Parallel reproducibility of the SHYFEM-MPI model

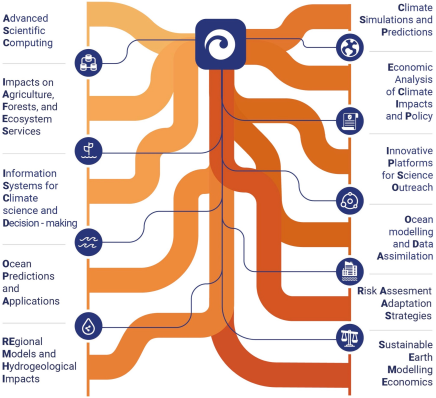
Francesco Carere[1], Giorgio Micaletto[1], Italo Epicoco[1,2], Francesca Mele [1]

November 10 2023,

Workshop on Correctness and Reproducibility for Climate and Weather Software

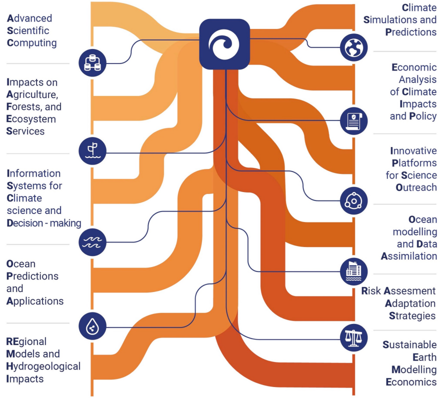
[1]Euro-Mediterranean Center on Climate Change (CMCC), Lecce, Italy
[2]Dep. Engineering for Innovation, University of Salento, Lecce, Italy

CMCC, ASC and SHYFEM-MPI

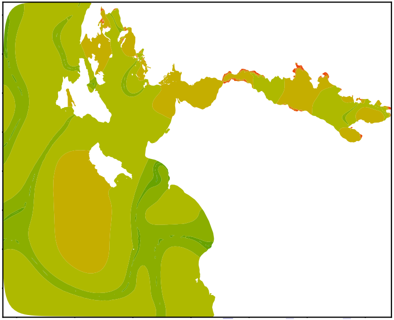


- W&C/society $\xleftrightarrow{\text{interdiscip. sci.}}$ policy
- Science (applied) vs. engineers (SW)

CMCC, ASC and SHYFEM-MPI



- W&C/society $\xleftrightarrow{\text{interdiscip. sci.}}$ policy
- Science (applied) vs. engineers (SW)
- ASC: develop SHYFEM-MPI
- Non-bitwise reproducible (non-BR)



Goal

Divide development in 3 consecutive stages

Goal

Divide development in 3 consecutive stages

Equations $\longrightarrow$ Correct, V&V code $\longrightarrow$ optimised code

Goal

Divide development in 3 consecutive stages

Equations $\longrightarrow$ Correct, V&V code $\longrightarrow$ optimised code

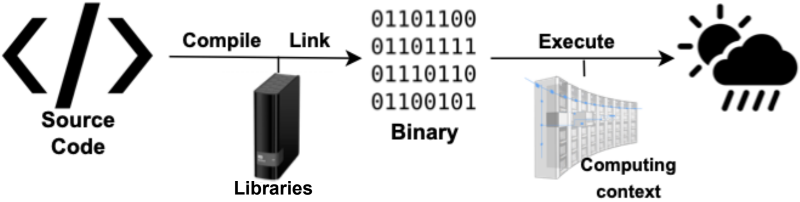
GOAL:

Propose : BR useful for second stage, but **not** needed after optimisation

Losing BR

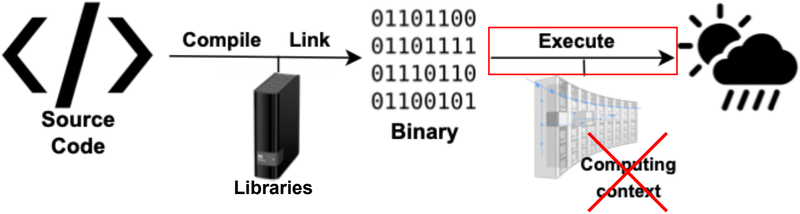
It is easy to lose BR

Losing BR



It is easy to lose BR

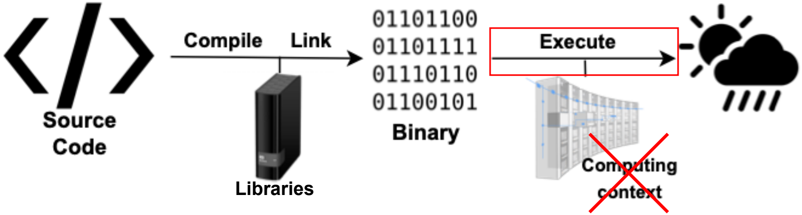
Losing BR



It is easy to lose BR

Our case: parallelized model introduces non-det. during execution/runtime (without changing comp. context)

