Vol.2021-HPC-180 No.2
2021/7/20

An effective parallel-in-time method for explicit time-marching schemes

Yen-Chen Chen[1,a)] Kengo Nakajima[1,b)]

Abstract: Various parallel-in-time methods have been studied since the year 1964. Methods such as parareal and multigrid reduction in time (MGRIT) have been shown to provide reasonable acceleration to PDE implicit schemes. However, only a few have applied pure explicit schemes. This research introduces a parallel-in-time method optimized for explicit schemes. The proposed method constructs a multiple coarsening layer structure similar to MGRIT and solves the parareal algorithm through coarse to fine layers and the relaxation method is defined to solve across the whole time segment divided by the number of processors. This research conducts numerical experiments for a 1-dimensional advection equation and a 2-dimensional simulation of compressible viscous flow around a circular cylinder, using explicit schemes. The research result shows that the proposed parallel-in-time method could improve the computation efficiency of explicit solvers compared to pure spatial parallelization.

Keywords: PinT, parallel-in-time, parareal, explicit time-marching scheme

1. Introduction

Parallel computation plays a critical role in modern numerical computations. With the rapid development of supercomputers, it has become common to apply thousands of parallel threads for each problem. However, researchers soon found out that there is a bottleneck to the maximum number of spatial parallelization for numerical PDE problems. In order to exploit more parallel performance, researchers developed parallel-in-time (PinT) methods that parallelize the problem in the time dimension. PinT methods have been developed for more than 50 years[2]. However, it was not until recent years that PinT methods gained particular attention. Although some PinT methods are proven to work with explicit schemes, none has been sufficient to work with a small number of cores to the best of our knowledge. The present work proposes a method that targets the best parallel performance for parallel-in-time with explicit schemes. This paper references two existing PinT method, *parareal*[3] and multigrid reduction-in-time (MGRIT) method[4]. The proposed PinT method in this work includes ideas from both *parareal* and MGRIT.

Hyperbolic equations have been one of the most challenging categories for parallel-in-time methods, which most methods show poor convergence. One of the most classic and commonly studied hyperbolic equations is the advection equation. Some of the most recent papers[5], [6], [7] has shown successful parallel-in-time results for advection equation. Another challenging application for parallel-in-time methods is Computational Fluid Dynamics (CFD). CFD simulations involve multiple differential equations and variables. The instability of CFD with coarse grids makes it even harder to apply parallel-in-time methods. So far the following works[7], [9], [10], [11], [15] have tried different parallel-in-time methods on CFD. However, most work with implicit schemes and not of which has very good efficiency. In the present work, the proposed method is tested on two of these challenging applications. First, a one-dimensional advection example is tested, and the performance is compared with the MGRIT method using xbraid library[1]. Numerical experiments in the result section choose a one-dimensional advection equation and a two-dimensional explicit CFD simulation example.

As an overview, existing methods *parareal* and MGRIT will be introduced in Section 2. Section 3 introduces the proposed PinT method, which is based on ideas from *parareal* and MGRIT. In Section 4, an one-dimensional advection example and a two-dimensional CFD simulation applies the proposed PinT method and compares to parallel-in-space and xbraid library (only for advection example) performances. Finally, Section 5 discusses how the proposed method could achieve such performance and conclusions and future works are summarized in Section 6.

2. Existing Methods

The present work proposes a method base on *parareal* and multigrid reduction-in-time (MGRIT) method. Therefore,

[1] The University of Tokyo, Bunkyo-ku, Tokyo 133–8656, Japan
[a)] yenchen_chen@mist.i.u-tokyo.ac.jp
[b)] nakajima@cc.u-tokyo.ac.jp

before giving a detailed explanation of the proposed method, this section provides a brief overview of the two methods.

2.1 Parareal Method

Parareal[3] is an iterative PinT method with two levels of coarsening. *Parareal* is composed of two solvers. A fine $\mathcal{F}$ solver provides the necessary precision, and a coarse solver $\mathcal{G}$ provides the necessary speedup.

$$y^k_{f,j} = \mathcal{F}(y^k_j, t_j, t_{j+1}), \quad y^k_{c,j} = \mathcal{G}(y^k_j, t_j, t_{j+1})$$

First, the method divides the whole timeline into smaller time intervals. The number of intervals depends on the number of available parallel processors. In each time interval, define "the jump" as the difference between results of fine and coarse solvers.

$$S^k_j = \mathcal{F}(y^k_j, t_j, t_{j+1}) - \mathcal{G}(y^k_j, t_j, t_{j+1})$$

Secondly, propagate the jumps from the first time interval to the last using a coarse solver. In general, we could use the same one $\mathcal{G}$ from the last step.

$$y^{k+1}_{j+1} = \mathcal{G}(y^{k+1}_j, t_j, t_{j+1}) + \mathcal{F}(y^k_j, t_j, t_{j+1}) - \mathcal{G}(y^k_j, t_j, t_{j+1})$$

Iterate the previous three steps until the result for the last time step converges to the desired accuracy.

