Introducing time parallelisation within data assimilation using parareal method

Rishabh Bhatt, Laurent Debreu, Arthur Vidard

Introducing time parallelisation within data assimilation using parareal method

Rishabh Bhatt, Laurent Debreu, and Arthur Vidard

INRIA, LJK, 38400 Saint-Martin-d'Hères, France

Abstract

Forecasts made by 4-D Var involves forward integration of model before proceeding for the minimisation process. While one reaches saturation in space parallelisation we try to obtain some speedup by integrating our model using the Parareal method. Our setting ensures that the minimum is obtained by solving a linear symmetric system. We use a modified version of the inexact conjugate gradient method where the matrix-vector multiplication is supplied by the parareal. This helps us to determine a specific stopping criterion for the parareal. The results are demonstrated by considering a 1-D linear shallow water model. Our method produces a speedup factor of 3 using fewer parareal iterations as compared to the same results obtained by the exact conjugate gradient method.

Keywords— Optimisation, Parareal, Variational Data Assimilation, High Performance Computing, Numerical Analysis, Krylov Subspace Methods

1 Introduction

Modern computation emphasises on reducing the clock time. With the advancement of massively fast computers millions of processors can be utilised at once. But when it comes to solving very large time-dependent problems, space parallelisation alone becomes insufficient. After one point it gets saturated even before effectively using up all the processors.

For some problems such as weather prediction even a small gain in computational speed matters. A simple example can be a climate simulation problem with 10^6 unknowns given $10,000$ cores. Using 10×10 unknowns for the horizontal domain would end up using 10^4 cores. Assigning lesser number of unknowns to each processor would make the computational time so less that the communication time would dominate the overall computing time. This means that after using certain number of cores, there will be a performance loss if more cores are utilised. This would make no sense in weather simulation if it gives us the result on the day for which we actually wanted to predict the weather.

One of the options is to look for the much needed parallelisation in the time domain. But time domain possesses an inherent property of causality which makes it hard to parallelise. It simply means that the solution at a given time can only be computed if we have some prior information about the solution at previous times. This dependency makes all the algorithms serial or sequential and this is where the concept of the Parallel-in-Time algorithms comes to circumvent the causality principle and minimise its influence. It is not surprising that the first ideas of time parallelisation were already introduced in the early 60s by Nivergelt [1] who proposed a multiple shooting method where the solutions at the subintervals are joined together by some interpolation.

Much work has been done since then and the state of the art review by Gander [2] provides a comprehensive knowledge of most of the important parallel-in-time algorithms. Our focus is on using these algorithms in the context of a Data Assimilation problem. The aim of the paper is to present whether we can exploit some of the available computational power to produce any possible speed ups. Operational 4-D VAR algorithms involves two serial operations of integrating the model trajectories and the minimisation process to provide the optimal initial condition for the model. We are going to make an attempt to replace the serial integration with time parallelisation to couple the two iterative processes

together. In the end we present our results and show an approximate measure of the speed up which can be obtained when the model trajectories are computed in parallel in time.

The paper is structured as follows. In section 2 we begin with Variational Data Assimilation applied on a linear model and define the setting on which we are going to work on. We will talk about how the time parallelisation could be introduced in its framework. Next we discuss about a specific kind of parallel-time-method called the Parareal algorithm which has been central to most of the recent developments in section 3. After the set-up we will talk about the optimisation techniques for minimising the 4-D VAR cost function for obtaining the accurate initial condition in Section 4. To have good efficiency we will use the inexact Krylov methods and later provide a modified version of an existing method to make it work with the parareal algorithm. Section 5 is left for the numerical experiments and results. Section 6 concludes the paper with some remarks and talking points for further work.

Throughout this article, we will adopt the following notation

- Upper case caligraphic fonts for nonlinear functions/operators: $\mathcal{M}, \mathcal{F}$
- Upper case italics for linear functions/operators: M, H
- Bold upper case for matrices: $\mathbf{A}, \mathbf{B}$
- Bold lower case for vectors: $\mathbf{x}, \mathbf{y}$

2 Data Assimilation

Let us describe our dynamical model depending on a state variable vector $\mathbf{x}$ by

$$\begin{aligned} \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{x}_i &= \mathcal{M}_{t_{i-1}\to t_i}(\mathbf{x}_{i-1}) \quad i = 1, ..., N \end{aligned} \tag{1}$$

where $\mathcal{M}$ is the discrete non-linear model operator, $\mathbf{x}_0$ is the initial condition and N is the number of time windows.

The problem of 4-D Variational Data Assimilation or 4-D Var amounts to correcting the model trajectory by striking a right balance between the a priori background information $\mathbf{x}_0^b \in \mathbb{R}^n$ and a set of observations $\mathbf{y}_i \in \mathbb{R}^m$ at time t_i [3]. This can be expressed as a least-squares fit problem whose cost function is

$$\mathcal{J}(\mathbf{x}_0) = \frac{1}{2}\,(\mathbf{x}_0 - \mathbf{x}_0^b)^T \mathbf{B}^{-1} (\mathbf{x}_0 - \mathbf{x}_0^b) \;+\; \frac{1}{2}\sum_{i=0}^{N} (\mathbf{y}_i^o - \mathcal{H}_i(\mathbf{x}_i))^T \mathbf{R}_i^{-1} (\mathbf{y}_i^o - \mathcal{H}_i(\mathbf{x}_i))$$

Here $\mathbf{B}$ and $\mathbf{R}$ are the background and observation covariance matrices respectively. $\mathcal{H} : \mathbb{R}^m \to \mathbb{R}^n$ is the observation operator which maps the state vectors to the observation space. Since the entire model trajectory can be described by the initial condition $\mathbf{x}_0$, the cost function can be minimised only with respect to $\mathbf{x}_0$. Thus the minimisation problem can be reformulated as

$$\begin{aligned} \min_{\mathbf{x}_0 \in \mathbb{R}^n} \mathcal{J}(\mathbf{x}_0) &= \frac{1}{2}\,\|\mathbf{x}_0 - \mathbf{x}_0^b\|_b^2 \;+\; \frac{1}{2}\sum_{i=0}^{N} \|\mathbf{y}_i^o - \mathcal{H}_i(\mathcal{M}_{t_0\to t_i}(\mathbf{x}_0))\|_o^2 \\ &= \underbrace{\mathcal{J}_b(\mathbf{x}_0)}_{\text{background}} \;+\; \underbrace{\mathcal{J}_o(\mathbf{x}_0)}_{\text{observation}} \end{aligned} \tag{2}$$

This is known as the strongly constrained 4-D VAR since the numerical model is assumed to be perfect. The norms $\|\cdot\|_b$ and $\|\cdot\|_o$ are the energy norms of the matrices $\mathbf{B}^{-1}$ and $\mathbf{R}^{-1}$ respectively. For notational simplicity a simplified observation operator $\mathcal{G}_i$ can be defined as the composition of the operators $\mathcal{M}_{t_0\to t_i}$ and $\mathcal{H}$, i.e. $\mathcal{G}_i \overset{\text{def}}{:=} \mathcal{M}_{t_0\to t_i} \circ \mathcal{H}_i$. This way the cost function in (2) can be re-written as

$$\mathcal{J}(\mathbf{x}_0) = \frac{1}{2}\,\|\mathbf{x}_0 - \mathbf{x}_0^b\|_b^2 \;+\; \frac{1}{2}\sum_{i=0}^{N} \|\mathbf{y}_i^o - \mathcal{G}_i(\mathbf{x}_0))\|_o^2 \tag{3}$$

Looking at (2) it is evident that the cost function $\mathcal{J}$ is non-convex because of the nonlinearites in $\mathcal{M}$ and $\mathcal{H}$. As a result in the absence of a unique minimum, the minimisation becomes very difficult. Instead of solving the large-scale minimisation process at one go, the problem is solved incrementally by taking successive approximations [4]. This is done by taking the following tangent linear approximations around the iterates with sufficiently small increments $\delta\mathbf{x}^{(k)}$

$$\mathcal{G}(\mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}) = \mathcal{G}(\mathbf{x}^{(k)}) + G\delta\mathbf{x}^{(k)}$$

where $G = \dfrac{\partial \mathcal{G}}{\partial \mathbf{x}}\Big|_{\mathbf{x}^{(k)}}$ is the linearised simplified observation operator.

The cost function obtained is now quadratic and much easier to solve. This formulation is known as the Incremental 4D-Var. We present the procedure as provided in [5]

i) For $k = 0$ we usually set $\mathbf{x}_0^{(0)} = \mathbf{x}_0^b$.

ii) At iteration k for a given increment defined as $\delta\mathbf{x}_0^{(k)} = \mathbf{x}_0^{(k+1)} - \mathbf{x}_0^{(k)}$ the cost function (2) can be written as

$$\begin{aligned} J(\delta\mathbf{x}_0^{(k)}) &= \frac{1}{2}\left\|\delta\mathbf{x}_0^{(k)} - (\mathbf{x}_0^b - \mathbf{x}_0^{(k)})\right\|_b^2 + \frac{1}{2}\sum_{i=0}^{N}\left\|\mathbf{y}_i - \mathcal{G}_i(\mathbf{x}_i^{(k)} + \delta\mathbf{x}_i^{(k)})\right\|_o^2 \\ &= \frac{1}{2}\left\|\delta\mathbf{x}_0^{(k)} - (\mathbf{x}_0^b - \mathbf{x}_0^{(k)})\right\|_b^2 + \frac{1}{2}\sum_{i=0}^{N}\left\|\mathbf{d}_i - G_i\delta\mathbf{x}_i^{(k)}\right\|_o^2 \end{aligned} \tag{4}$$

where we define the innovation vector or the departure as $\mathbf{d}_i^{(k)} = \mathbf{y}_i - \mathcal{G}_i(\mathbf{x}_i^{(k)})$

iii) The constraint is given by

$$\begin{aligned} \delta\mathbf{x}_i^{(k)} &= M_{t_0 \to t_i}\mathbf{x}_0^{(k)} \\ &= M_i M_{i-1} \ldots M_2 M_1 \mathbf{x}_0^{(k)} \end{aligned} \tag{5}$$

where $M = \dfrac{\partial \mathcal{M}}{\partial \mathbf{x}}\Big|_{\mathbf{x}^{(k)}}$ is the linearised model operator.

iv) Note that the control variable now is the correction $\delta\mathbf{x}$ to the model parameter $\mathbf{x}$. The new cost function now is quadratic with respect to $\delta\mathbf{x}_0^{(k)}$. The guess is updated by

$$\mathbf{x}_0^{(k+1)} = \mathbf{x}_0^{(k)} + \delta\mathbf{x}_0^{(k)}$$

Note: Incremental 4-D Var minimises the cost function on a lower resolution of the model. It is more efficient for solving quadratic problems but this does not count out the possibility that it can get stuck in a local minimum.

2.1 Introducing time parallelisation

Each minimisation iteration of a 4-D Var cycle begins by a forward integration of the model (1) to obtain and store the trajectory. A potential speed up can be achieved if these trajectories are instead calculated by integrating the model over the time windows parallel in time. To see how this could be done, let us suppose the length of one cycle to be T hours (say) which is made up of N time windows.

