PFEM performance meeting

Last updated on Monday 3rd May, 2021

Simon FÉVRIER

LIÈGE université
Sciences Appliquées

Table of contents

Available solvers

Monolithic PSPG (implicit)

Add a term proportional to the momentum equation to the continuity equation:

$$\begin{bmatrix} \frac{1}{\Delta t}\tilde{\mathbf{M}}_{\rho_0}^{n+1} + \mathbf{K}_{\mu}^{n+1} & -\mathbf{D}^{\mathsf{T}\,n+1} \\ \frac{1}{\Delta t}\mathbf{C}_{\tau}^{n+1} + \mathbf{D}^{n+1} & \mathbf{L}_{\tau/\rho}^{n+1} \end{bmatrix} \begin{pmatrix} \mathbf{v}^{n+1} \\ \mathbf{p}^{n+1} \end{pmatrix} = \begin{bmatrix} \mathbf{f}_{\text{ext}}{}^{n+1} + \frac{1}{\Delta t}\tilde{\mathbf{M}}_{\rho_0}^{n+1}\mathbf{v}^{n} \\ \mathbf{h}_{\tau}{}^{n+1} + \frac{1}{\Delta t}\mathbf{C}_{\tau}^{n+1}\mathbf{v}^{n} \end{bmatrix}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^{n} + \mathbf{v}^{n+1}\Delta t$$

- uses `Eigen::SparseLU` linear solver $\rightarrow$ no parallel computing,
- matrix not symmetric $\rightarrow$ cannot use iterative solver from `Eigen`,
- matrix of size (mesh dim $+ 1$) $\times$ N_{nodes}.

Fractional Step (implicit)

Split the computation in two step: compute an approximation of the velocity, then correct the pressure to obtain divergence free velocity:

$$\tilde{\mathbf{M}}_{\rho_0}^{n+1}\frac{\tilde{\mathbf{v}}^{n+1}-\mathbf{v}^n}{\Delta t}+\mathbf{K}_{\mu}^{n+1}\mathbf{v}^{n+1}-\gamma_{FS}\mathbf{D}^{\mathsf{T},n+1}\mathbf{p}^n=\mathbf{f}_{\text{ext}}^{n+1}$$

$$\frac{\Delta t}{\rho_0}\mathbf{L}\delta\mathbf{p}=-\mathbf{D}\tilde{\mathbf{v}}^{n+1}$$

$$\tilde{\mathbf{M}}_{\rho_0}^{n+1}\frac{\mathbf{v}^{n+1}-\tilde{\mathbf{v}}^{n+1}}{\Delta t}=\mathbf{D}^{\mathsf{T},n+1}\delta\mathbf{p}$$

$$\mathbf{x}^{n+1}=\mathbf{x}^n+\mathbf{v}^{n+1}\Delta t \quad \mathbf{p}^{n+1}=\gamma_{FS}\mathbf{p}^n+\delta\mathbf{p}$$

- all matrices are symmetric,
- iterative solver can thus be used(e.g. `Eigen::ConjugateGradient`,
- smaller matrices.

Weakly compressible (explicit)

Express the density as a function of the pressure and use an explicit compressible solver:

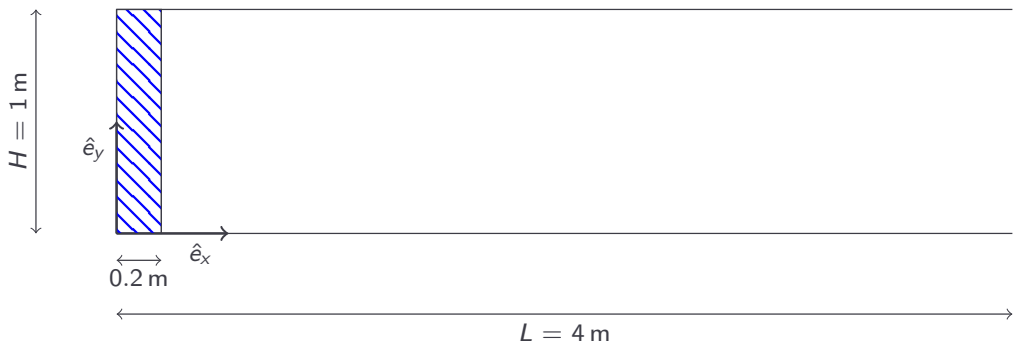
$$
\begin{aligned}
&\bar{\mathbf{v}}^{n+1} = \mathbf{v}^n + 0.5\Delta t \mathbf{a}_n \,, \\
&\bar{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}_n + 0.5\Delta t^2 \mathbf{a}_n = \mathbf{x}_n + \Delta t \bar{\mathbf{v}}^{n+1} \,, \\
&\mathbf{M}\boldsymbol{\rho}^{n+1} = \mathbf{M}\boldsymbol{\rho}^n - \Delta t \mathbf{D}_\rho \bar{\mathbf{v}}^{n+1} \,, \\
&\left[\mathbf{p}^{n+1}\right]_i = \rho^{-1}(\left[\boldsymbol{\rho}^{n+1}\right]_i) \,, \\
&\tilde{\mathbf{M}}_\rho \mathbf{a}^{n+1} = \mathbf{f}_{\text{ext}}^{n+1} - \mathbf{K}_\mu \bar{\mathbf{v}}^{n+1} + \mathbf{D}^T \mathbf{p}^{n+1} \\
&\mathbf{v}^{n+1} = \bar{\mathbf{v}}^{n+1} + 0.5\Delta t \mathbf{a}_{n+1} \\
&\mathbf{x}^{n+1} = \bar{\mathbf{x}}^{n+1}
\end{aligned}
$$

$$
\Delta t = C \min_e \left(\frac{h_e}{\max(\|\vec{v}\|, \sqrt{K/\rho})} \right)
$$

- mass matrices are lumped $\rightarrow$ trivial inversion,
- good parallel properties.

