

Evaluation Tools

Dr. Susanne Kilian

hhpberlin - Ingenieure für Brandschutz

10245 Berlin - Germany

Parameters for Pressure Solver

`&PRES VELOCITY_TOLERANCE = 0.001, MAX_PRESSURE_ITERATIONS = 100`



Iterate until normal components of velocity
towards internal solids and their differences along
mesh interfaces are smaller than tolerance

default: $\delta x/2$



Maximum number of iterations
which are allowed to be performed
(even if tolerance is not reached)

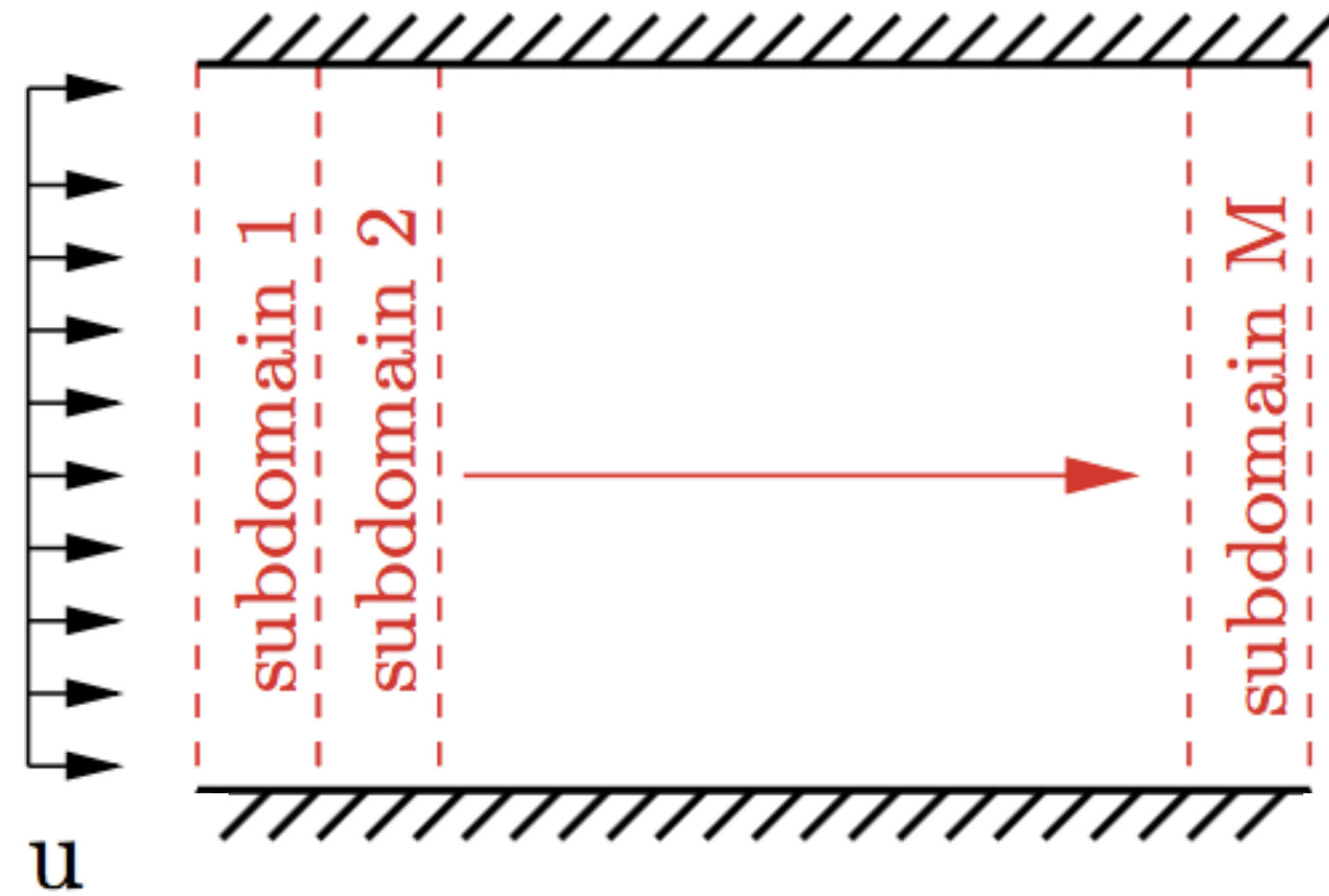
default: 10

→ Increased overhead per time step depending on individual situation

Optimal choice of velocity tolerance?

- depends on individual characteristics of given case
 - number of meshes
 - number/kind of internal obstructions
 - flow speed and directions
 - proper balance between accuracy and computational efficiency must be found
- Exemplary studies to analyze effects for simple test cases

Example 1: Flow through 2D-pipe



- nonviscid flow
- free-slip wall conditions
- open boundary on the right
- specified inflow $u(t)$ from the left

→ constant parallel flow for each time step

Analytical relation between pressure and velocity from momentum equation:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) + \nabla p = 0 \quad \xrightarrow{\text{advection term vanishes}} \quad \rho \frac{\partial \mathbf{u}}{\partial t} = -\nabla p$$

Example 1: Flow through 2D-pipe

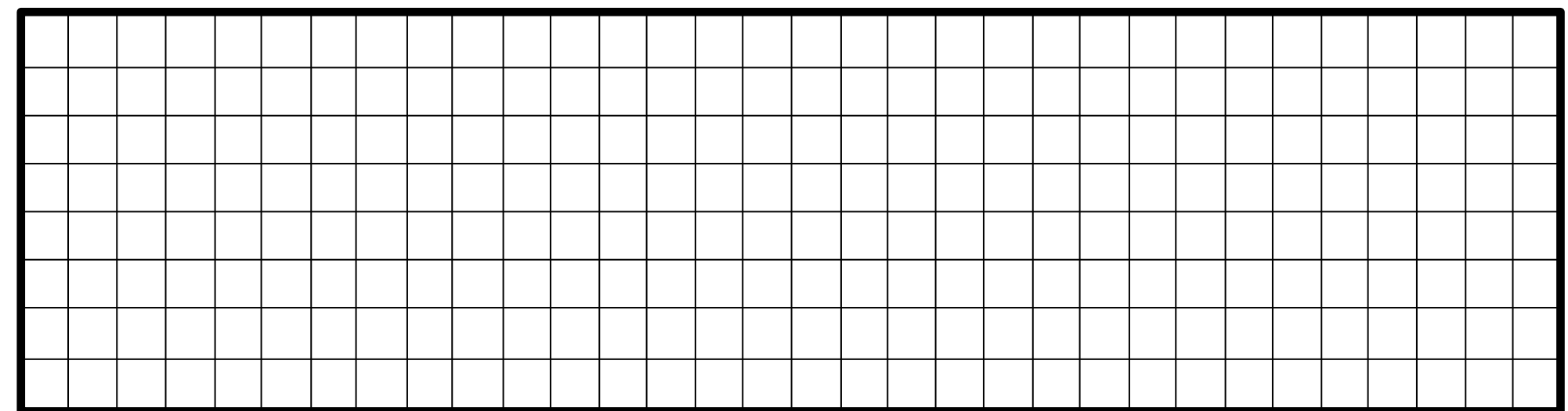
Test of two different inflow conditions

- linear inflow: $u(t) = u_0 t$ → Test case [linear2d](#)
- sinusoidal inflow: $u(t) = \sin(2\pi t)$ → Test case [sine2d](#)

Base pipe geometry with 1 mesh:

(with different grid resolutions)

M=1



Example 1: Flow through 2D-pipe

Test of two different inflow conditions

- linear inflow: $u(t) = u_0 t$ → Test case [linear2d](#)
- sinusoidal inflow: $u(t) = \sin(2\pi t)$ → Test case [sine2d](#)

Velocity field for 8 meshes:

(constant parallel flow)

M=8



Set of evaluation tools

Tools: Collection of simple python-tools to simplify case studies

- Split given rectangular geometry into multiple meshes
- Plot CSV-data of single decomposition
- Compare CSV-data of different decompositions

→ Analyze influences of domain decomposition and pressure velocity settings

Split rectangular single- into multi-mesh case

splitMesh.py

```
python <tools>/splitMesh.py <fdsfile> <command>
```

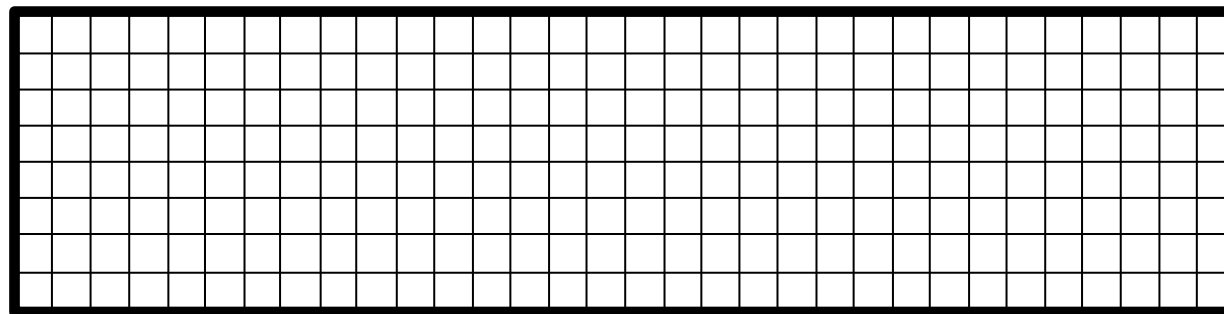
- <tools> is the path to the **Tools**-folder
- <fdsfile> is the path to the fds file to be split
- <command> is the split factor, number of digits in pattern corresponds to dimension, e.g. 2x2 for a 2D split or 3x1x4 for a 3D split

Note:

- based on &MULT-namelist
- new geometries are stored in **input**-folder, must be copied to desired location for job-start

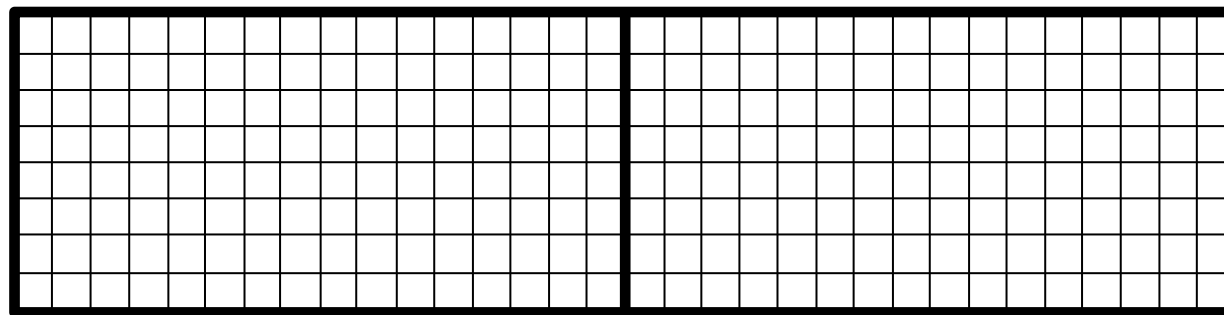
Exercise: Split pipe-case into multiple meshes

M=1



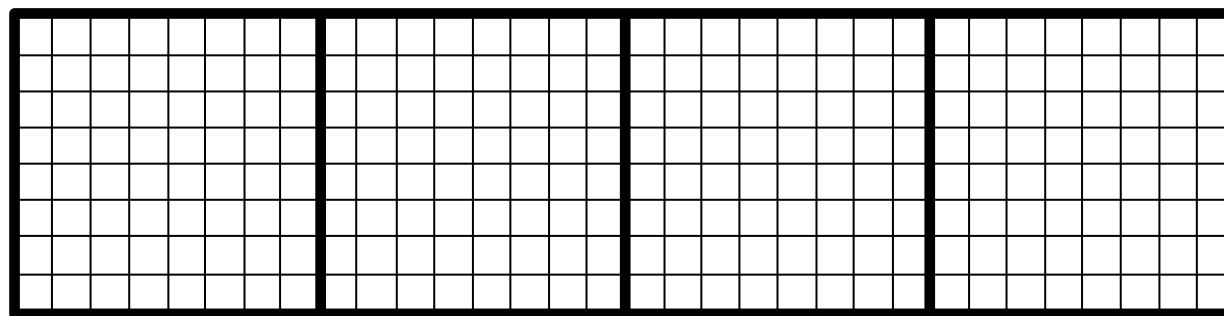
`sine2d.fds` (base geometry, 80x20 cells) → `sine2d_1x1.fds`

M=2



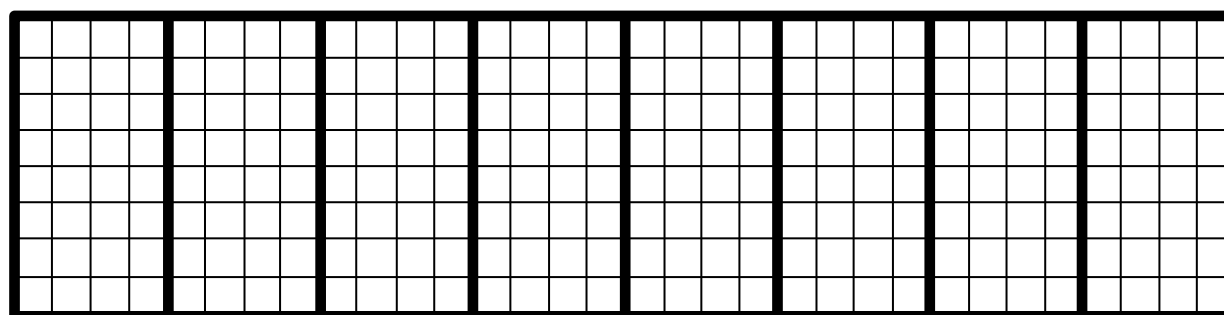
`python splitMesh.py sine2d.fds 2x1` → `sine2d_2x1.fds`

