An Investigation into Parareal

Abhijit Chowdhary

May 20, 2019

Contents

1 Introduction

2 Parareal

We would like to solve the autonomous ordinary differential equation:

$$\begin{cases} u'(t) = f(u), & t \in [t_0, t_f] \\ u(t_0) = u_0 \end{cases}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ and $u : \mathbb{R} \to \mathbb{R}^d$.

Parareal is an iterative scheme to approximate u for the above, which can be derived from the concept of *multiple shooting* methods. The idea behind these methods are to turn the problem into a nonlinear optimization problem, which then can be solved via Newton-Raphson. [?]

Take our time domain $[t_0, t_f]$ and partition it into N pieces $0 = t_0 < \cdots < t_N = t_f$. Looking specifically at the interval $[t_n, t_{n+1}]$, we pose the new ODE:

$$\begin{cases} u_n' = f(u_n), & t \in [t_n, t_{n+1}] \\ u(t_n) = U_n \end{cases}$$

' where U_n is such that $U_n - u_{n-1}(t_n, U_{n-1}) = 0$, i.e. the initial condition satisfies the solution to the previous time slice. The conditions on U_i form a nonlinear system $F(U) = 0$, which we can approximate using Newton-Raphson, so recieve the iteration $U^{k+1} = U^k - J_F^{-1}(U^k)F(U^k)$.

3 Implementation

3.1 Naive OpenMP

3.2 Pipelined OpenMP

4 Efficiency Analysis

Here we try to analyze the speedup of Parareal analytically, and then through scalability studies conducted on the supercomputer Prince, here at NYU.

4.1 Theoretical Results

We try to directly estimate the speedup of the Naive OpenMP implementation. We denote our speedup S by: ([2] slide 17)

$$S = \frac{\text{Time taken by fine solver}}{\text{Parareal time}}$$

Algorithm 1 Parareal Algorithm

```
Require: $y_0$ and course and fine solvers $\mathcal{G}$, $\mathcal{F}$.
  $y_c \leftarrow \mathcal{G}(t_f, t_0, y_0)$.                    ▷ Coursely approximate solution
  $y \leftarrow y_c$.
  while iter < max_iter && not converged do
      for $n = 0 \to P$ do                                          ▷ Parallel capable
          $y_f(n) = \mathcal{F}(t_{n+1}, t_n, y(n))$.               ▷ Note FSAL property
          $\delta y(n) = y_f(n) - y(n)$.                            ▷ corrector term.
      end for
      for $n = 0 \to P$ do
          $v = \mathcal{G}(t_{n+1}, t_n, y(n))$.                    ▷ Predict.
          $y(n) = v + \delta y(n)$.                                 ▷ Correct.
      end for
  end while
```

Let the time taken by the fine and course solvers respectively be denoted by T_f and T_g. Furthermore, suppose we have the optimal amount of processors P, such that $\Delta t = T/P$, where T is the time to integrate up to.

Considering this, at each step parareal makes a predictor and corrector computation. The predictor is just a run of the course solver, costing PT_g. The corrector has two parts, the fine and course portion. The fine portion is parallelized optimally, and therefore costs just T_f, instead of PT_f. However, the course portion of the corrector takes advantage of the *first same as last* property, and doesn't have to be counted sepreately. Recall, also, before the parareal iteration, we make an initial guess using one course solve, PT_g. Therefore, for a parareal with k iterations it would have speedup:

$$S = \frac{PT_f}{PT_g + k(PT_g + T_f)} = \frac{1}{\frac{T_g}{T_f} + k(\frac{T_g}{T_f} + \frac{1}{P})} = \frac{1}{\frac{T_g}{T_f}(1+k) + \frac{k}{P}}$$

Notice, under the limit:

$$\lim_{P\to\infty} S = \frac{1}{\frac{T_g}{T_f}(1+k)} = \frac{T_f}{T_g(1+k)}$$

Supposing we have the idealized case of $k = 1$, then we would find that this would result in speedup $\frac{T_f}{2T_g}$, which, given that $T_g << T_f$, could be significant. However, in practice, k has to be taken to be large enough so that S isn't too great. This is one of the first indicators we have of why $T_g << T_f$ must be true. In general, knowing that $T_g \leq T_f \implies S \leq \frac{1}{k+1}$, which is somewhat dismal.

