Stochastic rounding
for weather and climate models

Milan Klöwer[1], Adam Paxton[1], Matthew Chantry[1]
Peter Düben[2], Tim Palmer[1]

[1]University of Oxford, UK
[2]European Centre for Medium-Range Weather Forecasts, UK

StochasticRounding.jl

A registered Julia package, v0.5.1

Stochastic rounding with integer arithmetic

```
function Float32_stochastic_round(x::Float64)
    …
    xi = reinterpret(Int64,x)
    xi += rand(Int64) >> 35
    x = reinterpret(Float64,xi)
    return Float32(x)
end
```

1. Reinterpret same bits as signed integer

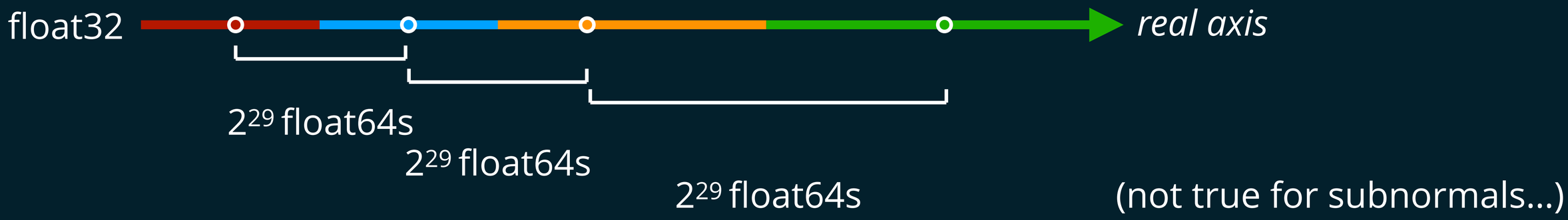
2. Create random bits and *arithmetic* bitshift >> for

```
00…00|01101… in [0,u/2)
11…11|10110… in [−u/2,0)
```

3. Round to nearest

Stochastic rounding with integer arithmetic

A high-precision format float64 (or float32) is **uniformly distributed**
with respect to a low precision format like float16 or bfloat16

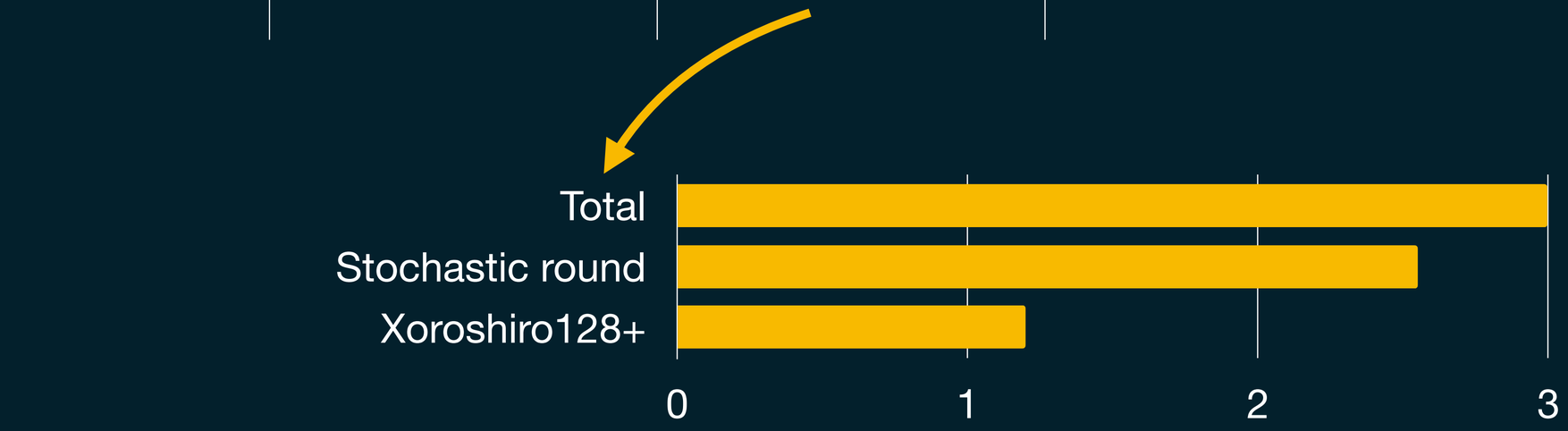


(not true for subnormals...)