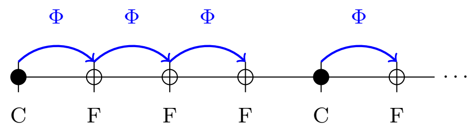
Generally, two solvers could be the same time-marching scheme with different time-step sizes. However, directly coarsening the time step size is dangerous while working with explicit schemes since it might violate the Courant-Friedrichs-Lewy (CFL) condition. More details will be explained in Section 3.1. A direct approach to overcome the CFL condition violation problem is to coarsen the mesh size Δx with the time step Δt. To do so, one also has to define restriction and prolongation in the spatial dimension explicitly. This technique will also be used in our proposed method.

Algorithm 1: parareal

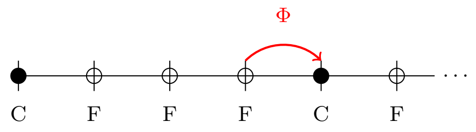
```
In parallel:
  - Solve on fine $y^k_{f,j} = \mathcal{F}(y^k_j, t_j, t_{j+1})$
  - Solve on coarse $y^k_{c,j} = \mathcal{G}(y^k_j, t_j, t_{j+1})$
Sequentially:
  - Update from first time segment to the last
      $y^{k+1}_{j+1} = \mathcal{G}(y^{k+1}_j, t_j, t_{j+1}) + \mathcal{F}(y^k_j, t_j, t_{j+1}) - \mathcal{G}(y^k_j, t_j, t_{j+1})$
```

2.2 Multigrid Reduction-in-Time Method

The multigrid reduction-in-time (MGRIT)[4] method is also a iterative parallel-in-time method. MGRIT is a multigrid method on the time dimension with specified relaxation, restriction, and prolongation methods. Although [4] argues that *parareal* method can be written as a type of MGRIT, the two methods usually have different settings in structures and relaxation methods; therefore, we explain the two methods separately in this paper.



(a) Explicit F-relaxation updates from each C points sequentially to all F points until before the next C point.



(b) Explict C-relaxation updates each C point from its previous F point

Fig. 1: F-relaxation (blue) and C-relaxation (red) for explicit schemes for the MGRIT method.

The MGRIT method chooses coarse time points (C points) from the timeline and constructs the multigrid method on the time dimension. The merit of the MGRIT method is that it can construct more than one layer of coarse time grids and thus leads to more efficient convergence. The MGRIT method was first developed to work with implicit schemes using iterative solver as the multigrid method's relaxation method. The MGRIT was later proven that it also works with explicit schemes with the proper settings.[7]

The MGRIT method first constructs a linear system of equations for all time steps.

$$Au = \begin{bmatrix} I & & & \\ -\Phi & I & & \\ & \ddots & \ddots & \\ & & -\Phi & I \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N_t} \end{bmatrix} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{N_t} \end{bmatrix}$$

Before applying the multigrid method on the time dimension, pick coarse grid points (C points) in the timeline with some ratio m. For every m fine grid point (F points), there is one coarse grid point (C point). Restriction R and prolongation P are then defined to move data between different levels of coarsening.

Like *parareal*, MGRIT requires a coarse operator to propagate error from one C point to the next. Since the CFL condition must be satisfied for explicit time-marching schemes, one must ensure that the coarse operator for the MGRIT method satisfies the CFL condition.

While working with explicit schemes, define F-relaxation as solving all F points sequentially between each C point and defining C-relaxation as solving all C points, as shown in Figure 1. The most used relaxation methods for the MGRIT method are F-relaxation and FCF-relaxation, which is F-relaxation followed by C-relaxation, followed by F-relaxation.

3. Proposed Method

The present work proposes a parallel-in-time method targeting problems solved by explicit time-marching schemes.

Algorithm 2: MGRIT(l)[4]

```
if $l$ is the coarsest level $L$ then
    - Solve coarse grid system $A_L u^{(L)} = g^{(L)}$
else
    - Relax on $A_l u^{(l)} = g^{(l)}$
    - Compute and restrict residual
      $g^{(l+1)} = R_l(g^{(l)} - A_l u^{(l)})$
    - Solve on next level $MGRIT(l+1)$
    - Correct using interpolation $u^{(l)} \leftarrow u^{(l)} + Pu^{(l+1)}$
end
```

The challenges of PinT methods for explicit schemes are first listed in the following subsection, then the proposed method and how it overcomes the challenges will be explained in the next one.

