Towards 16bit weather and climate models: Posits as an alternative to floats

Milan Klöwer¹, Peter Düben², Tim Palmer¹

¹Atmospheric, Oceanic and Planetary Physics, University of Oxford, Oxford, UK
²European Centre for Medium-Range Weather Forecasts, Reading, UK
Weather forecasting: Where are we?

Goal for the next decades
- ~1km (cloud resolving)

Requires exascale supercomputers

Forecast skill mainly improved through
- Satellite data
- Better models
- Faster supercomputers

Strongly linked to horizontal resolution:
- 9km (2019)
- 25km (2009)

Forecast skill at the European Weather Centre (UK)

Forecast skill mainly improved through
- Satellite data
- Better models
- Faster supercomputers

Strongly linked to horizontal resolution:
- 9km (2019)
- 25km (2009)

Goal for the next decades
- ~1km (cloud resolving)

Requires exascale supercomputers
1. Initial and boundary condition errors (observations, data assimilation, climate change)
2. Model error (approximations in equations of motion; clouds, precipitation, radiation, ...)
3. Discretisation error (finite spatial & temporal resolution)
4. Rounding error of double precision floats

Allow for bigger rounding errors they will be dwarfed by other errors anyway!
Weather forecasting in single precision

Single precision yields almost identical results in ~60% of runtime.

Next step: 16bit?

Half precision floats bfloat16

From Váňa et al, 2017
Reduced precision software emulator

\[ c = \mathcal{R}_{10}(a + b), \quad a, b, c :: \text{DOUBLE} \]

From Dawson & Düben, 2017

Allowing testing of weather and climate models without specialised hardware.
Software emulation: Reduced precision

Posits for weather and climate models
CoNGA’19
Milan Klöwer

Hurricane Sandy
27/10/12 00:00
850hPa wind speed
T255L91 ~ 80km

8sbits not a representative potential for the whole model!

Double precision 52sbits
Single precision 23sbits

Grid point calculations
Spectral transform
Spectral inverse
Linear terms in reduced precision

From Matthew Chantry

8 significant bits
The complexity of a weather/climate model

Cost (% of CPU time)

<table>
<thead>
<tr>
<th>Cost Component</th>
<th>9km (operational)</th>
<th>5km (goal for 2025)</th>
<th>1.25km (no ocean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>24%</td>
<td>2%</td>
<td>2%</td>
</tr>
<tr>
<td>Legendre (Mat x Mat)</td>
<td>26%</td>
<td>14%</td>
<td>20%</td>
</tr>
<tr>
<td>MPI comm</td>
<td>9%</td>
<td>31%</td>
<td>41%</td>
</tr>
<tr>
<td>Elliptic solver, Multi-grid precon</td>
<td>30%</td>
<td>38%</td>
<td>38%</td>
</tr>
<tr>
<td>Finite difference, Semi-implicit solver, Interpolations, MPI comm</td>
<td>3%</td>
<td>3%</td>
<td>2%</td>
</tr>
<tr>
<td>Cubic interpolation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Communication volume

<table>
<thead>
<tr>
<th>Cost Component</th>
<th>IFS (spectral), global MPI comm</th>
<th>DG (grid point), local MPI comm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>427 TB</td>
<td>34 TB</td>
</tr>
<tr>
<td>MPI procs</td>
<td>2880</td>
<td>2880</td>
</tr>
<tr>
<td>Runtime</td>
<td>12min</td>
<td>4h</td>
</tr>
</tbody>
</table>
A posit of posits?

Posits for weather and climate models
CoNGA’19
Milan Klöwer

Posits: A number format much better suited for weather and climate models?
Figure 2.3: Example use of the posit emulator `SigmoidNumbers` in the Julia shell.

```julia
julia> # define posit environment
julia> using SigmoidNumbers
julia> Posit161 = Posit{16,1};
julia> # convert float to 16bit posit, add
julia> a = Posit161(12.3);
julia> c = a+a;
julia> # bits split in sign, regime, exponent and fraction
julia> bits(c," ")
"0 1110 0 1000100110"
julia> Float64(c)  # convert back to double
24.59375
```
The simplest chaotic system: Lorenz 1963

\[
\begin{align*}
\dot{x} &= \sigma (y - x) \\
\dot{y} &= x(\rho - z) - y \\
\dot{z} &= xy - \beta z
\end{align*}
\]

Rescale the system:

\[
\hat{x} = s x
\]

Posits for weather and climate models
CoNGA’19
Milan Klöwer
Decimal precision

Reduced precision modelling:
• Rescale equations to fit number system (limited)
• Find number system that fits computed numbers

What would be the optimal decimal precision distribution?

\[
\text{decimal precision} = -\log_{10} \left| \log_{10} \left( \frac{x_{\text{repr}}}{x_{\text{exact}}} \right) \right|
\]
Shallow water equations I

\[
\frac{\partial u}{\partial t} + (u \cdot \nabla) u + f \hat{z} \times u = -g \nabla \eta - \nu \nabla^4 u - ru + F
\]

Conservation of momentum

\[
\frac{\partial \eta}{\partial t} + \nabla \cdot (uh) = 0.
\]

Conservation of mass

\[
\frac{\partial q}{\partial t} + u \cdot \nabla q = 0.
\]

