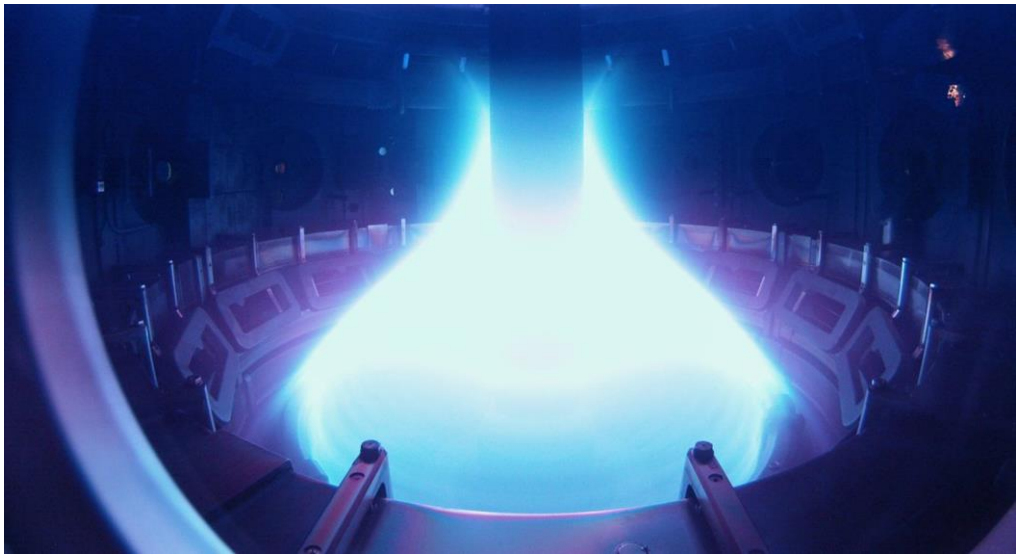
EPSRC
Engineering and Physical Sciences Research Council

InFoMM
Industrially Focused Mathematical Modelling

EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling



Mathematical formulation of the coarse predictor for the Parareal algorithm in fusion plasma divertor simulations

Federico Danieli

UNIVERSITY OF OXFORD

CCFE
CULHAM CENTRE FOR FUSION ENERGY

Contents

1. Introduction

Background

The *Culham Centre for Fusion Energy* (CCFE) is a leading research centre in the development of the technology necessary to attain a sustained fusion reaction. Achieving this ambitious task would represent a major step forward towards the solution of the energetic problem, providing a chance to tap from a sustainable, clean and durable energy source.

Much of this challenge consists in controlling the behaviour of the hot plasma inside the fusion reactor, for example by mean of strong induced magnetic fields. However, the physical laws that describe the whole system are of extreme complexity, and in most cases not yet fully understood. To gain more insight on the phenomena that occur inside a fusion reactor, researchers at CCFE rely heavily on numerical simulations, that allow the approximate evolution of the system to be determined, starting from some basic laws that describe its dynamics. These laws are expressed as *Partial Differential Equations* (PDEs), which are solved using the computational power provided by supercomputers.

Parallel computing

The power of a single CPU is limited. Parallel computing allows for multiple CPUs to work together on the same task, thus reducing the time necessary to complete it.

The growing complexity of the PDEs involved in the description of plasma physics demands ever increasing computational power in order to obtain numerical solutions in reasonable amounts of time. The frequency of computations that can be carried onto a single processor is, however, limited and represents an upper bound for the efficiency of serial algorithms. To overcome this limit, much attention has been directed towards the implementation of algorithms that work in parallel, thus providing an alternative to sheer computational power to speed up the computation of numerical solutions.

If the quantity we are solving for depends on position in space, then one way to parallelise consists on subdividing this space into smaller subdomains, and have different processors taking care of finding the solution in each of these. For example, if we solve a model to determine the evolution of the temperature inside a room, we might assign half of the room to one CPU, and the other half to another one. After the solutions are found, we have to match them at the interface of the subdomains. The matching comes with a cost, so further subdividing might be detrimental to the purpose of speeding up the computation.

