Introducing time parallelisation within data assimilation using parareal method

Rishabh Bhatt, Laurent Debreu, Arthur Vidard

INTRODUCING TIME PARALLELISATION WITHIN 4D-VAR USING PARAREAL METHOD*

RISHABH BHATT†, LAURENT DEBREU†, AND ARTHUR VIDARD†

Abstract. Four dimensional variational data assimilation (4DVAR) methods, in their incremental formulation, are based on optimisation algorithms which require the integration of the forward and adjoint versions of the original model in order to compute its gradient. For their use on parallel computers, these models are classically parallelised only on the spatial dimension and this is a limiting factor on the maximum number of computing cores that can be used. We here present a novel approach of introducing additional time parallelisation using the parareal method. This approach is used here for the integration of the forward model. We use a modified version of the inexact conjugate gradient method where the matrix-vector multiplication is supplied through the parareal algorithm. The use of this inexact conjugate gradient and the associated convergence conditions allows to determine precisely the stopping criterion of the iterations of the parareal method. The results are demonstrated by considering a one dimensional shallow water model. They are presented in terms of the accuracy (in comparison with the original exact conjugate gradient) and in terms of the number of required iterations of the parareal algorithm.

Key words. Variational data assimilation, Parareal, Parallel-in-time methods, Optimisation, Shallow water equations

AMS subject classifications. 86A22, 68W10, 65L05, 65K10

1. Introduction. Predicting weather forecasts accurately for long periods remains a challenge for meteorological institutions. The observations available in the form of in-situ and satellite data are often error-prone and scarcely scattered. While the numerical models expressing the state of the atmosphere comes with assumptions and are discretisations of partial differential equations. The data assimilation methods are run with the objective that an optimum initial state could be recovered which corrects the previous forecasts and gives an improved prediction of the atmosphere. Operational variational data assimilation algorithms [16] are highly sequential in the sense that one needs to perform multiple forward and a backward model integrations to do gradient based minimisation. With billions of observations available along with an extremely high dimensional state variable, this process requires immense computing power.

This is one of the reasons why modern computation emphasises on reducing the clock time. With the advancement of massively fast computers millions of processors can be utilised at once. Speeding up data assimilation with space parallelisation by the means of domain-decomposition is a natural option [29]. Rantakokko [23] provided strategies to calculate the cost function and gradient in parallel by distributing the data (observations, grid point variables) using partitioning algorithms.

Space parallelisation however has its own limits and for large time-dependent problems it alone is insufficient. After one point it gets saturated even before effectively using up all the processors. Again for a climate simulation problem with 10^5 unknowns per model levels, using 10×10 unknowns for the horizontal domain would end up using 10^3 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. The time domain is notorious for its property of causality which makes parallelisation a difficult task. 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 minimise the influence of causality. It is not surprising that the first ideas of time parallelisation were already introduced in the early 60s by Nivergelt [21] 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 [9] provides a comprehensive knowledge of most of the important parallel-in-time algorithms. For a more

* **Funding:** This work was supported by ANR-ADOM (ANR-18-CE46-0008).

†Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France (rishabh.bhatt@inria.fr, laurent.debreu@inria.fr, arthur.vidard@inria.fr).

applications-based review article talking about the efficiency and speedup see [22].

There have been a few attempts to introduce time parallelism in data assimilation. Rao and Sandhu [24] presented the strong-constrained 4D-Var as a problem of minimising the Lagrangian with the model trajectories at the sub-intervals acting as constraints. This way the cost function and the gradient evaluations can be performed in parallel. Fischer and Gürol developed a parallel weak-constraint 4D-Var algorithm which works on finding the saddle-point of the Lagrangian [7]. To the authors' knowledge no work has been done to couple proper parallel-in-time method with 4D-Variational data assimilation. Our aim is to introduce time parallelisation with one such method (Parareal) in 4D-Var. We then see if we can exploit some of the available computational power to produce any possible speedup. We present our results for a 1-D linear shallow water model and show a theoretical measure of the speedup when an inexact version of the conjugate gradient method is used for the minimisation.

The paper is structured as follows. In section 2 we briefly talk about Variational data assimilation and how its incremental approach is related to our implementation for a linear model. 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 4D-Var cost function in Section 4. To maintain efficiency we will use the inexact Krylov subspace 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 calligraphic fonts for nonlinear functions/operators: $\mathcal{F}, \mathcal{J}$
- Upper case italics for linear functions/operators and matrices: G, H
- Bold lower case for vectors: $\mathbf{x}, \mathbf{y}$
- Greek letters and lower case for constants and variables

A list of the notation for all the important quantities used in the article is provided in the table below.

Table of quantities used	
Symbol	Meaning
$\mathbf{x}_i$	state variable at time t_i
$\Delta\mathbf{x}$	state variable increment
Δx	spatial grid stepping
$\mathbf{y}_i$	observation at time t_i
F	fine propagator/model
G	coarse propagator
E	perturbation matrix
$\mathbf{r}_j$	inexact residual at inexact CG iteration j
$P(k)$	parareal operator at kth iteration
$\mathbf{p}_j$	conjugate direction at inexact CG iteration j
ω_j	tolerance for $\|E_j\|_{A^{-1},A}$
ξ_j	tolerance for $\|E_j\mathbf{p}_j\|_{A^{-1}}$
$\kappa(A)$	condition number of a matrix A

2. Data assimilation. Let us describe our dynamical model depending on a state variable vector $\mathbf{x} \in \mathbb{R}^n$ by

$$\begin{aligned} \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{x}_i &= \mathcal{F}_{t_{i-1}\to t_i}(\mathbf{x}_{i-1}) \quad i = 1, ..., N \end{aligned} \tag{2.1}$$

where $\mathcal{F}$ 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 4D-Variational data assimilation or 4D-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 $\mathbf{y}_i^o \in \mathbb{R}^m$ at time t_i [3]. This can be expressed as a least-squares fit problem whose cost function is

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

Here B and R are the background and observation covariance matrices respectively. $\mathcal{H} : \mathbb{R}^n \to \mathbb{R}^m$ 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

$$\begin{aligned}\min_{\mathbf{x}_0 \in \mathbb{R}^n} \mathcal{J}(\mathbf{x}_0) &= \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}_0^b\|_b^2 + \frac{1}{2}\sum_{i=0}^{N}\|\mathbf{y}_i^o - \mathcal{H}_i(\mathcal{F}_{t_0\to t_i}(\mathbf{x}_0))\|_o^2 \\ &= \underbrace{\mathcal{J}_b(\mathbf{x}_0)}_{\text{background}} + \underbrace{\mathcal{J}_o(\mathbf{x}_0)}_{\text{observation}}\end{aligned} \tag{2.2}$$

This is known as the strongly constrained 4D-Var since the numerical model is assumed to be perfect. The norms $\|\cdot\|_b$ and $\|\cdot\|_o$ are the energy norms of the matrices B^{-1} and R^{-1} respectively.

Looking at (2.2) it is evident that the cost function $\mathcal{J}$ is non-convex because of the nonlinearites in $\mathcal{F}$ 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 $\Delta\mathbf{x}^{(k)}$

$$\begin{aligned}\mathcal{F}(\mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}) &= \mathcal{F}(\mathbf{x}^{(k)}) + F|_{\mathbf{x}^{(k)}}\,\Delta\mathbf{x}^{(k)} \\ \mathcal{H}(\mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}) &= \mathcal{H}(\mathbf{x}^{(k)}) + H|_{\mathcal{F}(\mathbf{x}^{(k)})}\,\Delta\mathbf{x}^{(k)} \\ \mathcal{H}\circ\mathcal{F}(\mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}) &= \mathcal{H}\circ\mathcal{F}(\mathbf{x}^{(k)}) + HF\Delta\mathbf{x}^{(k)}\end{aligned}$$

where $F = \dfrac{\partial\mathcal{F}}{\partial x}\Big|_{\mathbf{x}^{(k)}}$ and $H = \dfrac{\partial\mathcal{H}}{\partial x}\Big|_{\mathcal{F}(\mathbf{x}^{(k)})}$ are the linearised model and observation operator respectively.

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 [15]

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

2. 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 (2.2) can be written as

$$\begin{aligned}J(\Delta\mathbf{x}_0^{(k)}) &= \frac{1}{2}\|\Delta\mathbf{x}_0^{(k)} - (\mathbf{x}_0^b - \mathbf{x}_0^{(k)})\|_b^2 + \frac{1}{2}\sum_{i=0}^{N}\|\mathbf{y}_i - \mathcal{H}_i\circ\mathcal{F}(\mathbf{x}_i^{(k)} + \Delta\mathbf{x}_i^{(k)})\|_o^2 \\ &= \frac{1}{2}\|\Delta\mathbf{x}_0^{(k)} - (\mathbf{x}_0^b - \mathbf{x}_0^{(k)})\|_b^2 + \frac{1}{2}\sum_{i=0}^{N}\|\mathbf{d}_i - H_i F\Delta\mathbf{x}_i^{(k)}\|_o^2\end{aligned} \tag{2.3}$$

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

3. The constraint is given by

$$\begin{aligned}\Delta\mathbf{x}_i^{(k)} &= F_{t_0\to t_i}\Delta\mathbf{x}_0^{(k)} \\ &= F_i F_{i-1}\ldots F_2 F_1\Delta\mathbf{x}_0^{(k)}\end{aligned} \tag{2.4}$$

4. 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: 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 4D-Var (or incremental 4D-Var inner) cycle begins by a forward integration of the model (2.1) for calculating the cost function value. The obtained trajectory is stored for running the adjoint model for gradient computation. A potential speedup 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.