Losing BR

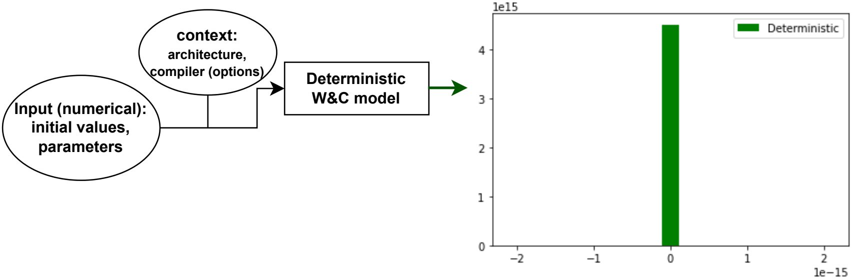


It is easy to lose BR

Our case: parallelized model introduces non-det. during execution/runtime (without changing comp. context)

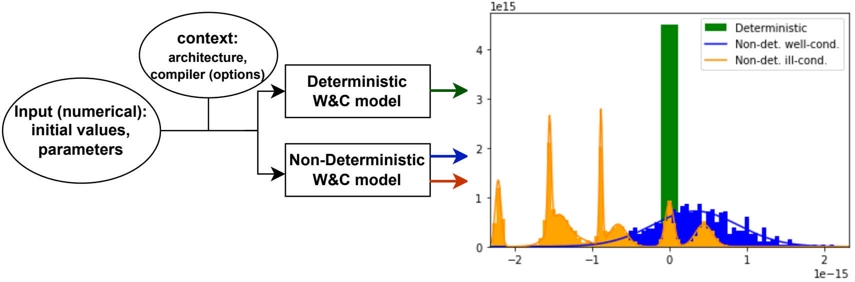
non-BR $\longleftrightarrow$ rounding error

W&C models: probability distributions



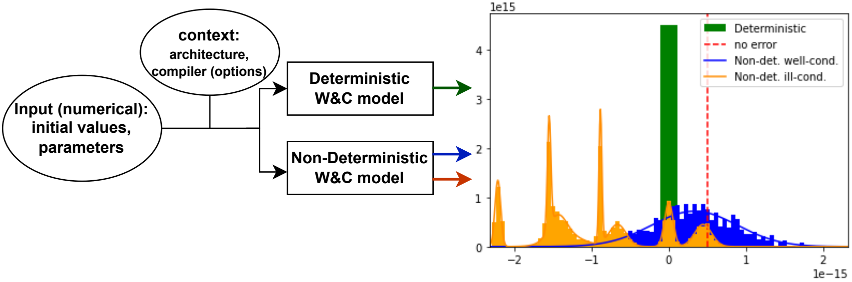
Have sequential model

W&C models: probability distributions



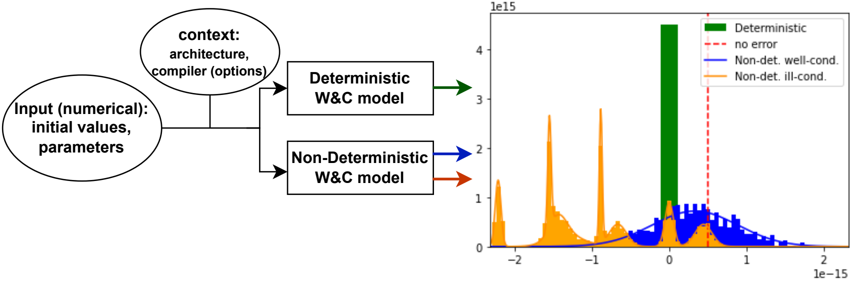
Have sequential model → parallelised (lose BR)

W&C models: probability distributions



Have sequential model → parallelised (lose BR)

W&C models: probability distributions



Have sequential model → parallelised (lose BR)

Parallel model: (part of) rounding error emerges

Non-BR: what to do?

Scientists and engineers unhappy with loss of BR. Solutions?

1. Force back BR (e.g. det. comm., little compiler optim., ...)
2. Reachieve BR (e.g. reproBLAS)
3. No BR (influence BR by e.g. precision [Nhe16; Pic18])

Non-BR: what to do?

Scientists and engineers unhappy with loss of BR. Solutions?