3.1 Challenges in Applying PinT to Explicit Time-Marching Scheme

Despite years of developments for parallel-in-time methods[2], there is yet a PinT method directly targeting explicit schemes. However, for some specific applications, such as supersonic flow simulation, explicit schemes are usually preferred over implicit ones. Due to its high scalability in space, spatial parallelization has been sufficient for many years. Nevertheless, as computing power grows, spatial parallelization has gradually reached the saturation point for parallelization in the spatial dimensions. This motivates us to study PinT methods for explicit schemes despite that it is highly scalable in space. Furthermore, the proposed method in this research can achieve similar parallel-in-time scalability compared to that of spatial parallelization.

A necessary restricting condition while solving with an explicit time-marching scheme is the CFL condition[12]. For example, a one-dimensional advection problem must satisfy the following CFL condition while solved by an explicit scheme.

$$C = \frac{u\Delta t}{\Delta x} \leq C_{max}$$

Where u is the wave velocity, Δx is the mesh size, and Δt is the time step. The Courant number C has to be less than some value C_{max}, which changes by different explicit schemes. C_{max} is determined base on various schemes and discretization methods.

Simple observation tells that the Courant number grows with the size of the time step. Since most parallel-in-time methods involve time-steppings on coarse time grids, as explained previously for *parareal* and MGRIT, it is crucial to be aware of the CFL condition while designing the PinT method for explicit schemes.

3.2 Explicit time-marching optimized parallel-in-time method.

The proposed method targets problems with explicit schemes and aims to achieve high scalability. The proposed method is based on *parareal* but has multiple level structures like the MGRIT method.

First, the algorithm follows the idea from *parareal* and divides the whole timeline into time segments of the number of parallel processors. Because we are working with explicit schemes, we argue that defining the same coarse points despite the number of available processors like the MGRIT method is inefficient. Furthermore, since the relaxation method for explicit time-marching schemes is solved for one step, calculating an FCF-relaxation with coarsening ratio of 2 is more expensive than a direct sequential solve. Therefore, to optimize the performance for the parallel in time method, the relaxation operator is defined as sequential explicit time-marching on the defined coarsening grid within each time interval assigned to the core.

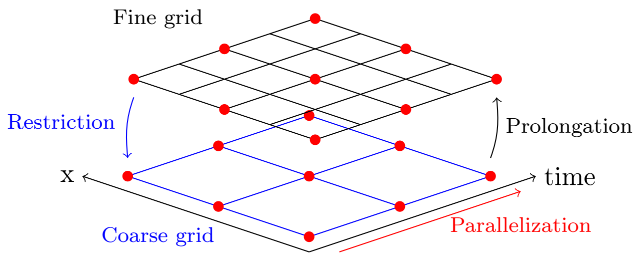


Fig. 2: Out proposed method extract coarse grid by coarsening both the time grid and the x grid.

The spatial grid is coarsened as the same ratio as the time grid, as shown in Figure 2, in order to satisfy the CFL condition for all coarsening levels. CFL condition is satisfied in all coarsening levels if both the time step size Δt and mesh size Δx are coarsened by the same ratio.

$$C = \frac{v\Delta t}{\Delta x}$$
$$C' = \frac{v(r\Delta t)}{r\Delta x} = \frac{v\Delta t}{\Delta x} = C$$

Since the defined relaxation (sequential time-marching within the current time segment) calculates results only from the previous time step, we do not need to define relation and prolongation in the time direction. However, the values on the whole space grid are used for the explicit time-marching schemes. Thus, this method defines restriction and prolongation for the spatial direction. For one-dimension cases, we could directly take every other point as a coarse point, similar to the multigrid method's restriction method. For prolongation, there are several options. The most simple one is, of course, linear interpolation. In previous studies, [13], using linear interpolation as prolongation method has proven to result in slow convergence while working with *parareal*. Therefore, this method defines prolongation as updating average jump values of *parareal*. More details are explained in the Numerical experiment in Sectioin 4.

Different from *parareal*, instead of solving on two levels of grids with fine and coarse solvers accordingly, we construct more than two levels of coarsening grids, each solves with the same explicit schemes but with a different size of the time step. The solvers in each level are called fine solvers, coarse solvers, and the coarsest solvers in the following. The fine solver solves on the whole space grid, while the coarsest solver solves on the least points of all.

