Vol.2022-HPC-185 No.25
2022/7/29

A Parallel-in-Time Method for Compressible Fluid Explicit Simulation

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

Abstract: High-Performance Computing (HPC) methods for time-dependent problems reach an acceleration in space as the problem size grows. This restriction leads to the development of parallel-in-time (PinT) methods. Despite many PinT methods have been introduced, very few PinT methods have been tested with explicit schemes to our best knowledge. This research introduces a PinT method that works with explicit schemes. This research constructs a multi-layer hierarchy in time and space and solves it through coarse to fine layers. The proposed Cascadic Parareal method is optimized based on the number of available cores to improve the efficiency of parallel-in-time solvers with a limited number of processors. The numerical experiment solves for a compressible fluid simulation around a cylinder. The research result shows that the proposed parallel-in-space/time (PinST) method could solve faster than traditional spatial parallelization with less than 100 cores.

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

1. Introduction

Parallel-in-Time (PinT) studies have received growing attention recently. PinT methods are methods that were used to achieve further parallel acceleration after spatial parallelization saturates. So far, many PinT methods have been proposed[1]. PinT methods such as parareal[2] and MGRIT[3], [5] are popular PinT methods that are commonly used. These methods have been shown to work on various applications from simple equations like Poisson's equation and advection equation to computational fluid dynamics simulation[4], [5], [9], [10], [11]. However, it is also observed that PinT methods with explicit time-marching schemes converge slower and require a large number of cores to achieve faster computation time[6]. Cascadic Parareal[7] is a PinT method developed to optimize the parallel efficiency of PinT methods with explicit time-marching schemes.

Computational Fluid Dynamics (CFD) has been a challenging application for PinT methods. CFD simulations involve solving multiple differential equations and variables. One has to solve four governing equations by finite volume method (FVM), for example, for the compressible fluid simulation. The instability of CFD with coarse meshes makes it even harder to apply PinT methods. So far the following works[6], [9], [10], [11], [17] have PinT methods on different CFD applications. Falgout et al.(2015)[10] achieves 7.5 times computation acceleration with 4096 cores for a compressible fluid simulation by MGRIT. Howse et al.(2019) achieve computation faster than spatial parallelization with 4096 to 16384 cores for the Burger's equation. Christopher et al.(2019) shows the convergence of MGRIT method for Transient Couette flow application. These researches all require a large number of cores for PinT acceleration and cannot compete with the parallel efficiency of spatial parallelization for explicit methods.

Cascadic Parareal shows better parallel efficiency than the existing PinT method with explicit time-stepping schemes. This paper will show that Cascadic Parareal converges well for subsonic compressible flow. Moreover, the accuracy for supersonic flow can be improved by applying adaptive mesh refinement on coarse meshes. Details will be explained in the result section.

Existing method *parareal* will be overviewed in section 2. Based on *parareal*, section 3 explains the Cascadic Parareal method. An experimental result for subsonic compressible flow will then be provided in section 4. Section 5 discusses the convergence of supersonic flow and how to improve its accuracy. Finally, the conclusion and future works are summarized in Section 6.

2. Pararael

The Cascadic Parareal method is based on the *parareal* method. This section introduces the *parareal* method and the Cascadic Parareal method will be explained in the next section.

2.1 Parareal Method

Parareal[2] is an iterative PinT method with two levels of time resolution. Consider a general PDE.

[1] The University of Tokyo, Bunkyo-ku, Tokyo 133–8656, Japan
[a)] chen-yenchen842@g.ecc.u-tokyo.ac.jp
[b)] nakajima@cc.u-tokyo.ac.jp

$$\frac{\partial u}{\partial t} = f(u,t) \quad ,t \in [0,T] \tag{1}$$

Parareal solves in parallel on time subintervals $[T^n, T^{n+1}]$, $n = 0, 1, 2, \ldots, N-1$. Within each time subinterval, the PDE is solved by a fine solver $\mathcal{F}_{dt}(u_n^k, T^n, T^{n+1})$ in parallel.

