Parallelization of elliptic solver for solving 1D Boussinesq model

IOP Conf. Series: Journal of Physics: Conf. Series **971** (2018) 012036 doi:10.1088/1742-6596/971/1/012036

Parallelization of elliptic solver for solving 1D Boussinesq model

D Tarwidi and D Adytia

School of Computing, Telkom University, Jalan Telekomunikasi No. 1 Terusan Buah Batu, Bandung 40257, Indonesia

E-mail: dedetarwidi@telkomuniversity.ac.id

Abstract. In this paper, a parallel implementation of an elliptic solver in solving 1D Boussinesq model is presented. Numerical solution of Boussinesq model is obtained by implementing a staggered grid scheme to continuity, momentum, and elliptic equation of Boussinesq model. Tridiagonal system emerging from numerical scheme of elliptic equation is solved by cyclic reduction algorithm. The parallel implementation of cyclic reduction is executed on multicore processors with shared memory architectures using OpenMP. To measure the performance of parallel program, large number of grids is varied from 2^8 to 2^{14}. Two test cases of numerical experiment, i.e. propagation of solitary and standing wave, are proposed to evaluate the parallel program. The numerical results are verified with analytical solution of solitary and standing wave. The best speedup of solitary and standing wave test cases is about 2.07 with 2^{14} of grids and 1.86 with 2^{13} of grids, respectively, which are executed by using 8 threads. Moreover, the best efficiency of parallel program is 76.2% and 73.5% for solitary and standing wave test cases, respectively.

1. Introduction

Boussinesq type of models have been widely used to solve various coastal engineering problems such as for simulating wave propagation against breakwater for harbor design, tsunami generation and propagation, wave-structure interaction, floating oscillating water column, and ship mooring system. The Boussinesq model was first introduced by J.V. Boussinesq in 1872 [1]. Unlike the Shallow Water Equations (SWE), the Boussinesq model can simulate dispersion effect in wave propagation. Some of numerical methods that have been developed to solve Boussinesq models are finite difference, finite element, finite volume, and spectral method; see for example [2, 3, 4, 5] for these numerical implementations. The model in [2] has been used for simulating long wave such as tsunami propagation (see [5, 6]), as well as short wave such as wind wave (see [2, 7, 8]).

In modeling and simulation of fluid dynamics, number of complex calculations and large data have conduced very long computational time. On the other hand, numerical simulation results are required quickly and precisely, as example in weather and ocean wave forecasting. In last decades, the development of multicore computers to get fast simulation results is growing rapidly. However, the existing code of computer programs only utilized single core even it was executed in multicore computer architecture. Hence, it is required to develop a parallel algorithm that can be implemented in distributed or shared memory architecture. OpenMP (Open Multi-Processing) is an API (application program interface) that supports shared memory multiprocessing. OpenMP

has been developed widely in scientific simulation. Marrone et al. [9] studied simulation of ship wave breaking pattern using OpenMP. Wen et al. [10] developed OpenMP parallel computing to simulate wave-structure interactions. Moreover, Michailidis et al. [11] solved tridiagonal system with pipelining technique in OpenMP.

Tridiagonal systems often appear in various scientific computing problems. Tridiagonal system arises from numerical solution of partial differential equation such as an elliptic equation which is solved by finite difference, finite volume, and finite element method. The most known of the tridiagonal system solver is Thomas algorithm [12]. The algorithm is simple and fast to solve tridiagonal system. Unfortunately, for large number of linear equations, Thomas algorithm is not efficient. Besides, Thomas algorithm is not suitable for parallelization. In last few decades, some parallel algorithms are developed to solve tridiagonal system especially those involving large size of linear equations. Allmann et al. [13] developed cyclic reduction algorithm to solve tridiagonal system in shared memory machine. Further, Stone et al. [14] used recursive doubling to large number of linear equation in tridiagonal system. In addition, Qeusada-Barriuso [15] implemented Bondeli's algorithm in OpenMP to solve tridiagonal system.