StochasticRounding.jl

Benchmark for adding two arrays of 1,000,000 elements

Rounding mode	Float64	Float32	Float16	BFloat16
Round to nearest	1	0.4	0.55	0.5
Stochastic rounding	-	2.4	3	3



Why Julia ???

As long as +-*/... are defined

In Julia type-flexible functions are JIT compiled to *any* number format

```
julia> using ShallowWaters, StochasticRounding

julia> RunModel(Float32,nx=100,time_scheme="SSPRK3")

julia> RunModel(Float32sr,nx=100,time_scheme="SSPRK3")
100% Integration done in 14.5s

julia> RunModel(Float16sr,nx=100,time_scheme="SSPRK3")
100% Integration done in 17.8s
```

StochasticRounding and ShallowWaters are completely independent

number format as argument

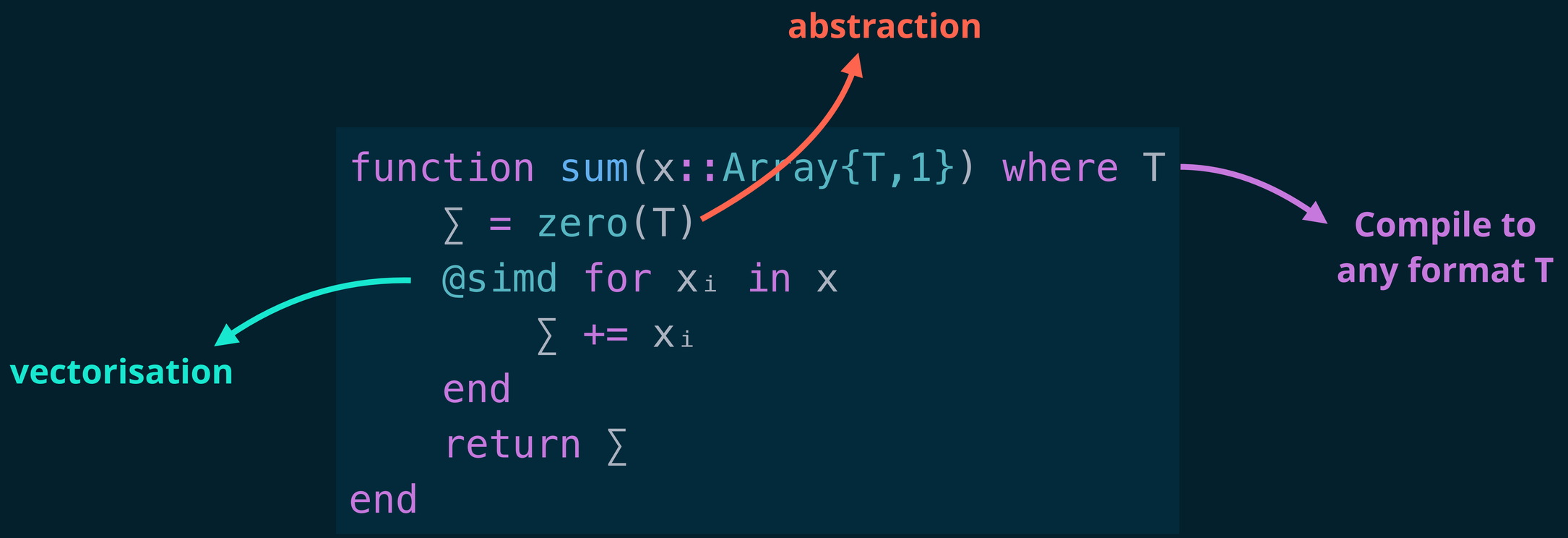
or similarly for an LU decomposition

```
julia> A = Float32sr.(rand(100,100))
julia> b = Float32sr.(rand(100))

julia> x = A\b    # LU decomposition

julia> A*x ≈ b
true
```

just works out of the box

Why Julia ???

