# Inexact computers for more accurate forecasts: Are we over-engineering numerical precision in weather and climate models?

Peter Düben, Stephen Jeffress, Tobias Thornes, Tim Palmer
S Dolaptchiev, J Schlachter, Parishkrati, S Yenugula,
J Augustine, C Enz, K Palem, F Russell, X Niu, W Luk

University of Oxford, University of Frankfurt, EPFL, IITM, Rice University,
Imperial College

# Why should we use inexact hardware in weather and climate predictions?

- Numerical models are crucial for reliable forecasts of future weather and climate.
- The quality of these forecasts dependents strongly on the resolution and complexity of the numerical models used.
- Resolution is limited by the computational power of state-of-the-art super computers.

# Why should we use inexact hardware in weather and climate predictions?

- Numerical models are crucial for reliable forecasts of future weather and climate.
- The quality of these forecasts dependents strongly on the resolution and complexity of the numerical models used.
- Resolution is limited by the computational power of state-of-the-art super computers.

- The free lunch is over: We can not assume that the steady increase in resolution and computational power will continue.

# Why should we use inexact hardware in weather and climate predictions?

- Silicon technology is reaching it's limits due to transistor sizes. Moore's law is under threat.
- Excessive parallelisation will increase power consumption linearly with the number of processors.
- Running one ensemble member of the ensemble forecast of ECMWF is equivalent to leaving the Kettle on for 3 hours (thanks Nils Wedi).

# What is inexact hardware?

**My definition: Inexact hardware is using a level of numerical precision which is smaller than double precision (64 bits).**

- Inexact hardware allows a reduction of power consumption and/or an increase in performance and therefore a reduction of computational cost.
- This would allow simulations at higher resolution and possibly more accurate forecasts.

# What is inexact hardware?

**My definition: Inexact hardware is using a level of numerical precision which is smaller than double precision (64 bits).**

- Inexact hardware allows a reduction of power consumption and/or an increase in performance and therefore a reduction of computational cost.
- This would allow simulations at higher resolution and possibly more accurate forecasts.

**It turns out that there are plenty of different approaches in hardware development that study trade-offs between precision and performance!**

# What is inexact hardware?

**My definition: Inexact hardware is using a level of numerical precision which is smaller than double precision (64 bits).**

- Inexact hardware allows a reduction of power consumption and/or an increase in performance and therefore a reduction of computational cost.
- This would allow simulations at higher resolution and possibly more accurate forecasts.

**It turns out that there are plenty of different approaches in hardware development that study trade-offs between precision and performance!**

**Easiest way: double $\rightarrow$ single ($\rightarrow$ half).**

# A short introduction to bit representation

- The computer represents an integer number as a string of 32 bits. Each bit represents a power of two:

$$102090 = 0\cdot 2^0 + 1\cdot 2^1 + 0\cdot 2^2 + 1\cdot 2^3 + 0\cdot 2^4 + 0\cdot 2^5 + 1\cdot 2^6 \ldots = \sum_{i=0}^{31} b_i 2^i$$

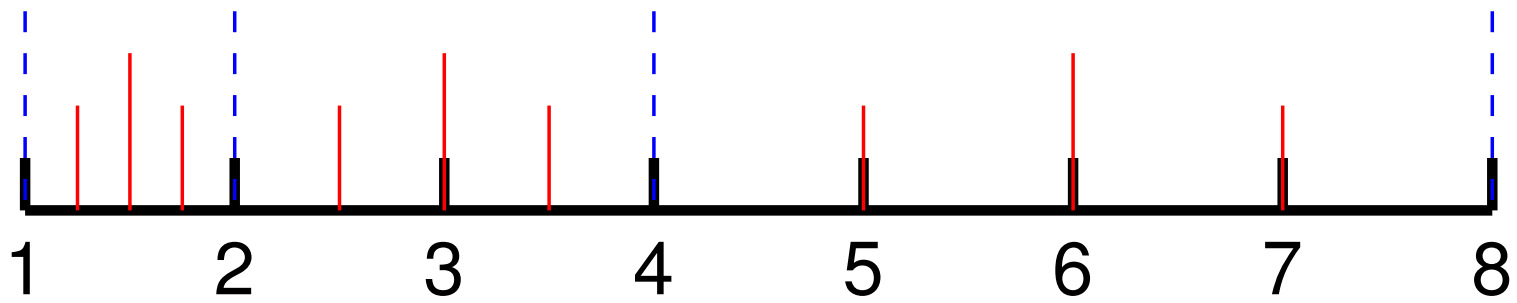
# A short introduction to bit representation

- The computer represents an integer number as a string of 32 bits. Each bit represents a power of two:

$$102090 = 0\cdot 2^0 + 1\cdot 2^1 + 0\cdot 2^2 + 1\cdot 2^3 + 0\cdot 2^4 + 0\cdot 2^5 + 1\cdot 2^6 \ldots = \sum_{i=0}^{31} b_i 2^i$$

- A real number $a$ is represented as a 64 bit floating point number:

$$a = (-1)^S \left(1 + \sum_{i=1}^{52} b_{-i} 2^{-i}\right) 2^E, \quad \text{where} \quad E = \left(\sum_{i=0}^{10} e_i 2^i\right) - 1023.$$



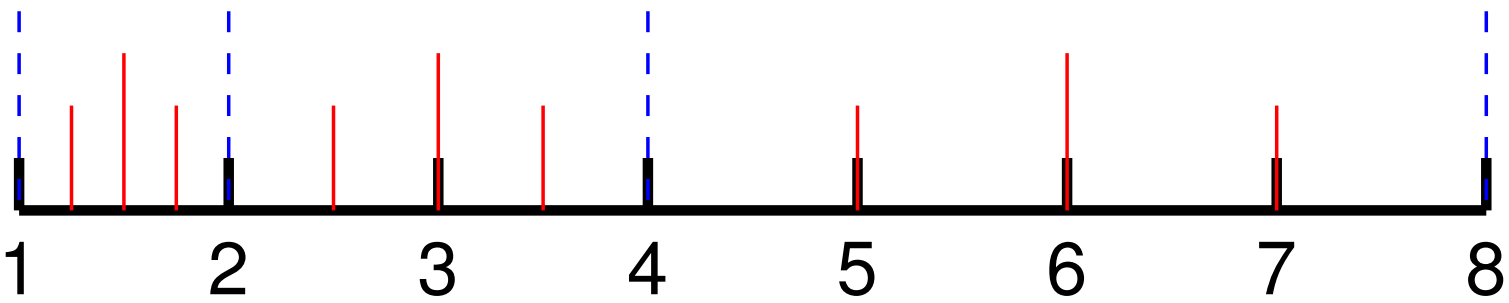
# A short introduction to bit representation

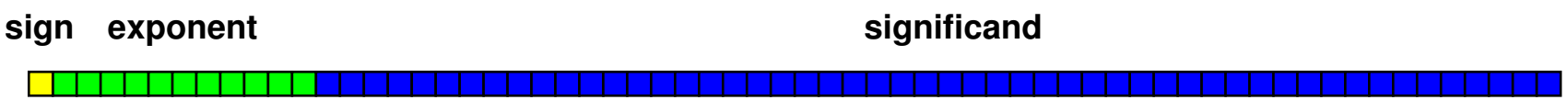
- The computer represents an integer number as a string of 32 bits. Each bit represents a power of two:

$$102090 = 0\cdot 2^0 + 1\cdot 2^1 + 0\cdot 2^2 + 1\cdot 2^3 + 0\cdot 2^4 + 0\cdot 2^5 + 1\cdot 2^6 \ldots = \sum_{i=0}^{31} b_i 2^i$$

- A real number $a$ is represented as a 64 bit floating point number:

$$a = (-1)^S \left(1 + \sum_{i=1}^{52} b_{-i} 2^{-i}\right) 2^E, \quad \text{where} \quad E = \left(\sum_{i=0}^{10} e_i 2^i\right) - 1023.$$

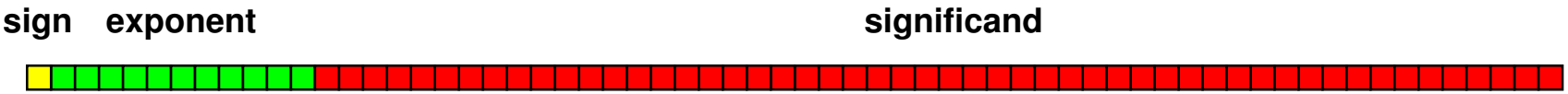


# Three approaches to inexact floating point units

## Stochastic processor

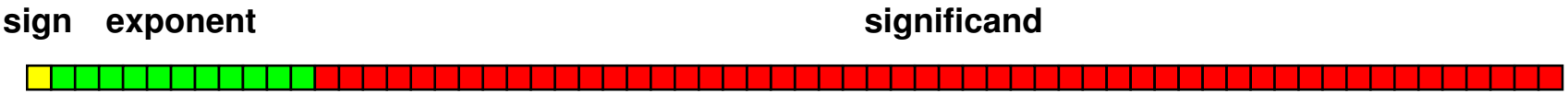
- If we reduce the applied voltage or the wall clock time beyond a certain level, we will get hardware errors, but we will save power.
- The error rate of a stochastic processor can be reduced massively, if the architecture is changed.



# Three approaches to inexact floating point units

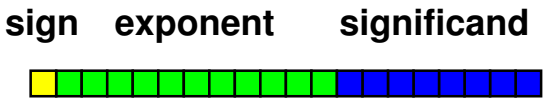
## Stochastic processor

- If we reduce the applied voltage or the wall clock time beyond a certain level, we will get hardware errors, but we will save power.
- The error rate of a stochastic processor can be reduced massively, if the architecture is changed.




## Pruning

Parts of the CPU that are hardly used or do not have a strong influence on significant bits are removed.

