Comparing the Laplace Transform and Parareal Algorithms

Craig C. Douglas and Li Deng Douglas
University of Wyoming
Laramie, WY, USA and
e-mail: cdougla6@uwyo.edu

Hyoseop Lee
Alcatel-Lucent Bell Labs - Seoul
Seoul, South Korea
e-mail: hyoseop.lee@gmail.com

Dongwoo Sheen
Seoul National University
Seoul, South Korea
e-mail: sheen@snu.ac.kr

Abstract—**Both the Laplace Transform and the Parareal family of algorithms promise to provide completely parallel in time and space computational results. Given a random time dependent partial differential equation it is unclear which algorithm will run faster. We define the algorithms in question, including more specific details than usual. We define interesting parallel environments, some of which do not exist yet. Finally, we demonstrate some computational environments in which one of the algorithms can be expected to be faster than the other algorithm.**

Keywords-parallel computing, time-space parallelism, partial differential equations, PDE solvers

I. Introduction

We are interested in comparing two algorithms that allow for both time and space parallelism in solving either an ordinary differential equation,

$$u' = f(u), \tag{1}$$

or a parabolic equation,

$$\frac{du}{dt} = L(u) + f. \tag{2}$$

Classical methods for solving either (1) or (2) are based on simple time stepping over an ordered set with N_T time steps

$$\left\{t_1, t_2, \cdots, t_{N_T-1}, t_{N_T}\right\}. \tag{3}$$

Consider the computational complexity of one method for solving (1)-(3), namely using backward Euler for time stepping combined with a multigrid. For either two or three spatial dimensions with N_S total spatial points, the cost of solving the problems is one of the following two cases:

Serial	$O(N_T \cdot N_S)$
Parallel	$O(N_T \cdot \log^2(N_S))$.

The central question that this paper is interested in investigating is the following:

Can we robustly solve parts of the problem later in time before fully approximating the solution earlier in time using something similar to a traditional numerical algorithm for solving time dependent ordinary or parabolic differential equations?

Over a 60 year period, the answer was repeatedly proposed to be yes, but the general algorithmic technique in all cases was conclusively proven to be in error [1]. Then the Parareal algorithm [2] was published using a very different algorithmic technique.

The remainder of this paper is organized by algorithms and numerical experiments. In Section II, we define the Parareal algorithm and its interpretations. In Section III, we define the Laplace transform and its relevant properties. In Section IV, we compare the two algorithms in a simple way. In Section V, we draw some conclusions about which of the two algorithms to try first based on the number of computing cores in a parallel computer.

II. Parareal

Consider solving (1) starting from $u_1 = u(t_1)$. Use two time propagation operators of the form,

$$F(t_2, t_1, u_1), \tag{4}$$
$$G(t_2, t_1, u_1), \tag{5}$$

where *F* and *G* are the fine quality and coarse (or rough) operators for time stepping, respectively.

For example, both time propagation operators could be the same time stepping scheme, but on nested time meshes, i.e., *F* can be a standard time stepping operator (e.g., backward Euler or Crank-Nicolson) on (3), but *G* could be the same time stepping scheme except on every τ^{th} point in (3). If the time mesh spacing is Δt for *F*, then the time mesh spacing for *G* is $\Delta T = \tau\Delta t$.

Parareal starts with an initial guess U_n^0 at time points $t_1, t_2, \cdots, t_N$ and computes U_n^k by a series of correction iterations. Parareal is formally defined by

```
for $k = 0, 1, \rightleftharpoons$                                        Loop over iterations
  for $n = 0, 1, \rightleftharpoons, N-1$                                    time steps
    $U_{n+1}^{k+1} = F(t_{n+1}, t_n, U_n^k) + G(t_{n+1}, t_n, U_n^{k+1}) - G(t_{n+1}, t_n, U_n^k)$
```

The dominant part of the computation in Parareal is computing *F* at every time step each outer iteration.

