Mixed-precision arithmetic in the ENDGame dynamical core of the Unified model, a numerical weather prediction and climate model code.

C. M. Maynard[a,b], D. N. Walters[a]

[a]*Met Office, FitzRoy Road, Exeter, EX1 3PB*
[b]*Department of Computer Science, Polly Vacher Building, University of Reading, Reading, UK, RG6 6AY*

Abstract

The Met Office's weather and climate simulation code the Unified Model is used for both operational Numerical Weather Prediction and Climate modelling. The computational performance of the model running on parallel supercomputers is a key consideration. A Krylov sub-space solver is employed to solve the equations of the dynamical core of the model, known as ENDGame. These describe the evolution of the Earth's atmosphere. Typically, 64-bit precision is used throughout weather and climate applications. This work presents a mixed-precision implementation of the solver, the beneficial effect on run-time and the impact on solver convergence. The complex interplay of errors arising from accumulated round-off in floating-point arithmetic and other numerical effects is discussed. A careful analysis is required. The mixed-precision solver is now employed in the operational forecast to satisfy run-time constraints without compromising the accuracy of the solution.

Keywords: Weather and Climate, Krylov sub-space solver, floating-point error, precision, convergence

---

1. Introduction

Numerical simulations of the fluid dynamics of the Earth's atmosphere, which are central to all weather and climate models, require solutions to a non-linear system of partial differential equations (PDEs). These PDEs are recast into a discrete system of algebraic equations on a grid covering the Earth. For each grid cell, the fluid dynamic properties of the atmosphere, the velocity, temperature, pressure and density of the moist air, are then to be determined.

When deriving these algebraic equations, which arise from the continuous form of the compressible Euler equations, there is a choice of whether to integrate forward in time explicitly (a direct calculation) or (semi-)implicitly. The latter involves a global matrix inversion of some form. For Numerical Weather Prediction (NWP) simulations,

Email address: `c.m.maynard@reading.ac.uk` (C. M. Maynard) particularly those discretised onto a longitude-latitude (lon-lat) grid, the nature of the grid means that it is generally not feasible to use an explicit scheme as it would severely restrict the length of the time step employed in the simulation. A typical operational global NWP simulation is required to simulate seven to fourteen days within one-to-two hours of wall-clock time and the number of small time steps of the explicit scheme would either be prohibitively expensive, or severely limit the physical complexity or spatial resolution of an affordable simulation.

Semi-implicit schemes treat the fast acoustic-gravity modes implicitly and in combination with a semi-Lagrangian advection scheme, allow a stable integration of the flow around the polar singularity and in regions with strong horizontal flows. To an extent, these schemes also allow more flexibility in the formulation of the algebraic system and in particular the form of the global matrix inversion. In line with incompressible and low Mach number flows, a pressure correction equation is derived which takes the form

$$A \cdot \mathbf{x} = \mathbf{b}, \tag{1}$$

where A is a large, sparse (banded) matrix of order n, $\mathbf{x}$ is the pressure correction / tendency and $\mathbf{b}$ contains the explicit forcing terms. This can be solved using a combination of Krylov sub-space iterative solvers, pre-conditioners and parallel computers to satisfy wall-clock time constraints such as those described above.

The current operational configuration of the Met Office Unified Model (UM) [1] solves the dynamical system described above using the "ENDGame" dynamical core [2]. The operational deterministic global forecast is discretised onto an lon-lat grid with 2560×1920 grid points, giving it an average grid distance of approximately 10 km at mid-latitudes. With 70 discrete vertical levels, the number of degrees of freedom n for the pressure correction system is large at approximately 350 million; the full algebraic system is six times larger. For such a large system, a parallel supercomputer is required. However, even on state-of-the-art parallel computers such as the Met Office's Cray XC40 machine, both algorithmic and code optimisations are necessary to scale to a large degree of parallelism[1] and to execute the program as quickly as possible.

One potential optimisation is to consider the numerical precision to which variables in the code are computed and stored, as reducing the precision can provide performance advantages. In common with many scientific numerical applications, weather and climate codes are memory bound.

Using smaller data types reduces the total data movement between memory and CPU and reduces the amount of data for remote communication between processors. For fixed bandwidth, more data items can be transferred in a fixed time. It can also use less energy [3]. Moreover, caches can be utilised more effectively. For weather and climate models, however, knowledge of accuracy and uncertainty are important. There are many processes that affect the evolution of the system that are either not resolved at the cut-off scale imposed by the discretisation or are not represented by the dynamical core. These processes are represented by physical parametrizations, which often include many branches in the code. For example, the presence or not of liquid

[1]The operational global forecast model uses around 500 compute nodes, each with 36 CPU cores.

water in a grid cell will determine whether latent heat can be released through the freezing of that water; this in turn will affect the temperature of the atmosphere and so influence the evolution of the dynamics. An aversion to the possibility of compromising the accuracy of a simulation through either accumulated round-off errors, or round-off errors triggering different branches with parametrizations, has led to the “safety first” approach of using high precision arithmetic throughout, and the assumption that this is both necessary and more accurate. In common with many scientific applications, therefore, 64-bit floating point arithmetic – often referred to as double precision – is used by default and has become the accepted standard.

Recent work [4, 5, 6, 7] and [8] has explored the potential benefits of moving away from the use of 64-bit arithmetic throughout an atmospheric model. These papers examine a variety of approaches to reducing precision, from low precision emulators, Field Programmable Gate Arrays (FPGAs), toy models, right through to running a full atmospheric model in 32-bit precision (*i.e.* single precision) to examine the effect of precision on forecast accuracy. Whilst it is true that 64-bit precision may be desirable, if not required, for some portions of the simulation, many schemes within a model will have either physical or parametric uncertainty, or will be designed to reach an approximate solution, with accuracy far removed from the numerical precision of the variables within the scheme.