1. Force back BR (e.g. det. comm., little compiler optim., ...)
2. Reachieve BR (e.g. reproBLAS)
3. **No BR** (influence BR by e.g. precision [Nhe16; Pic18])

- Section 1: (dis)advantages of BR
- Section 2: introducing parallel reproducibility via permutations
- Section 3: SHYFEM-MPI

1 BR: when to use it

2 Parallel reproducibility: Statistical approach

3 SHYFEM-MPI

Table of Contents

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	**Scientist**

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	**Scientist**
Debugging	
Verification&Validation	
Regression test	

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	**Scientist**
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	**Scientist**
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive	Slow/inefficient

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	**Scientist**
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive	Slow/inefficient

Left: BR indeed useful. Right: We think optimised (non-BR) code should be used

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	**Scientist**
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive	Slow/inefficient

Left: BR indeed useful. Right: We think optimised (non-BR) code should be used

- Sequential: round-off err. **fixed**. Parallel: **not fixed**

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	Scientist
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive	Slow/inefficient

Left: BR indeed useful. Right: We think optimised (non-BR) code should be used

- Sequential: round-off err. **fixed**. Parallel: **not fixed**
- Right: use correct, validated and verified code

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	Scientist
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive	Slow/inefficient

Left: BR indeed useful. Right: We think optimised (non-BR) code should be used

- Sequential: round-off err. **fixed**. Parallel: **not fixed**
- Right: use correct, validated and verified code
- Lose validity and verification when rounding error not fixed?

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	Scientist
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive	Slow/inefficient

Left: BR indeed useful. Right: We think optimised (non-BR) code should be used

- Sequential: round-off err. **fixed**. Parallel: **not fixed**
- Right: use correct, validated and verified code
- Lose validity and verification when rounding error not fixed?
- Rare behaviour? Decreasing instead of wanting non-BR?

BR: (Dis)advantages

Table: Generally mentioned (dis)advantages of BR

Engineer	**Scientist**
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive	Slow/inefficient

If optimised (non-BR) code is correct, verified, validated.
Use for science!

Bitwise reproducibility or an alternative?

Engineer	Scientist
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive/BR easily lost	Slow/inefficient

BR: useful when developing | Add type of reproducibility

Bitwise reproducibility or an alternative?

Engineer	Scientist
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive/BR easily lost	Slow/inefficient

BR: useful when developing | Add type of reproducibility

Define/measure/influence reproducibility in larger sense than BR?

Bitwise reproducibility or an alternative?

Engineer	Scientist
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive/BR easily lost	Slow/inefficient

BR: useful when developing | Add type of reproducibility

Define/measure/influence reproducibility in larger sense than BR?
In what sense trust (correctness, V&V) parallelised code?

Bitwise reproducibility or an alternative?

Engineer	Scientist
Debugging	Reviewing papers
Verification&Validation	Policy/applications
Regression test	Ill-cond. system (chaot/bifurc)
Restrictive/BR easily lost	Slow/inefficient

BR: useful when developing | Add type of reproducibility

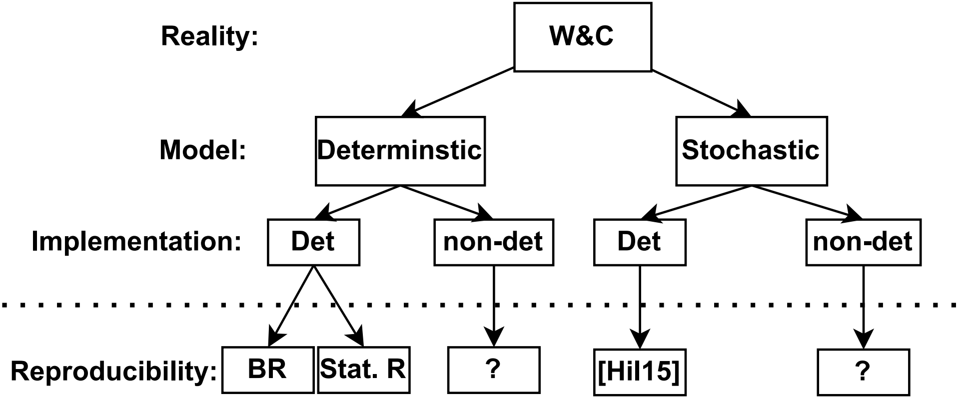
Define/measure/influence reproducibility in larger sense than BR?

In what sense trust (correctness, V&V) parallelised code?

