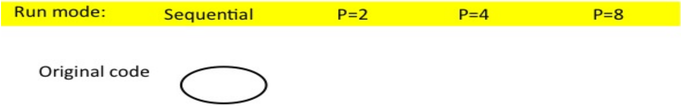
Webminar on Reproducible Research: Numerical Reproducibility
May 3rd, 2016, Grenoble, France

Reproducible Finite Element Simulation: A Case Study

Philippe Langlois, Rafife Nheili

DALI, Université de Perpignan Via Domitia
LIRMM, UMR 5506 CNRS - Université de Montpellier

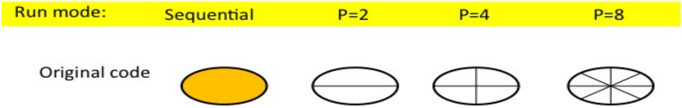
DALI, Digits, Architectures et Logiciels Informatiques


Run mode:
Sequential
P=2
P=4
P=8
Original code

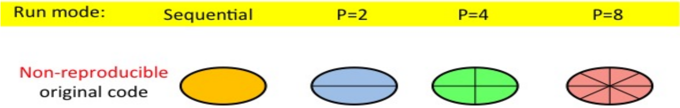
Bitwise identical results for every p-parallel run, $p \geq 1$ II

Run mode:	Sequential	P=2	P=4	P=8
Original code				

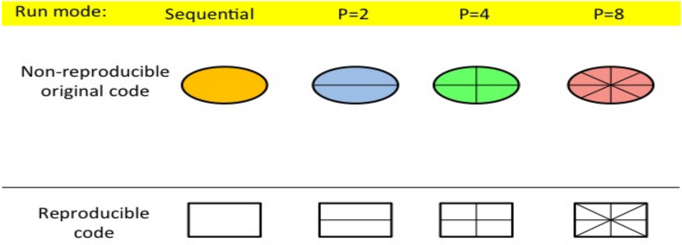
Bitwise identical results for every p-parallel run, $p \geq 1$ III



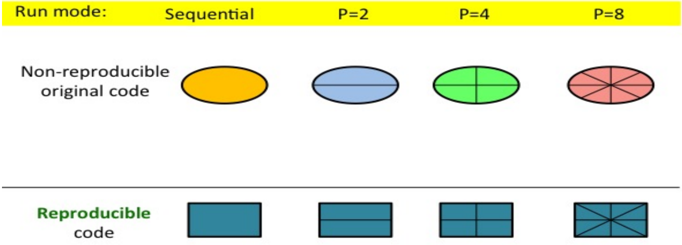
Bitwise identical results for every p-parallel run, $p \geq 1$ IV



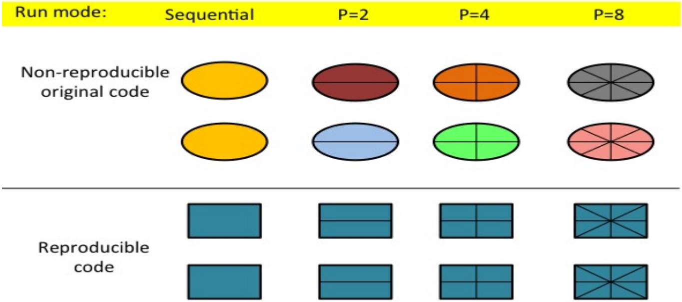
Bitwise identical results for every p-parallel run, $p \geq 1$ V



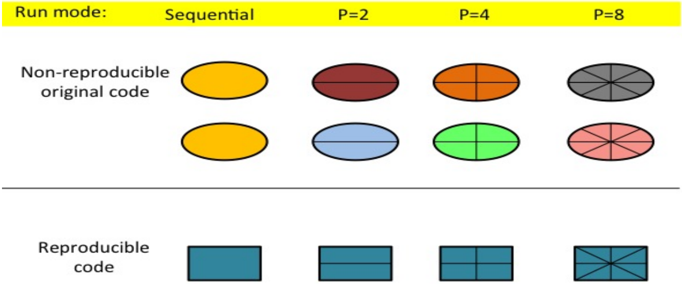
Bitwise identical results for every p-parallel run, $p \geq 1$ VI



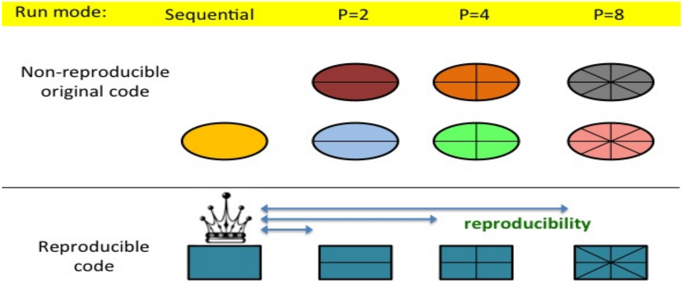
Bitwise identical results for every p-parallel run, $p \geq 1$ VII



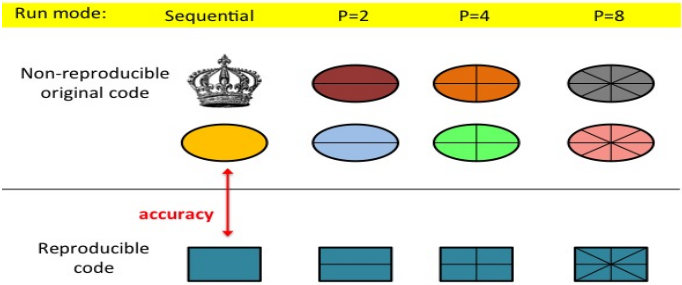
Bitwise identical results for every p-parallel run, $p \geq 1$ VIII



Bitwise identical results for every p-parallel run, $p \geq 1$ IX



Bitwise identical results for every p-parallel run, $p \geq 1$ X



Reproducibility failure of one industrial scale simulation code

- Simulation of free-surface flows in 1D-2D-3D hydrodynamics
- 300 000 loc. of open source Fortran 90
- 20 years, 4000 registered users, EDF R&D + international consortium

Telemac 2D [3]

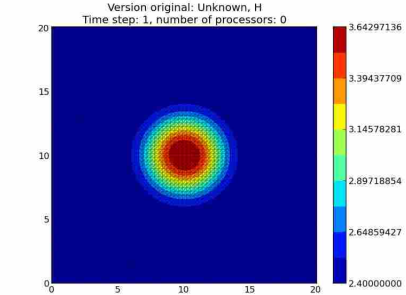
- 2D hydrodynamic: Saint Venant equations
- Finite element method, triangular element mesh, sub-domain decomposition for parallel resolution
- Mesh node unknowns: water depth (H) and velocity (U,V)

Telemac2D: the simplest gouttedo simulation

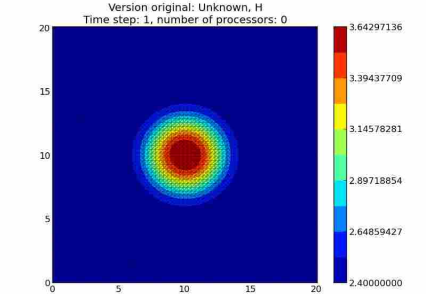
The gouttedo test case

- 2D-simulation of a water drop fall in a square bassin
- Unknown: water depth for a 0.2 sec time step
- Triangular mesh: 8978 elements and 4624 nodes

Expected numerical reproducibility (time step = 1, 2, ...)



Sequential

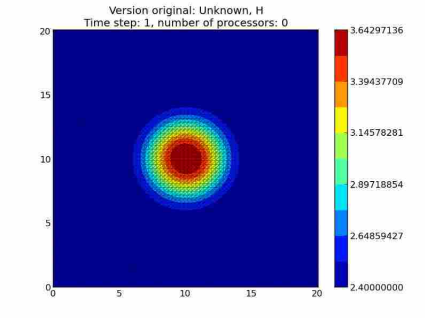


Parallel $p = 2$

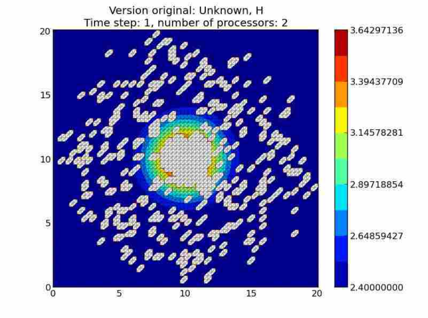
A white plot displays a non-reproducible value

Numerical reproducibility?

time step $= 1$



Sequential

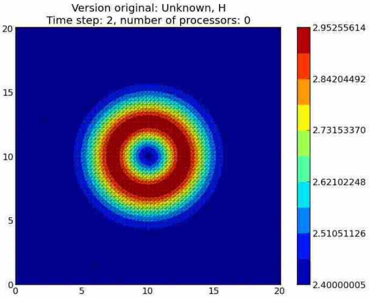


Parallel $p = 2$

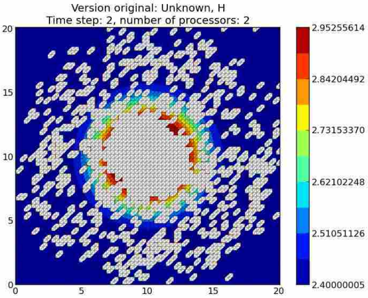
A white plot displays a non-reproducible value

Numerical reproducibility?

time step $= 2$



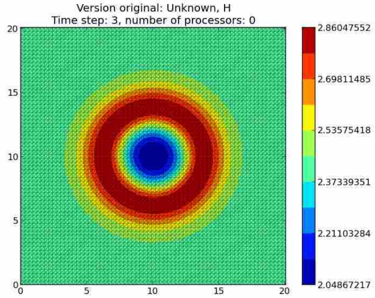
Sequential



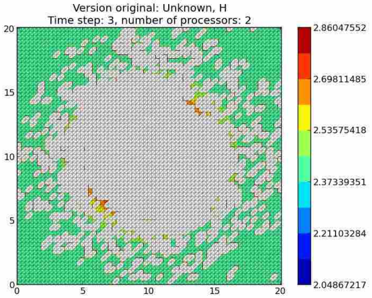
Parallel $p = 2$

Numerical reproducibility?