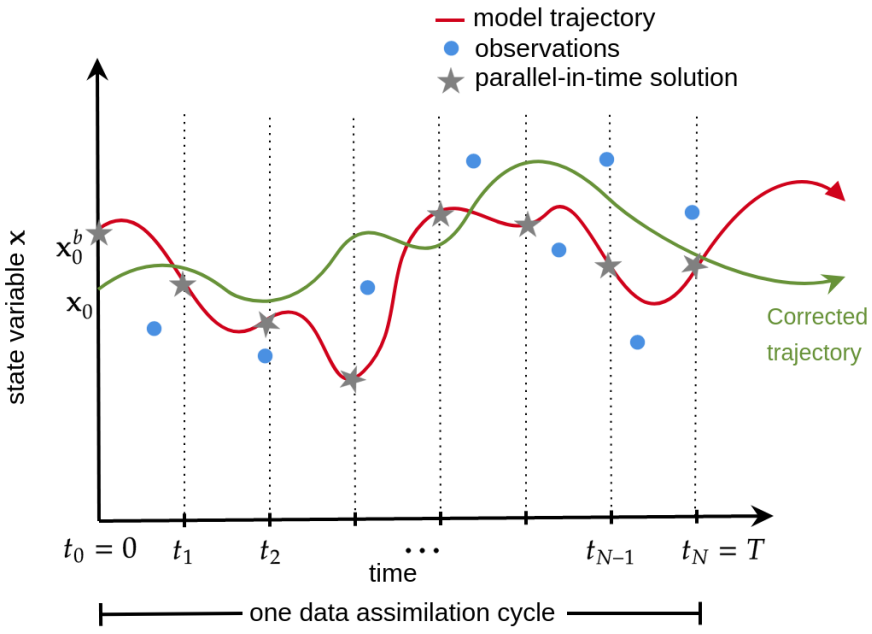


FIG. 2.1. *Using time parallelisation for running the forward model*

Let place ourselves in the case of a linear forward model

$$\begin{aligned} \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{x}_{i+1} &= F\mathbf{x}_i \quad i = 1, \dots N \end{aligned} \tag{2.5}$$

The model operator F is the same as the one used in (2.4). The only difference is that we have now replaced the state variable $\Delta\mathbf{x}$ with $\mathbf{x}$ to simplify notations.

Let us now set up our data assimilation problem. Again for the sake of simplicity 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 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

$$J(\mathbf{x}_0) = \frac{1}{2}\|F^N\mathbf{x}_0 - \mathbf{y}\|_2^2 \tag{2.6}$$

with gradient,

$$\nabla J(\mathbf{x}_0) = (F^N)^T(F^N\mathbf{x}_0 - \mathbf{y})$$

where $(F^N)^T$ 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 P (can be iterative or

direct), we have

$$J(\mathbf{x}_0) \approx \frac{1}{2}\|P\mathbf{x}_0 - \mathbf{y}\|_2^2$$
$$\nabla J(\mathbf{x}_0) \approx (F^N)^T(P\mathbf{x}_0 - \mathbf{y}) \tag{2.7}$$

Minimising this cost function requires setting the gradient $\nabla J(\mathbf{x}_0)$ to 0. This is equivalent to solving the system

$$A\mathbf{x} = \mathbf{b} \qquad \text{with} \quad A = (F^N)^T P \ , \ \mathbf{b} = (F^N)^T\mathbf{y}$$

At the end of each minimisation iteration we expect to have an initial condition with a lower cost function value. This in turn will be used by P to provide new trajectories for the next iteration. The process will end depending on the stopping criterion of the minimisation.

When $\kappa(A)$, the conditioning of A, is large, a regularisation term α can be used in the cost function which depends on the spatial derivative of the initial state $\mathbf{x}_0$. Thus,

$$J(\mathbf{x}_0) \approx \frac{1}{2}\|P\mathbf{x}_0 - \mathbf{y}\|_2^2 + \frac{\alpha}{2}\left\|\frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}\mathbf{x}}\right\|_2^2$$

The spatial derivative can be written as

$$\frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}\mathbf{x}} = \frac{\mathbf{x}_{0,i+1} - \mathbf{x}_{0,i}}{\Delta x} = \frac{1}{\Delta x} Q_1\mathbf{x}_0$$

where

$$Q_1 = \begin{pmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \\ & & & -1 \end{pmatrix}$$

Therefore,

$$\left\|\frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}\mathbf{x}}\right\|_2^2 = \frac{1}{\Delta x^2}\langle Q_1\mathbf{x}_0, Q_1\mathbf{x}_0\rangle$$

The simplified cost function can be written as

$$J(\mathbf{x}_0) \approx \frac{1}{2}\|P\mathbf{x}_0 - \mathbf{y}\|_2^2 + \frac{\alpha}{2\Delta x^2}\|Q_1\mathbf{x}_0\|_2^2$$

Written like this, the gradient should be

$$\nabla J(\mathbf{x}_0) \approx (F^N)^T(P\mathbf{x}_0 - \mathbf{y}) + \alpha Q_2\mathbf{x}_0$$

where

$$Q_2 = \frac{1}{\Delta x^2} Q_1^T Q_1 = \frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

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

$$A = (F^N)^T P + \alpha Q_2 \quad , \qquad \mathbf{b} = (F^N)^T\mathbf{y}$$

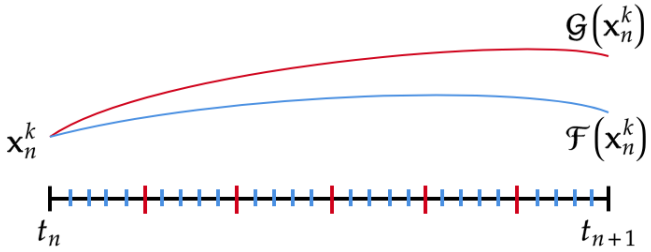
REMARK 2.1. *Using an iterative minimisation algorithm for the above linear system is equivalent to solving the incremental 4D-Var problem.*

Depending on the choice of the operator P 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 P is replaced by the parareal algorithm, which we will present in the next section.

3. Parareal method. Parareal was introduced in [17] and later reformulated in [1] 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 discrete model

$$\begin{aligned} \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{x}_{i+1} &= \mathcal{F}\mathbf{x}_i \quad i = 1, \dots N \end{aligned} \tag{3.1}$$

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 $\mathcal{F}$. $\mathcal{G}$ is chosen so that it is a very cheap approximation of $\mathcal{F}$, with large space/time steps making it less accurate. On the other hand, $\mathcal{F}$ has small space/time steps making it a highly accurate and computationally expensive solver. In the examples of this paper, only the time discretisation is different between $\mathcal{F}$ and $\mathcal{G}$, respectively denoted δt and Δt.



The parareal algorithm proceeds iteratively by computing the solution $\mathbf{x}_{n+1}^k \approx \mathbf{x}(t_{n+1})$ at the sub-intervals for given $\mathbf{x}_n^k$ at time t_n and iteration k. An initial configuration ($k = 0$) is obtained by a serial run of the coarse propagator $\mathcal{G}$ over the whole time domain. Thus,

$$\begin{aligned} \mathbf{x}_0^0 &= \mathbf{x}_0 \\ \mathbf{x}_{n+1}^0 &= \mathcal{G}(t_{n+1}, t_n, \mathbf{x}_n^0) \quad n = 0, 1, \dots, N-1 \end{aligned}$$

With the solution values now available at all the grid points, the fine propagator $\mathcal{F}$ can be run to provide more accurate estimates. 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

$$\begin{aligned} \mathbf{x}_0^{k+1} &= \mathbf{x}_0 \\ \mathbf{x}_{n+1}^{k+1} &= \underbrace{\mathcal{G}(t_{n+1}, t_n, \mathbf{x}_n^{k+1})}_{\text{Prediction}} + \underbrace{\mathcal{F}(t_{n+1}, t_n, \mathbf{x}_n^k) - \mathcal{G}(t_{n+1}, t_n, \mathbf{x}_n^k)}_{\text{Correction}} \end{aligned} \tag{3.2}$$

Thus parareal can be seen as a predictor-corrector algorithm where at each iteration the predicted term $\mathcal{G}(t_{n+1}, t_n, \mathbf{x}_n^{k+1})$ is corrected by the solutions from $\mathcal{F}$ and $\mathcal{G}$ at the previous iteration. The iterate update step (3.2) 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 (3.3) 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 (p-iteration) the trajectory is corrected to its exact solution for exactly one time window. After N iterations, the parareal algorithm will produce exactly the same solution 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.

Algorithm 3.1 Parareal Algorithm

{Initialisation}
for $n \leftarrow 0$ to N **do**
  $\mathbf{x}_{n+1}^0 = \mathcal{G}(t_{n+1}, t_n, \mathbf{x}_n^0)$
end for
{Iterations}
$k = 0$
repeat
  {parallel prediction step}
  for $n \leftarrow 0$ to N **do**
    $\tilde{\mathbf{x}}_{n+1}^k = \mathcal{F}(t_{n+1}, t_n, \mathbf{x}_n^k)$
  end for
  {Serial correction step}
  for $n \leftarrow 0$ to N **do**
    $\mathbf{x}_{n+1}^{k+1} = \mathcal{G}(t_{n+1}, t_n, \mathbf{x}_n^{k+1}) + \tilde{\mathbf{x}}_{n+1}^k - \mathcal{G}(t_{n+1}, t_n, \mathbf{x}_n^k)$
  end for
  $k = k + 1$
until $k = k_{\max}$

If the model (3.1) is linear, we can re-write the correction step (3.2) as

$$\begin{aligned} \mathbf{x}_{n+1}^{k+1} &= F\mathbf{x}_n^k + G\mathbf{x}_n^{k+1} - G\mathbf{x}_n^k \\ &= G\mathbf{x}_n^{k+1} + (F - G)\mathbf{x}_n^k \end{aligned} \tag{3.3}$$

where F and G are the matrix representation of the fine and coarse propagator.