The coarsest solver is the baseline accuracy of the result, and therefore, we start by initializing the values with a sequential solve across the timeline with the coarsest solver. Then, from coarse to fine levels, we perform the *parareal* algorithm after prolonging the result from the last level. In each level, we choose the solver on the current level as the high precision solver and the coarsest solver as the low precision solver, which is used to pass on the jumps. This way, for each level of *parareal*, a close-to-result initial value is provided, and the iteration number until convergence is reduced. The summarized algorithm for our method can be found at Algorithm 3.

Algorithm 3: Proposed method

```
Get coarsest values with restriction.
Initialization: sequential time-marching on the coarsest
  level L.
for level $\ell$ = L-1 to 1 do
    for iterate until residual tolerance do
        On current processor:
        Solve on the current level $y_f = \mathcal{F}_\ell(y_j, t_j, t_{j+1})$.
        Solve on the coarsest level $y_c = \mathcal{G}(y_j, t_j, t_{j+1})$.
        for Processor p = 1 to P do
            Solve on the coarsest level
              $y_{j+1}^{k+1} = \mathcal{G}(y_j^{k+1}, t_j, t_{j+1}) + \mathcal{F}_\ell(y_j^k, t_j, t_{j+1}) -$
              $\mathcal{G}(y_j^k, t_j, t_{j+1})$
            Update values to level $\ell$ with prolongation.
        end
    end
end
```

The proposed method looks similar to the MGRIT method but is different in two perspectives. The greatest difference between the proposed method and the MGRIT method is their relaxation method. The explicit MGRIT uses FCF-relaxation, which updates values between three C points, while the proposed method updates sequentially across the whole time interval assigned to each core. The two relaxation methods would be similar given a sufficient amount of cores. However, with a limited number of cores assigned to the time dimension, the proposed method could better reduce the computation time. Secondly, the proposed method does not follow V-cycle as the MGRIT does. Instead, the proposed method solves until convergence in each coarser level before moving to its finer upper level, aiming to reduce the number of iterations on fine levels (ideally to 1). On the downside, the iteration number of the proposed method grows slightly along with the number of cores. Note that the MGRIT method distributes the same computation into available cores. Thus the iteration number of the MGRIT method does not change upon the number of cores.

4. Result

The numerical result demonstrates the convergence and scalability of a one-dimensional advection example and a two-dimensional simulation of compressible viscous subsonic flow around a circular cylinder solving the Navier-Stokes equations.

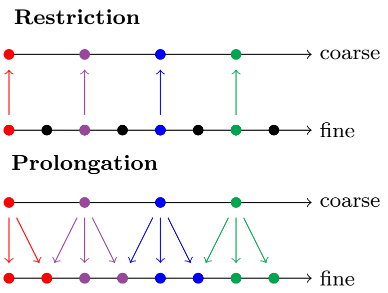


Fig. 3: Data values between fine and coarse levels are passed by restriction and prolongation.

4.1 One-dimensional advection equation

This example is a one-dimensional advection equation example with problem settings the same as [6]. A general one-dimensional advection equation can be written as the following, where c is the speed of the wave.

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x} \tag{1}$$

For convenience, the velocity c is set to 1 in this example. Furthermore, the initial condition is set as $u(x, 0) = \sin^4(\pi x)$, and the boundary is set as a periodic boundary.

A simple and effective explicit scheme is for the advection equation is the Lax-Wendroff method, as described in Equation 2

$$\begin{aligned} u_i^{n+1} = u_i^n &- \frac{\Delta t}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) \\ &+ \frac{\Delta t^2}{2\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + O(\Delta t^3) \end{aligned} \tag{2}$$

Recall that in order to prevent the violation of the CFL condition while solving with an explicit scheme, the present work coarsens the space grid along with the coarsening of the time grid. As shown in Figure 3, restriction and prolongation are defined to pass data values between fine and coarse space grids. Assume that $\{Y_i\}_{i=0}^N$ represents the fine grid values and $\{y_j\}_{j=0}^n$ represents the coarse grid values, where $N = 2n$. Restriction is defined as follows:

$$y_j = Y_{2j} \quad \forall 0 \le j \le n$$

And prolongation is defined as follows:

$$Y_i = \begin{cases} Y_i + \Delta y_j & i = 2j \\ Y_i + (\Delta y_j + \Delta y_{j+1})/2 & i = 2j+1 \end{cases} \quad \forall 0 \le j \le n$$

The following performance result uses tolerance of average residual norm 0.001 for both the proposed method and xbraid MGRIT. The result is generally accurate, but there are some small oscillations near 0 points.

