D Samaddar et al.

Temporal parallelization of edge plasma simulations using the parareal algorithm and the SOLPS code

Preprint of Paper to be submitted for publication in
Computer Physics Communications

Temporal parallelization of edge plasma simulations using the parareal algorithm and the SOLPS code

D. Samaddar[a,*], D.P. Coster[b], X. Bonnin[c,d], C. Bergmeister[a], E. Havlíčková[a],
L.A. Berry[e], W.R. Elwasif[e], D.B. Batchelor[e]

[a]*CCFE, Culham Science Centre,Abingdon, Oxon, OX14 3DB, UK*
[b]*Max-Planck-Institut für Plasmaphysik, Germany*
[c]*CNRS-LSPM, Université Paris 13, Sorbonne Paris Cit, Villetaneuse,France*
[d]*ITER Organization, Route de Vinon sur Verdon, 13115 St Paul Lez Durance, France*
[e]*Oak Ridge National Laboratory, Oak Ridge, USA*

Abstract

It is shown that numerical modelling of edge plasma physics may be successfully parallelized in time. The parareal algorithm has been employed for this purpose and the SOLPS code package coupling the B2.5 finite-volume fluid plasma solver with the kinetic Monte-Carlo neutral code Eirene has been used as a test bed. The complex dynamics of the plasma and neutrals in the scrape-off layer (SOL) region makes this a unique application. It is demonstrated that a significant computational gain (more than an order of magnitude) may be obtained with this technique. The use of the IPS framework for event-based parareal implementation optimizes resource utilization and has been shown to significantly contribute to the computational gain.

Keywords: time parallelization, parareal algorithm, edge plasma, scrape off layer (SOL), SOLPS, Eirene, magnetically confined plasmas
PACS: 02.70.-c, 52.40.Hf, 52.55.Fa, 52.65.-y, 02.90.+p, 07.05.Tp

1. Introduction

One of the biggest challenges in computational science is often to find the optimum balance between feasible wallclock time and the complexity of the

---

*Corresponding author
Email address: `Debasmita.Samaddar@ukaea.uk` (D. Samaddar)

simulation. Modern supercomputers with ever increasing number of cores require codes that sufficiently scale to maximize resource utilization. Parallelizing these computations is an option - and traditional techniques of parallelization, such as space parallelization indeed reduce the computational time, but leave a lot of room for further improvement. This is often because the computational gain reaches saturation and increasing the number of processors does not necessarily improve the gain beyond a certain point. There are also parts of the code-packages where standard parallelization does not have a large impact and hence act as bottlenecks in terms of computational time. This acts as a motivation to explore another dimension for parallelization which is time. Moreover, it must be clarified that temporal parallelization is not a replacement for any existing form of parallelization, but may be added to them to further, often significantly, improve performance. Temporal parallelization using the parareal algorithm was originally proposed in [1] in 2001, and has been applied to a variety of problems from relatively simple ones [2, 3, 4, 5, 6, 7, 8] to ones with complex dynamics [9, 10, 11, 12].

This work explores the application of the parareal algorithm in an area in fusion plasma. Simulations involving the edge and divertor regions of fusion devices are extremely challenging due to their complex physics as well as requirements for large wall-clock times. These computations involving the plasma edge are important for modeling plasma surface interactions along with designing the wall of fusion devices. The dynamics is complex at the scrape-off layer (SOL) - which is the region between the device wall and the Last Closed Flux Surface (LCFS). Impurities, which may originate primarily from plasma interactions with the materials of the wall, may greatly dilute the plasma and negatively affect the plasma heating. The presence of impurities is also necessary, to some extent, for radiative cooling of the divertor region to mitigate the very high heat and particle fluxes onto the divertor targets, thereby achieving (partial) plasma detachment. A comprehensive modelling of the SOL is one of the greatest challenges of magnetically confined fusion plasmas. The SOL is characterized by open field lines. In addition to dominant parallel flows along these field

lines, perpendicular flows become important in this region of the plasma. Also, alongside charged particle transport, neutral particle transport is dominant in some parts of the SOL. Neutral particles are externally introduced into the plasma through neutral beam injection, gas puffs and outgassing and removed by pumping. Processes within the plasma like recycling, sputtering and volume recombination are also hugely responsible for the presence of neutrals at the plasma edge of a fusion device [13, 14], and processes like ionization contribute to reducing their number. A range of codes, such as BOUT[15, 16], UEDGE[17], DIVIMP[18], ERO-JET[19], ASCOT[20], Edge-SOL[21], SOLPS[22, 23], EMC3-Eirene[24] and EDGE2D-Nimbus[25], SOLEDGE-2D[26] with varied degrees of complexity have been used to study different aspects of the plasma edge. Incorporating optimum physics into these simulations is often a challenge due to the computational cost. With simulations targeting more ITER-like plasmas, the wallclock time may often exceed weeks or months. An example of such cases are studies involving particle-balance in the simulations [27, 28, 29].

Exploring the advantages of the parareal algorithm in edge plasma simulations is different in many ways from previous applications. Fully developed turbulence simulations in the steady state where the statistical study of properties such as the probability distribution function, vorticity and velocity are relevant are dealt with in [9, 10]. [11] applies to burning plasma scenarios but the details of the physics of the edge involving impurities and neutral transport are not given significant importance. This work studies the application of the parareal algorithm to SOL simulations using the SOLPS-Eirene package [22]. It is shown that if the algorithm is optimally applied, a speed-up of more than a factor of ten may be achieved for the cases studied here. The SOLPS-Eirene code package has been extensively used for existing tokamaks[22, 23, 29] as well as island divertor configurations [30, 31] and is also being used at ITER [29, 32]. This application is significantly different from previous applications. The application to the SOLPS-Eirene package is unique and promising as it allows the possibility of applications to core-edge coupled codes for efficient simulations of ITER-like plasmas. The parareal implementation was performed using the IPS

framework developed at ORNL, USA, as part of the SWIM project [33, 34]. The framework allows the use of 'event-based parareal' [35] which significantly improves performance and is discussed in more detail in sections 2.3 and 3.3.

This paper is organized as follows. Section 2 describes the parareal algorithm and the techniques employed for its application. Section 2.1 gives a brief summary of the model employed for the parareal application. The results are described in section 3 and the findings are summarized in section 4.

2. Parareal algorithm

With increasing demands for reducing wallclock time and improving the scaling of computational gains with cores on modern supercomputers, parallelization of simulations has gained tremendous importance. Spatial parallelization is a common approach although the technique often reaches saturation preventing optimum resource usage. Exploiting the time domain allows one to utilize more cores and further speed up the code. Temporal parallelization is non-intuitive as it apparently violates causality. However, a number of predictor-corrector approaches to achieve time-parallelization (PITA [36], PFASST [37], parareal [1]) have been proposed which have garnered increasing attention in recent years. The technique of the parareal algorithm has been discussed in detail in [1, 9]. It involves dividing a time series into slices which are solved in parallel. This is illustrated in fig. 1, where individual time slices may be considered as t_0 to t_1, t_1 to t_2, t_2 to t_3 and so on. Each individual time slice is solved in parallel by different processors represented by P0, P1, P2, etc in the figure. The algorithm requires a coarse predictor (G) which is computationally fast but gives only a coarse estimate of the solution. The fine (F) solver or the corrector is computationally slower but generates a solution with a higher accuracy. A parareal iteration is represented by k each of which comprises of a G and F run. The coarse solver is always applied serially across processors, while the fine computation is performed in parallel, leading to a computational speed-up. At $k = 1$, G is applied to give each processor an initial value. Then, F is applied to that

4

(wrong) initial value in parallel across every time slice. At the start of each subsequent parareal iteration, k, the parareal correction is applied to the initial value across each processor (or time slice, i) and this correction is given by eq.1.

$$\lambda_{i+1}^k = F(\lambda_i^{k-1}) + G(\lambda_i^k) - G(\lambda_i^{k-1}) \tag{1}$$

λ_{i+1}^k is the initial state for the $(i+1)^{th}$ time slice at the kth iteration. At the kth iteration λ_i^k is evolved to λ_{i+1}^k using $F(\lambda_i^k)$ and $G(\lambda_i^k)$.