In Julia type-flexible functions are JIT compiled to *any* number format



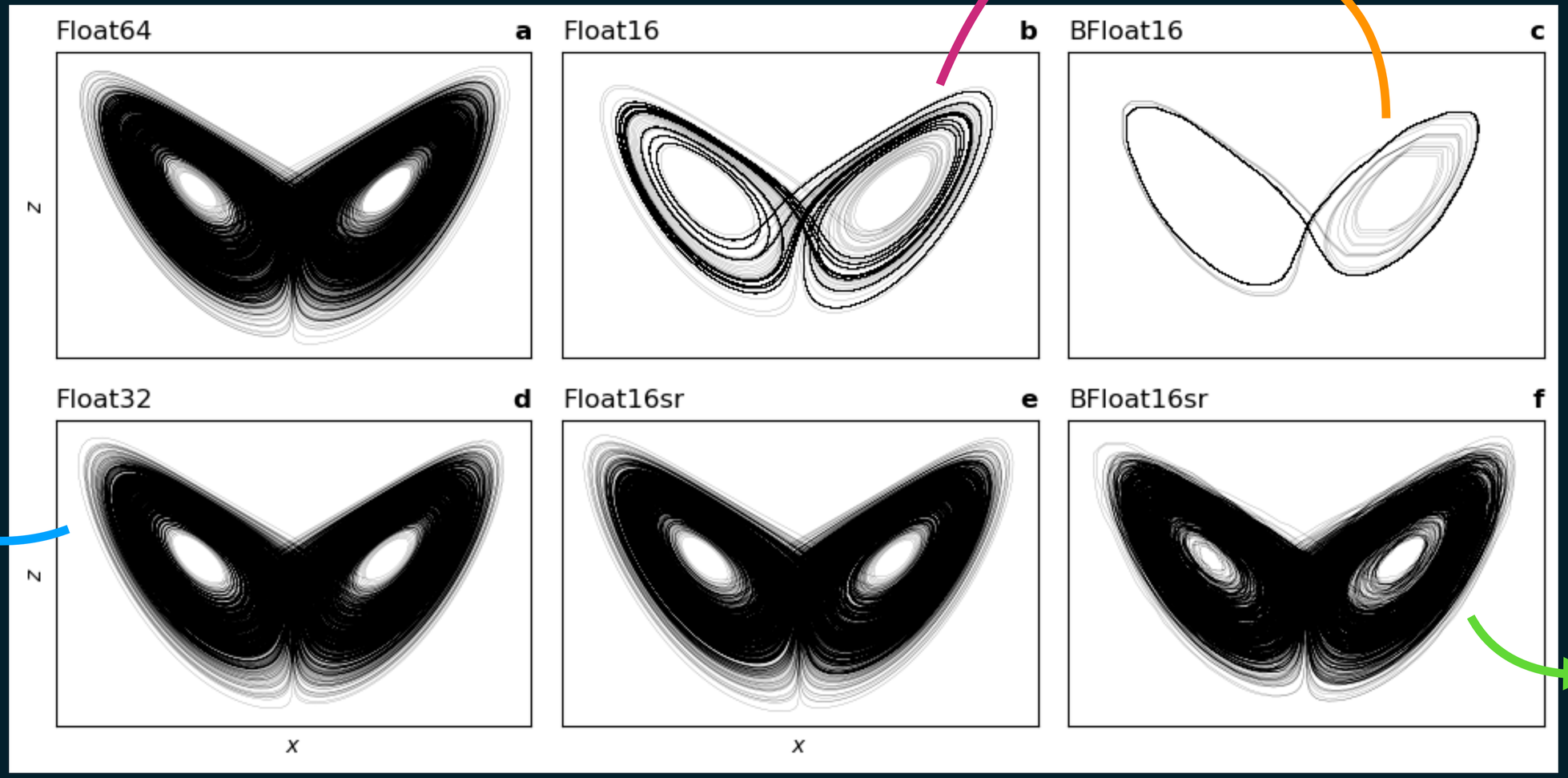
As fast as Julia's inbuilt *sum*,
3x faster than numpy's *sum*

