
# News on pre-processing tool and boundary conditions

JOSÉ M. DOMÍNGUEZ

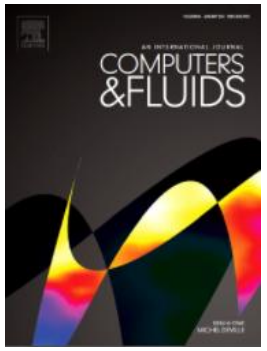
**UNIVERSIDADE DE VIGO**

# OUTLINE

- **Introduction**
  - **New boundary conditions: mDBC**
    - Motivation: DBC drawbacks
    - Fluid properties from ghost nodes
    - mDBC vs. DBC
    - mDBC requirements
      - **New options on Preprocessing**
        - Free drawing mode (without lattice limitation)
        - Automatic multi-layer drawing
        - XML programming style
        - Automatic calculation of normals (for mDBC)
          - **Conclusions & future improvements**

# Main improvements in SPH formulation of DualSPHysics v5

## ✓ Density Diffusion Term by G. Fourtakas




G. Fourtakas, J.M. Domínguez, R. Vacondio, B.D. Rogers. 2019. **Local Uniform Stencil (LUST) boundary condition for arbitrary 3-D boundaries in parallel smoothed particle hydrodynamics (SPH) models.** Computers & Fluids, 190: 346-361. doi.org/10.1016/j.compfluid.2019.06.009.

## ✓ Modified Dynamic Boundary Conditions by A. English


A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

# MOTIVATION: DBC DRAWBACKS

## Dynamic Boundary Conditions (DBC)

- DBC are simple, fast, reliable and very versatile.
- Allow very complex geometries in 2-D and 3-D simulations.
- Used in a large number of works. *Crespo et al., 2007* *with 249 citations!*

## However…

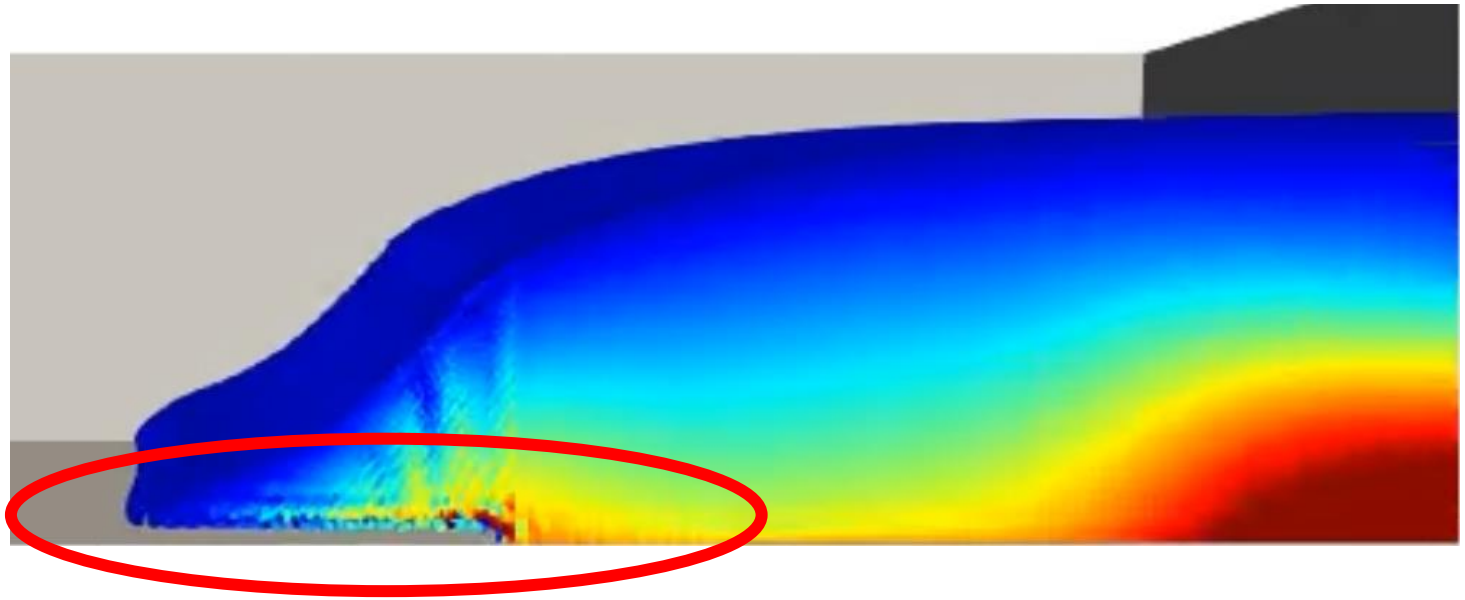
# MOTIVATION: DBC DRAWBACKS

## Dynamic Boundary Conditions (DBC)

- DBC are simple, fast, reliable and very versatile.
- Allow very complex geometries in 2-D and 3-D simulations.
- Used in a large number of works. *Crespo et al., 2007 with 249 citations!*

## However…

- **Gap between boundaries and fluid.**



**Gap in the advance of a 3-D dam break**

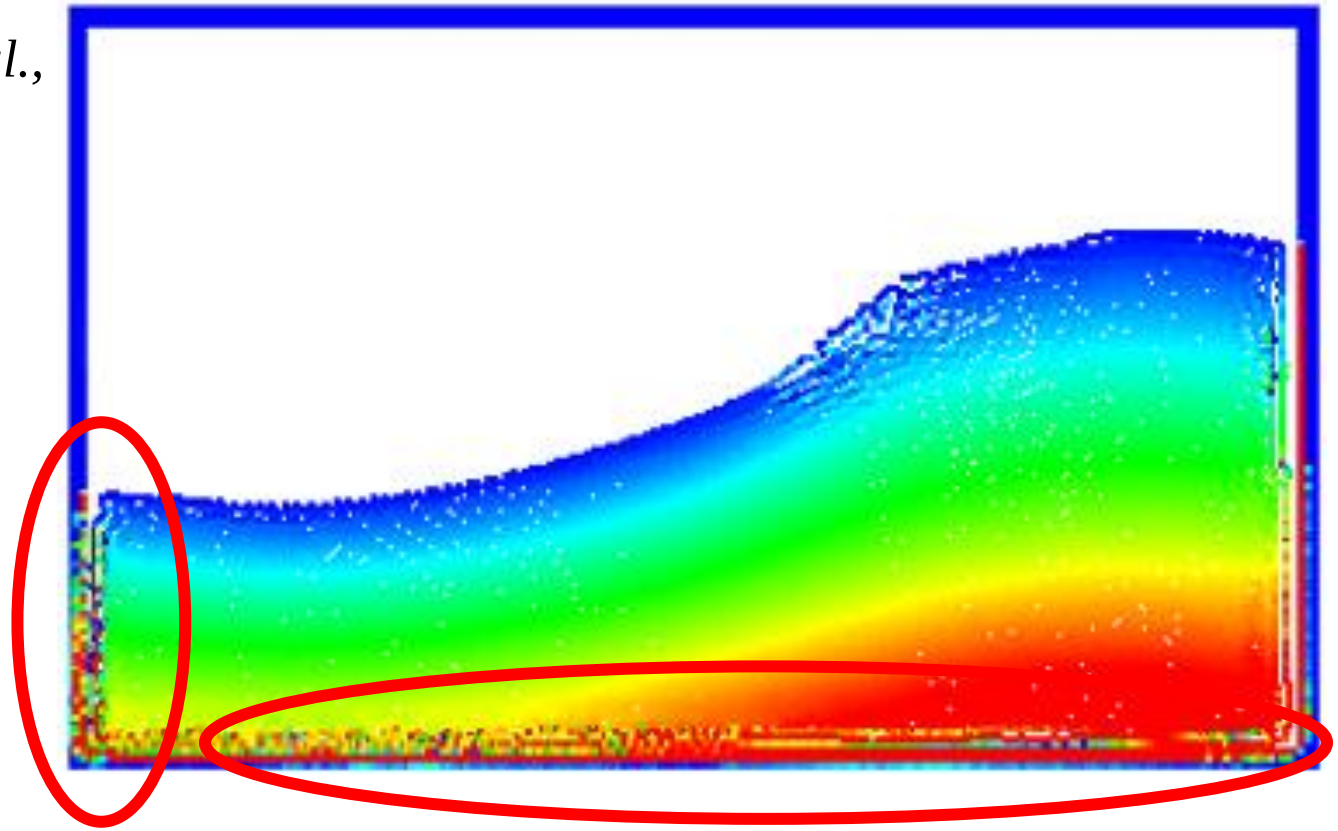
# MOTIVATION: DBC DRAWBACKS

## Dynamic Boundary Conditions (DBC)

- DBC are simple, fast, reliable and very versatile.
- Allow very complex geometries in 2-D and 3-D simulations.
- Used in a large number of works. *Crespo et al.*,

## However…

- Gap between boundaries and fluid.
- **Non-physical density on the boundaries.**



**Wrong density on boundaries**

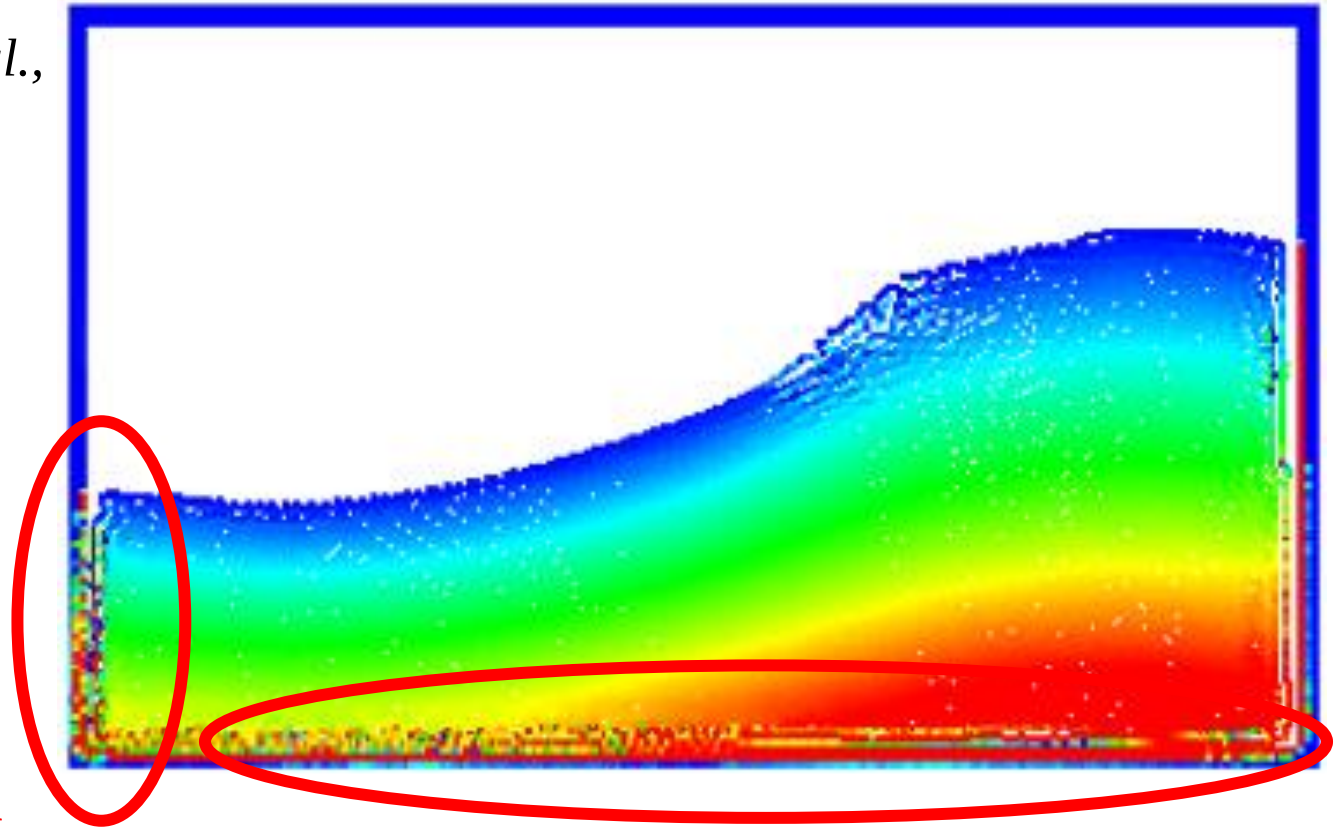
# MOTIVATION: DBC DRAWBACKS

## Dynamic Boundary Conditions (DBC)

- DBC are simple, fast, reliable and very versatile.
- Allow very complex geometries in 2-D and 3-D simulations.
- Used in a large number of works. *Crespo et al.*,

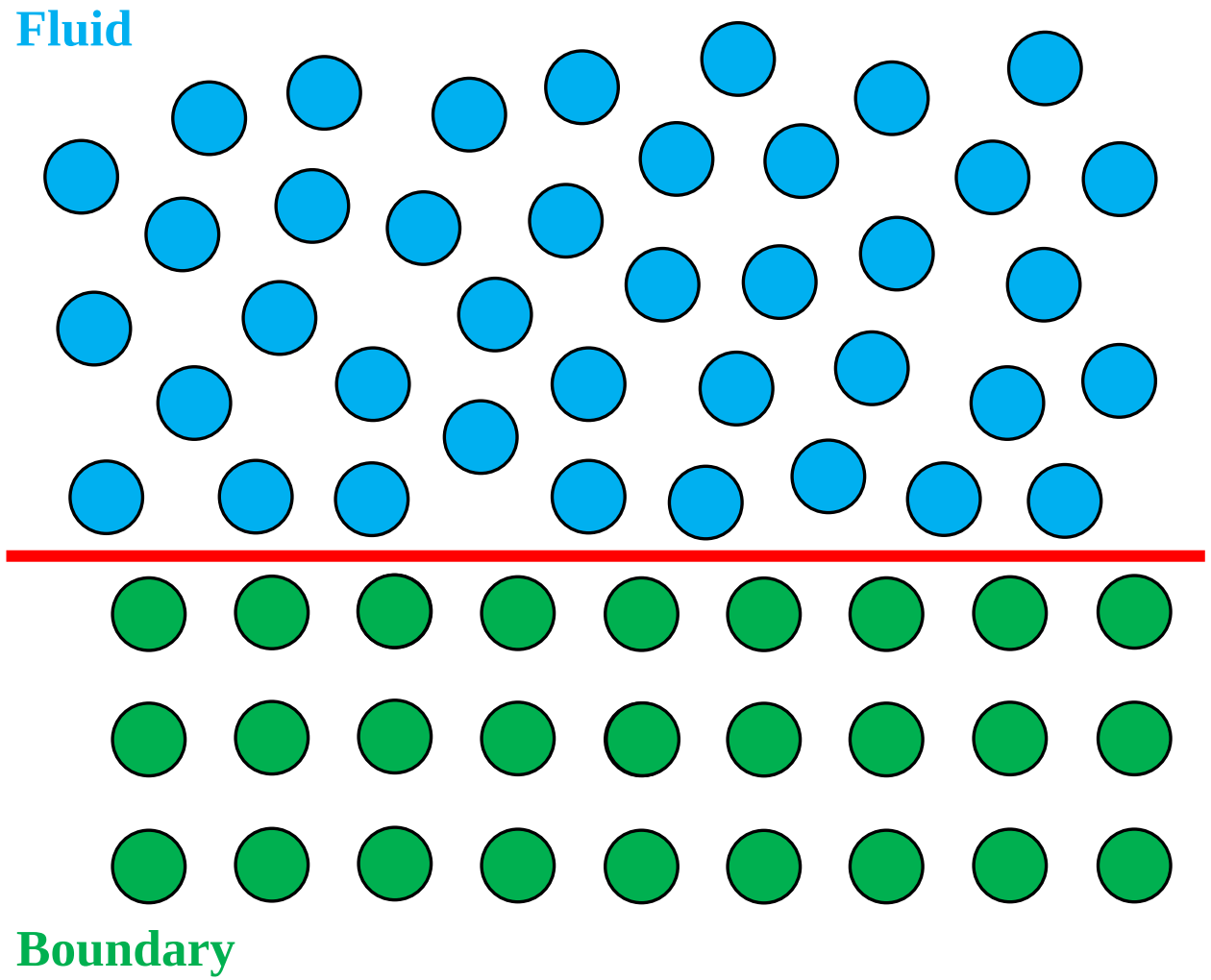
## However…

- Gap between boundaries and fluid.
- Non-physical density on the boundaries.
- **Pressure measurement is very noisy close to the boundaries.**



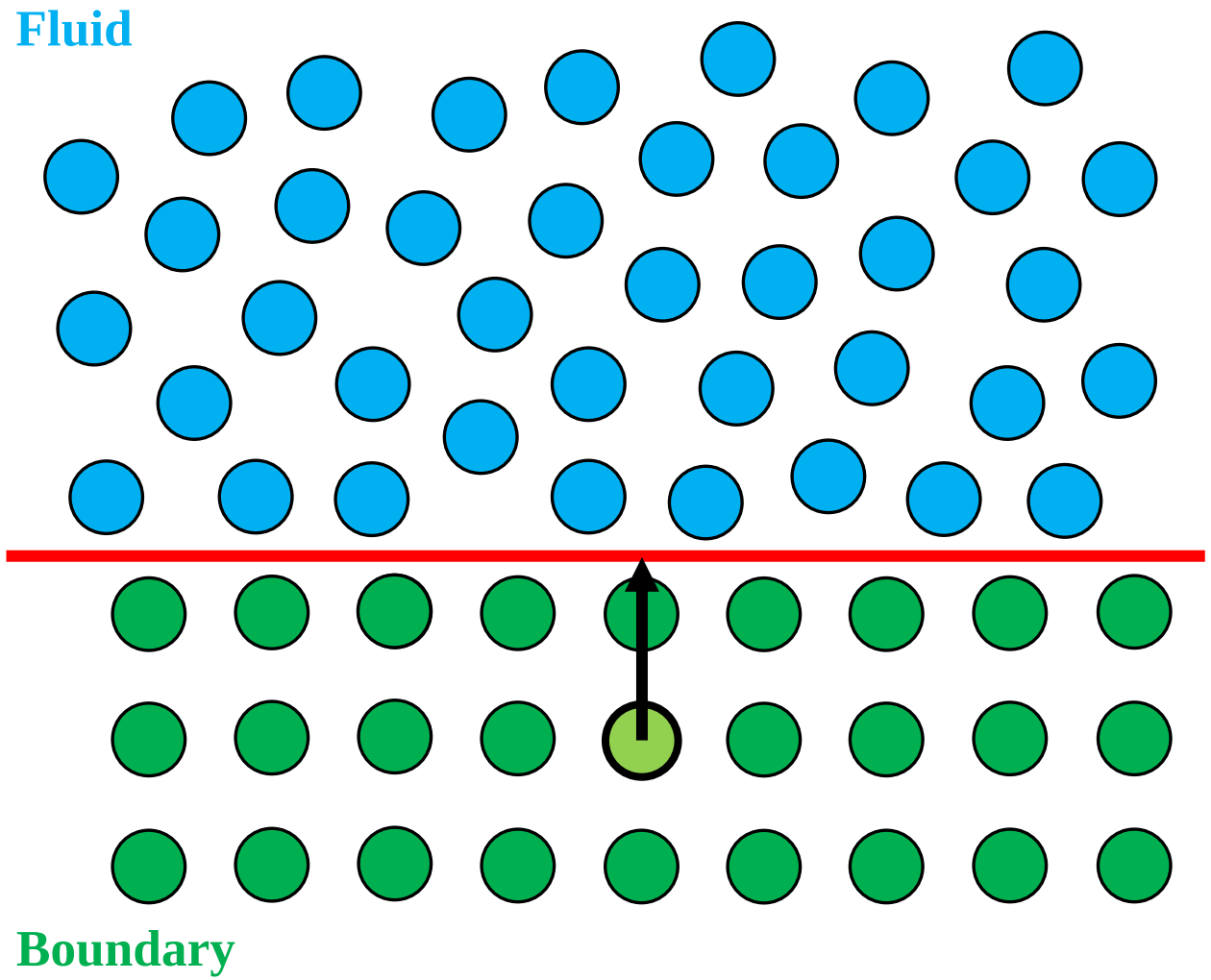
Wrong density on boundaries
**generates noise density in fluid**

# mDBC: Fluid properties from ghost nodes




Use a mirroring procedure based on the approach by Marrone et al. (2011) to mirror ghost nodes into the fluid region.

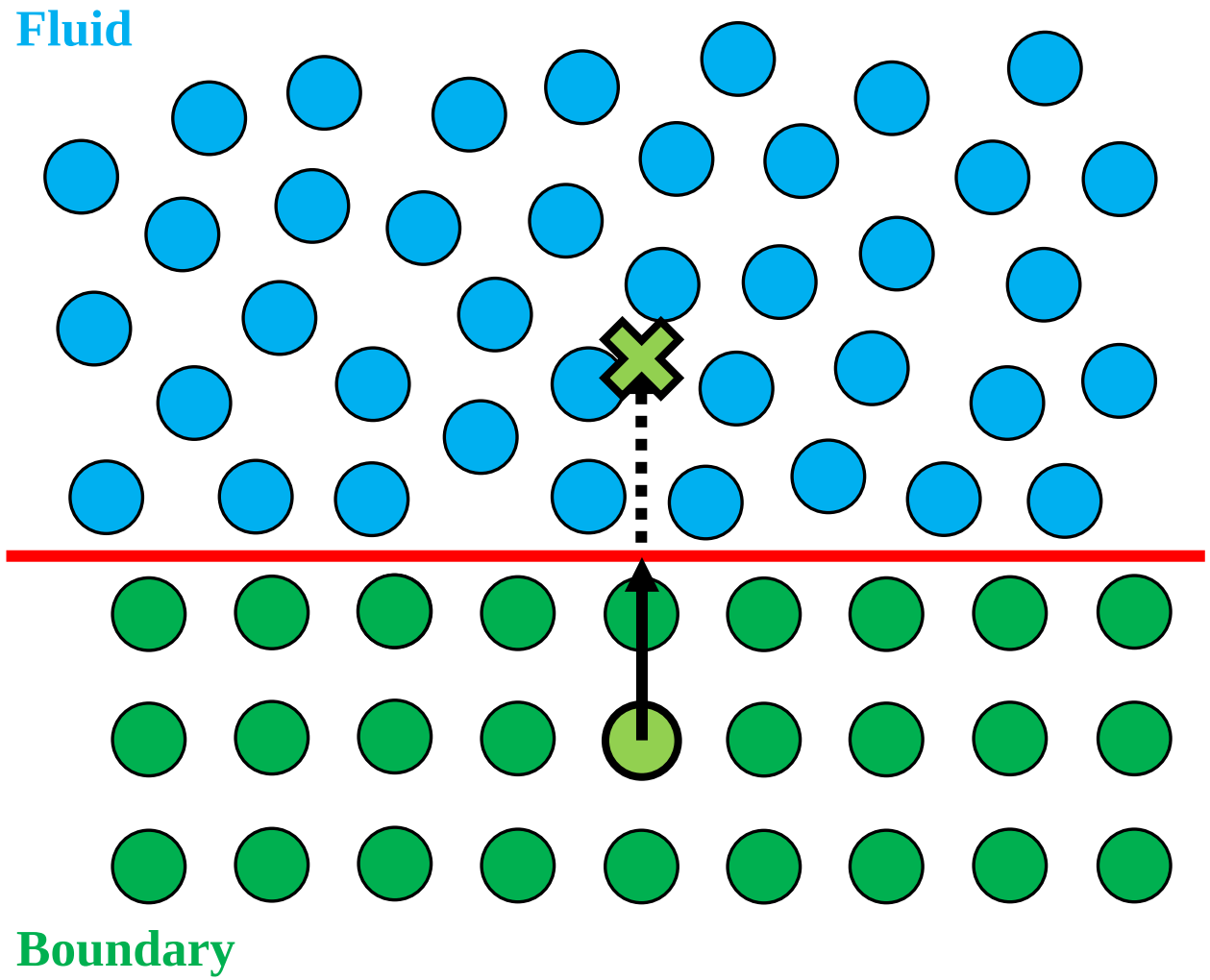
# mDBC: Fluid properties from ghost nodes




Use a mirroring procedure based on the approach by Marrone et al. (2011) to mirror ghost nodes into the fluid region.

Each **boundary particle** has a **normal vector** to **boundary limit**.

# mDBC: Fluid properties from ghost nodes

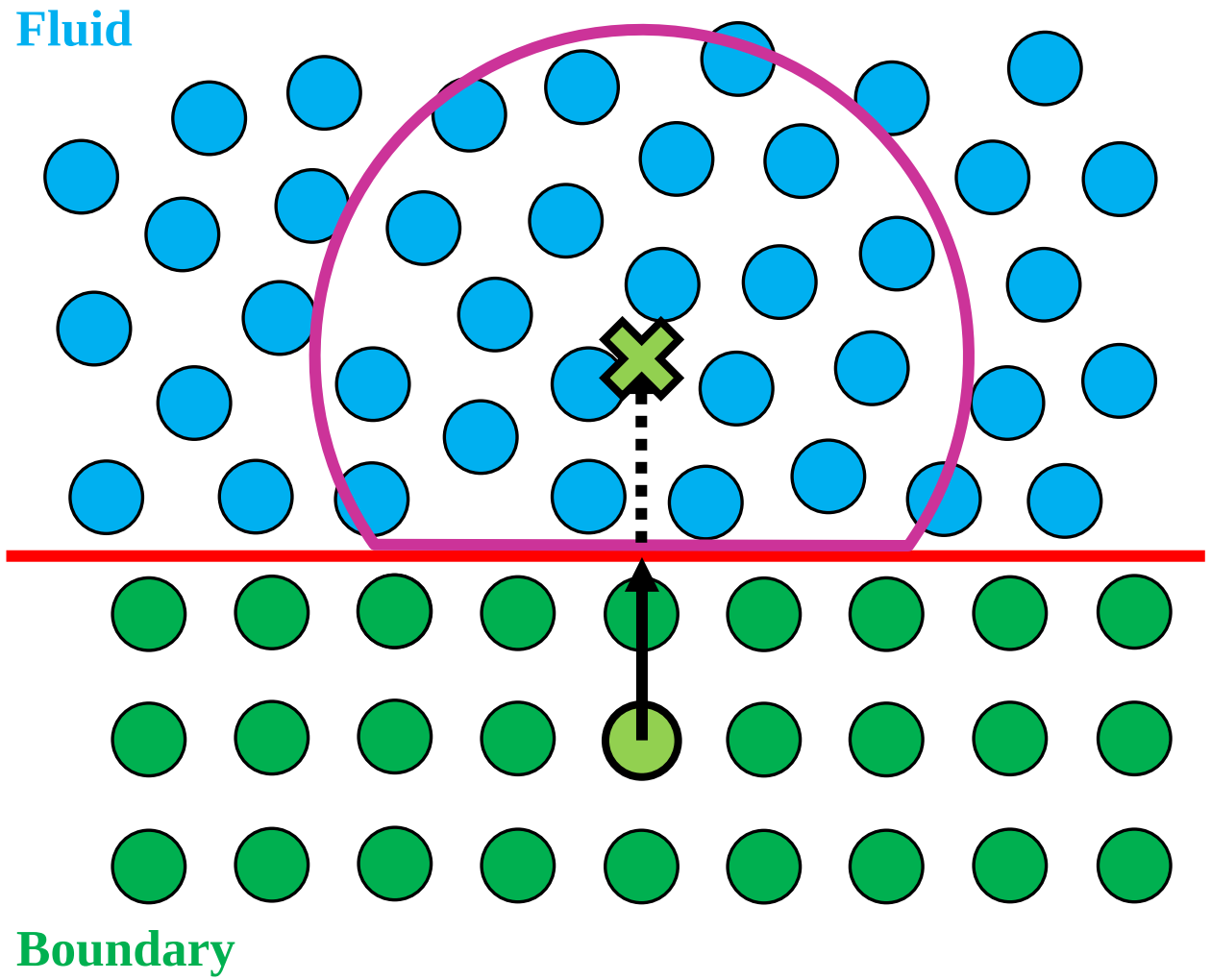



Use a mirroring procedure based on the approach by Marrone et al. (2011) to mirror ghost nodes into the fluid region.