4.2 Scalability

5 Stability Analysis

With regards to the stability analysis, we aim to try and prove some results on the "stability function" $R(z)$ as seen in Levecue's [3] chapter five through eight and in Staff [5]. We heavily follow the arguments in these sources to derive such results.

Suppose we have the following ordinary differential equation:

$$\begin{cases} u' = \mu u, & \mu < 0, t > 0 \\ u(0) = u_0 \end{cases} \tag{1}$$

We would like to analyze the stability of parareal on this system with a course operator $\mathcal{G}$ and a fine operator $\mathcal{F}$.

5.1 A stability function | Inspired by Euler Methods

First we restrict ourselves to the infamous explicit and implicit Euler, with the hope of deriving some stability criteria for them. Recall, that the explicit and implicit euler scheme is:

$$u_{n+1} = u_n + \mu\Delta t u_n = (1 + \mu\Delta t)u_n \qquad \text{(Explicit Euler)}$$
$$u_{n+1} = u_n + \mu\Delta t u_{n+1} = (1 - \mu\Delta t)^{-1}u_n \qquad \text{(Implicit Euler)}$$

It's easy to see, after unrolling the recurrence on u_n, that these translate to $u_{n+1} = (1+\mu\Delta t)^n u_0$ or $(1-\mu\Delta t)^{-n}$ for explicit and implicit Euler respectively. Calling $R_e(z) = 1+z$ and $R_i(z) = (1-z)^{-1}$, we can analyze the stability of the method through these, specifically we desire that $\left|R(z)\right| \leq 1$, so that $R(z)^n$ doesn't blow up in n.

With respect to Parareal, let's try and write our iteration in the form $\lambda_n^k = H(n,k)\lambda_0$. Applying the explicit Euler iteration to **??**, we see that it goes to:

$$\begin{aligned} \lambda_{n+1}^{k+1} &= \mathcal{G}(t^{n+1}, t^n, \lambda_n^{k+1}) + \mathcal{F}(t^{n+1}, t^n, \lambda_n^k) - \mathcal{G}(t^{n+1}, t^n, \lambda_n^k) \\ &= R_e(\mu\Delta t)\lambda_n^{k+1} + R_e(\mu\delta t)^s\lambda_n^k - R_e(\mu\Delta t)\lambda_n^k \end{aligned}$$

where we say $s = \Delta t/\delta t$, i.e. how many fine steps are needed to make one course step. Combining like terms results in:

$$\lambda_{n+1}^{k+1} = R_e(\mu\Delta t)\lambda_n^{k+1} + \left(R_e(\mu\delta t)^s - R_e(\mu\Delta t)\right)\lambda_n^k$$

Now, if we were to note the terms on λ, we note that we have something very similar to the recurrence relation on combinations $\binom{n}{k} = \binom{n}{k-1} + \binom{n-1}{k-1}$. Exploiting that relationship, we can unroll our recursion into:

$$\lambda_{n+1}^{k+1} = \left(\sum_{i=0}^{k}\binom{n}{i}\left[R_e(\mu\delta t)^s - R_e(\mu\Delta t)\right]^i R_e(\mu\Delta t)^{n-i}\right)\lambda_0 = H_e(\mu, n, k, \delta t, \Delta t)\lambda_0$$

In doing this, we notice that any method for which we could write as $u_{n+1} = R(z)u_n$ will have a very similar stability region, with $R_e \to R$. See figure 1 for the regions of stability under the Forward Euler method.