We try a statistical definition (not epistemological)

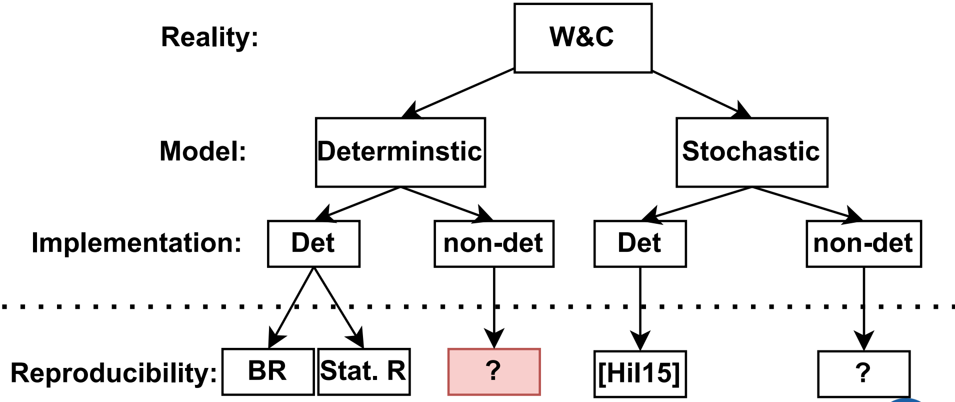
Table of Contents

Statistical framework: reproducibility



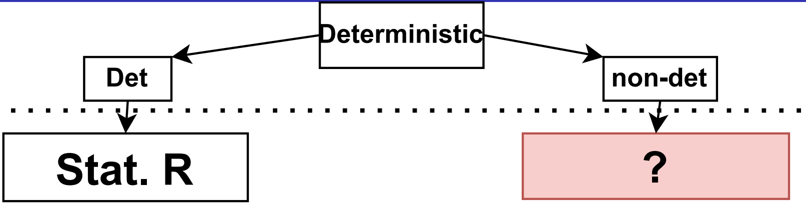
Statistical reproducibility exists [Mah+19] (as for stoch. det. [Hil15])

Statistical framework: reproducibility



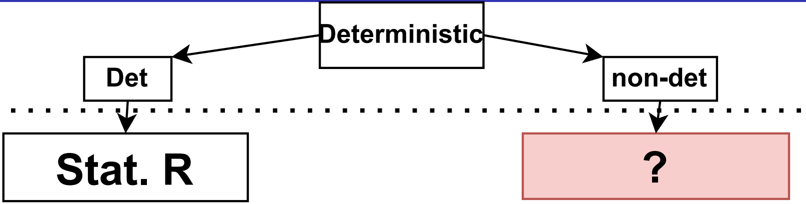
Problem: reproducibility for non-det. implementations of det. models

Parallel reproducibility: first try



1. Two version x, y of code, both det.
2. Vary init. vars, to get samples X_i, Y_i
3. Two-sample test using probability metric $d(\{X_i\}_i, \{Y_i\}_i)$
4. Stat. reproducibility: tolerance for test, e.g. [Mah+19]

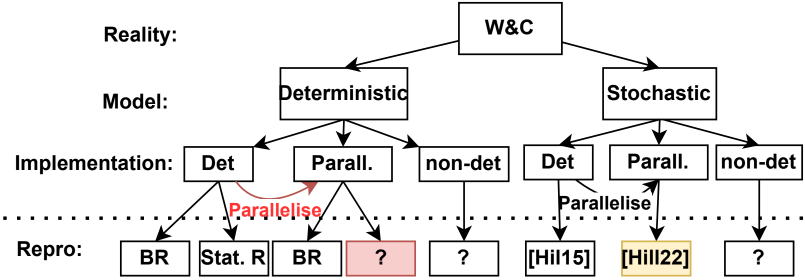
Parallel reproducibility: first try



1. Two version x, y of code, both det.
2. Vary init. vars, to get samples X_i, Y_i
3. Two-sample test using probability metric $d(\{X_i\}_i, \{Y_i\}_i)$
4. Stat. reproducibility: tolerance for test, e.g. [Mah+19]

1. One version of code z, non-det
2. Run multiple times to get sample Z_i
3. One-sample test??
4. Reproducibility?

Statistical framework: **parallel** reproducibility



- Goal: **parallel reproducibility** for parallelised model
- [Hil22] defined/treated it in stochastic case

Example: parallel summation

BLAS $\oplus$ not associative: $(x \oplus y) \oplus z \neq x \oplus (y \oplus z)$

Example: parallel summation

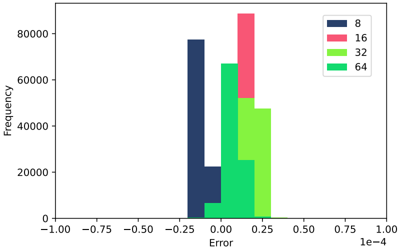
BLAS $\oplus$ not associative: $(x \oplus y) \oplus z \neq x \oplus (y \oplus z)$

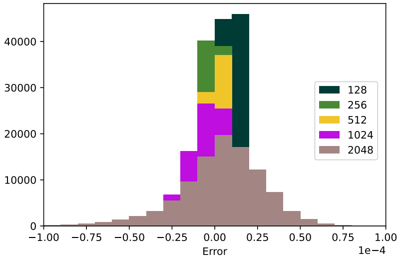
Parallelized evaluation of sum $S := \sum_{i=1}^{n} a_i$ not BR.