The coarse and fine calculations are repeated across a number of parareal iterations until convergence is achieved. Parareal convergence across a processor is said to have been achieved when the relative error between two successive fine calculations (k and $k-1$) across that processor is below a tolerance value. The sum of the relative error is defined by

$$\sigma_i^k = \int_{t_{i-1}}^{t_i} \frac{\left|\lambda^k(t) - \lambda^{k-1}(t)\right|}{|\lambda^{k-1}(t)|} dt. \tag{2}$$

The solution is then converged for time slice i if,

$$\sigma_i^k < \text{tol}, \quad \forall i \leq I. \tag{3}$$

2.1. SOLPS-Eirene code package

The interactions between the plasma and the wall in tokamaks and other fusion devices play an important role in the success of the experiment. The region of interest for this purpose is known as the scrape-off layer or SOL. The SOL is bounded by the wall of the machine and the last closed flux surface as shown in fig.2. The charged particles travel along the field lines and reach the divertor or target plates. A variety of configurations may be possible, two of which have been explored in this work. Depending on the number of null points or x-points in the magnetic field, a tokamak may have single, double or multiple (snowflake) null geometry. One of the tokamaks explored in this work for parareal simulations in the SOL is MAST with double null geometry and two sets of upper and lower divertors, illustrated in fig.2. DIII-D is the other fusion device simulated here and it has single null geometry and only lower
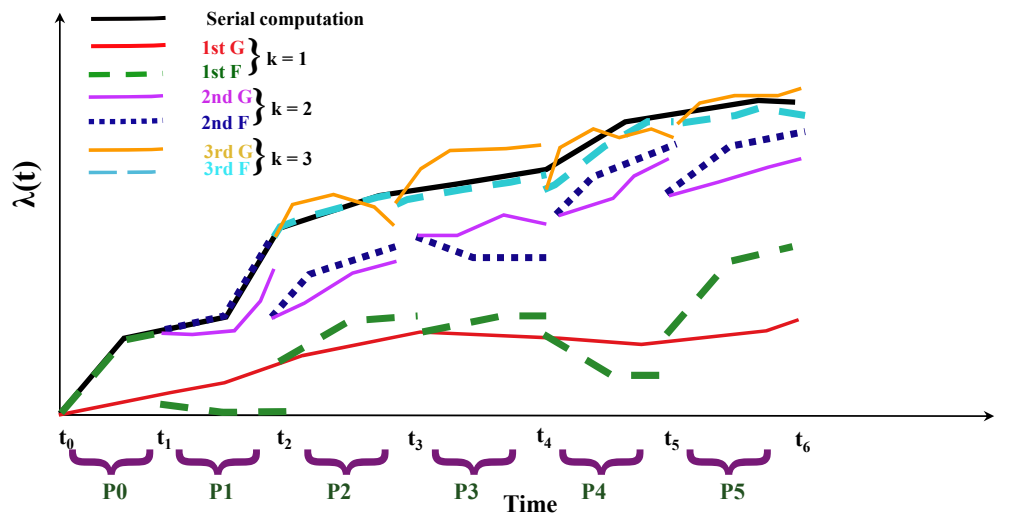
Figure 1: The parareal algorithm involves a coarse(G) and a fine(F)solver in each parareal iteration, k. After several iterations, the parareal solution approaches that achieved by serial computation.

divertors. The purpose of this paper was demonstrating the effectiveness of the parareal algorithm in simulations of two separate machines thus paving the way to broadening of the areas of application. In the SOL, the plasma may be treated as a fluid while the neutral particles may be simulated as Monte-Carlo histories. The SOLPS (Scrape Off Layer Plasma Simulator) code package consists of the 2D multifluid transport code B2.5 coupled with a neutrals transport model. The B2.5 model is based on the Braginskii equations [38] in magnetically aligned curvilinear orthogonal coordinates. The set of equations include the continuity and momentum conservation equations for each ion species, current continuity equation and the energy balance equations for both ions and electrons, which are described in detail in [22, 39]. Electron density is determined via quasineutrality. In the cases explored in this work, the $\mathbf{E} \times \mathbf{B}$ and diamagnetic drifts [40] are absent.

The transport of neutrals is modelled using the Eirene package[23]. Eirene uses Monte-Carlo treatment of neutral particle transport solving the Boltzman equation for distribution functions for neutrals. The kinetic model is detailed in physics and computationally robust but is expensive in terms of wall-clock time. In fact, Eirene is often responsible for significantly slowing SOLPS simulations.

2.2. Coarse solvers for SOLPS

As has been discussed in [9, 10, 11], a variety of choices may be employed to achieve shorter wallclock time accompanied by coarseness in the solution. However, the choice is certainly non trivial. A coarse solver that is computationally fast may actually deviate the solution very far away from the fine computation, which may lead to a high number of parareal iterations to achieve convergence or convergence may not be achieved at all. On the other hand, the solver may be relatively slower, but the solution may be corrected quickly to approach the fine solution requiring fewer parareal iterations. The condition of the 'coarseness' of the fast solver has been outlined in [1], but it is almost impossible to translate that for a complex physics problem.

Two coarse solvers or G of the solution have been explored in this work for

various implementations of the parareal algorithm to the SOLPS code package. As already mentioned in section 2.1 the fine or F version of SOLPS utilizes B2.5 along with Eirene for calculating the neutral transport in the plasma. Since Eirene is computationally expensive, replacing it by a fluid neutrals model simplifies the computation. As an example, the source term, S^n in the continuity equation in its most general form given by eq.4 is computed differently in the two models.

$$\frac{\partial n}{\partial t} + \nabla \cdot \vec{flux} = S^n \tag{4}$$

The reduced model as well as the Monte-Carlo technique is discussed in detail in [22, 41, 42, 43]. Modifying the transport coefficients (and eventually boundary conditions) for the neutrals (and ion energy equation) to improve the coarse estimate reduced the number of parareal iterations required for convergence. The results of using the neutral fluid model in place of Eirene are discussed in section 3.1. The second coarse solver is implemented using a coarse mesh yielding very interesting results. The technique of converting between coarse and fine grids is discussed in detail in [44]. Using a coarser mesh in space allows one to take bigger timestep sizes without violating the CFL condition. This technique is very useful in significantly speeding up the computation and the results are discussed in section 3.2. The two coarse solvers studied in this work are independently implemented. A combination of the two is likely to yield the most computational gain.

2.3. Event-based parareal with IPS framework

The parareal implementation of SOLPS was achieved using MPI paralleliza- tion via the IPS (Integrated Plasma Simulator) framework. The IPS framework written in Python [33, 34, 35] launches an event-based parareal application using MPI. In the traditional technique, for every parareal iteration k, all processors need to undergo 'mpi-wait' until computations for every time slice is completed for a given k. This significantly increases wall-clock time.