One such example is the numerical solution of Equation (1). There has been some analysis of the convergence properties and performance benefits of mixed-precision solvers [9, 10, 11, 12] and indeed, the emergence of lower precision hardware support in architectures such as GPUs (see for example [13]) has made the question of precision much more relevant. However, in this work the interaction between a mixed-precision solver and the evolution of a dynamical system such as the dynamical core of a NWP simulation is considerd. In [14], the authors examine the effect on model evolution of treating different length scales with different precision. Whilst this is a promising approach, it can only be employed in spectral transformation algorithms.

The solution for the pressure correction $\mathbf{x}$ is obtained through an iterative solver, in which each variable is typically defined to double precision, but in which the calculation is halted when the normalised error has been reduced several of orders of magnitude, typically three or four. Whilst the solver is only a small proportion of the total code base when measured by the number of lines of code, at the resolutions and node counts described for operational global NWP above, these routines constitute approximately one quarter of the run time of the full simulation[2]. In this work, a mixed-precision solver is defined, in which the pressure field itself is held in double precision throughout, but the majority of calculations, and hence the majority of memory transfers, are performed in single precision. In section 2 the ENDGame dynamical core, the Helmholtz equation to be solved for the pressure correction and the pre-conditioner used for this are described. In section 3 the mixed-precision solver is

[2]Individual model components have different computation and communication patterns and thus scale differently but for the global model at the resolution described, the approximate time proportions are one quarter for the solver, one quarter for advection and half for the total cost of the various physical parameterisation schemes. This excludes the time spent in doing I/O.

described, whilst in section 4 the effect this has on convergence to the solution as well as the computational speed up obtained is discussed. Finally, conclusions are drawn in section 5.

2. The ENDGame Dynamical core

Symbolically, the linear system of equations to be solved is of the form

$$\begin{pmatrix} I & 0 & 0 & 0 & 0 & G_u \\ 0 & I & 0 & 0 & 0 & G_v \\ 0 & 0 & I & B_i & 0 & G_w \\ 0 & 0 & B_{ii} & I & 0 & 0 \\ D_x & D_y & D_z & 0 & I & 0 \\ 0 & 0 & 0 & E_\theta & I & E_\pi \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ \theta \\ \rho \\ \Pi \end{pmatrix} = \begin{pmatrix} R_u \\ R_v \\ R_w \\ R_\theta \\ R_\rho \\ R_\Pi \end{pmatrix} \tag{2}$$

where I represents identity matrices, $G_{u,v,w}$ are discrete pressure gradient terms, $D_{x,y,z}$ are components of the divergence operator, B_i and B_{ii} are the coupling between the potential temperature and the vertical component of velocity due to gravity and $E_{\theta,\rho}$ arise from linearisation of the equation of state. The solution of this system corresponds to the fluid velocity components (u, v, w) and the tendencies (change between time steps) of the thermodynamic variables ρ (density), Π (Exner pressure) and θ (potential temperature[3]), where Exner pressure is defined as

$$\Pi = \left(\frac{P}{P_0}\right)^{\kappa}, \ \kappa = \frac{R}{C_p} \tag{3}$$

where P is the pressure, $P_0 = 10^5$ Pa, a reference surface pressure, $R = 287.05$ Jkg^{-1}K^{-1} is the specific gas constant and $C_P = 1005$ Jkg^{-1}K^{-1} is the specific heat capacity at constant pressure, of dry air. Note that within the ENDGame formulation, due to non-linearities and the change to a terrain following coordinate system, the right hand side terms involve contributions from the previous iterates of the current time step. Typically this linear system is solved four times per time step and so the solution method needs to be efficient.

This matrix is of the general form

$$\begin{pmatrix} P & Q \\ R & S \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{s} \\ \mathbf{t} \end{pmatrix} \tag{4}$$

which can be solved using the Schur compliment as

$$\mathbf{y} = P^{-1}(\mathbf{s} - Q\mathbf{x}); \quad A\mathbf{x} = (S - RP^{-1}Q)\mathbf{x} = \mathbf{t} - RP^{-1}\mathbf{s} = \mathbf{b}. \tag{5}$$

The matrix A has a 7-band structure similar to what would be expected from discretising the Laplacian operator, which, would be its exact equivalent in the Boussinesq incompressible limit. In the full system, however, it differs in some key aspects. Firstly,

[3]The absolute temperature $T = \theta \times \Pi$.

the matrix is non-symmetric and it is not constant coefficient; *i.e.*, the matrix rows are all different due to the spherical nature of the Earth and variations in temperature and orography (the height of the lower boundary). Secondly, as may be anticipated from the previous sentence, the matrix is time varying and so needs to be recomputed at every time step. This precludes the ability to pre-factorise the matrix off-line. Finally, because the vertical length scales are much shorter than the horizontal and gravitational effects are far larger than the vertical accelerations, some care is needed in order to solve this system of equations. In particular, it is necessary to ensure that the large scale hydrostatic balance is maintained during the iterations. It should be noted that, since this equation arises from a non-linear algebraic system, there are terms involving **x** which are lagged and so appear as forcing in **b** (see [2] for further details).

The system in equation (5) is solved using a Krylov subspace solver based on a variant of the BiCGstab algorithm [15] that is pre-conditioned on the right (post-conditioned) [16] and shown as pseudo-code in appendix Appendix A, algorithm 3. Equation (5) is replaced by

$$D_g^{-1}ACC^{-1}\mathbf{x} = D_g^{-1}\mathbf{b} \quad \equiv \quad \hat{A}CC^{-1}\mathbf{x} = \hat{\mathbf{b}}, \tag{6}$$

where D_g is the diagonal of A and C is the pre-conditioner. BiCGstab is used rather than the Conjugate Gradient algorithm [17] due to the lack of symmetry in the derived matrix A. The application of the diagonal is performed outside the solver and reduces the need to access this term during the application of $A\mathbf{x}$ within the Krylov method. The application of C is performed through a few (typically three) iterations of a second stationary method derived from the Successive-Over-Relaxation method (SOR), but with a decomposition that better reflects the nature of the atmospheric problem. This is achieved by writing

$$\hat{A} = L + T + U, \tag{7}$$

where T is a tridiagonal matrix arising from the vertical discretisation, L is a lower triangular matrix and U is an upper triangular matrix. Following the standard derivation of SOR, the fixed point method is obtained.

