Project Proposal

Abhijit Chowdhary

April 29, 2019

Proposal | Parareal: Parallel in Time PDE Method

For both my High Performance Computing and Numerical Methods II courses, I have I have chosen to investigate the Parareal algorithm[1] for its theoretical numerical properties, and then try to optimize for efficiency as much as possible.

Parareal

Parareal is a parallel in time technique for the numerical solution of ODEs. Given a course and cheap solver $\mathcal{G}$ for the ODE, and a high accuracy and potentially more expensive solver $\mathcal{F}$, we first solve the sytem using $\mathcal{G}$, and in combination of ths solutions of $\mathcal{F}$ we correct the system in parallel. Thus, this is a *parallel in time* technique. The iteration would look like:[2]

$$y_{j+1}^{k+1} = \mathcal{G}(y_j^{k+1}, t_j, t_{j+1}) + \mathcal{F}(y_j^k, t_j, t_{j+1}) - \mathcal{G}(y_j^k, t_j, t_{j+1}).$$

See Figure 1 for an illustration of how the correction would occur in parallel.

Desired high performance computing points to investigate

The *parallel in time* capability of Parareal is its main selling point, and I plan to investigate how far I can take its efficiency. I propose to:

- Implement Parareal using OpenMP (or MPI/CUDA see Technical details section).
- Compare the result to prevailing serial algorithms to demonstrate the benefit (or lack of) of parallelism.
- Investigate the scalability of the algorithm, as the number of processors / (CUDA Threads) are increased. Pose problems for which this would be a good technique, and vice versa.
- Investigate individual choices of $\mathcal{F}$ and $\mathcal{G}$ to try and improve the computational intensity. In particular, try to choose $\mathcal{F}$ so that time spent serially computing is minimized and choose $\mathcal{G}$ so that to maximize parallel thread computation.
- Consider a combination of parareal and *parallel in the system* techniques during each parallel $\mathcal{G}$ run. For example, if we have more processors than the optimal for parallelizing the correction

[1] https://parallel-in-time.org/. A resource I've been using to research.
[2] https://en.wikipedia.org/wiki/Parareal