The 'event-based parallelism' makes optimum use of resources possible. Here, for any iteration k, if the coarse calculation has been completed for a time slice, $i = \alpha$ (say), the fine computation may be done on all time slices with $i \leq \alpha$, while the coarse calculation may be carried out simultaneously on slices $i > \alpha$. The positive impact of the event-based approach on the parareal gain and efficiency is discussed in section 3.3. Moreover, the IPS framework simplifies the implementation of the algorithm in comparison to a traditional MPI approach to any code. Unlike a standard MPI application which involves introducing significant changes to the original serial code, there was no rewriting of the underlying physics codes in case of the IPS framework.

3. Results

With the purpose of exploiting the parareal algorithm in edge plasma simulations, our studies involved two tokamaks with different geometries. MAST is a spherical tokamak with double null geometry as described in fig. 2.The other machine used as a testbed for our simulations is DIII-D, which has a single x-point and hence single null geometry. Fig. 3 shows the fine mesh for SOLPS parareal simulations which is the same as the one used for serial computations.

The cases explored in this work had low density at the targets and the temperature of the plasma at the targets were ($\sim 40eV$) in MAST [39] and ($\sim 10eV$) in DIII-D. For both machines, the plasma species consisted of Deuterium and Carbon ions. The primary variables of the code are ion density (na, m^{-3}), electron density (ne, m^{-3}), electron (te, eV) and ion temperatures (ti, eV), parallel velocity ($u_{\parallel}$, m/s) and electrostatic potential (ϕ, V). The parareal correction, given by eq.1 was applied to these six variables at the beginning of a time slice for a given parareal iteration, k. The maximum total power fluxes (W/m^2), pwmxip and pwmxap, on the inboard and outboard divertor respectively were chosen as convergence metrics and are represented by λ in eq.2. After eq.2 is applied to both these variables, convergence at a given iteration k is said to be achieved when the errors for both these variables are $\leq 5 \times 10^{-2}$. The
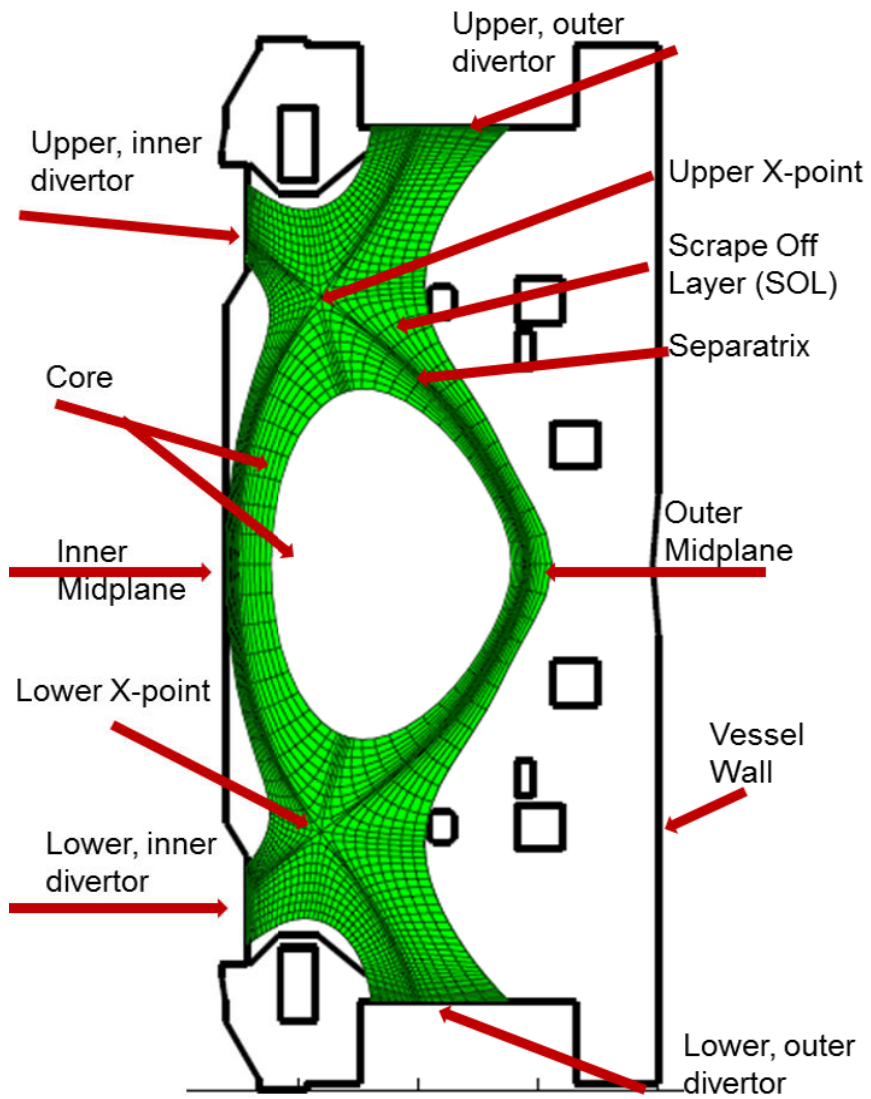
9

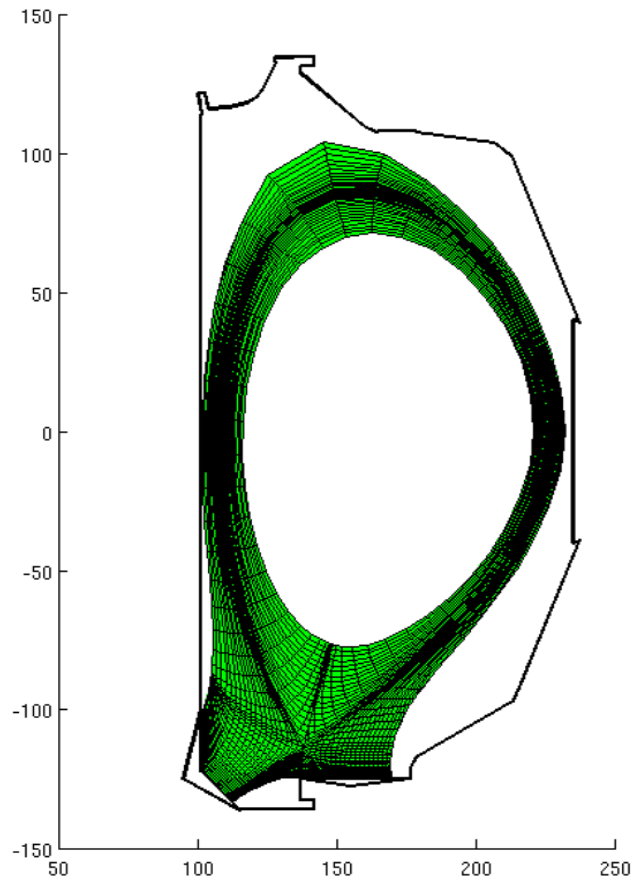Figure 2: The mesh used for SOLPS simulations of MAST.

Figure 3: The fine grid layout of the DIII-D tokamak used for SOLPS simulations. This grid of size 96×36 was used for serial computations as well as the 'fine' or F run of the parareal implementation.