Stochastic rounding in Lorenz system

Orbital length

1825Δt

191Δt



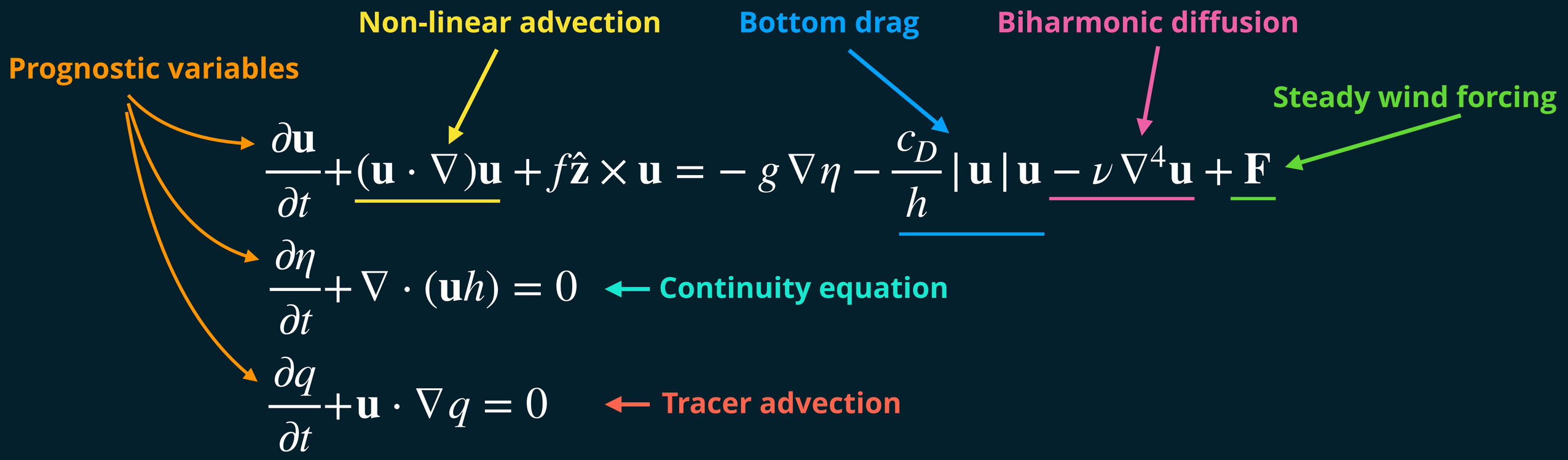
Orbital length

$>10^8\Delta t$

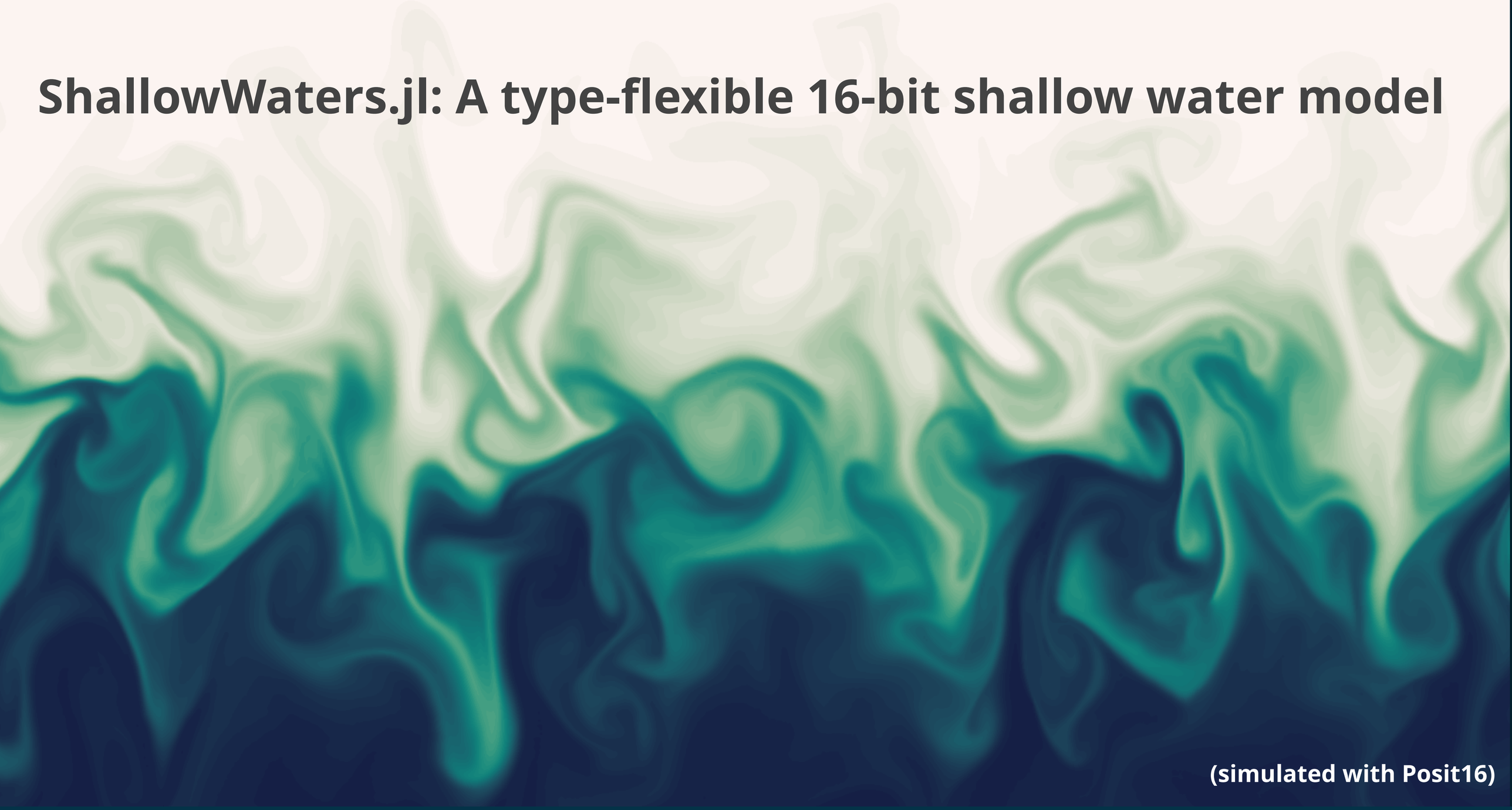
Wobbly solution

For Lorenz96,
every additional variable increases the orbital length by x10-100

ShallowWaters.jl: A type-flexible 16-bit shallow water model



ShallowWaters.jl: A type-flexible 16-bit shallow water model



(simulated with Posit16)

ShallowWaters.jl: A type-flexible 16-bit shallow water model

Some steps for 16-bit arithmetics

Semi-implicit time stepping

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} + f\hat{\mathbf{z}}\times\mathbf{u} = -g\nabla\eta - \frac{c_D}{h}|\mathbf{u}|\mathbf{u} - \nu\nabla^4\mathbf{u} + \mathbf{F}$$