M=4



`python splitMesh.py sine2d.fds 4x1` → `sine2d_4x1.fds`

M=8



`python splitMesh.py sine2d.fds 8x1` → `sine2d_8x1.fds`

Note: For reasons of notational consistency a corresponding `sine2d_1x1.fds` is automatically generated (same as `sine2d.fds`)

Identify device definitions for later comparisons

`linear2d.fds` \triangleq `linear2d_1x1.fds`, `sine2d.fds` \triangleq `sine2d_1x1.fds`

- based on default settings for pressure solver
- include the following device definitions

```
&DEVC XYZ=-0.4,0.0,0.0, QUANTITY='U-VELOCITY', ID='ULEFT' /  
&DEVC XYZ= 0.4,0.0,0.0, QUANTITY='U-VELOCITY', ID='URIGHT' /
```

```
&DEVC XYZ= 0.0,0.0,0.0, QUANTITY='MAXIMUM VELOCITY ERROR', ID='VEL_ERROR' /  
&DEVC XYZ= 0.0,0.0,0.0, QUANTITY='PRESSURE ITERATIONS' , ID='PRES_ITE' /
```

Exercise: Run the different cases

```
mpirun -np M fds linear2d_Mx1.fds  
mpirun -np M fds sine2d_Mx1.fds
```

Compare CSV-data within a single simulation

plotCSV.py

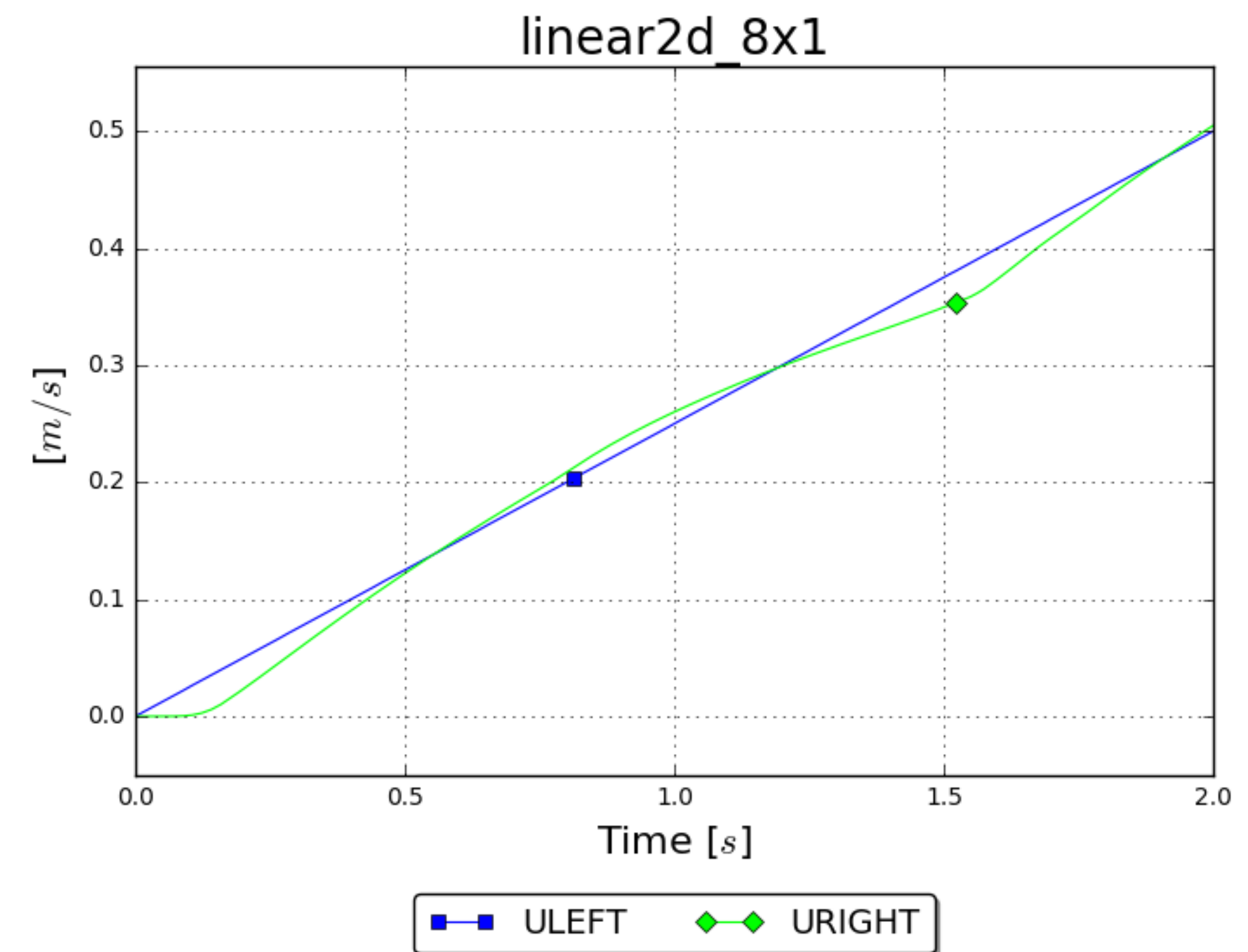
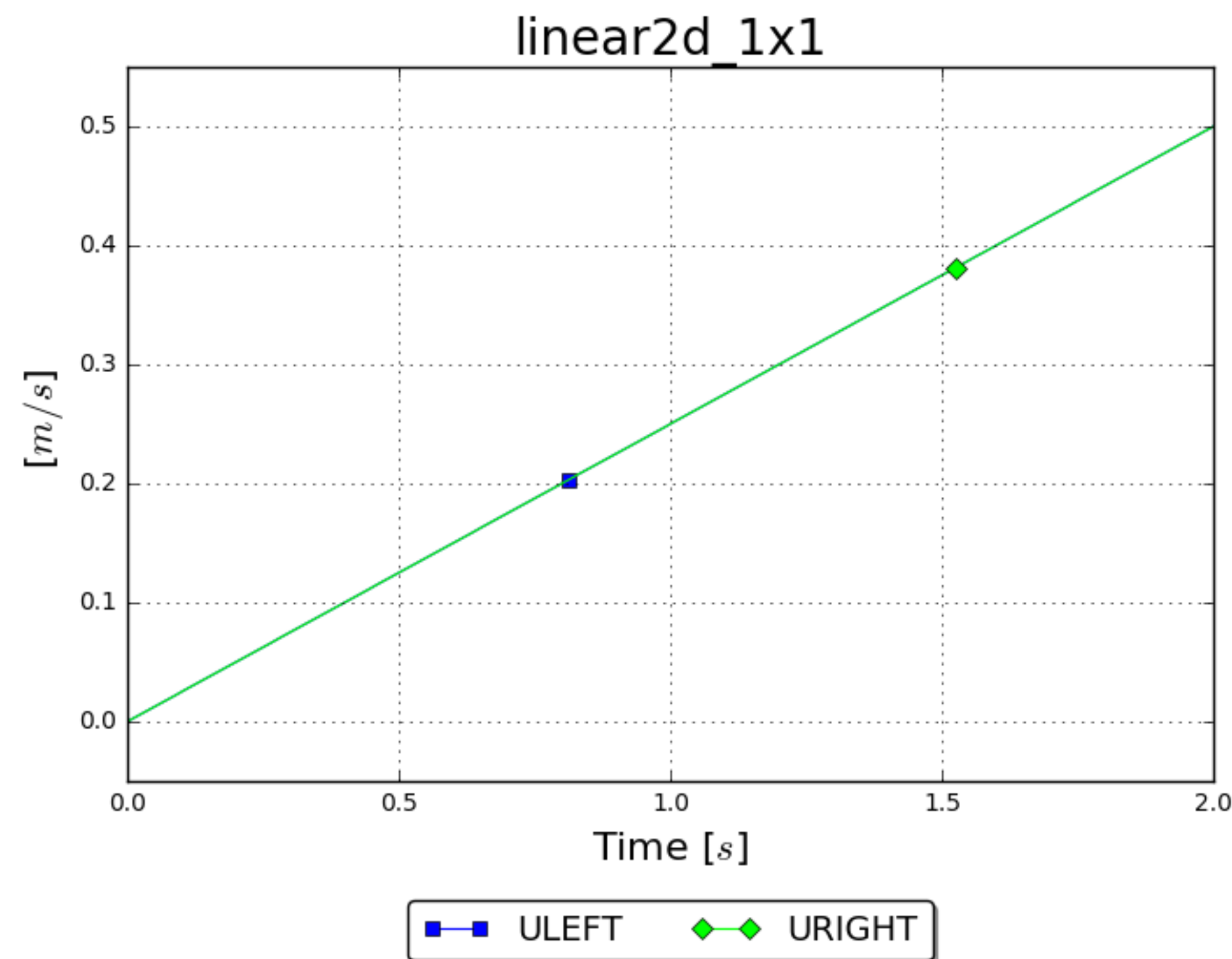
```
python <tools>/plotCSV.py <csvfile> <values> [-o <plot>]
```

- <tools> is the path to the **Tools**-folder
- <csvfile> is the path to the csv file containing the plot data
- <values> can be any list of device quantities appearing in the specified <csvfile>
- <plot> is the optional name of the output file. If not specified, an accordingly named png will be created in a **pictures** sub-folder

Note: If multiple devices shall be illustrated in one plot, units must match!

linear2d: Compare in- and outflow for different M

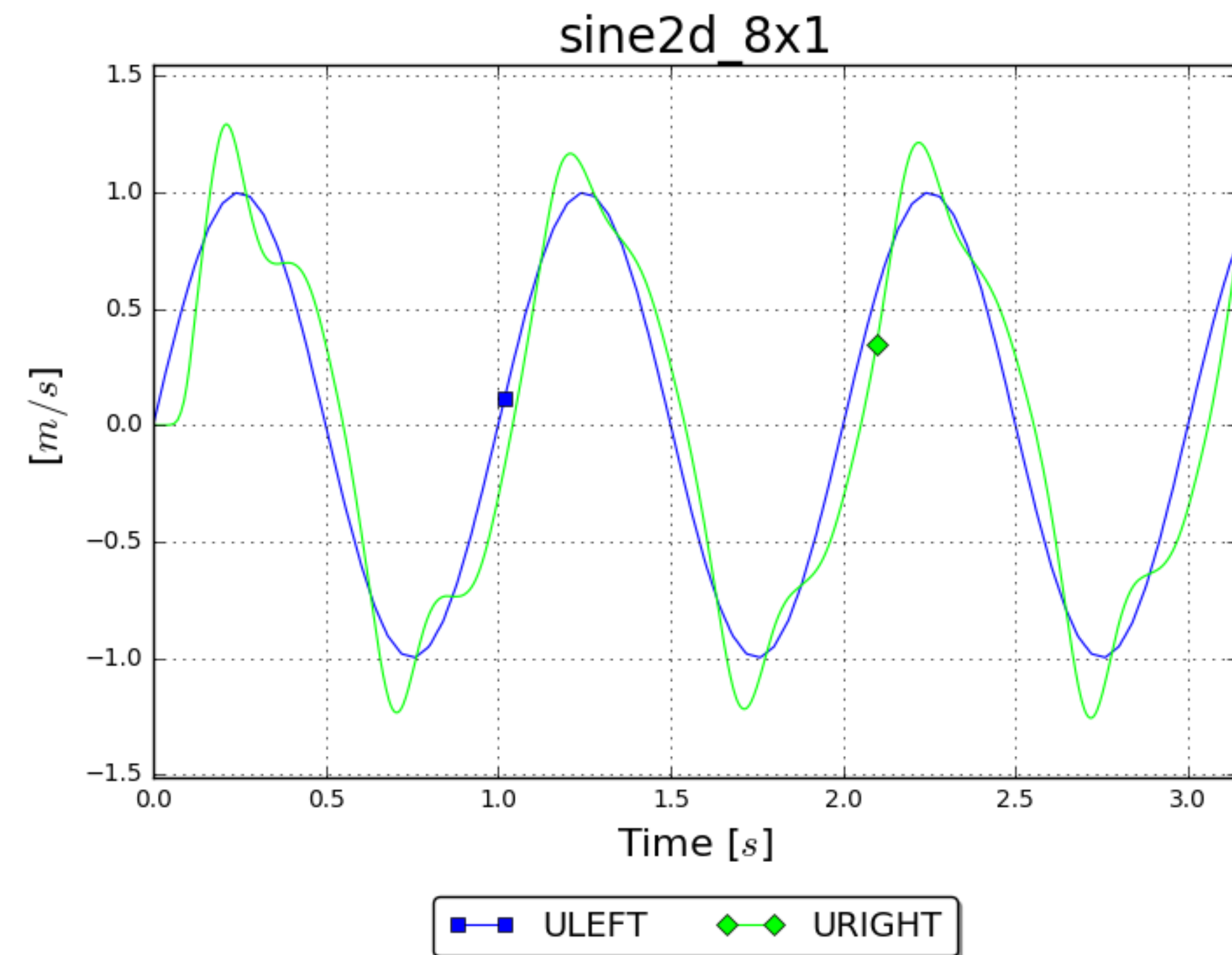
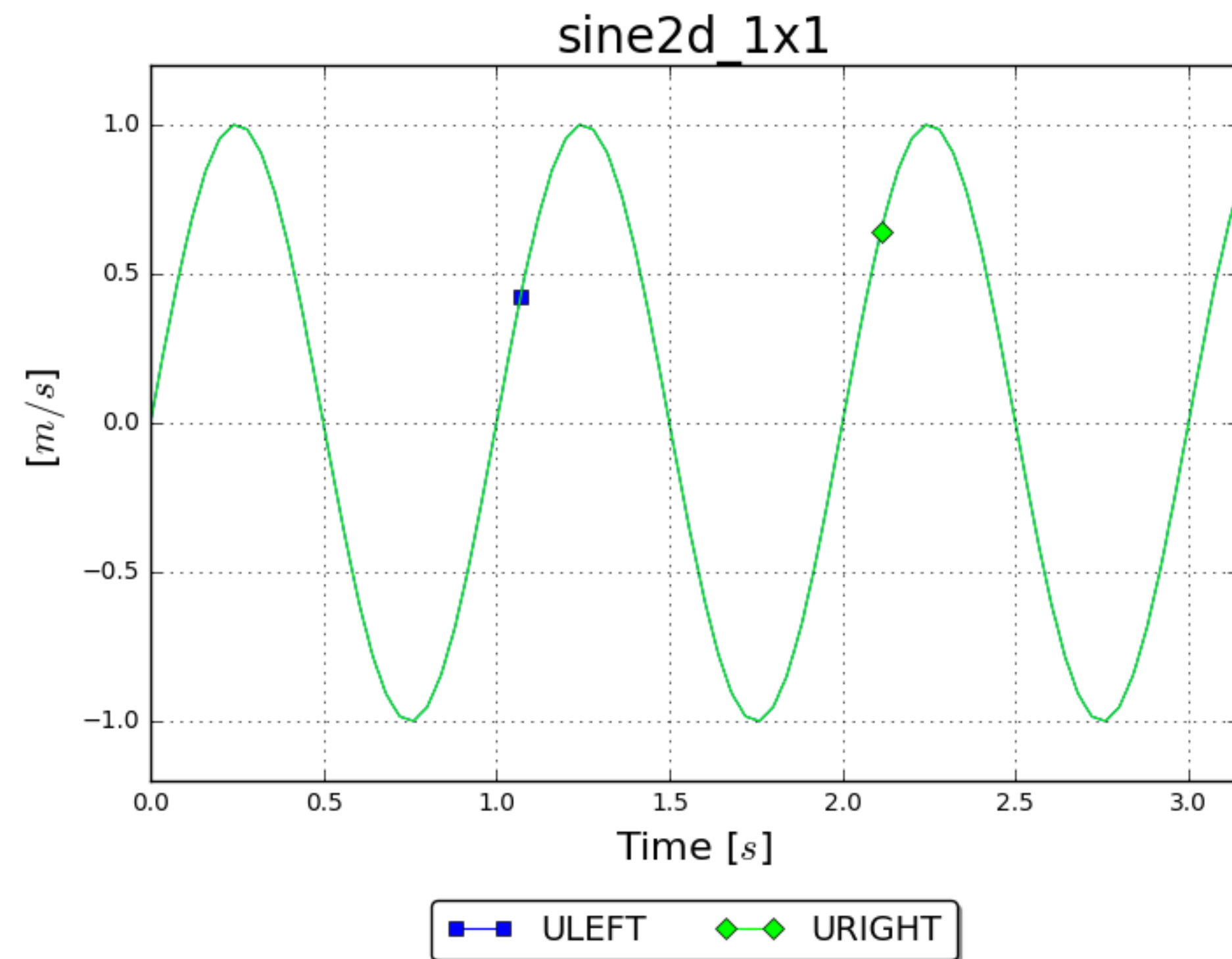
`python plotCSV.py linear2d_Mx1_devc.csv ULEFT URIGHT -o pictures/out.png`