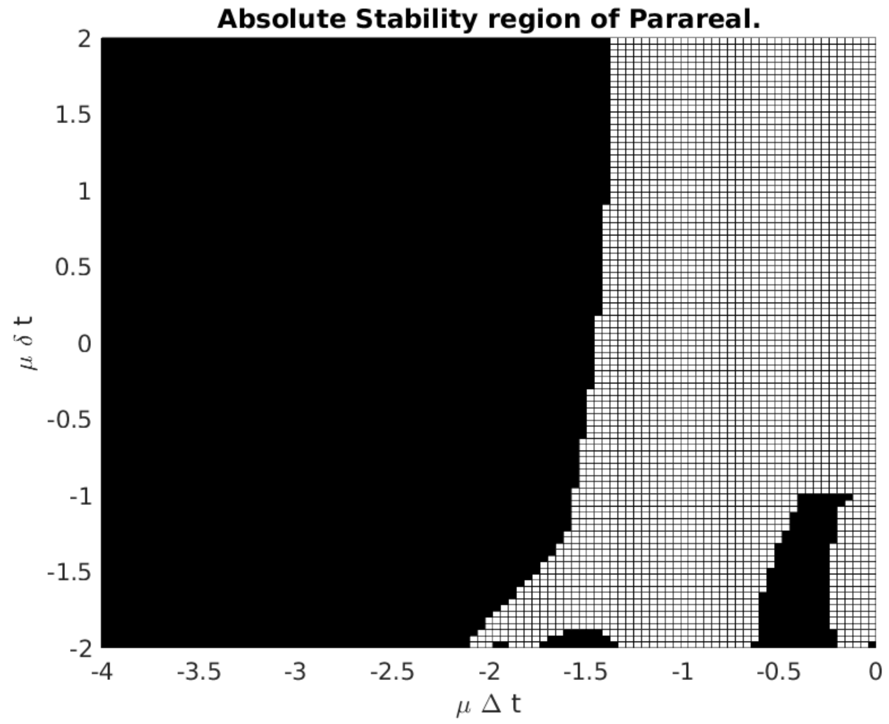

Figure 1

6 Convergence Analysis

Now down to the main point, we would like to analyze and confirm the convergence of the Parareal method. First we derive a theoretical result, and then confirm it through numerical expirements.

6.1 Theoretical Convergence

Suppose we have the following ordinary differential equation:

$$\begin{cases} u' = f(t,u), \quad t > 0 \\ u(0) = u_0 \end{cases} \tag{2}$$

In addition suppose we have the course operator $\mathcal{G}(t^{n+1}, t^n, u^n)$ and the fine operator $\mathcal{F}(t^{n+1}, t^n, u^n)$ with the following properties:

1. On $\mathcal{G}(t^{n+1}, t^n, u^n)$:
   - Suppose this operator has order m.
   - Suppose it's Lipschitz in the initial condition:

     $$\left\|\mathcal{G}(t^{n+1}, t^n, u) - \mathcal{G}(t^{n+1}, t^n, v)\right\| \leq C\|u - v\|$$

     In particular we write $C = (1 + L\Delta t)$.

2. With respect to $\mathcal{F}(t^{n+1}, t^n, u^n)$, we suppose it's accurate enough to be assumed to be the true solution u^*. This means that if $\mathcal{G}$ is accurate with order m to the true solution, then it too will be so to $\mathcal{F}$.

Then we can prove the following theorem:

Theorem. *The parareal method with course operator $\mathcal{G}$ and fine operator $\mathcal{F}$ has order of accuracy mk, where $k-1$ is the number of parareal iterations made. [1] [2]*

Proof. We proceed via induction on k and n. Suppose $k=1$, then it is trivial, this is the course operator, and for $n=0$, this is the initial condition which we know to any accuracy.

