Long Window 4D-Var

Mike Fisher[*], **Harri Auvinen**[†]

[*]*ECMWF, Shinfield Park, Reading*
RG2 9AX, United Kingdom
mike.fisher@ecmwf.int
[†]*Lappeenranta University of Technology, Finland*

1 Introduction

The potential for improving the quality of 4D-Var analyses by extending the analysis window was demonstrated by Fisher *et al.* (2005). They showed that analyses equal in quality to those of the extended Kalman filter could be achieved, at least for perfect-model analyses for the Lorenz (1995) system, if the analysis window was sufficiently long (of order 10 days).

Fisher *et al.* (*op cit.*) used a data-denial experiment to estimate the length of analysis window that would be required to achieve equivalence with the extended Kalman filter in an analysis system for numerical weather prediction (NWP). They estimated that a window of approximately seven days would be needed.

It is likely that the length of window required to achieve 4D-Var analyses equal in quality to those of the extended Kalman filter is strongly dependent on the observing system. Arguably, Fisher *et al*'s estimate of seven days applies to a system which does not assimilate satellite radiance data. In section 2, we present results from a data reinstatement experiment that suggest that a window of just three days may be sufficient.

The Lorenz (1995) model used by Fisher *et al.* (*op cit.*) is extremely simple. Although it provides an interesting analogue for mid-latitude atmospheric dynamics, it lacks many of the characteristics of a full, three-dimensional system. Moreover, the experiments conducted by Fisher *et al.* used a perfect model (that is, the assimilating model was identical to the model that defined "truth", and from which observations were taken). In section 4, we address these shortcomings by presenting results from a more realistic model, and by introducing model error into the analysis.

Current 4D-Var algorithms are highly sequential, making them unattractive for implementation on future computer architectures. In section 5, we address the possibilities for parallelising 4D-Var, and introduce a new formulation of 4D-Var that allows parallelisation in time.

2 Persistence of Past Information

All data assimilation algorithms combine information from the past with current information, in order to form an estimate of the current state of a system. Typically, the past information takes the form of a forecast from an earlier analysis. It may also include covariance information, as in the Kalman filter. Given this dependence on past information, it is of interest to investigate how far into the past information remains useful.

One way to evaluate the value of past information is to perform a data-reinstatement experiment. In this type of experiment, two identical analysis systems are run for a period of time with one system assimilating all available observations and the other system denied one type of observation (for example,

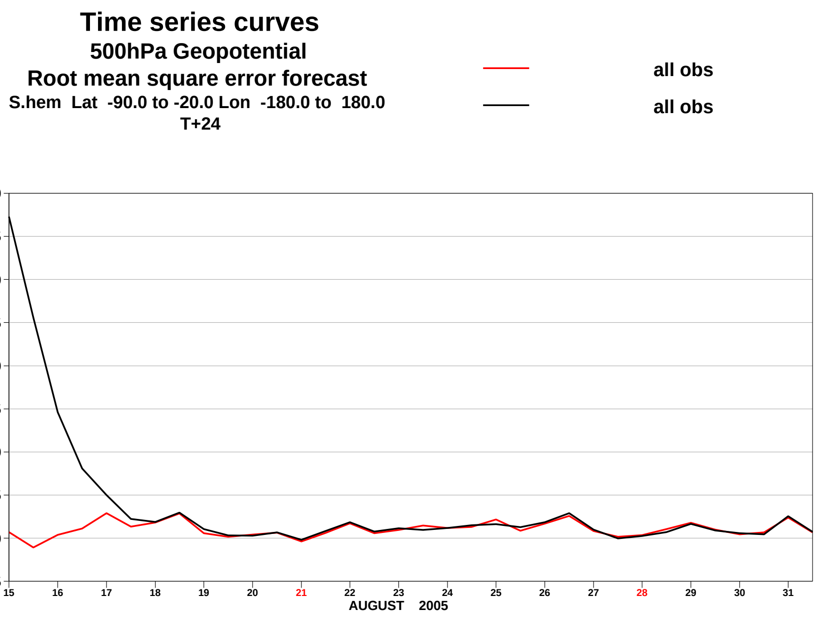

Figure 1: Time series from 15–31 August 2005 of 500hPa geopotential root-mean-square difference between 24 hour forecasts and verifying analyses averaged over the southern hemisphere. The red curve is for forecasts from an analysis system that used all available observations. The black curve is for forecasts from a system that was denied satellite radiances before 15 August.

satellite radiances). Once the two systems have reached consistent levels of analysis error (one would expect the system that has been denied a subset of observations to have larger analysis errors than the system that includes all the data), the withheld observation type is reinstated so that, for all subsequent analysis cycles, both systems assimilate identical observations.

Since the two analysis systems are identical, the differences between their analyses after reinstating the withdrawn observations are determined by their memory for the period during which they assimilated different observations. Assuming that this memory fades with time, we expect the analyses to converge over subsequent cycles. The rate at which this convergence takes place gives an estimate of the memory of the analysis system for past information.

Figure 1 shows, for two experiments, time series of the root-mean-square difference between 24-hour forecasts and their verifying analyses for 500hPa geopotential height averaged over the southern hemisphere. Both experiments were started from identical initial conditions on 1 August 2005. The experiment shown by the red curve assimilated all available observations. The experiment shown by the black curve was denied satellite observations until 15 August, after which they were reinstated. The errors of the 24-hour forecasts from this second experiment reduced over a period of three days, until they reached the same level as the first experiment. After this period of convergence, the two experiments are effectively indistinguishable. Time series for longer range forecasts (not shown) show a similar convergence over three days.

