Application of the parareal algorithm to CORSICA simulations of advanced plasma operation scenarios at ITER

D. Samaddar[1], T.A. Casper [1], S.H. Kim [1], L.A. Berry [2], W.R. Elwasif [2], D. Batchelor [2], W.A. Houlberg[1]

[1] *ITER Organization, Route de Vinon sur Verdon, 13115 Saint Paul Lez Durance, France*

[2] *Oak Ridge National Laboratory, Oak Ridge, USA*

Introduction

Detailed studies of plasma scenarios are considered to play a very important role in providing support to ITER design choices and in preparing for the successful operation of ITER. These studies are typically carried out using codes such as CORSICA [1, 2]. CORSICA combines a 2D free boundary equilibrium package with various transport and source models. These simulations are, however, computationally very intensive. Parallelization is the only way to shorten the wallclock time for these calculations, which is necessary for incorporating realistic physics and engineering constraints in these simulations.

For simulations of this nature, space parallelization has so far been the most common technique (to the best of our knowledge). Using CORSICA as a test-bed, this work explores the possibilities of parallelizing another domain, time, to achieve computational speedup. The parareal algorithm [3] is used to apply temporal parallelization to CORSICA. Preliminary results suggest that a significant computational gain may be achieved through temporal parallelization. Prior to its application to a complex system like turbulence [4], the parareal algorithm had been applied to relatively simpler problems [5, 6, 7, 8, 9]. Since plasma scenario simulations involve a large degree of non linearity and coupled physics, time parallelization of CORSICA involves new challenges that have previously not been encountered.

Parareal Algorithm

The parareal algorithm is described in detail in [3, 4]. It employs a predictor-corrector approach to split a time series into slices. Each slice is then solved in parallel across individual processors. A coarse solver (G) and a fine solver (F) are required for an implementation of the algorithm. G is characterized by a very small computation time as compared to F but yielding a coarse estimate of the solution to a time dependent initial value problem. G always operates as a serial process. F on the other hand, is computationally very slow but generates the correct estimate. F is applied in parallel, across individual processors. G and F are alternated across parareal iterations to achieve temporal parallelization. A solution across an individual processor is considered to have attained convergence when the relative error between two successive

fine solutions reaches a value less than or equal to a prescribed value set as the convergence criterion.

In case of CORSICA, the application has been achieved by the implementation of the parareal framework [10] written in Python. This framework has been developed at ORNL as part of the SWIM IPS project. As an alternative, parareal implementation could be carried out using MPI and OpenMP parallelization. That method is considerably more tedious. Moreover, such implementation is problem specific - which means each separate problem has to be treated individually. Using a Python based framework, such as the one used here, allows incorporating different problems in a relatively simpler fashion.

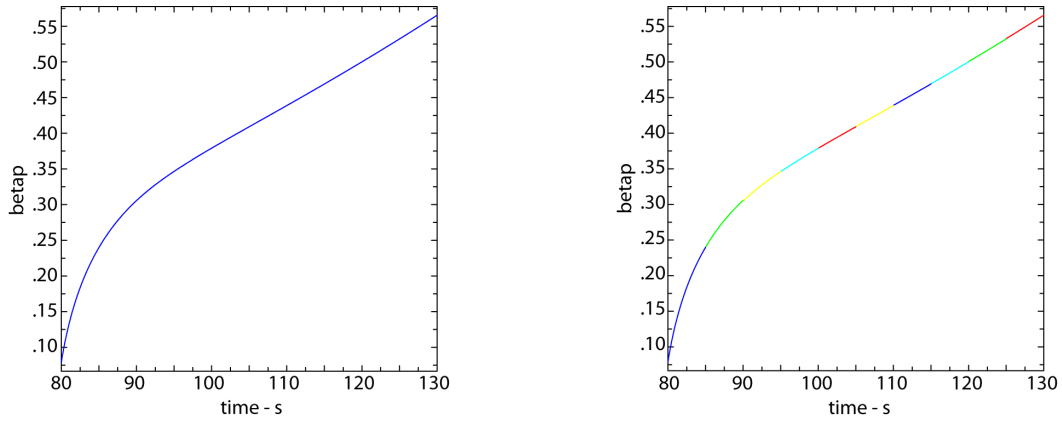
The framework also addresses another problem commonly faced with the application of the parareal algorithm. The prime challenge of a successful implementation depends on the choice of the optimum coarse solver, G for the best gain and efficiency. This task is relatively simpler when using the framework. The parareal framework depends on a I/O file based system which may often contribute to a computational overhead.

Results

The initial application of the parareal algorithm to CORSICA generates promising results. As the main goal in this work was to explore the applicability of the algorithm to a code as complex as CORSICA, a relatively simple case was chosen for the implementation. The simulations involved analytic sources as simulating the sources would make the problem more complex.

