algebra, finite difference method, conjugate gradient method.

Introduction

Our model problem is represented by the stationary reaction-diffusion equation, which discretization via the finite difference method leads to the system of linear algebraic equations.

We choose the numerical conjugate gradient method for solving the system of linear algebraic equations in this paper and we try to apply this method in parallel algorithm implemented in Fortran. The aim of this paper is to find how we save the calculating time computing the system of linear algebraic equations on 2, 3, or 4 processors instead of on one processor. We perform our calculation in numerical experiment in which we compute the vector of the approximate solution (for different mesh sizes of the grid) of the system of linear difference equations by the conjugate gradient method.

We have discussed the analogous problem using the method of steepest descent in [1].

1. Setting of Stationary Reaction-Diffusion Problem

We consider the two-dimensional boundary value problem in this chapter. It is the analogy of a one-dimensional boundary value problem mentioned in [2].
The task is to find a function u: $\Omega \to \mathbf{R}$ fulfilling the stationary reaction-diffusion equation

$$-\varepsilon\,\Delta u + k \cdot u = f \text{ in } \Omega = \langle a, b\rangle \times \langle c, d\rangle,\ a, b, c, d \in \mathbf{R},\ a < b,\ c < d,\ \varepsilon > 0,\ k \in \mathbf{R} \quad (1)$$

$$u\,\big|_{\delta\Omega} = g \text{ on } \delta\Omega, \quad (2)$$

where f: $\Omega \to \mathbf{R}$ is the given function and where g: $\delta\Omega \to \mathbf{R}$ represents a Dirichlet boundary condition.

For $m, n \in \mathbf{N}$ we bring up $(m+1)\times(n+1)$ of evenly spaced points $X_{i,j} = \lfloor x_i, y_j \rfloor = [a + i\cdot h, c + j \cdot h]$, where $i = 0, 1, \dots, m$, $j = 0, 1, \dots, n$ and where h is the spatial step. We denote $U_{i,j}$ the approximate solution at the points $X_{i,j}$, i. e. $U_{i,j} \approx u(x_i, y_j) \approx u(X_{i,j})$, and we put $F_{i,j} = f(x_i, y_j) = f(X_{i,j})$.

For every regular knot $X_{i,j}$ we use three-point stencil for approximation the second derivatives, i.e.

$$\frac{\delta^2 u}{\delta x^2}(x_i, y_j) \approx \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2}, \quad \frac{\delta^2 u}{\delta y^2}(x_i, y_j) \approx \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h^2}.$$

Applying this to (1) we obtain the difference equations for the regular knots $X_{i,j} = \lfloor x_i, y_j \rfloor$, it means we have the equations of the form

$$-U_{i-1,j} - U_{i,j-1} + \left(4 + k \cdot h^2\right)\cdot U_{i,j} - U_{i+1,j} - U_{i,j+1} = \frac{h^2}{\varepsilon} F_{i,j}, \qquad i = 1, 2, \dots, m-1, \quad j = 1, 2, \dots, n-1, \quad k \in \mathbf{R}.$$

This is the standard five-point scheme. The approximate values of the sought function u in knots of the given grid are represented by the numerical solution of the system of (linear) algebraic equations

$$\mathbf{AU} = \mathbf{F} \tag{3}$$

with an unknown vector

$$U = \left(U_{1,1}, \dots, U_{1,n-1}, U_{2,1}, \dots, U_{2,n-1}, \dots, U_{m-1,1}, \dots, U_{m-1,n-1}\right)^{\mathbf{T}} \in \mathbf{R}^{(m-1)\times(n-1)}$$

and a matrix **A**, which we can write in the following way

$$\mathbf{A} = \begin{pmatrix} \mathbf{L} & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ -\mathbf{I} & \mathbf{L} & \ddots & \ddots & \vdots \\ \mathbf{0} & \ddots & \ddots & \ddots & \mathbf{0} \\ \vdots & \ddots & \ddots & \mathbf{L} & -\mathbf{I} \\ \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{I} & \mathbf{L} \end{pmatrix},$$

where

$$\mathbf{L} = \begin{pmatrix} 4 + k\cdot h^2 & -1 & 0 & \cdots & 0 \\ -1 & 4 + k \cdot h^2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 4 + k\cdot h^2 & -1 \\ 0 & \cdots & 0 & -1 & 4 + k \cdot h^2 \end{pmatrix},$$