The experiment shows that the analysis system is effectively independent of observations and background states more than three days old. Note that this estimate is shorter than the seven days suggested by Fisher *et al.* (2005). The difference may be attributed to that fact that Fisher *et al.* estimated the memory following the withdrawal of satellite observations, rather than after their reinstatement. It is likely that the memory of an assimilation system for old information depends on the observing network. Fisher *et al*'s result shows that a system from which satellite radiances have been withdrawn has a mem-

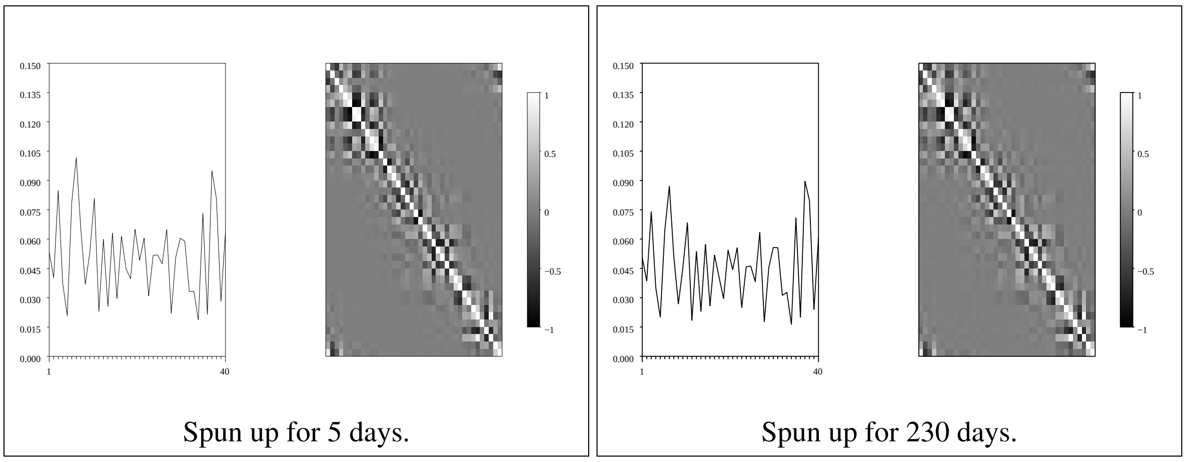

Figure 2: Variance and correlation of background error at the end of 230 days of assimilation for an extended Kalman filter applied to the Lorenz (1995) model. The left panel is for an experiment in which the covariance matrix was replaced by a static matrix during the first 225 days. The right panel is for an experiment in which the covariance matrix was allowed to evolve throughout the period.

ory of around seven days, whereas our data-reinstatement experiment suggests a memory of three days for a system that assimilates all available observations.

A further demonstration of the limited memory of data assimilation systems was given by Fisher *et al.* (*op cit.*) for an extended Kalman filter applied to the Lorenz (1995) model. They ran two experiments over a 230-day period. In one experiment, the covariance matrix was replaced at every cycle for the first 225 days by a static covariance matrix, after which it was allowed to evolve according to the Kalman filter equations. In the other experiment, the covariance matrix was allowed to evolve throughout the 230 days.

Figure 2 shows the background error variances and correlations at the end of the 230 days for both experiments. The covariance matrices for the two experiments are almost indistinguishable, apart from a small ($< 20\%$) change in the largest variances, showing that there is little memory for differences in background states or covariances older than five days. It is worth commenting that the model used for these experiments was perfect. That is, it was identical to the model used to perform the “truth run” from which the observations were taken. The covariance evolution equation of the extended Kalman filter included a contribution to account for linearisation error, but it was very small. The main contribution to the limited memory of the filter came from the effect of the chaotic dynamics of the system, not from the well-known “forgetting factor” effect of the model error covariance matrix.

3 Equivalence of 4D-Var and the Kalman Smoother

For a linear system, 4D-Var is algebraically equivalent to the Kalman smoother (see, for example, Ménard and Daley, 1996). That is, for a linear model and over a given interval during which observations are available, 4D-Var and the Kalman Smoother produce exactly the same sequence of states $x_0, \ldots, x_N$, given the same initial state x_b and covariance matrix B. At the end of the interval, both are equivalent to the Kalman Filter.

As shown in the previous section, if the interval is sufficiently long, the Kalman filter analysis x_N at the end of an interval is insensitive to old information from the beginning of the interval: i.e. to x_b and B. In this case, the Kalman filter analysis and error covariance at the end of the window are effectively indistinguishable from that of a Kalman filter that has been running indefinitely. Since both the Kalman smoother and 4D-Var give identical analyses at the end of the interval to that of the Kalman filter, it follows that 4D-Var gives effectively the same analysis x_N as a Kalman Filter that has been running indefinitely.

Viewed in this way, it is clear that for a sufficiently long analysis interval, 4D-Var is capable of generating the Kalman filter solution at the end of the interval, and may therefore be regarded as an algorithm for solving the Kalman Filter equations.

Strictly, the equivalence between 4D-Var and the Kalman filter holds only for a linear system. We believe that this is not fundamental to the argument. Both algorithms can be applied to nonlinear systems provided a suitable linearisation of the problem can be found. Although 4D-Var and the extended Kalman filter make different linearisation assumptions, it is difficult to argue that one is inherently better than the other in this respect. A practical demonstration of the equivalence of 4D-Var and the Kalman filter for a nonlinear system was given by Fisher *et al.* (*op cit.*). Moreover, there are significant similarities between the linearisation adopted by 4D-Var and the Iterated Extended Kalman Filter (Wishner *et al.*, 1969; Bell, 1994.)

4 Experiments with a Quasi-Geostrophic Model

Fisher *et al*'s (*op cit.*) results were for a somewhat unrealistic system. They used the Lorenz (1995) model, which is an extremely simple analogue for mid-latitude dynamics. Moreover, their experiments were conducted using a perfect assimilating model; the analysis cost function did not have a background term; and the system was rather well observed compared with a typical numerical weather prediction (NWP) system (60% of gridpoints observed).

Recent work has addressed these shortcomings, and is presented in this section. Specifically, we present results for a weak-constraint 4D-Var analysis for a two-level quasi-geostrophic channel model. We have verified that the model has realistic error-growth and nonlinearity characteristics. The error doubling time for the model is approximately 30 hours, and the nonlinearity index defined by Gilmour *et al.* (2001) reaches 0.7 (indicating a breakdown of linearity) after about 60 hours.

We introduced realistic model error into the assimilating model by perturbing model parameters (specifically, the depths of the two layers). This produces model error that is flow-dependent, time-correlated, strongly anisotropic and inhomogeneous, and which contains a significant systematic component. The error is only poorly described by the assumed covariance matrix, which represents isotropic homogeneous temporally-white noise with a Gaussian spatial correlation structure.

In contrast to the experiments presented by Fisher *et al.*, the new experiments included a background error covariance matrix (with a similar structure to that of the model error covariance matrix), and only 1.25% of gridpoints were observed every 6 hours.