$$(T + \omega L)\mathbf{x}_n = (1 - \omega)T\mathbf{x}_{n-1} + \omega(r - U\mathbf{x}_{n-1}), \tag{8}$$

where $\omega = 1.5$ is the over-relaxation parameter and the only difference from the standard method is the appearance of the tridiagonal matrix T in place of the diagonal. Note that the lower/upper-factorisation of the matrix T can be pre-computed (at each horizontal grid point) to aid the inversion process. Furthermore, there is no parallel decomposition in the vertical, which simplifies the process of applying T^{-1} considerably.

Algorithmic pseudo-code for the ENDgame dynamical core algorithms of the main loop, inner loop and Helmholtz solve and the BiCGStab algorithm are shown in appendix Appendix A.

3. The Mixed-Precision Solver

The accuracy of the solution reached by the BiCGstab solver is determined by the halting criterion

$$||\mathbf{r}_i|| = ||\mathbf{b} - A \cdot \mathbf{x}_i||, \tag{9}$$

where $\mathbf{x}_i$ is the solution after i iterations and $\mathbf{r}_i$ is known as the residual vector. If the L_2 norm of $\mathbf{r}_i$ is small, then $\mathbf{x}_i$ is close to the solution $\mathbf{x}$. The halting criterion for this algorithm is to break out of the iteration loop when the residual (either absolute or relative) has fallen below some threshold. The relative norm of the residual (R), is defined as

$$\mathrm{R} = \frac{||\mathbf{r}_i||}{||\mathbf{b}||} \tag{10}$$

and then the halting criterion is defined as when $\mathrm{R} < \mathrm{R}_c$, where R_c is known as the critical residual or the "solver tolerance". The optimum value of R_c is determined by a compromise between the desired *accuracy* of the solution and its computational *cost*.

In floating point arithmetic, the Unit of Least Precision (ULP) is defined in [18] as follows. The ULP in x is the distance between the closest two *straddling* IEEE floating point numbers a and b, *i.e.* those with $a \leq x \leq b$. For numbers $O(1)$ this is $O(10^{-7})$ for 32-bit numbers and $O(10^{-15})$ for 64-bit. If the stopping criterion

$$\mathrm{R}_c \gg \epsilon_{32}, \tag{11}$$

where ϵ_{32} is the ULP or machine precision for 32-bit floating point numbers, then 32-bit floating point numbers can satisfy the desired accuracy. Iterative solvers and associated errors for weather and climate applications are discussed in [19, 20]. Iterative methods in general are discussed in [21]. In chapter 4, section 2, on the stopping criterion, the authors discuss accuracy, precision and error. They show that the stopping criterion should not be set such that $\mathrm{R}_c < \epsilon$, Where ϵ is the precision used in the calculation. Conversely, decreasing ϵ from, for instance, 10^{-7} for 32-bit precision to 10^{-15} for 64-bit precision has no effect on the accuracy of the solution if $\mathrm{R} \gg \epsilon_{32}$.

In the case of the semi-implicit time stepping scheme, the solution to the linear system is part of the solution procedure [2] to a larger, non-linear system and the accuracy of the solve is dictated by the need for stability of the time stepping scheme. Moreover, it is only an indirect measure of the error. It is worth noting that since the finite difference approximations to the pressure gradient in the UM are at best second order, there is a limit to the effect of tightening the solver tolerance on the pressure. *i.e.* once the error in the solver is sufficiently small, the discretisation error becomes dominant.

In its first implementation, all operational systems using ENDGame used a tolerance of $\mathrm{R}_c = 10^{-3}$, which easily satisfies the condition laid out in equation (11). These calculations can be done in 32-bit without loss of accuracy compared to a calculation done in 64-bit, which offers a significant optimisation opportunity, especially as the solver and the pre-conditioner are memory bound. A 32-bit version of the code has in effect twice the cache memory available compared to a 64-bit version. This motivated the use of the mixed-precision solver, which contributed to the optimisations that made the implementation computationally affordable.

In principle, the whole BiCGStab algorithm could be written in 32-bit. However, the rest of the model remains 64-bit. The two components have been combined and it is natural to encapsulate the solver algorithm as a subroutine. It is not possible in Fortran to coerce a 64-bit real to a 32-bit real as an argument to a subroutine, modify its value and promote it back safely to a 64-bit real when the subroutine exits. So, rather than pay the cost of a whole domain memory copy from 64- to 32-bit when entering the routine and again on exit, the pressure field is kept as a 64-bit data-type and so mixed precision arithmetic is required. Most of the operations, including communications, can be performed in single precision and the Fortran run-time can coerce or promote the remaining operations automatically.

Following the initial implementation of ENDGame, numerical noise was observed in the horizontal wind field near the pole. It is common for models defined to longitude/latitude grids to horizontally filter their fields close to the poles to remove noise caused by inconsistencies in their formulation across the polar singularity. Unlike many such models — including previous revisions of the Met Office Unified Model — the ENDGame dynamical core is discretised self-consistently (*i.e.* the equations are written down and a single approach to discretising them is applied). Moreover, this is done in a manner that is physically consistent across the pole, by using a C-grid staggering with the meridional wind held at the pole, following the recommendations in [22]. This, coupled with the semi-Lagrangian scheme (and its scale selective filtering), should be sufficient to solve the dynamics without the need for additional filtering. Experimentation showed that ENDGame's noise at the pole could be alleviated by tightening the solver tolerance [23], although it remains unclear whether this noise was a direct feature of insufficient convergence in the vicinity of the pole, or was created by some other numerical issue and removed/reduced through additional iterations of the pressure solver.