Now suppose for $k, n > 1$, that we know:

$$\left\|u(t^n) - u_k^n\right\| \leq \|u_0\| \, C(\Delta t)^{mk},$$

We want to show that:

$$\left\|u(t^n) - u_{k+1}^n\right\| \leq \|u_0\| \, C(\Delta t)^{m(k+1)}$$

To proceed, recall that $\mathcal{F}$ is assumed to be a good approximation for $u(t^n)$, so we may write:

$$\begin{aligned}
\left\|u(t^n) - u_{k+1}^n\right\| &= \left\|\mathcal{F}(u(t^{n-1})) - \mathcal{G}(u_{k+1}^{n-1}) - \mathcal{F}(u_k^{n-1}) + \mathcal{G}(u_k^{n-1})\right\| \\
&= \left\|\mathcal{G}(u(t^{n-1})) + \delta\mathcal{G}(u(t^{n-1})) - \mathcal{G}(u_{k+1}^{n-1}) - \delta\mathcal{G}(u_k^{n-1})\right\| \\
&\leq \left\|\mathcal{G}(u(t^{n-1})) - \mathcal{G}(u_{k+1}^{n-1})\right\| + \left\|\delta\mathcal{G}(u(t^{n-1})) - \delta\mathcal{G}(u_k^{n-1})\right\| \\
&\leq (1 + L\Delta t)\left\|u(t^{n-1}) - u_{k+1}^{n-1}\right\| + C(\Delta t)^{m+1}\left\|u(t^{n-1}) - u_k^{n-1}\right\| \\
&\leq (1 + L\Delta t)\left\|u(t^{n-1}) - u_{k+1}^{n-1}\right\| + C(\Delta t)^{m+1}(\Delta t)^{mk}\|u_0\| \\
&\leq (1 + L\Delta t)\left\|u(t^{n-1}) - u_{k+1}^{n-1}\right\| + C(\Delta t)^{m(k+1)+1}\|u_0\|
\end{aligned}$$

At this point, note that the left hand term is the approximation of u at the previous time step, which we can assume to also be of the order $m(k+1)$. Therefore, we can say that $\left\|u(t^n) - u_{k+1}^n\right\| = \mathcal{O}(\Delta t^{m(k+1)})$, completing our inductive step. □

6.2 Numerical Results and Validation

Here we seek to confirm the theoretical order derived in the last section (from [1] [2]). To see a clear example, consier the Euler methods introduced back in section **??**. Recall that the explicit Euler method is of order 1, so theoretically if we have a fine method of high enough order, we should see a method whose order is k, for an order k Parareal. And indeed, as we expect, we do indeed see such such results, see figure 2 for the numerical plots.

Using this information, we might begin to consider how can we leverage this mk convergence rate. It's immediately clear, that if we want k to be as small as possible (as noted in the efficiency section), that we want m to be as large as possible. However, m corresponds to the order of the course integrator, and typically we want this integrator to be as cheap and inexpensive as possible. Therefore, we have an optimization problem here between the course integrator and number of parareal iterations. Of course, all of this is assuming that the fine integrator performs to the desired accuracy, otherwise the former is useless.