The cycling scheme is key to the success of long-window 4D-Var. We illustrate the approach in figure 3. The bottom two boxes marked "Long-window analysis" represent two consecutive cycles of analysis. At each cycle, the analysis interval is advanced by six hours, regardless of the length of the interval. For analysis intervals longer than six hours this results in an overlap between successive analysis intervals, which allows the starting point of the minimisation (marked "First Guess" in figure 3) for the overlapping part of the interval to be taken from the preceding analysis cycle. (The starting point of the minimisation for the final six hours of the analysis interval must be provided by a short forecast.) This ensures that the starting point for the minimisation is close to the optimal solution over most of the interval. As a consequence, the analysis makes only small adjustments to its first guess, and the linearisation is accurate, even for windows much longer than the time-to-nonlinearity of the underlying model.

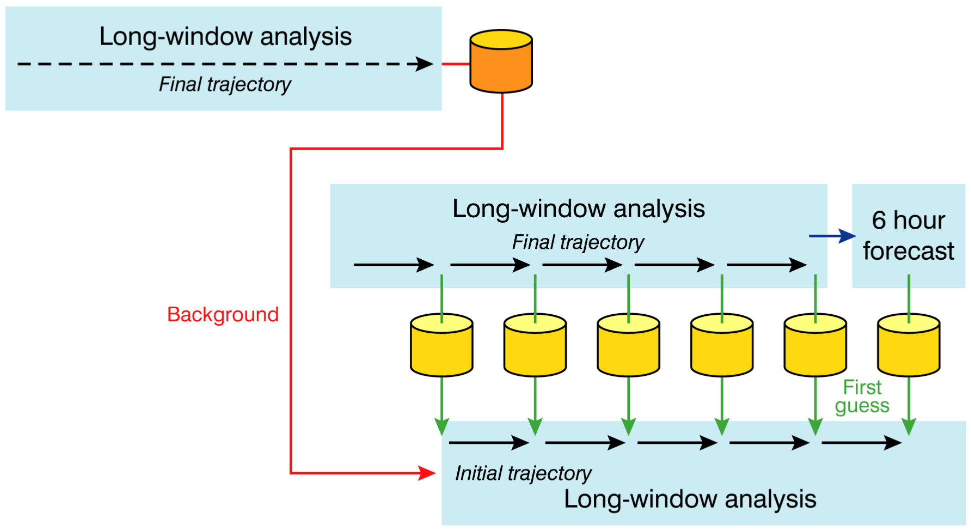

Figure 3: Schematic diagram showing the cycling algorithm used in the long window 4D-Var experiments described in this paper.

There is no statistical inconsistency in taking the starting point of the minimisation from the preceding analysis cycle, since there is no constraint for the analysis to stay close to the starting point. However, statistical correctness requires that the background state is taken from an earlier analysis that has not assimilated any of the current observations. This is indicated in figure 3 by the red line labelled "background", which shows that the background is taken from an earlier analysis that does not overlap the current cycle.

Since the model error induced by a change of model parameters has a large systematic component, it was found to be important to introduce a mean component of model error, $\bar{q}$, to account for it. The resulting weak-constraint 4D-Var cost function can be written as:

$$\begin{aligned} J(x_0,x_1,\ldots,x_N) \quad = \quad & \frac{1}{2}(x_0-x_b)^{\mathrm{T}}B^{-1}(x_0-x_b) \\ & +\frac{1}{2}\sum_{k=0}^{N}(y_k-\mathscr{H}_k(x_k))^{\mathrm{T}}R_k^{-1}(y_k-\mathscr{H}_k(x_k)) \\ & +\frac{1}{2}\sum_{k=1}^{N}(q_k-\bar{q})^{\mathrm{T}}Q_k^{-1}(q_k-\bar{q}) \end{aligned} \tag{1}$$

where $q_k = x_k - \mathscr{M}_k(x_{k-1})$, with $\mathscr{M}$ representing a model integration from the time of x_{k-1} to that of x_k.

Here, the analysis interval is divided into sub-windows, and the cost function is regarded as a function of the states $x_0,x_1,\ldots,x_N$ defined at the start of each sub-window. The operator $\mathscr{H}_k$ maps the state x_k to the observations within the sub-window. (In principle therefore, $\mathscr{H}_k$ may incorporate a model integration from the start of the sub-window to the times of the observations within the sub-window. For the experiments reported here observations were contemporaneous with the states $x_0,x_1,\ldots,x_N$, and no model integration was required.) The matrices R_k are covariance matrices of model error, B is the covariance matrix of background error, and x_b is the background (prior) state.

The final term of the cost function represents model error that is independent at each sub-window with covariance matrix Q_k and mean $\bar{q}$. We estimated the mean model error and the covariance matrices of model error and background error from large samples of forecasts. For the covariance matrices we then chose approximately representative error variance and correlation length scales which were used in the