The latest operational global configuration uses a tolerance of $R_c = 10^{-4}$, which although an order of magnitude tighter than originally used, still satisfies the inequality in (11).

4. Numerical and computational performance

4.1. Idealised model

To test the convergence of the solver, an idealised or serial toy-model of the ENDGame solver was constructed. The toy-model is simply the BiCGstab solver and pre-conditioner with $\mathbf{x}$ initialised to a representative pressure field, the Helmholtz matrix with the correct structure for ENDGame and representative right-hand side $\mathbf{b}$ values. No halting criterion is used, but instead the solver is run to very high levels of convergence, to study the convergence behaviour of R. Shown in figure (1) is the value of $||R||$ for 32-bit, 64-bit and 128-bit solvers as a function of iteration number.

All the solvers show the non-monotonic convergence typical of the BiCGStab algorithm. For the first few iterations, convergence is sufficiently similar to be indistinguishable. To reach a residual of $R = 10^{-3}$ or less takes 18, 15, 16 iterations for the 32-, 64- and 128-bit solvers respectively. Both the 64- and 128-bit solver residual values plunges quickly thereafter. As the 128-bit data type is represented in software it is much slower than either of the lower precision solvers.

The "iteration gap" between the 32-bit and the higher precision solvers grows as the residual falls. The 64-bit and 128-bit solver residuals remain close and this can be interpreted as the 64-bit solver being free of round-off errors. At tighter convergence, the 32-bit version takes 35 iterations to reach $R = 10^{-7}$ compared to 20 for the 64-bit version. This may well be due to the loss of orthogonality in the Krylov sub-space

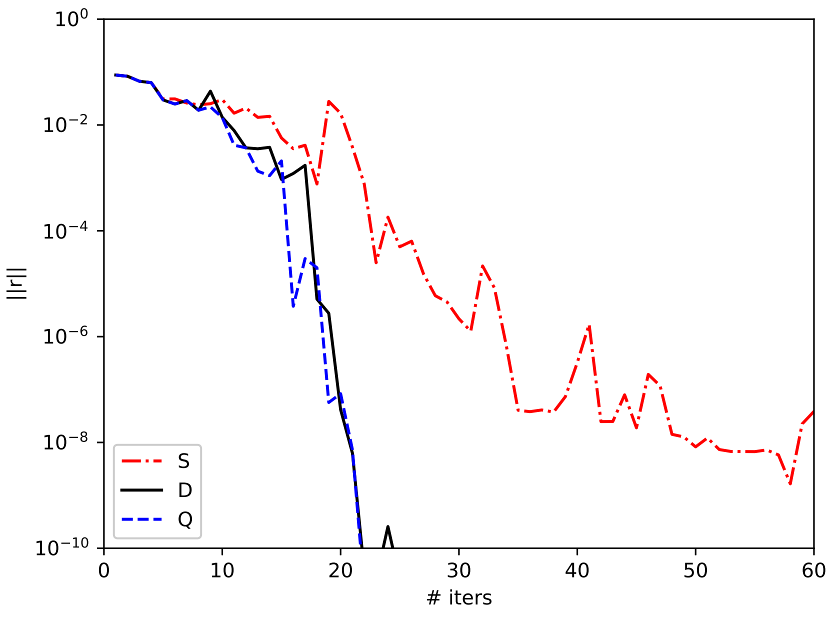

Figure 1: The relative norm of residual, R defined in equation (10) versus iteration number. The red (dot-dashed) line, labelled "S" shows data from the 32-bit solver, the black (solid) line labelled "D" the 64-bit solver and the blue (dashed) line labelled "Q" the 128-bit solver.

vectors. In the BiCGstab algorithm, these vectors are updated each iteration, they are not re-computed. Accumulated round-off errors set in earlier for the lower precision versions. The estimate of these vectors becomes worse with increasing iteration number resulting in a loss of efficiency. This can be recovered by restarting the algorithm; this comes at the cost of re-computing the vectors, but this new sequence of vectors is then (initially) closer to being orthogonal.

BiCGStab employs two Krylov subspaces, from algorithm 3 in Appendix A, they are:

$$\mathcal{K}_p = \text{span}\{p, Ap, \cdots\} \tag{12}$$

and

$$\mathcal{K}_C = \text{span}\{C, AC, \cdots\} \tag{13}$$

In common with most Krylov subspace solvers, BiCGstab employs only the first two vectors of each space. The inner product of these vectors is a measure of their orthogonality

$$O(\mathcal{K}_p) = p \cdot Ap \tag{14}$$

and similarly for C. This quantity can be computed every iteration. The orthogonality measure falls rapidly as the algorithm converges making it difficult to compare. However, the 128-bit data can be considered to have much less accumulated round off

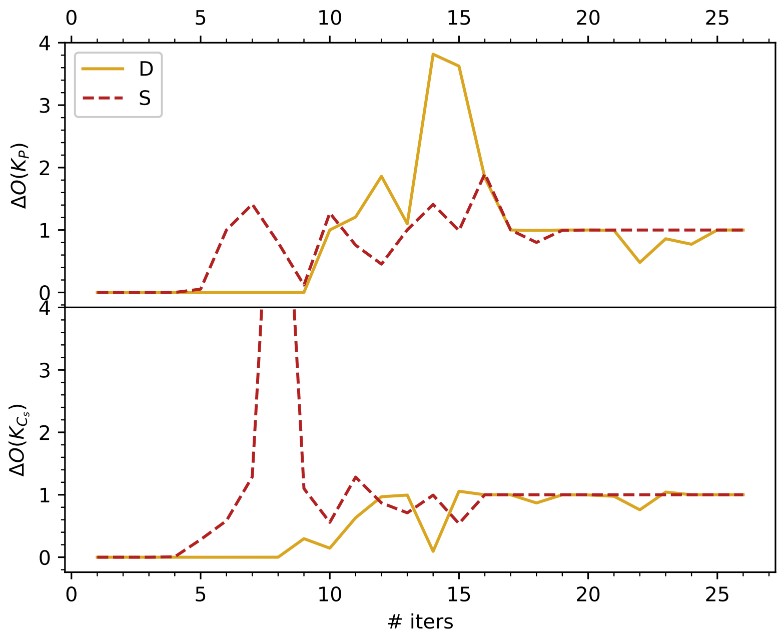

Figure 2: The relative difference of the orthogonalty measure for 32- and 128-bit solver data, labelled 'S' in red (dashed line) and the relative difference of the orthogonality measure for the 64- and 128-bit solver data, labelled 'D' in yellow (solid line), versus iteration number. Top panel shows the data for the p subspace and the bottom panel for the C_s subspace.