→ perfect match for 1-mesh, slight differences for 8-mesh case

sine2d: Compare in- and outflow for different M

`python plotCSV.py sine2d_Mx1_devc.csv ULEFT URIGHT -o pictures/out.png`



→ perfect match for 1-mesh, obvious differences for 8-mesh case

Compare single CSV-value of different simulations

compareCSV.py

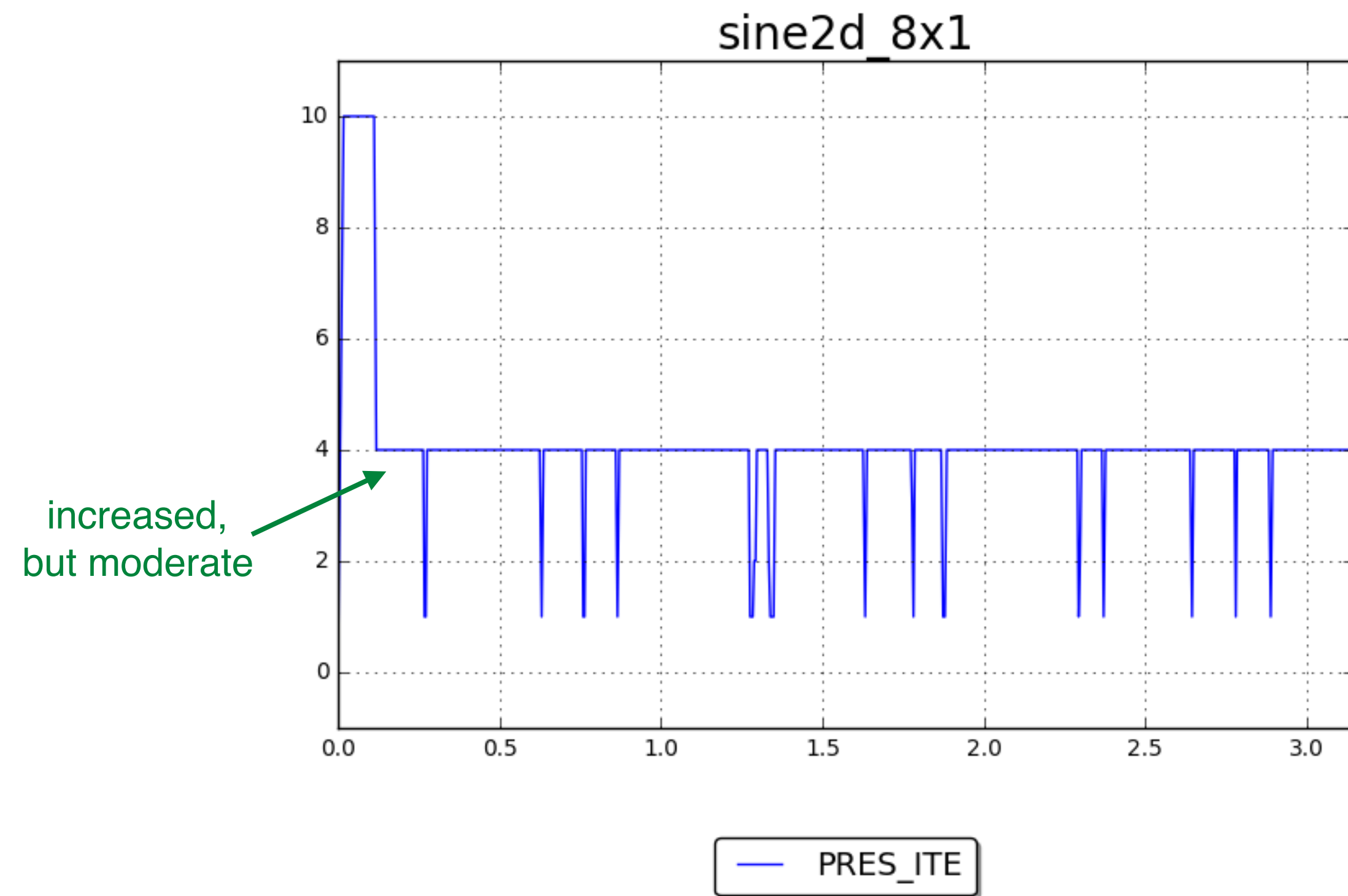
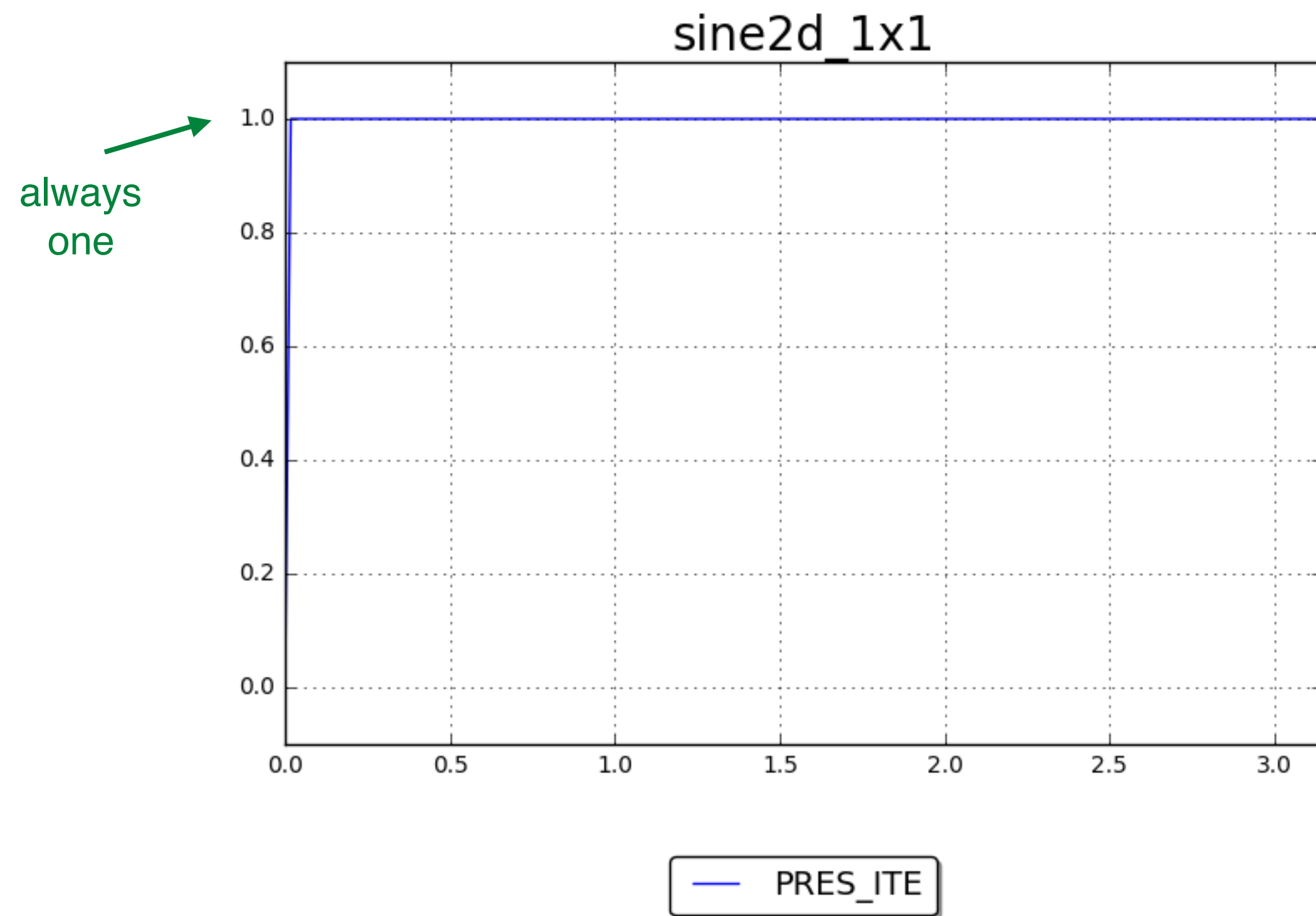
```
python <tools>/compareCSV.py <csvfiles> [-o <plot>] <value>
```

- <tools> is the path to the **Tools**-folder
- <csvfiles> is a blank separated list of paths to the csv files containing the plot data
- <plot> is the optional name of the output file. If not specified, an accordingly named png will be created in a **pictures** sub-folder
- <value> can be any device quantity appearing in **all** of the specified <csvfiles>

Note: Device value must be the same in all specified simulations, set of csv files can also be collected in input file which is specified with -i option (see later examples)