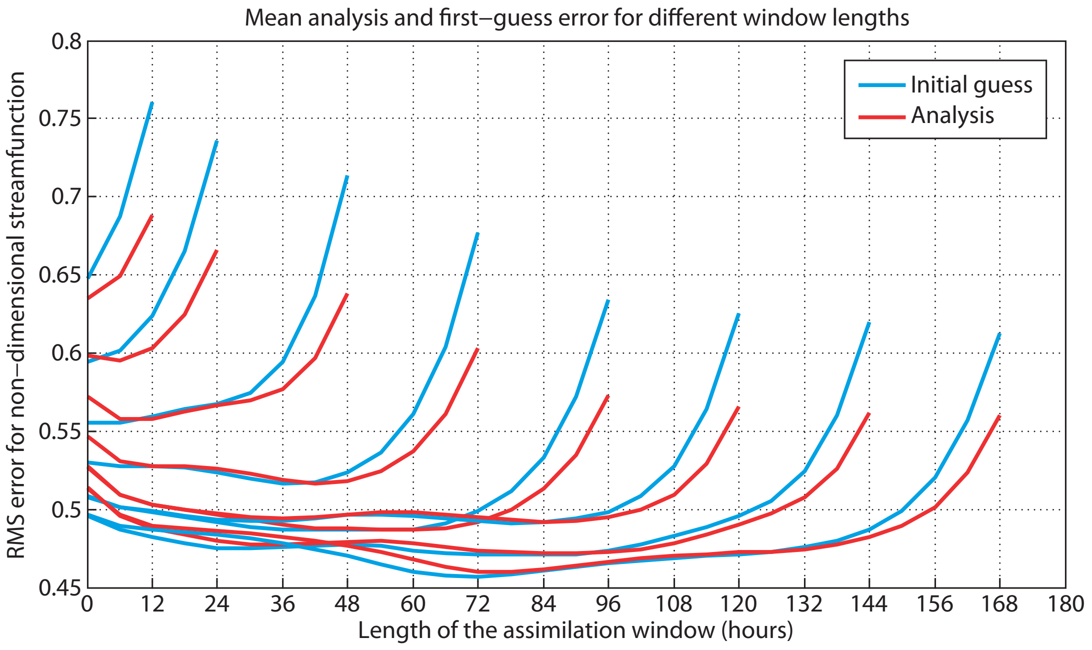

Figure 4: Root-mean-square error for streamfunction for analyses with different-length assimilation intervals. The red curves show analysis error during the assimilation interval. The blue curves show the error in the starting point for the minimisation.

isotropic, homogeneous covariance models.

Figure 4 shows the root-mean-square streamfunction error for the analysis and for the starting point of the minimisation, averaged over both levels of the model and over a large sample of cases. Values are plotted for a set of experiments with different-length analysis intervals. For each experiment, values corresponding to the states $x_0, x_1, \ldots, x_N$ are joined by blue and red lines.

It is clear from figure 4 that, throughout much of the assimilation interval, the starting point for the minimisation (indicated by the blue lines, and marked "initial guess") has errors similar in magnitude to the analysis error. This is unsurprising, since the starting point is taken from the preceding analysis cycle. The analysis (marked in red) improves on the starting point towards the end of the interval, where it benefits from an additional 6 hours of observations. Similarly, the loss of observations in the 6 hours preceding the analysis interval reduces the accuracy of the analysis near the beginning of the interval to below that of the initial guess for all except the shortest analysis interval.

The analysis error at the end of the assimilation interval decreases as the assimilation interval is extended. However, there is relatively little improvement for intervals longer than 96 hours. It is worth noting that the most accurate estimate of the state lies towards the middle of the analysis interval. This has interesting implications for re-analysis and for retrospective verification of analyses and forecasts.

The results presented in this section confirm those obtained by Fisher *et al.* (2005), while addressing some of the shortcomings of that study. It is clear that Long-Window 4D-Var is capable of producing good analyses even for a poorly observed system with an imperfect model and poorly described model error statistics. This is an encouraging result that strongly suggests that the method could be applied successfully in the NWP context. We note also that good analyses were obtained for analysis intervals significantly longer than the time-to-nonlinearity of the underlying model.

5 Parallel Algorithms for Weak-Constraint 4D-Var

In its usual formulation, 4D-Var is a highly sequential algorithm. Iterations of the minimisation algorithm are performed sequentially. Within each iteration, tangent linear and adjoint integrations run one after the other, and for each integration the model timesteps follow each other in sequence.

Computers are becoming ever more parallel, but processors are not getting faster, so that unless something can be done to make 4D-Var more parallel, it is likely to become un-affordable on future hardware. Up to now parallelisation of 4D-Var has benefitted from efforts to parallelise the model. However, the scope for further increases in the parallelism of the model is limited. The inner loops of the ECMWF 4D-Var analysis are currently run with just a few tens of grid columns per processor. This is barely enough to mask inter-processor communication costs, so that a further reduction in the number of grid columns per processor is unlikely to give much benefit. There is a need, therefore, to find increased parallelism in other aspects of 4D-Var.

As described above (see equation 1), weak constraint 4D-Var splits the analysis window into a set of sub-windows. At the beginning of each iteration of minimisation, the state at the start of each sub-window is known. In principle, this allows the model integrations ($\mathscr{M}_k$) and applications of the observation operator ($\mathscr{H}_k$), that are required to evaluate the terms in the two sums in equation 1, to be performed in parallel. As will be seen below, exploiting this parallelism is straightforward in the case of the outer loop of incremental 4D-Var, but is more difficult in the case of the inner loop.

5.1 Parallelising the Outer Loop

In strong-constraint 4D-Var, the inner loop calculates a single three-dimensional increment, δx_0, which is used to update the three-dimensional state x_0 at the outer loop. By contrast, in weak-constraint 4D-Var the inner loop produces a four-dimensional increment: $\delta x_0, \ldots, \delta x_N$. This increment is used to update the four-dimensional state $x_0, x_1, \ldots, x_N$ at the outer loop:

$$x_k^{(n)} = x_k^{(n-1)} + \delta x_k^{(n-1)} \tag{2}$$

where (n) is the outer loop index.

Following this update, integrations of the outer-loop model and applications of the observation operator are required to calculate quantities required by the next inner loop minimisation:

$$\begin{aligned} c_k^{(n)} &= \bar{q} - x_k^{(n)} + \mathscr{M}_k(x_{k-1}^{(n)}) \qquad (3) \\ d_k^{(n)} &= y_k - \mathscr{H}(x_k^{(n)}) \qquad (4) \end{aligned}$$

These calculations are independent for each sub-window, and can be performed in parallel.

5.2 Parallelising the Inner Loop

Linearising equation 1 about the trajectory $x_0^{(n)}, x_1^{(n)}, \ldots, x_N^{(n)}$ gives the inner-loop cost function:

$$\begin{aligned} \hat{J}(\delta x_0^{(n)}, \ldots, \delta x_N^{(n)}) = & \frac{1}{2}\left(\delta x_0 - b^{(n)}\right)^{\mathrm{T}} B^{-1} \left(\delta x_0 - b^{(n)}\right) \\ & + \frac{1}{2}\sum_{k=0}^{N} \left(H_k^{(n)} \delta x_k - d_k^{(n)}\right)^{\mathrm{T}} R_k^{-1} \left(H_k^{(n)} \delta x_k - d_k^{(n)}\right) \qquad (5) \\ & + \frac{1}{2}\sum_{k=1}^{N} \left(\delta q_k - c_k^{(n)}\right)^{\mathrm{T}} Q_k^{-1} \left(\delta q_k - c_k^{(n)}\right) \end{aligned}$$

where $\delta q_k = \delta x_k - M_k^{(n)} \delta x_{k-1}$, with $M_k^{(n)}$ denoting the tangent linear model; $H_k^{(n)}$ is the tangent liner observation operator; and $b^{(n)}$, $c_k^{(n)}$ and $d_k^{(n)}$ come from the outer loop:

$$\begin{aligned} b^{(n)} &= x_b - x_0^{(n)} && (6)\\ c_k^{(n)} &= \bar{q} - q_k^{(n)} && (7)\\ d_k^{(n)} &= y_k - \mathscr{H}_k(x_k^{(n)}) && (8) \end{aligned}$$