error, so by comparing the difference between the 128-bit data and the data for the other solvers, an estimate of the error in the orthogonality can be made. The relative difference Δ is defined as

$$\Delta(a, b) = \frac{\text{abs}(a - b)}{\text{abs}(a + b)} \tag{15}$$

where abs is the absolute value. This is plotted in figure 2.

For both spaces the value starts at near zero (no error) and tends towards one (error dominates) and moreover, the 32-bit data both deviates from near zero and tends towards one at a lower iteration number than the 64-bit data. This analysis suggests that the lower precision solver is indeed suffering from a loss of orthogonality in the Krylov subspaces.

4.2. Full model

The decision to implement the mixed-precision solver in the full model was based on tests using comprehensive weather and climate model simulations. These included high resolution 5-day NWP forecast runs and further testing in an operational-like research NWP system. Results with the mixed-precision and double precision solvers showed no noticeable scientific differences and hence the mixed-precision solver was

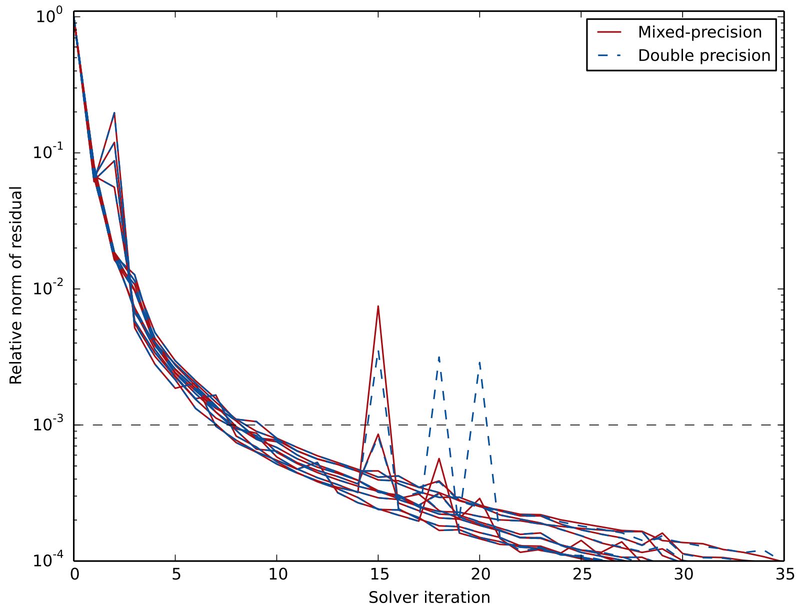

Figure 3: The relative norm of residual, R defined in equation (10) versus iteration number from the first call to the solver in 11 simulation dates each run with the mixed- and double precision solver.

accepted. To study the behaviour of the solver in more detail in an operational-like forecast, a set of N1280 global model simulations were run for a series of 11 dates each separated by one week, each on 96 nodes of the Met Office Cray XC40[4]. Mixed- and double precision solver runs for each date were initialised from 64-bit model states **x**, saved from a previous short forecast[5]. Each XC40 node is dual socket, each with an 18-core Intel Xeon (Broadwell) processor. The model is parallelised with both MPI and OpenMP by domain decomposition across the two dimensional horizontal domain. It was run with six MPI ranks per socket and three OpenMP threads per MPI rank. Again, the convergence was measured using the residual R defined in equation (10). Shown in figure (3) is the value of R versus the iteration number, for the mixed- and double precision version, up to the operational convergence criterion of $R_c = 10^{-4}$. The data for convergence of the solvers is also shown in table (1).

The results presented from each date are from the first call to the solver in the first

[4]To improve the robustness of the timing information, each forecast was submitted 3 times, to sample additional run-to-run variability. For a given date/experiment, the results of these resubmissions are identical to the bit level, so these do not add to the sampling of iteration counts or solver convergence.

[5]Operational forecast runs usually start from 32-bit truncated states for efficiency, and will usually have some special treatment ahead of and during the first time step, such as the inclusion of data assimilation increments and the use of a fully implicit first time step to better handle any imbalance these may cause. Using 64-bit dumps and switching these additional options off makes the first time step of these runs representative of later "typical" model time steps.

model time step, so that the prognostic fields on entering the solver are identical between the mixed- and double precision versions, so as to make it simple to compare the behaviour between the two. There are several features in the figure worth remarking on. Firstly, for the first five iterations, the fall in the size of the residual is quite steep, showing quick convergence, although four of the eleven sets of simulations demonstrate the lack of guaranteed monotonicity, even this early on. Secondly, up to and slightly beyond the original operational halting criterion of $R_c = 10^{-3}$, the values of R_{MP} and R_{DP} are very close, where MP denotes mixed-precision and DP double or 64-bit precision. After a single iteration, these agree for every date to seven significant figures, *i.e.* roughly the precision of a 32-bit number. Also, for each date, the residual reaches the halting criterion in an identical number of iterations, and the values of R_{MP} and R_{DP} at that point all agree to at least four significant figures. Between $R = 10^{-3}$ and $R = 10^{-4}$, the rate of convergence slows down and in a few cases jumps to a value an order of magnitude higher before dropping back down again, although these jumps are present in both the mixed- and double precision solver, and appear not to be related to the precision of the calculations.