Each **boundary particle** has a **normal vector** to **boundary limit**.

**Ghost node (X)** is projected into the fluid across a **boundary limit**.

# mDBC: Fluid properties from ghost nodes



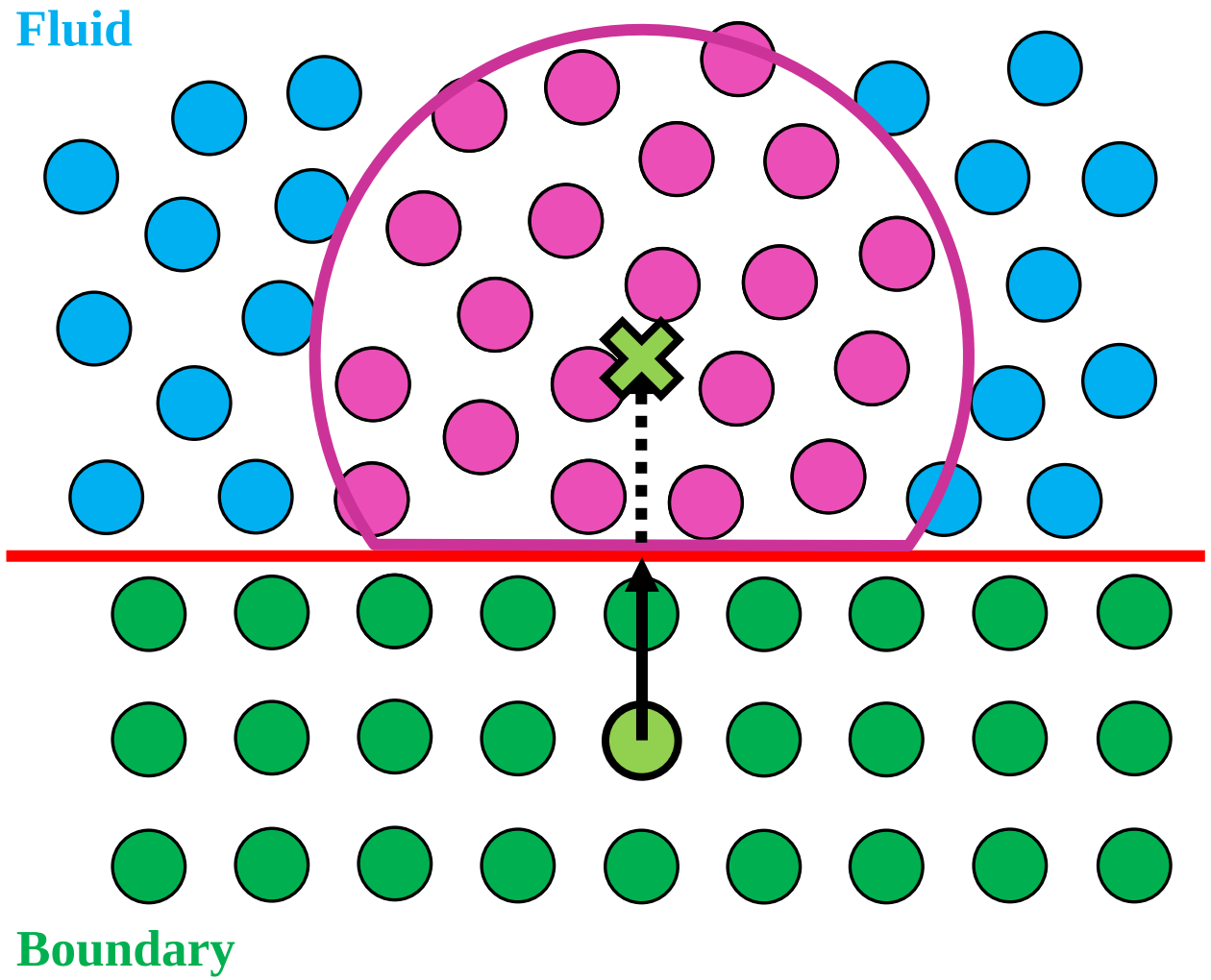

Use a mirroring procedure based on the approach by Marrone et al. (2011) to mirror ghost nodes into the fluid region.

Each **boundary particle** has a **normal vector** to **boundary limit**.

**Ghost node (X)** is projected into the fluid across a **boundary limit**.

Fluid properties are then computed at the ghost nodes according to the **surrounding fluid** using a corrected SPH approximation proposed by Liu and Liu (2006).

# mDBC: Fluid properties from ghost nodes



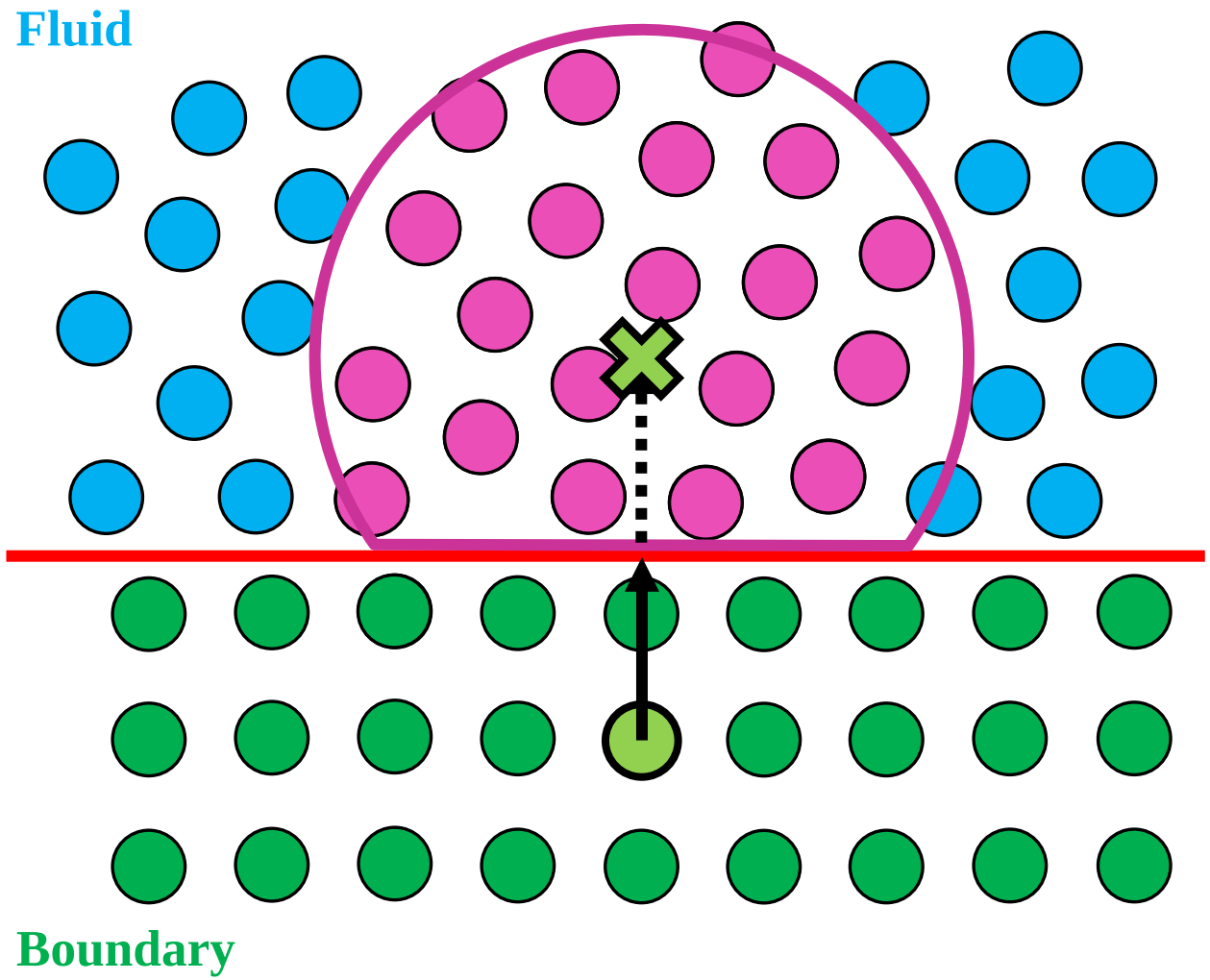

Use a mirroring procedure based on the approach by Marrone et al. (2011) to mirror ghost nodes into the fluid region.

Each **boundary particle** has a **normal vector** to **boundary limit**.

**Ghost node (X)** is projected into the fluid across a **boundary limit**.

Fluid properties are then computed at the ghost nodes according to the **surrounding fluid** using a corrected SPH approximation proposed by Liu and Liu (2006).

# mDBC: Fluid properties from ghost nodes




Use a mirroring procedure based on the approach by Marrone et al. (2011) to mirror ghost nodes into the fluid region.

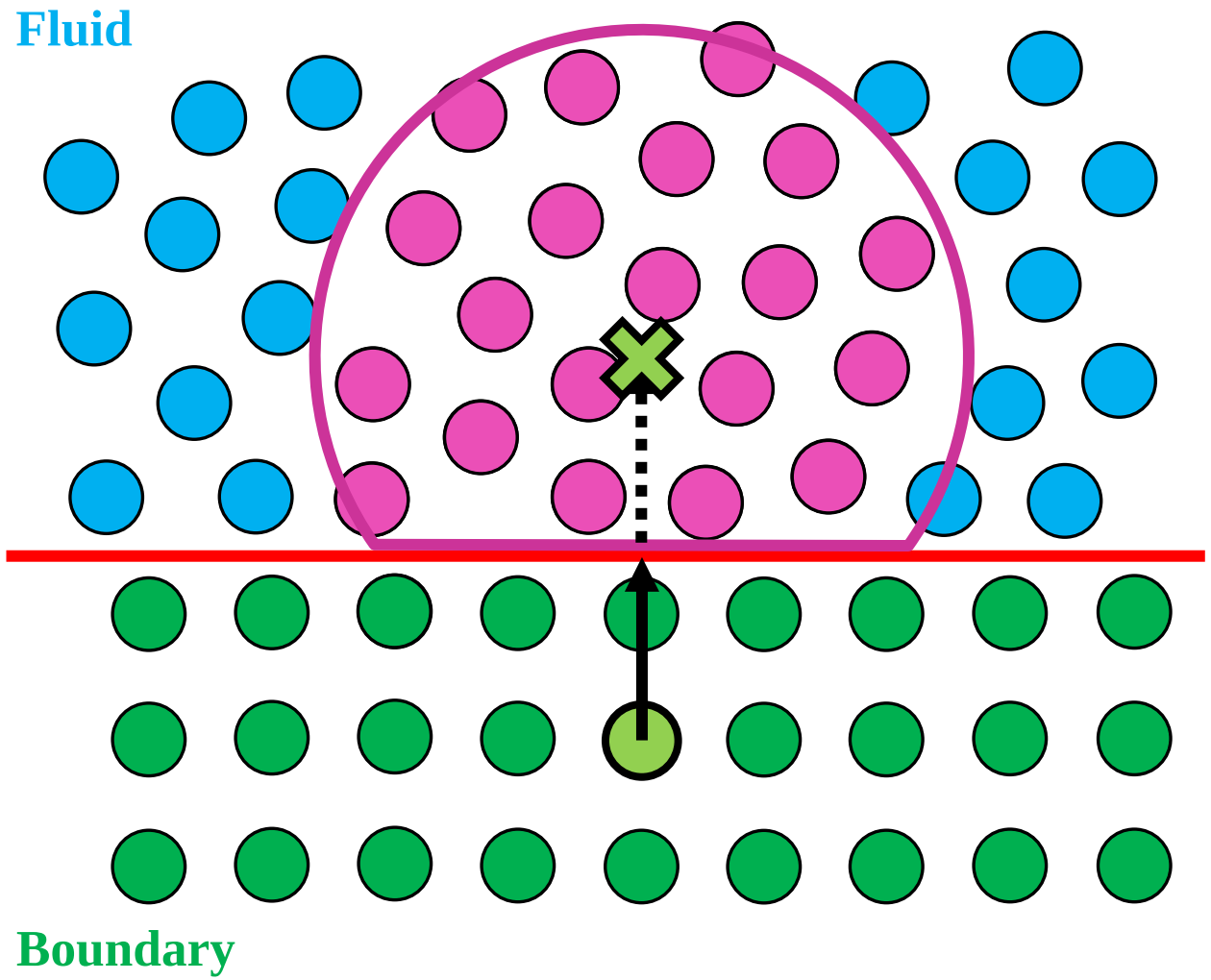
Each **boundary particle** has a **normal vector** to **boundary limit**.

**Ghost node (X)** is projected into the fluid across a **boundary limit**.

Fluid properties are then computed at the ghost nodes according to the **surrounding fluid** using a corrected SPH approximation proposed by Liu and Liu (2006).

And finally, fluid properties (density and reversed velocity) are mirrored back to **boundary particles**.

# mDBC: Fluid properties from ghost nodes




Use a mirroring procedure based on the approach by Marrone et al. (2011) to mirror ghost nodes into the fluid region.

Each **boundary particle** has a **normal vector** to **boundary limit**.

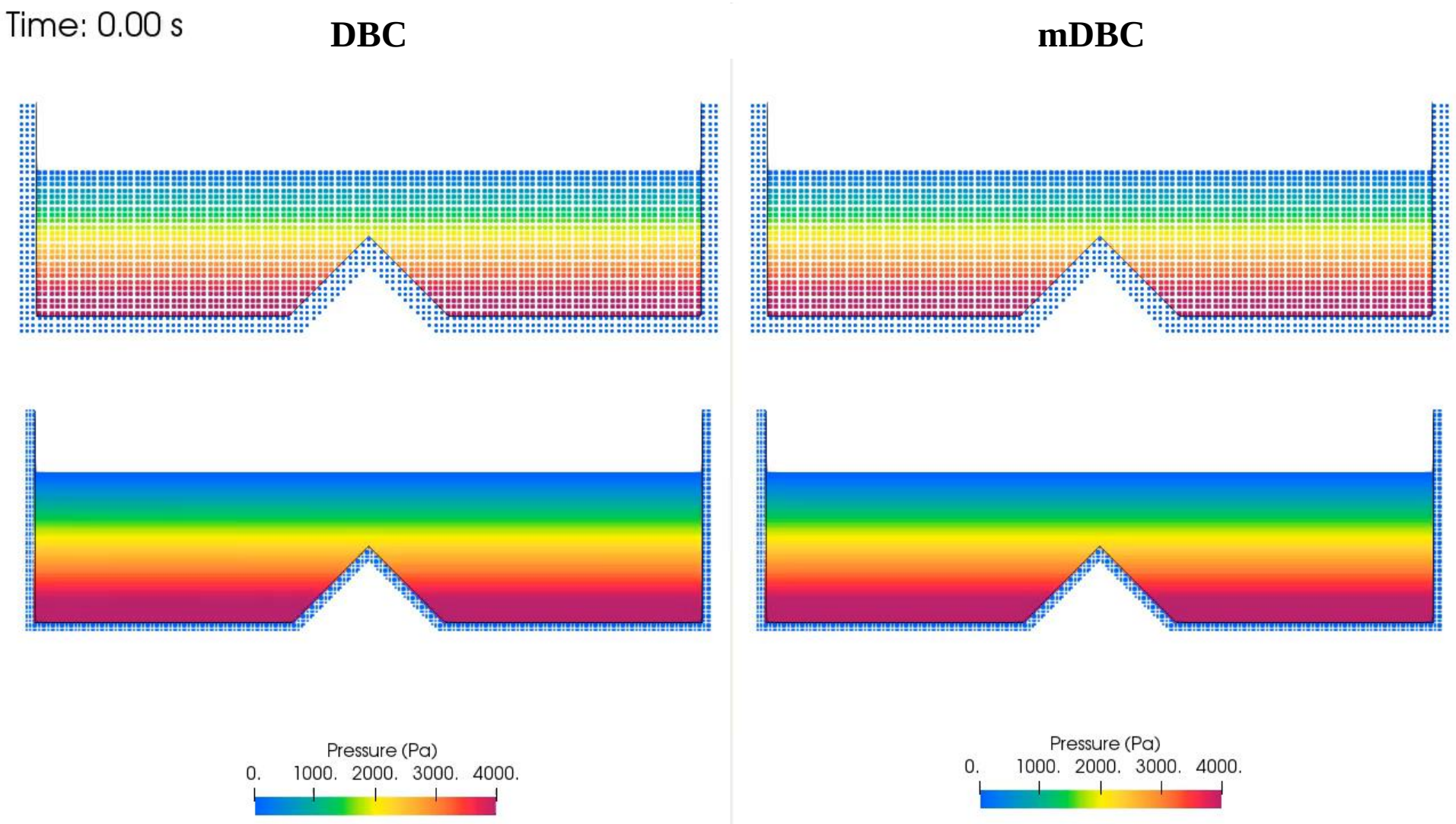
**Ghost node (X)** is projected into the fluid across a **boundary limit**.

Fluid properties are then computed at the ghost nodes according to the **surrounding fluid** using a corrected SPH approximation proposed by Liu and Liu (2006).

And finally, fluid properties (density and reversed velocity) are mirrored back to **boundary particles**.

**mDBC is a correction applied over DBC** when boundary particle has a non-zero normal. Thus it is possible to combine DBC with mDBC.

# DBC vs. mDBC: Still water with a wedge



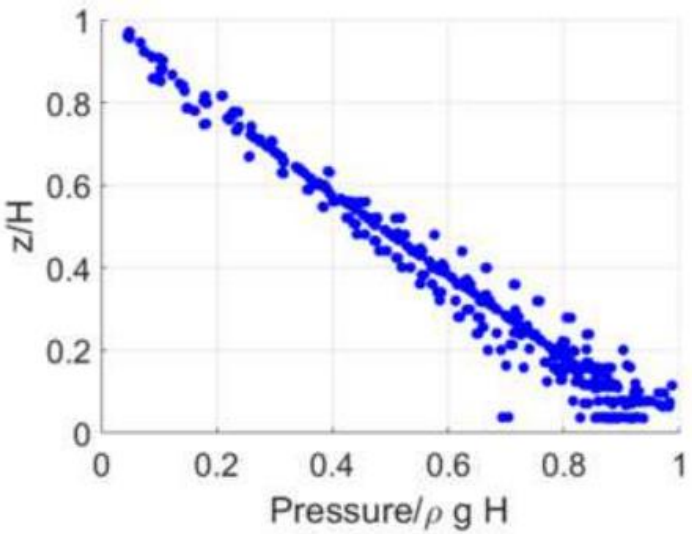
# DBC vs. mDBC: Still water with a wedge

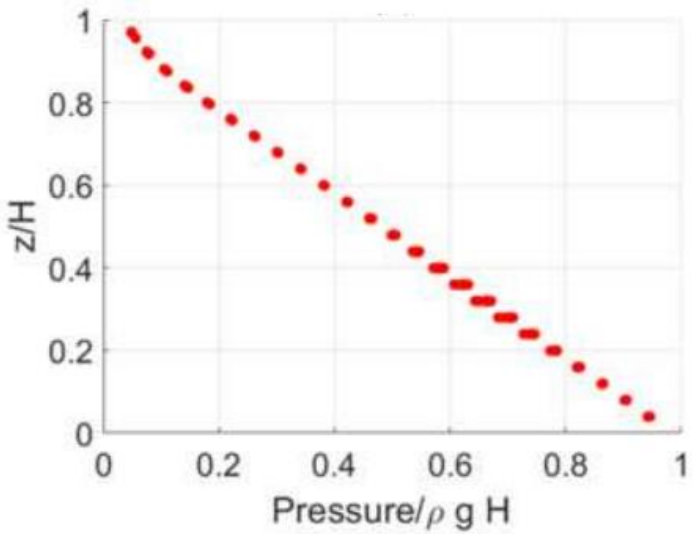
A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