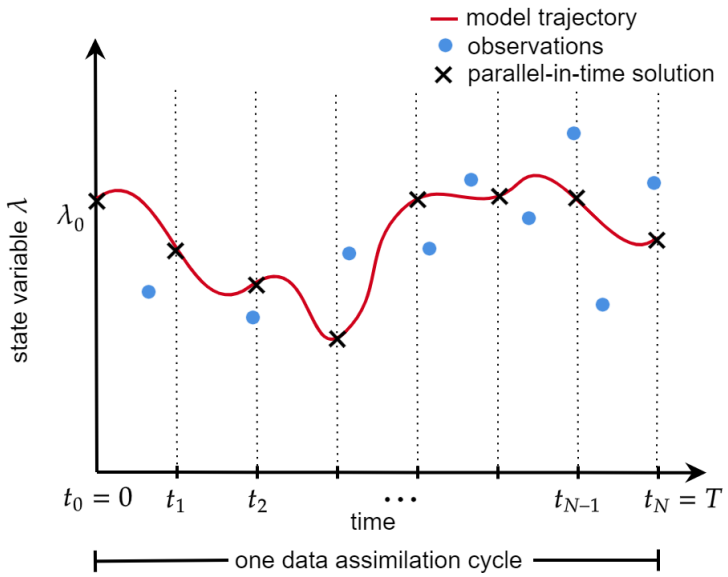


Figure 1: Using time parallelisation for running the forward model

We consider the linear model

$$\begin{aligned} \lambda(t_0) &= \lambda_0 \\ \lambda_{i+1} &= M\lambda_i \quad i = 1, \dots N \end{aligned} \tag{6}$$

The model operator M is the same as the one used in (5). The only difference is that we have now replaced the state variable $\delta\mathbf{x}$ with λ for simplicity and to maintain consistency.

Let us now set up our data assimilation problem. We assume that we have no background information about the model, that is there is no background term in the cost function. Also let there be only one true observation $\mathbf{y}$ at the end of the integration time $t_N = T$. This means that the covariance matrix $\mathbf{R}_i$ is equal to the identity matrix while the observation operator $\mathcal{H}_i$ is an identity map for each time t_i. Thus our cost function can be written as

$$J(\lambda_0) = \frac{1}{2}\|M^N\lambda_0 - \mathbf{y}\|_2^2 \tag{7}$$

with gradient,

$$\nabla J(\lambda_0) = (M^N)^T(M^N\lambda_0 - \mathbf{y})$$

where $(M^N)^T$ is the adjoint matrix.

As stated before, if we now carry out the computation of the forward model (involved in the misfit between the model trajectory and the observation) by a parallel-in-time based operator $\mathcal{T}$ (can be iterative or direct), we have

$$\begin{aligned} J(\lambda_0) &= \frac{1}{2}\|\mathcal{T}\lambda_0 - \mathbf{y}\|_2^2 \\ \nabla J(\lambda_0) &= (M^N)^T(\mathcal{T}\lambda_0 - \mathbf{y}) \end{aligned} \tag{8}$$

Minimising this cost function requires setting the gradient $\nabla J(\lambda_0)$ to 0. This is equivalent to solving the system

$$\mathbf{A}\lambda = \mathbf{b} \qquad \text{with} \quad \mathbf{A} = (M^N)^T\mathcal{T}\ ,\ \mathbf{b} = (M^N)^T\mathbf{y}$$

At the end of each minimsation iteration we will have an initial condition with a lower cost function value. This in turn will be used by $\mathcal{T}$ to provide new trajectories for the next iteration. The process will

end depending on the stopping criteria of the minimisation.

When the condition number $\kappa(\mathbf{A})$ is large, a regularisation term α can be used in the cost function which depends on the spatial derivative of the initial state λ_0^b. Thus,

$$J(\lambda_0) = \frac{1}{2}\|\mathcal{T}\lambda_0 - \mathbf{y}\|_2^2 + \frac{\alpha}{2}\left\|\frac{\mathrm{d}\lambda_0}{\mathrm{d}x}\right\|_2^2$$

The spatial derivative can be written as

$$\frac{\mathrm{d}\lambda_0}{\mathrm{d}x} = \frac{\lambda_{0,i+1} - \lambda_{0,i}}{\Delta x} = \frac{1}{\Delta x}\mathbf{P}\lambda_0$$

where

$$\mathbf{P} = \begin{pmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \\ & & & -1 \end{pmatrix}$$

Therefore,

$$\left\|\frac{\mathrm{d}\lambda_0}{\mathrm{d}x}\right\|_2^2 = \frac{1}{\Delta x^2}\left\langle \mathbf{P}\lambda_0, \mathbf{P}\lambda_0 \right\rangle$$

The simplified cost function can be written as

$$J(\lambda_0) = \frac{1}{2}\|\mathcal{T}\lambda_0 - \mathbf{y}\|_2^2 + \frac{\alpha}{2\Delta x^2}\|\mathbf{P}\lambda_0\|_2^2$$

Written like this, the gradient should be

$$\nabla J(\lambda_0) = (M^N)^T(\mathcal{T}\lambda_0 - \mathbf{y}) + \alpha\mathbf{Q}\lambda_0$$

where

$$\mathbf{Q} = \frac{1}{\Delta x^2}\mathbf{P}^T\mathbf{P} = \frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

Again, this can be put up as a linear system $\mathbf{A}\lambda = \mathbf{b}$ with

$$\mathbf{A} = (M^N)^T\mathcal{T} + \alpha\mathbf{Q} \quad , \qquad \mathbf{b} = (M^N)^T\mathbf{y}$$

Remark 2.1. Using an iterative minimisation algorithm for the above linear system is equivalent to solving the incremental 4-D VAR problem.

Remark 2.2. The true observation $\mathbf{y}$ at t_N is the reference parallel-in-time solution at the end of integration time. By taking $\mathbf{y}$ from the exact solution of the model, the minimisation complexity can be further increased. That is,

$$\begin{aligned} \mathbf{y} &= \lambda_0 \mathrm{e}^{\mathbf{B}t_N} \\ \mathbf{b} &= (M^N)^T(\lambda_0\, \mathrm{e}^{\mathbf{B}t_N}) \end{aligned} \tag{9}$$

where $\mathbf{B}$ is the coefficient matrix of the model M

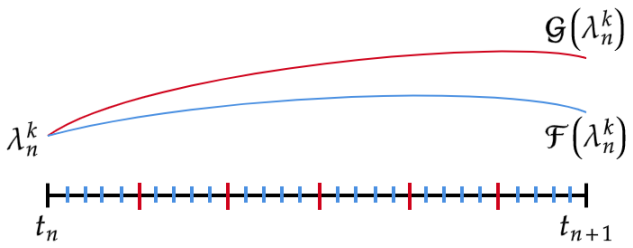
Depending on the choice of the operator $\mathcal{T}$ the two iterative processes of forward model integration and minimisation can be coupled in different ways. We are going to focus on the case where $\mathcal{T}$ is replaced by the Parareal algorithm.

3 Parareal Method

Parareal was introduced in [6] and later reformulated [7] as a kind of parallel-in-time method where the solution at a given time step is computed independently of the solution at previous time steps. For a general system of ODE

$$\begin{aligned} \frac{\mathrm{d}\lambda}{\mathrm{dt}} &= \mathbf{B}\lambda \\ \lambda(t=0) &= \lambda_0, \quad t \in [0, T] \end{aligned} \tag{10}$$

the strategy relies on partitioning the time domain $[0, T]$ into N time windows or sub-intervals $[t_n, t_{n+1}]$ of length $\Delta T = T/N$. Within each such sub-interval, a two-level grid system is defined using a coarse propagator $\mathcal{G}$ and a fine propagator $\mathcal{F}$. By construction, $\mathcal{G}$ is a very cheap solver with a large time step Δt making it less accurate. On the other hand, $\mathcal{F}$ has a small time step δt making it a highly accurate and computationally expensive solver.



The parareal algorithm proceeds iteratively by computing the solution $\lambda_{n+1}^k \approx \lambda(t_{n+1})$ at the sub-intervals for given λ_n^k at time t_n and iteration k. An initial configuration is obtained by a serial run of the coarse propagator $\mathcal{G}$ over the whole time domain. Thus,

$$\begin{aligned} \lambda_0^0 &= \lambda_0 \\ \lambda_{n+1}^0 &= \mathcal{G}(t_{n+1}, t_n, \lambda_n^0) \quad n = 0, 1, \ldots, N-1 \end{aligned}$$

With the solution now available at all the grid points, the fine propagator $\mathcal{F}$ is run to provide more accurate solution. As $\mathcal{F}$ is expensive, all the computations performed by $\mathcal{F}$ are done in parallel by assigning a processor to each sub-interval. Finally the iterates are updated by a correction procedure

$$\begin{aligned} \lambda_0^{k+1} &= \lambda_0 \\ \lambda_{n+1}^{k+1} &= \underbrace{\mathcal{G}(t_{n+1}, t_n, \lambda_n^{k+1})}_{\text{Prediction}} + \underbrace{\mathcal{F}(t_{n+1}, t_n, \lambda_n^k) - \mathcal{G}(t_{n+1}, t_n, \lambda_n^k)}_{\text{Correction}} \end{aligned} \tag{11}$$

Thus parareal can be seen as a predictor-corrector algorithm where at each iteration the predicted term $\mathcal{G}(t_{n+1}, t_n, \lambda_n^{k+1})$ is corrected by the solutions from $\mathcal{F}$ and $\mathcal{G}$ at the previous iteration. The correction step (11) is sequential as one has to do a serial coarse integration first to get the prediction term and only then the pre-computed correction terms can be added.

Theoretically, the maximum accuracy that the parareal algorithm can achieve is that of the fine solver. Taking the limit $k \to \infty$, the solutions from the coarse solver in (12) cancel out each other and what is left is just the solution from $\mathcal{F}$. Another thing to note is that after each parareal iteration the trajectory is corrected to its exact solution for exactly one time window. After N iterations, the parareal solution will exactly be the same as the solution obtained from the serial integration by the fine solver. For the algorithm to be practical it is important that $k \ll N$, otherwise there will be no speedup.

If the model is linear, we can re-write the correction step (11) as

$$\begin{aligned} \lambda_{n+1}^{k+1} &= \mathbf{F}\lambda_n^k + \mathbf{G}\lambda_n^{k+1} - \mathbf{G}\lambda_n^k \\ &= \mathbf{G}\lambda_n^{k+1} + (\mathbf{F} - \mathbf{G})\lambda_n^k \end{aligned} \tag{12}$$

Algorithm 1: Parareal Algorithm

```
Initialisation
for n ← 0 to N do
    λ^0_{n+1} = G(t_{n+1}, t_n, λ^0_n)
Iterations
k = 0
repeat
    Parallel prediction step
    for n ← 0 to N do
        λ̃^k_{n+1} = F(t_{n+1}, t_n, λ^k_n)
    Serial correction step
    for n ← 0 to N do
        λ^{k+1}_{n+1} = G(t_{n+1}, t_n, λ^{k+1}_n) + λ̃^k_{n+1} − G(t_{n+1}, t_n, λ^k_n)
    k = k + 1
until k = k_max
```

where $\mathbf{F}$ and $\mathbf{G}$ are the matrix representation of the fine and coarse propagator.