I is the identity matrix, and **0** is the zero matrix. As well as the vector of the right side of the mentioned system of (linear) difference equations we can write as follows

$$\mathbf{F} = \begin{pmatrix} \frac{h^2}{\varepsilon}F_{1,1} + U_{0,1} + U_{1,0} \\ \frac{h^2}{\varepsilon}F_{1,2} + U_{0,2} \\ \vdots \\ \frac{h^2}{\varepsilon}F_{1,n-1} + U_{0,n-1} + U_{1,n} \\ \frac{h^2}{\varepsilon}F_{2,1} + U_{2,0} \\ \frac{h^2}{\varepsilon}F_{2,2} \\ \vdots \\ \frac{h^2}{\varepsilon}F_{2,n-1} + U_{2,n} \\ \\ \vdots \\ \\ \frac{h^2}{\varepsilon}F_{m-1,1} + U_{m,1} + U_{m-1,0} \\ \frac{h^2}{\varepsilon}F_{m-1,2} + U_{m,2} \\ \vdots \\ \frac{h^2}{\varepsilon}F_{m-1,n-1} + U_{m,n-1} + U_{m-1,n} \end{pmatrix}.$$

According to (2) we consider the following Dirichlet boundary conditions:

$$U_{i,0} = g\left(X_{i,0}\right),\ U_{i,n} = g\left(X_{i,n}\right) \quad \forall\, i = 1, 2, \ldots, m-1,$$
$$U_{0,j} = g\left(X_{0,j}\right),\ U_{m,j} = g\left(X_{m,j}\right) \quad \forall\, j = 1, 2, \ldots, n-1.$$

2. Parallel approach to the solution of the system of linear algebraic equations

2.1. Matrix mapping in parallel linear algebra

The way, in which the matrix is mapped on the net of processors, determines efficiency and elegance of an algorithm in most the cases. We have chosen the striped mapping cyclically by rows of a matrix for our stationary reaction-diffusion problem.

2.2. The striped mapping of the matrix cyclically by rows

In this paragraph we describe how we map a matrix using MPI in parallel algorithms.

Let us assume that there is given the matrix **A** of the mentioned shape and of the type (n, n). In the parallel algorithm we work generally on p processors. The first task is to map the matrix **A** onto the particular processors. The matrix **A** will be mapped by the striped mapping cyclically by rows onto the particular processors. The striped mapping assumes that the processors are connected into the linear virtual array and that they are numbered 0, 1, 2, …, p – 1, where p is the number of used processors, in general. The matrix **A** (denoted A_global) is generated and known only by so called master processor. The master processor distributes data (the row vectors of the matrix **A)** into the particular processors. We obtain new matrices (denoted A_local) of the type ($\lceil n/p \rceil$, n), where n is the number of rows of original matrix and where p is the number of used processors, by the data distribution. Matrices "A_local" consist of the appropriate rows of the matrix **A**.

The analogous situation is true for mapping the vector **F** (of the right side of the system of linear algebraic equations). There is one small difference, only one element (not the whole row) is sent in every step of the distribution.

2.3. Product of a matrix and of a vector in parallel algorithm

We use the iterative methods for solving the system of linear algebraic equations. Part of every iterative method is the product of a matrix and of a vector. That is why we describe product of a matrix **A** and of a vector **U** from the matrix equation

$$\mathbf{AU} = \mathbf{F}$$

using MPI in parallel algorithms in this paragraph.

Every processor knows only a part of the vector **U**, with whom we want to multiply the matrix **A**, on the basis of the previous mapping of a vector onto the particular processors. (We assume that the given vector **U** is mapped in the same way in which the vector **F** was mapped). That is the reason why every processor has to find the rest part of the vector **U**, which it does not know. Every processor can call the command “MPI_ALLGATHER” by which it can find the missing information about the rest parts of the vector **U**.