This paper focuses on parallelization of a tridiagonal system of an elliptic solution which is part of numerical solution of Boussinesq model in [2, 7, 8]. Cyclic reduction is applied to solve tridiagonal system and is implemented on OpenMP parallel architecture. More discussion of cyclic reduction can be found in [16]. Speedup and efficiency of the parallel program is analyzed to evaluate the performance of parallel solver of elliptic equation. This paper is organized as follows. In Section 2, 1D Boussinesq model and its numerical solution based staggered grid is briefly discussed. Parallelization of elliptic solution is presented in Section 3. Further, performance of parallel program is discussed in Section 4. The conclusion of the numerical experiment is given in Section 5.

2. Boussinesq Model & Numerical Solution

There are some mathematical models to simulate water wave propagation. The simplest one is the Shallow Water Equations (SWE). The SWE consists of continuity and momentum equation. However, the drawback of the SWE is the effect of dispersion on wave propagation that is not taken into account. In this study, a Boussinesq model is used to simulate water wave. In the Boussinesq type of model, the dispersion effect is taken into account when deriving the model. In this paper, we use the Boussinesq model that is introduced in [2, 7, 8]. In this Boussinesq model, there is an additional term in continuity equation which represents dispersion effect. Asides from continuity and momentum equation, in the Boussinesq model of [8], there is an auxiliary equation, i.e. an elliptic equation, that has to be solved for correcting dispersion effect.

Let $\eta(x,t)$ and $u(x,t)$ represent elevation of water surface and velocity of water wave in $x-$direction, respectively. Let $H(x,t) = \eta(x,t) + h(x)$ be the total depth of the fluid where $h(x)$ is depth from mean water level. Moreover, $\psi(x,t)$ is an auxiliary variable which depends on horizontal variable. The 1D Boussinesq model that is proposed by [2, 8] is given by

$$\partial_t \eta = -\partial_x(Hu) - \partial_x(\beta \partial_x \psi) \quad (1)$$

$$\partial_t u = -g\partial_x \eta - u\partial_x u - c_f \frac{|u|u}{H} \quad (2)$$

$$-\partial_x(\alpha \partial_x \psi) + \gamma\psi = \partial_x(\beta u), \quad (3)$$

where g and c_f denote gravitational acceleration and bottom friction coefficient, respectively. Here, $\alpha(z), \beta(z), \gamma(z)$ are coefficients which depend on vertical profile and given by

$$\alpha = \frac{2}{15}H^3, \quad \beta = -\frac{H^2}{3}, \quad \gamma = \frac{H}{3}. \quad (4)$$

Note that, as denoted in shallow water equation, equation (1) and (2) are also called continuity and momentum equation with additional term $-\partial_x(\beta\partial_x\psi)$. Furthermore, equation (3) is called the elliptic equation. In next section, numerical solution of the elliptic equation will be parallelized by using cyclic reduction algorithm.

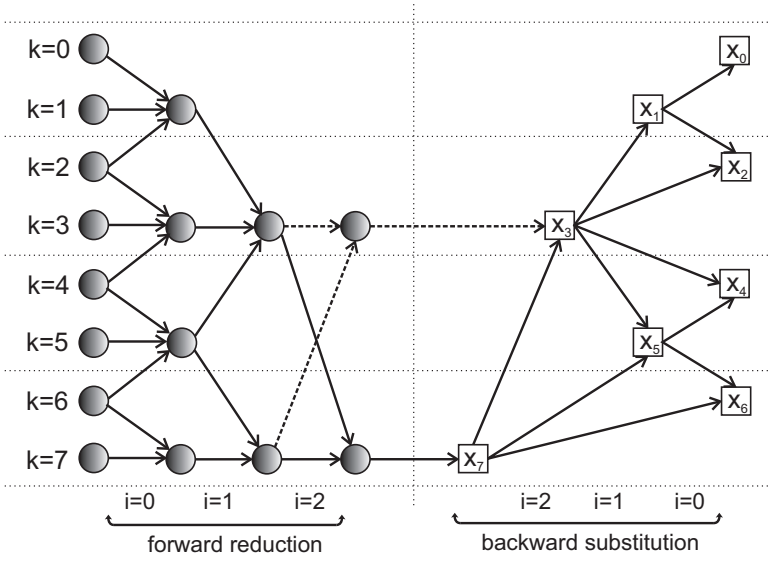

Figure 1. Illustration of parallelization of cyclic reduction algorithm for 8 linear equations and 4 processors.