Performance with number of nodes

Solving time - Problem used



Initial geometry of the flow between two plates problem.

ρ	$1000\,\mathrm{kg\,m^{-3}}$
μ	$0.001\,\mathrm{Pa\,s}$

Incompressible fluid parameters

$\rho^\star$	$1000\,\mathrm{kg\,m^{-3}}$
μ	$0.001\,\mathrm{Pa\,s}$
K_0	$2\,200\,000\,\mathrm{Pa}$
K_0'	7.6

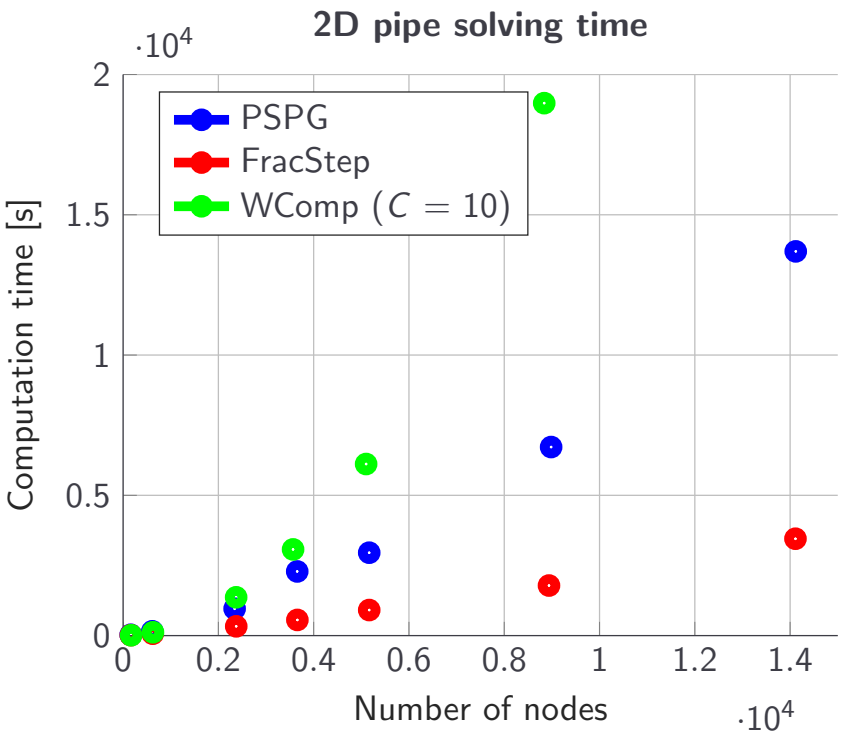
Weakly compressible fluid parameters

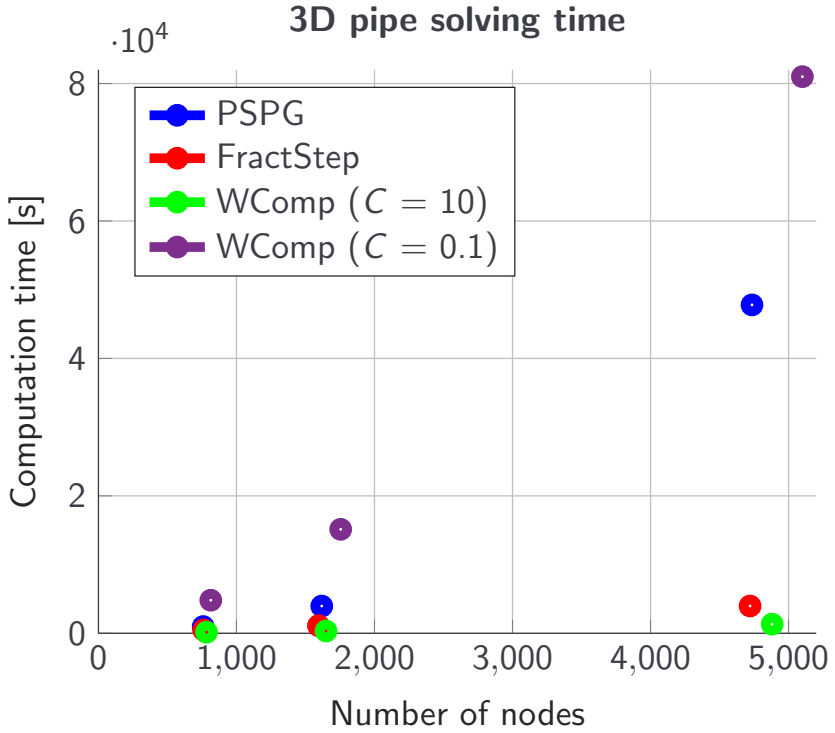
$h_{2D}(m)$	0.2	0.1	0.05	0.04	0.033	0.025	0.02
$h_{3D}(m)$	0.2	0.15	0.1				

Mesh characteristic size

N.B.: computation time computed on the whole run.