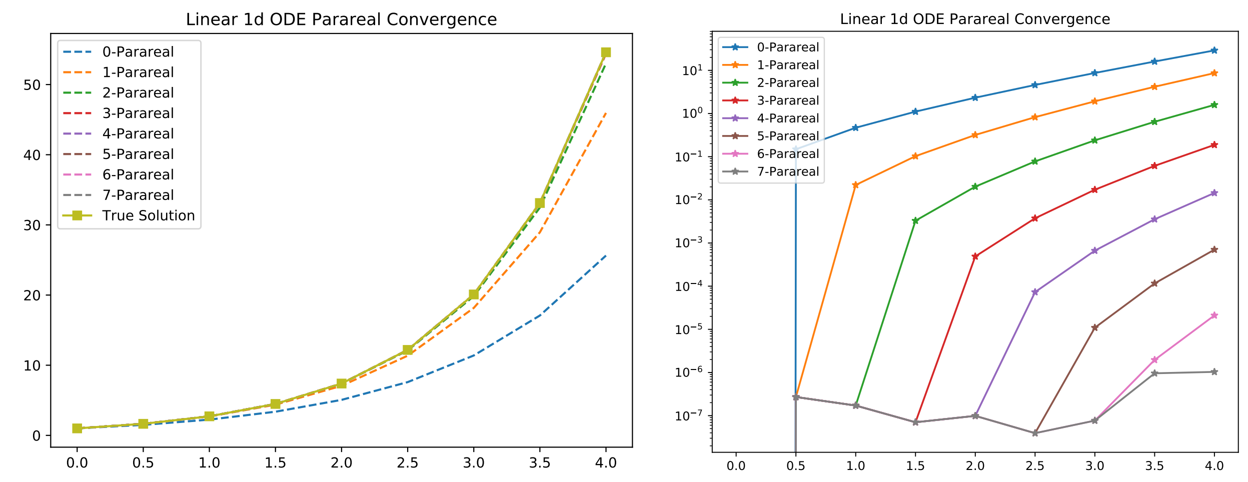

Figure 2: The figure on the left is showing the resulting solutions to the ODE $u' = u, u(0) = 1$ for $t \in [0, 4]$, and the right is showing the errors under the 1 norm. Critically, this plot reveals that our implementation of Parareal conforms to the theoreical convergence rate, which validates our method. The last iterate has strange behavior, but this is because I could not decrease the accuracy of the fine integrator (forward Euler) anymore without running out of memory.

For example, see figure 3 for a examination of the errors of parareal underneath the Runge-Kutta methods. In addition, see figure 4 for a more confirmation of the mk order, for the Runge-Kutta methods.

7 Conclusion

References

[1] Bal. *On the Convergence and the Stability of the Parareal Algorithm to solve Partial Differential Equations*. Columbia University, APAM

[2] Fields. *Parareal Methods*. `http://www.cfm.brown.edu/people/jansh/page5/page10/page40/assets/Field_Talk.pdf`

[3] Levecue. *Finite Difference Methods for Ordinary and Partial Differential Equations*. `https://epubs.siam.org/doi/abs/10.1137/1.9780898717839.fm`.

[4] Ruprecht. *A shared memory implementation of pipelined Parareal*. DOI:10.1007/978-3-319-64203-1_48.

[5] Staff and Rønquist. *Stability of the Parareal Algorithm*. Norweigian University of Science and Technology, Department of Mathematical Sciences.