Parallelising the minimisation of the inner loop is not trivial. It is unlikely that simultaneous calculations of gradients of the cost function will provide significant benefit. This is because the conjugate gradient algorithm, which is the obvious choice for minimising the inner loop cost function, relies on building up a high order Krylov space based on the initial cost-function gradient. Building up this space requires sequential gradient calculations with each calculation providing input to the next gradient calculation. It seems likely, therefore, that parallelising the minimisation of the inner loop must be achieved by parallelising the computations within each iteration, rather than by parallelising the iterations themselves.

From now on, for convenience of notation, we will drop the outer-loop index (n) from the terms in equation 5. We can simplify the equation further by defining some four-dimensional vectors and matrices:

$$\delta\mathbf{x} = \begin{pmatrix} \delta x_0 \\ \delta x_1 \\ \vdots \\ \delta x_N \end{pmatrix} \qquad \delta\mathbf{p} = \begin{pmatrix} \delta x_0 \\ \delta q_1 \\ \vdots \\ \delta q_N \end{pmatrix}$$

These vectors are related through $\delta q_k = \delta x_k - M_k \delta x_{k-1}$, which we can write in in matrix form as:

$$\delta\mathbf{p} = \mathbf{L}\delta\mathbf{x} \tag{9}$$

where:

$$\mathbf{L} = \begin{pmatrix} I & & & & \\ -M_1 & I & & & \\ & -M_2 & I & & \\ & & \ddots & \ddots & \\ & & & -M_N & I \end{pmatrix} \tag{10}$$

We will also define:

$$\mathbf{R} = \begin{pmatrix} R_0 & & & \\ & R_1 & & \\ & & \ddots & \\ & & & R_N \end{pmatrix}, \qquad \mathbf{D} = \begin{pmatrix} B & & & \\ & Q_1 & & \\ & & \ddots & \\ & & & Q_N \end{pmatrix}, \tag{11}$$

$$\mathbf{H} = \begin{pmatrix} H_0 & & & \\ & H_1 & & \\ & & \ddots & \\ & & & H_N \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} b \\ c_1 \\ \vdots \\ c_N \end{pmatrix} \qquad \mathbf{d} = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_N \end{pmatrix}.$$

With these definitions, we can write the inner-loop cost function either as a function of $\delta\mathbf{x}$:

$$J(\delta\mathbf{x}) = (\mathbf{L}\delta\mathbf{x} - \mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\mathbf{L}\delta\mathbf{x} - \mathbf{b}) + (\mathbf{H}\delta\mathbf{x} - \mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\delta\mathbf{x} - \mathbf{d}), \tag{12}$$

or as a function of $\delta\mathbf{p}$:

$$J(\delta\mathbf{p}) = (\delta\mathbf{p} - \mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\delta\mathbf{p} - \mathbf{b}) + (\mathbf{H}\mathbf{L}^{-1}\delta\mathbf{p} - \mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\mathbf{L}^{-1}\delta\mathbf{p} - \mathbf{d}). \tag{13}$$

Let us now consider how the cost function (or its gradient) can be evaluated using equations 12 and 13.

In the case of equation 12, the initial four-dimensional increment, $\delta\mathbf{x}$, is known at the start of the calculation. Applying $\mathbf{H}$ to $\delta\mathbf{x}$ requires application of H_k to δx_k for $k = 0 \ldots N$. These applications can be performed in parallel. Likewise, evaluation of the first term of the cost function requires application of $\mathbf{L}$ to $\delta\mathbf{x}$, which in turn requires the application of M_k to δx_{k-1} for $k = 1 \ldots N$. These model integrations can also be performed in parallel. We see, therefore, that evaluation of the cost function (or its gradient) can be parallelised over sub-windows if the cost function is represented as a function of $\delta\mathbf{x}$.

Equation 13, however, cannot easily be parallelised. This version of the cost function requires the application of $\mathbf{L}^{-1}$. This is equivalent to solving the linear system $\mathbf{L}\delta\mathbf{x} = \delta\mathbf{p}$ for $\delta\mathbf{x}$ by forward substitution, which amounts to the following sequential algorithm:

```
for $k = 1 \ldots N$
    $\delta x_k = \delta q_k + M_k \delta x_{k-1}$
end
```

Equation 13 has a strong resemblance to the familiar strong-constraint 4D-Var and 3D-Var cost functions. In particular, we can precondition it using the covariance matrix that appears in its first term, to give:

$$J(\chi) = \chi^{\mathrm{T}}\chi + (\mathbf{H}\mathbf{L}^{-1}\delta\mathbf{p} - \mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\mathbf{L}^{-1}\delta\mathbf{p} - \mathbf{d}) \tag{14}$$

where $\delta\mathbf{p} = \mathbf{D}^{1/2}\chi + \mathbf{b}$.

The first term of equation 14 contributes an identity matrix to the Hessian matrix of second derivatives, that guarantees that no eigenvalues of the Hessian are smaller than one. Since the rate of convergence of the minimisation is largely determined by the ratio of the largest to smallest eigenvalues of the Hessian, preconditioning using $\mathbf{D}$ is effective in ensuring the convergence rate is not spoiled by the presence of small Hessian eigenvalues.