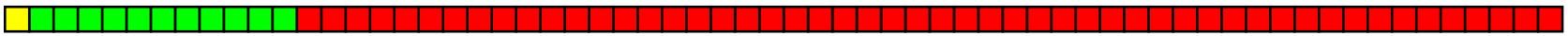


# Three approaches to inexact floating point units

## Stochastic processor

- If we reduce the applied voltage or the wall clock time beyond a certain level, we will get hardware errors, but we will save power.
- The error rate of a stochastic processor can be reduced massively, if the architecture is changed.

**sign** **exponent** **significand**



## Pruning

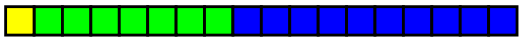
Parts of the CPU that are hardly used or do not have a strong influence on significant bits are removed.

**sign** **exponent** **significand**

## Field Programmable Gate Array (FPGA)

- FPGAs are integrated circuits that can be configured by the user.
- Numerical precision can be customised to the application.

**sign** **exponent** **significand**



# A scale-selective approach

Spectral models allow to treat different scales at different levels of precision.

# A scale-selective approach

Spectral models allow to treat different scales at different levels of precision.

**We can push the small scales harder than the large scales.**

# A scale-selective approach

Spectral models allow to treat different scales at different levels of precision.

**We can push the small scales harder than the large scales.**

**This is intuitive due to the high inherent uncertainty in small scale dynamics (parametrisation, viscosity, data-assimilation).**

# A scale-selective approach

Spectral models allow to treat different scales at different levels of precision.

**We can push the small scales harder than the large scales.**

**This is intuitive due to the high inherent uncertainty in small scale dynamics (parametrisation, viscosity, data-assimilation).**

**The smallest scales are the most expensive once.**

# Our vision....

**A global circulation model which is using just the right level of precision and reducing numerical precision with scales.**

**Large scales: Double precision, Small scales: Half precision**

# Our vision....

**A global circulation model which is using just the right level of precision and reducing numerical precision with scales.**

**Large scales: Double precision, Small scales: Half precision**

**How will rounding errors affect the solution?**

# Reduced precision in an atmosphere model

- We calculate weather forecasts with a spectral dynamical core (IGCM) in a “Held-Suarez world” and compare results against a high resolution truth.
- Floating point precision for the significand is reduced to 8 or 10 bits instead of 52 bits for double precision using an emulator.
- In the reduced precision setup, only 2% of the computational cost of the control simulation is calculated in double precision.
- Scale separation turned out to be really important.
- We also made successful tests with climate type simulations.

# What are the savings?

- In cooperation with Rice University (USA) and EPFL (Switzerland) we derive hardware setups of the floating point unit, memory and cache that show comparable error pattern.
- We analyse the possible savings and trace the application to obtain an estimate for the power consumption on the exact and inexact hardware.

# What are the savings?

- In cooperation with Rice University (USA) and EPFL (Switzerland) we derive hardware setups of the floating point unit, memory and cache that show comparable error pattern.
- We analyse the possible savings and trace the application to obtain an estimate for the power consumption on the exact and inexact hardware.

| Resolution | Precision FP significand | Normalised Energy Demand | Forecast error day 2 |
|---|---|---|---|
| 235 km | 52 | 1.0 | 2.3 |
| **315 km** | 52 | 0.47 | 4.5 |
| 235 km | **10** | 0.32 | 2.3 |
| 235 km | **8** | 0.29 | 2.5 |

Forecast error: Mean error in geopotential height.

# What are the savings?

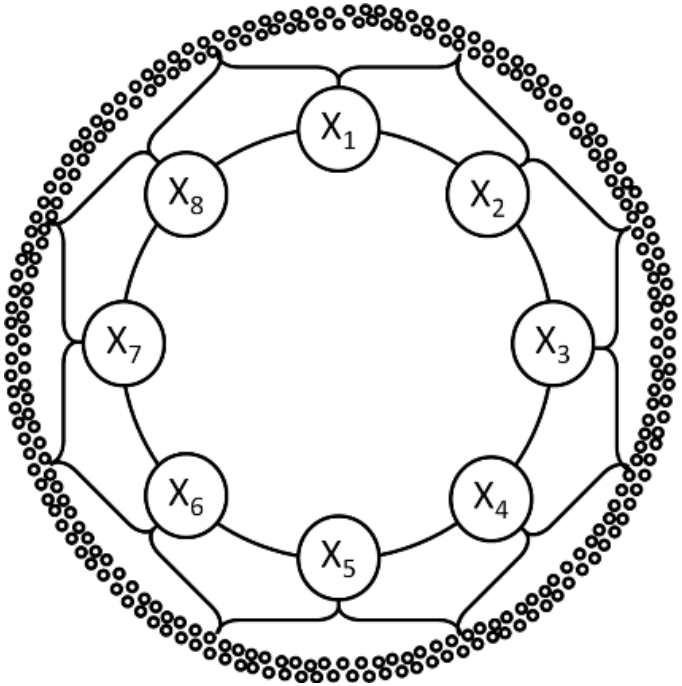
- In cooperation with Rice University (USA) and EPFL (Switzerland) we derive hardware setups of the floating point unit, memory and cache that show comparable error pattern.
- We analyse the possible savings and trace the application to obtain an estimate for the power consumption on the exact and inexact hardware.

| Resolution | Precision FP significand | Normalised Energy Demand | Forecast error day 2 |
|---|---|---|---|
| 235 km | 52 | 1.0 | 2.3 |
| **315 km** | 52 | 0.47 | 4.5 |
| 235 km | **10** | 0.32 | 2.3 |
| 235 km | **8** | 0.29 | 2.5 |

Forecast error: Mean error in geopotential height.

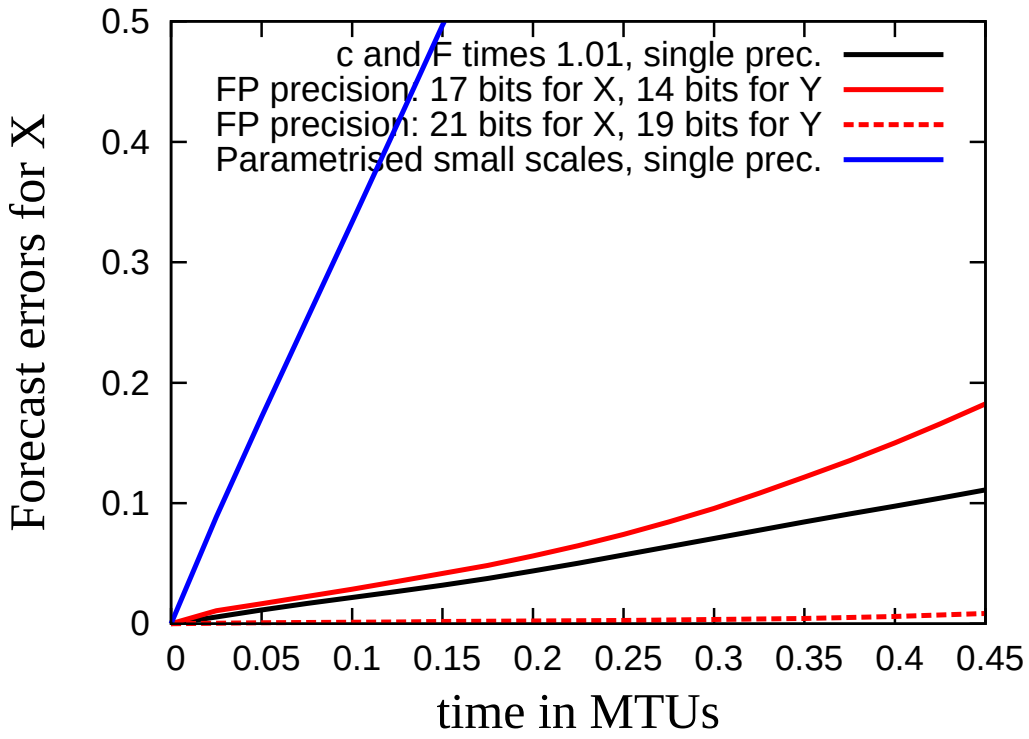
**To save power a reduction in precision is much more efficient than a reduction in resolution!**

# Lorenz ’96 on an FPGA

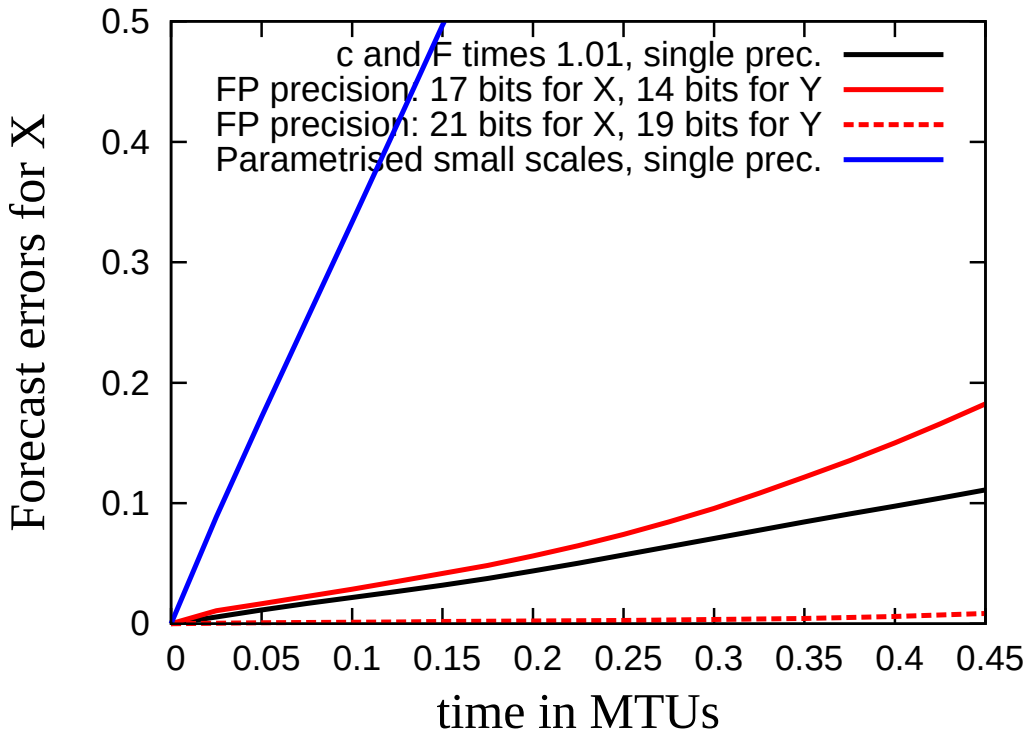



- We implemented the Lorenz ’96 model on an FPGA in cooperation with Xinyu Niu, Francis Russel and Wayne Luk from Imperial College.
- We scale the size of Lorenz ’96 to the size of a high performance application (up to more than 100 million degrees-of-freedom).
- We compare results with reduced precision against results with perturbed parameters (by 1 %) or parametrised small scales.