Numerical solution of Boussinesq model is solved by staggered grid discretization which is proposed in [17]. More review about staggered grid can be found in [18]. Implementation of staggered grid to (1) - (3) can be described as follows. Suppose $\Omega = [L_x^-, L_x^+]$ be the computational domain of fluid. The region of interest is partitioned into N_x subregions or subintervals. Here, N_x is also called number of grids. Moreover, the length of subinterval is $\Delta x = (L_x^+ - L_x^-)/N_x$. Let subscript $i+1/2$ be the interface between node i and $i+1$. Here, $x_{i+1/2}$ is a node between x_i and x_{i+1} and is called a half grid. Moreover, x_i denotes a full grid. In staggered grid scheme, the discretization of $\eta(x,t)$ and $u(x,t)$ is assigned in difference position of grid. $\eta(x,t)$ is assigned in full grid meanwhile $u(x,t)$ is assigned in half grid. Moreover, $\psi(x,t)$ is calculated in full grid. Discretization of (1) - (3) based on staggered grid is given by

$$\frac{\eta_i^{n+1} - \eta_i^n}{\Delta t} = -\frac{{}^*H_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n - {}^*H_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^n}{\Delta x} - \frac{{}^*\beta_{i+\frac{1}{2}}^n \left(\psi_{i+1}^n - \psi_i^n\right) - {}^*\beta_{i-\frac{1}{2}}^n \left(\psi_i^n - \psi_{i-1}^n\right)}{(\Delta x)^2} \tag{5}$$

$$\frac{u_{i+\frac{1}{2}}^{n+1} - u_{i+\frac{1}{2}}^n}{\Delta t} = -g\frac{\eta_{i+1}^{n+1} - \eta_i^{n+1}}{\Delta x} - (u\partial_x u)_{i+\frac{1}{2}}^n - c_f \frac{|u|u}{H}\bigg|_{i+\frac{1}{2}}^n \tag{6}$$

$$-\frac{{}^*\alpha_{i+\frac{1}{2}}^n \left(\psi_{i+1}^n - \psi_i^n\right) - {}^*\alpha_{i-\frac{1}{2}}^n \left(\psi_i^n - \psi_{i-1}^n\right)}{(\Delta x)^2} + {}^*\gamma_i \psi_i = \frac{{}^*\beta_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n - {}^*\beta_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^n}{\Delta x} \tag{7}$$

The superscript * denotes the variable is still unknown yet. Here, ${}^*H$ is approximated by first-order upwind method:

$${}^*H_{i+\frac{1}{2}} = \begin{cases} H_i, & \text{if } u_{i+\frac{1}{2}} \geq 0 \\ H_{i+1}, & \text{if } u_{i+\frac{1}{2}} < 0. \end{cases} \tag{8}$$

The first-order upwind method is also applied to ${}^*\alpha$, ${}^*\beta$, and ${}^*\gamma$.

Finally, the computational procedure in one time step to solve 1D Boussinesq model based on staggered grid is summarized as follows.

(i) Compute ${}^*\alpha$, ${}^*\beta$, and ${}^*\gamma$ using first-order upwind method and equation (4).

(ii) Solve elliptic equation (7) to obtain ψ_i^n .

(iii) Update surface elevation η_i^{n+1} base on equation (5).

(iv) Update velocity $u_{i+\frac{1}{2}}^{n+1}$ based on equation (6).