time step $= 3$



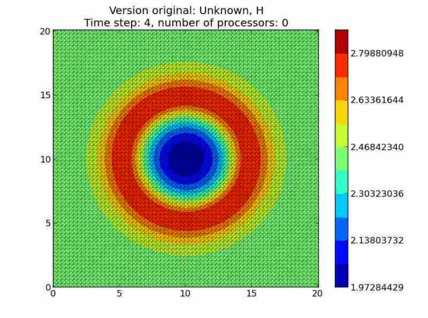
Sequential



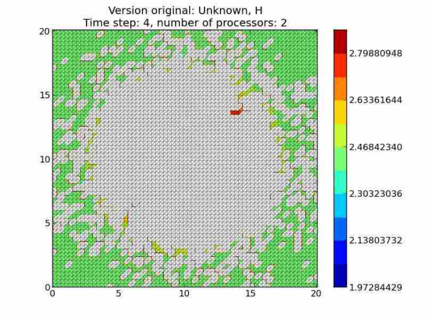
Parallel $p = 2$

Numerical reproducibility?

time step = 4



Sequential

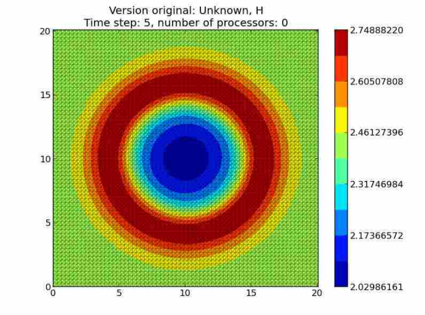


Parallel $p = 2$

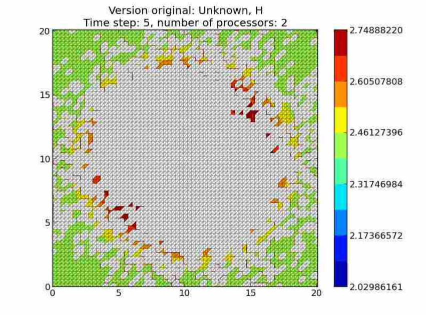
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 5



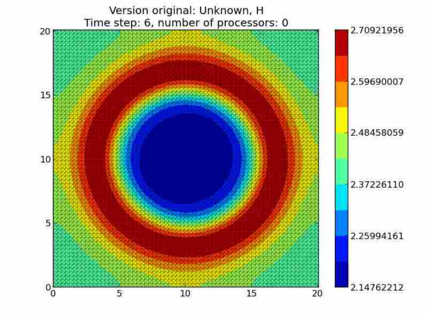
Sequential



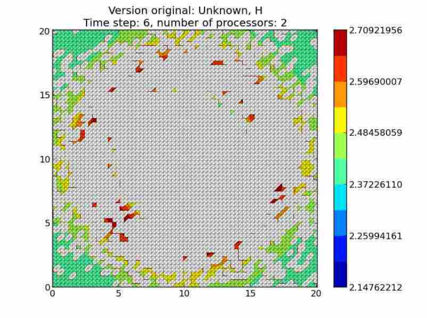
Parallel $p = 2$

Numerical reproducibility?

time step = 6



Sequential

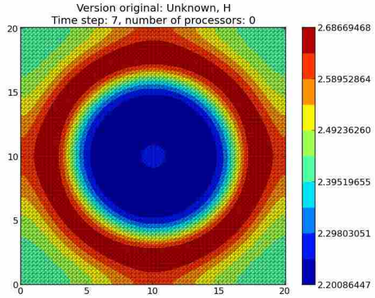


Parallel $p = 2$

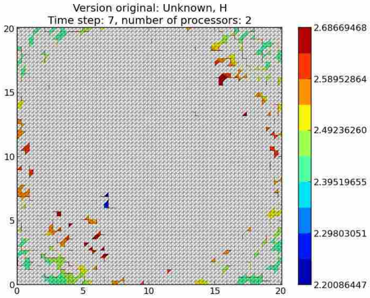
A white plot displays a non-reproducible value

Numerical reproducibility?

time step $= 7$



Sequential

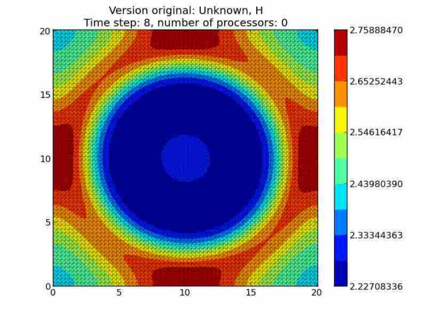


Parallel $p = 2$

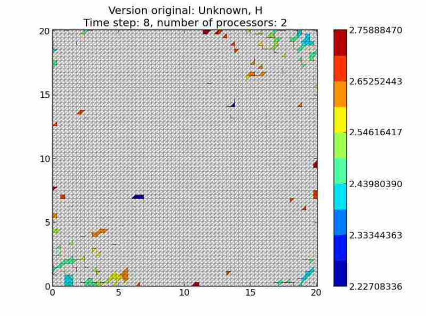
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 8



Sequential

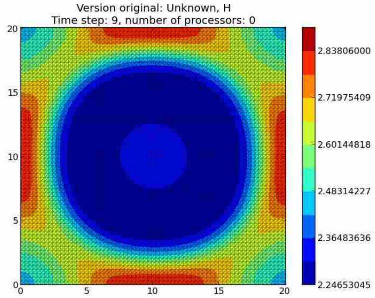


Parallel $p = 2$

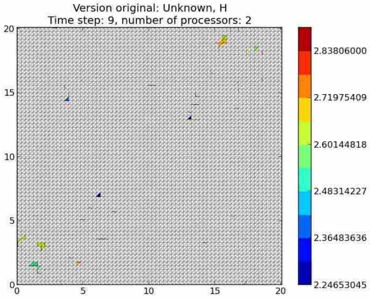
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 9



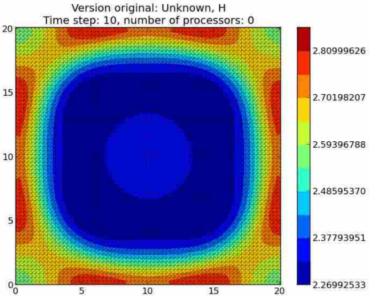
Sequential



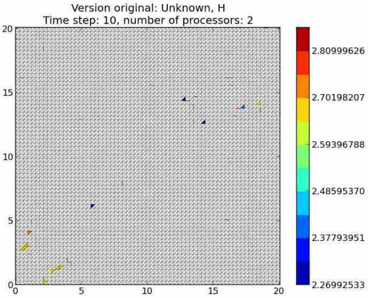
Parallel $p = 2$

Numerical reproducibility?

time step $= 10$



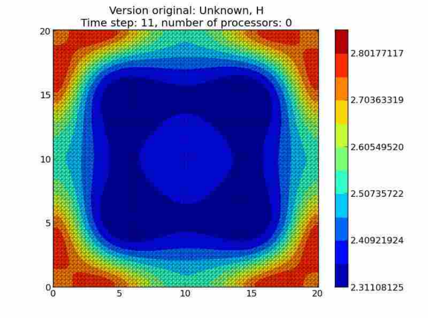
Sequential



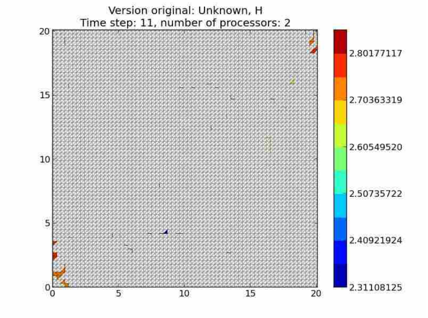
Parallel $p = 2$

Numerical reproducibility?

time step = 11



Sequential

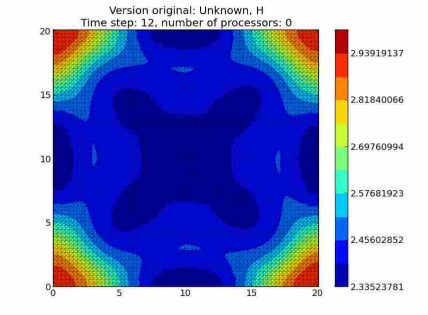


Parallel $p = 2$

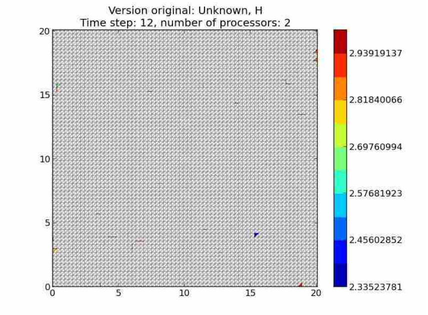
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 12



Sequential

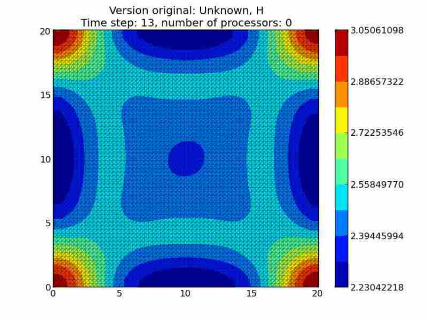


Parallel $p = 2$

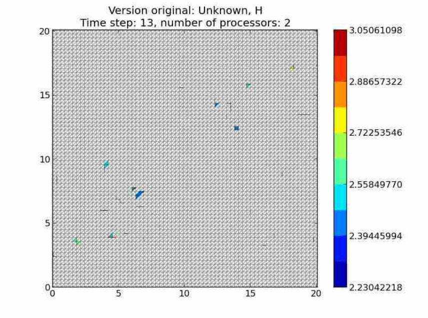
A white plot displays a non-reproducible value

Numerical reproducibility?

time step $= 13$



Sequential

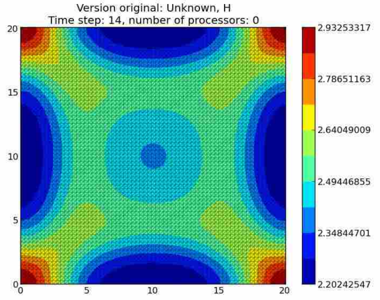


Parallel $p = 2$

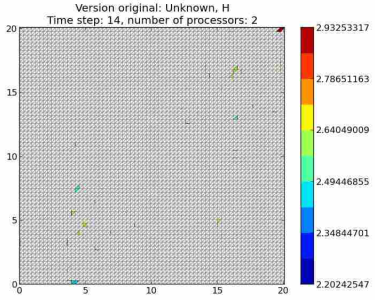
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 14



