



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*



# Work on numerical precision in NEMO

Oriol Tintó Prims  
PhD Candidate

4/04/2019

ESCAPE-2 VVUQ Workshop

A satellite view of the Earth, showing the Mediterranean Sea, the Middle East, and parts of Europe and Africa. The image is positioned on the left side of the slide, with the Earth's curvature visible.

# Outline

- Introduction
- Precision analysis algorithm
- Verifying a modification
- NEMO precision analysis!
- Conclusions

# Introduction



**Barcelona  
Supercomputing  
Center**

Centro Nacional de Supercomputación

# Motivation

Nowadays,  
**only computational models have the potential**  
to provide geographically and physically consistent estimates.

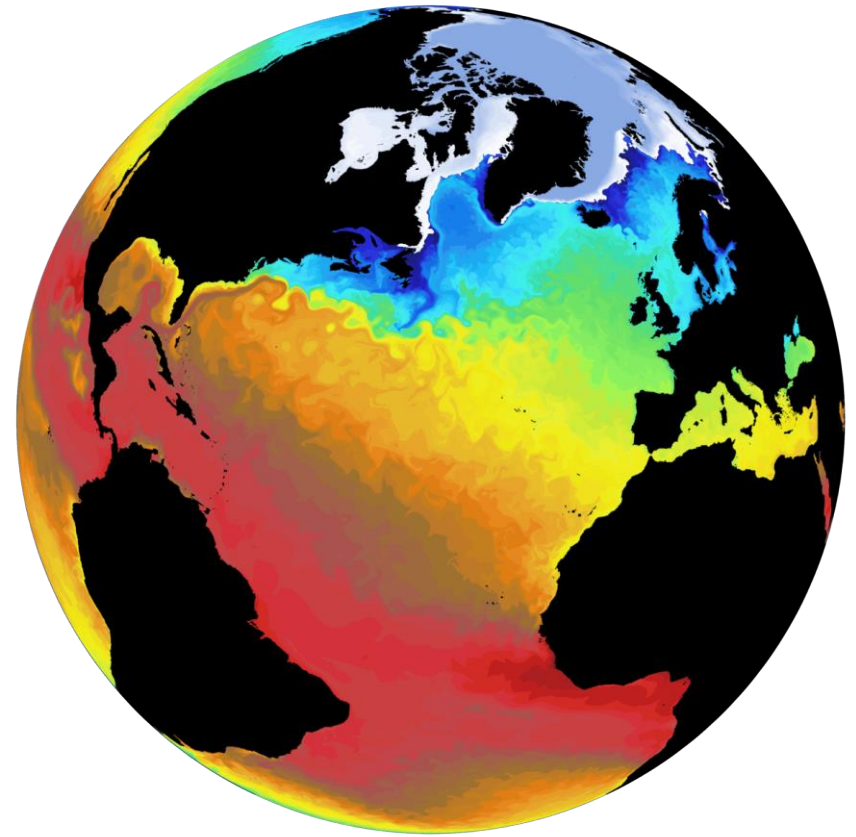


**Nucleus for European Modeling of the Ocean (NEMO)** is a **state-of-the-art** global ocean model

It is used in oceanographic research, operational oceanography, seasonal forecast and climate studies

**The objective of my thesis:**

Improve model's capacity to exploit modern supercomputers.



Sea Surface  
Temperature  
Velocity

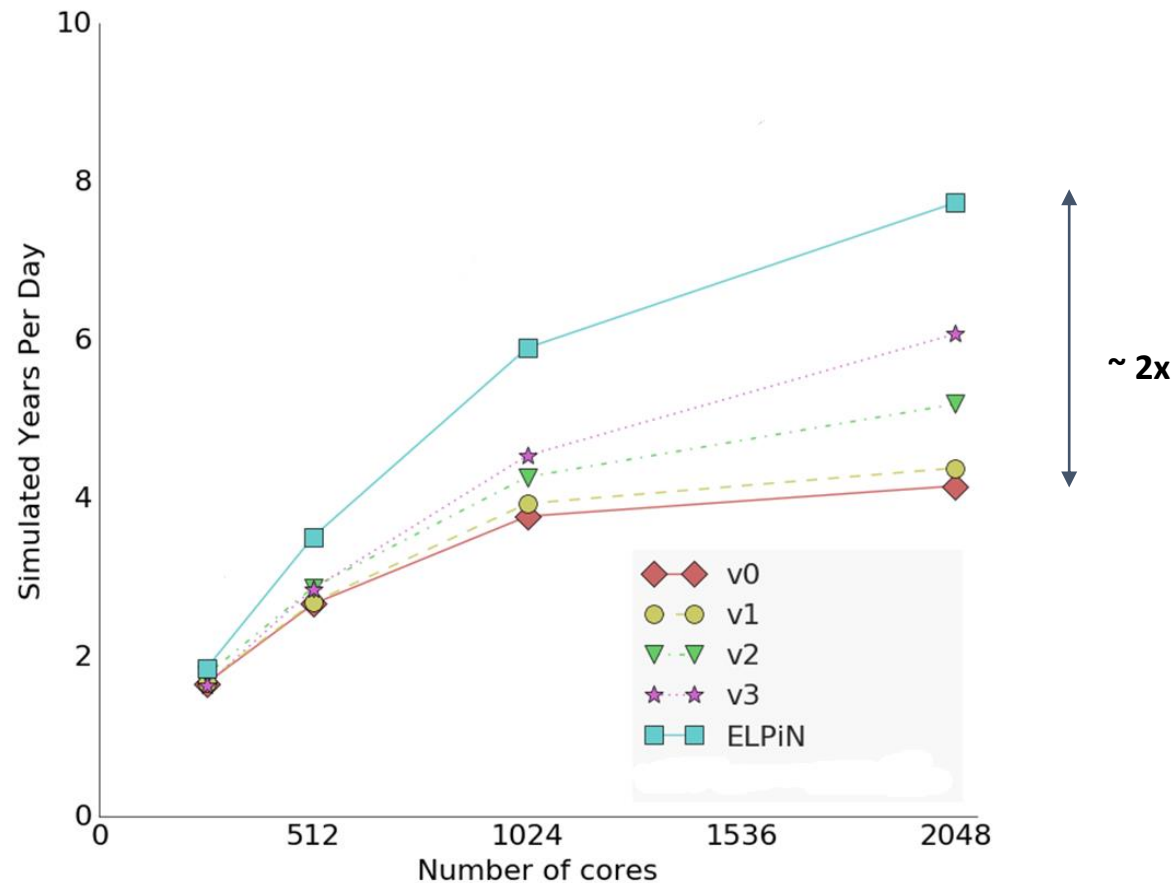
# Motivation

- We want to make models faster and cheaper to:
  - ... reduce costs.
  - ...allow new kinds of experiments.
  - ... better use the available resources.
  - ... make better science!