If all the processors know all the elements of the vector **U**, we can perform dot product and calculate the appropriate element of the result vector.

2.4. Scalar product of two vectors in parallel algorithm

Except of the product of a matrix and of a vector, the scalar product of two vectors occurs in the iterative methods, too. In this paragraph we describe process of computing a scalar product of two vectors in parallel algorithms.

Every processor knows only a part of the vector **U** as well as of the vector **V** (which we want to multiply scalarly) on the basis of the previous mapping (the striped mapping cyclically by rows) of a vector onto the particular processors. That is why we firstly multiply scalarly the appropriate elements of the vectors **U**, **V**. Secondly, we call the command “MPI_ALLREDUCE” that sums over all processors and distributes the resulting sum onto all the processors. All the processors finally know value of the scalar product of the vectors **U**, **V**.

3. Numerical experiment of stationary reaction-diffusion problem

We consider the two-dimensional stationary reaction-diffusion problem

$$-\varepsilon\,\Delta u + k \cdot u = f \ \text{ in } \Omega = [0,1]^2, \tag{4}$$

$$u\,\big|_{\delta\Omega} = 0 \ \text{ on } \ \Omega. \tag{5}$$

We set $\varepsilon = 1, k = 1$ and cover the domain Ω by the grid of knots with the spatial step h in the direction of both the coordinate axes x and y. We choose function f in the problem (4) so that the function

$$u(x, y) = 64 \cdot x \cdot y \cdot (1-x) \cdot (1-y) \cdot (y-x) \tag{6}$$

is the exact solution.

The approximate values of the solution u in all the inner regular knots of the given grid are the numerical solution of the problem (4) – (5). We find the numerical solution by the conjugate gradient method with prescribed tolerance 10^{-5} for the norm of residue.

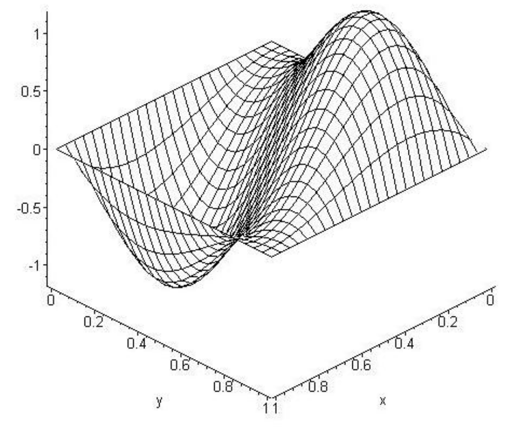
Table 1 illustrates the development of calculation times according to the number of used processors with respect to spatial step.

Tab. 1. *Calculation times needed for computing the vector* **U** *of the approximate solution for different mesh sizes of the grid on one, two, three, and four processors*

Step	Number of knots	Number of iterations	Condition number	1 processor	2 processors	3 processors	4 processors
h = 1/25	576	56	260	**1** s	**1** s	**1** s	**1** s
h = 1/49	2 304	110	963	**15** s	**10** s	**7** s	**5** s
h = 1/73	5 184	163	2 112	**2** min **35** s	**1** min **13** s	**49** s	**35** s
h = 1/97	9 216	217	3 704	**13** min **01** s	**5** min **50** s	**3** min **45** s	**2** min **57** s

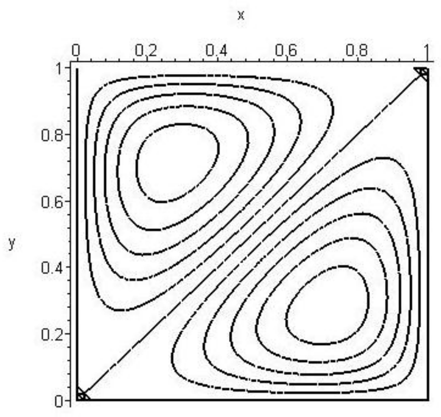
Source: Own based on computations on the cluster

There is shown the graph of the exact solution of the problem (4) – (5) in Fig. 1a and there are drawn the contours of the exact solution of the problem (4) – (5) in Fig. 1b.