This numerical experiment for a one-dimensional advection example is conducted on the Oakbridge-CX cluster at the University of Tokyo with the described initial state. Oakbridge-CX is an Intel Xeon Platinum 8280 system with

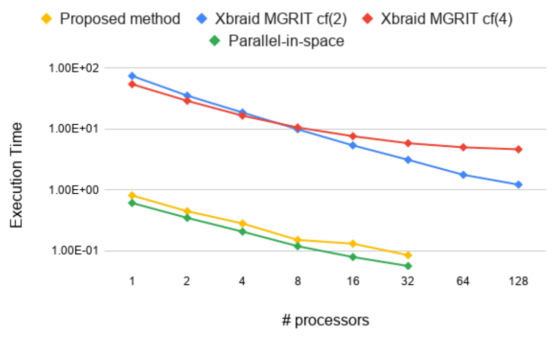


Fig. 4: Execution time for advection equation using 4096 x points and solves for 8192 time steps. The explicit scheme applied for this experiment is ERK3-U3[6]. This figure compares the result of that of parallel-in-space, parallel-in-time(proposed method) ,and MGRIT with xbraid library. The value in cf brackets is the coarse-fine ratio for MGRIT method.

28 CPUs per node and two cores per CPU with a memory size 96 GB and memory bandwidth 101 GB/s.

The most recent PinT result with advection using explicit scheme we could find is in [6] using the MGRIT method with the xbraid library. Figure 4 shows the runtime result of our method compared to that of sequential and MGRIT method by xbraid. The experiment uses the order 3 explicit Runge-Kutta and order 3 upwind scheme (ERK3-U3) in [6] and has 4096 number of x grids, 8192 number of time steps. The xbraid results were tuned with the best parameters as possible as we can with version 3.0.0. The proposed method shows that the parallel-in-time method could be helpful even if one only has a few extra processors to spare.

As shown in Figure 4, the parallel performance of the proposed method is similar to spatial parallelization, and much better compares to xbraid MGRIT. The figure also shows that the proposed method could serve as an efficient PinT method even with a small number of cores. Figure 5 adds the results for parallel in space *and* time using 2 and 4 group of cores in the time dimension accordingly. The number in the brackets denotes the number of cores assigned to the time dimension. In the case of 32 MPI processors, if the number of threads assigned to the time dimension is 2, then the number of threads assigned to the space dimension would be 16, so on so forth.

However, we could also observe that the iteration number increases while the number of processors increases, resulting in choosing the number of time intervals according to the number of available processors. This is different from the MGRIT method, in which iteration number does not depend on the number of processors. The improvement for convergence with a large number of processors is left as future work. From the numerical experiments, we also observe that the residual decreases very fast at first to about $O(10^-3)$, but it takes many iterations to achieve very high precision. The restriction and prolongation operators produce errors that cause slow convergence in the space dimension. Because

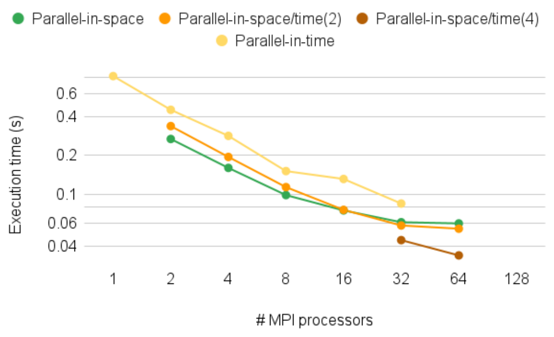


Fig. 5: Execution time for advection equation using 4096 x points and solves for 8192 time steps. The explicit scheme applied for this experiment is ERK3-U3[6]. The number in the barckets denotes the number of threads assigned to the time dimension.

the proposed method update jumps with coarse grid time stepping, the update for the error caused by prolongation is inefficient. How to improve the convergence for the spatial coarsening error is also left as future work.

4.2 Two-dimensional compressible subsonic flow around cylinder

For the two-dimensional example, we choose the simulation of compressible viscous flow around a circular cylinder. A two-dimensional Compressible compressible fluid dynamics (CFD) can be solved by solving the following Navier-Stokes equations.[14]

$$\frac{\partial U}{\partial t} + \bigtriangledown \cdot (F\hat{i} + G\hat{j}) = \bigtriangledown \cdot (R\hat{i} + S\hat{j}) \qquad (3)$$

where the state vector U, convective flux vector F, G and viscous flux vector R, S can be written as following.

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \quad F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ Eu + pu \end{pmatrix}, \quad G = \begin{pmatrix} \rho v \\ \rho v u \\ \rho v^2 + p \\ Ev + pv \end{pmatrix}$$

$$R = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{pmatrix}, S = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - q_y \end{pmatrix}$$

where ρ is density, u, v are velocities in x and y directions, E is total energy, $\tau_x x, \tau_x y, \tau_y y$ are viscous stresses and q_x, q_y are heat fluxes.