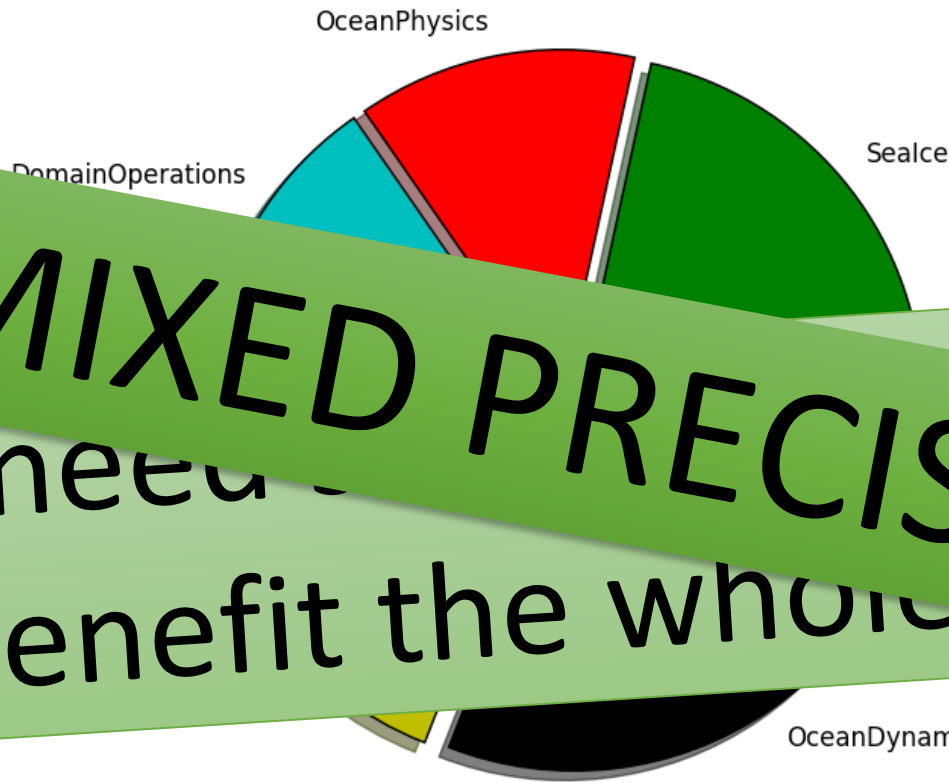
# Motivation

## Previous work:

Impact on ORCA 1/4 NEMO 3.6 – LIM3 running on Marenostrum 3



# Why mixed precision?





# Why mixed-precision has appeared as a performance opportunity?

## How was the situation?

- Reticence to use less precision.
- 64-bit arithmetic performance was comparable to 32-bit performance.

# Risks of low precision



source: <http://www-users.math.umn.edu/~arnold/disasters/ariane.html>

# Why mixed-precision has appeared as a performance opportunity?

## What changed now?

- Lower precision – better performance
- Vendors turning to half-precision arithmetic due to the ML hype.

# What does it mean?

There are smart strategies to **use less precision to solve problems that originally required higher precision.**

Many times, **that's not even necessary** because the higher precision is not required in the first place.

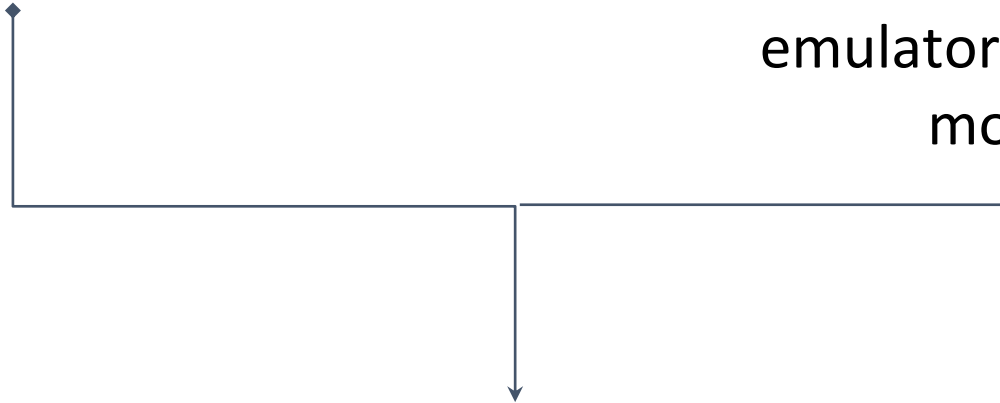
In Earth System models, few alternatives can provide benefits of the same magnitude for the same amount of effort.

We can benefit from reduced precision performance if we learn how to **“safely”** use less precision

# How we can do that?

Being able to verify  
the results

Being able to use the  
reduced precision  
emulator in the full  
model



Perform a precision analysis!

# Precision Analysis Algorithm



**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación

# Analysis Algorithm

- Basic ideas:
  - If we are able to **launch simulations using reduced precision**, we can directly **evaluate the impact it has on the outputs**.
  - If we can find a way to **easily change the precision** used by each real variable, we can develop an **automatic algorithm** to find out which variables can use reduced precision and which ones need to keep high-precision.

# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]





# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4] ?

[5 6 7 8 9] ?

# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4] 

[5 6 7 8 9] 

[0 1 2]  [3 4] 

# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4]

[5 6 7 8 9]

[0 1 2] [3 4]

[3] [4]

# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4]

[5 6 7 8 9]

[0 1 2] [3 4]

[3] [4]

# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4]

[5 6 7 8 9]

[0 1 2] [3 4]

[3] [4]

# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4] ?

[5 6 7 8 9] ✓

[0 1 2] ✓ [3 4] ✓

[3] ✓ [4] ✗

# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4] ✓

[5 6 7 8 9] ✓

[0 1 2] ✓ [3 4] ✓

[3] ✓ [4] ✗

# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4] ✓

[5 6 7 8 9] ✓

[0 1 2] ✓ [3 4] ✓

[3] ✓ [4] ✗



# Analysis Algorithm

[0 1 2 3 4 5 6 7 8 9]



[0 1 2 3 4] ✓

[5 6 7 8 9] ✓

[0 1 2] ✓ [3 4] ✓

[3] ✓ [4] ✗



Variable 4 must be kept in double-precision.