sine2d: Pressure iterations for different M

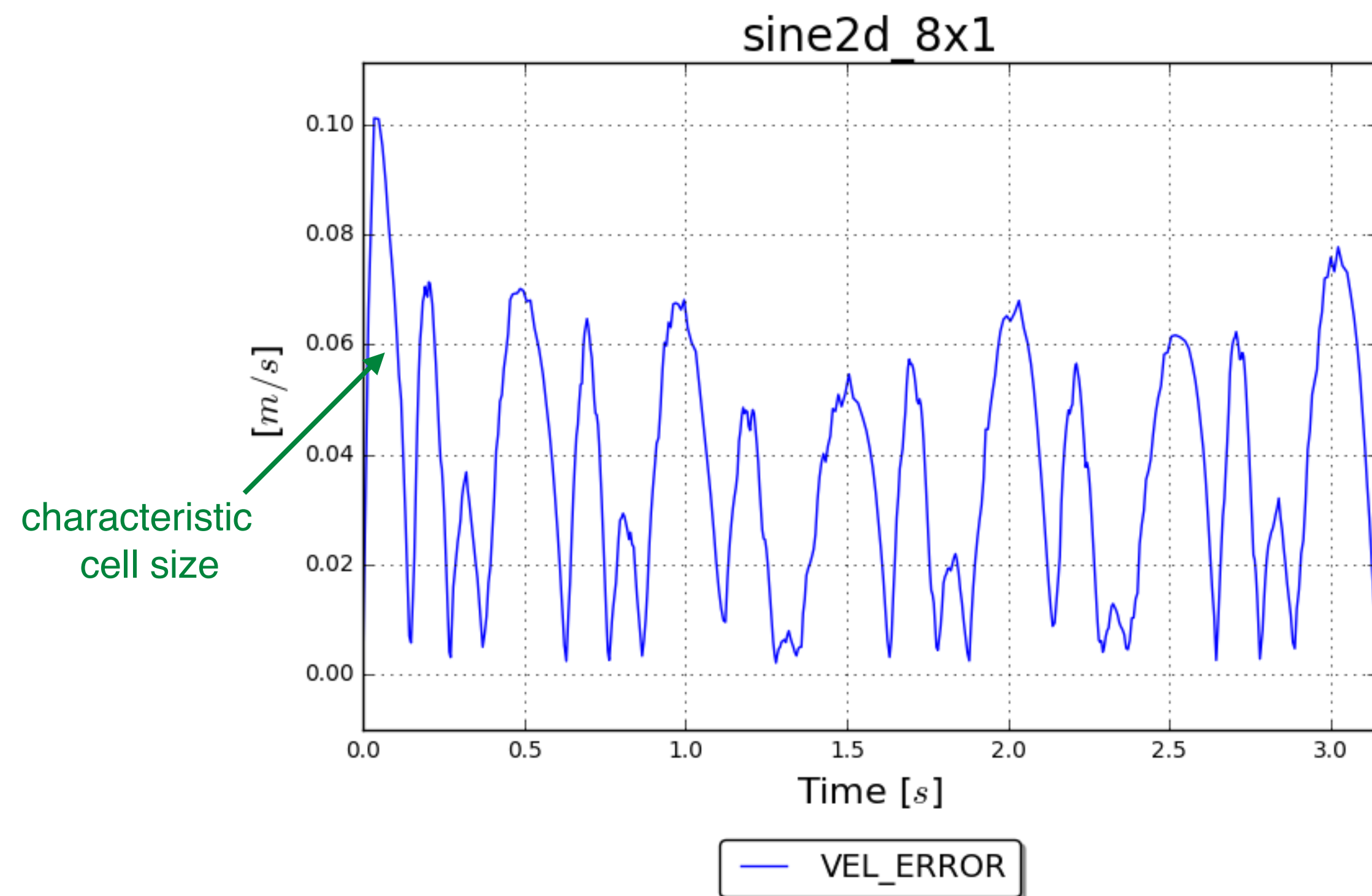
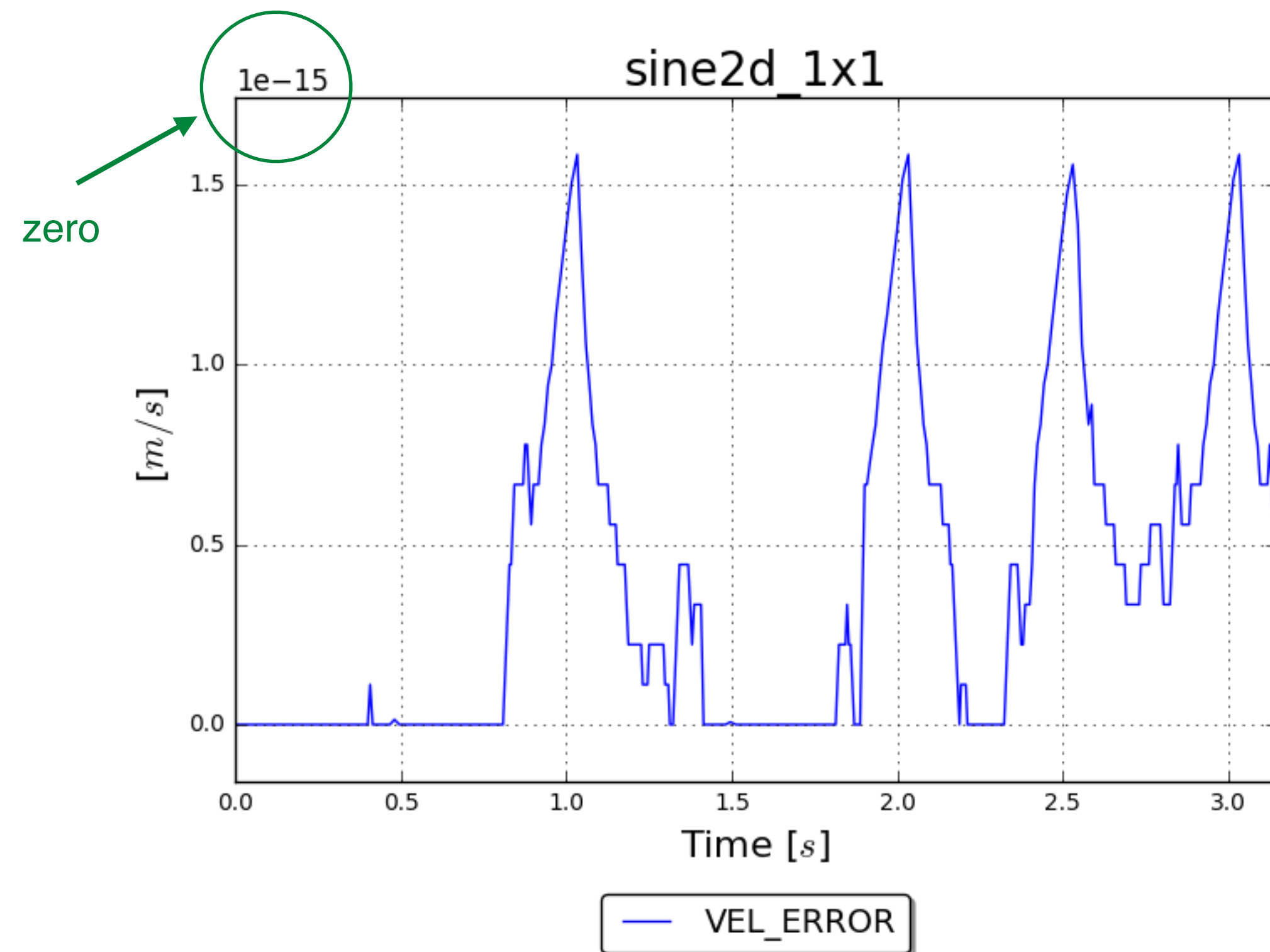
`python plotCSV.py sine2d_Mx1_devc.csv PRES_ITE -o pictures/out.png`



→ slightly increased number of pressure iterations für 8-mesh case

sine2d: Velocity error for different M

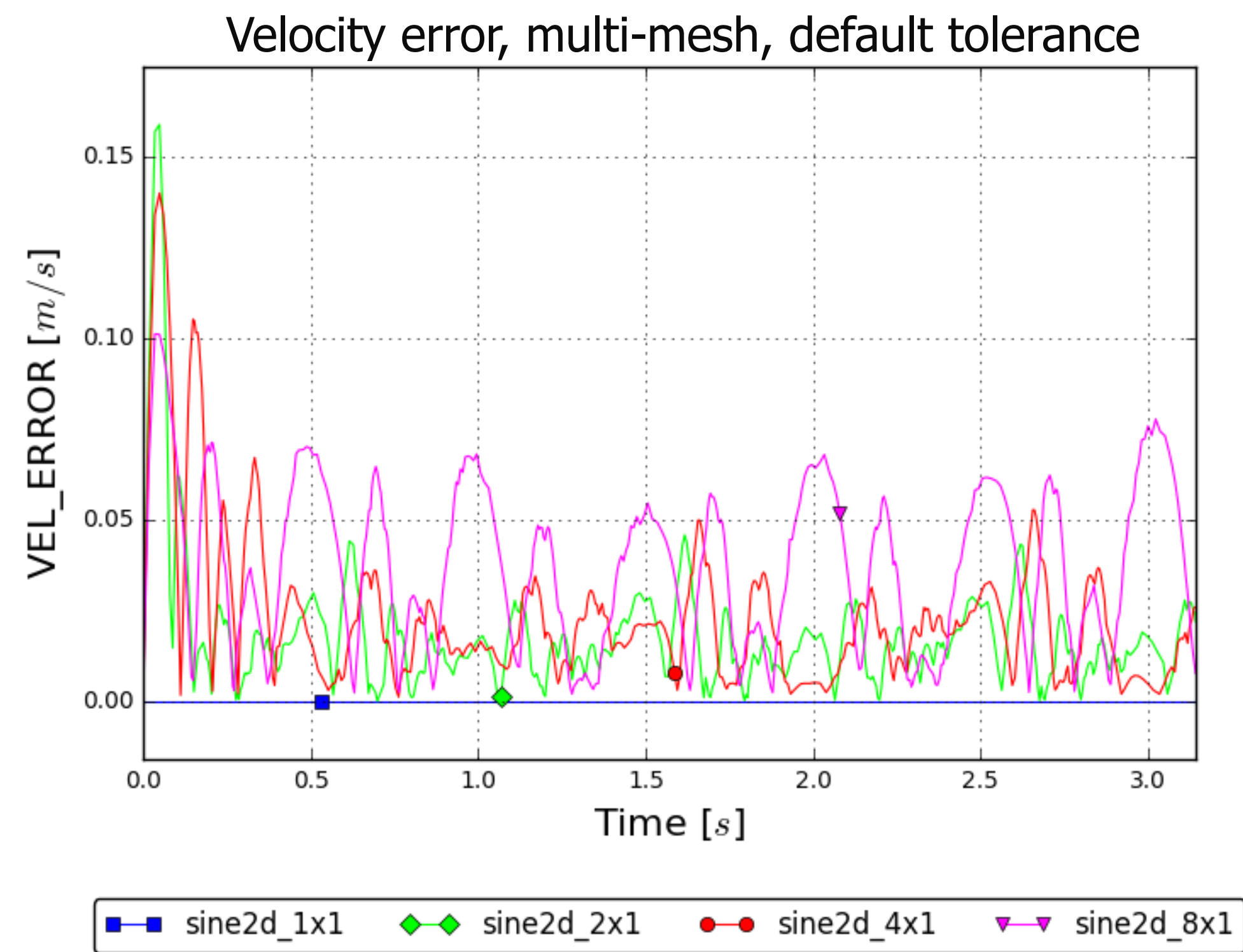
`python plotCSV.py sine2d_Mx1_devc.csv VEL_ERROR -o pictures/out.png`



→ zero velocity error for 1-mesh, comprehensible velocity error for 8-mesh case

sine2d: Velocity errors for all M in one plot

`python compareCSV.py -i compareCSV.txt -o pictures/out.png VEL_ERROR`



default
tolerance,
different
geometries

compareCSV.txt:

sine2d_1x1_devc.csv

sine2d_2x1_devc.csv

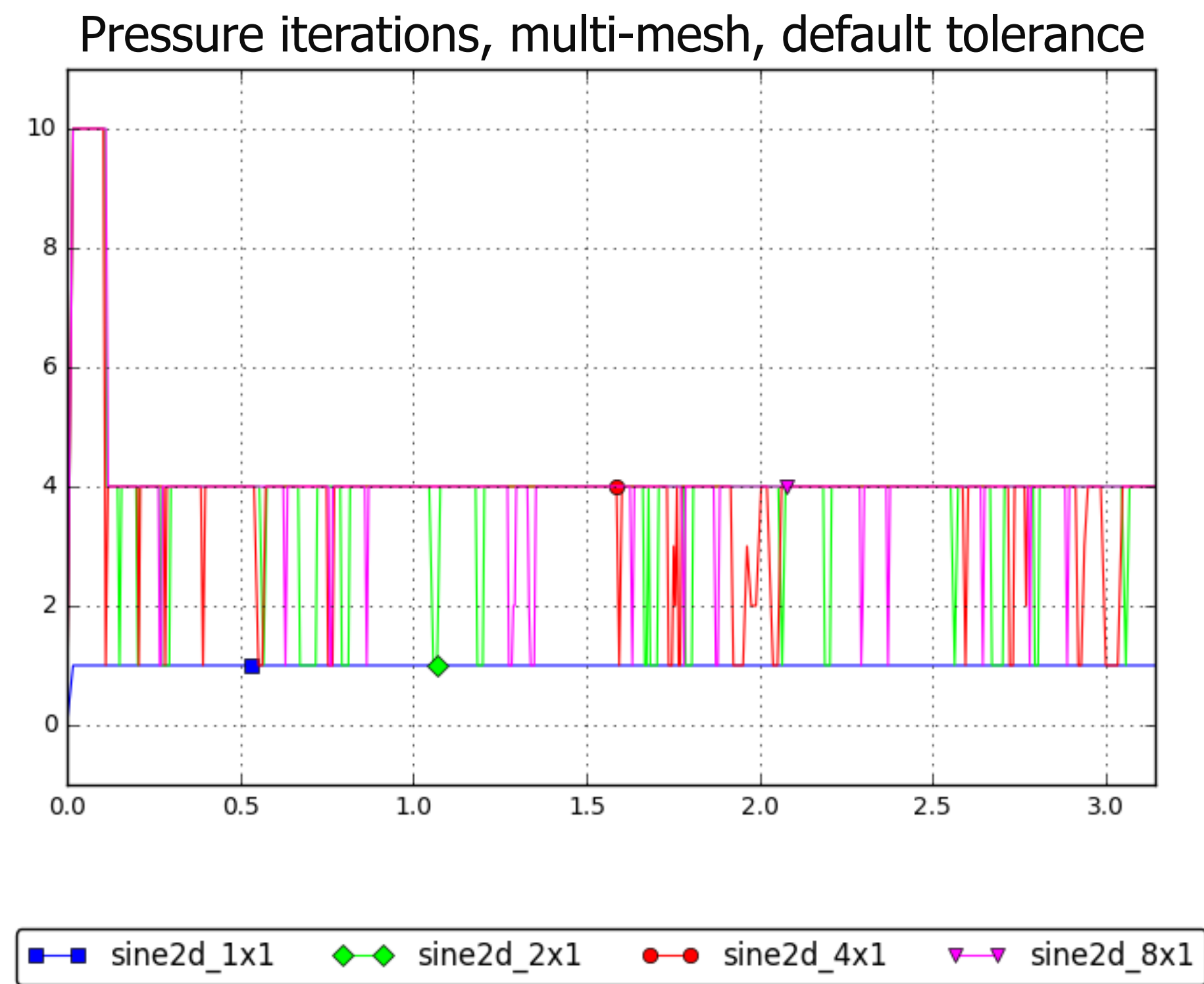
sine2d_4x1_devc.csv

sine2d_8x1_devc.csv

→ velocity error grows with increasing number of meshes

sine2d: Pressure iterations for all M in one plot

```
python compareCSV.py -i compareCSV.txt -o pictures/out.png PRES_ITE
```



default
tolerance,
different
geometries

compareCSV.txt:

sine2d_1x1_devc.csv
sine2d_2x1_devc.csv
sine2d_4x1_devc.csv
sine2d_8x1_devc.csv

→ number of pressure iterations grows for increasing number of meshes

sine2d: Stricter tolerances for pressure solver

Use smaller tolerances in &PRES line to reduce velocity error:

- **sine2d_tol-3.fds:**

```
&PRES VELOCITY_TOLERANCE = 1E-3, MAX_PRESSURE_ITERATIONS = 100 /
```

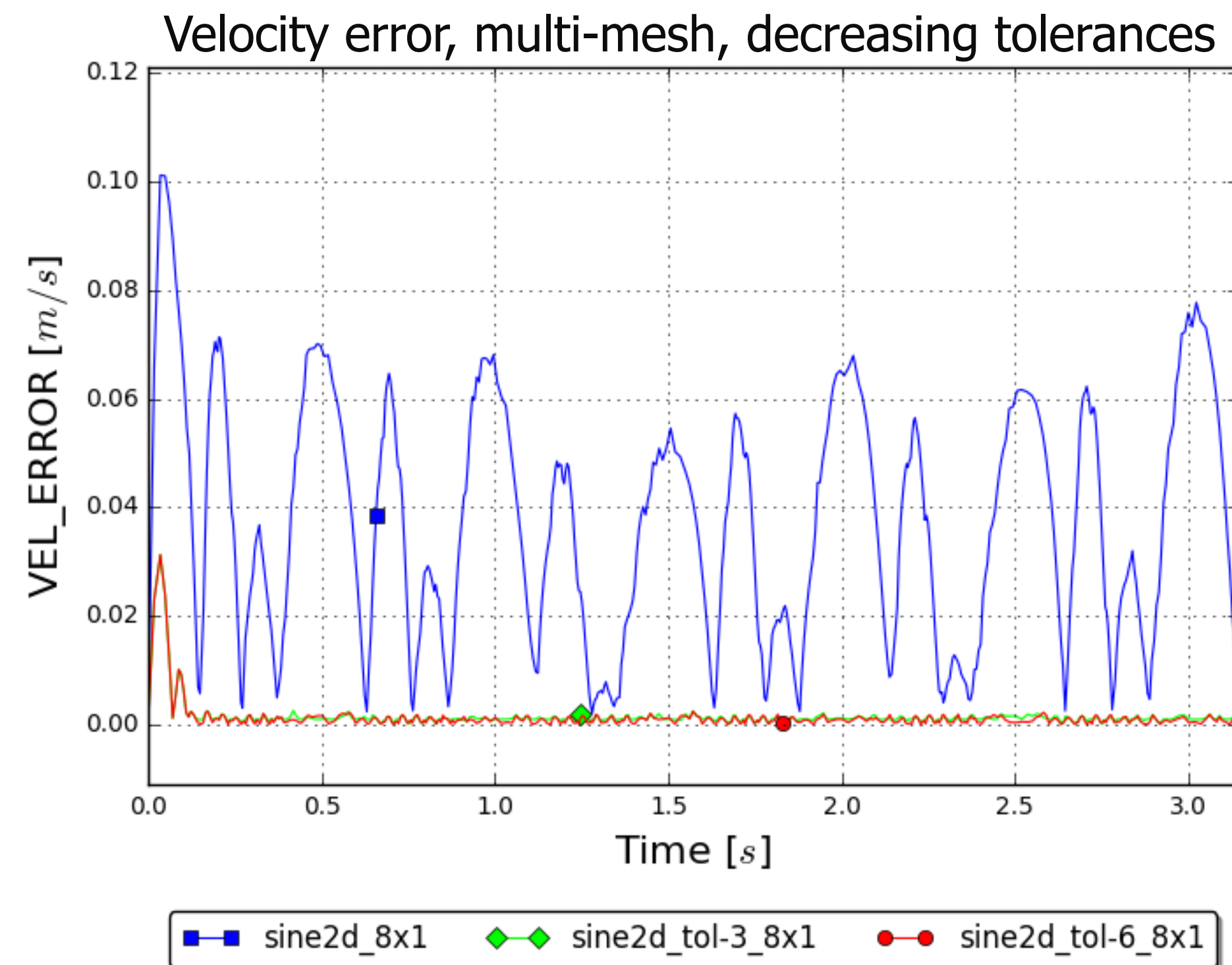
- **sine2d_tol-6.fds:**

```
&PRES VELOCITY_TOLERANCE = 1E-6, MAX_PRESSURE_ITERATIONS = 100 /
```

Exercise: apply [splitMesh.py](#), [plotCSV.py](#) and [compareCSV.py](#) for stricter tolerances

sine2d: Stricter tolerances for pressure solver

```
python compareCSV.py -i compareCSV.txt -o pictures/out.png VEL_ERROR
```



different
tolerances,
same
geometry

compareCSV.txt:

sine2d_8x1_devc.csv

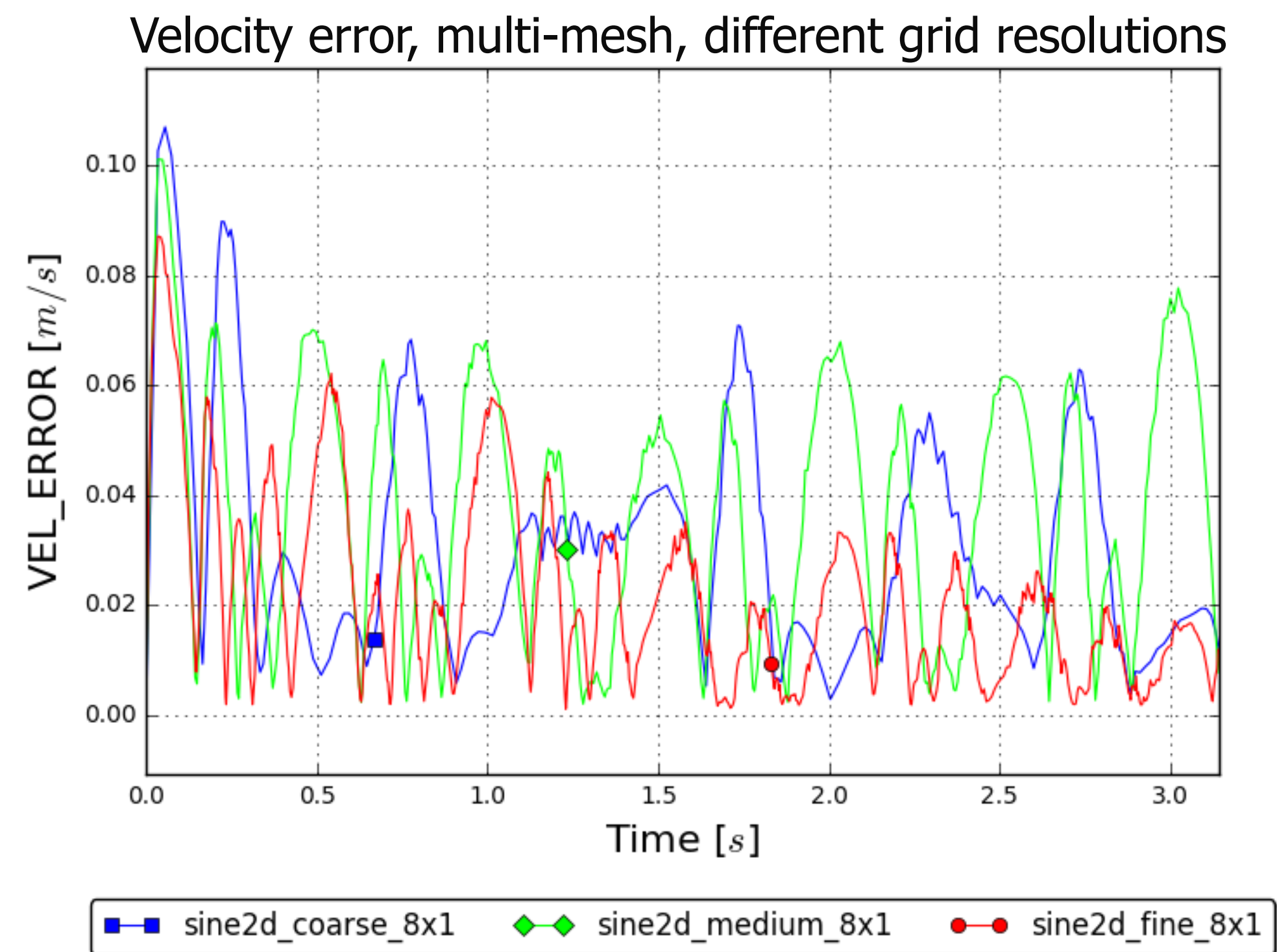
sine2d_tol-3_8x1_devc.csv

sine2d_tol-6_8x1_devc.csv

→ tolerance of 10^{-3} is already enough, no need to use 10^{-6}

sine2d: Compare different grid resolutions

`python compareCSV.py -i compareCSV.txt -o pictures/out.png VEL_ERROR`



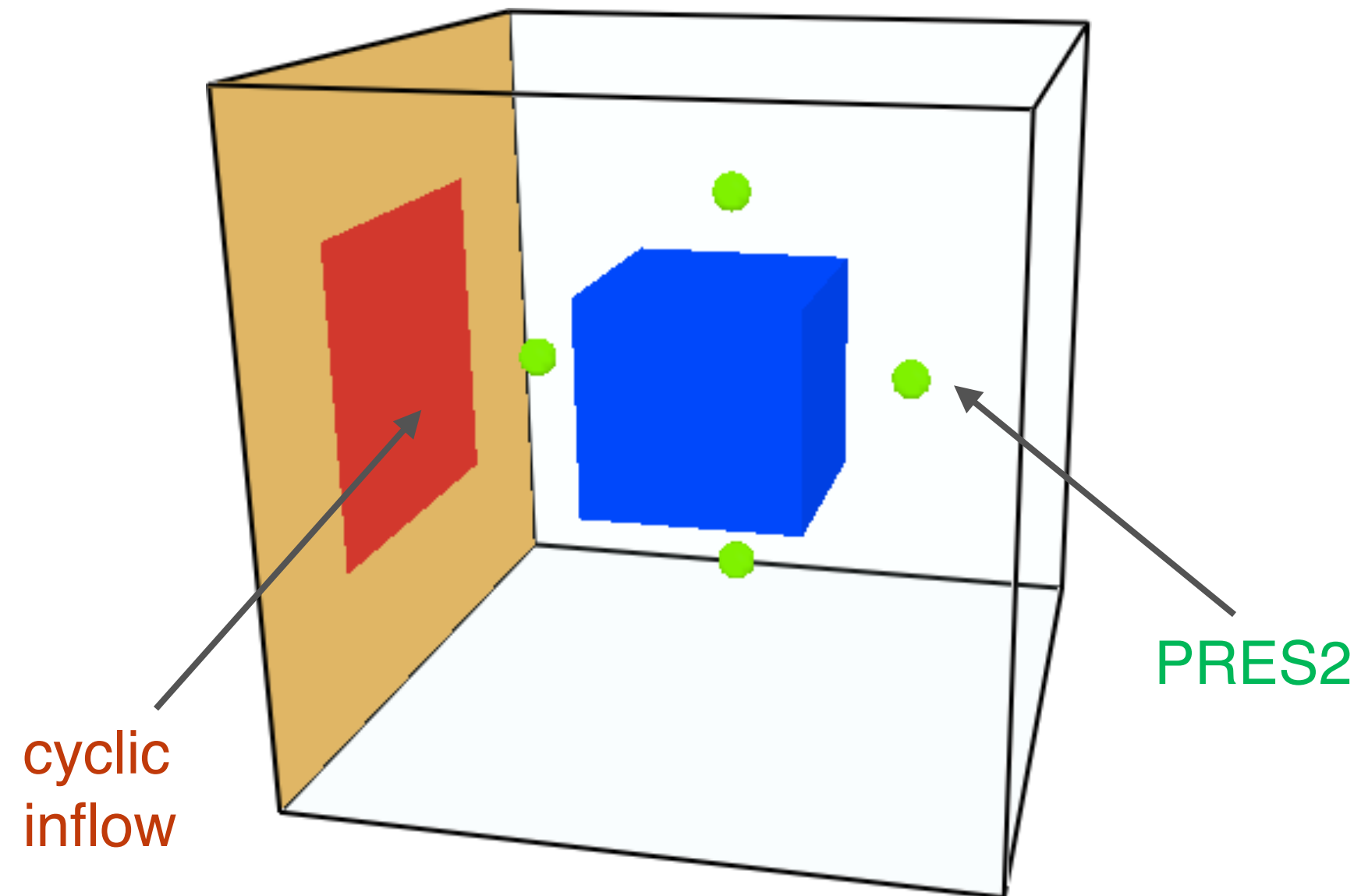
[compareCSV.txt:](#)

40x10 cells	→	sine2d_coarse_8x1_devc.csv
80x20 cells	→	sine2d_medium_8x1_devc.csv
160x40 cells	→	sine2d_fine_8x1_devc.csv