The four Navier-Stokes equations can be written in a similar form and can be solved using the same grid partition and explicit scheme. For compressible CFD, we have five unknowns $\rho, u, v, E,$ and P. For a perfect gas, we have Equation 4 to solve pressure P from total Energy E.

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2) \qquad (4)$$

The goal is to solve an initial uniform velocity condition to the right ($+u$ direction) until a steady-state of viscous

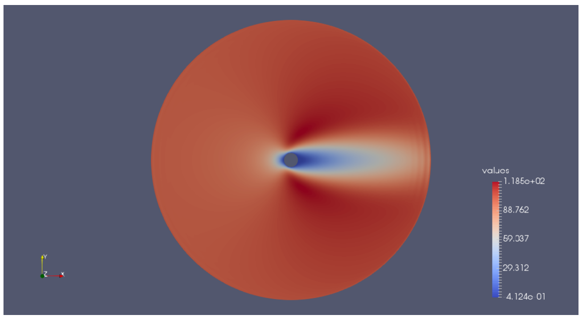
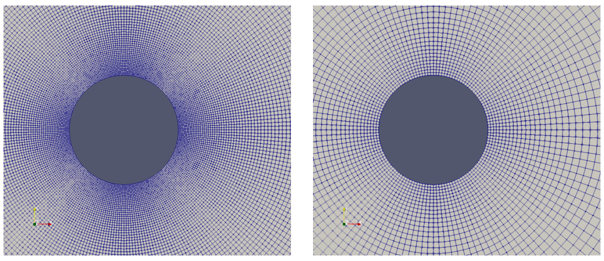


Fig. 6: X-velocity distribution of flow around cylinder stable result with initial velocity 0.3 Mach.



(a) Fine grid with 256 points on the angular coordinate.

(b) Coarse grid with 128 points on the angular coordinate.

Fig. 7: Grid points on different levels.

flow around a cylinder. Figure 6 shows a sample result of a steady-state with initial flow velocity 0.3 Mach to the right, where the color indicates the value of the rightward velocity of the flow.

Grid points are taken on the polar coordinates. Grid points for each coarsening level are defined such that the height is the same as the lower width. Figure 7 shows the grid taken with different solutions.

$$\Delta r = r\Delta\theta$$

The grid points for coarse and fine grids do not match except for the inner boundary. Thus, both restriction and prolongation should be defined by interpolation for the space dimension. We interpolate the values of each coarse and fine grid from its closest grid mesh Ω using bicubic interpolation[8]. Bicubic interpolation approximates values in a grid mesh using a cubic equation of x and y.

$$f(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^{i}y^{j} \quad \forall(x,y)\in\Omega$$

The parameters a_{ij} of the cubic is solved with values $f(x,y)$ and derivatives f_x, f_y, f_{xy} on the four corners of the mesh. The solved cubic function is then used to derive internal points. In this experiment, both restriction and prolongation use bicubic interpolation on polar coordinate to get coarse and fine grid point values.

For the explicit scheme, we rewrite Equation 2 into a two-dimensional version on polar coordinates.

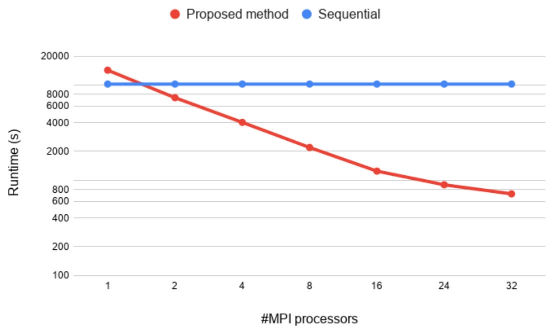


Fig. 8: MPI runtime comparison of 2D CFD flow simulation example with proposed parallel-in time method

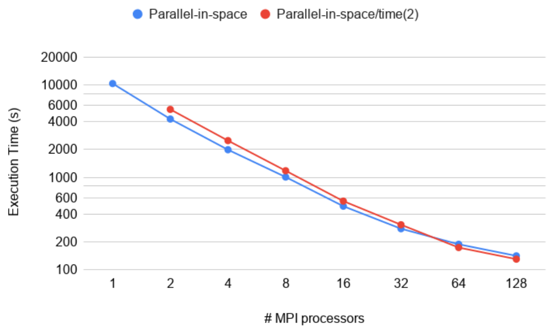


Fig. 9: Two-dimensional compressible flow execution time with 32000 grid points and solves for 100000 time steps. The blue line parallels only in the space dimension. The red line divide 2 time intervals and the and the rest of the cores are used to parallel in the space dimension.