Sequential

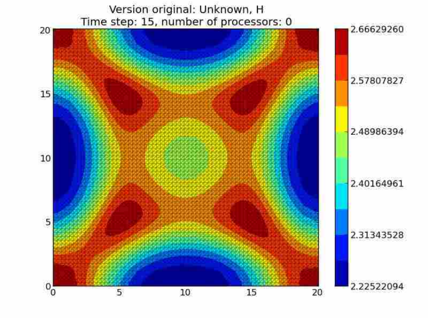


Parallel $p = 2$

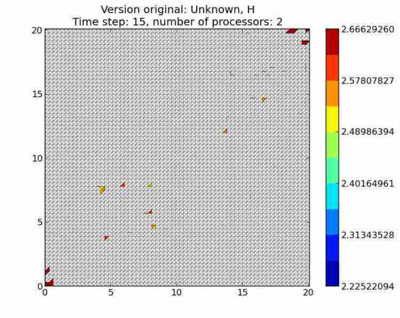
A white plot displays a non-reproducible value

NO numerical reproducibility!

time step $= 15$

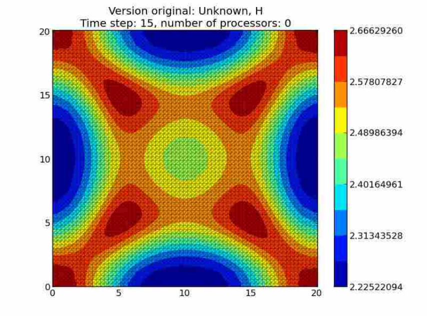


Sequential

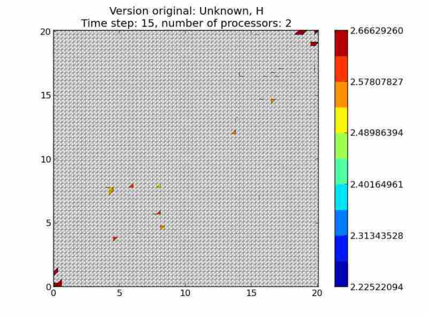


Parallel $p = 2$

NO numerical reproducibility!



Sequential



Parallel $p = 2$

Motivations

Reproducibility failures reported in numerical simulation in many domains

- energy [10],
- dynamic weather science [2],
- dynamic molecular [9],
- dynamic fluid [8]

Motivations

- How to debug?
- How to test?
- How to validate?
- How to receive legal agreements?

Today's issues

Case study

- Industrial scale software: openTelemac-Mascaret
- Finite element simulation, domain decomposition, linear system solving
- 2 modules: Tomawac, Telemac2D

Today's issues

Case study

- Industrial scale software: openTelemac-Mascaret
- Finite element simulation, domain decomposition, linear system solving
- 2 modules: Tomawac, Telemac2D

Feasibility

- How to recover reproducibility?
- Sources of non-reproducibility?
- Do existing techniques apply? how easily?
- Compensation yields reproducibility here!

Today's issues

Case study

- Industrial scale software: openTelemac-Mascaret
- Finite element simulation, domain decomposition, linear system solving
- 2 modules: Tomawac, Telemac2D

Feasibility

- How to recover reproducibility?
- Sources of non-reproducibility?
- Do existing techniques apply? how easily?
- Compensation yields reproducibility here!

Efficiency

- How much to pay for reproducibility?
- $\times 1.2 \leftrightarrow \times 2.3$ extra-cost which decreases as the problem size increases
- OK to debug, to validate and even to simulate!

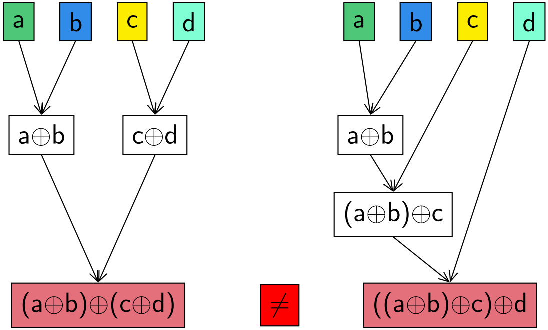
Basic ingredients

Parallel reduction and compensation techniques

Weakness of floating point arithmetic

- Rouding errors, non associative floating-point addition
- The computed value depends on the operation order

Parallel reduction: undefined order generates reproducibility failure

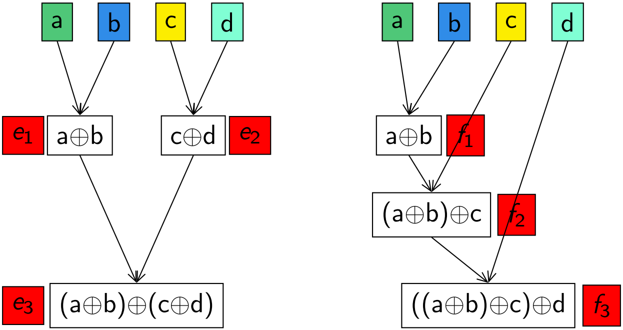


Parallel reduction and compensation techniques

Weakness of floating point arithmetic

- Rouding errors, non associative floating-point addition
- The computed value depends on the operation order

Compensation principle



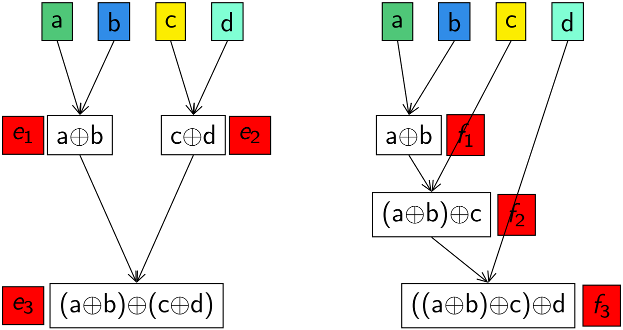
$$((a \oplus b) \oplus (c \oplus d)) \oplus ((e_1 \oplus e_2) \oplus e_3) = (((a \oplus b) \oplus c) \oplus d) \oplus ((f_1 \oplus f_2) \oplus f_3)$$

Parallel reduction and compensation techniques

Weakness of floating point arithmetic

- Rouding errors, non associative floating-point addition
- The computed value depends on the operation order

Compensation principle



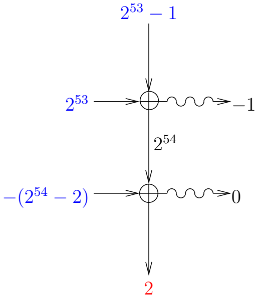
$$((a\oplus b)\oplus(c\oplus d)) \oplus ((e_1 \oplus e_2) \oplus e_3) = (((a\oplus b)\oplus c)\oplus d) \oplus((f_1 \oplus f_2) \oplus f_3)$$

Should be repeted for too ill-conditionned sums

Compensated summation: one example

IEEE binary64 (double): $x_1 = 2^{53} - 1$, $x_2 = 2^{53}$ and $x_3 = -(2^{54} - 2)$.
Exact sum: $x_1 + x_2 + x_3 = 1$.

Classic summation

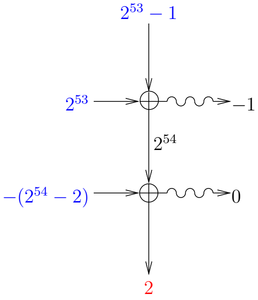


Relative error = 1

Compensated summation: one example

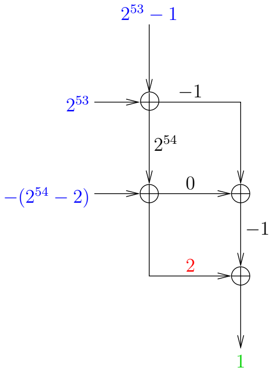
IEEE binary64 (double): $x_1 = 2^{53} - 1$, $x_2 = 2^{53}$ and $x_3 = -(2^{54} - 2)$.
Exact sum: $x_1 + x_2 + x_3 = 1$.

Classic summation



Relative error = 1

Compensation of the rounding errors

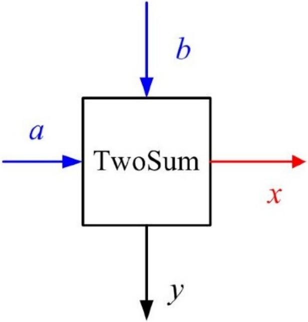


The exact result is computed

Rounding errors are computed with EFT

2Sum (Knuth, 65), Fast2Sum (Dekker, 71) for base ≤ 2 and RTN.

$$a + b = x + y, \text{ with } a, b, x, y \in \mathbb{F} \text{ and } x = a \oplus b.$$



Algorithm (Knuth)

function [x,y] = 2Sum(a,b)

$x = a \oplus b$

$z = x \ominus a$

$y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$

Algorithm ($|a| > |b|$, Dekker)

function [x,y] = Fast2Sum(a,b)

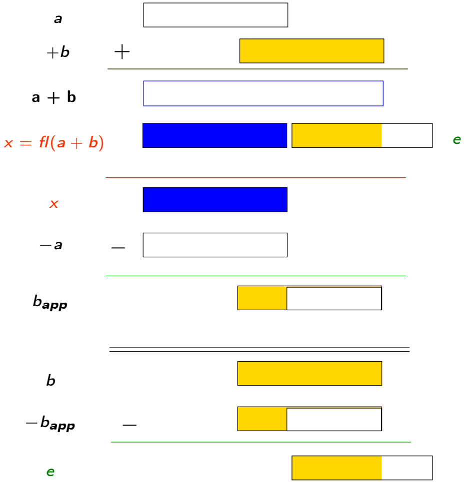
$x = a \oplus b$

$z = x \ominus a$

$y = b \ominus z$

The underlined idea of EFT

Fast2Sum computes the rouding error e in $a \oplus b$ ($|a| > |b|$)



Algorithm ($|a| > |b|$, Dekker)

function [x,y] = Fast2Sum(a,b)

$x = a \oplus b$

$z = x \ominus a$

$y = b \ominus z$

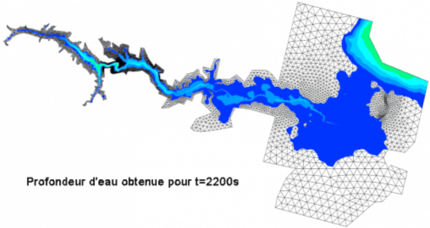
Other EFT for the product, FMA, ...