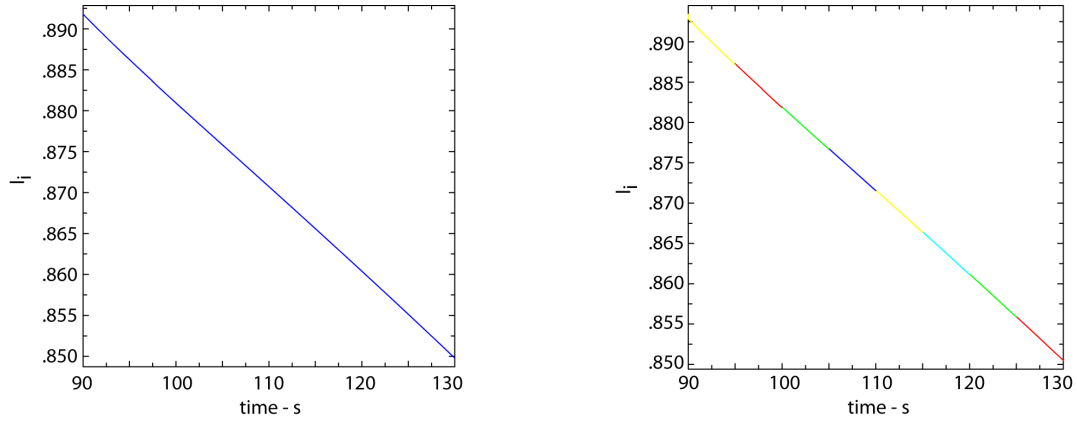
In this application, bigger timesteps were used to achieve the coarse solution. The coarse time step was 0.4 seconds while the fine solution required a time step size of 0.01 seconds. The typical walltime for a serial computation for 50 seconds is 13 hours when using short time steps (0.01 to 0.05 seconds). With the application of the parareal algorithm a computational gain of 8.32 was achieved with 12 processors. Figs. (1(a)) and (1(b)) represent the variation of the plasma poloidal beta (β_p) (normalized stored energy) with time. The variation of the internal inductance (li), a measure of the current density profile evolution, is shown in Figs. (2(a)) and Figs. (2(b)). Figs.(1(a)) and (2(a)) are the solutions obtained from a serial run on 1 processor, while Figs.(1(b)) and (2(b)) are the same ones obtained using the parareal algorithm across 10 processors. For the parareal simulations, the entire time series was split into slices, each of length 5 seconds, to be solved across multiple processors. Each color on the parareal solutions (Figs.(1(b)) and (2(b))) represents the calculation across a different processor. The error tolerance for parareal convergence was set to $5e-3$ for this simulation. The entire time series across 10 processors required only 2 parareal iterations to attain convergence. However, a longer time series across more processors may require an increased number of parareal iterations to attain

convergence.



(a) Serial solution (b) Parareal solution

Figure 1: The variation of poloidal beta β_p with time. The serial solution closely resembles the parareal solution. The colors on the plot for the parareal solution represent the calculations across different processors.



(a) Serial solution (b) Parareal solution

Figure 2: The variation of internal inductance li with time. The serial solution closely resembles the parareal solution. The colors on the plot for the parareal solution represent the calculations across different processors.

Conclusions and future work

Preliminary application of the parareal algorithm to CORSICA generates a significant computational gain (of 8.32 with 12 processors). The value may be expected to be enhanced with an increase in the number of processors.

More options for the choice of the coarse solver may be explored in the future. Future applications should also include increasing complexities in the simulations - for example, the introduction of simulated source terms, such as Monte-Carlo neutral-beam injection, ray tracing

electron-cyclotron heating, full-wave ion-cyclotron heating, free-boundary equilibrium boundary control. These applications may introduce restrictions to the choice of the coarse solver and may require multi level concurrencies.

A successful implementation of the parareal algorithm to CORSICA makes the study of plasma scenarios at ITER significantly more feasible. The reduced simulation times would allow for systematic parameter variation studies needed for scenario evolution and discharge development and validation. The performance can be greatly enhanced if the temporal parallelization is combined with spatial parallelization to achieve maximum computational gain.

Disclaimer

The views and opinions expressed here do not necessarily represent those of the ITER Organization.

References

[1] J A Crotinger et al, LLNL Report UCRL-ID-126284; NTIS PB2005-102154. (1997).

[2] T A Casper et al, 23rd Int. Conf. on Fusion Energy Conference (Daejeon, Korea) ITR/P1-19, accepted for publication in Nuclear Fusion (2010).

[3] J. Lions, Y. Maday and G. Turinici, CR Acad. Sci. I - Math. 332 (7), pp. 661-668 (2001).

[4] D. Samaddar,D. E. Newman and R. Sanchez, Journal of Computational Physics 229 (18), pp. 6558-6573(2010).

[5] L. Baffico, S. Bernard, et. al, Phys. Rev. E 66 (5), 057706 (2002).

[6] G. Bal, Y. Maday, Lecture Notes in Computer Science and Engineering, vol. 23, Springer, Berlin, 2002.

[7] I. Garrido, M.S. Espedal, G.E. Fladmark, Proceedings of Fifteenth International Conference on Domain Decomposition Methods, vol. 40, Springer, Berlin, pp. 469-476.

[8] P.F. Fischer, F. Hecht, Y. Maday, Proceedings of Fifteen International Conference on Domain Decomposition Methods, vol. 40, Springer Verlag, 2004, pp. 433-440, (2004).

[9] M.J. Gander, E. Hairer, Domain Decomposition Methods in Science and Engineering XVII,vol. 60, Springer, pp. 45-56, (2008).

[10] L. A. Berry, W. Elwasif, J. Reynolds, D. Samaddar, R. Sanchez and D. E. Newman, Journal of Computational Physics (In Press).