$$u_{n+1} = \mathcal{F}_{dt}(u_n, T^n, T^{n+1}) \tag{2}$$

Because the time subintervals are solved independently, the values at T^n are discontinuous. The Errors at T^n are propagated from the start time by a coarse solver $\mathcal{G}_{\Delta T}(u_n, T^n, T^{n+1})$, where $\Delta T = T^{n-1} - T^n$. Repeating the previous steps iteratively is the *parareal* method. The general equation of *parareal* can be written as follows.

$$\begin{aligned} u_{n+1}^{k+1} =& \mathcal{G}_{\Delta T}(u_n^{k+1}, T^n, T^{n+1}) \\ &+\mathcal{F}_{dt}(u_n^k, T^n, T^{n+1}) \\ &-\mathcal{G}_{\Delta T}(u_n^k, T^n, T^{n+1}) \end{aligned} \tag{3}$$

where fine solvers $\mathcal{F}$ are solved in parallel and coarse solvers $\mathcal{G}$ are solved sequentially. Algorithm 1 summarizes the parareal method.

The fine and coarse solvers in parareal can be any solvers for the time-dependent problem, either explicit or implicit. This research focuses on explicit methods for all fine and coarse solvers.

Algorithm 1: parareal

```
- Initialize with coarse solver $u_{n+1} = \mathcal{G}_{\Delta T}(u_n, T^n, T^{n+1})$
for iterations $k = 0, 1, \ldots$ do
    In parallel:
    - Solve with the fine solver $\mathcal{F}_{dt}(u_n, T^n, T^{n+1})$
    - Result from last iteration $\mathcal{G}_{\Delta T}(u_n, T^n, T^{n+1})$
    Sequentially:
    - Update from the first time segment to the last
    $u_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(u_n^{k+1}, T^n, T^{n+1}) + \mathcal{F}_{dt}(u_n^k, T^n, T^{n+1}) -$
      $\mathcal{G}_{\Delta T}(u_n^k, T^n, T^{n+1})$
end
```

2.2 Spatial Coarsening

When solving a time-dependent problem with an explicit method, it is essential that the time step satisfies the Courant-Friedrichs-Lewy (CFL) condition[12]. The CFL condition is a necessary condition of stability for explicit methods while solving certain PDEs. For example, an advection equation $\frac{\partial u}{\partial t} = -a\frac{\partial u}{\partial x}$ has to satisify the following CFL condition to solve stably with an explicit method.

$$C = \frac{a\Delta t}{\Delta x} \leq C_{\max} \tag{4}$$

where a is the flow speed, Δt is the time step size, and Δx is the mesh size. $C_{\max}$ is the maximum courant number, which depends on the explicit method.

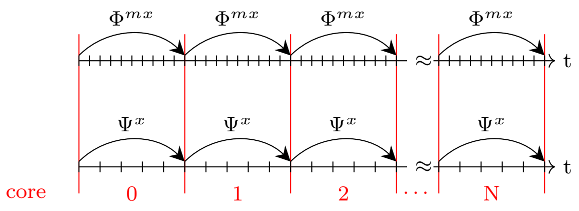

Fig. 1: The whole time interval is divided by the number of cores N. The coarse time step Δt is defined by coarsening from the fine time grid. The coarse and fine solver solves sequentially in each time subinterval.

Because most PinT methods involves coarse time-steppings (solvers with larger time step size), CFL condition is important for PinT methods with explicit methods. Spatial coarsening is a commonly used technique for explicit PinT methods to make sure that the results with coarse time steps are stable. When the mesh size is coarsened by the same ratio as the time step, the courant number remains the same. Therefore, suppose that the same explicit method is used for all PinT levels, the coarse time-stepping will be stable if the fine time-stepping is stable.

$$C_{coarse} = a\frac{\Delta T}{\Delta X} = a\frac{mdt}{mdx} = a\frac{dt}{dx} \leq C_{\max} \tag{5}$$

3. Cascadic Parareal

For a conventional *parareal* implementation, cores of the number of the coarse time steps are used to give the best efficiency of the parallelization. However, Cascadic Parareal maximizes the parallel efficiency by dividing the time subintervals base on the available cores. Unlike implicit schemes, which are often solved by iterative methods, explicit schemes solve for each time step directly. Thus, we argue that instead of distributing all works of a predefined algorithm to available cores, it is more efficient to divide the time dimension by the number of available cores N. Each core solves sequentially on both coarse and fine grids in the assigned time interval ($[T^n, T^{n+1}]$, $n = 0, 1, \ldots, N-1$) so that the limited computational resource is best used. Assume that there are m time steps in each time interval, for some explicit fine solver Φ and coarse solver Ψ, equation (3) is written as follows.

$$u_{n+1}^{k+1} = \Psi^x(u_n^{k+1}) + \Phi^{mx}(u_n^k) - \Psi^x(u_n^k) \tag{6}$$

where x is the number of coarse time steps in each time subinterval. Because the coarse/fine ratio is m, there are mx fine steps in each time subinterval and the total time step $T = mNx$.