11

computational gain is calculated by comparing fine and coarse run times, given by t_F and t_{PR} respectively, over the same simulation length. So, computational gain is defined as

$$gain = \frac{T_F}{t_{PR}} \tag{5}$$

3.1. Coarse solver using neutral fluid model

Replacing the kinetic model in Eirene with the neutral fluid model speeds the simulation by approximately 49 times but the accuracy of the solution is compromised. For example, the outer (nesepm1) and inner (nesepm2) midplane separatrix densities for the electrons are shown in figs. 4 and 5, where the final desired fine computation is significantly different from the initial coarse estimate. This coarse predictor with simplified physics was used in simulations of MAST.
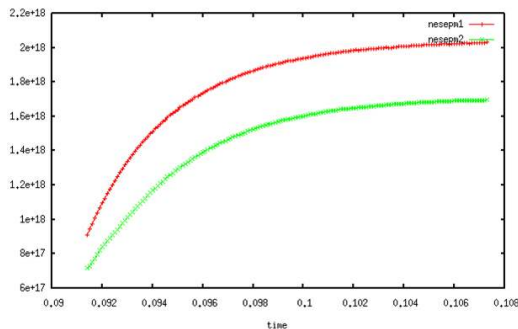

Figure 4: Density calculated using the SOLPS-Eirene package (fine solver). The parareal computation matches this solution.
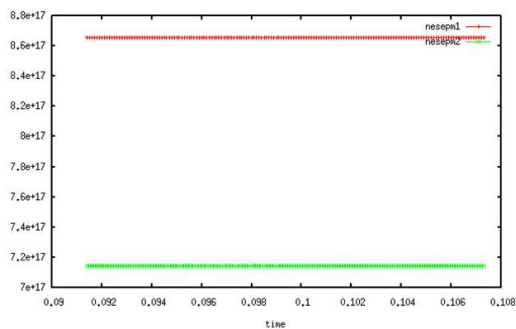
Figure 5: Density computed using the coarse solver (Eirene replaced by fluid neutral model) is different from the fine solution.

Using the case of fig.5 as a coarse predictor, the fine solution may be recovered in much shorter wall-clock time using the parareal algorithm. With increasing parareal iterations, k, the solution approaches the value that is typically obtained using a serial computation with the fine solver (as in fig.4). The evolution of the parareal solution for the density at the separatrix (nesepm) across iterations is illustrated in fig.6. Figs. 7a, 7b, 7c demonstrate that the
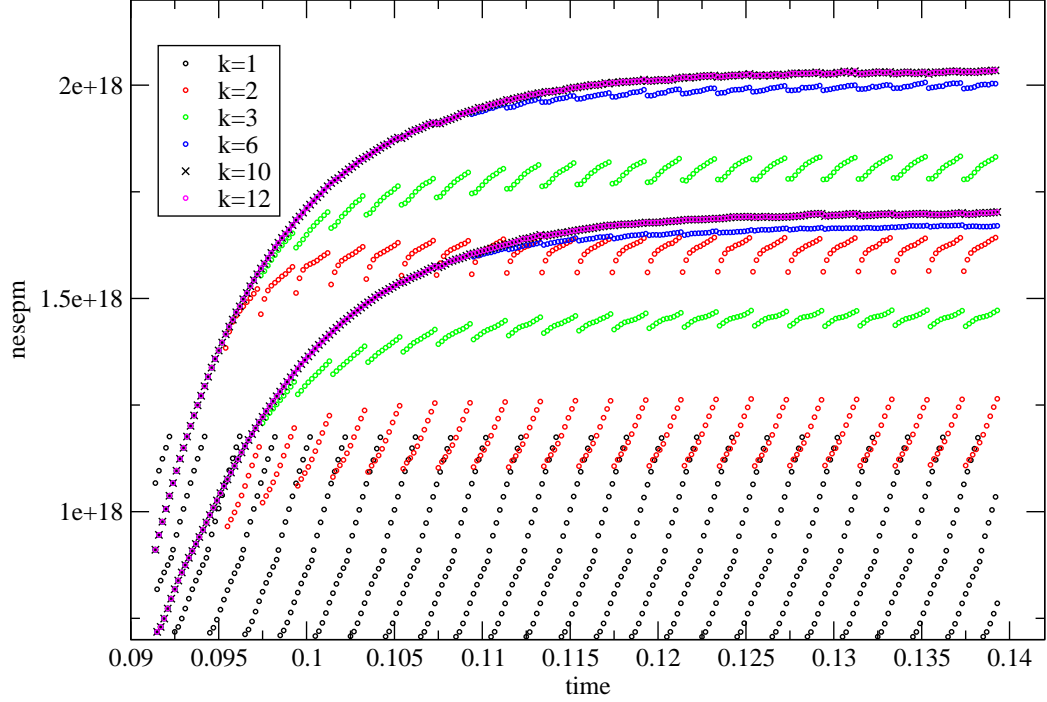
12

Figure 6: The plot of density at the separatrix plotted for various parareal iterations, k. At the final iteration, $k = 12$, the solution matches that of the fine solver in fig.4.

parareal solution (in this case, the maximum total power flux at the outer divertor) is identical to the serial solution, despite starting the simulation with the coarse computation in fig. 7a. Since MAST has two sets of divertors (upper and lower) resulting from the double null configuration shown in fig. 2, in the plots figs. 7a, 7b, 7c, $pwmxap_1$ refers to the lower outer divertor target while $pwmxap_2$ represents the value at the upper outer divertor target.
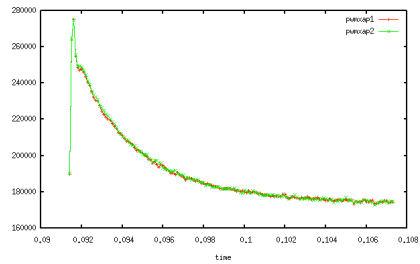
A gain of 12.58 was obtained with 240 processors for the case just discussed, where each processor solved a timeslice of 10 steps. Convergence was achieved
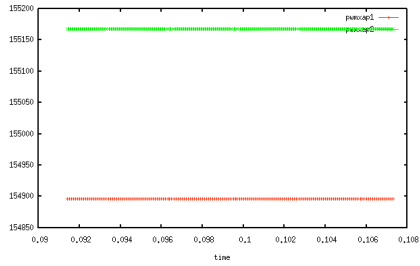
(a) Coarse(G) solution.
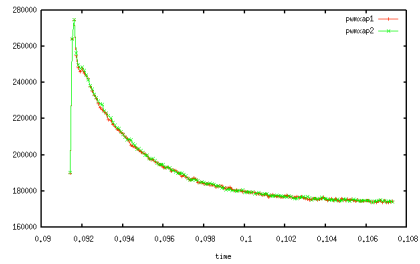
(b) Fine(F) solution.

(c) Parareal solution.

Figure 7: Maximum total power flux at the outer divertor. $pwmxap_1$ and $pwmxap_2$ are values at the lower and upper divertors respectively. For the coarse (G) computation, the transport of neutral particles was computed using a fluid model. The final parareal solution matches the serial solution.

14

after $k = 12$ parareal iterations in this case. It was interesting to explore if the size of the timeslice had an impact on the parareal convergence. It was observed that the size of 10 was optimum. Parareal convergence was not attained when the number of timesteps solved on each time slice was ≤ 5 or ≥ 40. For slices ≤ 5, it is likely that the simulation per processor was not long enough to attain the fine values before it was restarted with a new initial value for the next iteration. This is illustrated in fig.8 where even after $k = 11$ parareal iterations, the relative error between $k = 11$ and $k = 10$ is not below $5E - 2$.

It was also observed that an unlimited increase of the size of the timeslice solved per processor may negatively impact convergence. This is likely because the coarse solution is allowed to deviate very far from the fine solution which essentially violates the condition for parareal correction to rectify the solution as formulated in [1]. A similar behaviour with parareal simulations has been observed and studied in [9, 10].