There are different interpretations of parareal where it can be seen as a type of a multiple shooting method or like a time-multigrid method [12]. A thorough convergence study for the algorithm has been done by Gander and Vandewalle [12]. 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 [10].

3.1. Error analysis. We now focus on obtaining a general expression of the error for a linear system. We write the correction step (3.3)

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

Let us assume that the parareal reference solution is the solution obtained by integrating the model using the fine solver. Let $\mathbf{x}_f$ be the reference solution at the start of the time window n (i.e. $\mathbf{x}_f = F^n\mathbf{x}_0$). Then the error $\mathbf{e}_n^k = \mathbf{x}_f - \mathbf{x}_n^k$ satisfies the evolution equation

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

and thus

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

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

$$\mathbf{e}_n^k = \mathbf{x}_0 \sum_{p=k+1}^{n} C_p^n (F - G)^p G^{n-p}, \quad C_p^n = \frac{n!}{p!(n-p)!}$$

Since our cost function (2.7) involves an observation at the end of the integration time, we can accordingly define an approximate parareal based integrator over $[0, T]$ by

$$P(k)\,\mathbf{x}_0 = \mathbf{x}_N^k \tag{3.4}$$

and thus

$$F^N - P(k) = \sum_{p=k+1}^{N} C_p^N (F-G)^p G^{N-p} \tag{3.5}$$

3.2. 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 speedup ψ can be defined as the ratio of the time taken to perform a task sequentially to the time to do the same task in parallel.

$$\psi = \frac{\text{Serial computation time}}{\text{Parallel computation time}} \tag{3.6}$$

To derive the speedup for the parareal algorithm we follow along the lines of [20]. 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 F and similarly τ_g for the coarse propagator G. Thus for a single time window, $N_f\tau_f$ is the total cost of F and $N_g\tau_g$ for 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 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 p-iterations,

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

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

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

As stated in [26] we get different bounds for the speedup and they compete with each other.

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

The coarse propagator should be really cheap and fast in order to get a good speedup but at the same time there will be more p-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. For a more rigorous explanation about the speedup and efficiency of the parareal see [2].

3.3. Coupling. We now replace the parallel-in-time operator P in (2.7) by the parareal algorithm. In this case the sequential model trajectories are obtained by the reference solution which is nothing but the solution from the fine propagator F. We now look at the cost function (2.6)

$$J(\mathbf{x}_0) = \frac{1}{2}\|F^N\mathbf{x}_0 - \mathbf{y}\|_2^2$$

and its gradient

$$\nabla J(\mathbf{x}_0) = (F^N)^T(F^N\mathbf{x}_0 - \mathbf{y})$$

Again instead of using the reference solution for the forward integration, if we use the parareal based integrator $P(k)$, (2.6) becomes

$$J(\mathbf{x}_0) \approx \frac{1}{2}\|P(k)\mathbf{x}_0 - \mathbf{y}\|_2^2 \tag{3.7}$$

In this respect, the gradient can be written as

$$\nabla J(\mathbf{x}_0) \approx (F^N)^T(P(k)\mathbf{x}_0 - \mathbf{y})$$

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

REMARK 3.1. *Since the fine propagator is replaced by the parareal algorithm in (3.7), we will denote the exact matrix by A_* and the inexact (approximate) matrix by A. Thus $A_* = (F^N)^T F^N$ and $A = (F^N)^T P(k)$.*

4. Minimisation. For solving large linear systems, choosing a suitable Krylov subspace method is a logical option [14]. Krylov methods are iterative in nature with the matrix-vector multiplication as their most dominant operation. In some situations, there might be the case when 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 [27, 30]. By doing so, the same level of accuracy can be maintained as expected by using the usual Krylov subspace methods.

In our case we want to minimise a quadratic cost function of the data assimilation problem which leads to solving a symmetric system $A\mathbf{x} = \mathbf{b}$. As seen before if parareal is used for the forward integration then $A = (F^N)^T P(k)$ and $\mathbf{b} = (F^N)^T\mathbf{y}$. $P(k)$ is an operator which acts on the initial condition to give the parareal solution at the end of the integration time. Thus, 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 inexact conjugate gradient method which is effective for linear symmetric systems.

4.1. Inexact conjugate gradient. We can rewrite the 4D-Var cost function in terms of A_* and $\mathbf{b}$ leading to the following convex quadratic minimisation problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A_*\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

with symmetric positive definite matrix $A_* \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$
As mentioned before, this problem can also be seen as solving the symmetric positive-definite linear system $A_*\mathbf{x} = \mathbf{b}$ 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. [13] 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 $\mathbf{r}(\mathbf{x}) = \mathbf{b} - A_*\mathbf{x}$ is defined to be the residual of the system then this relationship can be described as [13]

$$\frac{1}{2}\|\mathbf{r}(\mathbf{x})\|^2_{A_*^{-1}} = J(\mathbf{x}) - J(\mathbf{x}_*) \tag{4.1}$$

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

We will adopt this idea to deal with the inexactness introduced by the use of a parareal method. Let us represent the source of inexactness in the matrix A by the perturbation matrix E. In our context the error manifests from the parareal method in equation (3.5). By construction, the E matrix at any

parareal iteration k can be written as

$$\begin{aligned} E(k) &= A - A_* \\ &= (F^N)^T P(k) - (F^N)^T F^N \\ &= -(F^N)^T (F^N - P(k)) \end{aligned}$$

Using (3.5) we get,

$$E(k) = -(F^N)^T \sum_{p=k+1}^{N} C_p^N (F-G)^p G^{N-p} \tag{4.2}$$

From (4.1) it is clear that a sufficiently accurate residual $\mathbf{r}(\mathbf{x})$ and its A_*^{-1} norm is required to monitor $\|\mathbf{r}(\mathbf{x})\|_{A_*^{-1}}$. But the presence of errors at each inexact CG iteration (icg-iteration) j also results in the computed residual $\mathbf{r}_j$ being different from the exact residual $\mathbf{r}(\mathbf{x}_j) = \mathbf{b} - A_*\mathbf{x}_j$. In order to still keep track of the quadratic change, it is necessary to appropriately bound the inexact residual norm $\|\mathbf{r}_j\|_{A^{-1}}$ and the residual gap norm $\|\mathbf{r}(\mathbf{x}_j) - \mathbf{r}_j\|_{A^{-1}}$. We are also going to need the primal-dual norm of the perturbation matrix defined as

$$\|E_j\|_{A^{-1},A} = \sup_{\mathbf{x}\neq 0} \frac{\|E_j\mathbf{x}\|_{A^{-1}}}{\|\mathbf{x}\|_A} = \|A^{-1/2} E_j A^{-1/2}\|_2$$

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 icg-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 [13], where $\mathbf{p}$ is taken as the conjugate direction vector.

THEOREM 4.1. *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

$$\|E_i\|_{A^{-1},A} \leq \omega_i = \frac{\sqrt{\epsilon}\|\mathbf{b}\|_{A^{-1}}\|\mathbf{p}_i\|_A}{2\phi_{i+1}\|\mathbf{r}_i\|_2^2 + \sqrt{\epsilon}\|\mathbf{b}\|_{A^{-1}}\|\mathbf{p}_i\|_A} \tag{4.3}$$

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

$$\|\mathbf{r}(\mathbf{x}_j) - \mathbf{r}_j\|_{A^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|\mathbf{b}\|_{A^{-1}} \tag{4.4}$$

Additionally if

$$\|\mathbf{r}_j\|_{A^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|\mathbf{b}\|_{A^{-1}} \tag{4.5}$$

then $|J(\mathbf{x}_j) - J(\mathbf{x}_*)| \leq \epsilon |J(\mathbf{x}_*)|$.