To satisify the CFL condition, spatial coarsening is applied. Coarse solvers for parareal are performed on coarse meshes, as shown in Figure 2. Coarse mesh U is coarsened from the original fine mesh u and restriction operator R and prolongation operator P are defined to pass data between different mesh resolutions.

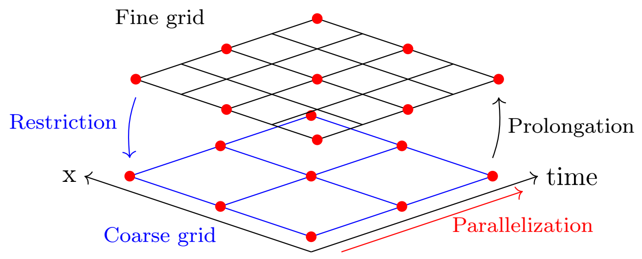

Fig. 2: Spatial coarsening stabilizes the coarse grid result by coarsening both the time grid and the x grid.

$$U = Ru, \quad u = PU \tag{7}$$

The equation (6) with spatial coarsening can be written as follows.

$$u_{n+1}^{k+1} = P\Psi^x(U_n^{k+1}) + \Phi^{mx}(u_n^k) - P\Psi^x(U_n^k) \tag{8}$$

Cascadic Parareal applies the cascadic multigrid[14] method on these time meshes using *parareal*, and always use the coarsest mesh as the coarse solver. The cascadic multigrid method is a multigrid method that only moves from coarse to fine grids. The *parareal* method is iterated at each multigrid level until the error is reduced to the same order as the discretization error, and then the converged result is moved to the next finer level.

Because Cascadic Parareal uses more than two levels of refinement, we define results at the spatial grid on level ℓ as follows, where L levels $\ell = 0, 1, \ldots, L-1$ are from fine to coarse.

$$\begin{aligned} u_\ell = R_{\ell-1}^\ell u_{\ell-1}, \quad u_{\ell-1} =& P_\ell^{\ell-1} u_\ell, \\ & \forall \ell = 1, 2, \ldots, L-1 \end{aligned} \tag{9}$$

where $u_0 = u$ is the finest grid and $u_{L-1} = U$ is the coarsest grid. $R_{\ell-1}^\ell$ is the restriction operation that interpolates from the grid of level $\ell - 1$ to that of level ℓ and $P_\ell^{\ell-1}$ is the prolongation operation that interpolates from the grid of level ℓ to that of level $\ell - 1$.

Cascadic Parareal uses the coarsest level U as the coarse solver for the parareal algorithm at each level. Therefore, we can write the Cascadic Parareal algorithm at level ℓ and at iteration k_ℓ as follows.

$$\begin{aligned} u_{\ell,n+1}^{k_\ell+1} =& P_{L-1}^\ell \Psi^x(U_n^{k_\ell+1}) \\ & + \Phi_\ell^{m^{L-1-\ell}x}(u_{\ell,n}^{k_\ell}) \\ & - P_{L-1}^\ell \Psi^x(U_n^{k_\ell}) \\ \forall \ell = L-&2, L-3, \ldots, 0 \end{aligned} \tag{10}$$

The converged result at coarser level ℓ are prolonged to the next finer level $\ell - 1$ as an initial guess of the parareal algorithm in the next level.

$$u_{\ell-1,n}^0 = P_\ell^{\ell-1} u_{\ell,n}^{k_\ell} \quad \forall n = 0, 1, \ldots, N-1 \tag{11}$$