# Validating a simulation



**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación

# ~~Validating a~~ simulation



**Barcelona  
Supercomputing  
Center**

Centro Nacional de Supercomputación

# Verifying a simulation



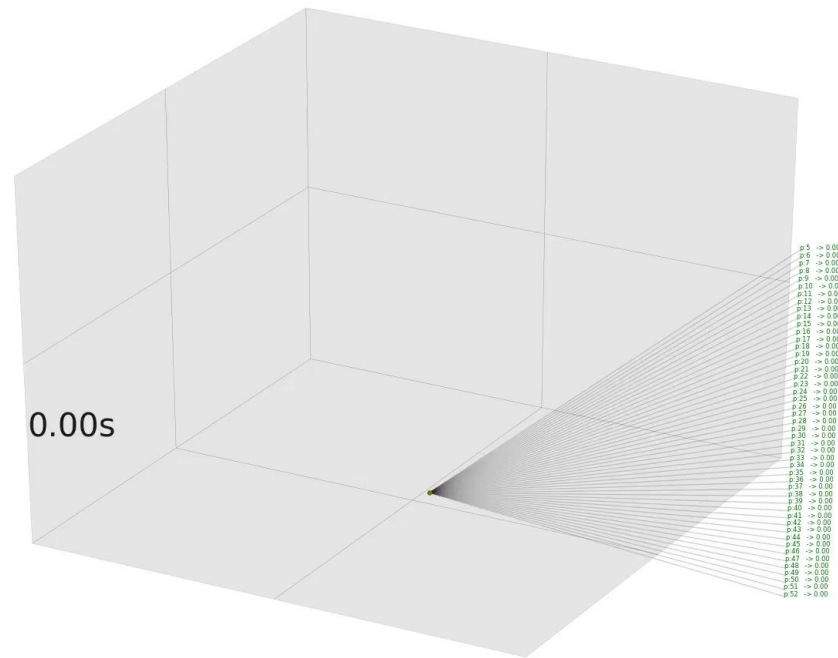
**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación

# Verifying a non-linear model: a simple example

- A simple example:
  - Lorenz system

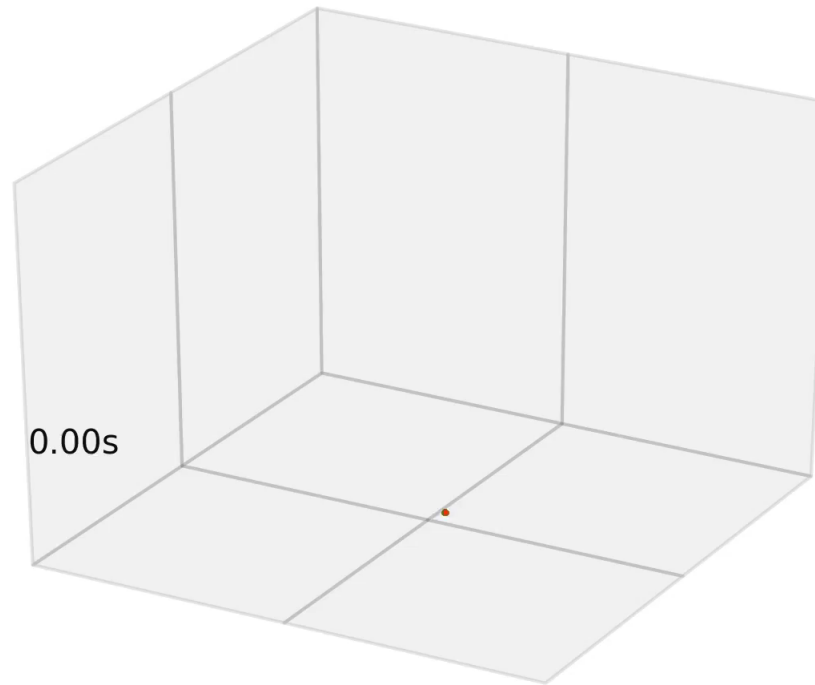
$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= x(\rho - z) - y, \\ \frac{dz}{dt} &= xy - \beta z.\end{aligned}$$

# Verifying a non-linear model: a simple example



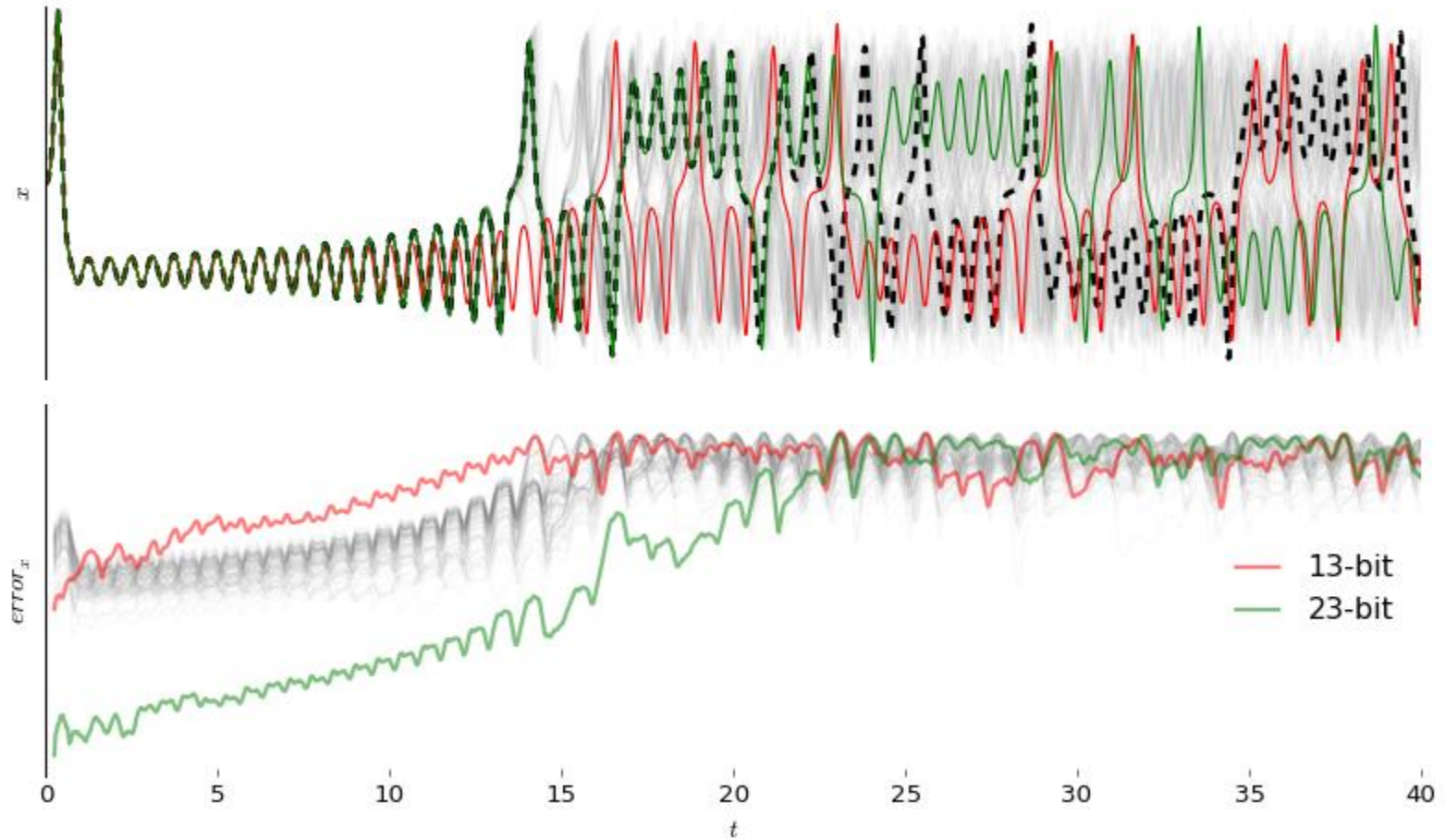
<https://drive.google.com/open?id=1FeVIOX4DL8GIXAAKdAPytQCQgNFdNJ5H>