Parareal algorithm can also be seen as a type of a multiple shooting method [8] where the Jacobian is approximated by the differences between two successive approximations of the coarse discretisation. A thorough convergence study for the algorithm has been done by Gander and Vandewalle[8]. For a linear system with bounded time domain one gets superlinear convergence but it becomes linear when the time domain is unbounded. For a general nonlinear system the convergence remains superlinear[9].

3.1 Error analysis

We now focus on obtaining a general expression of the error for a linear system. We write the correction step (12)

$$\lambda_n^k = \mathbf{G}\lambda_{n-1}^k + (\mathbf{F} - \mathbf{G})\lambda_{n-1}^{k-1}$$

Let us assume that the parareal reference solution is the solution obtained by integrating the model using the fine solver. Let $\lambda_{n,e}$ be the reference solution at the start of the time window n (i.e. $\lambda_{n,e} = \mathbf{F}^n\lambda_0$). Then the error $\mathbf{e}_n^k = \lambda_{n,e} - \lambda_n^k$ satisfies the evolution equation

$$\mathbf{e}_n^k = \mathbf{G}\mathbf{e}_{n-1}^k + (\mathbf{F} - \mathbf{G})\mathbf{e}_{n-1}^{k-1}$$

and thus

$$\mathbf{e}_n^k = (\mathbf{F} - \mathbf{G})\sum_{p=k}^{n-1} \mathbf{G}^{n-p-1}\mathbf{e}_p^{k-1}$$

where we have $\mathbf{e}_0^0 = 0$. If the initial states are obtained by a coarse grid integration (i.e. $\lambda_n^0 = \mathbf{G}^n\lambda_0$), we have $\mathbf{e}_n^0 = (\mathbf{F} - \mathbf{G})\lambda_n^0$ and we get a general expression for $\mathbf{e}_n^k$:

$$\mathbf{e}_n^k = \lambda_0 \sum_{p=k+1}^{n} C_p^n(\mathbf{F} - \mathbf{G})^p\mathbf{G}^{n-p}, \quad C_p^n = \frac{n!}{p!(n-p)!}$$

We can now define an approximate Parareal based integrator over $[0, T]$ by

$$P(k)\lambda_0 = \lambda_N^k \tag{13}$$

and

$$\mathbf{F}^N - P(k) = \sum_{p=k+1}^{N} C_p^N(\mathbf{F} - \mathbf{G})^p\mathbf{G}^{N-p} \tag{14}$$

3.2 Coupling

We replace the parallel-in-time operator $\mathcal{T}$ in (8) by the parareal algorithm. In this case the model trajectories are obtained by the reference solution which is nothing but the solution from the fine propagator $\mathbf{F}$. The cost function (7) becomes

$$J(\lambda_0) = \frac{1}{2}\|\mathbf{F}^N\lambda_0 - \mathbf{y}\|_2^2 \tag{15}$$

Its gradient is given by

$$\nabla J(\lambda_0) = (\mathbf{F}^N)^T(\mathbf{F}^N\lambda_0 - \mathbf{y})$$

This way, the linear system Now again instead of using the reference solution for the forward integration, if we use the Parareal based integrator $P(k)$ we have

$$J(\lambda_0) = \frac{1}{2}\|P(k)\lambda_0 - \mathbf{y}\|_2^2 \tag{16}$$

In this respect, the gradient can be written as

$$\nabla J(\lambda_0) = (\mathbf{F}^N)^T(P(k)\lambda_0 - \mathbf{y})$$

The corresponding linear system $\mathbf{A}\lambda = \mathbf{b}$ to solve has matrix $\mathbf{A} = (\mathbf{F}^N)^T P(k)$ and the vector $\mathbf{b} = (\mathbf{F}^N)^T\mathbf{y}$

3.3 Speedup

We assume that the we have N number of processors available so that each processor can be assigned to one time window. We also assume that the communication time between any two processors is so small that it can be ignored.

The theoretical speed up denoted by S is defined as the ratio of the time taken to perform a task sequentially to the time to do the same task in parallel.

$$S = \frac{\text{Serial computation time}}{\text{Parallel computation time}} \tag{17}$$

To derive the speedup for the parareal algorithm we follow along the lines of [10]. We begin by defining some quantities. Let N_f and N_g denote the number of fine and coarse time steps per time window respectively. Let τ_f be the time taken by the processor to perform a single step of the numerical scheme for the fine propagator $\mathbf{F}$ and similarly τ_g for the coarse propagator $\mathbf{G}$. Thus for a single time window, $N_f\tau_f$ is the total cost of $\mathbf{F}$ and $N_g\tau_g$ for $\mathbf{G}$. To simplify notations, let $\gamma_f = N_f\tau_f$ and $\gamma_g = N_g\tau_g$.

We take the solution from the fine propagator as our reference solution. So the serial integration time is nothing but the time taken by the fine solver to solve the problem sequentially. Since the initial configuration for the parareal is fed through a serial coarse integration, it is given by $N\gamma_g$. Now one iteration of parareal involves a prediction by $\mathbf{G}$ which is again sequential and the correction step in parallel. The total cost per parareal iteration is given as $N\gamma_g + \gamma_f$. Therefore for k parareal iterations,

$$\begin{aligned} S &= \frac{N\gamma_f}{N\gamma_g + k(N\gamma_g + \gamma_f)} \\ &= \frac{\gamma_f}{\gamma_g + k(\gamma_g + \frac{\gamma_f}{N})} \end{aligned}$$

Let us define $\beta = \dfrac{\gamma_g}{\gamma_f}$, we have

$$S = \frac{1}{\beta + k\left(\beta + \dfrac{1}{N}\right)}$$

As stated in [11] we get different bounds for the speedup and they compete with each other.

$$S \leq \min\left(\frac{N}{k}, \frac{\text{fine time}}{\text{coarse time}}\right)$$

The coarse propagator should be really cheap and fast in order to get a good speed up but at the same time there will be more parareal iterations which will reduce the speedup according to the first bound. Thus the optimal speedup must balance the two bounds in the best possible way.

4 Minimisation

For solving large linear systems, choosing a suitable Krylov subspace method is a logical option [12]. Krylov methods are iterative in nature with the matrix-vector multiplication as their most dominant operation. In some situations, there might be the case that the matrix-vector product is not exact because the matrix comes from a discretisation scheme of a differential equation. Another reason could be the lack of access to the matrix, and it is only available as a product by a vector.

However, under certain conditions, this undesirable inexactness could be used to better effects in terms of some efficiency gains. In literature, such methods are known as the inexact Krylov subspace methods. They are based on a counterintuitive principle that the error in the matrix-vector multiplication is permitted to grow as we move ahead with the iterations [13, 14]. By doing so, the same level of accuracy can be maintained as expected by using the usual Krylov subspace methods.

Here we are dealing with a quadratic cost function of the Data Assimilation problem which leads to solving a symmetric system $\mathbf{A}\lambda = \mathbf{b}$. In this case $\mathbf{A} = (\mathbf{F}^N)^T P(k)$ and $\mathbf{b} = (\mathbf{F}^N)^T \mathbf{y}$ since the matrix-vector multiplication is provided by the parareal method which is itself iterative and bound to some stopping tolerance. We are going to use the conjugate gradient method which is effective for linear symmetric systems.

4.1 Inexact conjugate gradient

We are given the following convex quadratic minimisation problem

$$\min_{\lambda \in \mathbb{R}^n} \; q(\lambda) = \frac{1}{2}\lambda^T \mathbf{A}\lambda - \mathbf{b}^T\lambda$$

with symmetric positive definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$

This problem can also be seen as solving the symmetric positive-definite linear system $\mathbf{A}\lambda = \mathbf{b}$ by a suitable krylov subspace method. We would like to look for any efficiency gains in the sense that relaxing the accuracy of the matrix-vector products (thus allowing inexactness) would still give the same levels of efficiency.

The inexact Conjugate Gradient (inexact CG) method proposed by Gratton et al. [15] monitors the decrease in the quadratic when the matrix-vector multiplications are inexact. The reason behind focusing on the quadratic is the direct relationship of its change with the energy norm of the residual which leads to a better stopping criterion. If $\mathbf{r}(\lambda) = \mathbf{b} - \mathbf{A}\lambda$ is defined to be the residual of the system then this relationship can be described as [15]

$$\frac{1}{2}\|\mathbf{r}(\lambda)\|^2_{\mathbf{A}^{-1}} = q(\lambda) - q(\lambda_*) \tag{18}$$

where $\lambda_* = \mathbf{A}^{-1}\mathbf{b}$ is taken as the exact solution of the system.

We will adopt this idea and introduce the inexactness with the help of parareal method. Let us represent the source of inexactness in the matrix $\mathbf{A}$ by the perturbation matrix $\mathbf{E}$. In our context the error manifests from the parareal method in equation (14). By construction, the $\mathbf{E}$ matrix at any parareal iteration k can be written as

$$\begin{aligned}\mathbf{E}(k) &= \mathbf{A}^{\text{approx}} - \mathbf{A}^{\text{exact}} \\ &= (\mathbf{F}^N)^T P(k) - (\mathbf{F}^N)^T \mathbf{F}^N \\ &= -(\mathbf{F}^N)^T (\mathbf{F}^N - P(k))\end{aligned}$$

Using (14) we get,

$$\mathbf{E}(k) = -(\mathbf{F}^N)^T \sum_{p=k+1}^{N} C_p^N (\mathbf{F}-\mathbf{G})^p \mathbf{G}^{N-p} \tag{19}$$

From (18) it is clear that a sufficiently accurate residual $\mathbf{r}(\lambda)$ and its $\mathbf{A}^{-1}$ norm is required to monitor $\|\mathbf{r}(\lambda)\|_{\mathbf{A}^{-1}}$. But the presence of errors at each inexact iteration j also results in the computed residual $\mathbf{r}_j$ being different from the exact residual $\mathbf{r}(\lambda_j) = \mathbf{b} - \mathbf{A}\lambda_j$. In order to still keep track of the quadratic change, it is necessary to appropriately bound the inexact residual norm $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ and the residual gap norm $\|\mathbf{r}(\lambda_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}}$. We are also going to need the primal-dual norm of the perturbation matrix defined as

$$\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}} = \sup_{\lambda \neq 0} \frac{\|\mathbf{E}_j\lambda\|_{\mathbf{A}^{-1}}}{\|\lambda\|_{\mathbf{A}}} = \|\mathbf{A}^{-1/2}\mathbf{E}_j\mathbf{A}^{-1/2}\|_2$$

We have dropped the variable k while defining the error norm for clarity. So in the later sections it will be present only when it is explicitly required in the context. Now for some permissible error at iteration j, if the inexact residual norm and the residual gap norm can be suitably bounded, then the change in the quadratic can be controlled. The following theorem captures the essence of the above idea [15],

Theorem 1. *Let $\epsilon > 0$ and let $\phi \in \mathbb{R}^j$ be a positive vector satisfying*

$$\sum_{i=1}^{j} \frac{1}{\phi_i} \leq 1$$

Suppose that

$$\|\mathbf{E}_i\|_{\mathbf{A}^{-1},\mathbf{A}} \leq \omega_i = \frac{\sqrt{\epsilon}\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_i\|_{\mathbf{A}}}{2\phi_{i+1}\|\mathbf{r}_i\|_2^2 + \sqrt{\epsilon}\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_i\|_{\mathbf{A}}} \tag{20}$$

for all $i \in \{0, ..., j-1\}$. Then

$$\|\mathbf{r}(\lambda_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}} \tag{21}$$

Additionally if

$$\|\mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}} \tag{22}$$

then $|q(\lambda_j) - q(\lambda_*)| \leq \epsilon|q(\lambda_*)|$.

