INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2008; **57**:1793–1804
Published online 7 January 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1703

Modified propagators of parareal in time algorithm and application to Princeton Ocean model

Yueming Liu*,† and Jianwei Hu

School of Mathematical Sciences and LPMC, Nankai University, Tianjin, China

SUMMARY

In this paper, two modified propagators of parareal in time algorithm are presented and applied to the Princeton Ocean model (POM). The parareal algorithm was pioneered by Lions *et al.* (*C. R. Acad. Sci. Paris Sér. I Math.* 2001; **332**:661–668) and later improved in a paper by Bal and Maday (*Proceedings of the Workshop on Domain Decomposition*, Zurich, Switzerland, Lecture Notes in Computer Science and Engineering Series, vol. 23. Springer: Berlin, 2002). Alternative formulations have also been presented, where the parallel in time algorithm proposed by Farhat and Chandesris (*Int. J. Numer. Methods Eng.* 2003; **58**:1397–1434) is the most important one. This algorithm enables parallel computation using a decomposition of the interval of time integration. Solutions are obtained sequentially on the coarse time grid and on the fine time grid in parallel. A practical problem for the Bohai Sea is calculated on the supercomputer cluster Nankai Stars. The properties of the modified propagators are analyzed in this paper.

Received 14 January 2007; Revised 30 October 2007; Accepted 31 October 2007

KEY WORDS: parareal; POM; Bohai Sea

1. INTRODUCTION

The parareal in time algorithm was first presented by Lions *et al.* [1]. It uses a coarse and a fine grid in time. These two grids are combined in a predictor–corrector scheme [2], creating an update for the entire time interval. The coarse grid and the update scheme are strictly sequential, while the fine grid can be computed in parallel on each subinterval of the coarse grid. The time discretization scheme of the Princeton Ocean model (POM) is 'partially' explicit [3]. The coarse time step length of the parareal algorithm may not be large enough due to the constraint of the Courant–Friedrichs–Lewy (CFL) condition. The speedup efficiency of the algorithm, to a large

*Correspondence to: Yueming Liu, School of Mathematical Sciences and LPMC, Nankai University, Tianjin, China.
†E-mail: cs_opohotmail.com

Contract/grant sponsor: Nature Science Foundation of China; contract/grant number: 10571094

degree, depends on the coarse time step length. To enlarge the coarse time step length, an implicit time scheme may be a good choice, although it would be a burden to modify the POM code. Here, we do not wish to directly consider enlarging the time step length to obtain more speedup efficiency. However, there are some other problems that have occurred when we apply this algorithm to the POM. The model usually needs a long fixed time interval (sometimes several whole days) to be computed in our application to the Bohai Sea and there are problems of adjusting the time step length and making the best use of CPUs, especially in the case where a huge number of CPUs are not available. Here we give an approach to deal with these problems, which arise from the implementation of the parareal algorithm. Two modified propagators of the parareal algorithm are given in the following sections, where the original propagator can be regarded as a special case. The speedup efficiency of the algorithms that involve these modified propagators are comparable to the original one. There are some interesting advantages of our modified propagators. For problems (e.g. POM) that need a fixed, long time interval to be computed and if we do not have enough CPUs to compute the whole interval at one time, we have to split the interval into several small intervals to fit the number of available CPUs. The length of the small time interval depends on the number of CPUs. This constraint can be removed, if the first modified propagator is used. The advantages of the second modified propagator is that its stability behavior seems better than the original one and this can be seen from the results of our numerical experiments.

2. A REVIEW OF THE PARAREAL IN TIME ALGORITHM

Let us consider the scalar linear problem of ordinary differential equations

$$u' = -au, \quad u(0) = u_0, \quad t \in [0, T] \tag{1}$$

The interval of time $[0, T]$ is decomposed into N subintervals $[T^n, T^{n+1}]$ of size ΔT and each subinterval is decomposed again with a smaller time step δt. Let $M = \Delta T / \delta t$. The parareal algorithm is defined using two propagation operators.

The coarse propagator $G(T^n, U^n)$ with time steps ΔT gives less accurate, whereas the fine propagator $F(T^n, U^n)$ with time steps δt on each subinterval gives a more accurate solver than $G(T^n, U^n)$. The algorithm starts with an initial approximation U_0^n, $n = 0, 1, \ldots, N$, at time $T^0, T^1, \ldots, T^N$, given, for example, by the sequential computation of $U_0^{n+1} = G(T^n, U_0^n)$, with $U_0^n = u_0$, here u_0 is the initial value. The iteration gives

$$U_{k+1}^{n+1} = G(T^n, U_{k+1}^n) + F(T^n, U_k^n) - G(T^n, U_k^n) \tag{2}$$

for $k = 0, 1, 2, \ldots$, where $F(T^n, U_k^n)$, $n = 0, 1, \ldots, N-1$, can be computed in parallel.