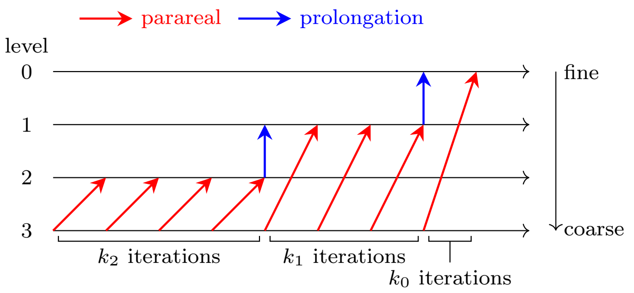

Fig. 3: Cascadic Parareal solves from coarse levels to fine levels. The result is prolonged into a finer level as an initial guess after convergence of the parareal method at the current level. Each red arrow represents an iteration of the parareal algorithm between the two levels. Each blue arrow represents the prolongation operator that gets the initial guess for the finer level.

suppose that the parareal algorithm converges at level ℓ with k_ℓ iterations.

Different from *parareal*, instead of solving on two levels of grids with fine and coarse solvers accordingly, Cascadic Parareal construct more than two levels of coarsening grids, each solves with the same explicit schemes but with a different size of the time step. Figure 3 shows the solving process of Cascadic Parareal of three levels. The red arrows indicates a iteration of *parareal* with the head as fine level and the tail as coarse level. After convergence at each coarse level ℓ, the blue arrow interpolates the converged values to the upper level $\ell - 1$ as an initial guess for the next level of *parareal*. The cascadic multigrid structure enables the Cascadic Parareal method to reduce the required iteration number at the finest level as much as possible. Therefore achieve better parallel efficiency. Algorithm 2 is the summary of Cascadic Parareal.

Algorithm 2: Cascadic Parareal

```
Explicit time-marching on the coarsest level L.
$U_n = R_0^{L-1} u_n$ for $n = 0, \ldots, N$
$U_{n+1} = \Psi^x(U_n)$ for $n = 0, \ldots, N-1$
for level l = L-2 to 0 do
    Take initial values from level $\ell + 1$.
    $u_\ell = P_{\ell+1}^\ell u_{\ell+1}$
    for iterate until residual tolerance do
        On current processor:
        Solve on the current level $u_{n+1} = \Phi^{mx}(u_n)$.
        for Processor p = 1 to P do
            Solve on the coarsest level
            $u_{\ell,n+1}^{k_\ell+1} = P_{L-1}^\ell \Psi^x(U_n^{k_\ell+1}) +$
            $\Phi_\ell^{m^{L-1-\ell}x}(u_{\ell,n}^{k_\ell}) - P_{L-1}^\ell \Psi^x(U_n^{k_\ell})$
            Update values to level 1 with prolongation
        end
    end
end
```

4. Result

A two-dimensional simulation of the compressible fluid flow around a cylinder is used as a numerical example for Cascadic Parareal.

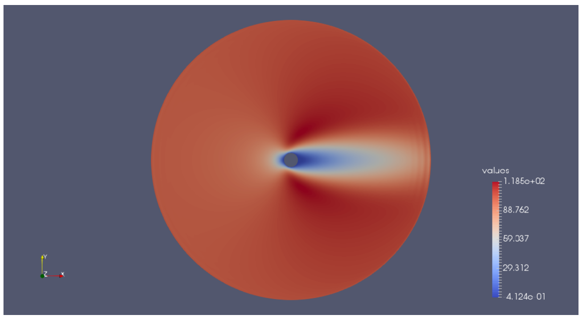

Fig. 4: X-velocity distribution of flow around cylinder stable result with initial velocity 0.3 Mach.

4.1 Two-dimensional compressible subsonic flow around cylinder

This experiment solves the compressible flow from a uniform velocity to a steady state around a cylinder obstacle without perturbation. Figure 4 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. A two-dimensional compressible compressible fluid dynamics (CFD) can be solved by solving the following governing equations.[15]

$$\frac{\partial U}{\partial t} + \bigtriangledown \cdot (F\hat{i} + G\hat{j}) = \bigtriangledown \cdot (R\hat{i} + S\hat{j} \qquad (12)$$

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

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

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

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.

There are five unknowns ρ, u, v, E, and P. For a perfect gas, equation 13 is included to solve with the four governing equations.

