MATEMATIKA, 2015, Volume 31, Number 2, 127–134

Parallel Localized Differential Quadrature Approach in Boundary Value Problem

Hui Ting Cheong, [1]Su Hoe Yeak and Che Lokman Jaafar
Department of Mathematical Sciences, Faculty of Science,
Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia
e-mail: [1]s.h.yeak@utm.my

Abstract In this paper, we need to developed a numerical strategy by implementing FORTRAN programming and OpenMP parallelization in Localized Differential Quadrature method in boundary value problem. Localized Differential Quadrature method is developed from the DQ method which used to improve the accuracy of the results by increasing the number of grid points. In order to reduce the execution time for sequential algorithm in solving boundary value problem using LDQ method, parallel programming of shared memory architecture (OpenMP) is introduced. With the introduction of share memory architecture, the computational time can be further speed up using parallel approach.

Keywords FORTRAN programming; OpenMP parallelization; localized differential quadrature; grid points; boundary value problem

2010 Mathematics Subject Classification 30E25

1 Introduction

The Differential Quadrature (DQ) method was first proposed by Bellman and his associates [1, 2] in early 1970s. This method has been adopted in science and engineering fields. According to C. Shu [3], DQ method is a highest order Finite Difference (FD) method, which is the extension of the low order FD scheme. The low order numerical methods such as FD, Finite Element (FE) and Finite Volume (FV) methods may be relatively easy to use but limited to special cases, for example, need initial trial functions, or require large amounts of computational effort and consequently high cost [4]. In order to overcome these several undesirable limitations, Differential Quadrature (DQ) method has been developed and discussed in this study. The basic idea to DQ method is to determine the weighting coefficients for any order derivatives by using a weighted sum of functional values at a set of selected grid points.

Although DQ method is a highly efficient method by using a small number of grid point, but it is not efficient when the number of grid points is large and it is also sensitive to grid point distribution [5-7]. To overcome the limitations for the applications of DQ method, there are many researchers have given much efforts in developing and improving DQ method use in more useful application. Quan and Chang [8, 9] developed a set of explicit formula to obtain the weighting coefficients for first and second order derivatives using the Lagrange interpolation polynomial for the distributed grid points in solving distributed system equation. Besides, Malik and Civan [10] introduced Differential Cubature (DC) method by replacing quadrature rule with cubature rule can act as a competitive numerical technique in solving multi-dimensional problems. Since the formulation of weighting coefficients cannot be applied directly into an irregular domain, therefore, DC method is an alternative in

treatment of irregular domain. These previous works have shown that the selection of grid distribution has significant influence on the accuracy of DQ results.

In this paper, we propose Localized Differential Quadrature (LDQ) method in solving partial differential equations (PDEs) with proper boundary conditions such as the diffusion equation and wave equation. The diffusion equation is a partial differential equation which describes density fluctuations in a material undergoing diffusion. It is an example of a parabolic equation. Makbule [11] introduced Differential Quadrature (DQ) method for solving time-dependent diffusion equation. Besides, we also discuss the wave equation in this study. The wave equation is a second-order linear partial differential equation. Zong and Lam [12, 13] had introduced Localized Differential Quadrature (LDQ) method by approximating the derivatives of the weighted sum of the points in its neighbourhood for solving wave equation.

In order to reduce the execution time for sequential algorithm in solving boundary value problem using LDQ method, parallel programming of shared memory architecture (OpenMP) is introduced. Parallel programming is a programming in a language that allows you to explicitly indicate how portions of computation may be executed concurrently by different processors [14]. Parallel computing has been considered to be "the high end of computing", and has been used to model difficult problems in many areas of sciences and engineering. OpenMP is an implementation of multithreading, a method of parallelizing whereby a master thread (a series of instructions executed consecutively) forks a specified number of slave threads and a task is divided among them. The threads then run concurrently, with the runtime environment allocating threads to different processors. According to Amdahl's Law [14], the non-parallelized part of program can affect the scalability of the program. Therefore, parallelization need to be done on the part of source code where the majority of the execution time is spent on. In this paper, parallel programming of shared memory architecture is introduced and used to reduce the execution time for sequential algorithm in solving boundary value problem using LDQ method.

2 Numerical Methodology

In this paper, initially the problem is to be solved on a global grid point by using the LDQ method. We generalize the LDQ method in two-dimensions. The two-dimensional domain of interest is discretized by $N \times M$ regular grid points.