# Verifying a non-linear model: a simple example



<https://drive.google.com/open?id=14H-2dsZI0zkcGcxiaFIBnMAUdo5kuqHO>

# Verifying a non-linear model: a simple example





# NEMO

## Precision Analysis



**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación

# Reduced Precision Emulator

## Overview

The library contains a derived type: `rpe_var`. This type can be used in place of real-valued variables to perform calculations with floating-point numbers represented with a reduced number of bits in the floating-point significand.

## Basic use of the reduced-precision type

The `rpe_var` type is a simple container for a double precision floating point value. Using an `rpe_var` instance is as simple as declaring it and using it just as you would a real number:

```
TYPE(rpe_var) :: myvar  
  
myvar = 12  
myvar = myvar * 1.287  ! reduced-precision result is stored in `myvar`
```

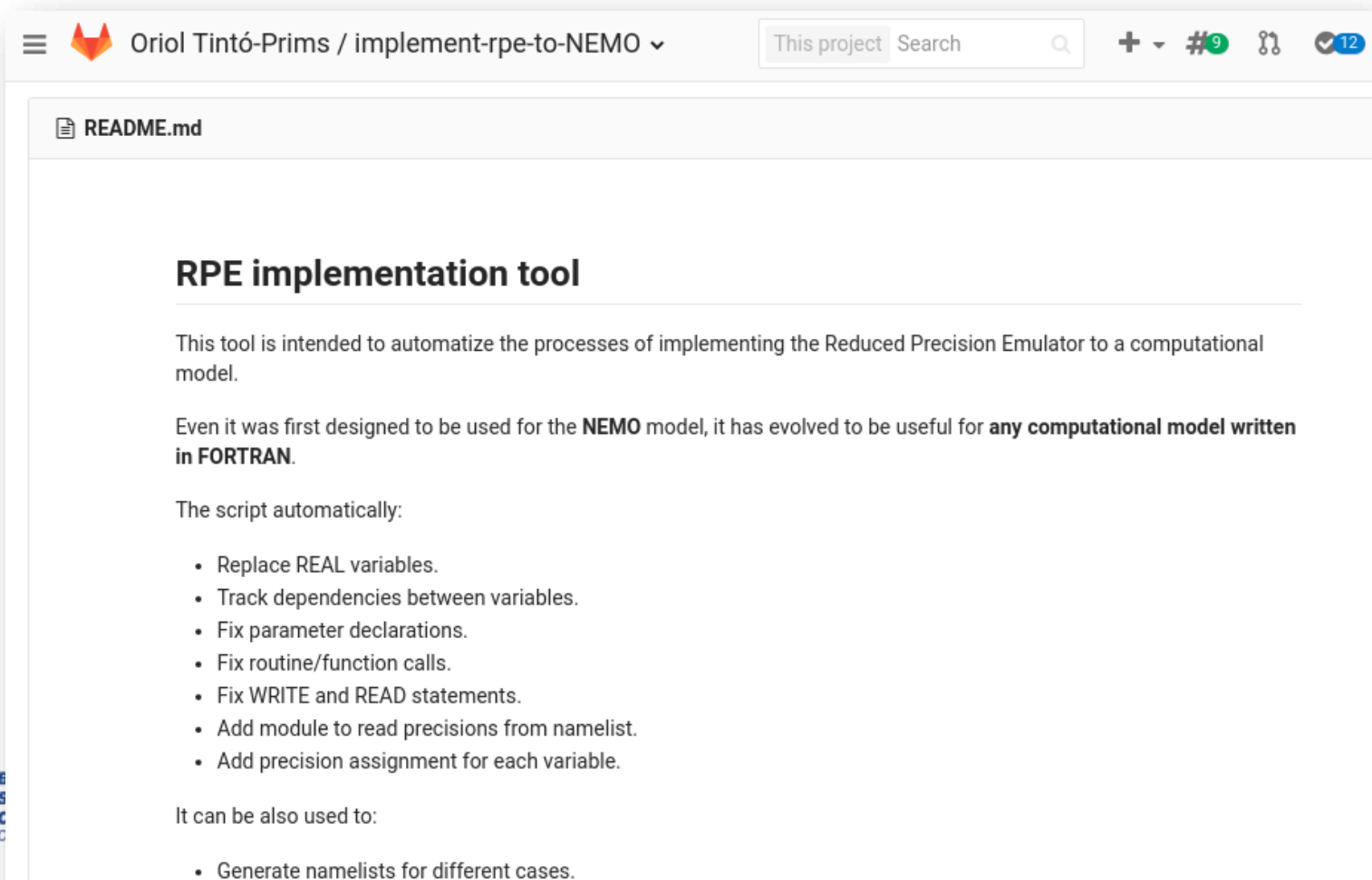
## Controlling the precision


The precision used by reduced precision types can be controlled at two different levels. Each reduced precision variable has an `sbits` attribute which controls the number of explicit bits in its significand. This can be set independently for different variables, and comes into effect after it is explicitly set.

```
TYPE(rpe_var) :: myvar1  
TYPE(rpe_var) :: myvar2  
  
! Use 16 explicit bits in the significand of myvar1, but only 12 in the  
! significand of myvar2.  
myvar1%sbits = 16  
myvar2%sbits = 12
```

# Implementing the emulator

- A Python tool to automate the implementation process was created.



☰  Oriol Tintó-Prims / implement-rpe-to-NEMO ▾ This project Search 🔍 + ▾ #9 🔗 12

📄 README.md

## RPE implementation tool

This tool is intended to automatize the processes of implementing the Reduced Precision Emulator to a computational model.


Even it was first designed to be used for the **NEMO** model, it has evolved to be useful for **any computational model written in FORTRAN**.

The script automatically:

- Replace REAL variables.
- Track dependencies between variables.
- Fix parameter declarations.
- Fix routine/function calls.
- Fix WRITE and READ statements.
- Add module to read precisions from namelist.
- Add precision assignment for each variable.