REMARK 4.2. *The errors are permitted to grow as we proceed with the icg-iterations. As a consequence the search directions are no longer conjugate to each other and the (inexact) residuals lose their orthogonality. In the theorem it is assumed that the inexact residual norm $\|\mathbf{r}_j\|_{A^{-1}}$ 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 4.3. *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 (4.5) and the number of inner iterations on ω_j in equation (4.3).*

4.2. Stopping criterion. In Theorem 4.1, the quantity ω_j acts as a threshold for how large a perturbation can be allowed at a particular icg-iteration j. In the beginning one starts with a small value of ω_j with more and more error allowed (and thus gradual increase in ω_j) in the subsequent iterations. When $\kappa(A)$ is large, the primal-dual norm of the perturbation matrix $\|E_j\|_{A^{-1},A}$ is quite large and it usually goes down below 1 only at the last few p-iterations. In terms of efficiency the number of p-iterations being used in criterion (4.3) does not reflect upon the actual number of p-iterations needed. We make a few modifications in the original method to fix this issue. In the proof of Theorem 4.1, the bound (4.4) is obtained using the inequality

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

We noticed that utilising $\|E_j\mathbf{p}_j\|_{A^{-1}}$ instead of $\|E_j\|_{A^{-1},A}$ for the parareal stopping criterion gives us fairly reasonable results. It can be clearly seen that this in no way affect the original method. To bound our new norm we introduce a quantity ξ_j which satisfies,

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

Thus from (4.3),

$$\xi_j = \frac{\sqrt{\epsilon}\|\mathbf{b}\|_{A^{-1}}\|\mathbf{p}_j\|_A^2}{2\phi_{j+1}\|\mathbf{r}_j\|_2^2 + \sqrt{\epsilon}\|\mathbf{b}\|_{A^{-1}}\|\mathbf{p}_j\|_A} \tag{4.6}$$

Note: $\|E_j\mathbf{p}_j\|_{A^{-1}}$ 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 4.4. *The modified criterion does not make use of $\|E_j\|_{A^{-1},A}$ 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 $P(k)\mathbf{p}_j - A\mathbf{p}_j$ is nothing but $E_j(k)\mathbf{p}_j$.*

Figure 4.1 below shows the number of p-iterations for a shallow water model if we use the original criterion and the modified criterion for a large $\kappa(A)$. 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.

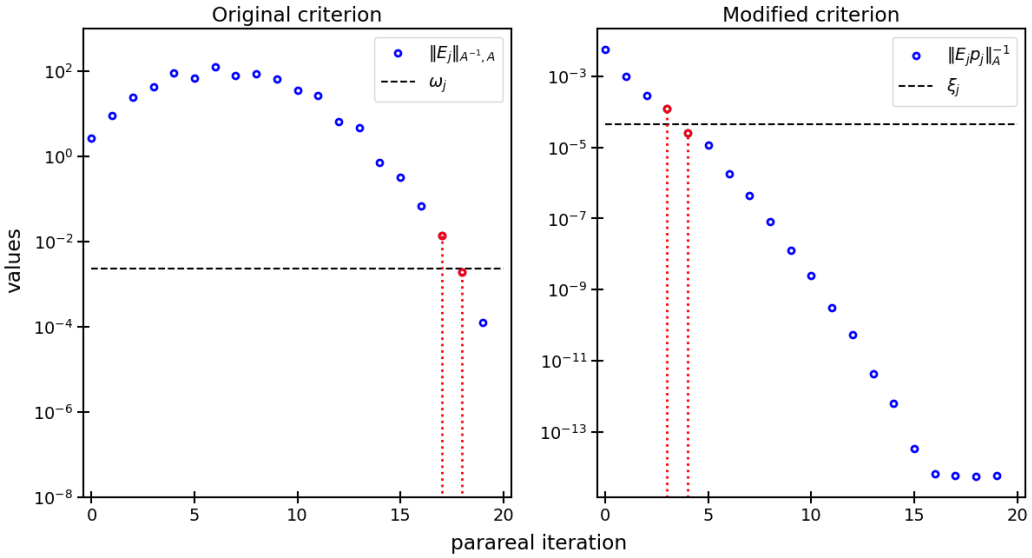


FIG. 4.1. *Estimates and bounds for icg-iteration 6, $N = 20, \kappa(A) = 1099070.55$*

From the figure on the left the curve resembles to the kind of parareal convergence results as reported in [8] 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 speedup. The error starts increasing and only comes down at the last iteration.

The relative decrease in the curve of $\|E_j\mathbf{p}_j\|_{A^{-1}}$ can be explained by the fact that even if there is an increase in $\|E_j\|_{A^{-1},A}$, theoretically the conjugate direction norm $\|\mathbf{p}_j\|_A \to 0$ as $j \to j_{\max}$.

Till now there has been no talk about the role of ϕ_j which is discussed in the original method [13, 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 icg-iterations. The corresponding $\hat{\phi}_{j+1}$ can be obtained from $\hat{\xi}_j = \xi_j(\hat{\phi}_{j+1})$ in (4.6) as

$$\hat{\phi}_{j+1} = \frac{(\|\mathbf{p}_j\|_A - \hat{\xi}_j)}{\hat{\xi}_j} \frac{\sqrt{\epsilon}\|\mathbf{b}\|_{A^{-1}}\|\mathbf{p}_j\|_A}{2\|\mathbf{r}_j\|_2^2} > \phi_{j+1} \tag{4.7}$$

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

4.3. Modified inexact 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 A and its inverse. But realistically one might not even have access to the matrix A, let alone its inverse. Thus in practice we will use the following approximations [13] of all the quantities which are unfeasible to compute.

- $\|\mathbf{p}_j\|_A \approx \sqrt{\frac{1}{n}\mathrm{Tr}(A)}\,\|\mathbf{p}_j\|_2$
- $J(\mathbf{x}_j) \approx J_j \overset{\text{def}}{=} -\frac{1}{2}\mathbf{b}^T\mathbf{x}_j$
- $\|\mathbf{b}\|_{A^{-1}} \approx \frac{\|\mathbf{b}\|_2}{\sqrt{\lambda_{\max}}} \quad j = 0,\ \mathbf{x}_0 = 0 \quad \text{and} \quad \|\mathbf{b}\|_{A^{-1}} \approx \sqrt{2|J_j|} \quad j = 1, 2, ..., j_{\max}$
- Termination test (4.5) by $J_{j-d} - J_j \leq \frac{1}{4}\epsilon|q_j|$ for some integer d

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

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

THEOREM 4.5. *If F is invertible, $\|E_j(k)\mathbf{p}_j\|_{A^{-1}}$ and $\|P(k)\mathbf{p}_j - \mathbf{p}_*\|_2$ are rigorously the same whatever the parareal approximation is. $P(k)\mathbf{p}_j$ 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,

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

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

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

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 4.2 we notice that the curve profile of $\|E_j\mathbf{p}_j\|_{A^{-1}}$ does not change much for given j. Then replacing $\|P(k)\mathbf{p}_j - \mathbf{p}_*\|_2$ by $\|P(k)\mathbf{p}_j - \mathbf{p}_{**}\|_2$ 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.

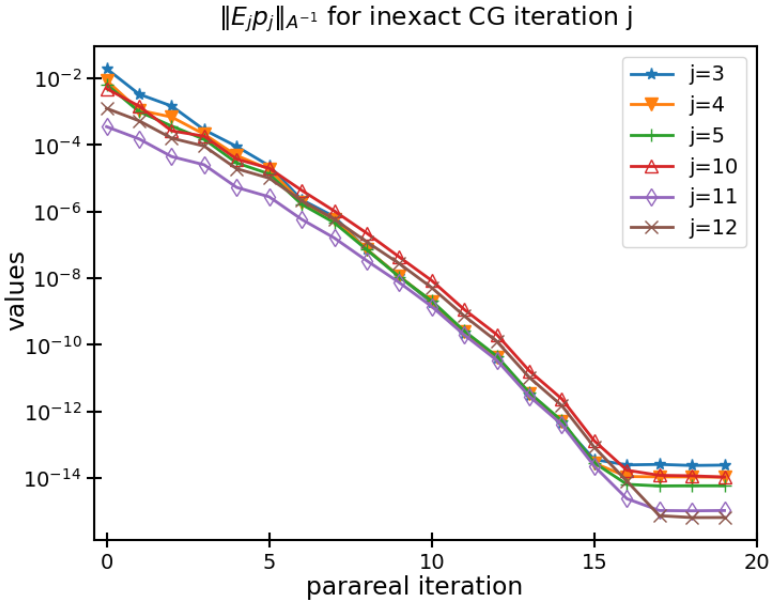


FIG. 4.2. *Profile of* $\|E_j\mathbf{p}_j\|_{A^{-1}}$ *for different CG iterations*

For a clear explanation, suppose for instance that the initial (accurate) parareal solution $P(k)\mathbf{p}_0$ takes $\bar{k}$ iterations. Then for the next icg-iteration, $\|P(k)\mathbf{p}_1 - \mathbf{p}_*\|_2$ is approximated by $\|P(k)\mathbf{p}_0 - \mathbf{p}_{**}\|_2$ where $\mathbf{p}_{**} \approx P(\bar{k})\mathbf{p}_0$. We then find the required k which satisfies $\|P(k)\mathbf{p}_0 - \mathbf{p}_{**}\|_2 \leq \xi_1$, say $k = k_1$. Now the parareal solution $P(k_1)\mathbf{p}_1$ will be used to compute the matrix-vector multiplication . Going to the next iteration, $\|P(k)\mathbf{p}_2 - \mathbf{p}_*\|_2$ will be approximated by $\|P(k)\mathbf{p}_1 - \mathbf{p}_{**}\|_2$ with $\mathbf{p}_{**} \approx P(k_1)\mathbf{p}_1$. 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 $\|P(k)\mathbf{p}_j - \mathbf{p}_{**}\|_2$ 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 $\|P(k_1)\mathbf{p}_0 - \mathbf{p}_{**}\|_2 \leq \xi_1$ but $\xi_1 \in [\bar{k}-1, \bar{k})$. Like before we run the parareal for k_1 iterations to get $P(k_1)\mathbf{p}_1$. What changes now is that we look for k such that $\|P(k)\mathbf{p}_1 - P(k_1)\mathbf{p}_1\|_2 \leq \xi_1$. 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 $k_1 := k_1 + k_*$. This discussion has been put in the form of an algorithm for $\mathbf{p}_*$ approximation which can be implemented.