**Pressure vs. depth – DBC**

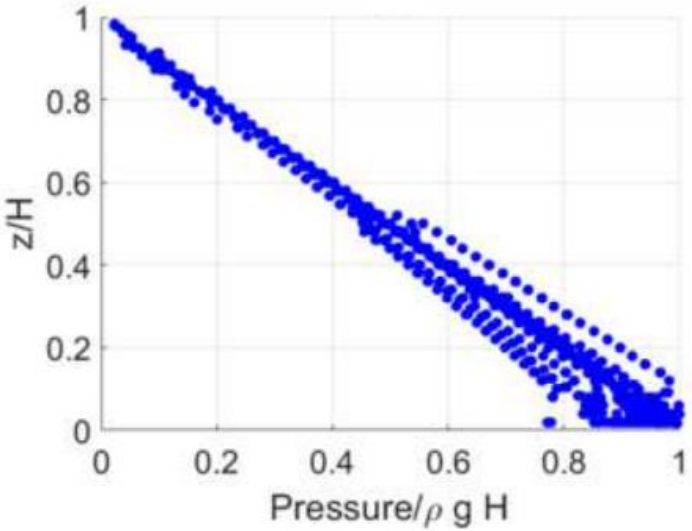
**Pressure vs. depth - mDBC**

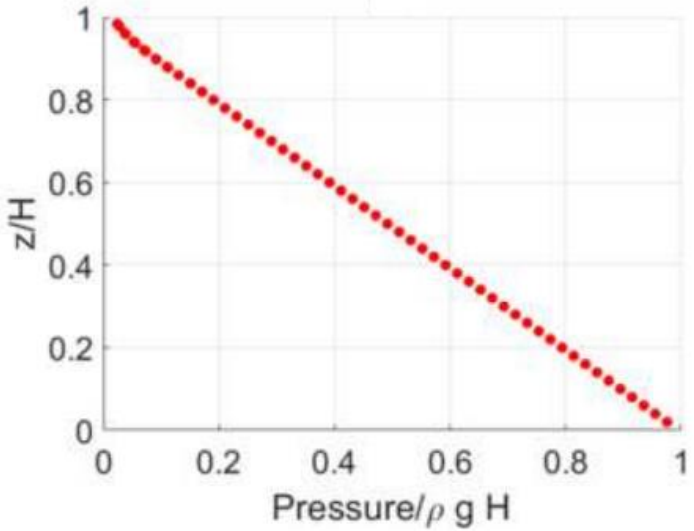
**Low resolution**



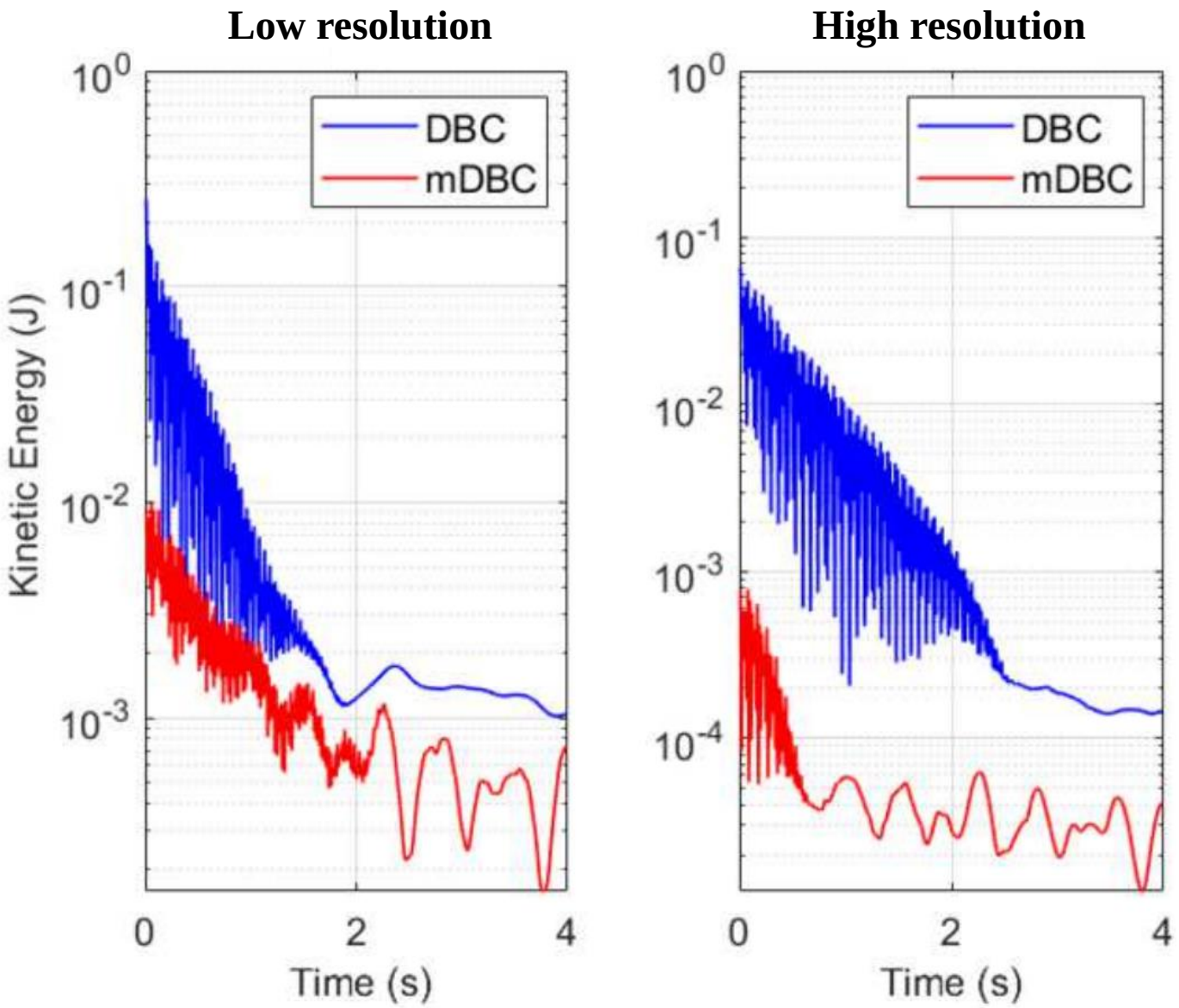




**High resolution**



# DBC vs. mDBC: Still water with a wedge




A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

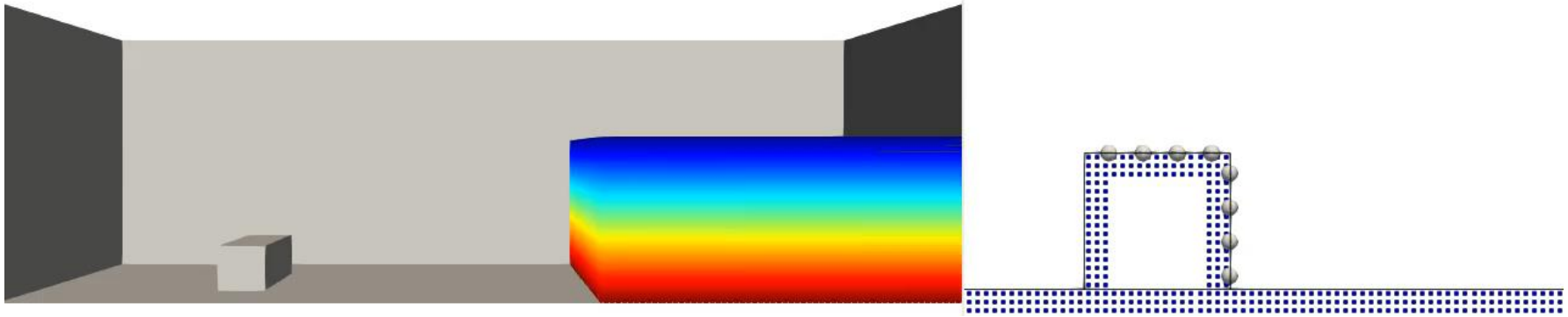
# DBC vs. mDBC: 3-D Dam break

A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

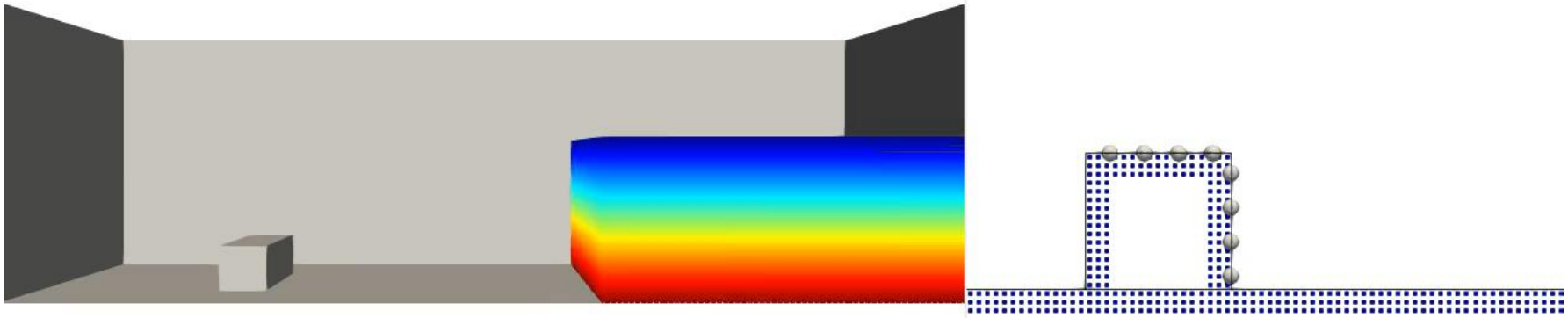
## SPHERIC Benchmark Testcase 2

Time: 0.00 s

**DBC**



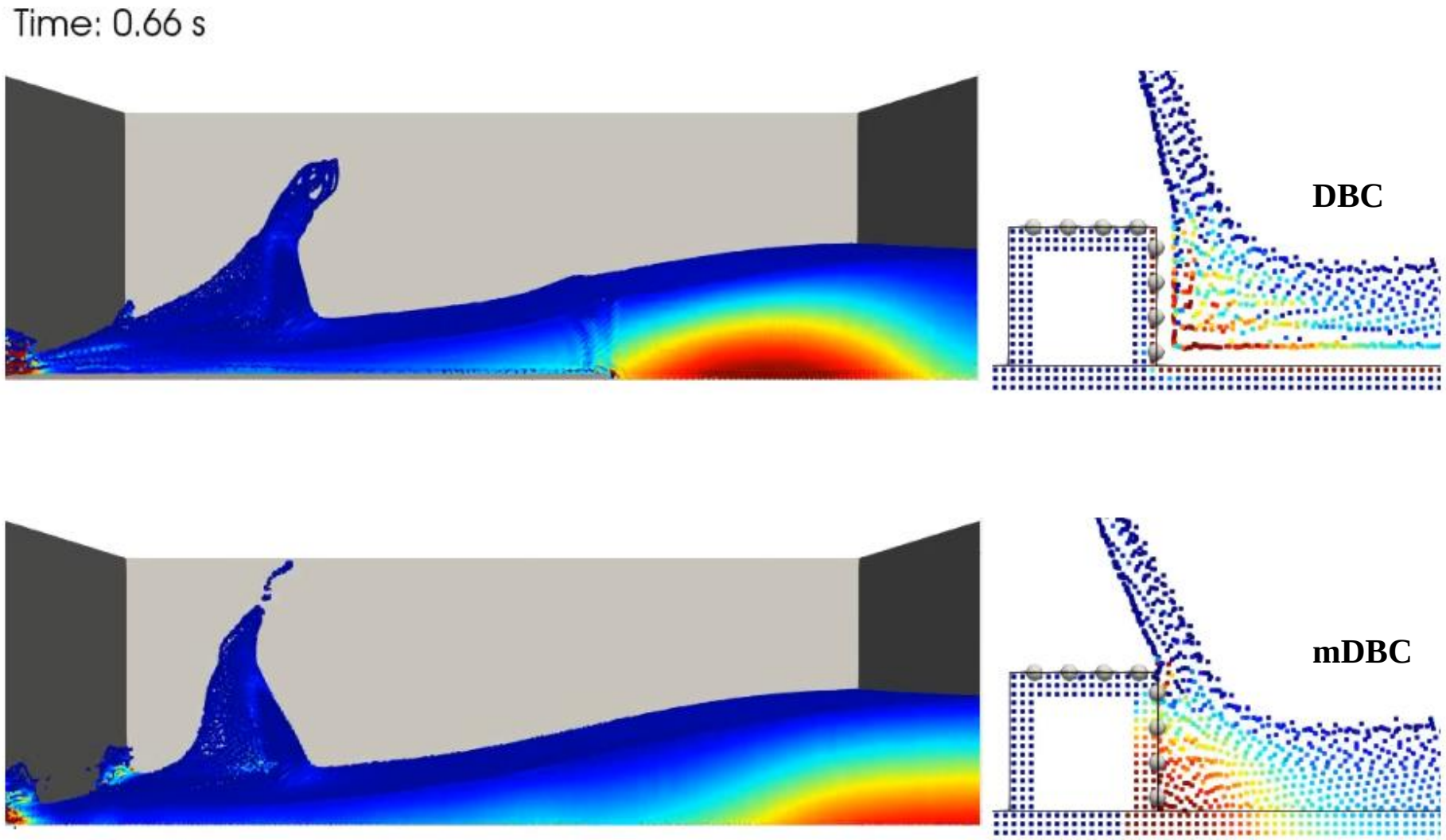
**mDBC**



# DBC vs. mDBC: 3-D Dam break

A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

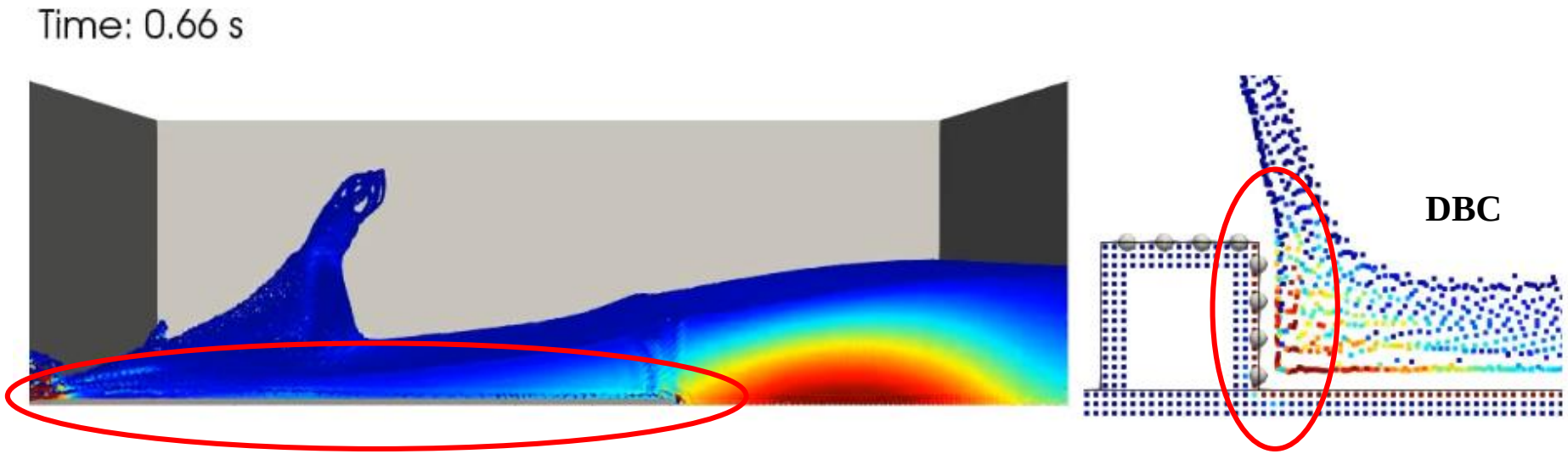
## SPHERIC Benchmark Testcase 2



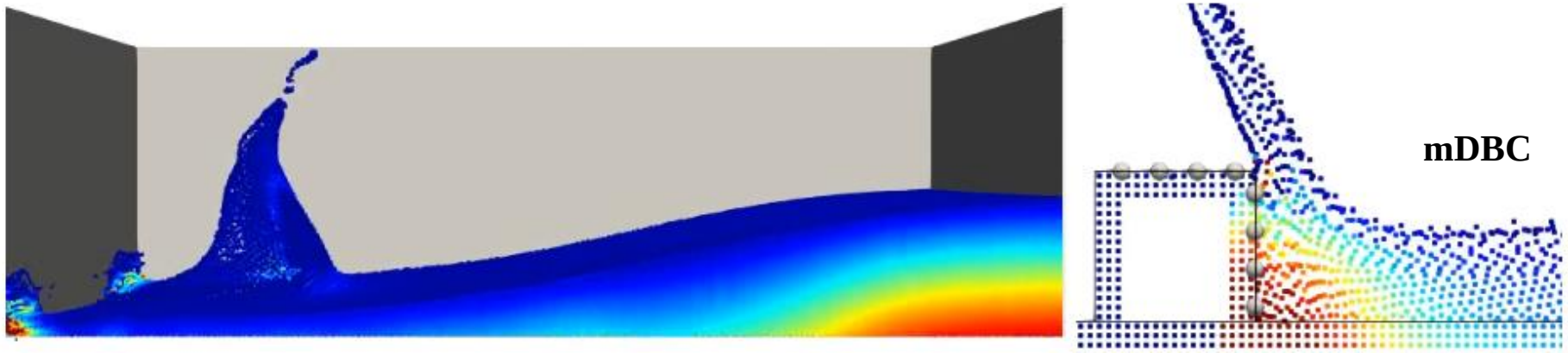
# DBC vs. mDBC: 3-D Dam break

A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

## SPHERIC Benchmark Testcase 2



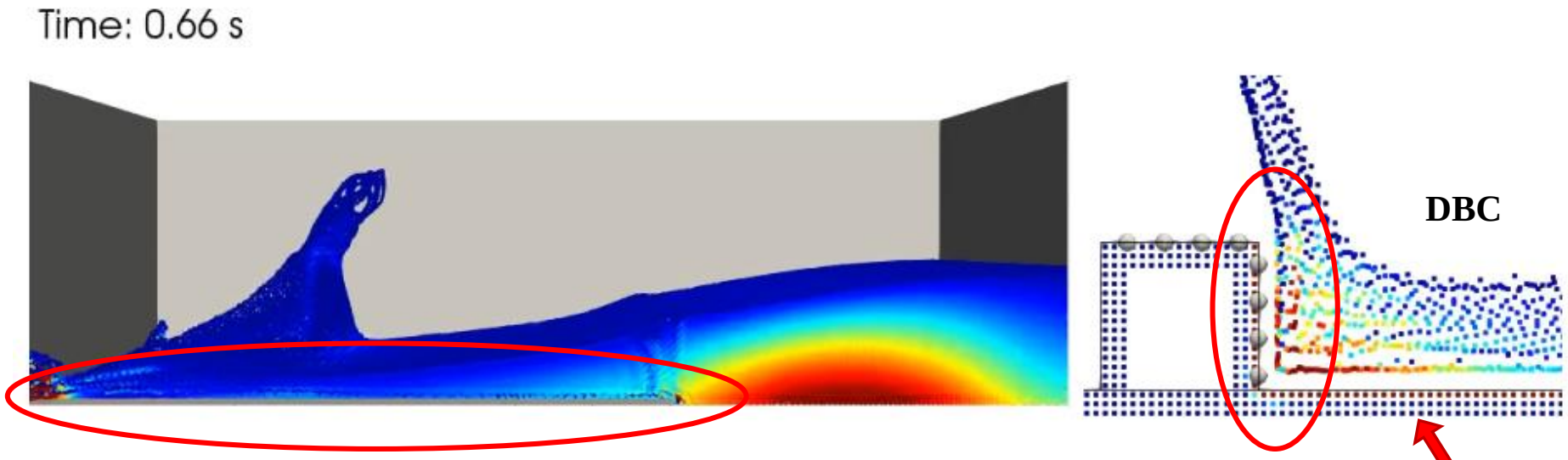

**Gap between fluid and boundaries**



# DBC vs. mDBC: 3-D Dam break

A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

## SPHERIC Benchmark Testcase 2

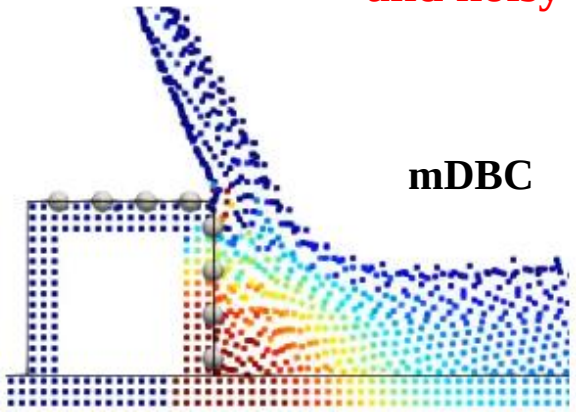





**DBC**

**Gap between fluid and boundaries**

**Non-physical density on boundary and noisy density in fluid**

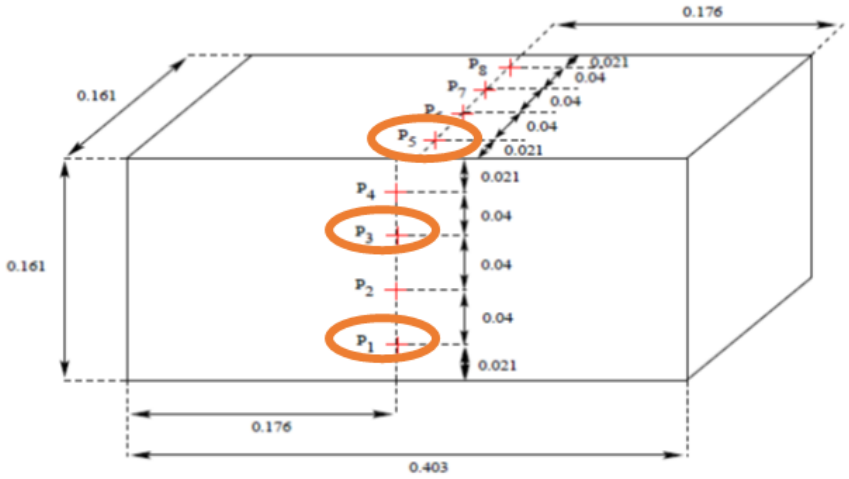


**mDBC**

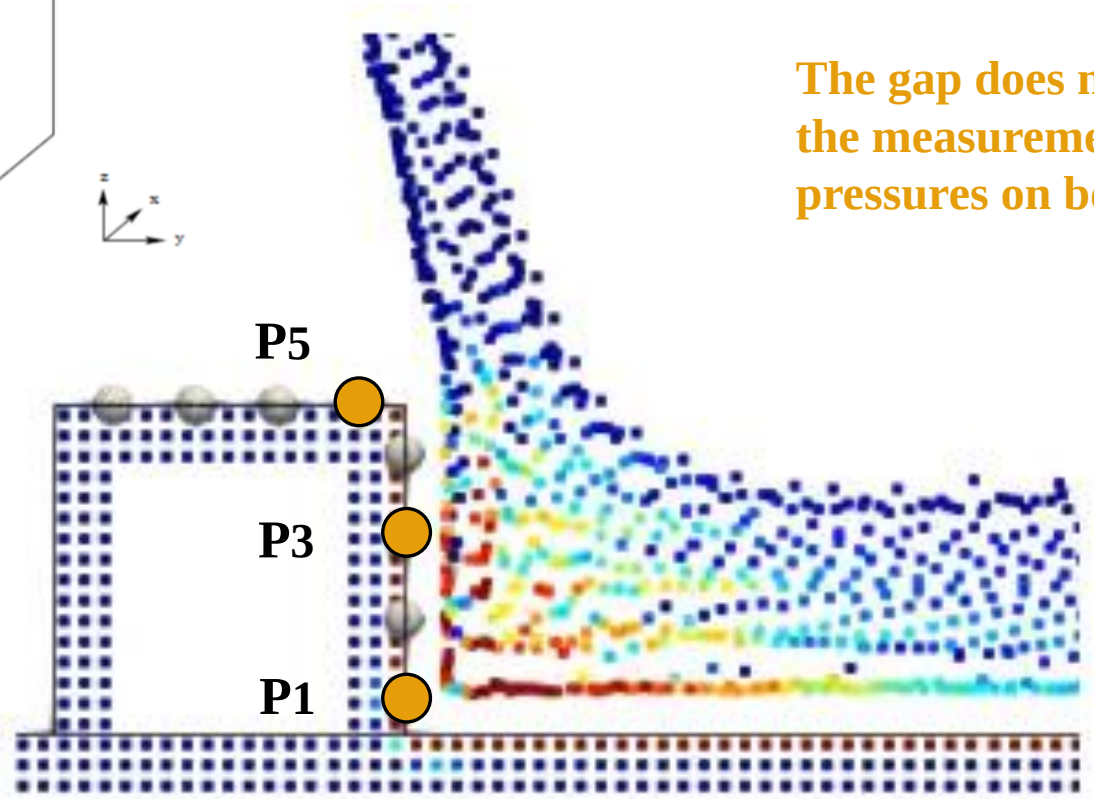
# DBC vs. mDBC: 3-D Dam break

A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

## SPHERIC Benchmark Testcase 2




**Pressure measurements** on the obstacle at positions: **P1**, **P3** and **P5**

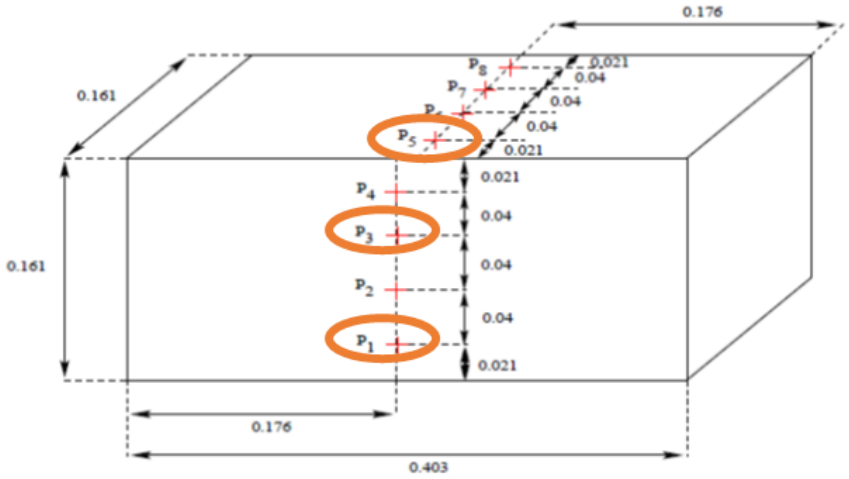



**The gap does not allow the measurement of pressures on boundary.**

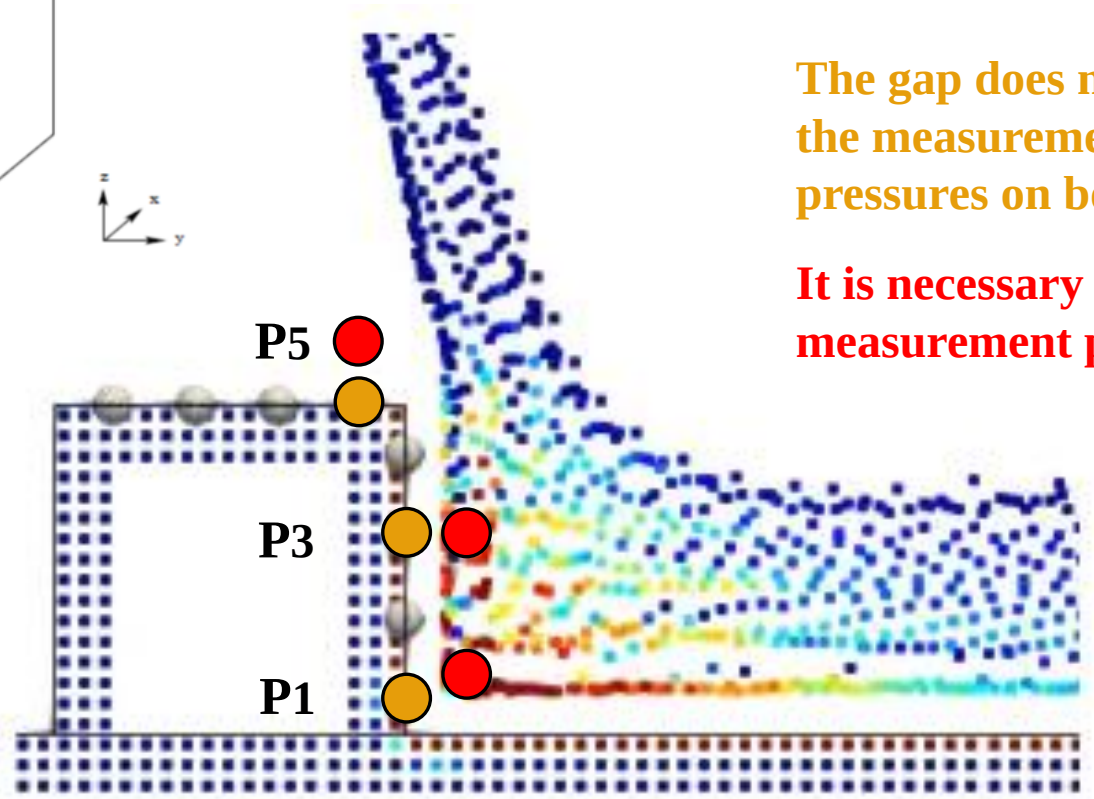
# DBC vs. mDBC: 3-D Dam break

A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

## SPHERIC Benchmark Testcase 2




**Pressure measurements** on the obstacle at positions: **P1**, **P3** and **P5**




**The gap does not allow the measurement of pressures on boundary.**

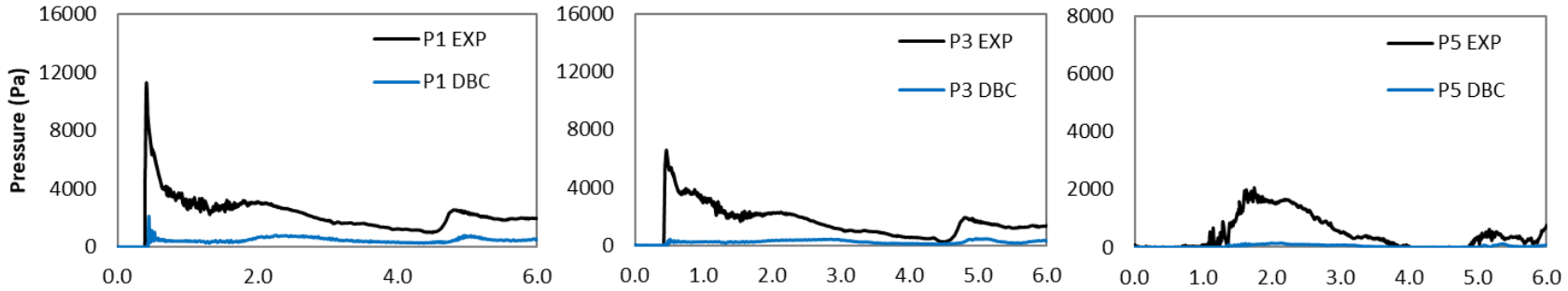
**It is necessary to move the measurement position +h**

# DBC vs. mDBC: 3-D Dam break

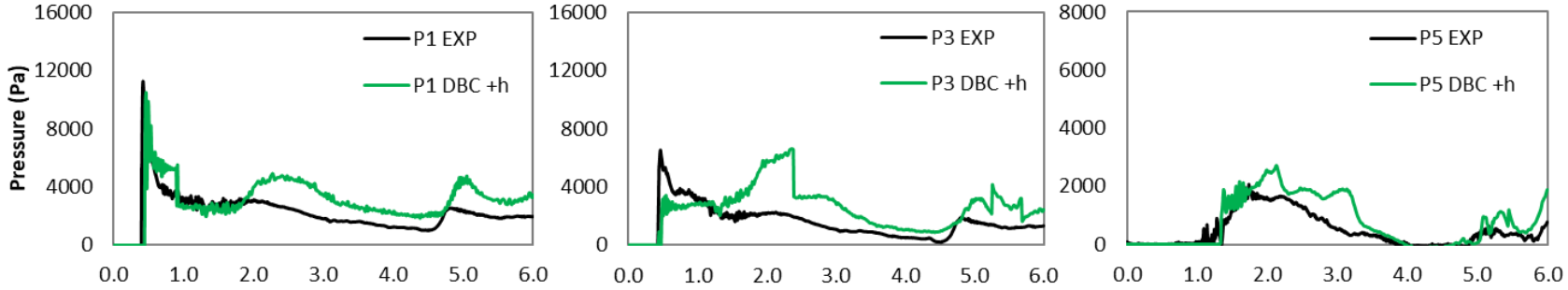
A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