If the algorithm converges (F and G should satisfy some stability and approximation conditions, see the references), the global accuracy of the iterative process after a few iterations may be comparable to that of a fine discretization with time step δt used over the whole interval $[0, T]$. The order of accuracy of the algorithm is given as follows.

Proposition 2.1
Let $T < \infty, \Delta T = T/N, T^n = n\Delta T, n = 0, 1, \ldots, N$. Let $F(T^n, U_k^n)$ be the exact solution at T^{n+1} of (1) with $u(T^n) = U_k^n$, and let $G(T^n, U_k^n) = R(a\Delta T)U_k^n$ be a one-step method, characterized by its

DOI: 10.1002/fld

stability function $R(z)$ [4]. The local truncation error of G is bounded by $C\Delta T^{q+1}$, with $q>0$ and C a constant, then we have

$$\max_{1\leqslant n\leqslant N}|u(T^n)-U_k^n|\leqslant\frac{(CT)^k}{k!}\Delta T^{qk}\max_{1\leqslant n\leqslant N}|u(T_n)-U_0^n| \tag{3}$$

Proof
See Reference [5]. □

3. MODIFIED PROPAGATORS OF THE PARAREAL ALGORITHM

While the original propagator performs corrections on every coarse time step, our first modified propagator performs corrections on every $m\geqslant 1$ coarse time steps. The second propagator contains the parallel fine time solver, which is defined on $p\geqslant 1$ coarse time steps, in contrast to the original propagator, which is only defined on one coarse time step. If $m=1$ or $p=1$ these propagators reduce to the original one.

Propagator 1
Let $\tilde{G}(\tilde{T}^n,U_k^n)=R(a\Delta\tilde{T}/m)U_k^n$ be a choice for the coarse solver G. The iteration is

$$U_{k+1}^{n+1}=\tilde{G}^m(\tilde{T}^n,U_{k+1}^n)+F(\tilde{T}^n,U_k^n)-\tilde{G}^m(\tilde{T}^n,U_k^n) \tag{4}$$

where $\tilde{G}^m(\tilde{T}^n,\cdot)=\underbrace{\tilde{G}\cdot\tilde{G}\cdot\cdots\cdot\tilde{G}}_{m}(\tilde{T}^n,\cdot)$.

Proposition 3.1
Let $T<\infty,\Delta\tilde{T}=T/N,T^n=n\Delta\tilde{T},n=0,1,\ldots,N$. Let $F(\tilde{T}^n,U_k^n)$ be the exact solution at $\tilde{T}^{n+1}$ of (1) with $u(\tilde{T}^n)=U_k^n$, and let $\tilde{G}(\tilde{T}^n,U_k^n)=R(a\Delta\tilde{T}/m)U_k^n$ be a one-step method. the local truncation error of G^m is bounded by $C(\Delta\tilde{T}/m)^{q+1}$, with $q>0$ and C a constant, then we have

$$\max_{1\leqslant n\leqslant N}|u(\tilde{T}^n)-U_k^n|\leqslant\frac{(CT)^k}{k!}\left(\frac{\Delta\tilde{T}}{m}\right)^{qk}\max_{1\leqslant n\leqslant N}|u(\tilde{T}_n)-U_0^n| \tag{5}$$

Proof
It can be easily obtained from Proposition 2.1. Indeed, just replace the local truncation error $C\Delta\tilde{T}^{q+1}$ by $C(\Delta\tilde{T}/m)^{q+1}$. □

Remark 3.2

- Actually, propagator 1 can also be regarded as a special version of the general description of the parareal algorithm as a scheme consisting of a coarse and a fine propagator. There is a close connection between multiple steps of one-leg methods, extrapolation methods and multi-stage methods. There is really no significant difference between taking a one-leg method m times or using an m-stage Runge–Kutta method for propagator 1.
- Note that the time step length $\Delta\tilde{T}/m$ of explicit schemes for parabolic problems is constrained by the CFL stability condition, but $\Delta\tilde{T}$ can be chosen independent of this constraint, since only m need to be selected such that $\Delta\tilde{T}/m$ satisfies the CFL condition. This can be seen from the results of our numerical experiments in the last section.