Parallelising over a time interval is challenging: the solution in the future depends on values from the past

Nevertheless, if the solution depends on time, we can still think of applying similar techniques, and subdivide a time interval instead of a space interval, thus allowing for *time parallelisation*. Performing this task is not straightforward, and the additional challenge lies in the fact that, in this case, there is a specific direction in the flow of information: the solution at previous instants impacts that at following ones. This makes time parallelisation counterintuitive, but despite these difficulties many algorithms have been proposed that can make use of this additional dimension for parallelisation. The *Parareal* algorithm is one such algorithm. Our aim is to study ways to improve its performance, for the purpose of speeding up plasma physics simulations.

Glossary of terms

- Differential equations: The evolution of a physical system can be described by means of differential equations, which describe laws that define how some quantities (like temperature or density of a gas) vary. These equations are called *ordinary differential equations* (ODEs) if only changes in one variable (usually time) are considered; if variations in more variables (like space and time) are linked, then the equations are called *partial differential equations*. If these variations do not

depend explicitly on the position or instant considered, but are only influenced by the state of the system itself, then the equations are said to be *autonomous*. Another classification is between *linear* and *non-linear* equations: the first ones are much simpler, while the latter present non-trivial interactions that can give rise to much more complicated behaviours.

- Numerical solver: For many complex systems of differential equations, an explicit analytical solution is not available. To overcome this issue, we can be satisfied with approximating such solution to various degrees of accuracy using a computer. The algorithm used to compute such solution numerically is called a numerical solver (or scheme, method, integrator, etc). We focus on two different schemes: a *fine solver*, which is more accurate and more expensive to use, and a *coarse solver*, which is cheaper to compute but less precise.
- Interpolant: This is a curve that connects some points (called samples). For example, if we are interested in describing the evolution in time of the temperature inside a room, we can imagine measuring such temperature at constant intervals (sampling), and then draw a curve that passes through the measured values, so to be able to guess the temperature even at instants in-between. There are many ways to achieve this, from the naïve "connect-the-dots" using straight lines to more sophisticated methods that generate smoother curves.

2. Description of the Parareal algorithm

To parallelise in time, we first make a guess for the solution at the beginning of each subinterval and then iteratively correct the guess

The first step in the Parareal algorithm consists of dividing the time interval over which we are interested in tracking the evolution of our solution into subintervals (time chunks) of equal size. We assign the task of computing the solution on each time chunk to an independent processor. However, in order to start the computations, we need to know the values of the solution at the beginning of each subinterval, so that we can evolve it from there. These values are called *initial conditions*. These are, of course, not known *a priori*, so we need to make a guess for the initial conditions, *verify* them, and possibly *correct* them iteratively until we reach a precise enough result. Verifying our guess is simple: we integrate using a fine solver, and we check the eventual mismatch between final conditions in one subinterval and initial conditions in the following one. The procedure for correcting (or updating) the initial conditions so that they are more accurate at the next iteration is not as straightforward. The Parareal algorithm uses the following formula to conduct the update:

$$\lambda_i^{k+1} = \mathcal{F}\left(\lambda_{i-1}^{k}\right) + \mathcal{G}\left(\lambda_{i-1}^{k+1}\right) - \mathcal{G}\left(\lambda_{i-1}^{k}\right),$$

where λ_i^k represents the initial conditions. The subscript i indicates the time chunk the initial condition refers to, while the superscript k indicates the iteration of the algorithm. We see how the update for each subinterval depends on values from the previous time chunk, collected both before (λ_{i-1}^{k}) and after (λ_{i-1}^{k+1}) the update. $\mathcal{F}(\lambda)$ represents the result of integrating using the fine solver, starting from λ, while $\mathcal{G}(\lambda)$ is the result from the application of the coarse solver. Moreover, in order to start the computation, we need an initial guess for the very first iteration. This can be provided by an initial sweep of the coarse solver on the whole time domain. We present a schematic of the initialisation of the algorithm in Figure 1. The solution we are interested in approximating is represented by the black curve. In order to parallelise the computation, we divide the time domain of interest (from 0 to T) into a number N of subintervals of equal size. The initial conditions for each of the subintervals, λ_i^0, are guessed by applying the coarse solver serially (red dotted line). The blue lines represent the evolution of the solution within each time chunk, as computed using the fine integrator. We note that, in general, the results from the numerical integrations at the end of each time chunk do not correspond to the initial conditions we chose for the following time chunk. Hence, we cannot assume that we can

connect together all the bits of the solution in one single continuous curve, and there is a corresponding error in the representation of the actual evolution. Iteratively applying the update formula is expected to reduce such error.

