Immersed Boundary Methods for Simulating Fluid-Structure Interaction

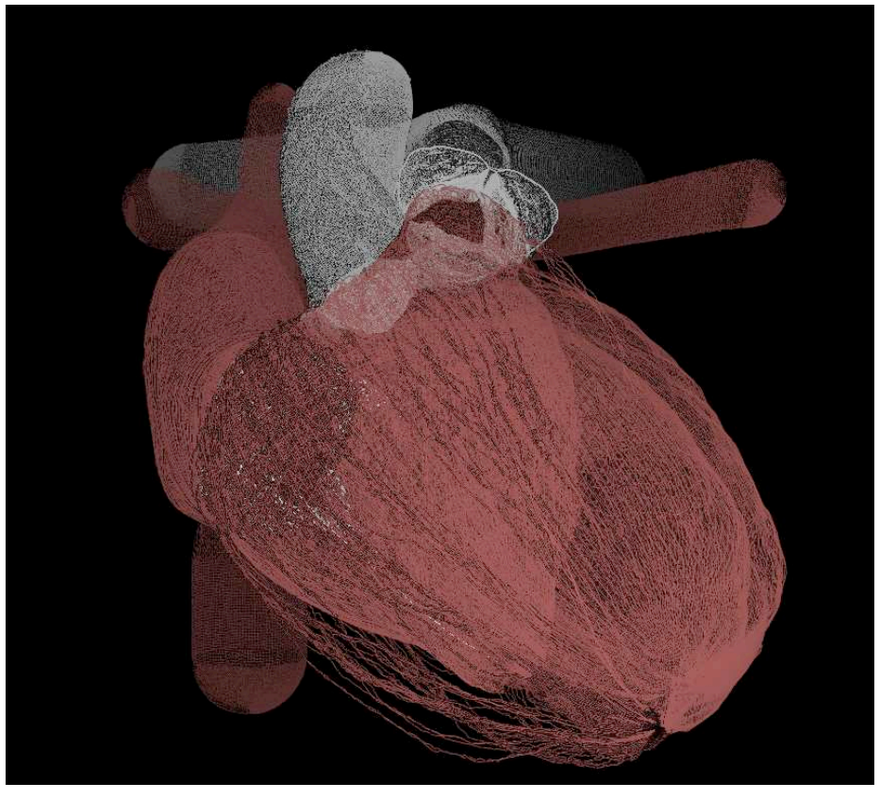
Boyce Griffith
Leon H. Charney Division of Cardiology-Department of Medicine
Computational Biology Training Program-Sackler Institute
Affiliated Faculty-Center for Health Informatics and Bioinformatics
New York University School of Medicine
email: boyce.griffith@nyumc.org
web: http://www.cims.nyu.edu/~griffith

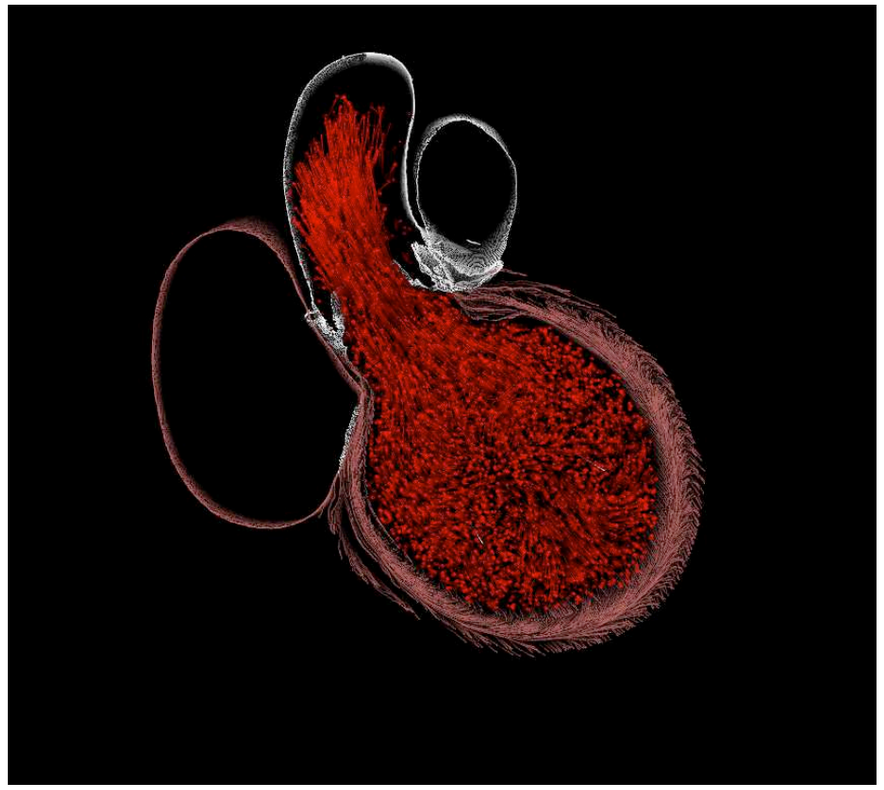
Spring Workshop on Nonlinear Mechanics
Xi'an Jiaotong University, China
2011.04.06

NYU School of Medicine

American Heart Association
Learn and Live

NSF





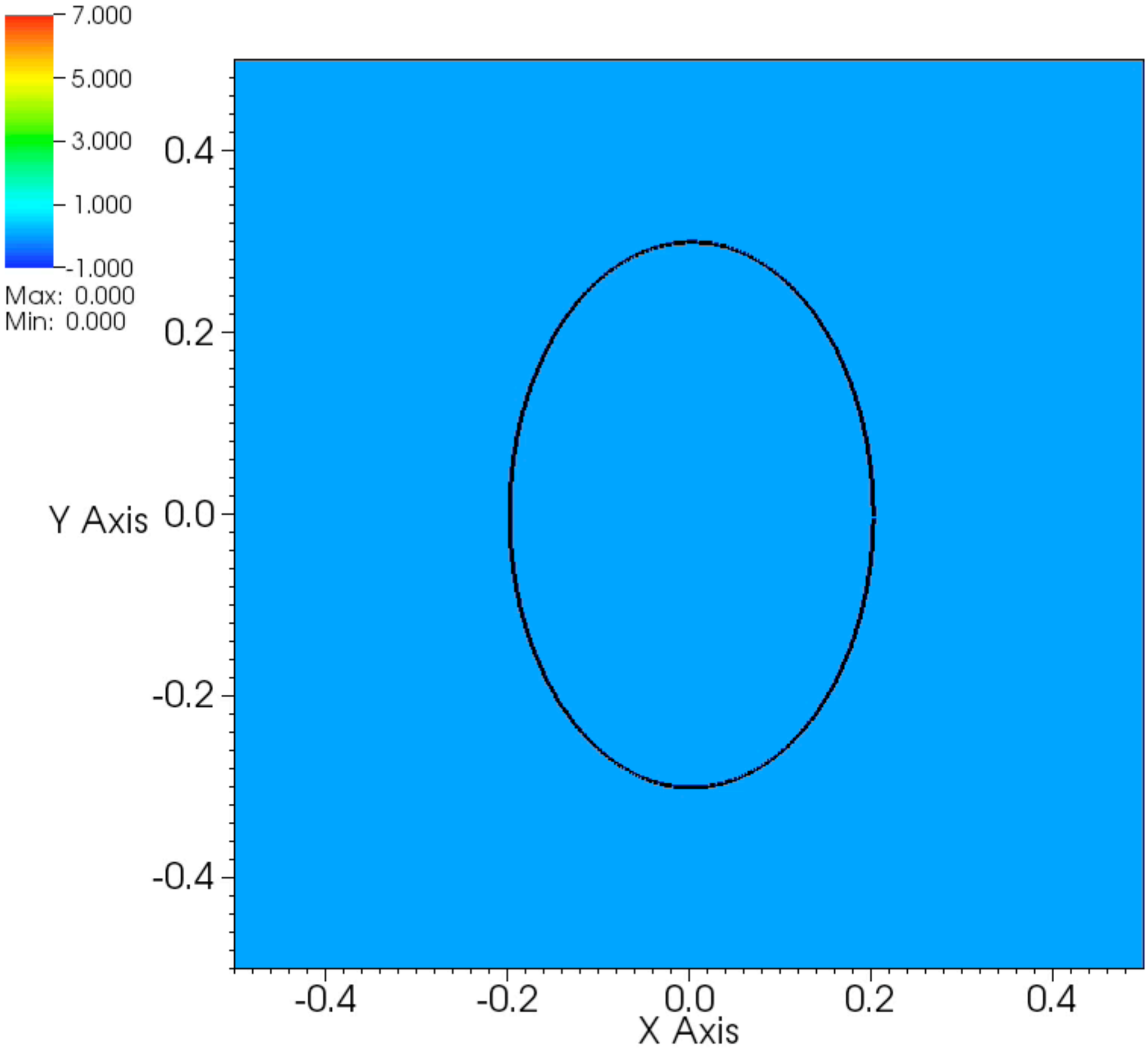
The *immersed boundary (IB) method* is an approach to modeling fluid-structure interaction (FSI) that we (and others) use to simulate cardiac and cardiovascular fluid dynamics and other problems of biological fluid dynamics.

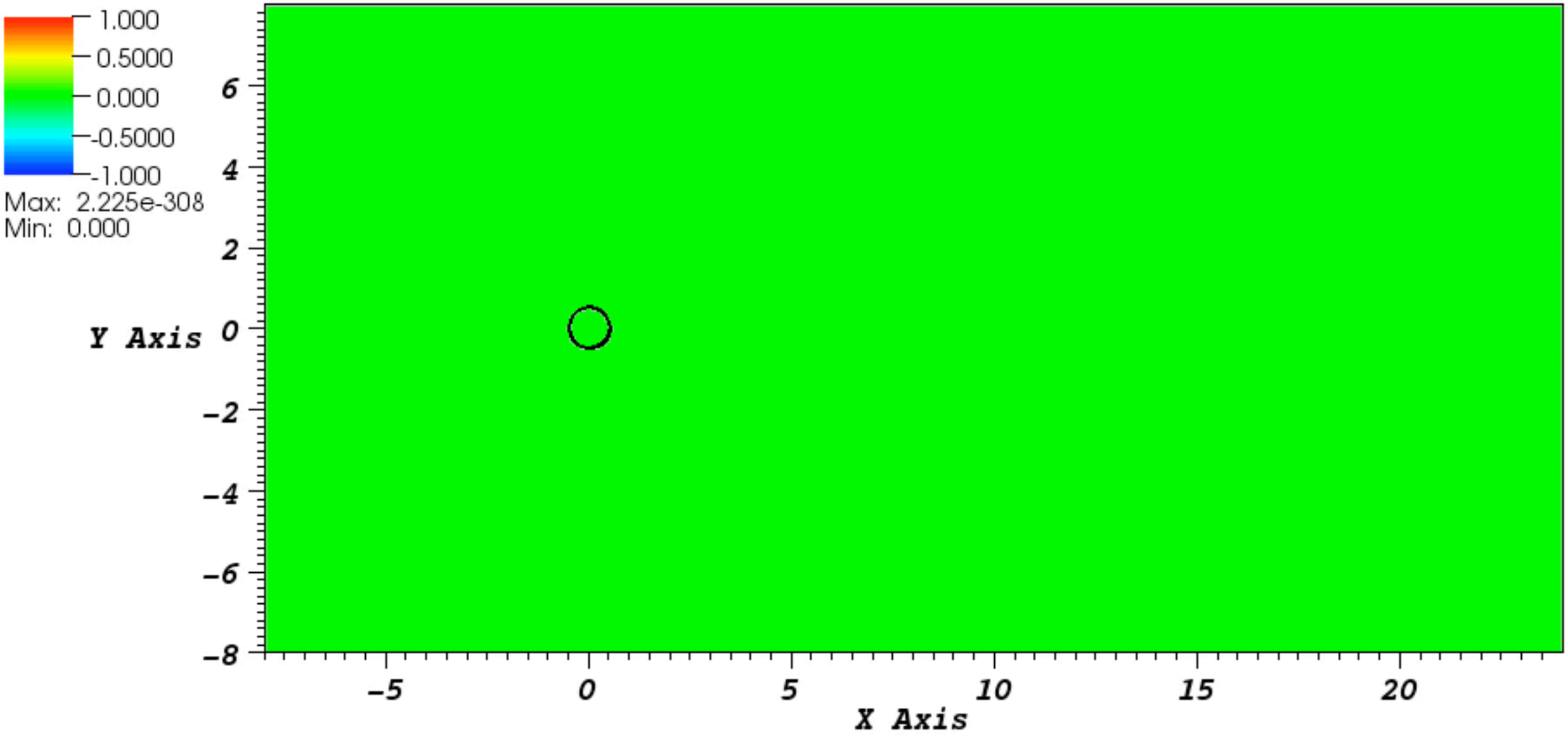
Over the last decade, the IB method also has become increasingly used for more “standard” engineering problems.

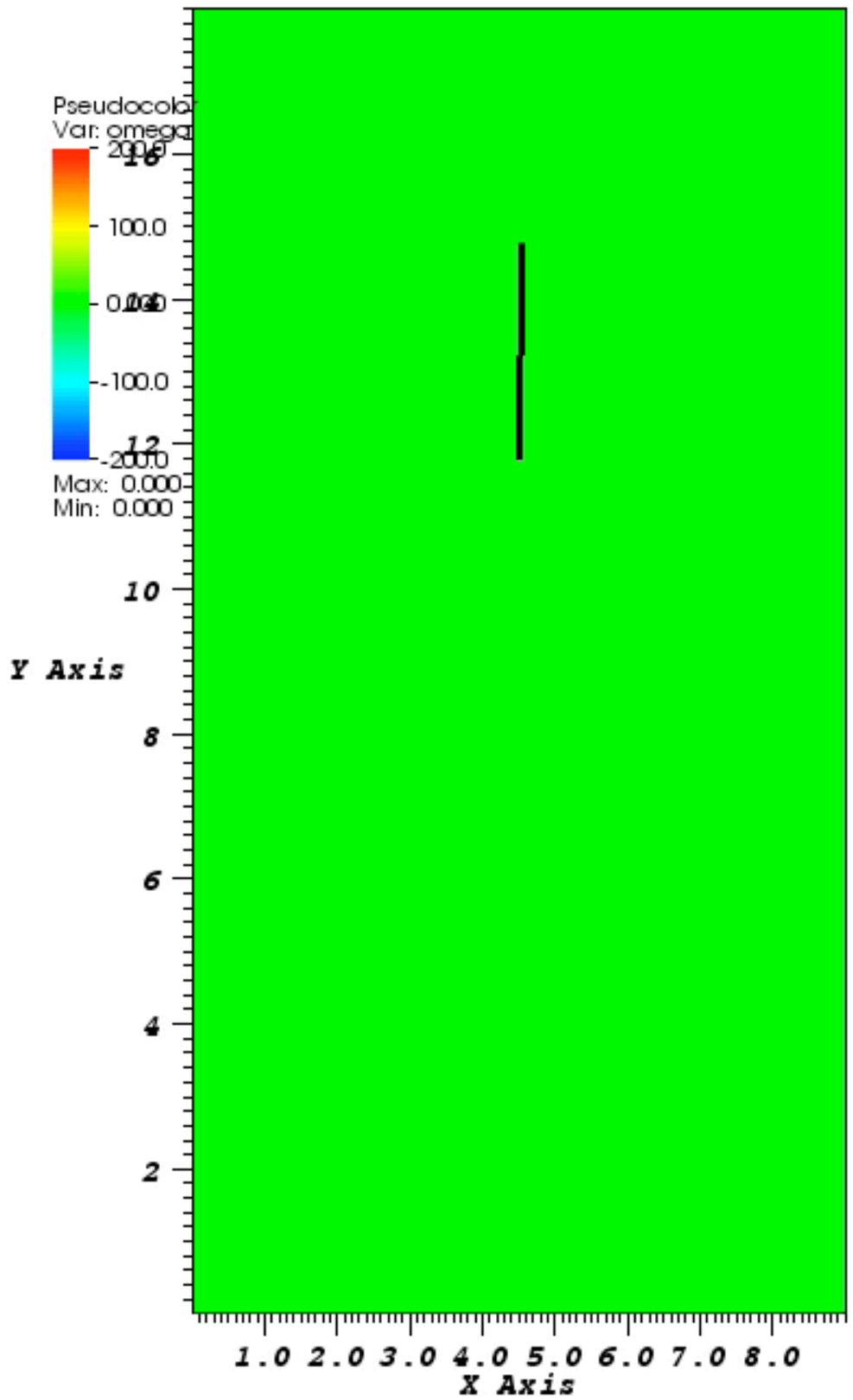
Why study cardiovascular fluid dynamics?

- Cardiovascular diseases have been the leading causes of death in Western countries for more than a century. . .
- And they are now also the leading causes of death in China.
- Cardiovascular diseases kill 2.6 million people in China annually.
- And the prevalence of cardiovascular diseases in China is currently forecast to increase by $\geq$50% between 2010 and 2030.

We are working to develop detailed, realistic models of the heart and vasculature that can be used to develop improved treatments and therapies for patients suffering from cardiac or cardiovascular diseases.


7.000
5.000
3.000
1.000
-1.000
Max: 0.000
Min: 0.000
0.4
0.2
0.0
-0.2
-0.4
Y Axis
-0.4
-0.2
0.0
0.2
0.4
X Axis


1.000
0.5000
0.000
-0.5000
-1.000
Max: 2.225e-308
Min: 0.000
Y Axis
6
4
2
0
-2
-4
-6
-8
-5
0
5
10
15
20
X Axis


Pseudocolor
Var: omega
200.0
100.0
0.000
-100.0
-200.0
Max: 0.000
Min: 0.000
16
12
10
8
6
4
2
Y Axis
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0
X Axis

Eulerian and Lagrangian descriptions

To describe the motion of a fluid, it is generally most convenient to use an *Eulerian* or *spatial* description.

An Eulerian description is one that is in terms of a *fixed* coordinate system.

We shall use the notation $\mathbf{x} = (x, y, z) \in \Omega$ to denote fixed physical coordinates. Then, for instance, we shall describe the fluid motion in terms of the fluid velocity field $\mathbf{u}(\mathbf{x}, t) = (u(\mathbf{x}, t), v(\mathbf{x}, t), w(\mathbf{x}, t))$, and the fluid pressure field $p(\mathbf{x}, t)$.

To describe the elasticity of a structure, it is generally most convenient to use a *Lagrangian* or *material* description.

A Lagrangian description is one that is in terms of material coordinates that are *attached to* and *move with* the structure.

We shall use the notation $\mathbf{s} = (q, r, s) \in U$ to denote the material coordinate system, and the notation $\mathbf{X}(\mathbf{s}, t) \in \Omega$ to denote the physical position of material point $\mathbf{s}$ at time t.

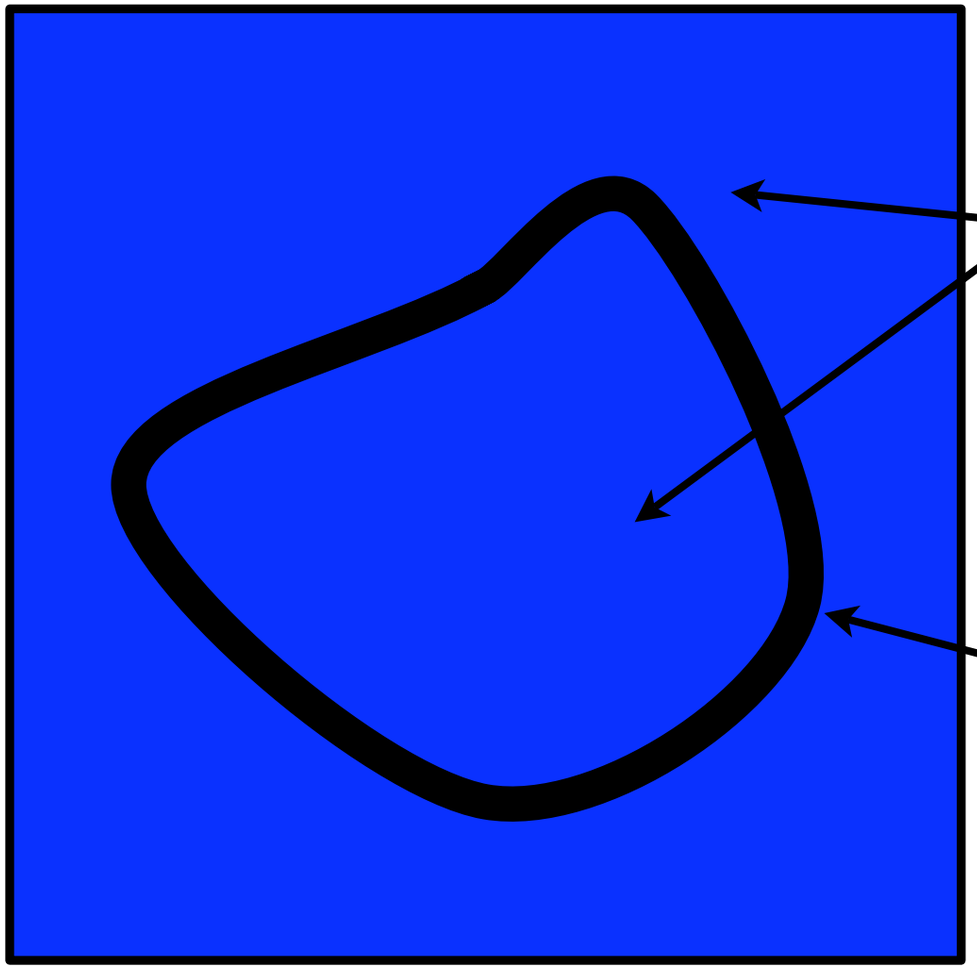
Eulerian and Lagrangian descriptions

Remark: A purely Lagrangian description *is not* especially useful for fluid dynamics, because fluid particles that are initially close to one another do not generally remain so.

On the other hand, a Lagrangian description *is* well-suited for elasticity problems because elastic forces generally are present that act to keep nearby material points close to one another.

The IB approach to fluid-structure interaction

- Use an Eulerian description of the viscous incompressible fluid;
- Use a Lagrangian description of the immersed elastic structure; and
- Use integral transforms with delta-function kernels to mediate interaction between Lagrangian and Eulerian variables.



$\mathbf{x} = (x, y, z)$ are Cartesian (physical) coordinates;
$\mathbf{u}(\mathbf{x}, t)$ is the Eulerian (fluid) velocity; and
$p(\mathbf{x}, t)$ is the Eulerian (fluid) pressure.

$\mathbf{s} = (q, r, s)$ are Lagrangian (material) coordinates;
$\mathbf{X}(\mathbf{s}, t)$ is the position of $\mathbf{s}$ at time t; and
$\mathbf{F}(\mathbf{s}, t)$ is the Lagrangian elastic force density.

The equations of motion for the fluid-structure system

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}(\mathbf{x},t)+\mathbf{u}(\mathbf{x},t)\cdot\nabla\mathbf{u}(\mathbf{x},t)\right)=-\nabla p(\mathbf{x},t)+\mu\nabla^2\mathbf{u}(\mathbf{x},t)+\mathbf{f}(\mathbf{x},t),$$

$$\nabla\cdot\mathbf{u}(\mathbf{x},t)=0,$$

$$\mathbf{f}(\mathbf{x},t)=\int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s},$$

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t)=\int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x},$$

$$\mathbf{F}(\mathbf{s},t)=\boldsymbol{\mathcal{F}}[\mathbf{X}(\mathbf{s},t);\mathbf{s},t],$$

in which

- $\mathbf{x}=(x,y,z)\in\Omega$ are Cartesian (physical) coordinates;
- $\mathbf{s}=(q,r,s)\in U$ are Lagrangian (material) coordinates;
- $\mathbf{u}(\mathbf{x},t)$ is the Eulerian material velocity field;
- $\mathbf{f}(\mathbf{x},t)$ is the Eulerian elastic force density;
- $\mathbf{X}(\mathbf{s},t)$ is the position of material point $\mathbf{s}$ at time t; and
- $\mathbf{F}(\mathbf{s},t)$ is the Lagrangian elastic force density.

The boundary advection equation

The material points of the structure move according to:

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}.$$

The defining property of the Dirac delta function therefore implies that:

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x} = \mathbf{u}(\mathbf{X}(\mathbf{s},t),t).$$

This equation states that the immersed structure moves at the local fluid velocity, i.e., it sticks to the fluid.

This equation can therefore be interpreted as a formulation of the *no-slip condition*, which generally appears as a boundary condition in the equations of fluid dynamics. Here, however, instead of appearing as a *constraint* on the fluid motion, the no-slip condition *determines* the motion of the immersed structure.

Equivalence of $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{F}(\mathbf{s}, t)$ as densities

Let $V \subseteq U$ be an arbitrary region of the Lagrangian coordinate space, and let $V_t = \mathbf{X}(V, t)$ be the current configuration of material region V at time t. Then,

$$\begin{aligned}\int_{V_t} \mathbf{f}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} &= \int_{V_t} \int_U \mathbf{F}(\mathbf{s}, t)\, \delta(\mathbf{x} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}\, \mathrm{d}\mathbf{x} \\ &= \int_U \int_{V_t} \mathbf{F}(\mathbf{s}, t)\, \delta(\mathbf{x} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{s} \\ &= \int_U \mathbf{F}(\mathbf{s}, t) \int_{V_t} \delta(\mathbf{x} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{s}.\end{aligned}$$

The defining property of the Dirac delta function implies that

$$\int_{V_t} \delta(\mathbf{x} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{x} = \begin{cases} 1, & \text{if } \mathbf{X}(\mathbf{s}, t) \in V_t \iff \text{if } \mathbf{s} \in V, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

Therefore,

$$\int_{V_t} \mathbf{f}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} = \int_V \mathbf{F}(\mathbf{s}, t)\, \mathrm{d}\mathbf{s}.$$

Because $V \subseteq U$ is arbitrary, $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{F}(\mathbf{s}, t)$ are *equivalent as densities*.

Equivalence of $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{F}(\mathbf{s}, t)$ as densities

In fact, we can make a statement of equivalence between $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{F}(\mathbf{s}, t)$ that is stronger than simply that

$$\int_{V_t} \mathbf{f}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} = \int_{V} \mathbf{F}(\mathbf{s}, t)\, \mathrm{d}\mathbf{s}$$

for $V \subseteq U$ arbitrary, with $V_t = \mathbf{X}(V, t)$.

Equivalence of $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{F}(\mathbf{s}, t)$ as densities

Let $\mathbf{v}(\mathbf{x})$ be a smooth test function defined on Ω. Then

$$\begin{aligned}\int_\Omega \mathbf{f}(\mathbf{x}, t) \cdot \mathbf{v}(\mathbf{x})\, \mathrm{d}\mathbf{x} &= \int_\Omega \left(\int_U \mathbf{F}(\mathbf{s}, t)\, \delta(\mathbf{x} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s} \right) \cdot \mathbf{v}(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= \int_U \mathbf{F}(\mathbf{s}, t) \cdot \left(\int_\Omega \mathbf{v}(\mathbf{x})\, \delta(\mathbf{x} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{x} \right) \mathrm{d}\mathbf{s} \\ &= \int_U \mathbf{F}(\mathbf{s}, t) \cdot \mathbf{v}(\mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}.\end{aligned}$$

Defining $\mathbf{V}(\mathbf{s}, t) = \mathbf{v}(\mathbf{X}(\mathbf{s}, t))$, we have that, for all $\mathbf{v}(\mathbf{x})$,

$$\int_\Omega \mathbf{f}(\mathbf{x}, t) \cdot \mathbf{v}(\mathbf{x})\, \mathrm{d}\mathbf{x} = \int_U \mathbf{F}(\mathbf{s}, t) \cdot \mathbf{V}(\mathbf{s}, t)\, \mathrm{d}\mathbf{s}.$$

Notice that, by choosing $\mathbf{v}(\mathbf{x})$ appropriately, we can recover the conclusion that

$$\int_{V_t} \mathbf{f}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} = \int_V \mathbf{F}(\mathbf{s}, t)\, \mathrm{d}\mathbf{s}$$

for $V \subseteq U$ arbitrary, with $V_t = \mathbf{X}(V, t)$.

Equivalence of $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{F}(\mathbf{s}, t)$ as densities

Suppose further that $\mathbf{F}(\mathbf{s}, t)$ is smooth. Then from

$$\int_{V_t} \mathbf{f}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} = \int_{V} \mathbf{F}(\mathbf{s}, t)\, \mathrm{d}\mathbf{s},$$

for $V \subseteq U$ arbitrary, it follows that $\mathbf{f}(\mathbf{x}, t)$ and $\mathbf{F}(\mathbf{s}, t)$ are pointwise equivalent in the sense that

$$\mathbf{f}(\mathbf{x}, t) = \begin{cases} \frac{1}{J(\mathbf{X}^{-1}(\mathbf{x}, t), t)} \mathbf{F}(\mathbf{X}^{-1}(\mathbf{x}, t), t), & \text{for } \mathbf{x} \in U_t = \mathbf{X}(U, t), \\ 0, & \text{otherwise.} \end{cases}$$

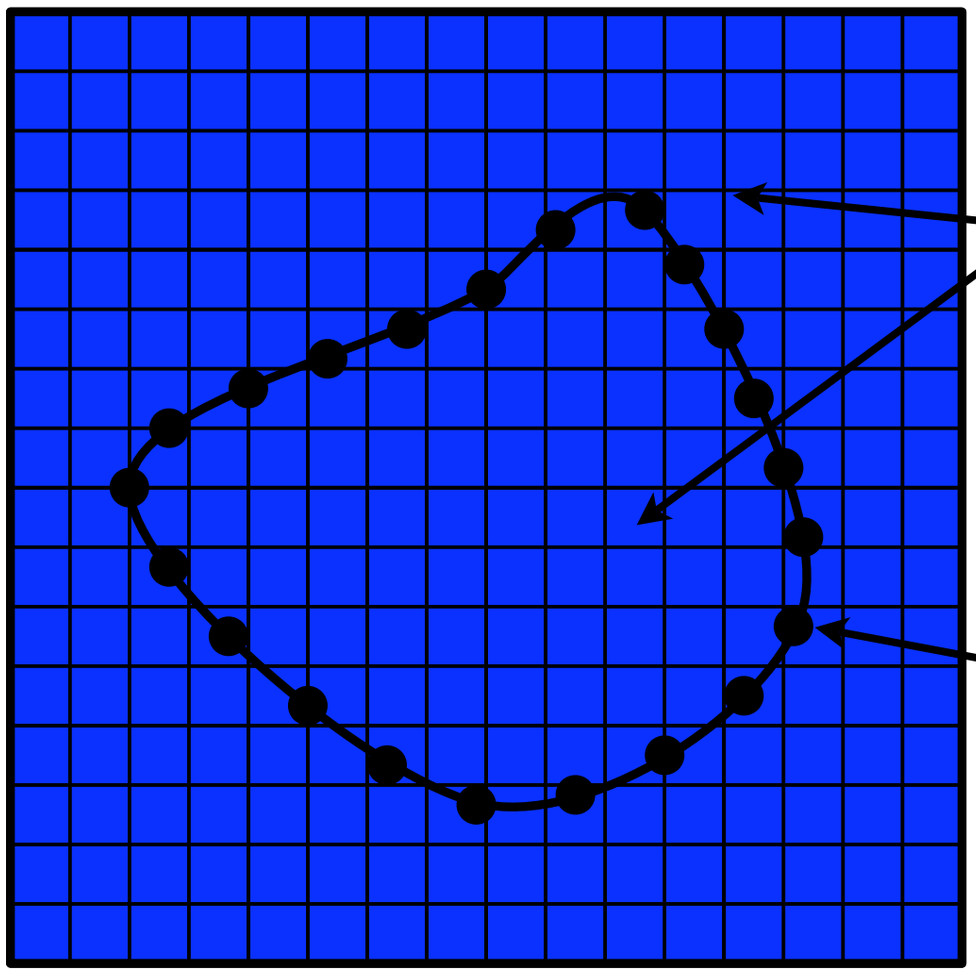
in which $J(\mathbf{s}, t) = \det\left(\mathbb{F}(\mathbf{s}, t)\right) = \det\left(\frac{\partial \mathbf{X}}{\partial \mathbf{s}}(\mathbf{s}, t)\right)$ is the Jacobian determinant of the *deformation gradient tensor* $\mathbb{F}(\mathbf{s}, t) = \frac{\partial \mathbf{X}}{\partial \mathbf{s}}(\mathbf{s}, t)$.

For this definition to make sense, clearly the mapping $(\mathbf{s}, t) \mapsto \mathbf{X}(\mathbf{s}, t)$ must be invertible.

Remark: $\mathbf{F}(\mathbf{s}, t)$ may not always be sufficiently smooth for this pointwise equivalence to be well defined.

The IB approach to fluid-structure interaction

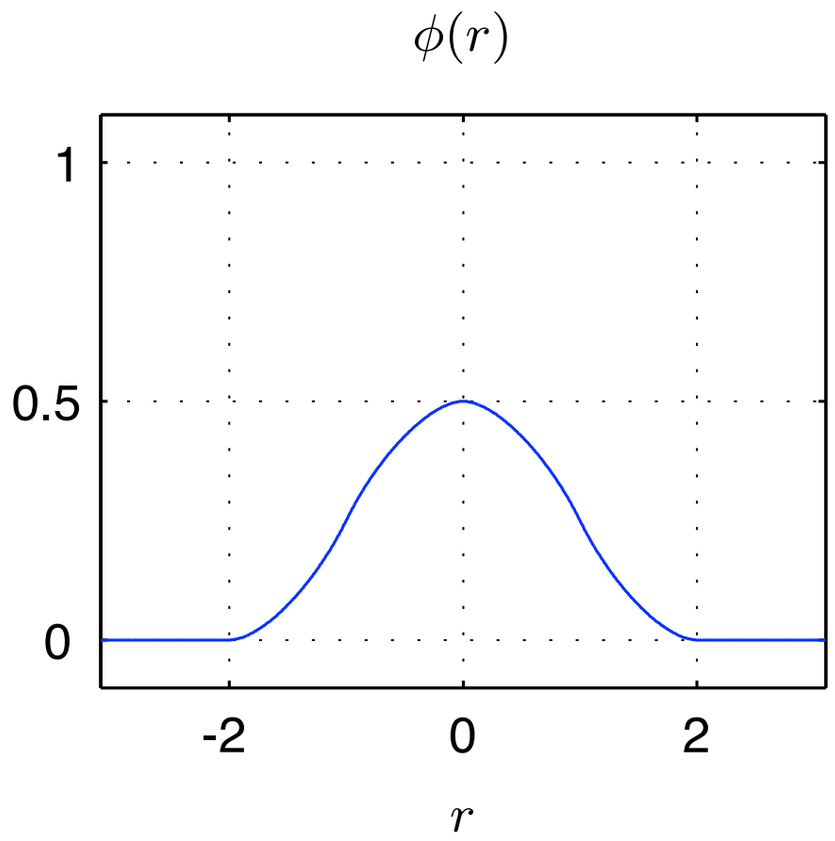
- Discretize the Eulerian equations on a Cartesian grid;
- Discretize the Lagrangian equations on a moving curvilinear mesh; and
- Discretize the interaction equations via regularized delta functions.



(i,j,k) indicate Cartesian grid points;
$\mathbf{x}_{i,j,k}$ is the position of grid point (i,j,k);
$\mathbf{u}_{i,j,k}$ is the velocity at $\mathbf{x}_{i,j,k}$; and
$p_{i,j,k}$ is the pressure at $\mathbf{x}_{i,j,k}$.

(q,r,s) indicate curvilinear mesh nodes;
$\mathbf{X}_{q,r,s}$ is the position of mesh node (q,r,s); and
$\mathbf{F}_{q,r,s}$ is the force at mesh node (q,r,s).

Regularized approximations to the Dirac delta function



Three-dimensional regularized delta function $\delta_h(\mathbf{x})$ is the tensor product of one-dimensional regularized delta functions $\delta_h(x)$:

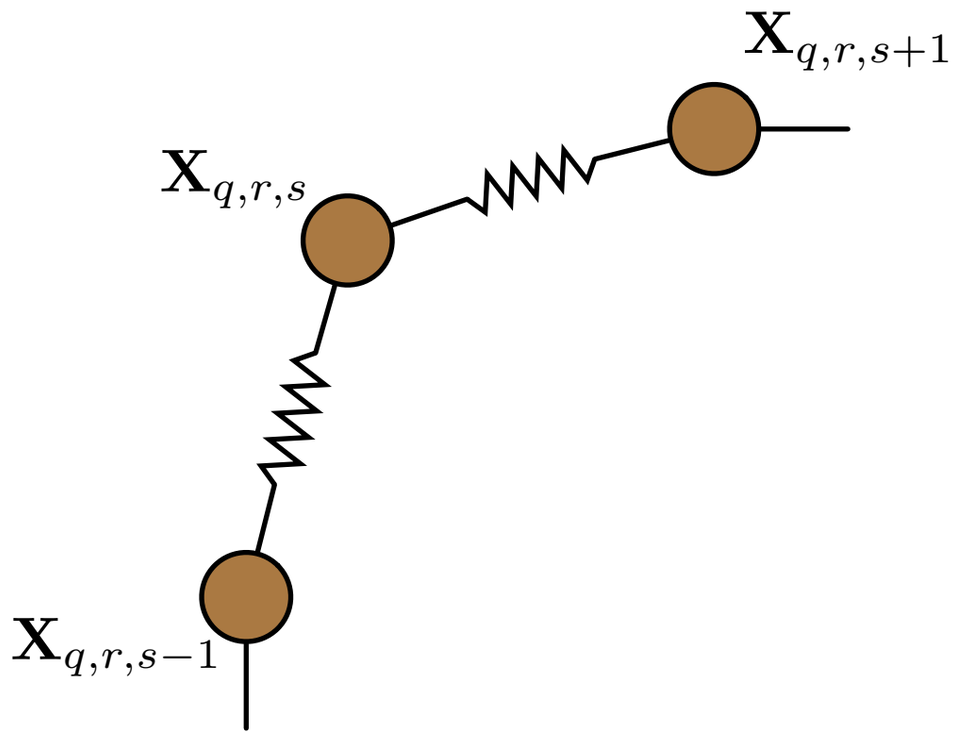
$$\delta_h(\mathbf{x}) = \delta_h(x)\,\delta_h(y)\,\delta_h(z)$$

with:

$$\delta_h(x) = \frac{1}{h}\,\phi\left(\frac{x}{h}\right),$$

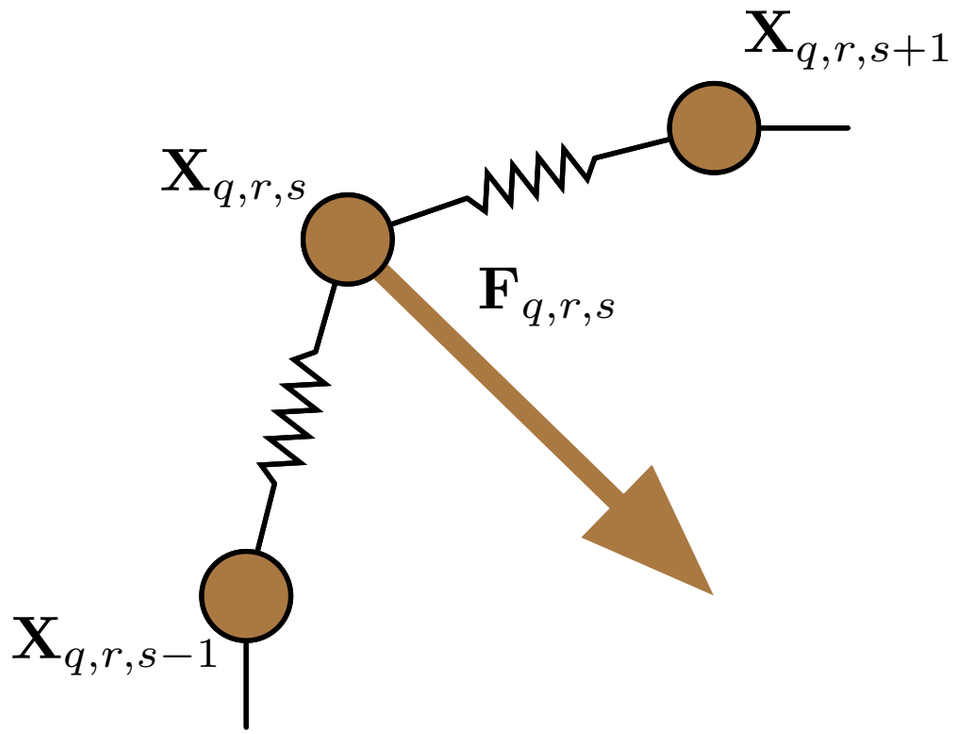
in which $h = \Delta x = \Delta y = \Delta z$ is the Cartesian grid spacing.

Spread force to the Eulerian grid



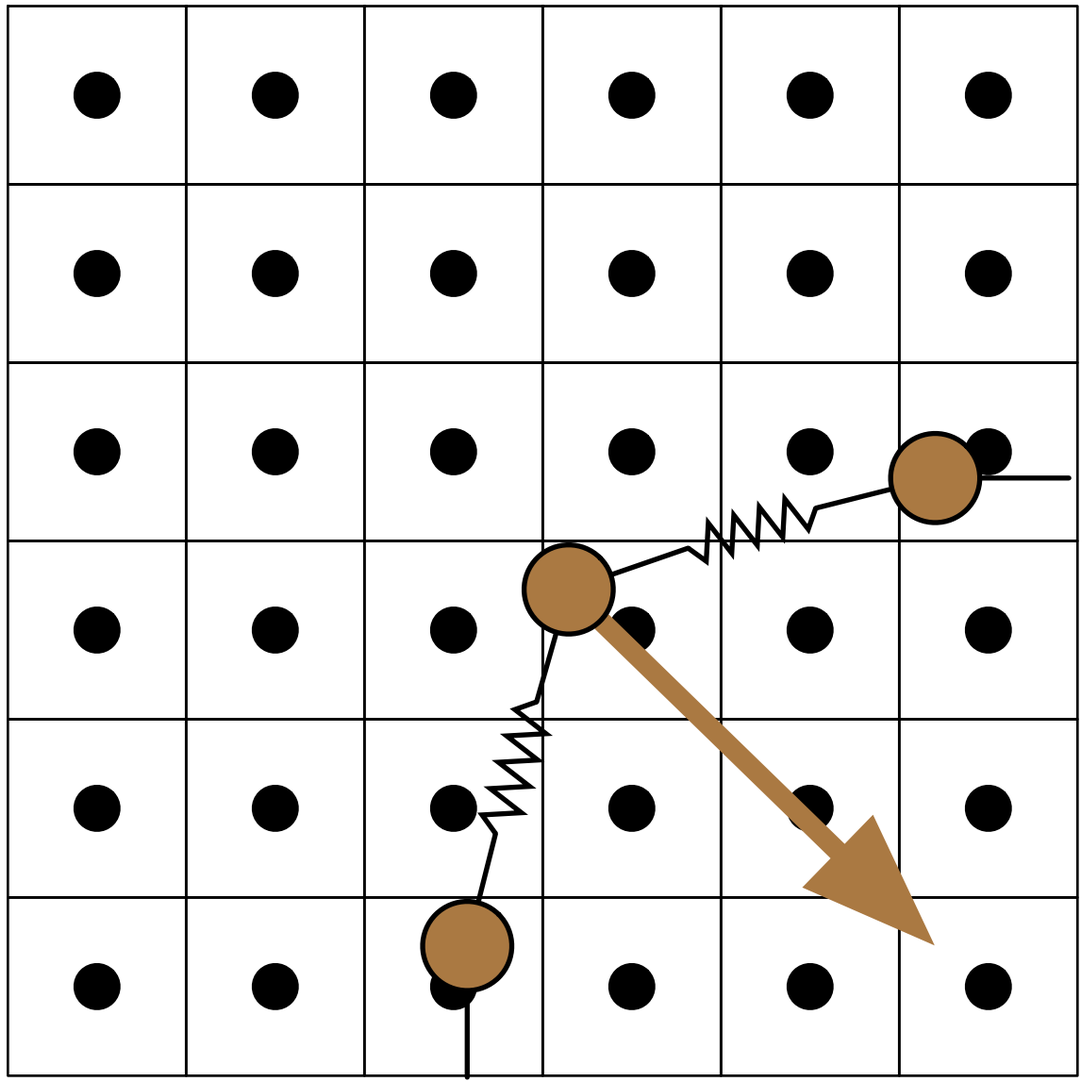
$$\begin{aligned}\mathbf{f}(\mathbf{x},t) &= \int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} \\ &\approx \sum_{q,r,s} \mathbf{F}_{q,r,s}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,\Delta q\,\Delta r\,\Delta s\end{aligned}$$

Spread force to the Eulerian grid



$$\mathbf{f}(\mathbf{x},t) = \int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}$$
$$\approx \sum_{q,r,s} \mathbf{F}_{q,r,s}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,\Delta q\,\Delta r\,\Delta s$$

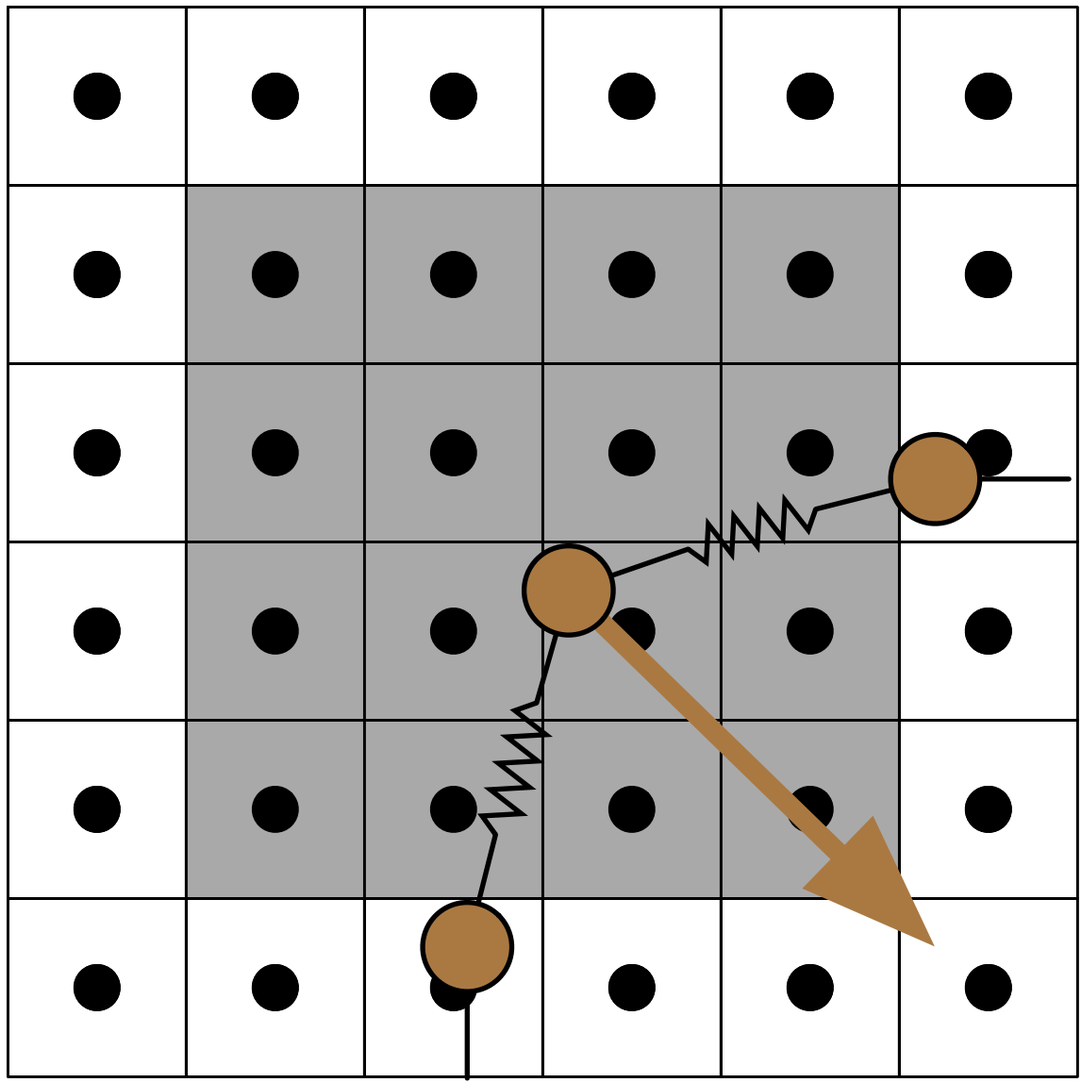
Spread force to the Eulerian grid



$$\mathbf{f}(\mathbf{x},t) = \int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}$$

$$\approx \sum_{q,r,s} \mathbf{F}_{q,r,s}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,\Delta q\,\Delta r\,\Delta s$$

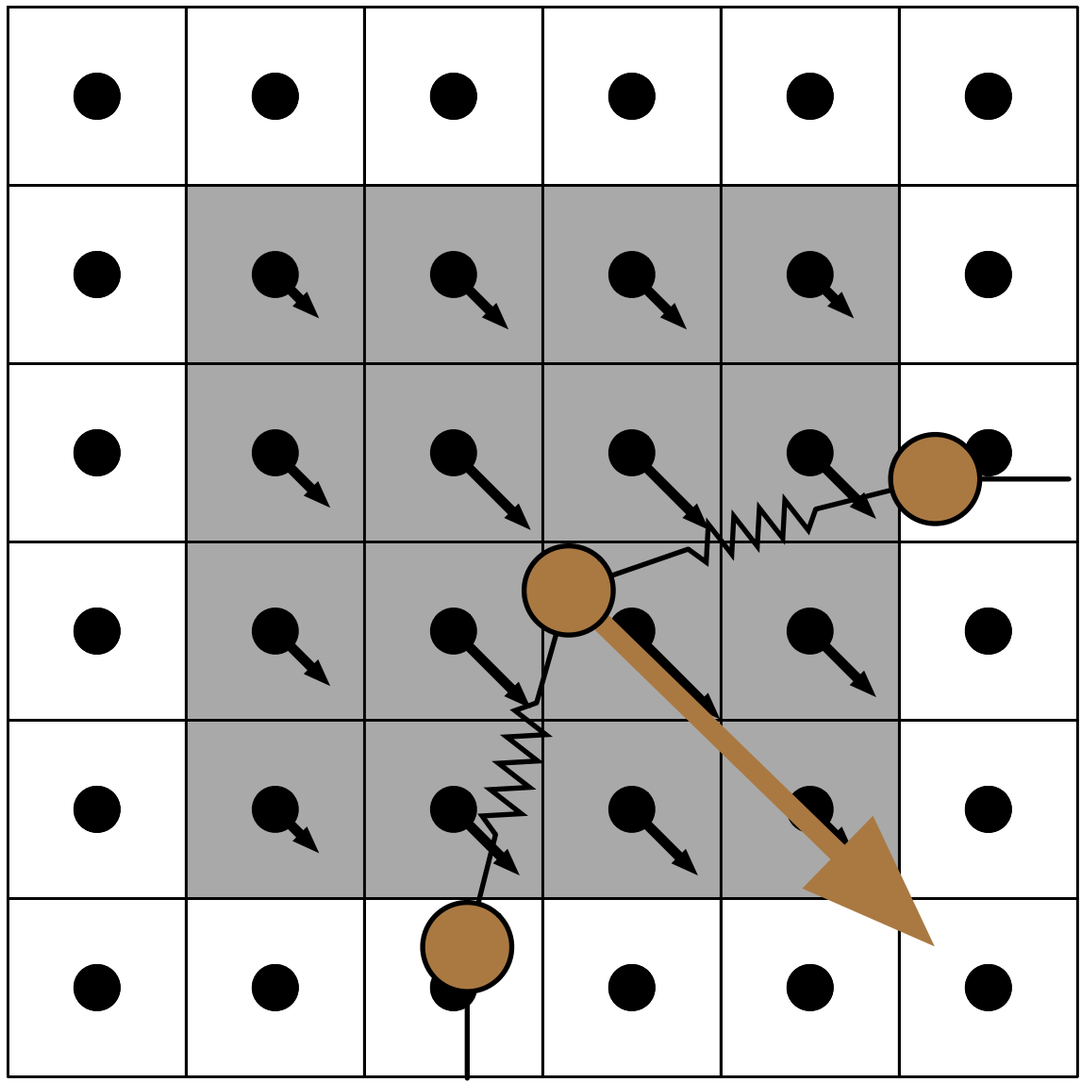
Spread force to the Eulerian grid



$$\mathbf{f}(\mathbf{x},t) = \int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}$$

$$\approx \sum_{q,r,s} \mathbf{F}_{q,r,s}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,\Delta q\,\Delta r\,\Delta s$$