This test is illustrated in Figure 4.3 for icg-iteration 17. We can see that when for the last parareal

Algorithm 4.1 p_* approximation test

Given: icg-iteration j
Find ξ_j from equation (4.6) {Last parareal test}
if $j = 0$ **then**
  Compute the parareal solution $P(k)\mathbf{p}_0$ with $\epsilon_{\text{p}} = \epsilon_{\text{cg}}/10$. Let the total iterations taken be k_0
  Set $k_{\text{sol}} = k_0$, $\mathbf{p}_{**} = P(k)\mathbf{p}_0(k_{\text{sol}})$
else
  Find $0 \leq k < k_{\text{sol}}$ such that $\|P(k)\mathbf{p}_{j-1}(k) - \mathbf{p}_{**}\|_2 \leq \xi_j$; save LHS as $\hat{\xi}_j$
  $k_{\text{sol}} = k$
  if $k = k_{\text{sol}} - 1$ **then**
    cp = True {Use current parareal iterate}
  end if
end if
Compute the current parareal solution p_j for k_{sol} iterations
Set $\mathbf{p}_{**} = P(k)\mathbf{p}_j(k_{\text{sol}})$
if cp = True **then**
  Find $0 \leq k < k_{\text{sol}}$ such that $\|P(k)\mathbf{p}_j - P(k)\mathbf{p}_j(k_{\text{sol}})\|_2 \leq \xi_j$; save LHS as $\hat{\xi}_j$
  if $k = k_{\text{sol}} - 1$ **then**
    Perform one more parareal iteration and set $k_{\text{sol}} = k_{\text{sol}} + 1$
  end if
  Repeat steps in lines 15-17 till $k \neq k_{\text{sol}} - 1$
  $k_{\text{sol}} = k$
end if
Calculate the inexact matrix-vector multiplication $\mathbf{c_j} = (F^N)^T\mathbf{p}_j(k_{\text{sol}})$

Algorithm 4.2 Modified inexact conjugate gradient

Given: symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, $\epsilon > 0$, reorth, $\mathbf{x}_0 = 0, r_0 = -\mathbf{b}, \mathbf{p}_0 = \mathbf{b}, \beta_0 = \|\mathbf{b}\|_2^2, \mathbf{u}_1 = \mathbf{b}/\beta_0, \phi_0 = j_{\max}$ 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 (4.7)
  $\Phi_{j+1} = \Phi_j - \hat{\phi}_j^{-1}$
  if $j < j_{\max}$ **then**
    $\phi_{j+1} = (j_{\max} - j - 1)/\Phi_{j+1}$
  else
    $\phi_{j+1} = \phi_j$
  end if
  $\alpha_j = \beta_j/\mathbf{p}_j^T\mathbf{c}_j$
  $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$
  $\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j\mathbf{c}_j$
  if $(J_{j+1-d} - J_{j+1}) \leq \frac{1}{4}\epsilon|J_{j+1}|$ **then**
    break
  end if
  if (reorth) **then**
    for $i = 1, 2, .., j$ **do**
      $\mathbf{r}_{j+1} = \mathbf{r}_{j+1} - (\mathbf{u}_i^T\mathbf{r}_{j+1})\mathbf{u}_i$
    end for
    $\beta_{j+1} = \mathbf{r}_{j+1}^T\mathbf{r}_{j+1}$
    $\mathbf{u}_{j+1} = \mathbf{r}_{j+1}/\sqrt{\beta_{j+1}}$
  else
    $\beta_{j+1} = \mathbf{r}_{j+1}^T\mathbf{r}_{j+1}$
  end if
  $\mathbf{p}_{j+1} = -\mathbf{r}_{j+1} + (\beta_{j+1}/\beta_j)\mathbf{p}_j$
end while

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.

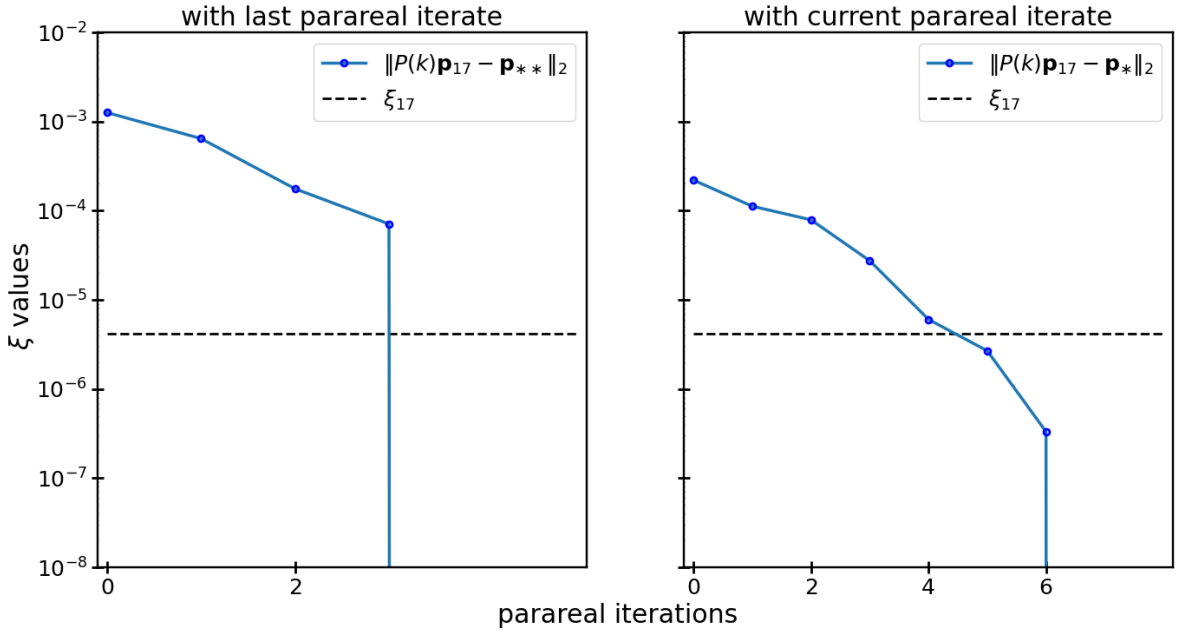


FIG. 4.3. *Using last and current parareal iterate*

5. Numerical experiments. One of the main objectives of coupling two inherently sequential methods is to use it for the applications related to ocean or climate modelling. In general, such problems are solved at a low resolution by integrating the model for a very long period of time for example 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.

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= -h\frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial t} &= -g\frac{\partial \eta}{\partial x} \end{aligned} \tag{5.1}$$

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 [5]. The modified equations are given as

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= -h\frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial t} &= -g\frac{\partial \eta}{\partial x} + \mu\frac{\partial^2 u}{\partial x^2} \end{aligned} \tag{5.2}$$

where μ is the diffusion constant. The numerical discretisiation is done using Arakawa C-grid or a staggered grid.

$$\begin{aligned} \frac{\eta_{i+1/2}^{k+1} - \eta_{i+1/2}^{k}}{\Delta t} &= -h\,\frac{u_{i+1}^{k} - u_{i}^{k}}{\Delta x} \\ \frac{u_{i+1}^{k+1} - u_{i+1}^{k}}{\Delta t} &= g\frac{\eta_{i+1/2}^{k} - \eta_{i-1/2}^{k}}{\Delta x} + \frac{u_{i+1}^{k} - 2u_{i}^{k} + u_{i-1}^{k}}{\Delta x^2} \end{aligned} \tag{5.3}$$

The above discretisation can be expressed as an ODE system

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{dt}} = C\mathbf{x} \tag{5.4}$$

where

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

and $\mathbf{x} = (\eta_0, \eta_1, \dots \eta_{59}, u_0, u_1, \dots u_{59})^T$. We take $h = 0.9, g = 10$ and the characteristic length $L = 120$. If the number of spatial grid points are 120, then $\Delta x = L/120 = 1$. To solve the system (5.4) we use a weighted θ-scheme

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

which we can rewrite as

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

We can now define the propagators for implementing the parareal algorithm. For each time window, we associate N_f and N_g as the number of fine time steps and coarse time steps respectively. Clearly N_g has to divide N_f. If δt is the fine step length, then the coarse step length Δt is given as

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

while the total integration time is $T = \delta t \times N_f \times N$. Here, $N = 20$, $N_f = 100$ and $N_g = 25$. The fine and coarse propagators are defined as

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

With our model in hand, we solve the following data assimilation problem

$$\min\ J(\mathbf{x}_0) = \frac{1}{2}\|F^N \mathbf{x}_0 - \mathbf{y}\|_2^2 + \frac{\alpha}{2}\left\|\frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}\mathbf{x}}\right\|_2^2$$

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 $\mathbf{x}_0 = \mathrm{e}^{[(x-L/2)/(L/15)]^2}$

5.1. Analysis. First we check how our data assimilation problem performs with our shallow water model when we run the exact CG method with exact matrix-vector multiplication. The stopping criterion depends on the 2-norm of the residual. For a tolerance ϵ_{cg} of 10^{-4}, it takes 24 minimisation iterations.