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

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

4th order Runge-Kutta

semi-Lagrangian advection scheme

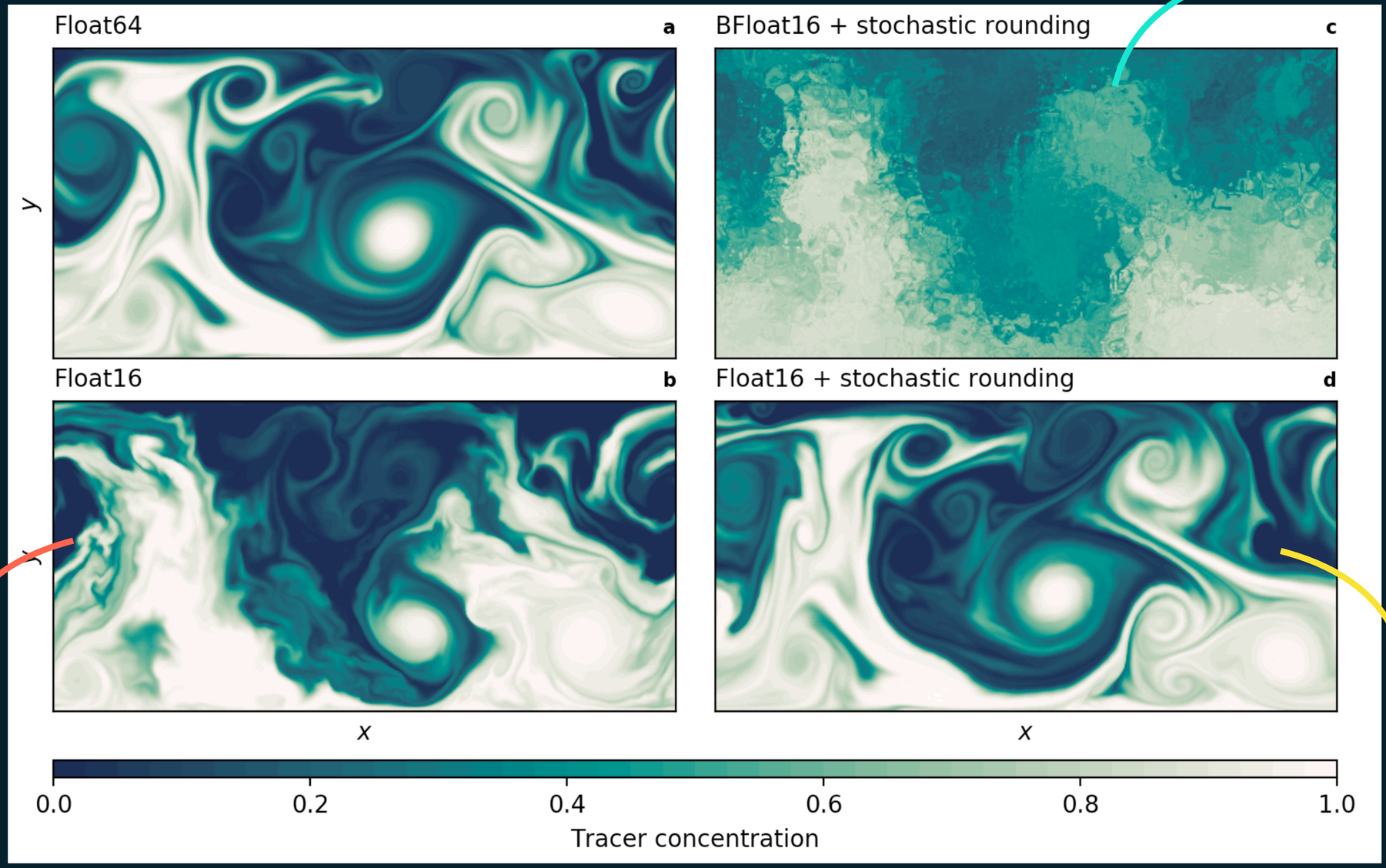
Rescaling the equations

$$\Delta t \to \frac{\Delta t}{\Delta x},$$