Source: Own based on exact solution of (6)

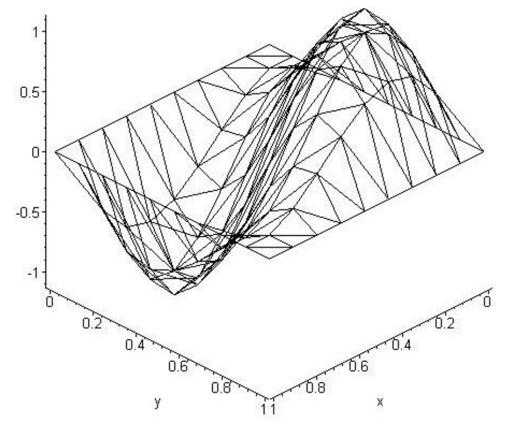
Fig. 1a. *Exact solution*



Source: Own based on exact solution of (6)

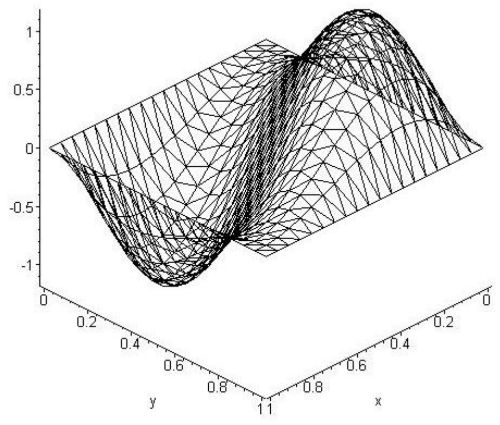
Fig. 1b. *Contours of the exact solution*

Furthermore, there are drawn the graphs of the approximate solutions of the problem (4) – (5) for the different mesh sizes in Fig 2a – 2d. The graph of the approximate solution for the mesh size $h = \frac{1}{9}$ is sketched in Fig. 2a, for the mesh size $h = \frac{1}{21}$ in Fig. 2b, for the mesh size $h = \frac{1}{41}$ in Fig. 2c, and finally for the mesh size $h = \frac{1}{61}$ in Fig. 2d.



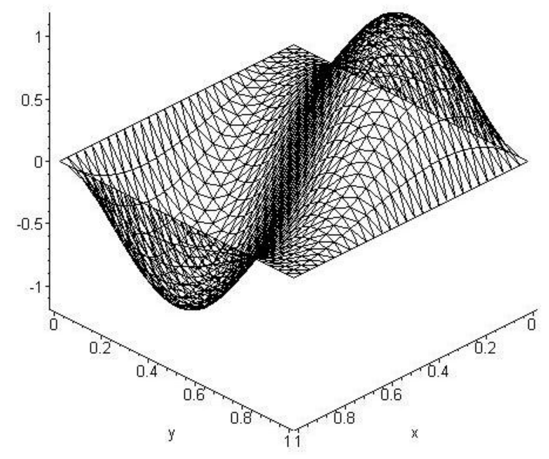
Source: Own based on numerical experiment

Fig. 2a. *Approximate solution, mesh size* $h = \frac{1}{9}$



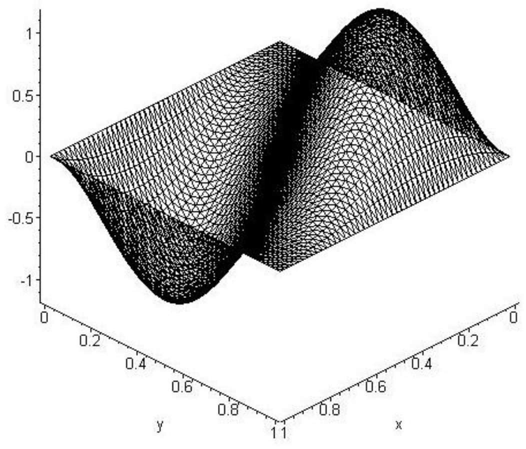
Source: Own based on numerical experiment