$$\begin{aligned}
&U^{n+1} = U^n + \left(\frac{\partial U}{\partial t}\right)^n \Delta t + \frac{1}{2}\left(\frac{\partial^2 U}{\partial t^2}\right)^n \Delta t^2 \\
&\left(\frac{\partial U}{\partial t}\right)^n = \nabla\cdot(\vec{R}-\vec{F}), \quad \left(\frac{\partial^2 U}{\partial t^2}\right)^n = \nabla^2(\vec{R}-\vec{F}) \\
&\nabla\cdot\vec{F} = \frac{(\vec{F}_{i+1,j}-\vec{F}_{i-1,j})}{2\Delta r} + \frac{(\vec{F}_{i,j+1}-\vec{F}_{i,j}+\vec{F}_{i,j}-\vec{F}_{i,j-1})}{2r\Delta\theta} \\
&\nabla^2\vec{F} = \frac{(\vec{F}_{i+1,j}-2\vec{F}_{i,j}+\vec{F}_{i-1,j})}{\Delta r^2} + \frac{(\vec{F}_{i,j+1}-2\vec{F}_{i,j}+\vec{F}_{i,j-1})}{r^2\Delta\theta^2}
\end{aligned} \tag{5}$$

where $\vec{F} = F\hat{i} + G\hat{j}$, $\vec{R} = R\hat{i} + S\hat{j}$.

With this explicit two-dimensional scheme, the experiment compares the runtime for spatial parallelization and parallel-in-time. For this numerical simulation, the problem has 32,000 space grid points and computes for 100,000 time steps. This experiment is also conducted on the Oakbridge-CX cluster at the University of Tokyo. We can see from Figure 8 that our method could achieve faster runtime for more than two MPI processes. Similar to the last example, we tested the performance of parallel-in-space/time with the proposed method in Figure 9.

5. Discussion

The parallel-in-time method requires more computations compare to the spatial parallelization of the explicit scheme. Therefore, it might seem odd that the proposed method could achieve similar scalability compared to spatial parallelization. This section provides a breakdown analysis of the computation cost, communication cost, and the number of synchronizations for both parallel-in-space and parallel-in-time (proposed method).

Assume that the target problem has N_x grid points (fine grid), N_t number of time steps, and the target machine has P parallel processors. Suppose that one sequential explicit time-marching without parallelization costs T_{ex} computations. Generally, with P processors, the computation complexity for spatial parallelization would be

$$N_t \times \frac{1}{P} T_{ex}$$

For the proposed method, suppose that there are L levels, with the coarsening ratio of 2 between each adjacent level. The iteration number in each level is $m_l,\ l = 1, \ldots, L-1$. The computation complexity for the proposed method would be.

$$\frac{1}{P}(m_{L-1} \times \frac{1}{2^{n-1}} T_{ex} + \ldots + m_1 \times T_{ex}) + \frac{1}{2^L} T_{ex} \times (1 + m_{L-1} + \ldots + m_1)$$

The first term is for the parallelization part, and the second term is for the sequential part. According to the numerical experiment, we can reduce the m_l to nearly one or two and still get decent results; therefore, the difference in computation cost is not significant.

Assume that for spatial parallelization, the explicit scheme exchanges C_{ex} data with its neighbor processors. The communication cost for spatial communication cost could be written as

$$N_t \times C_{ex} P$$

For the proposed method, the communication cost is as follows

$$(m_1 \frac{1}{2} N_x + m_2 \frac{1}{2^2} N_x + \ldots + m_{L-1} \frac{1}{2^n} N_x) \times (P-1)$$

The communication cost is also similar when the number of grids N_x is similar to the number of time steps N_t. Another critical factor for parallel computation is the number of synchronizations. For spatial parallelization, the number of synchronization is the number of time steps

$$N_t$$

For the proposed method, the number of synchronization equals the summary of iteration numbers.

$$\sum_{l=1}^{L-1} m_l \leq LP$$

Since the number of processors bounds the iteration number in each level, we have an upper bound LP for the number of synchronization, which is much smaller than N_t for a large problem.

The present work presented two examples with very different computation complexity. According to the above analysis, the proposed method could achieve better performance when synchronization time is larger than the computation complexity difference. Therefore, if the fine grid iteration number could be successfully reduced, the parallel performance of parallel-in-time could be similar to that of parallel-in-space. Moreover, combining parallel-in-space and parallel-in-time might be faster than that of pure parallel-in-space with sufficient number of cores, which fits the results we observe from Figure 5 and Figure 9.

