Computing and Visualization in Science manuscript No.
(will be inserted by the editor)

A relaxation approach to time-parallelization: a numerical study

Benjamin W. Ong · Felix Kwok

Received: date / Accepted: date

Abstract The idea behind this parallel-in-time scheme is to simultaneously compute many Jacobi or Newton iterates at different time levels, relaxing to the final solution. Although this work began as an extension of an adaptive Schwarz waveform relaxation scheme [2], a similar algorithm was previously proposed in 1990 by Womble [3]. However, we believe that this manuscript contains the first numerical results that embeds a Newton iterative scheme (for solving systems of nonlinear equations) within the parallel-in-time relaxation framework. A numerical study shows that some parallel speedup can be observed for provided examples.

Keywords Waveform Relaxation · Time-Parallelization · Iterative Methods

Mathematics Subject Classification (2010) 65-Y05, 65-F10

1 Introduction

To solve time-dependent partial differential equations,

$$u_t = f(t,u), \quad t \in [T_0, T_f], \quad x \in \Omega, \tag{1}$$

sequential time stepping methods are typically applied to approximate solutions to the PDE at discrete time levels

$$T_0 = t_0 < t_1 < \cdots < t_{n-1} < t_n < \cdots < t_N = T_F.$$

Benjamin W. Ong
Michigan Technological University
Department of Mathematical Sciences
E-mail: ongbw@mtu.edu

Felix Kwok
Hong Kong Baptist University
Department of Mathematics
E-mail: felix_kwok@hkbu.edu.hk

If a sequential time-stepper is used, the solution at time level t_{n-1} has to be available before one seeks a solution at time level t_n.

Similar to a previously introduced parallel time stepping method [3], we are interested in introducing parallelism-in-time when an iterative scheme is used to find the solution at each time level. If an implicit time integrator is used, in conjunction with some spatial discretization, to approximate solutions to eq. (1), one often obtains (i) a linear system of equations for the unknown solution at each time level, whereby an iterative solver can be used to solve the linear system at each time level, or (ii) a non-linear system of equations for the unknown solution at each time level, whereby a Newton solver might be employed to solve the system of nonlinear equations. In this latter case, one could, in addition, envision a nested iterative scheme where an iterative linear solver is used within the Newton method. We do not address the nested iteration case in this manuscript. For the two non-nested cases, a sequential time integrator which uses an iterative method to advance the solution at each time level can be described by Algorithm 1, where Q_{n-1} is the iteration function for time level n.

```
for n = 1, ..., N do
    set k = 1;
    set u_n^[0] = u_{n-1};
    while not converged do
        u_n^[k] = Q_n(u_n^[k-1], u_{n-1});
        k ← k + 1;
    end
end
```

Algorithm 1: Sequential time integration, where an iterative method is used to advance the solution at each time level.

Observe that the iteration update for time level n,

$$u_n^{[k]} = Q_{n-1}(u_n^{[k-1]}, u_{n-1}) \tag{2}$$

requires a previous iterate at the current time level n, and the (converged) solution from the previous time level, $n-1$. If one is willing to use an approximation to u_{n-1} in the iteration formula, eq. (2), then a parallel iterative formula can be obtained. Suppose that an initial guess for the solution at each discrete time level, $u_n^{[0]}, n = 1, \dots N$ is available, and that one is willing to use this initial guess to simultaneously advance the solution at each time level, i.e.,

$$u_n^{[1]} = Q_{n-1}^{[0]}(u_n^{[0]}, u_{n-1}^{[0]}), \quad n = 1, \dots, N.$$

One can repeat this process iteratively,

$$u_n^{[k]} = Q_{n-1}^{[k-1]}(u_n^{[k-1]}, u_{n-1}^{[k-1]}), \quad n = 1, \dots, N.$$

Switching the order of the loops leads to Algorithm 2, where the outer while-loop is iterated until some global convergence criteria is met. Algorithm 2 is a simplified variant of "Method 2" that was previously proposed in [3], where N processors can be used to compute solutions to the N discrete time-levels. Although convergence of Algorithm 2 can be shown for linear PDEs [3], this algorithm is not efficient for various choices of iterative schemes [1], and special care must be taken to observe parallel speed-up.

```
for n = 0, ..., N do in parallel
    set u_n^[0] = u_0; // initial guess
end
set k = 1;
while not converged do
    for n = 1, ..., N do in parallel
        u_n^[k] = Q_{n-1}^[k-1](u_n^[k-1], u_{n-1}^[k-1]);
    end
    k ← k + 1;
end
```

Algorithm 2: Parallel time integration. N processors are used to simultaneously iterate on all N time levels.

