Weather and climate models in 16-bit arithmetics: Number formats, error mitigation and scope

M. Klöwer[1], P. D. Düben[2], and T. N. Palmer[1]

[1]Atmospheric, Oceanic and Planetary Physics, University of Oxford, Oxford, UK

[2]European Centre for Medium-Range Weather Forecasts, Reading, UK

Key Points:

○ Posit numbers have smaller rounding errors compared to floating-point numbers in weather and climate applications, enabling reliable shallow water simulations computed entirely with 16-bit arithmetic.

○ Errors caused by 16-bit floating-point arithmetic are strongly reduced with critical computations in 32 bit, which can be implemented on present-day hardware.

○ 16 or even 8-bit communication between processors, preferably encoded as posit numbers, introduces negligible errors, providing a perspective for reduced data communication for weather and climate models.

Corresponding author: M. Klöwer, `milan.kloewer@physics.ox.ac.uk`

Abstract

The need for high precision calculations with 64-bit or 32-bit floating-point arithmetic for weather and climate models is questioned. Lower precision numbers can accelerate simulations and are increasingly supported by modern computing hardware. This paper investigates the potential of 16-bit arithmetic when applied within a shallow water model that serves as a medium complexity weather or climate application. There are several 16-bit number formats that can potentially be used (IEEE half precision, BFloat16, posits, integer and fixed-point). It is evident that a simple change to 16-bit arithmetic will not be possible for complex weather and climate applications as it will degrade model results by intolerable rounding errors that cause a stalling of model dynamics or model instabilities. However, if the posit number format is used as an alternative to the standard floating-point numbers the model degradation can be reduced to a tolerable minimum. Furthermore, a number of mitigation methods, such as rescaling, reordering and mixed-precision, are available to make model simulations resilient against a precision reduction. If mitigation methods are applied, 16-bit floating-point arithmetic can be used successfully within the shallow water model. The results show the potential of 16-bit formats for at least parts of complex weather and climate models where rounding errors would be entirely masked by intitial condition, model or discretization error.

Plain Language Summary

64-bit floating-point numbers are the standard number format for scientific computing in fluid dynamics, which allows for very precise calculations with negligible rounding errors. The need for calculations at this precision level has been questioned for weather and climate models, as errors are caused primarily by insufficient observations or deficiencies of the models themselves. Reducing numerical precision can accelerate simulations and low precision number formats are increasingly supported by modern computers. This paper investigates the potential of low numerical precision with numbers that only use 16 bit of information, when applied within weather and climate applications. There are several 16-bit number formats that can potentially be used, all of which have considerably larger rounding errors than the standard 64-bit numbers. In this paper, the different number formats are applied in a two-dimensional oceanic or atmospheric circulation model. A simple change to 16-bits for all calculations will not be possible as it will degrade simulation results. However, if mitigation methods are applied, 16-bit cal-

culations can be used successfully within the applications of this paper. The results show the potential of 16-bit number formats for at least parts of complex weather and climate models.

1 Introduction

Weather and climate models provide predictions that are of great importance for society and economy. The Earth's climate system remains very difficult to predict even with the computational resources of the world's largest supercomputers, due to its complexity and non-linear dynamics that couple all features from the smallest time and length-scales to the largest. The forecast error of a weather forecast model has several origins (T. N. Palmer, 2012; T. Palmer, 2015): (i) Initial and boundary condition errors, which result from observations, data assimilation, model errors and external factors; (ii) model error, i.e. the difference between the mathematical model and the real world; (iii) discretisation error resulting from a finite spatial and temporal resolution of the discretised equations and (iv) rounding errors with finite precision arithmetic. In general, the forecast error is dominated by (i-iii), depending on the forecast variable and the forecast lead time. In contrast, rounding errors are usually negligible if the IEEE 754 standard on 64 bit double precision floating-point numbers (Float64) is used, which is the standard for the majority of operational weather forecasts and in climate models.

Research on reduced precision floating-point arithmetics is motivated by the potential for faster processing and communication between different elements of the computing architecture. The gained speed can be traded for increased complexity of simulations, resulting in more accurate predictions of weather and climate. The Integrated Forecast System at the European Centre for Medium-Range Weather Forecasts can be run almost entirely with 32-bit single precision (Float32) without a decrease in forecast skill (Váňa et al., 2017), but in 60% of the run-time. Similar progress was made at MeteoSwiss with their weather forecast model COSMO (Rüdisühli et al., 2013). For the European ocean model NEMO a mix of 32-bit and 64-bit arithmetic is a promising approach to keep accuracy-critical parts in high precision while increasing performance in others (Tintó Prims et al., 2019).

Deep learning algorithms require only low numerical precision but high computational performance. The recent boom of machine learning applications increased the de-

mand on hardware-accelerated reduced precision calculations, such that hardware developments increasingly offer more flexibility on low precision number formats. While 16-bit arithmetic was not available for use on commodity supercomputing hardware in the past, today most hardware vendors offer the use of 16-bit formats on the next generation of hardware. Graphic processing units started to support Float16 for increased performance (Markidis et al., 2018). Google's tensor processing units (TPU, Jouppi et al. (2017, 2018)) support the 16-bit BFloat16 format, a truncated version of Float32, as this format is sufficient for many deep learning applications (Kalamkar et al., 2019; Burgess et al., 2019; Gupta et al., 2015).

Using simplistic chaotic models, it was shown that the majority of 64 bits with Float64 do not contain real information (Jeffress et al., 2017). Running algorithms used for weather forecast models at precision lower than 32 bit, for example with 16-bit half precision floats (Float16), is an active field of research, but remains challenging (Düben & Palmer, 2014; Thornes et al., 2017; Hatfield et al., 2018; Düben, 2018). Most research on reduced precision modelling for weather and climate applications makes use of software emulators (Dawson & Düben, 2017) that provide other arithmetics than Float32 and Float64 which are widely supported on today's hardware. This comes with the disadvantage that simulations are orders of magnitude slower. However, software emulation allows a scientific evaluation of the use of reduced numerical precision for weather and climate simulations with no need to port the models to special hardware (such as FPGAs, Russell et al. (2017)).

Posit™ numbers are a recently proposed alternative to floating point numbers and claim to provide more precision in arithmetic calculations with fewer bits in algorithms of linear algebra or machine learning (J. L. Gustafson & Yonemoto, 2017; Langroudi et al., 2019). However, posits remain untested for weather and climate simulations.

To get a better impression whether 16-bit arithmetic can be useful within weather and climate applications, which 16-bit formats are most promising, and how model simulations can be made resilient against a reduction in precision to 16 bits, this study will apply common types of 16-bit arithmetic in weather and climate applications and test approaches to mitigate negative impact if 16-bit arithmetic cannot be used for the entire model.

The study is structured as follows: Section 2 outlines the different number formats, the concept of decimal precision, and mitigation methods how weather and climate mod-

els can be made resilient against a reduction in numerical precision of 16-bit formats. Section 3 presents results of various implementations of 16-bit arithmetic in the shallow water model. Section 4 discusses the results and provides the conclusions.

2 16-bit number formats and mitigation methods

This section discusses different types of 16-bit arithmetic, and similarities and differences between them. Mitigation methods that allow for the use of 16-bit arithmetic within weather and climate applications are introduced.

2.1 16-bit number formats

2.1.1 The integer and fixed point number format

The simplest way to represent a real number in bits is the integer format. An n-bit signed integer starts with a sign bit followed by a sequence of integer bits, that are decoded as a sum of powers of 2 with exponents $0, 1, ..., n-2$. The largest representable number $maxpos$ for a signed integer format is $2^{n-1} - 1$. Fixed-point numbers extend the integer format by adding n_f fraction bits to decode an additional sum of powers of 2 with negative exponents $-1, -2, ..., -n_f$. Every additional fraction bit reduces the number of integer bits. For example, Q6.10 is the 16-bit fixed-point format with 6 signed integer bits and 10 fraction bits.

The arithmetic of fixed-point numbers is very similar to integers. The range of representable numbers with integer arithmetic can therefore be changed with fixed-point numbers, providing some flexibility for integer arithmetics (Russell et al., 2017). However, the width of the dynamic range $\log_{10}(maxpos/minpos)$ is always limited and less than five orders of magnitude for any 16-bit integer or fixed-point format – too small for most applications. We will therefore focus on the discussion of the other number formats in the rest of the study.

2.1.2 The floating-point number format

The IEEE standard on floating-point arithmetic defines how floats encode a real number x in terms of a sign, and several exponent and significant bits

$$x = (-1)^{sign\ bit} \cdot 2^{e-bias} \cdot (1 + f). \tag{1}$$

The exponent bits e are interpreted as unsigned integers, such that $e - bias$ converts them effectively to signed integers. The fraction (or significant) bits f_i are defined as before, such that the significand $(1 + f)$ is in the bounds $[1, 2)$. An 8-bit float encodes a real number with a sign bit (red), $n_e = 3$ exponent bits (blue) and $n_f = 4$ fraction bits (black) as illustrated in the following example

$$3.14 \approx 01001001_{\text{Float8}} = (-1)^0 \cdot 2^{4-bias} \cdot (1 + 2^{-1} + 2^{-4}) = 3.125 \tag{2}$$

with $bias = 2^{n_e-1} - 1 = 3$. Exceptions to Eq. 1 occur for subnormal numbers, infinity (Inf) and Not-a-Number (NaN) when all exponent bits are either zero (subnormals) or one (Inf when f=0, or NaN else). 16-bit half-precision floating point numbers (Float16) have 5 exponent bits and 10 significant bits. A truncated version of the Float32 format (8 exponent bits, 23 significant bits) is BFloat16 with 8 exponent bits and 7 signficant bits. A format with more exponent bits has a wider dynamic range of representable numbers but lower precision, as fewer bits are available for the significand. All floating-point formats have a fixed number of bits in the significand, consequently, they have a constant number of significant digits throughout their range of representable numbers (subnormals excluded).