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2) \qquad (13)$$

The four coupled equations (12) are solved with the finite volume method (FVM)[15] with artificial dissipation to improve its stability[16]. FVM solves on dual meshes Ω_N with each grid point N in the center.

$$\int_{\Omega_N} \left(\frac{\partial U}{\partial t}\right) d\Omega + \int_{\Omega_N} \left(\bigtriangledown \cdot \vec{F} - \bigtriangledown \cdot \vec{R}\right) d\Omega = 0 \qquad (14)$$

Two-dimensional Lax-Wendroff method is chosen as the explicit method to solve the governing equations.

$$\begin{aligned} U^{n+1} &= U^n + \left(\frac{\partial U}{\partial t}\right)^n \Delta t + \frac{1}{2}\left(\frac{\partial^2 U}{\partial t^2}\right)^n \Delta t^2 \\ \left(\frac{\partial U}{\partial t}\right)^n &= \bigtriangledown \cdot (\vec{R} - \vec{F}), \quad \left(\frac{\partial^2 U}{\partial t^2}\right)^n = \bigtriangledown^2(\vec{R} - \vec{F}) \\ \bigtriangledown \cdot \vec{F} &= \frac{(\vec{F}_{i+1,j} - \vec{F}_{i-1,j})}{2\Delta r} + \frac{(\vec{F}_{i,j+1} - \vec{F}_{i,j} + \vec{F}_{i,j} - \vec{F}_{i,j-1})}{2r\Delta\theta} \\ \bigtriangledown^2 \vec{F} &= \frac{(\vec{F}_{i+1,j} - 2\vec{F}_{i,j} + \vec{F}_{i-1,j})}{\Delta r^2} + \frac{(\vec{F}_{i,j+1} - 2\vec{F}_{i,j} + \vec{F}_{i,j-1})}{r^2\Delta\theta^2} \end{aligned} \qquad (15)$$

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

Second-order and fourth-order artificial dissipations are added to the governing equations with a pressure switch $\triangle P$.

$$\begin{aligned} \triangle p &= \frac{\sum_{i=1}^{n} p_i - p_0}{\sum_{i=1}^{n} p_i + p_0} \\ U &= U + \triangle p \epsilon_2 U^{(2)} + (1 - \triangle p)\epsilon_4 U^{(4)} \end{aligned} \qquad (16)$$

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

Both restriction and prolongation should be defined by interpolation for the space dimension. Spatial restriction and prolongation are performed using bicubic interpolation[8]. Bicubic interpolation approximates a value point in a grid mesh using a cubic equation of r and θ.

$$f(r, \theta) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} r^i \theta^j \quad \forall (r, \theta) \in \Omega \qquad (17)$$

The parameters a_{ij} of the cubic is solved with values $f(r, \theta)$ and derivatives $f_r, f_\theta, f_{r\theta}$ on the four corners of the mesh Ω. 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.

With this explicit two-dimensional scheme, the experiment compares the runtime for spatial parallelization and parallel-in-space/time. For this numerical simulation, the problem has 32,000 space grid points and computes for 100,000 time steps. This experiment is conducted on the Oakbridge-CX cluster at the University of Tokyo. Figure 5 shows the execution time comparison of Cascadic Parareal to spatial parallelization. The figure shows that even with only 2 time subintervals, Cascadic Parareal can execute faster than spatial parallelization after 64 cores. This result requires much less cores than existing researches on CFD simulations.[6], [9], [10]

5. Supersonic Flow

5.1 Convergence of Supersonic flow

Supersonic flow is nonlinear and creates shock wave in front of the cylinder. Figure 6 shows an example of supersonic flow around a cylinder. Table 1 shows the finest converge iteration of Cascadic Parareal for compressible flow of different flow speed. The table shows that Cascadic Parareal does not converge well for supersonic flow.