One approach to improving parallel efficiency is to modify Algorithm 2 to only iterate on P time levels ($P < N$) simultaneously using P processors, and not expending any effort to update the remaining $P - N$ levels. Intuitively, this makes sense on two levels: (1) if a solution at an earlier time-level has already converged, there is no reason to continue the iterative update; (2) perhaps it is not useful to start iterating a later time-level until a reasonable initial guess is available. Indeed, in the extreme case when $P = 1$, one would like the parallel algorithm to recover algorithm 1, iterating each time level until convergence before advancing to the next time level. An algorithm that utilizes P processors is summarized in Algorithm 3. Each iteration in Algorithm 3,

$$u_n = Q_{n-1}(v_n, v_{n-1}),$$

uses the most accurate iterate at the current time level n, v_n, and the most accurate solution from the previous time level $(n-1)$, v_{n-1}. Convergence is checked before the new iterates are stored in v_n. A similar algorithm with a slightly more sophisticated starting procedure was recently analyzed in the context of Schwarz waveform relaxation [2]. We extend that work now to consider more general iterative solvers using Algorithm 3.

```
for n = 0, ..., N do in parallel
    set v_n = u_0; // initial guess
end
set k = 1;
set m = 0; // number of converged intervals
while m < N do
    set i_s = m + 1; // first unconverged interval
    set i_e = min(N, m + P) // at most P intervals
    for n = i_s, ..., i_e do in parallel
        u_n = Q_{n-1}(v_n, v_{n-1})
    end
    while ||u_{m+1} - v_{m+1}|| < TOL do
        m ← m + 1; // update # converged
    end
    for n = i_s, ..., i_e do in parallel
        v_n ← u_n;
    end
end
```

Algorithm 3: Parallel time integration. P processors are used to simultaneously iterate on all N time levels.

2 Numerical Studies

2.1 Implicit Euler and Jacobi

We begin with the canonical example, the parabolic heat equation with Dirichlet boundary conditions,

$$u_t = u_{xx}, \quad x \in [0,1], \quad t \in [0, 0.1],$$
$$u(0,x) = \sin \pi x, \quad u(t,0) = u(t,1) = 0.$$

Applying a method of lines approach, for example with centered-differences in space, a semidiscretized equation is obtained,

$$\frac{d\,u}{dt} = L\,u.$$

Applying an implicit Euler approximation, one arrives at an iteration formula

$$A\,u_n = u_{n-1}, \tag{3}$$

where A is a constant tri-diagonal matrix which is strictly diagonally dominant. If matrix A is decomposed into it's diagonal component D and remainder R, i.e., $A = D + R$, then the solution to the linear system, eq. (3), can be obtained iteratively using the Jacobi method,

$$u_n^{[k+1]} = D^{-1}(u_{n-1} - R\,u_n^{[k]}). \tag{4}$$

Since A is strictly diagonally dominant, it can be shown that eq. (4) converges for any initial guess, $u_n^{[0]}$.

To apply the parallel time integrator, Algorithm 3, one uses the modified iteration function

$$Q(v_n, v_{n-1}) =: Q_n(v_n, v_{n-1}) = D^{-1}(v_{n-1} - R\,v_n).$$

Figure 1 shows the number of Jacobi iterations required at each time step for varying number of time-parallel tasks. 100 spatial intervals and 20 time steps were used, and the Jacobi iteration was terminated after a tolerance of 10^{-4} was reached. As P increases, an increasing number of Jacobi iterates are needed at each time step, consistent with the analysis in [1]. The benefit of using parallel time integrators however, is that several Jacobi iterations can be simultaneously computed. Table 1 shows the number of parallel Jacobi iterations, speedup, and efficiency for parallel time integrators with varying number of time-parallel tasks. Here, Jacobi iterates that can be simultaneously computed count as a single parallel Jacobi iterate. For a small number of time-parallel tasks, reasonable speedup and efficiency is observed.