→ effects of different grid resolutions can be analyzed

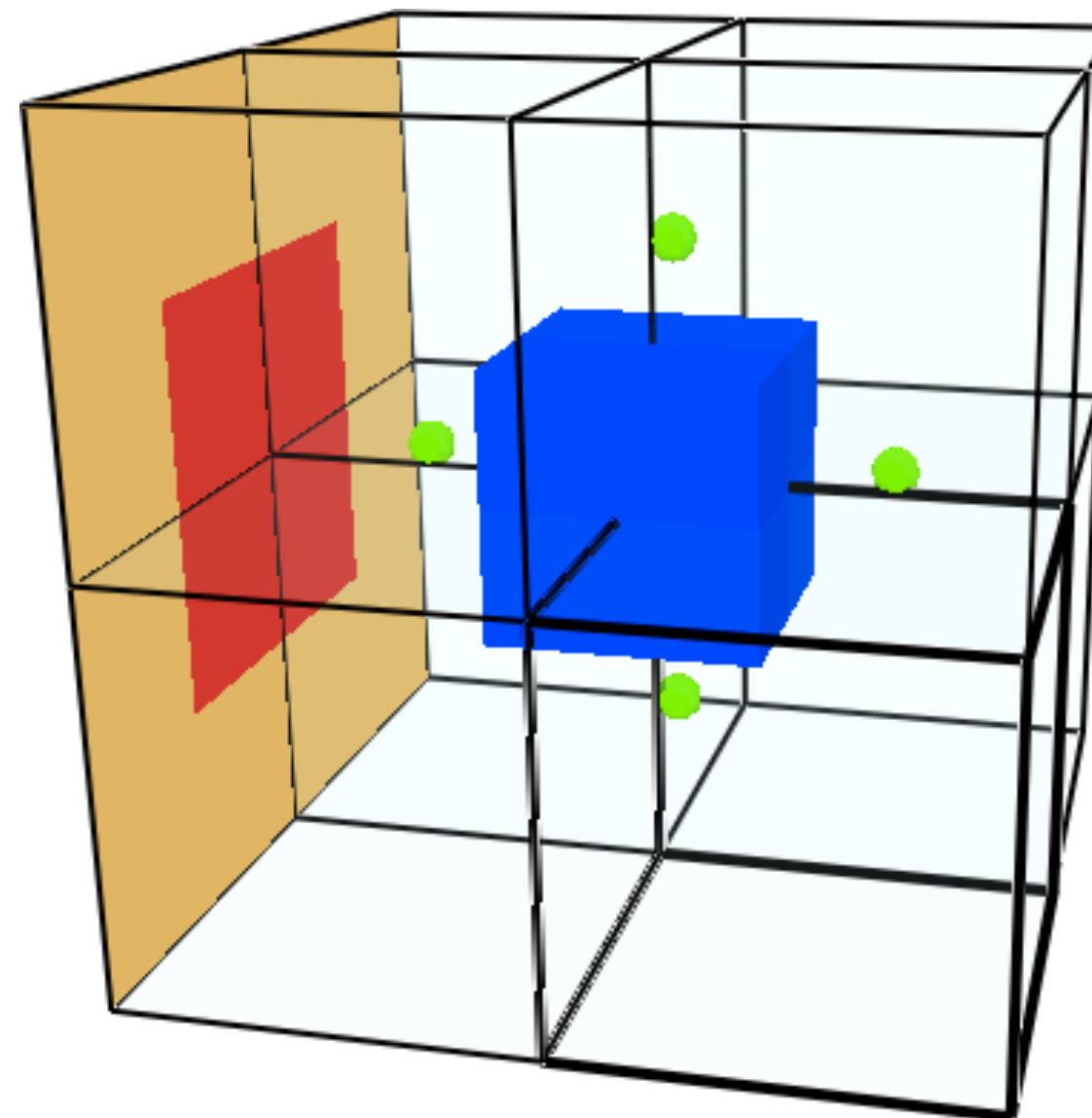
Example 2: Flow through a cube with obstruction

cube3d_1x1x1



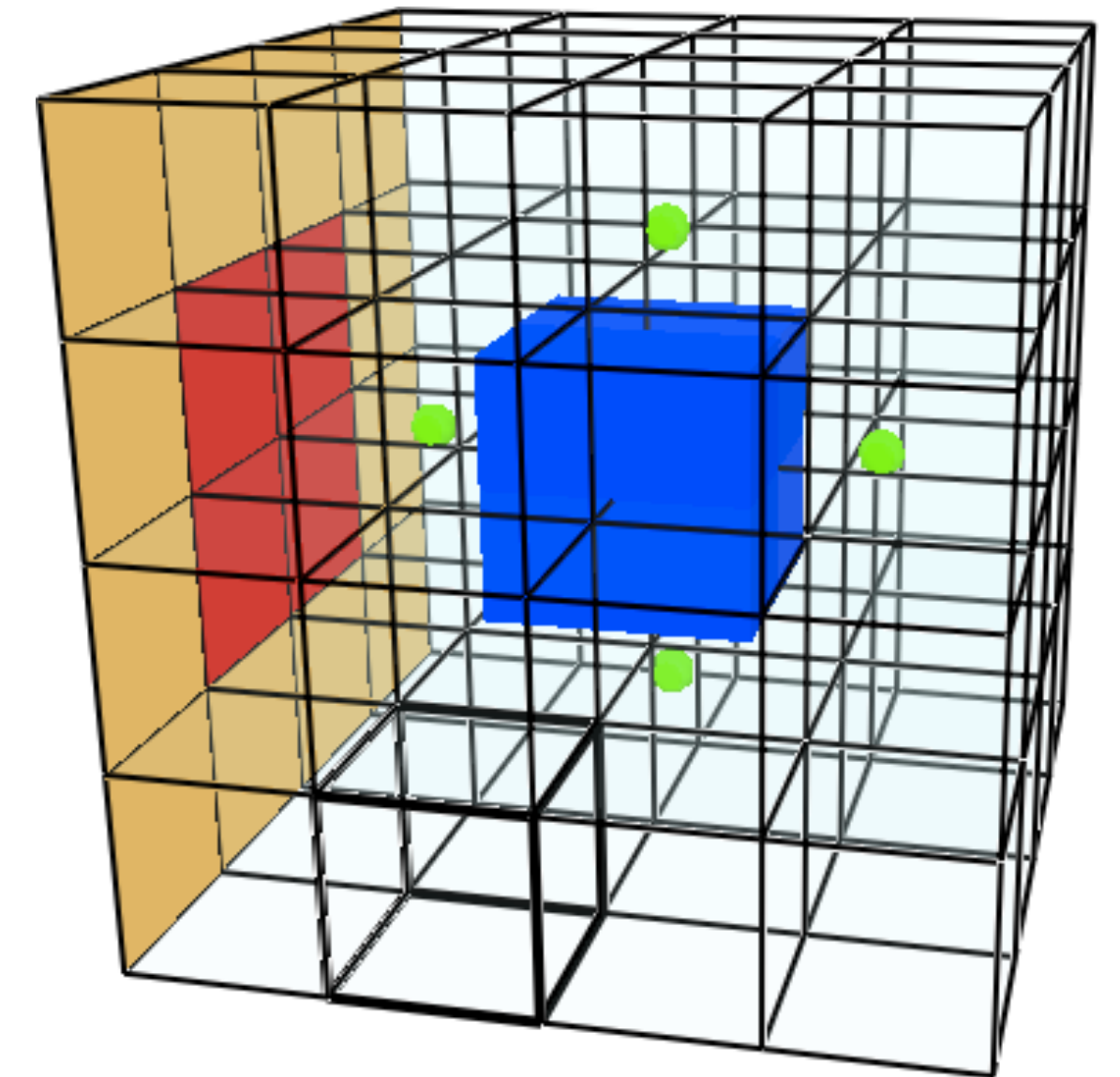
corresponds to `cube3d.fds`

cube3d_2x2x2



`python splitMesh.py cube3d.fds 2x2x2`

cube3d_4x4x4



`python splitMesh.py cube3d.fds 4x4x4`

Exercise: produce different cases with `splitMesh.py` based on `cube3d.fds`,
use 3-digit pattern in `<command>` due to 3D-case

Example 2: Flow through a cube with obstruction

`cube3d.fds` \triangleq `cube3d_1x1x1.fds`

- based on the default settings for pressure solver
- includes the following device definitions

```
&DEVC XYZ= 1.4, 0.1, 0.1, QUANTITY='PRESSURE', ID='PRES2' /
```

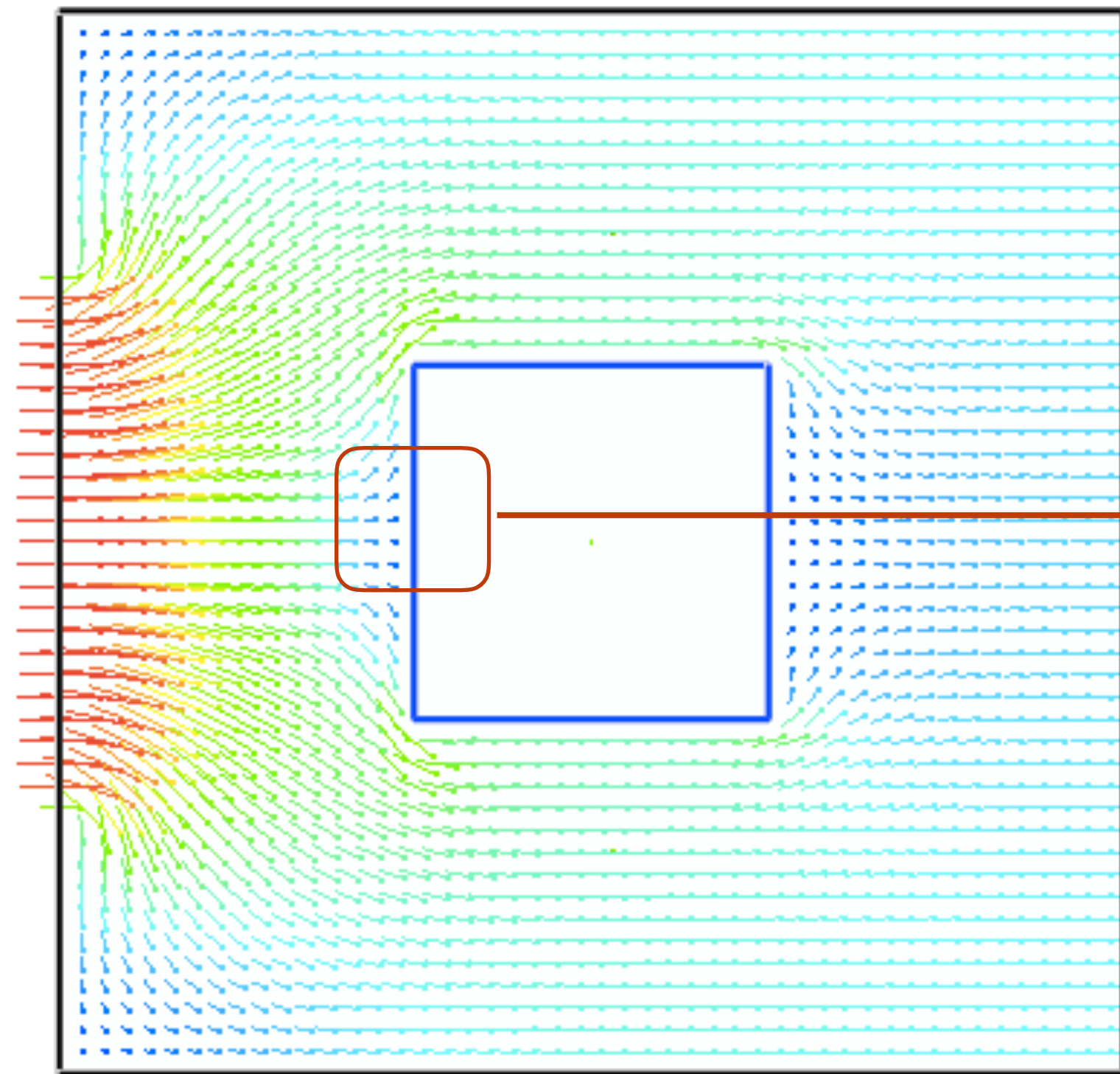
```
&DEVC XYZ= 0.0, 0.0, 0.0, QUANTITY='MAXIMUM VELOCITY ERROR', ID='VEL_ERROR' /  
&DEVC XYZ= 0.0, 0.0, 0.0, QUANTITY='PRESSURE ITERATIONS', ID='PRES_ITE' /
```