Algorithm 1 Solve tridiagonal system $A\mathbf{x} = \mathbf{f}$ using Cyclic Reduction

Input: matrix size: n; tridiagonal matrix: $A[i][j], \quad i = 0, 1, 2, ..., n-1, j = 0, 1, 2, ..., n-1$ with $A[i][i] \neq 0$; right hand side: $f[i], \quad i = 0, 1, 2, ..., n-1$

Output: $x[i], i = 0, 1, 2, ..., n-1$

{Forward Reduction}
for $i = 0$ to $(\log_2 n) - 1$ **do**
  for $j = 2^{i+1} - 1$ to $n-1$, $j = j + 2^{i+1}$ **do**
    $idx_1 \leftarrow j - 2^i$
    $idx_2 \leftarrow j + 2^i$
    $\alpha \leftarrow A[j][idx_1]/A[idx_1][idx_1]$
    $\gamma \leftarrow A[j][idx_2]/A[idx_2][idx_2]$
    for $k = 0$ to $n-1$ **do**
      $A[j][k] \leftarrow A[j][k] - (\alpha A[idx_1][k] + \gamma A[idx_2][k])$
    end for
    $f[j] \leftarrow f[j] - (\alpha f[idx_1] + \gamma F[idx_2])$
  end for
end for
{Backward Substitution}
$idx_2 \leftarrow n - 1$
$idx_1 \leftarrow n/2 - 1$
$x[idx_2] = f[idx_2]/A[idx_2][idx_2]$
$x[idx_1] = (f[idx_1] - A[idx_1][idx_2] \cdot x[idx_2])/A[idx_1][idx_1]$
for $i = (\log_2 n) - 2$ to 0 **do**
  for $j = 2^{i+1} - 1$ to $n-1$, $j = j + 2^{i+1}$ **do**
    $idx_1 \leftarrow j - 2^i$
    $idx_2 \leftarrow j + 2^i$
    $sum_1 = f[idx_1]$
    $sum_2 = f[idx_2]$
    for $k = 0$ to $n-1$ **do**
      $sum_1 \leftarrow sum_1 - A[idx_1][k] \cdot x[k]$
      $sum_2 \leftarrow sum_2 - A[idx_2][k] \cdot x[k]$
    end for
    $x[idx_1] \leftarrow (sum_1 + A[idx_1][idx_1] \cdot x[idx_1])/A[idx_1][idx_1]$
    $x[idx_2] \leftarrow (sum_2 + A[idx_2][idx_2] \cdot x[idx_2])/A[idx_2][idx_2]$
  end for
end for

3. Parallelization of Elliptic Solver

Discretization of the elliptic equation using staggered grid is given by (7). In solving (7), there is an expensive computational cost due to large number of grids. By rearranging (7) in term of ψ, the linear tridiagonal system is obtained and is given by

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{N_x-1} & b_{N_x-1} & c_{N_x-1} \\ 0 & & & a_{N_x} & b_{N_x} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N_x-1} \\ \psi_{N_x} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N_x-1} \\ f_{N_x} \end{bmatrix} \tag{9}$$

where

$$\begin{aligned} a_i &= -\frac{{}^*\alpha_{i-\frac{1}{2}}}{(\Delta x)^2}, \quad i = 2, \cdots, N_x \\ b_1 &= \frac{{}^*\alpha_{\frac{3}{2}}}{(\Delta x)^2} + {}^*\gamma_1 \\ b_i &= \frac{{}^*\alpha_{i-\frac{1}{2}} + {}^*\alpha_{i+\frac{1}{2}}}{(\Delta x)^2} + {}^*\gamma_i, \quad i = 2, \cdots, N_x - 1 \\ b_{N_x} &= \frac{{}^*\alpha_{N_x-\frac{1}{2}}}{(\Delta x)^2} + {}^*\gamma_{N_x} \\ c_i &= -\frac{{}^*\alpha_{i+\frac{1}{2}}}{(\Delta x)^2}, \quad i = 1, \cdots, N_x - 1 \\ f_i &= \frac{{}^*\beta_{i+\frac{1}{2}} u_{i+\frac{1}{2}} - {}^*\beta_{i-\frac{1}{2}} u_{i-\frac{1}{2}}}{\Delta x} \end{aligned}$$

Cyclic reduction algorithm is used to solve tridiagonal system in (9). This method uses elimination step which is similar to Gaussian elimination. However, cyclic reduction can be implemented into parallel program. In general, cyclic reduction works by eliminating variable x_{i-1} and x_{i+1} from equation i by utilizing equations $i-1$ and $i+1$. By this elimination, the number of equation is reduced to a half number of equations. In the new linear system, the same elimination is also assigned.