The time taken to reach $R \leq 10^{-3}$ and $R \leq 10^{-4}$ is shown in Table 1. The mixed-precision BiCGStab is almost twice as fast per iteration than the 64-bit version, for the reasons outlined in section 1, *viz.* that the smaller data type halves the data movement both from memory to CPU and remote communication and doubles the number of data items in the memory caches. For example, the local volume for each MPI rank/OpenMP thread is

$$LV = (l_x + 2h_x) \times (l_y + 2h_y) \times n_{levels}/n_{threads} \tag{16}$$

where l is local domain size, h the halo size, x the East-West direction, y the North-South directions, n_{levels} is the number of vertical levels and $n_{threads}$ is the the number of OpenMP threads the data are shared between. For a 96 node (3456 core) job, a typical decomposition might be 72 MPI ranks East-West and 16 North-South with 3 OpenMP threads resulting in a local volume of

$$LV = \left(\frac{2560}{72} + 2\right) \times \left(\frac{1920}{16} + 2\right) \times 70/3 = 111720 \tag{17}$$

The pre-conditioner routine, a tridiagonal SOR pre-conditioner, requires ten domain valued arrays for the 7-point stencil and back substitution. Thus the size of the working set, N, is

$$N = 111720 \times 10 = 1117200 \tag{18}$$

The memory footprint for a working set of this size is 4468800 bytes in 32-bit and 8937600 bytes in 64-bit. The Met Office supercomputer processors are Intel Xeon E5-2695 v4, which has a level 2 (L2) cache of size 4718592 bytes. The working set would fit into L2 cache in 32-bit. Of course, what data are resident in cache is much more complicated than simple size. However, the working set clearly cannot fit into L2 cache in 64-bit. Other levels of cache, main memory bandwidth and remote communication bandwidth will all affect the speed of computation and by using smaller data these can be exploited more effectively.

Table 1: The mean (standard error) iteration count and wall clock time taken in seconds for the mixed- and double-precision solvers to reach the converge to the halting criterion R_c.

	Mixed-precision		Double precision	
R	Iterations	Time (s)	Iterations	Time (s)
10^{-3}	8.4(2)	0.346(6)	8.4(2)	0.592(9)
10^{-4}	29.3(9)	1.18(2)	29.3(9)	2.06(3)

To see the effect on the iteration count and convergence of the algorithm for this work, the simulations from figure (3) have been extended to use a tighter halting criterion of $R_c = 10^{-5}$, which is shown in figure (4). This shows that when $R < 10^{-4}$, the BiCGstab algorithm starts to break down as beyond this point, the value of the residual falls slowly and often jumps to a higher value between iterations. The simulations with the double precision solver take hundreds of iterations to converge to $R \leq 10^{-5}$, and can experience many jumps before this occurs. Below $R \approx 10^{-4}$, however, it is clear that the mixed-precision approach suffers from severe errors and after 500 iterations, only two of the simulations have converged, whilst others have become unstable and the residual has started growing rather than reducing with time.

In operational simulations, occasional problems with convergence have been observed even with $R_c = 10^{-4}$. In these cases, the scalar weights of the BiCGStab algorithm are close to or equal to zero and dividing by these very small numbers can result in floating-point overflows and the algorithm failing. The scalars are calculated by a global sum, which in the original mixed-precision solver were performed in 32-bit, with the local summation performed first, then an MPI reduction call on a single number. To protect against the scalars summing to zero, however, the global sum only has since been reverted to 64-bit. This has a negligible cost as the MPI reduction is the dominant cost of the procedure and this is latency bound. Reverting the global sums to 64-bit reduces the likelihood of the model failing with overflowing fields, but the problem of slow convergence remains. Slow convergence can be the result of a loss of orthogonality of the Krylov sub-space, although there are multiple reasons for which BiCGStab can break down [24]. To address this and improve the robustness of operational systems, the implementation of BiCGStab was modified to include a mandatory restart if convergence has not been achieved after a fixed number of iterations. The restart threshold is set at a relatively high number of iterations of 150. With this modification, the mixed-precision solver is found to be both computational efficient and robust enough for operational use.

5. Conclusion

Speed, accuracy and stability of computation are all important criteria for numerical calculations, particularly for an operational NWP system. When carefully controlled, the use of reduced precision can have a big impact on the speed of a calculation without affecting its accuracy or stability. However, as shown in this work, if the numerical algorithms employed are susceptible to non-convergence, reduced precision can only be safely utilised within a certain regime.

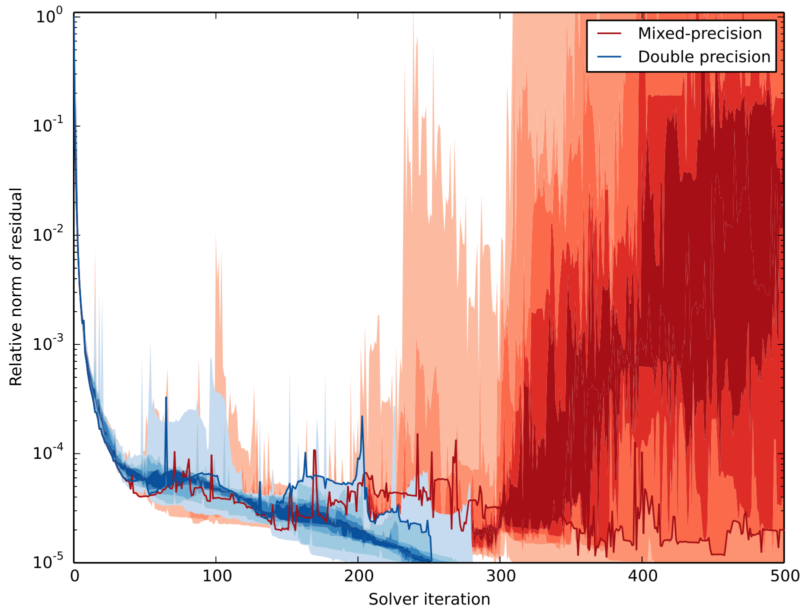

Figure 4: The relative norm of residual, R defined in equation (10) versus iteration number from the same simulations presented in figure 3, but with the tighter halting criterion of $R_c = 10^{-5}$. Shading is used to highlight the ranges of R per iteration, with a "typical" single case highlighted in the solid lines.