2.1.3 The posit number format

Posit numbers arise from a projection of the real axis onto a circle (Fig. 1), with only one bitpattern for zero and one for Not-a-Real (NaR, also called *complex infinity*), which serves as a replacement for Not-a-Number (NaN) as well as positive and negative infinity. The circle is split into *regimes*, determined by a constant *useed*, which always marks the north-west on the posit circle (Fig. 1b). Regimes are defined by $useed^{\pm 1}$, $useed^{\pm 2}$, $useed^{\pm 3}$, etc. which set a wide dynamic range of representable numbers. To encode these regimes into bits, posit numbers use regime bits which extend the standard on floating-point arithmetic. Regime bits are a sequence of identical bits after the sign bit, terminated by an opposite bit. As the number of regime bits is not fixed but flexible, the significand increases in length for numbers towards ± 1, when fewer regime bits are needed. Consequently, a higher precision around one can be achieved with posits, which is traded against a gradually lower precision for very large or very small numbers.

A positive posit number p is decoded as (J. L. Gustafson & Yonemoto, 2017; J. Gustafson, 2017; Chen et al., 2018; Klöwer et al., 2019)

$$p = (-1)^{sign\ bit} \cdot useed^k \cdot 2^e \cdot (1 + f). \tag{3}$$

k is the number of regime bits. e is the unsigned integer encoded by the exponent bits and f is the fraction which is represented in the fraction (or significant) bits. The base of the regime bits $useed = 2^{2^{e_s}}$ is determined by the number of exponent bits e_s. More exponent bits $e_s = 1, 2, 3, ...$ increase $useed = 4, 16, 256, ...$ and therefore widen the dynamic range of representable numbers but reduce the precision around ± 1. The exponent bits themselves fill gaps of powers of 2 spanned by $useed$ and so do not affect the dynamic range directly by changing the value of 2^e in Eq. 3. Consequently, every posit number, except for zero and NaR, can be written as $p = \pm 2^i \cdot (1 + f)$ for a given integer i. In the following, a posit number format with n bits including e_s exponent bits is denoted as Posit(n, e_s).

We provide an example in the Posit(8,1)-system (i.e. $useed = 4$), which encodes the number 14 as follows

$$01101110_{\text{Posit}(8,1)} = (-1)^0 \cdot 4^1 \cdot 2^1 \cdot (1 + 2^{-1} + 2^{-2}) = 14. \tag{4}$$

The sign bit is red, regime bits orange, the terminating regime bit brown, the exponent bit blue and the fraction bits are black. The k-value follows from the number of regime bits n_r. For regime bits being 0 $k = -n_r$, for regime bits being 1 $k = n_r - 1$, in this example $k = 1$. The exponent bits are interpreted as unsigned integer and the fraction bits follow the IEEE floating-point standard for significant bits.

In order to use posits on a conventional processor we developed for the Julia programming language (Bezanson et al., 2017) the posit emulator *SoftPosit.jl* (Klöwer & Giordano, 2019), which are bindings for the C-based library SoftPosit (Leong, 2020). A standardized posit processor is not yet available, but current research focuses on hardware implementations (?, ?; van Dam et al., 2019; Chen et al., 2018; Chaurasiya et al., 2018; Glaser et al., 2017).

2.2 The concept of decimal precision and summary of number formats

Most arithmetic operations include rounding of an exact result x_{exact} to a representable number x_{repr}. Based on the decimal rounding error $|\log_{10}(\frac{x_{\text{repr}}}{x_{\text{exact}}})|$ the decimal
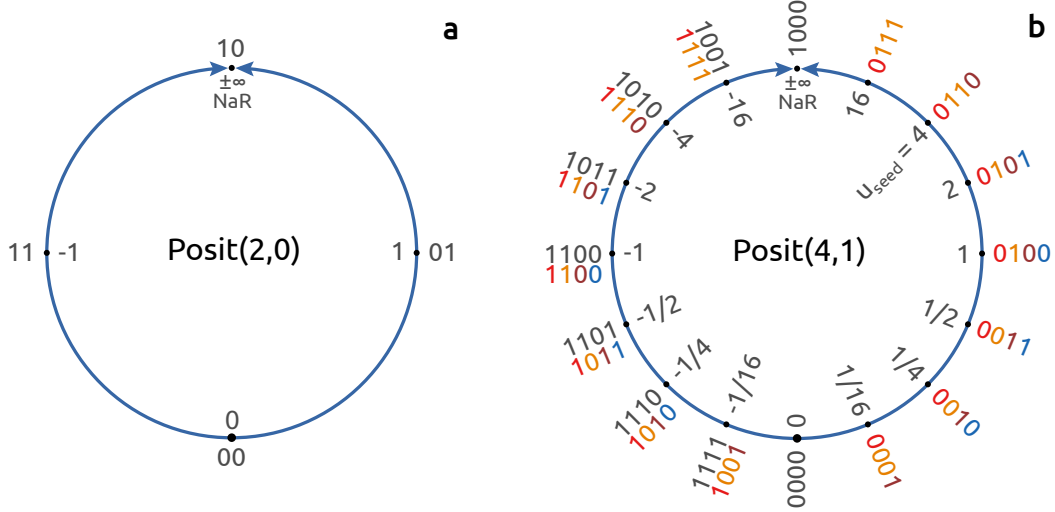
Figure 1. Two posit number formats obtained by projecting the real axis onto a circle. (a) 2-bit Posit(2,0) and (b) 4-bit Posit(4,1). The bit patterns are marked on the outside and the respective values on the inside of each circle. Bit patterns of negative numbers (black) have to be converted to their two's complement (colours) first (see text). At the top of every circle is complex infinity ($\pm\infty$) or NaR (Not-a-Real). After J. Gustafson (2017).

precision is defined as (J. L. Gustafson & Yonemoto, 2017; Klöwer et al., 2019)

$$\text{decimal precision} = -\log_{10}|\log_{10}(\frac{x_{\text{repr}}}{x_{\text{exact}}})|. \tag{5}$$

The decimal precision measures the number of correct decimal places after rounding. The decimal precision goes to infinity when the exact result aprroaches a representable number. Rounding an exact result half-way between two representable numbers maximises the decimal error and minimises the decimal precision. This minimum is the *worst-case* decimal precision, which measures the number of decimal places that are at least correct after round-to-nearest. In the following we will refer to the worst-case decimal precision simply as decimal precision. The machine epsilon ϵ, a relative rounding error in floating-point arithmetic, is commonly used to measure the precision of number formats. It is defined as the distance δ between 1 and the next largest representable number, and can be given in terms of decimal precision as $\epsilon = -\log_{10}(\log_{10}(1 + \frac{\delta}{2}))$.

The decimal precision for various 16 and 8-bit floats and posits, 16-bit integers and the fixed-point format Q6.10 (6 integer bits, 10 fraction bits) is presented in Fig. 2a. Floats have an exponent that is evenly spaced in logarithmic-space, which results in a nearly
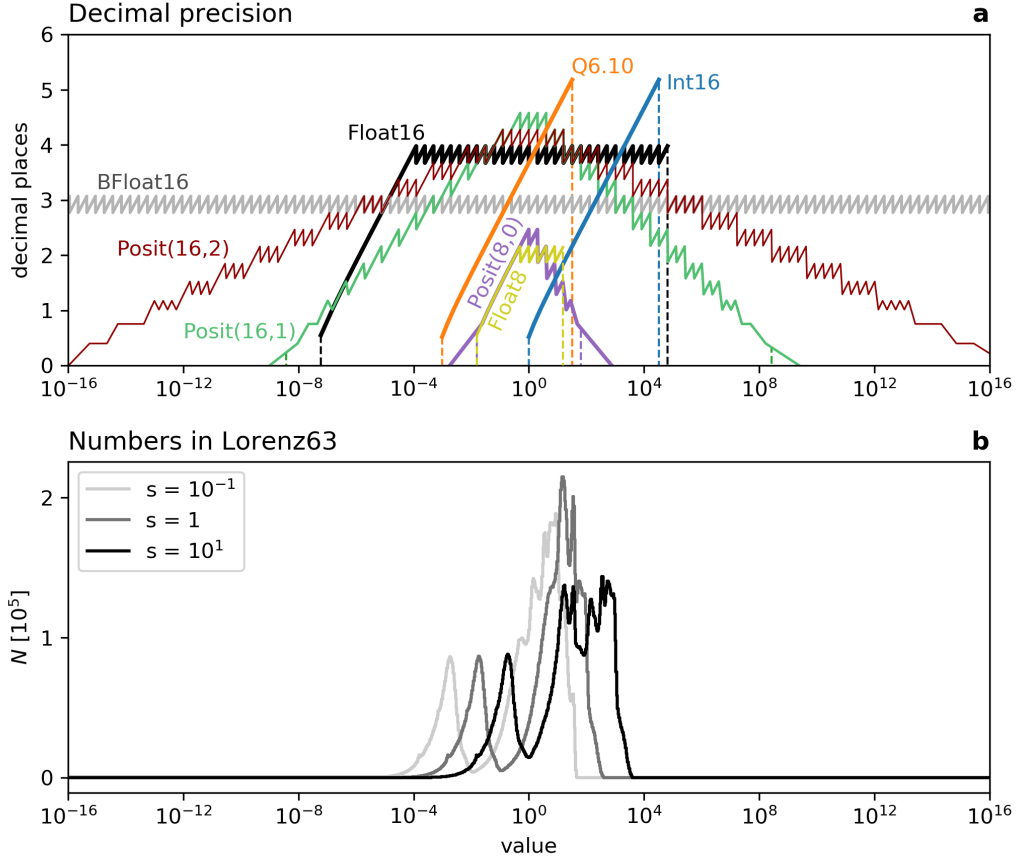
Figure 2. (a) Decimal precision for various number formats. Dashed vertical lines indicate for each format the range from *minpos* to *maxpos* of representable numbers. Float64, Float32 and Posit32 are beyond the axes limits. (b) Histogram of results of the absolute values of all arithmetic operations in Lorenz system rescaled by *s*.