DOI: 10.1002/fld

- For problems, e.g. POM, the whole time interval may be dozens of days and $[0, T]$ has to be split into several small intervals to fit the number of available CPUs, if we do not have enough CPUs to compute the whole interval at one time. In the formula of the original propagator, the length of the small time interval depends on the number of CPUs, because it needs one CPU for every one coarse time step. This constraint on the length of small time interval may be removed, if the first modified propagator is involved.
- The speedup efficiency can be roughly computed from the ratio between the number of steps of a fine sequential solver with δt and the number of total sequence steps of the parareal algorithm [1]. From a simple computation, we can see that the speedup efficiency of the first propagator with coarse time step $\Delta\tilde{T}$ is the same as the original one with coarse time step ΔT, where $\Delta\tilde{T}/m=\Delta T$. In fact, if the total number of coarse time steps is N and the number of CPUs is $C+1$ (for simplicity of our programming we need 1 master CPU and C slave CPUs). Let $S=N/C$ (assume S is integer), which denotes that we 'restart' the parareal algorithm S times. The total number of sequence steps of the original parareal algorithm is $(C+(M+C)k)S$, where the first 'C' in the expression denotes the number of prediction steps, the second 'C' denotes the correction steps,'M' the number of steps of the parallel solver on every parallel CPU, 'k' denotes the number of iteration steps. In contrast, the total number of steps of the algorithm with the first propagator is $(Cm+(Mm+Cm)k)(S/m)=(C+(M+C)k)S$, here assume C/m is integer and we 'restart' the algorithm C/m times.

The second propagator of the parareal algorithm is given as follows.

Propagator 2
The total computation time interval of each parallel solver takes $p\Delta T$ instead of ΔT. The correction formula is modified as

$$U_{k+1}^{n+1}=G(T^n,U_{k+1}^n)+F^p(T^{n-p+1},U_k^{n-p+1})-G(F^{p-1}(T^{n-p+1},U_k^{n-p+1})) \tag{6}$$

where $F^p(T^{n-p+1},\cdot)=\underbrace{F\cdot F\cdot\cdots\cdot F}_{p}(T^{n-p+1},\cdot)$ and $F^0=I$ means the identity operator.

Proposition 3.3
Let $T<\infty, \Delta T=T/N, T^n=n\Delta T, n=0,1,\ldots,N$. Let $F(T^n,U_k^n)$ be the exact solution at T^{n+1} of (1) with $u(T^n)=U_k^n$, and let $G(T^n,U_k^n)=R(a\Delta T)U_k^n$ be a one-step method. The local truncation error of G is bounded by $C\Delta T^{q+1}$, with $q>0$ and C a constant, then we have

$$\max_{1\leqslant n\leqslant N}|u(T^n)-U_k^n|\leqslant\frac{(CT)^k}{k!}(\Delta T)^{qk}\max_{1\leqslant n\leqslant N}|u(T_n)-U_0^n| \tag{7}$$

Proof
We denote by e_k^n the error at iteration step k of this modified propagator at time T^n, that is $e_k^n=u(T^n)-U_k^n, n=0,1,\ldots,N$. With (6) and an induction argument on n, this error satisfies

$$\begin{aligned}e_{k+1}^n&=R(a\Delta T)e_{k+1}^{n-1}+(e^{a\Delta T}-R(a\Delta T))e_k^{n-p+1}\\&=(e^{a\Delta T}-R(a\Delta T))\Sigma_{j=1}^{n-1}R(a\Delta T)^{n-j-1}e_k^{n-p+1}\end{aligned}$$

Let $\mathbf{e}_k=(e_k^{N-p+1},e_k^{N-p},\ldots,e_k^{2-p})^{\mathrm{T}}$, if the element upper index of $\mathbf{e}_k$ is less than zero, it is set to zero. The above relation can be expressed in the matrix form by collecting the error components

DOI: 10.1002/fld

of the kth iteration e_k^n. Then we have

$$\mathbf{e}_{k+1} = (e^{a\Delta T} - R(a\Delta T))M(R(a\Delta T))\mathbf{e}_k$$

where

$$M(\beta) = \begin{pmatrix} 0 & 1 & \beta^1 & & \beta^{N-2} \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta^1 \\ & \ddots & & \ddots & 1 \\ 0 & & & & 0 \end{pmatrix}$$

By induction on k, we obtain

$$\mathbf{e}_k = (e^{a\Delta T} - R(a\Delta T))^k M(R(a\Delta T))^k \mathbf{e}_0$$

and taking norms, we have

$$\|\mathbf{e}_k\|_\infty \leqslant |(e^{a\Delta T} - R(a\Delta T))|^k \|M(R(a\Delta T))\|_\infty^k \|\mathbf{e}_0\|_\infty$$