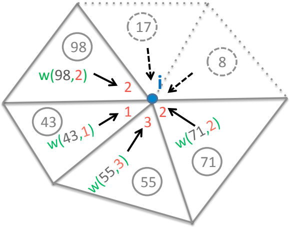
Reproducibility failure in a FE simulation

Finite element assembly: the sequential case

Assembly step principle: $V(i) = \sum_{elements} W_e(i)$

- compute the inner node values $V(i)$
- accumulating local W_e for every *ielem* that contains *i*





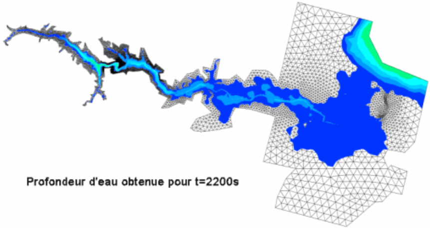
The assembly loop:

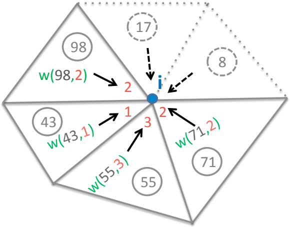
```
for  idp = 1, ndp          //idp: triangular local number(ndp=3)
  for ielem = 1, nelem
     i = IKLE(ielem,  idp)
     V(i) = V(i) +  W(ielem, idp)   //i: domain global number
```

Finite element assembly: the sequential case

Assembly step principle: $V(i) = \sum_{elements} W_e(i)$

- compute the inner node values $V(i)$
- accumulating local W_e for every *ielem* that contains i



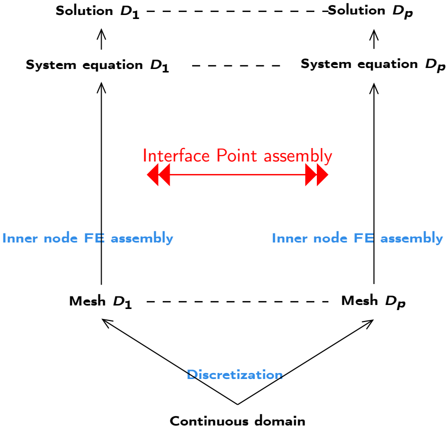


The assembly loop: managing local vs. global numbers

```
for  idp = 1, ndp          //idp: triangular local number(ndp=3)
  for ielem = 1, nelem
    i = IKLE(ielem,  idp)       <-- LOOP INDEX INDIRECTION
    V(i) = V(i) +  W(ielem, idp)   //i: domain global number
```

Finite element assembly: the parallel case

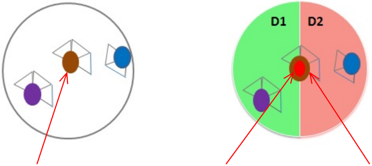
Parallel FE: subdomain decomposition



$\boxed{\text{IP assembly = communications and reductions}}$

Exact arithmetic

sequential | parallel



$V(i) = a$ $\quad V_{D1}(i) = b \quad V_{D2}(i) = c$

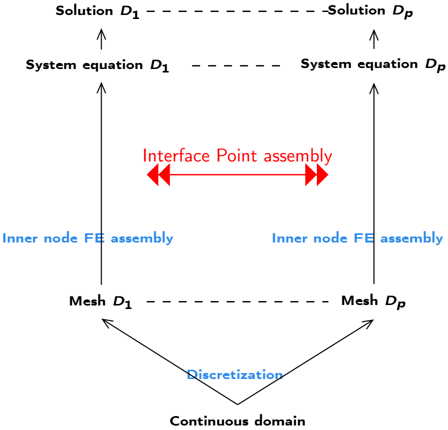
Interface point assembly

$V(i) = b + c = a$

$V(i) = \sum_{D_k} V_{D_k}(i)$
subdomains D_k, $k = 1...p$

Finite element assembly: the parallel case

Parallel FE: subdomain decomposition

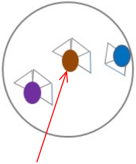


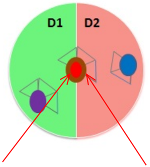
IP assembly =
communications and reductions

Floating point arithmetic

sequential

parallel





$V(i) = \widehat{a}$

$V_{D1}(i) = \widehat{b}$ $\quad V_{D2}(i) = \widehat{c}$

Interface point assembly

$V(i) = \widehat{b} \oplus \widehat{c} \neq \widehat{a}$

$V(i) = \sum_{D_k} V_{D_k}(i)$

subdomains $D_k, k = 1...p$

The parallel finite element assembly is non-reproducible

Tomawac: the Telemac's module for wave propagation in coastal areas

- Transport equation: 1st order PDE $\rightarrow$ ODEs along its characteristic curves
- Integration reduces to accumulation (and interpolation) over the finite element mesh
- Unknowns: significant wave height, mean wave frequency and direction
- The Nice test case:
  effect of high-speed ferry waves

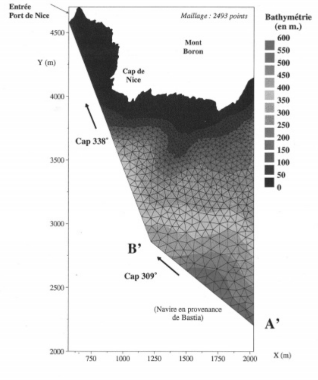


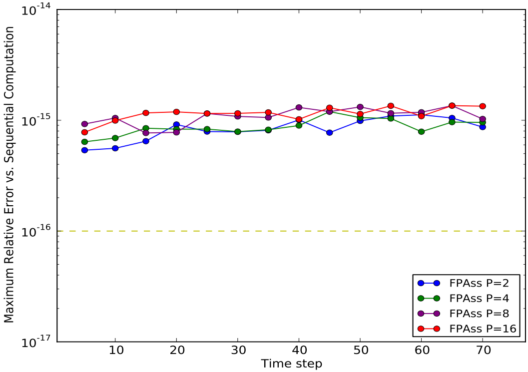
Figure 1 - Bathymétrie et maillage du domaine étudié

Reproducibility failure of the FE assembly step

Sequential vs. p-parallel results differ for $p = 2, 4, 8, 16$

- Assembly with the classical floating-point accumulation
- sequential *FPAss* vs. p-parallel $FPAss_p$

$$\max |FPAss_p - FPAss| / |FPAss|$$



Mean frequency wave, Nice test case, Tomawac

Basic ingredients

1. Failure of numerical reproducibility: what, when, why
   - Motivation
   - Today's case study
   - Compensation in floating-point arithmetic
2. Reproducibility failure in a finite element simulation
   - Sequential and parallel FE assembly
   - Non reproducible Tomawac
   - Sources of non reproducibility in Telemac2D
3. Recovering reproducibility
   - Reproducible parallel FE assembly
   - Reproducible algebraic operations
   - Reproducible conjugate gradient
   - Reproducible Telemac2D
4. Efficiency
5. Conclusion and work in progress

Sources of non reproducibility in Telemac2D

Culprits: theory

1. Building step: interface point assembly
2. Solving step (with conjugate gradient): parallel matrix-vector and dot products

Sources of non reproducibility in Telemac2D

Culprits: theory

1. Building step: interface point assembly
2. Solving step (with conjugate gradient): parallel matrix-vector and dot products

Culprits: practice = optimizations

- Element-by-element storage of the FE matrix
  - Everything is vector, no matrix!
  - No BLAS parallel matrix-vector product
- Wave equation, "mass-lumping" and associated algebraic transformations
  - System decoupling and many diagonal matrices
  - Everything is vector, no matrix!
- Interface point assembly and linear system solving are merged

Sources of non reproducibility in Telemac2D

The Telemac2D FE steps

Solution U,V

↑ wave equation

Solution H

↑ conjugate gradient

System equation $Ay = b$

$$\begin{pmatrix} AM1 & 0 & 0 \\ 0 & AM2 & 0 \\ 0 & 0 & AM3 \end{pmatrix} \begin{pmatrix} H \\ U \\ V \end{pmatrix} = \begin{pmatrix} CV1 \\ CV2 \\ CV3 \end{pmatrix}$$

↑ algebraic transformation

$$\begin{pmatrix} A_{hh} & A_{hu} & A_{hv} \\ A_{uh} & A_{uu} & 0 \\ A_{vh} & 0 & A_{vv} \end{pmatrix} \begin{pmatrix} H \\ U \\ V \end{pmatrix} = \begin{pmatrix} B_h \\ B_u \\ B_v \end{pmatrix}$$

↑ FE assembly + algebraic computation

Mesh (elements, nodes)

↑ Discretization

---

wave equation →

$CV2 = B_u - A_{uh}H,$

$CV3 = B_v - A_{vh}H.$

Interface node assembly:

$CV2, CV3$

conjugate gradient →

Interface node assembly:

$AM1$ in each iteration

algebraic transformation →

$AM2, AM3$: diagonal matrices,

$AM1 = A_{hh} - A_{hu}AM2^{-1}A_{uh} - A_{hv}AM3^{-1}A_{vh},$

$CV1 = B_h - A_{hu}AM2^{-1}B_u - A_{hv}AM3^{-1}B_v,$

Interface node assembly:

$AM2, AM3, CV1$

H

AM2,AM3

AM1,CV1

Sources of non reproducibility in Telemac2D

The Telemac2D FE steps

Solution U,V

wave equation

$$CV2 = B_u - A_{uh}H,$$
$$CV3 = B_v - A_{vh}H.$$

Interface node assembly:

$CV2$, $CV3$

Solution H

conjugate gradient

Interface node assembly:

$AM1$ in each iteration

H

AM2,AM3

AM1,CV1

System equation $Ay = b$

$$\begin{pmatrix} AM1 & 0 & 0 \\ 0 & AM2 & 0 \\ 0 & 0 & AM3 \end{pmatrix} \begin{pmatrix} H \\ U \\ V \end{pmatrix} = \begin{pmatrix} CV1 \\ CV2 \\ CV3 \end{pmatrix}$$

algebraic transformation

$AM2, AM3$: diagonal matrices,

$$AM1 = A_{hh} - A_{hu}AM2^{-1}A_{uh} - A_{hv}AM3^{-1}A_{vh},$$
$$CV1 = B_h - A_{hu}AM2^{-1}B_u - A_{hv}AM3^{-1}B_v,$$