Exercise: Run the different cases

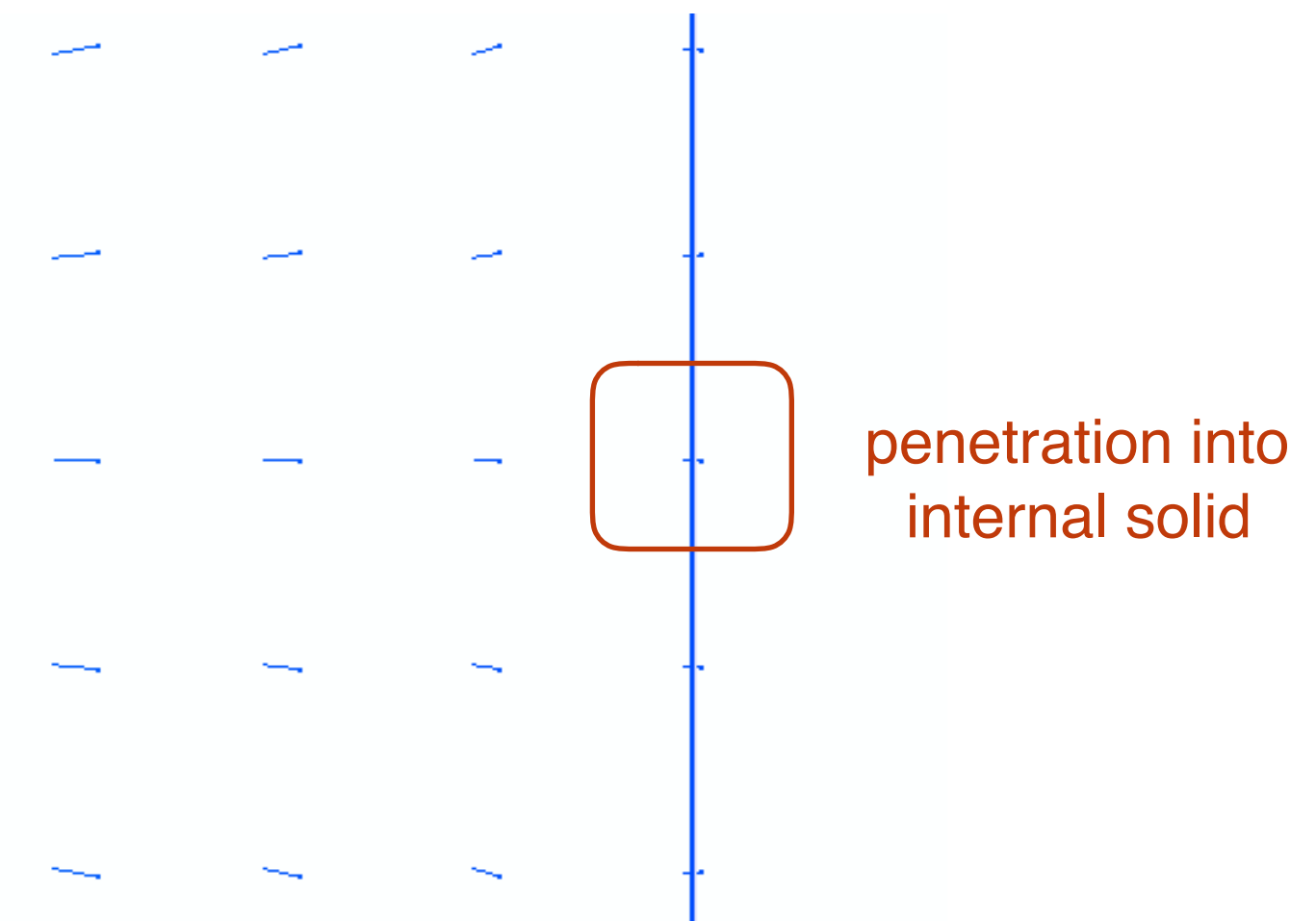
```
mpirun -np M3 fds cube3d_MxMxM.fds
```

cube3d: Velocity error along obstruction

velocity field along $y=0$



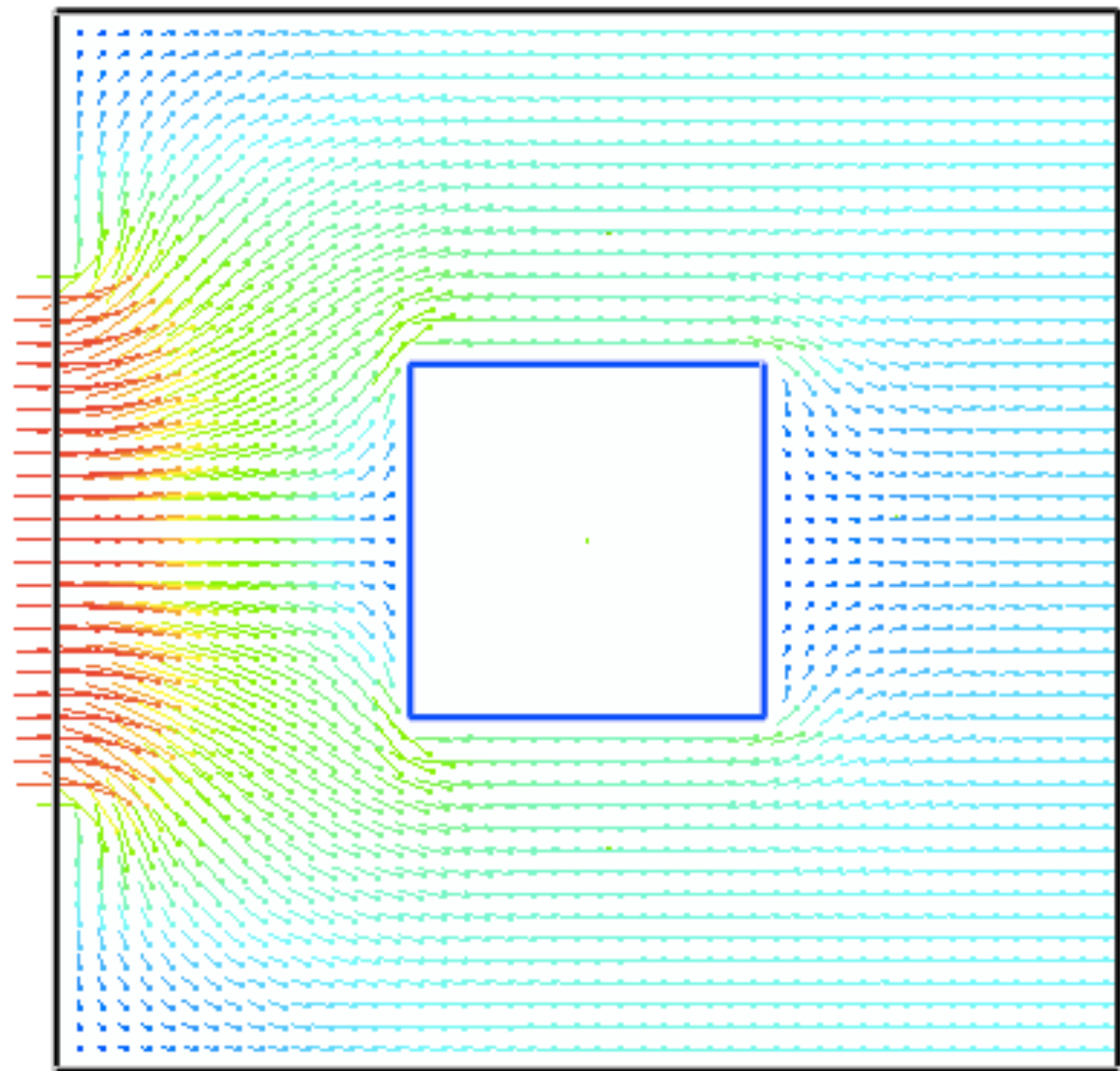
zoom of velocity at internal obstruction



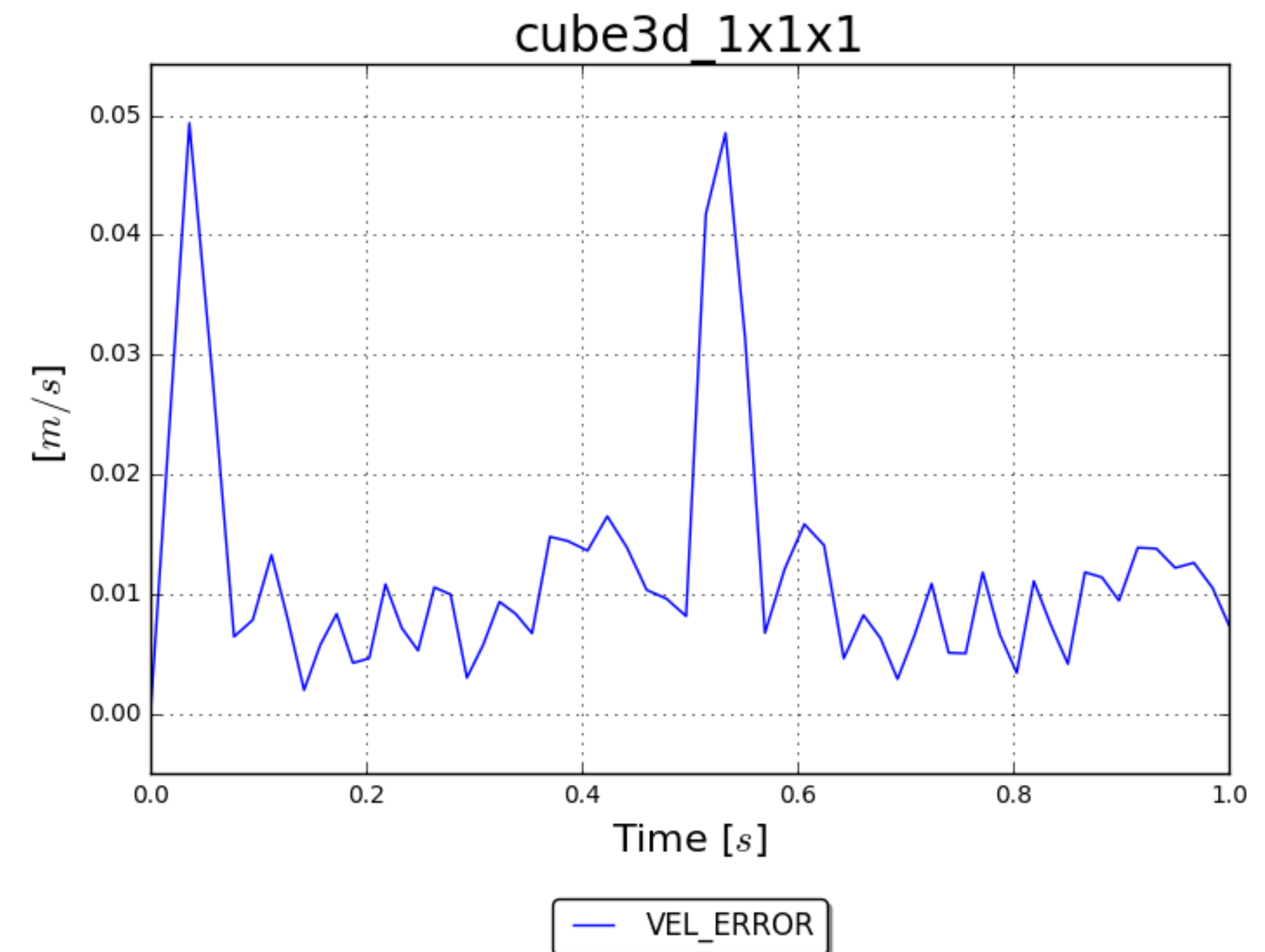
→ penetration of velocity field into internal obstruction for default tolerance

cube3d: Velocity error along obstruction, 1-mesh

velocity field along $y=0$



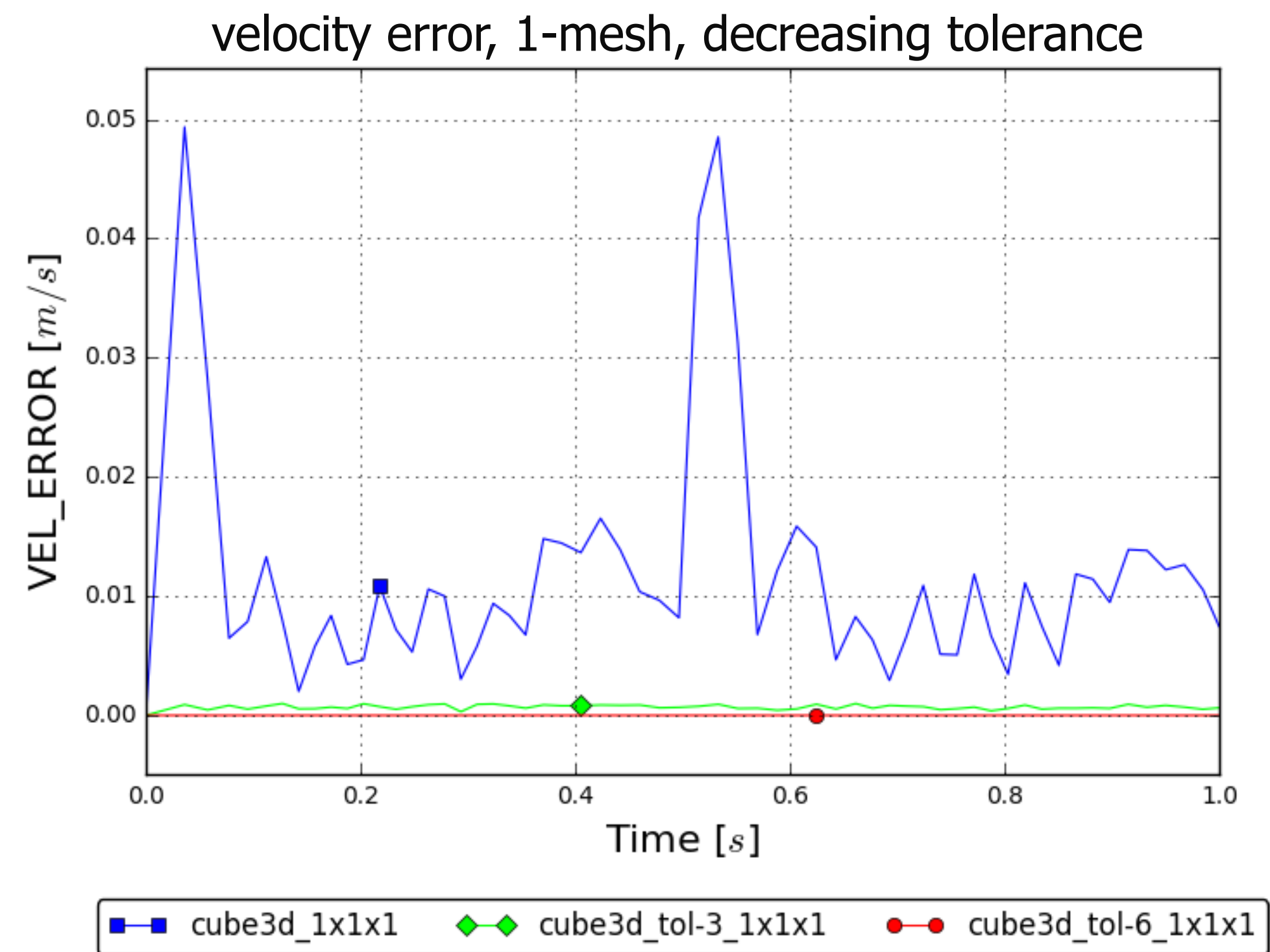
`python plotCSV.py cube3d_1x1x1_devc.csv VEL_ERROR -o pictures/out.png`