There are two phases to solve tridiagonal system by using cyclic reduction algorithm. The first phase is forward reduction. In each step of forward reduction, half of the variables in the tridiagonal system is eliminated. The process is repeated until two equations with two unknown variables are left. Here, two unknown variables with two linear equations is then solved by simple elimination. Cyclic reduction algorithm to solve tridiagonal system $Ax = f$ is given by Algorithm 1. The detail of the algorithm can be found in [19]. Illustration to solve 8-element of tridiagonal system using 4 threads is shown by Figure 1.

Implementation of cyclic reduction in OpenMP is described in Figure 2. At first, N_x/p equations are distributed among the threads. Here, p denotes the number of threads used in parallel program. Tridiagonal matrix A and right hand side f is assigned as shared variables in forward reduction phase. Furthermore, in backward substitution, x, A, f, and N_x is considered as shared variables. Note that, in the simulation, the tridiagonal system solver function is called in each time step.

4. Results and Discussion

The computational time in solving the Boussinesq model numerically has become an important part in the simulation of water waves. One of the numerical solution of the Boussinesq model

```
/*forward substitution*/
for(i=0; i<log2(Nx); i++){
n = pow(2,i+1);
# pragma omp parallel shared ( A, f, Nx ) \
private ( j, k, index1, index2, alpha, gamma )
{
# pragma omp for
for(j=pow(2,i+1)-1;j<Nx;j=j+n){
...
}
}
}
/*backward substitution*/
...
for(i=log2(Nx)-2;i>=0;i--){
n = pow(2,i+1);
# pragma omp parallel shared ( x, A, f, Nx ) \
private ( j, k, index1, index2, sum1, sum2 )
{
# pragma omp for
for(j=pow(2,i+1)-1;j<Nx;j=j+n){
...
# pragma omp critical
for(k=0;k<Nx;k++){
...
}
...
}
}
}
```

Figure 2. Parallel implementation of cyclic reduction algorithm in OpenMP.

that can be optimized its computational time is elliptic solver. As described in Section 3, the numerical solution of elliptic equation is in the form of tridiagonal system. In this case, the cyclic reduction algorithm is used to solve the tridiagonal system. The numerical experiment is conducted in personal computer with specification: Intel Xeon CPU E3-1230 v5 @3.4 GHz, 16 GB of RAM, and 64-bit OS type.

Performance of cyclic reduction algorithm implemented in OpenMP parallel architecture is evaluated. Two test cases, i.e. solitary and standing wave, are employed to measure the performance of parallel program. The benefit of parallel program is measured by the quantity of speedup and efficiency. The speedup represents the comparison of execution time between parallel program (multiprocessor) relative to sequential program (single processor). The speedup $S_p(n)$ is defined as

$$S_p(n) = \frac{T^*(n)}{T_p(n)},$$

where $T^*(n)$ and $T_p(n)$ are execution time of sequential and parallel program with size n, respectively. Here, p is the number of threads used to solve the problem. Moreover, the efficiency

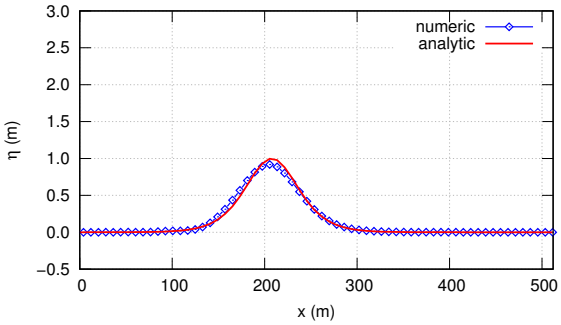

Figure 3. Numerical vs analytical solution for solitary wave.