Example: parallel summation

BLAS $\oplus$ not associative: $(x \oplus y) \oplus z \neq x \oplus (y \oplus z)$

Parallelized evaluation of sum $S := \sum_{i=1}^{n} a_i$ not BR.





How reproducible is parallel summation?

Parallel summation: rounding error

FLOP $\oplus$ calculates (rel. err. $|\delta_i| \leq \epsilon =$ round. precision)

$$S := \sum_{i=1}^{n} a_i,$$

Parallel summation: rounding error

FLOP $\oplus$ calculates (rel. err. $|\delta_i| \leq \epsilon =$ round. precision)

$$S := \sum_{i=1}^{n} a_i, \quad s_{i+1} := s_i \oplus a_i = (s_i + a_i)(1 + \delta_i), \qquad s_1 := a_1.$$

Parallel summation: rounding error

FLOP $\oplus$ calculates (rel. err. $|\delta_i| \leq \epsilon =$ round. precision)

$$S := \sum_{i=1}^{n} a_i, \quad s_{i+1} := s_i \oplus a_i = (s_i + a_i)(1 + \delta_i), \qquad s_1 := a_1.$$

Either estimated error [Rou16]

$$\textbf{estimated:}\ |S - s_n| \leq \frac{(n-1)\epsilon}{1-(n-1)\epsilon} \sum_{i=1}^{n} |a_i| \quad \text{for } n \leq \epsilon{-}1,$$

Parallel summation: rounding error

FLOP $\oplus$ calculates (rel. err. $|\delta_i| \leq \epsilon =$ round. precision)

$$S := \sum_{i=1}^{n} a_i, \quad s_{i+1} := s_i \oplus a_i = (s_i + a_i)(1 + \delta_i), \qquad s_1 := a_1.$$

Either estimated error [Rou16]

$$\textbf{estimated:}\ |S - s_n| \leq \frac{(n-1)\epsilon}{1-(n-1)\epsilon} \sum_{i=1}^{n} |a_i| \quad \text{for } n \leq \epsilon{-1},$$

or expected [Hen64; Vig93] error:

$$\textbf{expected:}\ S - s_n \sim \mathcal{N}(0, \sqrt{n}\,\sigma) \quad \text{if } \delta_i \sim \mathcal{N}(0,\sigma) \text{ iid } (\sigma = \frac{1}{\sqrt{12}}\epsilon).$$

Parallel reproducibility: try 2

Parallel reproducibility: try 2.

Parallel reproducibility: try 2

Parallel reproducibility: try 2.

Sample P_i, parallel code. Probability distribution S of rounding error.

Method 1:

1. Choose probability metric d and $0 < \alpha < 1$
2. Perform one-sample test between P_i and S (e.g. KS)
3. Accept test if passes with tolerance α

Parallel reproducibility: try 2

Parallel reproducibility: try 2.

Sample P_i, parallel code. Probability distribution S of rounding error.

Method 1:

1. Choose probability metric d and $0 < \alpha < 1$
2. Perform one-sample test between P_i and S (e.g. KS)
3. Accept test if passes with tolerance α

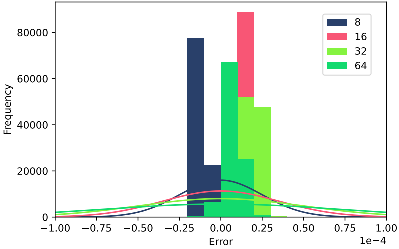
.

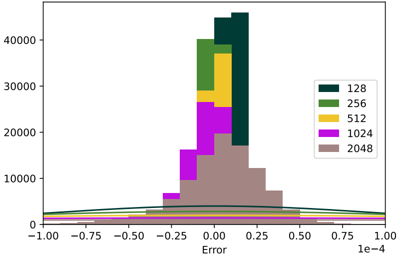
Method 2:

1. Choose tolerance $0 < \alpha < 1$
2. α-confidence interval of mean (of S)
3. Check if P_i lies in α-confidence interval

Parallel summation

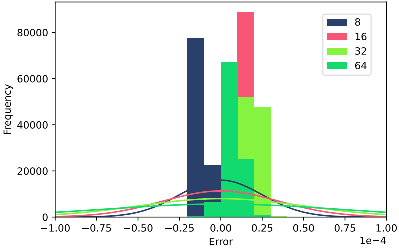
Method 2:

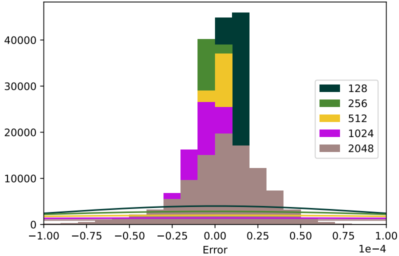




Parallel summation

Method 2:





Method 1:

Kolmogorov-Smirnov test: negative outcome. Not drawn from the same distribution

Rounding-error and non-associativity

Problem with this approach:

Distribution of rounding error often hard to find.