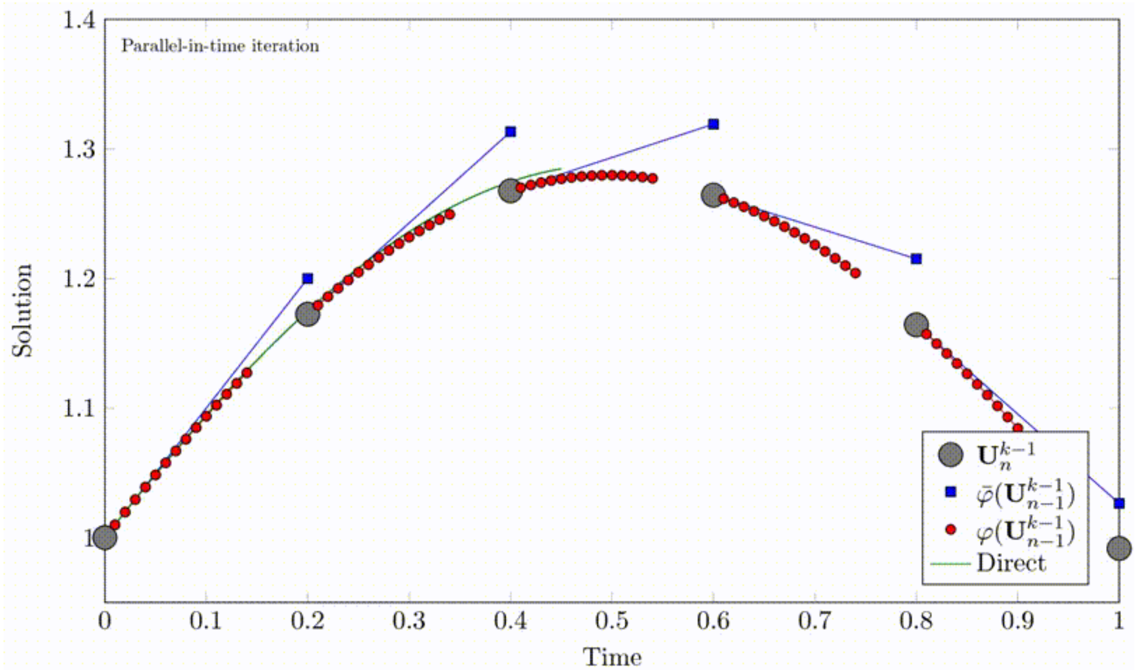

Figure 1: A sample Parareal solve mid iteration. $\hat{\phi}$ is the course solver, and ϕ is the fine solver. Credits wikipedia.

of $\mathcal{F}$, we can try to speed up the individual computation of each correction by using a $\mathcal{F}$ that can take advantage of the extra cores.

Specifically for the last two bullets, I'm not sure if there is a solution that makes sense and works well, but part of my investigation will be to see what doesn't work as well.

Desired theoretical points to investigate

While Parareal itself was built with computational efficiency in mind, there are quite a few interesting analytical properties of the method. While using it to solve a few model ODEs and perhaps the diffusion equation $u_t = \kappa u_{xx}$, I propose to try and experiment with different choices of fine and course solvers with the intention of:

- Examining the robustness of the method as we tend down Δt for different choices of solvers, and potentially examining accuracy as a function of the course/finer solver for fixed Δt.
- Examining the order of accuracy of the method as a function of $\mathcal{F}$ and $\mathcal{G}$, trying to find properties of $\mathcal{G}$ that would be best for the correction step of $\mathcal{F}$.
- Experiment with the resulting `parareal` on a stiff problem, and examine how to correct despite wanting an explicit solver for $\mathcal{G}$.
- Understanding what the region of stability seems to be for the combination of our methods $\mathcal{F}$ and $\mathcal{G}$ utilizing some linear model problems.
- One can also experiment with adaptive schemes in the fine solver, for example the Bogacki-Shampine method we wrote earlier in class.

I'm not sure how difficult it will be to derive analytical results on the following, though it seems that if we consider the method in serial and do our analysis on the resulting serial method, we might be able to use the theory we explored in chapters 6-8 in Levecue [3]to create criteria for convergence/stability.

Techincal details

Since this will be parallelized, it will have to be written in a language other that MATLAB/Octave, likely I will choose C++. As far as framework for parallelization, it will be performed on a CPU, so I have the options:

- OpenMP:
  - OpenMP is easy in terms of syntax and resource allocation.
  - I don't think OpenMP will scale to very large scale problems, as it seems the optimal number of processors to have here will be number of time steps of $\mathcal{G}$, which for stiffer problems will be large!
- MPI:
  - More low level, a bit more precarious, but with the potential to scale across the network.
  - Will probably be able to scale to as many processors as I could want, but I'm skeptical of how much the network bandwidth across systems will harm us when sending pieces of the course solver to be solved in parallel.

I will spend the first part of this project investigating the optimal parallelization technique to take, and then figuring out which framework works best for it.

Proposed Schedule

- Week 4/22-4/28:
  - Read papers and understand basic Parareal.
  - Begin writing basic serial version in Matlab for correctness (almost done).
- Week 4/29-5/05:
  - Finish writing Serial version, and port code to C++.
  - Write data structures and helper code for ODE systems and solvers.
  - Perform numerical analysis on convergence, stability, and robustness of methods as internal solvers vary.
- Week 5/06-5/12:
  - Take serial version and write basic OpenMP version.
    * Compare efficiency.
    * Model computational intensity, and think about how to maximize it.
  - Begin seriously writing and wrapping up an MPI version to scale to larger systems.
  - Once this is done, try it on Prince.
- Week 5/13-5/19:
  - Construct plots and figures for both report and presentation.
  - Finalize mathematics behind optimizing parallel efficiency and computational intensity. Confirm with professors about they accuracy.
  - Finish typesetting both.

[3]https://epubs.siam.org/doi/book/10.1137/1.9780898717839