constant decimal precision. The deviations from a constant decimal precision, which can be seen as regular spikes in Fig. 2a, are due to a linearly-spaced significand. The decimal precision for floats decreases for the subnormal numbers towards the smallest representable number *minpos*. Posits have an increased decimal precision around 1 and a yet a wide dynamic range, due to the tapered precision towards *minpos* and the largest representable number *maxpos*. The decimal precision for posits is above zero outside the dynamic range as posits have a no overflow/no underflow-rounding mode. Integers are linearly-spaced, consequently their precision increases towards *maxpos* for 16-bit signed integers. The decimal precision of fixed-point numbers has an identical shape, but is shifted towards smaller numbers by a factor of $\frac{1}{2}$ for each additional fraction bit. Consequently, many arithmetic calculations should be placed close to *maxpos*, but the small range of high precision is restrictive for many applications. No convincing results with integer or fixed-point arithmetic was achieved in this study, especially as the limited dynamic range of less than five orders of magnitude imposes a very difficult constrain to avoid under- or overflows.

Format	bits	exp bits	*minpos*	*maxpos*	ϵ	% NaR
Float64	64	11	$5.0 \cdot 10^{-324}$	$1.8 \cdot 10^{308}$	16.3	0.0
Float32	32	8	$1.0 \cdot 10^{-45}$	$3.4 \cdot 10^{38}$	7.6	0.4
Float16	16	5	$6.0 \cdot 10^{-8}$	65504	3.7	3.1
BFloat16	16	8	$9.2 \cdot 10^{-41}$	$3.4 \cdot 10^{38}$	2.8	0.4
Float8	8	3	$1.5 \cdot 10^{-2}$	15.5	1.9	12.5
Posit32	32	2	$7.5 \cdot 10^{-37}$	$7.5 \cdot 10^{37}$	8.8	0.0
Posit(16,1)	16	1	$3.7 \cdot 10^{-9}$	$3.7 \cdot 10^{9}$	4.3	0.0
Posit(16,2)	16	2	$1.4 \cdot 10^{-17}$	$1.4 \cdot 10^{17}$	4.0	0.0
Posit(8,0)	8	0	$1.5 \cdot 10^{-2}$	64	2.2	0.4
Int16	16	0	1	32767	0.8	0
Q6.10	16	0	$9.8 \cdot 10^{-4}$	32.0	3.7	0

Table 1. Characteristics of various number formats. *minpos* is the smallest representable positive number, *maxpos* the largest. The machine error ϵ is here given as decimal precision (i.e. correct decimal places after rounding). % NaR denotes the percentage of bit patterns that represent not a number (NaN), infinity or not a real (NaR).

Characteristics of various formats are summarised in Table 1. Float64 has more than 10^{15} bitpatterns reserved for NaN, but these only make up $< 0.05\%$ of all available bit patterns. However, the percentage of redundant bitpatterns for NaN increases for floats with fewer exponent bits and poses a noticable issue for Float16 and Float8. Every posit number format has only one bitpattern reserved for NaR, such that this issue is negligible for posits.

2.3 Mitigation measures to allow for the use of 16-bit arithmetic in weather and climate applications

2.3.1 Mixed precision arithmetic

In many models, it will not be possible (or useful) to use 16-bit arithmetic throughout the entire model. Some model components will be more sensitive to a reduction in precision when compared to others and it often makes sense to reduce precision only in those components where results are not deteriorated, while keeping precision high in precision-sensitive components. This approach is called *mixed-precision* and is already used for the reduction to single precision in ocean and atmosphere models (Váňa et al., 2017; Tintó Prims et al., 2019).

2.3.2 Algorithmic changes: Rescaling, reordering and precomputations

Equations can be *rescaled* via multiplication with a constant rescaling factor s to shift the dynamic range occurring in an algorithm towards larger numbers (for $s > 1$) or towards smaller numbers ($s < 1$). Rescaling can be used to adjust the number range to the decimal precision of the number formats. If non-linear terms are considered, this multiplicative rescaling is ineffective, as the rescaling factor s appears inside the non-linear terms. The non-linear terms are therefore effectively invariant under multiplicate rescaling and only the linear terms are scaled by s. Fig. 2 includes histograms of numbers occurring in the rescaled Lorenz system (Lorenz, 1963; Kwasniok, 2014; Jeffress et al., 2017; Tantet et al., 2018) as an example from Klöwer et al. (2019). For posit arithmetic it is preferable to use $s = \frac{1}{10}$ in the Lorenz system to scale the prognostic variables to match the range of highest decimal precision around ± 1, which increases the complexity of the Lorenz attractor and decreases the average rounding error.

Furthermore, it is sometimes possible to avoid intermediate arithmetic results, which may be outside the dynamic range of a number format, by changing the order in which multiplications and divisions are executed. In general, it is preferable to combine such operations to a single multiplication with a constant which can be precomputed. Although this will have a negligible effect on the rounding error for floating-point arithmetic due to the approximately constant decimal precision throughout the range of numbers (subnormals excluded), it reduces the risk of over- or underflow. Re-ordering the shallow water equations is discussed in section .

2.3.3 Reduced precision communication

Complex weather and climate models rely on parallelisation to distribute the computational cost of simulations efficiently among the processing units in a large cluster or supercomputer. Parallel execution typically requires domain decomposition, where the spatial domain is split into many subdomains to be calculated separately on individual processing units. Domain decomposition requires communication of the boundary values of a subdomain with the neighbouring subdomains. If 16-bit arithmetic cannot be used within the entire model, it may still be possible to reduce precision in the communication between processors.

Not all weather and climate models would benefit from a reduced precision communication as the acceleration potential depends on many factors specific to a model and the used hardware, e.g. number of nodes in a cluster and how shared and distributed memory is managed. It will also be important whether communication is latency or volume bound. Latency bound communication is bound by the time a package of information requires to travel between processors. In contrast, volume bound communication is limited by the bandwidth that is available for communication. Only the latter will benefit from a reduction in data volume, which can be achieved with reducing precision.

However, if communication volume is an identified bottleneck in a given application, which is often the case in weather and climate models, reliable model simulations might be possible with 16 or even 8-bit communication, allowing for a signficant reduction in computing time. In general, various lossy and lossless data compression techniques can be used to reduce communication volume (?, ?). We restrict ourselves to lossy type

conversions, which introduce likely a negligible overhead due to encoding before and decoding after communication.

3 A shallow water model with 16-bit arithmetic

In this section, the different number formats Float16, BFloat16, Posit(16,1) and Posit(16,2) are evaluated when solving the shallow water equations. The shallow water model usede here is an updated version of Klöwer et al. (2019), and most details described therein are also valid here. A vertical integration of the Navier-Stokes equations yields the shallow water equations, that can be used to understand many features of the general circulation of atmosphere and ocean as well as some two-dimensional non-linear interactions on shorter length and time scales (Gill, 1982; Vallis, 2006). The shallow water equations for the prognostic variables velocity $\mathbf{u} = (u, v)$ and sea surface elevation η are

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + f\hat{\mathbf{z}} \times \mathbf{u} = -g\nabla\eta + \mathbf{D} + \mathbf{F} \tag{6a}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (\mathbf{u}h) = 0. \tag{6b}$$

η can be interpreted as pressure for the atmosphere (Gill, 1982). The shallow water system is forced with a zonal wind stress $\mathbf{F}$. The dissipation term $\mathbf{D}$ removes energy on large scales (bottom friction) and on small scales (diffusion). The non-linear term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ represents advection of momentum. The term $f\hat{\mathbf{z}} \times \mathbf{u}$ is the Coriolis force and $-g\nabla\eta$ is the pressure gradient force, with g being the gravitational acceleration. Eq. 6b is the shallow water-variant of the continuity equation, ensuring conservation of mass.

The shallow water equations are solved in the (x, y)-plane over the zonally periodic rectangular domain $L_x \times L_y$, of size $2000\,\text{km} \times 1000\,\text{km}$. We associate x with the zonal and y with the meridional direction. The domain is centred at 45N with the beta-plane approximation (Vallis, 2006). The boundary conditions are periodic in zonal direction and no-slip at the northern and southern boundary. The layer thickness is $h = \eta + H(x)$, with

$$H(x) = H_0 - H_1 \exp\left(-H_\sigma^{-2}(x - \tfrac{L_x}{2})^2\right) \tag{7}$$

being the undisturbed depth, representing a meridional mountain ridge at $x = \frac{L_x}{2}$ spanning from the southern to the northern boundary. The standard depth is $H_0 = 500\,\text{m}$. The ridge has a maximum height of $H_1 = 50\,\text{m}$ and a characteristic width of $H_\sigma = 300\,\text{km}$,

308   which makes the zonal current barotropically unstable. The flow regime is therefore gov-

309   erned both by eddy-mean flow as well as eddy-eddy interactions.