First, we consider the distance between any two points is denoted as the following:

$$r_{ik} = |x_i - x_k| \ , \qquad i, k = 1,\ 2,\ \ldots,\ N \tag{1}$$

$$p_{jk} = |y_j - y_k| \ , \qquad j, k = 1,\ 2,\ \ldots,\ M \tag{2}$$

Through comparison, one can find the permutation $s(1), s(2),\ \ldots,\ s(N)$ and $q(1),\ q(2),\ \ldots,\ q(M)$ such that

$$r_{is(1)} \leq r_{is(2)} \leq\ \ldots\ \leq r_{is(N)} \tag{3}$$

$$p_{jq(1)} \leq p_{jq(2)} \leq\ \cdots\ \leq p_{jq(M)}. \tag{4}$$

The number of points in neighbourhood is fixed at m ($m \leq N,\ M$). S_{ij} and Q_{ij} define the

neighbourhood of the grid point of interest. Then the neighbourhood index set as

$$S_{ij} = (s(1),\ s(2),\ \ldots,\ s(m)), \tag{5}$$

$$Q_{ij} = (q(1),\ q(2),\ \ldots,\ q(m)), \quad i = 1,\ 2,\ \ldots,\ N\ , \quad j = 1,\ 2,\ \ldots,\ M \tag{6}$$

where S_{ij} and Q_{ij} define the neighbourhood of the grid point.

The DQ approximation is in this neighbourhood is in the form:

$$u_x(x_i, y_j, t) = \sum_{k \in S_{ij}} a_{ik} u(x_k, y_j, t)\ ; \quad u_{xx}(x_i, y_j, t) = \sum_{k \in S_{ij}} b_{ik} u(x_k, y_j, t) \tag{7}$$

$$u_y(x_i, y_j, t) = \sum_{k \in Q_{ij}} a_{jk} u(x_i, y_k, t)\,; \quad u_{yy}(x_i, y_j t) = \sum_{k \in Q_{ij}} b_{jk} u(x_i, y_k, t) \tag{8}$$

$i = 1,\ 2,\ \ldots,\ N,\ j = 1,\ 2,\ \ldots,\ M$ where the weighting coefficients for the first-order derivative are

$$a_{ij} = \frac{1}{x_j - x_i} \prod_{k \in S_i, k \neq i,j} \frac{x_j - x_k}{x_j - x_i}, \quad j \in S_i,\ \ j \neq i \tag{9}$$

$$a_{ii} = -\sum_{j \in S_i, j \neq i} a_{ij}, \qquad i = 1,\ 2,\ \ldots,\ N. \tag{10}$$

For the second-order derivative, the weighting coefficients are

$$b_{ij} = 2\left(a_{ij} a_{ii} - \frac{a_{ij}}{x_i - x_j}\right), \quad i, j = 1,\ 2,\ \ldots,\ N\ , \qquad i \neq j, \tag{11}$$

$$b_{ii} = -\sum_{j \neq i}^{N} b_{ij}, \quad i = 1,\ 2,\ \ldots,\ N\ . \tag{12}$$

The formulation of weighting coefficients for the first- and second-order with respect to the coordinate y is same with the formulation of weighting coefficients for the first- and second-order with respect to the coordinate x.

In this paper, we consider discretization of the two-dimensional wave equation and diffusion equation as follows:

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \\ \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \end{aligned} \tag{13}$$

From the equation (13), we discretize the value of $u(x, y, t)$ on the grid point (x_i, y_j) as $u_{ij}(t)$, and its velocity is $v_{ij}(t)$.

Then rewrite equation (13) in the following form,

$$\frac{\partial u_{ij}}{\partial t} = v_{ij} \tag{14}$$

$$\frac{\partial v_{ij}}{\partial t} = \sum_{k \in S_{ij}} b_{ik} u(x_k, y_j, t) + \sum_{k \in Q_{ij}} b_{jk} u(x_i, y_k, t). \tag{15}$$

From equations (14) and (15), the matrix form for $u_{ij}(t)$ and $v_{ij}(t)$ can be written as

$$\tilde{w} = \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} = \begin{bmatrix} u_{11} \\ \vdots \\ u_{NM} \\ v_{11} \\ \vdots \\ v_{NM} \end{bmatrix}, \tag{16}$$

$$\frac{\partial}{\partial t}\left[\tilde{w}\right] = \left[B_{(2ij)}\right]\left[\tilde{w}\right]. \tag{17}$$

Runge-Kutta method is used to numerically integrate the equation (17) in time direction.

3 Parallel Computing

In this section, the parallelization of the LDQ method will be executed by OpenMP compiler based on the boundary value problem. Parallelism across space is used in this paper and it is efficient when the systems have a regular structure as shown in Figure 1.