It can be also used to:

- Generate namelists for different cases.



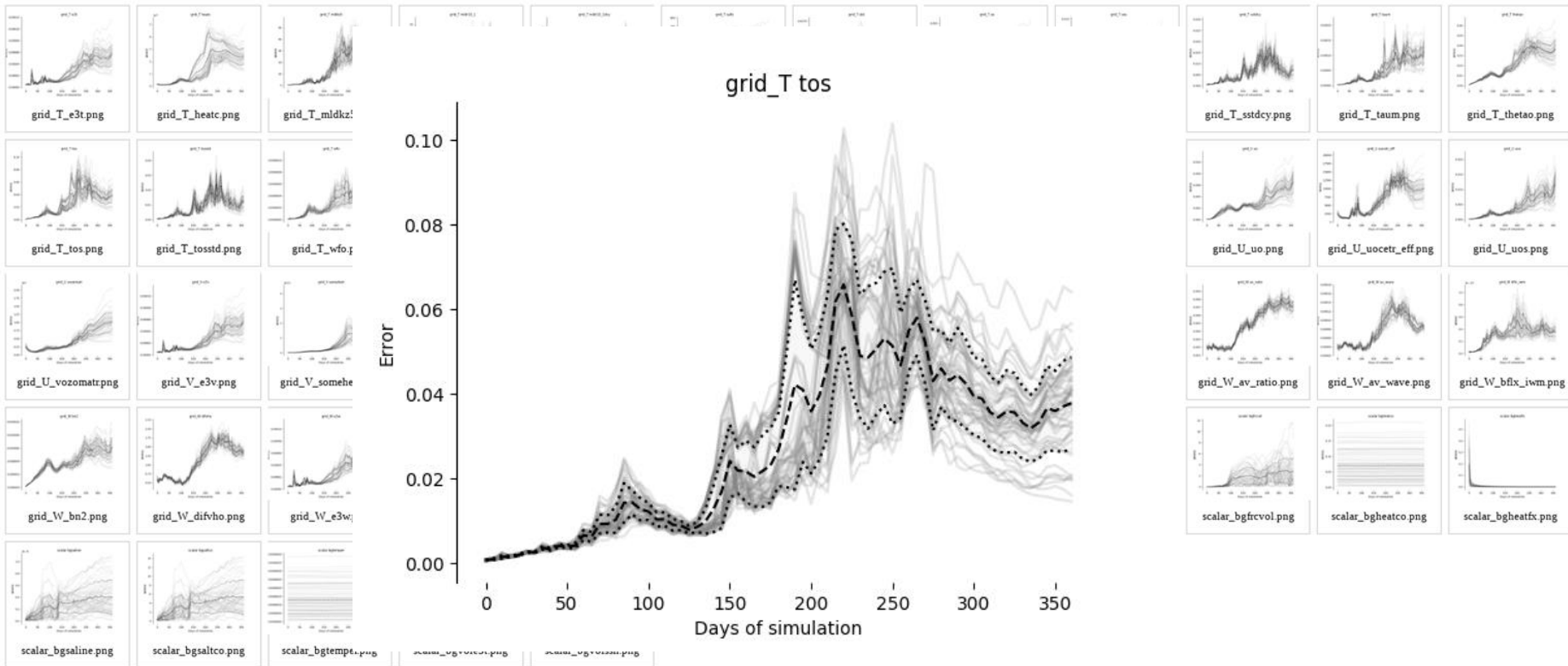
# RPE in NEMO: What we can do with it?

With a **single binary**, we can specify the number of significant bits used for each real variable declaration within the code through a **namelist**.

```
1 ! namelist variable precisions
2 &precisions
3 emulator_variable_precisions(1) = 10 ! Variable: ad_u Routine: ad_balance_tile Module: ad_balance
4 emulator_variable_precisions(2) = 10 ! Variable: ad_v Routine: ad_balance_tile Module: ad_balance
5 emulator_variable_precisions(3) = 10 ! Variable: ad_zeta Routine: ad_balance_tile Module: ad_balance
6 emulator_variable_precisions(5) = 10 ! Variable: pc_r2d Routine: ad_balance_tile Module: ad_balance
7 emulator_variable_precisions(6) = 10 ! Variable: r_r2d Routine: ad_balance_tile Module: ad_balance
8 emulator_variable_precisions(7) = 10 ! Variable: br_r2d Routine: ad_balance_tile Module: ad_balance
9 emulator_variable_precisions(8) = 10 ! Variable: p_r2d Routine: ad_balance_tile Module: ad_balance
10 emulator_variable_precisions(9) = 10 ! Variable: bp_r2d Routine: ad_balance_tile Module: ad_balance
11 emulator_variable_precisions(30) = 10 ! Variable: dTdz Routine: ad_balance_tile Module: ad_balance
12 emulator_variable_precisions(31) = 10 ! Variable: dSdz Routine: ad_balance_tile Module: ad_balance
13 emulator_variable_precisions(32) = 10 ! Variable: ad_gradP Routine: ad_balance_tile Module: ad_balance
14 emulator_variable_precisions(33) = 10 ! Variable: ad_phi Routine: ad_balance_tile Module: ad_balance
15 emulator_variable_precisions(34) = 10 ! Variable: ad_gradPx Routine: ad_balance_tile Module: ad_balance
16 emulator_variable_precisions(35) = 10 ! Variable: ad_gradPy Routine: ad_balance_tile Module: ad_balance
17 emulator_variable_precisions(36) = 10 ! Variable: ad_A Routine: ad_bc_r2d_tile Module: ad_bc_2d
18 emulator_variable_precisions(37) = 10 ! Variable: ad_A Routine: ad_bc_u2d_tile Module: ad_bc_2d
19 emulator_variable_precisions(38) = 10 ! Variable: ad_A Routine: ad_bc_v2d_tile Module: ad_bc_2d
20 emulator_variable_precisions(39) = 10 ! Variable: ad_A Routine: ad_dabc_r2d_tile Module: ad_bc_2d
21 emulator_variable_precisions(40) = 10 ! Variable: ad_A Routine: ad_dabc_u2d_tile Module: ad_bc_2d
22 emulator_variable_precisions(41) = 10 ! Variable: ad_A Routine: ad_dabc_v2d_tile Module: ad_bc_2d
23 emulator_variable_precisions(42) = 10 ! Variable: ad_A Routine: ad_bc_r2d_bry_tile Module: ad_bc_bry2d
24 emulator_variable_precisions(43) = 10 ! Variable: ad_A Routine: ad_bc_u2d_bry_tile Module: ad_bc_bry2d
25 emulator_variable_precisions(44) = 10 ! Variable: ad_A Routine: ad_bc_v2d_bry_tile Module: ad_bc_bry2d
26 emulator_variable_precisions(45) = 10 ! Variable: ad_A Routine: ad_conv_r2d_bry_tile Module: ad_conv_bry2d
27 emulator_variable_precisions(46) = 10 ! Variable: ad_Awkr Routine: ad_conv_r2d_bry_tile Module: ad_conv_bry2d
28 emulator_variable_precisions(47) = 10 ! Variable: ad_FE Routine: ad_conv_r2d_bry_tile Module: ad_conv_bry2d
29 emulator_variable_precisions(48) = 10 ! Variable: ad_FX Routine: ad_conv_r2d_bry_tile Module: ad_conv_bry2d
30 emulator_variable_precisions(49) = 10 ! Variable: Hfac Routine: ad_conv_r2d_bry_tile Module: ad_conv_bry2d
```