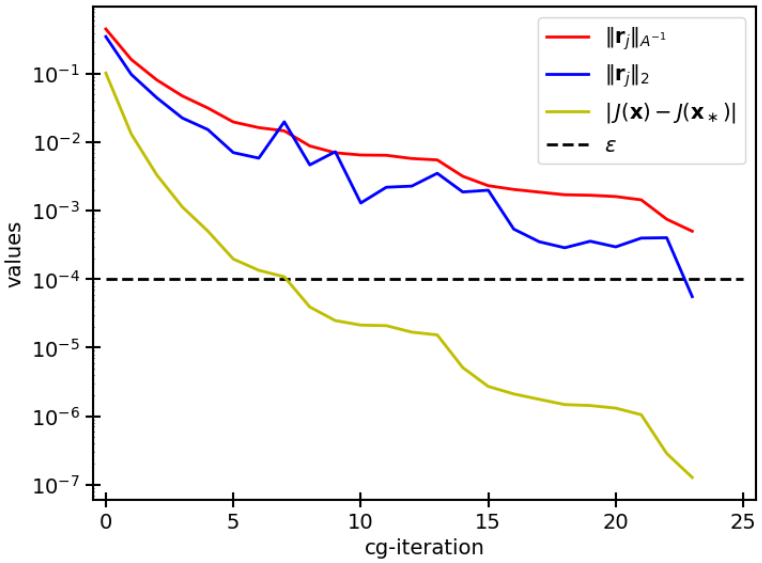


FIG. 5.1. *Exact CG with exact matrix-vector product*

From the above exact CG solution with $\epsilon_{\text{cg}} = 10^{-4}$ in Figure 5.1, we find the value of $\|\mathbf{r}_k\|_{A^{-1}}$ and use it in (4.5) to calculate a suitable stopping criteria for the inexact CG. We find that $\epsilon_{\text{icg}} = 1.12 \times 10^{-7}$. As talked earlier the residuals generated are no longer exact and for that purpose we have added a reorthogonlisation step at each icg-iteration.

We first show the result when we use all theoretical estimates as well as the norm $\|E_j\|_{A^{-1},A}$ and why it is not a good idea to keep the primal-dual norm as the stooping criteria. While we are still able to converge after 24 minimisation iterations, but only at the expense of 464 p-iterations. This makes it an average of 18.5 p-iterations per cg-iteration which is not feasible at all. Replacing $\|E_j\|_{A^{-1},A}$ with $\|E_j\mathbf{p}_j\|_{A^{-1},A}$ as the stopping criterion, it takes 25 minimisation iterations but the total p-iterations goes down to 165 and we have an average of 6.6 p-iterations per cg-iteration.

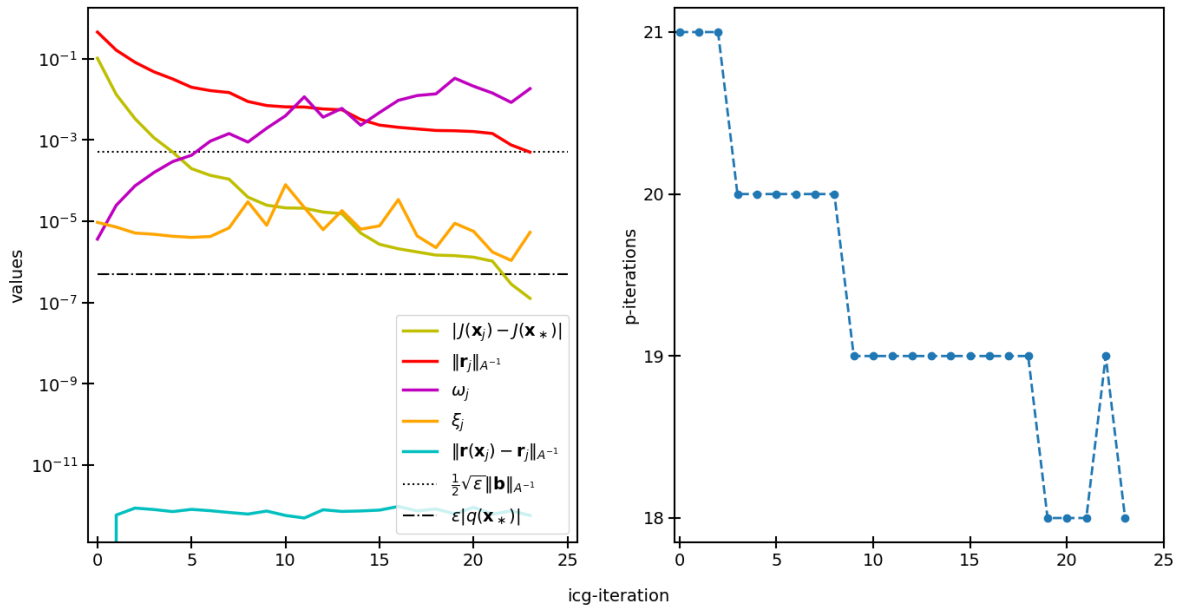


FIG. 5.2. *Theoretical inexact CG with* $\|E_j\|_{A^{-1},A}$ *as the stopping criteria*

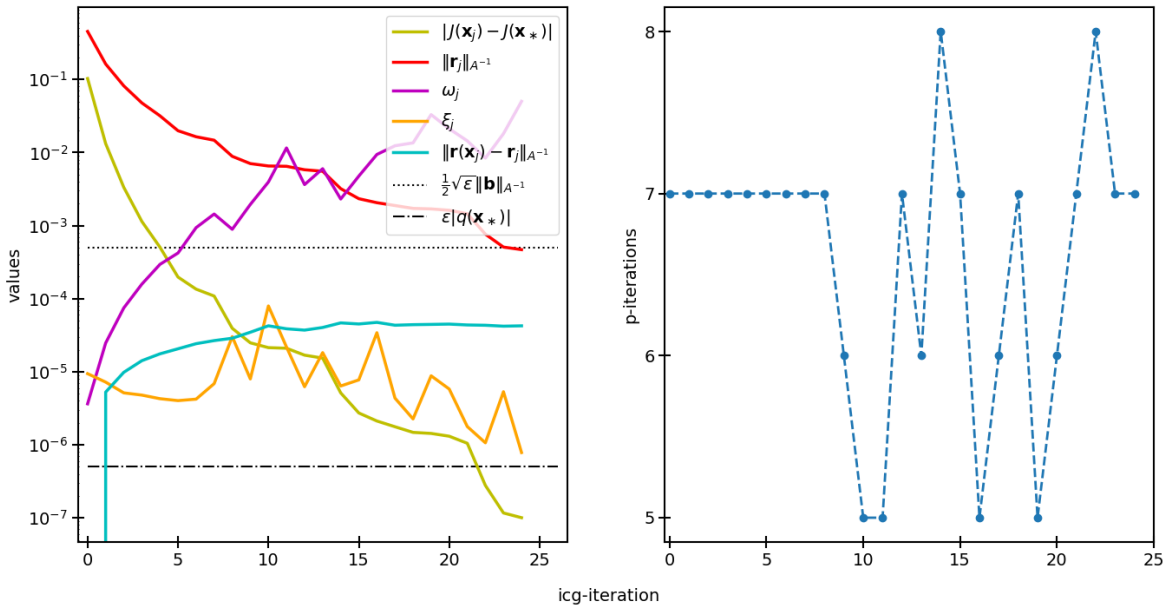


FIG. 5.3. *Theoretical inexact CG with* $\|E_j p_j\|_{A^{-1}}$ *as the stopping criteria*

With the approximations of the quantities from Section 4.3 and the $\|E_j \mathbf{p}_j\|_{A^{-1}}$, the practical inexact CG performs equally well. The integer value taken for the stopping criterion is $d = 2$. The minimisation ends in 26 iterations with a slightly lower average of 6.2 p-iterations per cg-iteration.

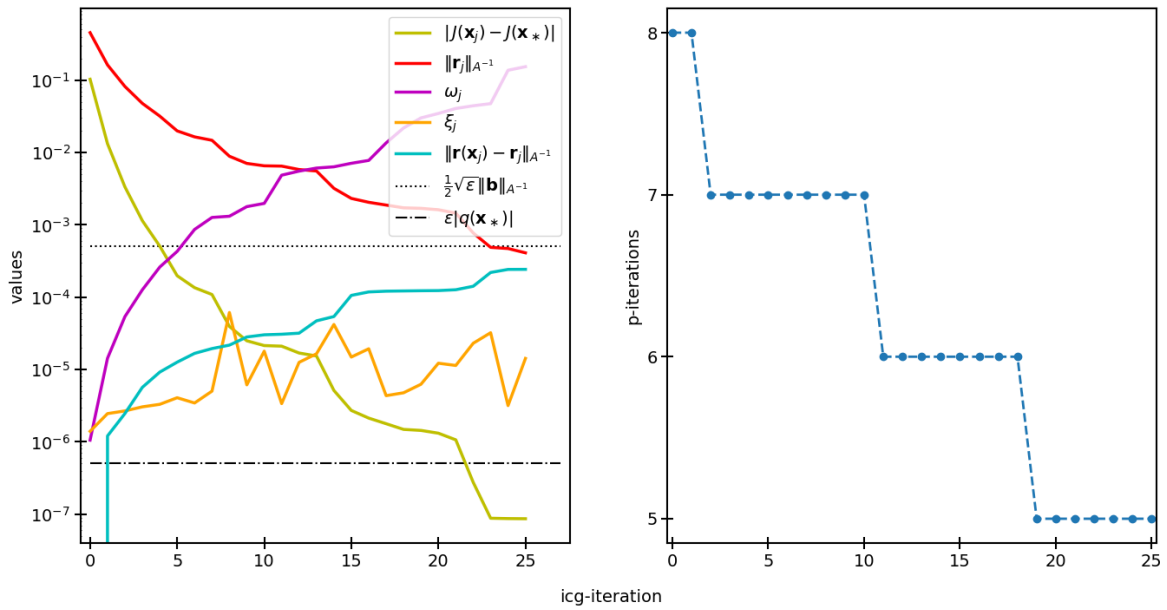


FIG. 5.4. *Practical inexact CG with* $\|E_j p_j\|_{A^{-1}}$ *as the stopping criteria*

Finally, we show the main result by combining the practical estimates and the approximation for the modified stopping criterion. We choose $d = 10$. Still, the method gives almost identical results as seen in the last two cases with the minimisation and p-iterations being 25 and 159 respectively. The average p-iterations per CG iterations remains at 6.36. The p-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.