[3] See [PN20] for similar tests

Rounding-error and non-associativity

Problem with this approach:

Distribution of rounding error often hard to find.

Solution

Sample rounding-error. But how?

[3] See [PN20] for similar tests

Rounding-error and non-associativity

Problem with this approach:

Distribution of rounding error often hard to find.

Solution

Sample rounding-error. But how?

Non-associativity → reorder index set **of BR code**[3].

[3] See [PN20] for similar tests

Parallel reproducibility: definition

Samples parallel/sequential P_i and S_i.

Two methods

Two-sample test:

1. Given probability metric d, tolerance $0 < \alpha < 1$
2. Perform two-sample test
3. Accept if p-value smaller than tolerance

Parallel reproducibility: definition

Samples parallel/sequential P_i and S_i.
Two methods

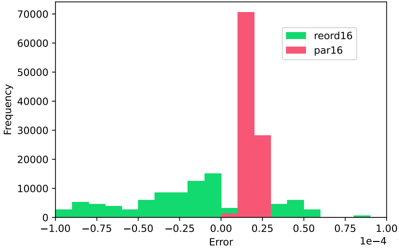
Two-sample test:

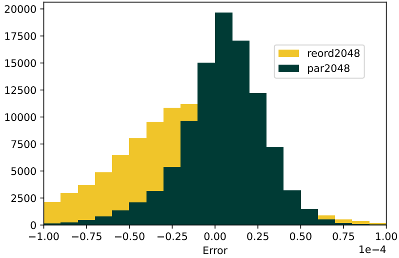
1. Given probability metric d, tolerance $0 < \alpha < 1$
2. Perform two-sample test
3. Accept if p-value smaller than tolerance

Confidence interval

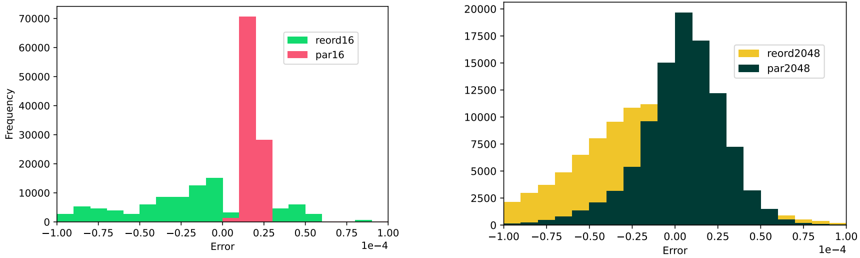
1. Given tolerance $0 < \alpha < 1$
2. Assume $S_i \sim \mathcal{N}(\mu, \sigma)$ (assume CLT)
3. Check if P_i in confidence interval for given tolerance?

Parallel reproducibility for sums





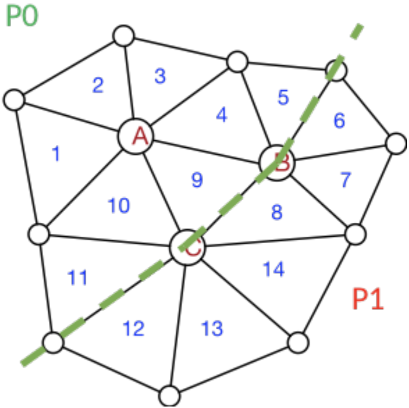
Parallel reproducibility for sums



Again negative KS test: negligible value of hypothesis statistic (not equal)

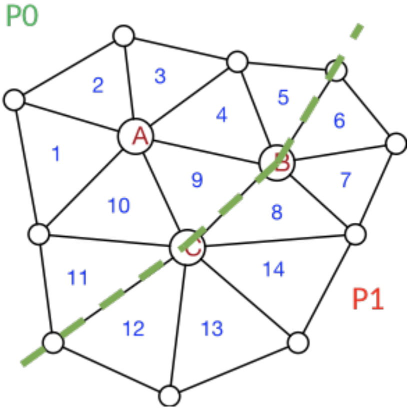
Table of Contents

SHYFEM-MPI: reproducibility



- Domain partitioned
- Communication over boundaries
  Non-associativity → BR

SHYFEM-MPI: reproducibility



- Domain partitioned
- Communication over boundaries
  Non-associativity → BR
- Sample by reordering grid

SHYFEM-MPI: reproducibility

Difference parallel runs (non-BR) 2 causes [Mic+22]:

1. MPI communications: different order of operations in **reductions** and **non-blocking recv-send**
2. Assembly of matrix by PETSc library