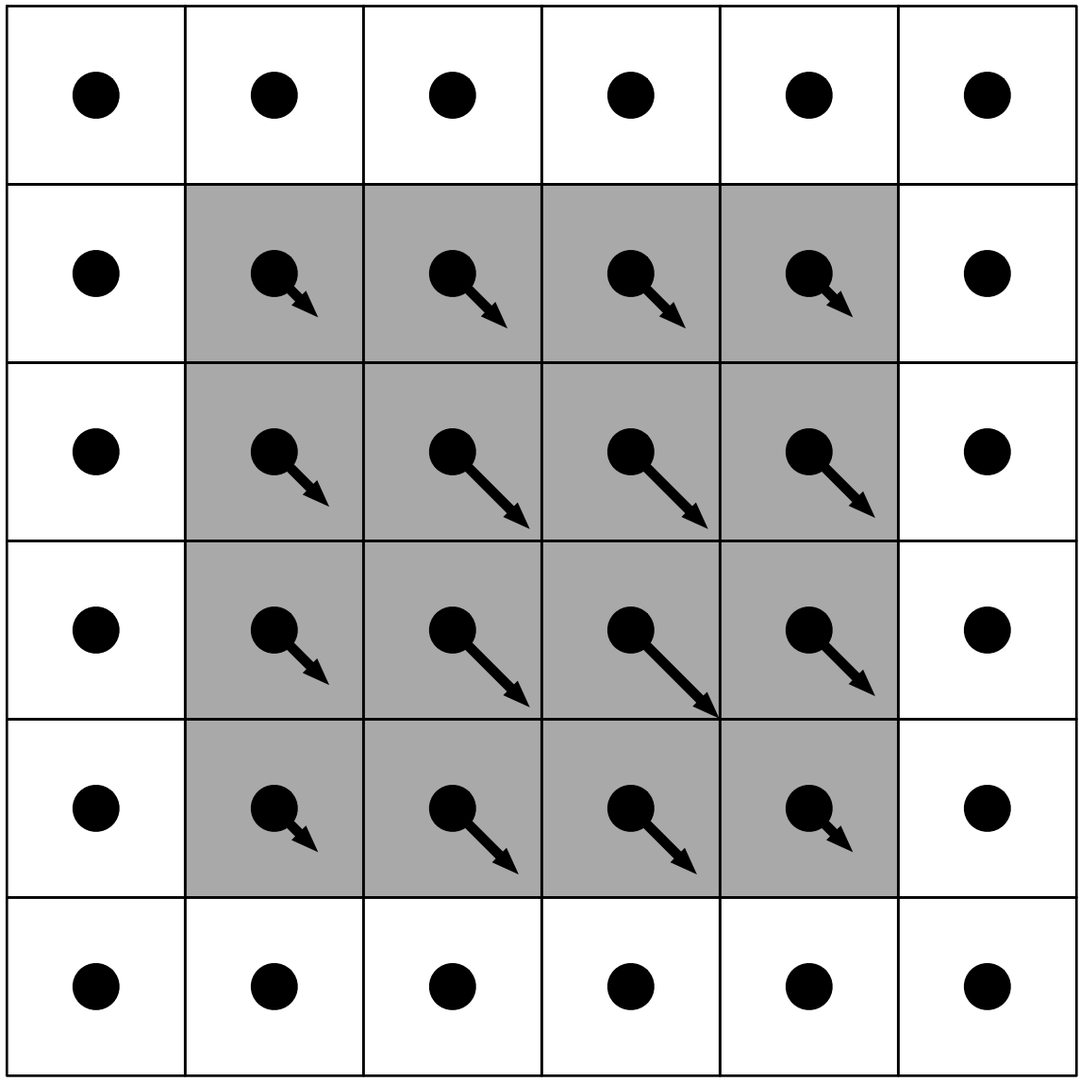
Spread force to the Eulerian grid



$$\mathbf{f}(\mathbf{x},t) = \int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}$$

$$\approx \sum_{q,r,s} \mathbf{F}_{q,r,s}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,\Delta q\,\Delta r\,\Delta s$$

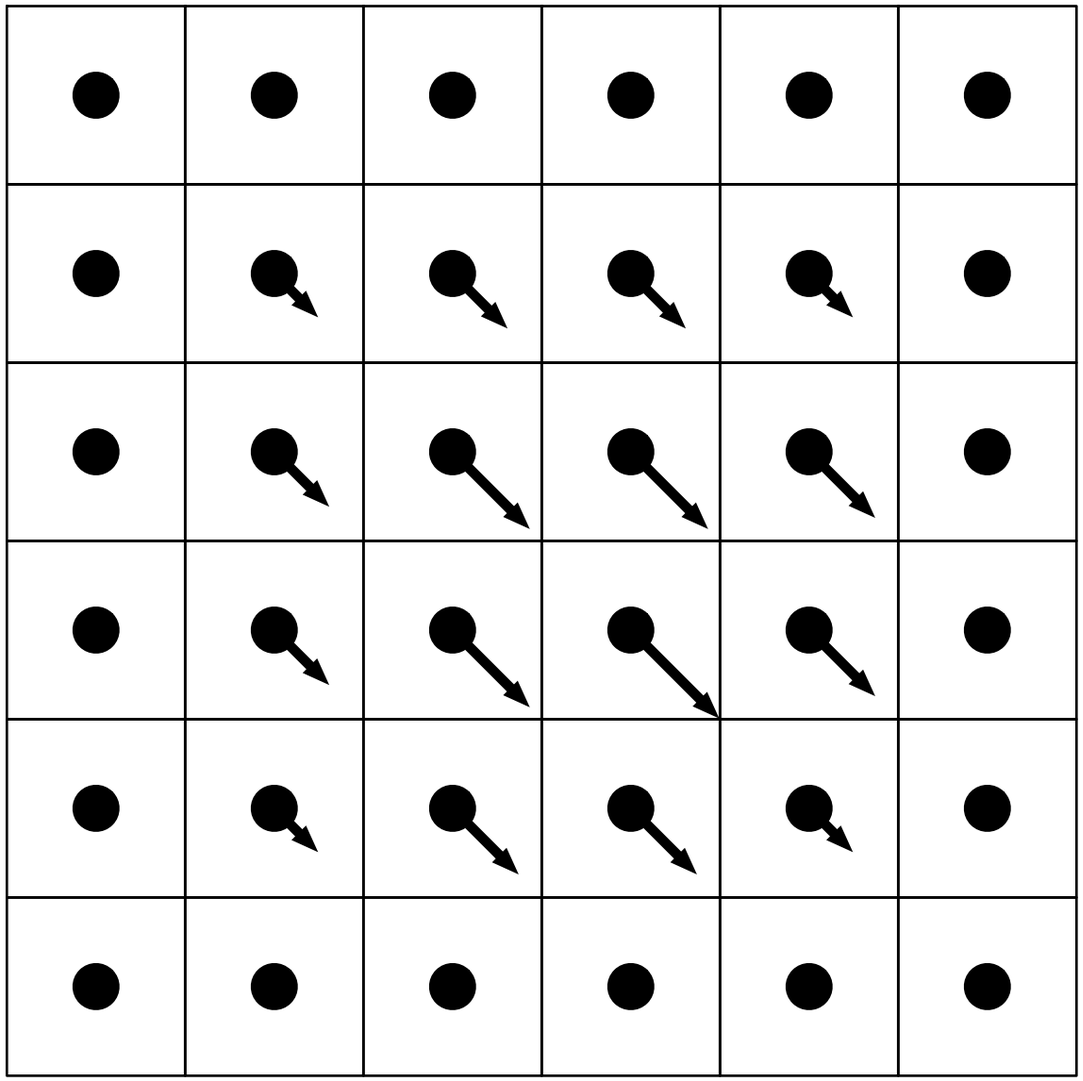
Spread force to the Eulerian grid



$$\mathbf{f}(\mathbf{x},t) = \int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}$$

$$\approx \sum_{q,r,s} \mathbf{F}_{q,r,s}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,\Delta q\,\Delta r\,\Delta s$$

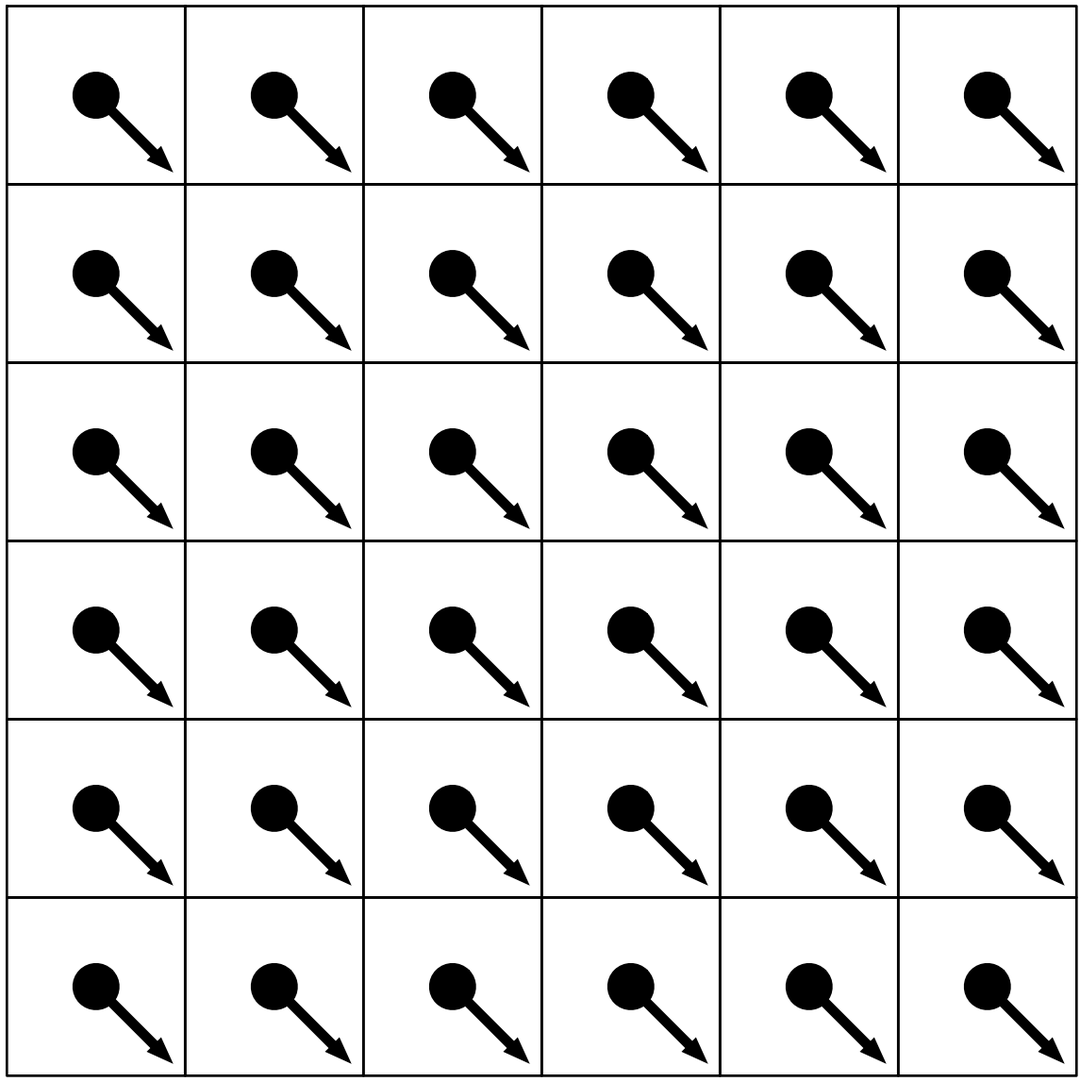
Spread force to the Eulerian grid



$$\mathbf{f}(\mathbf{x},t) = \int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}$$

$$\approx \sum_{q,r,s} \mathbf{F}_{q,r,s}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,\Delta q\,\Delta r\,\Delta s$$

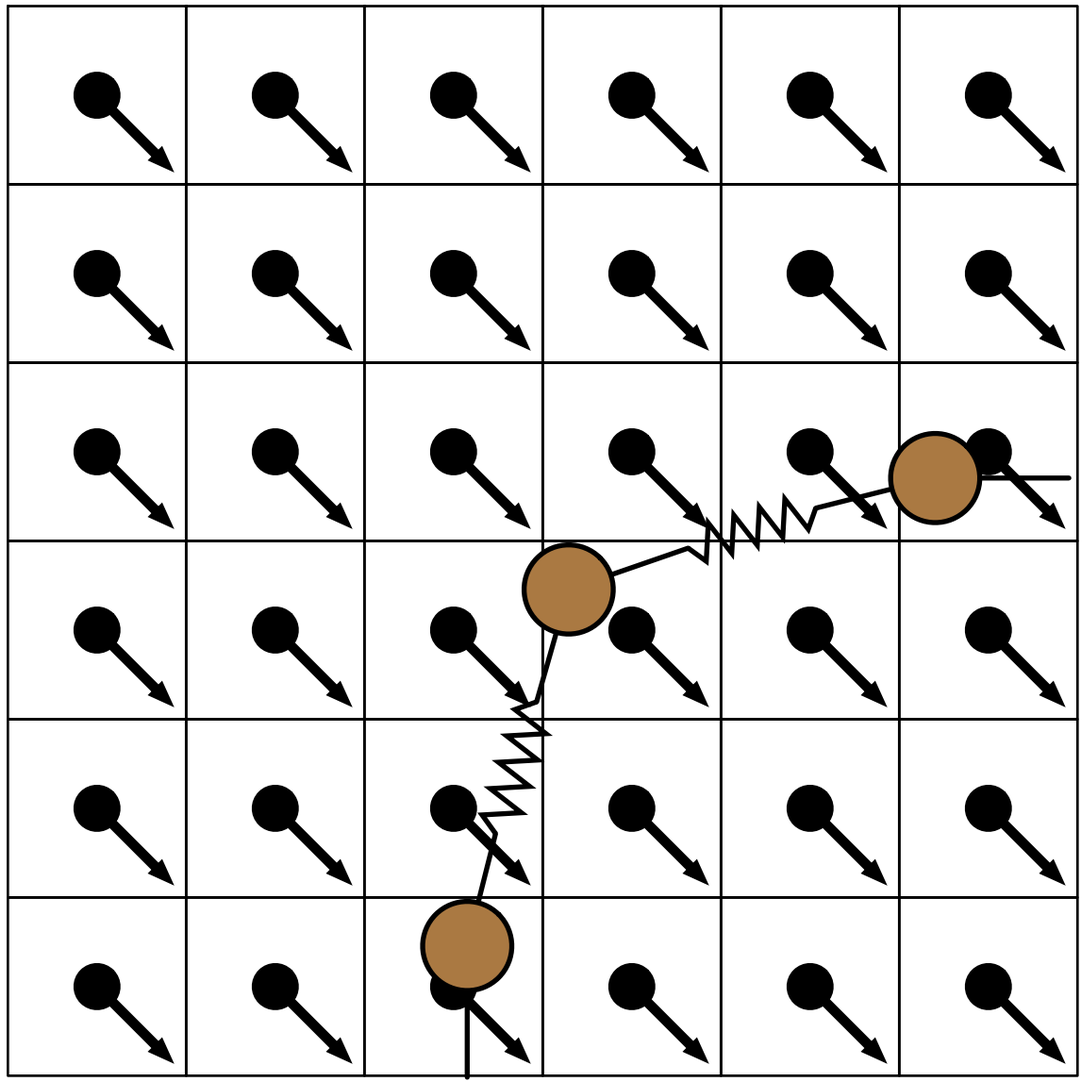
Restrict velocity to the Lagrangian mesh



$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_{\Omega} \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}$$

$$\approx \sum_{i,j,k} \mathbf{u}_{i,j,k}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,h^3$$

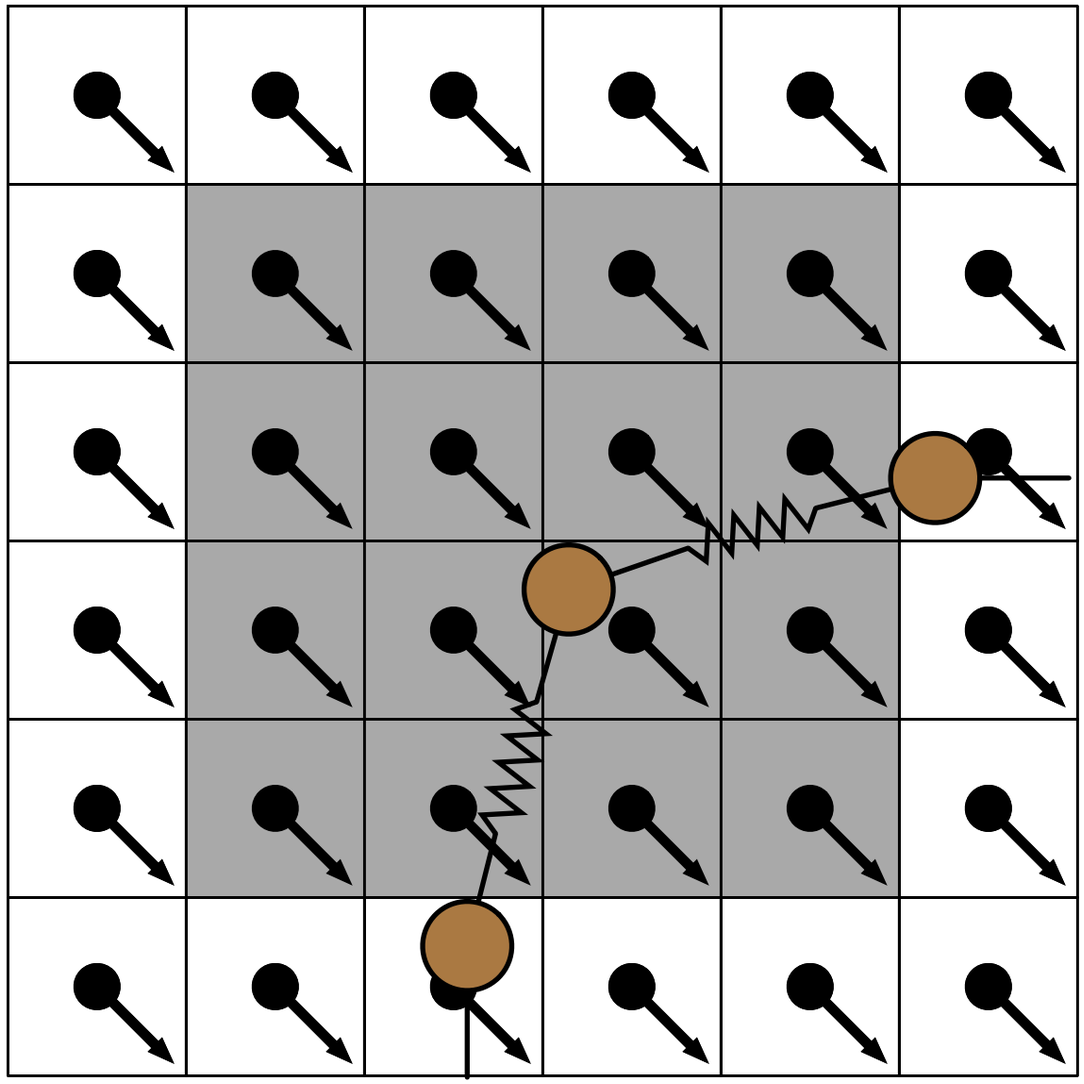
Restrict velocity to the Lagrangian mesh



$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_{\Omega} \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}$$

$$\approx \sum_{i,j,k} \mathbf{u}_{i,j,k}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,h^3$$

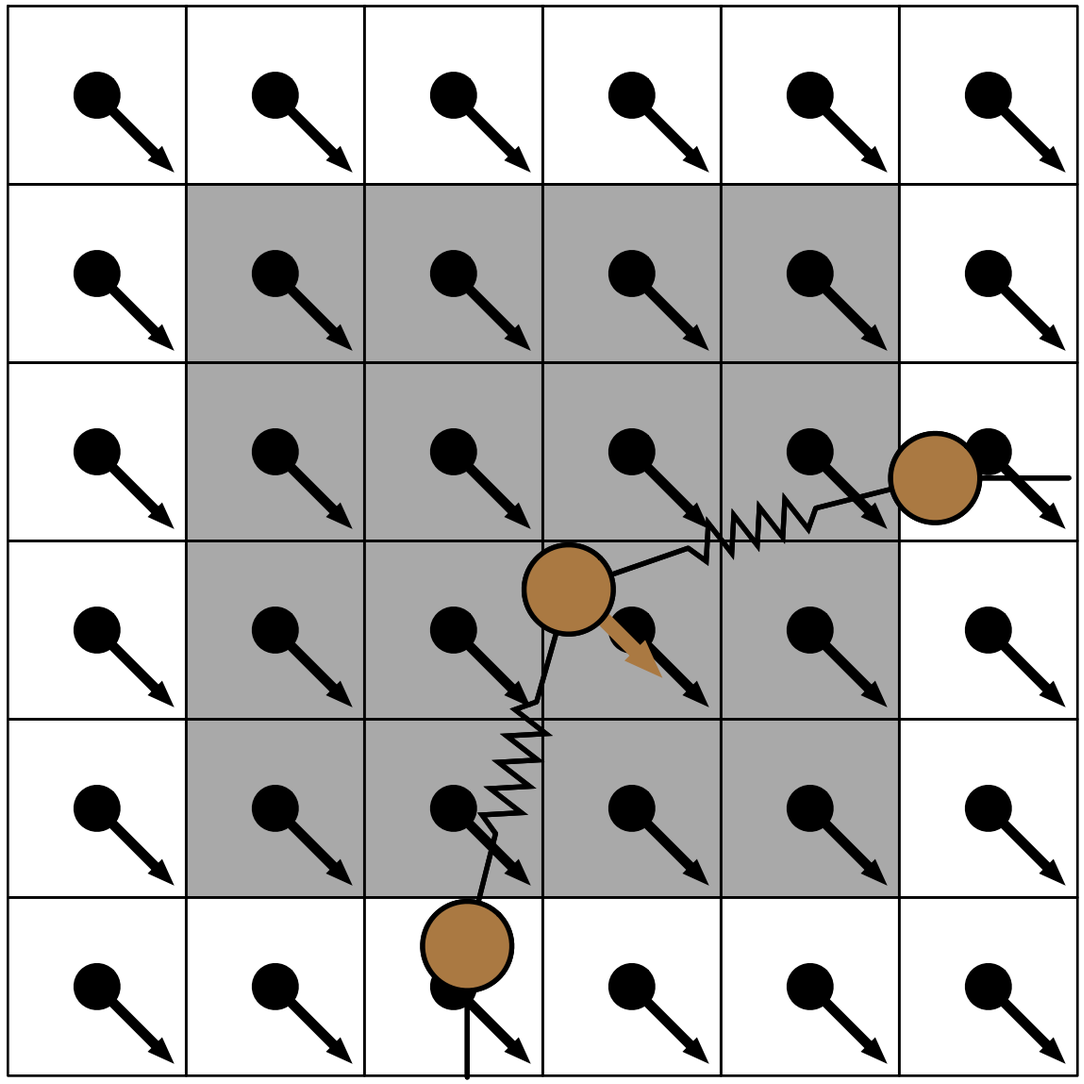
Restrict velocity to the Lagrangian mesh



$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}$$

$$\approx \sum_{i,j,k} \mathbf{u}_{i,j,k}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,h^3$$

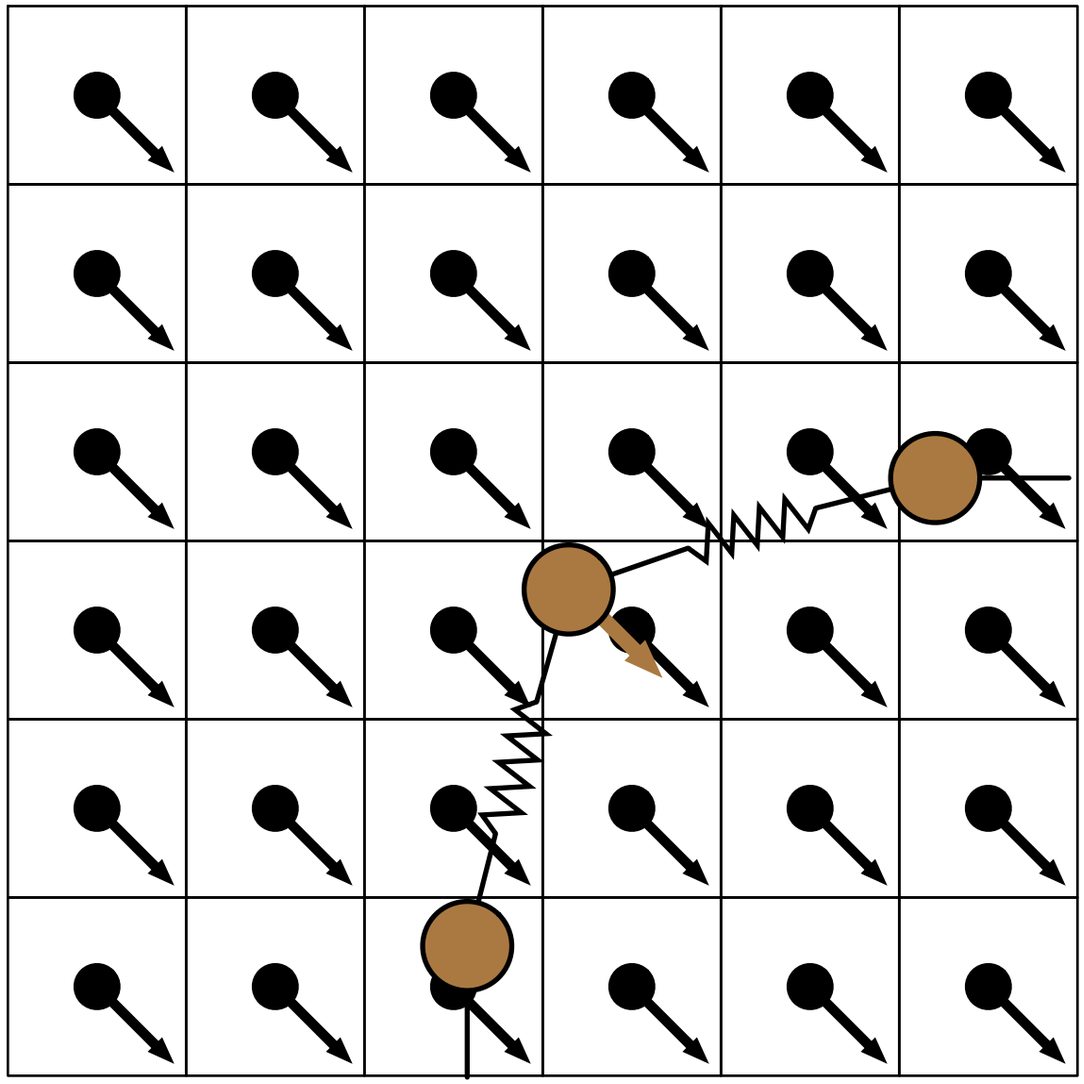
Restrict velocity to the Lagrangian mesh



$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}$$

$$\approx \sum_{i,j,k} \mathbf{u}_{i,j,k}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,h^3$$

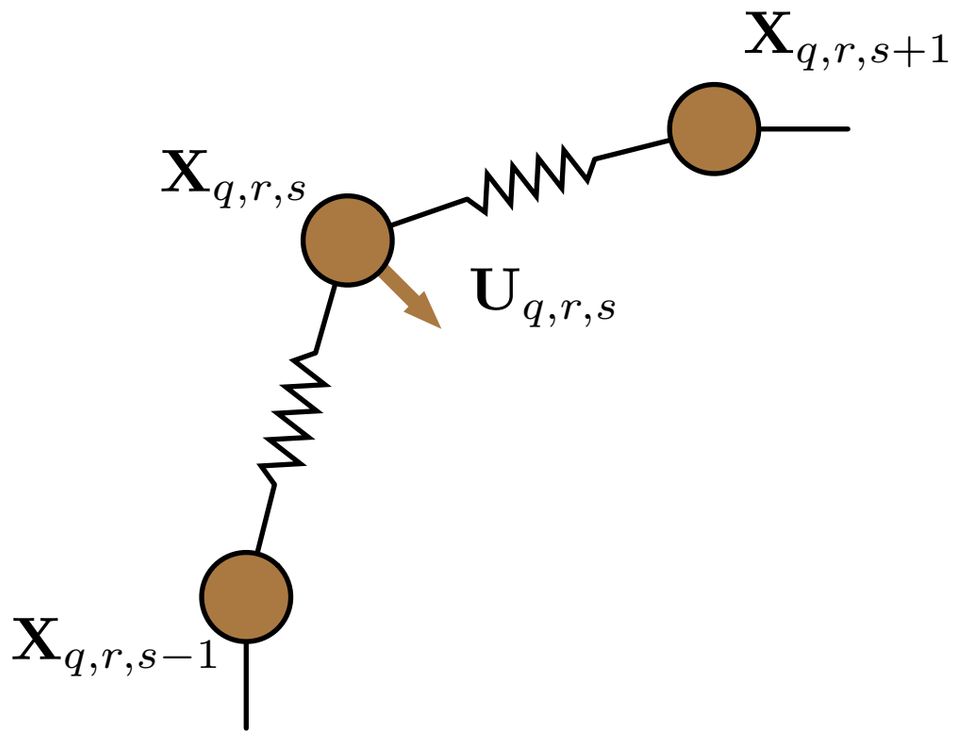
Restrict velocity to the Lagrangian mesh



$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}$$

$$\approx \sum_{i,j,k} \mathbf{u}_{i,j,k}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,h^3$$

Restrict velocity to the Lagrangian mesh



$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_{\Omega} \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}$$
$$\approx \sum_{i,j,k} \mathbf{u}_{i,j,k}\,\delta_h(\mathbf{x}_{i,j,k}-\mathbf{X}_{q,r,s})\,h^3$$

Regularized approximations to the Dirac delta function

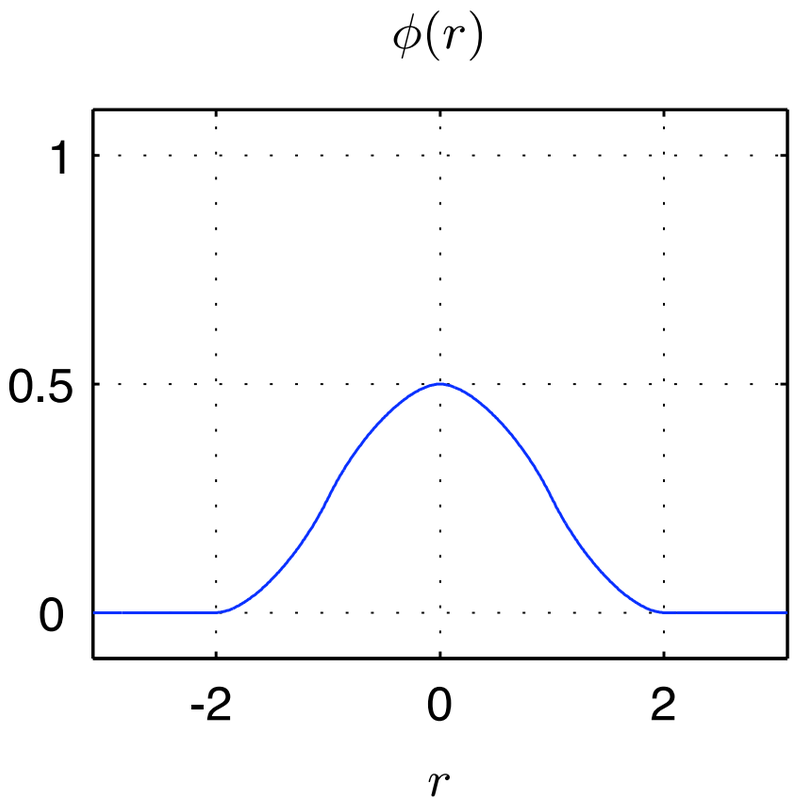
A commonly used choice for $\phi(r)$ is the *four-point delta function*, which satisfies $\forall r \in \mathbb{R}$:

$$\phi(r) = 0 \quad |r| \geq 2,$$

$$\sum_{j \in \mathbb{Z} \text{ even}} \phi(r-j) = \sum_{j \in \mathbb{Z} \text{ odd}} \phi(r-j) = \frac{1}{2},$$

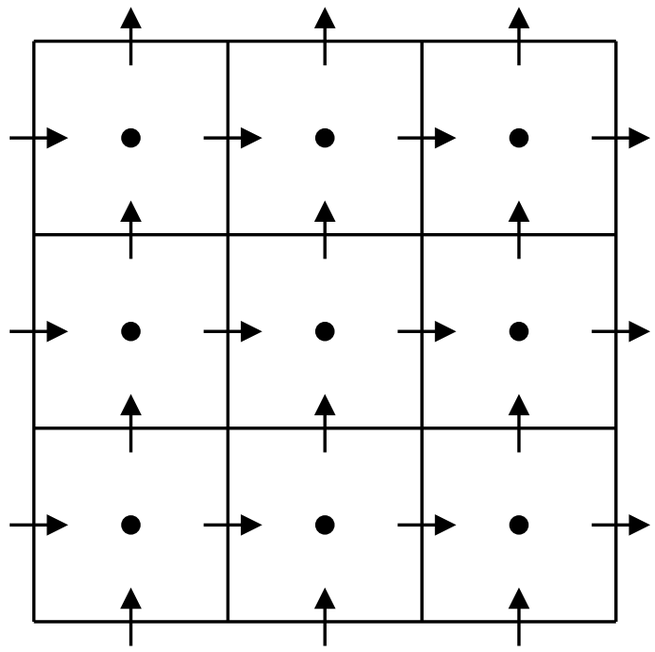
$$\sum_{j \in \mathbb{Z}} (r-j)\,\phi(r-j) = 0,$$

$$\sum_{j \in \mathbb{Z}} (\phi(r-j))^2 = C = \frac{3}{8}.$$



These properties ensure conservation of force and torque during Lagrangian-Eulerian interaction, and yield a second-order accurate interpolation scheme for smooth velocity fields.

Eulerian spatial discretization



Staggered-grid (MAC) discretization: Normal components of the velocity approximated on cell edges (faces in 3D); pressures approximated at cell centers:

$$(\nabla_h \cdot \mathbf{u})_{i,j} := \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\Delta y},$$

$$\left((\nabla_h^x p)_{i+\frac{1}{2},j}, (\nabla_h^y p)_{i,j+\frac{1}{2}}\right) := \left(\frac{p_{i+1,j} - p_{i,j}}{\Delta x}, \frac{p_{i,j+1} - p_{i,j}}{\Delta y}\right),$$

$$(\nabla_h \cdot \nabla_h p)_{i,j} = \left(\nabla_h^2 p\right)_{i,j} := \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{\Delta y^2}.$$

Notation for Lagrangian-Eulerian interaction

Force spreading:

$$\begin{aligned}\mathbf{f}_{i,j,k}^{n} &= \sum_{q,r,s} \mathbf{F}_{q,r,s}^{n}\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}_{q,r,s}^{n})\, \Delta q\, \Delta r\, \Delta s \\ &= \left(\mathcal{S}^{n}\mathbf{F}^{n}\right)_{i,j,k},\end{aligned}$$

in which $\mathcal{S}^{n} = \mathcal{S}\left[\mathbf{X}^{n}\right]$ is the *force-spreading operator*.

Velocity restriction:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{X}_{q,r,s}^{n} &= \sum_{i,j,k} \mathbf{u}_{i,j,k}^{n}\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}_{q,r,s}^{n})\, h^3 \\ &= \left(\mathcal{R}^{n}\mathbf{u}^{n}\right)_{q,r,s},\end{aligned}$$

in which $\mathcal{R}^{n} = \mathcal{R}\left[\mathbf{X}^{n}\right]$ is the *velocity-restriction operator*.

Notice: For simplicity, these operators are stated for a cell-centered discretization. Minor modifications are needed for the staggered-grid discretization.

Remark: Because we use δ_h for both force spreading and velocity restriction, $\mathcal{R} = \mathcal{S}^{*}$. This implies that the semi-discretized IB method conserves energy during Lagrangian-Eulerian interaction.

Numerical methods

Given $\mathbf{X}^n$, $\mathbf{u}^n$, and $p^{n-\frac{1}{2}}$, we wish to obtain $\mathbf{X}^{n+1}$, $\mathbf{u}^{n+1}$, and $p^{n+\frac{1}{2}}$ by discretizing the equations of motion.

A simple semi-implicit, first-order accurate discretization would be:

$$\begin{aligned}
\rho\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}+\mathbf{N}^n\right) &= -\nabla_h p^{n+\frac{1}{2}}+\frac{\mu}{2}\nabla_h^2\left(\mathbf{u}^{n+1}+\mathbf{u}^n\right)+\mathbf{f}^n,\\
\nabla_h\cdot\mathbf{u}^{n+1} &= 0,\\
\mathbf{f}^n &= \mathcal{S}^n\,\mathbf{F}^n,\\
\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t} &= \mathcal{R}^n\,\mathbf{u}^{n+1},\\
\mathbf{F}^n &= \boldsymbol{\mathcal{F}}\left[\mathbf{X}^n\right],
\end{aligned}$$

in which $\mathbf{N}^n \approx \left[\mathbf{u}\cdot\nabla\mathbf{u}\right]^n$.

It is straightforward to extend this discretization to second-order accuracy in time by using a Runge-Kutta-type approach.

Notice: To solve these equations for $\mathbf{X}^{n+1}$, $\mathbf{u}^{n+1}$, and $p^{n+\frac{1}{2}}$, we need only to solve a linearly implicit discretization of the incompressible Navier-Stokes equations.

The projection method

The projection method is a fractional-step timestepping scheme.

Step 1: Solve the momentum equation without the incompressibility constraint for an intermediate velocity $\mathbf{u}^*$:

$$\rho\left(\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + \mathbf{N}^{(n+\frac{1}{2})}\right) = \frac{\mu}{2}\nabla_h^2\left(\mathbf{u}^* + \mathbf{u}^n\right) + \mathbf{f}^{(n+\frac{1}{2})}.$$

Step 2: Impose the incompressibility constraint:

$$\rho\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t}\right) + \nabla_h\varphi = 0,$$

$$\nabla_h \cdot \mathbf{u}^{n+1} = 0.$$

Taking the divergence of the first of these equations, and using $\nabla_h \cdot \mathbf{u}^{n+1} = 0$, we have:

$$\nabla_h^2\varphi = \frac{\rho}{\Delta t}\nabla_h \cdot \mathbf{u}^*.$$

It follows that:

$$\mathbf{u}^{n+1} = \left(I - \nabla_h\left(\nabla_h^2\right)^{-1}\nabla_h\cdot\right)\mathbf{u}^* = P\,\mathbf{u}^*.$$

The operator $P = \left(I - \nabla_h\left(\nabla_h^2\right)^{-1}\nabla_h\cdot\right)$ is a projection operator ($P^2 = P$).

Step 3: Evaluate:

$$p^{n+\frac{1}{2}} = \left(I - \frac{\Delta t}{\rho}\frac{\mu}{2}\nabla_h^2\right)\varphi.$$

The projection method *as a preconditioner*

The projection method can be recast as a *preconditioner* for a Krylov method (e.g., GMRES).

Krylov methods are iterative methods for solving linear systems of equations $Ax = b$ by constructing *Krylov subspaces*, i.e.,

$$K_r(A, b) = \operatorname{span}\left(\left\{b, Ab, A^2b, \ldots, A^{r-1}b\right\}\right).$$

The convergence of such methods depends on the condition number of the matrix A.

For ill-conditioned matrices, it may be faster to apply the Krylov method to the *preconditioned* linear system

$$(BA)x = Bb.$$

To obtain good performance, the preconditioner B should be an "approximate inverse" of A that is inexpensive to compute.

The projection method *as a preconditioner*

Our semi-implicit discretization can be written in block matrix form:

$$Ax := \begin{pmatrix} \frac{\rho}{\Delta t} I - \frac{\mu}{2}\nabla_h^2 & \nabla_h \\ -\nabla_h \cdot & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}$$

To solve this system using a Krylov method, we need an effective *preconditioner*.

The projection method can be recast as a preconditioner B for the matrix A:

$$B := \begin{pmatrix} I & -\frac{\Delta t}{\rho}\nabla_h \\ 0 & I - \frac{\Delta t}{\rho}\frac{\mu}{2}\nabla_h^2 \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & (\nabla_h^2)^{-1} \end{pmatrix} \cdot$$

$$\begin{pmatrix} I & 0 \\ -\frac{\rho}{\Delta t}\nabla_h \cdot & -\frac{\rho}{\Delta t} I \end{pmatrix} \begin{pmatrix} (\frac{\rho}{\Delta t} I - \frac{\mu}{2}\nabla_h^2)^{-1} & 0 \\ 0 & I \end{pmatrix}$$

This approach can be viewed as using a Krylov method to eliminate the splitting errors of the basic projection method.

Key points:

- Need a *velocity subdomain solver* for the operator $\left(\frac{\rho}{\Delta t} I - \frac{\mu}{2}\nabla_h^2\right)$ and a *pressure subdomain solver* for the operator ∇_h^2.
- Exact subdomain solvers are **not** needed.
- The “artificial” boundary conditions required by the subdomain solvers **do not** affect the *actual* boundary conditions imposed on the coupled system.

Some loose ends. . .

What solvers do we use for the operators $\left(\frac{\rho}{\Delta t} I - \frac{\mu}{2}\nabla_h^2\right)$ and ∇_h^2?

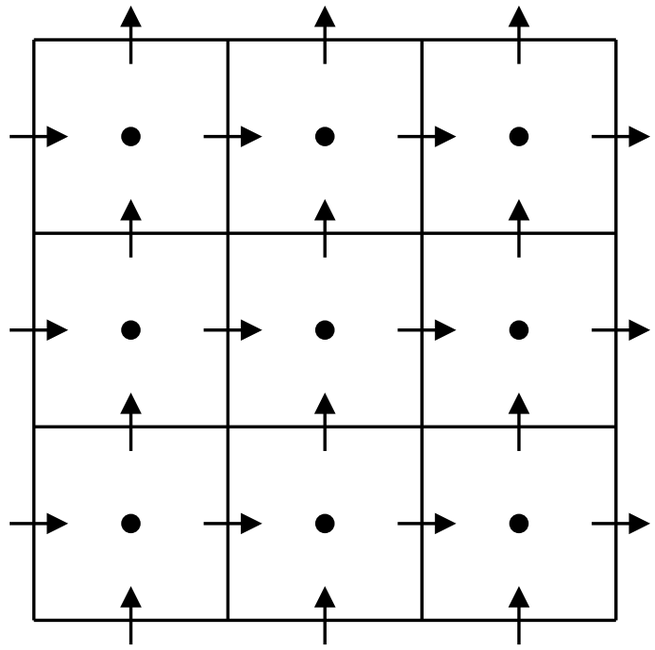
- FFT (for periodic domains).
- Multigrid or multigrid-preconditioned Krylov methods (for either periodic or nonperiodic domains).

Some loose ends. . .

How do we approximate $\mathbf{u} \cdot \nabla \mathbf{u}$?

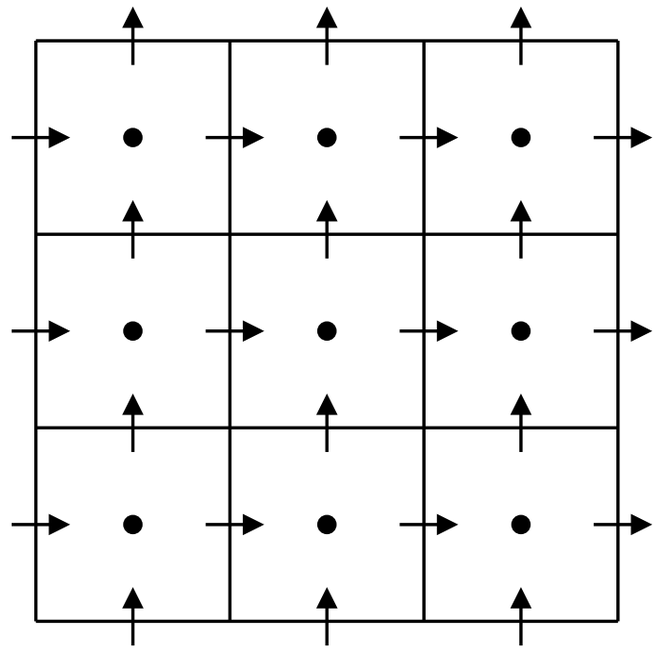
- Centered differences.
- Higher-order upwind differences.

Centered approximation to $\mathbf{u} \cdot \nabla \mathbf{u}$



We can easily compute centered approximations to $\mathbf{u} \cdot \nabla u$ at cell edges in *advection*, *conservation*, or *skew-symmetric* forms.

Centered approximation to $\mathbf{u} \cdot \nabla \mathbf{u}$



An approximation to $(\mathbf{u} \cdot \nabla u)$ at $\mathbf{x}_{i-\frac{1}{2},j}$ in advection form may be computed as

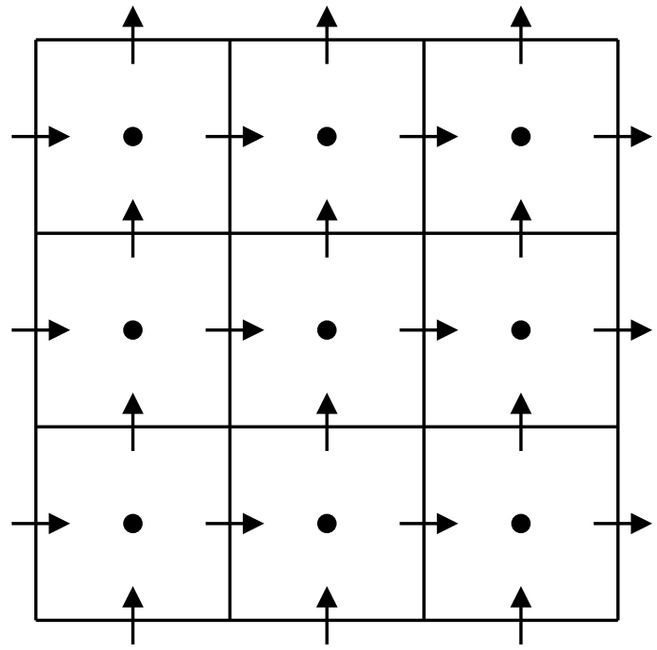
$$(\mathbf{u} \cdot \nabla u)_{i-\frac{1}{2},j} \approx A_x + A_y,$$

with:

$$A_x := \frac{1}{2}\left(\frac{u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}}{2}\,\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{h} + \frac{u_{i-\frac{1}{2},j} + u_{i-\frac{3}{2},j}}{2}\,\frac{u_{i-\frac{1}{2},j} - u_{i-\frac{3}{2},j}}{h}\right),$$

$$A_y := \frac{1}{2}\left(\frac{v_{i,j+\frac{1}{2}} + v_{i,j-\frac{1}{2}}}{2}\,\frac{u_{i,j+\frac{1}{2}} - u_{i,j-\frac{1}{2}}}{h} + \frac{v_{i,j-\frac{1}{2}} + v_{i,j-\frac{3}{2}}}{2}\,\frac{u_{i,j-\frac{1}{2}} - u_{i,j-\frac{3}{2}}}{h}\right).$$

Centered approximation to $\mathbf{u} \cdot \nabla \mathbf{u}$



An approximation to $(\mathbf{u} \cdot \nabla v)$ at $\mathbf{x}_{i,j-\frac{1}{2}}$ may be computed similarly.

Some loose ends. . .

What is $\mathbf{F} = \boldsymbol{\mathcal{F}}\left[\mathbf{X}; \mathrm{s}, t\right]$, and how do we compute it?

- For an elastic structure comprised of systems of elastic fibers, with (q, r) labeling individual fibers and s running along a particular fiber, $\mathbf{F} = \frac{\partial}{\partial s}\left(T\boldsymbol{\tau}\right)$, in which T is the fiber tension and $\boldsymbol{\tau}$ is the unit tangent vector in the fiber direction, i.e., $\boldsymbol{\tau} = \frac{\partial \mathbf{X}}{\partial s} \Big/ \left|\frac{\partial \mathbf{X}}{\partial s}\right|$.
  - In the simple special case in which $T = \kappa \left|\frac{\partial \mathbf{X}}{\partial s}\right|$, $\mathbf{F} = \kappa \frac{\partial^2 \mathbf{X}}{\partial s^2}$.
- If we can describe the elasticity of the structure in terms of an energy functional $E[\mathbf{X}(\cdot, t)]$, then $\mathbf{F} = -\frac{\delta E}{\delta \mathbf{X}}$, in which $\frac{\delta E}{\delta \mathbf{X}}$ is the Fréchet (total) derivative of E, i.e.,

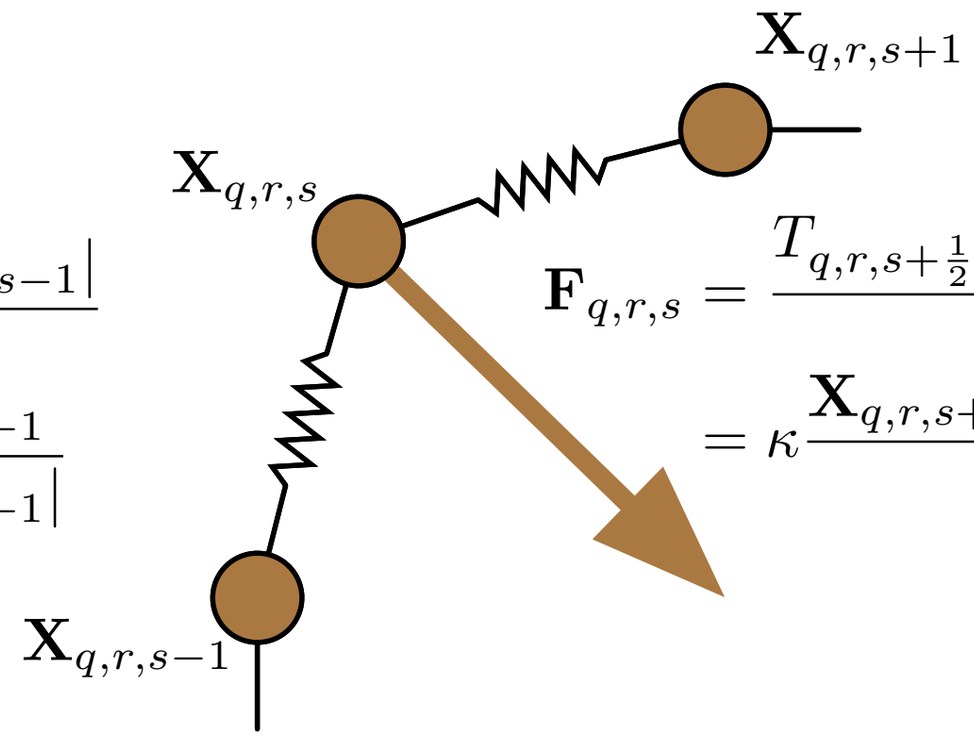
$$\delta E[\mathbf{X}(\cdot, t)] = -\int_U \mathbf{F}(\mathbf{s}, t) \cdot \delta \mathbf{X}(\mathbf{s}, t)\, \mathrm{d}\mathbf{s}.$$

  (Note that here, δ is the perturbation operator, not the Dirac delta function!)
- For fiber-based models, we can approximate $\mathbf{F}$ via centered finite differences. Typically, this amounts to describing the elasticity of the structure in terms of systems of springs and beams.