## SPHERIC Benchmark Testcase 2

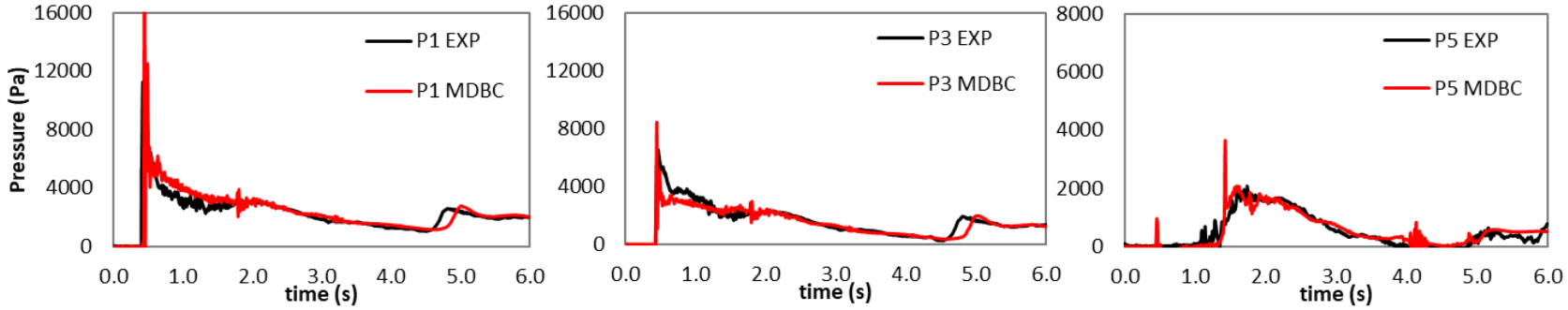
**DBC**

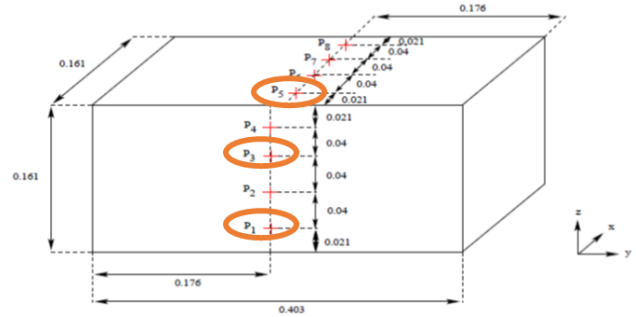



**DBC (+h)**




**mDBC**



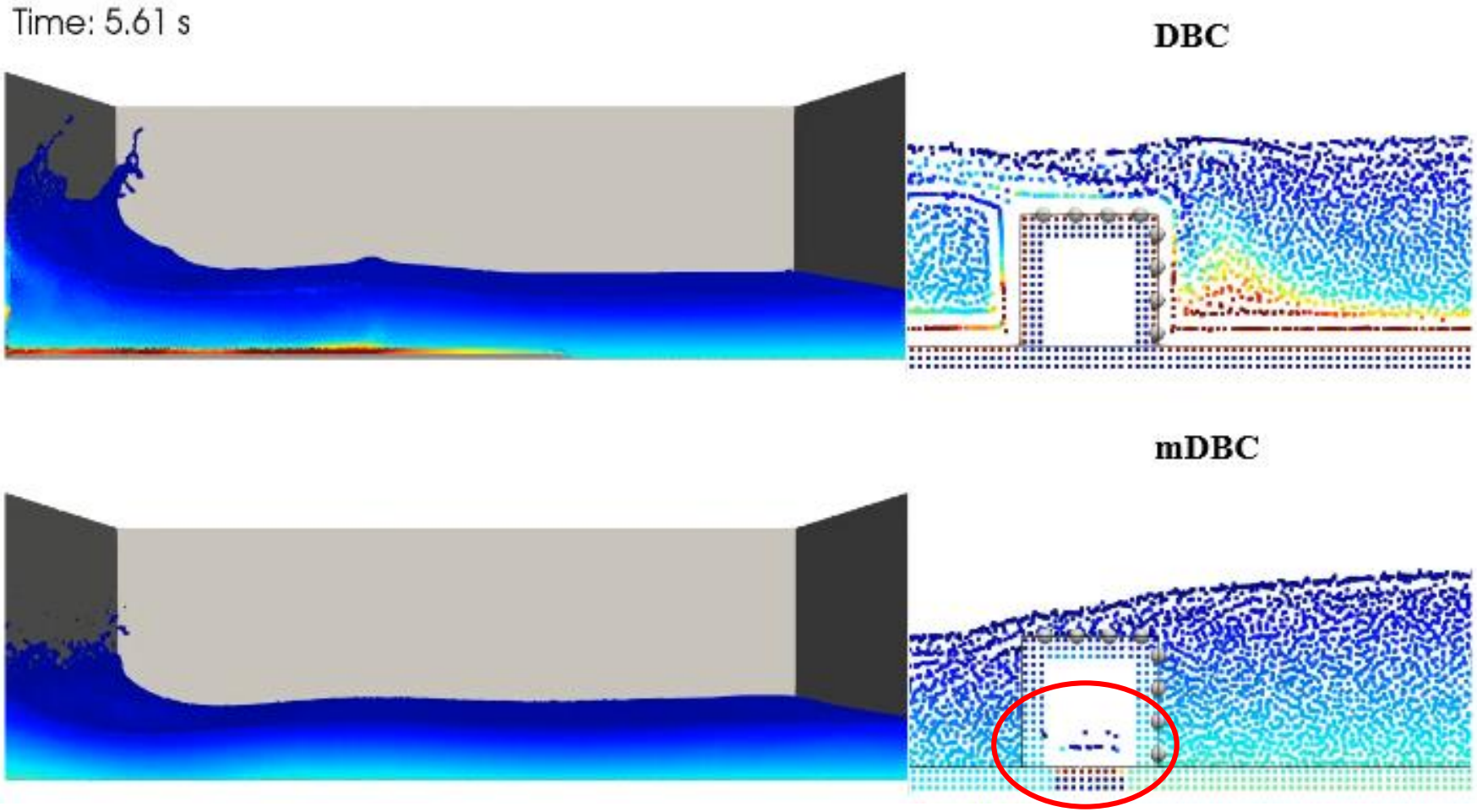



**Pressure measurements** on the obstacle at positions: **P1**, **P3** and **P5**

# DBC vs. mDBC: 3-D Dam break

A. English, J.M. Domínguez, R. Vacondio, A.J.C. Crespo, P.K. Stansby, S.J. Lind,L. Chiapponi, M. Gómez-Gesteira. 2021. **Modified dynamic boundary conditions (mDBC) for general purpose smoothed particle hydrodynamics (SPH): application to tank sloshing, dam break and fish pass problems.** Computational Particle Mechanics. **IN PRESS**

**SPHERIC Benchmark Testcase 2**




**Some particles go through the boundaries**

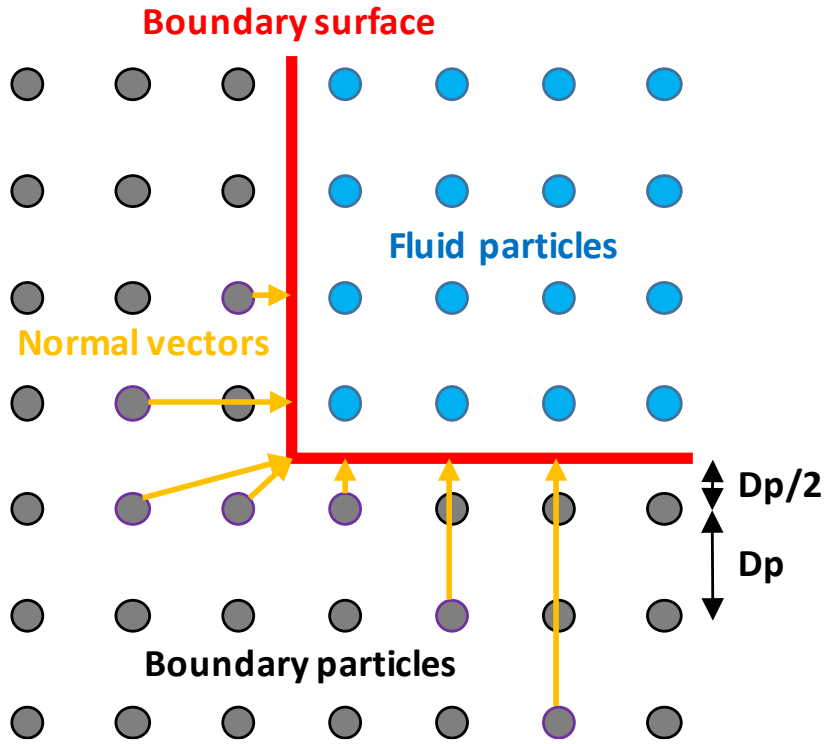
# mDBC REQUIREMENTS

mDBC presents several improvements regarding to DBC:

- Physical density/pressure values in boundary particles.
- Avoids the separation distance between boundary and fluid particles (GAP).

However, creating the initial condition for mDBC is more complicated since:

- Several boundary particles layers are necessary (**three layers or more**). It depends on the SPH smoothing length (h). Three layers are enough when $2h \leq 3Dp$.
- Distance between boundary limit and boundary particles should be half the initial inter-particle distance (**Dp/2**).
- Vectors from boundary particles to actual boundary limit are necessary (**normals**).



# mDBC REQUIREMENTS

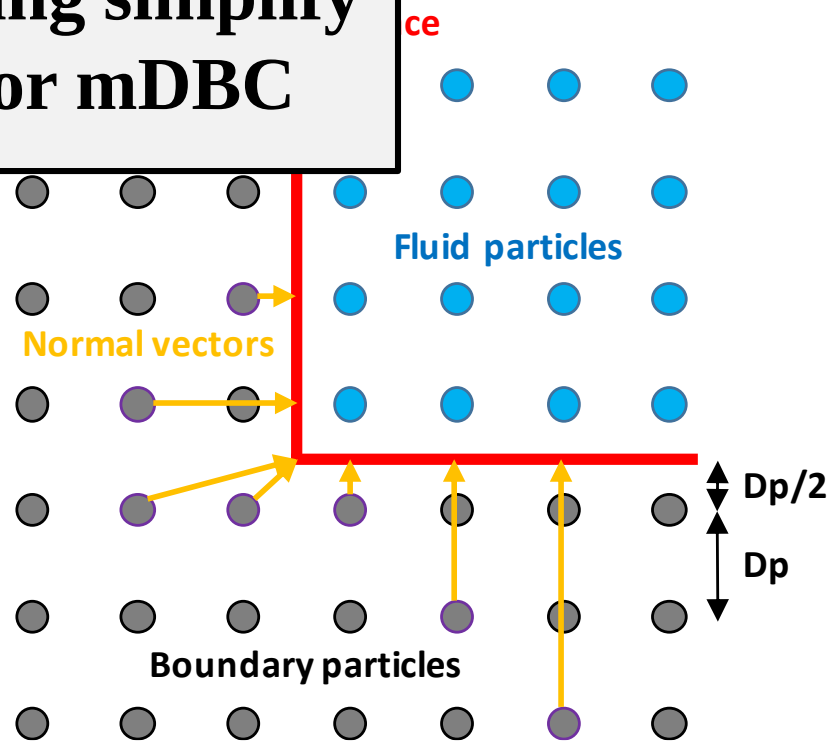
mDBC presents several improvements regarding to DBC:

- Physical density/pressure values in boundary particles.
- Avoids the separation distance between boundary and fluid particles (GAP).

However, creating the in

- Several boundary par **or more**). It depends layers are enough wh
- Distance between boundary limit and boundary particles should be half the initial inter-particle distance (**Dp/2**).
- Vectors from boundary particles to actual boundary limit are necessary (**normals**).

**New options in Pre-processing simplify the initial configuration for mDBC**



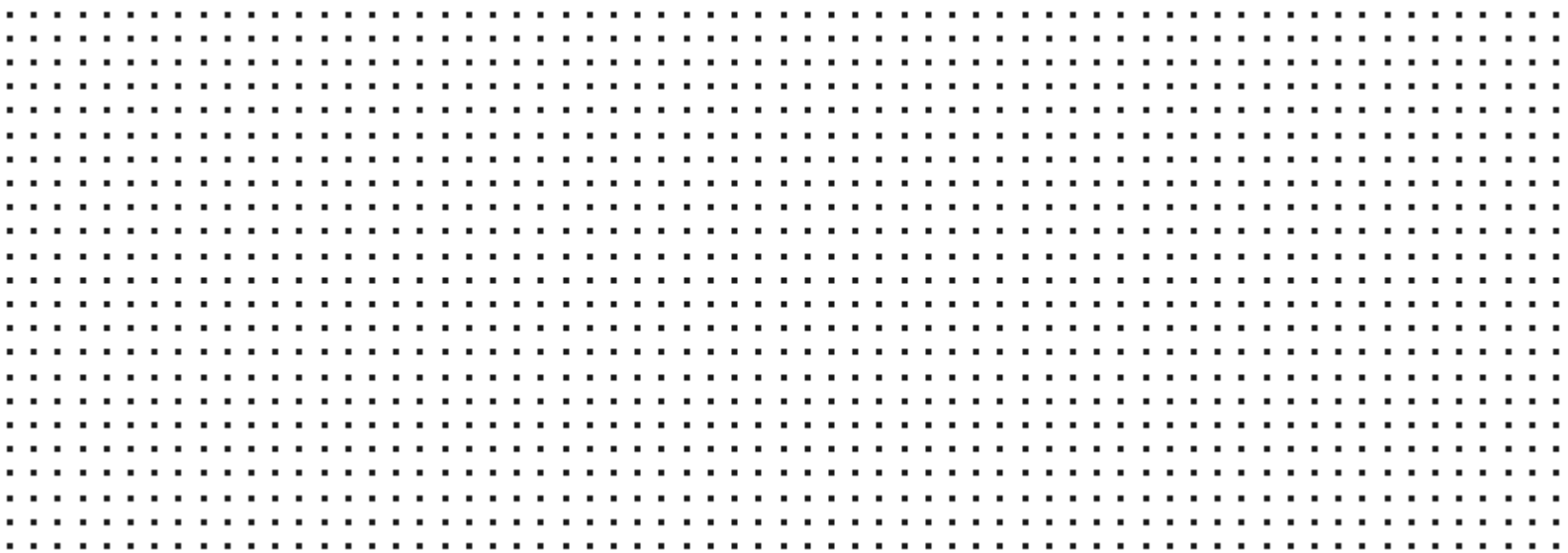
# OUTLINE

- Introduction
  - New boundary conditions: mDBC
    - Motivation: DBC drawbacks
    - Fluid properties from ghost nodes
    - mDBC vs. DBC
    - mDBC requirements
      - **New options on Preprocessing**
        - Free drawing mode (without lattice limitation)
        - Automatic multi-layer drawing
        - XML programming style
        - Automatic calculation of normals (for mDBC)
          - Conclusions & future improvements

# NORMAL DRAWING MODE (using cubic lattice)

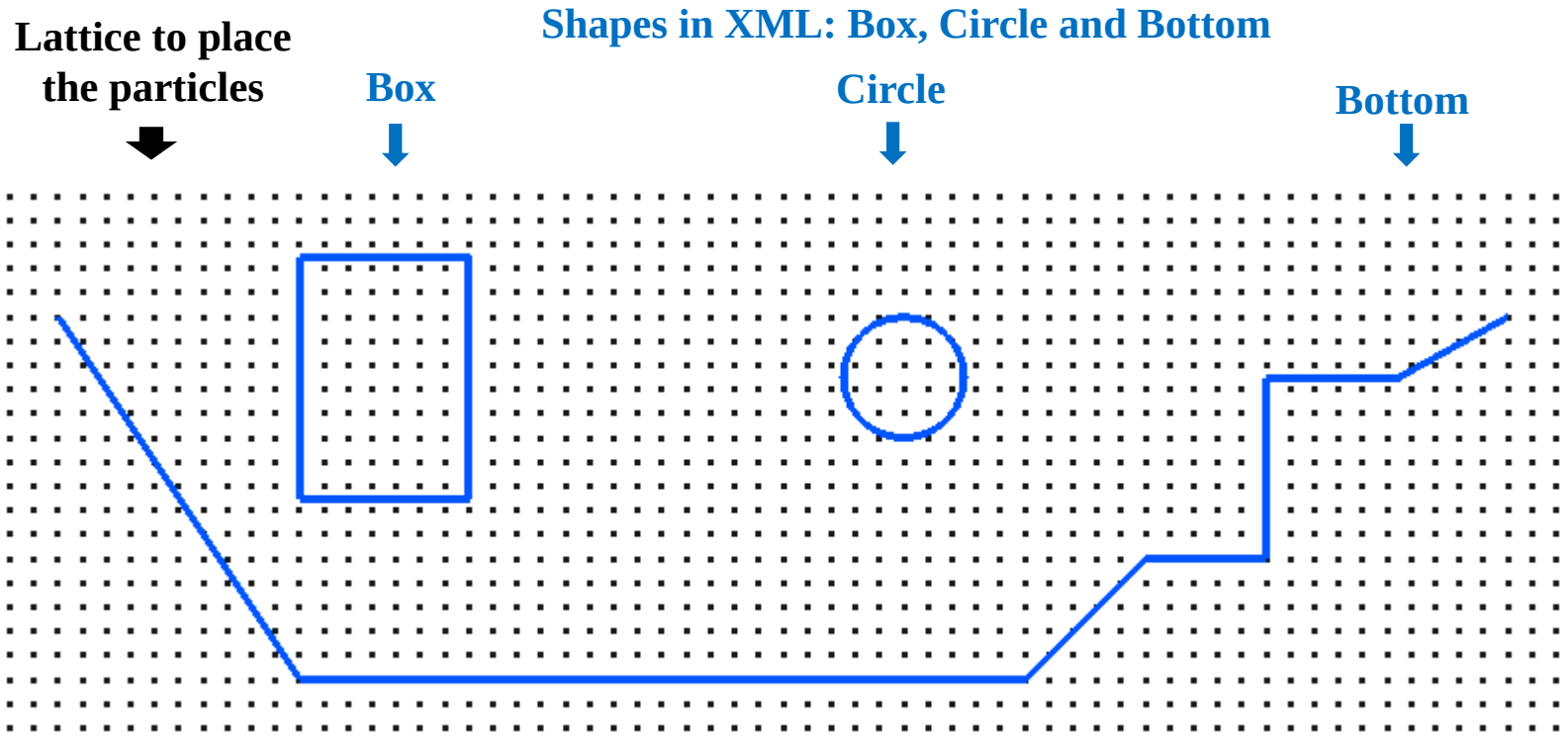
**The preprocessing tool (GenCase) uses a cubic lattice** to draw simple shapes (lines, triangles, spheres…), so particles can only be placed at points on this lattice.

**Lattice to place the particles**



# NORMAL DRAWING MODE (using cubic lattice)

**The preprocessing tool (GenCase) uses a cubic lattice** to draw simple shapes (lines, triangles, spheres…), so particles can only be placed at points on this lattice.




**Shapes in XML: Box, Circle and Bottom**

```
<setshapemode>actual | bound</setshapemode>
<setmkbound mk="0" />
```

**Box**

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

**Circle**

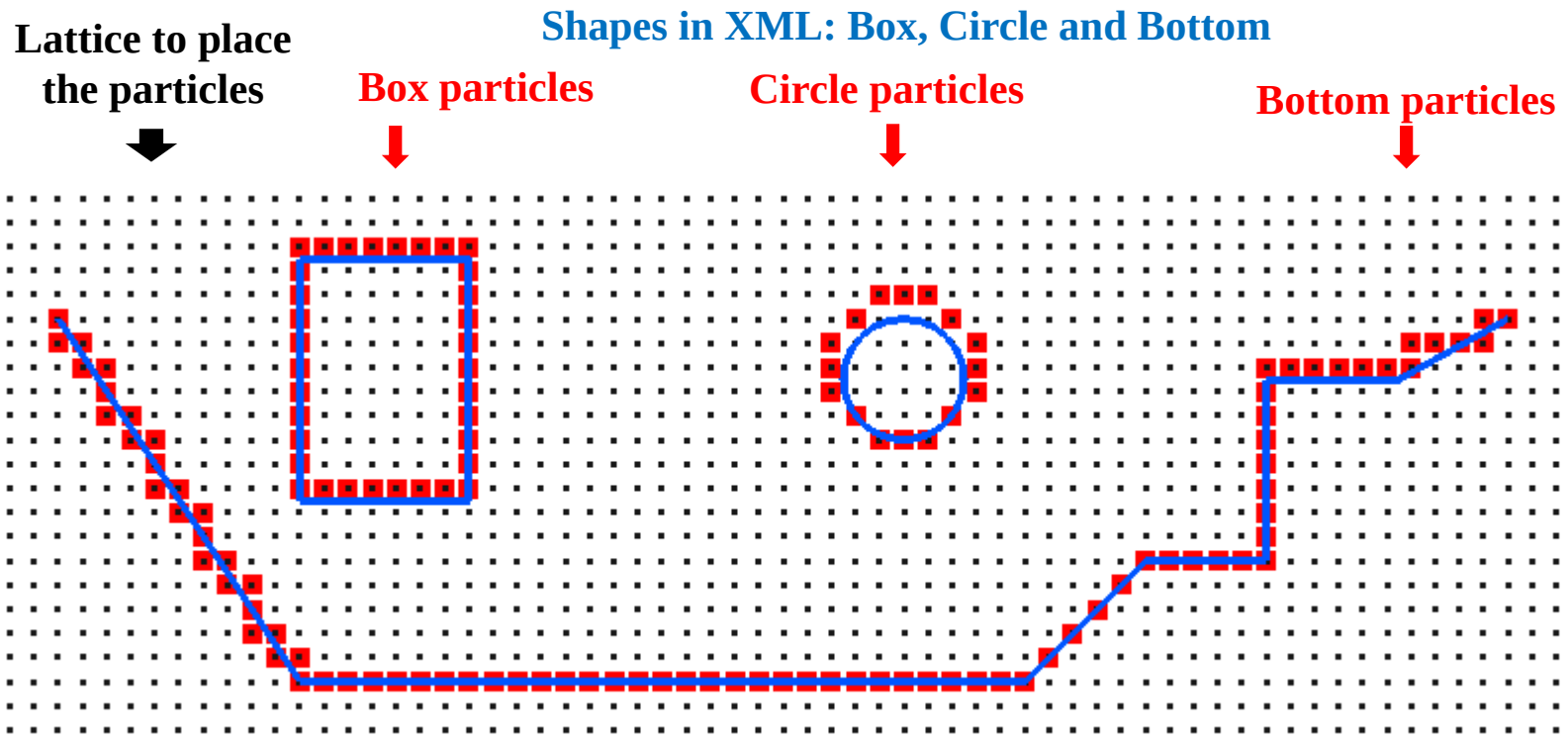
```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

**Bottom**

```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

# NORMAL DRAWING MODE (using cubic lattice)

**The preprocessing tool (GenCase) uses a cubic lattice** to draw simple shapes (lines, triangles, spheres…), so particles can only be placed at points on this lattice.




```
<setshapemode>actual | bound</setshapemode>
<setmkbound mk="0" />
```

Box

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

Circle

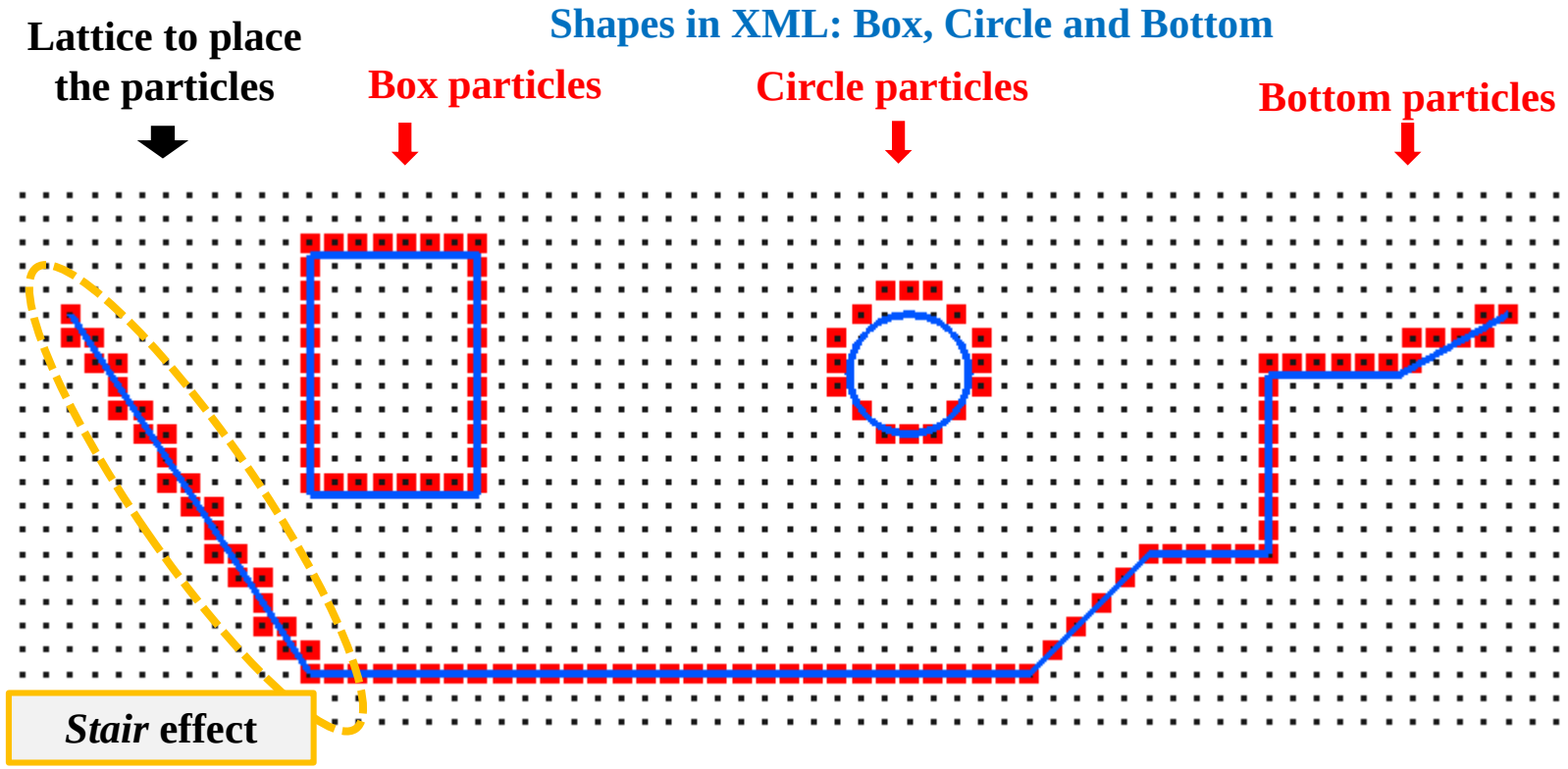
```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

Bottom