Remark 4.1. The errors are permitted to grow as we proceed with the inexact CG iterations. As a consequence the search directions are no longer conjugate and the (inexact) residuals lose their orthogonality. In the theorem it is assumed that the inexact residual norm $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ eventually becomes smaller but it is not guaranteed. Adding a reorthogonalisation step could ensure that the residuals converge to zero after at most n steps.

Remark 4.2. Since the inexactness is introduced by another iterative method (Parareal in this case), for each outer minimisation iteration j there are many inner iterations k of the Parareal. The number of outer iterations depend on the stopping criteria (22) and the number of inner iterations on ω_j in equation (20).

4.2 Modified method

The stopping criteria (20) fails to act as a good bound when the condition number of $\mathbf{A}$ is large. In terms of efficiency the number of parareal iterations being used does not reflect upon the actual number of parareal iterations needed. We make a few modifications in the original method to fix this issue. In the proof of Theorem 1, the residual gap norm (21) is bounded using the inequality

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}\|\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \omega_j\|\mathbf{p}_j\|_{\mathbf{A}^{-1}}$$

We noticed that utilising $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ itself for the parareal stopping criterion gives us fairly reasonable estimates. It can be clearly seen that this in no way affect the original method. We introduce a new quantity ξ_j which satisfies,

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \omega_j\|\mathbf{p}_j\|_{\mathbf{A}^{-1}} = \xi_j$$

Thus from (20),

$$\xi_j = \frac{\sqrt{\epsilon}\|\mathbf{b}\|_{A^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}^2}{2\phi_{j+1}\|\mathbf{r}_j\|_2^2 + \sqrt{\epsilon}\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}} \tag{23}$$

Note: $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ is the product of the error matrix and the conjugate direction vector $\mathbf{p}_j$ at a given parareal iteration. This means that $\mathbf{p}_j$ is the initial condition for the parareal algorithm.

Remark 4.3. The modified criterion does not make use of $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$ anywhere in the algorithm. Thus, we don't need to know the perturbation matrix explicitly. It is sufficient for us if we know it indirectly in the form of a matrix vector product. With a given conjugate direction $\mathbf{p}_j$ the product $P(k)\mathbf{p}_j - \mathbf{A}\mathbf{p}_j$ is nothing but $\mathbf{E}_j(k)\mathbf{p}_j$.

Figure 2 below shows the number of parareal iterations for a shallow water model if we use the original criterion and the modified criterion for a large $\kappa(\mathbf{A})$. The red marked symbols indicate the values on the either side of the tolerance. It can be seen that there is a huge difference between the two criteria. On the left image we see that the bound is satisfied only at the second last iteration whereas on the right image it is already satisfied in 5 iterations.

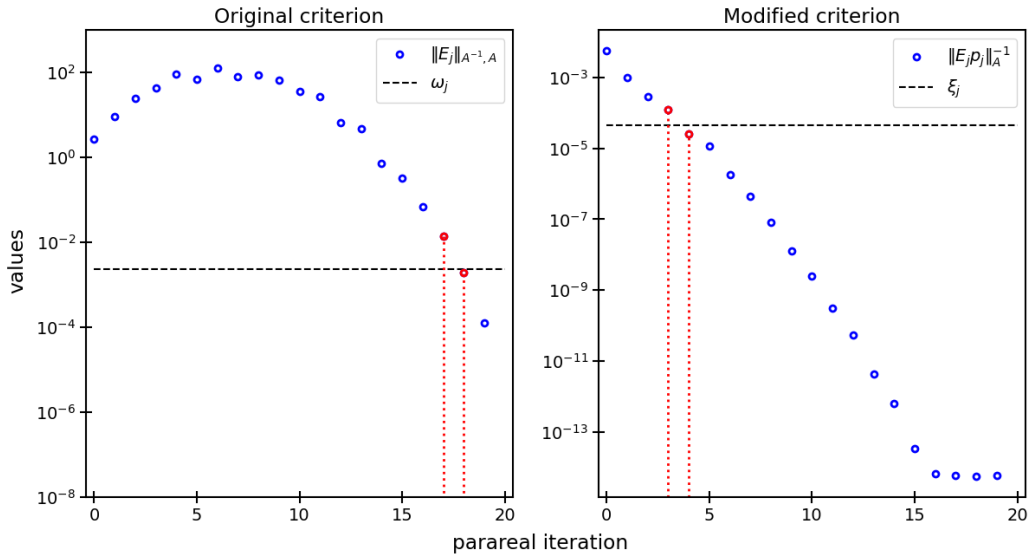


Figure 2: Estimates and bounds for inexact CG iteration 6, $N = 20, \kappa(\mathbf{A}) = 1099070.55$

From the figure on the left the curve resembles to the kind of convergence results as reported in [16] for pure advection problems. It says that even if the bound is superlinear the constants in the bounds for numerical method make it hard to get the speed up. The error starts increasing and only comes down

at the last iteration.

The relative decrease in the curve of $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ can be explained by the fact that even if there is an increase in $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$, theoretically the conjugate direction norm $\|\mathbf{p}_j\|_{\mathbf{A}} \to 0$ as $j \to j_{\max}$. While both criteria might give the same number of parareal iterations when the problem is well-conditioned, it is wise to use the modified criterion. For an ill-conditioned problem, it would be intuitive to see that even the norms depending on $\mathbf{A}^{-1}$ will not be really accurate.

Till now there has been no talk about the role of ϕ_j which is discussed in the original method [15, see Sec 3.1]. By definition ϕ_{j+1} puts a restriction on the bound ξ_j and so choosing a constant value of $\phi_j = j_{\max}$ limits the possibility of larger error allowance. In fact the smaller the value of ϕ_{j+1}, the larger the bound for ξ will be. This can actually be done by managing the inaccuracies obtained by the difference between the computed ξ_j and the allowed inexactness $\hat{\xi}_j$(say). This difference is unused and could be utilised by distributing to the subsequent inexact CG iterations. The corresponding $\hat{\phi}_{j+1}$ can be obtained from $\hat{\xi}_j = \xi_j(\hat{\phi}_{j+1})$ in (23) as

$$\hat{\phi}_{j+1} = \frac{(\|\mathbf{p}_j\|_{\mathbf{A}} - \hat{\xi}_j)}{\hat{\xi}_j} \frac{\sqrt{\epsilon}\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}}{2\|\mathbf{r}_j\|_2^2} > \phi_{j+1} \tag{24}$$

The inaccuracy can be distributed the same way as in the original algorithm.

4.3 Inexact para-conjugate gradient

So far we have discussed about the method from a theoretical point of view with an assumption that we have access to the quantities or norms concerning the matrix $\mathbf{A}$ and its inverse. But realistically one might not even have access to the matrix $\mathbf{A}$, let alone its inverse. Thus in practice we will use the following approximations [15] of all the quantities which are unfeasible to compute.

- $\|\mathbf{p}_j\|_{\mathbf{A}} \approx \sqrt{\frac{1}{n}\mathrm{Tr}(\mathbf{A})}\,\|\mathbf{p}_j\|_2$
- $q(\lambda_j) \approx q_j \overset{\text{def}}{=} -\frac{1}{2}\mathbf{b}^T\lambda_j$
- $\|\mathbf{b}\|_{\mathbf{A}^{-1}} \approx \frac{\|\mathbf{b}\|_2}{\sqrt{\lambda_{\max}}} \quad j = 0,\ \lambda_0 = 0 \quad \text{and} \quad \|\mathbf{b}\|_{A^{-1}} \approx \sqrt{2|q_j|} \quad j = 1, 2, ..., j_{\max}$
- Termination test (22) by $q_{j-d} - q_j \leq \frac{1}{4}\epsilon|q_j|$ for some integer d

Note: Here $\lambda_{\max}$ is the maximum eigenvalue of the matrix $\mathbf{A}$ and should not be confused with the state variable λ.

While $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ is a suitable stopping criterion for the parareal iterations, it is not easy to compute as well. It turns out that it can be replaced by another equivalent and computable quantity.

Theorem 2. *If $\mathbf{F}$ is invertible, $\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ and $\|\hat{\mathbf{p}}_j(k) - \mathbf{p}_*\|_2$ are rigorously the same whatever the parareal approximation is. $\hat{\mathbf{p}}_j(k)$ is the parareal approximation at iteration k with $\mathbf{p}_j$ as the initial condition and $\mathbf{p}_*$ is the corresponding exact solution.*

Proof. We have,

$$\begin{aligned}
\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}} &= \left\langle \mathbf{E}_j(k)\mathbf{p}_j, A^{-1}\mathbf{E}_j(k)\mathbf{p}_j \right\rangle \\
&= \left\langle (\mathbf{F}^N)^T\mathbf{F}^N - (\mathbf{F}^N)^T P(k)]\,\mathbf{p}_j,\ [(\mathbf{F}^N)^T\mathbf{F}^N]^{-1}\,[(\mathbf{F}^N)^T\mathbf{F}^N - (\mathbf{F}^N)^T P(k)]\,\mathbf{p}_j \right\rangle \\
&= \left\langle (\mathbf{F}^N)^T[(\mathbf{F}^N - P(k))\,\mathbf{p}_j\,],\ [(\mathbf{F}^N)^T\mathbf{F}^N]^{-1}\,(\mathbf{F}^N)^T[(\mathbf{F}^N - P(k))\,\mathbf{p}_j\,] \right\rangle \\
&= \left\langle (\mathbf{F}^N - P(k))\,\mathbf{p}_j\,,\ \mathbf{F}^N[(\mathbf{F}^N)^T\mathbf{F}^N]^{-1}\,(\mathbf{F}^N)^T[(\mathbf{F}^N - P(k))\,\mathbf{p}_j\,] \right\rangle \\
&= \left\langle (\mathbf{F}^N - P(k))\,\mathbf{p}_j\,,\ [\mathbf{F}^N(\mathbf{F}^N)^\dagger]\,[(\mathbf{F}^N - P(k))\,\mathbf{p}_j\,] \right\rangle
\end{aligned}$$

where $(\mathbf{F}^N)^\dagger$ is the Moore-Penrose generalised inverse or pseudo-inverse of $\mathbf{F}^N$. Thus when $\mathbf{F}$ is invertible (i.e. $(\mathbf{F}^N)^\dagger = (\mathbf{F}^N)^{-1}$) we have

$$\begin{aligned}\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}} &= \left\langle (\mathbf{F}^N - P(k))\,\mathbf{p}_j\,,\,(\mathbf{F}^N - P(k))\,\mathbf{p}_j\,)\right\rangle \\ &= \left\langle \mathbf{F}^N\mathbf{p}_j - P(k)\mathbf{p}_j\,,\,\mathbf{F}^N\mathbf{p}_j - P(k)\mathbf{p}_j\right\rangle \\ &= \|\hat{\mathbf{p}}_j(k) - \mathbf{p}_*\|_2\end{aligned}$$

□

One last hurdle is to find a computable estimate for $\mathbf{p}_*$ since we do not want the parareal algorithm to run with a very small tolerance each time. Looking at Figure 3 we notice that the curve profile of $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ does not change much for given j. Then replacing $\|\hat{\mathbf{p}}_j(k) - \mathbf{p}_*\|_2$ by $\|\hat{\mathbf{p}}_j(k) - \mathbf{p}_{**}\|_2$ can still lead to convergence where $\mathbf{p}_{**}$ is the last parareal iterate. All we need is an accurate parareal solution in the beginning of the inexact CG to ensure that the computed last parareal iterates are reliable as well.