A simple discretization of the Lagrangian force

$$T_{q,r,s+\frac{1}{2}} = \kappa \frac{|\mathbf{X}_{q,r,s+1} - \mathbf{X}_{q,r,s}|}{\Delta s}$$

$$\tau_{q,r,s+\frac{1}{2}} = \frac{\mathbf{X}_{q,r,s+1} - \mathbf{X}_{q,r,s}}{|\mathbf{X}_{q,r,s+1} - \mathbf{X}_{q,r,s}|}$$



$$T_{q,r,s-\frac{1}{2}} = \kappa \frac{|\mathbf{X}_{q,r,s} - \mathbf{X}_{q,r,s-1}|}{\Delta s}$$

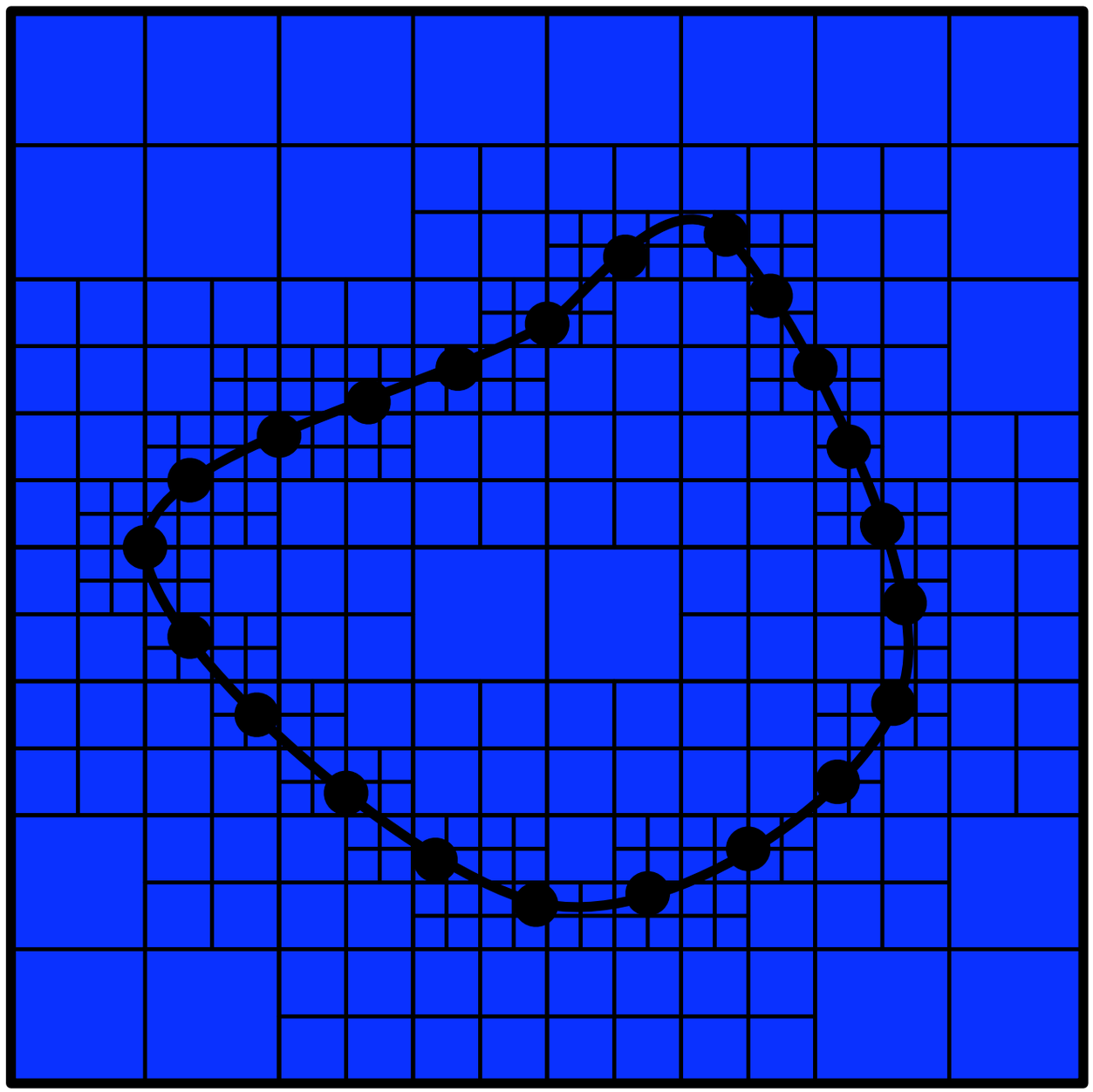
$$\tau_{q,r,s-\frac{1}{2}} = \frac{\mathbf{X}_{q,r,s} - \mathbf{X}_{q,r,s-1}}{|\mathbf{X}_{q,r,s} - \mathbf{X}_{q,r,s-1}|}$$

$$\mathbf{F}_{q,r,s} = \frac{T_{q,r,s+\frac{1}{2}}\tau_{q,r,s+\frac{1}{2}} - T_{q,r,s-\frac{1}{2}}\tau_{q,r,s-\frac{1}{2}}}{\Delta s}$$

$$= \kappa \frac{\mathbf{X}_{q,r,s+1} - 2\mathbf{X}_{q,r,s} + \mathbf{X}_{q,r,s-1}}{\Delta s^2}$$

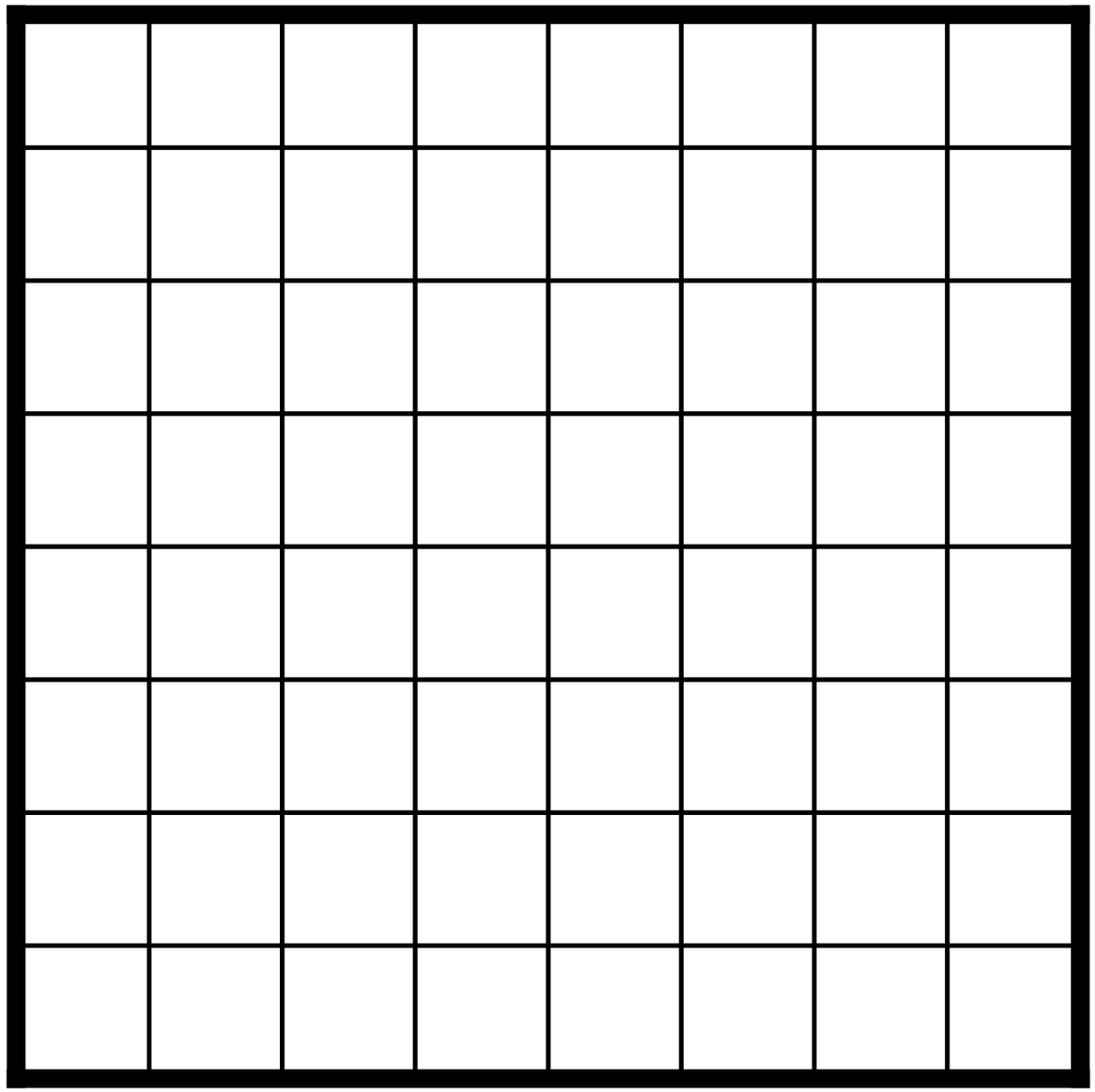
An *adaptive* IB approach to fluid-structure interaction

- Discretize the Eulerian equations on a *locally refined* Cartesian grid;
- Discretize the Lagrangian equations on a moving curvilinear mesh; and
- Discretize the interaction equations via regularized delta functions.



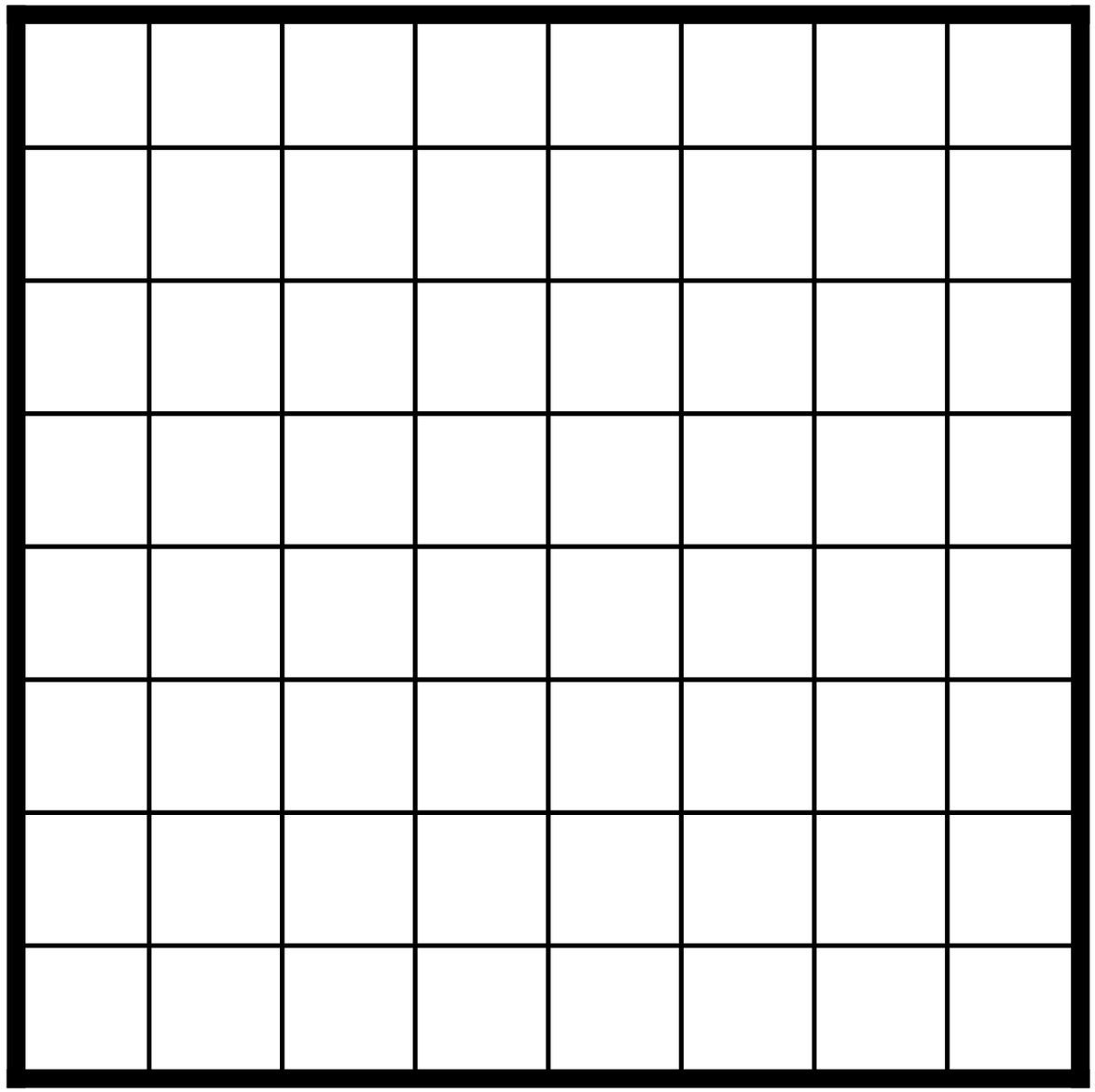
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



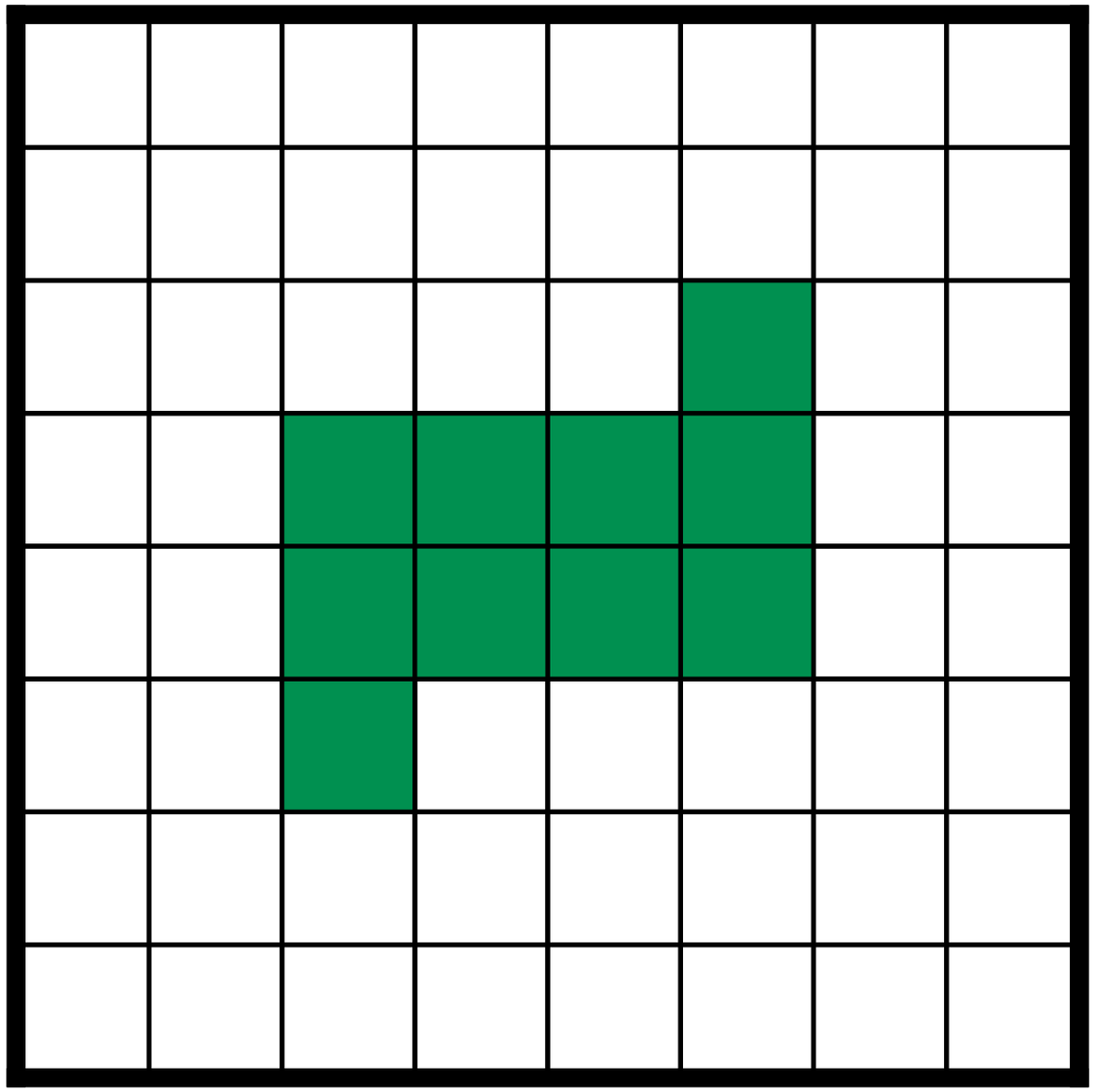
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



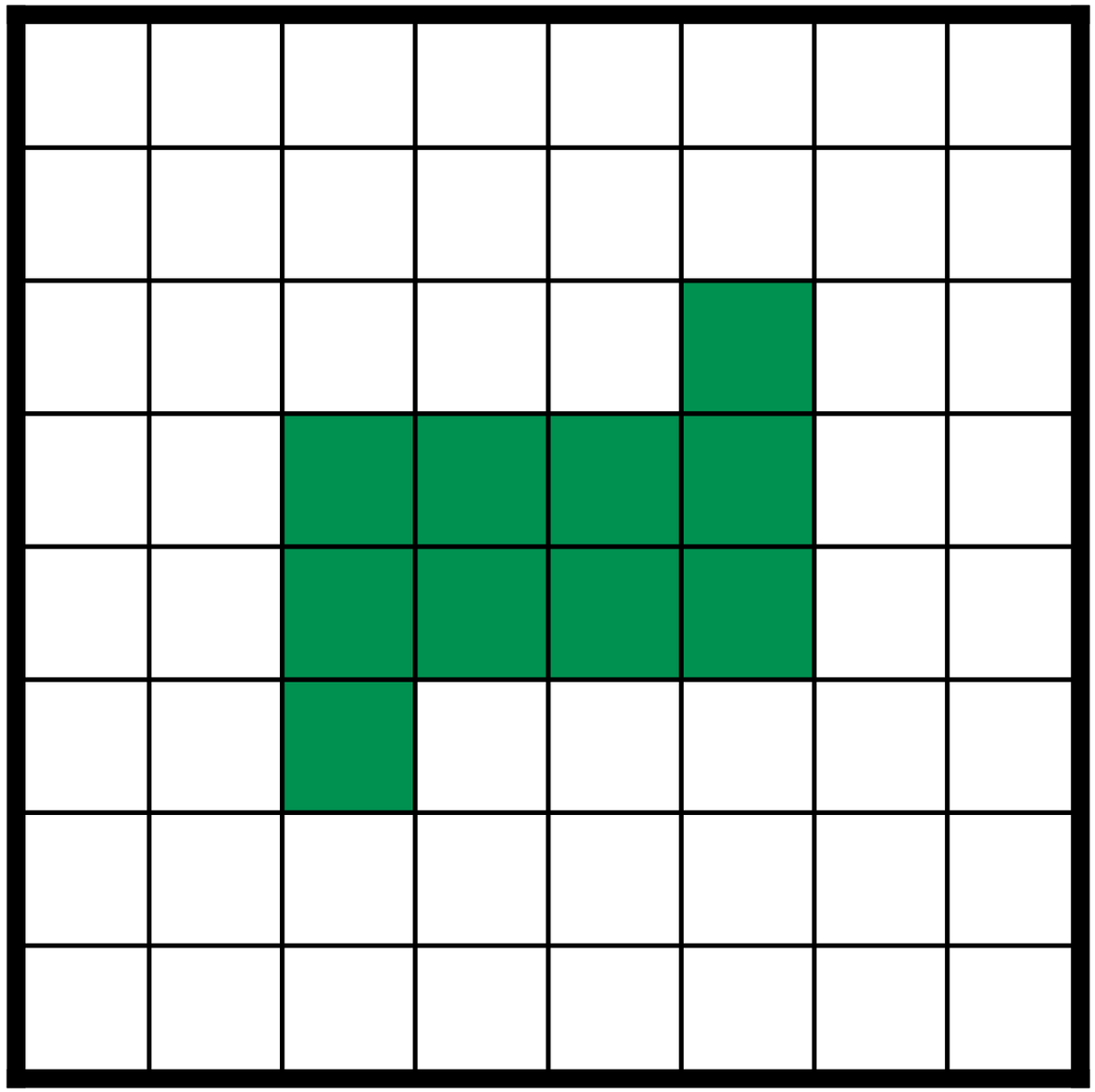
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



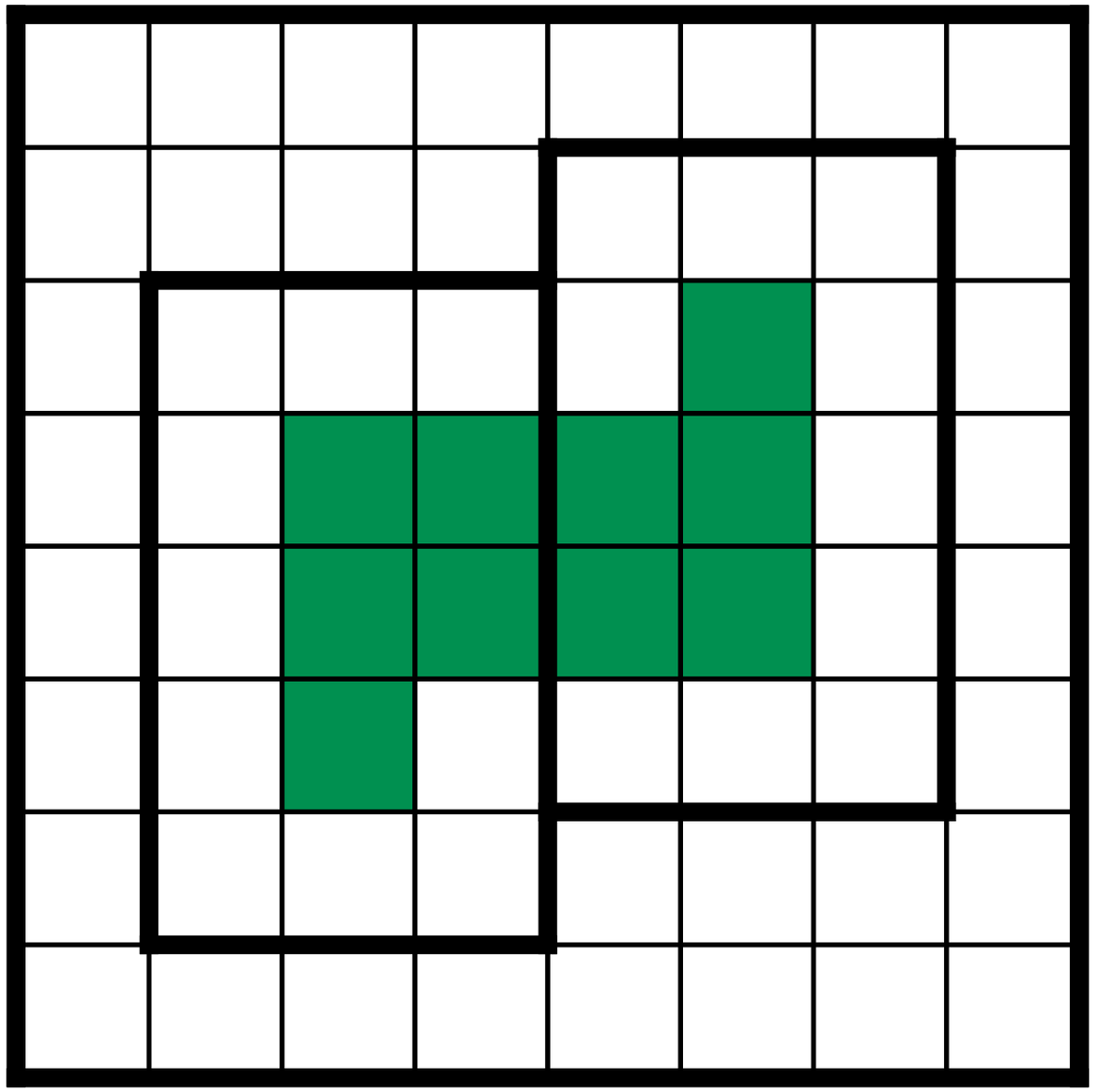
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



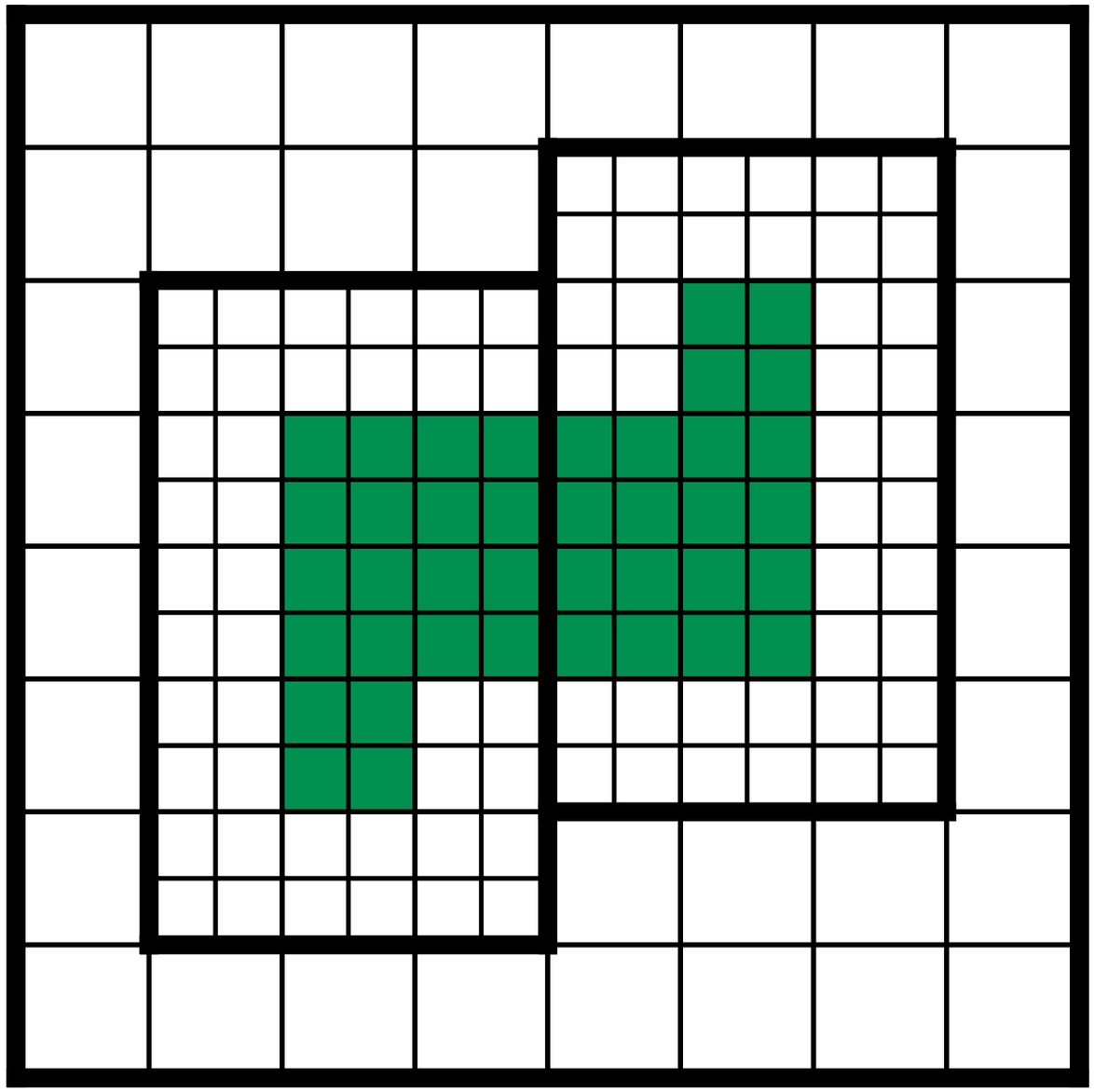
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



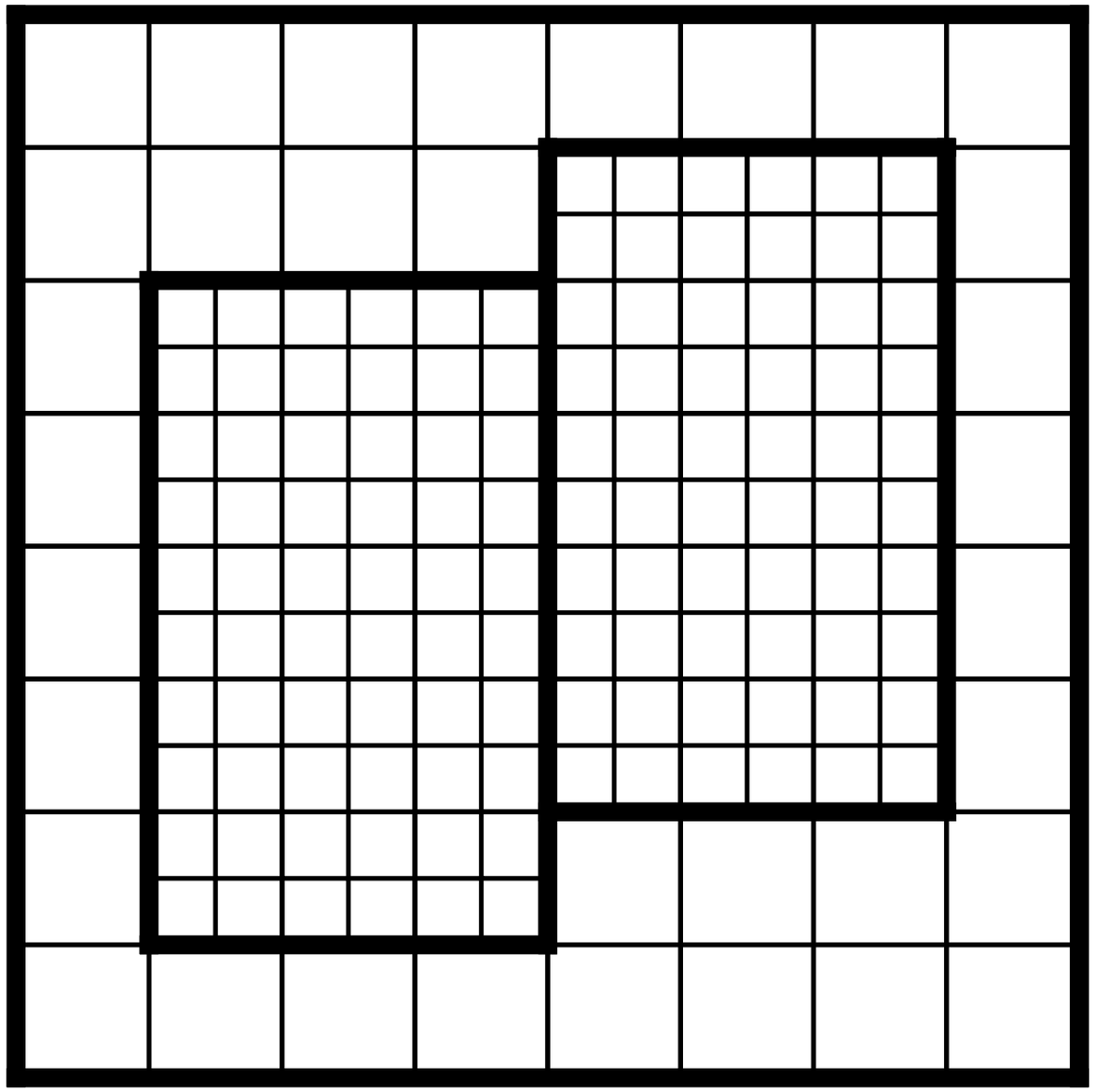
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



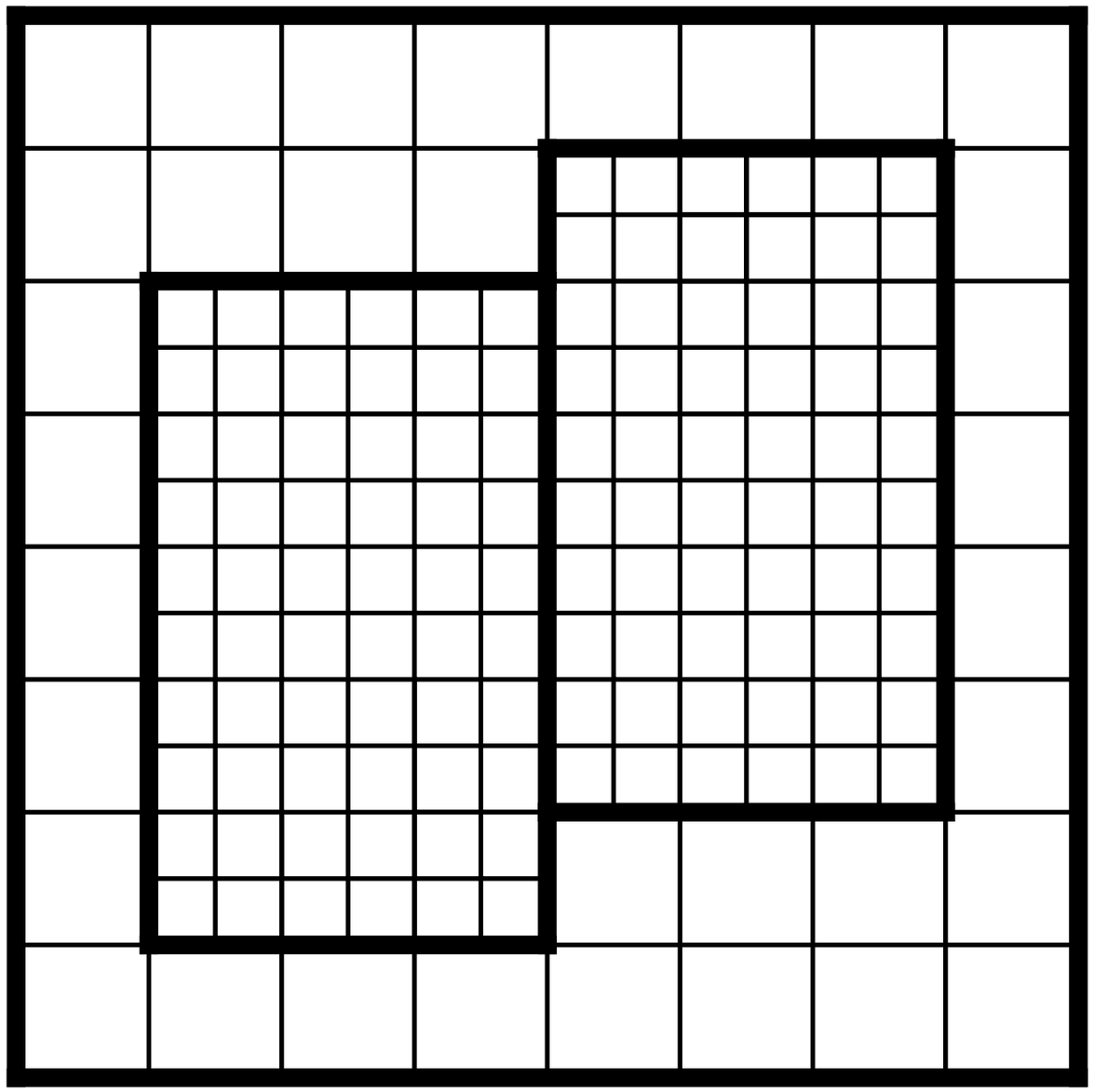
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



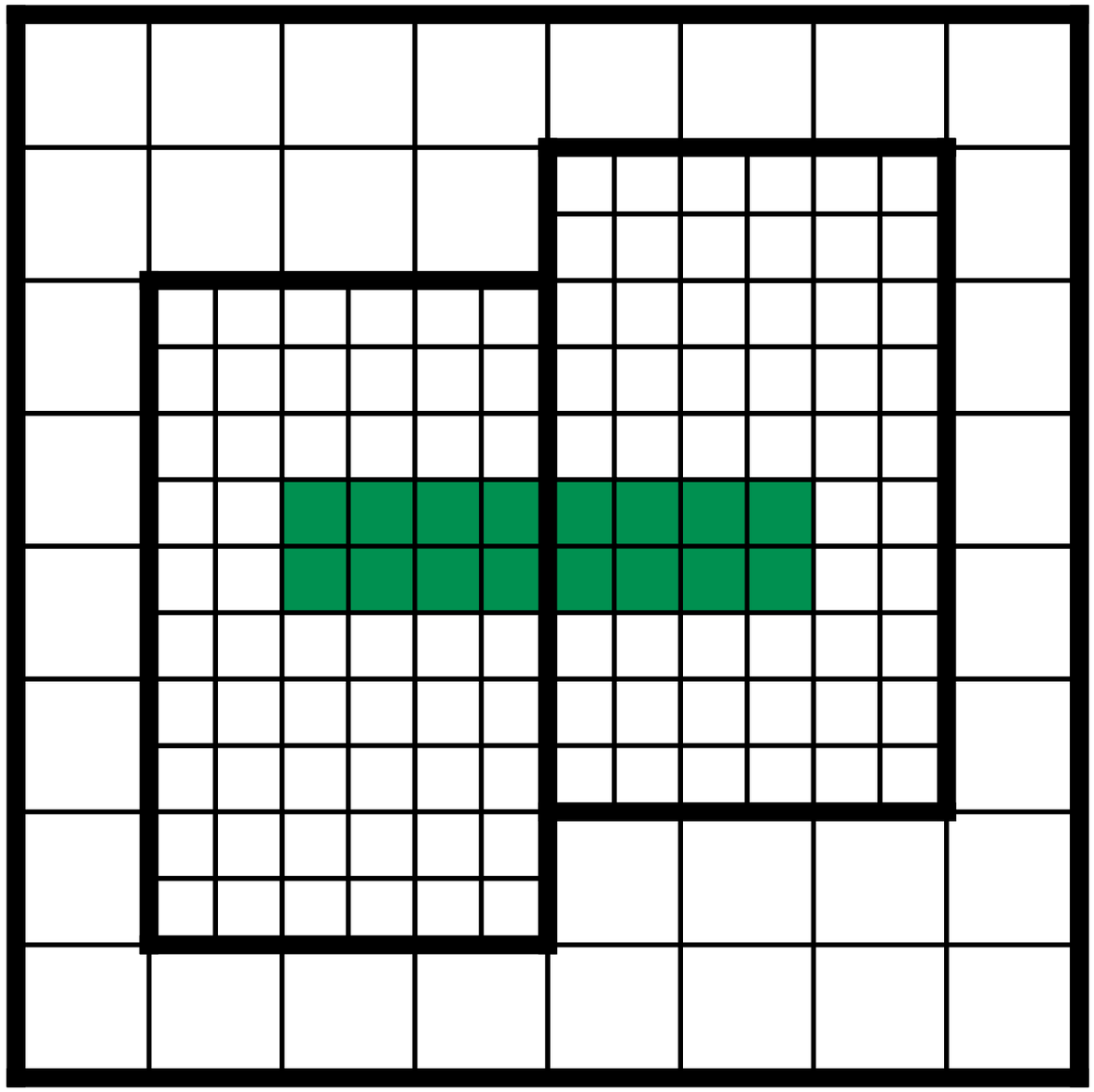
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



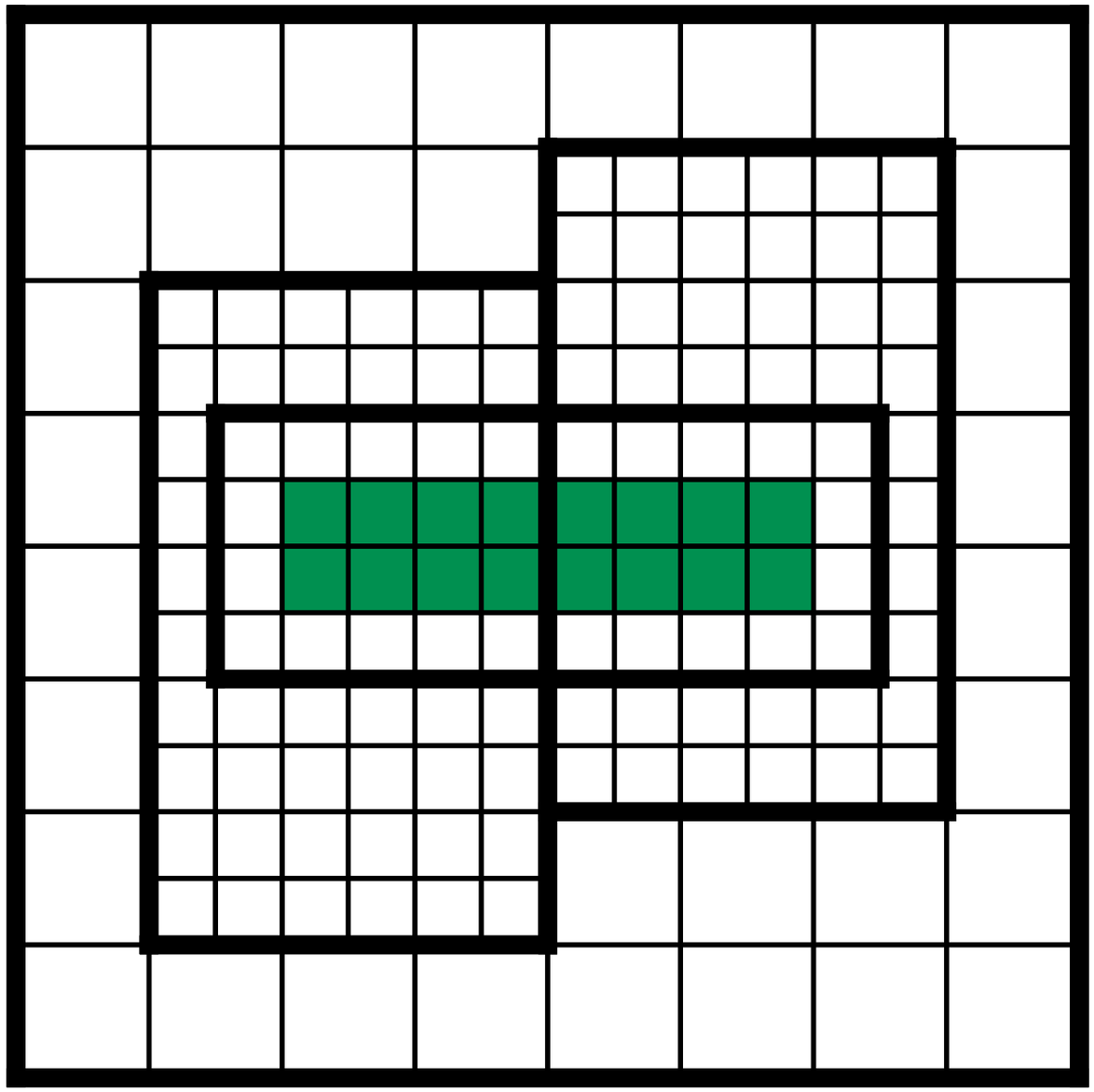
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



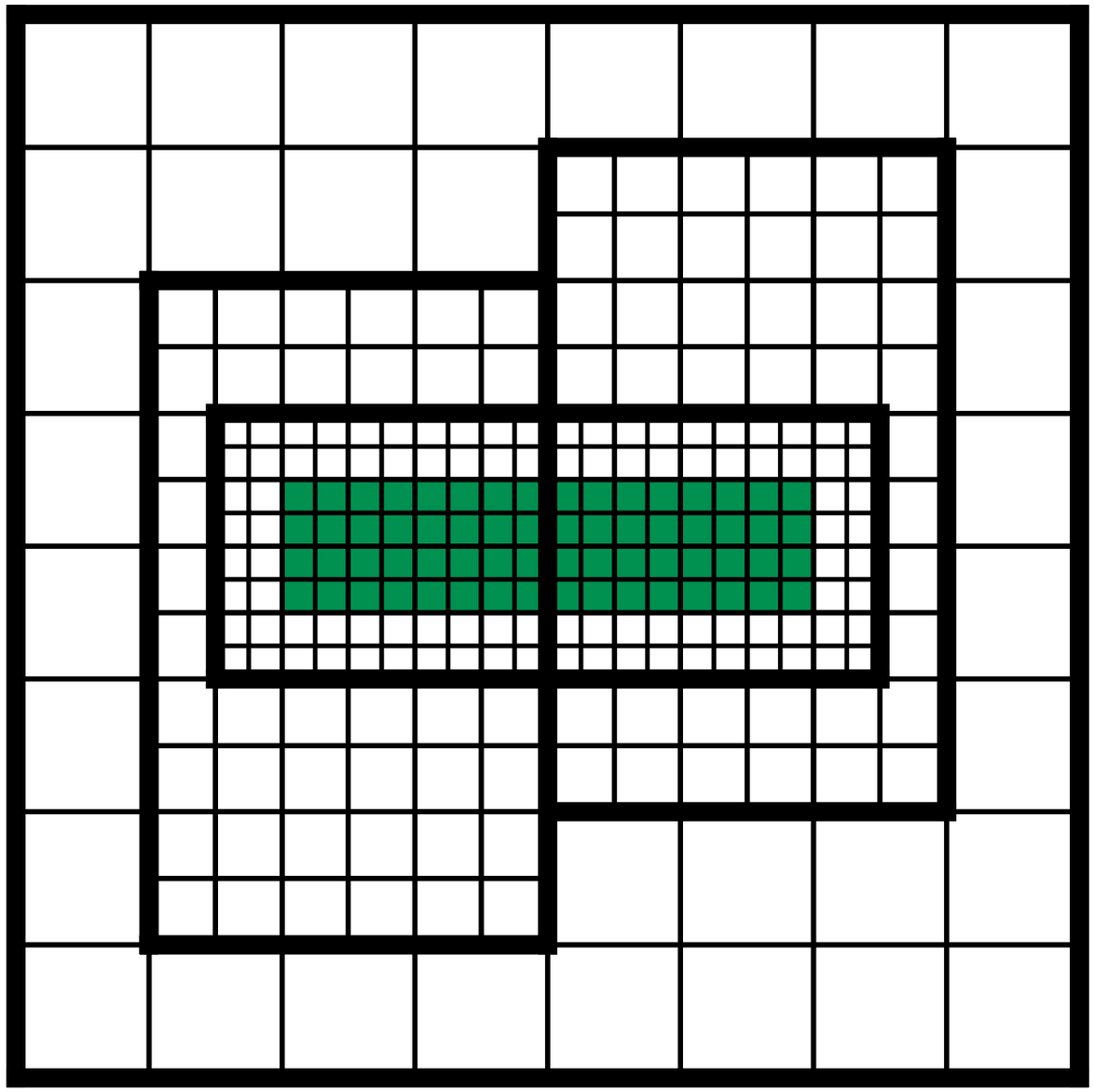
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



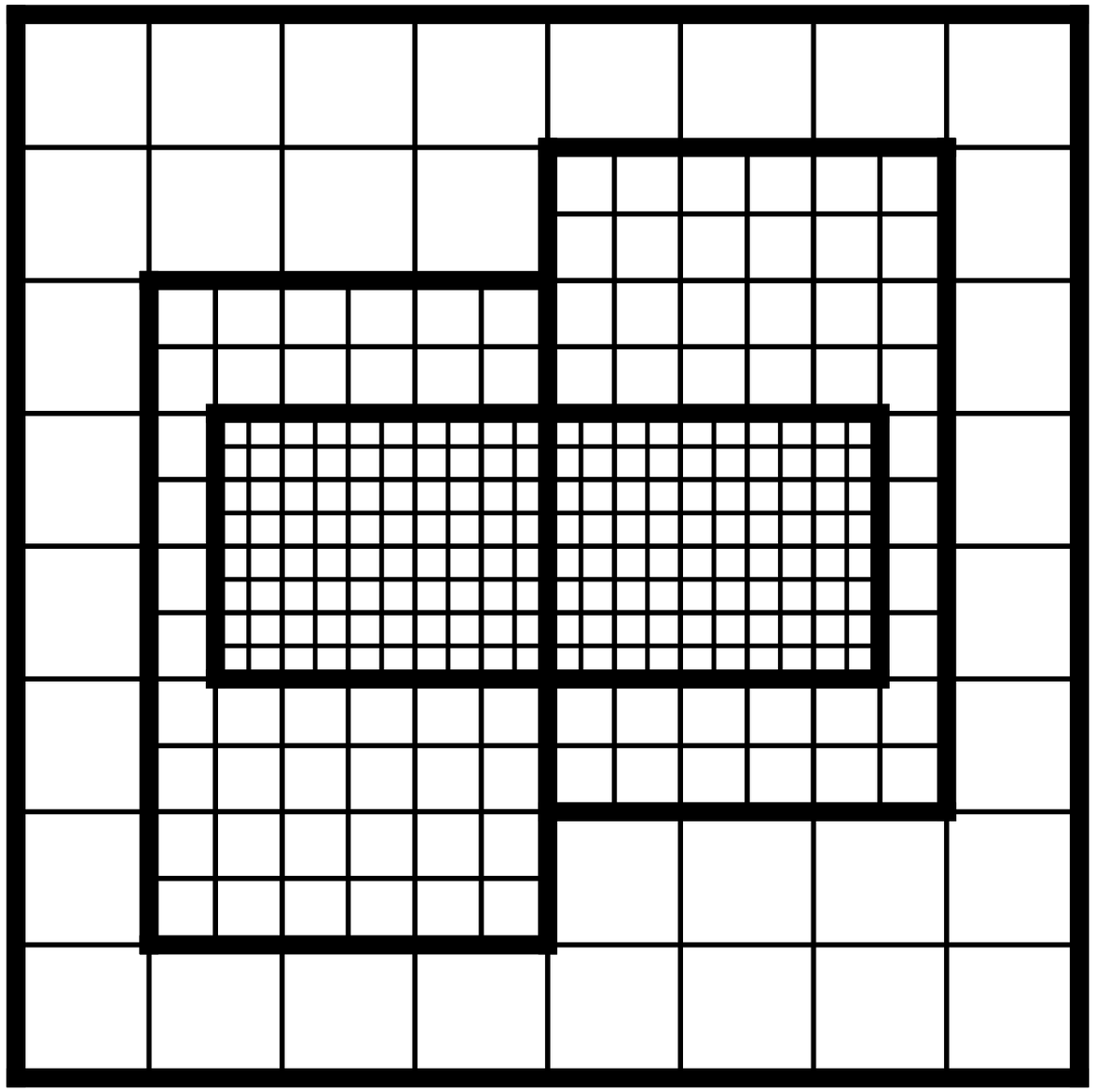
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.