# Lorenz ’96 on an FPGA: Weather



# Lorenz ’96 on an FPGA: Weather




**Changes in weather type forecasts are comparably small when precision is reduced.**

# Lorenz ’96 on an FPGA: Climate

| Precision | Hellinger distance |
|---|---|
| c and F times 1.01, single prec. | 0.0054 |
| Parametrised small scales, single prec. | 0.1137 |
| FP precision, 17 bits for X, 14 bits for Y | 0.0079 |
| FP precision, 21 bits for X, 19 bits for Y | 0.0029 |

The Hellinger distance describes the difference between two PDFs.

# Lorenz ’96 on an FPGA: Climate

| Precision | Hellinger distance |
|---|---|
| c and F times 1.01, single prec. | 0.0054 |
| Parametrised small scales, single prec. | 0.1137 |
| FP precision, 17 bits for X, 14 bits for Y | 0.0079 |
| FP precision, 21 bits for X, 19 bits for Y | 0.0029 |

The Hellinger distance describes the difference between two PDFs.

**Changes in climate type forecasts are comparably small when precision is reduced.**

# Lorenz ’96 on an FPGA: Speed and Power

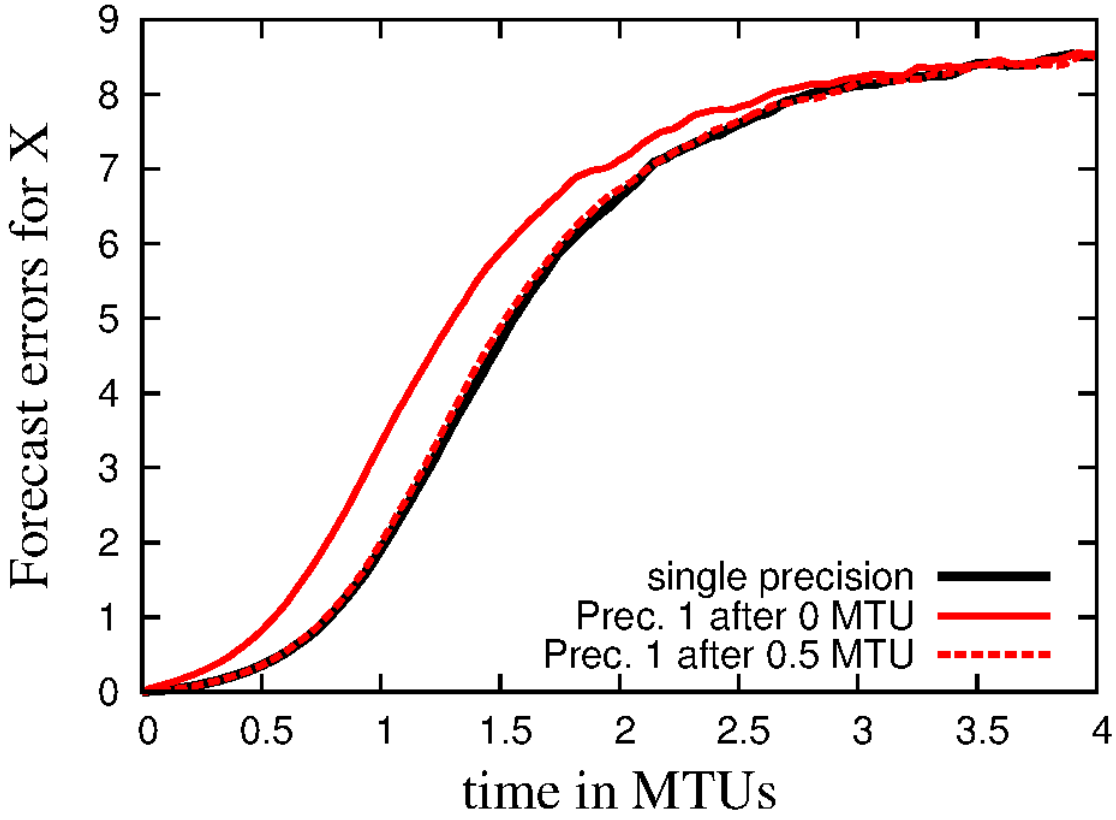
| Hardware | Speed | Energy efficiency |
|---|---|---|
| CPU, 12 cores, single precision | 1.0 | 1.0 |
| CPU, 12 cores, double precision | 0.5 | - |
| FPGA, single precision | 2.8 | 10.4 |
| FPGA, 17 bits for X, 14 bits for Y | 6.9 | 23.9 |
| FPGA, 21 bits for X, 19 bits for Y | 5.4 | 18.9 |

# Lorenz ’96 on an FPGA: Speed and Power

| Hardware | Speed | Energy efficiency |
|---|---|---|
| CPU, 12 cores, single precision | 1.0 | 1.0 |
| CPU, 12 cores, double precision | 0.5 | - |
| FPGA, single precision | 2.8 | 10.4 |
| FPGA, 17 bits for X, 14 bits for Y | 6.9 | 23.9 |
| FPGA, 21 bits for X, 19 bits for Y | 5.4 | 18.9 |

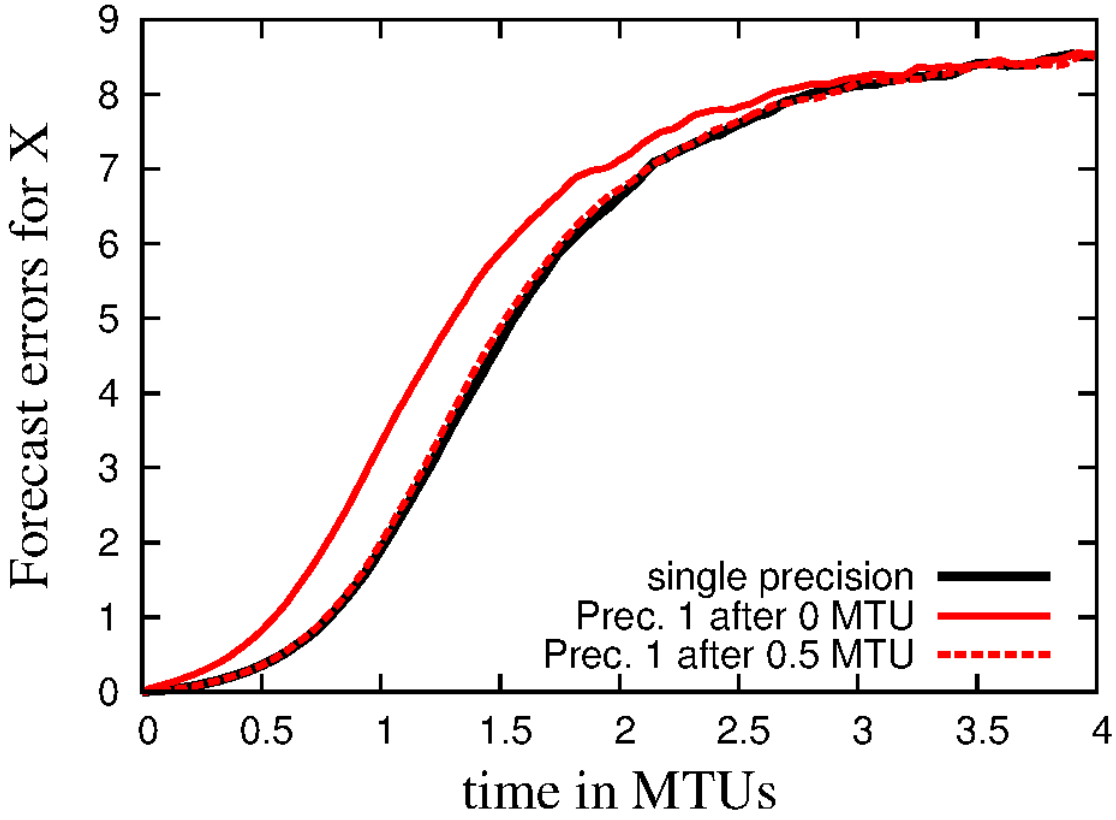
**We get significant savings in energy consumption and a significant increase in performance if we use FPGAs with reduced precision.**

# “Weather forecasts” with Lorenz ’95 that reduce precision with time




Prec 1 is using 6 bits in the exponents and 11 bits in the significand.

# “Weather forecasts” with Lorenz ’95 that reduce precision with time




Prec 1 is using 6 bits in the exponents and 11 bits in the significand.

**Precision can be reduced with time in weather-type forecasts.**

# Rounding errors in atmosphere models

- Rounding errors will generate a forcing that is added to the differential equations.
- The forcing is (almost) uncorrelated in space and time.
- We can influence the forcing by changing either the precision, or the model setup (time stepping scheme etc.)

# Rounding errors in atmosphere models

- Rounding errors will generate a forcing that is added to the differential equations.
- The forcing is (almost) uncorrelated in space and time.
- We can influence the forcing by changing either the precision, or the model setup (time stepping scheme etc.)