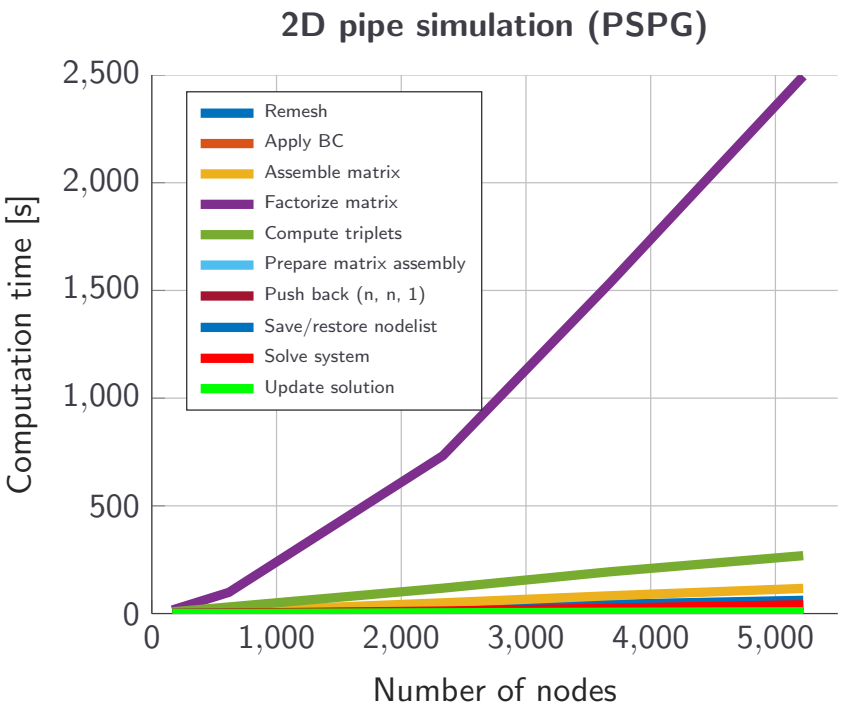
2D and 3D comparison of whole solvers

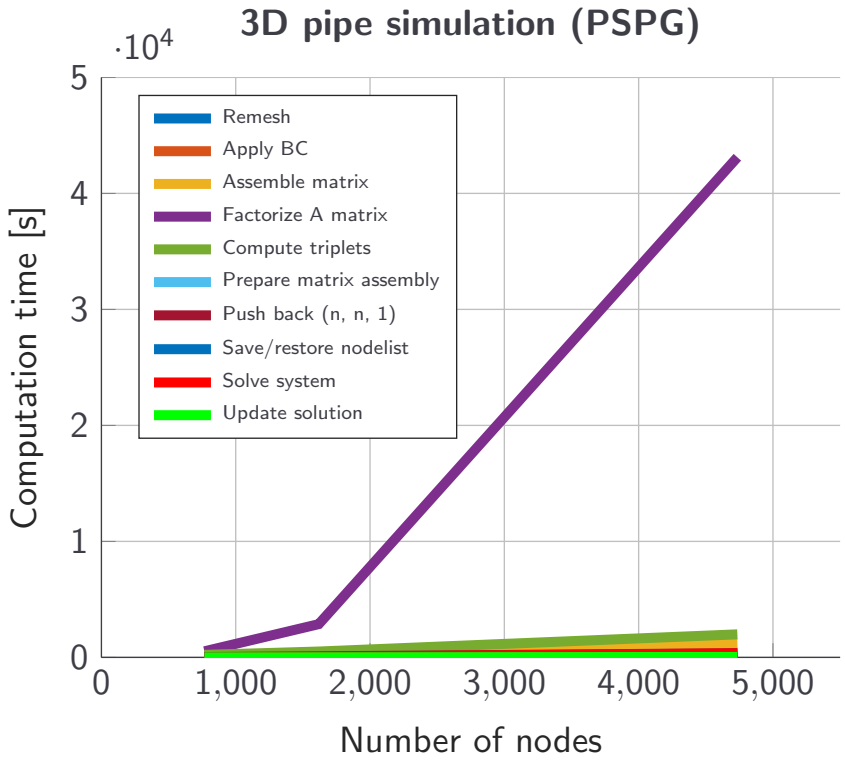




- In 2D, fractional step implicit solver is the best,
- In 3D, there is a way to find a combination of C and K_0 parameter such that the weakly compressible explicit code compete with the fractional step implicit one,
- PSPG code loose in each case.