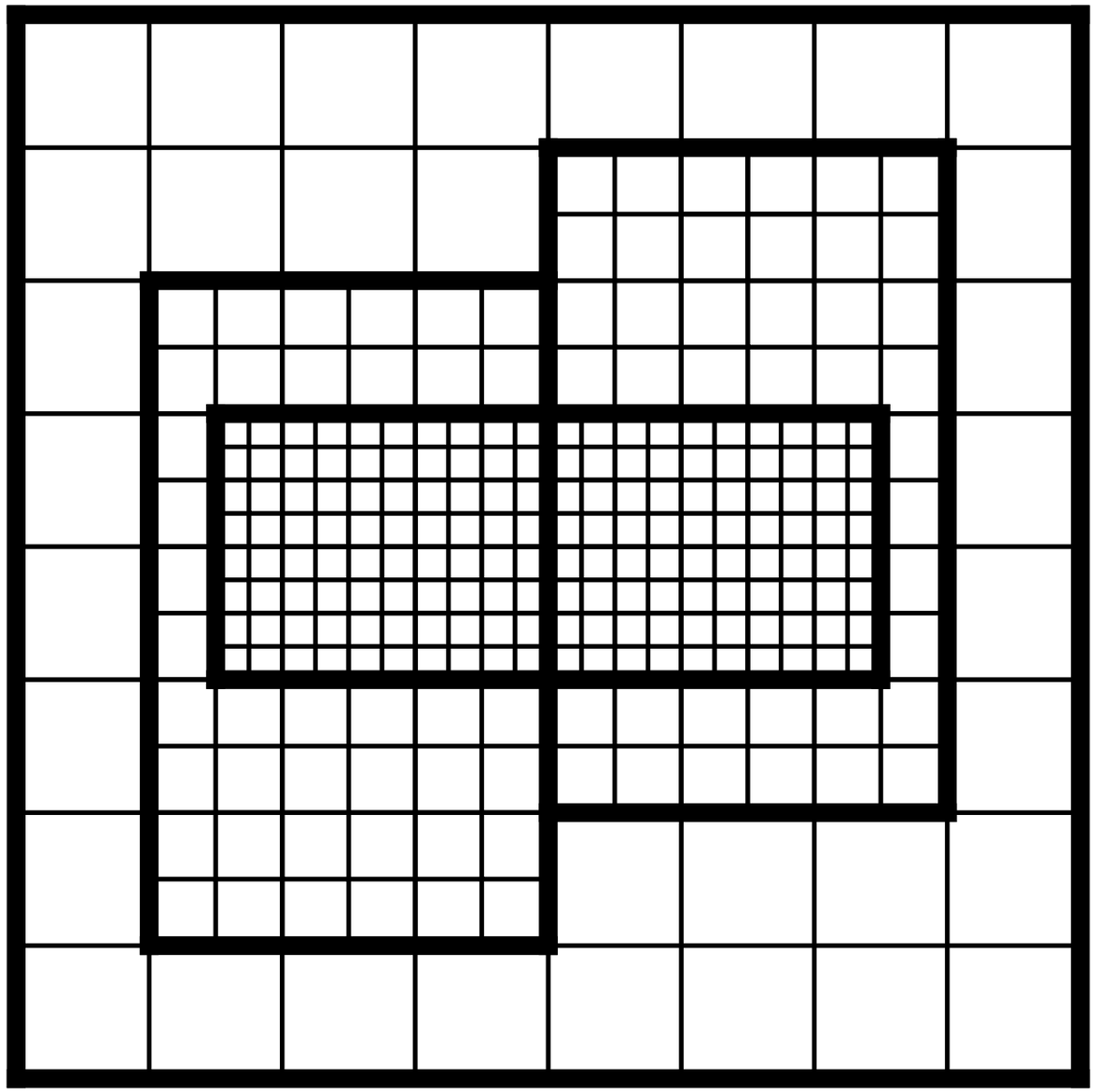
Structured Adaptive Mesh Refinement (AMR)

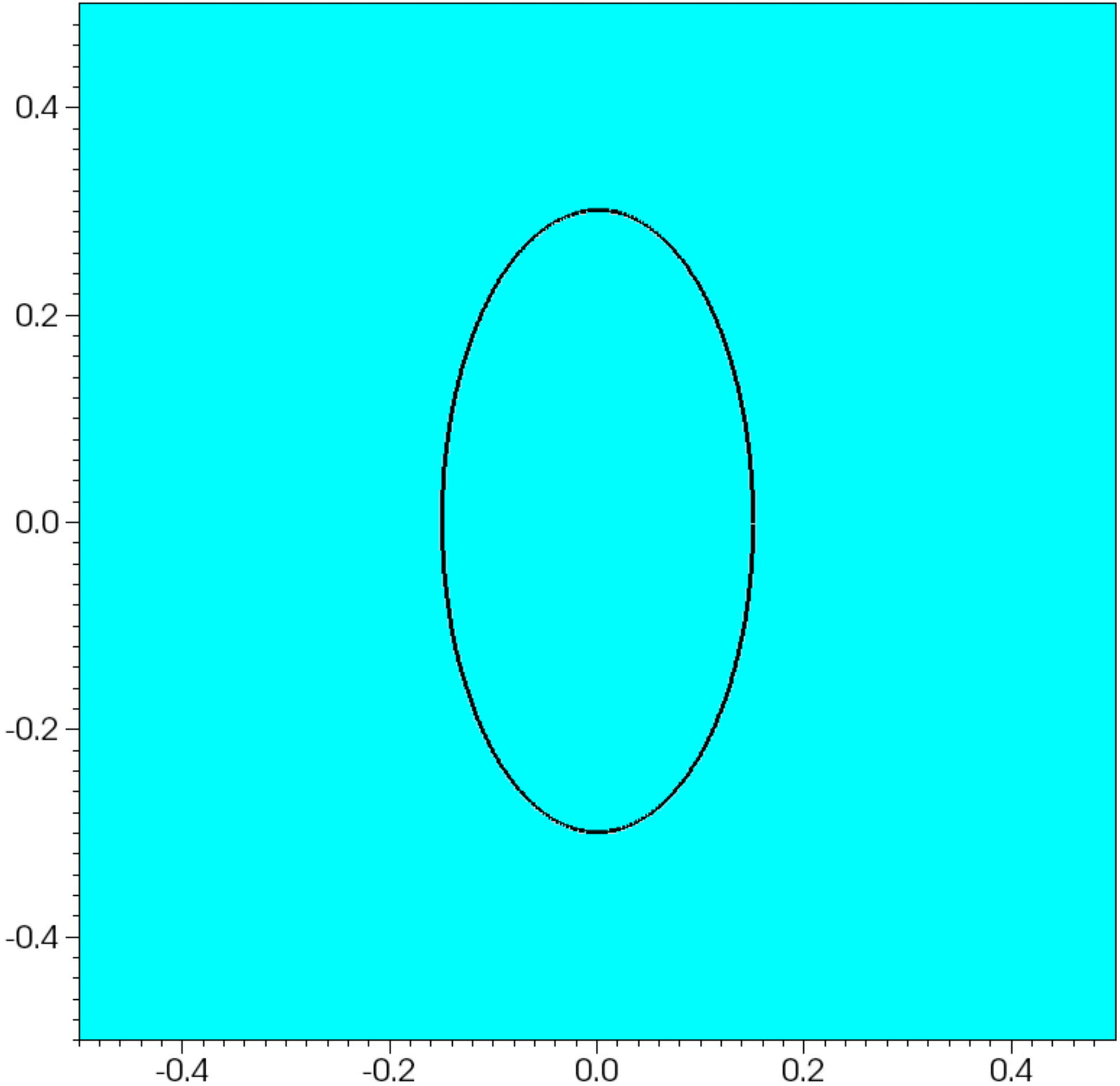
- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.

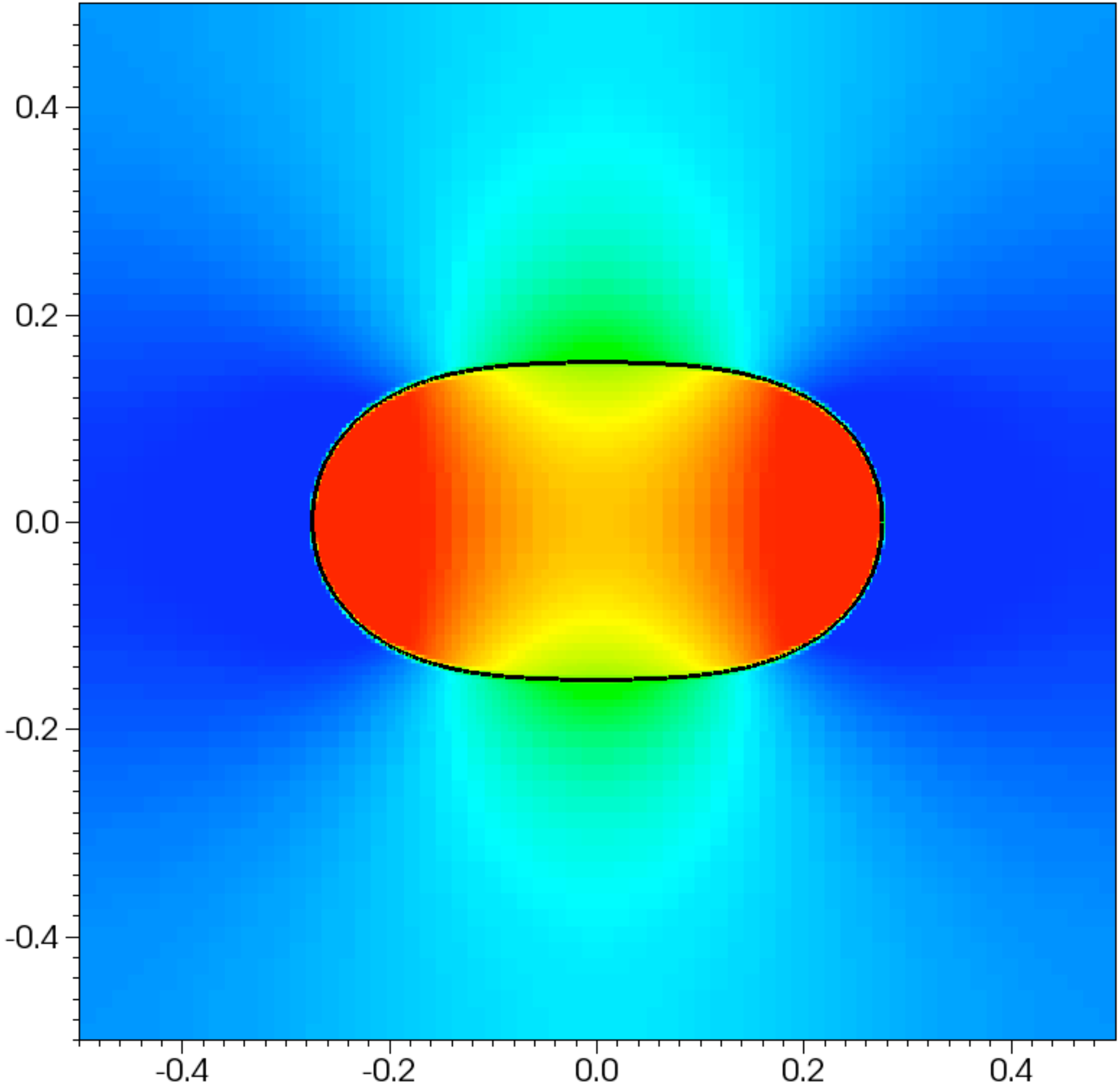


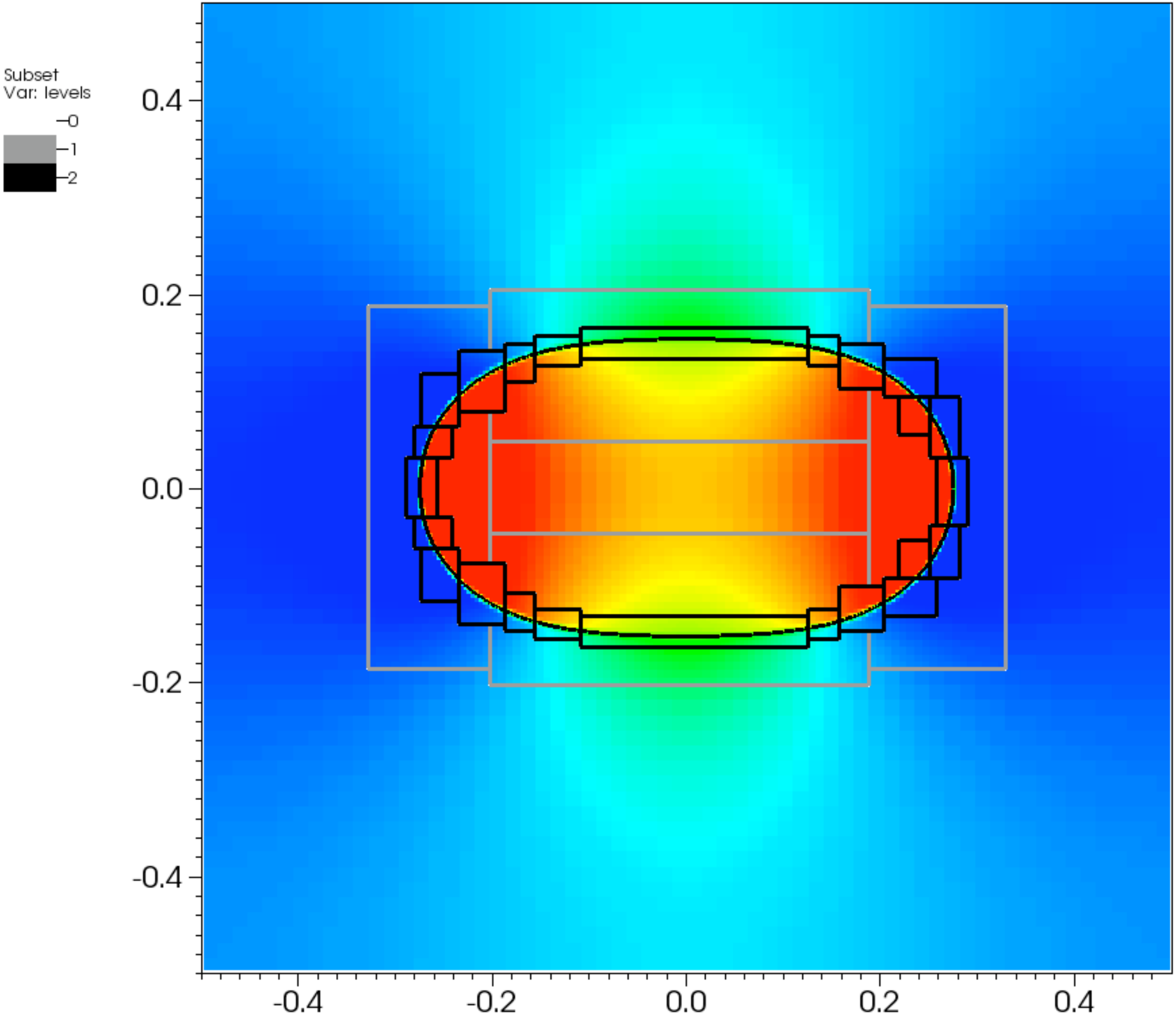
Structured Adaptive Mesh Refinement (AMR)

- Start with a global coarse grid that covers the physical domain
- Indicate the portions of the domain that require higher resolution by tagging cells for refinement
- Generate the rectangular patches that comprise the next finer level of the grid (e.g., using the Berger-Rigoutsos point clustering algorithm)
- Continue recursively until the maximum number of levels have been generated
- Cells are tagged for refinement whenever they contain curvilinear mesh nodes or large values of $\|\boldsymbol{\omega}(\mathbf{x},t)\| = \|\nabla \times \mathbf{u}(\mathbf{x},t)\|$.




0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4


0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4

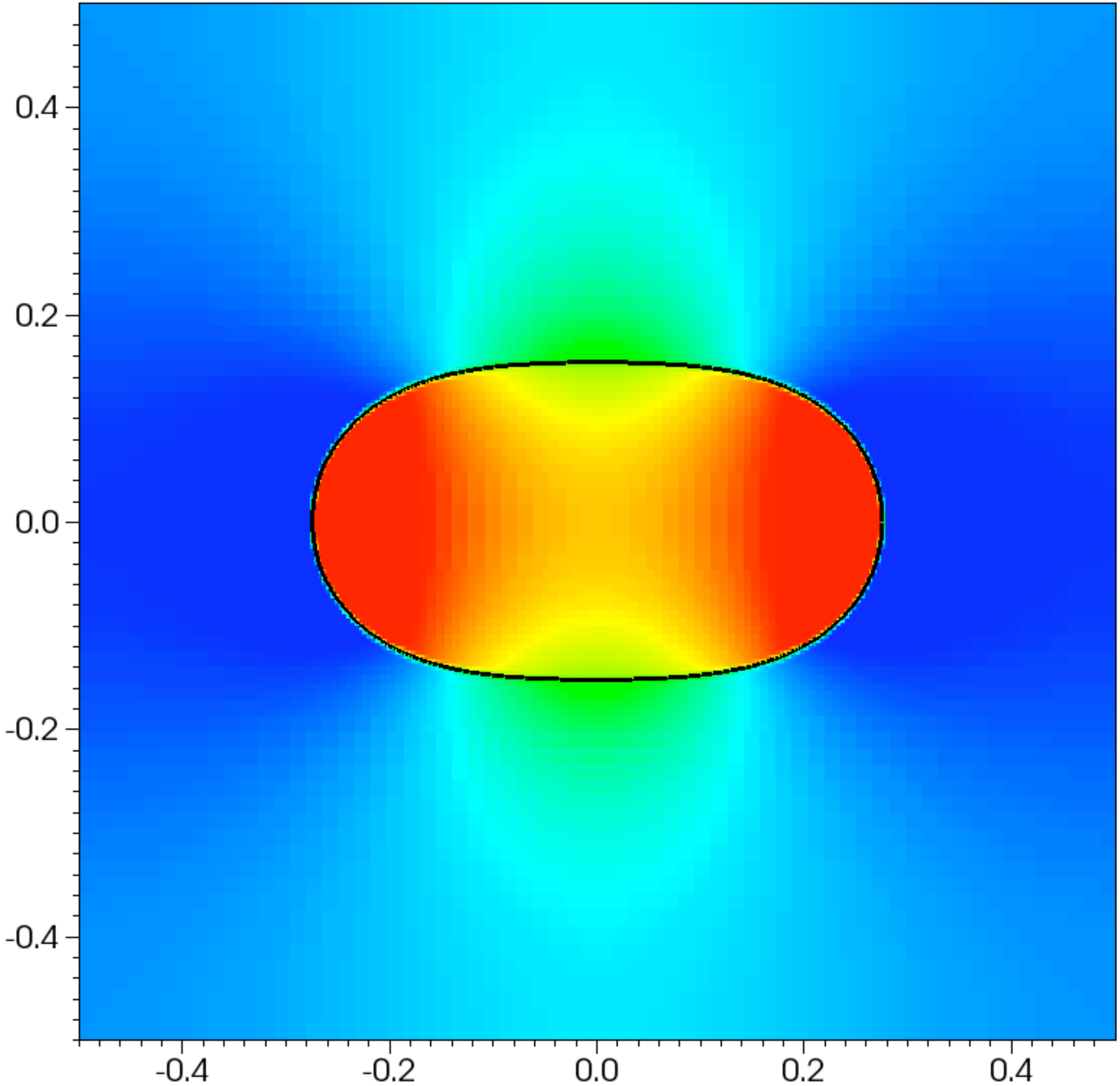

Subset
Var: levels
0
1
2
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4

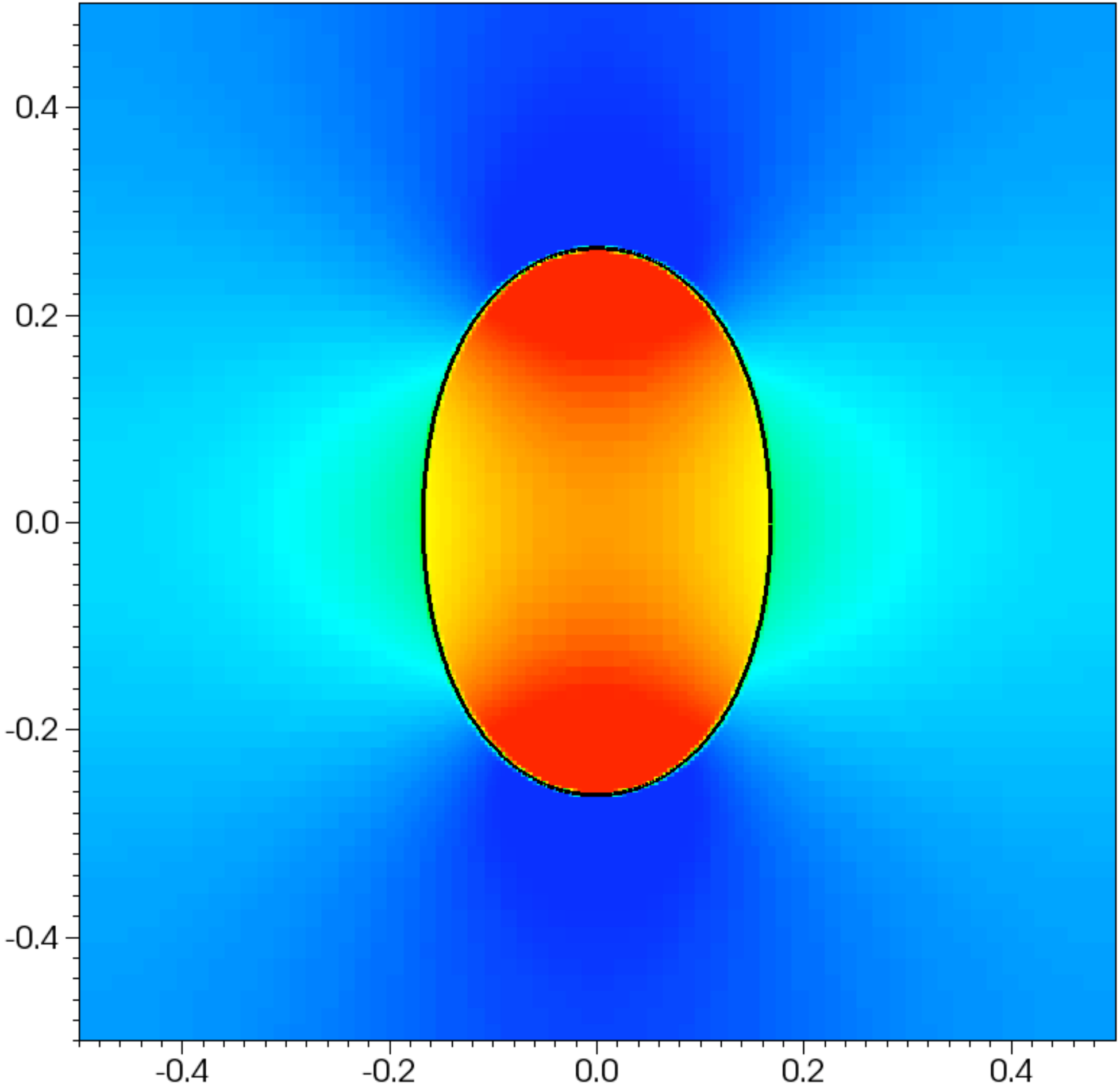
Subset
Var: levels
0
1
2

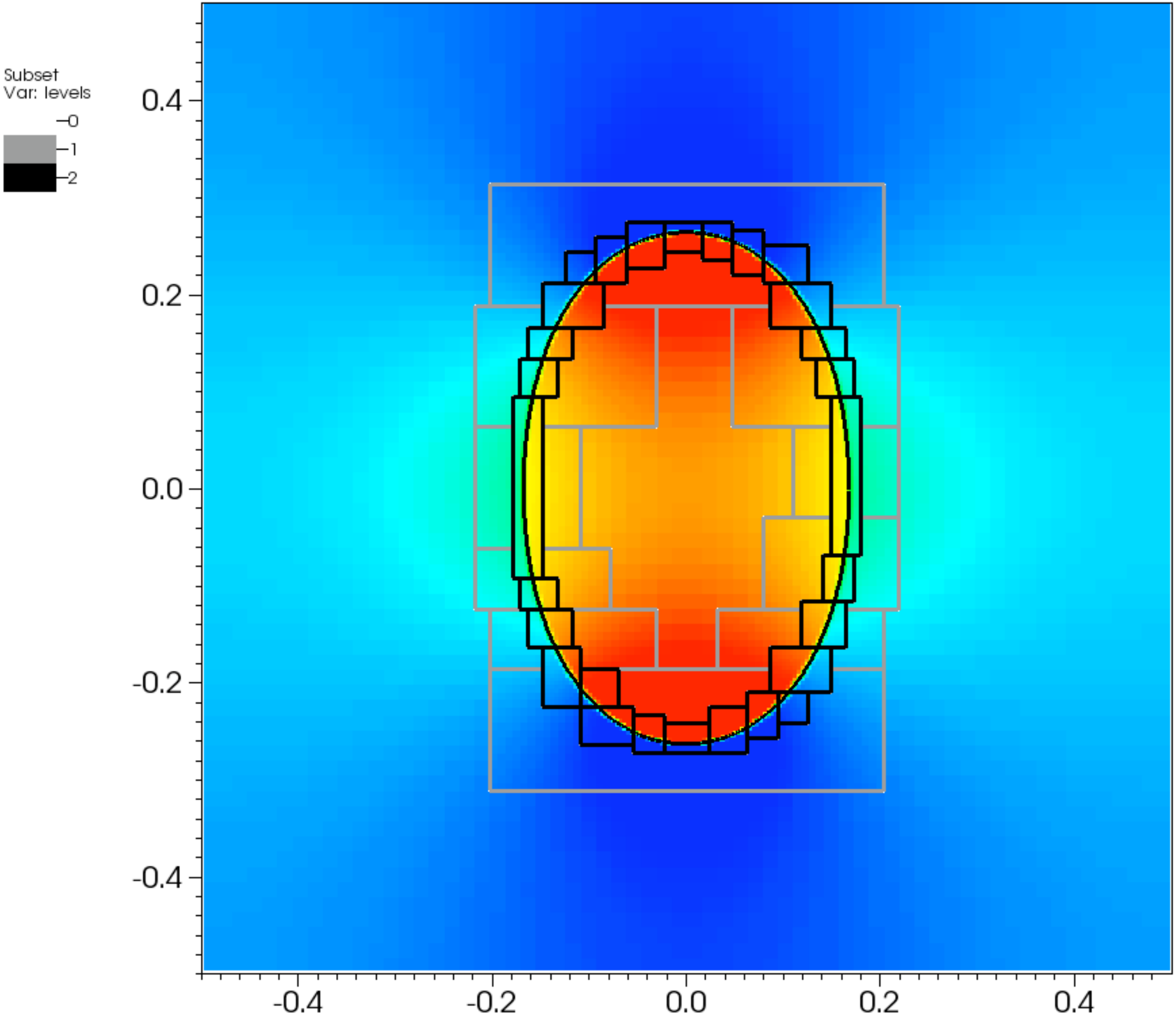
Mesh
Var: amr_mesh

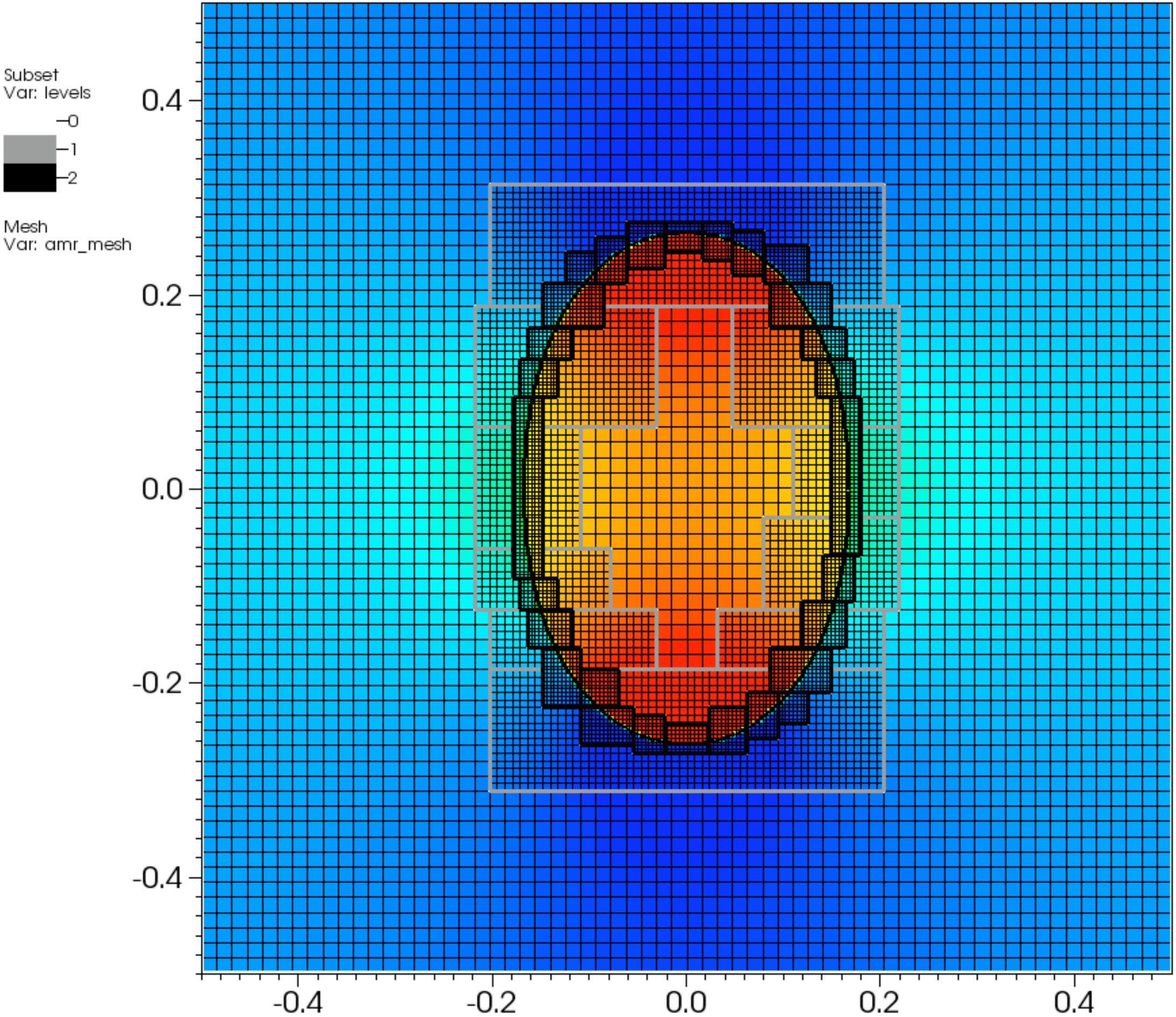
0.4
0.2
0.0
-0.2
-0.4

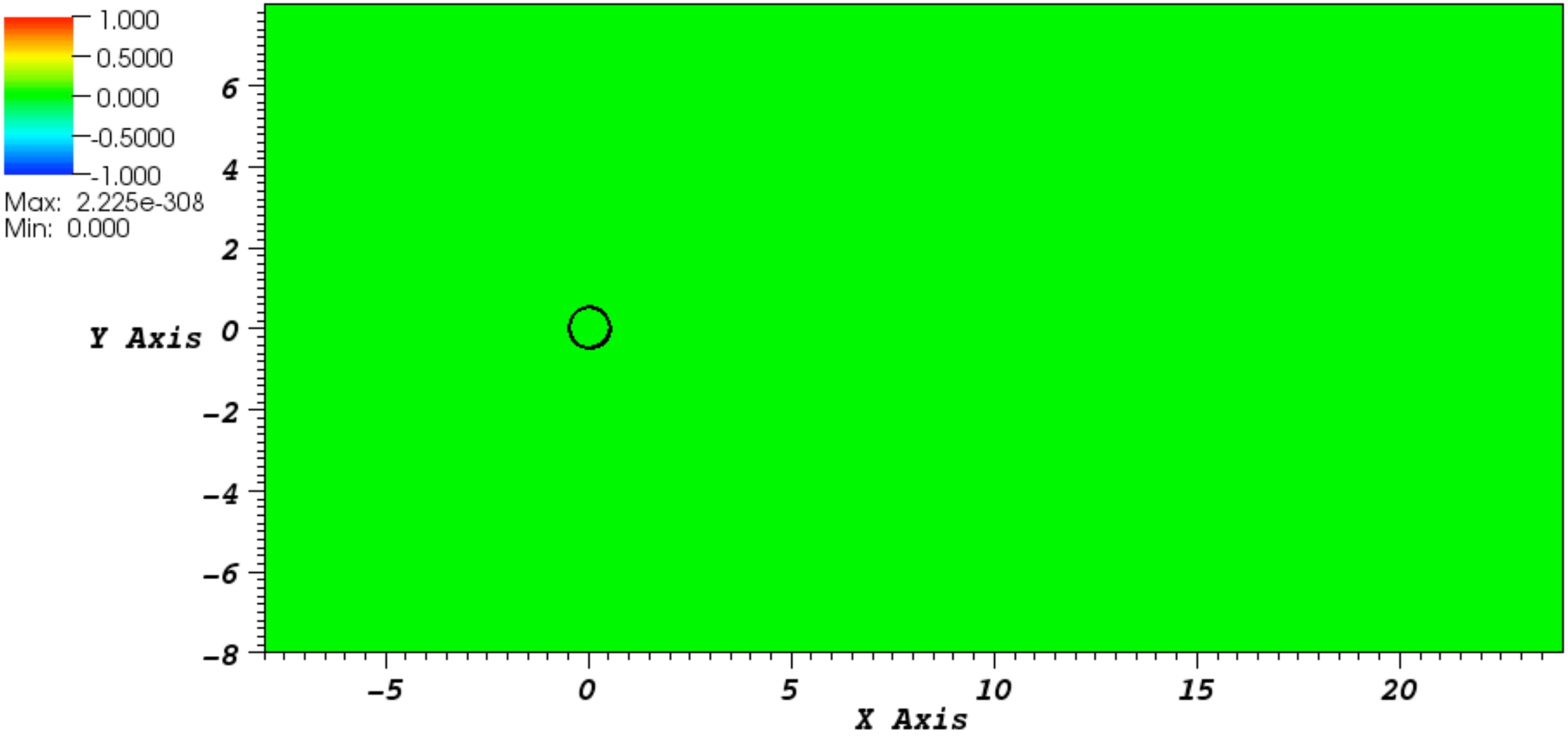
-0.4
-0.2
0.0
0.2
0.4

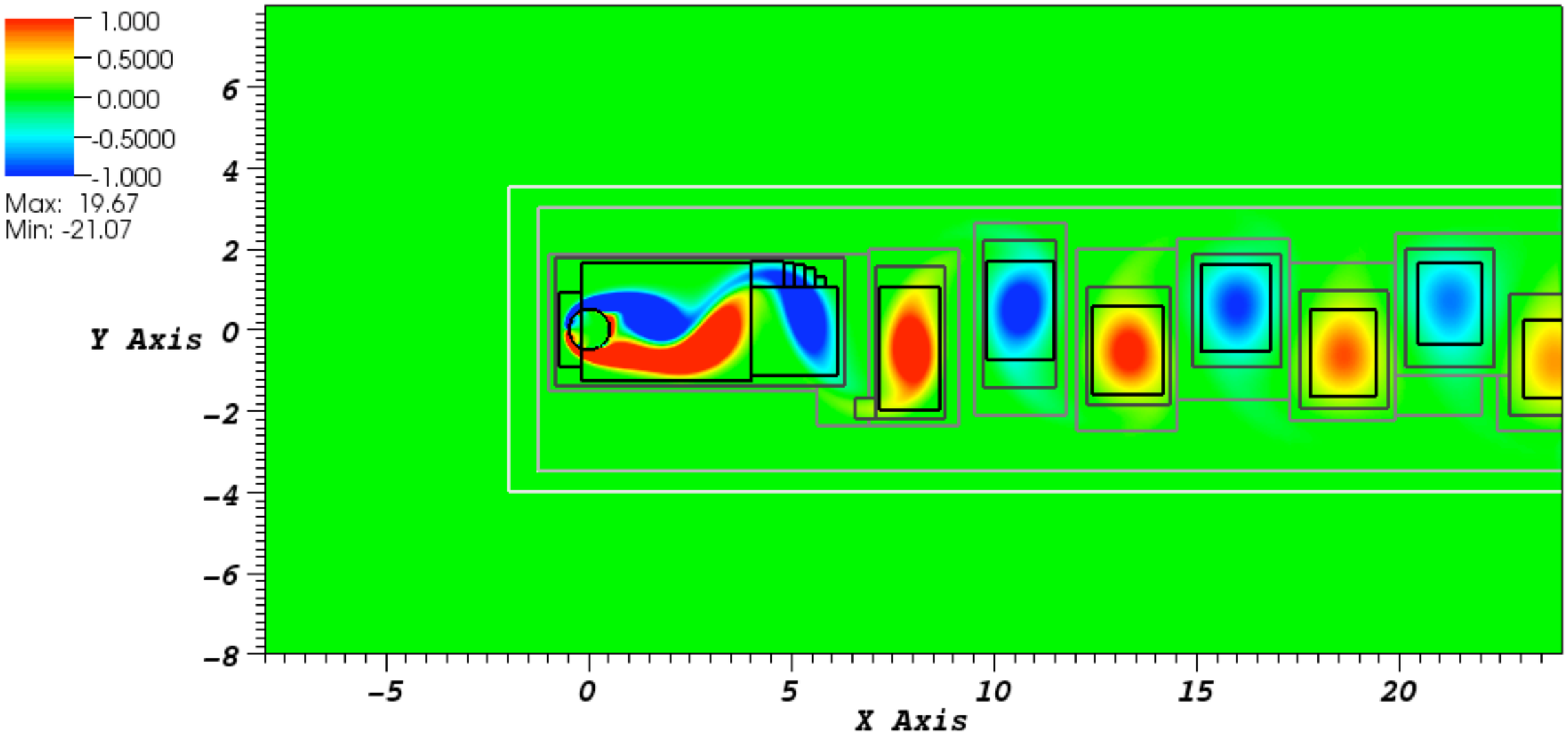

0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4

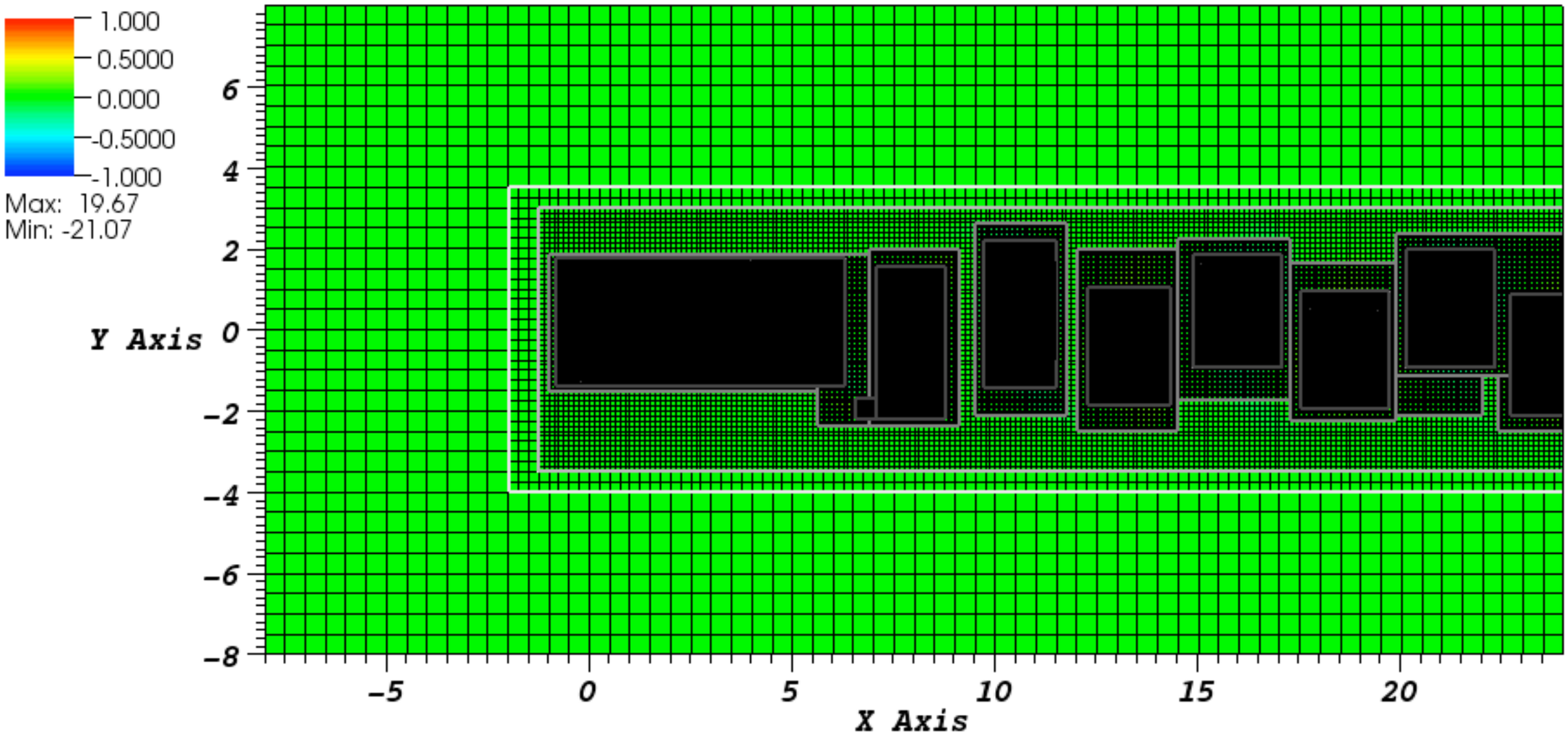

0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4

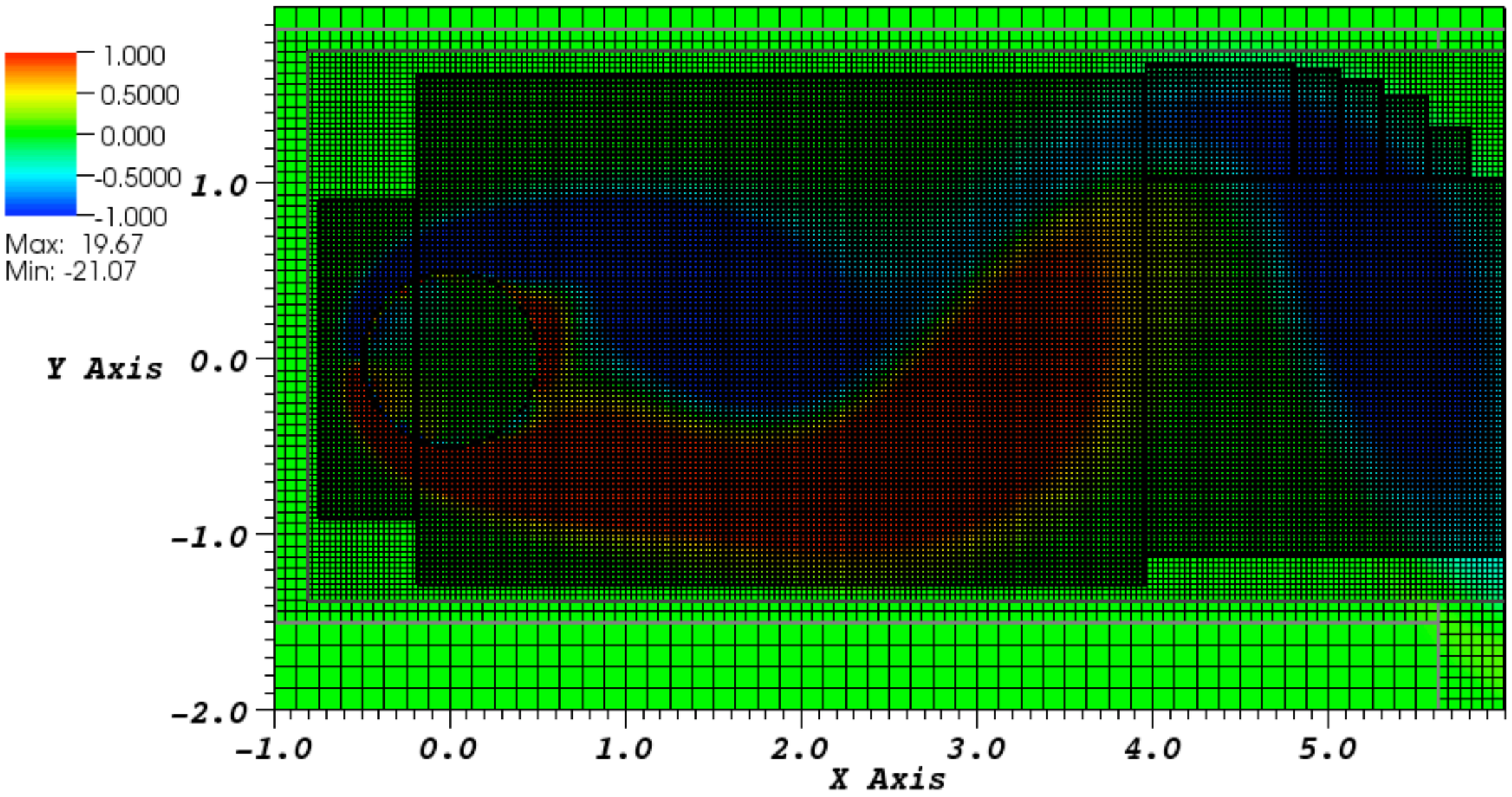

Subset
Var: levels
0
1
2
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4

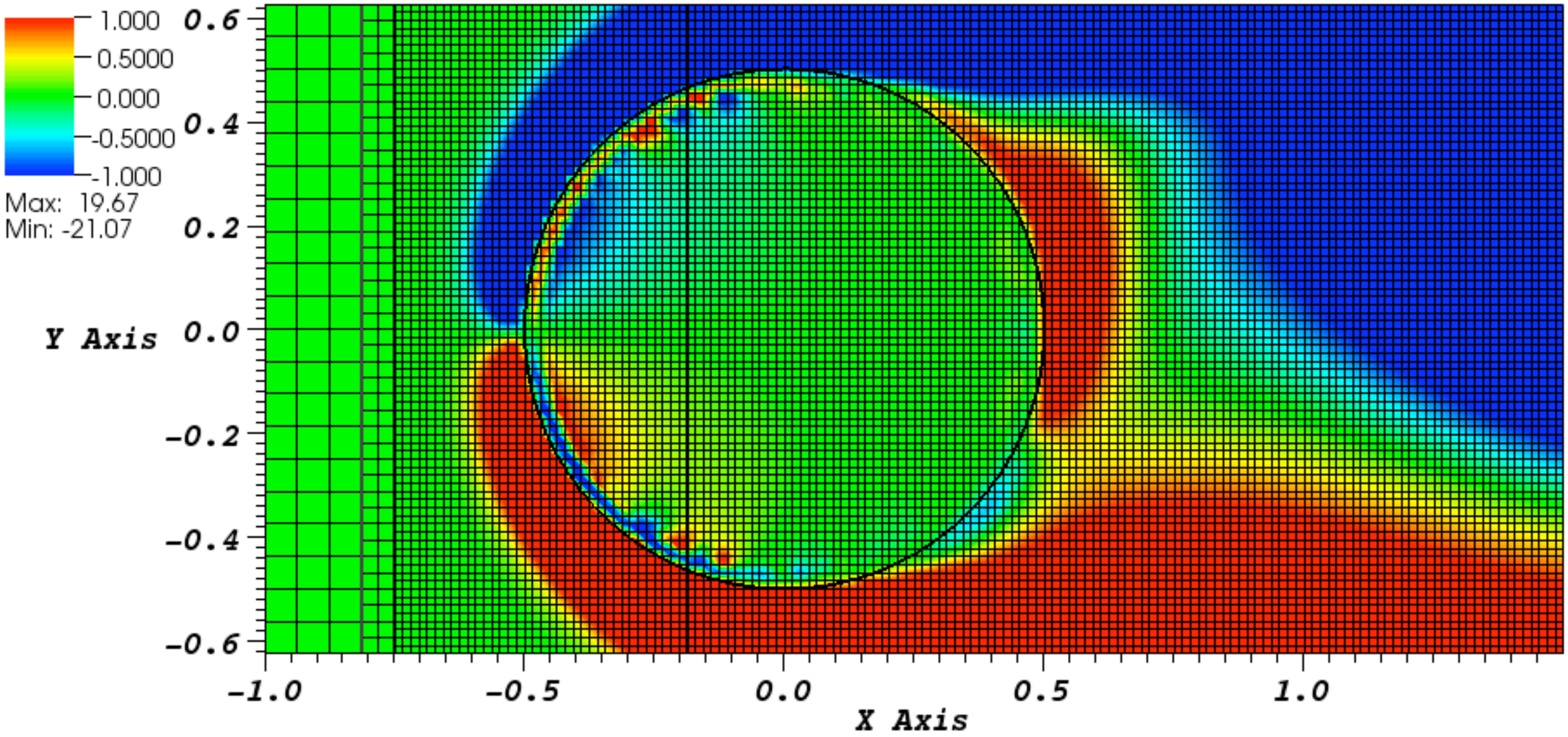

Subset
Var: levels
0
1
2
Mesh
Var: amr_mesh
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4

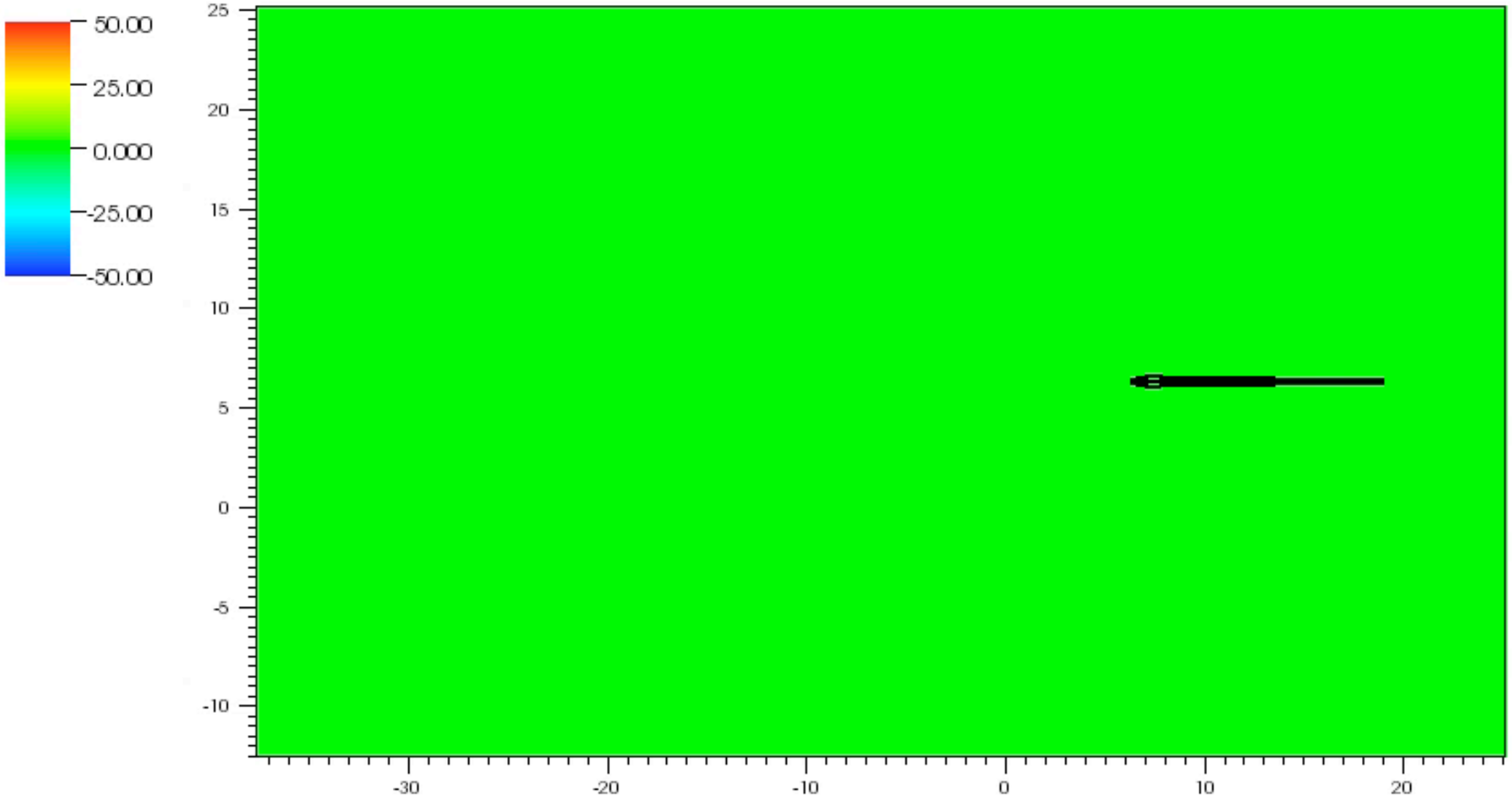

1.000
0.5000
0.000
-0.5000
-1.000
Max: 2.225e-308
Min: 0.000
Y Axis
6
4
2
0
-2
-4
-6
-8
-5
0
5
10
15
20
X Axis