310   The time step $\Delta t = 282\,\mathrm{s}$ is chosen to resolve surface gravity waves, traveling at

311   an estimated phase speed of $\sqrt{gH_0}$ with CFL number being close to 1 and gravitational

312   acceleration $g = 10\,\mathrm{ms^{-1}}$. The wind stress forcing $\mathbf{F} = (F_x, 0)$ is constant in time, acts

313   only on the zonal momentum budget

$$Fx = \frac{F_0}{\rho h} \cos\left(\pi \left(yL_y^{-1} - 1\right)\right)^2 \tag{8}$$

314   and vanishes at the boundaries. The water density is $\rho = 1000\,\mathrm{kg\,m^{-3}}$ and $F_0 = 0.12\,\mathrm{Pa}$.

315   The wind forcing acts as a contiunuous input of large-scale kinetic energy that is bal-

316   anced by the dissipation term.

317   The dissipation term $\mathbf{D}$ is the sum

$$\mathbf{D} = -\frac{c_D}{h}\|\mathbf{u}\|\mathbf{u} - \nu\nabla^4\mathbf{u} \tag{9}$$

318   of a quadratic bottom drag with dimensionless coefficient $c_D = 10^{-5}$ (Arbic & Scott,

319   2008) and a biharmonic diffusion with viscosity coefficient $\nu \approx 1.33\times10^{11}\,\mathrm{m^4\,s^{-1}}$ (Griffies

320   & Hallberg, 2000).

321   The shallow water equations are discretised using 2nd order centred finite differ-

322   ences on an Arakawa C-grid (Arakawa & Lamb, 1977) and the fourth order Runge-Kutta

323   method (Butcher, 2016) is used for time integration of the pressure, coriolis and advec-

324   tive terms, whereas a semi-implicit method is used for the dissipative terms $\mathbf{D}$. We present

325   results of simulations with three different levels of resolution: high resolution simulations

326   with a grid spacing of $\Delta = 5\,\mathrm{km}$ (400x200 grid points), medium resolution simulations

327   with a grid-spacing of $\Delta = 20\,\mathrm{km}$ (100x50 grid points) and low resolution with a grid-

328   spacing of $\Delta = 40\,\mathrm{km}$ (50x25 grid points). The advection terms are discretised using

329   an energy and enstrophy conserving scheme (Arakawa & Hsu, 1990).

330   The shallow water equations are extended with an advection equation for tracers.

331   Temperature and humidity or salinity are examples of tracers in the atmosphere and ocean.

332   For simplicity, we regard them as passive here, such that they do not influence the flow.

333   The advection of a passive tracer q given a velocity $\mathbf{u}$ is governed by

$$\frac{\partial q}{\partial t} + \mathbf{u}\cdot\nabla q = 0. \tag{10}$$

A semi-Lagrangian advection scheme (Smolarkiewicz & Pudykiewicz, 1992) is used to discretise Eq. 10. In this discretization, the tracer concentration for a given grid cell (i.e. the arrival point) is calculated from the concentration at a departure point at the previous time step. The departure point is determined from by the flow field by back-tracing the velocities from the arrival point. Departure points in general do not coincide with grid nodes, such that an interpolation from the surrounding grid points is required to find the concentration at the departure point, which is then directly the concentration at the arrival point.

3.0.1 Rescaling in the shallow water equations

The dynamic range of representable numbers with 16-bit arithmetic is for most formats discussed here considerably smaller than with Float32 or 64 (Fig. 2a and Table 1). It is therefore important to rescale the calculations to limit the range of arithmetic results to stay within the bounds of a given 16-bit format.

The prognostic variables in the shallow water equations are typically $\mathcal{O}(1\,\mathrm{ms}^{-1})$ for $\mathbf{u}$, $\mathcal{O}(1\,\mathrm{m})$ for η and $\mathcal{O}(1)$ for q. Their physical units are therefore retained in the discretised numerical model and we do not apply a rescaling of the shallow water equations as a whole. However, the grid spacing Δ in units of meter is large for geophysical flows. We therefore use dimensionless Nabla operators $\tilde{\nabla} = \Delta \nabla$. The continuity equation Eq. 6b, for example, reads then as

$$\eta^{n+1} = \eta^n + \tilde{\Delta}t \left(-\tilde{\nabla} \cdot (\mathbf{u}h)^n \right) \tag{11}$$

for an explicit time stepping scheme. $\tilde{\Delta}t$ is the rescaled time step, a Runge-Kutta coefficient times $\frac{\Delta t}{\Delta}$ to combine a division by a large value for Δ and a subsequent multiplication with a large value for Δt into a single multiplication with $\tilde{\Delta}t$. The other terms are rescaled accordingly ($\tilde{f} = f\Delta$; $\tilde{\mathbf{F}} = \mathbf{F}\Delta$). As these terms remain constant, they can be precomputed at higher precision during model initialisation. The momentum equations are rescaled similarly.

Diffusion is an example of a discretization scheme that requires rescaling for the arithmetic results to fit into the limited dynamic range. Biharmonic diffusion (Griffies & Hallberg, 2000) calculates a fourth derivative in space, which is often very small $\mathcal{O}(10^{-20})$ in geophysical applications, due to the large physical dimensions, when using meter as a unit for length. Contrarily, biharmonic viscosity coefficients are typically very large $\mathcal{O}(10^{11})$.

We therefore rescale $\mathbf{D}$ accordingly

$$\tilde{\mathbf{D}} = -\frac{\tilde{c_D}}{h}\|\mathbf{u}\|\mathbf{u} - \tilde{\nu}\tilde{\nabla}^4\mathbf{u} \tag{12}$$

with $\tilde{c_D} = c_D\Delta = 0.2\,\mathrm{m}$, and $\tilde{\nu} = \nu\Delta^{-3} \approx 0.16\,\mathrm{ms}^{-1}$, which are precomputed. The term $\tilde{\mathbf{D}}$ is computed instead of $\mathbf{D}$ and the scaling eventually undone when multiplying with the rescaled time step $\tilde{\Delta}t$.

The semi-Lagrangian advection scheme is reformulateed for 16-bit arithmetics. In the absence of sources and sinks, the Lagrangian point-of-view states that the tracer concentration q does not change following a flow trajectory. The concentration q at departure points $\mathbf{x}_d$ at time t is therefore the same as the concentration at time $t+\Delta t_{\mathrm{adv}}$ at arrival points $\mathbf{x}_a$, which are chosen to coincide with the grid points. Based on the flow velocity at the arrival point, the departure point is derived. To avoid large numbers for the coordinates (in our case $L_x = 2 \cdot 10^6\mathrm{m}$), non-dimensional departure points $\tilde{\mathbf{x}}_{d,rel}$ relative to the arrival point are computed as

$$\tilde{\mathbf{x}}_{d,rel} = -\mathbf{u}(\mathbf{x}_a, t + \Delta t_{\mathrm{adv}})\left(\frac{\Delta t_{\mathrm{adv}}}{\Delta}\right). \tag{13}$$

A scaling with the grid-spacing inverse Δ^{-1} is applied such that all terms are $\mathcal{O}(1)$ and therefore representable with 16-bit arithmetics. In practice, when converting the relative departure point $\tilde{\mathbf{x}}_{d,rel}$ to an array index for the interpolation, the floor function is used in combination with integer arithmetics. This essentially separates a computation with reals into two parts. One that can be computed with integers without rounding errors, and a calculation with reals, with a removed offset to reduce rounding errors.