However, this is embarrassingly parallel. Further, to implement Parareal to experiment with takes only five lines of Matlab.

The convergence required is the accuracy of the F time propagator at each time step t_n. We can see immediately that we are guaranteed this level of convergence after N steps since this is equivalent to just doing the original time stepping on (3). Recalling the central question of Section I, a reasonable question to ask is if this is all that Parareal can guarantee since if it really takes N steps, then all of the computations involving G are a waste of computational resources.

A typical theorem for Parareal is one proven in [2]:

Theorem 1: *For* $u' = -au$ and $u(0) = u_0$, *let* $F(t_{n+1}, t_n, U_n^k)$ *denote the exact solution at* t_{n+1} *and* $G(t_{n+1}, t_n, U_n^k)$ *be the backward Euler approximation with time step* ΔT. *Then*

$$\max_{1\le n\le N}\left|u(t_n) - U_n^k\right| \le C_k \Delta T^{k+1}.$$

The proof can be found in [2].

Before interpreting the significance of Parareal, consider Multiple Shooting Methods [3], an older set of algorithms from an earlier era before parallel computers were common. For N intervals for solving (1) with $u(0) = u_0$, $t \in [0,1]$, define

$$U_{n+1}^{k+1} = u_n(t_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)\left(U_n^{k+1} - U_n^k\right). \quad (6)$$

Then the following theorem can be proved. *Theorem 2*: *If in the multiple shooting method*,

$$u_n(t_{n+1}, U_n^k) \approx F(t_{n+1}, t_n, U_n^k), \quad (7)$$

$$\frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)\left(U_n^{k+1} - U_n^k\right) \approx \quad (8)$$
$$G(t_{n+1}, t_n, U_n^{k+1}) - G(t_{n+1}, t_n, U_n^k),$$

then the multiple shooting and Parareal methods coincide.

See [4] for the proof.

Parareal can be interpreted in three fundamentally different ways:

1. Just a solver for the F equations if Parareal iterates until convergence.
2. A new time integrator if the number of iterations is fixed in advance.
3. Different approximations to the Jacobian in (6)-(8) lead to different time-parallel algorithms.

Interpretations 2 and 3 are clearly of interest.

III. Laplace Transform

We solve (2) with $L(u) = -Au$ starting from $u(0) = u_0$. Given some $z \in \mathbb{C}$ and a function $u(\cdot,t)$, the Laplace transform in time is given by

$$\hat{u}(\cdot,z) \equiv L[u](z) = \int_0^\infty u(\cdot,t)e^{-zt}\,dt. \quad (9)$$

We are left solving (9) by any reasonable elliptic partial differential equation solver the transformed problem

$$\hat{u}(\cdot,z) = \left(zI + A\right)^{-1}\left(u_0(\cdot) + \hat{f}(\cdot,z)\right). \quad (10)$$

We assume for some $C_S \in \mathbb{R}^+$ the real parts of the singular points of $u_0(\cdot) + \hat{f}(\cdot,z) \le C_S$. Let the integral contour be a straight line parallel to the imaginary axis,

$$\Gamma \equiv \left\{z \in \mathbb{C} \mid z(\omega) = \alpha + i\omega, \alpha \ge C_S, \omega \in \mathbb{R}\right\}. \quad (11)$$

The Laplace inversion formula is given by

$$u(\cdot,t) = \frac{1}{2\pi i}\int_\Gamma \hat{u}(\cdot,z)e^{zt}\,dz. \quad (12)$$

When $|z| \gg 0$ and $z \in \Gamma$ has negative real parts, the discretization error in evaluating the integrals in (12) for $u(\cdot,t)$ is significantly reduced for all $t > 0$. We deform Γ to the left half plane with all singularities to its left and a hyperbola contour

$$\Gamma = \begin{cases} z \in \mathbb{C} \mid z(\omega) = \varsigma(\omega) + is\omega, \omega \in \mathbb{R}, \\ \varsigma(\omega) = \gamma - \sqrt{\omega^2 + v^2} \end{cases}. \quad (13)$$

In essence, the hyperbola contour must be kept away from the spectrum of $-A$ and the singular points of $\hat{f}(z)$.