Interface node assembly:

$AM2$, $AM3$, $CV1$

$$\begin{pmatrix} A_{hh} & A_{hu} & A_{hv} \\ A_{uh} & A_{uu} & 0 \\ A_{vh} & 0 & A_{vv} \end{pmatrix} \begin{pmatrix} H \\ U \\ V \end{pmatrix} = \begin{pmatrix} B_h \\ B_u \\ B_v \end{pmatrix}$$

FE assembly +

algebraic computation

Mesh (elements, nodes)

Discretization

Objective

Correct sources of non-reproducibility
to compute reproducible system and solutions

Recovering reproducibility in Telemac2D

Sources

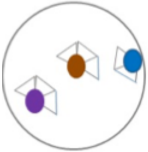
- FE assembly: matrix and second member
- Wave equation: algebraic transformations and diagonal resolutions
- Resolution: matrix-vector and dot products

Reproducible resolution: principles

- vector $V \rightarrow [V, E_V] \rightarrow V + E_V$
- Inner nodes: compensate FE assembly
- Interface nodes: propagate rounding errors
  and compensate within interface node assembly (also denoted IP)
- Compensate the EBE matrix-vector products that include IP assembly
- Compensate the MPI parallel dot products that include MPI reduction

Accurate compensated FE assembly: the sequential case

```
for idp= 1, ndp
  for ielem= 1, nelem
    i=IKLE(ielem, idp)
    X(i)=X(i)+W(ielem,idp)
```



D1

FE assembly

Contribution $W_i(e_1)$ for every node mesh i belonging to element $e_1 \ldots e_n$

Computation and accumulation of rounding errors Δ

$$X(i) = W_i(e_1) + W_i(e_2) + \ldots + W_i(e_n)$$

Δ_1 Δ_2 Δ_n

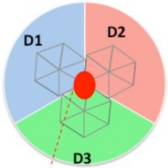
$$E(i) = \Delta_1 + \Delta_2 + \ldots + \Delta_n$$

Compensation of the FE assembly for every mesh node I

$$X(i)=X(i)+E(i)$$

The parallel case is easy to derive

```
for idp=1, ndp
  for ielem=1, nelem
    i=IKLE (ielem, idp)
    X(i)=X(i)+ W(ielem, idp)
```



i : Interface Point

Interface Point assembly

D1 | D2 | D3

Subdomain FE assembly.

Contribution $W_i(e_k)$ for every node mesh i belonging to elem $e_1, \ldots, e_n$

Computation and accumulation of rounding errors Δ

similar

$$X(i) = \underbrace{W_i(e_1)}_{\Delta_1} + \underbrace{W_i(e_2)}_{\Delta_2} + \ldots + \underbrace{W_i(e_n)}_{\Delta_n}$$

$$E(i) = \Delta_1 + \Delta_2 + \ldots + \Delta_n$$

similar

Interface point assembly

i : interface point between D1,D2,D3

Accumulation of errors Δ and $\mathcal{E}$

$X(i), E(i)$ ⟺ $X(i) = X(i) \underbrace{+}_{\mathcal{E}_1} X(i) \underbrace{+}_{\mathcal{E}_2} X(i)$ ⟺ $X(i), E(i)$

$$E(i) = ((E(i) + E(i) + \mathcal{E}_1) + E(i) + \mathcal{E}_2)$$

Compensation of the FE assembly for every mesh node i

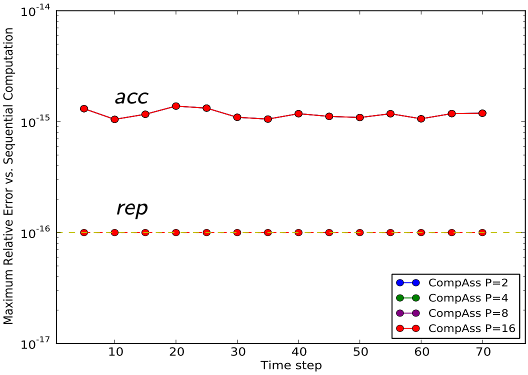
similar

$$X(i)=X(i)+E(i)$$

similar

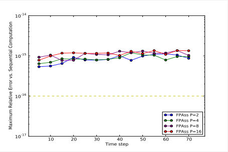
Accurate compensated assembly gives reproducibility

Reproducibility in Tomawac: the Nice test case



Mean frequency wave, Nice test case, Tomawac

rep in the original FE assembly



A^s: sequential, A^p: p-parallel
$\max_{rel}(A_1, A_2) = |A_1 - A_2|/|A_2|$

Reproducibility:
$rep =$
$\max_{rel}(CompAss^p, CompAss^s)$

Accuracy (vs. FPAss):
$acc = \max_{rel}(CompAss^p, FPAss^s)$

Reproducible algebraic operations

Algebraic operation: the vector case

Reproducible vector algebraic operations

- openTelemac's included library: BIEF
- Entry-wise vector ops: copy, opp/inv., add/sub, Hadamard prod., ...
- Applies also for diagonal matrix
- Every op. is applied to FE assembled vectors with/without IP assembly
- Principle: propagate rounding errors and compensate while assembling IP

Example:
Hadamard product

Original version

1. $V = X \circ Y$
2. IP assembly if necessary for interface node i

Reproducible version

1. compute errors:
   $[V, E_V] = [X, E_X] \circ [Y, E_Y]$
   with $(V, e_V) = 2\text{Prod}(X, Y)$
   and $E_V = X \circ E_Y + Y \circ E_X + e_V$
2. compensate at inner nodes k:
   $V(k) + E_V(k)$:
3. propagate and compensate at the IP assembly if necessary for IP i: $V(i) + E_V(i)$

What is reproducible now?

Most of the linear system:

- FE assembly
- algebraic vector operations
- interface point assembly

except:

- the matrix of the H system
- its dependencies: the second members of the U and V systems

Partially reproducible Telemac2D

wave equation :

$CV2 = B_u - A_{uh}H,$

$CV3 = B_v - A_{vh}H.$

Interface node assembly:

$CV2, CV3$

Solution U,V

H

AM2,AM3

conjugate gradient :

Interface node assembly:

$AM1d$ in each iteration

Solution H

AM1,CV1

algebraic transformation:

$AM2, AM3$: diagonal matrices,

$AM1 = A_{hh} - A_{hu}AM2^{-1}A_{uh} - A_{hv}AM3^{-1}A_{vh},$

$CV1 = B_h - A_{hu}AM2^{-1}B_u - A_{hv}AM3^{-1}B_v,$

Interface node assembly:

$AM2, AM3, CV1$

Towards a reproducible conjugate gradient

The last step

A=[$AM1$, E_{AM1}]

B=$CV1$

Initialization:

$$r^0 = AX^0 - B; \text{ a given } d^0$$

$$\rho^0 = \frac{(r^0, d^0)}{(Ad^0, d^0)}; \; X^1 = X^0 - \rho^0 d^0$$

Iterations:

$$r^m = r^{m-1} - \rho^{m-1} Ad^{m-1}$$

$$d^m = r^m + \frac{(r^m, r^m)}{(r^{m-1}, r^{m-1})} d^{m-1}$$

$$\rho^m = \frac{(r^m, d^m)}{(d^m, Ad^m)}$$

$$X^{m+1} = X^m - \rho^m d^m$$

X=H

Non-reproducibility: sources and solutions [5]

Towards a reproducible conjugate gradient

The last step

$A = [AM1,\ E_{AM1}]$

$B = CV1$

Initialization:

$$r^0 = AX^0 - B;\ \text{a given } d^0$$

$$\rho^0 = \frac{(r^0, d^0)}{(Ad^0, d^0)};\ X^1 = X^0 - \rho^0 d^0$$

Iterations:

$$r^m = r^{m-1} - \rho^{m-1} Ad^{m-1}$$

$$d^m = r^m + \frac{(r^m, r^m)}{(r^{m-1}, r^{m-1})} d^{m-1}$$

$$\rho^m = \frac{(r^m, d^m)}{(d^m, Ad^m)}$$

$$X^{m+1} = X^m - \rho^m d^m$$

$X = H$

Non-reproducibility: sources and solutions [5]

- EBE matrix-vector product
  - is followed by one IP assembly at every iteration
  - reproducible with compensated IP assembly

Towards a reproducible conjugate gradient

The last step

A=[$AM1$, E_{AM1}] → ; B=$CV1$ →

Initialization:

$$r^0 = AX^0 - B; \text{ a given } d^0$$

$$\rho^0 = \frac{(r^0, d^0)}{(Ad^0, d^0)}; \quad X^1 = X^0 - \rho^0 d^0$$

Iterations:

$$r^m = r^{m-1} - \rho^{m-1} Ad^{m-1}$$

$$d^m = r^m + \frac{(r^m, r^m)}{(r^{m-1}, r^{m-1})} d^{m-1}$$

$$\rho^m = \frac{(r^m, d^m)}{(d^m, Ad^m)}$$

$$X^{m+1} = X^m - \rho^m d^m$$

→ X=H

Non-reproducibility: sources and solutions [5]

- EBE matrix-vector product
  - is followed by one IP assembly at every iteration
  - reproducible with compensated IP assembly
- MPI `all_reduced` dot product
  - a parallel version of the compensated `dot2` [6]

Towards a reproducible conjugate gradient

The last step

A=[$AM1$, E_{AM1}]

B=$CV1$

```
Initialization:
```

$$r^0 = AX^0 - B; \text{ a given } d^0$$

$$\rho^0 = \frac{(r^0, d^0)}{(Ad^0, d^0)}; \; X^1 = X^0 - \rho^0 d^0$$

```
Iterations:
```

$$r^m = r^{m-1} - \rho^{m-1} Ad^{m-1}$$

$$d^m = r^m + \frac{(r^m, r^m)}{(r^{m-1}, r^{m-1})} d^{m-1}$$

$$\rho^m = \frac{(r^m, d^m)}{(d^m, Ad^m)}$$

$$X^{m+1} = X^m - \rho^m d^m$$

X=H

Non-reproducibility: sources and solutions [5]