$$\nu\nabla^4 \to \left(\frac{\nu}{\Delta x^3}\right)\tilde{\nabla}^4$$

Stochastic rounding in ShallowWaters.jl

Stagnation without stochastic rounding

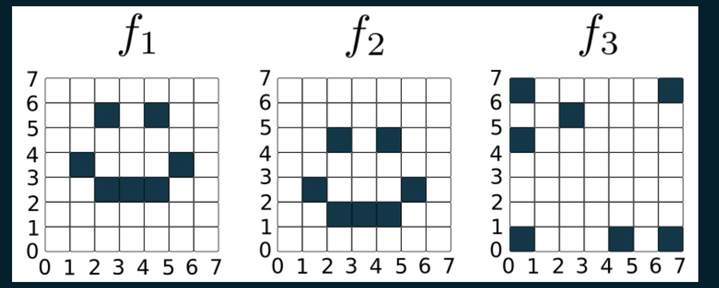


Rounding errors cause gravity waves

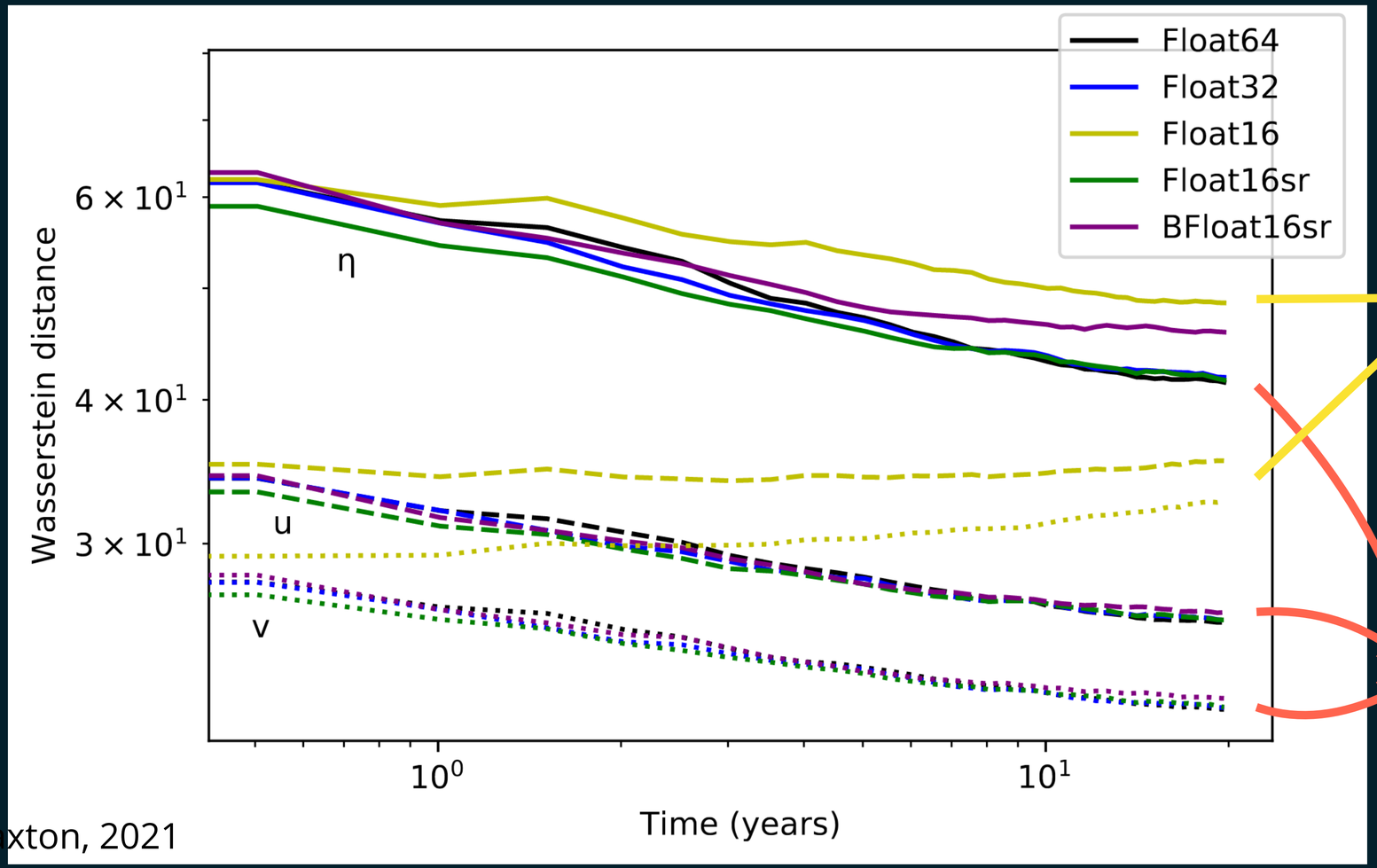
Stochastic rounding stabilises the flow and reduces the overall error