- Stochastic parametrisation schemes use “high quality” random noise with specific mean and variability.
- The noise is motivated by sub-grid-scale variability.

# Rounding errors in atmosphere models

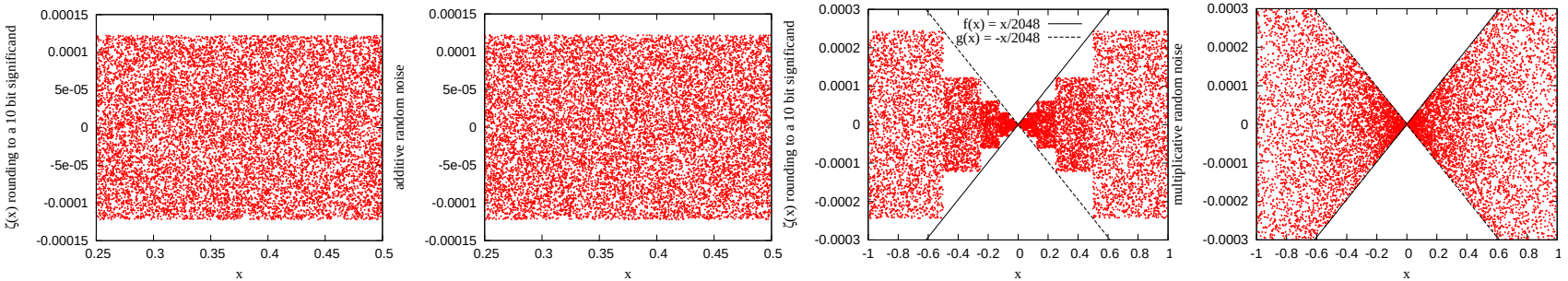
- Rounding errors will generate a forcing that is added to the differential equations.
- The forcing is (almost) uncorrelated in space and time.
- We can influence the forcing by changing either the precision, or the model setup (time stepping scheme etc.)

- Stochastic parametrisation schemes use “high quality” random noise with specific mean and variability.
- The noise is motivated by sub-grid-scale variability.

**Can we design noise from rounding errors and replace the random noise in stochastic parametrisation schemes?**

# Rounding errors in atmosphere models

- Rounding errors will generate a forcing that is added to the differential equations.
- The forcing is (almost) uncorrelated in space and time.
- We can influence the forcing by changing either the precision, or the model setup (time stepping scheme etc.)




**We can mimic additive and multiplicative noise with rounding errors.**

# Let's test this!

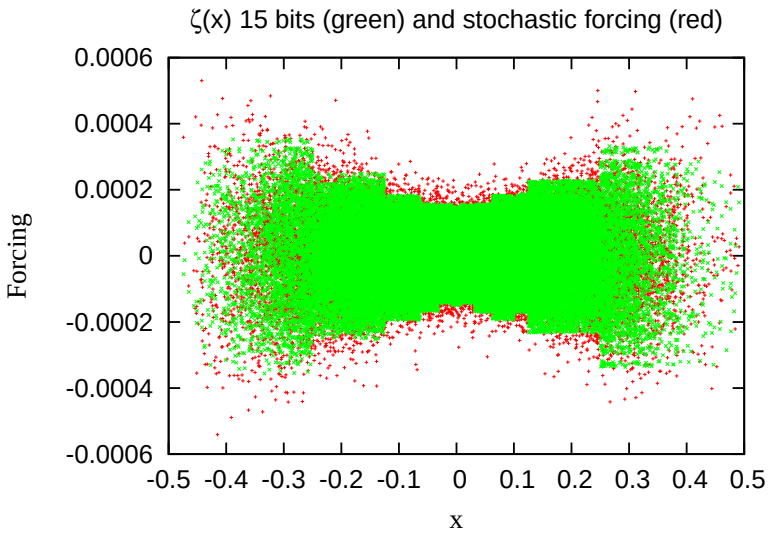
- We study a model of the randomly forced one-dimension Burgers equation with stochastic parametrisation for turbulent closure by Dolaptchiev, Timofeyev and Achatz
- The parametrisation scheme is based on the stochastic mode reduction strategy by Majda, Timofeyev and Vanden-Eijnden.
- Two thirds of the computational cost of the parametrised model is caused by the routine that calculates the random noise of the stochastic parametrisation scheme.

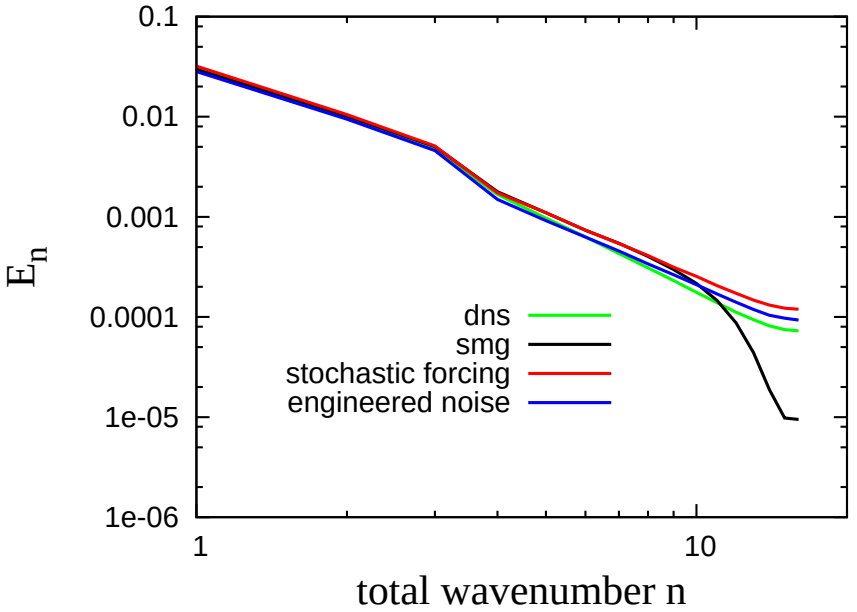
# Let's test this!

- We study a model of the randomly forced one-dimension Burgers equation with stochastic parametrisation for turbulent closure by Dolaptchiev, Timofeyev and Achatz
- The parametrisation scheme is based on the stochastic mode reduction strategy by Majda, Timofeyev and Vanden-Eijnden.
- Two thirds of the computational cost of the parametrised model is caused by the routine that calculates the random noise of the stochastic parametrisation scheme.

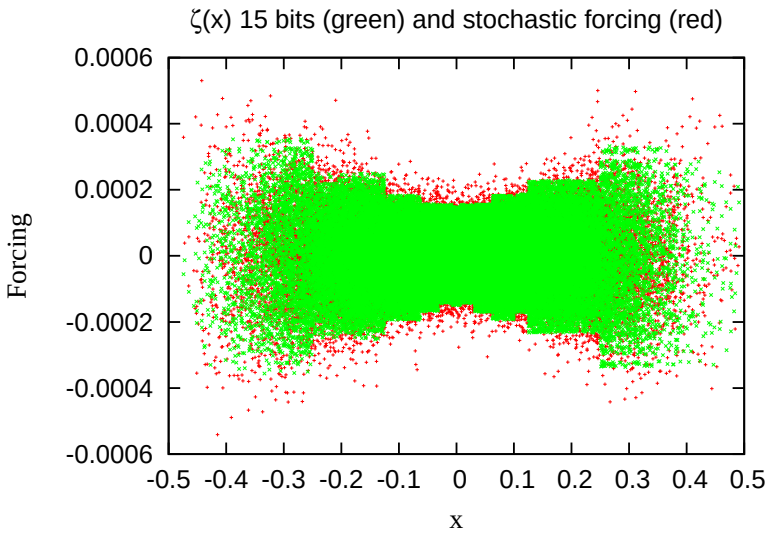
**We design noise from rounding errors by adding a small number of operations per prognostic variable per timestep to replace the random forcing in the parametrisation scheme.**

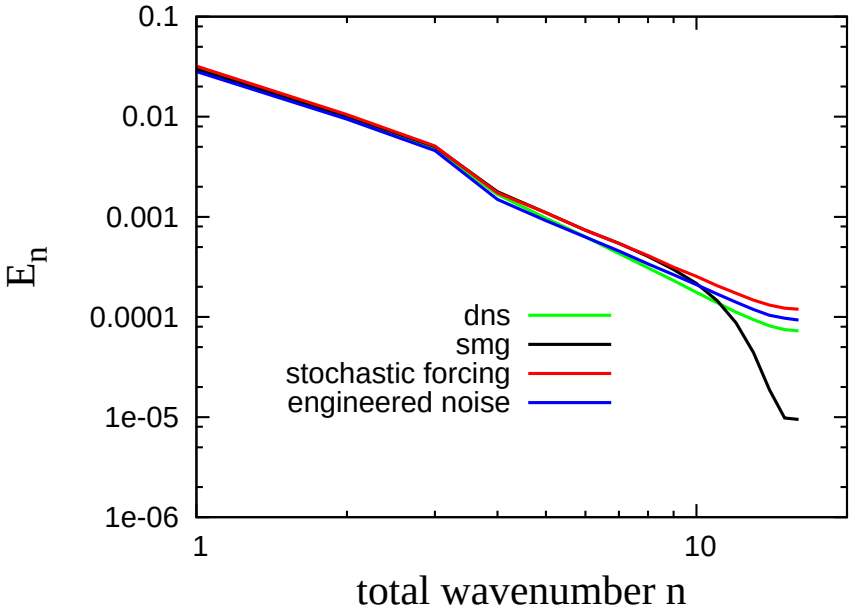
# Rounding errors can represent sub-grid-scale variability