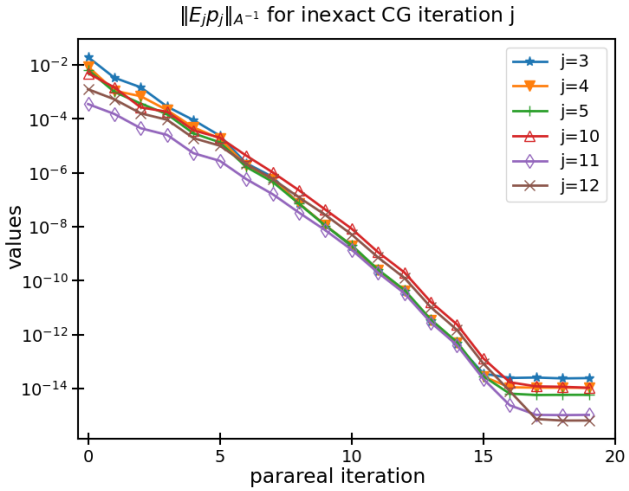


Figure 3: Profile of $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ for different CG iterations

For a clear explanation, suppose for instance that the initial (accurate) parareal solution $\hat{\mathbf{p}}_0$ takes $\bar{k}$ iterations. Then for the next inexact CG iteration, $\|\hat{\mathbf{p}}_1(k) - \mathbf{p}_*\|_2$ is approximated by $\|\hat{\mathbf{p}}_0(k) - \mathbf{p}_{**}\|_2$ where $\mathbf{p}_{**} \approx \hat{\mathbf{p}}_0(\bar{k})$. We then find the required k which satisfies $\|\hat{\mathbf{p}}_0(k) - \mathbf{p}_{**}\|_2 \leq \xi_1$, say $k = k_1$. Now the parareal solution $\hat{\mathbf{p}}_1(k_1)$ will be used to compute the matrix-vector multiplication . Going to the next iteration, $\|\hat{\mathbf{p}}_2(k) - \mathbf{p}_*\|_2$ will be approximated by $\|\hat{\mathbf{p}}_1(k) - \mathbf{p}_{**}\|_2$ with $\mathbf{p}_{**} \approx \hat{\mathbf{p}}_1(k_1)$. We proceed similarly for the subsequent minimisation iterations till convergence.

Since we are using the solution from the last parareal iterate we would like to make sure we always have reliable estimates when bounding $\|\hat{\mathbf{p}}_j(k) - \mathbf{p}_{**}\|_2$ with ξ_j. For this it is important that the value of ξ_j does not lie between the last two values of k (say). If this happens, then we move on to the current parareal iterate where we compute the solution with the current $\mathbf{p}_j$ as the initial condition. We proceed the same way as before but and check again if the bound ξ_j lies above the last two iterates. If not, we keep doing one more parareal iteration until our condition is satisfied.

Again from the above example, we use the current parareal iterate if $\|\hat{\mathbf{p}}_0(k_1) - \mathbf{p}_{**}\|_2 \leq \xi_1$ but $\xi_1 \in [\bar{k} - 1, \bar{k})$. Like before we run the parareal for k_1 iterations to get $\hat{\mathbf{p}}_1(k_1)$. What changes now is that we look for k such that $\|\hat{\mathbf{p}}_1(k) - \hat{\mathbf{p}}_1(k_1)\|_2 \leq \xi_1$. If the required k still falls between the last two values, we perform one more parareal iteration and keep doing it until our condition is satisfied. So if it takes k_* more iterations then we update $k1 := k1 + k_*$. This discussion has been put in the form of an algorithm for $\mathbf{p}_*$ approximation which can be implemented.

Algorithm 2: p_* approximation test

1 **Given**: inexact CG iteration j
2 Find ξ_j from equation (23)
3 `Last parareal test`
4 **if** $j = 0$ **then**
5 | Compute the parareal solution $\hat{\mathbf{p}}_0$ with $\epsilon_{\text{p}} = \epsilon_{\text{cg}}/10$. Let the total iterations taken be k_0
6 | Set $k_{\text{sol}} = k_0$, $\mathbf{p}_{**} = \hat{\mathbf{p}}_0(k_{\text{sol}})$
7 **else**
8 | Find $0 \leq k < k_{\text{sol}}$ such that $\|\hat{\mathbf{p}}_{j-1}(k) - \mathbf{p}_{**}\|_2 \leq \xi_j$; save LHS as $\hat{\xi}_j$
9 | $k_{\text{sol}} = k$
10 | **if** $k = k_{\text{sol}} - 1$ **then**
11 | | cp = True `Use current parareal iterate`
12 Compute the current parareal solution p_j for k_{sol} iterations
13 Set $\mathbf{p}_{**} = \hat{\mathbf{p}}_j(k_{\text{sol}})$
14 **if** cp = True **then**
15 | Find $0 \leq k < k_{\text{sol}}$ such that $\|\hat{\mathbf{p}}_j(k) - \hat{\mathbf{p}}_j(k_{\text{sol}})\|_2 \leq \xi_j$; save LHS as $\hat{\xi}_j$
16 | **if** $k = k_{sol} - 1$ **then**
17 | | Perform one more parareal iteration and set $k_{\text{sol}} = k_{\text{sol}} + 1$
18 | Repeat steps in lines 15-17 till $k \neq k_{\text{sol}} - 1$
19 | $k_{\text{sol}} = k$
20 Calculate the inexact matrix-vector multiplication $\mathbf{c_j} = (\mathbf{F}^N)^T \mathbf{p}_j(k_{\text{sol}})$

This test is illustrated in Figure 4 for inexact CG iteration 17. We can see that when for the last parareal iterate, ξ_{17} lies between the second last and the last iteration k. So we proceed for the current parareal iterate (on the right) and now ξ_{17} is above the last two k.

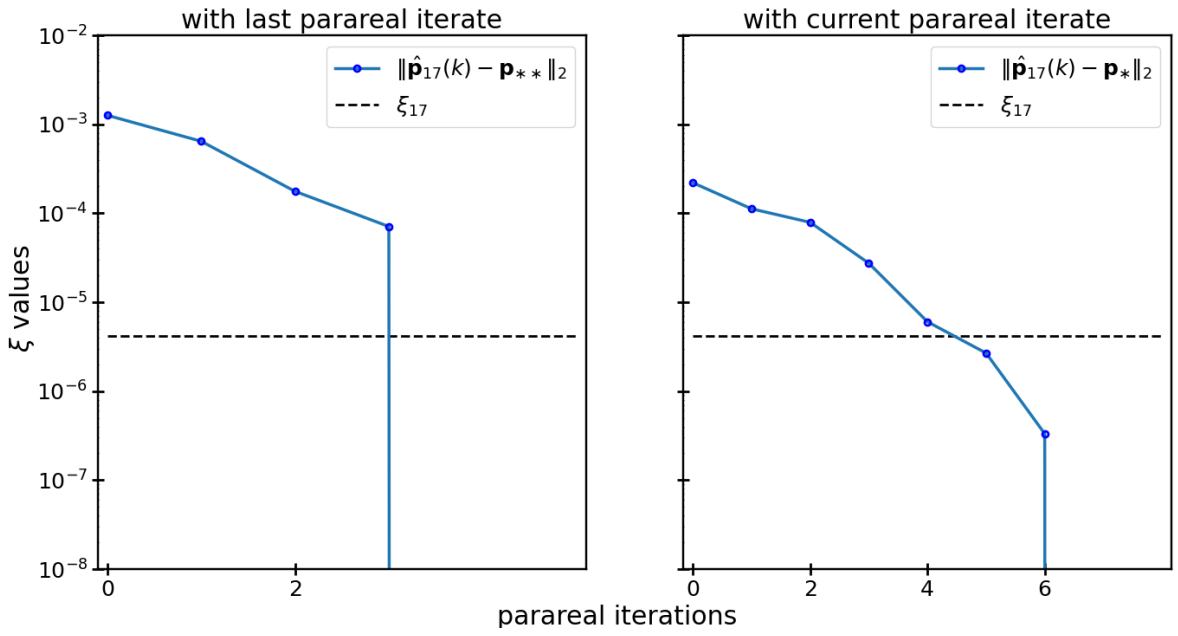


Figure 4: Using last and current parareal iterate

Algorithm 3: Inexact Para-conjugate Gradient

Given: symmetric positive definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, $\epsilon > 0$, reorth, $\lambda_0 = 0, r_0 = -\mathbf{b}, \mathbf{p}_0 = \mathbf{b}, \beta_0 = \|\mathbf{b}\|_2^2, \mathbf{u}_1 = \mathbf{b}/\beta_0, \phi_0 = j_{\max}$ and $\Phi_0 = 1$

`Iterations`

$j = 0$

while $j \leq j_{\max}$ **do**

  Use the $\mathbf{p}_*$ approximation test in Algorithm 3. Also store the value of $\hat{\xi}_j$.

  Compute $\hat{\phi}_j$ from (24)

  $\Phi_{j+1} = \Phi_j - \hat{\phi}_j^{-1}$

  if $j < j_{\max}$ **then**

    $\phi_{j+1} = (j_{\max} - j - 1)/\Phi_{j+1}$

  else

    $\phi_{j+1} = \phi_j$

  $\alpha_j = \beta_j/\mathbf{p}_j^T\mathbf{c}_j$

  $\lambda_{j+1} = \lambda_j + \alpha_j\mathbf{p}_j$

  $\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j\mathbf{c}_j$

  if $(q_{j+1-d} - q_{j+1}) \leq \frac{1}{4}\epsilon|q_{j+1}|$ **then**

    break

  if (reorth) **then**

    for $i = 1, 2, .., j$ **do**

      $\mathbf{r}_{j+1} = \mathbf{r}_{j+1} - (\mathbf{u}_i^T\mathbf{r}_{j+1})\mathbf{u}_i$

    $\beta_{j+1} = \mathbf{r}_{j+1}^T\mathbf{r}_{j+1}$

    $\mathbf{u}_{j+1} = \mathbf{r}_{j+1}/\sqrt{\beta_{j+1}}$

  else

    $\beta_{j+1} = \mathbf{r}_{j+1}^T\mathbf{r}_{j+1}$

  $\mathbf{p}_{j+1} = -\mathbf{r}_{j+1} + (\beta_{j+1}/\beta_j)\mathbf{p}_j$

5 Numerical Experiments

5.1 A simple example/First experiments

One of the main reasons of this coupling is to use it for the applications related to ocean or climate modelling. Such problems are generally solved at a low resolution which results in integrating the model for a very long period of time like some decades or centuries. This gives us an opportunity to investigate the parallel-in-time methods for large grid sizes. We start with the simplest case of a linearised one-dimensional shallow water equations.