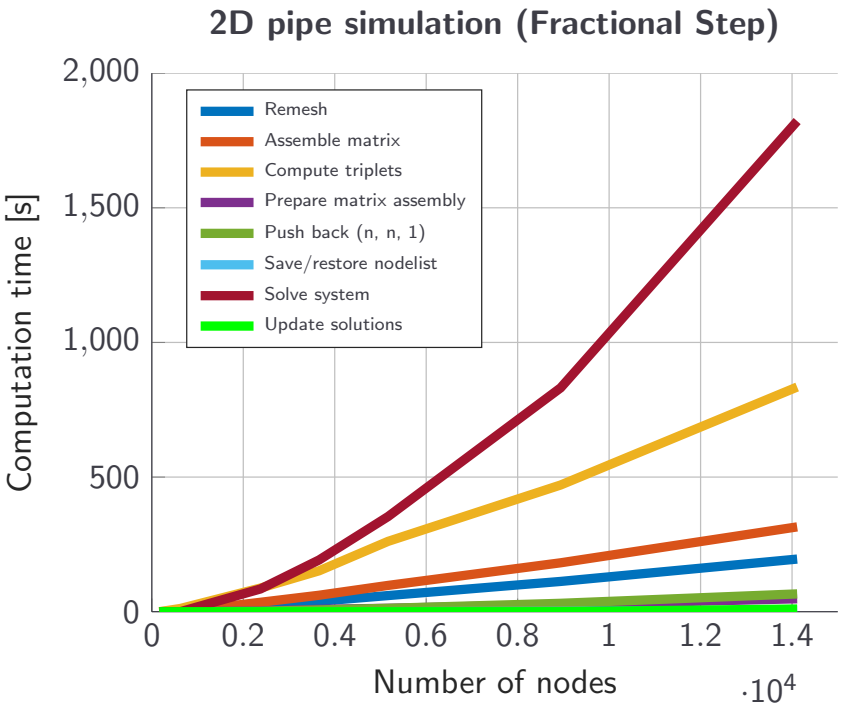
Repartition of computation time in PSPG

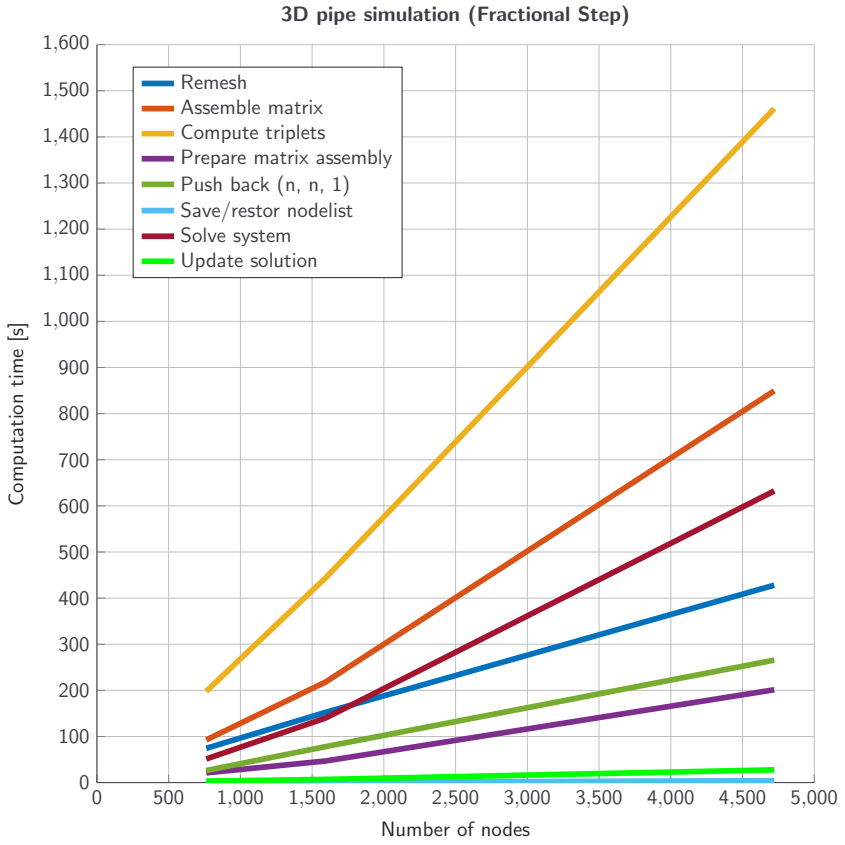




- the factorize matrix step takes most of the computation time $\rightarrow$ no need to check parallel efficiency of PSPG code,
- without finding a better solver than `Eigen::SparseLU`, nothing can be done.

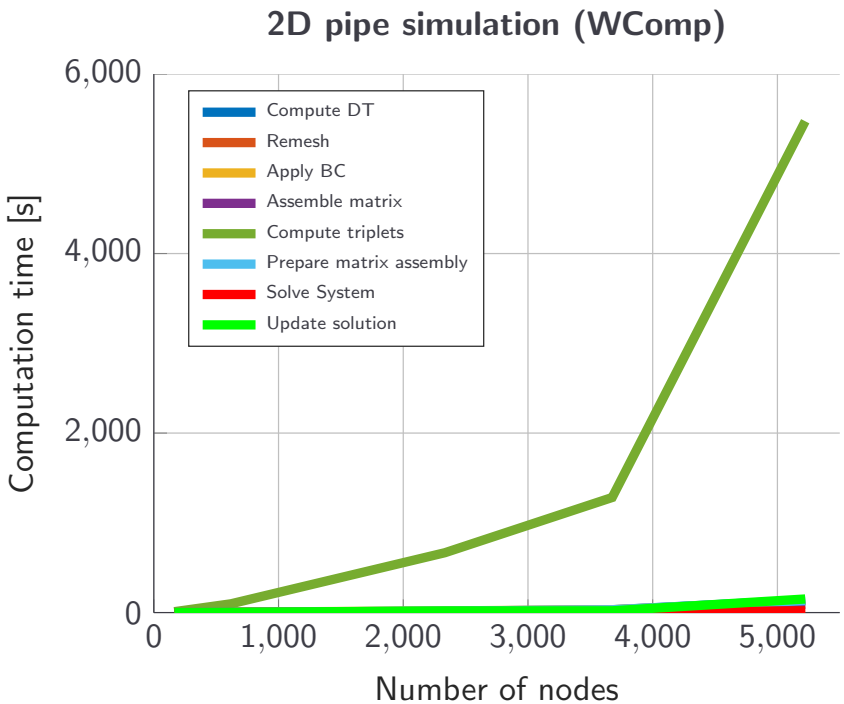
Repartition of computation time in Fractional Step