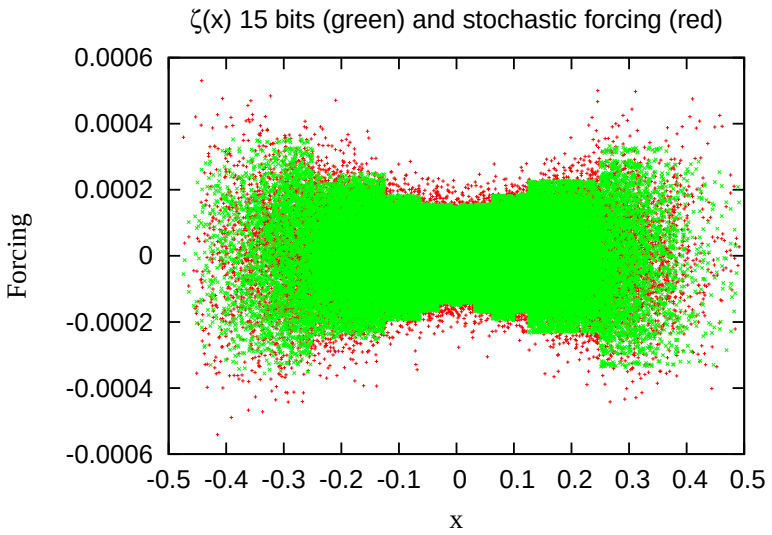
# Rounding errors can represent sub-grid-scale variability

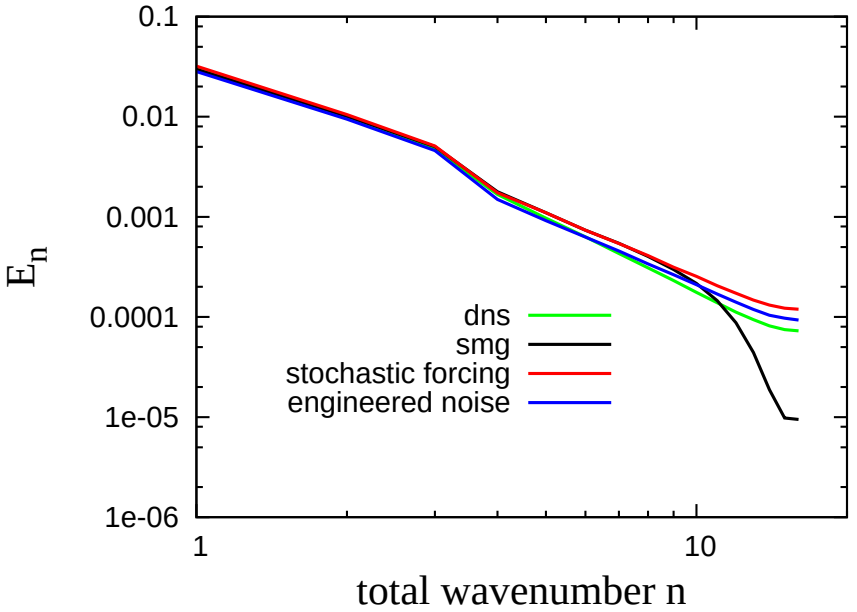



**We can replace the stochastic forcing of a stochastic parametrisation scheme by rounding errors.**

# Rounding errors can represent sub-grid-scale variability

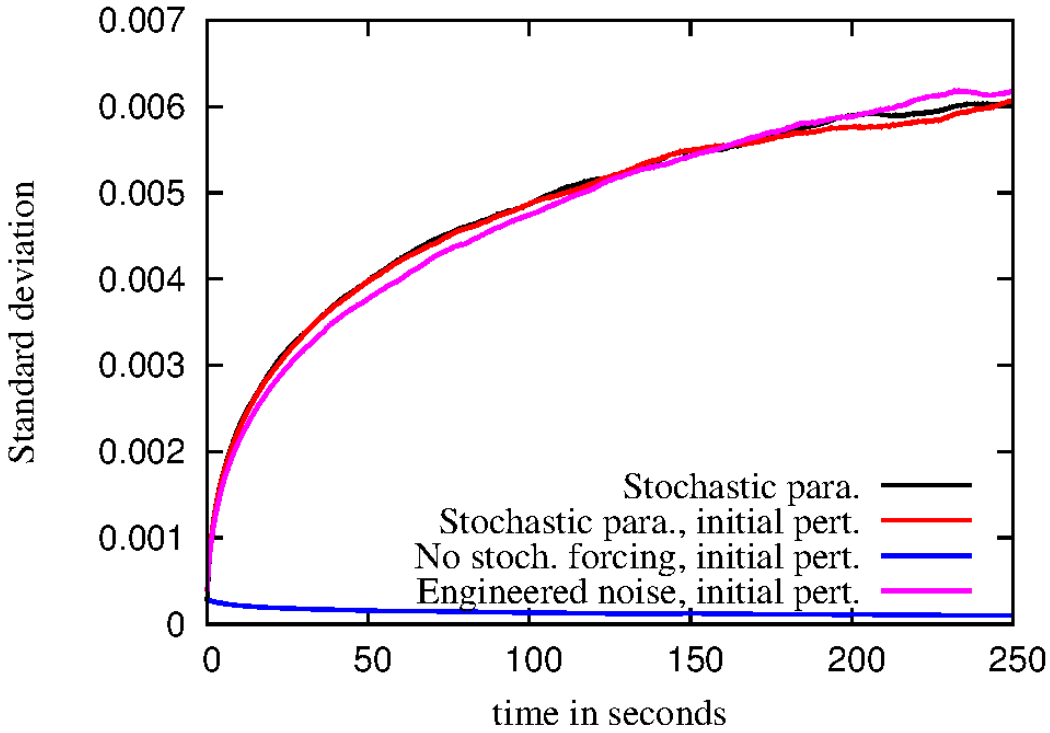



**We can replace the stochastic forcing of a stochastic parametrisation scheme by rounding errors.**

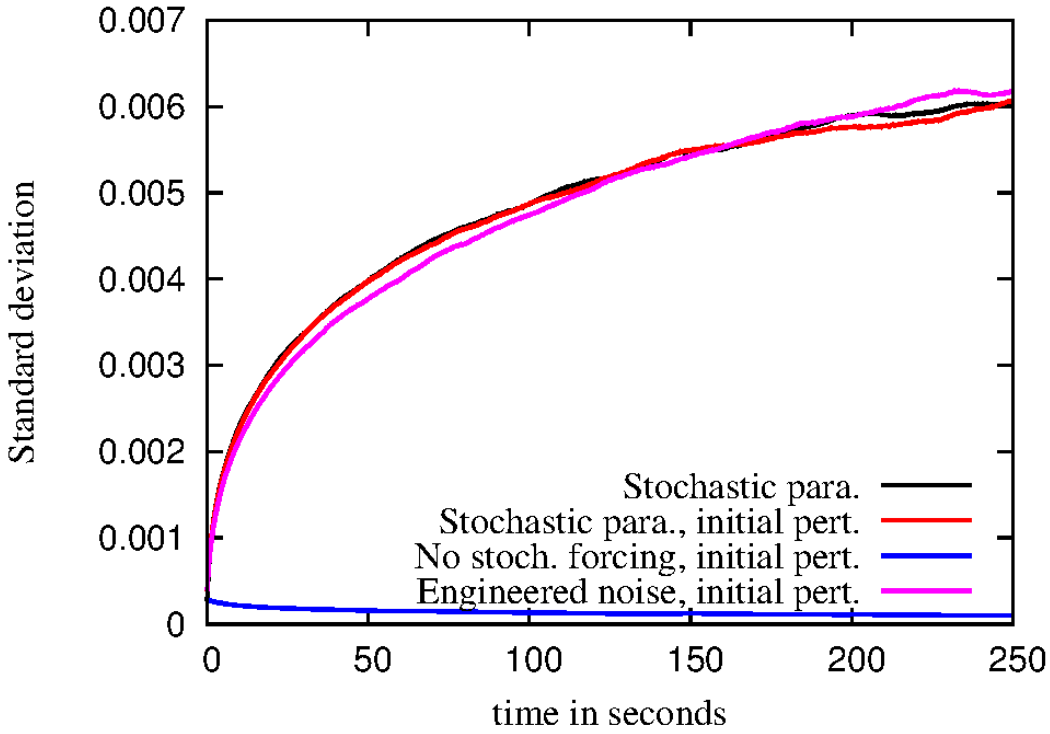
**This study is certainly extremely idealised!**

# Rounding errors can represent sub-grid-scale variability




If we add a tiny amount of random noise to the initial conditions, rounding errors will be uncorrelated in space and time.