It can be proved that when $|\beta| \leqslant 1$

$$\|M(\beta)^k\|_\infty \leqslant \binom{N-1}{k} \frac{\prod_{j=1}^k (N-j)}{k!}$$

then we have

$$\begin{aligned} \|\mathbf{e}_k\|_\infty &\leqslant \frac{|e^{a\Delta T} - R(a\Delta T)|^k}{k!} \prod_{j=1}^k (N-j) \|\mathbf{e}_0\|_\infty \\ &\leqslant \frac{C^k (\Delta T)^{(q+1)k}}{k!} N^k \|\mathbf{e}_0\|_\infty \\ &\leqslant \frac{C^k (\Delta T N)^k}{k!} \Delta T^{qk} \|\mathbf{e}_0\|_\infty \\ &\leqslant \frac{(CT)^k}{k!} \Delta T^{qk} \|\mathbf{e}_0\|_\infty \end{aligned}$$

□

Remark 3.4

- If $p = 1$, the second propagator reduces to the original one:

$$U_{k+1}^{n+1} = G(T^n, U_{k+1}^n) + F(T^n, U_k^n) - G(T^n, U_k^n)$$

- The jumps $F(T^n, U_k^n) - G(T^n, U_k^n)$ in the original propagator can be obtained in sequence correction processes, if $G(T^n, U_k^n)$ in the jumps are reused from the previously coarse

DOI: 10.1002/fld

computation. However, it will need much memory to store the previous results. An alternative way of obtaining the jumps without using so much memory is that we can compute the jumps in the parallel CPUs. That is to say, two solvers $F(T^n, U_k^n)$ and $G(T^n, U_k^n)$ based on the initial values U_k^n should be computed on every parallel CPU. From a speedup efficiency point of view, there is only one more step in every parallel CPU; hence, it has little effect on the speedup efficiency. The jumps $F^p(T^{n-p+1}, U_k^{n-p+1}) - G(F^{p-1}(T^{n-p+1}, U_k^{n-p+1}))$ in our second propagator should be obtained in this parallel way. $F^p(T^{n-p+1}, U_k^{n-p+1})$ are obtained based on the initial value U_k^{n-p+1} and $G(F^{p-1}(T^{n-p+1}, U_k^{n-p+1}))$ are obtained based on the initial value $F^{p-1}(T^{n-p+1}, U_k^{n-p+1})$, which are just given from the last step of computation of $F^p(T^{n-p+1}, U_k^{n-p+1})$.

- The second propagator may be more stable than the others, because the jumps are obtained from p time steps before and this may give more accurate result of jumps. This can be seen in our following numerical results.

4. PRINCETON OCEAN MODEL

The POM was created by Alan F. Blumberg and George L. Mellor in 1977. They give a detailed description of the numerical circulation model in [3]. A more complete list of the references is available on the POM home page http://www.aos.princeton.edu/WWWPUBLIC/htdocs.pom. POM is a three-dimensional, primitive equation, time-dependent, sigma coordinate, free surface, estuarine and coastal ocean circulation model. The prognostic variables are the three components of the velocity fields, temperature, salinity and two quantities that characterize the turbulence, the turbulence kinetic energy and the turbulence macroscale. A horizontally and vertically staggered lattice of grid points is used for the computations.

4.1. Basic equations

Consider a system of orthogonal Cartesian coordinates with x, y, z increasing eastward, northward and vertically upwards, respectively. $z=\eta(x, y, z)$ is the surface elevation and $z=-H(x, y)$ is the bottom topography.
The continuity equation is

$$\nabla\mathbf{V}+\frac{\partial W}{\partial z}=0 \tag{8}$$

The Reynolds momentum equations are

$$\frac{\partial U}{\partial t}+\mathbf{V}\cdot\nabla U+W\frac{\partial U}{\partial z}-fV=-\frac{1}{\rho_0}\frac{\partial P}{\partial x}+\frac{\partial}{\partial z}\left(K_M\frac{\partial U}{\partial z}\right)+Fx \tag{9}$$

$$\frac{\partial V}{\partial t}+\mathbf{V}\cdot\nabla V+W\frac{\partial V}{\partial z}-fU=-\frac{1}{\rho_0}\frac{\partial P}{\partial y}+\frac{\partial}{\partial z}\left(K_M\frac{\partial V}{\partial z}\right)+Fy \tag{10}$$

$$\rho g=-\frac{\partial P}{\partial z} \tag{11}$$

DOI: 10.1002/fld

The pressure at depth z is

$$P(x, y, z, t) = P_{\text{atm}} + g\rho_0\eta + g\int_z^0 \rho(x, y, z', t)\,\mathrm{d}z' \tag{12}$$