Reducing the Hessian of the first term of equation 12 to the identity matrix requires the following preconditioner:

$$\delta\mathbf{x} = \mathbf{L}^{-1}(\mathbf{D}^{1/2}\chi + \mathbf{b}). \tag{15}$$

The presence of $\mathbf{L}^{-1}$ in the preconditioner requires sequential model integrations, thereby making an otherwise parallel algorithm sequential. An obvious solution to this problem is to replace $\mathbf{L}^{-1}$ in equation 15 by a cheap approximation $\tilde{\mathbf{L}}^{-1}$. However, if we do this, we can no longer write the first term of the cost function as $\chi^{\mathrm{T}}\chi$. Rather, we have to calculate $\delta\mathbf{x}$, and then explicity evaluate the term as

$$(\mathbf{L}\delta\mathbf{x} - \mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\mathbf{L}\delta\mathbf{x} - \mathbf{b}). \tag{16}$$

In this case, the Hessian of of the first term of the cost function (with respect to χ) is no longer the identity matrix, but:

$$\mathbf{D}^{\mathrm{T}/2}\tilde{\mathbf{L}}^{-\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{L}\tilde{\mathbf{L}}^{-1}\mathbf{D}^{1/2}. \tag{17}$$

Unfortunately, the matrix $\mathbf{D}$, typically has some very small eigenvalues, corresponding (for example) to large spatial scales have near-zero variances. This makes the preconditioning extremely sensitive to the accuracy with which $\tilde{\mathbf{L}}$ approximates $\mathbf{L}$. It is difficult to choose an approximation that does not produce a range of very small Hessian eigenvalues, which ruin the conditioning of the minimisation.

So far, we have failed to find a preconditioner that is both effective and cheap to evaluate. This has prompted us to devise a new formulation of 4D-Var which we describe in the next section.

6 The Saddle Point Formulation of 4D-Var

Let us consider equation 12. At the minimum of the cost function, we have:

$$\nabla J = \mathbf{L}^{\mathrm{T}}\mathbf{D}^{-1}(\mathbf{L}\delta\mathbf{x}-\mathbf{b})+\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\delta\mathbf{x}-\mathbf{d}) = \mathbf{0} \tag{18}$$

Let us define:

$$\lambda = \mathbf{D}^{-1}(\mathbf{b}-\mathbf{L}\delta\mathbf{x}) \tag{19}$$
$$\mu = \mathbf{R}^{-1}(\mathbf{d}-\mathbf{H}\delta\mathbf{x}) \tag{20}$$

Then equations 19, 20 and 18 can be written as

$$\left.\begin{array}{rcl} \mathbf{D}\lambda+\mathbf{L}\delta\mathbf{x} & = & \mathbf{b} \\ \mathbf{R}\mu+\mathbf{H}\delta\mathbf{x} & = & \mathbf{d} \\ \mathbf{L}^{\mathrm{T}}\lambda+\mathbf{H}^{\mathrm{T}}\mu & = & \mathbf{0} \end{array}\right\} \Longrightarrow \begin{pmatrix} \mathbf{D} & \mathbf{0} & \mathbf{L} \\ \mathbf{0} & \mathbf{R} & \mathbf{H} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{H}^{\mathrm{T}} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \lambda \\ \mu \\ \delta\mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{d} \\ \mathbf{0} \end{pmatrix} \tag{21}$$

This is called the saddle point formulation of 4D-Var. The matrix in equation 21 is a saddle point matrix and is conventionally denoted $\mathscr{A}$. The matrix is real, symmetric, but indefinite. Note that the matrix contains no inverse matrices, and in particular does not contain $\mathbf{L}^{-1}$. This shows that we can apply $\mathscr{A}$ without requiring a sequential model integration (i.e. we can parallelise over sub-windows). The lack of inverses of covariance matrices in the saddle point matrix is also also an advantage, since it is often easier to construct the covariance matrices than their inverses.

Although the above derivation of the saddle-point formulation of 4D-Var is particularly simple, it does not give much insight into the formulation. An alternative derivation, which gives more insight, is possible if we write the minimisation of the original cost function (equation 12) as a constrained problem:

$$\min_{\delta\mathbf{p},\delta\mathbf{w}} J(\delta\mathbf{p},\delta\mathbf{w}) = (\delta\mathbf{p}-\mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\delta\mathbf{p}-\mathbf{b})+(\delta\mathbf{w}-\mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\delta\mathbf{w}-\mathbf{d}) \tag{22}$$
$$\text{subject to } \delta\mathbf{p}=\mathbf{L}\delta\mathbf{x} \text{ and } \delta\mathbf{w}=\mathbf{H}\delta\mathbf{x}.$$

To solve this constrained problem, we introduce the Lagrangian $\mathscr{L}$, together with vectors of Lagrange multipliers λ and μ for the constraints $\delta\mathbf{p}=\mathbf{L}\delta\mathbf{x}$ and $\delta\mathbf{w}=\mathbf{H}\delta\mathbf{x}$:

$$\begin{aligned}\mathscr{L}(\delta\mathbf{x},\delta\mathbf{p},\delta\mathbf{w},\lambda,\mu) = \; & (\delta\mathbf{p}-\mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\delta\mathbf{p}-\mathbf{b})+(\delta\mathbf{w}-\mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\delta\mathbf{w}-\mathbf{d}) \\ & +\lambda^{\mathrm{T}}(\delta\mathbf{p}-\mathbf{L}\delta\mathbf{x})+\mu^{\mathrm{T}}(\delta\mathbf{w}-\mathbf{H}\delta\mathbf{x})\end{aligned} \tag{23}$$

The solution of the constrained minimisation problem corresponds to the stationary point of the Lagrangian, which may be found by setting all of its first derivatives to zero:

$$\frac{\partial\mathscr{L}}{\partial\lambda}=\mathbf{0} \quad\Rightarrow\quad \delta\mathbf{p}=\mathbf{L}\delta\mathbf{x} \tag{24}$$
$$\frac{\partial\mathscr{L}}{\partial\mu}=\mathbf{0} \quad\Rightarrow\quad \delta\mathbf{w}=\mathbf{H}\delta\mathbf{x} \tag{25}$$
$$\frac{\partial\mathscr{L}}{\partial\delta\mathbf{p}}=\mathbf{0} \quad\Rightarrow\quad \mathbf{D}^{-1}(\delta\mathbf{p}-\mathbf{b})+\lambda=\mathbf{0} \tag{26}$$
$$\frac{\partial\mathscr{L}}{\partial\delta\mathbf{w}}=\mathbf{0} \quad\Rightarrow\quad \mathbf{R}^{-1}(\delta\mathbf{w}-\mathbf{d})+\mu=\mathbf{0} \tag{27}$$
$$\frac{\partial\mathscr{L}}{\partial\delta\mathbf{x}}=\mathbf{0} \quad\Rightarrow\quad \mathbf{L}^{\mathrm{T}}\lambda+\mathbf{H}^{\mathrm{T}}\mu=\mathbf{0} \tag{28}$$

Eliminating $\delta\mathbf{p}$ and $\delta\mathbf{w}$ using the first two equations, we are left with a set of three equations that may easily be rearranged to give the saddle-point system.