Define $\psi(\omega) = \tanh(\frac{\tau\omega}{2})$: $(-\infty,\infty) \to (-1,1)$. Then

$$u(t) = \frac{1}{2\pi i}\int_\Gamma e^{\left[z(\psi^{-1}(y))t\right]} \cdot \hat{u}\left(z(\psi^{-1}(y))\right) \cdot$$
$$\left\{\varsigma'(\psi^{-1}(y)) + is\right\}\frac{d\psi^{-1}}{dy}(y)dy,$$

which can be discretized using something as simple as the trapezoidal rule. While it is not obvious mathematically, this last step leads to a roundoff problem that limits how parallel the Laplace transform method is. While it appears that there is no limit to the number of points $z \in \Gamma$ can be chosen, with each point z leading to an elliptic problem that can be solved in parallel, that is not the case, unfortunately.

IV. Numerical Experiment

Consider

$$u_t - \frac{u_{xx}+u_{yy}}{5\pi^2} = 0, \qquad (14)$$
$$u(x,y,0) = e\sin(2\pi x)\sin(\pi y)$$

with the exact solution

$$u(x,y,t) = e^{1-t}\sin(2\pi x)\sin(\pi y).$$

The Laplace transform (9) is given by

$$z\hat{u} - \frac{\hat{u}_{xx}+\hat{u}_{yy}}{5\pi^2} = e\sin(2\pi x)\sin(\pi y)\ in\ (0,1)^2,$$
$$u = 0\ \partial(0,1)^2,$$

where

$$\Gamma = \left\{ \begin{array}{c} z\in\mathbb{C} \mid z(\omega)+is\omega,\ \mathrm{y}\in(-1,1), \\ \varsigma(\omega) = \gamma - \sqrt{\omega^2+\upsilon^2}, \\ \omega(y) = \frac{2}{\tau}\tan^{-1} y = \frac{1}{\tau}\log\frac{1+y}{1-y} \end{array} \right\}.$$

Using [5,7], we get $\alpha \approx 1.1721$. Hence, for (11) and (13) we get,

$$\Lambda(\alpha) = \cosh^{-1}\left(\frac{2\alpha}{(4\alpha-\pi)\sin\alpha}\right),$$
$$\gamma = \frac{(4\alpha-\pi)\pi N_z}{\Lambda(\alpha)t},$$
$$\upsilon = \gamma\sin(\alpha),\ s = \gamma\cot(\alpha),\ \text{and}$$
$$\tau = \frac{\log(2N_z-1)}{\gamma\sin(\alpha)\sinh\left(\Lambda(\alpha)\frac{N_z-1}{N_z}\right)},$$

which is quite complex, but simple to evaluate once α is known.

In our numerical experiments, in order to solve (10), we used $N_z = 32$ points on the hyperbola contour (13), a compact fourth order finite difference discretization on an $N_x \times N_x$ uniform mesh with $N_x = 160$, and MADPACK version 5 for the elliptic partial differential equation solver [6, 7]. In Fig. 1 we see the runtimes for each method. In Fig. 2 we see the speedups for each method. The Laplace transform method is the clear winner for this specific example based on runtimes.