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= -H\frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial t} &= -g\frac{\partial \eta}{\partial x} \end{aligned} \tag{25}$$

where η and u are the free surface and velocity respectively. H is the characteristic height and g is gravity. This is clearly a hyperbolic problem and to numerically solve it we are going to add an explicit diffusion. The reason being to explain the effects of the unresolved scales in terms of the large scales. This so called the closure of the system is performed by the Reynolds averaging. The modified equations are given as

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= -H\frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial t} &= -g\frac{\partial \eta}{\partial x} + \mu\frac{\partial^2 u}{\partial x^2} \end{aligned} \tag{26}$$

where μ is the diffusion constant.

The numerical discretisiation is done on an Arakawa C-grid or a staggered grid.

$$\begin{aligned}\frac{\eta_{i+1/2}^{k+1}-\eta_{i+1/2}^{k}}{\Delta t} &= -H\,\frac{u_{i+1}^{k}-u_{i}^{k}}{\Delta x}+\frac{u_{i+1}^{k}-2u_{i}^{k}+u_{i-1}^{k}}{\Delta x^2}\\ \frac{u_{i+1}^{k+1}-u_{i+1}^{k}}{\Delta t} &= g\frac{\eta_{i+1/2}^{k}-\eta_{i-1/2}^{k}}{\Delta x}+\frac{u_{i+1}^{k}-2u_{i}^{k}+u_{i-1}^{k}}{\Delta x^2}\end{aligned} \tag{27}$$

We take $H = 0.9, g = 10$ and the characteristic length $L = 120$. If the number of spatial grid points are 120, then $\Delta x = L/120 = 1$. This can be expressed as the same ODE system in (10)

$$\frac{\mathrm{d}\lambda}{\mathrm{dt}} = \mathbf{B}\lambda \tag{28}$$

where

$$\mathbf{B} = \left(\begin{array}{cccccc|ccccc} & & & & & & -H/\Delta x & & & & \\ & & & & & & H/\Delta x & -H/\Delta x & & & \\ & & & 0 & & & & \ddots & \ddots & & \\ & & & & & & & & H/\Delta x & -H/\Delta x & \\ & & & & & & & & & H/\Delta x & \\ \hline g/\Delta x & -g/\Delta x & & & & & -2\mu/\Delta x^2 & \mu/\Delta x^2 & & & \\ & g/\Delta x & -g/\Delta x & & & & \mu/\Delta x^2 & -2\mu/\Delta x^2 & \mu/\Delta x^2 & & \\ & & \ddots & \ddots & & & & \ddots & & & \\ & & & g/\Delta x & -g/\Delta x & & & \mu/\Delta x^2 & -2\mu/\Delta x^2 & \mu/\Delta x^2 & \\ & & & & & & & & \mu/\Delta x^2 & -2\mu/\Delta x^2 \end{array}\right)$$

and $\lambda = (\eta_0, \eta_1, \ldots \eta_{59}, u_0, u_1, \ldots u_{59})^T$

To solve this linear system of ODE (28) we use a weighted θ-scheme

$$\frac{\lambda^{k+1}-\lambda^{k}}{\Delta t} = \mathbf{B}[\,\theta\lambda^{k+1} + (1-\theta)\lambda^{k}\,]$$

which we can rewrite as

$$\lambda^{k+1} = \frac{[\,\mathbf{I}+\mathbf{B}(1-\theta)\Delta t\,]}{\mathbf{I}-\mathbf{B}\theta\Delta t}\,\lambda^{k}$$

Now we solve the above setup using the Parareal algorithm. The number of time windows N is taken to be 20. For each time window, we associate N_f and N_c as the number of fine time steps and coarse time steps respectively. Clearly N_c has to divide N_f. If δt is the fine step length, then the coarse step length Δt is given as

$$\Delta t = \frac{\delta t \times N_f}{N_c}$$

while the total integration time is $T = \delta t \times N_f \times N$

Here, $N_f = 100$ and $N_c = 25$. The fine and coarse propagators are defined as

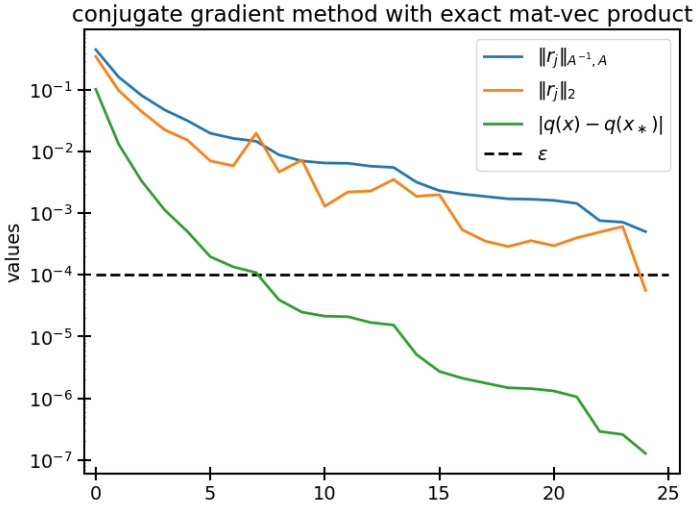
$$\mathbf{F} = \left(\frac{\mathbf{I}+(1-\theta)\delta t\mathbf{B}}{\mathbf{I}-\theta\delta t\mathbf{B}}\right)^{N_f} \quad , \quad \mathbf{G} = \left(\frac{\mathbf{I}+(1-\theta)\Delta t\mathbf{B}}{\mathbf{I}-\theta\Delta t\mathbf{B}}\right)^{N_c}$$

The following values of the parameters are used:

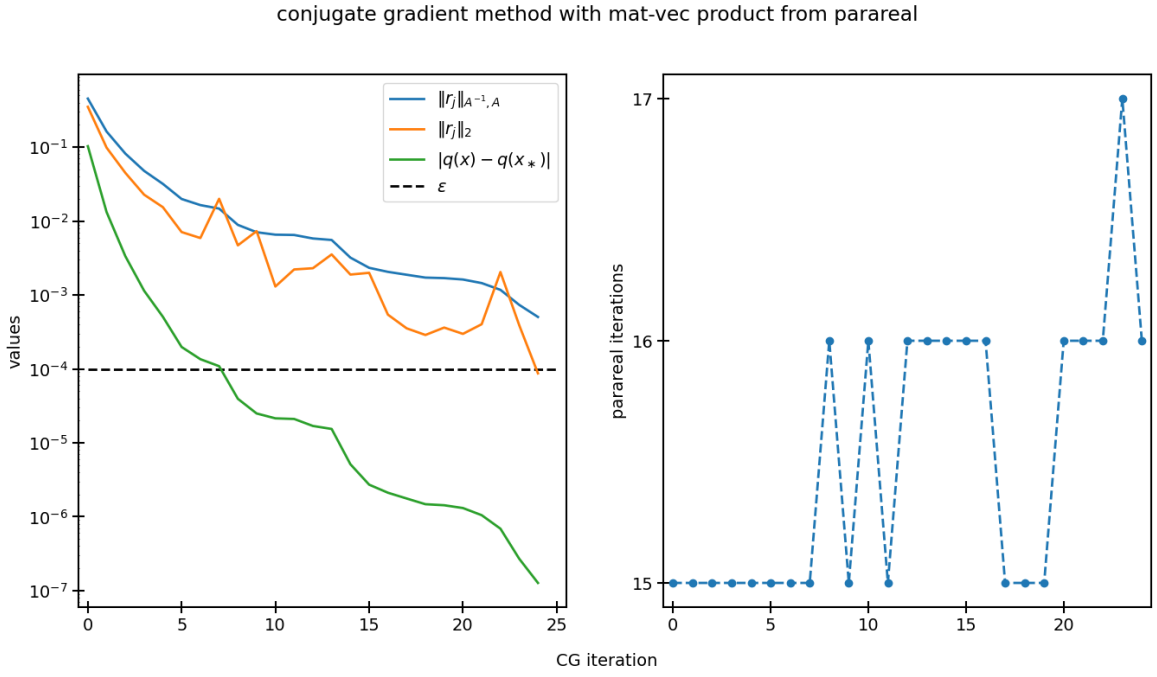
- $\theta = 0.51$
- regularisation parameter $\alpha = 10^{-5}$
- viscosity coefficient $\mu = 0.15$
- fine time step $\delta t = 0.05$
- initial condition : a Gaussian at the middle of the domain $\lambda_0 = \mathrm{e}^{[(x-L/2)/(L/15)]^2}$

5.2 Analysis

First is the result of the exact conjugate gradient method which uses exact matrix-vector product. The tolerance ϵ_{cg} is set to 10^{-4} and it takes 25 iterations.