3.0.2 *Shallow water simulations in 16-bit arithmetic*

The shallow water model simulates vigurous turbulence interacting with a zonal current (Fig. 3). Both float and posit arithmetic present very similar fluid dynamics in comparison to the Float64 reference in only 16 bit. A snapshot of tracer concentration many simulated days after initialisation reveals turbulent mixing of the tracer that is well simulated with posits. However, with Float16 the simulation deviates faster from the reference than with Posit(16,1) and to a lesser degree with Posit(16,2), presumably due to the small scale instabilities visible in the snapshot as wavy filaments and fronts. These instabilities are clearly triggered by Float16 arithmetics, but to a lower degree also visible for posits. This provides some visual evidence that accumulated rounding errors are
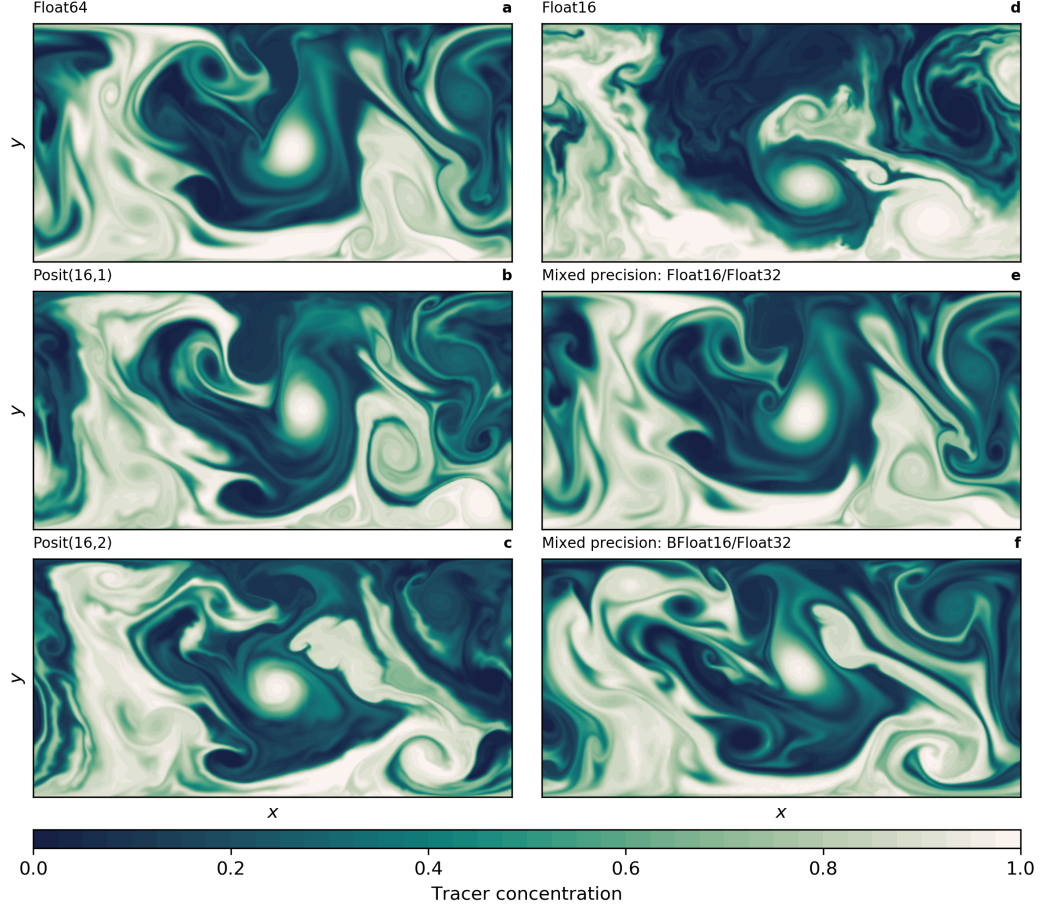
Figure 3. Snapshot of tracer concentration simulated by the shallow water model using different 16-bit number formats and the high-resolution configuration ($\Delta = 5$km). The mixed precision simulations presented in (e) and (f) are using Float32 for the representation of prognostic variables only. The tracer was injected uniformly in the lower half of the domain 50 simulation days before the time step shown.
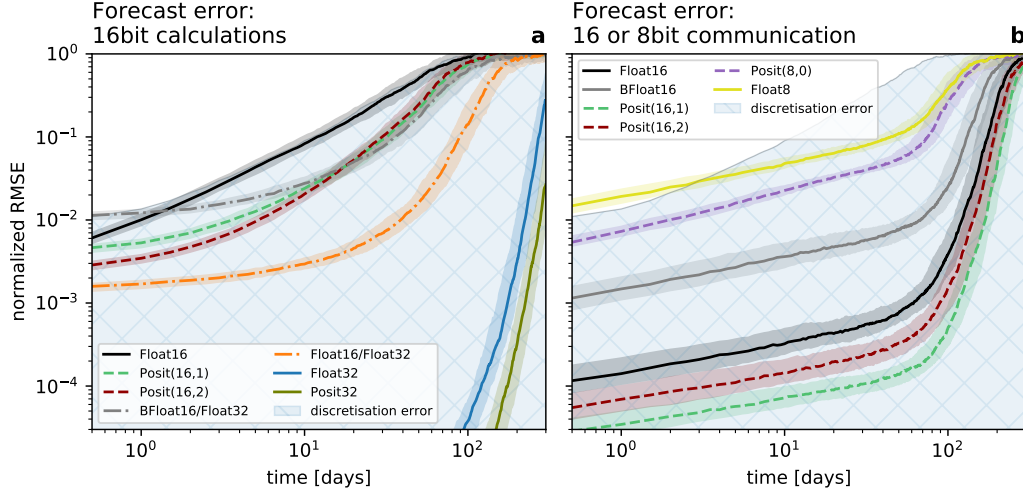
Figure 4. Forecast error of sea surface height η measured as root mean square error (RMSE) taking Float64 as reference. (a) Forecast error for various 16-bit number formats and mixed 16/32-bit simulations for which the prognostic variables are kept in Float32. (b) Forecast error for reduced precision communication in 8 or 16 bit with various number formats used for encoding, with Float64 used for all calculations. The communication of boundary values occurs at every time step for the prognostic variables. The RMSE is normalised by a mean forecast error at very long lead times. Solid lines represent the median of 200 forecasts per number format. The shaded areas of each model configuration denote the interquartile range of the forecast experiments.

reduced with posits, especially Posit(16,1). BFloat16 arithmetic is not able to simulate the shallow water dynamics, as tendencies are too small to be added to the prognostic variables. A stalling of the simulated flow is observed.

Short-term forecasts at medium-resolution ($\Delta = 10$km) are performed to analyse the differences between different 16-bit arithmetics. To quantify the error growth caused by rounding errors with different arithmetics in a statistically robust way, we create a number of forecasts with each member starting from one of 200 randomly picked start dates from a 50-year long control simulation. The forecast error in the shallow water model is computed as root mean square error (RMSE) of sea surface height η with respect to Float64 simulations. Other variables yield similar results. Each forecast is performed several times from identical initial conditions but with the various number formats. The error growth caused by rounding errors is additionally compared to the error introduced
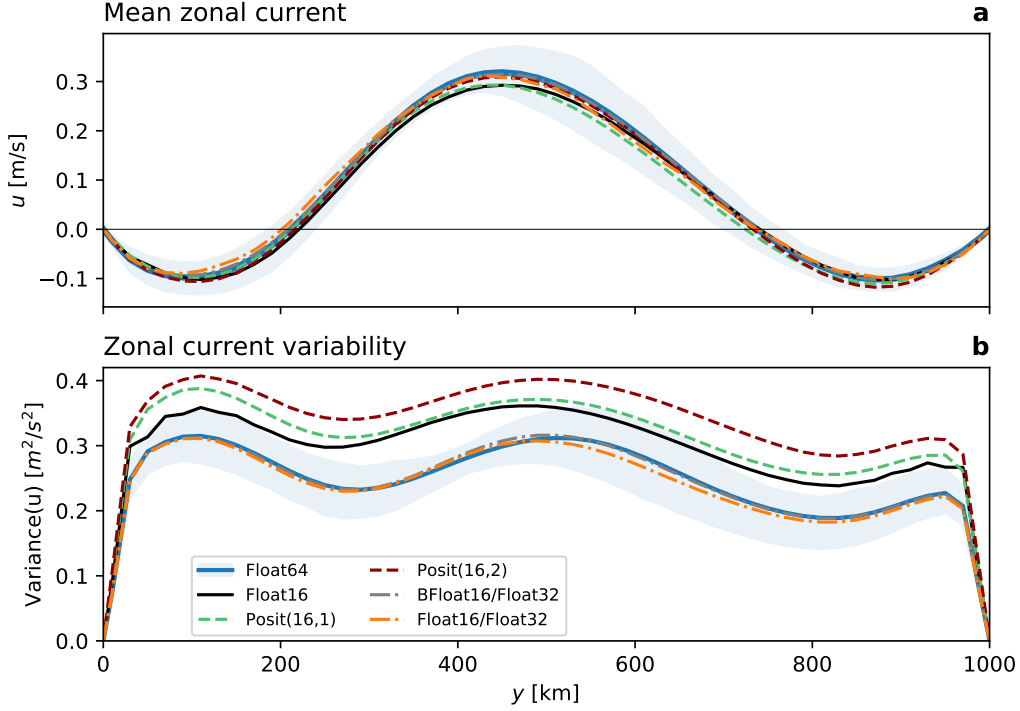
Figure 5. Climatology and variability of the zonal current in the medium-resolution simulations. (a) Zonally-averaged zonal current u as a function of the meridional coordinate y. (b) Zonal variance of the zonal current as a function of y. The shaded area denotes the interquartile temporal variability around the (a) mean and (b) variance of reference simulation with Float64.

by discretisation. A low-resolution model configuration with $\Delta = 20$km is used to quantify a realistic level of discretisation error. The RMSE is normalised by the climatological mean forecast error at very long lead times, which is the same for all model configurations. When the normalised RMSE reaches 1 all information on the initial conditions is removed by the chaotic evolution of the shallow water system.

The forecast error of Float16 is as large as the discretisation error and clearly outperformed by 16-bit posit arithmetic (Fig. 4a). Both Posit(16,1) and Posit(16,2) yield a forecast error that is several times smaller than Float16. The forecast error of 32-bit arithmetic is several orders of magnitude smaller and is only after 200 days as large as the error for 16-bit arithmetic at short lead times of about 10 days. Also at 32 bit, posits clearly outperform floats.
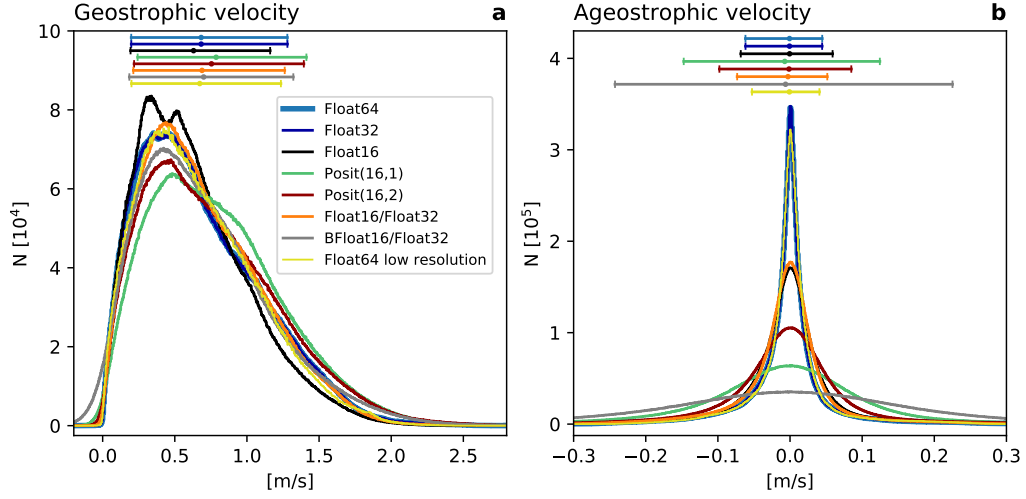
Figure 6. Geostrophic balance as simulated with different number formats. (a) Histograms of flow-parallel components of geostrophic velocity. (b) as (a) but for the ageostrophic velocities. Horizontal bars denote the mean, 10th and 90th-percentile in respective colours.