```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

# NORMAL DRAWING MODE (using cubic lattice)

**The preprocessing tool (GenCase) uses a cubic lattice** to draw simple shapes (lines, triangles, spheres…), so particles can only be placed at points on this lattice.




```
<setshapemode>actual | bound</setshapemode>
<setmkbound mk="0" />
```

Box

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

Circle

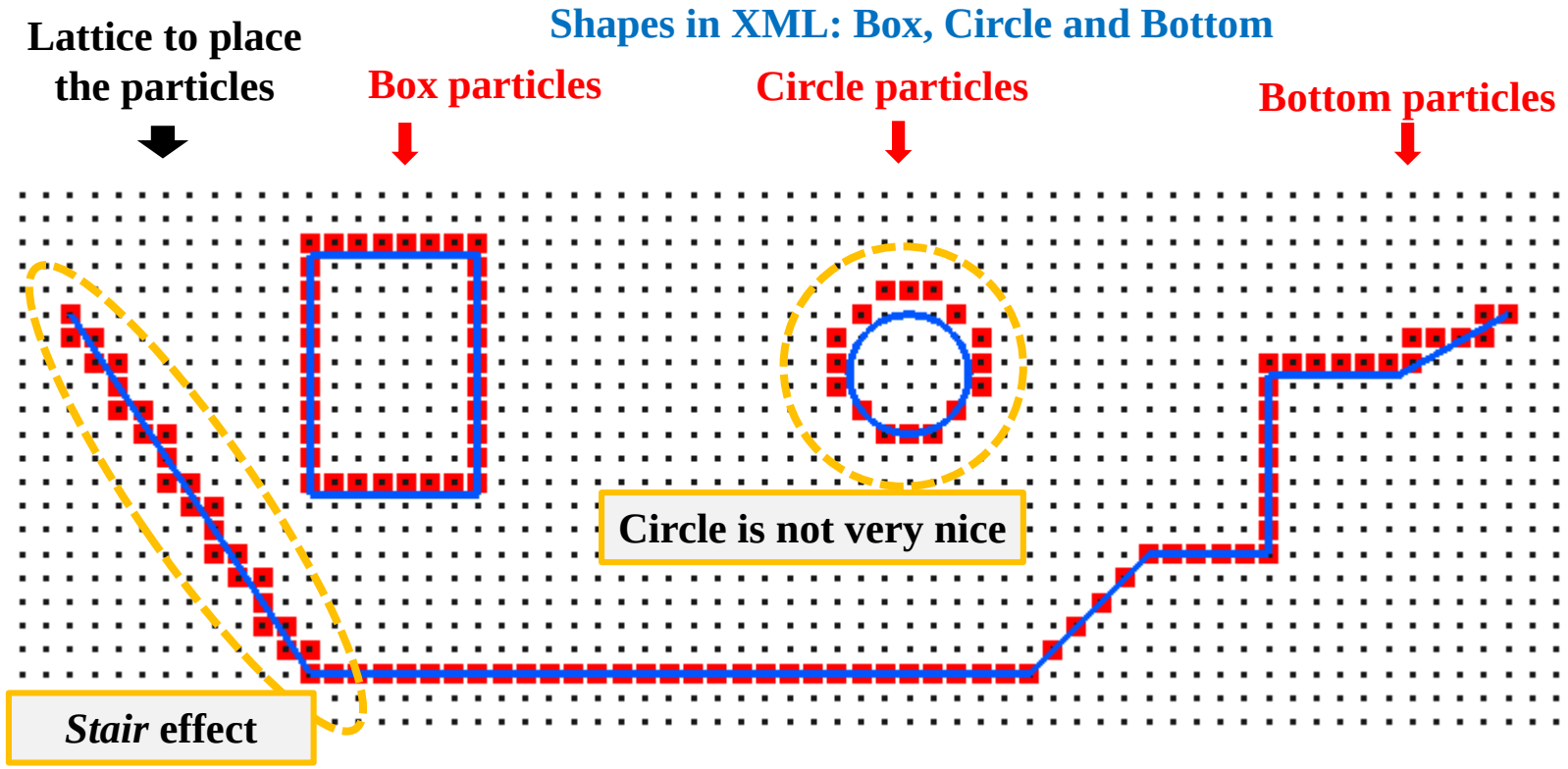
```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

Bottom

```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

# NORMAL DRAWING MODE (using cubic lattice)

**The preprocessing tool (GenCase) uses a cubic lattice** to draw simple shapes (lines, triangles, spheres…), so particles can only be placed at points on this lattice.




```
<setshapemode>actual | bound</setshapemode>
<setmkbound mk="0" />
```

Box

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

Circle

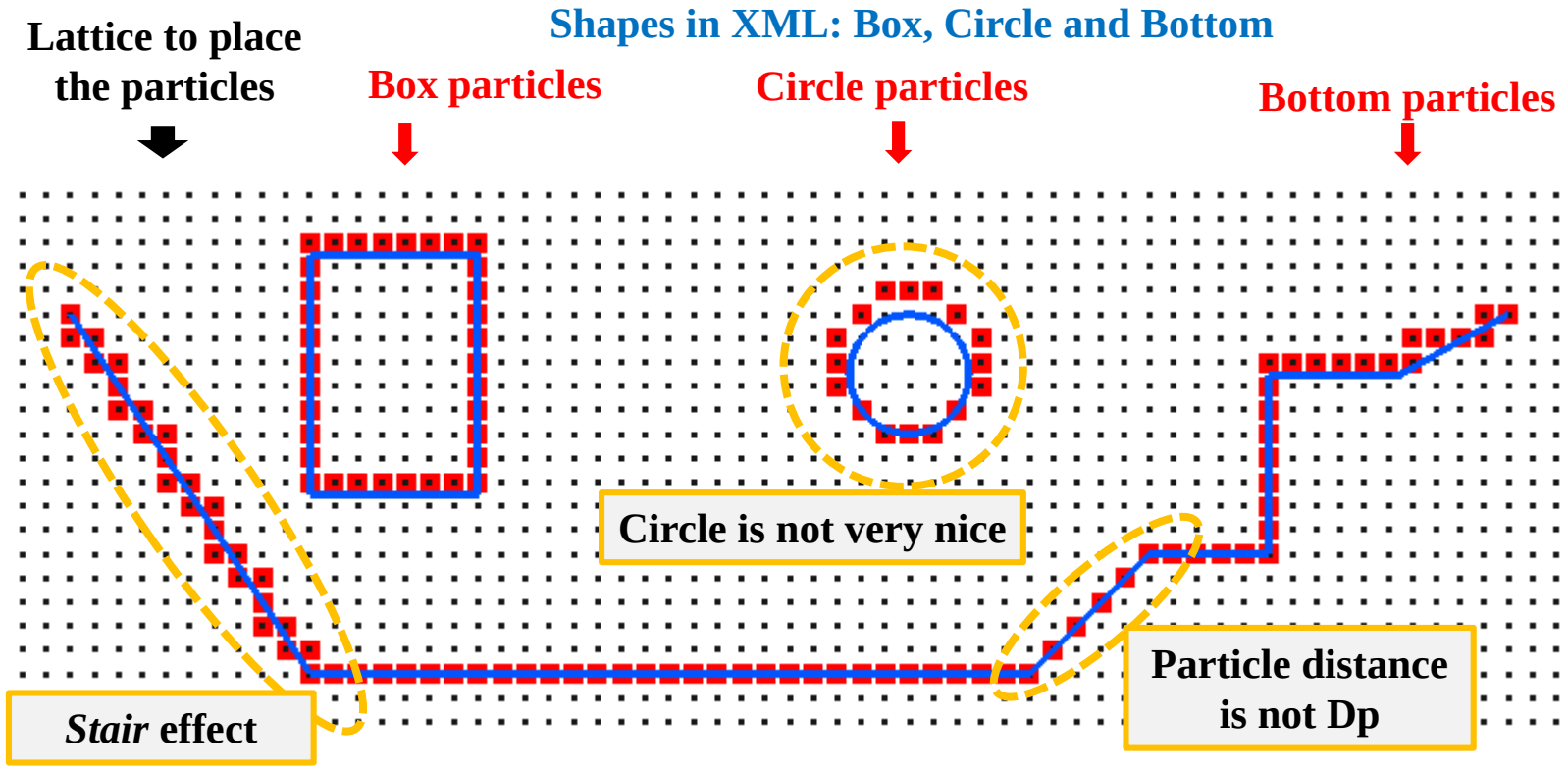
```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

Bottom

```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

# NORMAL DRAWING MODE (using cubic lattice)

**The preprocessing tool (GenCase) uses a cubic lattice** to draw simple shapes (lines, triangles, spheres…), so particles can only be placed at points on this lattice.




```
<setshapemode>actual | bound</setshapemode>
<setmkbound mk="0" />
```

Box

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

Circle

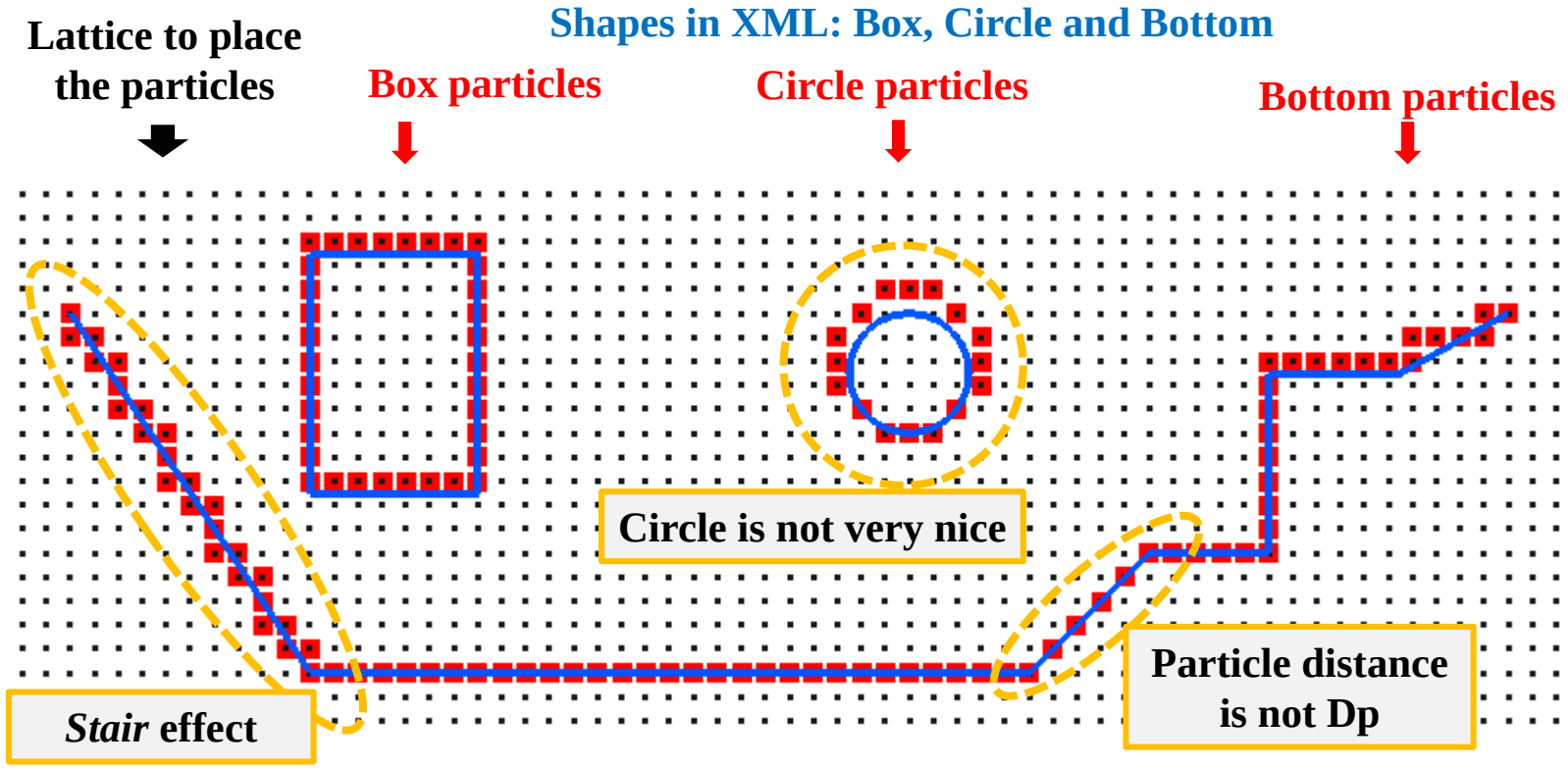
```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

Bottom

```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

# NORMAL DRAWING MODE (using cubic lattice)

**The preprocessing tool (GenCase) uses a cubic lattice** to draw simple shapes (lines, triangles, spheres…), so particles can only be placed at points on this lattice.




```
<setshapemode>actual | bound</setshapemode>
<setmkbound mk="0" />
```

Box

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

Circle

```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

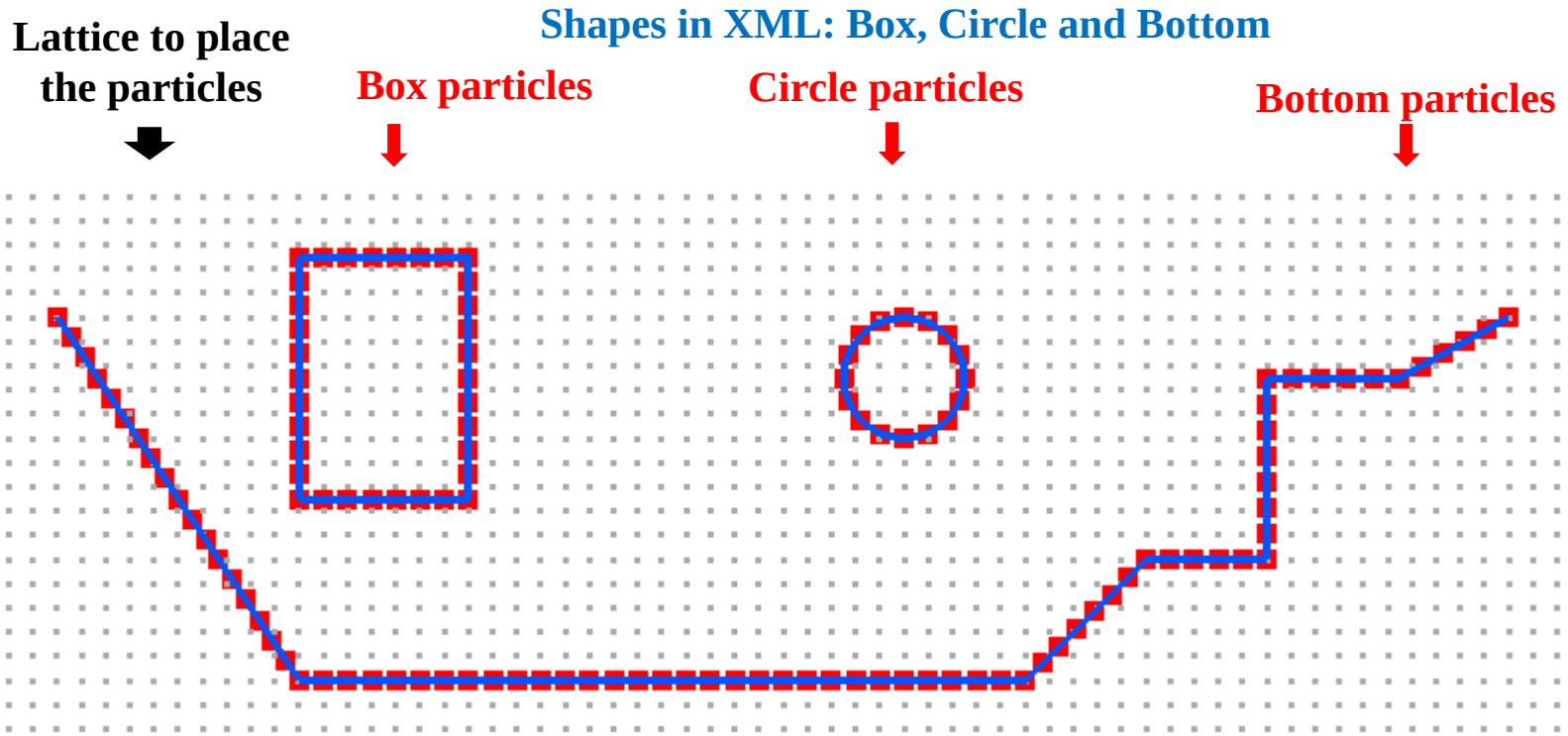
Bottom

```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

The error in the position of the particles depends on the resolution or distance between particles (*Dp*).
It is necessary to use higher resolution to minimize this inaccuracy.

# FREE DRAWING MODE (without cubic lattice)

**New free drawing mode to avoid the use of the cubic lattice.** It is a bit slower, but it places the particles in any position to match the drawn shape.




```
<setshapemode>actual | bound</setshapemode>
<setfrdrawmode auto="true" />
<setmkbound mk="0" />
```

Box

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

Circle

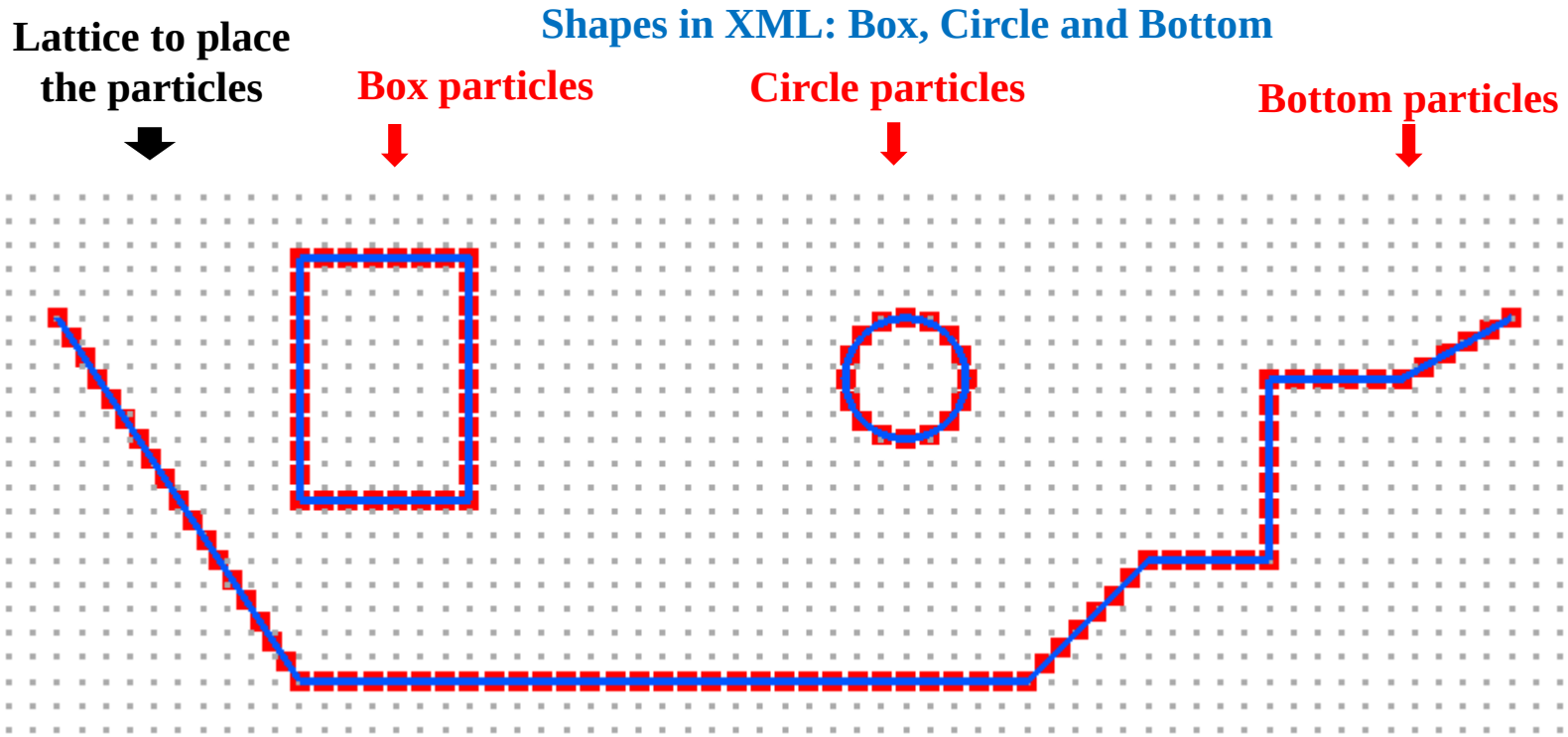
```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

Bottom

```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

# FREE DRAWING MODE (without cubic lattice)

**New free drawing mode to avoid the use of the cubic lattice.** It is a bit slower, but it places the particles in any position to match the drawn shape.




New free drawing mode is activated with a command and works automatically.

```
<setshapemode>actual | bound</setshapemode>
<setfrdrawmode auto="true" />
<setmkbound mk="0" />
```

Box

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

Circle

```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

Bottom

```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

# FREE DRAWING MODE (without cubic lattice)

**New free drawing mode to avoid the use of the cubic lattice.** It is a bit slower, but it places the particles in any position to match the drawn shape.

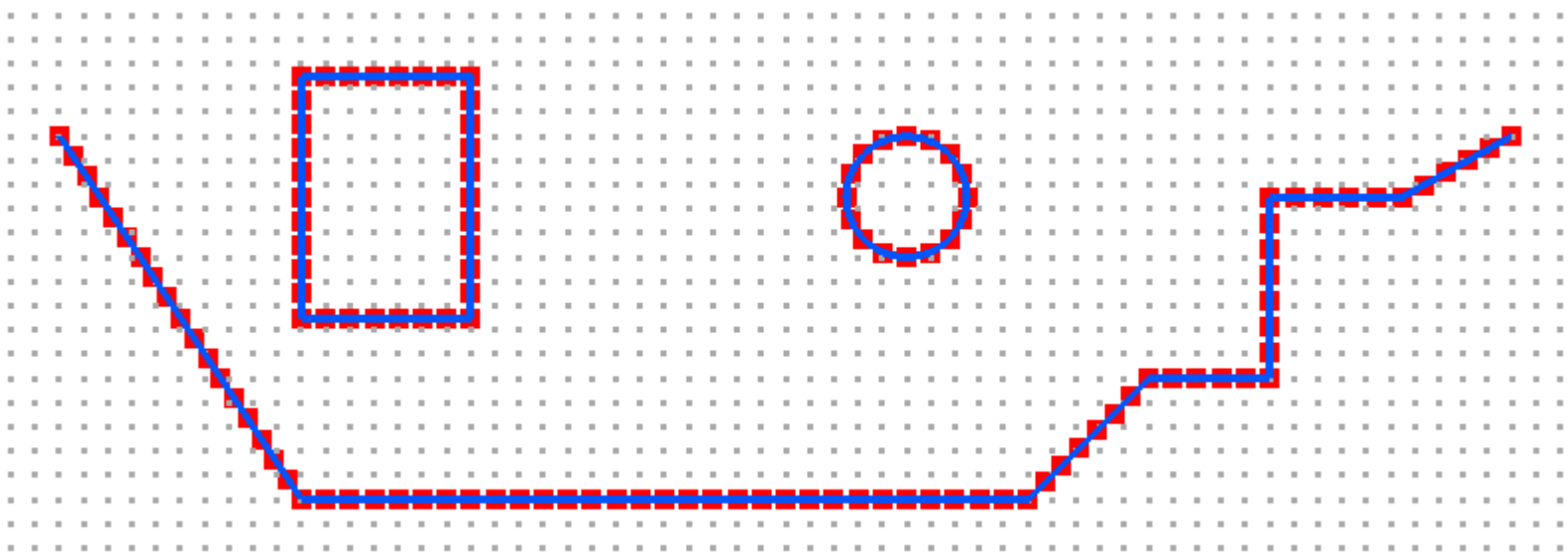
Lattice to place the particles

Shapes in XML: Box, Circle and Bottom

Box particles

Circle particles

Bottom particles



New free drawing mode is activated with a command and works automatically.

```
<setshapemode>actual | bound</setshapemode>
<setfrdrawmode auto="true" />
<setmkbound mk="0" />
```

Box

```
<drawbox>
    <boxfill>all</boxfill>
    <point x="2" y="-1" z="1.5" />
    <size x="1.4" y="2" z="2" />
</drawbox>
```

Circle

```
<drawcylinder radius="0.5" mask="1|2">
    <point x="7" y="-1" z="2.5" />
    <point x="7" y="1" z="2.5" />
</drawcylinder>
```

Bottom

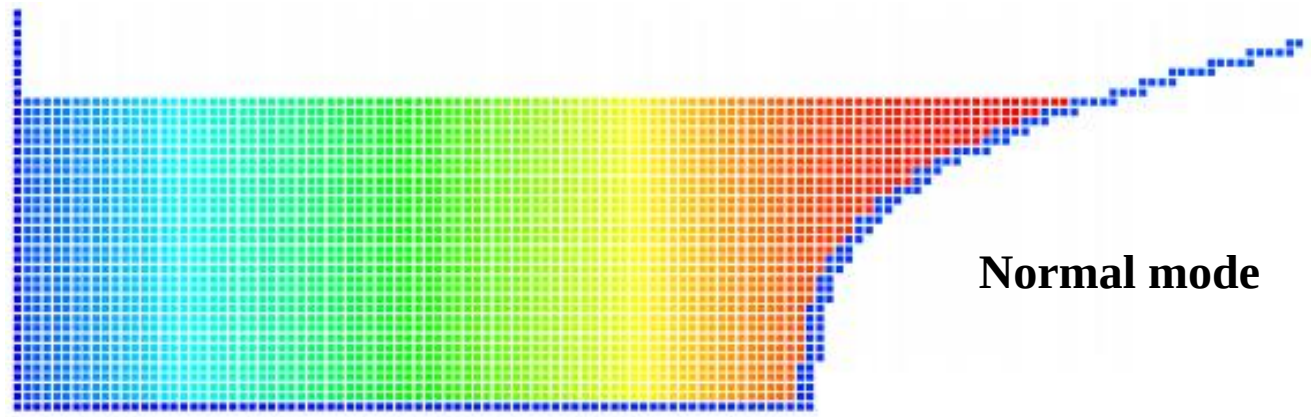
```
<drawextrude closed="false">
    <point x="0" y="0" z="3" />
    <point x="2" y="0" z="0" />
    <point x="8" y="0" z="0" />
    <point x="9" y="0" z="1" />
    <point x="10" y="0" z="1" />
    <point x="10" y="0" z="2.5" />
    <point x="11.1" y="0" z="2.5" />
    <point x="12" y="0" z="3" />
    <extrude x="0" y="1" z="0" />
</drawextrude>
<shapeout file="" />
```

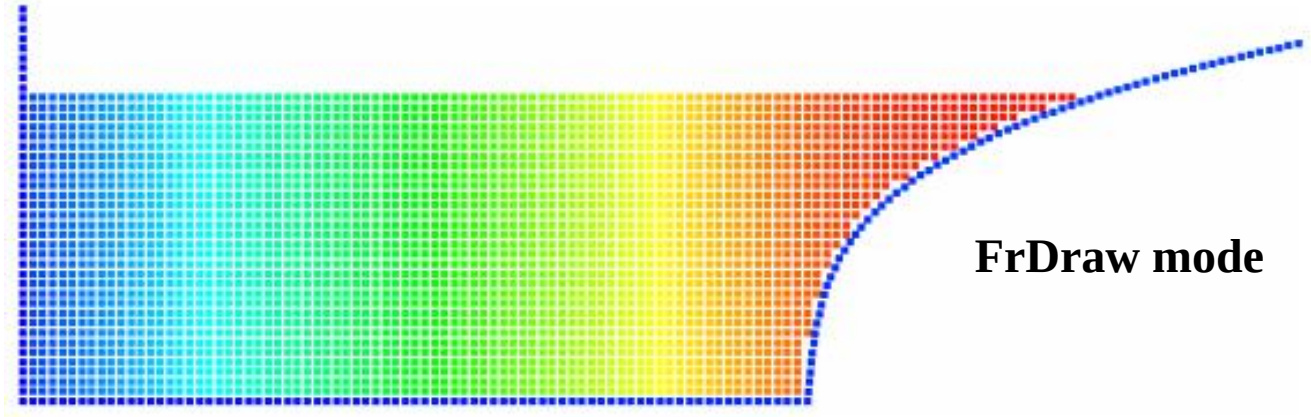
**The improvements are: 1) no stair effect, 2) particles match the dimension of the shape, 3) the circle is a circle.**

**Provides good results with low resolution.**

# FREE DRAWING MODE (without cubic lattice)

**New free drawing mode to avoid the use of the cubic lattice.** It is a bit slower, but it places the particles in any position to match the drawn shape.