Stochastic rounding in ShallowWaters.jl

Quantifying the error in the invariant measure



Wasserstein distance: Discrepancy between PDFs



Lower is better

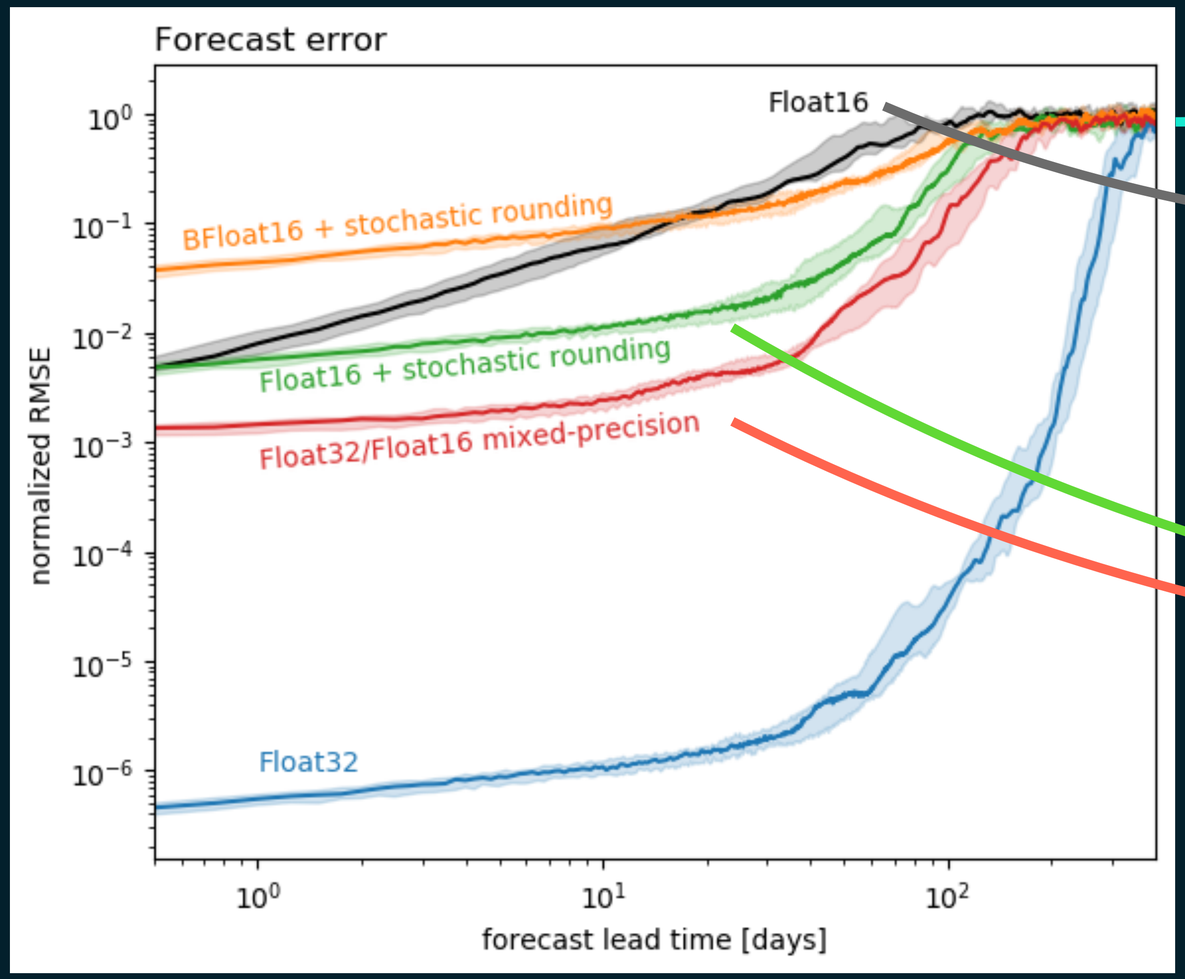
For Float16 and BFloat16 + SR the invariant measure differs from the reference

For Float32 and Float16 + SR the invariant measures converge to the reference

A Paxton, 2021

Error growth with stochastic rounding

Accumulated rounding error in ShallowWaters.jl



Error=1 means chaos removed all information from initial conditions

Round to nearest has a steeper error growth than stochastic rounding

Float16 + SR starts with a higher error in the initial conditions but has otherwise a similar error growth to mixed-precision

Summary

Stochastic rounding is great because

- Chaotic systems are finally chaotic again
- Dynamical systems explore a larger phase space
- Better sampled invariant measures
- Stabilises simulations and reduces error
- Makes 16-bit arithmetic more attractive for climate models

Packages on GitHub

- milankl/StochasticRounding.jl (v0.5.1)
- milankl/ShallowWaters.jl (v0.4)
- milankl/SoftPosit.jl (v0.3)
- eapax/EarthMover.jl