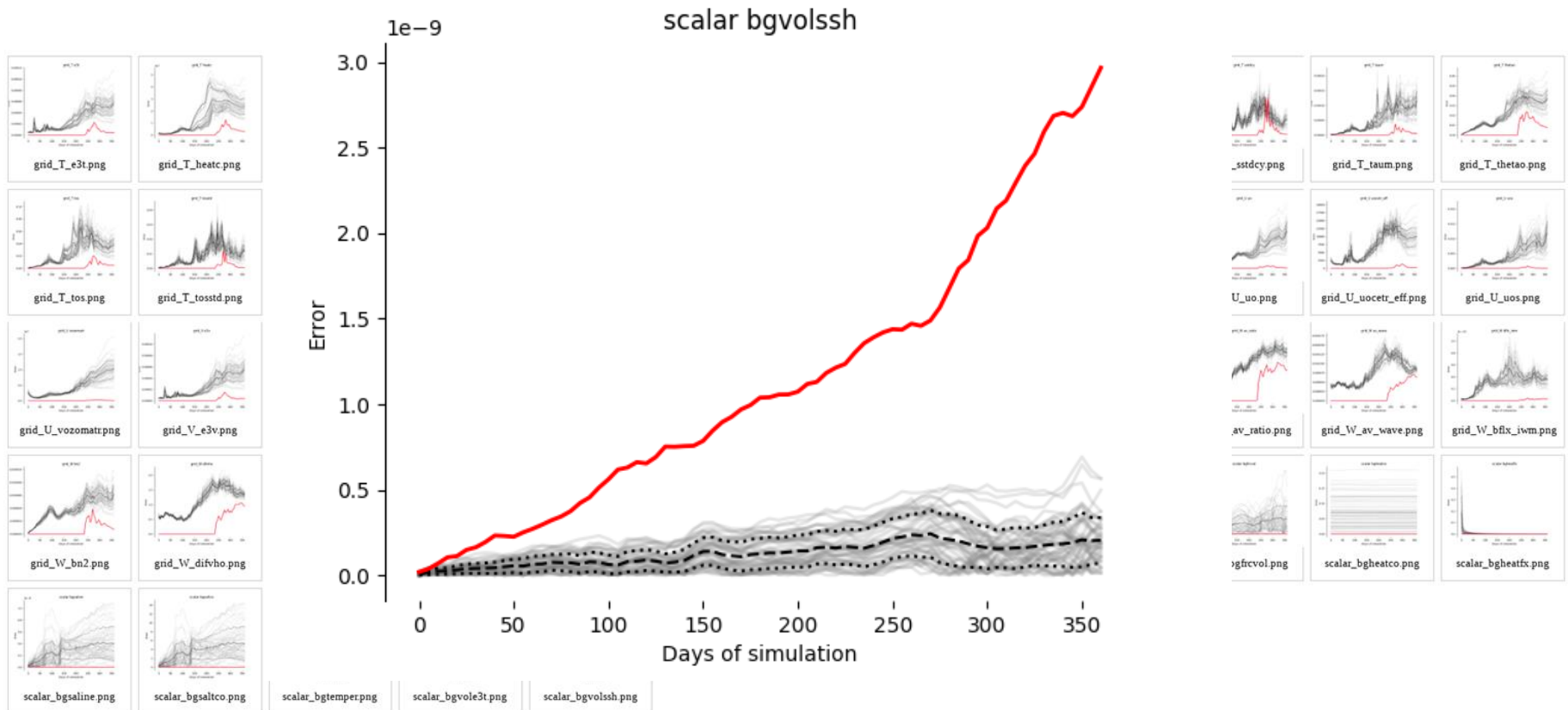
# Verifying NEMO

- Initial conditions perturbed with white noise in the 3D temperature field.
- Evaluating 53 output variables.