3.2. Coarse solver using reduced grid and bigger timesteps

Increasing the timesteps in the simulation allows faster computation but may introduce inaccuracy. The increase in timestep size is also limited by the CFL condition. The second coarse solver explored in this work is using a reduced grid with bigger timesteps for faster computations. The parareal implementation was carried out in simulations on both MAST and DIII-D cases. The grid size in a simulation is given by nx×ny where nx is the mesh size in the poloidal direction and ny is that in the radial direction. The fine computation for MAST used a grid given by 150×36, while for DIID simulations, the grid was of size 96×36. Various coarse grids were explored for both cases. For coarse computations of MAST simulations, grids of size 150×18, 76×36 and 76×18 were used. Grids 48×36 and 48×18 were explored for the same purpose in DIII-D simulations. Using a reduced grid implies bigger spatial step sizes, which allows bigger timesteps (dt) without violating the CFL condition [45].

The technique of using the reduced grid model for coarse (G) computations yielded excellent results for both MAST and DIII-D studies. For DIII-D simu-
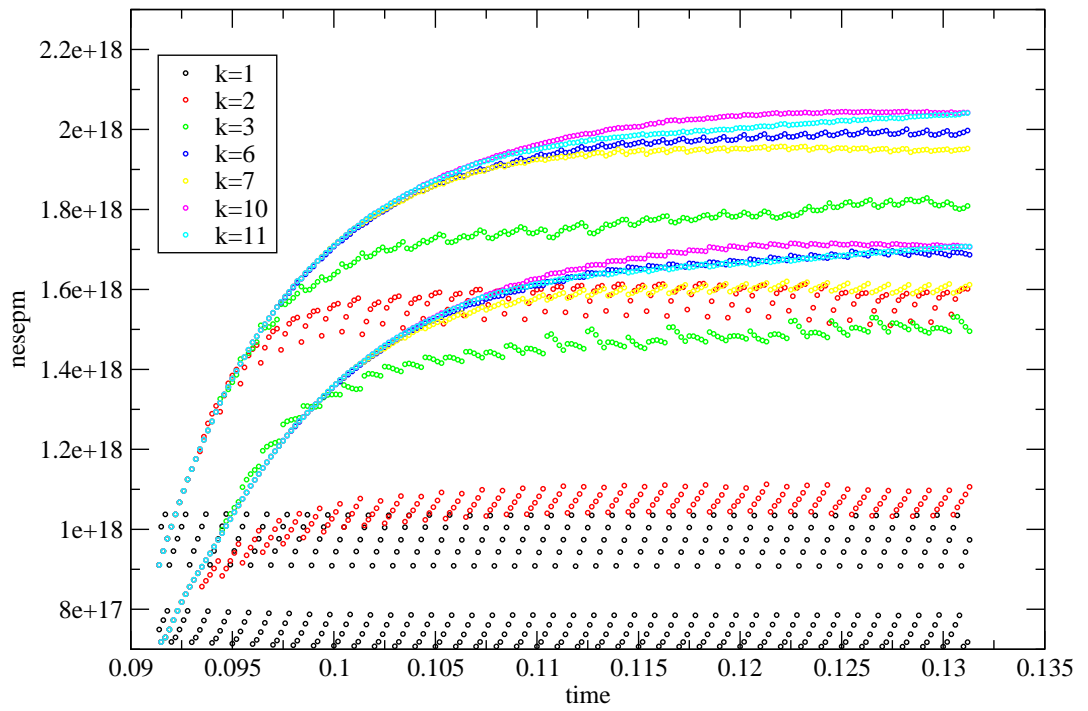
Figure 8: When the coarse solver used the fluid neutral model in place of Eirene, parareal convergence was not obtained when the size of the time-slice per processor was ≤ 5. The solutions never settle at a steady value even at large k (parareal iteration) until $k = i$ where i is the processor number or time slice.

lations, a coarse grid size of 48×36 and $dt_G = 30dt_F$ generated a computational gain (given by eq. 5) of 21.8 using 96 processors. A reduced grid of 150×18 and $dt_G = 8dt_F$ similarly yielded a parareal gain of 15.9 with 64 processors in MAST simulations. dt_F and dt_G are the fine (F) and coarse (G) timesteps respectively. The evolution of the solution in case of MAST with increasing parareal iteration k is illustrated as an example in fig.9. 'pwmxip' is the maximum total power flux impinging on the upper inboard divertor.

However, it was once again observed as in section 3.1 that the size of the time slice (number of timesteps solved per processor) strongly influenced the number of parareal iterations required for convergence. As a test case in DIII-D with 16 processors, fig. 10 illustrates this point. $NTIM_F$ and $NTIM_G$ are the number of timesteps solved on each processor for fine (F) and coarse (G) computations, respectively. It may be observed that there is a 'sweet spot' in the relation between $NTIM_F$ and $NTIM_G$ where the performance is best. Although the ratio of $NTIM_F : NTIM_G$ may stay constant, the computational gain changes depending on the exact values of $NTIM_F$ and $NTIM_G$. A similar behavior for MAST simulations was also observed as is illustrated in fig. 11. Here, the parareal iteration k required for convergence depended on $NTIM_F$, keeping $NTIM_G$ constant.

Finally, a scaling study further supports the above points. DIII-D cases are used as examples for this case. For the weak scaling where the size of a time-slice per processor is fixed but the number of time-slices are varied, $NTIM_F = 450$ and $NTIM_G = 15$ were chosen in fig. 12. The gain can always be maximized by optimizing $NTIM_F/NTIM_G$.

The strong scaling is shown in fig. 13 in a case where $\frac{NTIM_F}{NTIM_G} = 12.5$. In case of strong scaling, the total simulation length is fixed while the number of processors is varied. As a result, for high processor count, the time slice solved per processor (and hence $NTIM_F$ and $NTIM_G$) is significantly reduced. The reduction in the values of $NTIM_F$ and $NTIM_G$ reduces the gain reflecting the observations in fig. 10. It may also be noted that the performance using the coarse solver with reduced grid was more robust than the one in section
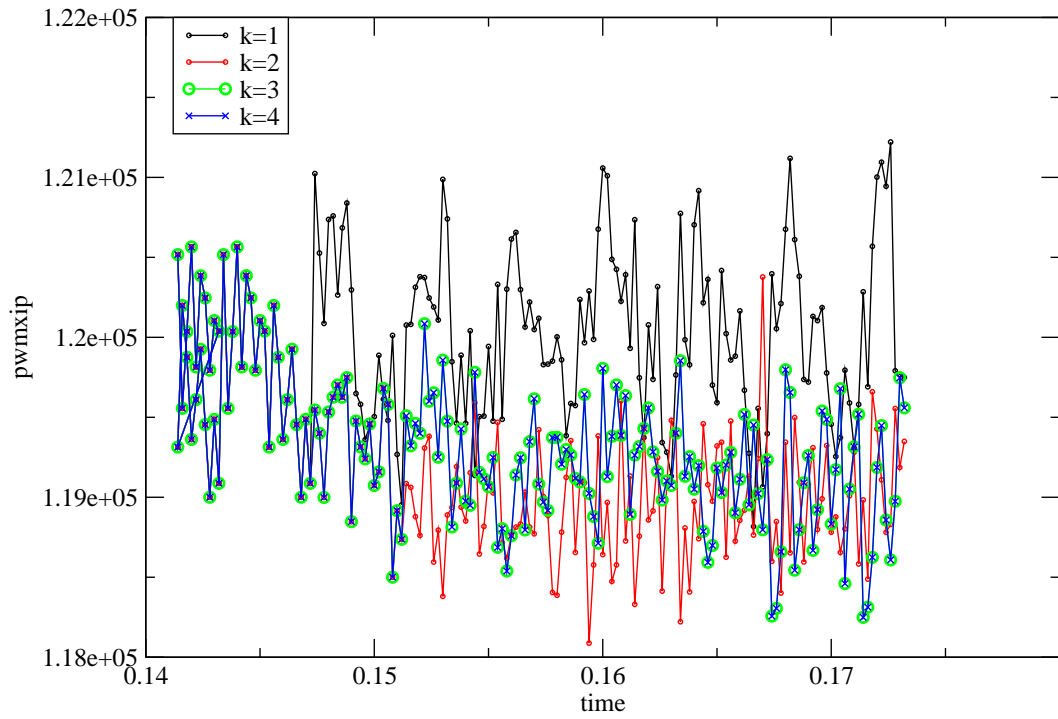
17