The diverged Cascadic Parareal results for supersonic flow are due to unstable solving at coarse levels. The values around shock wave is unstable on coarse meshes. One simple

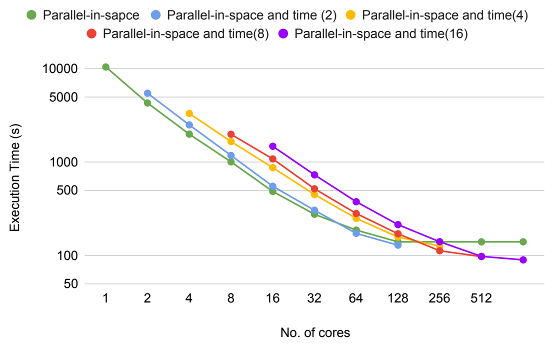

Fig. 5: Execution time of the CFD example with 32,000 grid points and solves for 100,000 time steps.The blue curve represents parallelization only in the space dimension. The parallel-in-space and time result used 2 to 16 cores in time the domain ($P_t = 2, 4, 8, 16$).

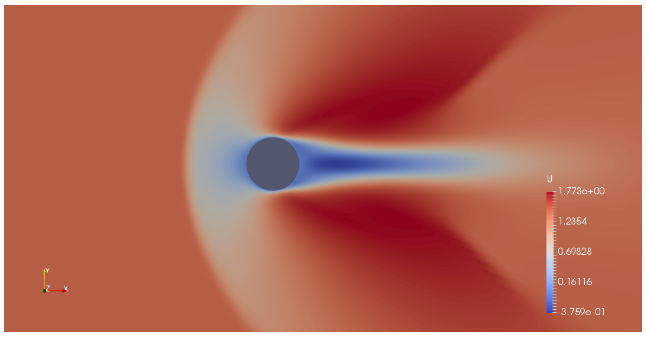

Fig. 6: Result of compressible fluid flow around a cylinder with shock wave. The flow speed is 1.4 Mach.

Table 1: Convergence of parallel-in-time algorithm for 2D compressible flow example using 64 cores. (number only shows the iteration number at the finest level, X means diverge)

Flow speed (Mach)	0.3	0.7	1.0	1.4	1.7
500000 time steps	3	4	5	6	X
1000000 time steps	3	4	5	5+	X
2000000 time steps	3+	3+	3+	X	X

solution to improve the stability is to increase the artificial dissipation coefficients for coarse meshes. However, the converged Cascadic Parareal, affect by the coarse level, shows smooth values around the shock wave, as shown in Figure 7.

This paper consider using adaptive mesh refinement at coarse grids to improve the accuracy of Cascadic Parareal as another solution for the convergence problem.

5.2 Cascadic Parareal with AMR

Adaptive mesh refinement (AMR) is a mesh refinement method that was used to reduce the computation for large-scale computations with non-uniform smoothness. On the contrary, AMR can improve CFD stability for supersonic flow by refining near the shock wave and the inner boundary. Figure 8 is an example of sequential solving with AMR around the cylinder.

Define AMR coarse mesh $\tilde{U}$ and Prolongation operator

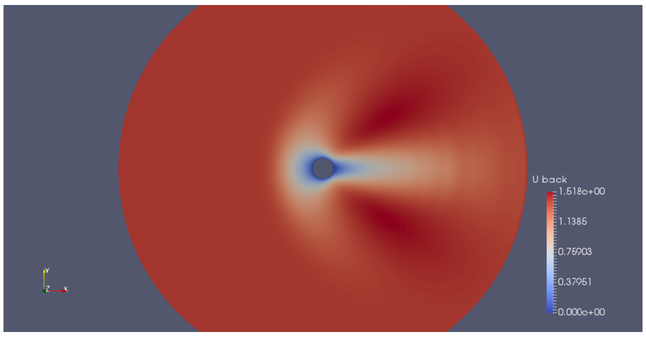

Fig. 7: Cascadic Parareal result of supersonic flow with large dissipation. The flow speed is 1.4 Mach.

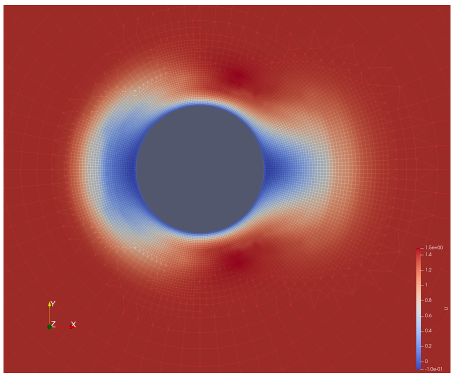

Fig. 8: Adaptive mesh refinement refines smaller meshes where value gradients are large.