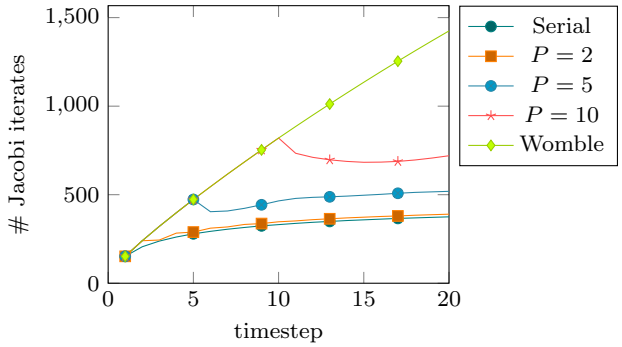

Fig. 1 Heat equation: # Jacobi iterations required at each time step for the serial and parallel time integrators with varying number of time-parallel tasks (P). As P increases, an increasing number of Jacobi iterates are needed at each time step, and parallel efficiency is lost.

P	# Parallel Iterations	speedup	efficiency
1 (serial)	6291	–	–
2	3392	1.85	0.93
3	2547	2.46	0.82
4	2167	2.90	0.73
5	1954	3.22	0.64
10	1540	4.09	0.41
20 (Womble)	1425	4.41	0.22

Table 1 Heat equation: time-parallel integration with varying number of (simultaneous) parallel Jacobi iterations. For a small number of time-parallel tasks, reasonable speedup and efficiency is observed.

2.2 Implicit Euler and Newton

Inviscid Burgers' equation is a non-linear time-dependent PDE,

$$u_t + \left(\frac{1}{2}u^2\right)_x = 0, \quad x \in [0,1], \quad t \in [0, 0.5],$$
$$u(0,x) = 1 - \cos 2\pi x, \quad u(t,0) = u(t,1) = 0.$$

If we utilize implicit Euler and an up-wind conservative spatial discretization, then the non-boundary nodes $U_j^n := u(t_n, x_j)$ satisfy

$$U_j^n = U_j^{n-1} - \frac{\Delta t}{\Delta x}\left[\frac{1}{2}\left(U_j^n\right)^2 - \frac{1}{2}\left(U_{j-1}^n\right)^2\right].$$

This gives rise to a non-linear system of equations,

$$F(u_n, u_{n-1}) = 0. \tag{5}$$

One approach to solving this non-linear system of equations is to use Newton's method, which generates an iteration formula of the form

$$u_n^{[k+1]} = u_n^{[k]} - (\nabla F(u_n^{[k]}, u_{n-1}))^{-1} F(u_n^{[k]}, u_{n-1}),$$

where ∇F is the Jacobian matrix. Hence, to apply the parallel time integrator, Algorithm 3, one uses the iteration function

$$Q_n(v_n, v_{n-1}) = v_n - (\nabla F(v_n, v_{n-1}))^{-1} F(v_n, v_{n-1}).$$

Figure 2 shows the number of Newton iterations required at each time step for varying number of time-parallel tasks. 2000 spatial intervals and 10 time steps were used, and the Newton iteration was terminated after a tolerance of 10^{-10} was reached. The number of Newton iterations needed by the serial backward Euler integrator varies throughout the domain, but for most of the domain, only $5 - 7$ Newton iterations are required, limiting the possibility for parallel speedup. As P increases, the total number of Newton iterations (area under each curve) increases

We expect limited benefit from using parallel time integrators since the sequential integrator only requires

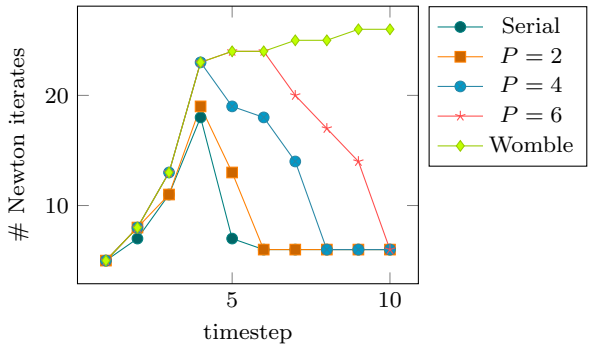

Fig. 2 Burgers' equation: # Newton iterations required at each time step for the serial and parallel time integrators with varying number of time-parallel tasks (P). As P increases, the total number of Newton iterations required (area under the curve) increases.