Figure 9: The parareal algorithm converges in 4 iterations for this case where the coarse model is a reduced grid. The relative error between the total power fluxes at the inboard divertor for iterations $k = 3$ and $k = 4$ is below $5E - 2$.
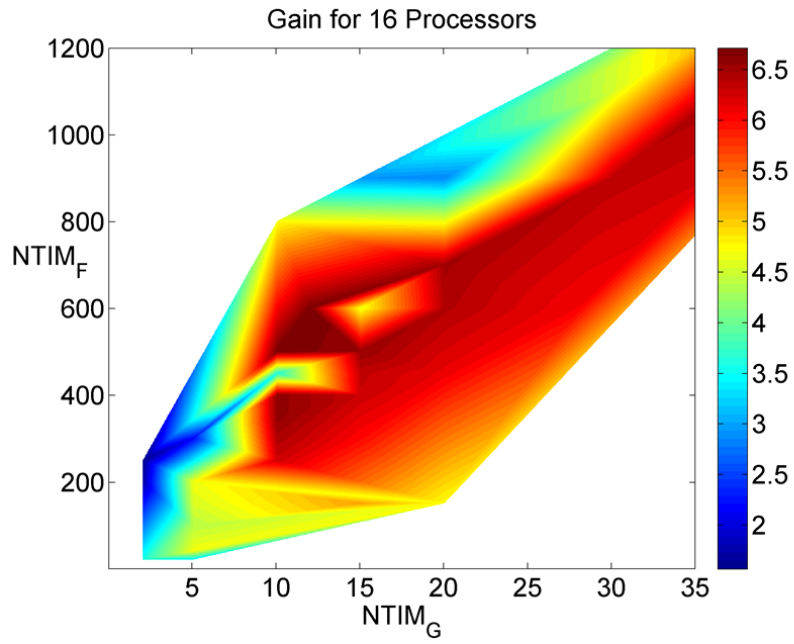
Figure 10: The parareal gain depends on the number of timesteps ($NTIM_F$ for fine and $NTIM_G$ for coarse) solved per processor.
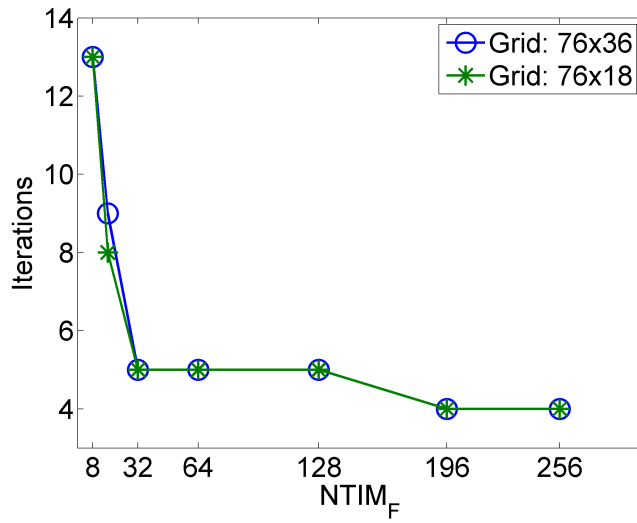

Figure 11: For the same number of timesteps solved per processor for the coarse computation, the parareal iteration k required for convergence depends on the number of timesteps used for the fine computation ($NTIM_F$).
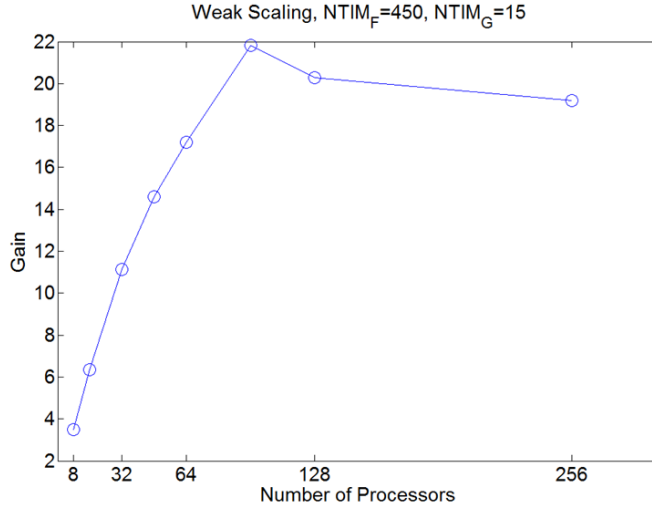
Figure 12: Weak scaling studies. Here $NTIM_F = 450$ and $NTIM_G = 15$.

3.1 as a wider range of timeslices were allowed where parareal convergence was achievable.

3.3. Parareal gain with IPS framework

The IPS (Integrated Plasma Simulator) framework was used to carry out the parareal implementation in this work. This allowed an event-based approach which positively impacted the computational gain.

For a long time series divided into N slices using N processors, let t_G be the time taken to perform the coarse(G) computation on a single time slice and let t_F be the same for a fine computation on a single time slice as well. If at iteration k, n_c slices have converged then $n_k = N - n_c$ slices remain to perform a G and F calculation. In a traditional MPI implementation, with G being a serial process and F being performed in parallel, the wallclock time required per iteration k can then be stated as:

$$t_k = t_G * n_k + t_F \tag{6}$$

If K is the total number of iterations required for convergence of all N time slices, then summing across all iterations gives the total time for a tradional
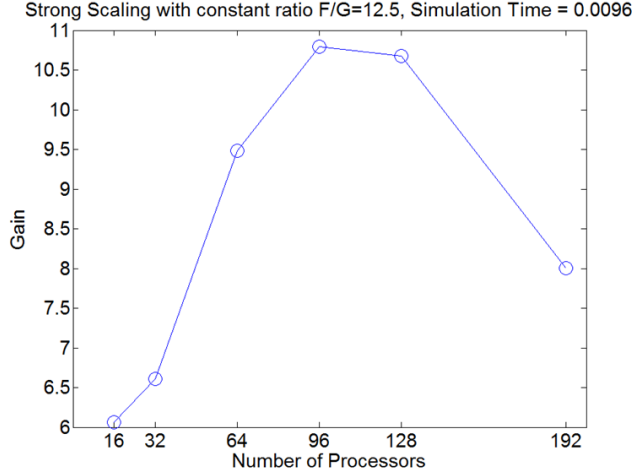
20

Figure 13: Strong scaling studies. The gain decreases at high processor count when the size of the time slice solved per processor is significantly reduced.

parareal implementation using MPI.

$$T_{trad} = \sum_{k=1}^{K} t_k \tag{7}$$

The computational gain for traditional implementation of the parareal algorithm using MPI may then be stated as:

$$gain_{trad} = \frac{t_F * N}{T_{trad}} \tag{8}$$

The fact that the event-based approach significantly improves performance is illustrated in fig.14 where the plot for the event-based implementation using IPS (in red) is generated by actual simulation results discussed in section 3.1. The other plot (in black) is a theoretical estimate using eq.(8) of the traditional MPI implementation of the algorithm. It may be added that the theoretical calculation of computational gain in fig.14 ignores MPI communication overheads. This clearly demonstrates the efficiency of the framework, where the computational gain grows almost linearly, while with traditional MPI, the gain would have flattened at a much smaller value.