Conservation of tracer (temperature, humidity, salinity, …)

Some features

› Energy and enstrophy conserving mom adv (Arakawa & Hsu, 1990)
› Smagorinsky–like biharmonic diffusion (Griffies & Hallberg, 2000)
› Time splitting (ideas from NEMO, ROMS)
› Semi–Lagrangian tracer advection (Diamantakis, 2014)
Shallow water equations II

\[
\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} + f \mathbf{\hat{z}} \times \mathbf{u} = -g \nabla \eta - \nu \nabla^4 \mathbf{u} - r \mathbf{u} + \mathbf{F}
\]

\[
\frac{\partial \eta}{\partial t} + \nabla \cdot (\mathbf{uh}) = 0.
\]

\[
\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q = 0.
\]

Prognostic variables

- \( u \) meridional velocity
- \( v \) zonal velocity
- \( \eta \) surface height, pressure
- \( q \) Tracer, e.g. temperature

Basically a 2D layer of the Navier–Stokes equations.
Rescaling: Biharmonic diffusion

\[ \frac{\partial u}{\partial t} = \ldots - \nu \nabla^4 u \]

For 10km model:

\[ \approx 10^{10} \text{ m}^4 \text{s}^{-1} \quad \mathcal{O} \left( \frac{1}{\Delta^4} \right) \approx 10^{-16} \text{ m}^{-4} \]

Rescaling algorithms is very important for 16bit arithmetics:

\[ u_{n+1} = u_n + \left( \frac{\Delta t}{\Delta} \right)( \ldots - \tilde{\nu} \tilde{\nabla}^4 u ) \]

\[ \approx 10^{-2} \text{ sm}^{-1} \quad \approx 10^{-2} \text{ ms}^{-1} \quad \mathcal{O}(1) \]

Which is indeed computable with 16bit arithmetics!
Shallow water simulation

16 bit posits with 2 exponent bits

The first 16 bit ocean/atmosphere model !?
Forecasts in the shallow water model

Float64  |  Posit(16,2)  |  Float16  |  Satellite

Truth  |  Spot the error!  |  From NASA

Posits for weather and climate models
CoNGA’19
Milan Klöwer
Forecasts in the shallow water model

Control simulation

- Spin-up
- n=280 random start dates
- 100 days
- 100 yrs

- Float64 (truth)
- Float16
- Posit16,1
- Posit16,2

Identical model, different arithmetics +,−,∗,/

Float64 + some discretisation error

Posits for weather and climate models
CoNGA’19
Milan Klöwer
Forecasts in the shallow water model

Useless forecast

Perfect forecast

Forecast error

Normalized RMSE

Rounding errors dominate

discretisation error

Acceptable error

0.0 20 40 60 80 100
time [days]
Forecasts in the shallow water model

Useless forecast

Perfect forecast

Forecast error

vertime [days]
Forecasts in the shallow water model

Perfect forecast

Useless forecast

Forecast error

0.0
0.2
0.4
0.6
0.8
1.0

0
20
40
60
80
100

time [days]

normalized RMSE

Float16

discretisation error

Posit(16,1)
Forecasts in the shallow water model

Perfect forecast

Useless forecast

Forecast error

time [days]

normalized RMSE

Float16

discretisation error

Posit(16,1)

Posit(16,2)
Forecasts in the shallow water model

Posits for weather and climate models
CoNGA'19
Milan Klöwer

Perfect forecast
Useless forecast

Forecast error

Normalized RMSE

time [days]

Posits clearly outperform floats!
Posit errors will be dwarfed by other sources of error.
Dynamic range of Posit(16,2) has a great potential for more complex models.
Conclusion on posits in the shallow water model

PDE-type problems / computational fluid dynamics benefit from a $10^{16} - 10^{-16}$ dynamic range and the increased precision around 1.

Float16 okay, but bfloat16 useless.

Posit(16,2) has a great potential for Earth-system models!
Summary

What we have

<table>
<thead>
<tr>
<th>CPU</th>
<th></th>
<th>PNU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float64</td>
<td></td>
<td>Posit(32,2)</td>
</tr>
<tr>
<td>Float32</td>
<td></td>
<td>Posit(16,2)</td>
</tr>
<tr>
<td>Int</td>
<td></td>
<td>Int</td>
</tr>
</tbody>
</table>

For now

> Use models with 32bit floats, 40–50% speed-up
> Rewrite algorithms for low precision
> Test Posit(16,2) in various models
> Wait

For future

> Switch Float32 -> Posit32
> Test Posit32 -> Posit16

What we want

For now

Get in touch!

milan.kloewer@physics.ox.ac.uk
@milankloewer
github.com/milankl/juls
Perspectives for quires

\[ u^{n+1} = u^n + RK_u \left( Qhv + \partial_x p + D_x + F_x \right) \]

(1) (2) (3) (4) (5)

Which is the dominant balance of the sum? Only clear at runtime! In Float16

\[-0.0002 \leftarrow 10^{-1} + 1 - 1 + 10^{-4} - 10^{-1} \]
\[-0.0005 \leftarrow 10^{-1} + 1 - 10^{-1} + 10^{-4} - 1 \]

But actually this is

\[ 0.0001 = 1 - 1 + 10^{-1} - 10^{-1} + 10^{-4} \]