- EBE matrix-vector product
  - is followed by one IP assembly at every iteration
  - reproducible with compensated IP assembly
- MPI `all_reduced` dot product
  - a parallel version of the compensated `dot2` [6]
- with ponderation for interface node shared by p subdomains
  - $(1/p, 1/p, \ldots, 1/p) \to (0, \ldots, 0, 1, 0 \ldots, 0)$

A reproducible conjugate gradient

$$\mathbf{A} = [AM1,\ E_{AM1}] \quad \mathbf{B} = CV1$$

Initialization:

$$r^0 = AX^0 - B;\ \text{a given } d^0$$

$$\rho^0 = \frac{(r^0, d^0)}{(Ad^0, d^0)};\ X^1 = X^0 - \rho^0 d^0$$

Iterations:

$$r^m = r^{m-1} - \rho^{m-1} Ad^{m-1}$$

$$d^m = r^m + \frac{(r^m, r^m)}{(r^{m-1}, r^{m-1})} d^{m-1}$$

$$\rho^m = \frac{(r^m, d^m)}{(d^m, Ad^m)}$$

$$X^{m+1} = X^m - \rho^m d^m$$

$$\longrightarrow \mathbf{X} = H$$

Not necessarily more accurate but reproducible

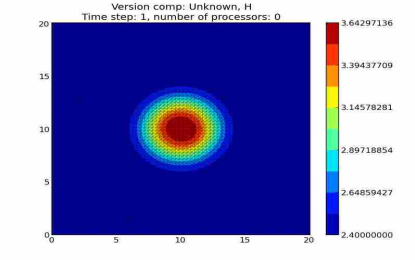
Same errors in the compensated values for both sequential and parallel executions

Recovering reproducibility in a finite element resolution

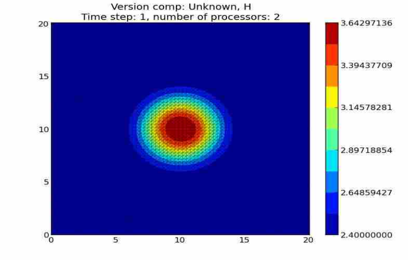
Reproducible gouttedo!

Time step 1

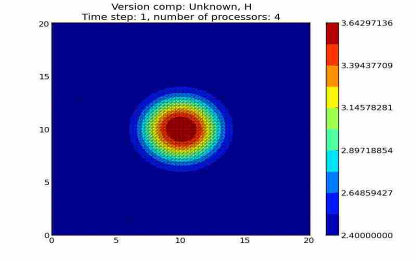
p=1



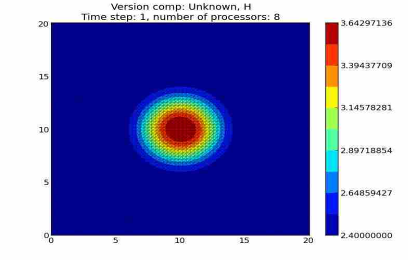
p=2



p=4



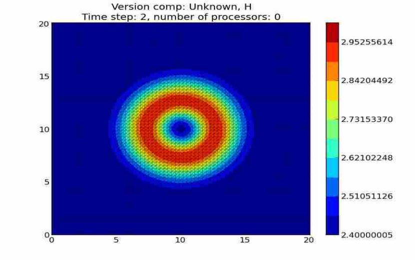
p=8



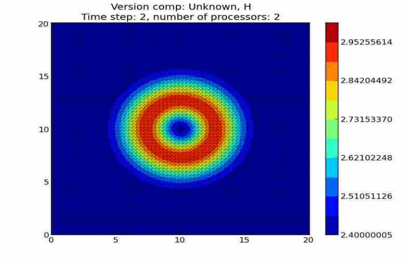
Reproducible gouttedo!

Time step 2

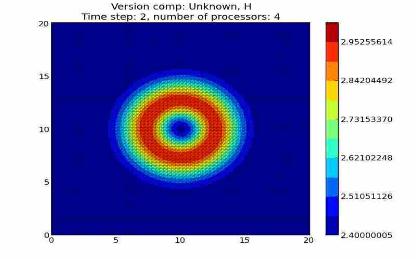
p=1



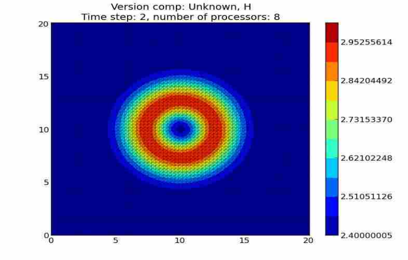
p=2



p=4



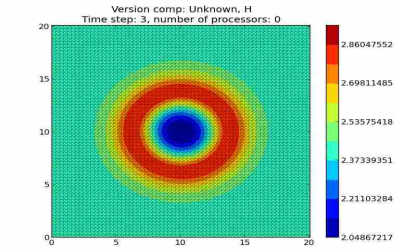
p=8



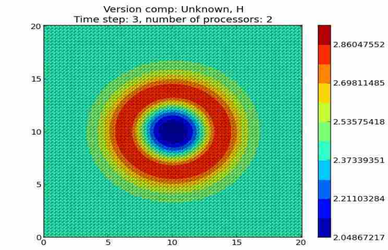
Reproducible gouttedo!

Time step 3

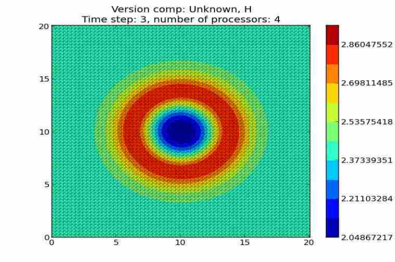
p=1



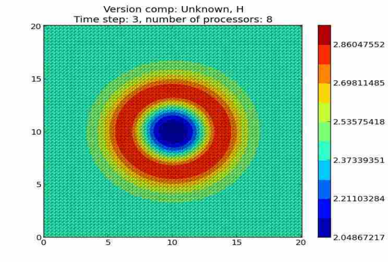
p=2



p=4

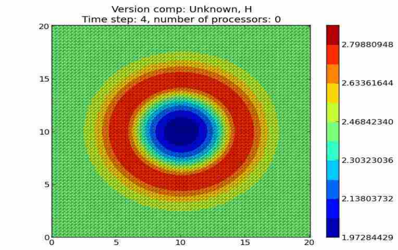


p=8

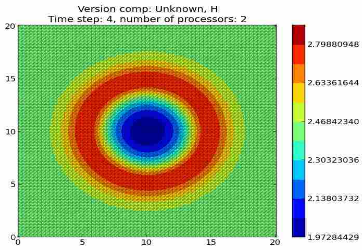


Time step 4

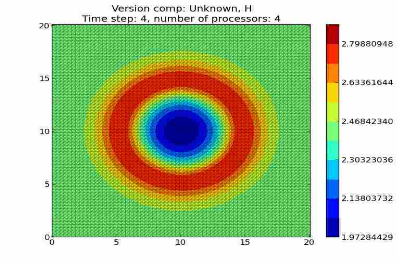
p=1



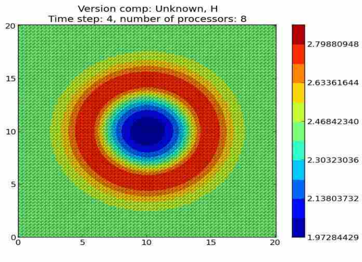
p=2



p=4



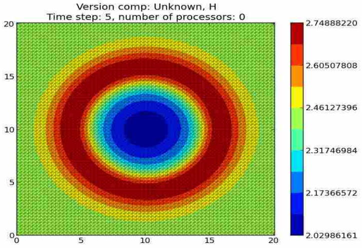
p=8



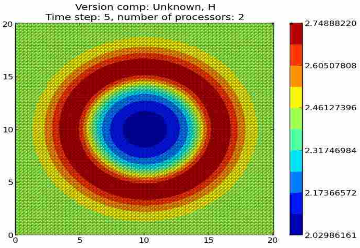
Reproducible gouttedo!