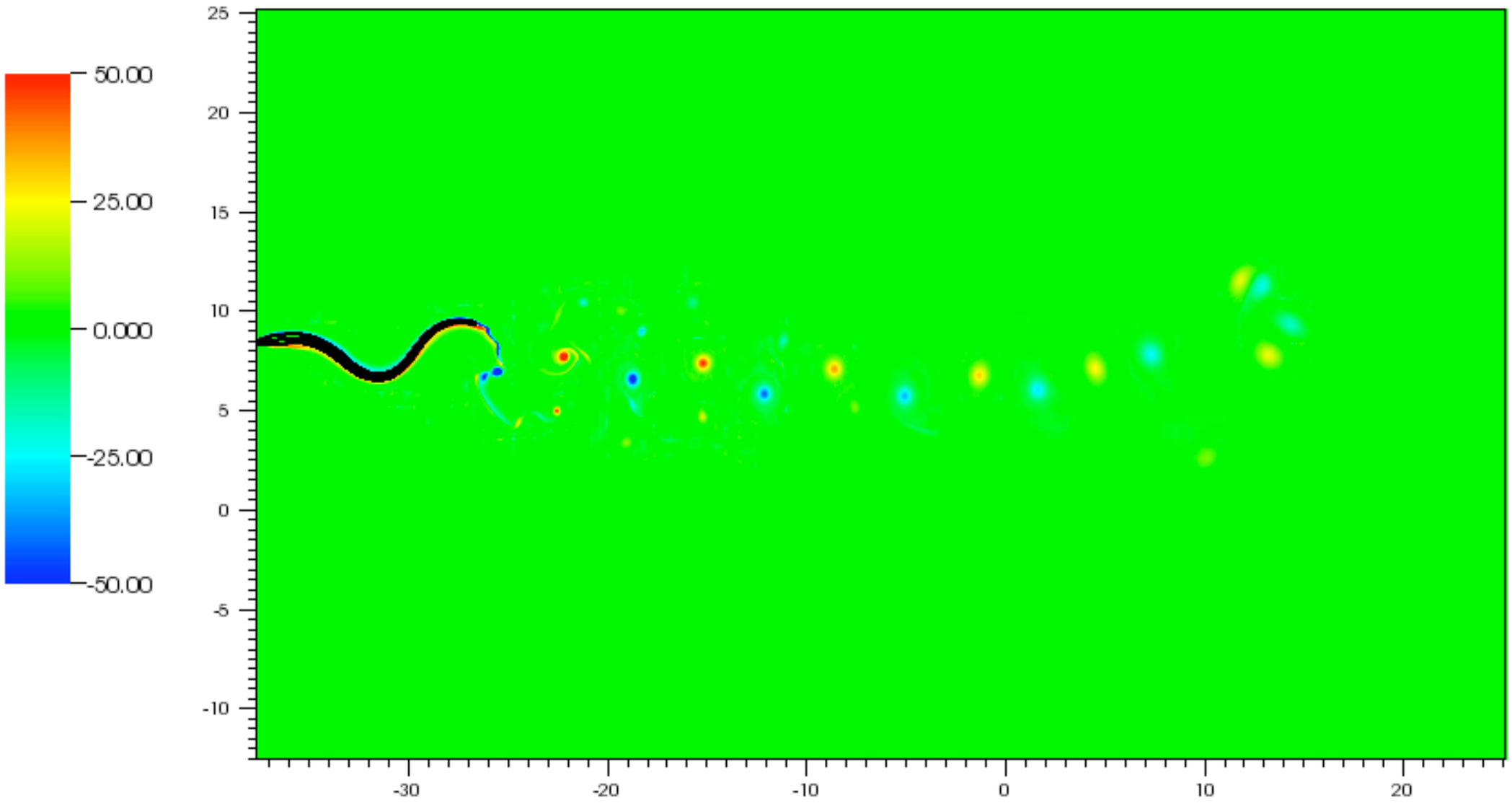

1.000
0.5000
0.000
-0.5000
-1.000
Max: 19.67
Min: -21.07
Y Axis
6
4
2
0
−2
−4
−6
−8
−5
0
5
10
15
20
X Axis

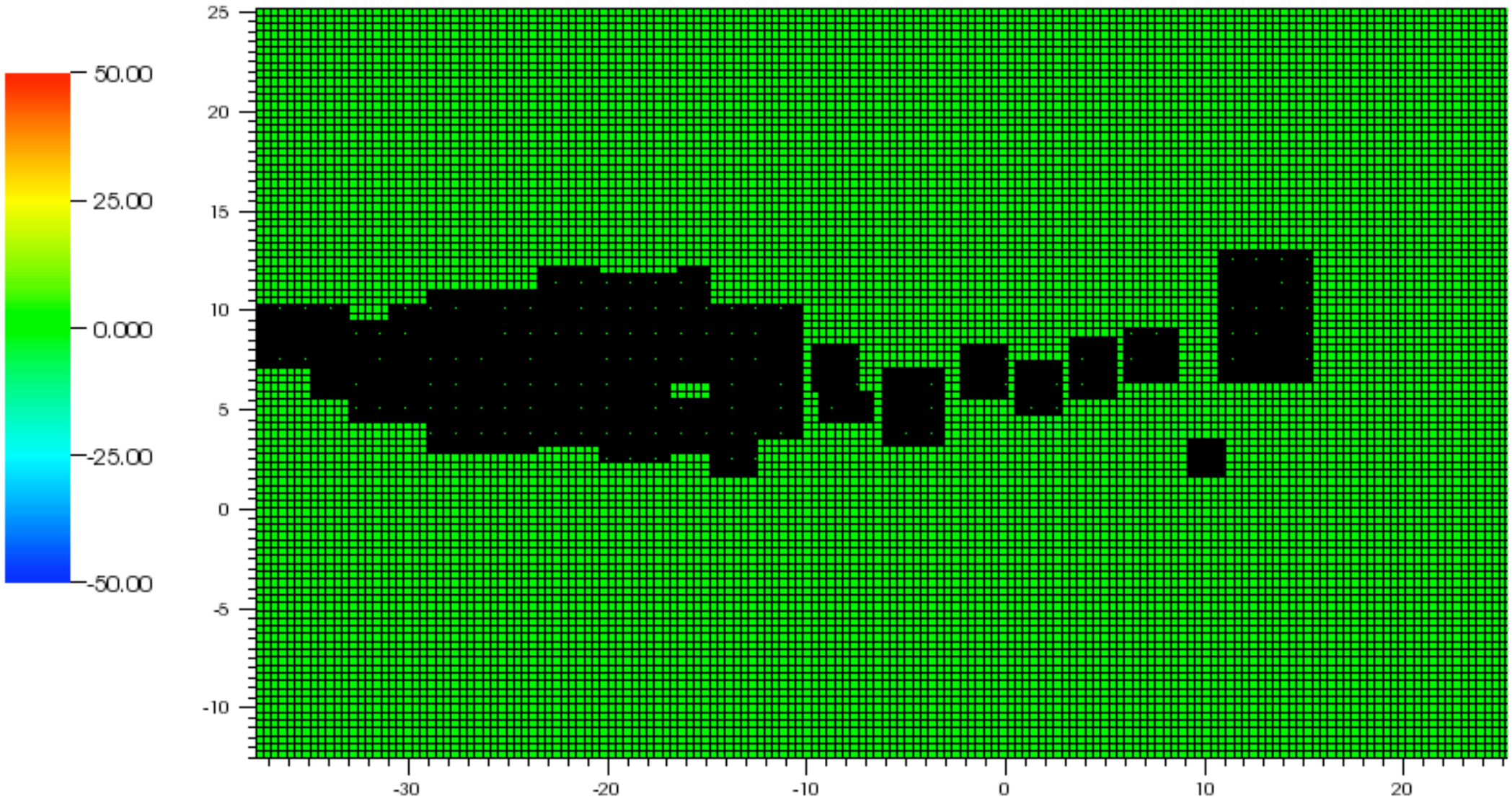

1.000
0.5000
0.000
-0.5000
-1.000
Max: 19.67
Min: -21.07
6
4
2
0
-2
-4
-6
-8
Y Axis
-5
0
5
10
15
20
X Axis

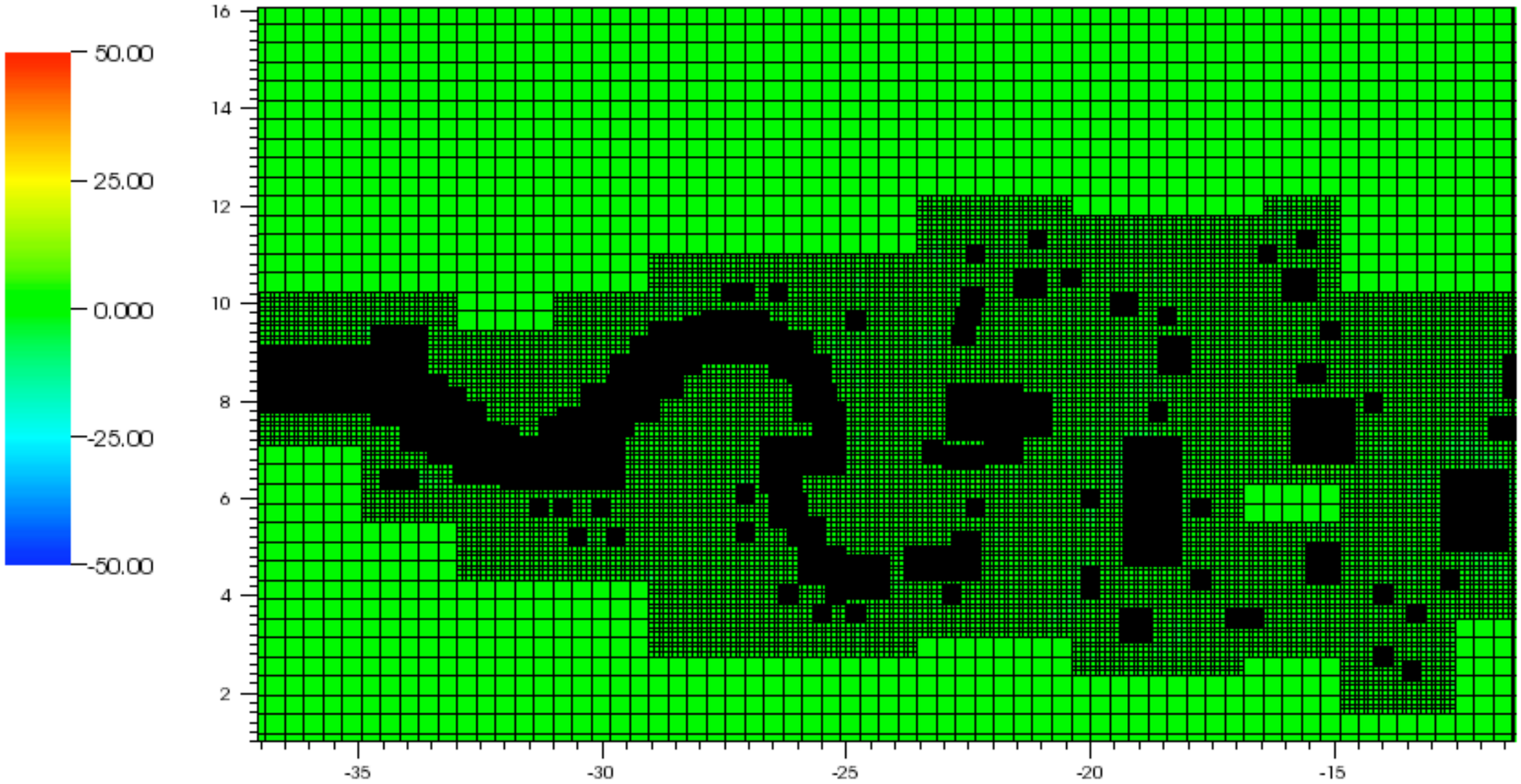

1.000
0.5000
0.000
-0.5000
-1.000
Max: 19.67
Min: -21.07
Y Axis
1.0
0.0
-1.0
-2.0
-1.0
0.0
1.0
2.0
3.0
4.0
5.0
X Axis

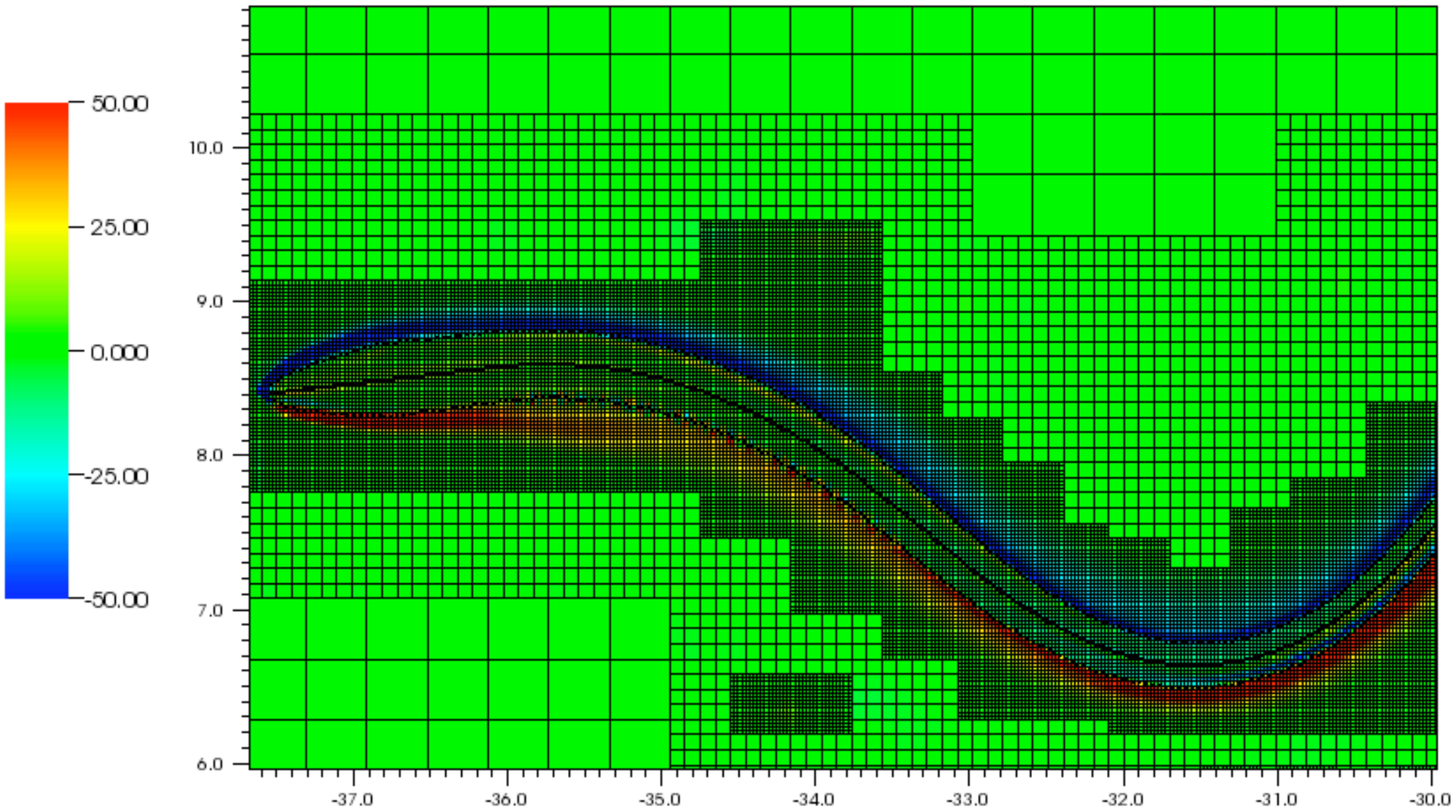

1.000
0.5000
0.000
-0.5000
-1.000
Max: 19.67
Min: -21.07
Y Axis
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-1.0
-0.5
0.0
0.5
1.0
X Axis


50.00
25.00
0.000
-25.00
-50.00
25
20
15
10
5
0
-5
-10
-30
-20
-10
0
10
20

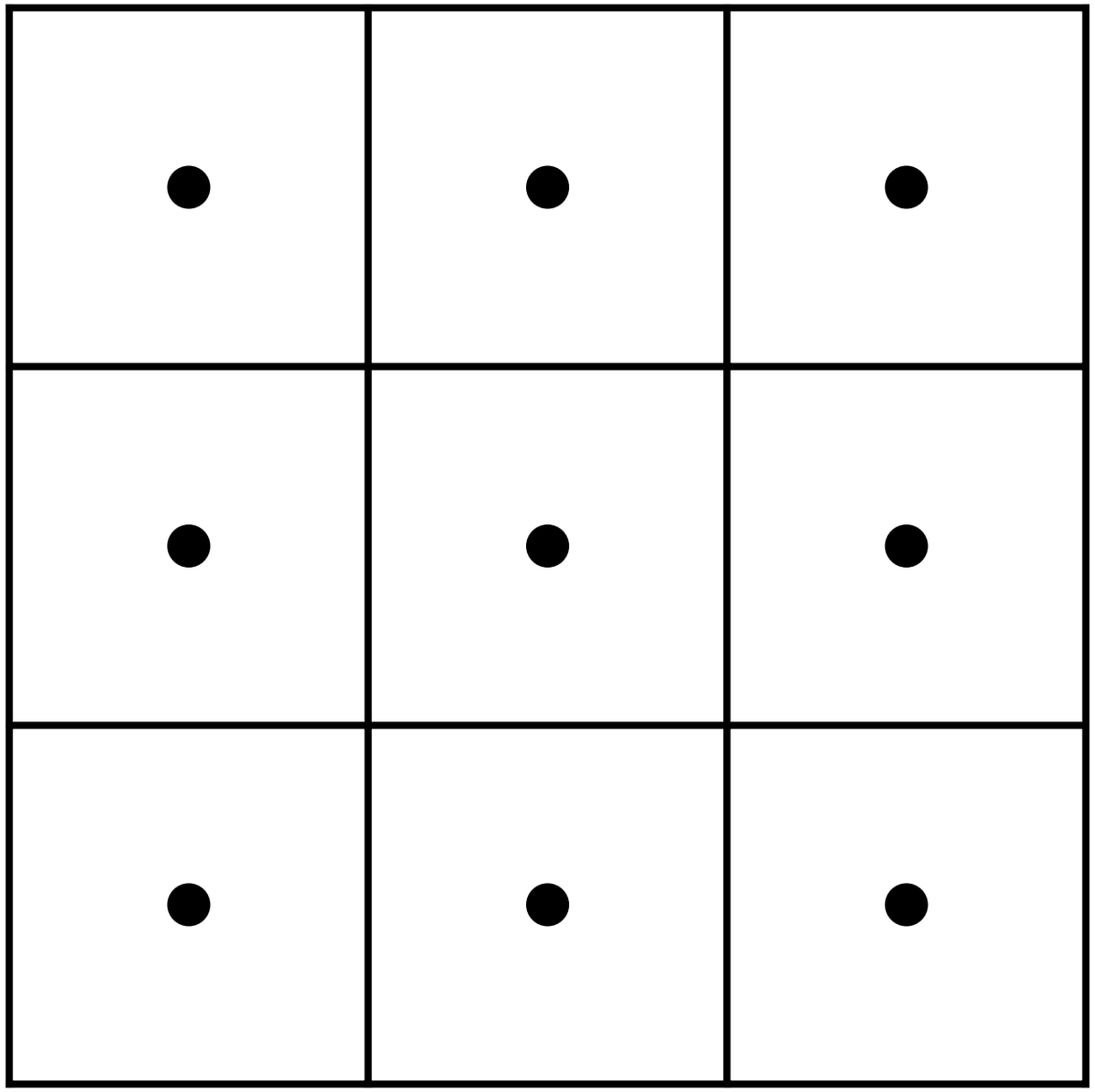

50.00
25.00
0.000
-25.00
-50.00
25
20
15
10
5
0
-5
-10
-30
-20
-10
0
10
20


50.00
25.00
0.000
-25.00
-50.00
25
20
15
10
5
0
-5
-10
-30
-20
-10
0
10
20

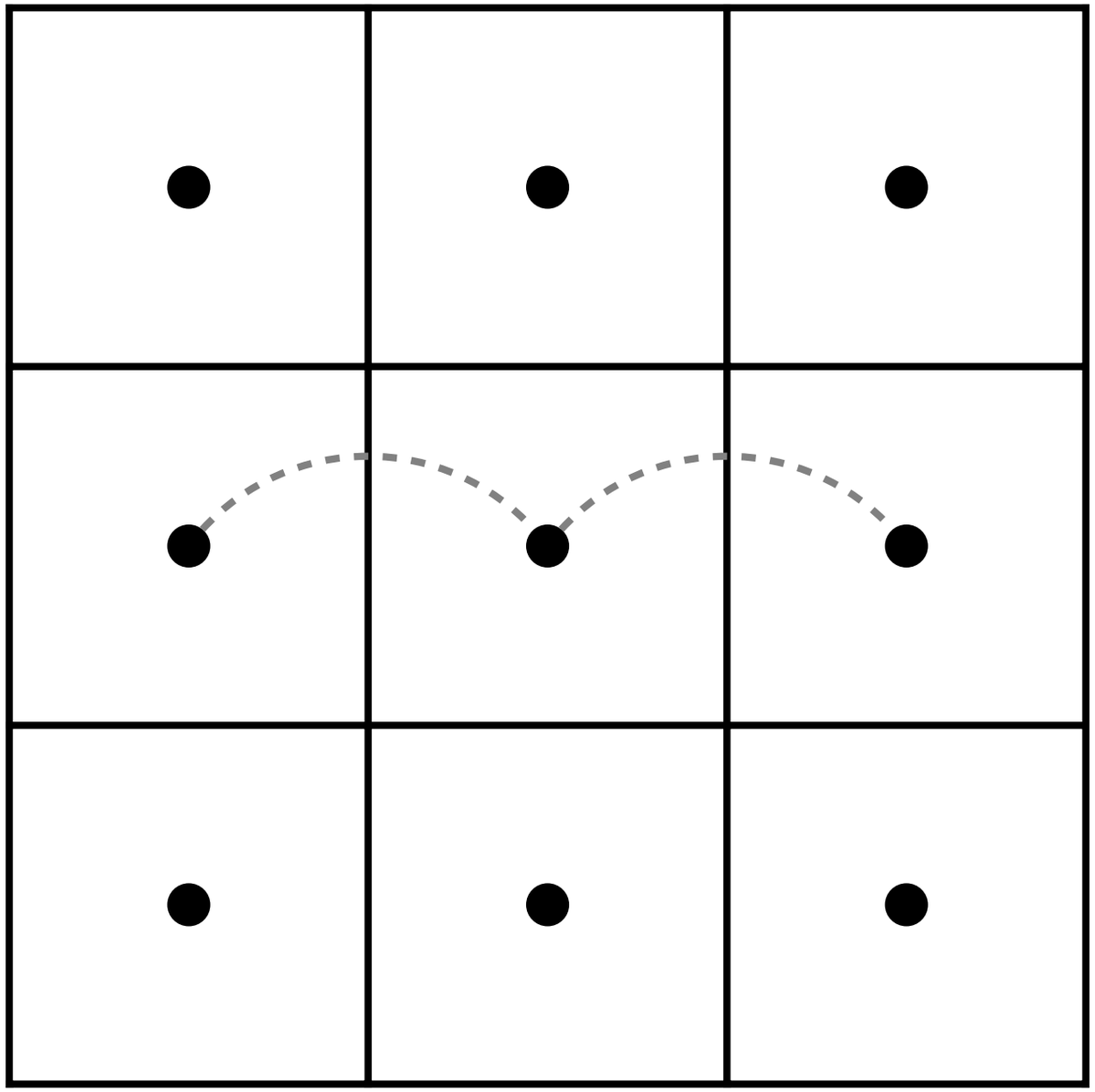

50.00
25.00
0.000
-25.00
-50.00
16
14
12
10
8
6
4
2
-35
-30
-25
-20
-15


50.00
25.00
0.000
-25.00
-50.00
10.0
9.0
8.0
7.0
6.0
-37.0
-36.0
-35.0
-34.0
-33.0
-32.0
-31.0
-30.0

Computing the x-component of the gradient on a uniform grid

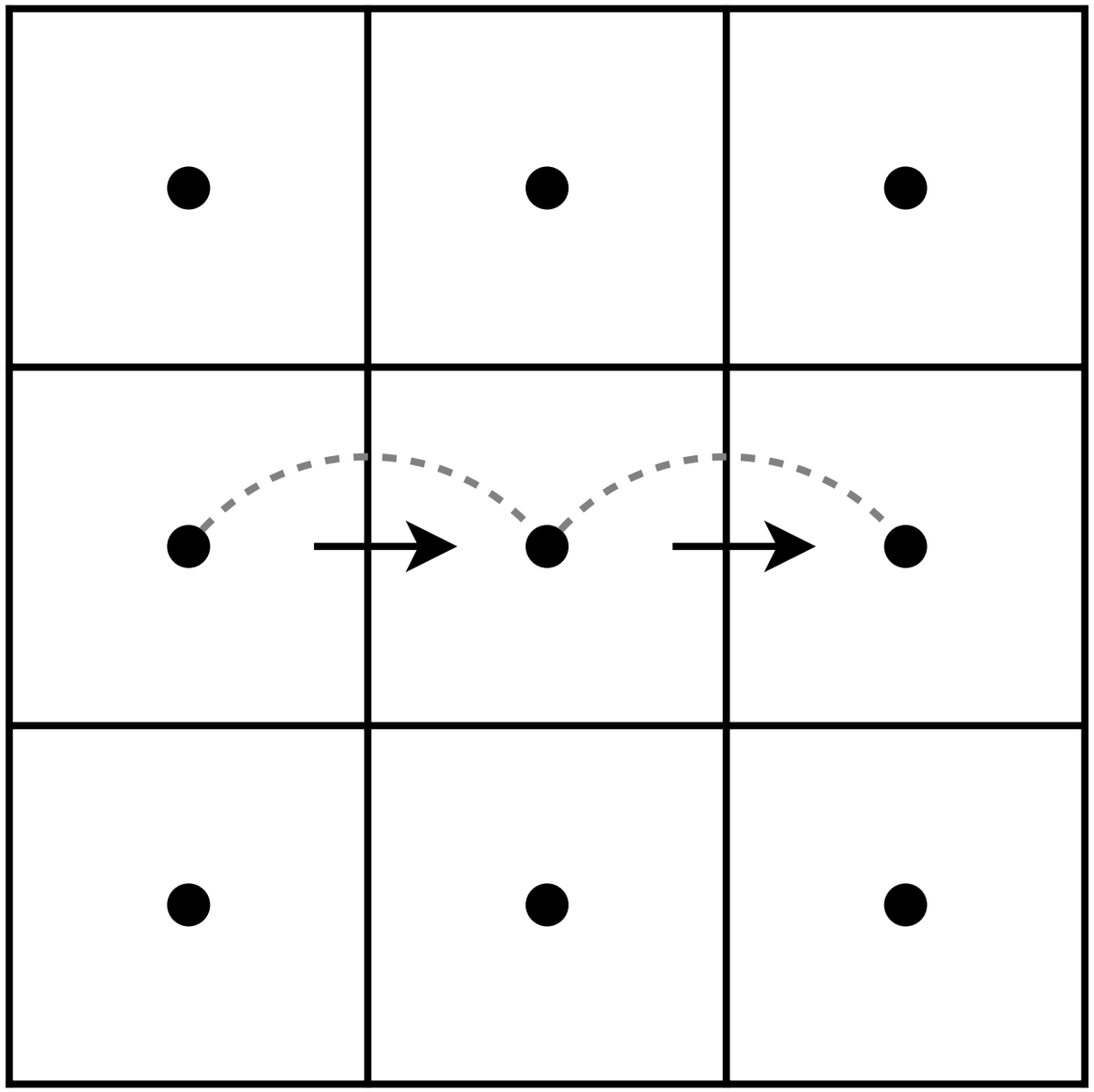


Computing the x-component of the gradient on a uniform grid



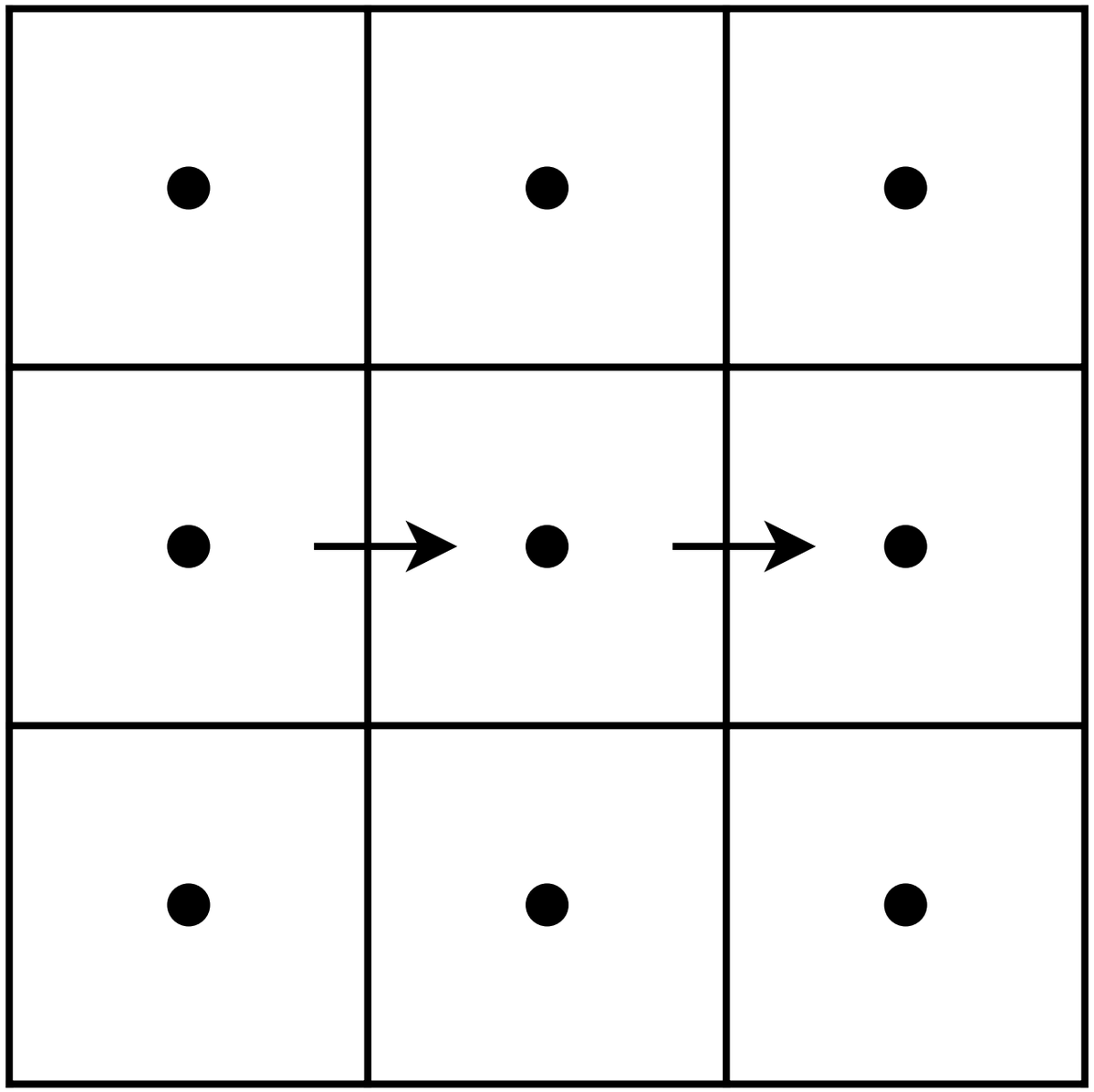
Difference neighboring cell-centered values.

Computing the x-component of the gradient on a uniform grid



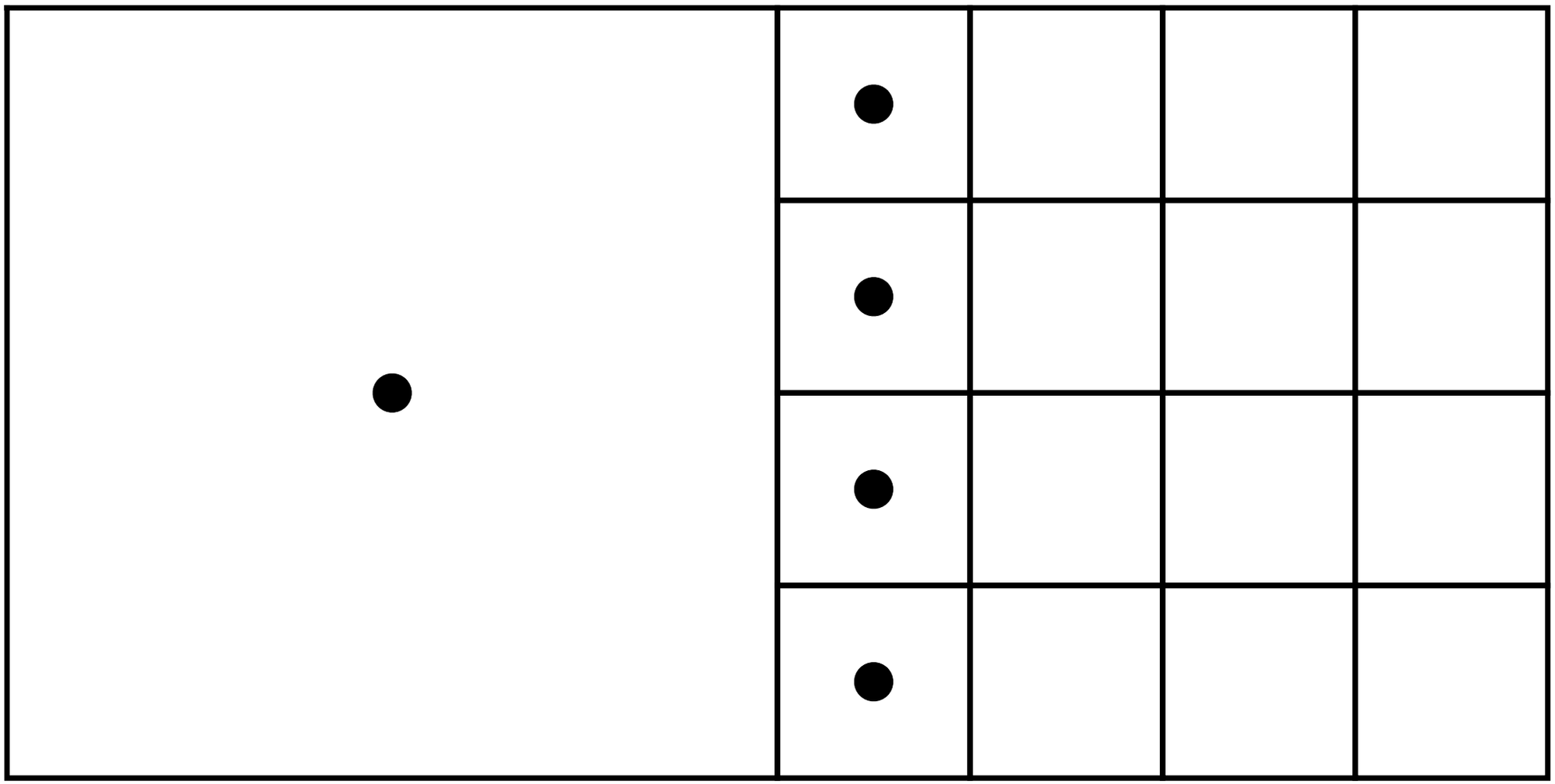
Difference neighboring cell-centered values.

Computing the x-component of the gradient on a uniform grid

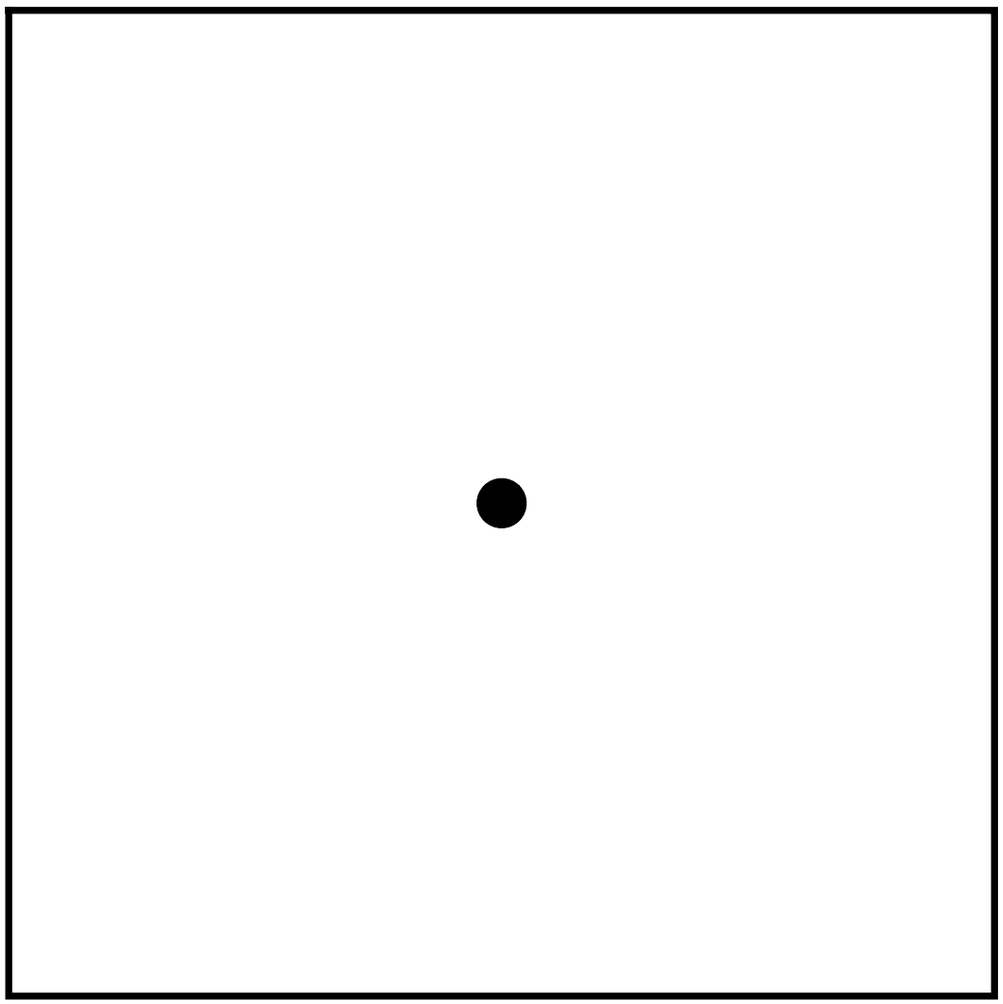


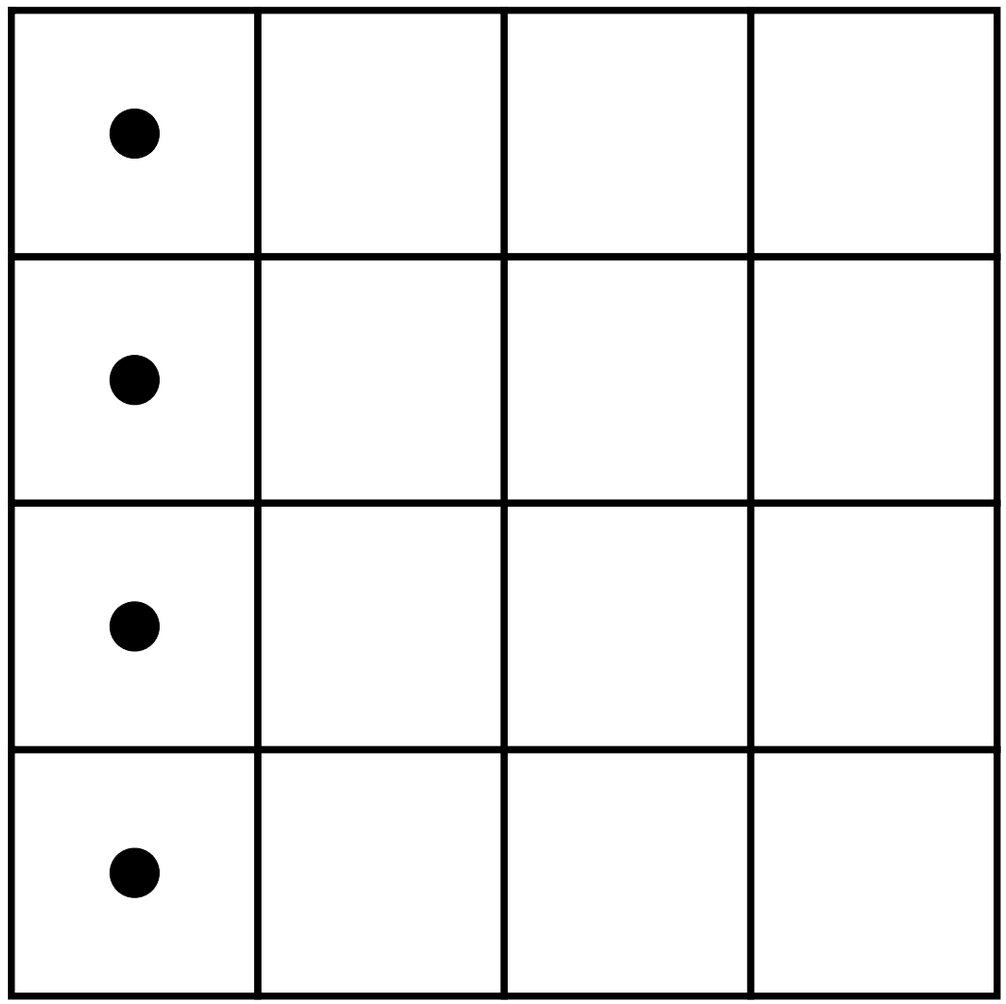
Difference neighboring cell-centered values.

Computing the x-component of the gradient on a locally refined grid

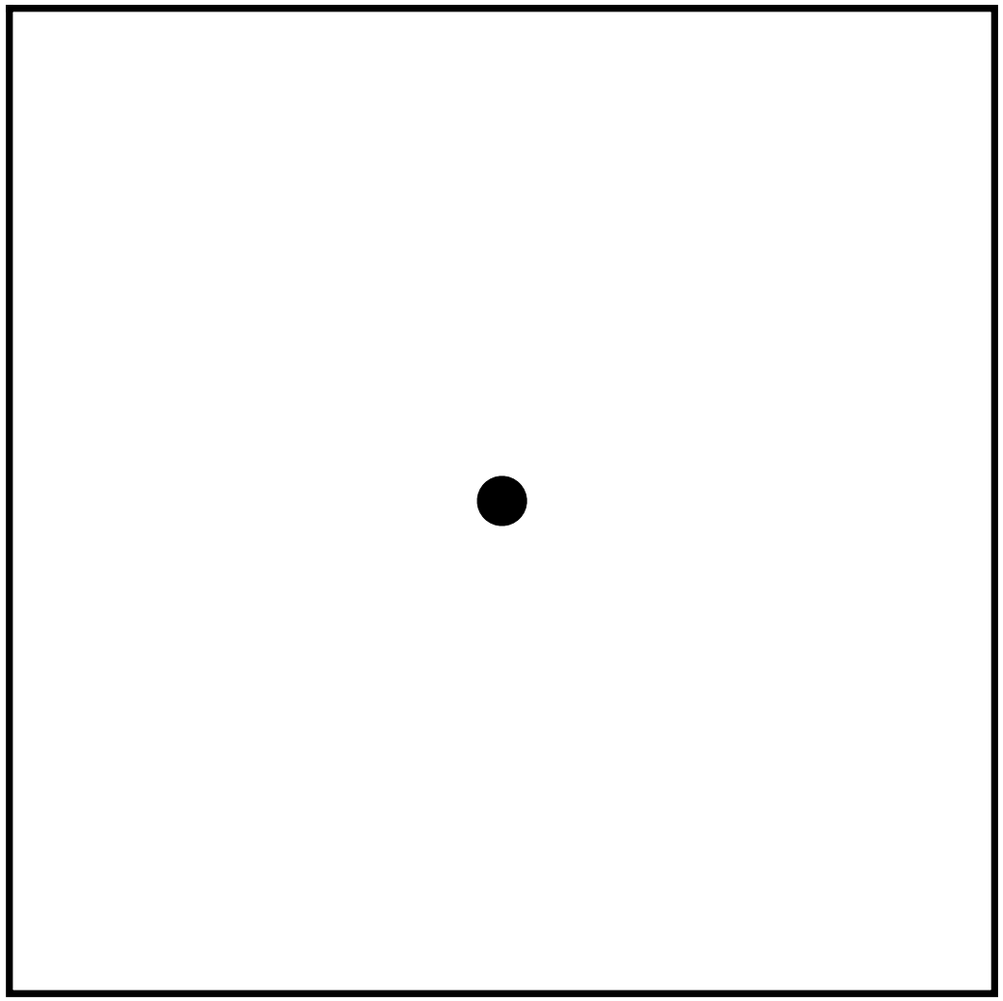


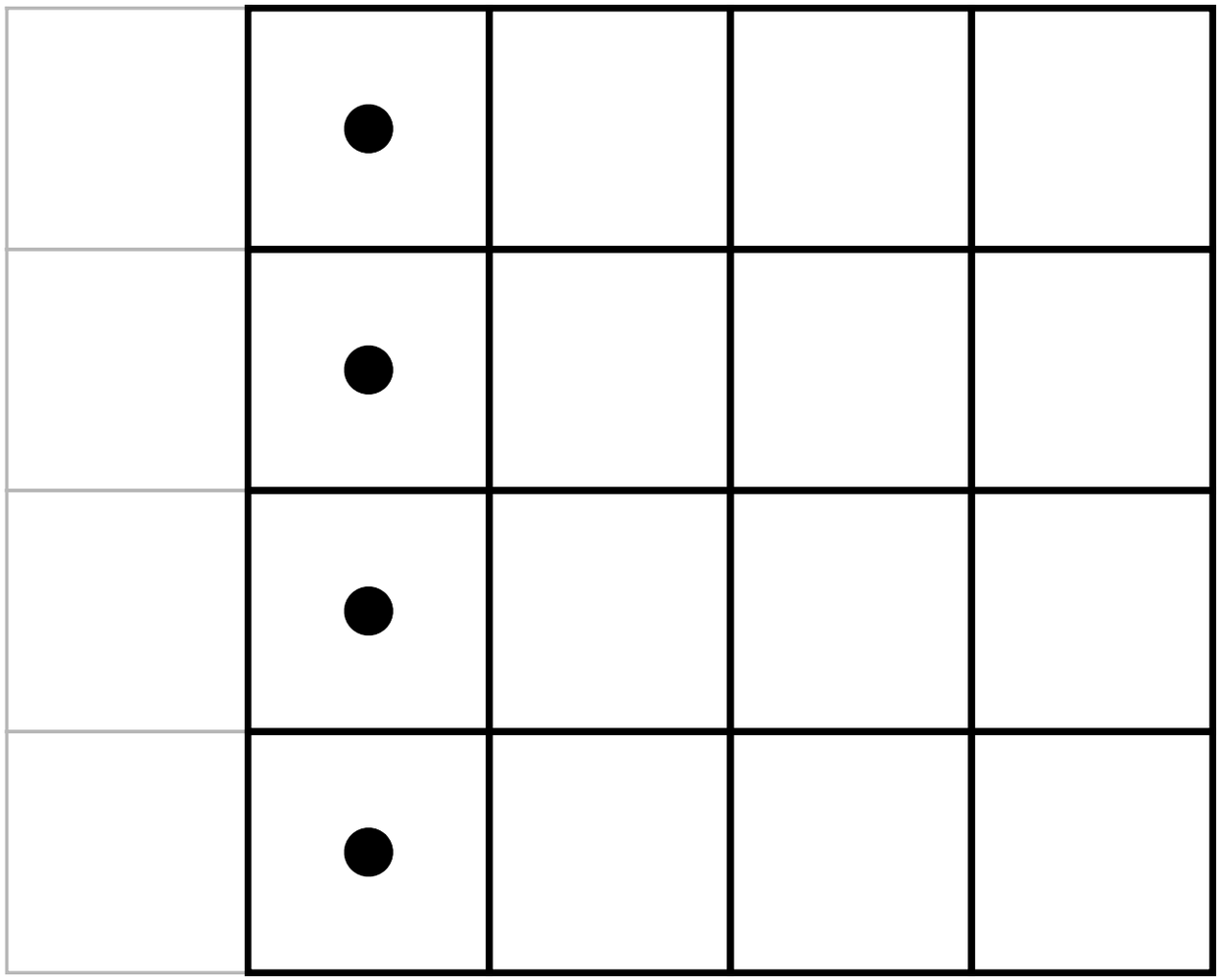
Computing the x-component of the gradient on a locally refined grid