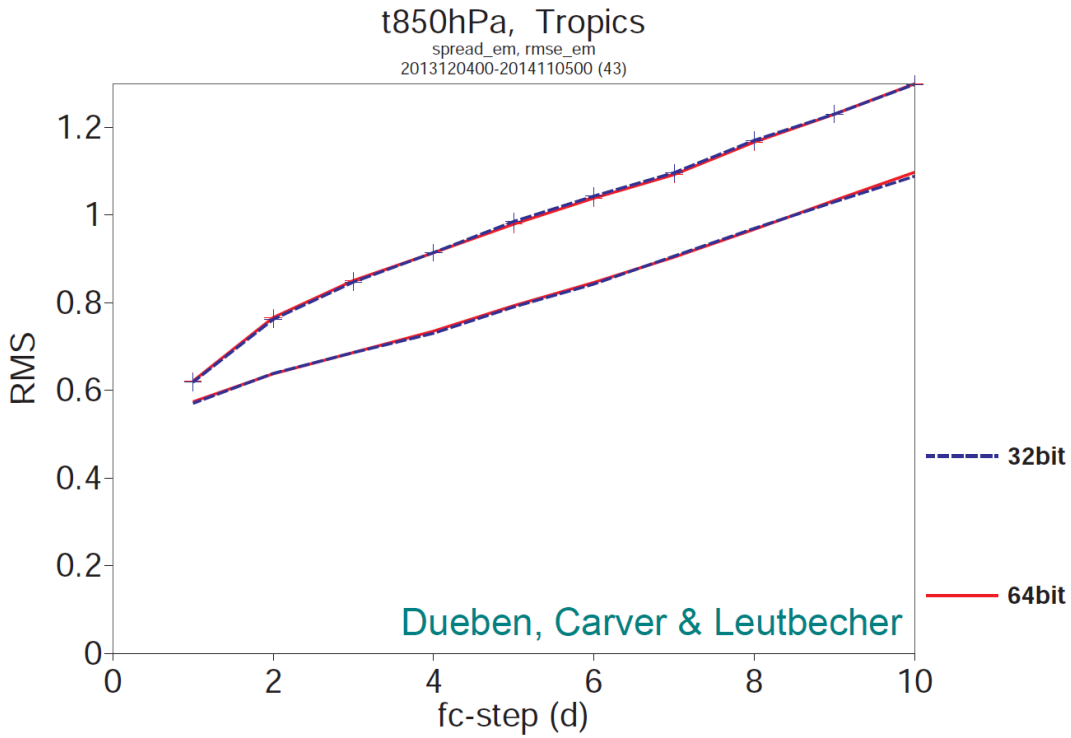
# Rounding errors can represent sub-grid-scale variability



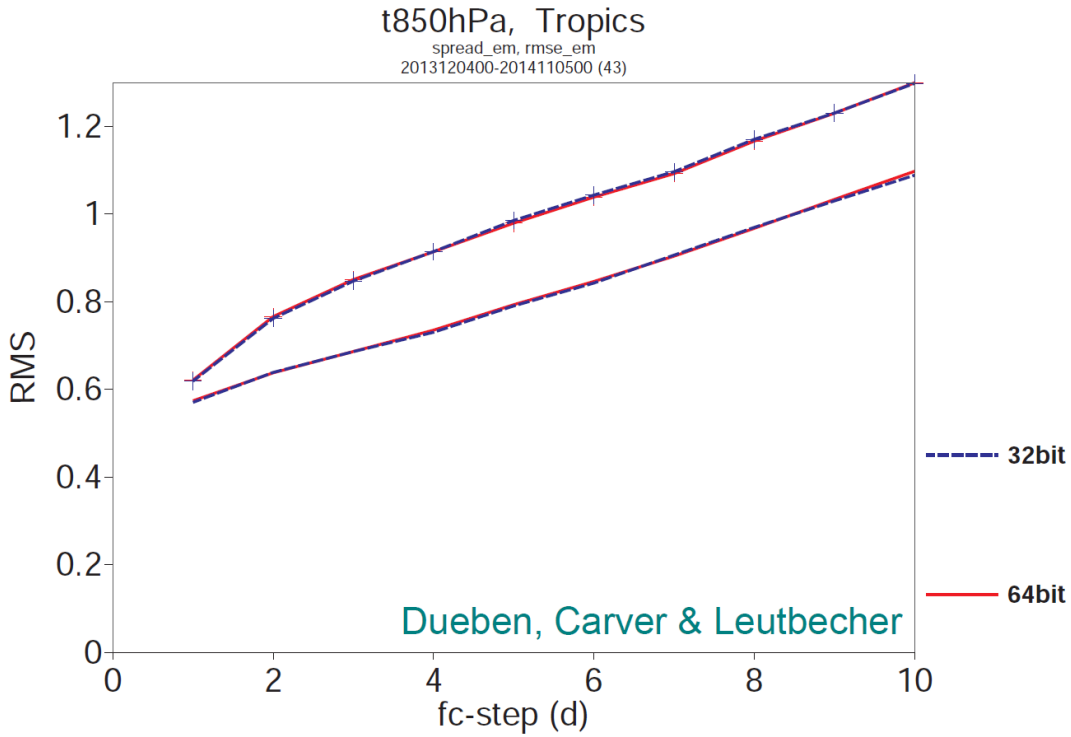

If we add a tiny amount of random noise to the initial conditions, rounding errors will be uncorrelated in space and time.

**Rounding errors can be used to represent sub-grid-scale variability and generate ensembles.**

# How to approach full-blown GCMs? OpenIFS in single precision



# How to approach full-blown GCMs? OpenIFS in single precision




- Approximately one third speed-up.
- Less computing nodes are needed due to reduced memory requirements.
- Filip Vana is now working on a single precision version of IFS.

# How to approach full-blown GCMs?
# Emulation of reduced precision

**Method:**
We define a new reduced-precision type that behaves like a floating point number, but reduces the precision when it is operated on, this allows the emulation of reduced precision and specific setups of inexact hardware in large models (maybe IFS?) with no need for extensive changes of model code.

**Example:**
Emulated 5 bit significand with reduced precision "+"

*Standard Fortran:*

```
REAL :: a,b,c
a = 1.442221
b = 2.136601
c = a+b
```

$\rightarrow$ *c=3.578822*

*Reduced precision declarations:*

```
TYPE(reduced_precision) :: a,b,c
a = 1.442221
b = 2.136601
c = a+b
```

$\rightarrow$ *c=3.562500*

# How to approach full-blown GCMs?

**If we use the emulator to reduce precision, how can we find the minimal level of precision that can be used?**

# How to approach full-blown GCMs?

**If we use the emulator to reduce precision, how can we find the minimal level of precision that can be used?**

**Shall we use the same precision level for a) the entire model, b) each module, c) each line of code, d) each variable?**

# How to approach full-blown GCMs?

**If we use the emulator to reduce precision, how can we find the minimal level of precision that can be used?**

**Shall we use the same precision level for a) the entire model, b) each module, c) each line of code, d) each variable?**

**The more we fine-grain, the more we will be able to a) reduce precision, b) learn about the information content of specific variables and c) learn about technical issues and bad programming.**

# How to approach full-blown GCMs?

**If we use the emulator to reduce precision, how can we find the minimal level of precision that can be used?**

**Shall we use the same precision level for a) the entire model, b) each module, c) each line of code, d) each variable?**

**The more we fine-grain, the more we will be able to a) reduce precision, b) learn about the information content of specific variables and c) learn about technical issues and bad programming.**

**But tests with a big model are expensive!**

# Information content and reduced precision

**We like to think that the minimal level of precision that can be used is the "information content" of a variable (e.g. for $\pi$).**

# Information content and reduced precision

**We like to think that the minimal level of precision that can be used is the "information content" of a variable (e.g. for $\pi$).**

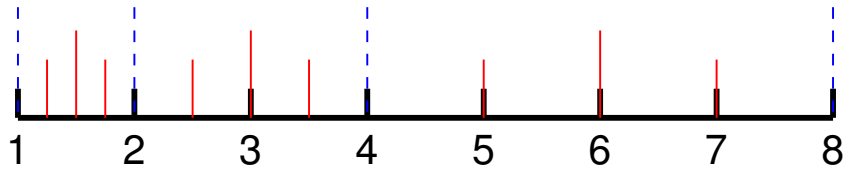
**If this would be true and if we could identify this level of precision, this would be great help for model development and the development of stochastic parametrisation schemes.**

# Information content and reduced precision

**We like to think that the minimal level of precision that can be used is the "information content" of a variable (e.g. for $\pi$).**

**If this would be true and if we could identify this level of precision, this would be great help for model development and the development of stochastic parametrisation schemes.**

**But this is not necessarily the case!**
**Example: Floating point representation of temperature in Kelvin (223-323) and in degree Celsius (-50-50) and model crashes.**

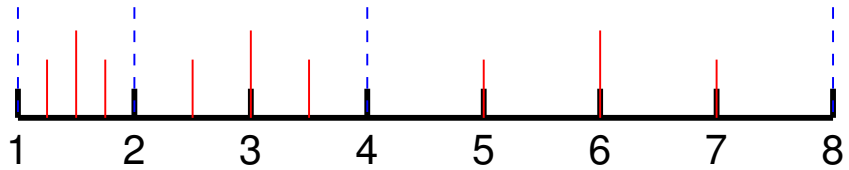


# Information content and reduced precision

**We like to think that the minimal level of precision that can be used is the "information content" of a variable (e.g. for $\pi$).**

**If this would be true and if we could identify this level of precision, this would be great help for model development and the development of stochastic parametrisation schemes.**

**But this is not necessarily the case!**
**Example: Floating point representation of temperature in Kelvin (223-323) and in degree Celsius (-50-50) and model crashes.**



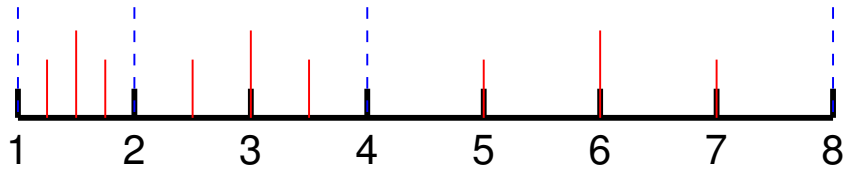

**We need to look into details to understand the information content.**

# Information content and reduced precision

**We like to think that the minimal level of precision that can be used is the "information content" of a variable (e.g. for $\pi$).**

**If this would be true and if we could identify this level of precision, this would be great help for model development and the development of stochastic parametrisation schemes.**

**But this is not necessarily the case!**
**Example: Floating point representation of temperature in Kelvin (223-323) and in degree Celsius (-50-50) and model crashes.**