21

As already mentioned in section 2.3, the implementation of the parareal scheme to any case is simplified by the use of the farmework. There are no massive changes to the original physics code (SOLPS) that are required, and the user can focus on varying and optimizing the coarse solvers instead of the numerics of the scheme.

4. Conclusion

This work has shown that time parallelization greatly contributes to speeding up of edge plasma simulations. The parareal algorithm has been applied to the SOLPS-Eirene code package and a gain larger that 20 has been achieved which is significant for edge simulation in plasmas. It has also been shown that the use of event-based parareal via the IPS framework has a very strong impact on the computational gain. The parareal approach was explored on simulations of two different machines performing experiments of magnetic fusion - namely the DIII-D and MAST tokamaks. Both studies generated promising results. The parareal algorithm, which uses a predictor-corrector technique, requires a coarse and a fine solver. The fine solver uses SOLPS B2.5 and Eirene code packages and is relatively computationally intensive. The choice of the coarse solver as in any work involving the parareal algorithm, is non trivial. Two coarse solvers have been explored in this work. The Eirene package, which solves the Boltzman equation for a kinetic treatment of neutral particles, has been replaced by a simpler fluid neutral model to serve as a coarse predictor. This has allowed significant speed-up of the code, but a strong sensitivity to the size of the time slice (number of simulation time steps solved per processor) has been observed. A size of 10 appears to be the optimum value for the cases considered. The second coarse solver uses a reduced grid model accompanied by bigger simulation time steps (dt). This coarse solver appears to have a more robust performance than the one discussed earlier and it, too, leads to significant computational gain. Scaling studies have also been performed. It was observed that the computational gain using the parareal algorithm depends on the size of
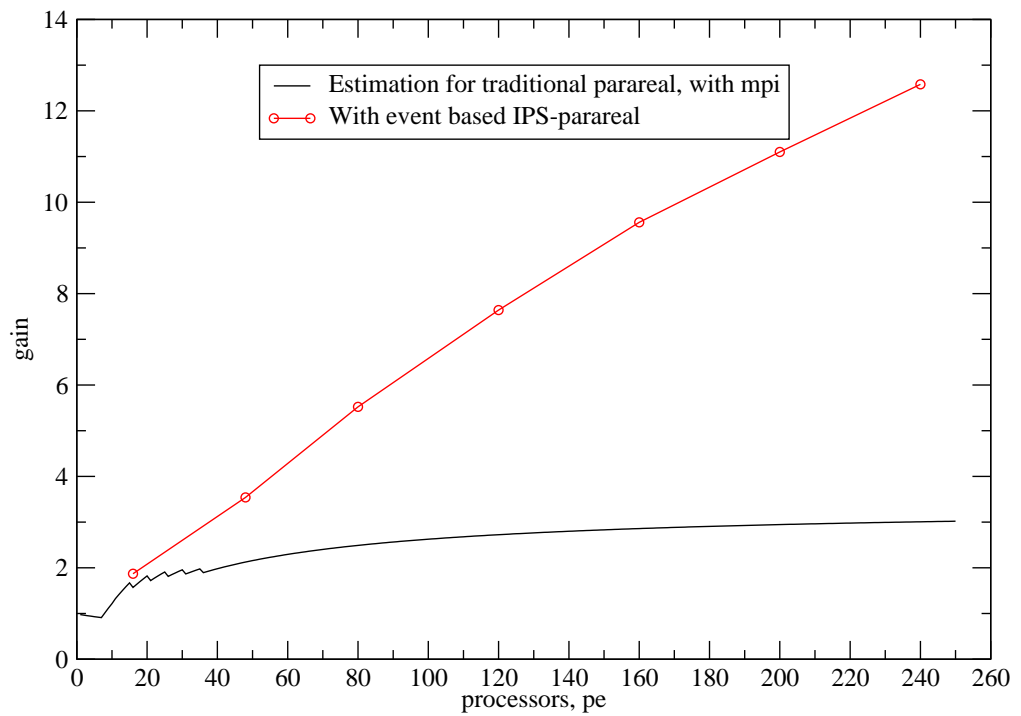
Figure 14: The computational gain is significantly larger when the event-based parareal is applied using the IPS framework. A theoretical estimate of the gain using traditional MPI implementation shows a much lower value and quick saturation with processor numbers.

time slices solved on each processor. The computational gains obtained may also be further improved by combining the two coarse solvers along with additional levels of parallelism, such as, space parallelization. The idea of the coarse solvers explored in this work may be expanded to other areas of computational physics. A reduced mesh for the coarse domain may be explored wherever applicable although limitations to their extent may be expected. Using reduced physics for the coarse model should be attempted with care, as has been done here, to ensure that the parareal correction at every iteration prevents the solution from deviating from the fine simulation.

5. Acknowledgements

References

[1] J. Lions, Y. Maday, G. Turinici, A "parareal" in time discretization of PDE's, Comptes Rendus de l'Académie des Sciences - Series I - Mathematics 332 (7) (2001) 661–668.

[2] L. Baffico, S. Bernard, Y. Maday, G. Turinici, G. Zérah, Parallel in time molecular dynamics simulations, Phys. Rev. E 66 (5) (2002) 057706.

[3] G. Bal, Y. Maday, A parareal time discretization for non-linear PDEs with application to the pricing of an American put, Vol. 23 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2002, p. 189202.

[4] G. Bal, Parallelization in time of (stochastic) ordinary differential equations (2003).
URL http://www.columbia.edu/~gb2030/PAPERS/paralleltime.pdf

[5] P. F. Fischer, F. Hecht, Y. Maday, A parareal in time semi-implicit approximation of the Navier-Stokes equations, in: Proceedings of Fifteen International Conference on Domain Decomposition Methods, Vol. 40, Springer Verlag, 2004, pp. 433–440.

[6] M. J. Gander, E. Hairer, Nonlinear convergence analysis for the parareal algorithm, in: Domain Decomposition Methods in Science and Engineering XVII, Vol. 60, Springer, 2008, p. 45.

[7] I. Garrido, M. S. Espedal, G. E. Fladmark, A convergent algorithm for time parallelization applied to reservoir simulation, in: Proceedings of Fifteen International Conference on Domain Decomposition Methods, Vol. 40, Springer, Berlin, 2004, p. 469.

[8] G. Staff, G. Rønquist, Stability of the parareal algorithm, in: Proceedings of Fifteen International Conference on Domain Decomposition Methods, Springer Verlag, 2003, pp. 449–456.

[9] D. Samaddar, D. E. Newman, R. Sánchez, Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm, Journal of Computational Physics 228 (2010) 6558.

[10] J. Reynolds-Barredo, D. E. Newman, R. Sánchez, D. Samaddar, L. A. Berry, W. R. Elwasif, Mechanisms for the convergence of time-parallelized, parareal turbulent plasma simulations, Journal of Computational Physics 231 (2012) 7851.

[11] D. Samaddar, T. Casper, S. Kim, L. A. Berry, W. R. Elwasif, D. B. Batchelor, W. Houlberg, Application of the parareal algorithm to advanced operation scenario simulations of ITER plasma using the CORSICA code, Applied Mathematics and Computation (submitted).