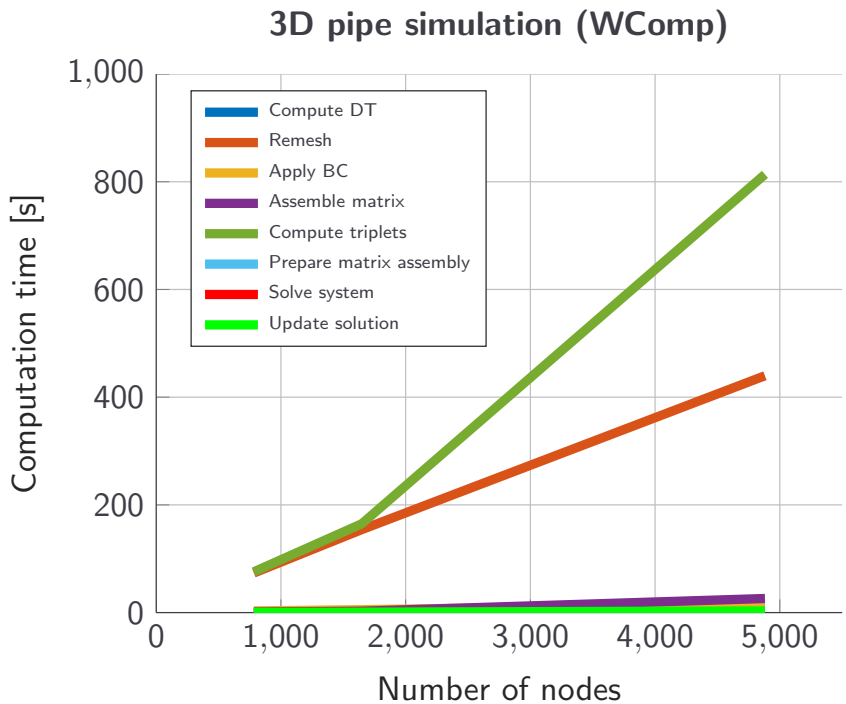




- no factorization step,
- "Solve system" and "Compute triplets" are parallel, "Assemble matrices" and "Remesh" are not.

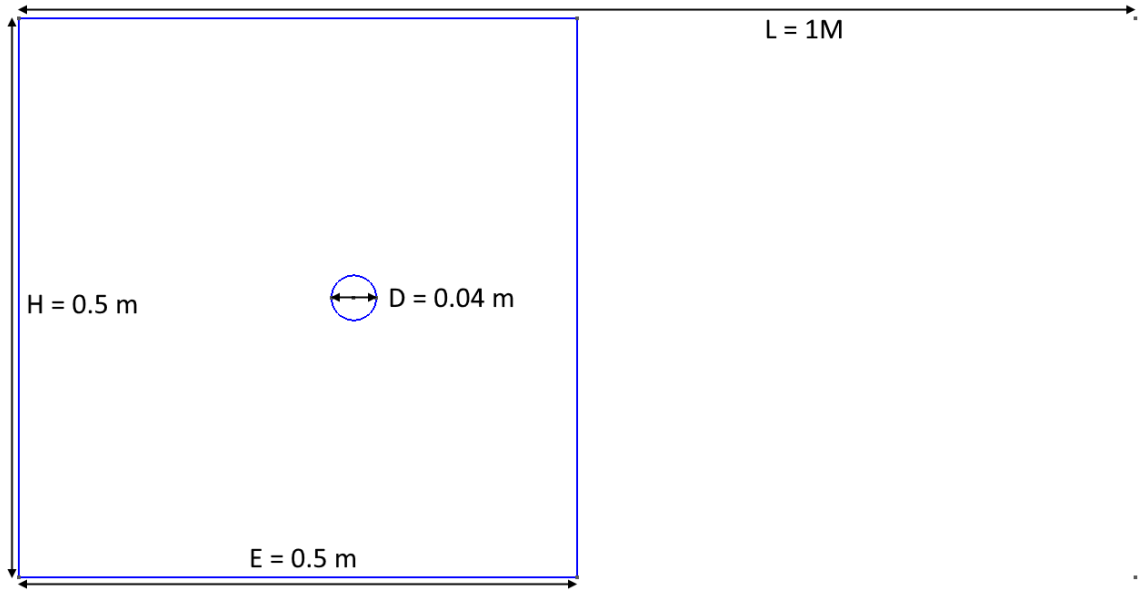
Repartition of computation time in Weakly Compressible





- "Compute triplets" takes a lot of time,
- but it is parallel.

Remeshing time - Problem used 2



Initial geometry of the problem (initial velocity at the left boundary, $0.04\,\mathrm{m\,s^{-1}}$).