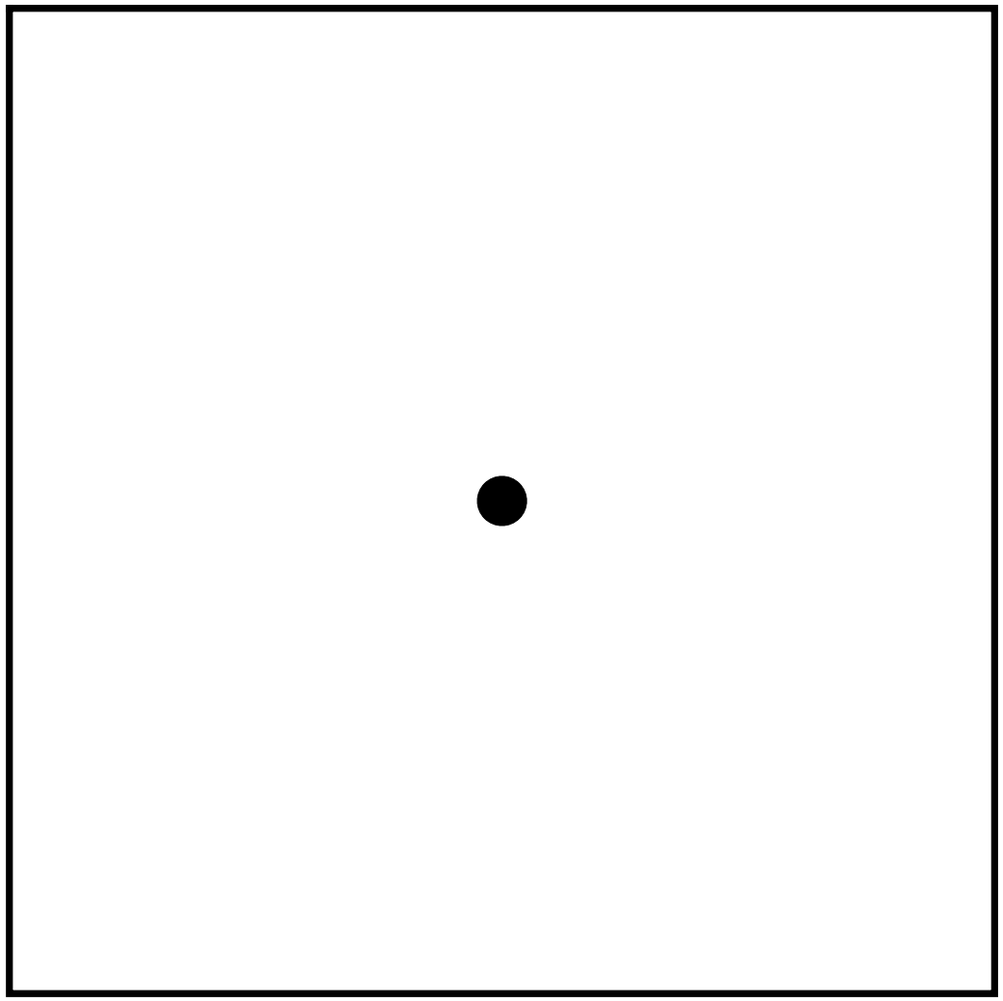


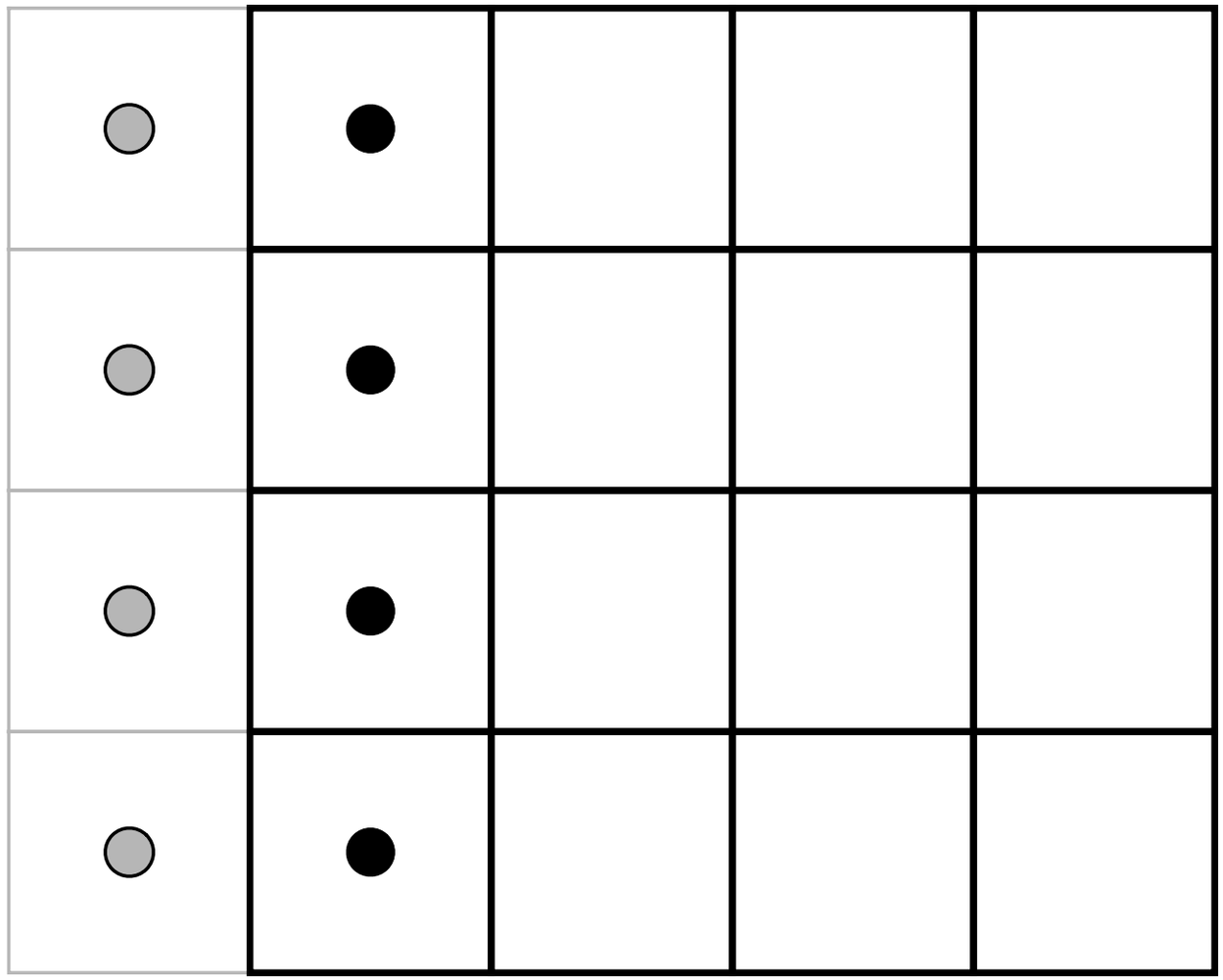
Computing the x-component of the gradient on a locally refined grid



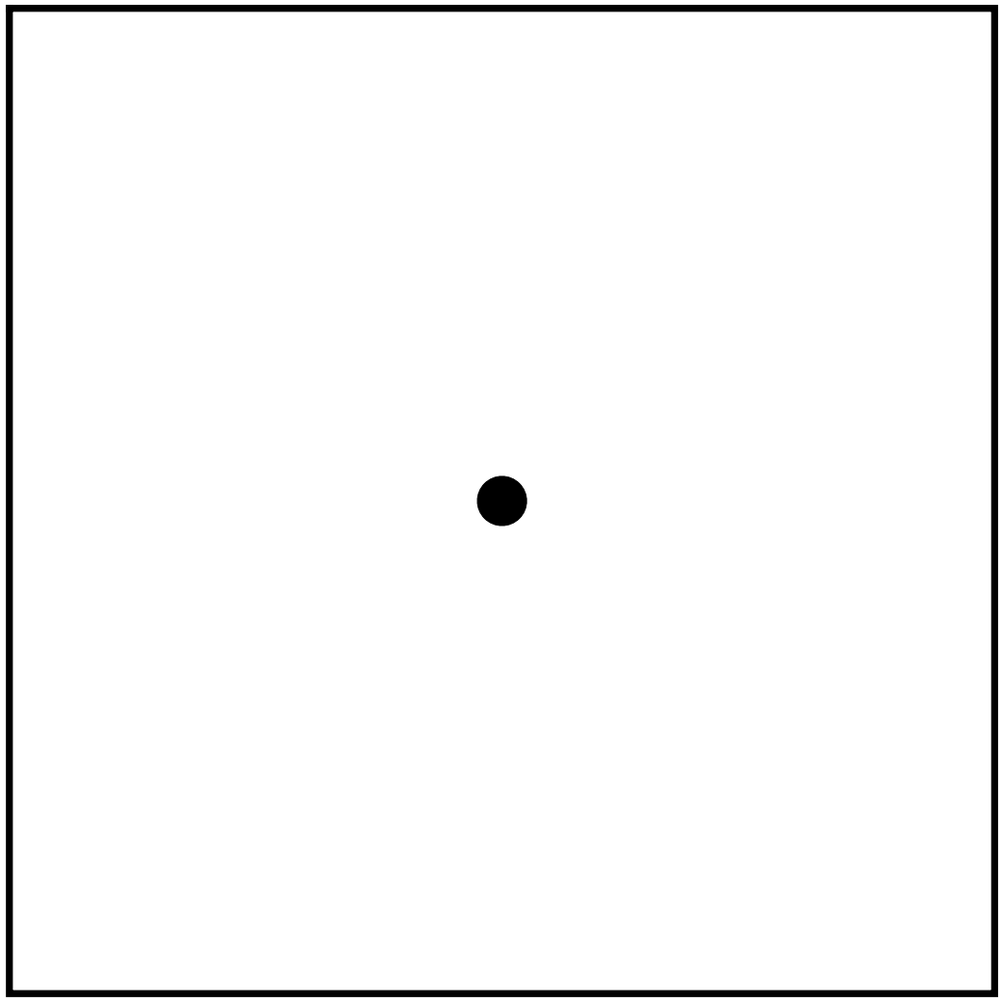


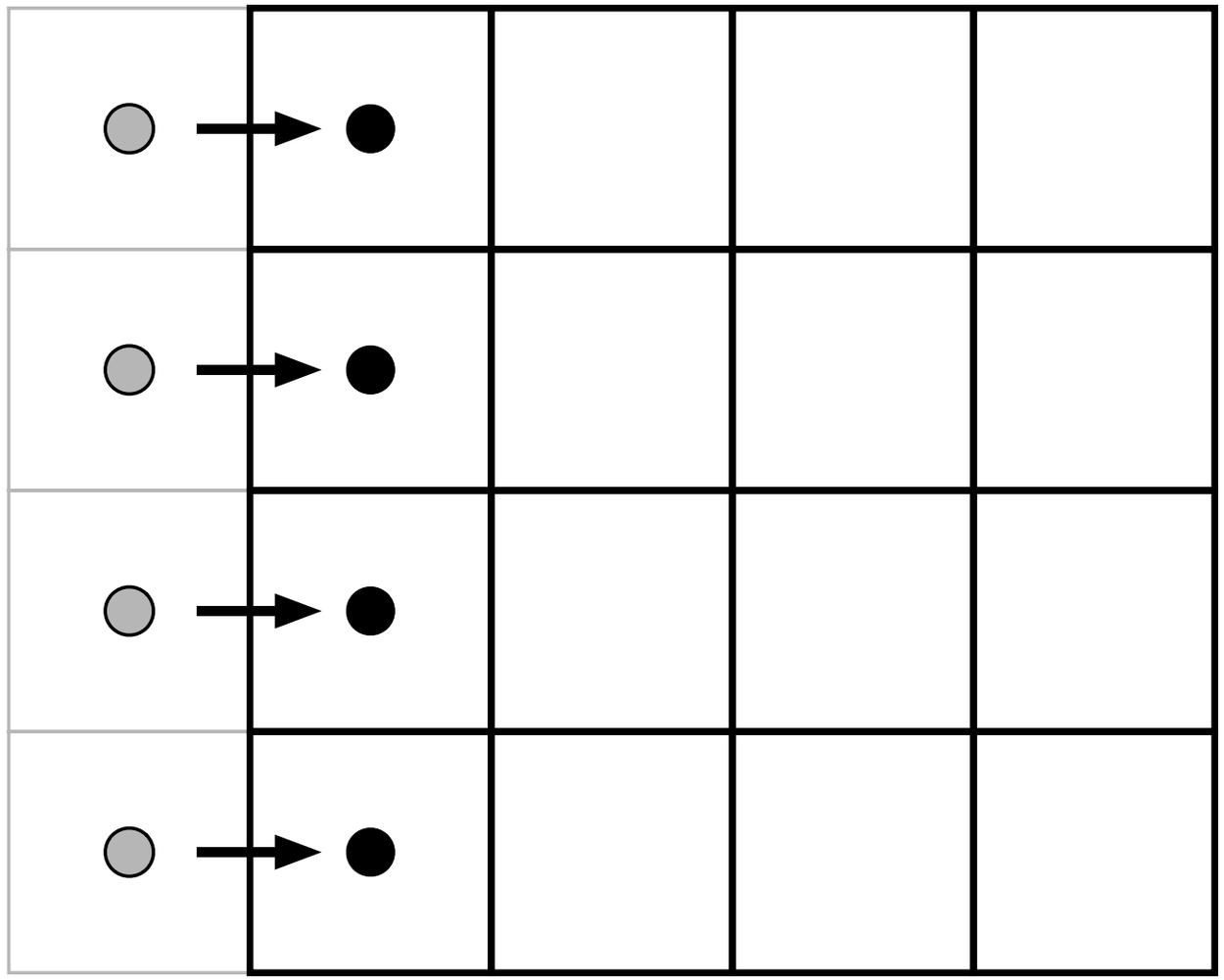
Computing the x-component of the gradient on a locally refined grid



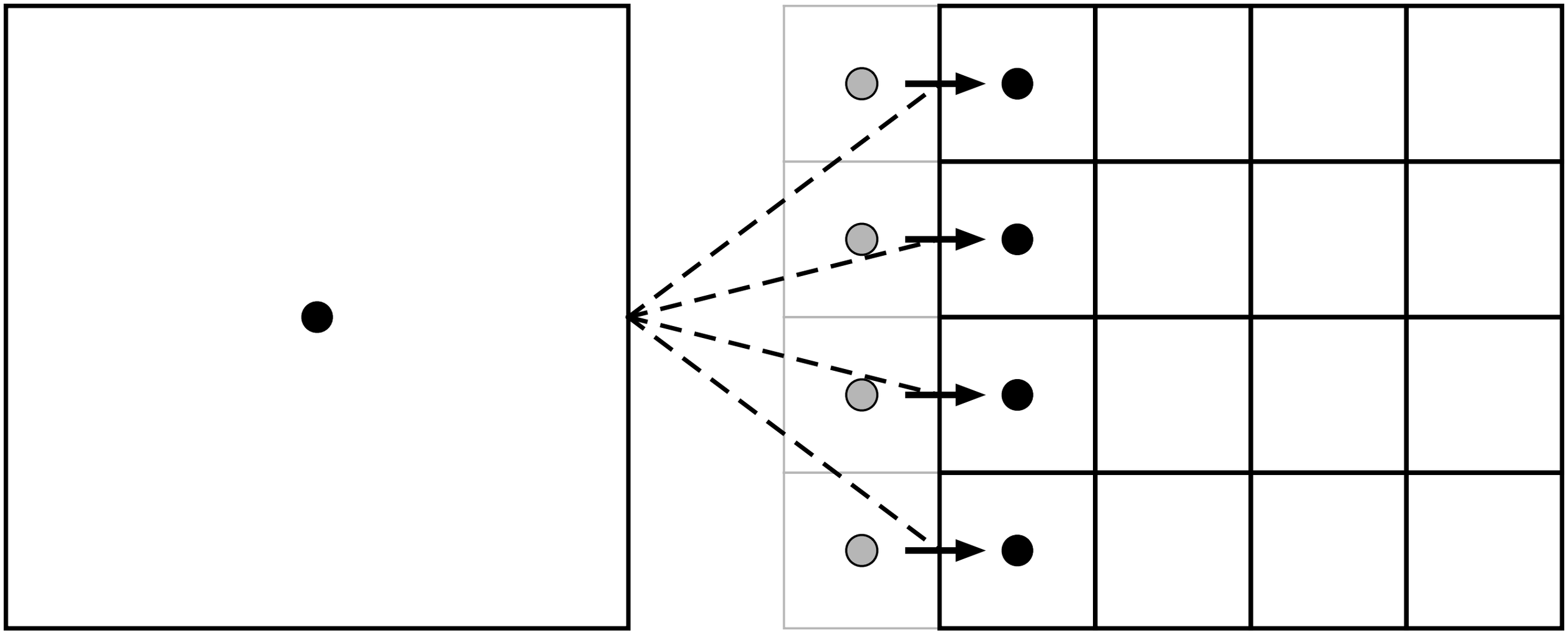


Computing the x-component of the gradient on a locally refined grid

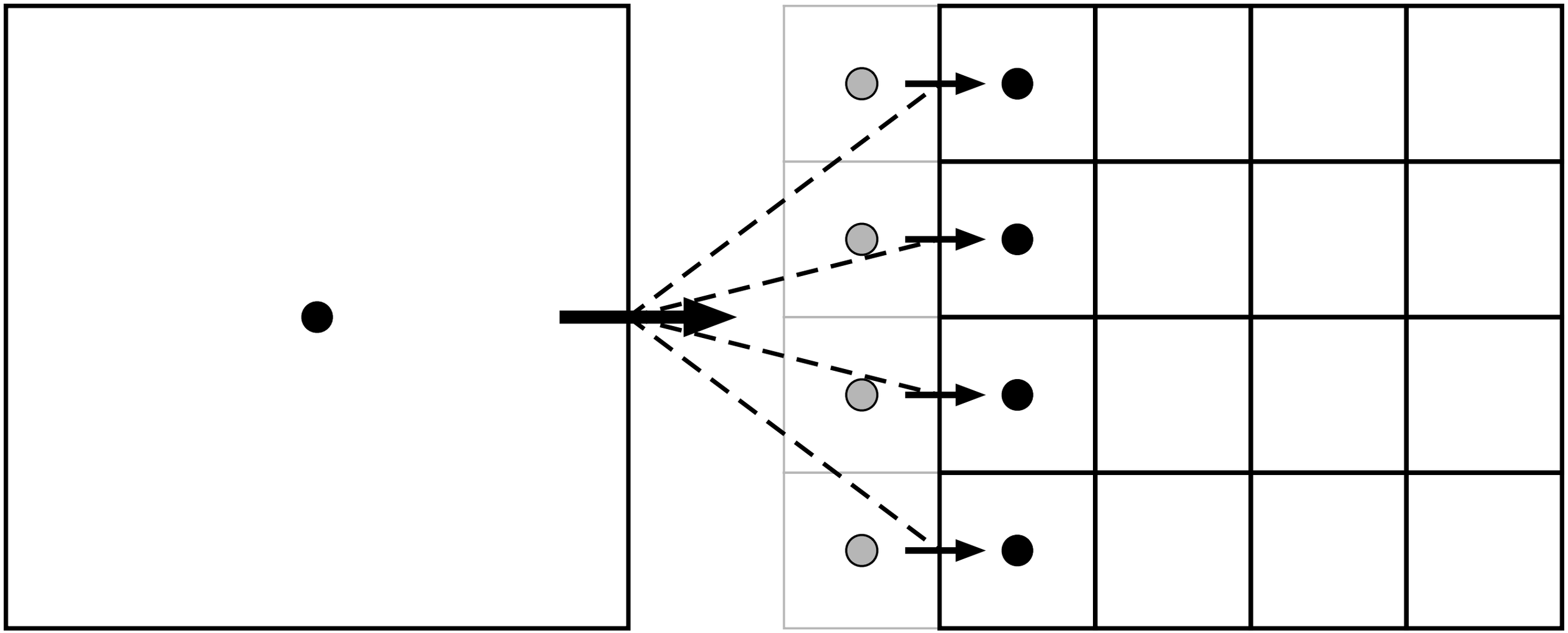




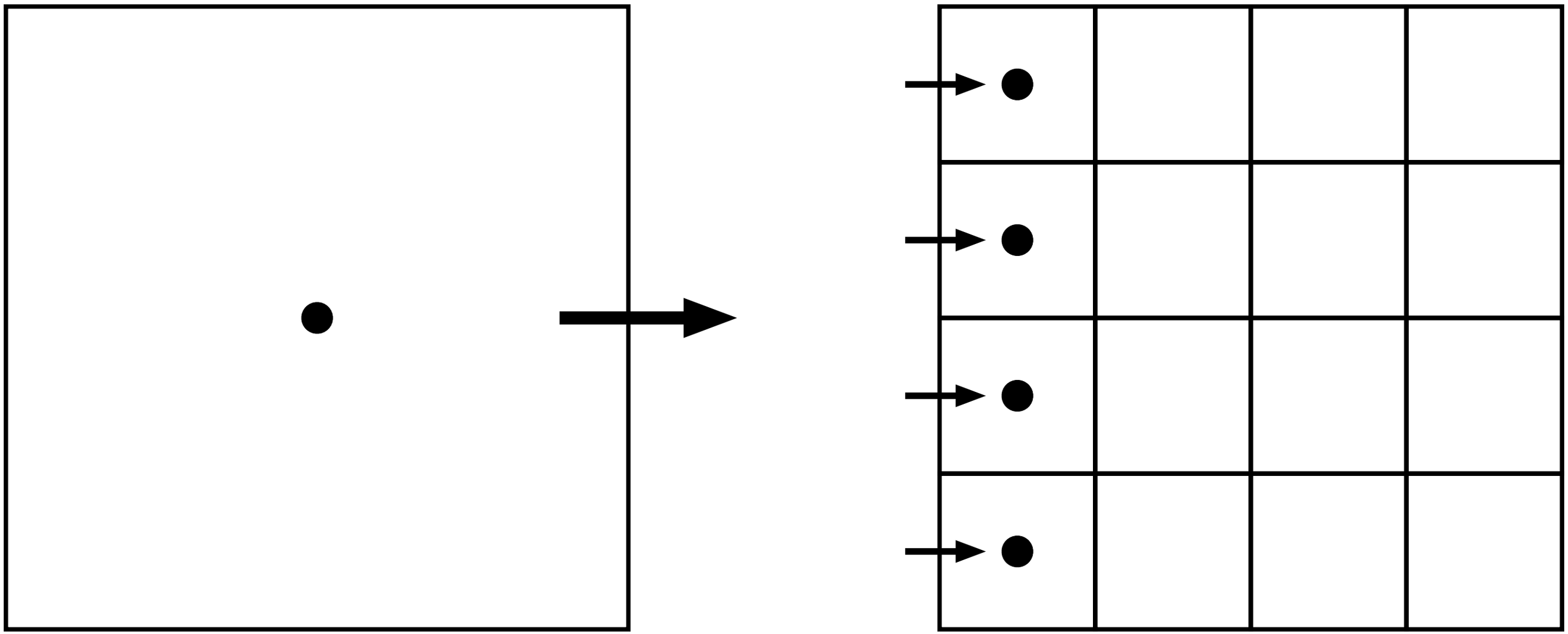
Computing the x-component of the gradient on a locally refined grid

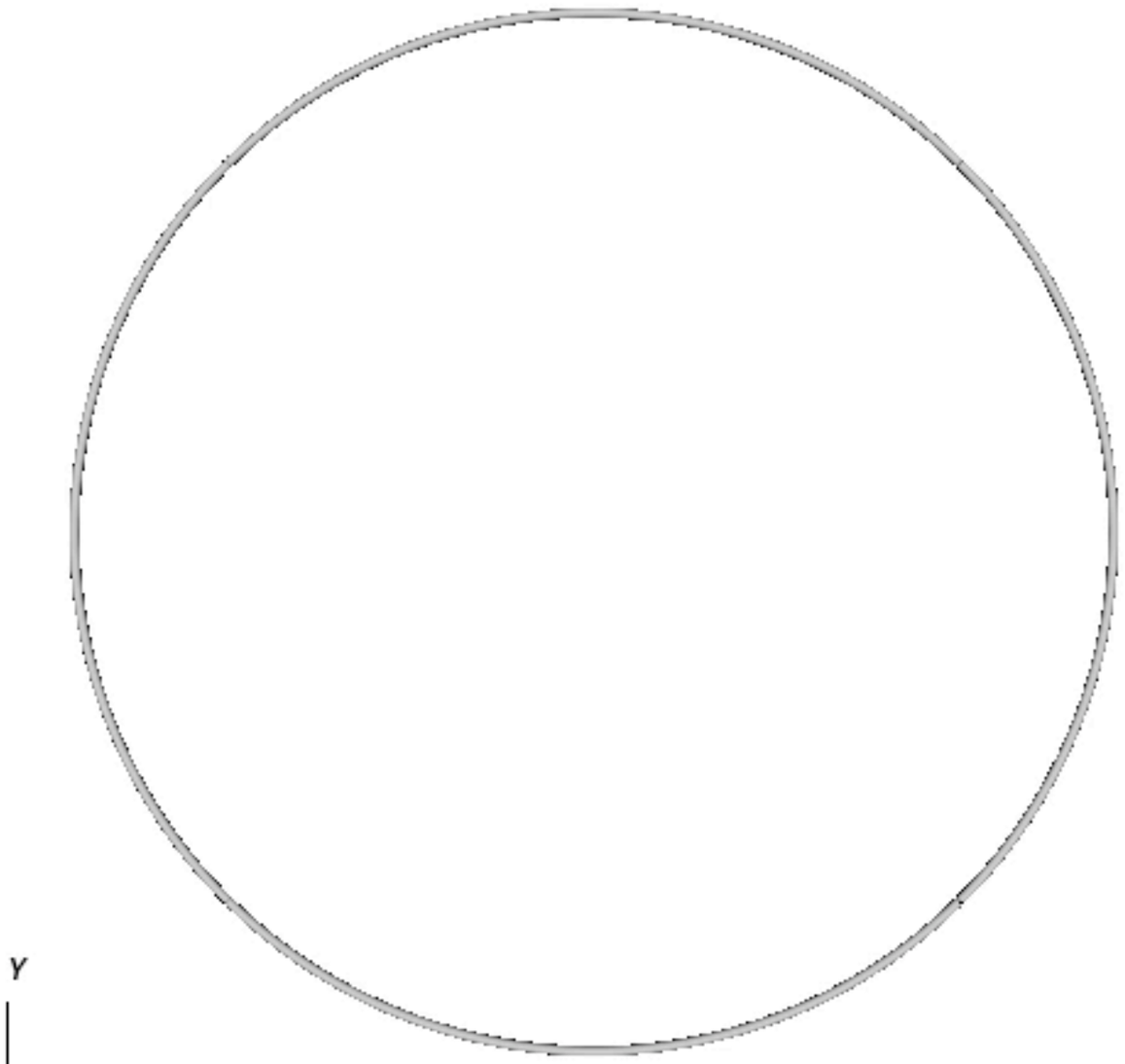


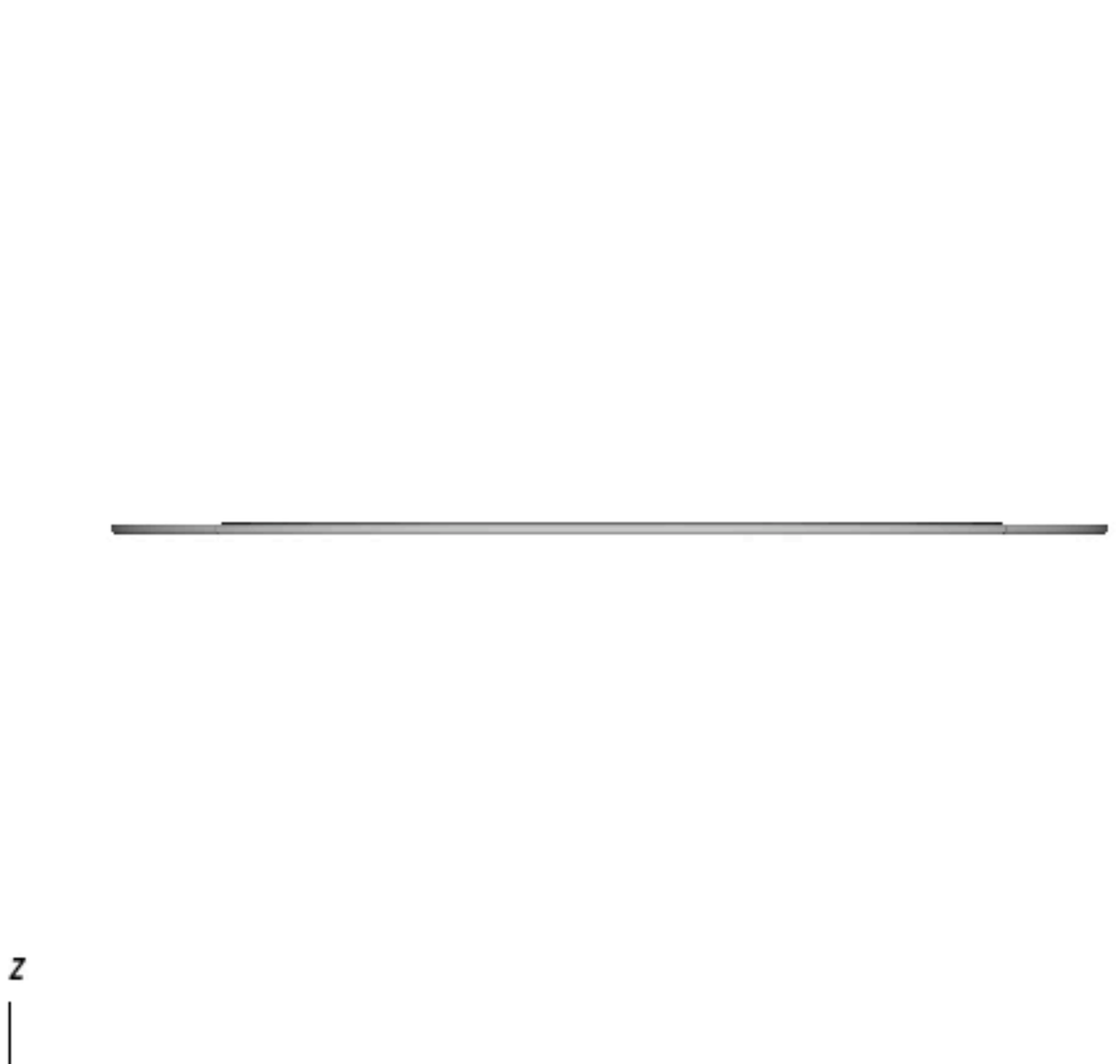
Computing the x-component of the gradient on a locally refined grid

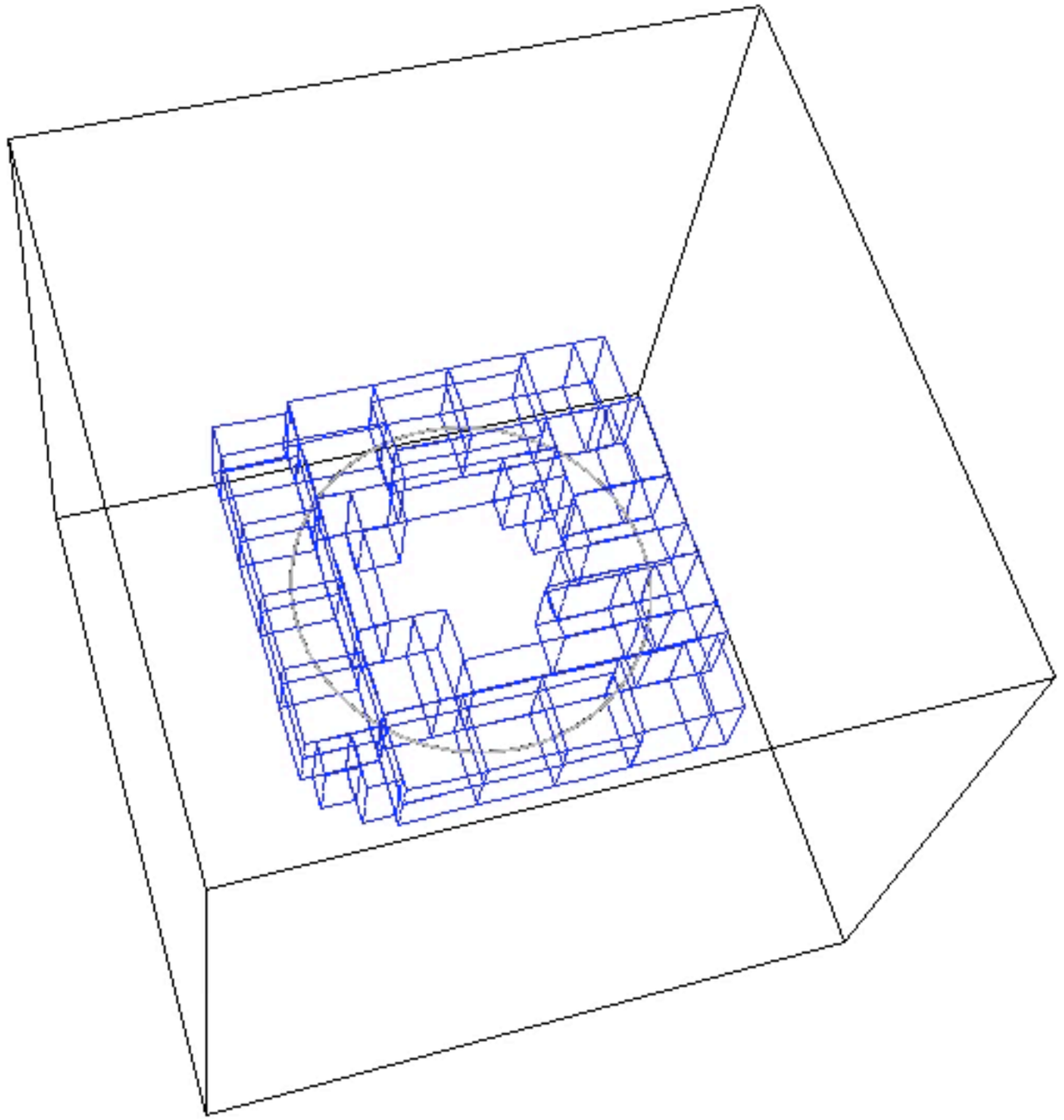


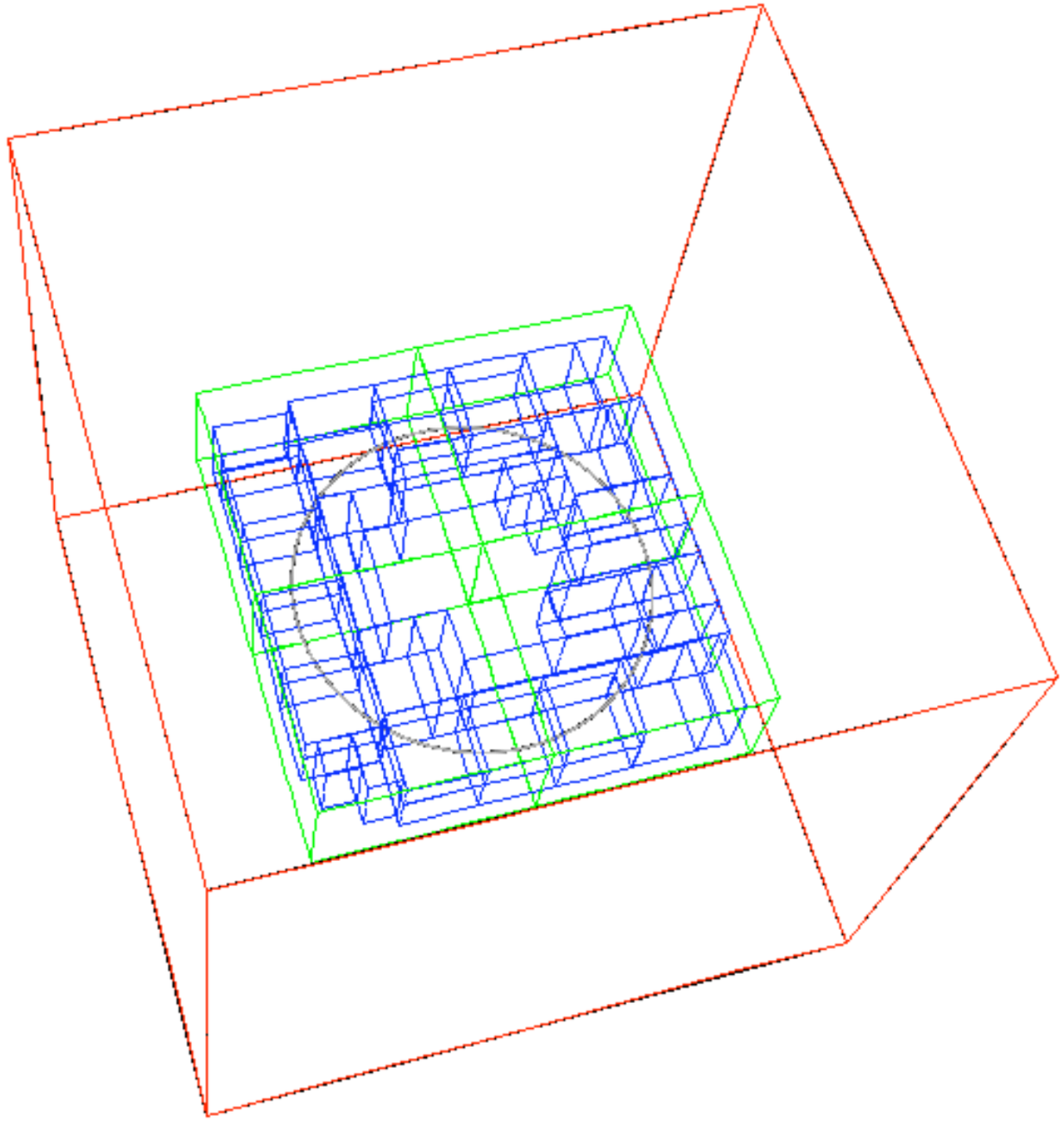
Computing the x-component of the gradient on a locally refined grid