To investigate the effect of rounding errors on the climatological mean state of the shallow water system, we zonally average the zonal velocity u. The mean state is an eastward flow of about 0.3 m/s, about 3 to 4 times weaker than individual velocities throughout the domain (Fig. 5a), which is typical for turbulent flows. A weak westward mean flow is found at the northern and southern boundary. No 16-bit format was found to have a significant impact on the mean state. The variability of the flow around its mean state is high throughout the domain (Fig. 5b). The variability is significantly increased by 10 − 30% with 16-bit arithmetic, especially with Posit(16,2). This is probably caused by rounding errors that are triggering local perturbations which increase variability.

The turbulence in shallow water simulations is largely geostrophic, such that the pressure gradient force opposes the Coriolis force. The resulting geostrophic velocities $\mathbf{u}_g$ can be derived from the sea surface height η as

$$\mathbf{u}_g = \frac{g}{f}\hat{\mathbf{z}} \times \nabla\eta \tag{14a}$$

$$\mathbf{u} = \mathbf{u}_g + \mathbf{u}_{ag} \tag{14b}$$

and deviations from the actual flow $\mathbf{u}$ are the ageostrophic velocity components $\mathbf{u}_{ag}$. We project both components on the actual velocities to obtain the flow-parallel components

429 $\tilde{u}_g$ and $\tilde{u}_{ag}$ via

$$\tilde{u}_g = \frac{\mathbf{u}_g \cdot \mathbf{u}}{\|\mathbf{u}\|}, \quad \tilde{u}_{ag} = \frac{\mathbf{u}_{ag} \cdot \mathbf{u}}{\|\mathbf{u}\|}. \tag{15}$$

430 The geostrophic velocities in the shallow water simulations can reach up to 2 m/s, are

431 hardly negative (i.e. against the flow) and have a mean of about 0.7 m/s (Fig. 6a). This

432 behaviour is well simulated with 16-bit number formats, although posits increase the strength

433 of geostrophic velocities slightly. Ageostrophic velocity components are found to be isotropic,

434 and are oriented equally frequent with and against the prevailing flow. They rarely ex-

435 ceed ± 0.1 m/s and are therefore comparably small, which is expected in geostrophically

436 balanced turbulence. Ageostrophic velocities can be seen as a measure of the physical

437 instabilities in the flow field and their variance is indeed increased when simulated with

438 16-bit number formats. Float16 and posits show clearly fewer ageostrophic velocities around

439 0, pointing towards an increased number of simulated instabilities. Especially Posit(16,1)

440 increases the variance of ageostrophic velocities by more than a factor of two. It is un-

441 clear where in the model integration rounding errors of 16-bit arithmetic trigger insta-

442 bilities that lead to the observed increase in ageostrophy. We conclude that although the

443 geostrophic balance in the simulations is maintained, rounding errors lead, likely due to

444 an increase in ageostrophy, to a higher variability in the flow field.

445 As 16-bit arithmetics have no significant impact on the climatological mean state,

446 histograms of prognostic variables are also not changed (Fig. 7a and b). However, the

447 tendencies are increased by orders of magnitude with 16-bit arithmetics (Fig. 7d and e),

448 as rounding errors cause gravity waves to radiate away from eddies (Fig. 7f). Gravity

449 waves are identified from the tendency of sea surface height. Comparing their propaga-

450 tion to the location of anomalous sea surface height, which is used as a proxy to locate

451 eddies, we assume that rounding errors in regions of high eddy activity lead to instabil-

452 ities that propagate away in the form of gravity waves. These gravity waves are not present

453 in Float64 simulations (Fig. 7c) and tend to have only a small impact on quasi-gestrophic

454 dynamics, as they act on different time and length scales. It is unclear but possible that

455 gravity waves cause the observed increased ageostrophic velocities for 16-bit arithmetic.

456 Tendencies are about 4 orders of magnitude smaller than the prognostic variables.

457 This poses a problem for number formats with a machine epsilon, measured as decimal

458 precision, significantly lower than 4 decimal places (Table 1). Float16 has a machine ep-

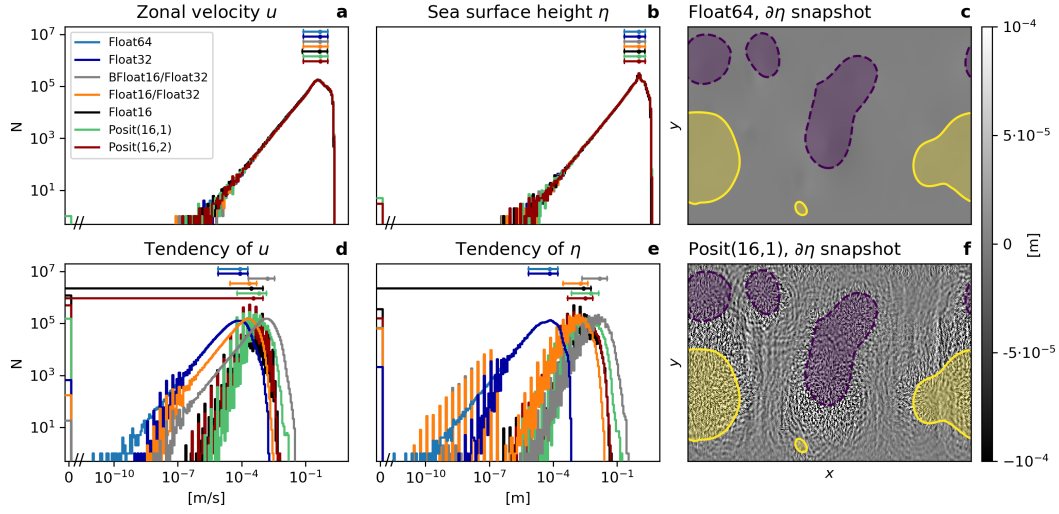459 silon of 3.7, which is presumably close to the lower limit beyond which the addition of

Figure 7. Histograms of the numeric values of the prognostic variables (a) zonal velocity u, (b) sea surface height η, and the respective tendencies of (d) u and (e) η, simulated with different 16, 32 and 64-bit number formats. Mean, 10th and 90th percentile are shown above the histograms in respective colors. Snapshots of the tendencies of η simulated with (c) Float64 and (f) Posit(16,1). Snapshots are similar for other 16-bit formats (not shown here). Areas of sea surface height anomalies exceeding ± 1.4 m are shown in purple (negativ) and yellow (positive). Note the break on the x-axis close to zero in (a,b,d) and (e).

460 tendencies will be round back. The BFloat16 number format has a machine epsilon of

461 2.8, which explains why flow is stalling when simulated with BFloat16.

3.0.3 *Mixed precision arithmetic in the shallow water model*

463   In the previous simulations the entire shallow water simulation was performed with

464 the specified number format. As the addition of tendencies to the prognostic variables

465 was identified as a key calculation that is error-prone, we investigate now the benefits

466 of mixed precision arithmetic, where Float32 is used for the prognostic variables but the

467 tendencies are computed with either Float16 or BFloat16, two number formats that have

468 the lowest decimal precision around 1. The prognostic variables are now reduced to Float16

469 or BFloat16 before calculations of the right-hand side and every term of the tendencies

470 is converted back before addition to the prognostic variables. Using subscripts 16 and

471 32 to denote variables held at 16 and 32-bit precision, respectively, and let Float32() be

472 the conversion function. The continuity equation (Eq. 6b) then becomes

$$\frac{\partial \eta_{32}}{\partial t} = -\,\mathrm{Float32}(\partial_x(u_{16}h_{16}) + \partial_y(v_{16}h_{16})) \tag{16}$$

473 and similar for u and v in Eq. 6a.

474   Snapshots of tracer concentration reveal well simulated geostrophic turbulence (Fig.

475 3e and f) with Float16/Float32 or BFloat16/Float32 and instabilites at fronts or in fil-

476 aments are visibly reduced compared to pure 16-bit arithmetic. The forecast error is strongly

477 reduced once the prognostic variables are kept as Float32 (Fig. 4a), supporting the hy-

478 pothesis that the addition of tendencies to the prognostic variables is a key computa-

479 tion with low rounding error-tolerance. Despite BFloat16 not being suitable for shallow

480 water simulations when applied to all computations, mixing BFloat16 with Float32 arith-

481 metic yields a similar error growth to posits, which is well below the discretization er-

482 ror. Mean state or variability are virtually identical for both mixed precision cases (Fig.