ρ	$1000\,\mathrm{kg\,m^{-3}}$
μ	$0.015\,\mathrm{Pa\,s}$

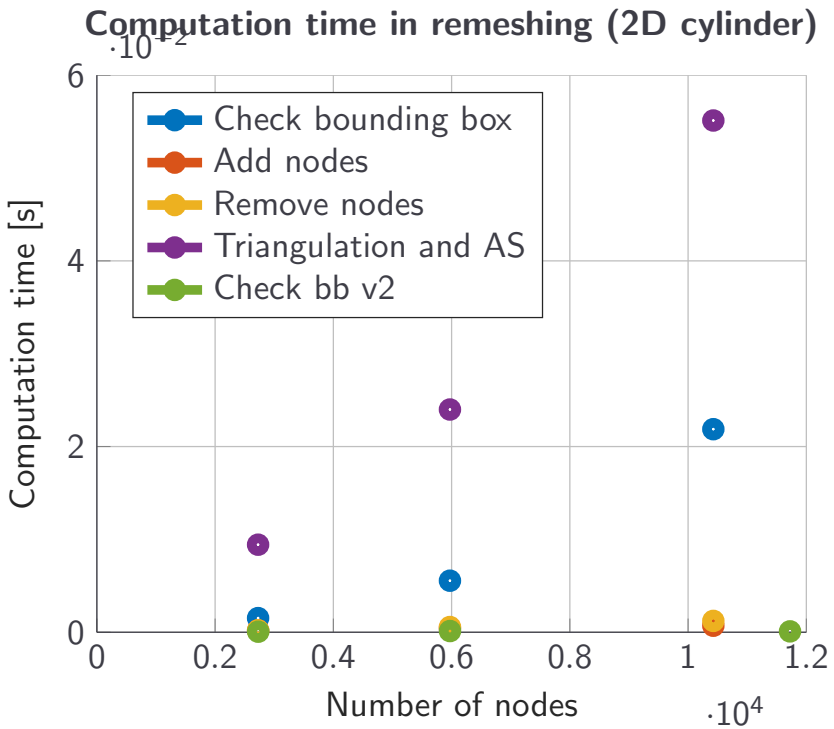
Incompressible fluid parameters

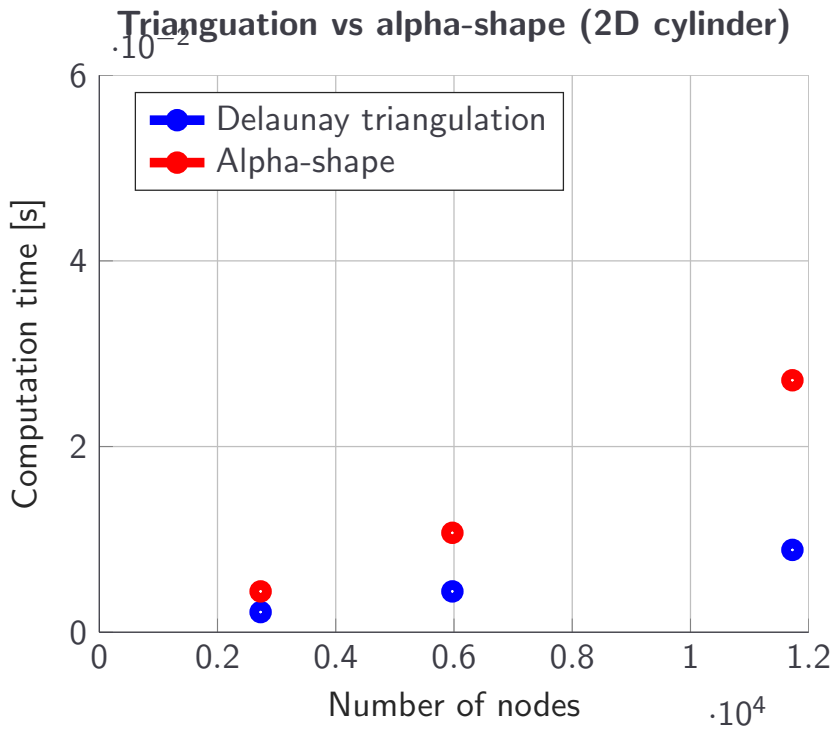
$h_{2D}(m)$	0.15	0.1	0.75

Mesh characteristic size

N.B.: computation time computed on one function execution.

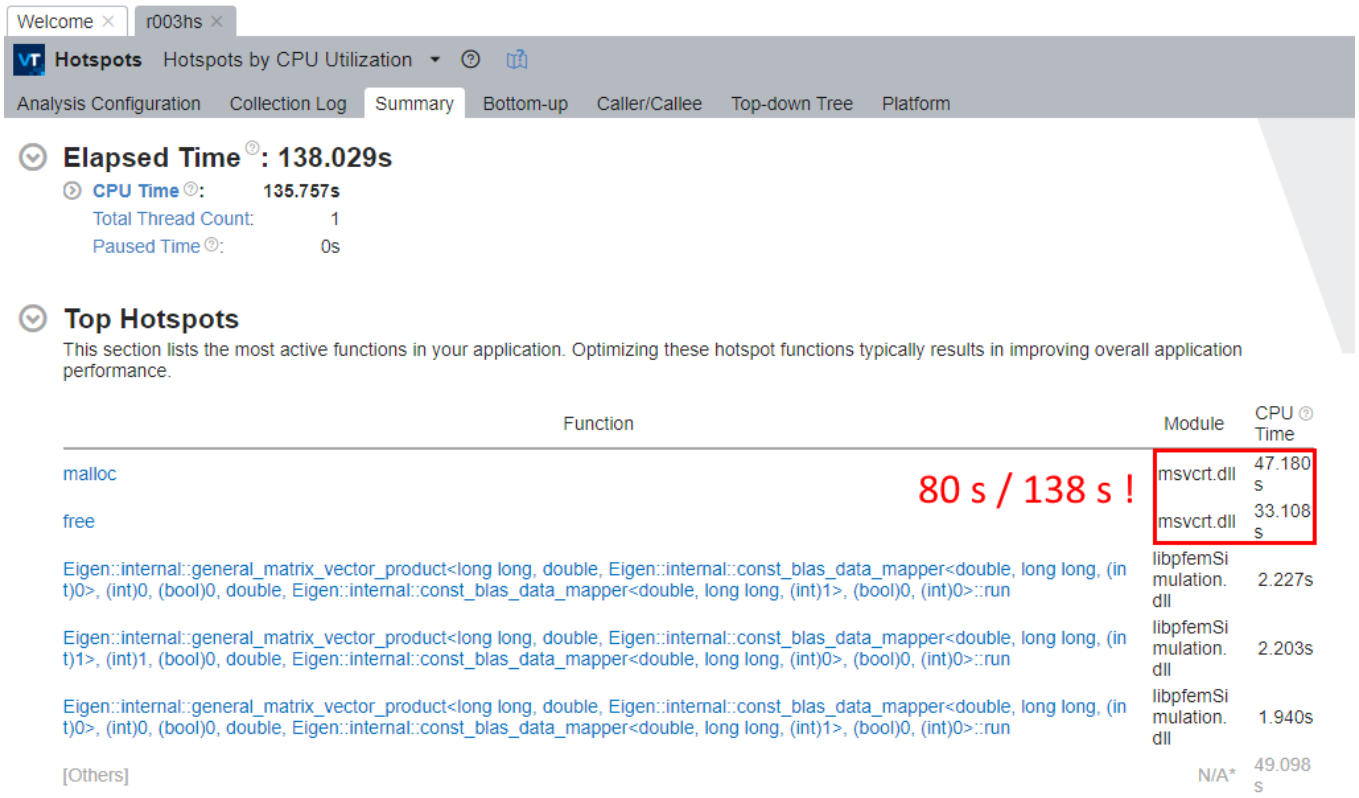
Repartition of computation time in remeshing