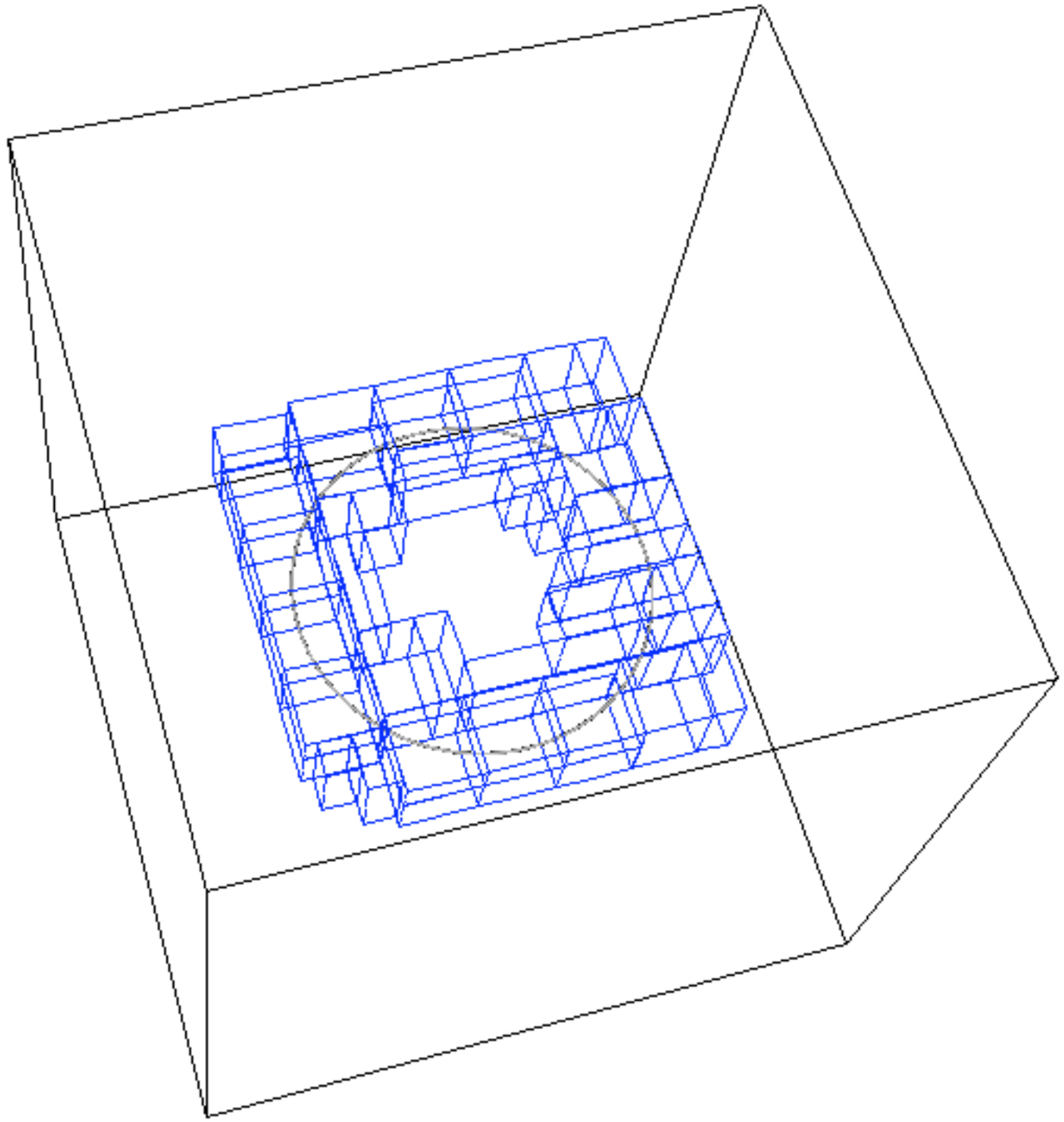


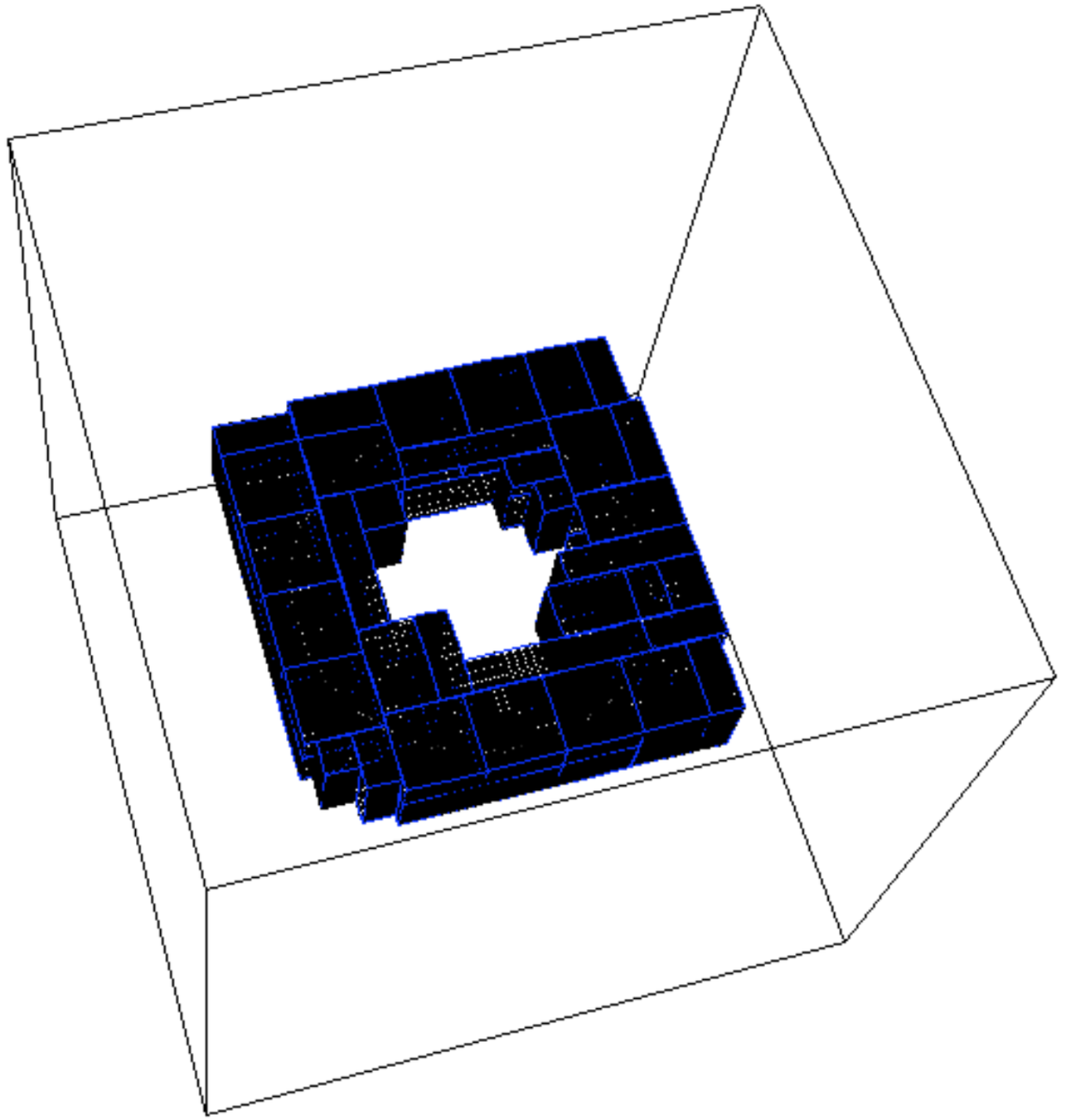

Y
Z
X

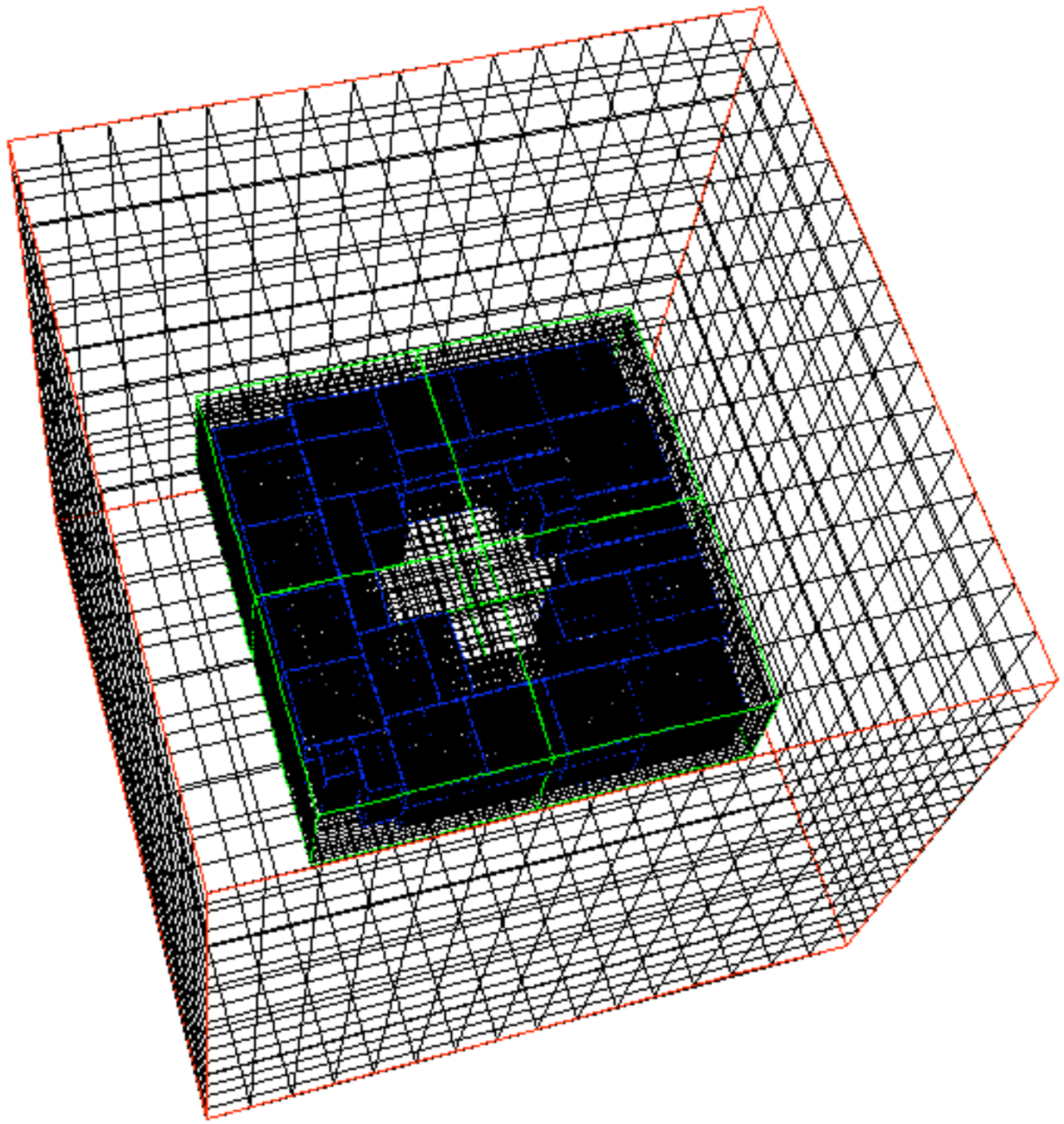

Z
Y
X

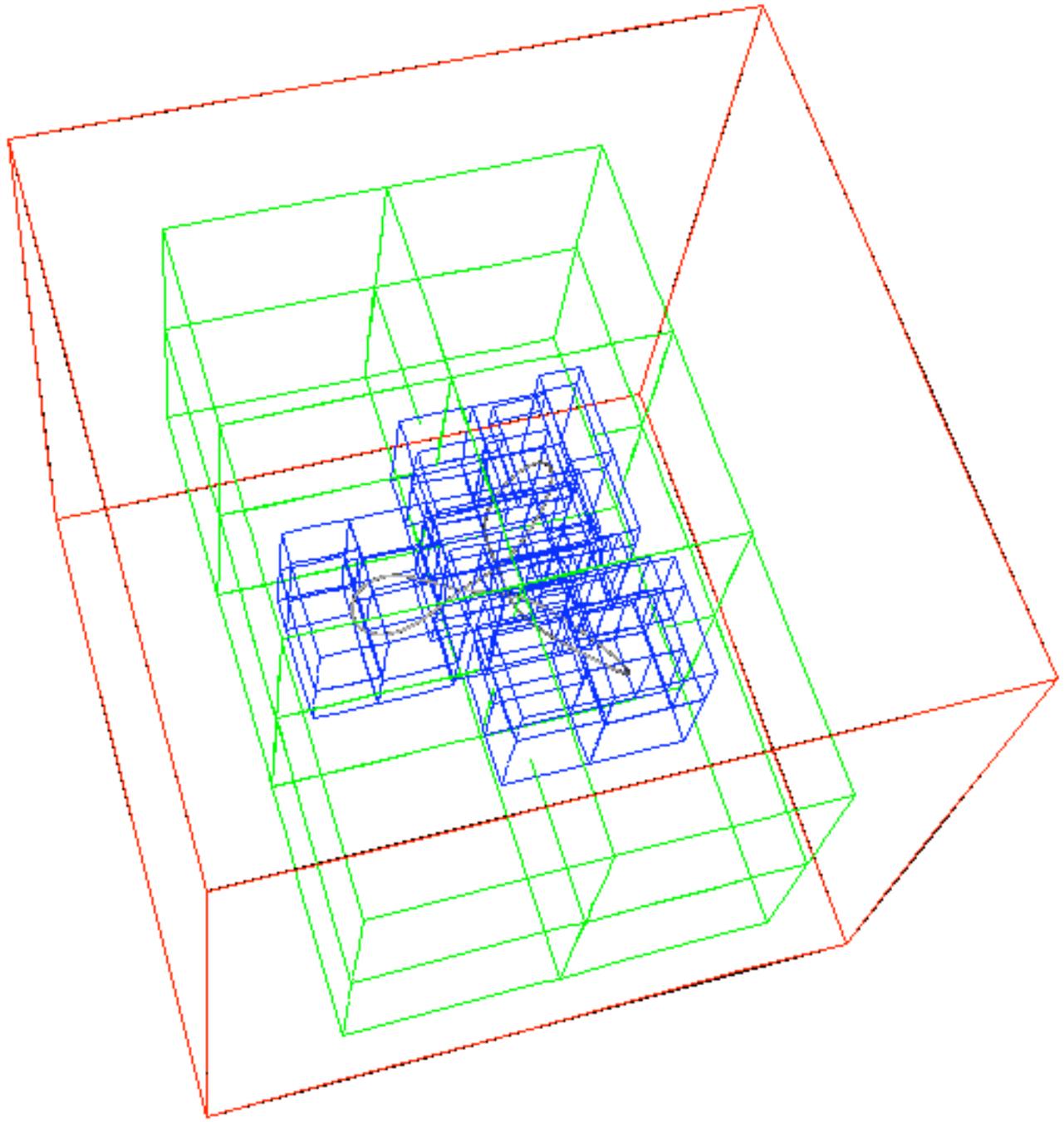


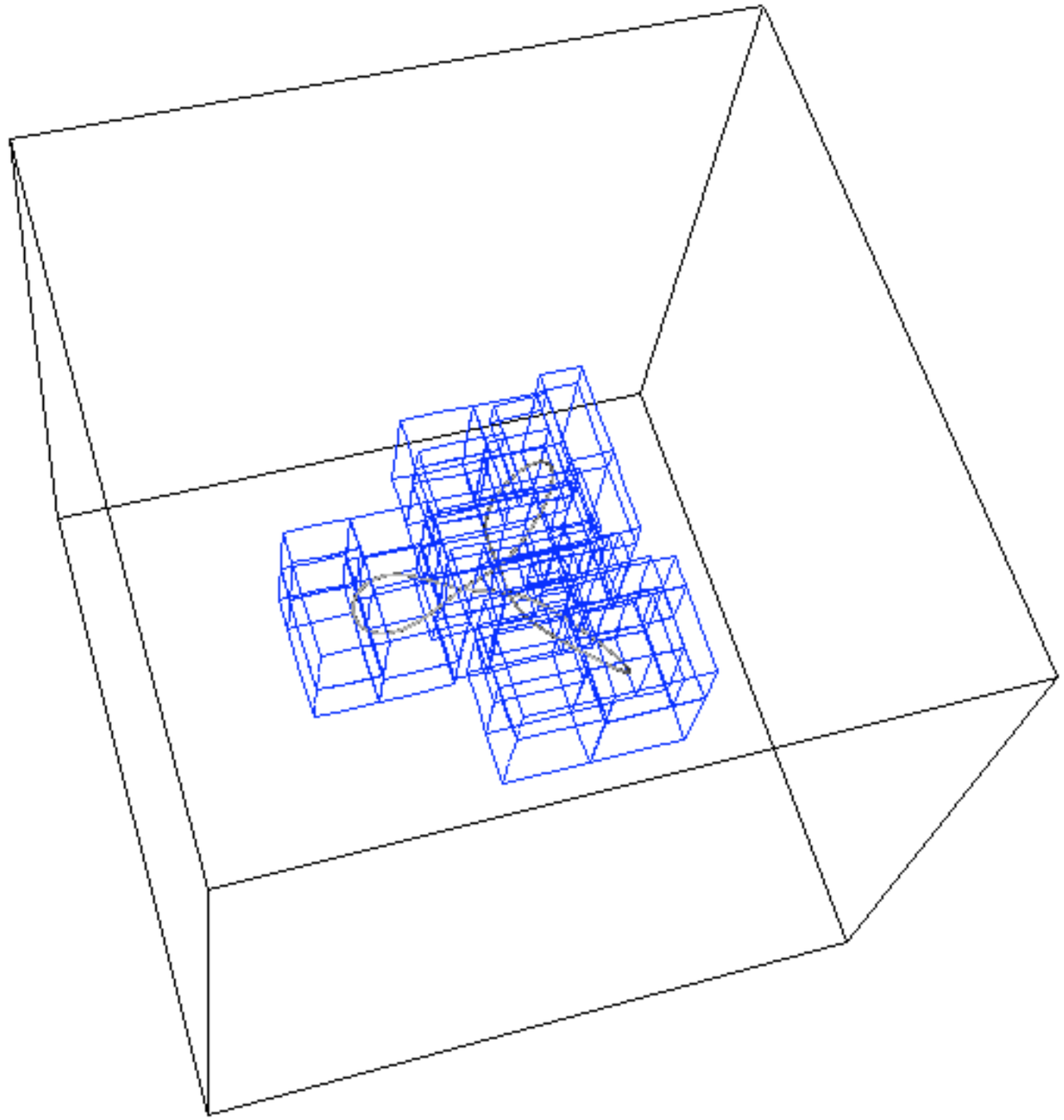


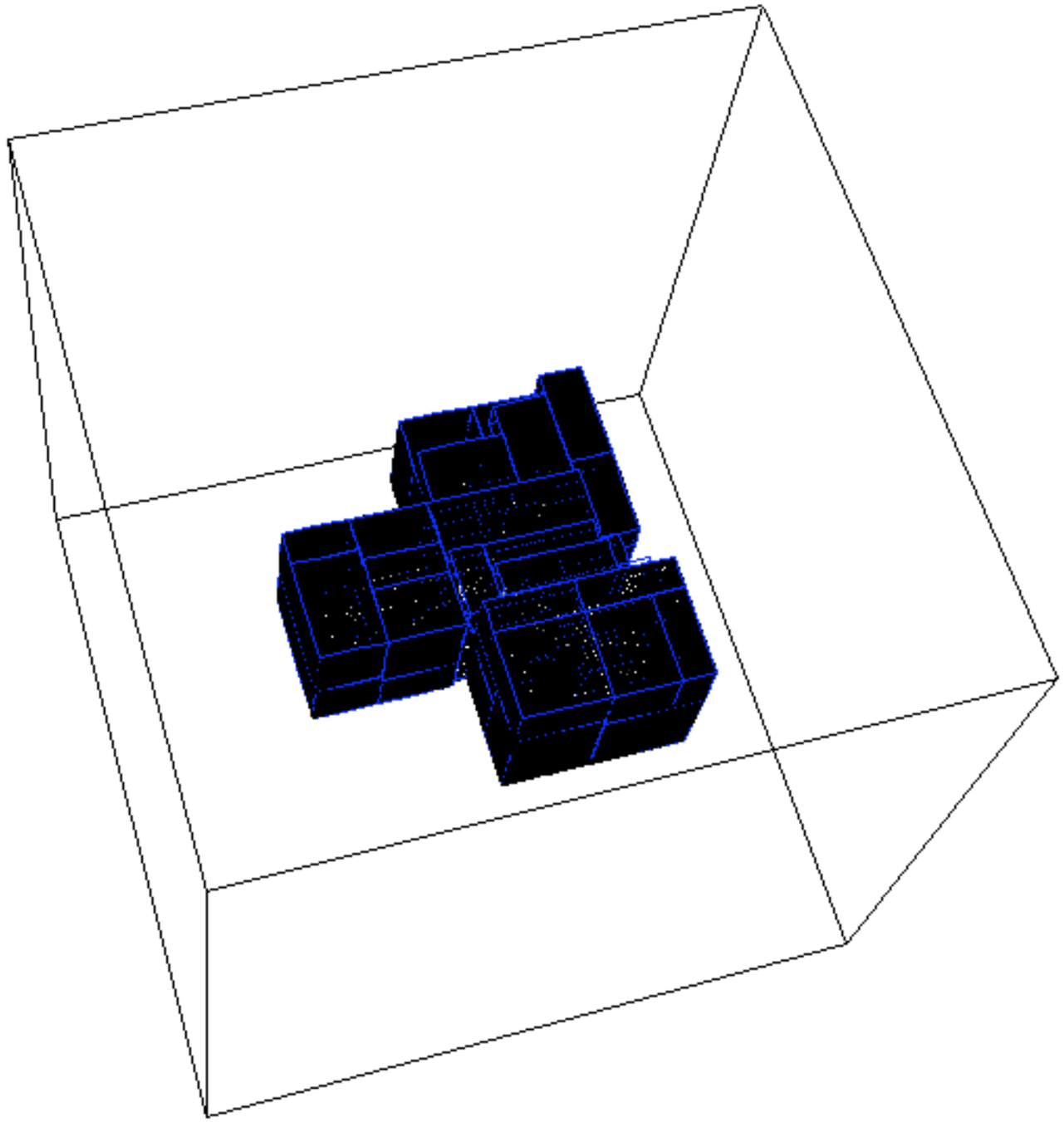


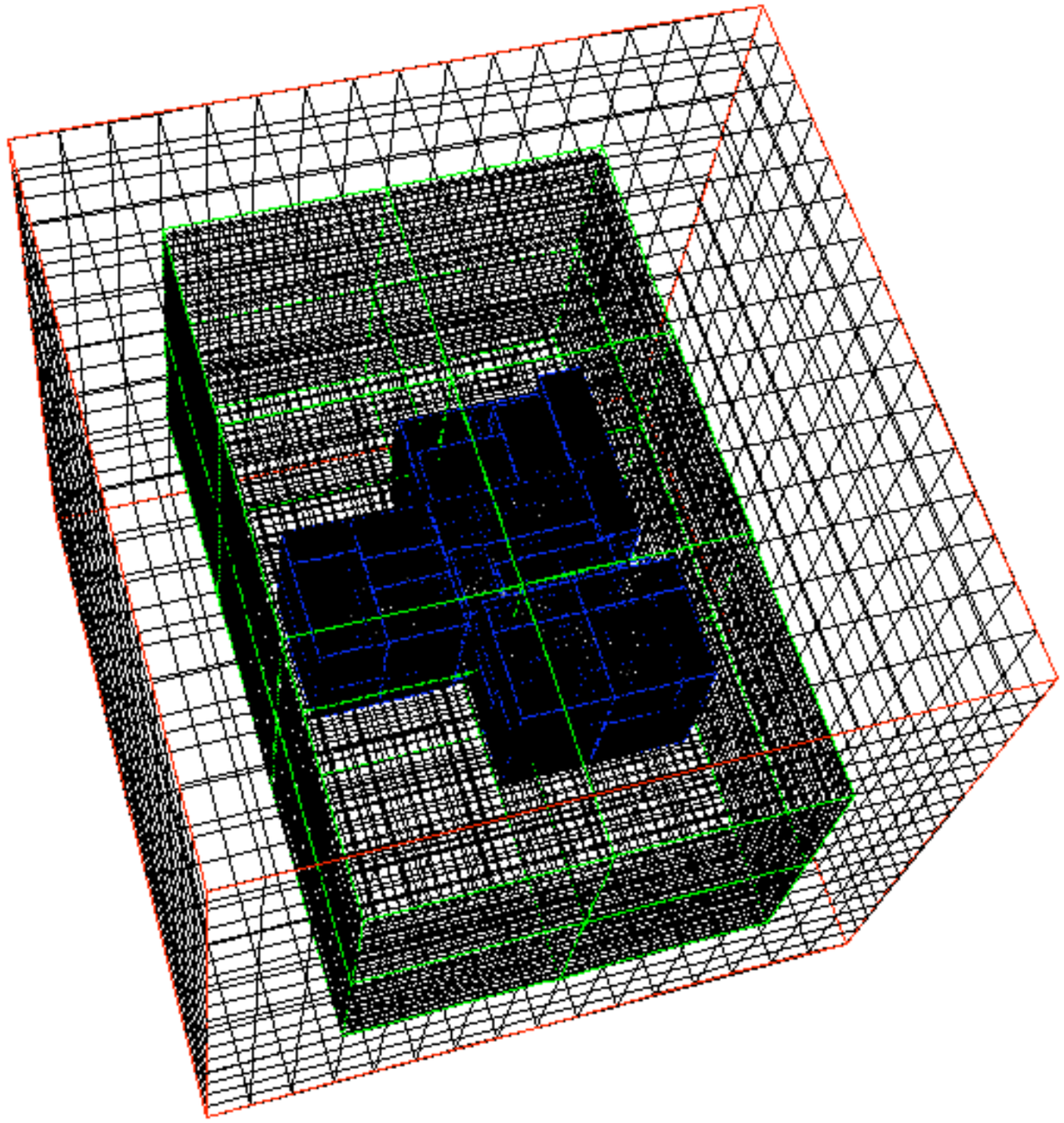




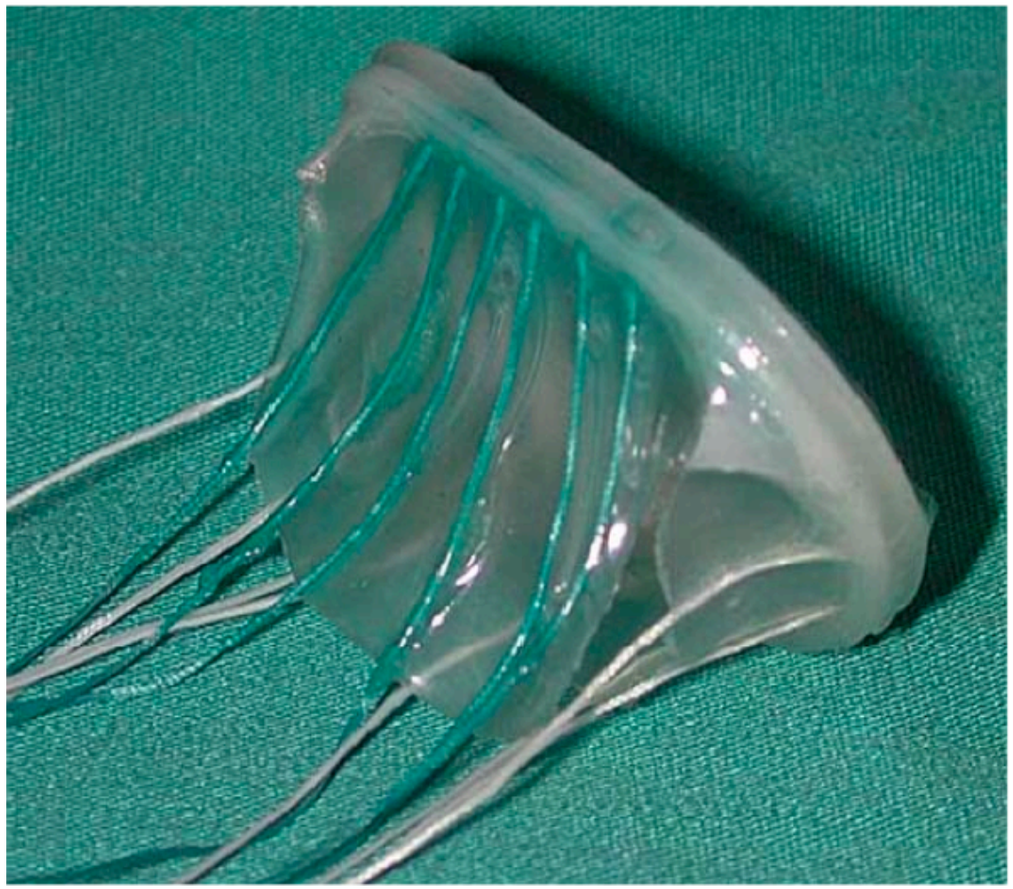


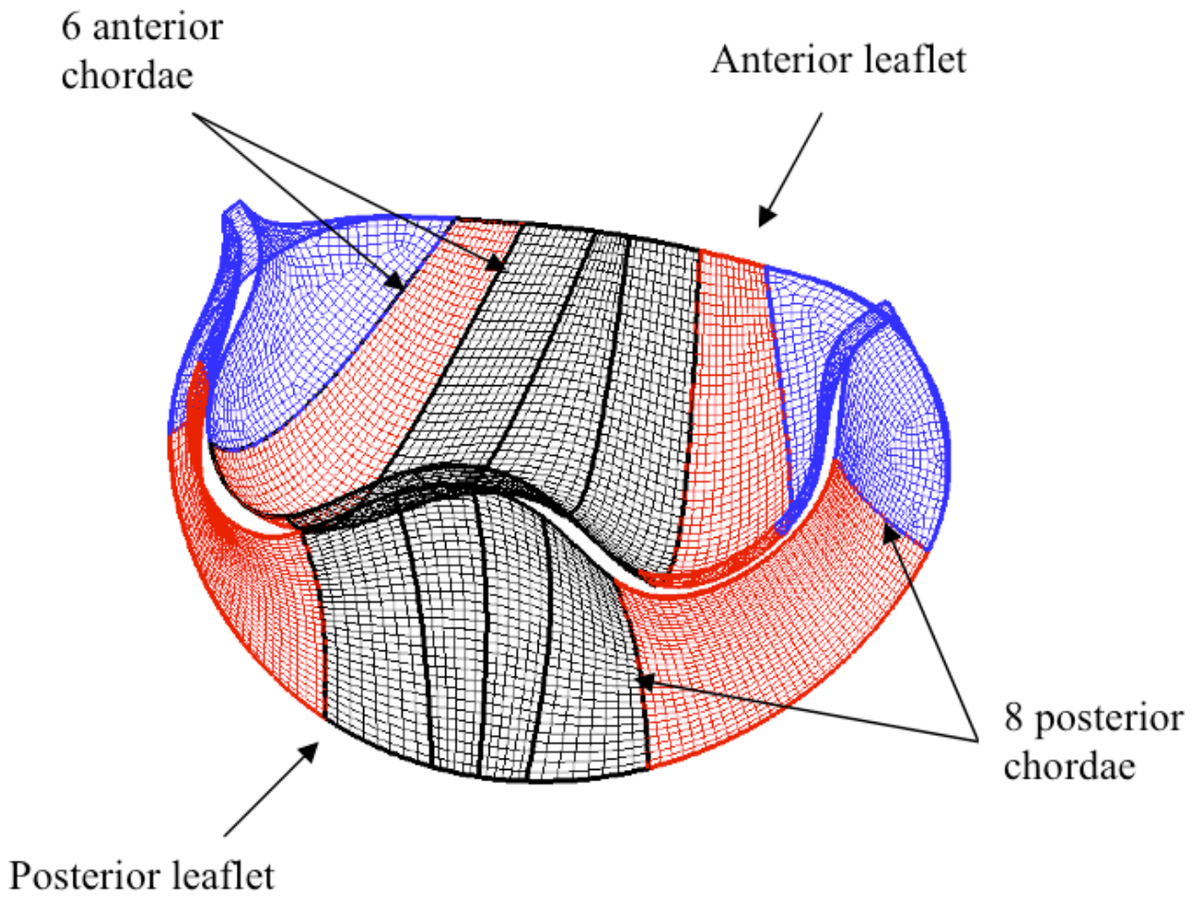




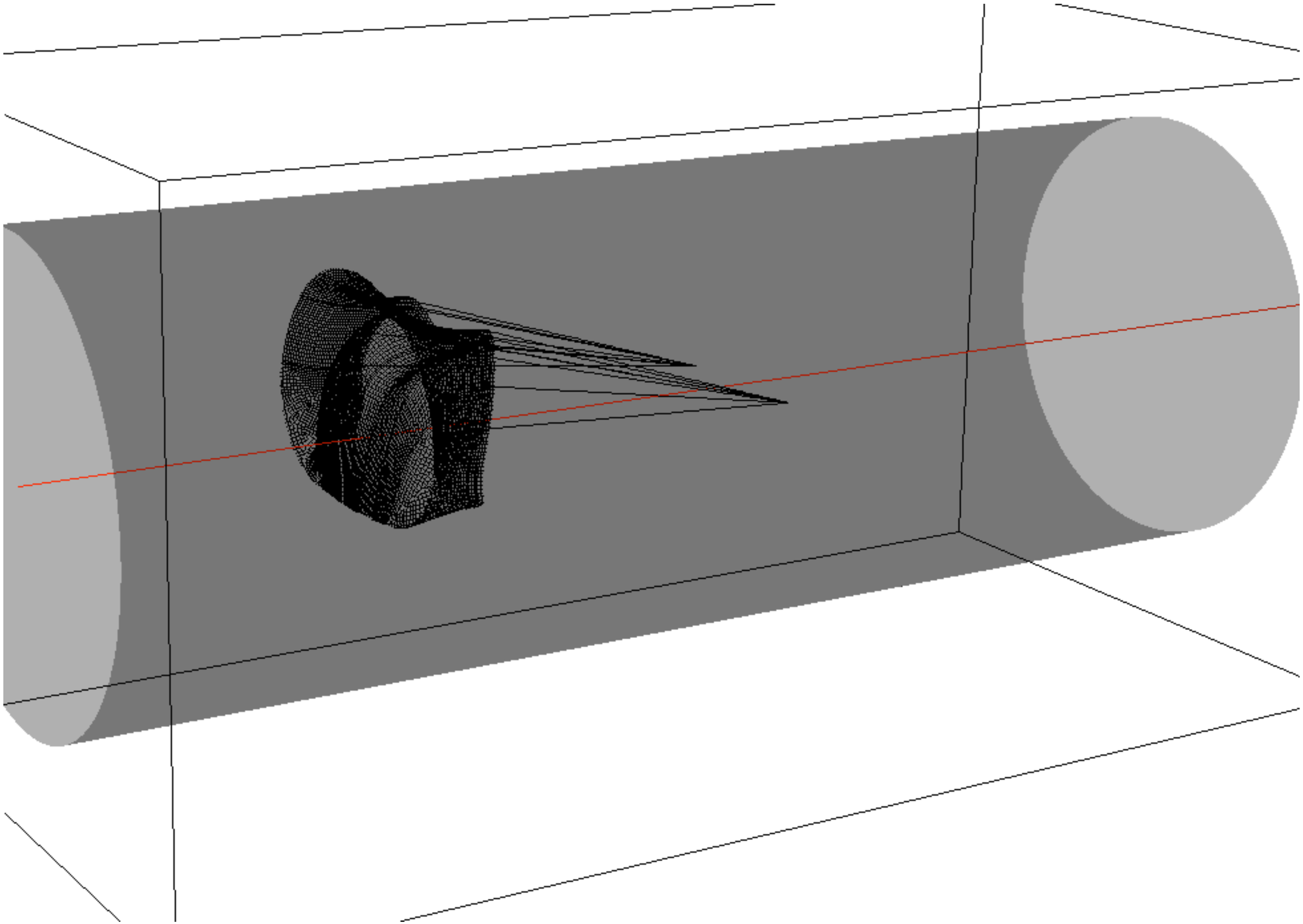


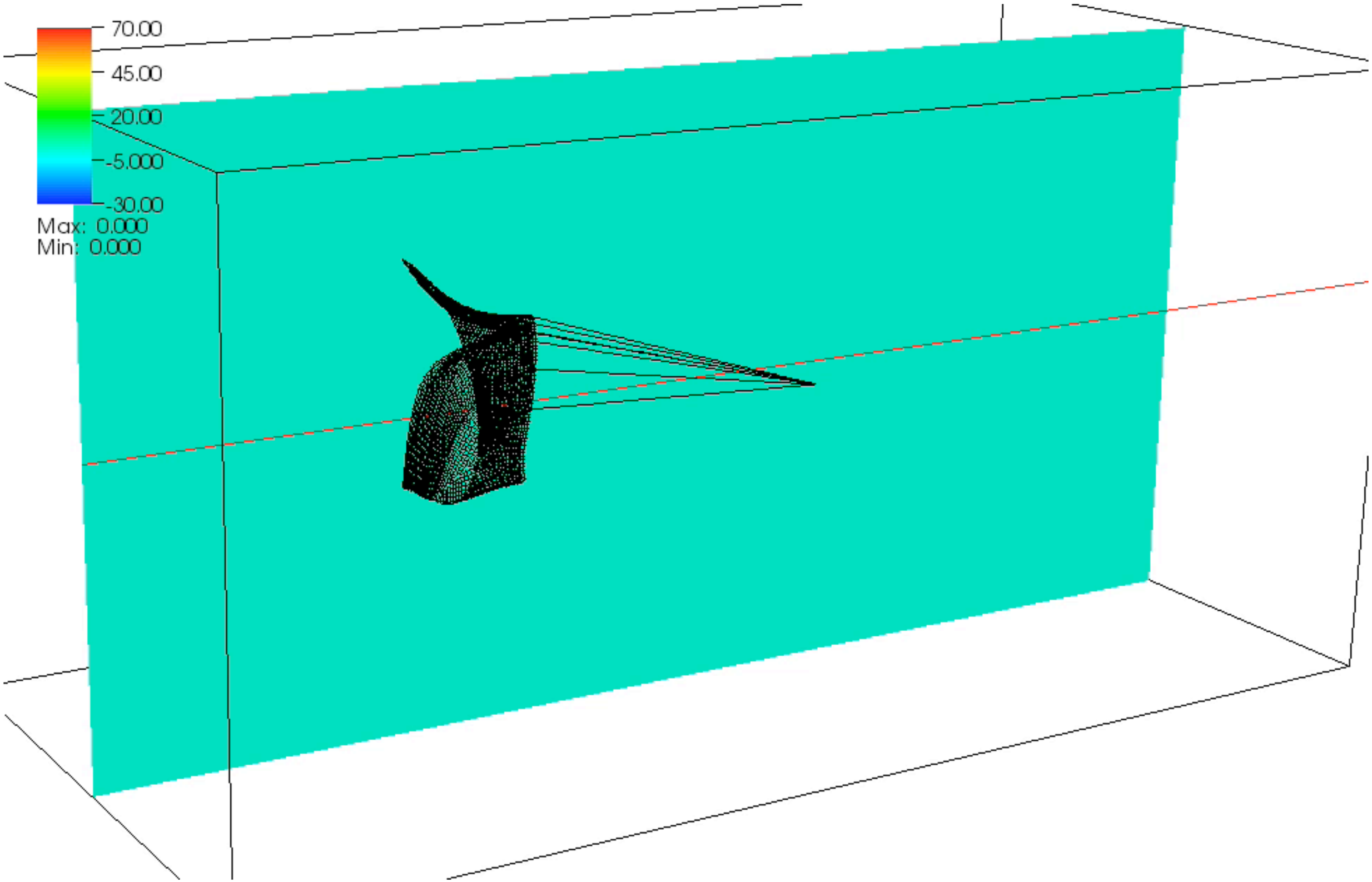
Simulating the fluid dynamics of a prosthetic mitral valve



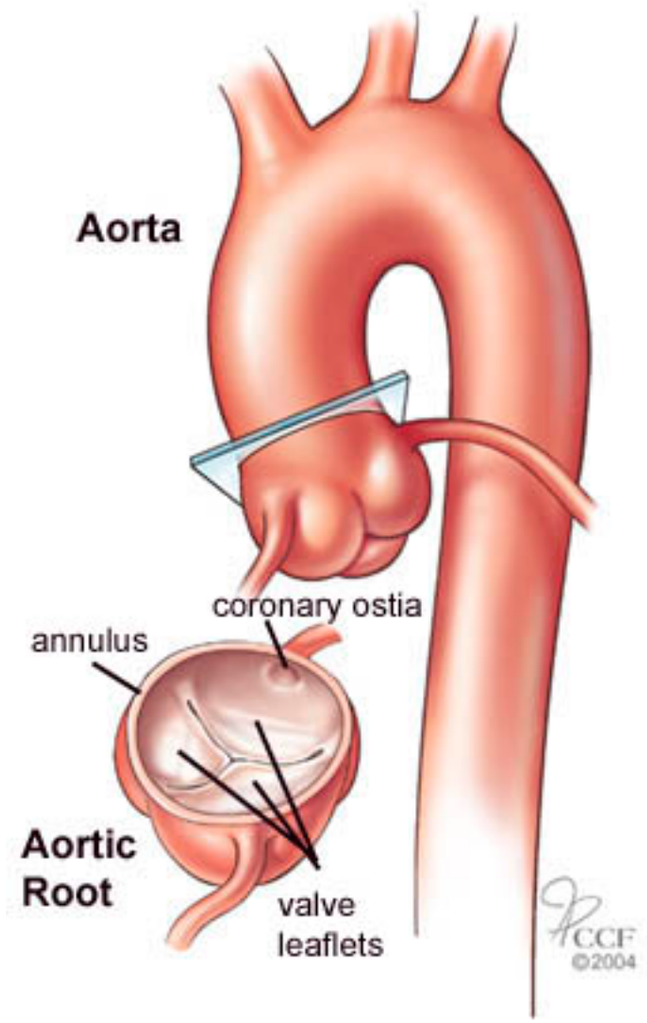


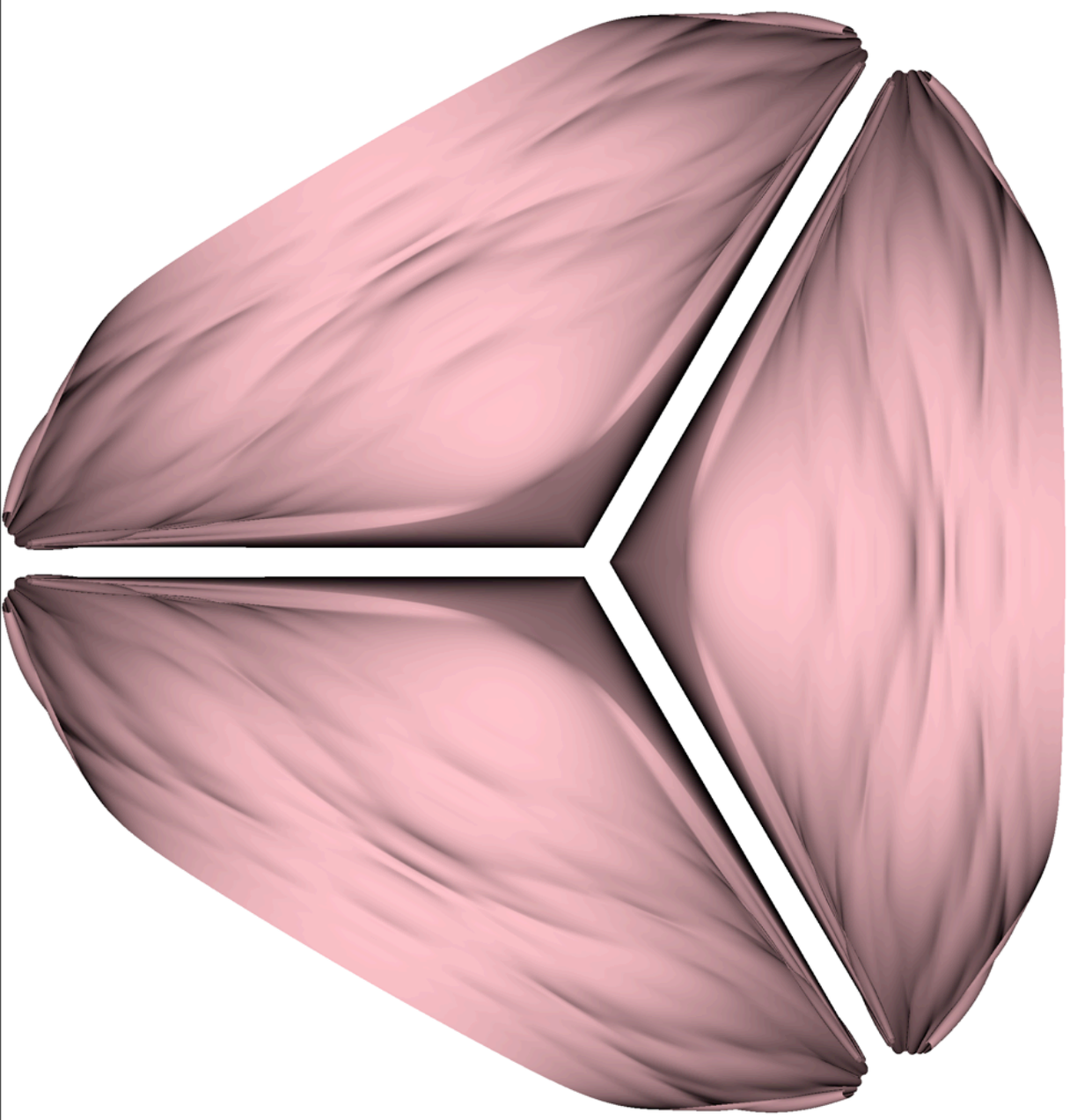
Left: Prosthetic mitral valve designed by D. Wheatley and his group at U. Glasgow. **Right:** Model of the prosthetic mitral valve developed by X. Luo and co-workers at U. Glasgow.

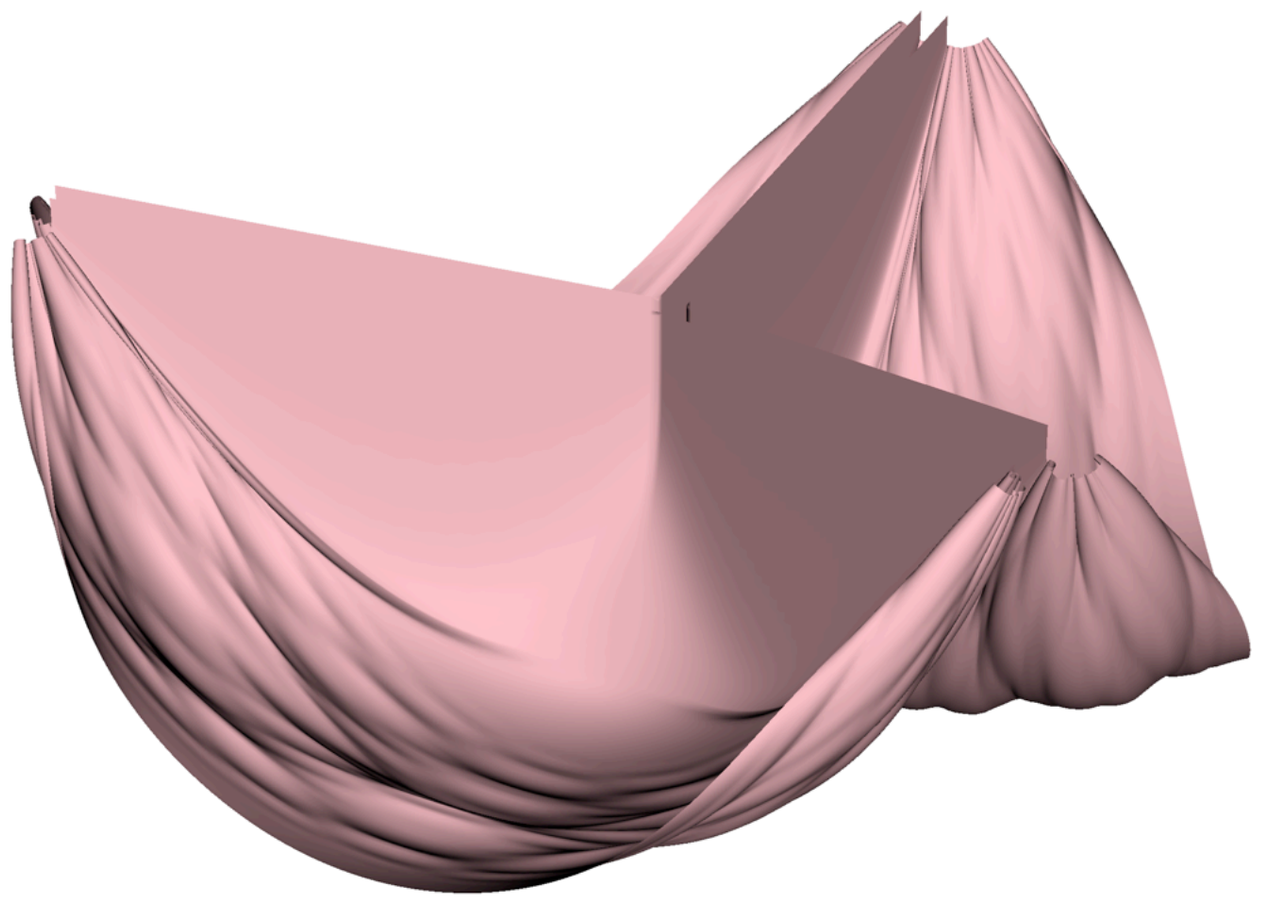


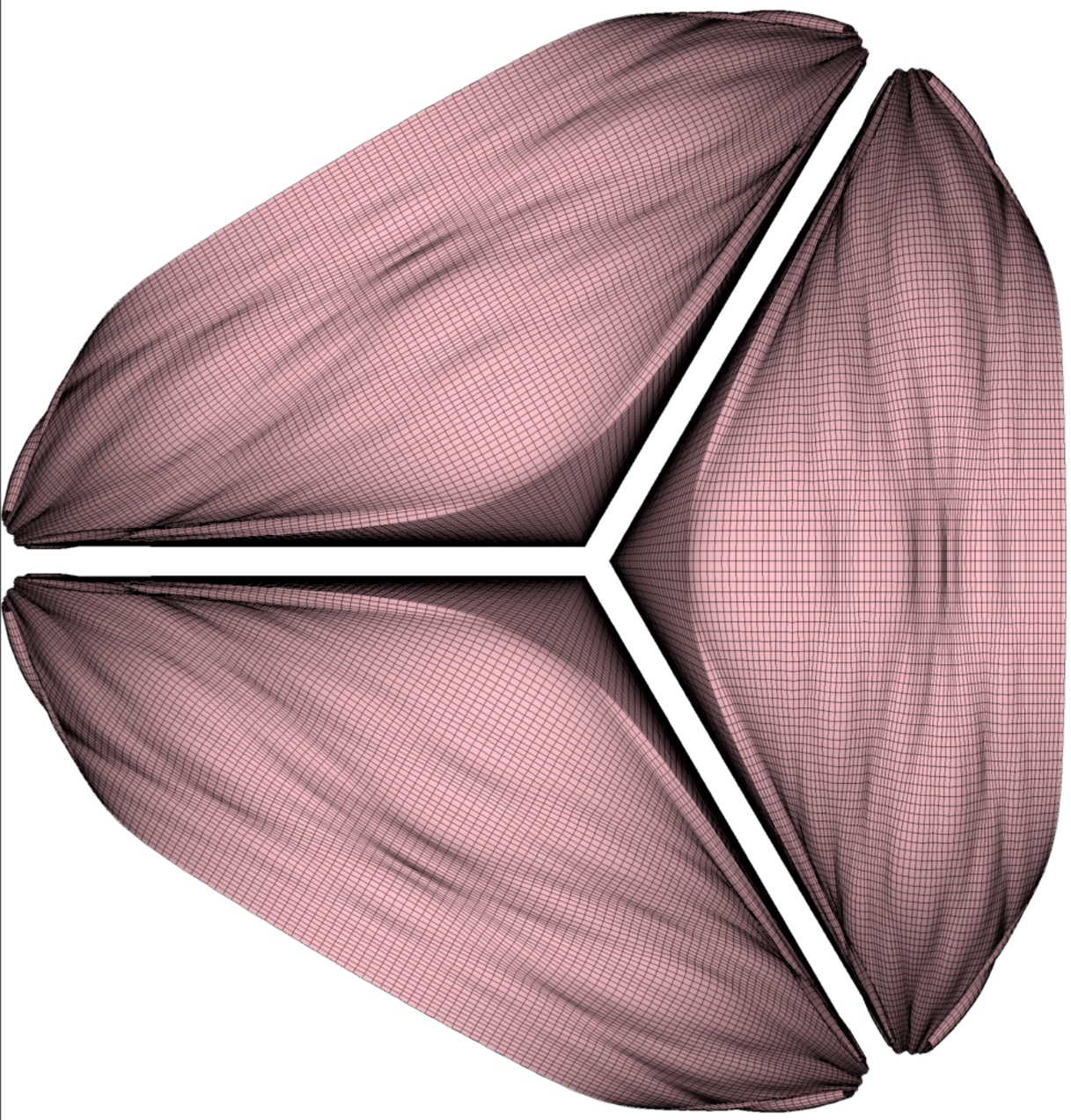

70.00
45.00
20.00
-5.000
-30.00
Max: 0.000
Min: 0.000

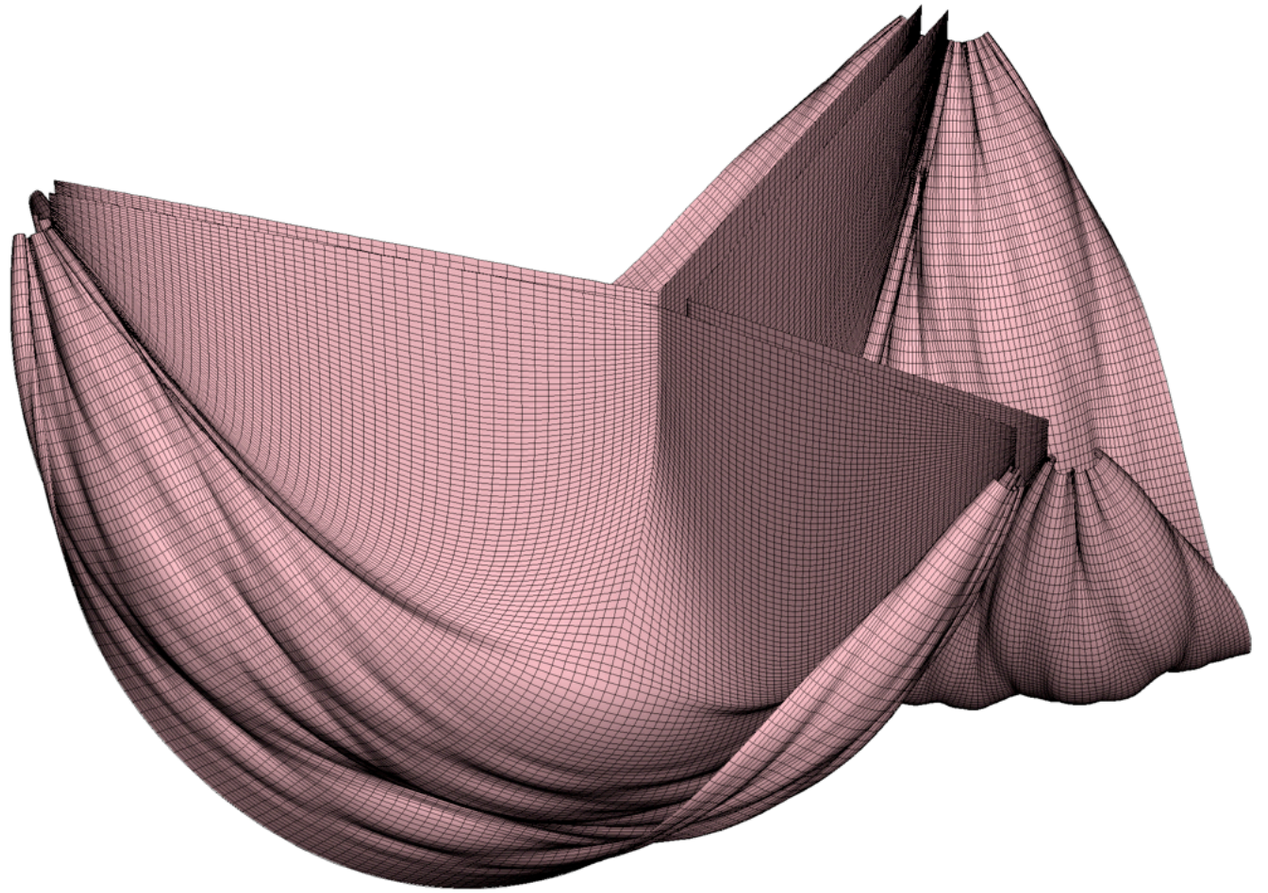
Simulating the fluid dynamics of a native aortic valve





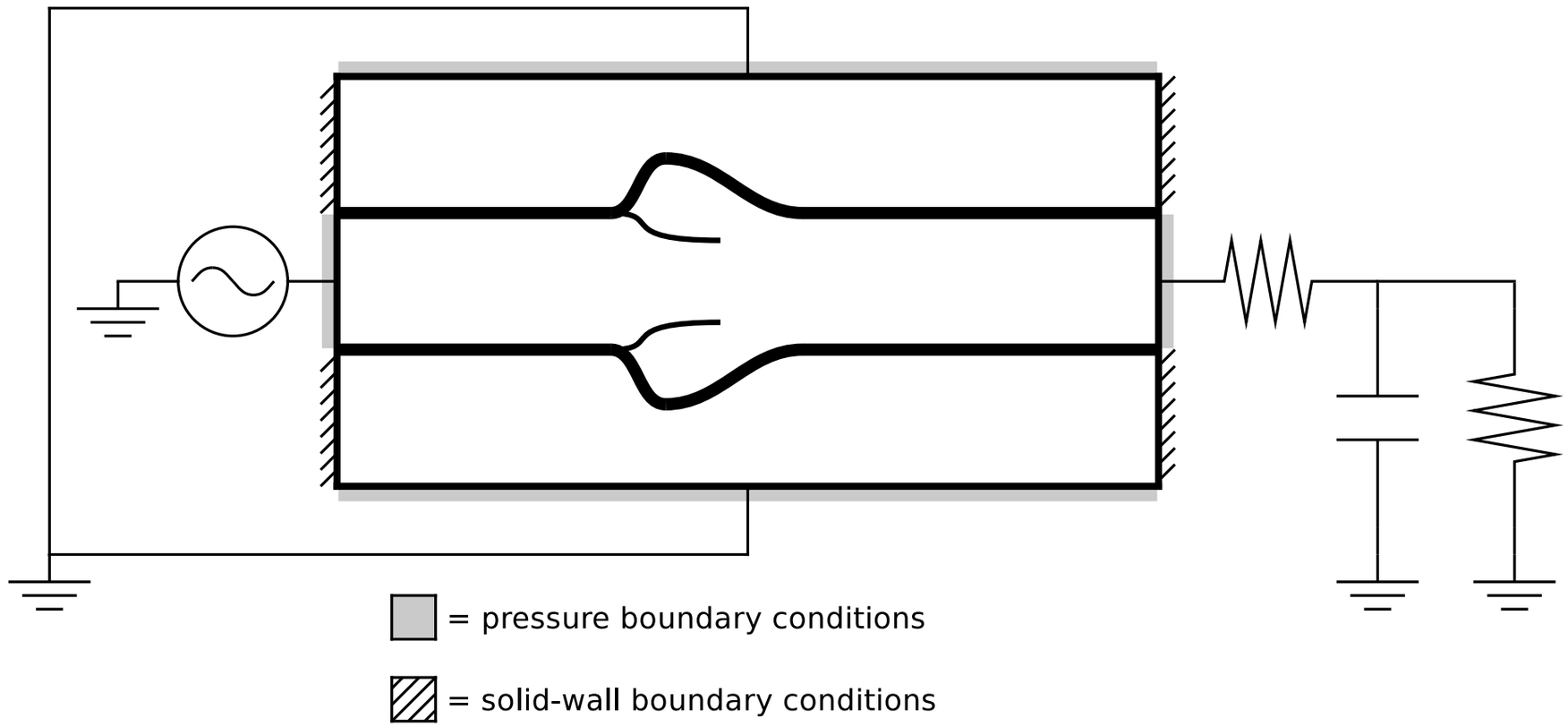






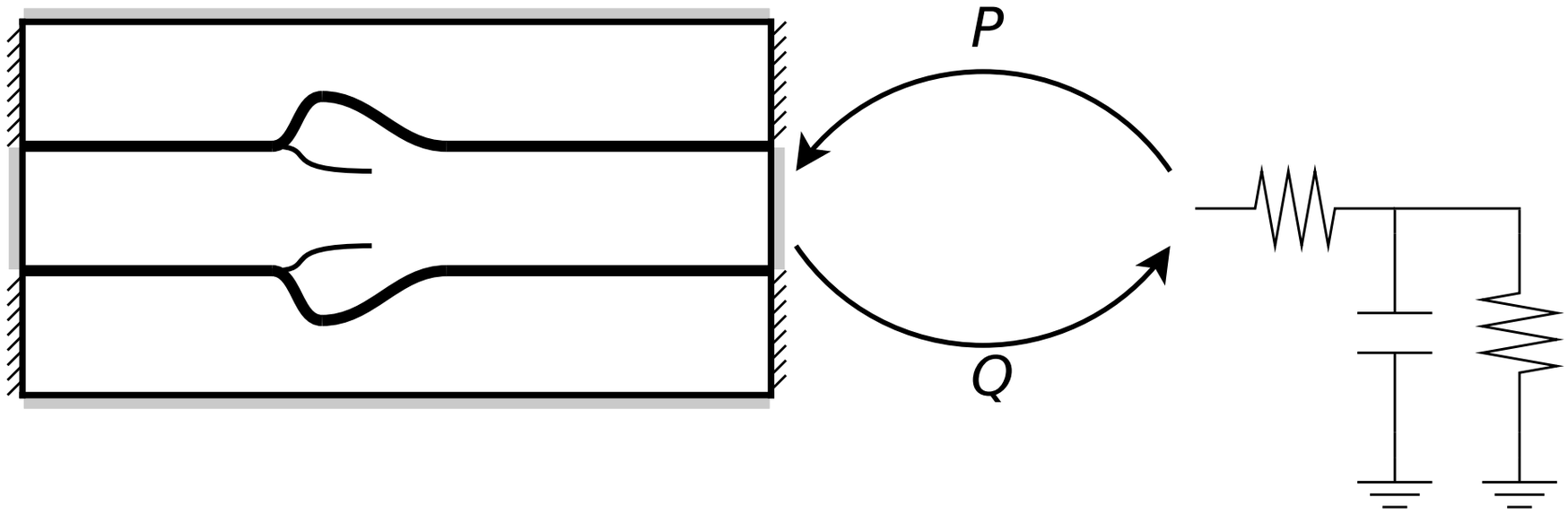
Coupling between *detailed* and *reduced* models

Use the reduced circulation model to provide pressure boundary conditions for the detailed FSI model, and use the detailed FSI model to provide flow boundary conditions for the reduced model.



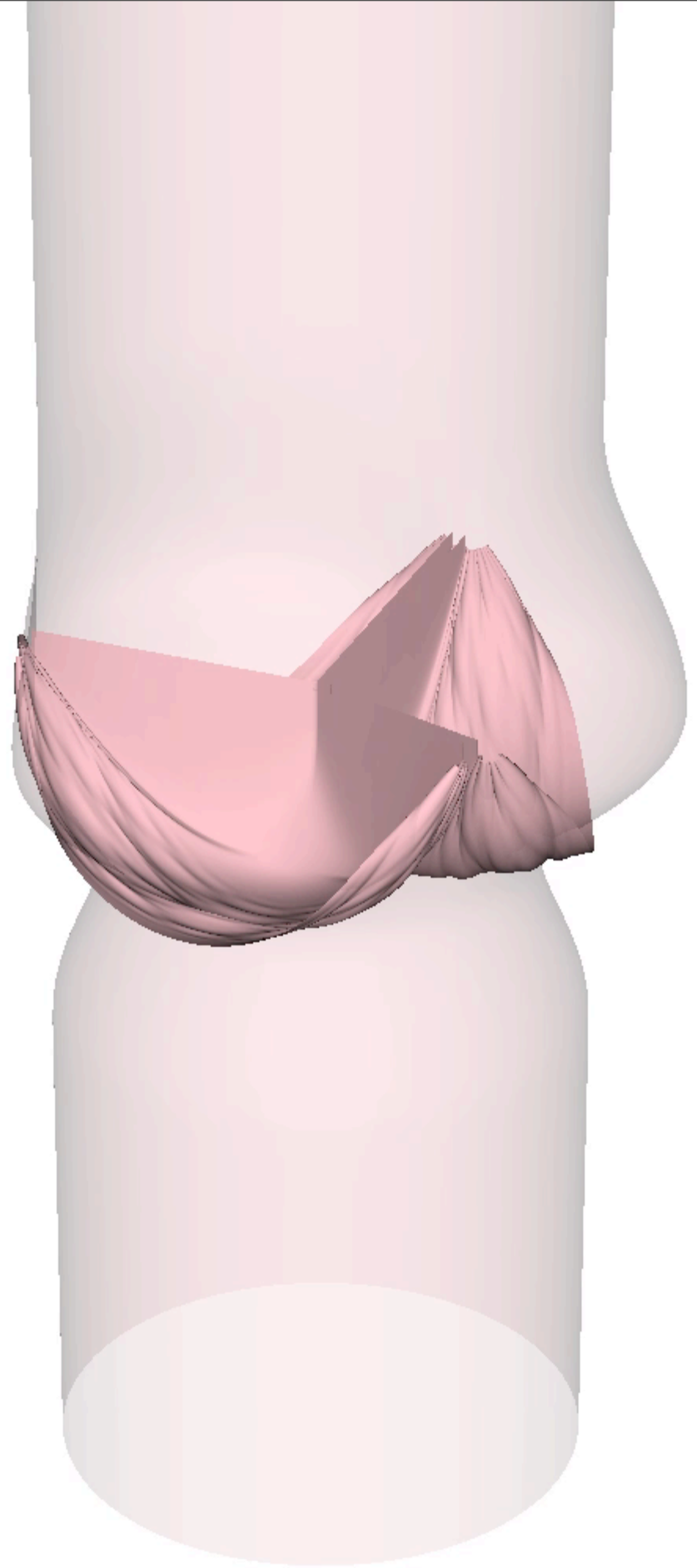
Coupling between *detailed* and *reduced* models

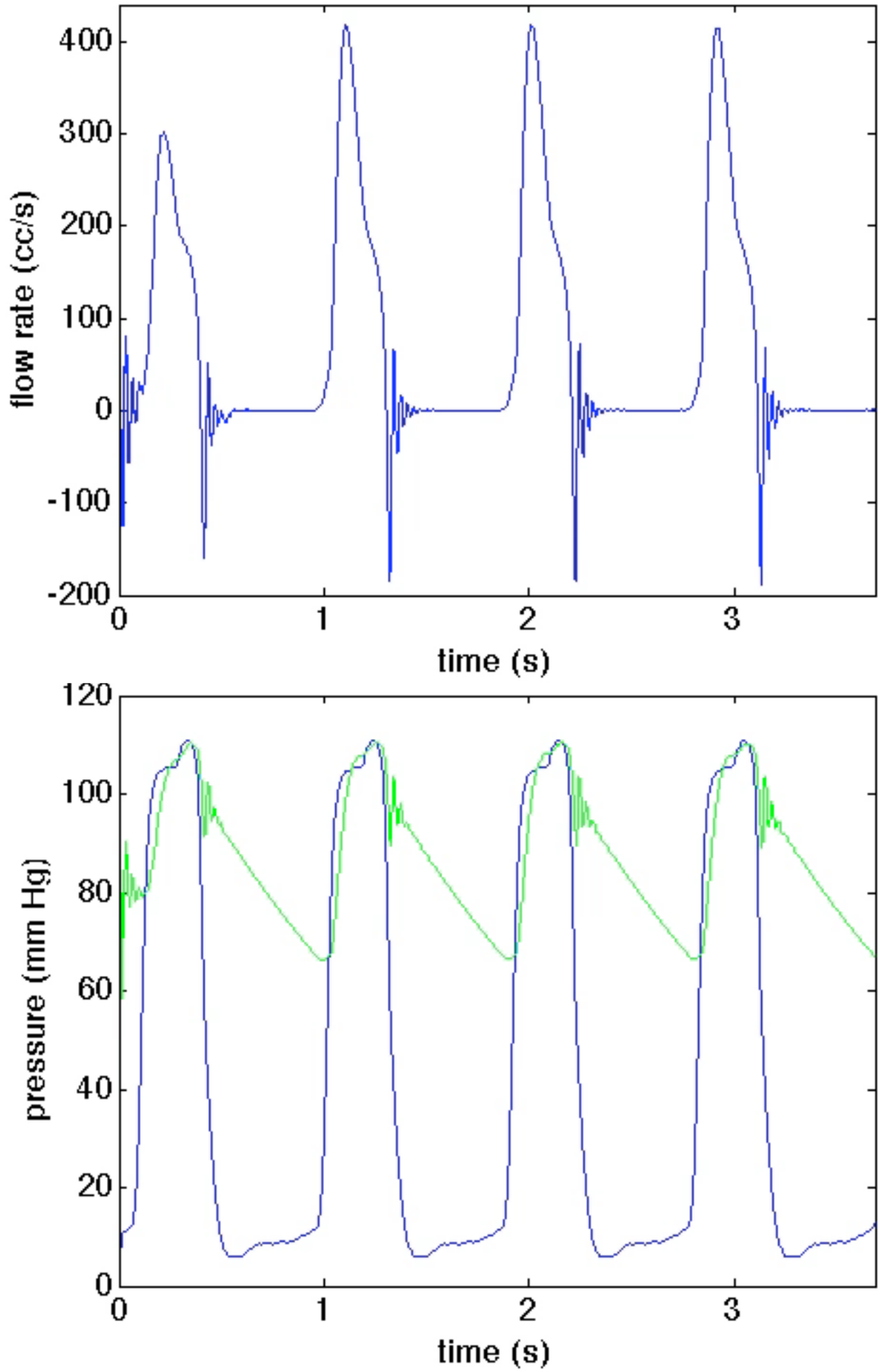
Use the reduced circulation model to provide pressure boundary conditions for the detailed FSI model, and use the detailed FSI model to provide flow boundary conditions for the reduced model.