There are undoubted benefits to reducing precision, in this case halving the run-time of the solver. However, other numerical instabilities such as the polar noise described in a previous section have forced the model to be run on a regime where the iterative, Krylov solver has problems with slow convergence. This makes the use of reduced precision all the more important as the run-times with a tighter stopping criteria are longer. Naively, it should be possible to run with reduced precision at tighter stopping condition than 10^{-4}, however, problems with slow convergence and numerical stability indicate that the algorithm used, BiCGStab, is failing, either to converge or doing so far too slowly. This is true for the double-precision version and so it is safe to conclude that these issues are not caused by using reduced precision. However, as the reduced precision version appears to suffer more severe problems, including a failure to converge, it appears that there is a complex interplay between accumulated round-off errors, such as a loss of orthogonality in the Krylov sub-space as demonstrated in the idealised example, and other errors introduced by the numerical scheme.

These issues, combined with the behaviour shown in figure (4), suggest that there could be some benefit from continuing to investigate the cause of the noise in the horizontal wind field near the pole. This may allow the operational forecasts to revert to the slacker solver tolerance of $R_c = 1.0 \times 10^{-3}$ and hence make the solution less susceptible to problems with convergence, whilst reducing computational cost. These results also make clear that the poor convergence is not directly related to the mixed-precision solver, and would still be present if the solver were routinely run solely in double precision. This should encourage atmospheric model developers to continue to consider reducing the precision of some calculations to improve the efficiency of their simulations.

6. Acknowledgements

The authors thank T. Allen and N. Wood, both from the Met Office, for in-depth discussions and advice.

Appendix A. Algorithms

The algorithms that make up the ENDgame dynamical core time-step, described in the main body of the text.

Algorithm 1: The Main ENDGame Loop

```
Input:  Model State (i.e.  previous time level values).
Calculate "slow physics" increments;
Calculate Sources to be evaluated at departure points;
Set guess at next time level to current model state;
Calculate and Store the scaled Helmholtz Matrix and vertical LU-factorisation;
for Outer_Loop = 1,2,... do
    Calculate departure points and interpolate sources to them;
    Add sources lagged in the inner loop;
    Advect tracers if (present and final outer loop);
    Calculate "fast physics" sources;
    Inner Loop: Solve Helmholtz Problem;
end
Copy new time level values to old ready for next time step;
Diagnostics, physics "tidy-up", etc
```

Where the RHS for the solver is derived from the “fast physics” sources and shown in equation (2).

Algorithm 2: The ENDGame Inner Loop

```
Input:  Current best estimate of next time level fields,
 sources from advection and physics.
Calculate Fixed part of the right-hand-side (RHS) to the Helmholtz problem;
Update moisture fields;
for Inner_Loop = 1,2,... do
    Calculate New time level sources and add additional terms;
    Build variable part of RHS for the Helmholtz equation;
    Solve linear system: Bi-CGSTAB;
    Update Prognostic fields for new time level;
end
```

Algorithm 3: The post-conditioned (or right pre-conditioned) BiCGSTAB method

```
Inputs:  $\boldsymbol{x}$, $\boldsymbol{r}$, $\epsilon_{tol}$, $\delta_{min}$
$\delta = \max(||\boldsymbol{r}||, \delta_{min})$;
$\boldsymbol{p} = \boldsymbol{v} = 0$;
$\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{Ax}$; $\boldsymbol{r}_0 = \boldsymbol{r}$;
$\alpha = \omega = n = 1$;
for $k$ = 1,2,... do
    $\rho = (\boldsymbol{r}, \boldsymbol{r}_0)$; $\beta = \alpha\rho/(n\omega)$;
    $\boldsymbol{t} = \boldsymbol{r} - \beta\omega\boldsymbol{v}$; $\boldsymbol{s} = \boldsymbol{Ct}$;
    $\boldsymbol{p} = \boldsymbol{s} + \beta\boldsymbol{p}$; $\boldsymbol{v} = \boldsymbol{Ap}$;
    $n = (\boldsymbol{v}, \boldsymbol{r}_0)$; $\alpha = \rho/n$;
    if $\omega < 10^{-12}$ then
        Convergence problem $\omega$ too small.
    end
    $\boldsymbol{s} = \boldsymbol{r} - \alpha\boldsymbol{v}$; $\tilde{\boldsymbol{s}} = \boldsymbol{Cs}$; $\boldsymbol{t} = \boldsymbol{A}\tilde{\boldsymbol{s}}$;
    $\omega = (\boldsymbol{t}, \boldsymbol{s})/||\boldsymbol{t}||^2$;
    $\boldsymbol{x} = \boldsymbol{x} + \alpha\boldsymbol{p} + \omega\tilde{\boldsymbol{s}}$; $\boldsymbol{r} = \boldsymbol{s} - \omega\boldsymbol{t}$;
    $n = \rho$; $\epsilon = ||\boldsymbol{r}||/\delta$;
    If $\epsilon < \epsilon_{tol}$ exit;
end
```

References

[1] A. Brown, S. Milton, M. Cullen, B. Golding, J. Mitchell, A. Shelly, Unified modeling and prediction of weather and climate: a 25 year journey, Bulletin of the American Meteorological Society 93 (2012) 1865–1877. doi:10.1175/BAMS-D-12-00018.1.