**Parabolic beach** defined by

$$\mathbf{x = Az^B + Cz}$$

**A** = 0.05
**B** = 3.5
**C** = 0.1

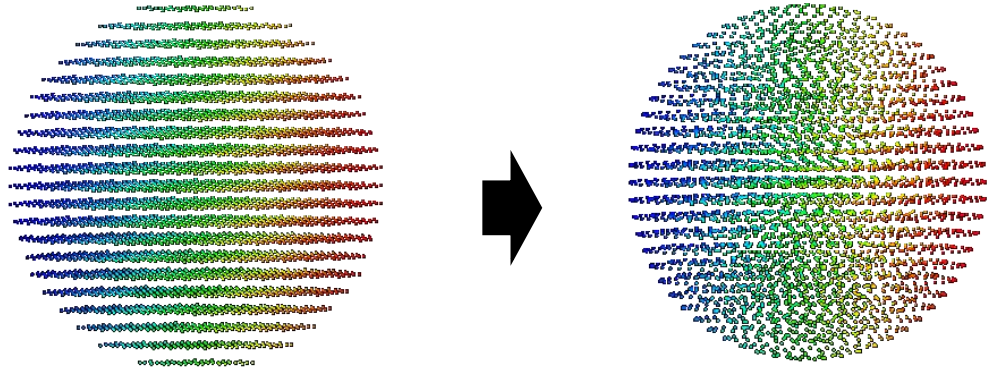
# FREE DRAWING MODE (without cubic lattice)

**New free drawing mode also works with 3D objects** such as spheres or cylinders.

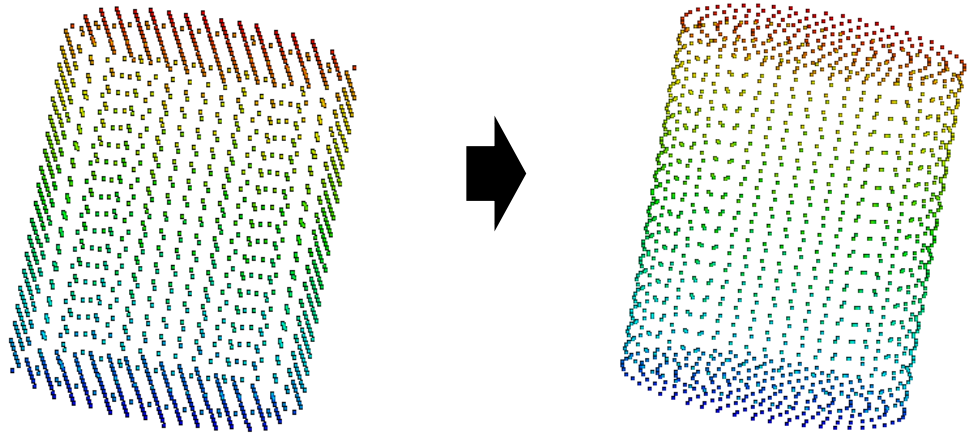
It allows to discretise rounded surfaces without staircase effect and this is essential in some special cases.

**Normal mode** **vs.** **FrDraw mode**

```
<setdrawmode mode="full" />
<setmkbound mk="0" name="Sphere" />
<setfrdrawmode auto="true" />
<drawsphere radius="5.0">
    <point x="0" y="0" z="5.0" />
</drawsphere>
<setfrdrawmode auto="false" />
```



```
<setdrawmode mode="face" />
<setmkbound mk="0" name="Cylinder" />
<setfrdrawmode auto="true" />
<drawcylinder radius="4.0" mask="0">
    <point x="0" y="0" z="0" />
    <point x="0" y="0" z="10" />
</drawcylinder>
<setfrdrawmode auto="false" />
```

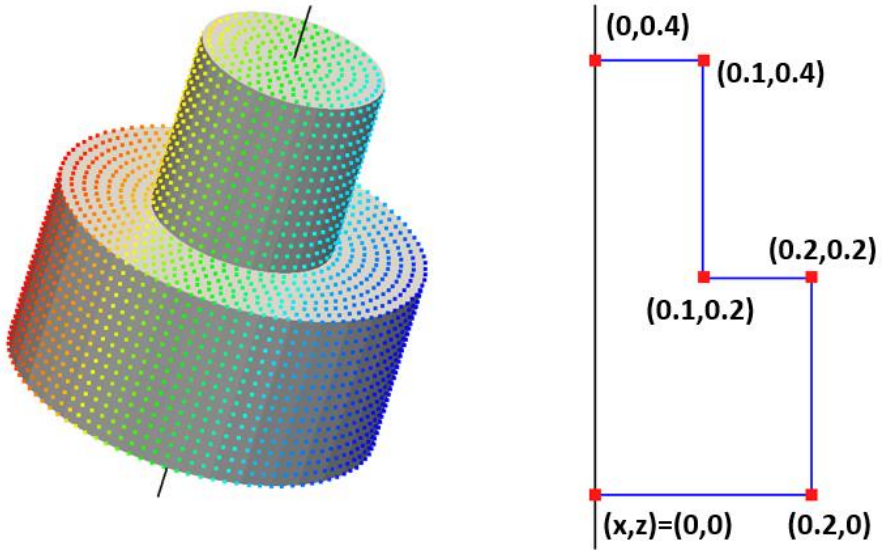


# FREE DRAWING MODE (without cubic lattice)

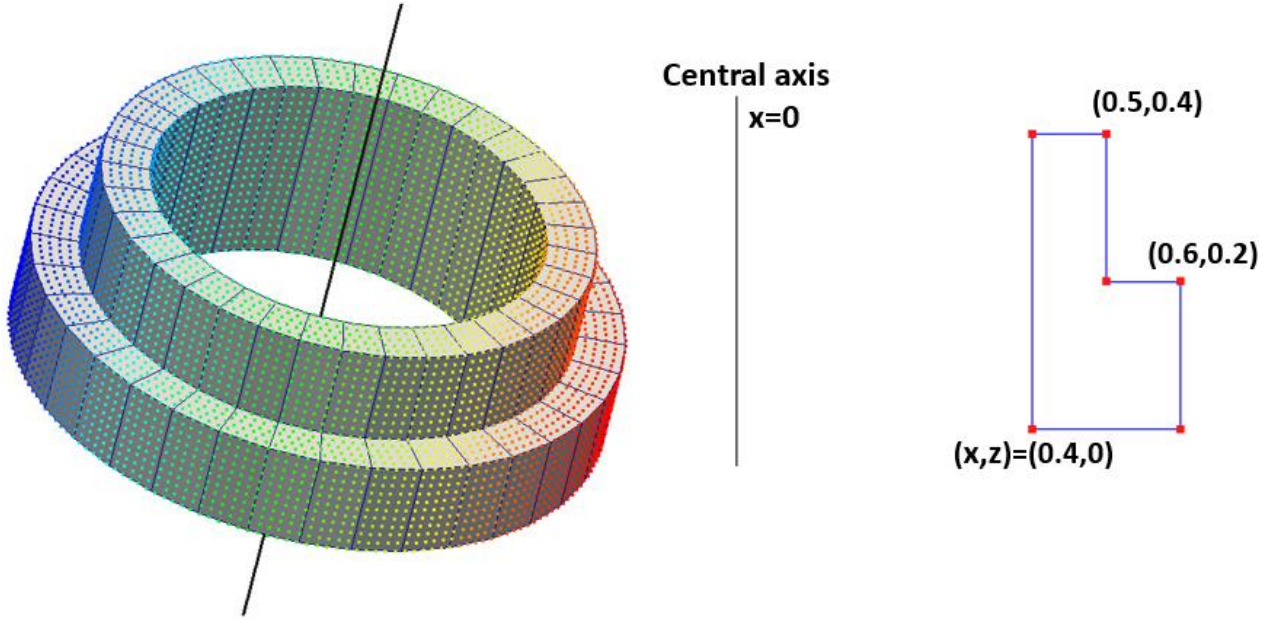
**New free drawing mode also works with 3D objects** such as spheres or cylinders.

It allows to discretise rounded surfaces without staircase effect and this is essential in some special cases.

Examples of **3-D Revolution objects using FrDraw mode:**




Example 1: Central axis within the object




Example 2: Central axis outside the object

# AUTOMATIC MULTI-LAYER DRAWING

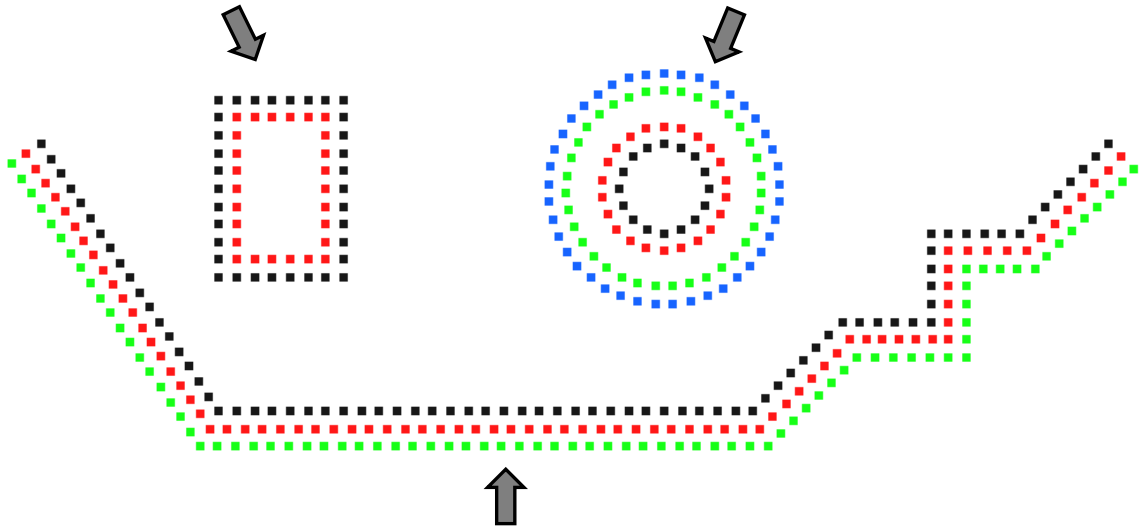
**The new option <layers> allows you to create shapes with multiple layers in an easy way.**

Each value creates an inner or outer layer using negative and positive values respectively according to Dp.

The asterisk defines the shapes saved by <shapeout>.

```
<drawbox>
   <layers vdp="0,-1"/>
</drawbox>
```

```
<drawcylinder>
   <layers vdp="0,1,3,4"/>
</drawcylinder>
```



```
<drawextrude>
   <layers vdp="0,-1,-2"/>
</drawextrude>
```

```
<mainlist>
    <setshapemode>actual | bound</setshapemode>
    <setfrdrawmode auto="true" />
    <setmkbound mk="0" />
    <drawbox>
        <boxfill>all</boxfill>
        <point x="2" y="-1" z="1.5" />
        <size x="1.4" y="2" z="2" />
        <layers vdp="0*,-1" />
    </drawbox>
    <drawcylinder radius="0.5" mask="2">
        <point x="7" y="-1" z="2.5" />
        <point x="7" y="1" z="2.5" />
        <layers vdp="0,1,3,4*" />
    </drawcylinder>
    <drawextrude closed="false">
        <point x="0" y="0" z="3" />
        <point x="2" y="0" z="0" />
        <point x="8" y="0" z="0" />
        <point x="9" y="0" z="1" />
        <point x="10" y="0" z="1" />
        <point x="10" y="0" z="2" />
        <point x="11" y="0" z="2" />
        <point x="12" y="0" z="3" />
        <extrude x="0" y="1" z="0" />
        <layers vdp="0*,-1,-2" />
    </drawextrude>
    <shapeout file="" reset="true" />
</mainlist>
```

# XML PROGRAMMING STYLE

**XML can be used as a programming language with variables, conditionals, loops, subroutines and input/output functions.** The execution of this code creates the initial simulation case for DualSPHysics.

The new version of GenCase allows the definition of variables in the XML and complex numerical expressions, where these variables and different functions (mathematical, statistical, logical…) are used to calculate the required value at runtime.

The commands to define **user variables** are:

```
<newvar size="0.4" _rem="Defines size" _print="1" />
<newvar var1="10" var2="25.5" var3="var1+var2" var4="true" />
<newvarcte cte1="1.0" cte2="var1+var2/size" cte3="(var1>=cte2)!=var3" />
<newvarstr file1="cube.stl" file2="sphere.stl" />
<newvarstrcte text="The file is [file1]." />
```

**<newvar>:** creates one or several numerical variables (using double precision).

Values ***true*** and ***false*** are stored as 1 and 0.

The variables are created in order so previous variables can be used in the following ones.

E.g.: `var1=10, var2=25.5, var3=var1+var2, var4=true`

**<newvarstr>:** creates one or several text variables.

**<newvarcte>** and **<newvarstrcte>:** creates variables (numerical and text) as constants, so these variables cannot be changed.

**_rem** attribute is used for comments and **_print** equal to true (or not zero) shows the values stored in the variables on the screen.

# XML PROGRAMMING STYLE

**Access to the general case constants such as gravity, dp or h for use in programming.**

GenCase creates some variables automatically (as constants) according to values in the XML file, so the user-defined variables can be created from the previous ones. The available constants are:

- **`Gravity_x, Gravity_y, Gravity_z, Rhop0`**
  Gravity values and reference density of the fluid are loaded at the beginning,
  before evaluating <predefinition> section.

- **`Dp`**
  Initial inter-particle distance loaded from XML and after evaluating <predefinition> section.
  Dp value can be modified using variables in <predefinition>.

- **`PosMin_x, PosMin_y, PosMin_z`**
  **`PosMax_x, PosMax_y, PosMax_z`**
  Domain limits loaded or calculated from XML file.

- **`CaseName`**
  **`Data2D, Data2DPosy`**
  **`H, MassBound, MassFluid`**
  Constant variables calculated automatically starting from previous constants and configuration in XML file. These constants are created before evaluating the drawing command lists.

# XML PROGRAMMING STYLE

The user-defined variables can be used in most of the drawing commands and the **different sections** of the XML for **GenCase and DualSPHysics** programs.

```
<mainlist>
    <newvarstr filestl="cube.stl"/>
    <newvar inix="0.2" iniz="0.1"/>
    <move x="#inix" y="0" z="#iniz" />
    <drawbox>
        <boxfill>bottom</boxfill>
        <point x="#PosMin_x" y="0" z="#PosMin_z" />
        <size x="#Dp*20" y="1" z="#Dp*8" />
    </drawbox>
    <drawfilestl file="$[filestl]" if="Dp<=0.1" />
    <runlist name="DrawShape" times="#10/inix" />
</mainlist>
...
<initials>
    <velocity mkfluid="1" x="0" y="0" z="#-Gravity_z*2"/>
</initials>
...
<floatings>
    <floating mkbound="2">
        <center x="#inix" y="0" z="#iniz" />
    </floating>
</floatings>
```

The use of **numerical variables or expressions** must start with **#** in order to be recognized and evaluated.
Numerical or logical expressions can be used in the attribute **`if`** recognized by all drawing commands.

The use of **text variables** or expressions must start with **$** and the name of the variable **in square brackets**.

User-defined variables can be used in the following sections of XML: **<geometry>**, **<initials>**, **<floatings>**, **<properties>**, **<motion>**, **<normals>**, **<special>** and **<parameters>**

# XML PROGRAMMING STYLE

The numerical or logical expressions can use: values, variables, mathematical operators, comparison operators, logical operators, parenthesis and functions.

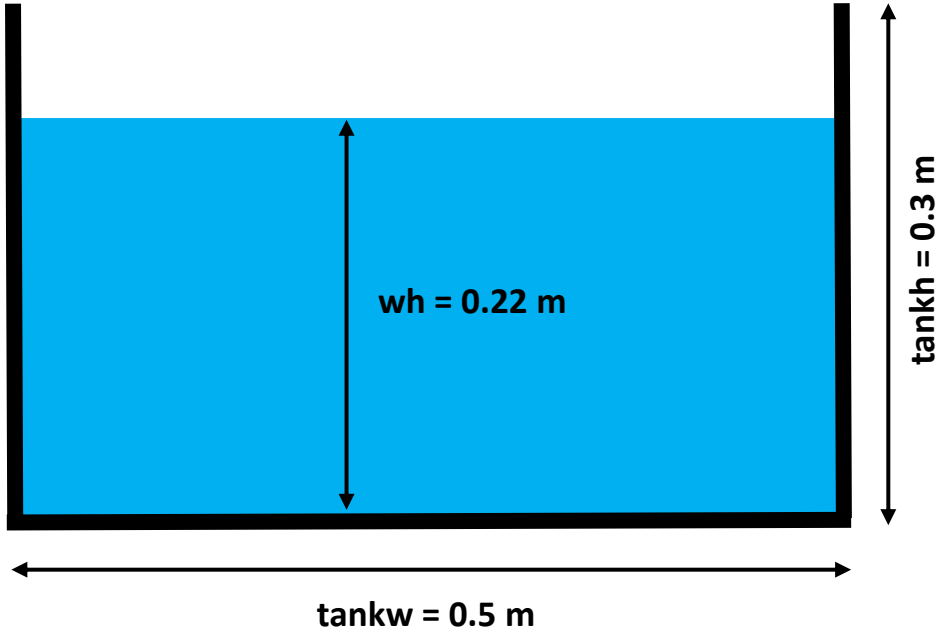
- Mathematical operators: **+**, **-**, *****, **/**
- Comparison operators: **==** (equal), **!=** (not equal), **<**, **>**, **<=**, **>=**
- Logical operators: **!** (not), **||** (or), **@@** (and)
- Functions:

  **pi**(), **e**(),
  **int**(a), **float**(a), **abs**(a), **floor**(a), **ceil**(a), **round**(a), **fmod**(a,b), **fmodr**(a,b)=round(fmod(a,b)),
  **min**(a,b), **min**(a,b,c) , **min**(a,b,c,d), **max**(a,b), **max**(a,b,c), **max**(a,b,c,d),
  **sqrt**(a), **exp**(a), **log**(a), **log10**(a), **pow**(a,b),
  **sin**(radians), **cos**(radians), **tan**(radians), **sindg**(degrees), **cosdg**(degrees), **tandg**(degrees),
  **sinh**(a), **cosh**(a), **tanh**(a), **asin**(a), **acos**(a), **atan**(a),
  **randinit**(seed), **random**(), //to initialise the random generator and to obtain random numbers.
  **eval**(condition, v1, v2), //returns v1 when condition is true or v2 in the other case.
  **wavelength**(gravity, depth, waveperiod) //returns wave length.

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.

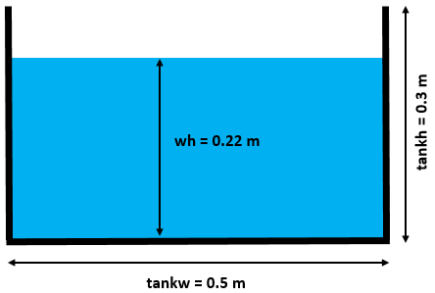
Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m



# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.

Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m

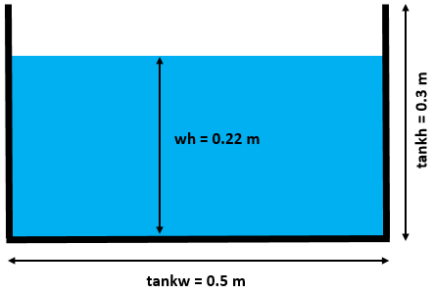



**XML for DBC:**

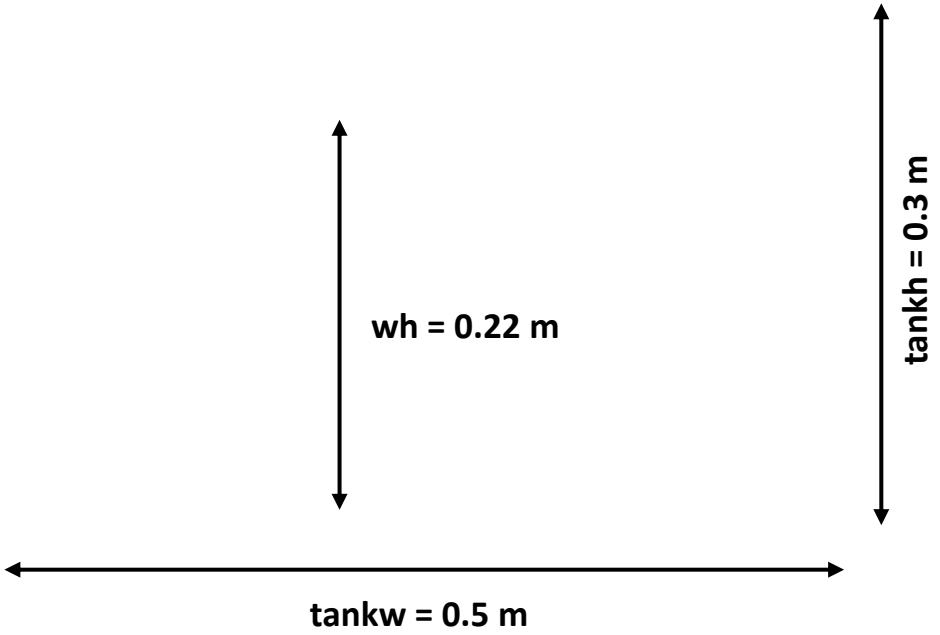
```
<!-- User-defined variables  -->
<newvarcte tankw="0.5" tankh="0.3" _rem="Tank size"/>
<newvarcte wh="0.22" _rem="Water level"/>

<!-- Tank particles -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh" />
</drawbox>

<!-- Fluid particles-->
<setmkfluid mk="0" />
<fillbox x="0" y="0" z="#Dp*2">
    <modefill>void</modefill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#wh"/>
</fillbox>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m

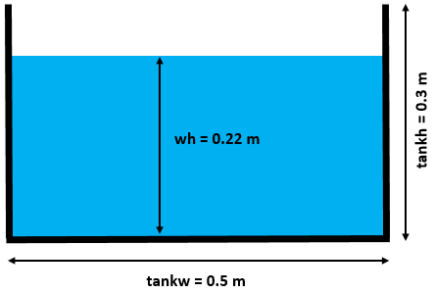



**XML for DBC:**

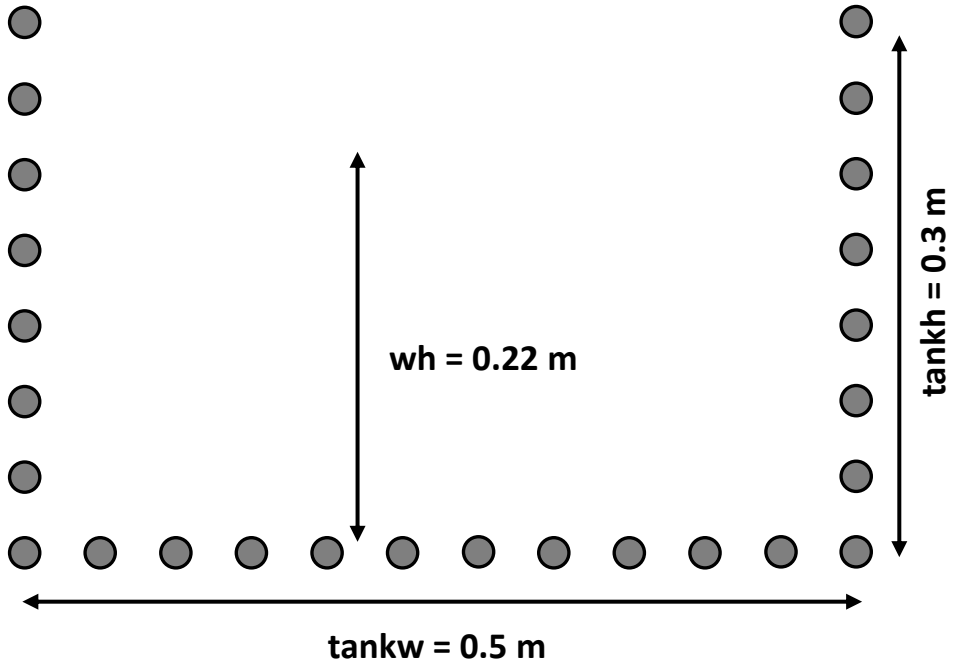
```
<!-- User-defined variables  -->
<newvarcte tankw="0.5" tankh="0.3" _rem="Tank size"/>
<newvarcte wh="0.22" _rem="Water level"/>