Time step 5

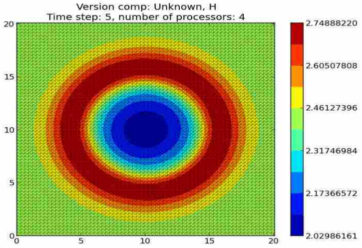
p=1



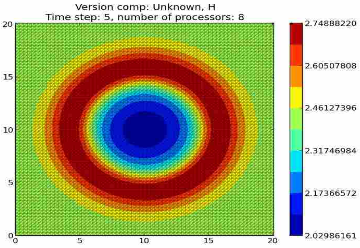
p=2



p=4

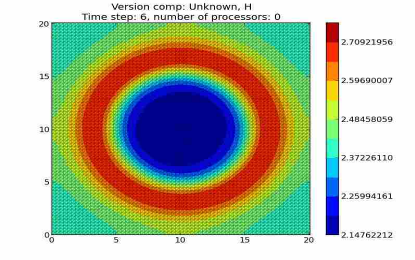


p=8

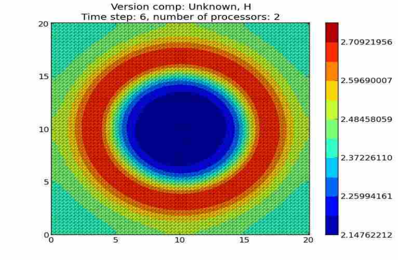


Time step 6

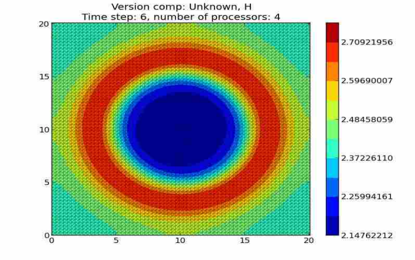
p=1



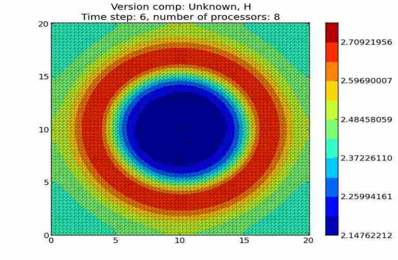
p=2



p=4

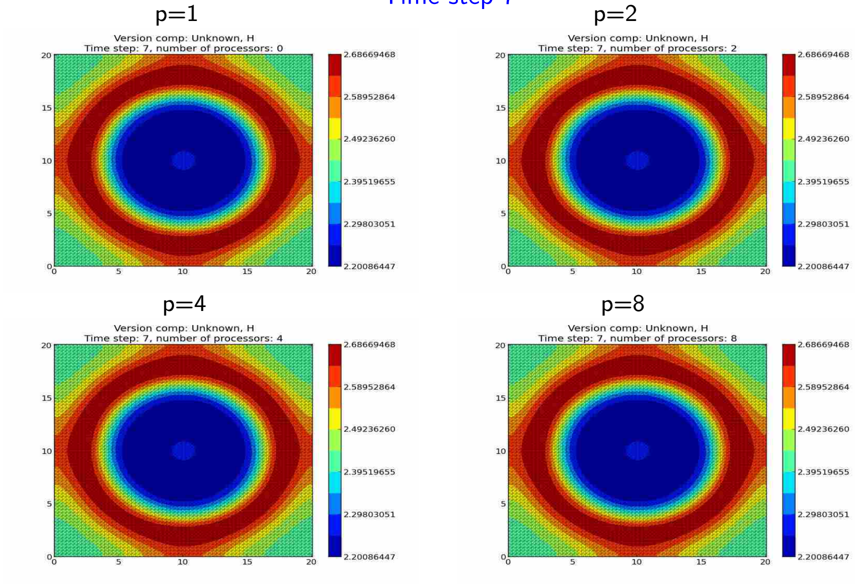


p=8



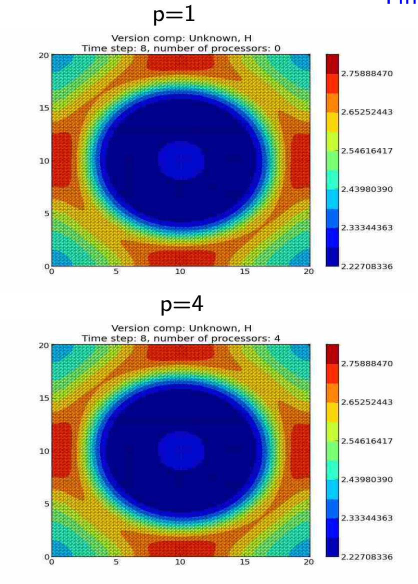
Reproducible gouttedo!

Time step 7



Reproducible gouttedo!

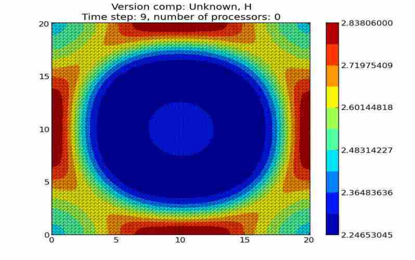
Time step 8



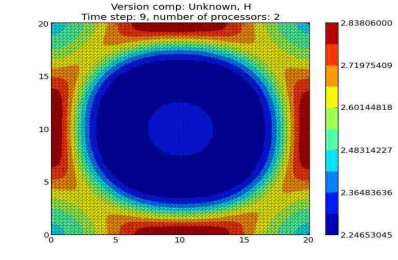
Reproducible gouttedo!

Time step 9

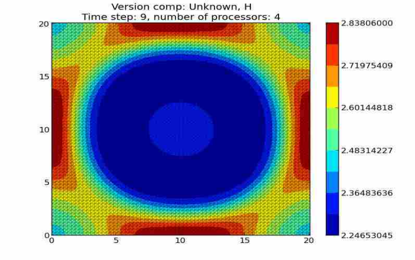
p=1



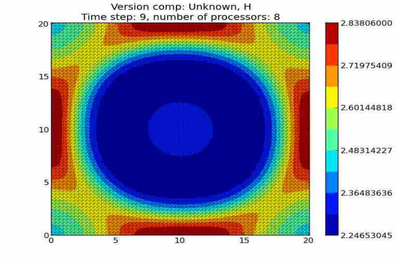
p=2



p=4



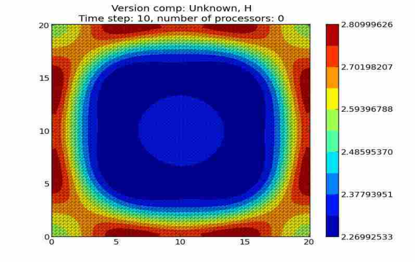
p=8



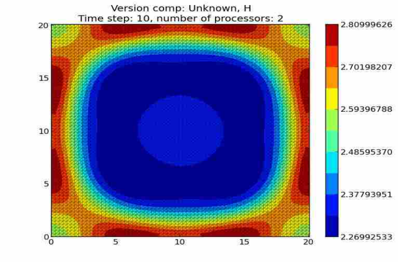
Reproducible gouttedo!

Time step 10

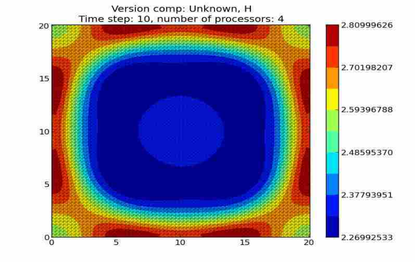
p=1



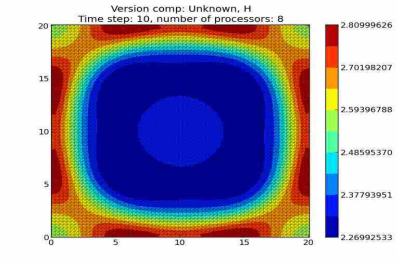
p=2



p=4



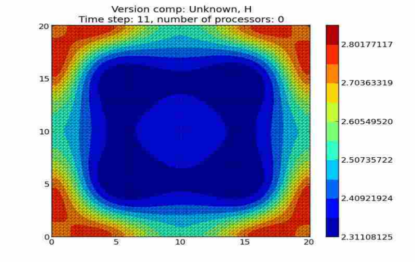
p=8



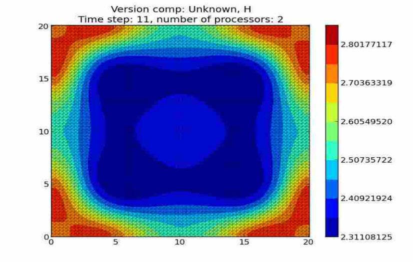
Reproducible gouttedo!

Time step 11

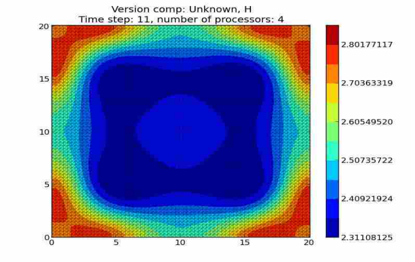
p=1



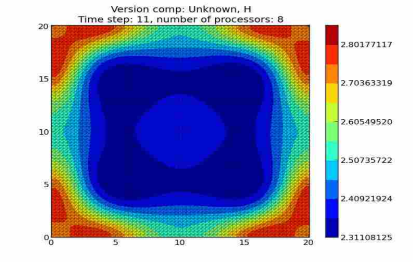
p=2



p=4



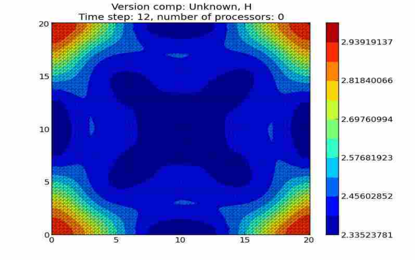
p=8



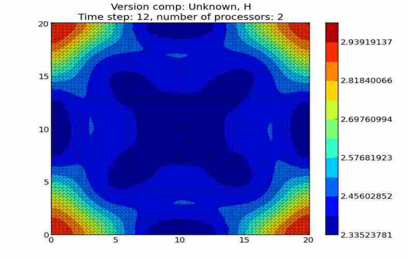
Reproducible gouttedo!

Time step 12

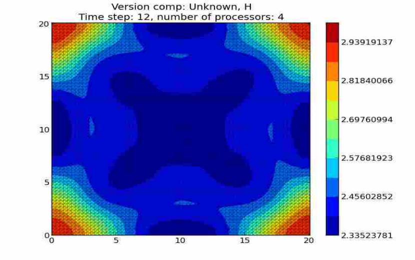
p=1



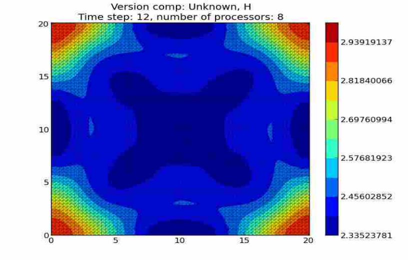
p=2



p=4



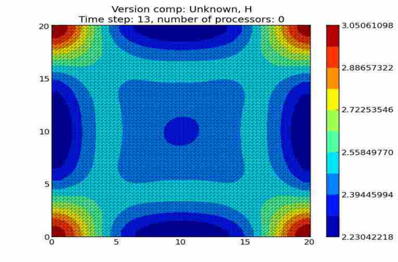
p=8



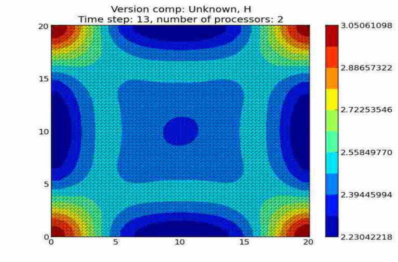
Reproducible gouttedo!

Time step 13

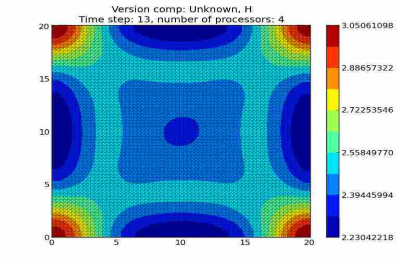
p=1



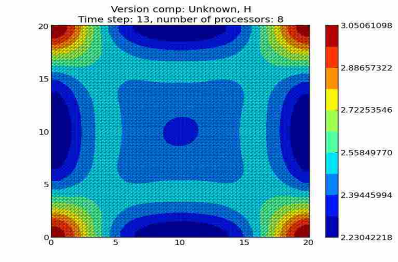
p=2



p=4



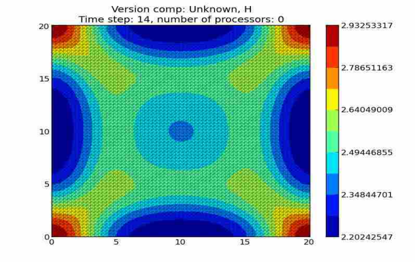
p=8



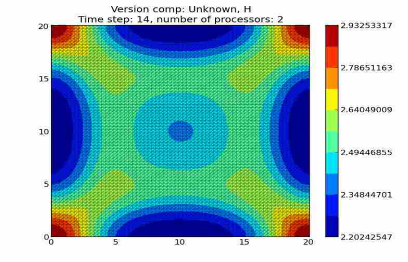
Reproducible gouttedo!