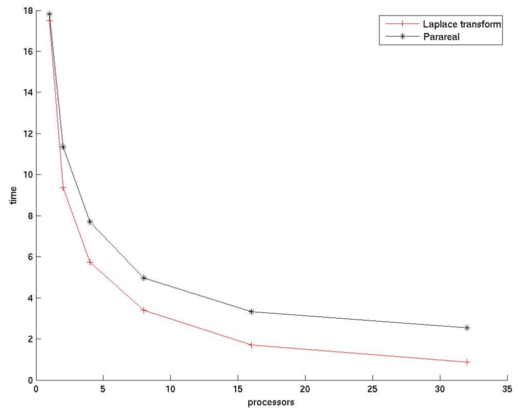

Figure 1. Wall clock time versus processors

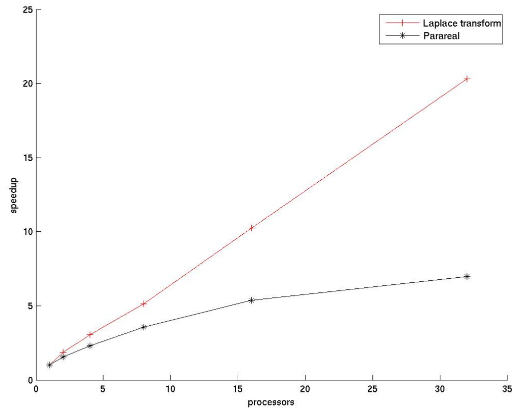

Figure 2. Speedup versus processors

V. CONCLUSIONS

There are at least two families of robust algorithms to create parallel in both time and space ordinary or parabolic differential equation solvers, namely Parareal and the Laplace transform.

For one processing core, neither is appropriate and both are guaranteed to run slower than standard serial solvers.

For a small number of processing cores, the Laplace transform approach will normally be the method of choice. Suppose that there are N_z chosen points on the contour hyperbola. For $P = C_{N_z}N_z$ processing cores, where $C_{N_z} \in \mathbb{N}$ is small, then the Laplace transform is the obvious choice, particularly if $C_{N_z} = 1$. This is because Parareal may use too many iterations, whereas we know we have N_z parallelism trivially with the Laplace transform (and possibly much more depending on the elliptic equation solver used).

For a (very) large number of processing cores, Parareal is the obvious method to try first. This is because a group of processors can be assigned to each time step in the time domain, where the size of the groups is determined by the parallel code used to advance the approximate solutions. Due to roundoff errors, there appears to be limits on how large N_z can be feasibly.

As to the principal question of the paper in Section I, there is no clear answer at this time.

Acknowledgment

This research was supported in part by the National Science Foundation grants CNS-1018072 and EPS-1135483, Air Force Office of Scientific Research FA9550-11-1-0341, National Research Foundation 2011-000344, the City of Seoul WR080951, and King Abdullah University of Science & Technology grant KUS-C1-016-04.

References

[1] A. Deshpande, S. Malhotra, C. C. Douglas, and M. H. Schultz, “A rigorous analysis of time domain parallelism,” Parallel Alg. Appl., vol. 6, 1995, pp. 53-62.

[2] J. L. Lions, Y. Maday, and G. Turinici, “A parareal in time discretization of PDEs,” C. R. Acad. Sci. Paris Ser. I Math., vol. 332, 2001, pp. 661-668.

[3] H. B. Keller, Numerical Methods for Two-Point Boundary Value Problems, Waltham, MA: Blaisdell, 1968.

[4] M. J. Gander and S. Vandewalle, “Analysis of the parareal time-parallel time-integration method,” SIAM J. Sci. Comput., vol. 29, 2007, pp. 556–578.

[5] J. A. C. Weideman and L. N. Trefethen, “Parabolic and hyperbolic contours for computing the Bromwich integral,” Math. Comp., vol. 76, 2007, pp. 1341–1356.

[6] C. C. Douglas, “Madpack: a family of abstract multigrid or multilevel solvers,” Comput. Appl. Math., vol. 14, 1995, pp. 3–20.

[7] C. C. Douglas, I. Kim, H. Lee, and D. Sheen, “Higher-order schemes for the Laplace transformation method for parabolic problems,” Comput. Visual Sci., vol. 14, 2011, pp. 39–47.