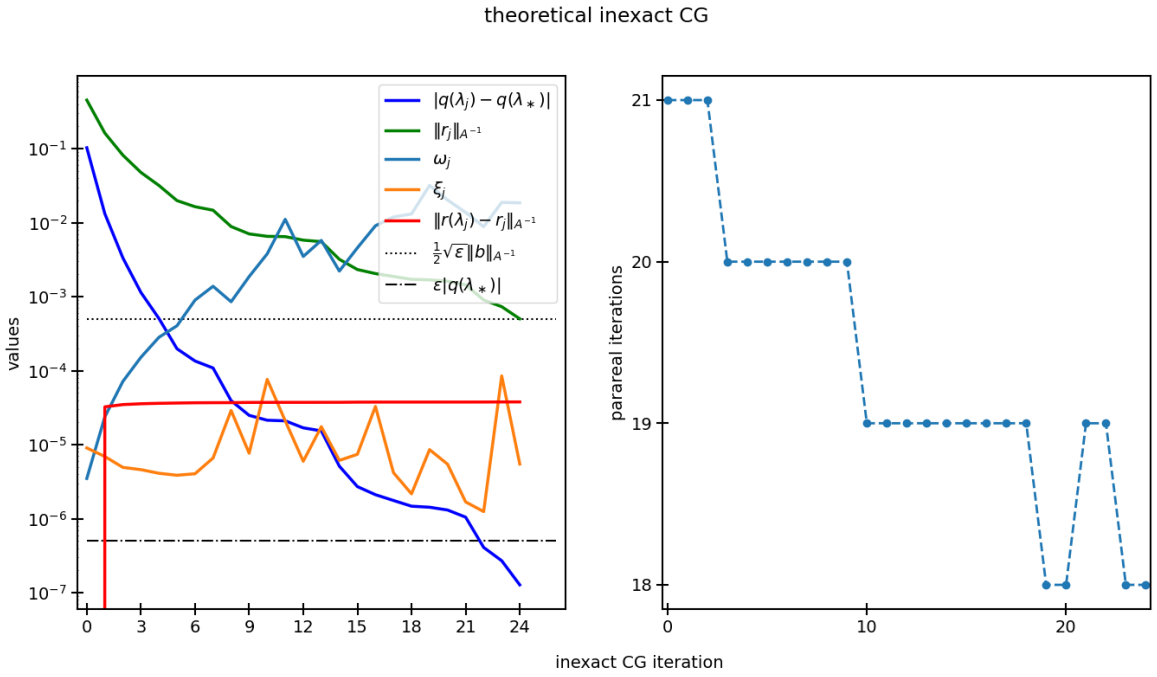


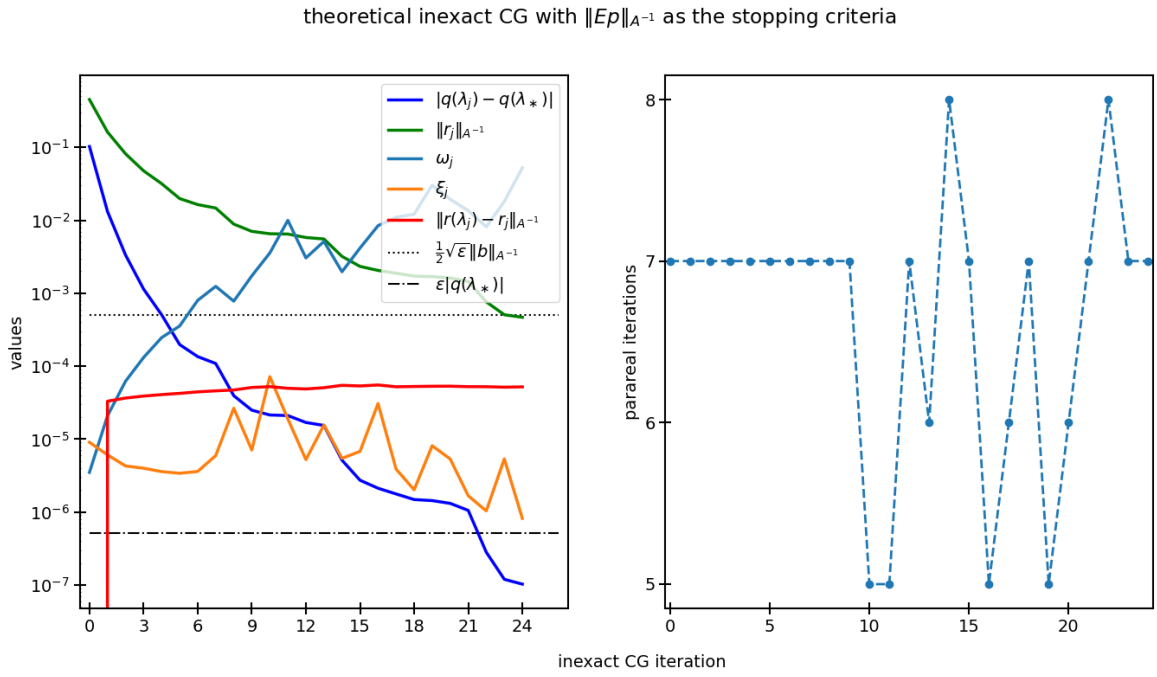
For the same minimisation tolerance now the matrix-vector product is supplied through the Parareal method. To mimic the same results obtained through the exact matrix-vector product we need to choose a very low tolerance for parareal. For $\epsilon_{\text{p}} = 10^{-12}$ as a fixed stopping criterion, it takes 388 parareal iterations.



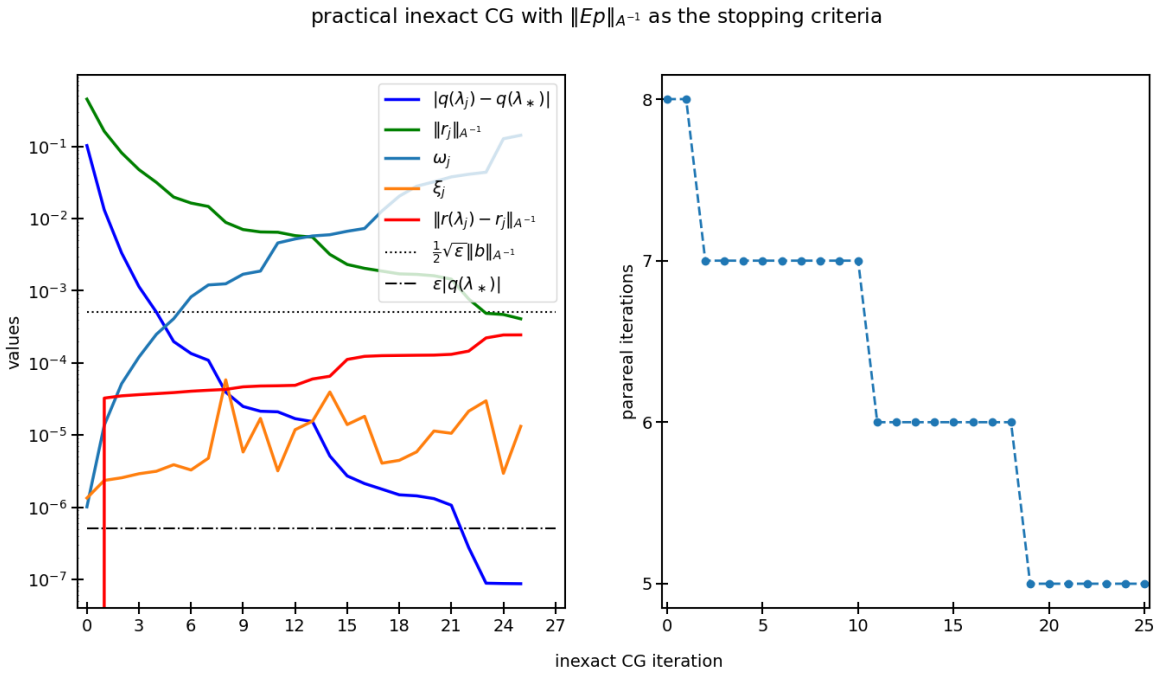
The above 1-D shallow water model with the given parameters is now considered in the context of Data Assimilation problem. From the above exact CG solution with $\epsilon_{\text{cg}} = 10^{-4}$, we find the value of $\|r_k\|_{A^{-1}}$ and use it in (22) to calculate a suitable stopping criteria for the inexact CG. We find that $\epsilon_{\text{icg}} = 1.12 \times 10^{-7}$.

With the theoretical estimates, we use the norm $\|E_j\|_{A^{-1},A}$ and it takes 25 minimisation iterations. But as expected it ends up using 484 parareal iterations which is not feasible at all. Next we replace $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$ with $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$ and keep everything the same. While the minimisation takes 31 iterations, the total parareal iterations goes down to 204.

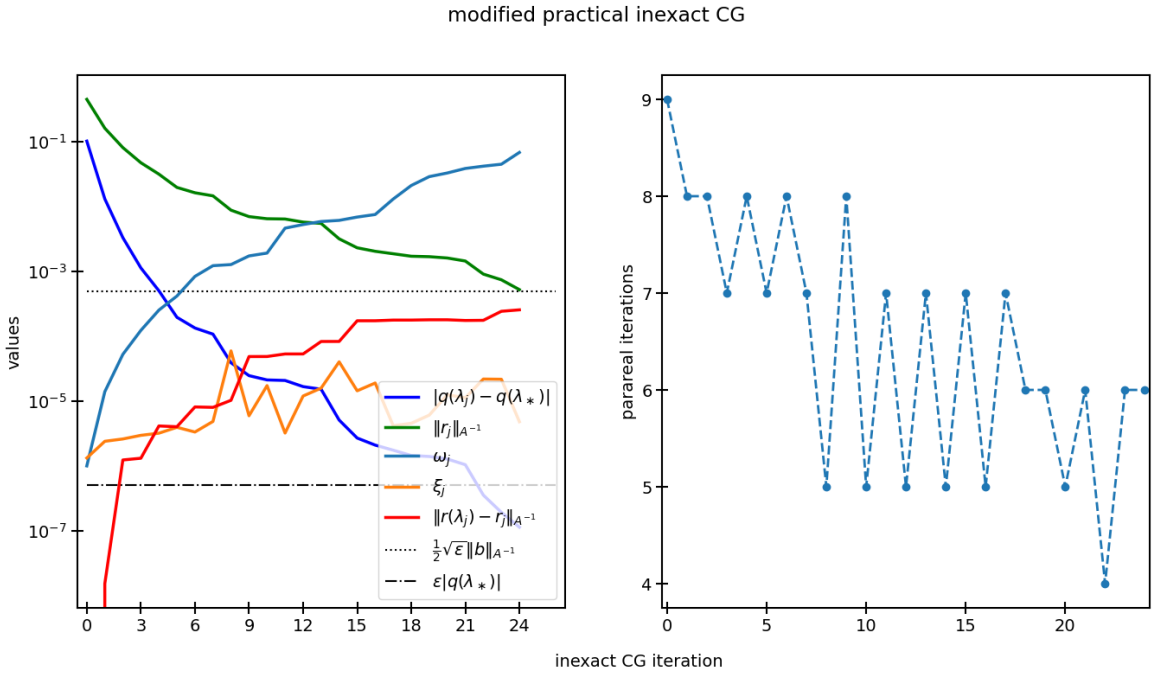




With the approximations of the quantities from Section 4.3 and the $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$, the practical inexact CG performs equally well. The integer value taken for the stopping criterion is $d = 5$. For 32 minimisation iterations, it used up 192 parareal iterations. As previously mentioned the inexact residuals quickly lose their orthogonality and this could lead to more iterations and less accurate results. Adding a reorthogonalisation step at each minimisation iteration fixes this issue. In this case, for $d = 2$, the minimisation ends in 26 iterations taking an overall of 162 parareal iterations.



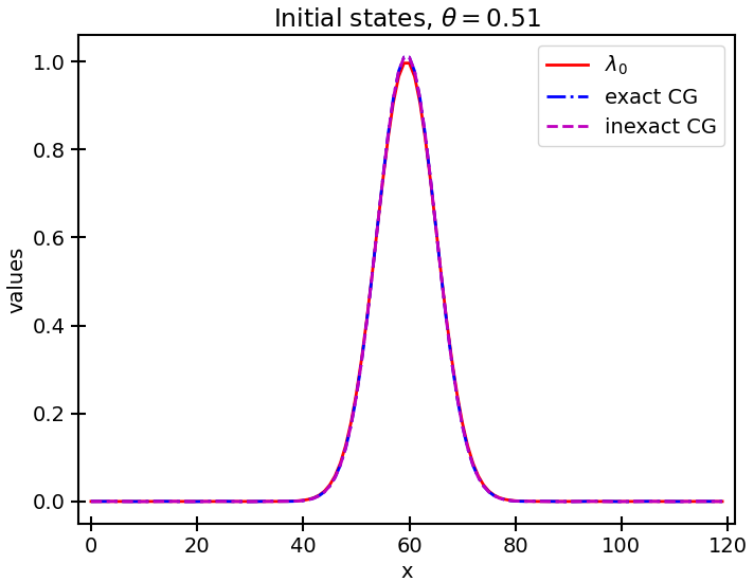
Finally, we show the main result by combining the practical estimates, the approximation for the modified stopping criterion and reorthogonalisation step. We choose $d = 5$. Remarkably, the minimisation and parareal iterations are similar as in the case of using $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ with 25 and 162 respectively. The parareal iterations oscillate by ± 1 which can be expected because we assume that the value of ξ_j should not lie between the last two parareal iterates.