SHYFEM-MPI: reproducibility

Difference parallel runs (non-BR) 2 causes [Mic+22]:

1. MPI communications: different order of operations in **reductions** and **non-blocking recv-send**
2. Assembly of matrix by PETSc library

Differences between sequential and parallel executions:

1. Different order of floating point operations (regardless of communications)
2. Internal optimization of PETSc
3. Compiler optimization (out of order execution, FMA, vectorization)

SHYFEM-MPI: reproducibility

Difference parallel runs (non-BR) 2 causes [Mic+22]:

1. MPI communications: different order of operations in **reductions** and **non-blocking recv-send**
2. Assembly of matrix by PETSc library

Differences between sequential and parallel executions:

1. Different order of floating point operations (regardless of communications)
2. Internal optimization of PETSc
3. Compiler optimization (out of order execution, FMA, vectorization)

Parallel reproducibility: multivariate case

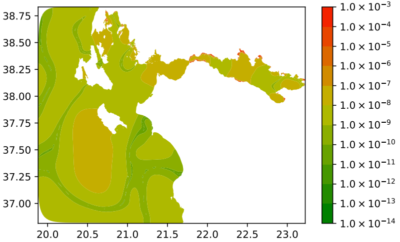


Figure: L_1 norm between parallel run and ensemble average of SST

Look at case study

- Grid: Zakynthos island
- #Processes fixed

Parallel reproducibility: multivariate case

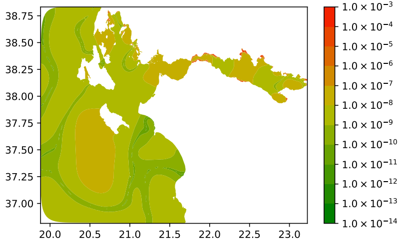


Figure: L_1 norm between parallel run and ensemble average of SST

Look at case study

- Grid: Zakynthos island
- #Processes fixed
- Interested in parallel reproducibility
- Ensemble runs (sequential: reordering)
- Long run (weather: 1 year)

Parallel reproducibility: multivariate case

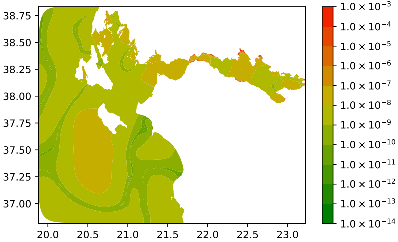


Figure: L_1 norm between parallel run and ensemble average of SST

Look at case study

- Grid: Zakynthos island
- #Processes fixed
- Interested in parallel reproducibility
- Ensemble runs (sequential: reordering)
- Long run (weather: 1 year)

Have statistical distribution over space and time.

Parallel reproducibility: Multivariate case

Two possibilities:

Multivariate version:

two-sample test and **confidence radius**

Parallel reproducibility: Multivariate case

Two possibilities:

Multivariate version:

two-sample test and **confidence radius**

Pointwise version &reduce:

two-sample test and **confidence interval** as above. Then reduce to one-dimension

Parallel reproducibility: Confidence interval

For every point have 2σ confidence interval of the mean

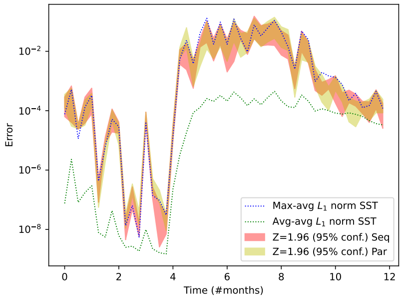


Figure: L_1 norm between ensemble and ensemble-mean. (Maximum and average over grid)

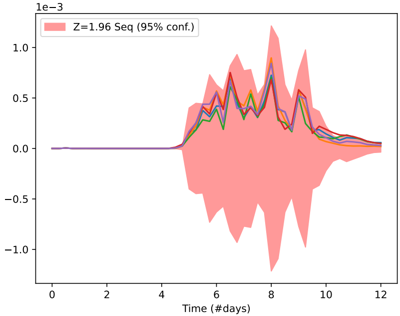


Figure: 90th percentile largest (over grid) 2σ confidence interval (over ensemble), and L_1 error of ensemble runs

Parallel reproducibility

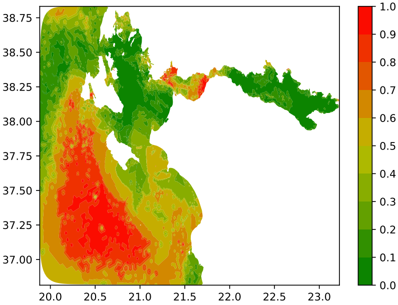


Figure: Kolmogorov-Smirnov test at final time

Define reproducibility using grid?

Parallel reproducibility: thoughts

Define/measure/influence reproducibility in larger sense than BR?