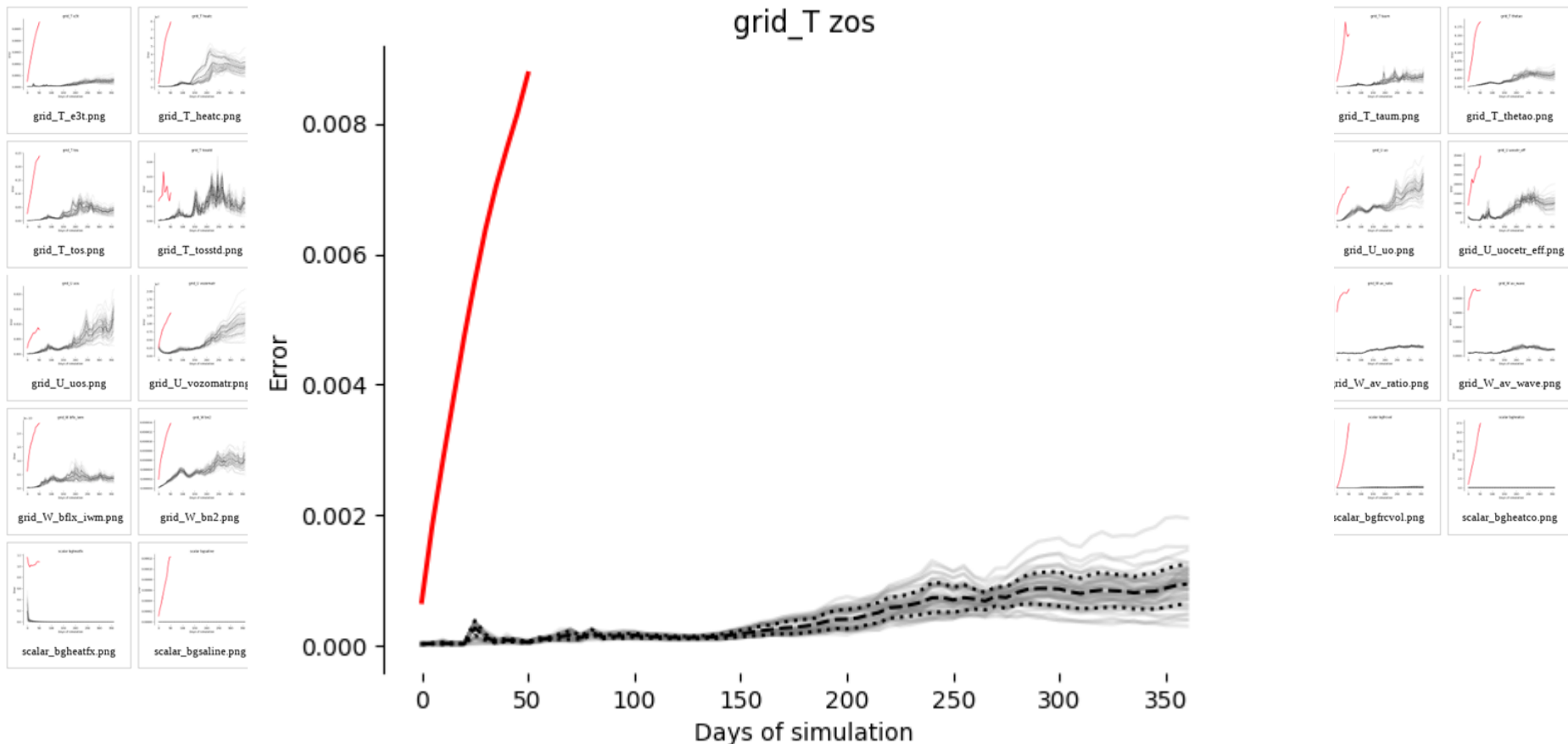
# Verifying NEMO

- Example: Compiling with **-xHost**



# Verifying NEMO

- Example: Everything in single precision:



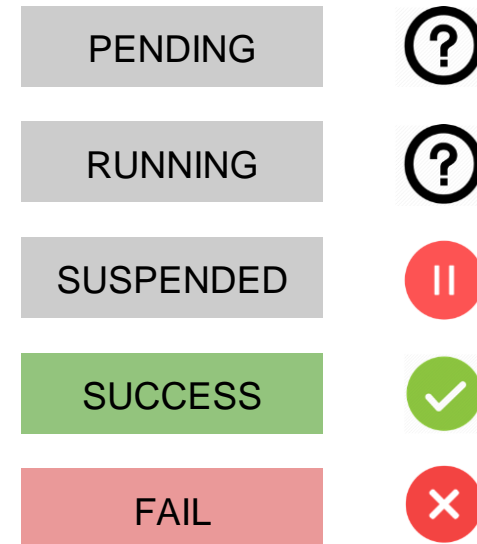
# Analysis algorithm: How is it implemented?

Implementation done in Python:

- **class** Job( variable\_set):
  - Submit job to remote machine.
  - Check job status.
  - Evaluate success.
  - Expand subgroups.
  - Check subgroups.

[0 1 2 3 4 5 6 7 8 9]

[0 1 2 3 4] [ 5 6 7 8 9]