Fig. 2b. *Approximate solution, mesh size* $h = \frac{1}{21}$



Source: Own based on numerical experiment

Fig. 2c. *Approximate solution, mesh size* $h = \frac{1}{41}$



Source: Own based on numerical experiment

Fig. 2d. *Approximate solution, mesh size* $h = \frac{1}{61}$

Conclusion

We discussed the two-dimensional stationary reaction-diffusion problem. We described the striped mapping of a matrix cyclically by rows in parallel programming. We showed the possibilities of applying the basic principles of linear algebra – product of a matrix and of a vector, scalar product of two vectors – in parallel algorithm. Finally, we commented the numerical experiment in which the mentioned theory is applied in practice.

We can summarize on the basis of the numerical experiment results that computing the approximate solution of the system of linear difference equations is approximately two times (three times, resp. four times) quicker, if we calculate the system of equations on two (three, resp. four) processors instead of on one processor. Mentioned results are best seen from the calculation times for bigger mesh sizes of the grid. The results are written in the Tab. 1.

Acknowledgements

The paper was supported by the project ESF, no. CZ.1.07/2.3.00/09.0155, “Constitution and improvement of a team for demanding technical computations on parallel computers at TU Liberec”.

Literature

[1] BÍMOVÁ, D.: *Parallel solution of Poisson Equation.* In: Proceedings of Aplimat 2012. Bratislava, Slovak University of Technology 2012, pp. 233-243. ISBN 978-80-89313-57-0.

[2] DOLEJŠÍ, V.; KNOBLOCH, P.; KUČERA, V.; VLASÁK, M.: *Finite element methods: theory, applications and implementation*. TUL, Liberec 2011. ISBN 978-80-7372-728-4.

Mgr. Daniela Bímová, Ph.D.

PARALELNÍ PŘÍSTUP K ŘEŠENÍ STACIONÁRNÍHO REAKČNĚ-DIFÚZNÍHO PROBLÉMU

Článek je věnován paralelnímu řešení dvoudimenzionálního stacionárního reakčně-difúzního problému. Pomocí paralelního přístupu k reprezentaci lineární algebry vytvoříme paralelní algoritmus pro výpočet numerického řešení dvoudimenzionálního stacionárního reakčně-difúzního problému. Porovnáme časy potřebné k výpočtu přibližného řešení systému (lineárních) diferenciálních rovnic pro různě velkou matici soustavy numerickou metodou sdružených gradientů na 1, 2, 3 a 4 procesorech.

PARALLERER ANSATZ ZUR LÖSUNG EINES STATIONÄREN REAKTIONSDIFFUSEN PROBLEMS

Dieser Artikel beschäftigt sich mit der parallelen Lösung eines zweidimensionalen stationären reaktionsdiffusen Problems. Mit Hilfe eines parallelen Ansatzes zur Repräsentation linearer Algebra bilden wir einen parallelen Algorithmus für die Berechnung einer numerischen Lösung eines zweidimensionalen stationären reaktionsdiffusen Problems. Wir vergleichen die Zeiten, die zur Berechnung einer annähernden Lösung des Systems (linearer) differenzialer Gleichungen einer großen Matrix eines Systems durch die numerische Methode dualer Gradienten auf 1, 2, 3 und 4 Prozessoren nötig sind.

PARALELNE PODEJŚCIE DO ROZWIĄZYWANIA STACJONARNEGO PROBLEMU REAKCYJNO-DYFUZYJNEGO

Artykuł poświęcony jest paralelnemu rozwiązywaniu dwuwymiarowego stacjonarnego problemu reakcyjno-dyfuzyjnego. Przy pomocy podejścia paralelnego do reprezentacji algebry liniowej stworzymy algorytm równoległy do wyliczenia numerycznego rozwiązania dwuwymiarowego stacjonarnego problemu reakcyjno-dyfuzyjnego. Porównamy czasy niezbędne do wyliczenia przybliżonego rozwiązania układu równań różniczkowych (liniowych) dla macierzy układu o różnej wielkości przy pomocy metody numerycznej gradientów sprzężonych na 1, 2, 3 i 4 procesorach.