5–7 Newton iterations at most of the time levels. Table 2 shows the number of parallel Newton iterations, speedup, and efficiency for parallel time integrators with varying number of time-parallel tasks. Here, Newton iterates that can be simultaneously computed count as a single parallel Newton iterate. For a small number of time-parallel tasks, reasonable speedup and efficiency is observed.

P	# Parallel Iterations	speedup	efficiency
1 (serial)	78	–	–
2	45	1.73	0.87
3	36	2.17	0.72
4	32	2.33	0.61
5	30	2.60	0.52
10 (Womble)	26	3.00	0.3

Table 2 Burgers' equation: time-parallel integration with varying number of (simultaneous) parallel Newton iterations. For a small number of time-parallel tasks, reasonable speedup and efficiency is observed.

3 Further Observations / Future Work

The observed work diagrams (number of Jacobi/Newton iterations), figs. 1 and 2, and the observed parallel speedup and efficiency, tables 1 and 2, are consistent with a previously developed error propagation model for a different relaxation scheme [2]. In that model, there are two sources of error that contribute to the error propagation model. There is the error arising due to advancing the initial guess from one time level to the next; this error may be amplified or decayed based on the discretization scheme. There is also the error that arises due to the iterative approximation. Denote the error due to advancing the initial guess from time level n to time level $(n+1)$ at iteration k as $G(n,k)$, and denote the error that arises due to the iterative update from iterate k to iterate $(k+1)$ at time level n as $H(n,k)$. The error propagation model [2] is built on a system of coupled recurrence equations

$$G(n,k) \leq \alpha G(n-1,k) + \quad H(n,k), \tag{6a}$$
$$H(n,k+1) \leq \quad G(n-1,k) + \beta H(n,k), \tag{6b}$$

where α and $\beta < 1$ are constants. The constant α measures the amplification or decay of the error in the initial condition for each time level if there is no contribution from the iterative error. This constant can be greater than 1. The constant β measures the contraction of the error in the iterative scheme when the initial guess is exact. This is typically a desired property of the iterative solver, and must be less than 1 *in some appropriate norm* if the original method converges.

The numerical evidence indicates that with some effort, one should be able to identify error measures $G(m,k)$ and $H(m,k)$ for the two numerical examples that satisfy eq. (6). The analysis is elusive and a topic for more exploration.

4 Conclusion

This work started as an extension of an adaptive Schwarz waveform relaxation algorithm proposed by the present authors [2]. The idea is to simultaneously compute many Jacobi or Newton iterates at different time levels, relaxing to the final solution. It turns out, that a similar algorithm was previously proposed in 1990 by Womble [3]. However, we believe that this manuscript contains the first numerical results that embeds a Newton iterative scheme (for solving systems of nonlinear equations) within the relaxation framework. Some parallel speedup is observed for both provided examples, and the work diagrams appear consistent with a previously develop error propagation model [2], though it remains to be shown that the same analysis can be applied.

References

1. Deshpande, A., Malhotra, S., Schultz, M., Douglas, C.C.: A rigorous analysis of time domain parallelism. Parallel Algorithms Appl. **6**, 53–62 (1995)
2. Kwok, F., Ong, B.: Schwarz waveform relaxation with adaptive pipelining. SIAM Journal on Scientific Computing **41**(1), A339–A364 (2019). DOI 10.1137/17M115311X. URL `https://doi.org/10.1137/17M115311X`
3. Womble, D.: A time-stepping algorithm for parallel computers. SIAM Journal on Scientific and Statistical Computing **11**(5), 824–837 (1990). DOI 10.1137/0911049. URL `https://doi.org/10.1137/0911049`