→ comprehensible velocity error for default tolerance, reflects cyclic inflow

cube3d: 1-mesh, stricter velocity tolerances

`python compareCSV.py -i compareCSV.txt -o pictures/out.png VEL_ERROR`



different
tolerances,
same
geometry

compareCSV.txt:

cube3d_1x1x1_devc.csv

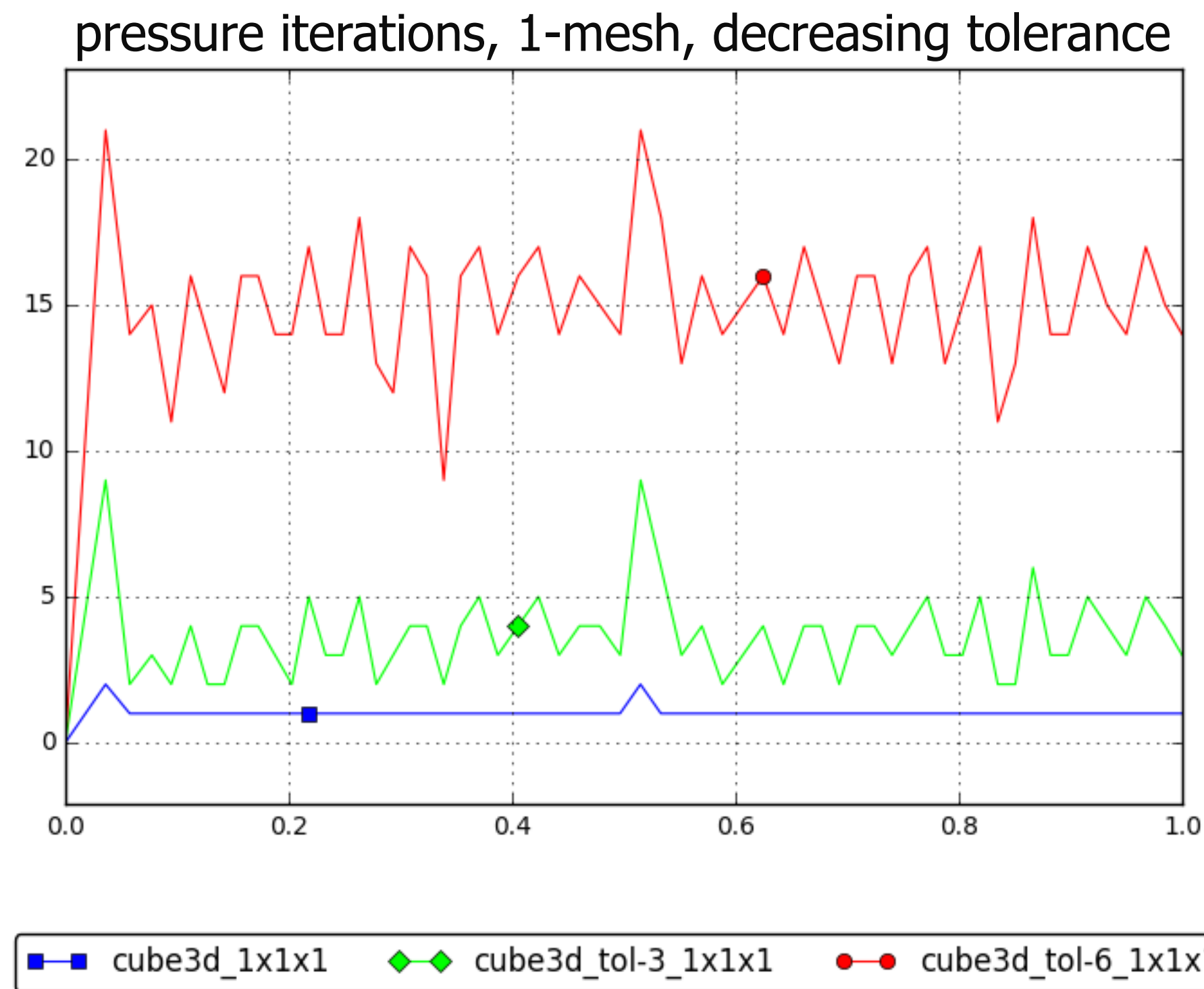
cube3d_tol-3_1x1x1_devc.csv

cube3d_tol-6_1x1x1_devc.csv

→ for 1-mesh case a tolerance of 10^{-3} is sufficient, no need to use 10^{-6}

cube3d: 1-mesh, stricter velocity tolerances

`python compareCSV.py -i compareCSV.txt -o pictures/out.png PRES_ITE`



different
tolerances,
same
geometry



[compareCSV.txt:](#)

`cube3d_1x1x1_devc.csv`

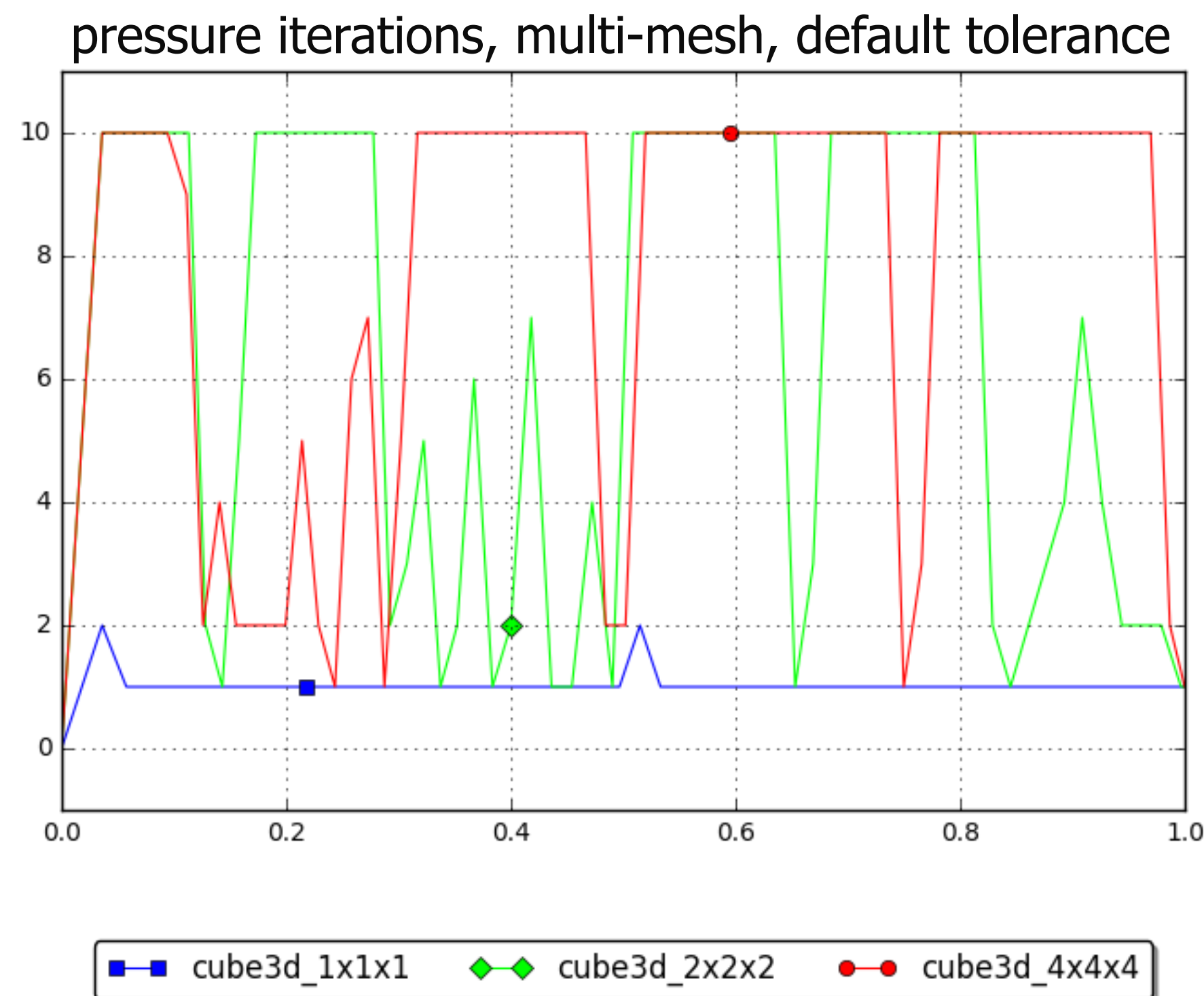
`cube3d_tol-3_1x1x1_devc.csv`

`cube3d_tol-6_1x1x1_devc.csv`

→ number of pressure iterations increases due to stricter tolerances

cube3d: Multi-mesh, default velocity tolerance

`python compareCSV.py -i compareCSV.txt -o pictures/out.png PRES_ITE`



default
tolerance,
different
geometries

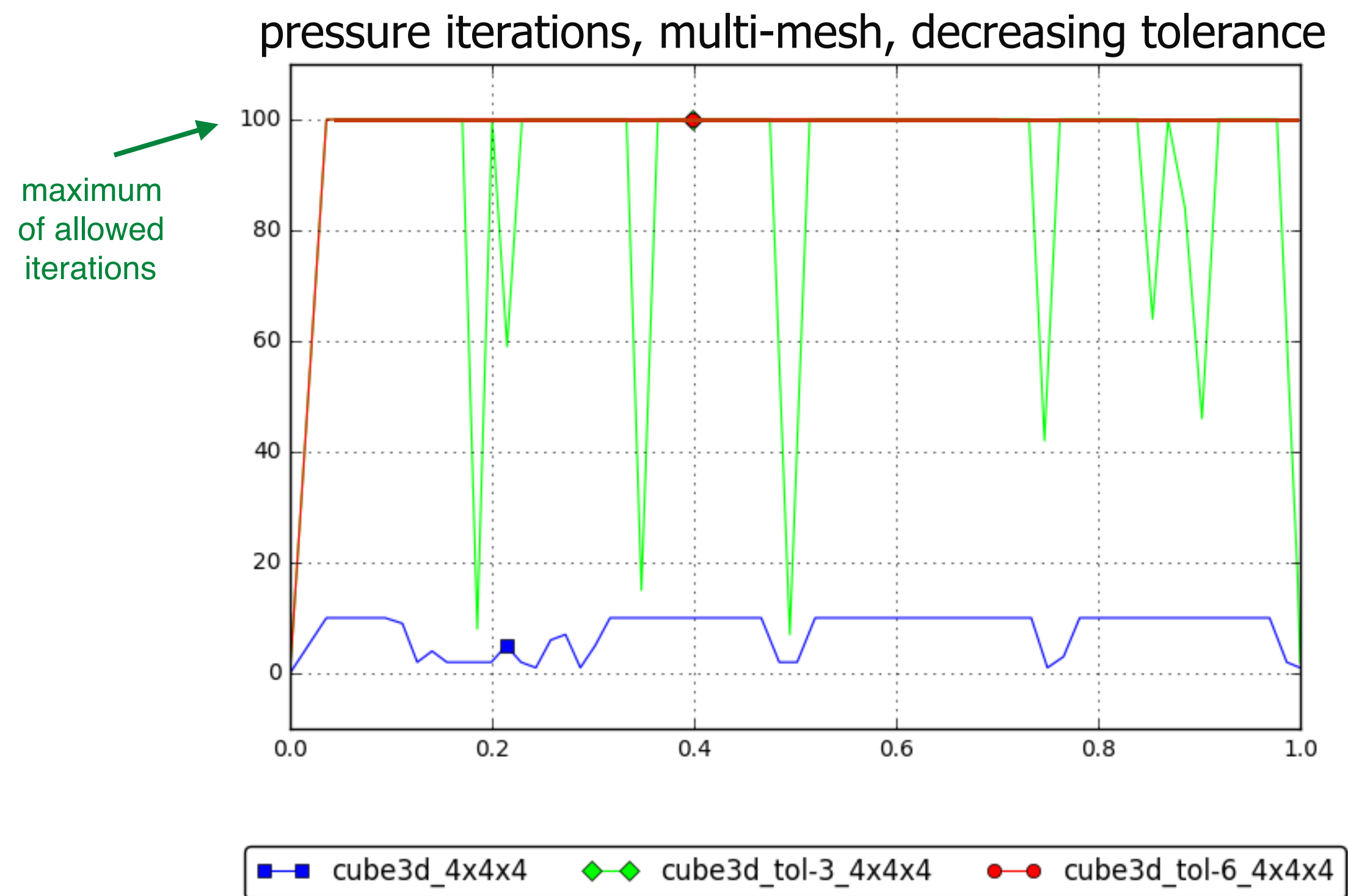
[compareCSV.txt:](#)

cube3d_1x1x1_devc.csv
cube3d_2x2x2_devc.csv
cube3d_4x4x4_devc.csv

→ slightly increased due to multi-mesh decomposition for default tolerance

cube3d: Multi-mesh, stricter velocity tolerances

```
python compareCSV.py -i compareCSV.txt -o pictures/out.png PRES_ITE
```



compareCSV.txt:

cube3d_4x4x4_devc.csv

cube3d_tol-3_4x4x4_devc.csv

cube3d_tol-6_4x4x4_devc.csv

→ big increase due to multi-mesh decomposition and decreasing tolerance

Typical procedure for case studies

- identify (rectangular) base geometry
 - identify devices of interest
 - identify parameters to vary (e.g. velocity tolerance, mesh decompositions)
 - generate correspondingly modified geometry files
-
- use [splitMesh.py](#) to generate different subdivisions (stored in [input](#)-folder)
 - run different cases in parallel
 - use [plotCSV.py](#) to evaluate one or more quantities of one selected simulation
 - use [compareCSV.py](#) to compare one common quantity for a set of simulations
(keep one parameter fixed, change another one, e.g. fix geometry, decreasing tolerances)