[12] A.-M. Baudron, J.-J. Lautard, Y. Maday, M. K. Riahi, J. Salomon, Parareal in time 3D numerical solver for the LWR Benchmark neutron diffusion transient model, Journal of Computational Physics (submitted).

[13] P. C. Stangeby, The plasma boundary of magnetic fusion devices, Vol. 7, Taylor & Francis, 2000.

[14] J. Wesson, Vol. 149, Oxford University Press, 2011.

[15] X. Q. Xu, R. H. Cohen, Scrape-Off Layer Turbulence Theory and Simulations, Contrib.Plasma Phys. 38 (1-2) (1998) 158.

[16] X. Q. Xu, R. H. Cohen, T. D. Rognlien, J. R. Myra1, A fully implicit, time dependent 2-D fluid code for modeling tokamak edge plasmas, Journal of Nuclear Materials 7 (2000) 1951.

[17] T. Rognlien, J. Milovich, M. Rensink, G. Porter, Scrape-Off Layer Turbulence Theory and Simulations, Contrib.Plasma Phys. 196-198 (1992) 347.

[18] P. Stangeby, J. Elder, Calculation of observable quantities using a divertor impurity interpretive code, DIVIMP, Journal of Nuclear Materials 196 (1992) 258.

[19] A. Kirschner, V. Philipps, J. Winter, U. Kögler, Simulation of the plasma-wall interaction in a tokamak with the Monte Carlo code ERO-TEXTOR, Nucl. Fusion 40 (5) (2000) 989.

[20] J. Heikkinen, T. Kiviniemi, T. Kurki-Suonio, A. Peeters, S. Sipilä, Particle Simulation of the Neoclassical Plasmas, Journal of Comp. Phys. 173 (2) (2001) 527.

[21] O. E. Garcia, N. H. Bian, Bursting and large-scale intermittency in turbulent convection with differential rotation, Phys. Rev. E 68 (2003) 047301.

[22] R. Schneider, X. Bonnin, K. Borrass, D. P. Coster, H. Kastelewicz, D. Reiter, V. A. Rozhansky, B. J. Braams, Plasma Edge Physics with B2-Eirene, Contrib.Plasma Phys. 46 (1-2) (2006) 3.

[23] P. B. D. Reiter, M. Baelmans, The EIRENE and B2-EIRENE Codes, Fusion Science and Technology 47 (2) (2005) 172.

[24] Y. Feng, F. Sardei, J. Kisslinger, P. Grigull, K. McCormick, D. Reiter, 3D Edge Modeling and Island Divertor Physics, Contrib. Plasma Phys. 44 (1) (2004) 57.

[25] R. Simonini, G. Corrigan, G. Radford, J. Spence, A. Taroni, Models and Numerics in the Multi-Fluid 2-D Edge Plasma Code EDGE2D/U, Contrib.Plasma Phys. 34 (2-3) (1994) 368.

[26] H. Bufferand, G. Ciraolo, L. Isoardi, G. Chiavassa, F. Schwander, E. Serre, N. Fedorczak, P. Ghendrih, P. Tamain, Applications of SOLEDGE-2D code to complex SOL configurations and analysis of Mach probe measurements, Journal of Nuclear Materials 415 (1, Supplement) (2011) S589S592.

[27] A. S. Kukushkin, H. D. Pacher, Divertor modelling and extrapolation to reactor conditions, Plasma Phys. Control. Fusion 44 (6) (2002) 931.

[28] A. Loarte, et al., Power and particle control, Nucl. Fusion 47 (2007) S203.

[29] A. S. Kukushkin, H. D. Pacher, V. Kotov, G. Pacher, D. Reiter, Finalizing the ITER divertor design: The key role of SOLPS modeling, Fusion Engineering and Design 86 (12) (2011) 2865.

[30] G. Herre, P. Grigull, R. Schneider, B2-Eirene code modelling of an island divertor, Journal of Nuclear Materials 266-269 (1999) 1015–1019.

[31] X. Bonnin, R. Schneider, D. Coster, V. Rozhansky, S. Voskoboynikov, Electric fields and currents in an island divertor configuration, Journal of Nuclear Materials 829835, 14th Int. Conf. on Plasma-Surface Interactions in Controlled Fusion Devices (2001) 7.

[32] R. A. Pitts, A. Kukushkin, A. Loarte, A. Martin, M. Merola, C. E. Kessel, V. Komarov, M. Shimada, Status and physics basis of the ITER divertor, Physica Scripta 2009 (2009) T138.

[33] W. Elwasif, D. Bernholdt, A. Shet, S. Foley, R. Bramley, D. B. Batchelor, L. A. Berry, The Design and Implementation of the SWIM Integrated Plasma Simulator, in: Distributed and Network-based Processing (PDP), Pisa, Italy, 2010.

[34] W. R. Elwasif, S. Foley, D. Bernholdt, L. A. Berry, D. Samaddar, D. E. Newman, R. Sanchez, A dependency-driven formulation of parareal: parallel-in-time solution of PDEs as a many-task application, in: MTAGS '11 Proceedings of the 2011 ACM international workshop on Many task computing on grids and supercomputers, ACM New York, NY, USA, 2011, p. 15.

[35] L. A. Berry, W. R. Elwasif, J. Reynolds-Barredo, D. Samaddar, R. Sanchez, D. E. Newman, Event-based parareal: A data-flow based implementation of parareal, Journal of Comp. Phys. 231 (17) (2012) 5945.

[36] C. Farhat, M. Chandesris, Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications., Int. J. Numer. Meth. Engng. 58 (9) (2003) 1397.

[37] M. Emmett, M. L. Minion, Toward an efficient parallel in time method for partial differential equations, Communications in Applied Mathematics and Computational Science 7 (1) (2012) 105.

[38] S. I. Braginskii, Transport Processes in a Plasma, Vol. 1, Consultants Bureau, New York, 1965, p. 205.

[39] E. Havlíčková, W. Fundamenski, F. Subba, D. Coster, M. Wischmeier, G. Fishpool, Benchmarking of a 1D scrape-off layer code SOLF1D with SOLPS and its use in modelling long-legged divertors, Plasma Phys. Control. Fusion 55 (6) (2013) 065004.

[40] V. Rozhansky, S. Voskoboynikov, E. Kaveeva, D. Coster, R. Schneider, Simulation of tokamak edge plasma including self-consistent electric fields, Nuclear Fusion 41 (4).

[41] D. P. Coster, Reduced physics models in SOLPS for reactor scoping studies, Contributions to Plasma Physics (2015) Accepted.

[42] D. Coster, X. Bonnin, B. Braams, D. Reiter, R. Schneider, the ASDEX Upgrade Team, Simulation of the Edge Plasma in Tokamaks, Physica Scripta 2004 (T108) (2004) 7.

[43] D. P. Coster, Detachment physics in SOLPS simulations, Journal of Nuclear Materials - Proceedings of the 19th International Conference on Plasma-Surface Interactions in Controlled Fusion. Volume 415, Issue 1, Supplement (2011) S545S548.

[44] X. Bonnin, H. Bürbaumer, R. Schneider, D. Coster, F. Aumayr, H. P. Winter, The Two-Mesh Grid Refinement Method for the B2 Code, Contributions to Plasma Physics 42 (2) (2002) 175.

[45] R. Courant, K. Friedrichs, H. Lewy, Über die partiellen Differenzengleichungen der mathematischen Physik, Mathematische Annalen 100 (1) (1928) 32.