This derivation highlights the fact that the saddle-point system is different from both 4D-Var and 4D-PSAS. Both the latter can be derived from the Lagrangian by combining equations 24–28 to produce functions of $\delta\mathbf{x}$ or $\delta\mathbf{p}$ with positive definite Hessians. Effectively, 4D-Var and 4D-PSAS convert the constrained minimisation problem into unconstrained problems, with 4D-PSAS being the Lagrangian dual of 4D-Var. By contrast, the saddle-point formulation directly determines the stationary point of the Lagrangian — a function that has both positive and negative curvature.

7 Preconditioning the Saddle Point System

To solve the saddle point system, we have to precondition it. Preconditioning saddle point systems is the subject of much current research (see, for example, Benzi and Wathen, 2008; Benzi, Golub and Liesen, 2005). One possible preconditioner is (c.f. Bergamaschi, *et al.*, 2011):

$$\tilde{\mathscr{P}} = \begin{pmatrix} \mathbf{D} & \mathbf{0} & \tilde{\mathbf{L}} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} \\ \tilde{\mathbf{L}}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{29}$$

in which $\tilde{\mathbf{L}}$ approximates $\mathbf{L}$.

It is readily verified that $\tilde{\mathscr{P}}^{-1}$ can be expressed as

$$\tilde{\mathscr{P}}^{-1} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \tilde{\mathbf{L}}^{-\mathrm{T}} \\ \mathbf{0} & \mathbf{R}^{-1} & \mathbf{0} \\ \tilde{\mathbf{L}}^{-1} & \mathbf{0} & -\tilde{\mathbf{L}}^{-1}\mathbf{D}\tilde{\mathbf{L}}^{-\mathrm{T}} \end{pmatrix} \tag{30}$$

Note that application of $\tilde{\mathscr{P}}^{-1}$ requires an approximate inverse of $\mathbf{L}$. As an initial approach, we have generated approximate inverses using the following, easily proved, identity:

$$\mathbf{L}^{-1} = \mathbf{I} + (\mathbf{I}-\mathbf{L}) + (\mathbf{I}-\mathbf{L})^2 + \ldots + (\mathbf{I}-\mathbf{L})^{N-1} \tag{31}$$

Since this is a (finite) power series expansion, it suggests truncating the series at some order $< N-1$.

Figure 5 shows the convergence of the saddle-point formulation of 4D-Var as a function of iteration for a weak constraint 4D-Var analysis using the quasi-geostrophic system described in an earlier section. For this experiment, the analysis interval was 24 hours, and was divided into eight sub-windows. The saddle-point system was solved using GMRES. For comparison, the convergence of the "forcing" formulation of 4D-Var is also shown. This corresponds to solution of equation 14 using a conjugate-gradient algorithm.

It is clear from figure 5 that the rate of convergence per iteration of the saddle-point algorithm tends to increase as the accuracy of the approximation to $\mathbf{L}^{-1}$ increases. However, even when the exact inverse of $\mathbf{L}$ is used (corresponding to the curve labelled "7"), the saddle-point algorithm requires more iterations to achieve a comparable accuracy to that of the forcing formulation.

The chief advantage of the saddle-point algorithm is its parallel nature. The forcing formulation of 4D-Var requires, at each iteration, a sequential integration of the tangent linear model over the entire assimilation interval, followed by a sequential integration of the adjoint model. By contrast, the sub-window integrations required to apply $\mathscr{A}$ can be run in parallel. Moreover, application of $\tilde{\mathbf{L}}^{-1}$ can also be parallelised over sub-windows. For a given truncation order, m, application of $\tilde{\mathbf{L}}^{-1}$ requires m applications of $\mathbf{L}$, and therefore m sequential integrations for each sub-window. A further application

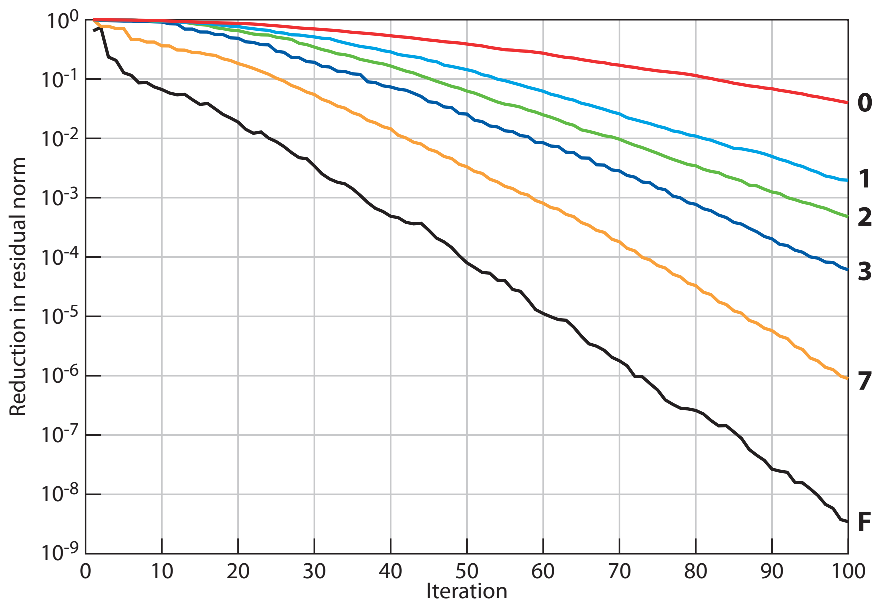

Figure 5: Convergence of the saddle-point algorithm as a function of iteration for different truncations of the series expansion for **L***. The numbers on the curves indicate the truncation order. The curve marked "F" shows the convergence of the "forcing formulation" of 4D-Var.*