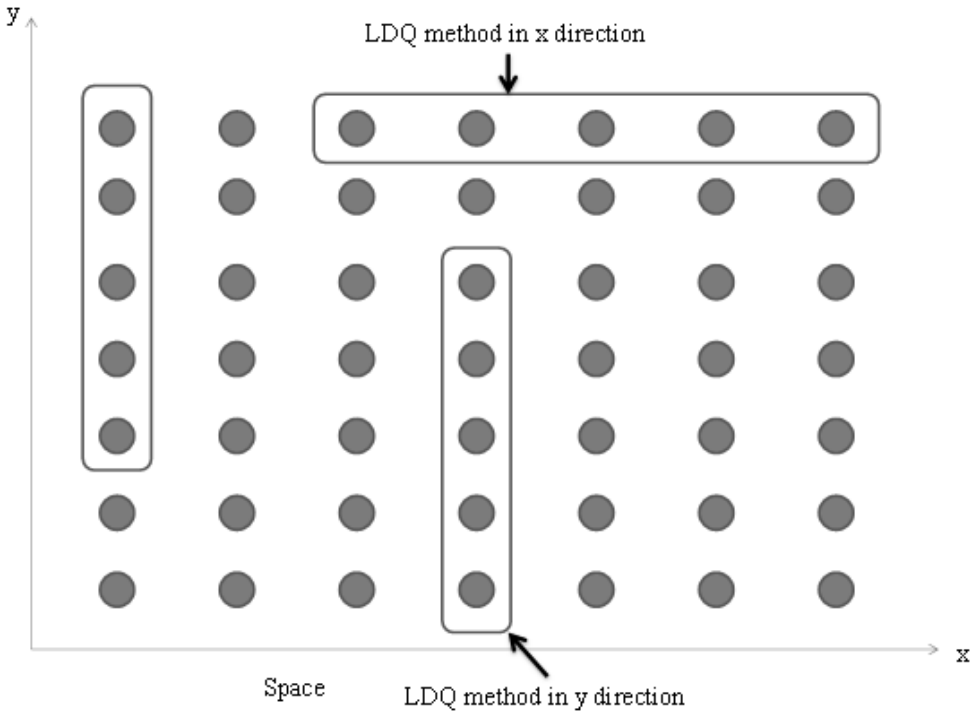

Figure 1: Computation Grid with Parallelism Across Space

In order to analyze the convergence behavior of the proposed method, we first apply the method to solve 2-D wave and diffusion equation with available analytical solution. We have chosen Windows Vista Ultimate, Intel(R) Core(TM) i5 CPU M460 @ 2.53GHz, 4 GB RAM and 32-bit operating system for studying the parallel performance of our implementation. In this paper, the speedup of the model is calculated. The speedup, S_p is given by

$$S(p) = t_s/t_p$$

where t is the execution time of sequential algorithm and t_p is the execution time of parallel algorithm with p processors.

For ideal speedup, a multi-core computer with p processors can reduce the execution time for parallel algorithm up to p times of the execution time for sequential algorithm. Then, the efficiency of the parallel algorithm which is given by the following formulation will be calculated:

$$E_p = S(p)/p, 0 \leq E \leq 1,$$

where p is the number of processors used with speedup, $S(p)$. If the efficiency of p processors, $Ep = 1$, then the parallel algorithm is a perfect algorithm with ideal speedup.

Example 1

We consider the two-dimensional wave equation

$$\frac{d^2u}{dt^2} = \frac{d^2u}{dx^2} + \frac{d^2u}{dy^2}$$

with the boundary conditions

$$\begin{aligned} u(0,y,t) &= e^{(-y-\sqrt{2}t)}, & u(1,y,t) &= e^{(-1-y-\sqrt{2}t)}, \\ u(x,0,t) &= e^{(-x-\sqrt{2}t)}, & u(x,1,t) &= e^{(-x-1-\sqrt{2}t)}. \end{aligned}$$

and initial condition

$$u(x,y,0) = e^{(-x-y)}$$

The exact solution is given in this form,

$$u(x,y,t) = e^{(-x-y-\sqrt{2}t)}$$

Example 2

We consider the two-dimensional diffusion equation with large localized gradient

$$\frac{du}{dt} = \frac{d^2u}{dx^2} + \frac{d^2u}{dy^2}$$

with the boundary conditions

$$\begin{aligned} u(0,y,t) &= e^{-0.25\alpha^2}e^{-\beta t}, \\ u(1,y,t) &= 1 + e^{-0.25\alpha^2}e^{-\beta t}, \\ u(x,0,t) &= x + e^{-\alpha^2(x-0.5)^2}e^{-\beta t}, \\ u(x,1,t) &= x + e^{-\alpha^2(x-0.5)^2}e^{-\beta t}. \end{aligned}$$

and initial condition

$$u(x, y, 0) = x + e^{-a^{2(x-0,5)^2}}.$$

The exact solution is given in the form:

$$u(x, y, t) = x + e^{-a^2(x-0.5^2)}e^{-\beta t}$$

with α =100 and β =0.1.