<!-- Tank particles -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh" />
</drawbox>

<!-- Fluid particles-->
<setmkfluid mk="0" />
<fillbox x="0" y="0" z="#Dp*2">
    <modefill>void</modefill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#wh"/>
</fillbox>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m




**XML for DBC:**

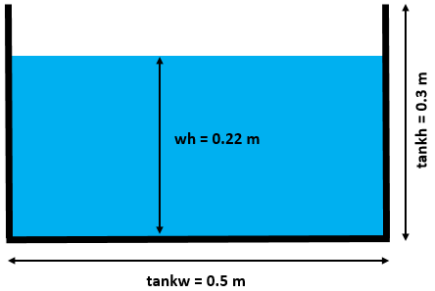
```
<!-- User-defined variables  -->
<newvarcte tankw="0.5" tankh="0.3" _rem="Tank size"/>
<newvarcte wh="0.22" _rem="Water level"/>
```

```
<!-- Tank particles -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh" />
</drawbox>
```

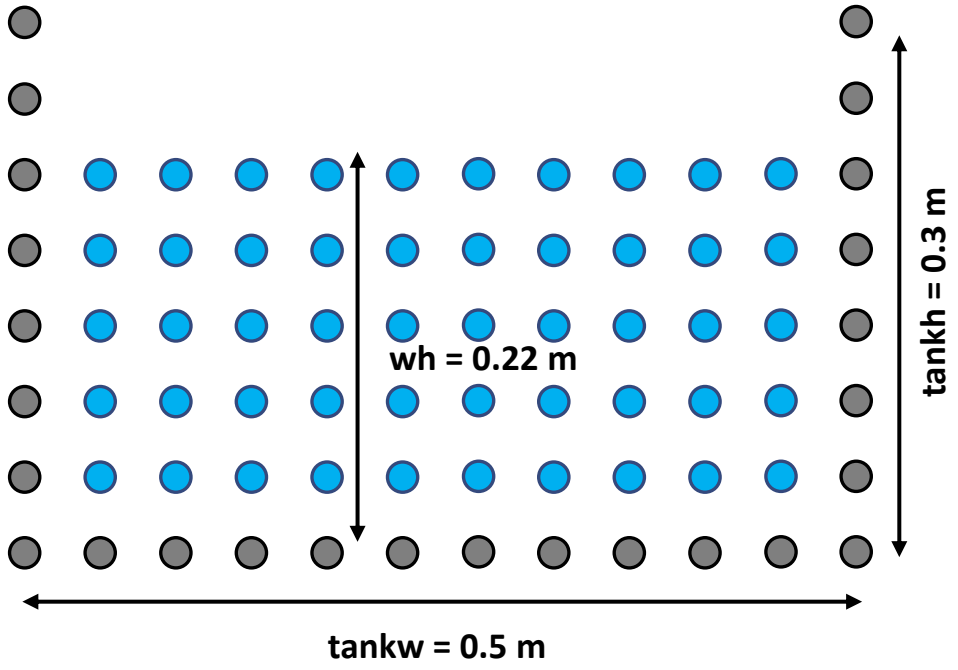
```
<!-- Fluid particles-->
<setmkfluid mk="0" />
<fillbox x="0" y="0" z="#Dp*2">
    <modefill>void</modefill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#wh"/>
</fillbox>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m



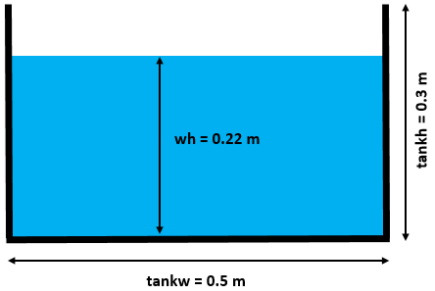

**XML for DBC:**

```
<!-- User-defined variables  -->
<newvarcte tankw="0.5" tankh="0.3" _rem="Tank size"/>
<newvarcte wh="0.22" _rem="Water level"/>
```

```
<!-- Tank particles -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh" />
</drawbox>
```

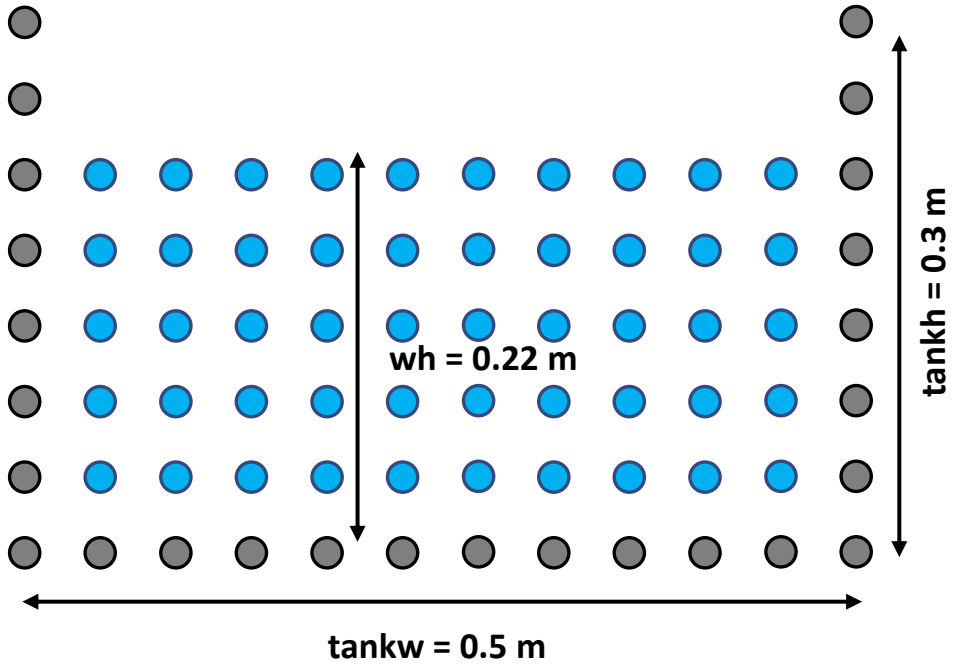
```
<!-- Fluid particles-->
<setmkfluid mk="0" />
<fillbox x="0" y="0" z="#Dp*2">
    <modefill>void</modefill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#wh"/>
</fillbox>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)




The new features in GenCase simplify the initial configuration for mDBC.

Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m



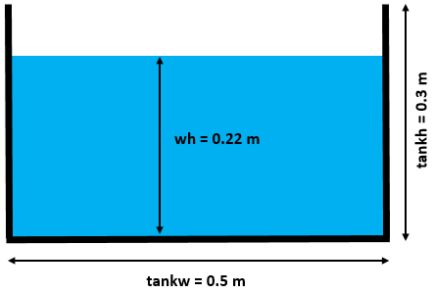

**XML for DBC:**

```
<!-- User-defined variables  -->
<newvarcte tankw="0.5" tankh="0.3" _rem="Tank size"/>
<newvarcte wh="0.22" _rem="Water level"/>

<!-- Tank particles -->
<setmkbound mk="0" />
<d
</drawbox>

<!-- Fluid particles-->
<setmkfluid mk="0" />
<fillbox x="0" y="0" z="#Dp*2">
    <modefill>void</modefill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#wh"/>
</fillbox>
```

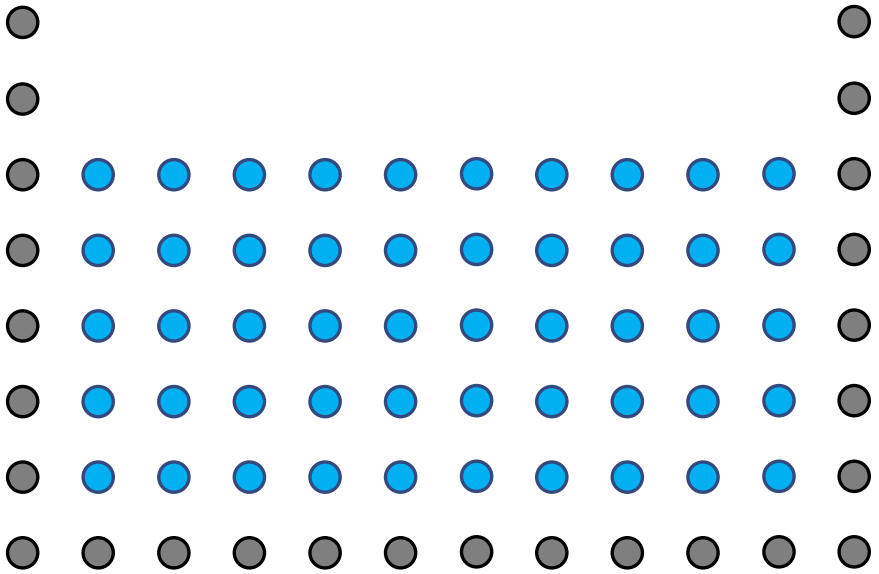
**XML is ready for DBC!!**

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m

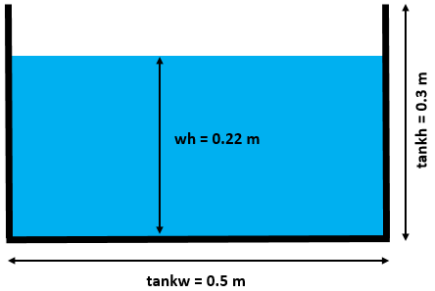


**XML for mDBC:**

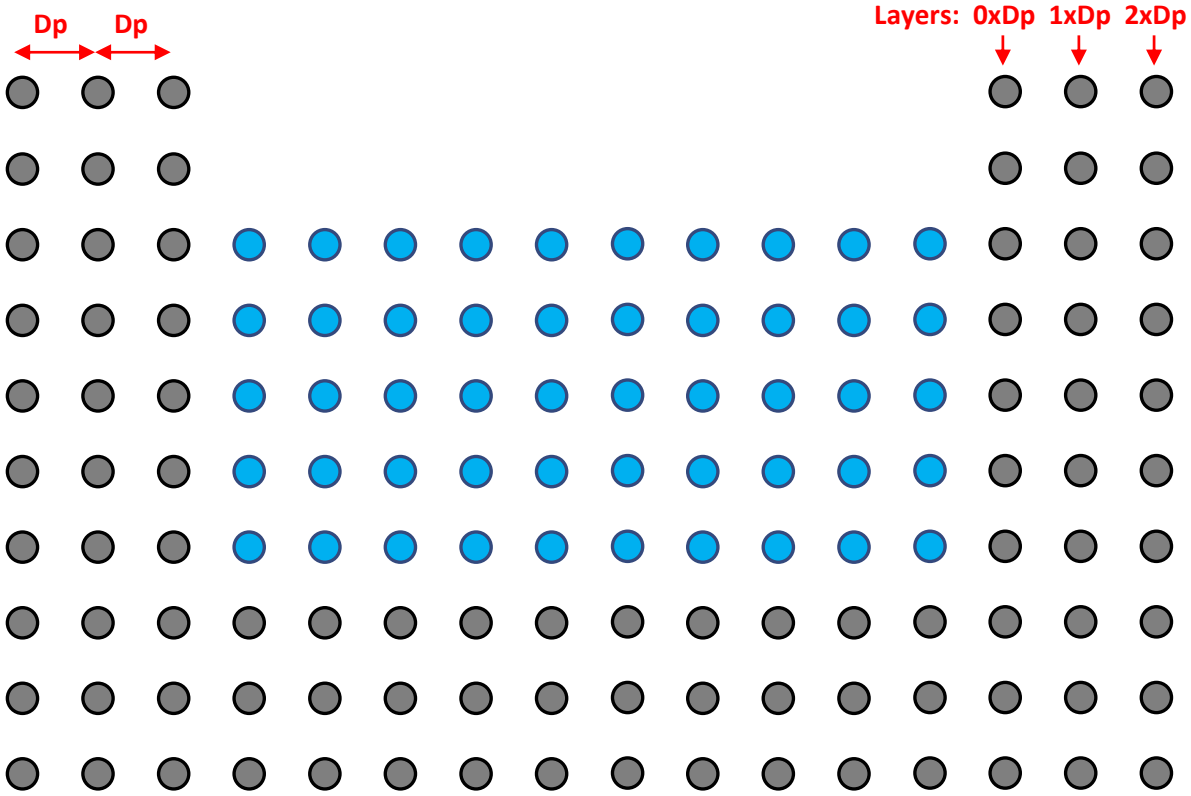
```
<!-- User-defined variables  -->
<newvarcte tankw="0.5" tankh="0.3" _rem="Tank size"/>
<newvarcte wh="0.22" _rem="Water level"/>

<!-- Tank particles -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh" />
    <layers vdp="0,1,2" />
</drawbox>

<!-- Fluid particles-->
<setmkfluid mk="0" />
<fillbox x="0" y="0" z="#Dp*2">
    <modefill>void</modefill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#wh"/>
</fillbox>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m



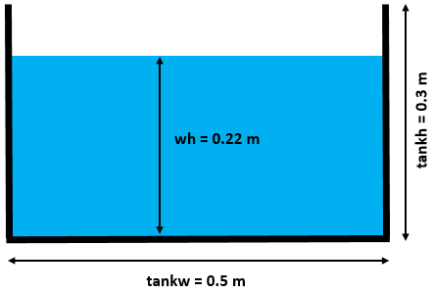

**XML for mDBC:**

```
<!-- User-defined variables  -->
<newvarcte tankw="0.5" tankh="0.3" _rem="Tank size"/>
<newvarcte wh="0.22" _rem="Water level"/>

<!-- Tank particles -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh" />
    <layers vdp="0,1,2" />
</drawbox>

<!-- Fluid particles-->
<setmkfluid mk="0" />
<fillbox x="0" y="0" z="#Dp*2">
    <modefill>void</modefill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#wh"/>
</fillbox>
```

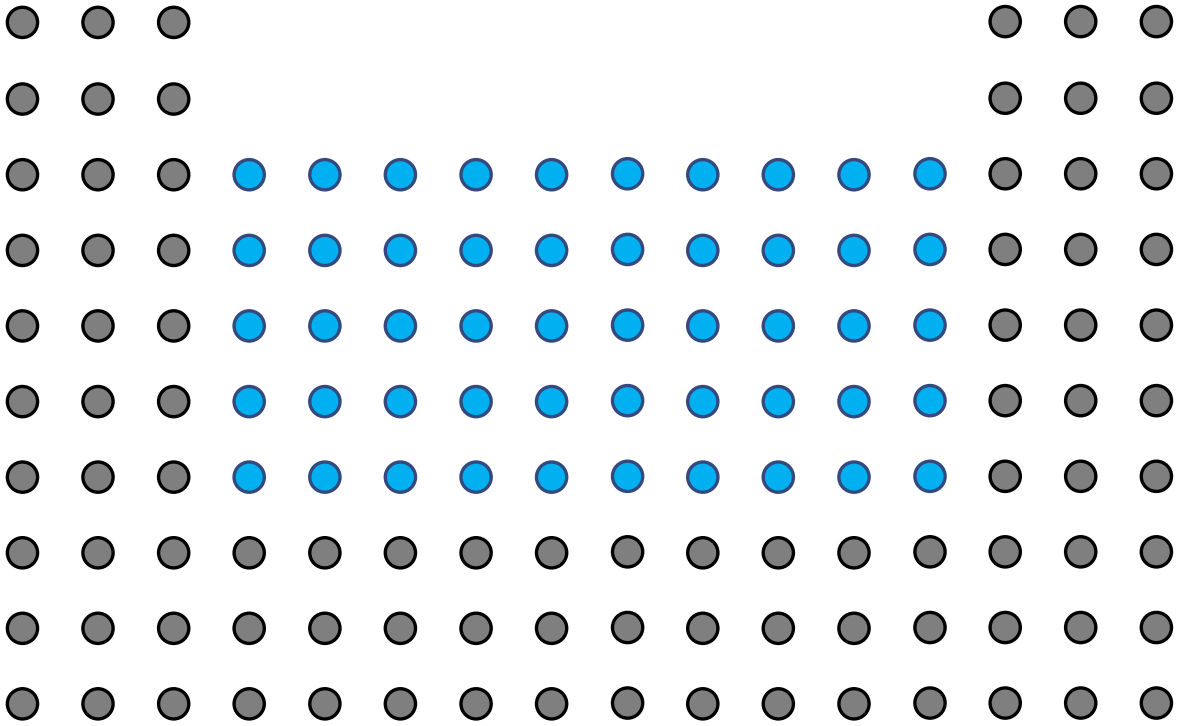
**mDBC needs sevaral boudary layers, so option <layers> is used.**

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m



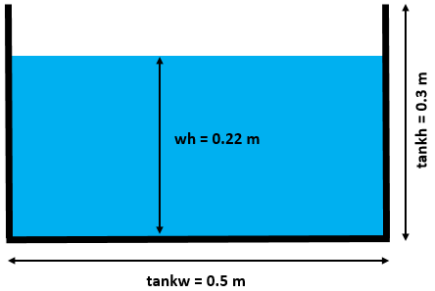
**XML for mDBC:**

```
<setshapemode>actual|bound</setshapemode>
<setactive drawpoints="false" />

<!-- Tank geometry -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh"/>
    <layers vdp="-0.5" />
</drawbox>

<!-- Save geometry file for normals -->
<shapeout file="hdp"/>

<setactive drawpoints="true"/>
```

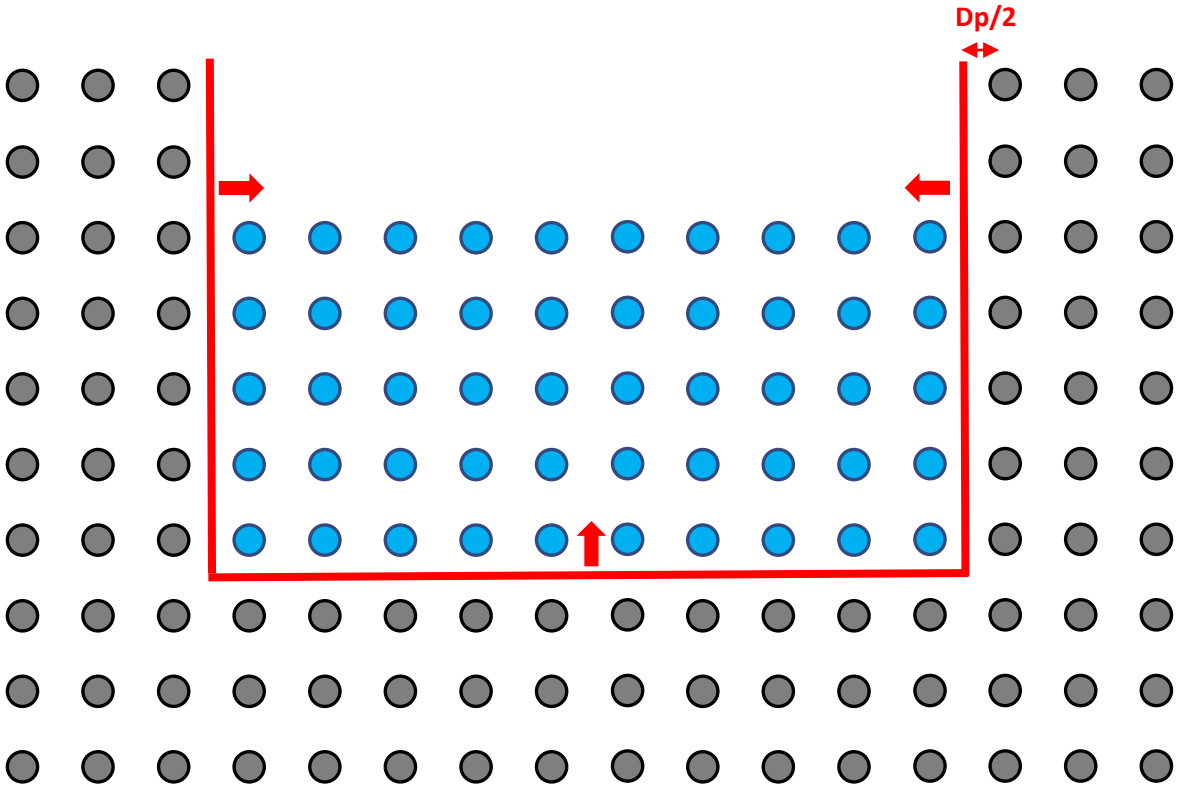
**Disables particle drawing**

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m




**XML for mDBC:**

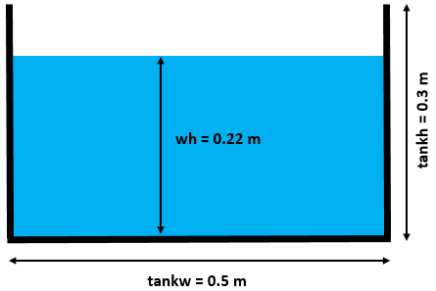
```
<setshapemode>actual|bound</setshapemode>
<setactive drawpoints="false" />

<!-- Tank geometry -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh"/>
    <layers vdp="-0.5" />
</drawbox>

<!-- Save geometry file for normals -->
<shapeout file="hdp"/>

<setactive drawpoints="true"/>
```

**Disables particle drawing**

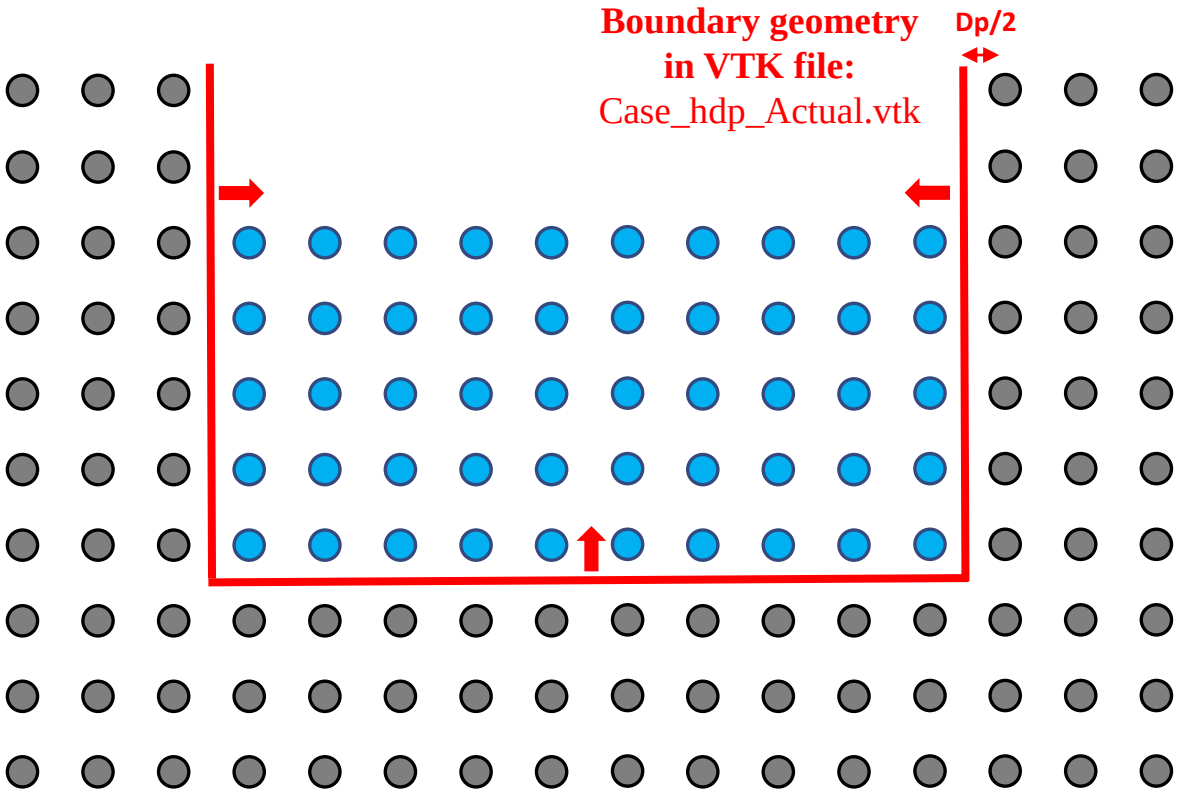
**Creates boundary limit at 0.5xDp**

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m




**XML for mDBC:**

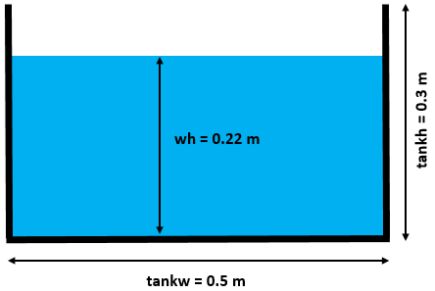
```
<setshapemode>actual|bound</setshapemode>
<setactive drawpoints="false" />

<!-- Tank geometry -->
<setmkbound mk="0" />
<drawbox>
    <boxfill>all^top</boxfill>
    <point x="0" y="-0.1" z="0" />
    <size x="#tankw" y="0.2" z="#tankh"/>
    <layers vdp="-0.5" />
</drawbox>