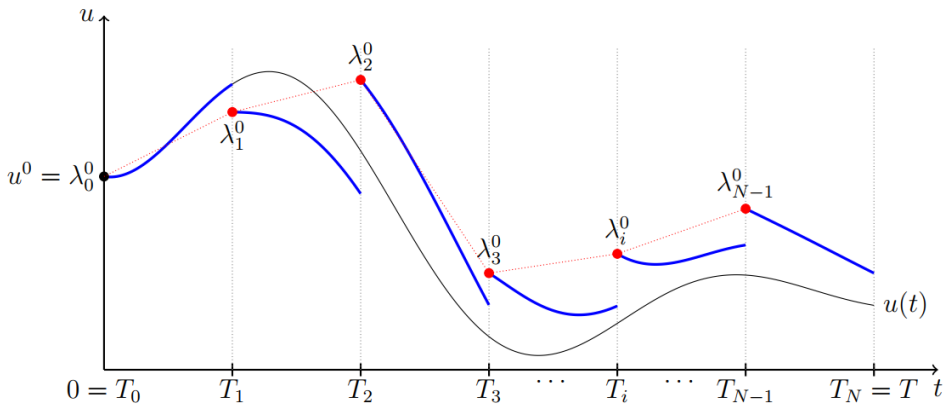


Figure 1: Schematic illustrating an example of the initialisation of the Parareal algorithm.

The Parareal algorithm alternates between an integration step (parallel) done with a fine solver, and an update step (serial) done with a coarse solver.

We can interpret the formula prescribed in the Parareal algorithm as follows: the initial conditions we pick are updated depending on the result of the integration from the initial conditions at the previous time chunk, plus a correction that comes from the fact that the previous initial conditions were inaccurate. The coarse solver takes care of this correction. Notice that the fine solver can be used in parallel, since all time chunks are independent, which means that we can afford to invest effort in performing the integration accurately. On the other hand, the coarse solver needs to be used in serial, to propagate the correction forward in time, so we are forced to use a quick and inaccurate integrator for the algorithm to be efficient.

The coarse solver $\mathcal{G}$ is crucial for the correct functioning of the algorithm: it should be at the same time accurate enough to resemble the action of $\mathcal{F}$, but cheap enough to be computed serially. Finding the optimal $\mathcal{G}$ is still an open challenge.

3. The Interparareal algorithm: an alternative to the coarse solver

Contrarily to the Parareal algorithm, Interparareal relies on a direct approximation of the action of the fine solver in order to conduct the update.

We now develop an alternative algorithm that removes altogether the necessity to use a coarse solver. Instead of relying on the action of $\mathcal{G}$, we can infer the effect that a variation on the initial conditions has on the final conditions directly from the evaluations of $\mathcal{F}$ conducted throughout the algorithm. If the differential equation is autonomous, in fact, each time we perform an integration on any of the time chunks, we are sampling the same function $\mathcal{F}(\lambda)$, which gives us the final conditions at the end of a subinterval starting from the initial conditions. Given these samples, we can then build an *interpolant*, Π (hence the name *Inter*-parareal), as an approximation of $\mathcal{F}$. Finally, the correction can be done using the derivative of the interpolant with respect to the initial conditions, $\mathcal{J}_\Pi$, which describes precisely the impact that varying λ has on the final conditions. Our update formula is:

$$\lambda_i^{k+1} = \mathcal{F}\left(\lambda_{i-1}^k\right) + \mathcal{J}_\Pi\left(\lambda_{i-1}^{k+1} - \lambda_{i-1}^k\right).$$

Comments

- The Interparareal algorithm has some strong limitations in its applicability: the task of building an accurate interpolant become increasingly challenging as we add more variables to the system of equations considered. This means that it can be efficiently used only in the case of systems of ODEs that are reasonably small in size.

- Extending the algorithm to PDEs poses an additional difficulty. PDEs can often be solved numerically by re-casting them as a very large system of ODEs, hence falling in the problematic case described in the first comment. However, while it is true that the number of equations is large, they are all similar in nature. Under some simplifying assumptions, we can then exploit the particular structure of the arising system to overcome this issue. In particular, we need to request that the way a solution evolves is independent on the point in space where we are evaluating it, and it is only affected by the values around it.