4 Results and Analysis

The following table shows the time execution for both parallel and sequential algorithm when compiling the program using multi-core computers with 4 processors. Then, the speedup, S(p) and the efficiency E_p is calculated.

Table 1: Table of Speedup, $S(p)$ and Efficiency, E_p for Different Number of Grid Points (Example 1)

Total of number grid points	$t_p(s)$	$t_s(s)$	$S(p)$	E_p
25	0.0320	0.0470	1.4688	0.3672
49	0.1090	0.2972	2.7264	0.6816
100	0.2500	0.5783	2.3130	0.5783

Table 2: Table of Speedup, $S(p)$ and Efficiency, E_p for Different Number of Grid Points (Example 2).

Total of number grid points	$t_p(s)$	$t_s(s)$	$S(p)$	E_p
25	0.2175	0.2180	1.0023	0.2506
49	0.2960	0.4584	1.5486	0.3872
100	0.5610	0.9094	1.6210	0.4053

From our speedup results, the execution time for parallel algorithm is decrease compare with the execution time for sequential algorithm. The efficiency of parallel algorithm in Table 1 showing that our program contains less than 70 percent of parallelized part. The efficiency of parallel algorithm in Table 2 showing that our program contains less than 40 percent of parallelized part. We found that the different boundary value problems are given different efficiency of parallel algorithm.

5 Conclusions

In this paper, the LDQ method is used with the fourth order Runge-Kutta method for solving boundary value problems. The Runge-Kutta method is used to numerically integrate the equation in time direction. The shared memory architecture of parallel programming

by using OpenMP is performed on the LDQ method in order to reduce the execution time in simulating results.

Acknowledgment

The authors would like to thank the MyBrain15 (MyPhD), Kementerian Pengajian Tinggi Malaysia and Research University Grant, Universiti Teknologi Malaysia Q.J130000.2626.09J86 for financial support.

References

[1] Bellman, R. and Casti, J. Differential quadrature and long-term integration. *J. Math. Anal. Appl.* 1971. 34: 235-238.

[2] Bellman, R., Kashef, B. G. and Casti, J. Differential quadrature: a technique for the rapid solution of nonlinear partial differential equations. *Journal of Computational Physics.* 1972. 10: 40–52.

[3] Shu, C. *Differential Quadrature and Its Application in Engineering.* Great Britain: Springer-Verlag London. 2000.

[4] Chen, W. *Differential quadrature method and its applications in engineering: Applying special matrix product to nonlinear computations and analysis.* Shanghai Jiao Tong University: Ph.D. Thesis. 1996.

[5] Civan, F. and Sliepcevich, C. M. Application of differential quadrature to transport processes. *J. Math Anal. Appl.* 1983. 93: 206-221.

[6] Shu, C. and Richards, B. E. Application of generalized differential quadrature to solve 2- incompressible Navier-Stokes equations. *Int. J. Numerical Method Fluid.* 1990. 15: 791-798, Iss 7.

[7] Zong, Z., Zhang, Y. *Advanced Differential Quadrature Methods.* New York: Chapman & Hall/CRC Applied Mathematics and Nonlinear Science Series. 2009.

[8] Quan, J. R. and Chang, C. T. New insights in solving distributed system equations by the quadrature method-I. *Analysis. Comput. Chem. Eng.* 1989. 13: 779-788.

[9] Quan, J. R. and Chang, C. T., New insights in solving distributed system equations by the quadrature method-II. *Analysis. Comput. Chem. Eng.* 1989. 13: 1017-1024.

[10] Malik, M., and Civan, F. A comparative-study of differential quadrature and cubature method vis-avis some conventional techniques in context of convection-diffusion-reaction problems. *Chem Eng Sci.* 1995. 50: 531-547, Iss3.

[11] Makbule, M. *Differential quadrature method for time-dependent diffusion equation.* Middle East Technical University: Ph.D. Thesis. 2003.

[12] Lam, K. Y., Zhang, J. and Zong, Z. A numerical study of wave propagation in a poroelastic medium by use of localized differential quadrature method. *J. Comput. Mech.* 2004. 28: 487-511.

[13] Zong, Z. and Lam, K. Y. A localized differential quadrature method and its application to 2D wave equation. *J. Comput. Mech.* 2002. 29: 382-391.

[14] Quinn, M. J. *Parallel programming in C with MPI and OpenMP.* Singapore International Edition: McGraw-Hill. 2003.