The conservation equations for temperature and salinity are

$$\frac{\partial\Theta}{\partial t} + \mathbf{V}\cdot\nabla\Theta + W\frac{\partial\Theta}{\partial z} = \frac{\partial}{\partial z}\left(K_H\frac{\partial\Theta}{\partial z}\right) + F_\Theta \tag{13}$$

$$\frac{\partial S}{\partial t} + \mathbf{V}\cdot\nabla S + W\frac{\partial S}{\partial z} = \frac{\partial}{\partial z}\left(K_H\frac{\partial S}{\partial z}\right) + F_S \tag{14}$$

The second-order turbulence closure schemes are

$$\frac{\partial q^2}{\partial t} + \mathbf{V}\cdot\nabla q^2 + W\frac{\partial q^2}{\partial z} = \frac{\partial}{\partial z}\left(K_q\frac{\partial q^2}{\partial z}\right) + 2K_M\left[\left(\frac{\partial U}{\partial z}\right)^2 + \left(\frac{\partial V}{\partial z}\right)^2\right] + \frac{2g}{\rho_0}K_H\frac{\partial\rho}{\partial z} - \frac{2q^3}{B_1 l} + F_q \tag{15}$$

$$\frac{\partial q^2 l}{\partial t} + \mathbf{V}\cdot\nabla(q^2 l) + W\frac{\partial q^2 l}{\partial z} = \frac{\partial}{\partial z}\left(K_q\frac{\partial q^2 l}{\partial z}\right) + lE_1K_M\left[\left(\frac{\partial U}{\partial z}\right)^2\left(\frac{\partial V}{\partial z}\right)^2\right] + \frac{lE_1 g}{\rho_0}K_H\frac{\partial\rho}{\partial z} - \frac{q^3}{B_1}\widetilde{W} + F_l \tag{16}$$

where $\mathbf{V}$ is the horizontal velocity vector with components (U, V), ∇ is the horizontal gradient operator, W is the vertical velocity, ρ_0 the reference density, ρ the *in situ* density, g the gravitational acceleration, P the pressure, K_M the vertical eddy diffusivity of turbulent momentum mixing, Θ is the potential temperature (or *in situ* temperature for shallow water applications).

Because the ordinary x, y, z coordinate system has certain disadvantages in the vicinity of large bathymetric irregularities, the sigma coordinate system equations are introduced in the POM to transform both the surface and the bottom into coordinate surfaces. A mode splitting technique is used in the POM. The equations governing the dynamics of coastal circulation contain propagation of fast moving external gravity waves and slow moving internal gravity waves. It is desirable in terms of computer economy to separate out the vertical integrated computations (external mode) from the vertical structure equations (internal mode). That is to say, the velocity external mode equations are obtained by integrating the internal mode equations over the depth, thereby eliminating all vertical structures. For a detailed description about the surface and boundary conditions, we refer to [3]. The discretization scheme of POM is a finite difference formulation. The special grid of POM is a staggered computational 'C' grid. The vertical diffusion terms are treated implicitly, while the other terms are treated explicitly, see Section 5 of [3].

4.2. Parareal algorithm applied to POM

The parareal algorithm with different propagators can be easily applied to POM. However, the time discretization scheme for POM is partially explicit [3]; hence, the constraint of the CFL

DOI: 10.1002/fld

computational stability condition must be considered. The CFL condition on the vertically integrated, external mode, transport equations limits the time step according to

$$\Delta\widetilde{t}\leqslant\frac{1}{C_t}\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)^{1/2} \tag{17}$$

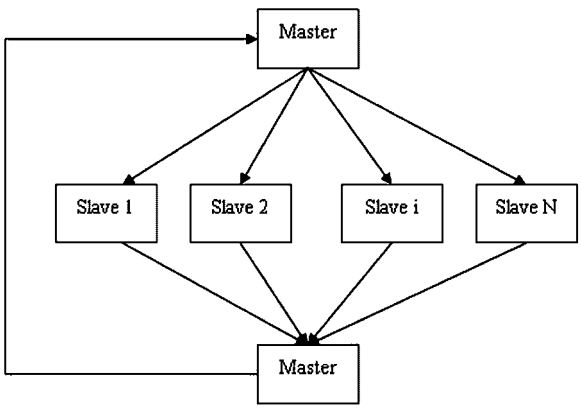
where $C_t=2(gH)^{1/2}+\overline{U}_{\max}$, $\overline{U}_{\max}$ is the maximum average velocity expected. The internal mode has a much less stringent time step:

$$\Delta\widetilde{T}\leqslant\frac{1}{C_T}\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)^{1/2} \tag{18}$$

where $C_T=2C+U_{\max}$, C is the maximum internal gravity wave speed, commonly of order 2 m/s and $U_{\max}$ the maximum advective speed. For typical coastal ocean conditions the ratio of the time steps, $\Delta\widetilde{T}/\Delta\widetilde{t}$, is often a factor of 80–100 and for more details we refer to [3].

4.3. Program design

The fortran77 parareal code is derived from the serial code POM98. The flow diagram of our program can be represented as follows:



The program steps of the original parareal algorithm are as follows.

Step 1: Master solves $U_0^{n+1}=G(T^n,U_0^n)$, $n=0,1,\ldots,N-1$, sequentially, then we obtain the results: $U_0^1,U_0^2,\ldots,U_0^N$ and then set $k=1$.

Step 2: Master sends $U_0^0,U_0^1,\ldots,U_0^{N-1}$ to Slave1, Slave2, …, Slave$N$, respectively.

Step 3 (*parallel*): Slave1, Slave2, …, Slave$N$ solve $u_k^n=F(T^n,U_k^n)$, $n=0,1,\ldots,N-1$ in parallel, respectively, and we obtain the results: $u_k^0,u_k^1,\ldots,u_k^{N-1}$.

Step 4: Master collects $u_k^0(T),u_k^1(T^2),\ldots,u_k^{N-1}(T^N)$ from Slave1, Slave2, …, Slave$N$.

Step 5: Master solves the propagation problem sequentially:

$$\begin{aligned}U_{k+1}^{n+1}&=G(T^n,U_{k+1}^n)+F(T^n,U_k^n)-G(T^n,U_k^n)\\&=G(T^n,U_{k+1}^n)+u_k^n(T^{n+1})-G(T^n,U_k^n)\end{aligned}$$

we obtain the corrected results: $U_{k+1}^1,U_{k+1}^2,\ldots,U_{k+1}^N$.

DOI: 10.1002/fld

Step 6: Master checks whether the results achieve the expected accuracy, if yes, stop iteration, otherwise set $k=k+1$ and go to Step 2.

The program steps of the algorithm with the first propagator are similar to the above steps and we just give the program steps of the algorithm with the second propagator, which can be expressed as follows:

Step 1: Master solves $U_0^{n+1}=G(T^n,U_0^n)$, $n=0,1,\ldots,N-p-1$, sequentially, then we obtain the results: $U_0^1,U_0^2,\ldots,U_0^{N-p}$ and then set $k=1$.

Step 2: Master sends $U_0^0,U_0^1,\ldots,U_0^{N-p-1}$ to Slave1, Slave2, ..., Slave$N-p$, respectively.

Step 3 (*parallel*): Slave1, Slave2, ..., Slave$N-p$ solve $u_k^{n-p+1}=F^p(T^{n-p+1},U_k^{n-p+1})$, $n=p-1,p,\ldots,N-1$ and $\tilde{U}_k^{n+1}=G(F^{p-1}(T^{n-p+1},U_k^{n-p+1}))$ in parallel, respectively, and we obtain the jumps: $S_k^{n+1}=u_k^{n-p+1}(T^n)-\tilde{U}_k^{n+1}$, $n=p-1,p,\ldots,N-1$.

Step 4: Master collects the jumps $S_k^p,S_k^{1+p},\ldots,S_k^{N-1}$ from Slave1, Slave2, ..., Slave$N-p$.

Step 5: Master solves the propagation problem sequentially:

$$\begin{aligned}U_{k+1}^{n+1}&=G(T^n,U_{k+1}^n)+F^p(T^{n-p+1},U_k^{n-p+1})-G(F^{p-1}(T^{n-p+1},U_k^{n-p+1}))\\&=G(T^n,U_{k+1}^n)+S_k^{n+1}\end{aligned}$$

we obtain the corrected results: $U_{k+1}^1,U_{k+1}^2,\ldots,U_{k+1}^{N-p}$.

Step 6: Master checks whether the results achieve the expected accuracy, if yes, stop iteration, otherwise set $k=k+1$ and go to Step 2.