483 5) compared to the Float64 reference. The geostrophic balance is largely unaffected, but

484 ageostrophic velocities increase in variance, especially for BFloat16 (Fig. 6). Gravity waves

485 are similarly present for mixed precision although weaker for tendencies computed with

486 Float16 (Fig. 7d) and, as discussed, they tend to not interact with the geostrophic time

487 and length scales. Although the results show that Float16 is generally a preferrable num-

488 ber format over BFloat16 for the applications presented here, we acknowledge that the

489 conversion between Float32 and Float16 will come with some computational cost. In con-

trast, the conversion between BFloat16 and Float32 is computationally very cheap as both formats have the same number of exponent bits. Removing significant bits, applying rounding, and padding trailing zeros, are the only operations for this conversion. Following the results here, mixing 16 and 32-bit precision is found to be an attractive solution to circumvent spurious behaviour due to 16-bit floating-point arithmetics. Performance benefits are still possible as most calculations are performed with 16 bit, with error-critical computations in 32 bit to reduce the overall error. The conversions between number formats are likely of negligible cost. This is an attractive solution as hardware-accelerated 16-bit floating-point arithmetic is already available on graphic or tensor processing units and implementations therefore do not rely on the development of future computing hardware, which is the case for posits.

3.0.4 *Reduced precision communication for the shallow water model*

A standard method to parallelise simulations is the distributed-memory parallelism via Message Passing Interface (MPI). We emulate MPI-like communication in the shallow water model with the copying of boundary values between the right and left boundary (periodic boundary conditions). Although the shallow water model does not run in parallel, reducing the precision in the copying of boundary values introduces an equivalent error as if reduced precision MPI communication was used between subdomains. Reduced precision is applied for the communication of the prognostic variables at every Runge-Kutta substep.

Regarding snapshots of tracer concentration simulated with reduced precision communication show a negligible error for Float16 and posits (Fig. 8). The error is largest at fronts and not concentrated around the boundaries. Encoding the communication with BFloat16 introduces a larger error than for the other 16-bit formats as the decimal precision is with 2.8 clearly lower (Table 1) for the range of values occuring within the prognostic variables (Fig. 7a and b). The errors are quantified by the RMSE of surface height η as before and are up to about two orders of magnitude smaller than the errors that result from 16-bit arithmetic. As even the worst 16-bit communication format, BFloat16, has a smaller error than the best mixed precision formats, Float16 with Float32, we extend the short-term forecast experiments to include two 8-bit formats, Posit(8,0) and Float8 (see Table 1 for a description). Both formats are found to be suitable for reduced precision communication here and do not introduce an error that is larger than the discretiza-
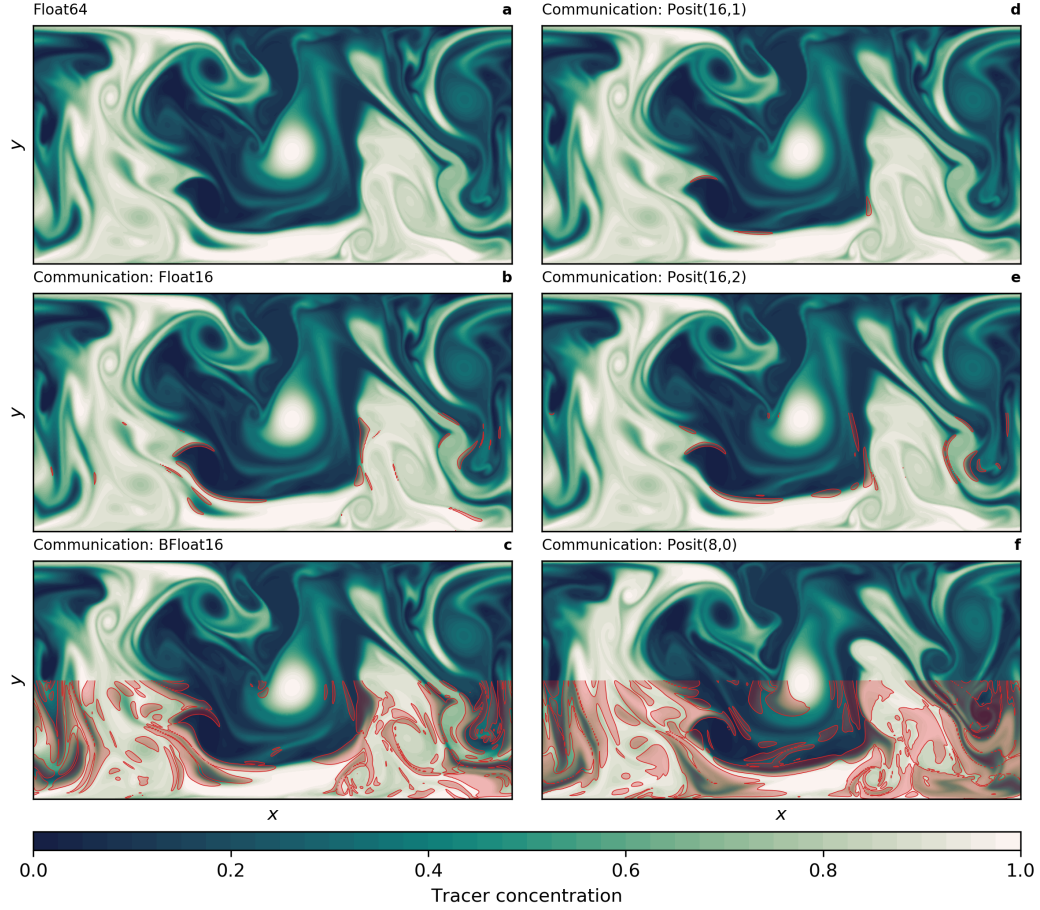
Figure 8. Snapshot of tracer concentration simulated by the shallow water model using reduced precision communication. The communication of boundary values occurs at every time step for the prognostic variables. Float64 was used for all calculations. Areas where the absolute error exceeds 0.05 are shaded in red only in the lower half of the domain. The tracer was injected uniformly in the lower half of the domain 50 days before. This simulation was run in the high-resolution configuration ($\Delta = 5$km).

tion error. Having said that, Float8 communication introduces an error that is comparably large in the first days but growths only linearly in the first 50 days of the simulation, which is in contrast to the exponential error growth observed for 16-bit arithmetic.

4 Conclusion and Discussion

Future high performance computing architecture will support some 16-bit arithmetics. The wall-clock time for weather and climate simulations could be greatly reduced if computationally demanding algorithms were run at such reduced precision. We tested a number of options for 16-bit arithmetic for weather and climate applications in a shallow water model. The best results were achieved with 16-bit posits (with either 1 or 2 exponent bits) which appear very promising for application in high performance computing for Earth System modelling. Float16 can be used to perform forecasts with the shallow water model while the application of BFloat16 or integer arithmetic was not successful.

In general, 16-bit arithmetics were not found to alter the climatological mean state or the large-scale dynamics. However, variability and ageostrophic velocities were increased, such that second and higher-order statistics should undergo testing to assess the models reliability. Depending on the application, an increased variability does not necessarily deteriorate the model, especially for more realistic model set-ups than considered here. However, our findings suggest that reduced precision changes need to be done carefully as specific simulation features can change without obvious impact on mean diagnostics.

Shallow water simulations with 16-bit arithmetic required rescaling of some terms but no major revisions of the model code or algorithms. Given that only floats are currently hardware-supported, we investigated mixed precision approaches. Keeping the prognostic variables at 32 bit while computing the tendencies in 16 bit reduced the rounding errors significantly. We also showed that numerical precision for communication between compute nodes can be greatly reduced down to 16 or even 8-bit without introducing a large error. Reduced precision communication was not found to have a significant impact on either mean state, variability, geostrophy or tendencies.

A *perfect model* is used in this study, such that any form of model or initial condition error is ignored and only the number format is changed between simulations. Solely discretisation errors are estimated by lowering the spatial resolution by a factor of 2. Al-

though this is essential here to analyse the impact of rounding errors isolated from other errors, it is in general not a realistic configuration for weather or climate models. More complex models include many other sources of forecast error, such that the contribution of rounding errors from 16-bit arithmetic would likely be dwarfed by model, discretisation or initial condition errors.

Only the most common discretisation method for fluid dynamics was used in this study: Finite differences with an explicit time stepping scheme. But various other discretisation methods exist, such as finite element or volume, spectral methods and implicit time stepping. These methods come with different algorithms and associated precision requirements. Consequently, some might be less tolerant to rounding errors than the method used in this study.

As there is no hardware available for posit arithmetic that we could have used for performance testing, we cannot draw any conclusion about the performance of posit arithmetic operations in comparison to Float16 or the other formats. Until progress is made on hardware implementations for posits, the results here suggest that also 16-bit float arithmetic can succesfully be used for parts of complex weather and climate models with the potential for acceleration on graphic and tensor processing units. It is therefore recommended to adapt a type-flexible programming paradigm, ideally in a language that supports portability, with algorithms written to reduce the dynamic range of arithmetic results. Hardware progress on central, graphic or tensor processing units, with various numbers formats supported, can subsequently be utilised to accelerate weather and climate simulations.

Data and scripts will be available at `https://github.com/milankl/ClimateModels16bit` upon acceptance. Relevant software is already available via `https://github.com/milankl` in the following repositories: SoftPosit.jl, ShallowWaters.jl, and Float8s.jl.