- The Interparareal algorithm has an additional, desirable, feature which makes it better than the Parareal algorithm. Our theoretical results show us that, if the equations are *linear*, then already after the first iteration our guess on the initial conditions will be exact. This is a very simple category of equations, but linear PDEs can already describe many relevant physical phenomena.

4. Comparison of the algorithms

To test the efficiency of the Interparareal algorithm against the original Parareal algorithm, we compare how the error we make in guessing the initial conditions varies as we proceed with the iterations for both algorithms. The error can be computed as the difference between our solution and the one obtained applying the fine solver on the whole domain. If the error approaches zero, the algorithm is said to achieve *convergence*: the faster this happens, the better the algorithm is performing.

To conduct our test, we consider the application of the two algorithms both to systems of ODEs and to PDEs, employing various kinds of numerical solvers which present different levels of accuracy.

Application to ODEs

When applied to linear equations, the Interparareal algorithm needs only one iteration to converge. In the majority of the other cases tested, the performance of our modified algorithm is better than that of the original Parareal.

Regarding ODEs, we are mostly interested in convergence results for systems that have a chaotic behaviour, that is, systems whose evolution can vary wildly and is strongly dependent on the initial conditions. This is both because they represent a first rough benchmark for simulations describing plasma behaviour, and because our algorithms become more delicate whenever correctly guessing initial conditions is crucial. The Lorenz system is considered the model problem for chaotic evolution: a comparison of the two algorithms when solving the Lorenz system is shown in Figure 2 (left). Furthermore, linear systems are also of particular relevance, because in that case, the properties of the Interparareal algorithm allow us to achieve convergence after a single iteration. The corresponding results are shown in Figure 2 (right). We can see for both cases that the error in the Interparareal algorithm reduces much more quickly than in the Parareal algorithm. Similar results are obtained when applying the algorithms to other systems of ODEs with different numbers of equations, which present a variety of behaviours in the nature of their evolution. In particular, in our test we consider (i) systems whose state tends towards equilibrium or away from it, (ii) ODEs that show an oscillatory evolution, and (iii) combinations of these. Increasing the accuracy of the numerical solvers used tends to decrease the gap in performance between the two algorithms, but in the majority of

cases considered Interpareareal outperforms Parareal, usually requiring little more than half the number of iterations in order to achieve the same level of precision.

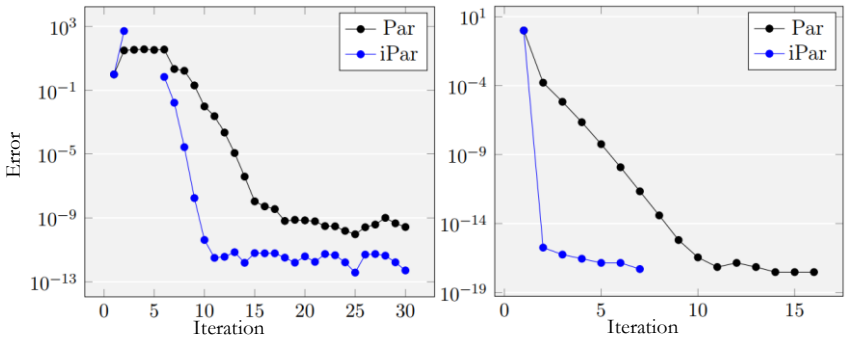


Figure 2: Error evolution for Parareal (black) and Interpareareal (blue) when applied to Lorenz's chaotic system (left) and a linear system of ODEs (right).

Application to PDEs

We now compare the performance of the Interpareareal and Parareal algorithms on some simple PDEs. The Parareal algorithm is known to have particular difficulties in achieving convergence when applied to a category of PDEs named *transport* equations. Among other things, transport equations are used to describe how substances propagate within a medium. Conversely, the Parareal algorithm has proven to be reasonably stable when applied to *diffusion* equations, which describe for example how the temperature distribution varies inside a room. A comparison of the error for both these cases using the two algorithms is shown in Figure 3. The transport equation chosen is linear, and we see in Figure 3 (left) that convergence is achieved immediately when using the Interpareareal algorithm. The diffusion equation has an additional non-linear component, which represents a reaction term: in our temperature example, it could represent for instance production or absorption of heat. From Figure 3 (right), we again see how quickly the Interpareareal algorithm can reach convergence, even when used for the solution of some types of PDEs.