for AMR mesh $\tilde{P}$, The Cascadic Parareal equation (10) can be written as follows.

$$\begin{aligned}
&\tilde{U}_{\ell,n} = AMR_{\min(\ell,j)}(U_n^{k_\ell=0}) \\
&u^1_{\ell,n+1} = \tilde{P}^\ell \tilde{\Psi}^x(\tilde{U}^1_{\ell,n}) + \Phi_\ell^{m^{L-1-\ell}x}(u^0_{\ell,n}) - \tilde{P}^\ell \tilde{\Psi}^x(\tilde{U}_{\ell,n}) \\
&u^{k_\ell+1}_{\ell,n+1} = \tilde{P}^\ell \tilde{\Psi}^x(\tilde{U}^{k_\ell+1}_{\ell,n}) + \Phi_\ell^{m^{L-1-\ell}x}(u^{k_\ell}_{\ell,n}) - \tilde{P}^\ell \tilde{\Psi}^x(\tilde{U}^{k_\ell}_{\ell,n}) \\
&\forall k_\ell = 1, 2, \ldots, k_{\ell_{\max}}
\end{aligned} \tag{18}$$

Because we are solving with explicit methods, the solver on the AMR-refined mesh $\tilde{\Psi}$ also has to satisfy the CFL condition. For smaller cells in the AMR mesh, we apply an explicit method with smaller time steps for more time steps. Suppose that a cell $\square$ is divided into four smaller cells $\boxplus$ by AMR. The mesh size h in x and y directions are both reduced to half. Therefore we apply an explicit time-marching scheme twice with the half time step on these smaller cells $2\phi_{\Delta t/2}$. Fig. 9 shows the explicit solving on a one-dimensional non-uniformed grid. When doing spatial parallelization, we have to keep in mind that small cells do two times more computation, or the load balancing will be bad. The same goes for even smaller cells refined by AMR. Suppose that the AMR-refined mesh $\tilde{U}$ has points $\tilde{U}_i$, we write the solver $\tilde{\Psi}$ as follows.

$$\tilde{\Psi}_{\Delta T}(\tilde{U}) = \begin{cases} \Psi^2_{\frac{\Delta T}{2}}(\tilde{U}_i) & i \text{ is in a refined cell} \\ \Psi_{\Delta T}(\tilde{U}_i) & \text{otherwise} \end{cases} \tag{19}$$

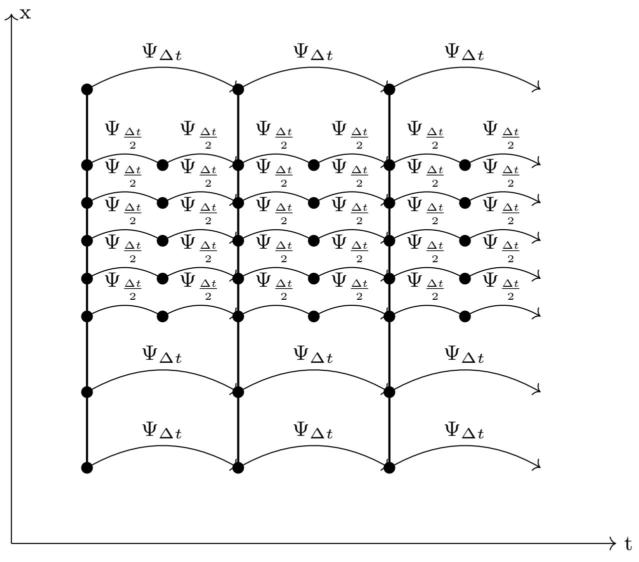

Fig. 9: Smaller time steps are applied on smaller cells on the AMR mesh.

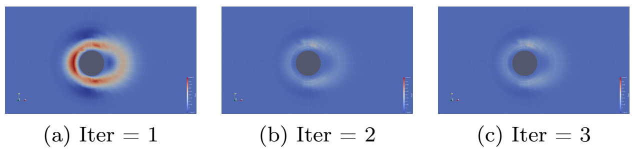

Fig. 10: Error of two-level Cascadic Parareal with double dissipation at uniform coarser level.