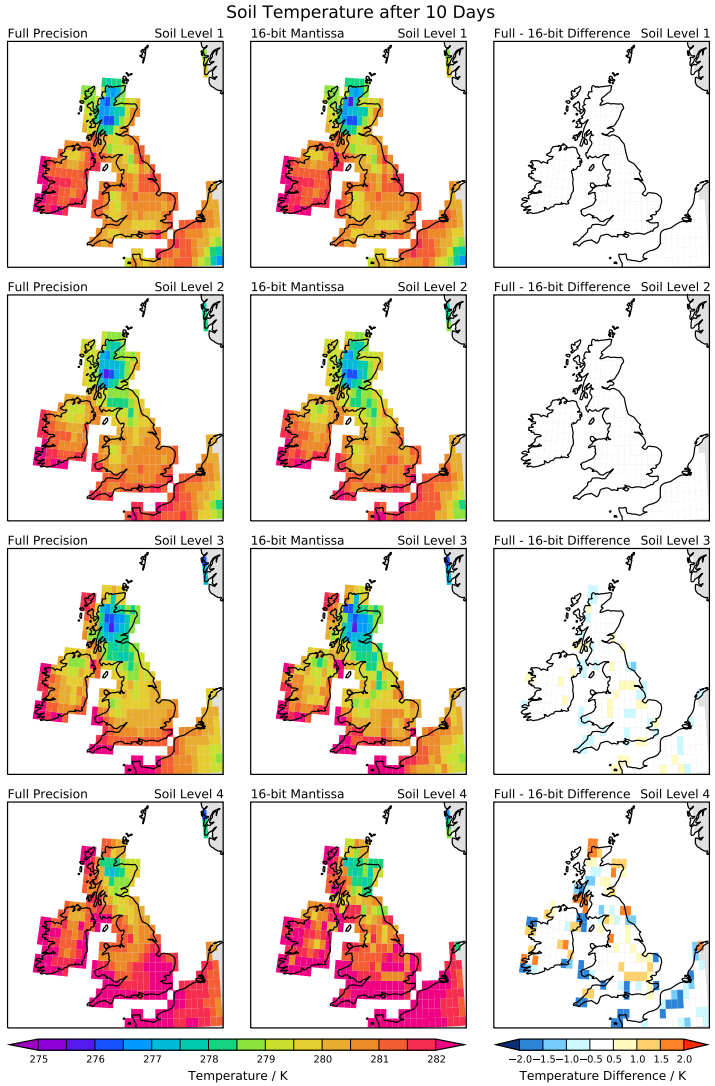



**We need to look into details to understand the information content.**

**We can certainly learn something about the forcings.**

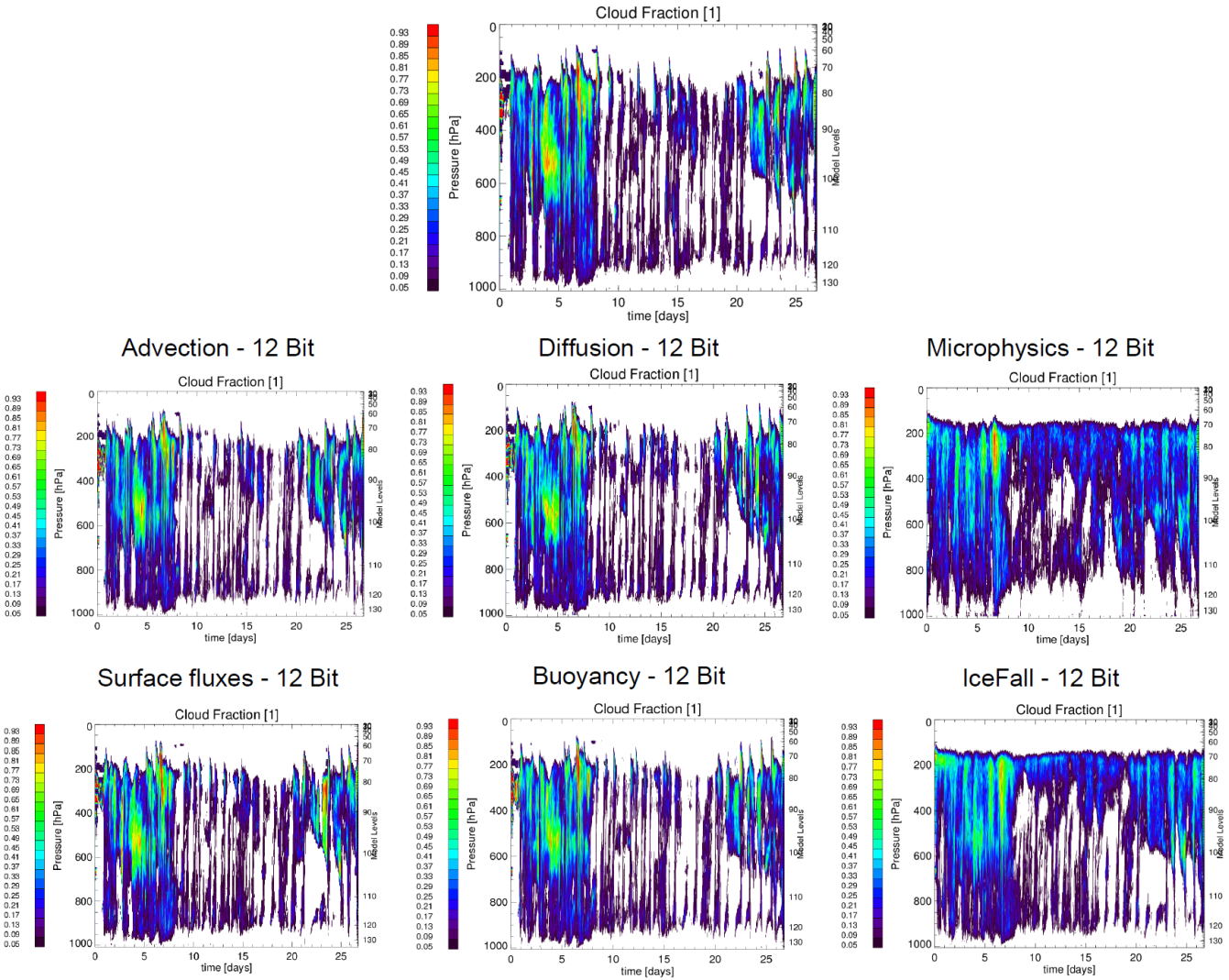
# “Coarse-graining” in large models

Preliminary results with the emulator in the land surface scheme from Andrew Dawson.



# “Coarse-graining” in large models

Preliminary results with the emulator in the cloud resolving model of the superparametrised IFS model from Aneesh Subramanian.



# “Fine-graining” in large models

**It is too expensive to run long term simulation with IFS to identify the right level of precision if we have many different precision levels!**

# “Fine-graining” in large models

**It is too expensive to run long term simulation with IFS to identify the right level of precision if we have many different precision levels!**

**What is a cheap way to identify the minimal level of precision that can be used?**

# How to identify the right level of precision that should be used?

**Tests with Lorenz ’95 show that short-term forecasts (50 timesteps) can provide useful information.**

However:

- Each quantity needs to be checked against it’s own reference.
- Simulations are too sensitive after a couple of timesteps.

# How to identify the right level of precision that should be used?

**Tests with Lorenz ’95 show that short-term forecasts (50 timesteps) can provide useful information.**

However:

- Each quantity needs to be checked against it’s own reference.
- Simulations are too sensitive after a couple of timesteps.

**This is consistent with the seamless prediction approach.**

# My recipe to reduce precision in IFS

1. Introduce the emulator into the model.
2. Use short term simulations of 50 timesteps to “fine-grain” precision levels.
3. Compare each “prognostic quantity” of a subroutine against a high precision reference after 50 timesteps.
4. Find an appropriate quality control and automatise the search.

# Tests with a C-grid shallow-water model

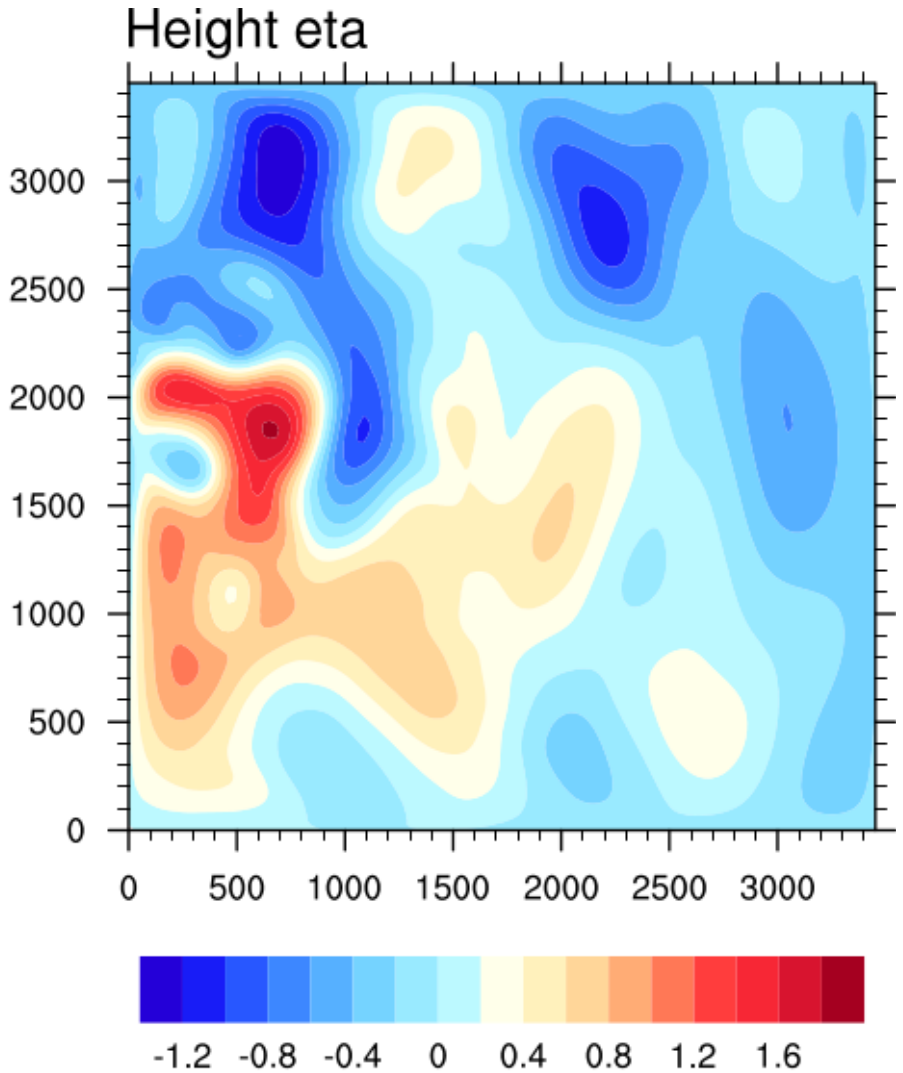
$$\partial_t \mathbf{u} + \mathbf{u} \cdot (\nabla \mathbf{u}) + f\mathbf{k} \times \mathbf{u} + g\nabla\eta = \nu\Delta\mathbf{u} + \boldsymbol{\tau}$$
$$\partial_t \eta + \nabla \cdot (h\mathbf{u}) = 0$$

**u**: horizontal velocity
$f$: Coriolis parameter
**k**: vertical unit vector
$g$: gravitational acceleration

$\eta$: surface elevation
$\nu$: eddy viscosity
$\boldsymbol{\tau}$: forcing term
$h$: height of the fluid column

# Tests with a C-grid shallow-water model

We run a Munk-double-gyre ocean testcase on a 128X128 grid with 500 meter depth.