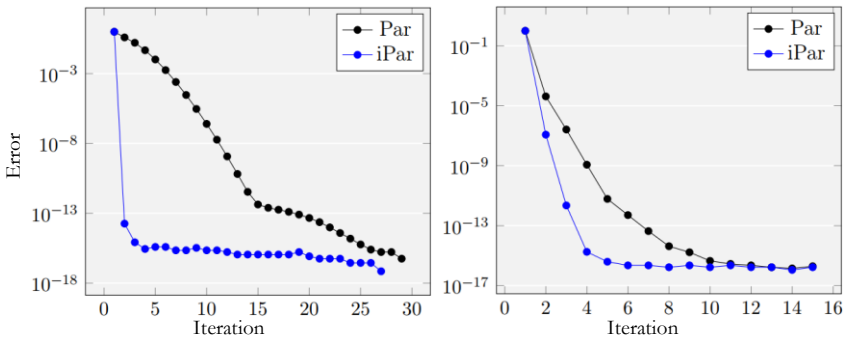


Figure 3: Error evolution for Parareal (black) and Interpareareal (blue) when applied to a linear transport PDE (left) and a simple non-linear diffusion-reaction PDE (right).

However, we point out that the range of PDEs our algorithm can be applied to is still very limited, and additional research is required to successfully and efficiently employ it for the solution of the kind of equations used in plasma physics.

5. Discussion, Conclusions & Recommendations

We have considered ways to improve the performance of the Parareal algorithm for time parallelisation for use in solving problems from plasma physics. We have developed a new version of the Parareal algorithm, which we have named the Interparareal algorithm. The Interparareal algorithm uses an alternative to the coarse solver found in the Parareal algorithm in order to conduct the update of the initial conditions. This requires computing the derivative of the function that describes the evolution in time of the solution with respect to the initial conditions. The Interparareal algorithm generates a direct approximation of this derivative using an interpolating function that employs the evaluations of the fine solver as sample points. This potentially allows for faster convergence at comparable cost.

We investigated numerous test cases covering a range of systems of ODEs presenting different behaviours in their evolutions, as well as some simple PDEs, solved using numerical schemes of different degrees of accuracy. The results from the comparison between the two algorithms were very promising, showing how the Interparareal algorithm needs a smaller number of iterations than Parareal to achieve convergence. This is definitely seen in applications to linear equations, but even when extended to some simple non-linear cases, its performance has proven to be not worse (and very often much better) than that of the original Parareal algorithm. For this reason it represents an alternative worth investigating and expanding further.

In order to apply the proposed Interparareal algorithm to the systems of PDEs that are of interest for plasma simulations, more research needs to be put into trying to overcome the requirements for linearity and space-independence. In particular, effort should be focused on trying to reduce the number of discretised equations involved in the simulation, which will simplify the task of building an accurate interpolant.

6. Potential Impact

Solving differential equations is of crucial importance in a variety of applications. In particular, researchers at CCFE use complex systems to describe the behaviour of hot plasma confined within a fusion reactor. However, the number of computations necessary to conduct these simulations is often such that a vast amount of time is required in order to find the solution. Hence, reducing this time even by a fraction is of extreme relevance. The Interparareal algorithm can allow for a faster resolution of systems of ODEs and PDEs, providing in many cases an option which has shown to be much better than the original Parareal algorithm for employing time parallelisation.

Dr. Debasmita Samaddar, Computational Plasma Physicist at CCFE, commented on the outcome of the project: "*Working with Federico has been an absolute pleasure. He started the mini project with the goal of identifying ways of choosing a 'good' coarse predictor for Parareal implementations in problems in fusion plasma. The project was challenging for a variety of reasons. Temporal parallelisation is often considered counter intuitive – so grasping the details of the Parareal algorithm was not expected to be easy. Federico studied a sufficient amount of literature to quickly find his way in this novel area of research. Federico then introduced a whole new approach to dealing with the Parareal correction. The rest of the project was focused on exploring this approach of 'Interparareal' to a variety of problems. The results are very promising although there are a large number of open questions. It will be extremely beneficial to the mathematical as well as plasma physics communities if Federico further explores the new approach for more complex non-linear problems for his PhD. If implemented efficiently, his results can usher in a breakthrough not just in fusion plasma simulations, but in computational physics in general.*"