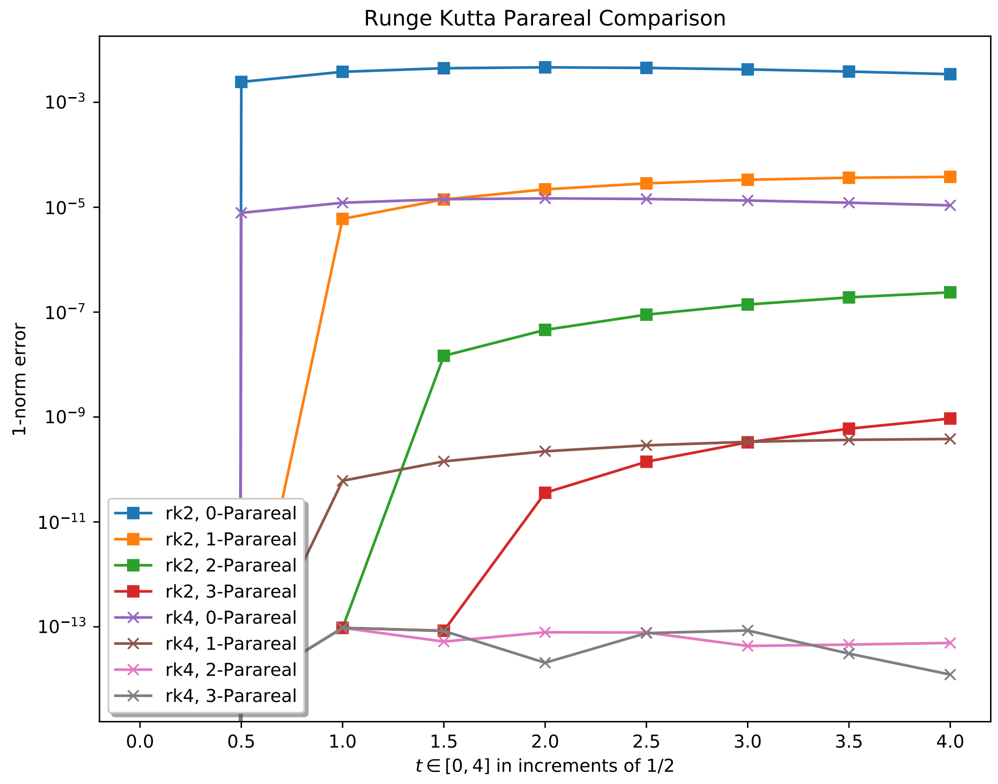

Figure 3: This figure is showing the resulting errors in the parareal approximation to to the ODE $u' = -\frac{1}{2}u, u(0) = 1$ for $t \in [0, 4]$ under the 1 norm. This plot reveals how, given a higher order course method, the parareal is able to make larger jumps of accuracy as the number of parareal iterations increases.

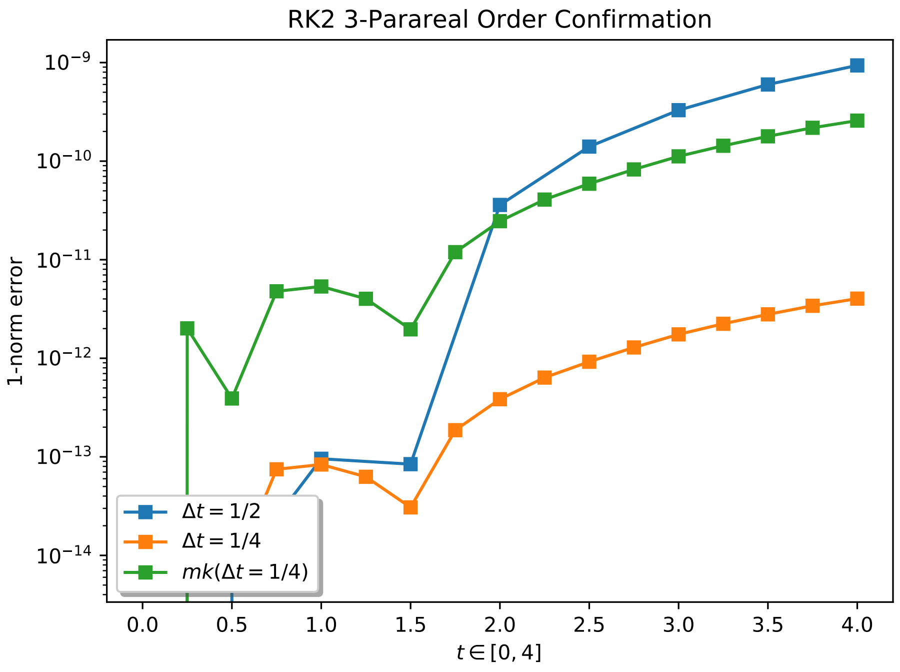

Figure 4: This figure shows that how, under refinement of Δt, the third parareal iteration of the Runge Kutta 2 method has order roughly 5. This is confirmed by taking the error of $\Delta t/2$ and shifting it by the order. They don't exactly land on top of each other, but I believe that to be an artifact of the approximation not being exact.