- Two-sample test ($S = P$) not right
- Confidence interval good: parallel code P "more reproducible" than seq. code S. In some sense

$$P \leq S.$$

Parallel reproducibility: thoughts

Define/measure/influence reproducibility in larger sense than BR?

- Two-sample test ($S = P$) not right
- Confidence interval good: parallel code P "more reproducible" than seq. code S. In some sense

$$P \leq S.$$

In what sense trust (correctness, V&V) parallelised code?

Parallel reproducibility: thoughts

Define/measure/influence reproducibility in larger sense than BR?

- Two-sample test ($S = P$) not right
- Confidence interval good: parallel code P "more reproducible" than seq. code S. In some sense

$$P \leq S.$$

In what sense trust (correctness, V&V) parallelised code? (sequential)?

Parallel reproducibility: thoughts

Define/measure/influence reproducibility in larger sense than BR?

- Two-sample test ($S = P$) not right
- Confidence interval good: parallel code P "more reproducible" than seq. code S. In some sense

$$P \leq S.$$

In what sense trust (correctness, V&V) parallelised code? (sequential)?

- If $X \leq Y$ V&V follows from the sequential code.

Good for well-conditioned systems (sum, SHYFEM-MPI without turbulence)

Thoughts and Conclusion

Conclusion

- BR useful for development of correct,V&V code. Should be relaxed when optimising code

Thoughts and Conclusion

Conclusion

- BR useful for development of correct,V&V code. Should be relaxed when optimising code
- Reproducibility of non-BR code via confidence interval test. Found by simulating (by reordering) round-off error in BR code
- V&V of non-BR code follows if confidence interval test good
- SHYFEM-MPI: reproducible in case study of long-time integration, if we use 90% percentile largest 2σ confidence interval

Thoughts and Conclusion

Conclusion

- BR useful for development of correct,V&V code. Should be relaxed when optimising code
- Reproducibility of non-BR code via confidence interval test. Found by simulating (by reordering) round-off error in BR code
- V&V of non-BR code follows if confidence interval test good
- SHYFEM-MPI: reproducible in case study of long-time integration, if we use 90% percentile largest 2σ confidence interval
- All of the above: No chaos/well-conditioned

Thoughts and Conclusion

Conclusion

- BR useful for development of correct,V&V code. Should be relaxed when optimising code
- Reproducibility of non-BR code via confidence interval test. Found by simulating (by reordering) round-off error in BR code
- V&V of non-BR code follows if confidence interval test good
- SHYFEM-MPI: reproducible in case study of long-time integration, if we use 90% percentile largest 2σ confidence interval
- All of the above: No chaos/well-conditioned

Thoughts:

- **Parallel reproducibility**: Useful if BR code in development $\rightarrow$ non-BR in optimisation (not necessarily parallelisation)

References I

[Hen64] Peter Henrici. “Elements of numerical analysis”. In: *(No Title)* (1964).

[Hil15] David RC Hill. “Parallel random numbers, simulation, and reproducible research”. In: *Computing in Science & Engineering* 17.4 (2015), pp. 66–71.

[Hil22] David RC Hill. “Reproducibility of simulations and High Performance Computing”. In: *ESM 2022, European Simulation and Modelling Conference*. 2022, pp. 5–9.

References II

[Mah+19] Salil Mahajan et al. "A multivariate approach to ensure statistical reproducibility of climate model simulations". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. 2019, pp. 1–10.

[Mic+22] G. Micaletto et al. "Parallel implementation of the SHYFEM (System of HydrodYnamic Finite Element Modules) model". In: *Geoscientific Model Development* 15.15 (2022), pp. 6025–6046. DOI: 10.5194/gmd-15-6025-2022. URL: https://gmd.copernicus.org/articles/15/6025/2022/.

References III

[Nhe16] Rafife Nheili. "How to improve the numerical reproducibility of hydrodynamics simulations: analysis and solutions for one open-source HPC software". PhD thesis. Université de Perpignan Via Domita, 2016.

[Pic18] Romain Picot. "Amélioration de la fiabilité numérique de codes de calcul industriels". PhD thesis. Sorbonne université, 2018.

[PN20] Samuel D Pollard and Boyana Norris. "A Statistical Analysis of Error in MPI Reduction Operations". In: *2020 IEEE/ACM 4th International Workshop on Software Correctness for HPC Applications (Correctness)*. IEEE. 2020, pp. 49–57.

References IV

[Rou16] Pierre Roux. “Formal Proofs of Rounding Error Bounds: With Application to an Automatic Positive Definiteness Check”. In: *Journal of Automated Reasoning* 57 (2016), pp. 135–156.

[Vig93] Jean Vignes. “A stochastic arithmetic for reliable scientific computation”. In: *Mathematics and computers in simulation* 35.3 (1993), pp. 233–261.