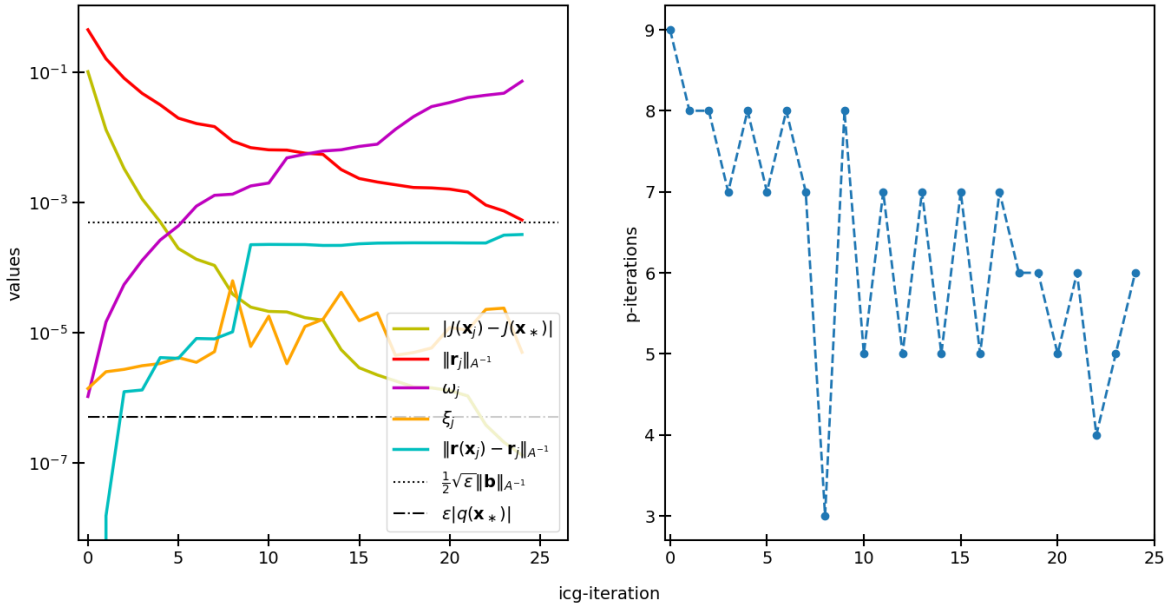


FIG. 5.5. *Modified practical inexact CG*

Looking at all the previous results, we observe that if we take the minimum of all the values of ξ_j which turns out to be approximately 10^{-6}, running the exact CG with now the matrix-vector multiplication coming through the parareal method, it should work and it does. For $\epsilon_{\mathrm{p}} = 10^{-6}$ as a fixed parareal stopping criterion, we end up using 193 p-iterations. However on average we use 7.72 p-iterations per cg-iteration which is slightly higher due to a fixed parareal tolerance.

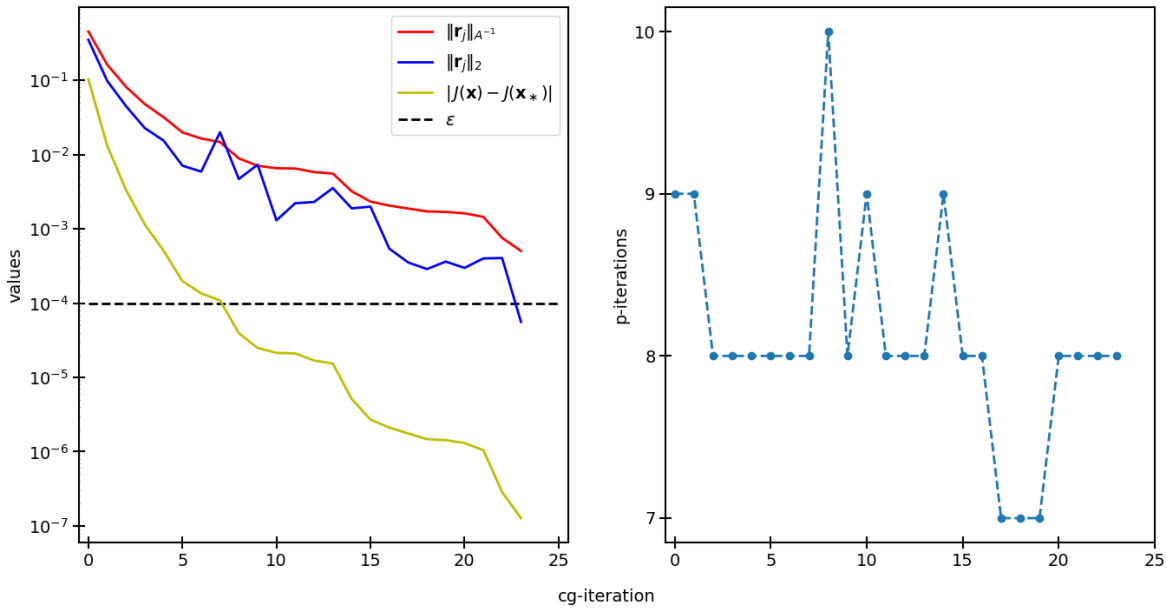


FIG. 5.6. *Exact CG with matrix-vector product from parareal*

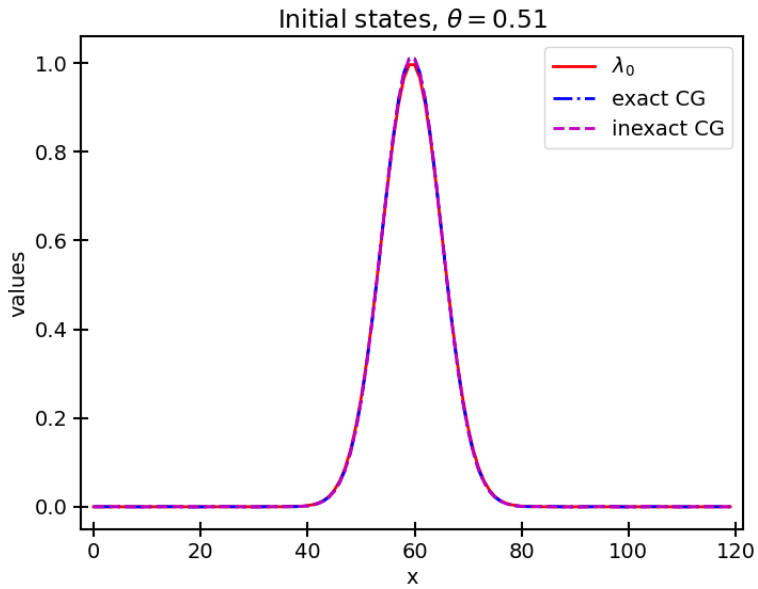
Some of the obtained parameter values

- courant number: 0.55
- diffusion courant number: 0.0075

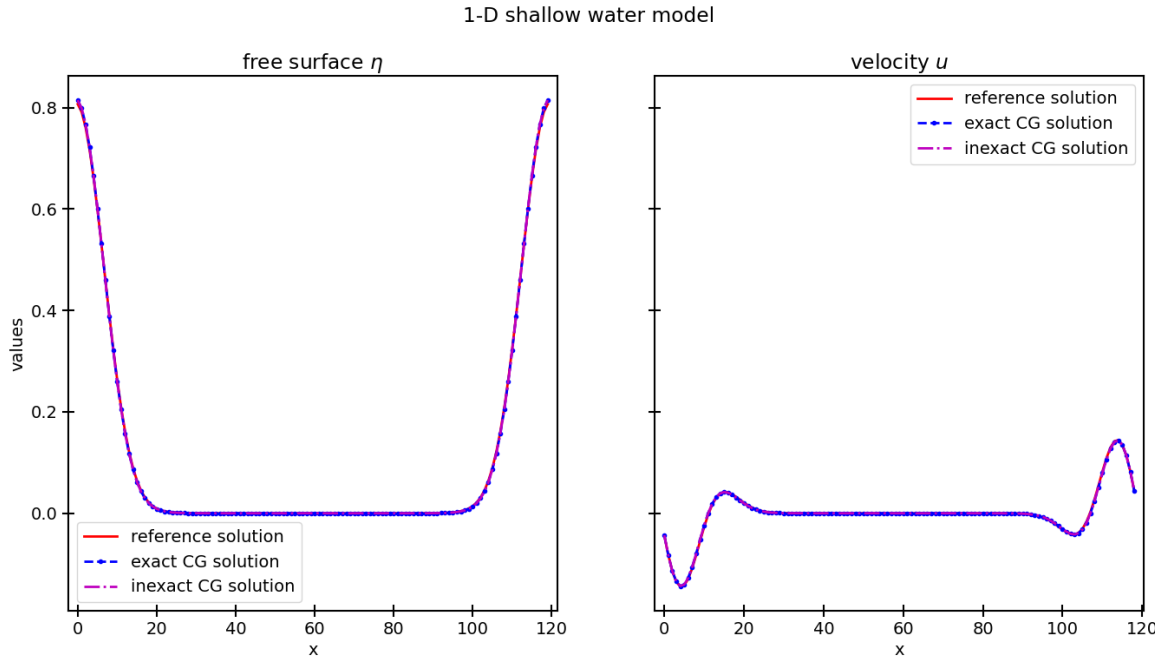
- expected speedup:

$$\psi = \frac{\text{number of time windows}}{\text{average p-iterations per cg-iteration}} = \frac{20}{6.36} = 3.145$$

The initial condition retrieved by 4D-Var by using exact and inexact CG is almost identical to the initial state $\mathbf{x}_0$



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-in-time method (parareal) with 4D-Var we see that if we use the proposed modified inexact CG method one would expect the optimum use of p-iterations. Even though the p-iterations do not decrease monotonically which could be accounted for by the values of ξ_j, we still get an expected speedup of 3. We can at least conclude that the inexact CG by Gratton et al. [13] provides a well defined stopping criterion for the parareal algorithm at each minimisation iteration. This way we don't have to run parareal with a very fine precision each

time but only till where the bound is satisfied.