Time step 14

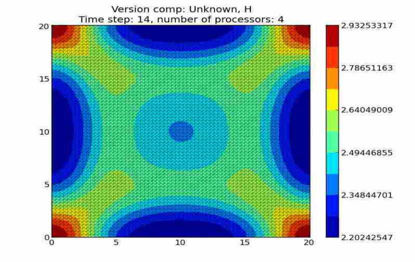
p=1



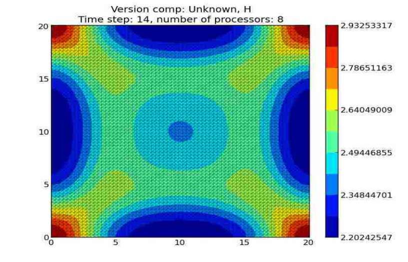
p=2



p=4



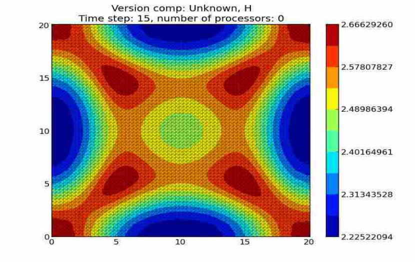
p=8



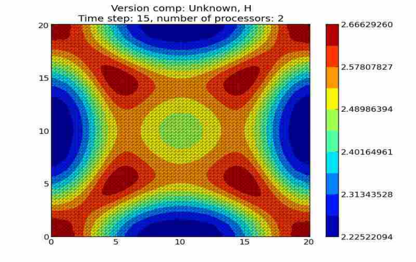
Reproducible gouttedo!

Time step 15

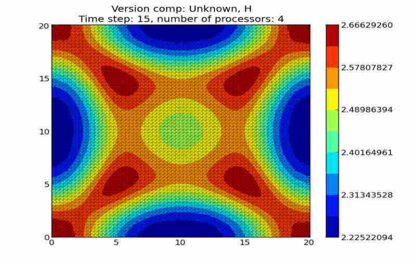
p=1



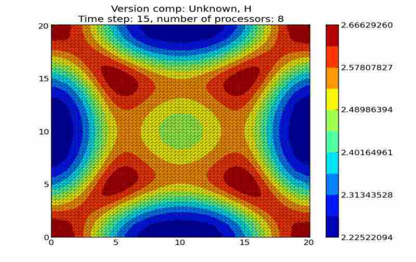
p=2



p=4



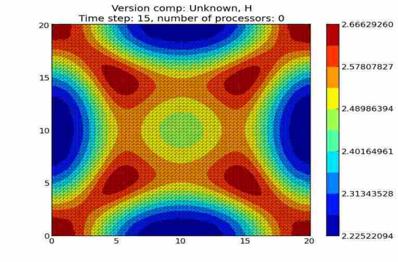
p=8



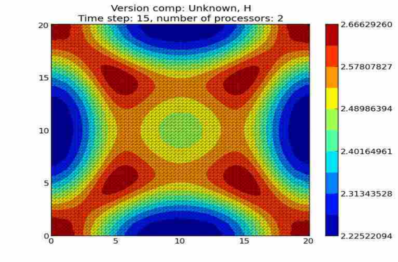
Reproducible gouttedo!

Time step 15

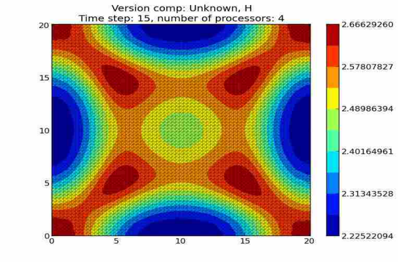
p=1



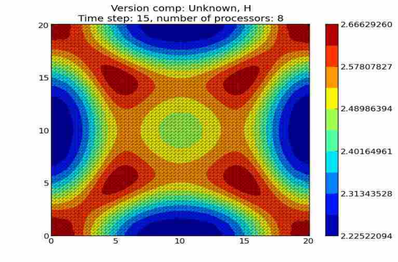
p=2



p=4



p=8



Runtime extra-cost for reproducible simulations

Measures, test cases and mesh sizes

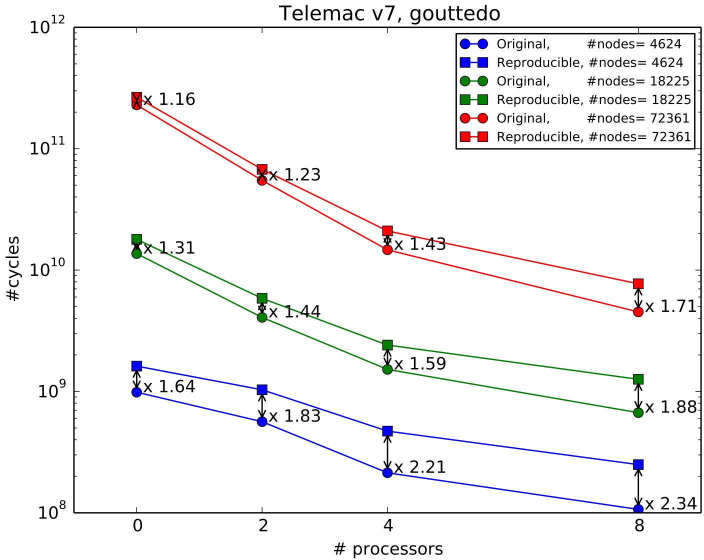
- hardware cycle counter: `rdtsc`
- `gouttedo`
- meshe sizes: 4624, 18225, 72361 nodes ($\approx 1, \times 4, \times 16$)

Hardware and software env.

- openTelemac v7.2
- socket: Intel Xeon E5-2660 2.20GHz (L3 cache = 20 M)
- 2 sockets of 8 cores each
- GNU Fortran 4.6.3, -O3
- OpenRTE (openMPI) 1.5.4
- Linux 3.5.0-54-generic

The core runtime extra-cost for reproducible gouttedo

gouttedo core: no input/output steps, just building+solving



Time to conclude

Conclusion

Recovering numerical reproducibility

- Industrial scale software: openTelemac-Mascaret
- Finite element simulation, domain decomposition, linear system solving
- 2 reproducible modules: Tomawac, Telemac2D
- Integration in the next openTelemac version: in progress

Conclusion

Recovering numerical reproducibility

- Industrial scale software: openTelemac-Mascaret
- Finite element simulation, domain decomposition, linear system solving
- 2 reproducible modules: Tomawac, Telemac2D
- Integration in the next openTelemac version: in progress

Feasibility

- How to recover reproducibility? Sources of non-reproducibility? Do existing techniques apply? how easily?
- Hand-made analysis of the computing workflow
- Compensation yields reproducibility here!
- Fits well to the openTelemac's vector library
- Other existing techniques also apply and more or less easily [4]

Conclusion

Efficiency

- How much to pay for reproducibility?
- $\times 1.2 \leftrightarrow \times 2.3$: OK to debug, to validate and even to simulate!

Conclusion

Efficiency

- How much to pay for reproducibility?
- $\times 1.2 \leftrightarrow \times 2.3$: OK to debug, to validate and even to simulate!

Reproducibility at a larger scale: the whole openTelemac software suite

- Does it still working for complex, large and real-life simulations?
- The two FE test cases are significant enough to validate the methodology
- Localization of the failure sources is difficult to automatize
- but it's easier to pass the methodology on to software developpers

Acknowledgment

Christophe Denis, EDF R&D, Clamart and CMLA, Cachan
Jean-Michel Hervouet, LNE, EDR R&D, Chatou
Chemseddine Chohra, DALI/LIRMM, UPVD, Perpignan

DALI, Digits, Architectures et Logiciels Informatiques

Références I

J. W. Demmel and H. D. Nguyen.
Fast reproducible floating-point summation.
In Proc. 21th IEEE Symposium on Computer Arithmetic. Austin, Texas, USA, 2013.

Y. He and C. Ding.
Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications.
J. Supercomput., 18:259–277, 2001.

J.-M. Hervouet.
Hydrodynamics of free surface flows: Modelling with the finite element method.
John Wiley & Sons, 2007.

P. Langlois, R. Nheili, and C. Denis.
Numerical Reproducibility: Feasibility Issues.
In NTMS'2015: 7th IFIP International Conference on New Technologies, Mobility and Security, pages 1–5, Paris, France, July 2015. IEEE, IEEE COMSOC & IFIP TC6.5 WG.

P. Langlois, R. Nheili, and C. Denis.
Recovering numerical reproducibility in hydrodynamic simulations.
In J. Hormigo, S. Oberman, and N. Revol, editors, 23rd IEEE International Symposium on Computer Arithmetic, pages 1–10. IEEE Computer Society, July 2016.
(Silicon Valley, USA. July 10-13 2016).

Références II

T. Ogita, S. M. Rump, and S. Oishi.
Accurate sum and dot product.
SIAM J. Sci. Comput., 26(6):1955–1988, 2005.

Open TELEMAC-MASCARET. v.7.0, Release notes.
www.opentelemac.org, 2014.

R. W. Robey, J. M. Robey, and R. Aulwes.
In search of numerical consistency in parallel programming.
Parallel Comput., 37(4-5):217–229, 2011.

M. Taufer, O. Padron, P. Saponaro, and S. Patel.
Improving numerical reproducibility and stability in large-scale numerical simulations on gpus.
In IPDPS, pages 1–9. IEEE, 2010.

O. Villa, D. G. Chavarría-Miranda, V. Gurumoorthi, A. Márquez, and S. Krishnamoorthy.
Effects of floating-point non-associativity on numerical computations on massively multithreaded systems.
In CUG 2009 Proceedings, pages 1–11, 2009.

Other existing techniques

Existing techniques to recover numerical reproducibility in summation

- Accurate compensated summation [6]
- Demmel-Nguyen's reproducible sums [1]
- Integer convertion [7]