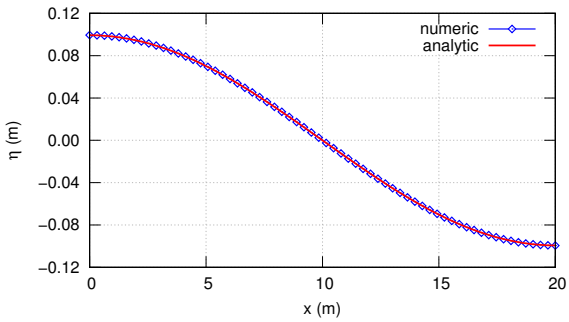

Figure 4. Numerical vs analytical solution for standing wave.

Table 1. Execution time of sequential program (1 thread) and parallel program (2, 4, and 8 threads) for Test Case 1.

# threads	Execution time (in seconds)						
	$N_x = 2^8$	$N_x = 2^9$	$N_x = 2^{10}$	$N_x = 2^{11}$	$N_x = 2^{12}$	$N_x = 2^{13}$	$N_x = 2^{14}$
1	0.57	5.10	17.71	74.29	273.30	1032.00	4201.60
2	0.89	4.55	13.93	49.00	185.11	694.00	2755.75
4	1.42	5.04	16.25	43.76	150.29	538.08	2141.17
8	1.38	6.21	16.12	47.13	137.23	504.02	2026.88

of a parallel program can be expressed as

$$E_p(n) = \frac{S_p(n)}{p}.$$

The efficiency expresses processor utilization of parallel program. In other words, it measures the fraction of time for which a processor is usefully exploited.

4.1. Test Case 1: Solitary wave

In this test case, a solitary wave is considered as initial condition of the Boussinesq model. Let $\Omega = [-512, 512]m$ be the domain of the simulation. The domain is discretized in to finite number of grids. To measure performance of the parallel program, the number of grids is varied from 2^8 to 2^{14}. Moreover, time step used in this simulation is $\Delta t = 0.01s$ with final time is $10s$. The initial condition for solitary wave is given by

$$\eta(x, 0) = 2A_0 \operatorname{sech}^2\left(\sqrt{\frac{3A_0}{4h_0^2(h_0 + A_0)}}\, x\right),$$

where $A_0 = 1$ and $h_0 = 10$.

Numerical solution of 1D Boussinesq model with solitary wave as initial condition is depicted in Figure 3. From the figure, it can be seen that the numerical solution shows a good agreement with the analytical solution. Table 1 displays the execution time of sequential and parallel program of Test Case 1 for various number of grids. Table 1 was obtained by executing the parallel program for 1, 2, 4, and 8 threads. When executing the parallel program, there was no

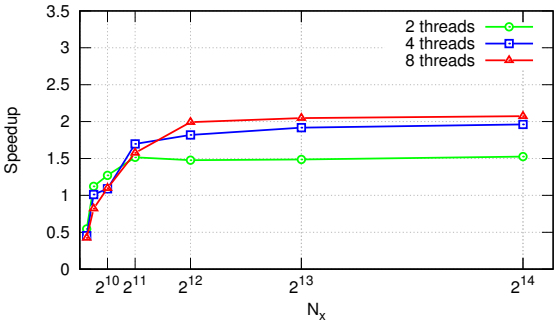

Figure 5. Speedup of parallel program for Test Case 1.

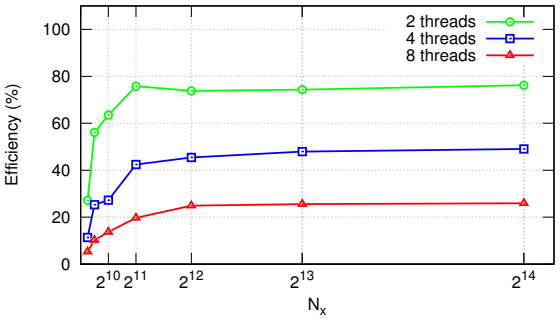

Figure 6. Efficiency of parallel program for Test Case 1.