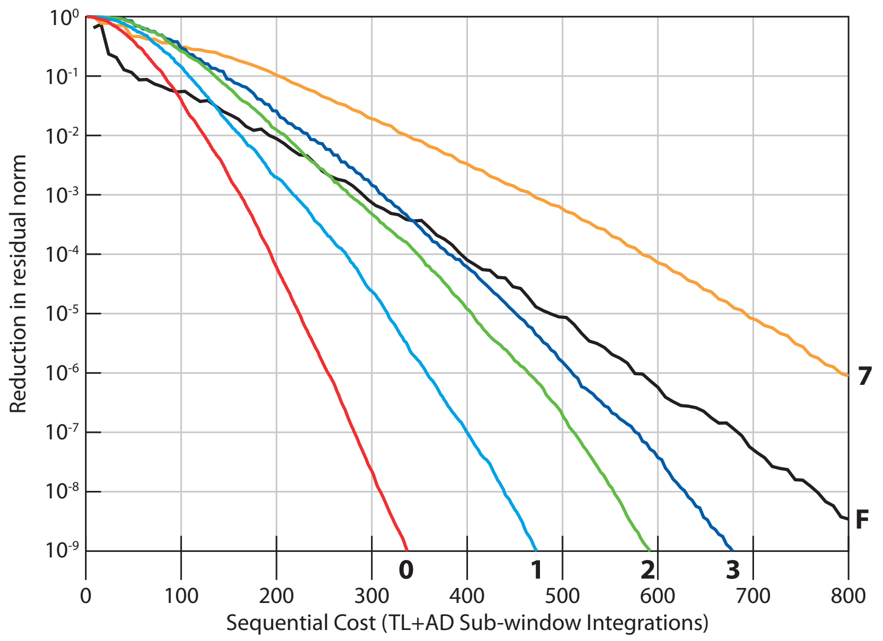

Figure 6: Convergence of the saddle-point algorithm as a function of sequential sub-window integrations for different truncations of the series expansion for **L**

of $\mathbf{L}$ is required to apply $\mathscr{A}$, making a total sequential cost for the saddle-point algorithm of $m+1$ sub-window integrations per iteration.

To give an indication of the possibility for parallel speed-up offered by the saddle-point formulation, we have re-plotted figure 5 with the horizontal axis showing necessarily-sequential sub-window integrations, assuming that tangent-linear and adjoint integrations are performed sequentially. (That is, we have multiplied the number of iterations by $m+1$, where m is the truncation order for $\tilde{\mathbf{L}}^{-1}$, or by eight in the case of the forcing formulation.) The result is shown in figure 6. For small M, the saddle-point formulation requires significantly fewer sequential integrations than the forcing formulation, and is particularly efficient if a zero-order (i.e. identity matrix) approximation to $\mathbf{L}^{-1}$ is used.

After producing this figure 6, we realised that a further increase in the parallel efficiency of the saddle point algorithm is possible if the tangent linear and adjoint integrations are performed simultaneously. (This possibility was also noted by Lagarde (2000), in the context of strong-constraint 4D-Var.) This would lead to an even greater parallel speed-up than is suggested by figure 6.

The results presented in this section are preliminary. We have investigated one possible preconditioner and one method of approximating $\mathbf{L}^{-1}$. It is possible that better preconditioners and better approximations can be found that would further reduce the computational cost of 4D-Var. In this context, we note that the matrix $\mathbf{L}$ plays an important role in the "Parareal" time-parallel integration algorithm (Lions *et al.*, 2001. See also Gander and Vandewalle, 2007). It is possible that exploiting this connection could lead to efficient and highly parallel preconditioners for weak constraint 4D-Var. It is also likely that, by taking the structure of the problem into account, more efficient solution algorithms than GMRES can be found.

8 Conclusions

The results presented above lend weight to the suggestion that long window 4D-Var is a viable data assimilation method for NWP. The data reinstatement experiment reported in section 2 hints that most of the benefits of extending the analysis window can be achieved by increasing the window from its current 12 hours to something less than three days. Experiments with a quasi-geostrophic model and with realistic model error are encouraging steps towards implementing long window 4D-Var in more complex models. Finally, we have presented an algorithm that allows a significant increase in the parallelism of 4D-Var. We expect this algorithm to allow 4D-Var to remain viable on next-generation supercomputers.

9 References

Bell B.M., 1994: The Iterated Kalman Smoother as a GaussNewton Method. *SIAM J. Optim.*, **4**, 626-636

Benzi M, Golub G H, and Liesen J, 2005: Numerical Solution of Saddle Point Systems, *Acta Numerica*, 1–137

Benzi M and Wathen A J, 2008: Some Preconditioning Techniques for Saddle Point Problems, *in W. Schilders, H. A. van der Vorst and J. Rommes, eds., Model Order Reduction: Theory, Research Aspects and Applications, Springer-Verlag (Series: Mathematics in Industry)*, 195–211.

Bergamaschi L., J. Gondzio, M. Venturin and G. Zilli, 2011: Erratum to: Inexact constraint preconditioners for linear systems arising in interior point methods. *Computational Optimization and Applications*, **49**, 401-406

Fisher, M., M. Leutbecher and G. Kelly, 2005: On the equivalence between Kalman smoothing and weak-constraint four-dimensional variational data assimilation. *Quart. J. Roy. Met. Soc*, **131**, 3235–3246.

Gander M.J. and Vandewalle S., 2007: Analysis of the Parareal Time-Parallel Time-Integration method, *SIAM Journal on Scientific Computing*, **29**, 2, 556–578.

Lagarde T, 2000: Nouvelle approche des methodes d'assimilation de données: les algorithmes de point selle. *Ph.D. thesis, Université Paul Sabatier (Toulouse III).*

Lions J-L, Y. Maday and G. Turinici, 2001: A "parareal" in time discretization of PDE's. *C. R. Acad. Sci. Paris Sér. I Math.*, **332**, 661–668.

Lorenz, E.N., 1995: Predictability: a problem partly solved, ECMWF Seminar on Predictability, 4-8 September 1995, 1–18.

Ménard, R. and R. Daley, 1996: The application of Kalman smoother theory to the estimation of 4D-Var error statistics. *Tellus*, **48A**, 221–237.

Wishner R.P., Tabaczynsk J.A. and Athans M., 1969; A comparison of three non-linear filters. *Automatica*, **5**, 487–496.