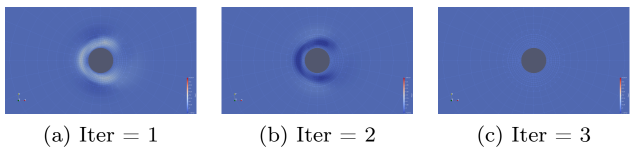

Fig. 11: Error of two-level Cascadic Parareal with AMR coarse level. The dissipation coefficient remains the same.

Consider a two level Cascadic Parareal with 4 time intervals. Figure 10 and Figure 11 are the values of the Cascadic Parareal results without and with AMR at each iteration substracted by the last iteration. The supersonic flow problem size is 2112 grid points with 2000 time steps. For Cascadic Parareal without AMR, two times larger dissipation is applied on the coarse level to improve stability. The figures shows that not only does Cascadic Parareal with AMR converges faster, the converged result is also more accurate.

However, because AMR coarse mesh computes with smaller time steps at refined meshes, AMR solver is naturally more expensive than the uniform coarse solver. Figure 12 compares the computation time of Cascadic Parareal with and without AMR coarse grid. Although Cascadic Parareal converges faster as shown in Table 2, the total computation time is larger than Cascadic Parareal with uniform coarse grid.

For supersonic flow simulation that requires higher accuracy, we use AMR coarse grid for Cascadic Parareal. For smooth functions, Cascadic Parareal with uniform coarse grid has the best parallel efficiency.

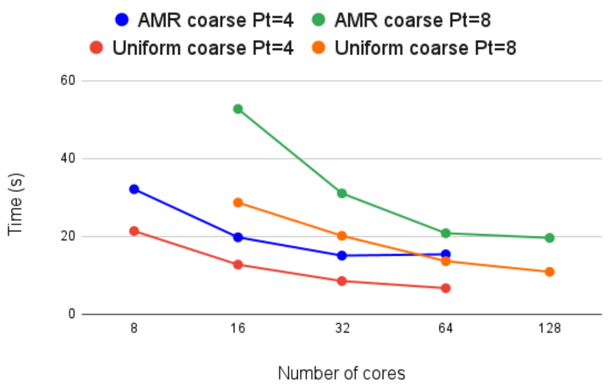

Fig. 12: Execution time comparison with and without AMR as coarse level.

Table 2: Number of iteration until convergence of supersonic flow by two-level Cascadic Parareal with and without AMR coarse level.

Coarse level	4 time subintervals	8 time subintervals
Uniform grid	3	6
AMR grid	2	4

6. Conclusion and Future Work

This paper applies Cascadic Parareal for compressible flow simulation with explicit methods. Cascadic Parareal is mainly based on *parareal* but has many layer hierarchies with a cascadic multigrid solving process. Cascadic Parareal has shown to provide great acceleration for subsonic flow. With only 64 cores, parallel-in-space and time by Cascadic Parareal could provide better parallel performance than that of pure spatial parallelization.

Experiments also show that using AMR coarse grid helps increasing accuracy for Cascadic Parareal for supersonic flow simulation. However, using AMR slightly increases the computation time. We conclude that the original Cascadic Parareal with spatial coarsening is the most efficient PinT method with explicit time-marching schemes to our best knowledge. When the target function is nonlinear and higher accuracy is required, Cascadic Parareal with AMR is a better choice.

For future work, Cascadic Parareal with AMR could be further experimented. For example, Cascadic Parareal with AMR at both fine and coarse grids. More complicated simulations such as vortex after the cylinder is also left as future work.

References

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

[2] 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.

[3] 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.

[4] 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.
[5] 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.
[6] 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.
[7] Y.-C. Chen and K. Nakajima, "Optimized Cascadic Multigrid Parareal Method for Explicit Time-Marching Schemes" *2021 12th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), IEEE*, pp. 9–18, 2021. DOI: 10.1109/ScalA54577.2021.00007
[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] F. A. Bornemann and P. Deuflhard, "The cascadic multigrid method for elliptic problems", *Numerische Mathematik*, vol. 75, No. 2, pp 135–152, 1996, DOI: 10.1007/s002110050234
[15] 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.
[16] T. H. Pulliam, "Artificial dissipation models for the Euler equations", *AIAA journal*, vol. 24, no. 12, pp 1931–1940, 1986, DOI: 10.2514/3.9550
[17] 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)