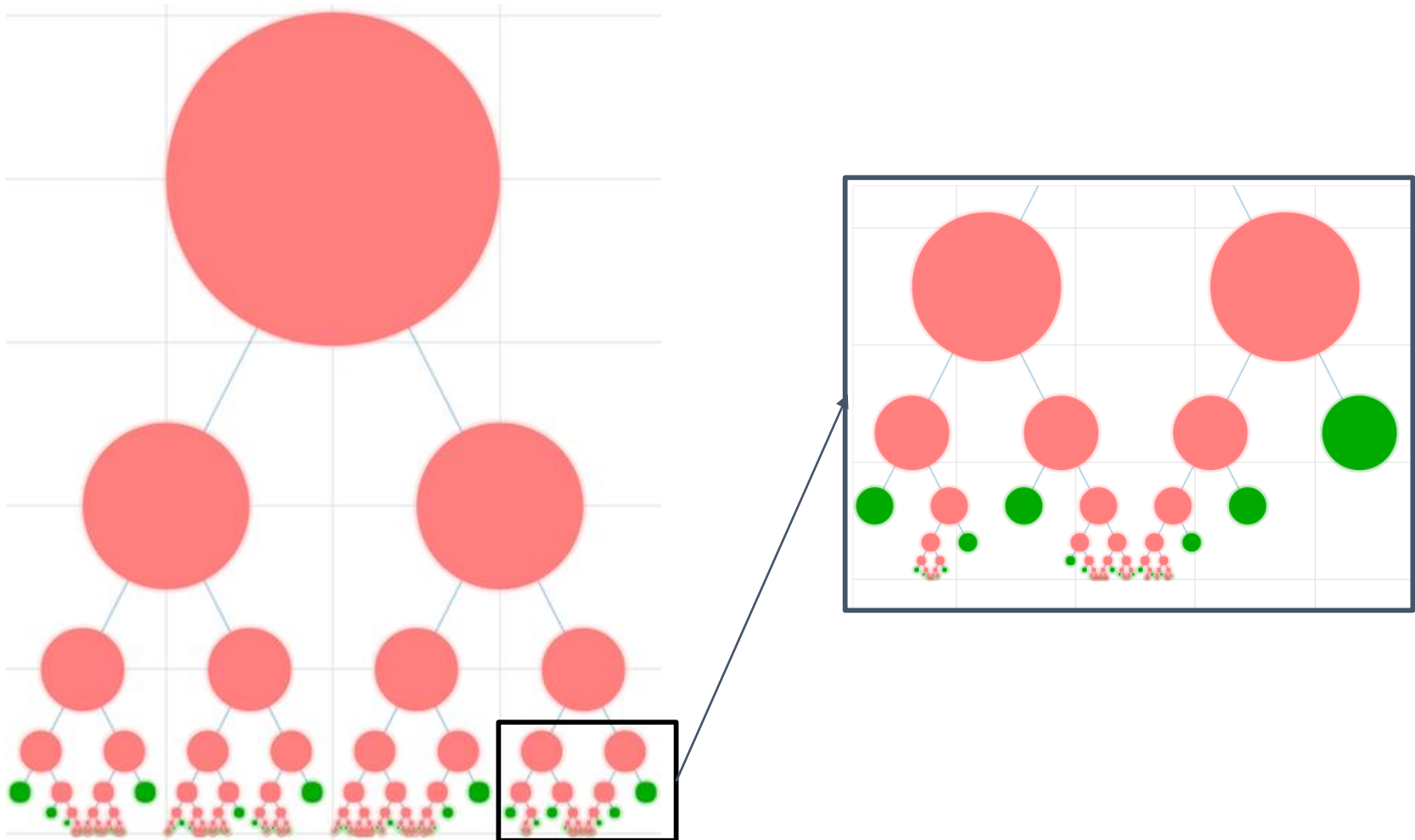




# Analysis algorithm: How is it implemented?

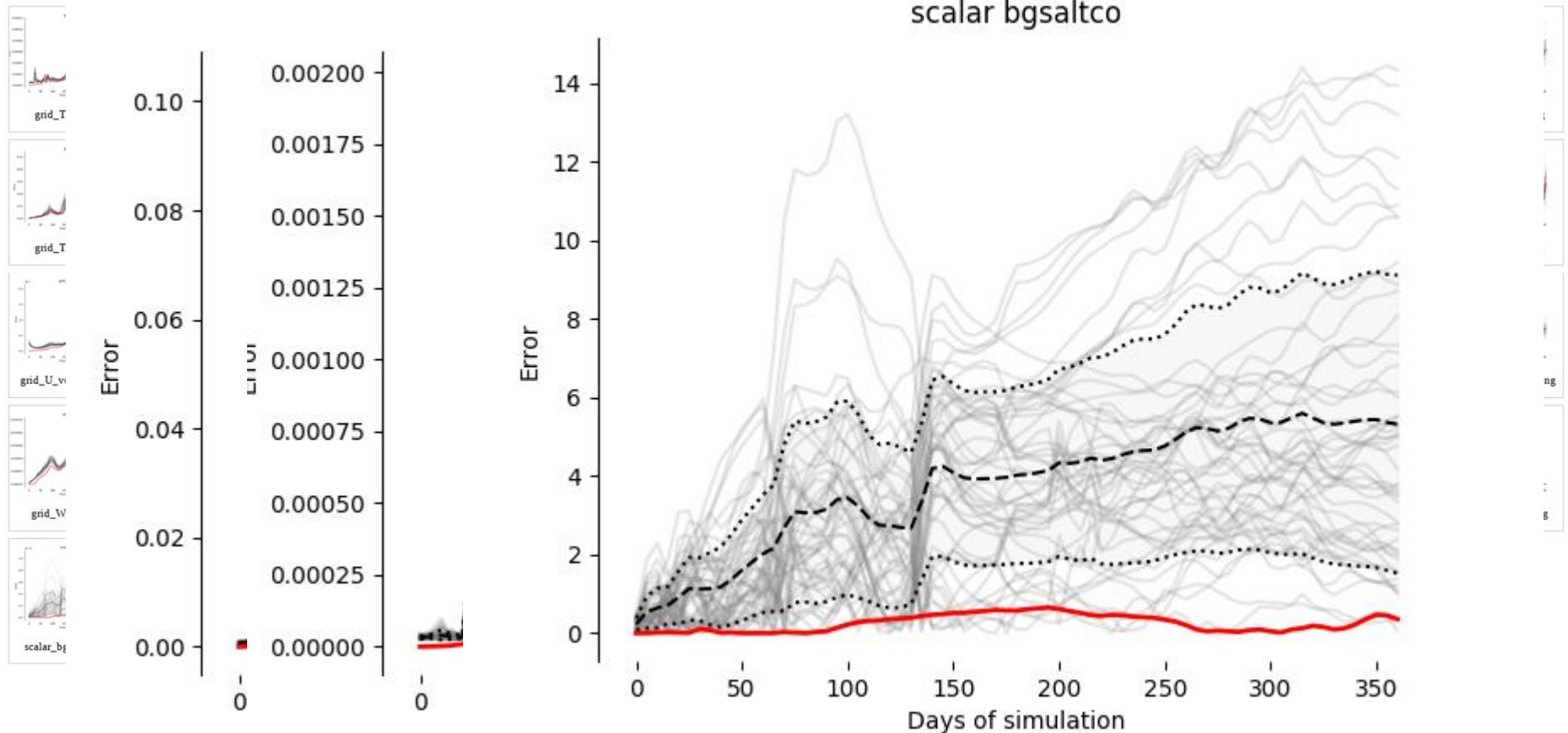
- Some details:
  - We use a simulation template which prepares everything required to execute our test case. The only thing that is modified from one simulation to the other is the precision namelist.
  - The size of the outputs for each individual simulation is about 8GB. To reduce the load to the filesystem the RMSE for all the fields is computed after the simulation and the outputs are deleted. We save a file with a python object which is ~0.5MB.
  - At the beginning of the analysis we don't have many changes to exploit trivial parallelism but at lower steps there are many simulations that can be executed in parallel.

# Precision Analysis



# Results (1)

- Using this verification test to run the analysis algorithm on a small part of the code:

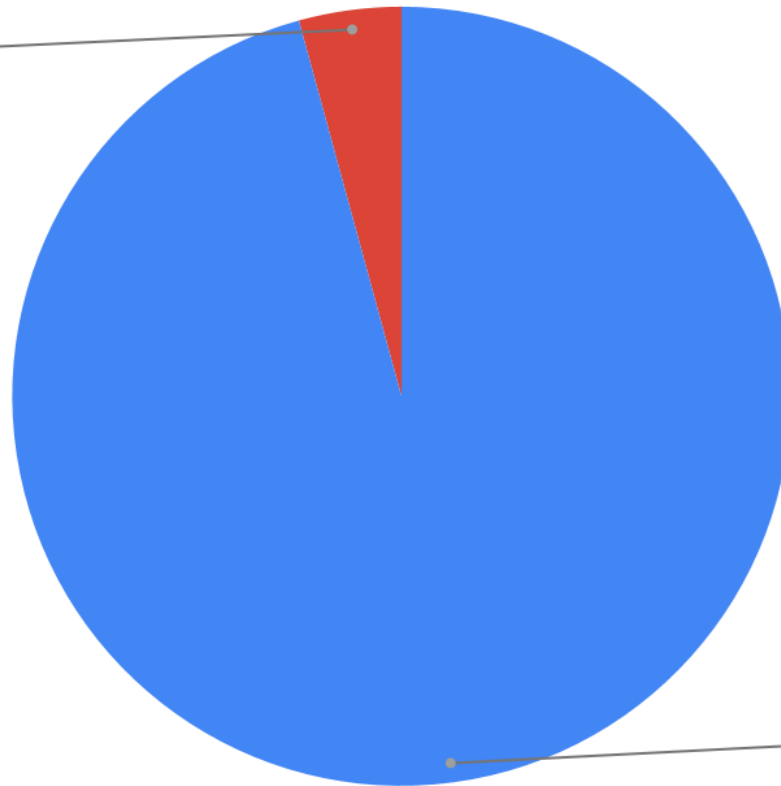


# Results(2)

- Using the old verification test on the full model:

## Results

Double Precision  
4.2%



Single Precision  
95.8%

# Conclusions and open questions



**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación

# Conclusions

- We can find which variables require double precision.
- There's a huge room for reducing the precision used in NEMO.

# Open Questions

- Transferability between different cases?
- Other ways of verifying the results?
- Reducing even more the precision for future architectures?



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*



**EXCELENCIA  
SEVERO  
OCHOA**

# Thank you

[oriol.tinto@bsc.es](mailto:oriol.tinto@bsc.es)