5. NUMERICAL RESULTS

A practical problem for the Bohai Sea is calculated using the Nankai Stars, a supercomputer cluster, developed by the Institute of Scientific Computation, Nankai University. The experiment has been done on the grid size $98\times55\times7$ and with program parameters Mode$=4$ (three-dimensional calculation with temperature and salinity held fixed) (Figure 1).

5.1. Stability and speedup results

First, we present the numerical results of our first propagator (denoted by **P1** and the original algorithm by **P0**). Suppose we have only 201 CPUs available and the time interval that needs to be computed is $T=4$ days. Thus, the computation cannot be completed at one time with 201 CPUs. We need to split the whole time interval T into several small time intervals. The results are given in the following table. $D=1$ day in the cells of the table. The word 'Fail' in the column 'CFL' means that the CFL condition cannot be satisfied in our computation, while 'OK' means that we can obtain the final results. The number in column 'Restart times' equals the number of the small time intervals. That is to say, it needs a 'restart' of the parareal algorithm on each small interval. 'Ref. speedup' is the speedup without considering communication cost, which could be approximated by the ratio between the number of steps of the fine sequential solver with δt over the whole time interval and the number of total sequence steps of the parareal algorithm. We take the iteration times $k=2$ in all of our computations.

DOI: 10.1002/fld

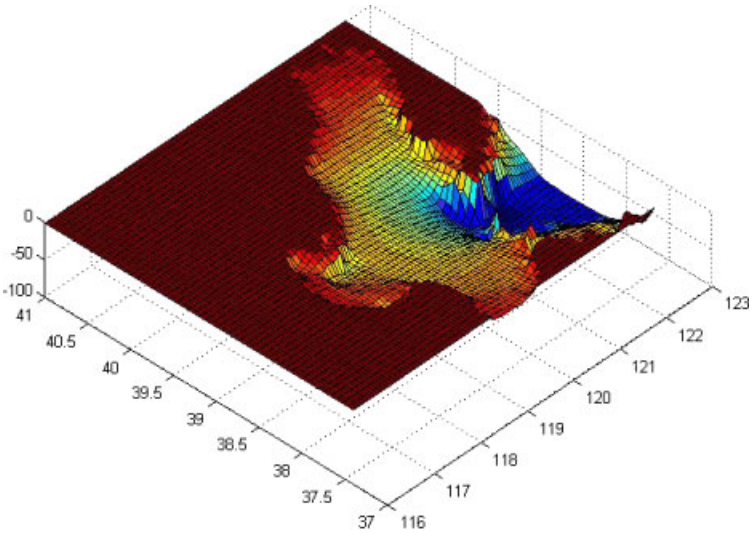


Figure 1. Bohai Sea topography and grid (longitude: 116–122.5°, latitude: 36.5–41°).

	ΔT	$\Delta\tilde{T}$	δt	CFL	Restart times	CPU	Speedup	Ref. speedup
P0	$\frac{D}{100}$	—	$\frac{D}{2400}$	Fail	2	201	—	9.7
	$\frac{D}{200}$	—	$\frac{D}{2400}$	OK	4	201	5.2	5.7
P1	$\frac{D}{100}$	$\frac{D}{50}$	$\frac{D}{2400}$	Fail	1	201	—	9.7
	$\frac{D}{200}$	$\frac{D}{100}$	$\frac{D}{2400}$	OK	2	201	5.4	5.7
	$\frac{D}{200}$	$\frac{D}{50}$	$\frac{D}{2400}$	OK	1	201	5.5	5.7

We can see from the results that the CFL constraint condition directly relates to ΔT, while $\Delta\tilde{T} = m\Delta T$ seems independent of the CFL condition, which can be enlarged by increasing the value of m. $\Delta\tilde{T}$ can be regarded as the step length of 'propagation'; hence, **P1** may have some flexibility in practical application.

The speedup efficiency in our configuration may be a little low. A speedup of 5.5 on 201 processors is only an efficiency of less than 3%. However, if we take a much smaller time step length of the fine solver, e.g. $\delta t = D/24000$, the speedup of **P0** or **P1** with $\Delta T = D/200$ would be about 37, that is an efficiency near 18%. The underlying reason is the selection of the ratio between ΔT and δt (see [1]). In our application, the selection of the ratio is enough.

Another interesting thing can be seen from the results that **P0** and **P1** have the same 'Ref. speedup', while having different values in the column 'Speedup'. The reason is that **P1** only needs communication at points of time step length of 'propagation'.