Table 2. Execution time of sequential program (1 thread) and parallel program (2, 4, and 8 threads) for Test Case 2.

# threads	Execution time (in seconds)						
	$N_x = 2^8$	$N_x = 2^9$	$N_x = 2^{10}$	$N_x = 2^{11}$	$N_x = 2^{12}$	$N_x = 2^{13}$	$N_x = 2^{14}$
1	1.28	10.13	79.41	600.26	3991.28	22671.6	175409
2	1.42	9.33	59.17	421.15	2968.81	15616.4	119344
4	1.77	9.78	54.66	349.93	2569.05	12339.6	109587
8	1.85	9.75	53.79	351.55	2528.66	12181.0	104278

any program that runs in the computer so that each scenario has the same treatment. It can be seen that the execution time of the sequential program is faster than the parallel program for $N_x = 2^8$ and $N_x = 2^9$. Furthermore, The execution time of parallel program is faster than the sequential program if the number of grids is more than 2^{10}.

The speedup of parallel program for Test Case 1 by using 2, 4, 8 threads can be seen in Figure 5. It can be seen that the speedup of parallel program grows sublinearly with respect to number of grids used. From the figure, it can be revealed that the speedup more than 1 if the number of grids is more than 2^{10}. It means that the parallel program is efficient if $N_x \geq 2^{10}$. However, the speedup for N_x greater than 2^{12} does not change significantly. The speedup by using 2, 4, and 8 threads are around 1.5, 1.9, and 2.0, respectively. However, the best speedup for solitary wave test case is 2.07 which is executed in 8 threads and using 2^{14} of grids.

Figure 6 shows efficiency of parallel program for Test Case 1. It can be seen that parallel program by using 2 threads is more efficient than 4 and 8 threads. The highest efficiency is about 76.2% and is achieved when the parallel program is using 2 threads.

4.2. Test Case 2: Standing Wave

Initial condition for the second test case is considered as standing wave. The initial condition of standing wave is given by

$$\eta(x, 0) = 2\, A_0 \cos\left(\frac{\pi}{20}\, x\right),$$

where $A_0 = 0.05$. The numerical domain of standing wave simulation is given by $\Omega = [0, 20]m$. Similar to Test Case 1, the standing wave simulation also uses various of number of grids from $N_x = 2^8$ to $N_x = 2^{14}$. The final time of the simulation is also 10 seconds. However, in the standing wave simulation, the time step Δt follows the number of grids that used in

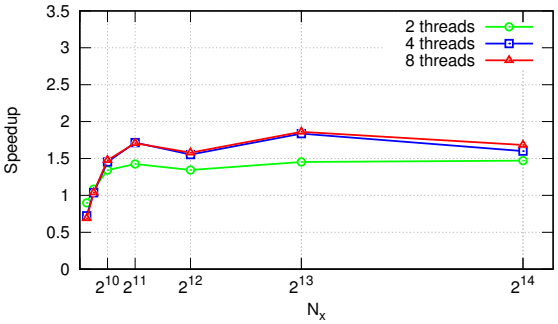

Figure 7. Speedup of parallel program for Test Case 2.

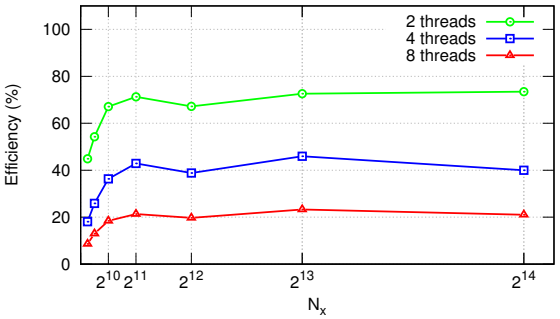

Figure 8. Efficiency of parallel program for Test Case 2.

the simulation. The more number of grids, the smaller time step used in the simulation. As consequence, the number of time iteration becomes difference in each number of grids used.