6. Conclusion and Future Work

This paper proposed a parallel-in-time method that is optimized for explicit time-marching schemes. The proposed method is mainly based on *parareal* but has many layer hierarchies like MGRIT. For a one-dimensional advection example and a two-dimensional compressible CFD simulation example, the proposed method has been proven to converge even with a small number of iterations. Also, the proposed has shown that with enough processors, combining parallel-in-space and time could provides better performance than that of pure spatial parallelization.

Despite the high scalability and convergence on both examples, the proposed method has a problem that iteration number grows slightly with the number of processors. Also, as the error analysis result shows, the discretization error caused by restriction and prolongation of the proposed method converges slowly. Therefore, the fast converging result, as shown in the result section, could only serve as a fast approximation of the original problem. More iteration is required for high precision results. At last, the proposed method has not yet provided a successful result for the compressible CFD with shock wave results (supersonic flow) due to the stability problem in the coarse layers.

In future work, the convergence of the method should be improved such that the discretization error could be reduced more efficiently. Secondly, the algorithm should be further adjusted in order to solve CFD problems with shock waves. We would also like to challenge applying parallel-in-time for three-dimensional examples and large-scale CFD applications.

References

[1] XBraid: Parallel multigrid in time. http://llnl.gov/casc/xbraid

[2] M. J. Gander, "50 years of time parallel time integration," in *Multiple shooting and time domain decomposition methods*. Springer, 2015, pp. 69–113.

[3] J. L. Lions, Y. Maday, and G. Turinici, "Résolution d'EDP par un schéma en temps «pararéel »," *Comptes Rendus de l'Academie des Sciences - Series I: Mathematics*, vol. 332, no. 7, pp. 661–668, 2001.

[4] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, "Parallel time integration with multigrid," *SIAM Journal on Scientific Computing*, vol. 36, no. 6, pp. C635–C661, 2014.

[5] O. A. Krzysik, H. De Sterck, S. P. MacLachlan, and S. Friedhoff, "On selecting coarse-grid operators for parareal and mgrit applied to linear advection," *arXiv preprint arXiv:1902.07757*, 2019.

[6] H. De Sterck, R. D. Falgout, S. Friedhoff, O. A. Krzysik, and S. P. MacLachlan, "Optimizing mgrit and parareal

coarse-grid operators for linear advection," *arXiv preprint arXiv:1910.03726*, 2019.

[7] A. J. Howse, H. D. Sterck, R. D. Falgout, S. MacLachlan, and J. Schroder, "Parallel-in-time multigrid with adaptive spatial coarsening for the linear advection and inviscid burgers equations," *SIAM Journal on Scientific Computing*, vol. 41, no. 1, pp. A538–A565, 2019.

[8] Keys, R.: Cubic convolution interpolation for digital image processing. IEEE Transactions on Acoustics, Speech, and Signal Processing **29**(6), 1153–1160 (1981). DOI: 10.1109/TASSP.1981.1163711

[9] J. Christopher, R. D. Falgout, J. B. Schroder, S. M. Guzik, and X. Gao, "A space-time parallel algorithm with adaptive mesh refinement for computational fluid dynamics," *Computing and Visualization in Science*, vol. 23, no. 1, pp. 1–20, 2020.

[10] R. D. Falgout, A. Katz, T. V. Kolev, J. B. Schroder, A. Wissink, and U. M. Yang, "Parallel time integration with multigrid reduction for a compressible fluid dynamics application," *Lawrence Livermore National Laboratory Technical Report, LLNL-JRNL-663416*, 2015.

[11] M. von Danwitz, V. Karyofylli, N. Hosters, and M. Behr, "Simplex space-time meshes in compressible flow simulations," *International Journal for Numerical Methods in Fluids*, vol. 91, no. 1, pp. 29–48, 2019.

[12] H. Lewy, K. Friedrichs, and R. Courant, "Über die partiellen differenzengleichungen der mathematischen physik," *Mathematische annalen*, vol. 100, pp. 32–74, 1928.

[13] D. Ruprecht, "Convergence of parareal with spatial coarsening," *PAMM*, vol. 14, no. 1, pp. 1031–1034, 2014.

[14] V. Parthasarathy, Y. Kallinderis, and K. Nakajima, "Hybrid adaptation method and directional viscous multigrid with prismatic-tetrahedral meshes," in *33rd Aerospace Sciences Meeting and Exhibit*, 1995, p. 670.

[15] Cortes Garcia, I., Kulchytska-Ruchka, I., Clemens, M., Schöps, S.: Parallel-in-time solution of eddy current problems using implicit and explicit time-stepping methods. arXiv e-prints pp. arXiv–2012 (2020)