# Tests with a C-grid shallow-water model

1. Our quality control: The mean difference between the double precision and the reduced precision simulation should be smaller than 0.1% of the standard deviation of $u$, $v$, and $\eta$.
2. We give each floating point field of the timestepping loop (29 in total) an individual level of precision (e.g. $\Delta$t, h, g,...).
3. We run a set of runs for which we reduce precision only for a specific variable. We start with 2 bits precision in the significand for the specific variable and increase this level until the run fulfils the quality control.
4. The results on the following slides are very preliminary!

# Tests with a C-grid shallow-water model

| Parameter | number of bits in the significand |
|---|---|
| $\eta$ | 12 |
| u | 12 |
| v | 12 |
| $\tau_x$ | 2 |
| $\tau_y$ | 2 |
| dh | 2 |
| du | 2 |
| dv | 2 |
| ab | 2 |
| h0 | 2 |
| fu | 4 |
| fv | 4 |
| b | 7 |
| zeta | 2 |
| vec | 2 |
| g | 5 |
| pi | 2 |
| f0 | 2 |
| beta | 2 |
| $\nu$ | 2 |
| ah | 2 |
| x0 | 2 |
| y0 | 2 |
| dx | 8 |
| dy | 8 |
| dt | 2 |
| slip | 2 |
| sigmax | 2 |
| sigmay | 2 |

We end up with precision levels that should be used for the significand of floating point numbers.

Precision can be reduced significantly!

We obtain information on the information content.

# Tests with a C-grid shallow-water model

| Parameter | number of bits in the significand |
|---|---|
| $\eta$ | 12 |
| u | 12 |
| v | 12 |
| $\tau_x$ | 2 |
| $\tau_y$ | 2 |
| dh | 2 |
| du | 2 |
| dv | 2 |
| ab | 2 |
| h0 | 2 |
| fu | 4 |
| fv | 4 |
| b | 7 |
| zeta | 2 |
| vec | 2 |
| g | 5 |
| pi | 2 |
| f0 | 2 |
| beta | 2 |
| $\nu$ | 2 |
| ah | 2 |
| x0 | 2 |
| y0 | 2 |
| dx | 8 |
| dy | 8 |
| dt | 2 |
| slip | 2 |
| sigmax | 2 |
| sigmay | 2 |

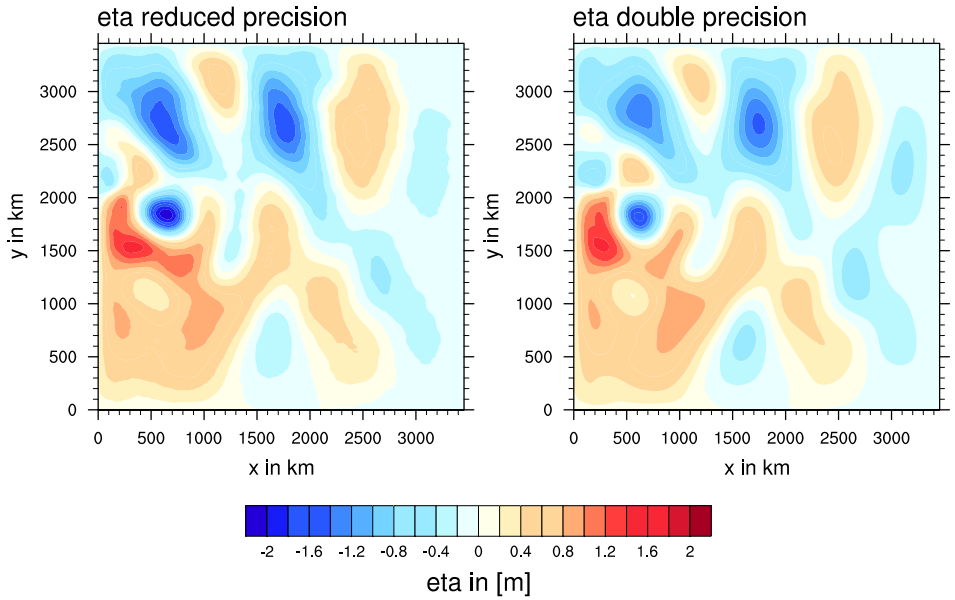
We end up with precision levels that should be used for the significand of floating point numbers.

Precision can be reduced significantly!

We obtain information on the information content.

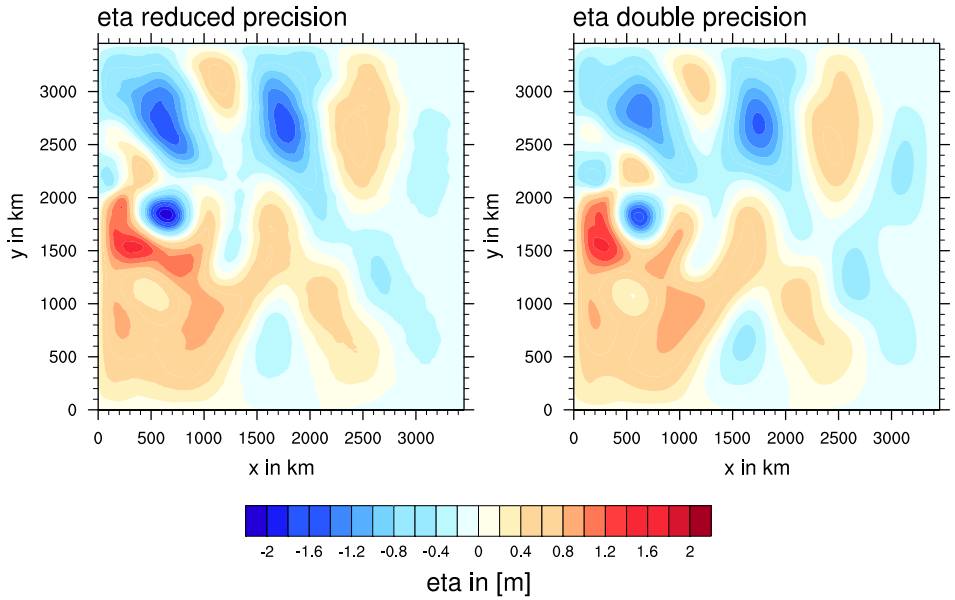
Expert knowledge is needed to obtain stable model simulations (increase precision for $\Delta t$ and ab).

# Tests with a C-grid shallow-water model




Height field after 28 days of simulations.

# Tests with a C-grid shallow-water model




Height field after 28 days of simulations.

**Results are very preliminary and more tests are needed!**

# How to approach full-blown GCMs?

How can we port a model to inexact hardware (whatever this may look like)?

- GCMs already face the challenge to be portable and scalable on different hardware architectures (e.g. GPU/CPU systems).
- If you want maximal performance, a rewrite of the most expensive parts of the model seems to be necessary already for exact hardware (flops/bits, parallelisations, GPUs,...).
- The use of a domain specific language will probably be the key!
- A study of inexact hardware would be much easier in a model which is based on a domain specific language.

# Conclusions 1

- Double precision as default is overcautious in earth system modelling.
- Inexact hardware allows significant savings. Freed resources can improve forecast quality.
- To save power, a reduction of precision is more efficient compared to a reduction in resolution.
- There are several different ways to trade precision against performance in hardware development (Stochastic processors, pruned FPUs, FPGAs, half precision,...).
- Rounding errors can represent sub-grid-scale variability (based on idealised study).
- Rounding errors of inexact hardware can be used to generate ensembles.

# Conclusions 2

- We can probably reduce precision with increasing lead time in weather forecasts.
- Overloaded operators might allow to introduce the emulator for inexact hardware into models of the size of a full GCM.
- Domain-specific languages will be the key to make models portable.

References:
PD Düben, J Joven, A Lingamneni, H McNamara, G De Micheli, KV Palem, TN Palmer, Phil. Trans. A, 2014
PD Düben, TN Palmer, H McNamara, JCP, 2014
TN Palmer, PD Düben, H McNamara, Phil. Trans. A, 2014
PD Düben, TN Palmer, Mon. Weath. Rev., 2014
PD Düben, J Schlachter, Parishkrati, S Yenugula, J Augustine, C Enz, K Palem and TN Palmer, DATE, 2015
F Russell, PD Düben, X Niu, W Luk, TN Palmer, FCCM, 2015
PD Düben, S Jeffress, TN Palmer, EMIT, 2015
PD Düben, SI Dolaptchiev, TCFD, 2015
PD Düben, F Russel, X Niu, W Luk, TN Palmer, submitted to JAMES