As a remark, it is a well known fact that the parareal algorithm shows instabilities when it is applied to hyperbolic problems and second order ODEs [25, 28]. 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 [8]. This parareal variant is known as the Krylov enhanced parareal method. As a side experiment, we ran our modified inexact CG method with the Krylov enhanced version and for 25 minimisation iterations the average p-iterations per cg-iteration almost get halved (3.24).

Clearly our task becomes much simpler. However, at each parareal iteration one has to store all the fine evaluations for constructing the subspace leading to computational inefficiency. Here it works well because the problem is a simple 1-D case but it would be really difficult to implement for higher dimensional problems. Choosing appropriate basis vectors which contribute to the subspace could be one of the solutions which can be seen as a topic of future study.

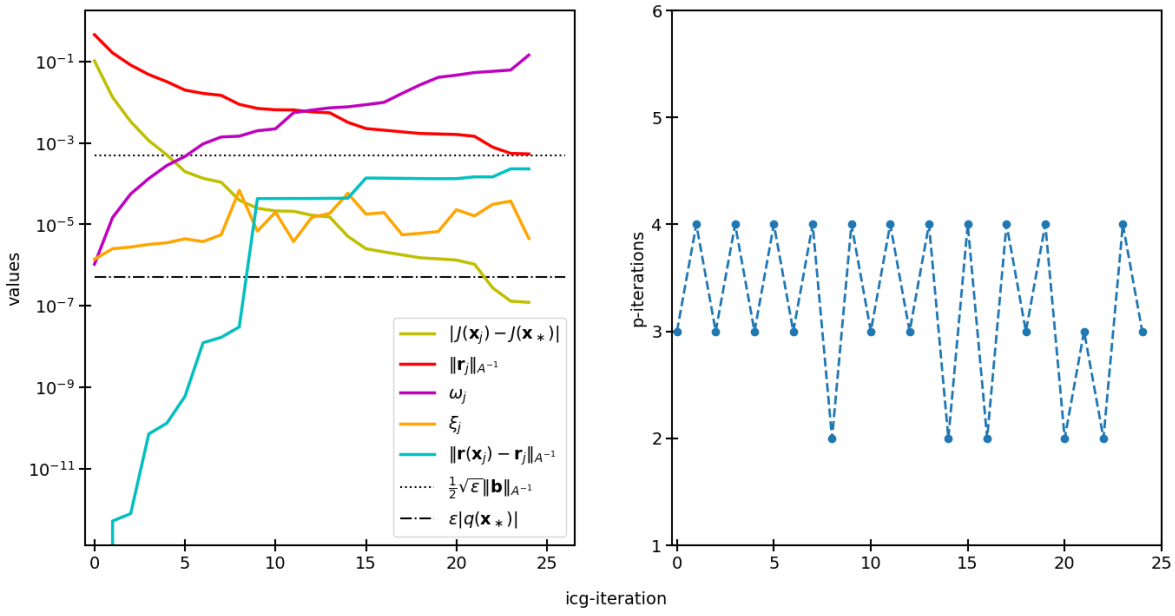


FIG. 6.1. *Modified practical inexact CG with krylov enhanced parareal*

One thing to note is that the adjoint in the cost function (2.7) 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 could be applied in exactly similar manner and can double the degree of parallelism in our algorithm. 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 [6] or the ParaOpt algorithm [11] where the forward and backward model are solved together as a single optimality system.

We are currently investigating the idea of introducing asynchronous parallel-in-time algorithms which makes the processors independent of one another and therefore we can cut the communication time and obtain further speedup. The idea would be to make these p-iterations asynchronous [18, 19] both for the forward and the backward model integration.

REFERENCES

[1] L. BAFFICO, S. BERNARD, Y. MADAY, G. TURINICI, AND G. ZÉRAH, *Parallel-in-time molecular-dynamics simulations*, Physical Review E, 66 (2002), p. 057701.
[2] G. BAL, *Parallelization in time of (stochastic) ordinary differential equations*, Math. Meth. Anal. Num.(submitted), (2003).
[3] F. BOUTTIER AND P. COURTIER, *Data assimilation concepts and methods march 1999*, Meteorological training course lecture series. ECMWF, 718 (2002), p. 59.
[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, 120 (1994), pp. 1367–1387.
[5] B. CUSHMAN-ROISIN AND J.-M. BECKERS, *Introduction to geophysical fluid dynamics: physical and numerical aspects*, Academic press, 2011.

[6] M. Emmett and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Communications in Applied Mathematics and Computational Science, 7 (2012), pp. 105–132.
[7] M. Fisher and S. Gürol, *Parallelization in the time dimension of four-dimensional variational data assimilation*, Quarterly Journal of the Royal Meteorological Society, 143 (2017), pp. 1136–1147.
[8] M. Gander and M. Petcu, *Analysis of a krylov subspace enhanced parareal algorithm for linear problems*, in ESAIM: Proceedings, vol. 25, EDP Sciences, 2008, pp. 114–129.
[9] M. J. Gander, *50 years of time parallel time integration*, in Multiple shooting and time domain decomposition methods, Springer, 2015, pp. 69–113.
[10] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the parareal algorithm*, in Domain decomposition methods in science and engineering XVII, Springer, 2008, pp. 45–56.
[11] M. J. Gander, F. Kwok, and J. Salomon, *Paraopt: A parareal algorithm for optimality systems*, SIAM Journal on Scientific Computing, 42 (2020), pp. A2773–A2802.
[12] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM Journal on Scientific Computing, 29 (2007), pp. 556–578.
[13] S. Gratton, E. Simon, D. Titley-Peloquin, and P. L. Toint, *Minimizing convex quadratics with variable precision conjugate gradients*, Numerical Linear Algebra with Applications, 28 (2021), p. e2337.
[14] I. C. Ipsen and C. D. Meyer, *The idea behind krylov methods*, The American mathematical monthly, 105 (1998), pp. 889–899.
[15] 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, 131 (2005), pp. 459–476.
[16] F.-X. Le Dimet and O. Talagrand, *Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects*, Tellus A: Dynamic Meteorology and Oceanography, 38 (1986), pp. 97–110.
[17] 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, 332 (2001), pp. 661–668.
[18] F. Magoules and G. Gbikpi-Benissan, *Asynchronous parareal time discretization for partial differential equations*, SIAM Journal on Scientific Computing, 40 (2018), pp. C704–C725.
[19] F. Magoulès, G. Gbikpi-Benissan, and Q. Zou, *Asynchronous iterations of parareal algorithm for option pricing models*, Mathematics, 6 (2018), p. 45.
[20] M. Minion, *A hybrid parareal spectral deferred corrections method*, Communications in Applied Mathematics and Computational Science, 5 (2011), pp. 265–301.
[21] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Communications of the ACM, 7 (1964), pp. 731–733.
[22] B. W. Ong and J. B. Schroder, *Applications of time parallelization*, Computing and Visualization in Science, 23 (2020), pp. 1–15.
[23] J. Rantakokko, *Strategies for parallel variational data assimilation*, Parallel Computing, 23 (1997), pp. 2017–2039.
[24] V. Rao and A. Sandu, *A time-parallel approach to strong-constraint four-dimensional variational data assimilation*, Journal of Computational Physics, 313 (2016), pp. 583–593.
[25] D. Ruprecht, *Wave propagation characteristics of parareal*, Computing and Visualization in Science, 19 (2018), pp. 1–17.
[26] D. Ruprecht and R. Krause, *Explicit parallel-in-time integration of a linear acoustic-advection system*, Computers & Fluids, 59 (2012), pp. 72–83.
[27] V. Simoncini and D. B. Szyld, *Theory of inexact krylov subspace methods and applications to scientific computing*, SIAM Journal on Scientific Computing, 25 (2003), pp. 454–477.
[28] G. A. Staff and E. M. Rønquist, *Stability of the parareal algorithm*, in Domain decomposition methods in science and engineering, Springer, 2005, pp. 449–456.
[29] Y. Trémolet and F.-X. Le Dimet, *Parallel algorithms for variational data assimilation and coupling models*, Parallel Computing, 22 (1996), pp. 657–674.
[30] J. Van Den Eshof and G. L. Sleijpen, *Inexact krylov subspace methods for linear systems*, SIAM Journal on Matrix Analysis and Applications, 26 (2004), pp. 125–153.