<!-- Save geometry file for normals -->
<shapeout file="hdp"/>

<setactive drawpoints="tru
```

**Disables particle drawing**

**Creates boundary limit at 0.5xDp**

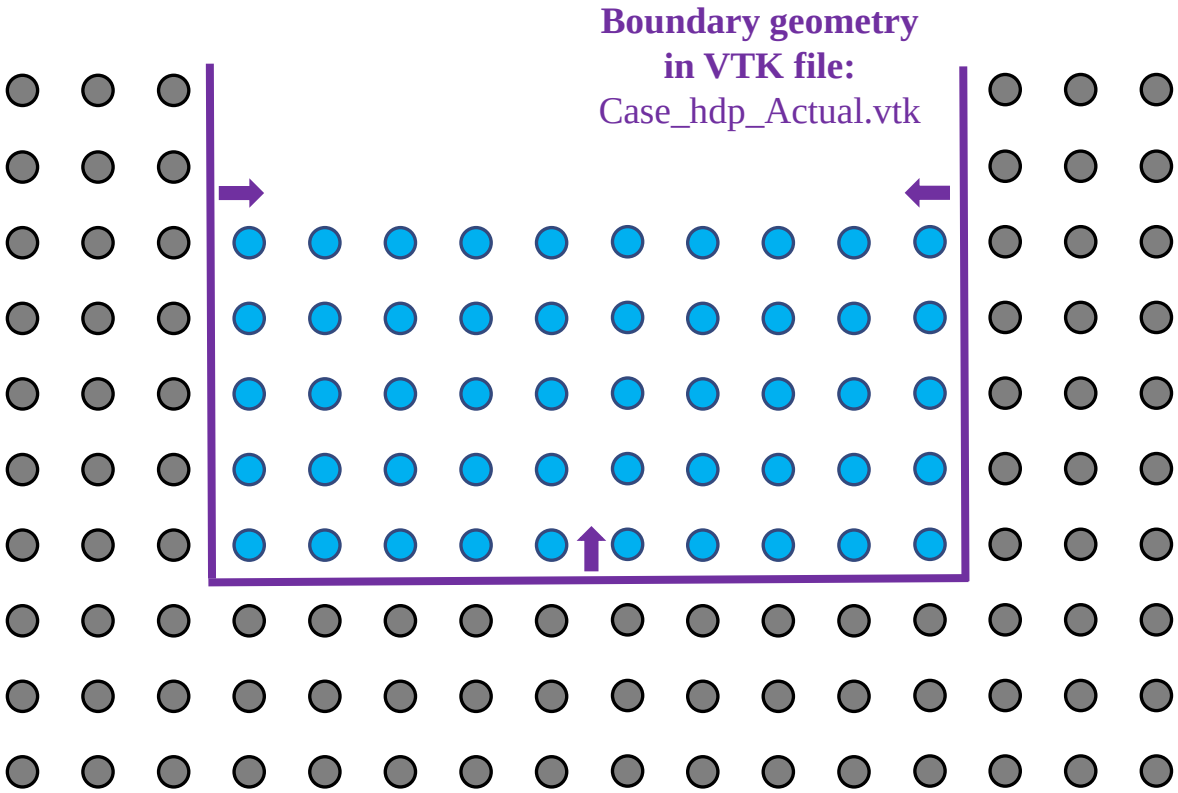
**Creates VKT file with boundary geometry:**
Case_hdp_Actual.vtk

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)




The new features in GenCase simplify the initial configuration for mDBC.

Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m

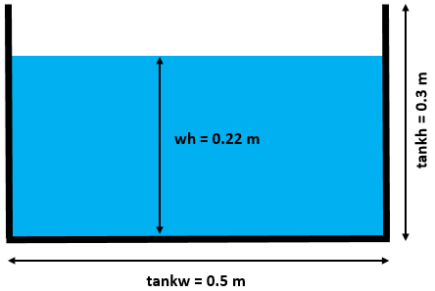



## XML for mDBC:

Section **<normals>** of XML configures the **automatic calculation of normal vectors** for each boundary particle starting from the geometry VTK file cretated before.

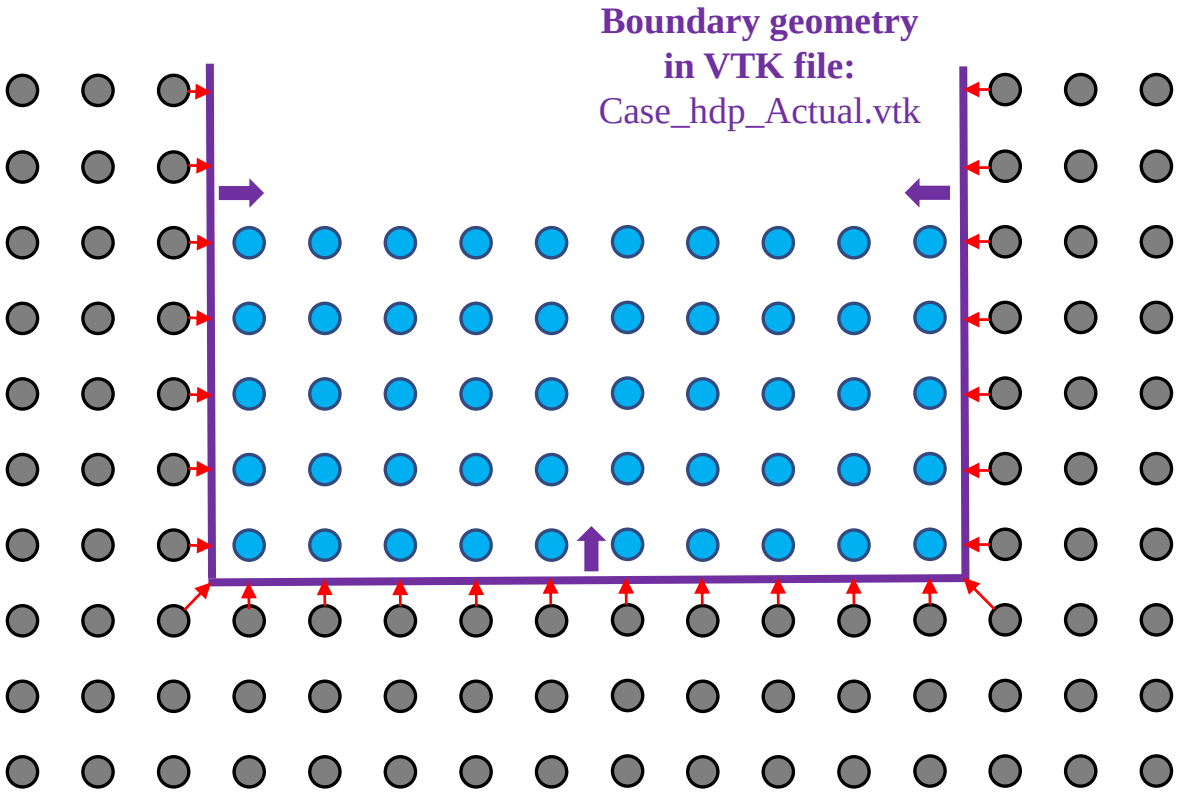
```
    </geometry>
    <normals>
        <distanceh value="3.0" comment="(default=2)"/>
        <geometryfile file="[CaseName]_hdp_Actual.vtk"/>
        <svshapes value="1" comment="(default=0)"/>
    </normals>
</casedef>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)




The new features in GenCase simplify the initial configuration for mDBC.

Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m



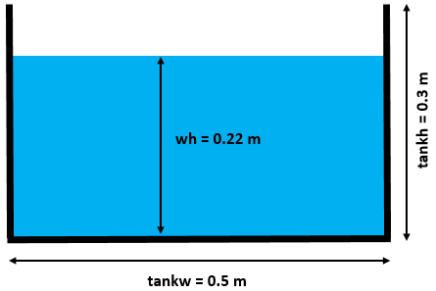

## XML for mDBC:

Section **<normals>** of XML configures the **automatic calculation of normal vectors** for each boundary particle starting from the geometry VTK file cretated before.

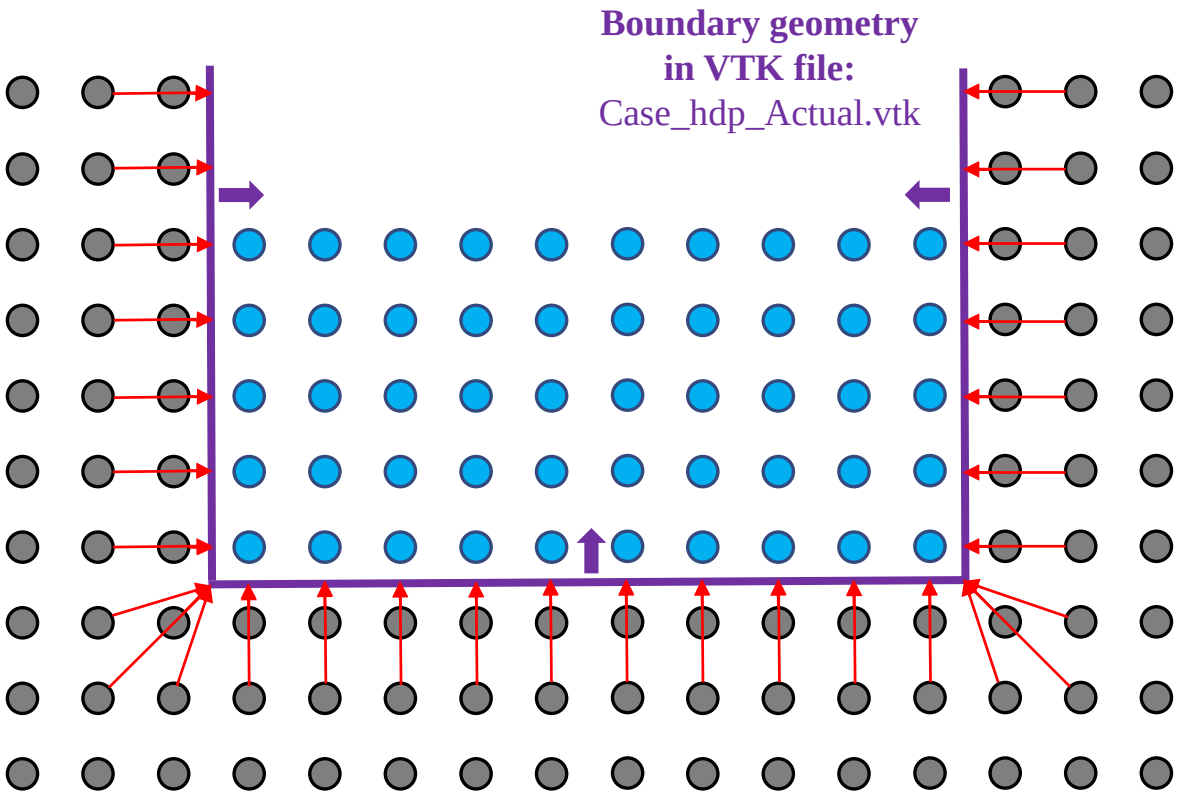
```
    </geometry>
    <normals>
        <distanceh value="3.0" comment="(default=2)"/>
        <geometryfile file="[CaseName]_hdp_Actual.vtk"/>
        <svshapes value="1" comment="(default=0)"/>
    </normals>
</casedef>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m

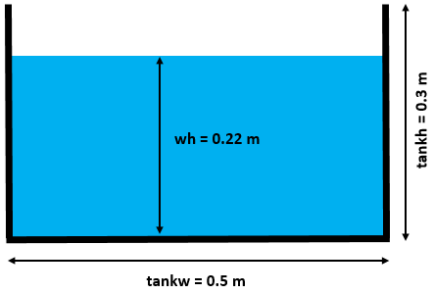



## XML for mDBC:

Section **<normals>** of XML configures the **automatic calculation of normal vectors** for each boundary particle starting from the geometry VTK file cretated before.

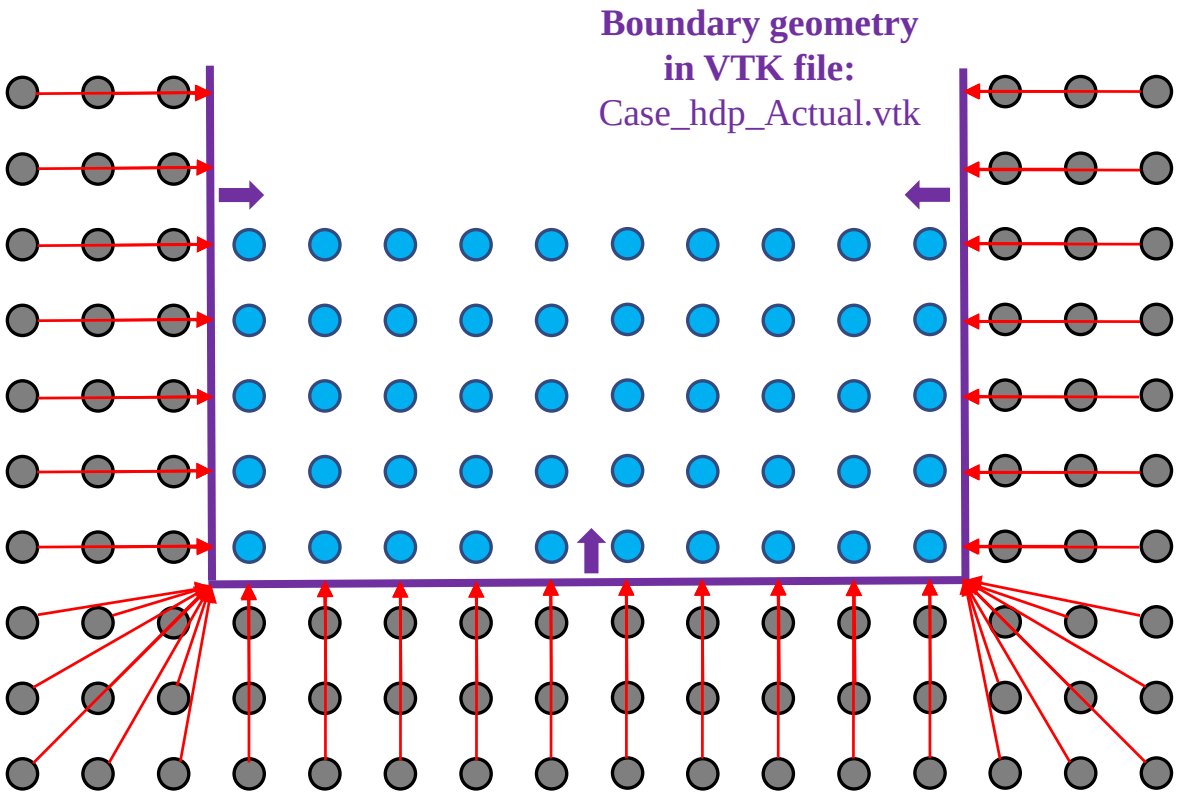
```
    </geometry>
    <normals>
        <distanceh value="3.0" comment="(default=2)"/>
        <geometryfile file="[CaseName]_hdp_Actual.vtk"/>
        <svshapes value="1" comment="(default=0)"/>
    </normals>
</casedef>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)




The new features in GenCase simplify the initial configuration for mDBC.

Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m



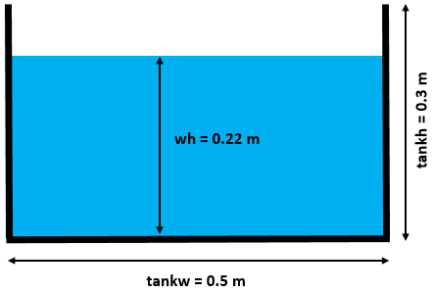

## XML for mDBC:

Section **<normals>** of XML configures the **automatic calculation of normal vectors** for each boundary particle starting from the geometry VTK file cretated before.

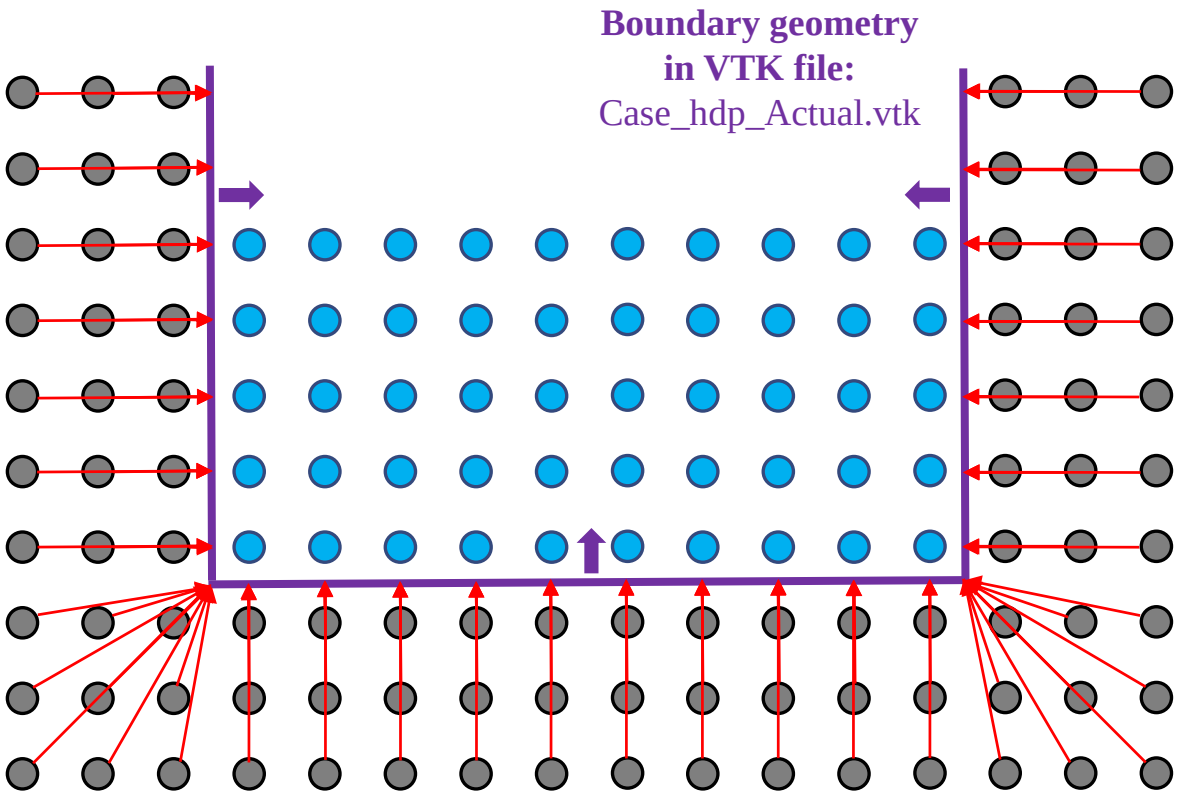
```
    </geometry>
    <normals>
        <distanceh value="3.0" comment="(default=2)"/>
        <geometryfile file="[CaseName]_hdp_Actual.vtk"/>
        <svshapes value="1" comment="(default=0)"/>
    </normals>
</casedef>
```

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

The new features in GenCase simplify the initial configuration for mDBC.




Example: Tank with dimensions 0.5m x 0.3m
and water height 0.22m




## XML for mDBC:

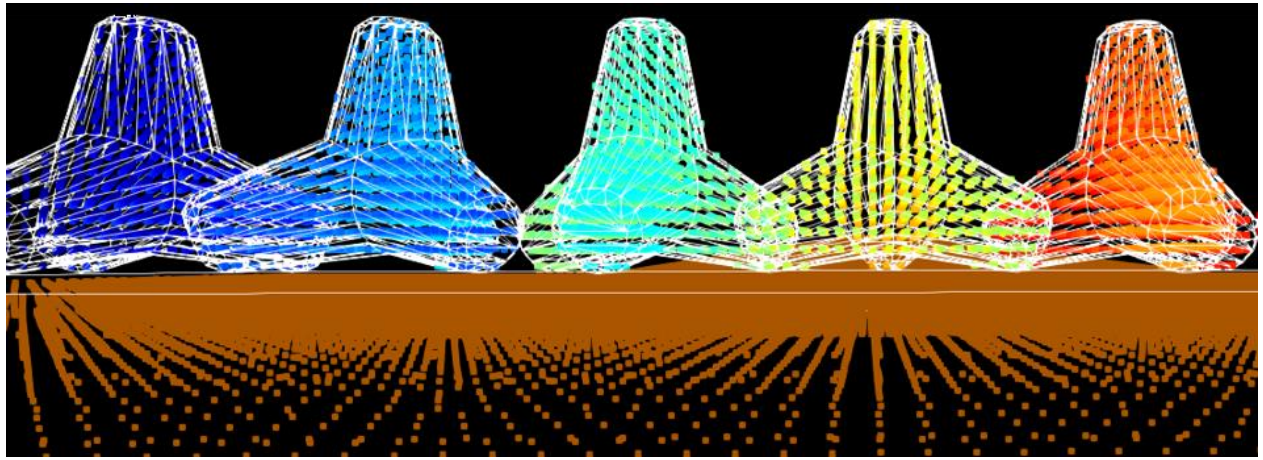
Section **<normals>** of XML configures the **automatic calculation of normal vectors** for each boundary particle starting from the geometry VTK file cretated before.

```
    </geometry>
    <normals>
        <distanceh value="3.0" comment="(default=2)"/>
        <geometryfile file="[CaseName]_hdp_Actual.vtk"/>
        <svshapes value="1" comment="(default=0)"/>
    </normals>
</casedef>
```

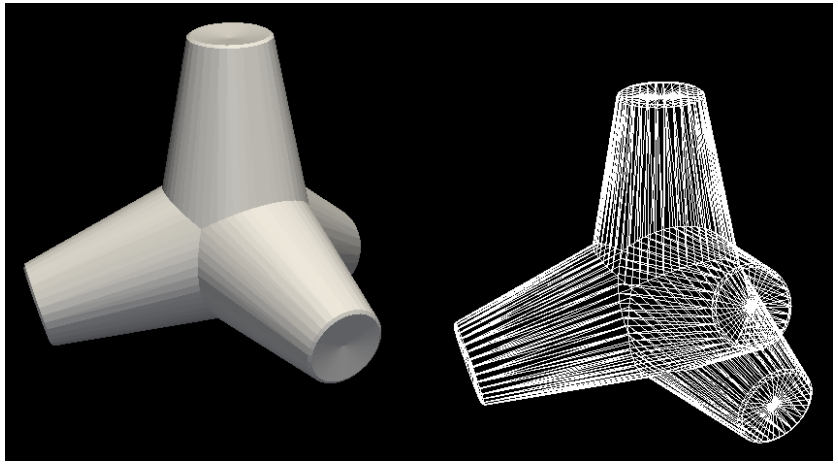
And the **normal vector** of each **boundary particle** is used by DualSPHysics to define the position of **ghost nodes** in mDBC correction.

# AUTOMATIC CALCULATION OF NORMALS (for mDBC)

**Boundary particles**

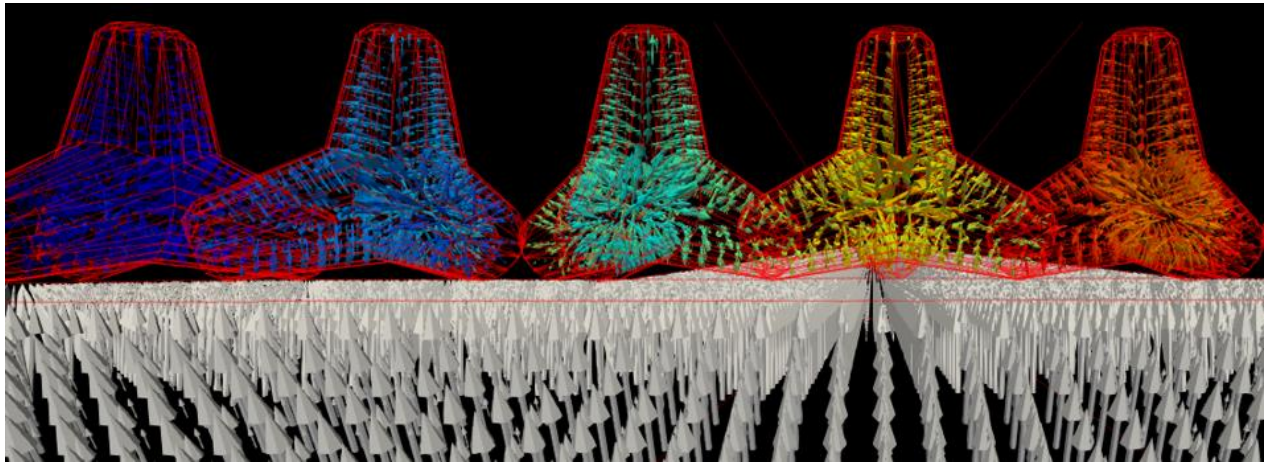


**Tetrapods geometry**



The same technique is valid for **complex 3-D objects** such as Tetrapods.

**Boundary with normal for mDBC**



# OUTLINE

- **Introduction**
  - **New boundary conditions: mDBC**
    - Motivation: DBC drawbacks
    - Fluid properties from ghost nodes
    - mDBC vs. DBC
    - mDBC requirements
      - **New options on Preprocessing**
        - Free drawing mode (without lattice limitation)
        - Automatic multi-layer drawing
        - XML programming style
        - Automatic calculation of normals (for mDBC)
          - **Conclusions & future improvements**

# CONCLUSIONS

- DualSPHysics v5.0 includes important improvements in formulation, pre-processing and post-processing that allow addressing applications impossible before.

**New reference paper for DualSPHysics v5**


J.M. Domínguez, G. Fourtakas, C. Altomare, R.B. Canelas, A. Tafuni, O. García-Feal, I. Martínez-Estévez, A. Mokos, R. Vacondio, A.J.C. Crespo, B.D. Rogers, P.K. Stansby, M. Gómez-Gesteira. 2021.
**DualSPHysics: from fluid dynamics to multiphysics problems.**
Computational Particle Mechanics. **IN PRESS**

# CONCLUSIONS

- DualSPHysics v5.0 includes important improvements in formulation, pre-processing and post-processing that allow addressing applications impossible before.
- DBC are simple, reliable and very versatile to simulate 2D & 3D complex geometries, but high resolution is necessary to minimise some problems.
- mDBC presents significant improvements regarding to DBC:
  - Avoids the gap between boundary and fluid particles.
  - Provides physical density/pressure values in boundary particles.
- The initial condition for mDBC is much more complicated because several boundary layers and normals are necessary.
- mDBC should be used when DBC problems are important for your application and also mDBC can focus only on the area of interest.
- New options in pre-processing simplify the initial configuration for mDBC.
- New free drawing mode without cubic lattice and programming functions in XML open up a world of possibilities.

J.M. Domínguez, G. Fourtakas, C. Altomare, R.B. Canelas, A. Tafuni, O. García-Feal, I. Martínez-Estévez, A. Mokos, R. Vacondio, A.J.C. Crespo, B.D. Rogers, P.K. Stansby, M. Gómez-Gesteira. 2021. **DualSPHysics: from fluid dynamics to multiphysics problems.** Computational Particle Mechanics. **IN PRESS**

# WORK IN PROGRESS

J.M. Domínguez, G. Fourtakas, C. Altomare, R.B. Canelas, A. Tafuni, O. García-Feal, I. Martínez-Estévez, A. Mokos, R. Vacondio, A.J.C. Crespo, B.D. Rogers, P.K. Stansby, M. Gómez-Gesteira. 2021. **DualSPHysics: from fluid dynamics to multiphysics problems.** Computational Particle Mechanics. **IN PRESS**

## On mDBC:

- Free-slip condition for mDBC (only no-slip is now available).
- Floating bodies with mDBC.
- New technique to avoid minor penetration issues.

## On pre-processing:

- New options on using external geometries for mDBC.
- Free drawing mode for complex external geometries.
- Free drawing mode for fluid filling operations.


# News on pre-processing tool and boundary conditions

JOSÉ M. DOMÍNGUEZ

jmdominguez@uvigo.es

**UNIVERSIDADE DE VIGO**