[2] N. Wood, A. Staniforth, A. White, T. Allen, M. Diamantakis, M. Gross, T. Melvin, C. Smith, S. Vosper, M. Zerroukat, J. Thuburn, An inherently mass-conserving semi-implicit semi-lagrangian discretization of the deep-atmosphere global non-hydrostatic equations, Quarterly Journal of the Royal Meteorological Society 140 (682) (2014) 1505–1520. doi:10.1002/qj.2235.

[3] D. Molka, D. Hackenberg, R. Schne, M. S. Mller, Characterizing the energy consumption of data transfers and arithmetic operations on x8664 processors, in: International Conference on Green Computing, 2010, pp. 123–133. doi:10.1109/GREENCOMP.2010.5598316.

[4] P. D. Düben, T. N. Palmer, Benchmark tests for numerical weather forecasts on inexact hardware, Monthly Weather Review 142 (10) (2014) 3809–3829. doi:10.1175/MWR-D-14-00110.1.

[5] P. D. Düben, H. McNamara, T. Palmer, The use of imprecise processing to improve accuracy in weather and climate prediction, Journal of Computational Physics 271 (2014) 2–18, Frontiers in Computational Physics. doi:10.1016/j.jcp.2013.10.042.

[6] P. D. Düben, F. P. Russell, X. Niu, W. Luk, T. N. Palmer, On the use of programmable hardware and reduced numerical precision in Earth-system modeling, Journal of Advances in Modeling Earth Systems 7 (3) (2015) 1393–1408. doi:10.1002/2015MS000494.

[7] F. Vana, P. Düben, S. Lang, T. Palmer, M. Leutbecher, D. Salmond, G. Carver, Single precision in weather forecasting models: An evaluation with the IFS, Monthly Weather Review 145 (2) (2017) 495–502. doi:10.1175/MWR-D-16-0228.1.

[8] M. Nakano, H. Yashiro, C. Kodama, H. Tomita, Single precision in the dynamical core of a nonhydrostatic global atmospheric model: Evaluation using a baroclinic wave test case, Monthly Weather Review 146 (2) (2018) 409–416. doi:10.1175/MWR-D-17-0257.1.

[9] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, S. Tomov, Accelerating scientific computations with mixed precision algorithms, Computer Physics Communications 180 (12) (2009) 2526 – 2533. doi:https://doi.org/10.1016/j.cpc.2008.11.005.

[10] A. Buttari, J. Dongarra, J. Langou, J. Langou, P. Luszczek, J. Kurzak, Mixed precision iterative refinement techniques for the solution of dense linear systems, International Journal of High Performance Computing Applications 21. doi:10.1177/1094342007084026.

[11] E. Carson, N. Higham, A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems, SIAM Journal on Scientific Computing 39 (6) (2017) A2834–A2856. doi:10.1137/17M1122918.

[12] E. Carson, N. J. Higham, Accelerating the solution of linear systems by iterative refinement in three precisions, SIAM J. Scientific Computing 40.

[13] A. Haidar, S. Tomov, J. Dongarra, N. J. Higham, Harnessing gpu tensor cores for fast fp16 arithmetic to speed up mixed-precision iterative refinement solvers, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC '18, IEEE Press, Piscataway, NJ, USA, 2018, pp. 47:1–47:11.
URL `http://dl.acm.org/citation.cfm?id=3291656.3291719`

[14] M. Chantry, T. Thornes, T. Palmer, P. Dben, Scale-selective precision for weather and climate forecasting, Monthly Weather Review 147 (2) (2019) 645–655. doi:10.1175/MWR-D-18-0308.1.

[15] H. A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 13 (2) (1992) 631–644. doi:10.1137/0913035.

[16] A. M. Bruaset, A survey of preconditioned iterative methods, Vol. 328, Harlow: Longman Scientific and Technical, 1995.

[17] M. Hestenes, E. Steifel, Methods of conjugate gradients for solving linear systems, J. Res. Nat. Bur. Stand. 49 (1952) 409–436.

[18] J. Harrison, A machine-checked theory of floating point arithmetic, in: Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, L. Théry (Eds.), Theorem Proving in Higher Order Logics: 12th International Conference, TPHOLs'99, Vol. 1690 of Lecture Notes in Computer Science, Springer-Verlag, Nice, France, 1999, pp. 113–130.

[19] P. Smolarkiewicz, L. Margolin, Variational solver for elliptic problems in atmospheric flows., Appl. Math. Comp. Sci. 4 (1994) 527–551.

[20] T. Allen, Comparison of iterative pressure solvers for turbulent flow over hills, Atmospheric Science Letters 8 (1) (2007) 21–28. doi:10.1002/asl.145.

[21] R. Barrett, M. Berry, T. F. Chan, J. D. James Demmel, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. V. der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, 1993.

[22] J. Thuburn, A. Staniforth, Conservation and linear rossby-mode dispersion on the spherical c grid, Monthly Weather Review 132 (2) (2004) 641–653. doi:10.1175/1520-0493(2004)132¡0641:CALRDO¿2.0.CO;2.

[23] D. Walters, I. Boutle, M. Brooks, T. Melvin, R. Stratton, S. Vosper, H. Wells, K. Williams, N. Wood, T. Allen, A. Bushell, D. Copsey, P. Earnshaw, J. Edwards, M. Gross, S. Hardiman, C. Harris, J. Heming, N. Klingaman, R. Levine, J. Manners, G. Martin, S. Milton, M. Mittermaier, C. Morcrette, T. Riddick, M. Roberts, C. Sanchez, P. Selwood, A. Stirling, C. Smith, D. Suri, W. Tennant, P. L. Vidale, J. Wilkinson, M. Willett, S. Woolnough, P. Xavier, The Met Office Unified Model Global Atmosphere 6.0/6.1 and JULES Global Land 6.0/6.1 configurations, Geosci. Model Dev. 10 (2017) 1487–1520. doi:10.5194/gmd-10-1487-2017.

[24] P. Graves-Morris, The breakdowns of BiCGStab, Numerical Algorithms 29 (1) (2002) 97–105. doi:10.1023/A:1014864007293.