Next, we consider the second propagator of the algorithm(**P2**). We performed two groups of computation. The first group is based on a zero initial velocity field and the second group is based

DOI: 10.1002/fld

on a 'good' initial velocity field, indeed, it is a serial result of finer time step with $T=3$ days. The whole time interval that needs to be computed is $T=1$ day.

The results of zero initial velocity field and 'good' initial velocity field are presented as follows:

		P0		**P2**		
ΔT	δt	CFL	CPU	p	CFL	CPU
$\frac{T}{100}$	$\frac{T}{2400}$	Fail	101	2	Fail	100
				4	Fail	98
$\frac{T}{150}$	$\frac{T}{2400}$	Fail	151	2	Fail	150
				4	Fail	148
$\frac{T}{200}$	$\frac{T}{2400}$	OK	201	2	OK	200
				4	OK	198

		P0		**P2**		
ΔT	δt	CFL	CPU	p	CFL	CPU
$\frac{T}{100}$	$\frac{T}{2400}$	Fail	101	2	Fail	100
				4	Fail	98
$\frac{T}{150}$	$\frac{T}{2400}$	Fail	151	2	OK	150
				4	OK	148
$\frac{T}{200}$	$\frac{T}{2400}$	OK	201	2	OK	200
				4	OK	198

When $\Delta T=T/150$ and a good initial velocity field is provided, **P2** does not violate the CFL constraint condition as indicated by the above results ($\Delta T=T/150$ satisfies the CFL condition for a serial computation of the given problem). This indicates that the behavior of the second propagator may be more stable than the original one, at least in the configuration of our calculation.

The speedup results of **P2** with $p=2$ based on the 'good' initial velocity field are given as follows:

	ΔT	δt	CFL	Relative error (%)	CPU	Speedup	Ref. speedup
P0	$\frac{T}{200}$	$\frac{T}{2400}$	OK	5.62	201	5.2	5.7
	$\frac{T}{150}$	$\frac{T}{2400}$	Fail	—	151	—	7.2
P2	$\frac{T}{200}$	$\frac{T}{2400}$	OK	5.62	200	4.9	5.4
	$\frac{T}{150}$	$\frac{T}{2400}$	OK	7.10	150	6.1	6.6

DOI: 10.1002/fld

The speedup of **P2** is lower than **P0**, because the parallel time interval of **P2** is larger than **P0**. Note that again, the low speedup efficiency because of the selection of the ratio between ΔT and δt.

6. CONCLUSION

The essential speedup efficiencies are comparable for all of the three propagators. Our modified propagators **P1** and **P2** may have some flexibility in the configuration in the time step ratio between the coarse and the fine time step and **P2** may behave in a more stable manner in some applications.

REFERENCES

1. Lions J-L, Maday Y, Turinici G. Résolution d'EDP par un schéma en temps pararél. *Comptes Rendus de l' Acadamié des Sciences Paris*, *Série I*, *Mathématique* 2001; **332**:661–668.
2. Maday Y, Turinici G. The parareal in time iterative solver: a further direction to parallel implementation. In *Proceedings of the 15th International Domain Decomposition Conference*, Kornhuber R, Hoppe PHW, Périaux J, Pironneau O, Widlund OB, Xu J (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, 2003; 731–733.
3. Blumberg AF, Mellor GL. A description of a three-dimensional coastal ocean circulation model. In *Three-dimensional Coastal Ocean Models*, Heaps N (ed.), vol. 4. American Geophysical Union: Washington, DC, 1987; 208.
4. Staff G, Rønquist E. Stability of the parareal algorithm. In *Proceedings of the 15th International Conference on Domain Decomposition Methods*. Springer: Berlin, 2003.
5. Gander MJ, Vandewalle S. Analysis of the parareal time-paralleled time-integration method. *Report TW443*, Department of Computer Science, K.U. Leuven, Leuven, Belgium, November 2005. Available at: URL=http://www.cs.kuleuven.ac.be/publicaties/rapporten/tw/TW443.abs.html.
6. Bal G, Maday Y. A 'parareal' time discretization for non-linear PDE's with application to the pricing of an American put. In *Proceedings of the Workshop on Domain Decomposition*, Zurich, Switzerland, Pavarino LF, Toselli A (eds). Lecture Notes in Computer Science and Engineering Series, vol. 23. Springer: Berlin, 2002.
7. User's Guide for A Three-dimensional, Primitive Equation, Numerical Ocean Model. Available at: http://www.aos.princeton.edu/ WWWPUBLIC/htdocs.pom.
8. Farhat C, Chandesris M. Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid–structure applications. *International Journal for Numerical Methods in Engineering* 2003; **58**:1397–1434.

DOI: 10.1002/fld