Numerical result of 1D Boussinesq model with standing wave against its analytical solution is shown by Figure 4. It can be revealed that numerical solution of standing wave shows good agreement with the analytical solution. Table 2 shows execution time of sequential and parallel program for $N_x = 2^8$ to $N_x = 2^{14}$. From this table, it can be seen that the execution time of parallel program is faster than the sequential program if $N_x \geq 2^9$. Speedup and efficiency of parallel program for Test Case 2 are presented in Figure 7 and Figure 8, respectively. It can be seen that the best speedup is 1.86 which is achieved for 8 threads and $N_x = 2^{13}$. The highest of efficiency is achieved for 2 threads and is about 73.5%. Moreover, the efficiency for 4 and 8 threads with $N_x = 2^{13}$ are 45.9% and 23.2%, respectively.

5. Conclusion

Parallelization of elliptic solver for solving 1D Boussinesq model has been successfully conducted. Cyclic reduction algorithm has been implemented in parallel program and executed on multicore processors with shared memory architectures using OpenMP. The best speedup of solitary and standing wave test cases is about 2.07 with 2^{14} of grids and 1.86 with 2^{13} of grids, respectively, which is executed by using 8 threads. By the numerical experiment, it can be concluded that the speedup does not change significantly if the number of grids is greater than 2^{12}. Moreover, the efficiency of parallel program is 76.2% and 73.5% for solitary and standing wave test cases, respectively.

References

[1] Boussinesq J 1872 *J. Math.* **17** 55–108
[2] Adytia D 2014 *The Twenty-fourth International Ocean and Polar Engineering Conference* (International Society of Offshore and Polar Engineers)
[3] Madsen P A and Sørensen O R 1992 *Coastal engineering* **18** 183–204
[4] Nwogu O 1993 *Journal of waterway, port, coastal, and ocean engineering* **119** 618–638
[5] van Groesen E, Adytia D and Andonowati 2008 *Natural Hazards and Earth System Sciences* **8** 175–185
[6] Adytia D and van Groesen E 2009 *Asian and Pacific Coasts 2009, Proceedings of the 5th International Conference on APAC 2009* (World Scientific) pp 122–128
[7] Adytia D 2012 *Coastal zone simulation with variational Boussinesq modelling* (PhD Thesis, University of Twente, The Netherlands)
[8] Adytia D and Lawrence 2016 *ASME 2016 35th International Conference on Ocean, Offshore and Arctic Engineering* (American Society of Mechanical Engineers)
[9] Marrone S, Bouscasse B, Colagrossi A and Antuono M 2012 *Computers & Fluids* **69** 54–66
[10] Wen H, Ren B, Dong P and Wang Y 2016 *Applied Ocean Research* **59** 366–377
[11] Michailidis P D and Margaritis K G 2011 *Journal of Computational and Applied Mathematics* **236** 326–341

[12] Thomas L H 1949 *Watson Sci. Comput. Lab. Rept., Columbia University, New York* **1**
[13] Allmann S, Rauber T and Runger G 2001 *Parallel and Distributed Processing, 2001. Proceedings. Ninth Euromicro Workshop on* (IEEE) pp 290–297
[14] Stone H S 1973 *Journal of the ACM (JACM)* **20** 27–38
[15] Quesada-Barriuso P, Lamas-Rodríguez J, Heras D B, Bóo M and Argüello F 2011 *Int. Conf. on Parallel and Distributed Processing Techniques and Applications (as part of WorldComp2011 Conference)*
[16] Rauber T and Rünger G 2013 *Parallel programming: For multicore and cluster systems* (Springer Science & Business Media)
[17] Stelling G S and Duinmeijer S A 2003 *International journal for numerical methods in fluids* **43** 1329–1354
[18] Pudjaprasetya S, Magdalena I and Tjandra S 2017 *Journal of Earthquake and Tsunami* **11** 1740002
[19] Karniadakis G E and Kirby II R M 2003 *Parallel scientific computing in C++ and MPI: a seamless approach to parallel algorithms and their implementation* (Cambridge University Press)