References

Arakawa, A., & Hsu, Y.-J. G. (1990, October). Energy Conserving and Potential-Enstrophy Dissipating Schemes for the Shallow Water Equations. *Monthly Weather Review*, *118*(10), 1960–1969. doi: 10.1175/1520-0493(1990)118⟨1960:ECAPED⟩2.0.CO;2

Arakawa, A., & Lamb, V. R. (1977, January). Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model. In J. Chang (Ed.), *Methods in Computational Physics: Advances in Research and Applications* (Vol. 17, pp. 173–265). Elsevier. doi: 10.1016/B978-0-12-460817-7.50009-4

Arbic, B. K., & Scott, R. B. (2008, January). On Quadratic Bottom Drag, Geostrophic Turbulence, and Oceanic Mesoscale Eddies. *Journal of Physical Oceanography*, *38*(1), 84–103. doi: 10.1175/2007JPO3653.1

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017, January). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, *59*(1), 65–98. doi: 10.1137/141000671

Burgess, N., Milanovic, J., Stephens, N., Monachopoulos, K., & Mansell, D. (2019, June). Bfloat16 Processing for Neural Networks. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)* (pp. 88–91). doi: 10.1109/ARITH.2019.00022

Butcher, J. C. (2016). *Numerical Methods for Ordinary Differential Equations* (3edition ed.). Chichester, West Sussex, United Kingdom: Wiley.

Chaurasiya, R., Gustafson, J., Shrestha, R., Neudorfer, J., Nambiar, S., Niyogi, K., . . . Leupers, R. (2018, October). Parameterized Posit Arithmetic Hardware Generator. In *2018 IEEE 36th International Conference on Computer Design (ICCD)* (pp. 334–341). Orlando, FL, USA: IEEE. doi: 10.1109/ICCD.2018.00057

Chen, J., Al-Ars, Z., & Hofstee, H. P. (2018). A matrix-multiply unit for posits in reconfigurable logic leveraging (open)CAPI. In *Proceedings of the Conference*

for Next Generation Arithmetic on - CoNGA '18 (pp. 1–5). Singapore, Singapore: ACM Press. doi: 10.1145/3190339.3190340

Dawson, A., & Düben, P. D. (2017, June). Rpe v5: An emulator for reduced floating-point precision in large numerical simulations. *Geoscientific Model Development*, *10*(6), 2221–2230. doi: 10.5194/gmd-10-2221-2017

Düben, P. D. (2018, November). A New Number Format for Ensemble Simulations. *Journal of Advances in Modeling Earth Systems*, *10*(11), 2983–2991. doi: 10.1029/2018MS001420

Düben, P. D., & Palmer, T. N. (2014, July). Benchmark Tests for Numerical Weather Forecasts on Inexact Hardware. *Monthly Weather Review*, *142*(10), 3809–3829. doi: 10.1175/MWR-D-14-00110.1

Gill, A. E. (1982). *Atmosphere-ocean dynamics* (No. 30). San Diego: Acad. Press. (OCLC: 249294465)

Glaser, F., Mach, S., Rahimi, A., Gürkaynak, F. K., Huang, Q., & Benini, L. (2017, December). An 826 MOPS, 210 uW/MHz Unum ALU in 65 nm. *arXiv:1712.01021 [cs]*.

Griffies, S. M., & Hallberg, R. W. (2000). Biharmonic Friction with a Smagorinsky-Like Viscosity for Use in Large-Scale Eddy-Permitting Ocean Models. *MONTHLY WEATHER REVIEW*, *128*, 12.

Gupta, S., Agrawal, A., Gopalakrishnan, K., & Narayanan, P. (2015, February). Deep Learning with Limited Numerical Precision. *arXiv:1502.02551 [cs, stat]*.

Gustafson, J. (2017). *Posit Arithmetic.*

Gustafson, J. L., & Yonemoto, I. (2017). Beating Floating Point at its Own Game: Posit Arithmetic. *Supercomputing Frontiers and Innovations*, *4*(2), 16.

Hatfield, S., Düben, P., Chantry, M., Kondo, K., Miyoshi, T., & Palmer, T. (2018, September). Choosing the Optimal Numerical Precision for Data Assimilation in the Presence of Model Error. *Journal of Advances in Modeling Earth Systems*, *10*(9), 2177–2191. doi: 10.1029/2018MS001341

Jeffress, S., Düben, P., & Palmer, T. (2017, September). Bitwise efficiency in chaotic models. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *473*(2205), 20170144. doi: 10.1098/rspa.2017.0144

Jouppi, N. P., Young, C., Patil, N., & Patterson, D. (2018, August). A domain-specific architecture for deep neural networks. *Communications of the ACM*,

648   *61*(9), 50–59. doi: 10.1145/3154484

649   Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... Yoon,

650   D. H. (2017, June). In-Datacenter Performance Analysis of a Tensor Pro-

651   cessing Unit. In *Proceedings of the 44th Annual International Symposium on*

652   *Computer Architecture* (pp. 1–12). Toronto, ON, Canada: Association for

653   Computing Machinery. doi: 10.1145/3079856.3080246

654   Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., ...

655   Dubey, P. (2019, June). A Study of BFLOAT16 for Deep Learning Training.

656   *arXiv:1905.12322 [cs, stat]*.

657   Klöwer, M., Düben, P. D., & Palmer, T. N. (2019). Posits as an alternative to floats

658   for weather and climate models. In *Proceedings of the Conference for Next*

659   *Generation Arithmetic 2019 on - CoNGA'19* (pp. 1–8). Singapore, Singapore:

660   ACM Press. doi: 10.1145/3316279.3316281

661   Klöwer, M., & Giordano, M. (2019, December). *SoftPosit.jl - A posit arithmetic em-*

662   *ulator.* Zenodo. doi: 10.5281/zenodo.3590291

663   Kwasniok, F. (2014, June). Enhanced regime predictability in atmospheric low-order

664   models due to stochastic forcing. *Philosophical Transactions of the Royal Soci-*

665   *ety A: Mathematical, Physical and Engineering Sciences*, *372*(2018), 20130286.

666   doi: 10.1098/rsta.2013.0286

667   Langroudi, H. F., Carmichael, Z., & Kudithipudi, D. (2019, July). Deep Learning

668   Training on the Edge with Low-Precision Posits. *arXiv:1907.13216 [cs, stat]*.

669   Leong, S. H. (2020, March). *SoftPosit.* Zenodo. (Language: eng) doi: 10.5281/

670   zenodo.3709035

671   Lorenz, E. N. (1963, March). Deterministic Nonperiodic Flow. *Journal of the Atmo-*

672   *spheric Sciences*, *20*(2), 130–141. doi: 10.1175/1520-0469(1963)020⟨0130:DNF⟩

673   2.0.CO;2

674   Markidis, S., Chien, S. W. D., Laure, E., Peng, I. B., & Vetter, J. S. (2018, May).

675   NVIDIA Tensor Core Programmability, Performance Precision. In *2018*

676   *IEEE International Parallel and Distributed Processing Symposium Workshops*

677   *(IPDPSW)* (pp. 522–531). doi: 10.1109/IPDPSW.2018.00091

678   Palmer, T. (2015, October). Modelling: Build imprecise supercomputers. *Nature*

679   *News*, *526*(7571), 32. doi: 10.1038/526032a

680   Palmer, T. N. (2012, April). Towards the probabilistic Earth-system simulator:

A vision for the future of climate and weather prediction. *Quarterly Journal of the Royal Meteorological Society*, *138*(665), 841–861. doi: 10.1002/qj.1923

Rüdisühli, S., Walser, A., & Fuhrer, O. (2013). COSMO in single precision. *Cosmo Newsletter*(14), 5–1.

Russell, F. P., Düben, P. D., Niu, X., Luk, W., & Palmer, T. (2017, December). Exploiting the chaotic behaviour of atmospheric models with reconfigurable architectures. *Computer Physics Communications*, *221*, 160–173. doi: 10.1016/j.cpc.2017.08.011

Smolarkiewicz, P. K., & Pudykiewicz, J. A. (1992, November). A Class of Semi-Lagrangian Approximations for Fluids. *Journal of the Atmospheric Sciences*, *49*(22), 2082–2096. doi: 10.1175/1520-0469(1992)049⟨2082:ACOSLA⟩2.0.CO;2

Tantet, A., Lucarini, V., & Dijkstra, H. A. (2018, February). Resonances in a Chaotic Attractor Crisis of the Lorenz Flow. *Journal of Statistical Physics*, *170*(3), 584–616. doi: 10.1007/s10955-017-1938-0

Thornes, T., Düben, P., & Palmer, T. (2017, January). On the use of scale-dependent precision in Earth System modelling: Scale-Dependent Precision in Earth System Modelling. *Quarterly Journal of the Royal Meteorological Society*, *143*(703), 897–908. doi: 10.1002/qj.2974

Tintó Prims, O., Acosta, M. C., Moore, A. M., Castrillo, M., Serradell, K., Cortés, A., & Doblas-Reyes, F. J. (2019, July). How to use mixed precision in ocean models: Exploring a potential reduction of numerical precision in NEMO 4.0 and ROMS 3.6. *Geoscientific Model Development*, *12*(7), 3135–3148. doi: https://doi.org/10.5194/gmd-12-3135-2019

Vallis, G. K. (2006). *Atmospheric and Oceanic Fluid Dynamics*. Cambridge University Press.

van Dam, L., Peltenburg, J., Al-Ars, Z., & Hofstee, H. P. (2019, March). An Accelerator for Posit Arithmetic Targeting Posit Level 1 BLAS Routines and Pair-HMM. In *Proceedings of the Conference for Next Generation Arithmetic 2019* (pp. 1–10). Singapore, Singapore: Association for Computing Machinery. doi: 10.1145/3316279.3316284

Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., & Carver, G. (2017, February). Single Precision in Weather Forecasting Models: An

714   Evaluation with the IFS.   *Monthly Weather Review*, *145*(2), 495–502.   doi:

715   10.1175/MWR-D-16-0228.1