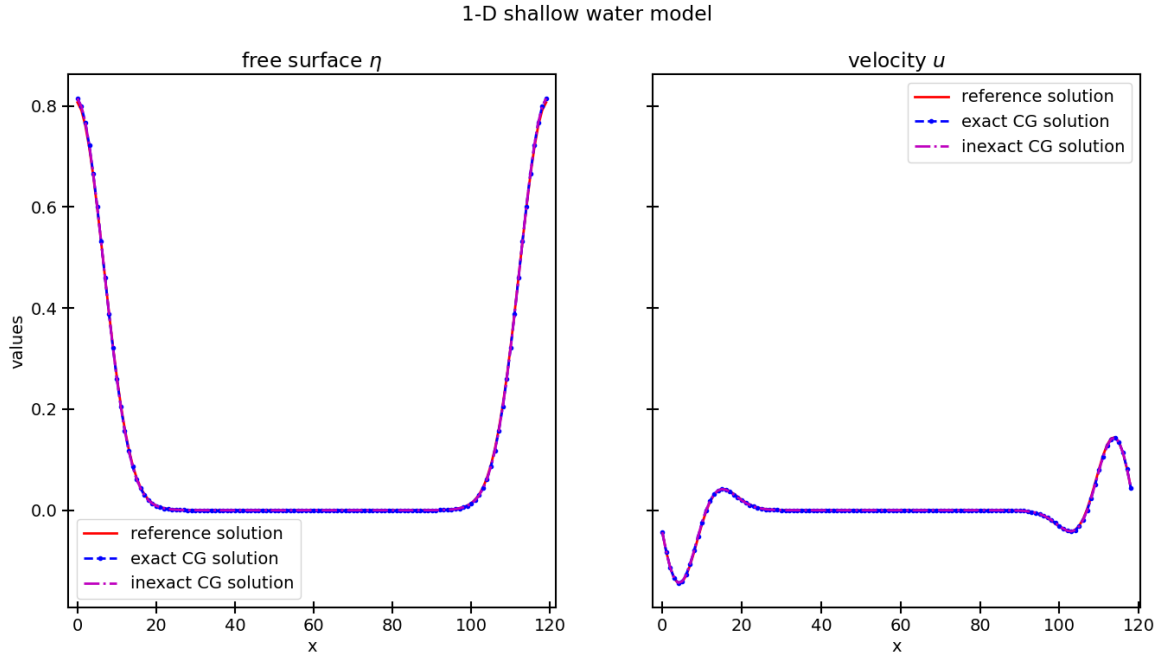
Some of the obtained parameter values

- courant number: 0.55
- diffusion courant number: 0.0075
- expected speedup: $\dfrac{20 \times 25}{162} = 3.08$

The initial condition retrieved by 4-D Var by using exact and inexact CG is almost identical to the initial state λ_0



As one would expect, the free surface and velocity values also coincide with the values obtained from the reference solution.



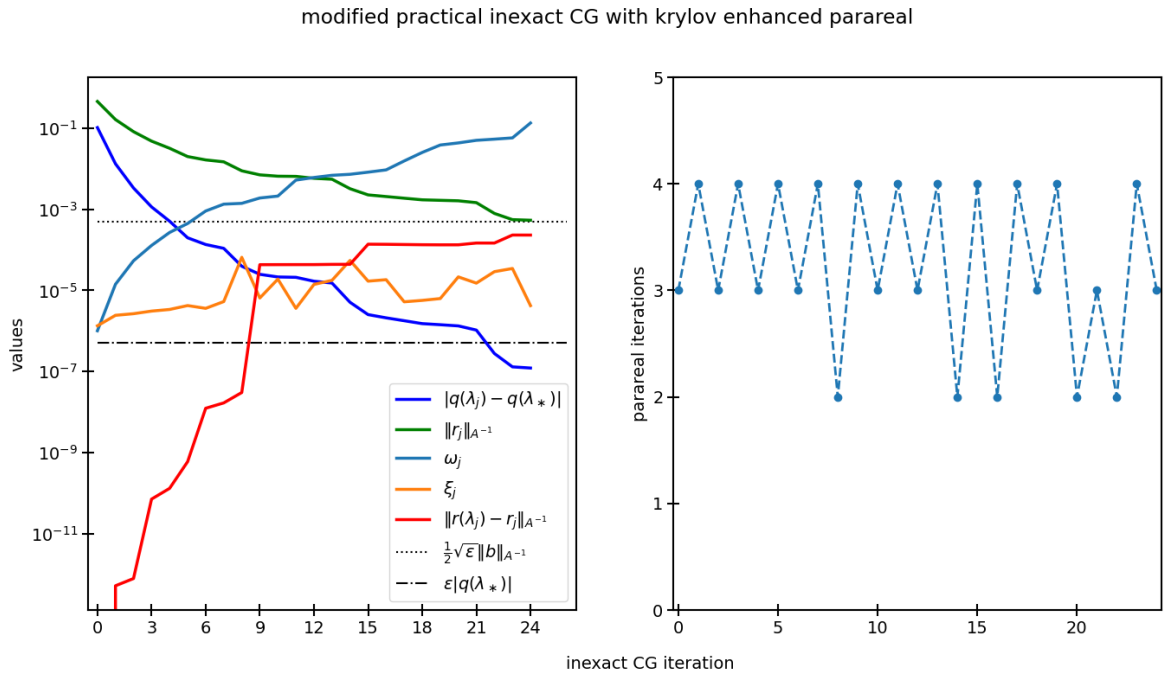
6 Conclusion and Future Work

In an attempt to couple a parallel-time method (here, Parareal) with Data Assimilation we see that one would expect the number of parareal iterations to go down eventually. But in fact it does not vary too much which could be accounted for by the values of ξ_j. We can at least conclude that the inexact conjugate method by Gratton et al. provides a well defined stopping criterion for the Parareal algorithm at each minimisation iteration if not any speedup. This way we don't have to run the parareal algorithm with very fine precision each time but only till where the bound is satisfied.

One thing to note is that the adjoint in the cost function (8) still uses the transpose of the matrix. We have not introduced time parallelisation for the backward model integration which could be the next

step. This can double the degree of parallelism in our algorithm.

Now it is well known that the Parareal algorithm shows instabilities when it is applied to hyperbolic problems and second order ODEs [17, 18]. One way to considerably increase the parareal's performance is by constructing a coarse solver from the subspace of the initial conditions at each time sub-domain [16]. This parareal variant is known as the Krylov enhanced parareal method. Running the modified inexact CG with Krylov enhanced version takes only 81 parareal iterations as compared to 162 parareal iterations with the usual version.



Clearly there is a huge improvement but it also has a drawback. Since one requires all the fine evaluations of the previous iterations beforehand, it faces a problem of storage and computational efficiency for higher dimensional problem. Using reduced order bases methods and choosing the right bases can be looked up as a direction for future research.

Finally parareal is just one of the techniques which can be employed for Data Assimilation. There are several other parallel-in-time methods for example the PFASST algorithm [19] or the ParaOpt algorithm [20] where the forward and backward model are solved together as a single optimality system.

References

[1] J. Nievergelt, "Parallel methods for integrating ordinary differential equations," *Communications of the ACM*, vol. 7, no. 12, pp. 731–733, 1964.

[2] M. J. Gander, "50 years of time parallel time integration," in *Multiple shooting and time domain decomposition methods*, pp. 69–113, Springer, 2015.

[3] F. Bouttier and P. Courtier, "Data assimilation concepts and methods march 1999," *Meteorological training course lecture series. ECMWF*, vol. 718, p. 59, 2002.

[4] P. Courtier, J.-N. Thépaut, and A. Hollingsworth, "A strategy for operational implementation of 4d-var, using an incremental approach," *Quarterly Journal of the Royal Meteorological Society*, vol. 120, no. 519, pp. 1367–1387, 1994.

[5] A. Lawless, S. Gratton, and N. Nichols, "An investigation of incremental 4d-var using non-tangent linear models," *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, vol. 131, no. 606, pp. 459–476, 2005.

[6] J.-L. Lions, Y. Maday, and G. Turinici, "Résolution d'edp par un schéma en temps pararéel," *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, vol. 332, no. 7, pp. 661–668, 2001.

[7] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, "Parallel-in-time molecular-dynamics simulations," *Physical Review E*, vol. 66, no. 5, p. 057701, 2002.

[8] M. J. Gander and S. Vandewalle, "Analysis of the parareal time-parallel time-integration method," *SIAM Journal on Scientific Computing*, vol. 29, no. 2, pp. 556–578, 2007.

[9] M. J. Gander and E. Hairer, "Nonlinear convergence analysis for the parareal algorithm," in *Domain decomposition methods in science and engineering XVII*, pp. 45–56, Springer, 2008.

[10] M. Minion, "A hybrid parareal spectral deferred corrections method," *Communications in Applied Mathematics and Computational Science*, vol. 5, no. 2, pp. 265–301, 2011.

[11] D. Ruprecht and R. Krause, "Explicit parallel-in-time integration of a linear acoustic-advection system," *Computers & Fluids*, vol. 59, pp. 72–83, 2012.

[12] I. C. Ipsen and C. D. Meyer, "The idea behind krylov methods," *The American mathematical monthly*, vol. 105, no. 10, pp. 889–899, 1998.

[13] V. Simoncini and D. B. Szyld, "Theory of inexact krylov subspace methods and applications to scientific computing," *SIAM Journal on Scientific Computing*, vol. 25, no. 2, pp. 454–477, 2003.

[14] J. Van Den Eshof and G. L. Sleijpen, "Inexact krylov subspace methods for linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 26, no. 1, pp. 125–153, 2004.

[15] S. Gratton, E. Simon, D. Titley-Peloquin, and P. L. Toint, "Minimizing convex quadratics with variable precision conjugate gradients," *Numerical Linear Algebra with Applications*, vol. 28, no. 1, p. e2337, 2021.

[16] M. Gander and M. Petcu, "Analysis of a krylov subspace enhanced parareal algorithm for linear problems," in *ESAIM: Proceedings*, vol. 25, pp. 114–129, EDP Sciences, 2008.

[17] D. Ruprecht, "Wave propagation characteristics of parareal," *Computing and Visualization in Science*, vol. 19, no. 1, pp. 1–17, 2018.

[18] G. A. Staff and E. M. Rønquist, "Stability of the parareal algorithm," in *Domain decomposition methods in science and engineering*, pp. 449–456, Springer, 2005.

[19] M. Emmett and M. Minion, "Toward an efficient parallel in time method for partial differential equations," *Communications in Applied Mathematics and Computational Science*, vol. 7, no. 1, pp. 105–132, 2012.

[20] M. J. Gander, F. Kwok, and J. Salomon, "Paraopt: A parareal algorithm for optimality systems," *SIAM Journal on Scientific Computing*, vol. 42, no. 5, pp. A2773–A2802, 2020.