- the `Mesh::triangulateAlphaShape2D` function takes a little bit more time than what CGAL do (expected but can it be decreased ?),
- the `Mesh::checkBoundingBox` takes a lot of time but it can be fixed by changing the order of the functions in `Mesh::remesh`.

Further work



Intel VTune Profiler hotspot analysis of weakly compressible explicit code.

- comes from the use of `Eigen::MatrixXd` which are dynamic and thus needs to allocate memory,
- can be fixed by using fixed size matrices but refactoring of code needed to handle correctly 2D and 3D cases (test: 1712 $\rightarrow$ 1000 s on a pipe problem with $h = 0.04$),
- implicit codes have less this problem (6s/26s).

OpenMP performance

Speed-up ratio and parallel efficiency definition

Let τ be a vector containing the execution time of the program for a certain number of thread such that τ_i is the execution time using i threads.

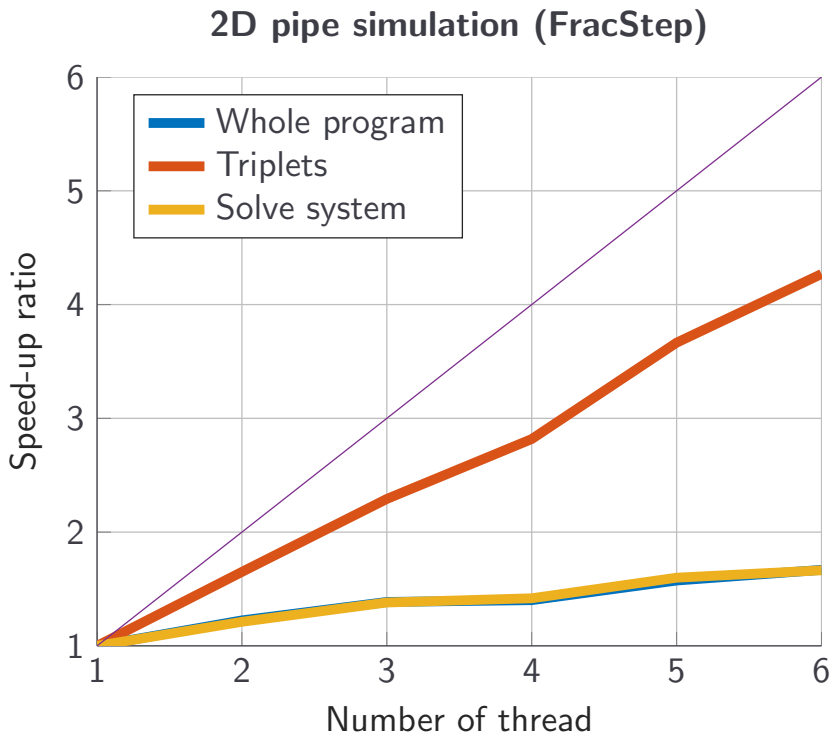
The speed-up ratio is defined as:

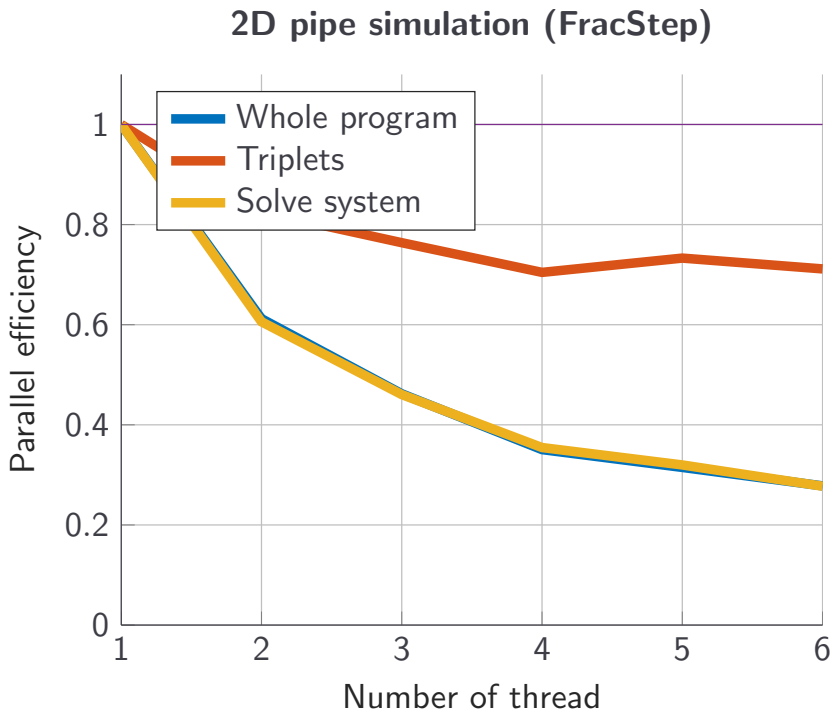
$$\gamma = \frac{\tau_1}{\tau_i}$$

The parallel efficiency is defined as:

$$\eta = \frac{\tau_1}{i\tau_i}$$

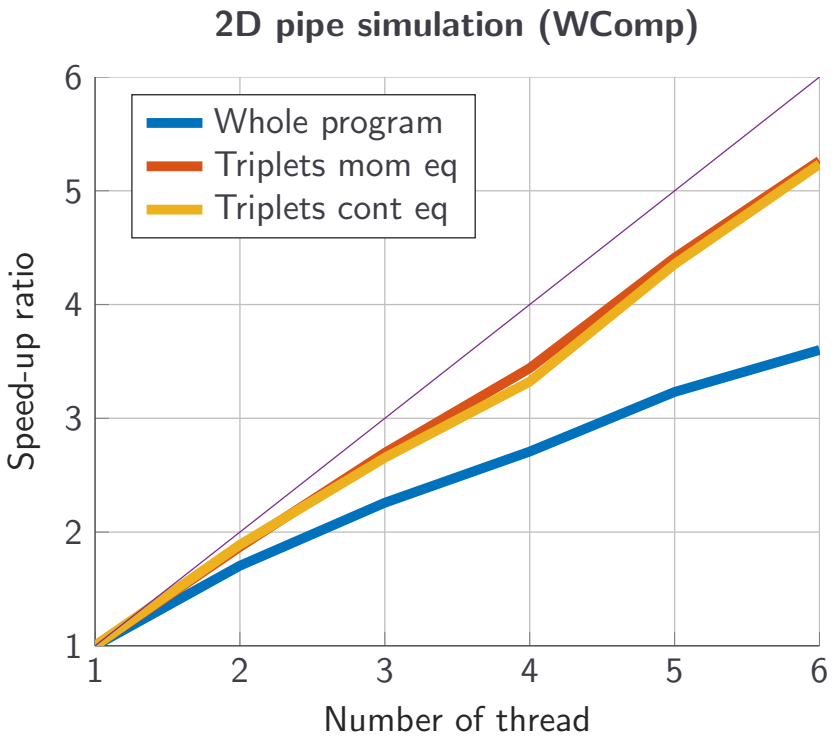
Speed-up ratio and parallel efficiency of Fractional Step solver

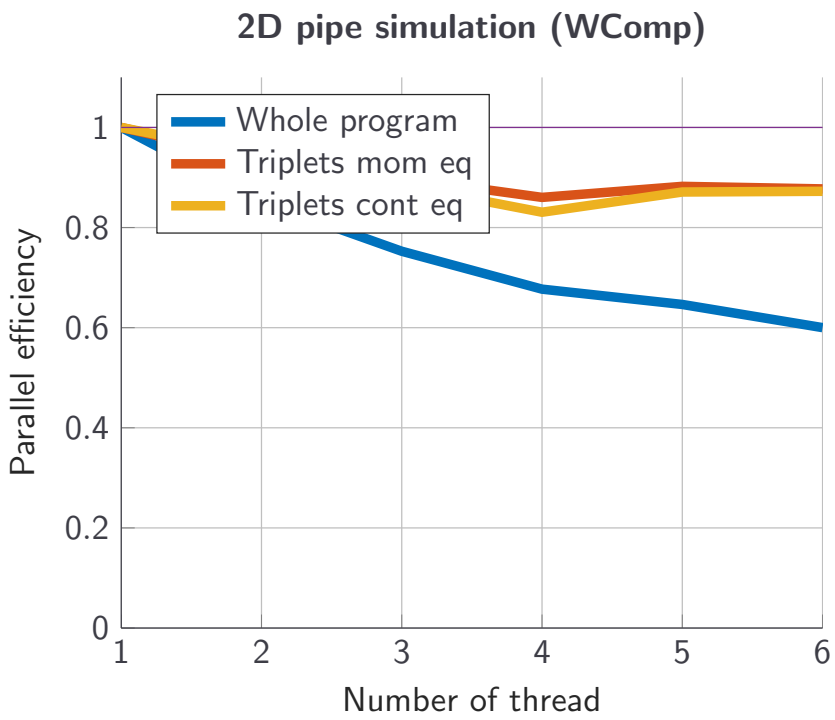




- efficiency is killed by `Eigen::ConjugateGradient` poor performance,
- look to improve triplets computation efficiency.

Speed-up ratio and parallel efficiency of Weakly compressible solver





- global efficiency is decreased due to serial part of the code.