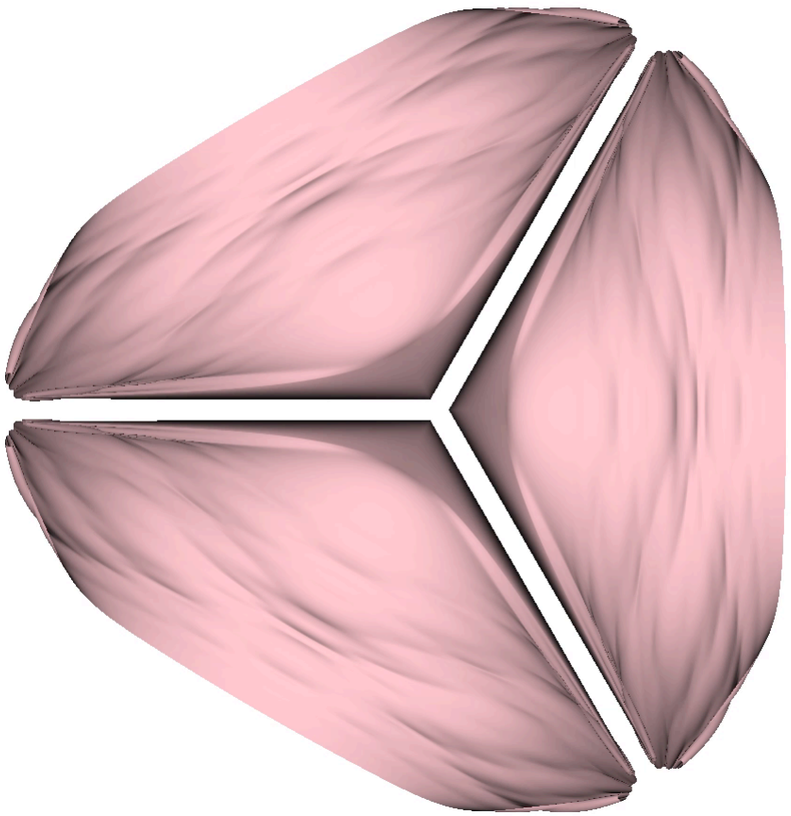


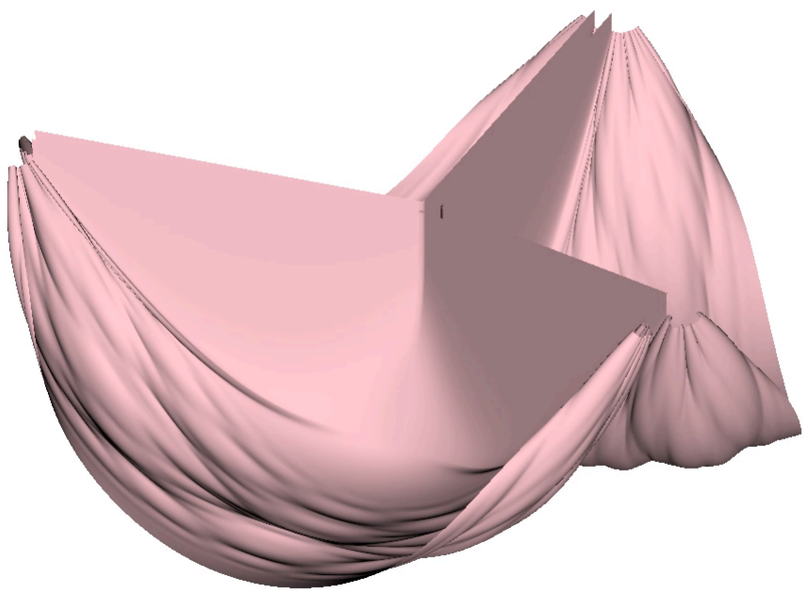
Step 1: Treat the stored pressure in the Windkessel model as fixed and solve the IB equations over a time increment Δt for $\mathbf{X}^{n+1}$, $\mathbf{u}^{n+1}$, and $p^{n+\frac{1}{2}}$.

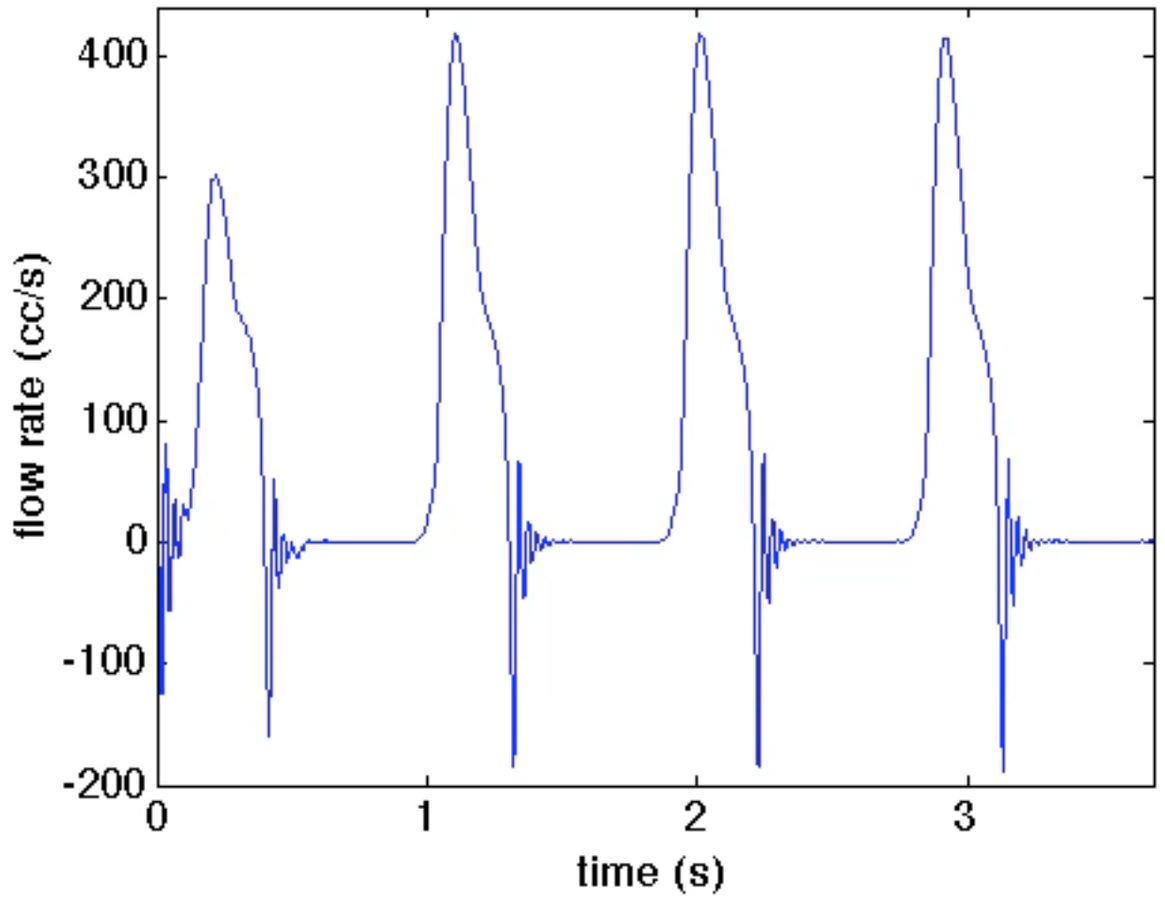
Step 2: Treat $\mathbf{X}^{n+1}$, $\mathbf{u}^{n+1}$, and $p^{n+\frac{1}{2}}$ as fixed, compute $Q = Q^{n+1}$, the instantaneous flow rate out of the detailed model vessel into the reduced model, and solve the reduced model equations over a time increment Δt for P^{n+1}.

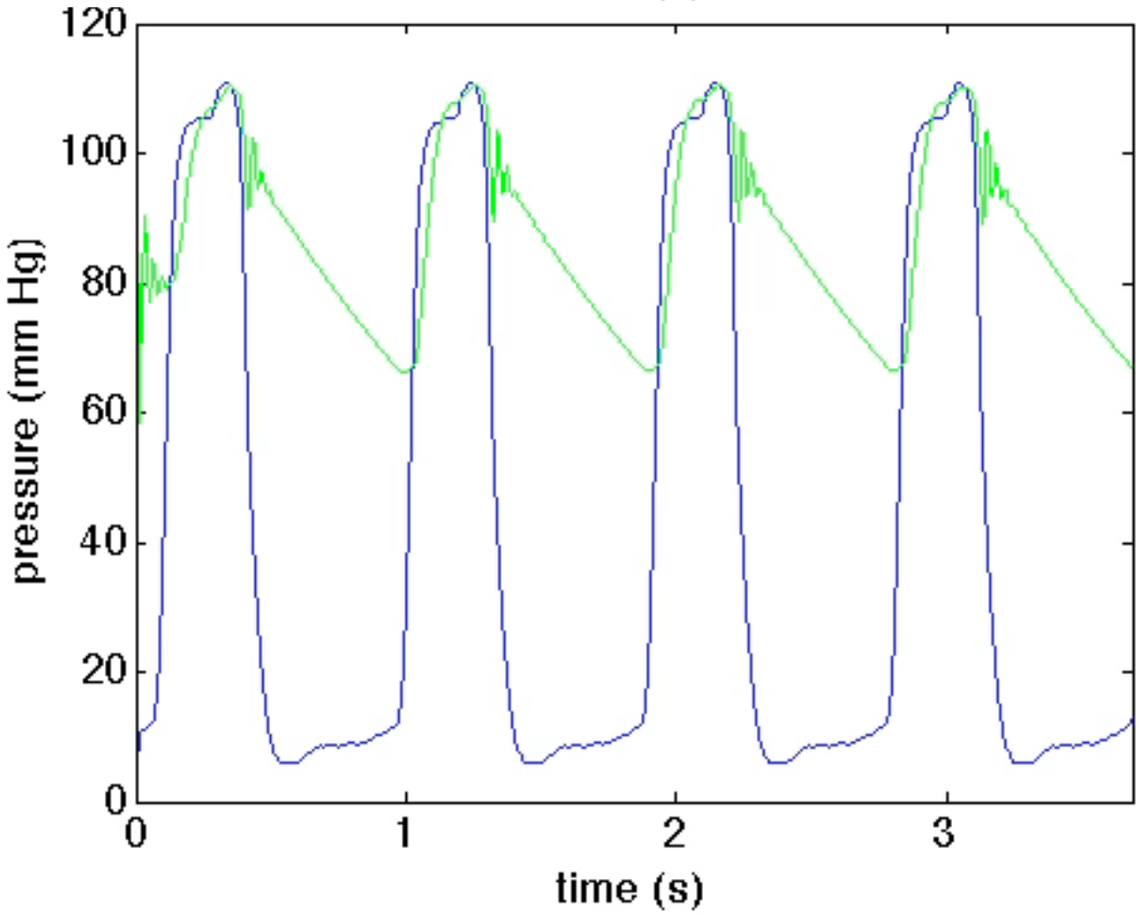


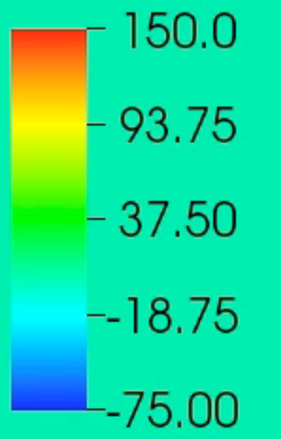

flow rate (cc/s)
400
300
200
100
0
-100
-200
0
1
2
3
time (s)
pressure (mm Hg)
120
100
80
60
40
20
0
0
1
2
3
time (s)

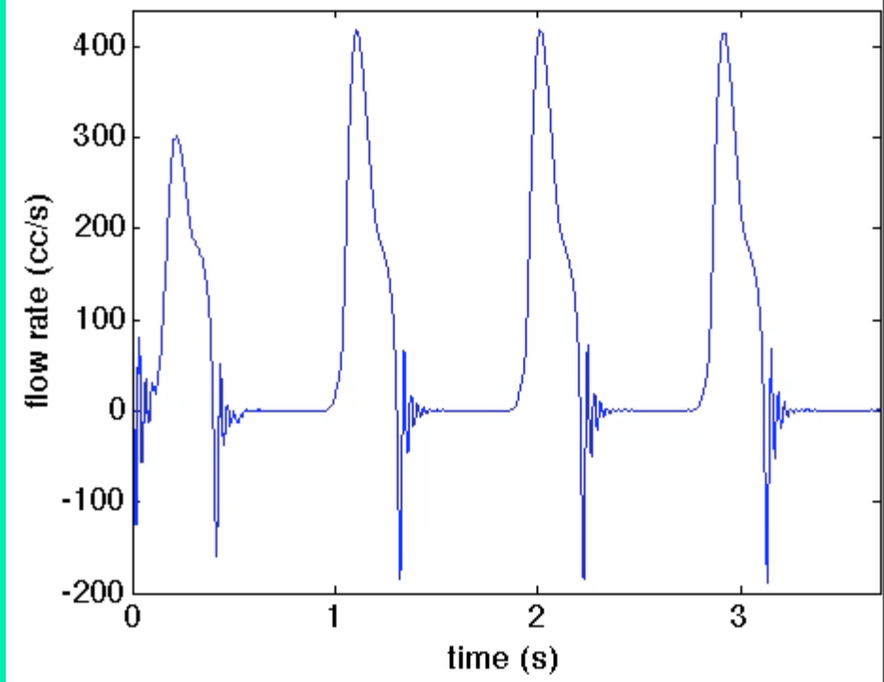


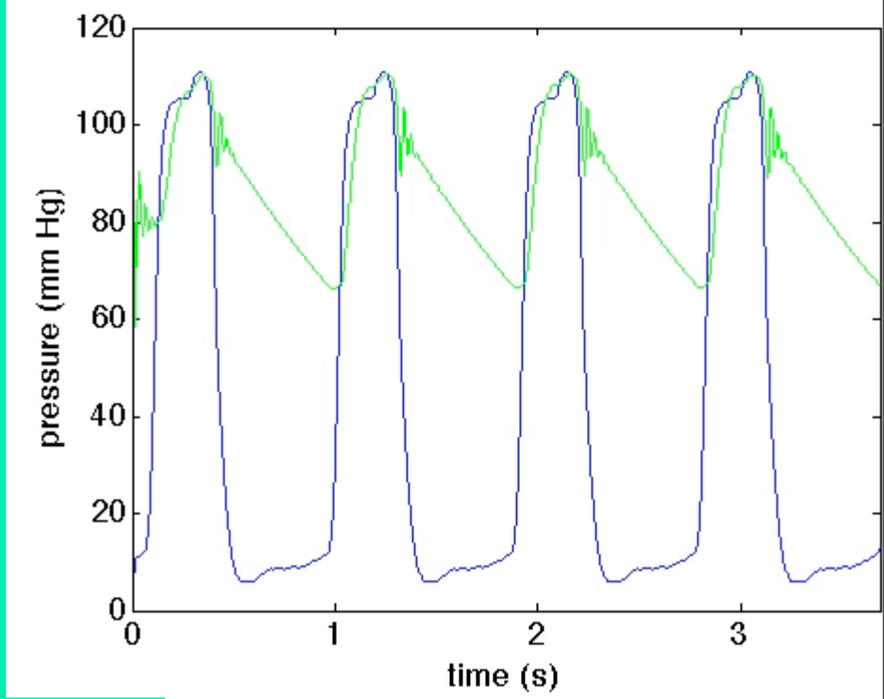


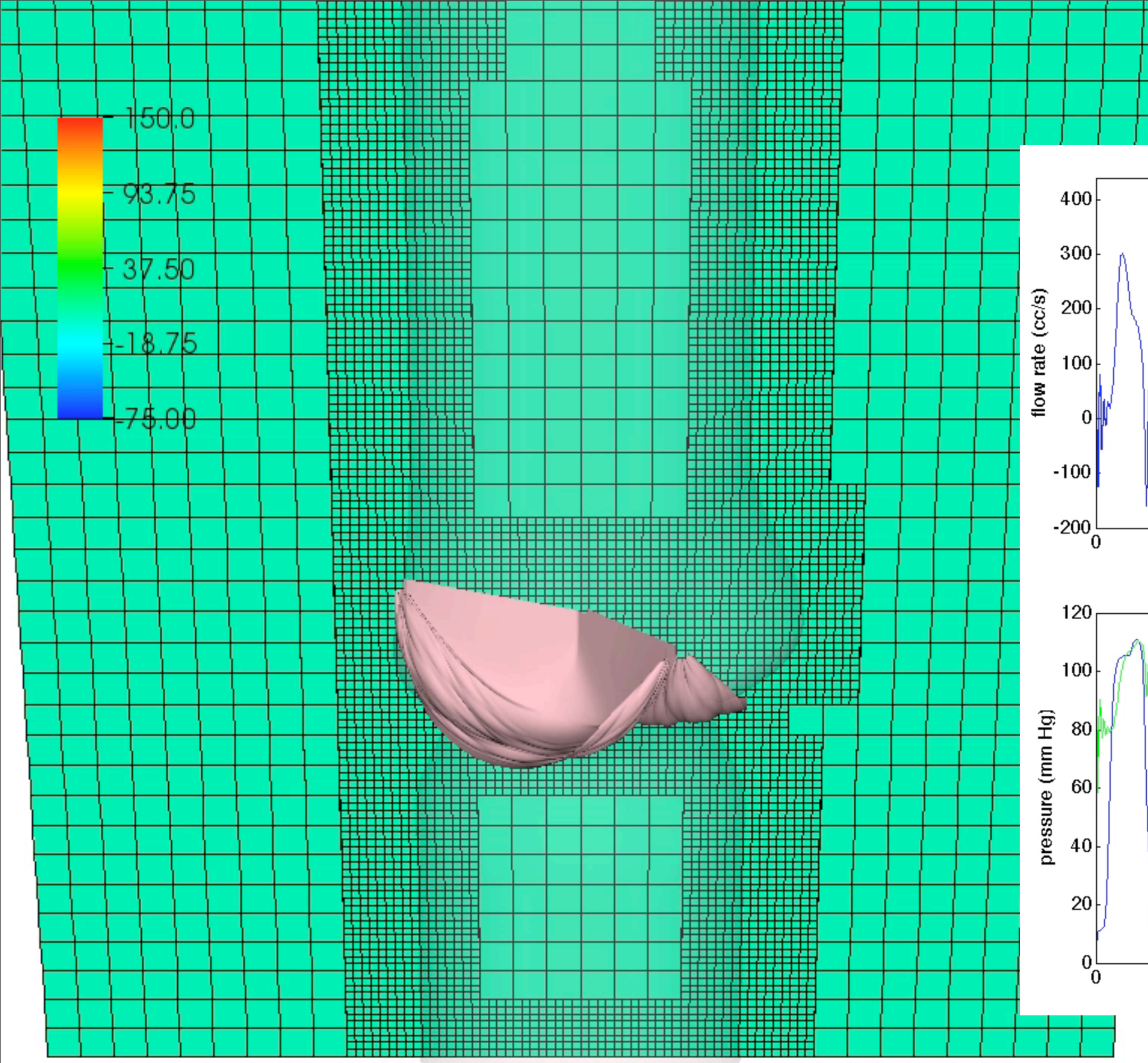

400
300
200
100
0
-100
-200
flow rate (cc/s)
0
1
2
3
time (s)

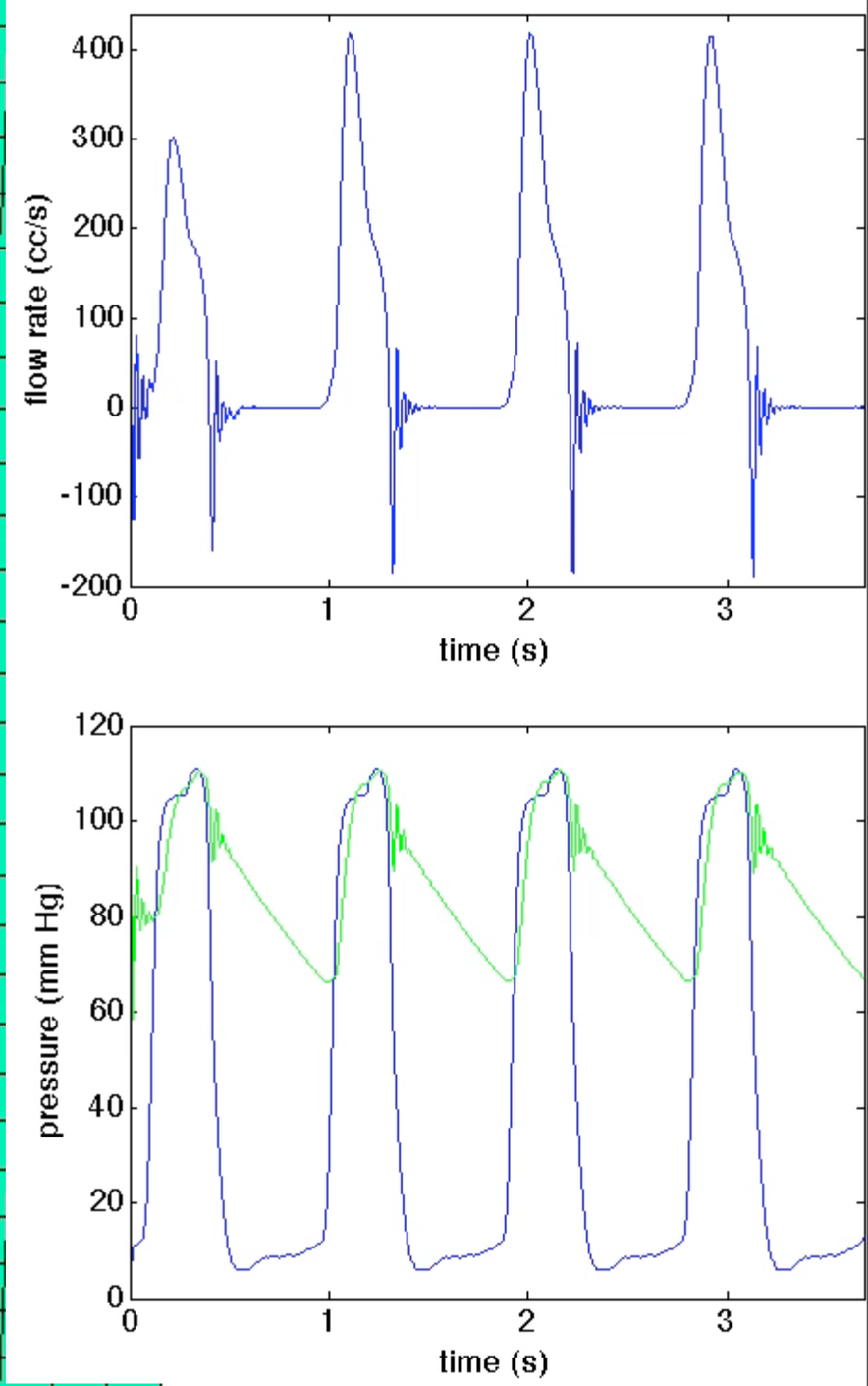

120
100
80
60
40
20
0
pressure (mm Hg)
0
1
2
3
time (s)

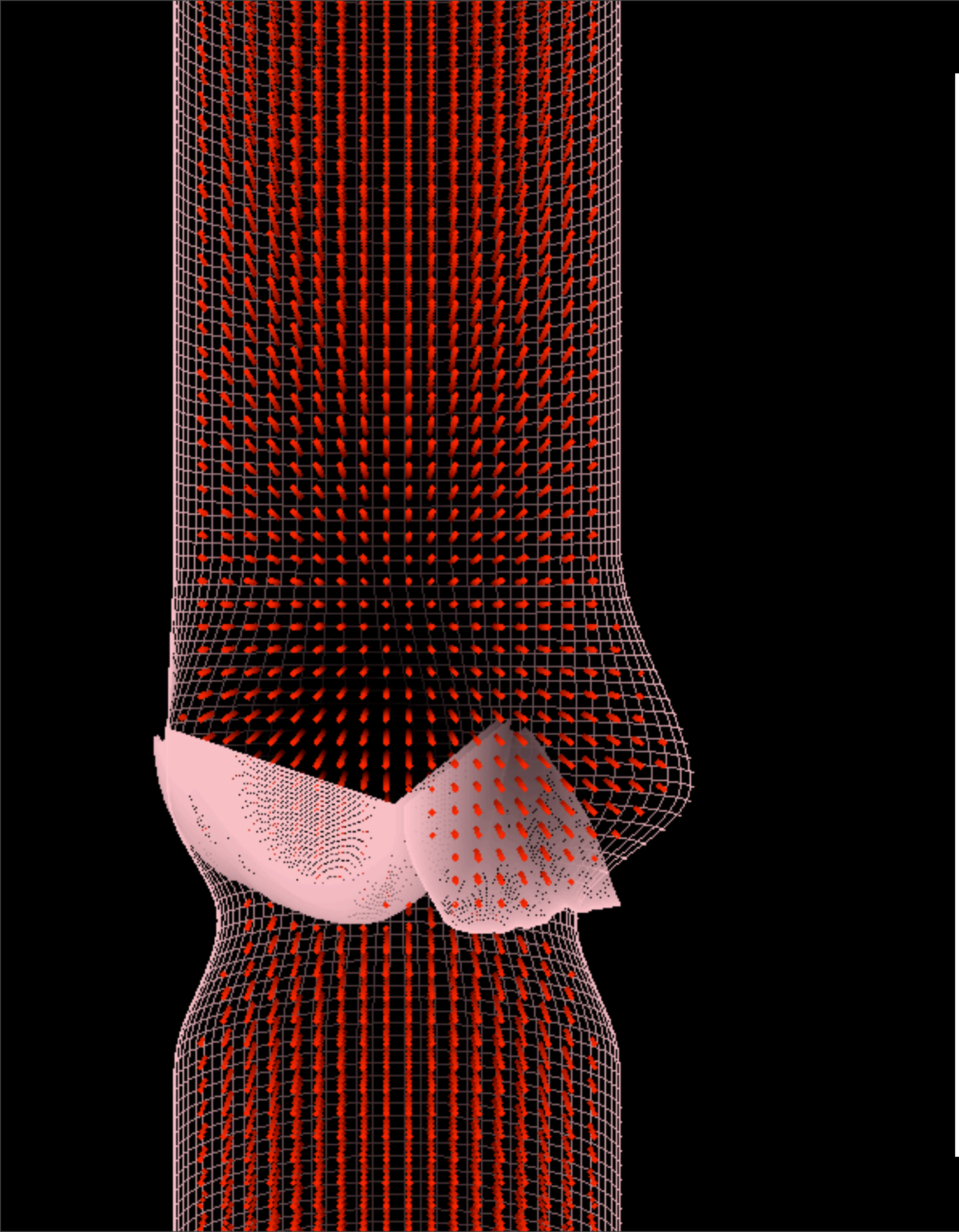

150.0
93.75
37.50
-18.75
-75.00
Max: 0.000
Min: 0.000

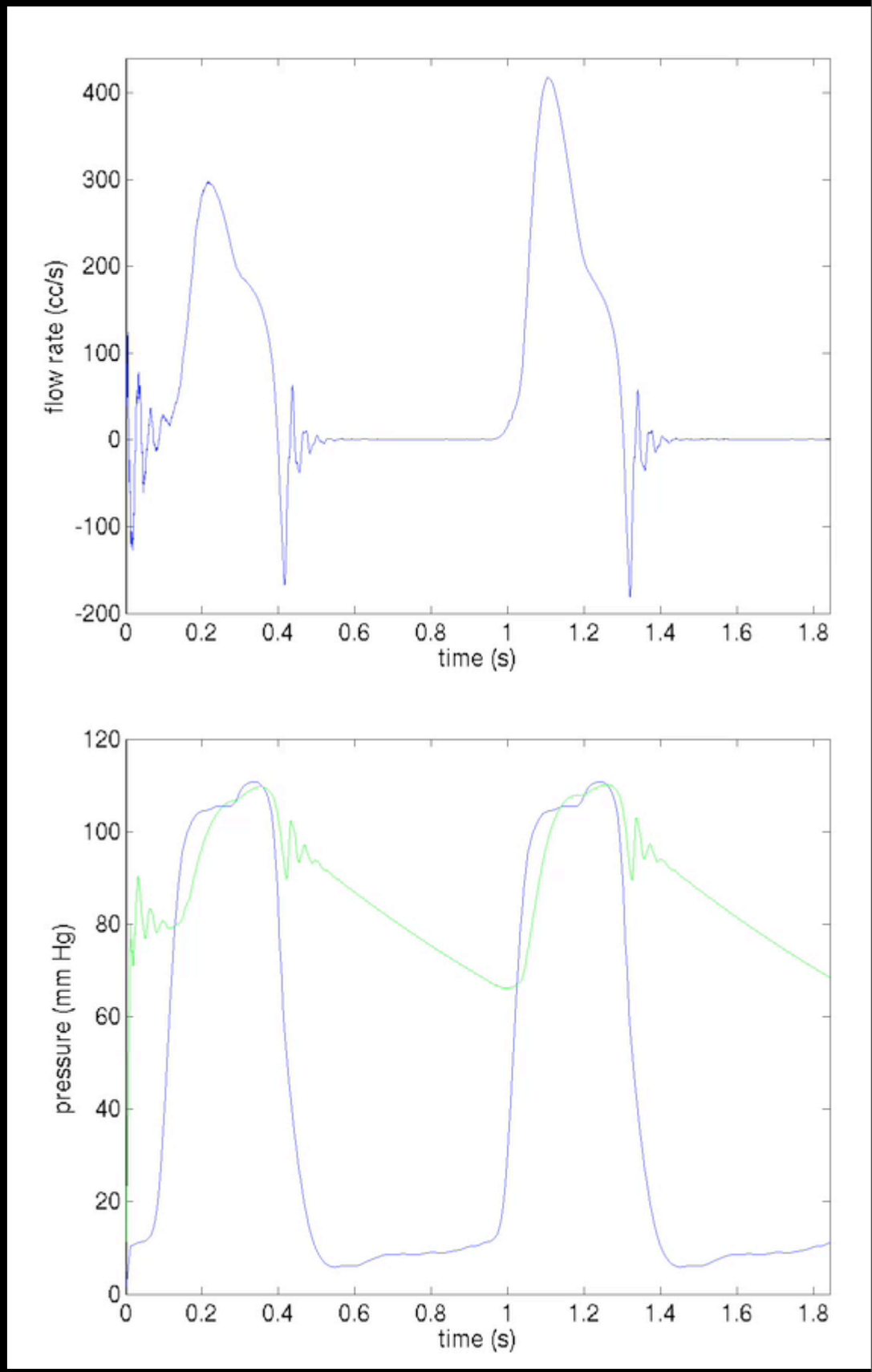

flow rate (cc/s)
400
300
200
100
0
-100
-200
0
1
2
3
time (s)


pressure (mm Hg)
120
100
80
60
40
20
0
0
1
2
3
time (s)


150.0
93.75
37.50
-18.75
-75.00

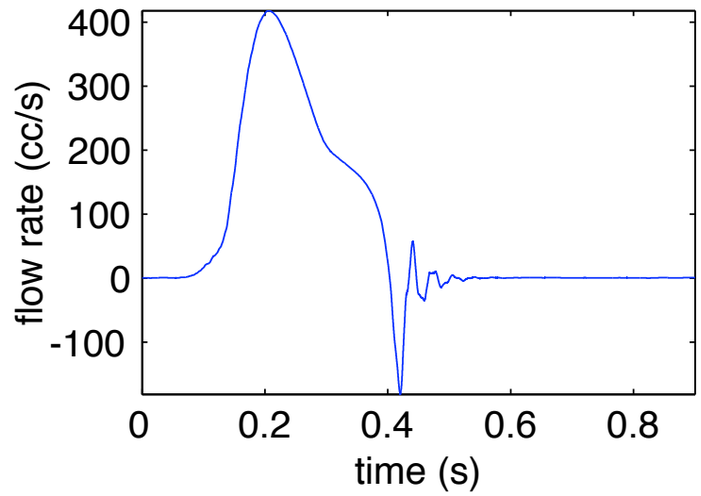

flow rate (cc/s)
400
300
200
100
0
-100
-200
0
1
2
3
time (s)
pressure (mm Hg)
120
100
80
60
40
20
0
0
1
2
3
time (s)



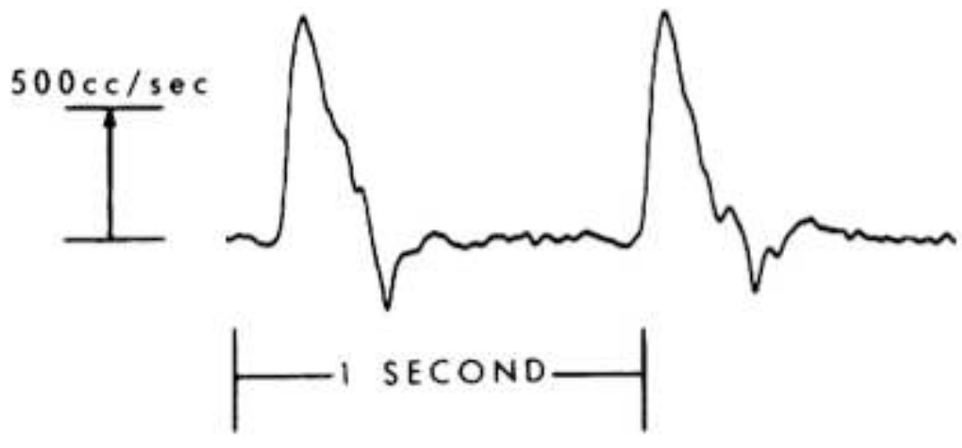

flow rate (cc/s)
400
300
200
100
0
-100
-200
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
time (s)
pressure (mm Hg)
120
100
80
60
40
20
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
time (s)

Comparison of simulated and clinical flow rates through the aortic valve

A.



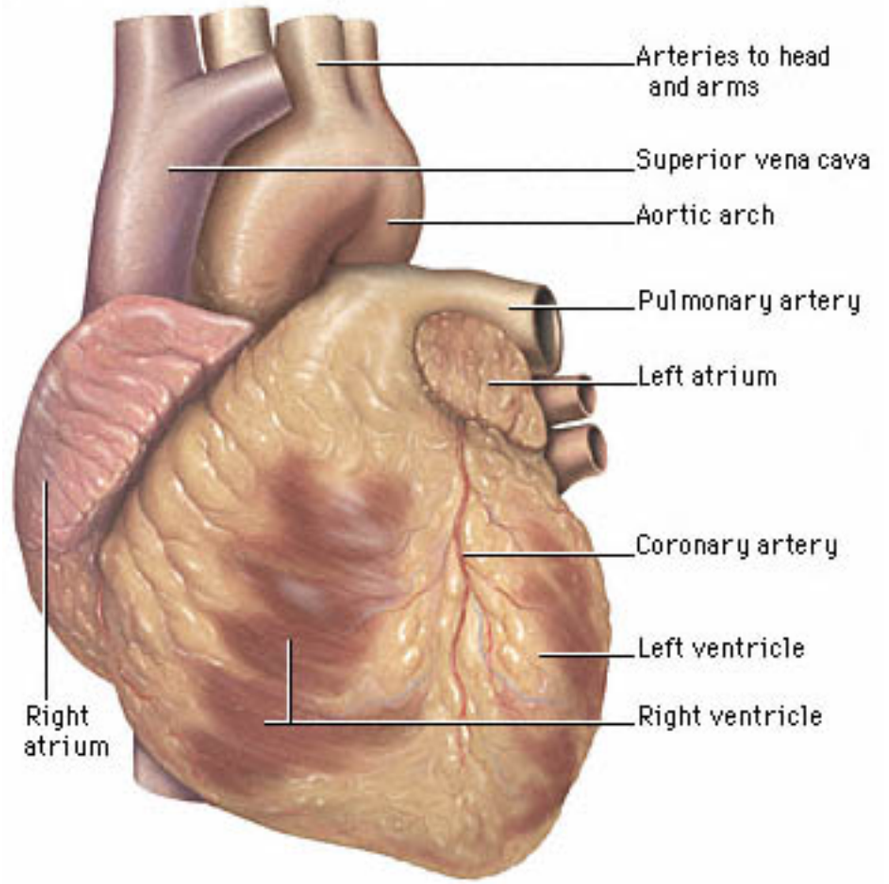
B.



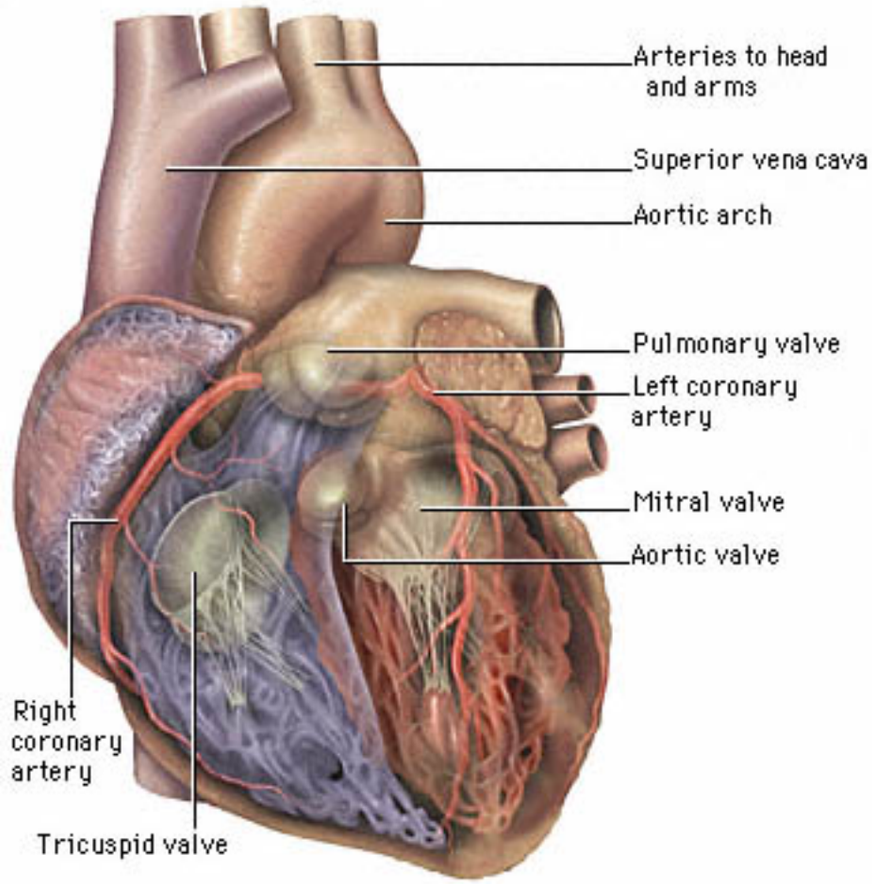
A: Flow rate through the model valve. **B:** Aortic flow in a healthy human subject (from Murgo et al., *Circulation*, 1980).

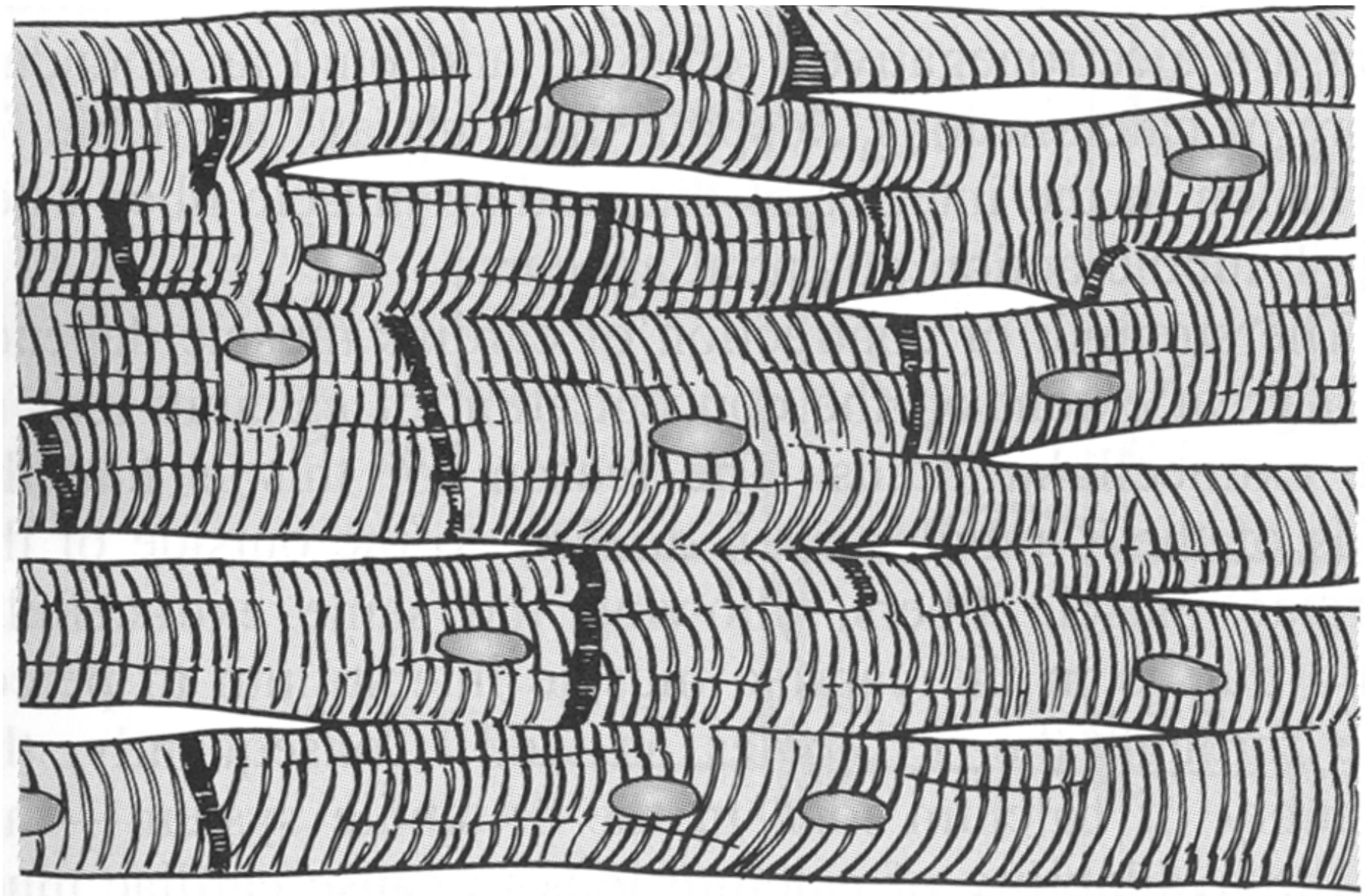
Note: In the model, only the upstream driving pressure and downstream pressure load are specified. The flow rate *is not* specified in the model; instead, it *emerges* from the FSI simulation.

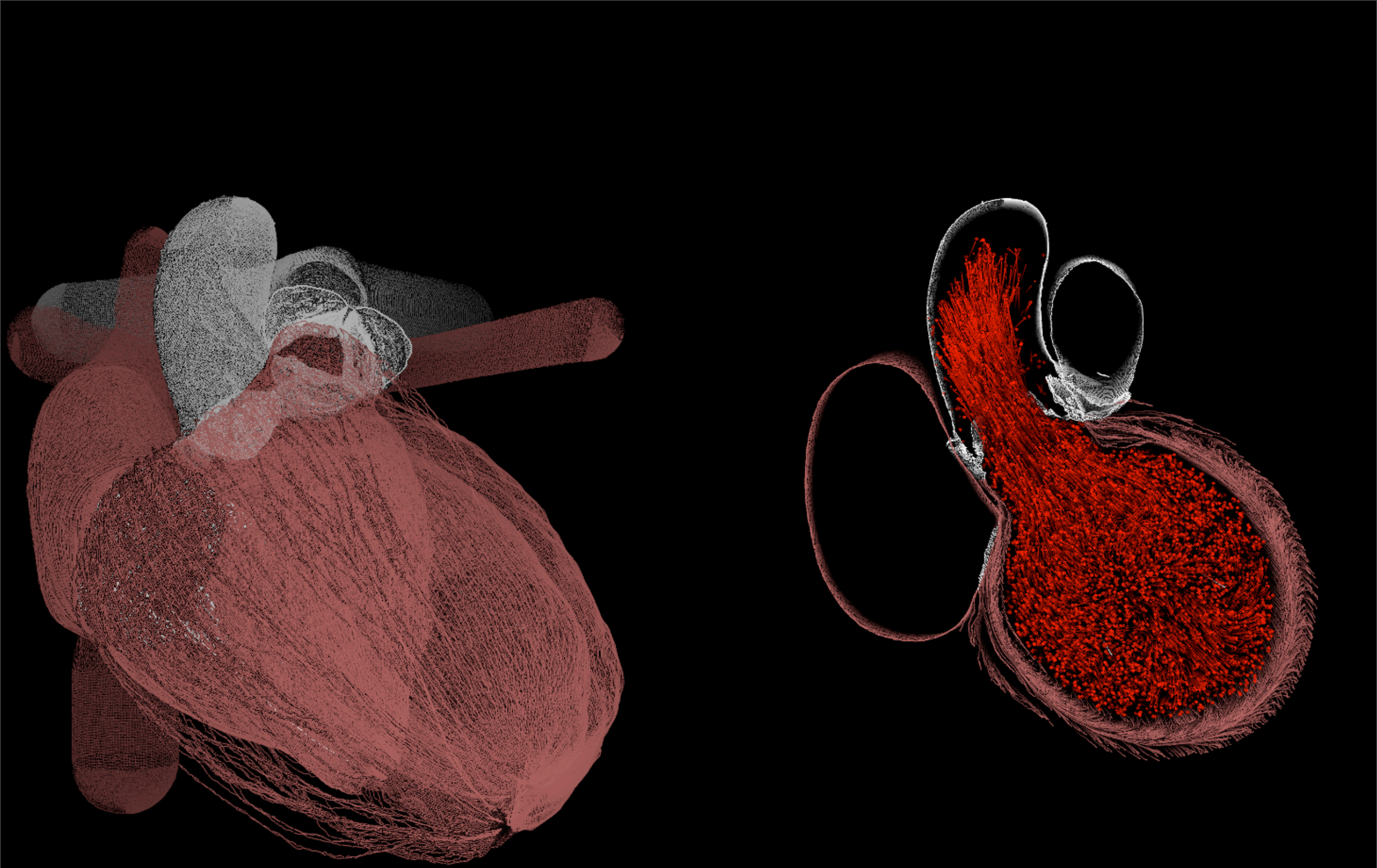
Exterior structures of the heart

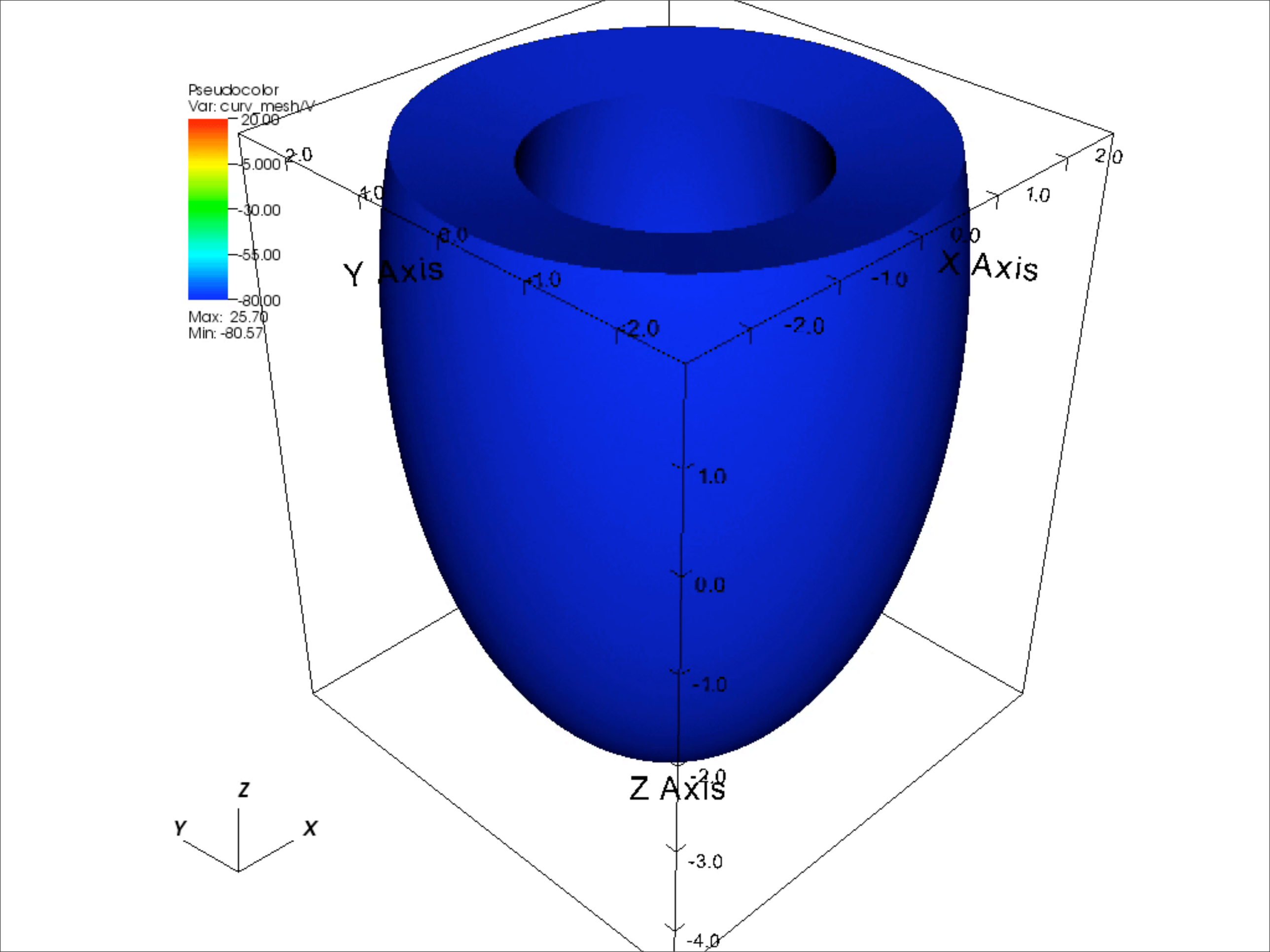


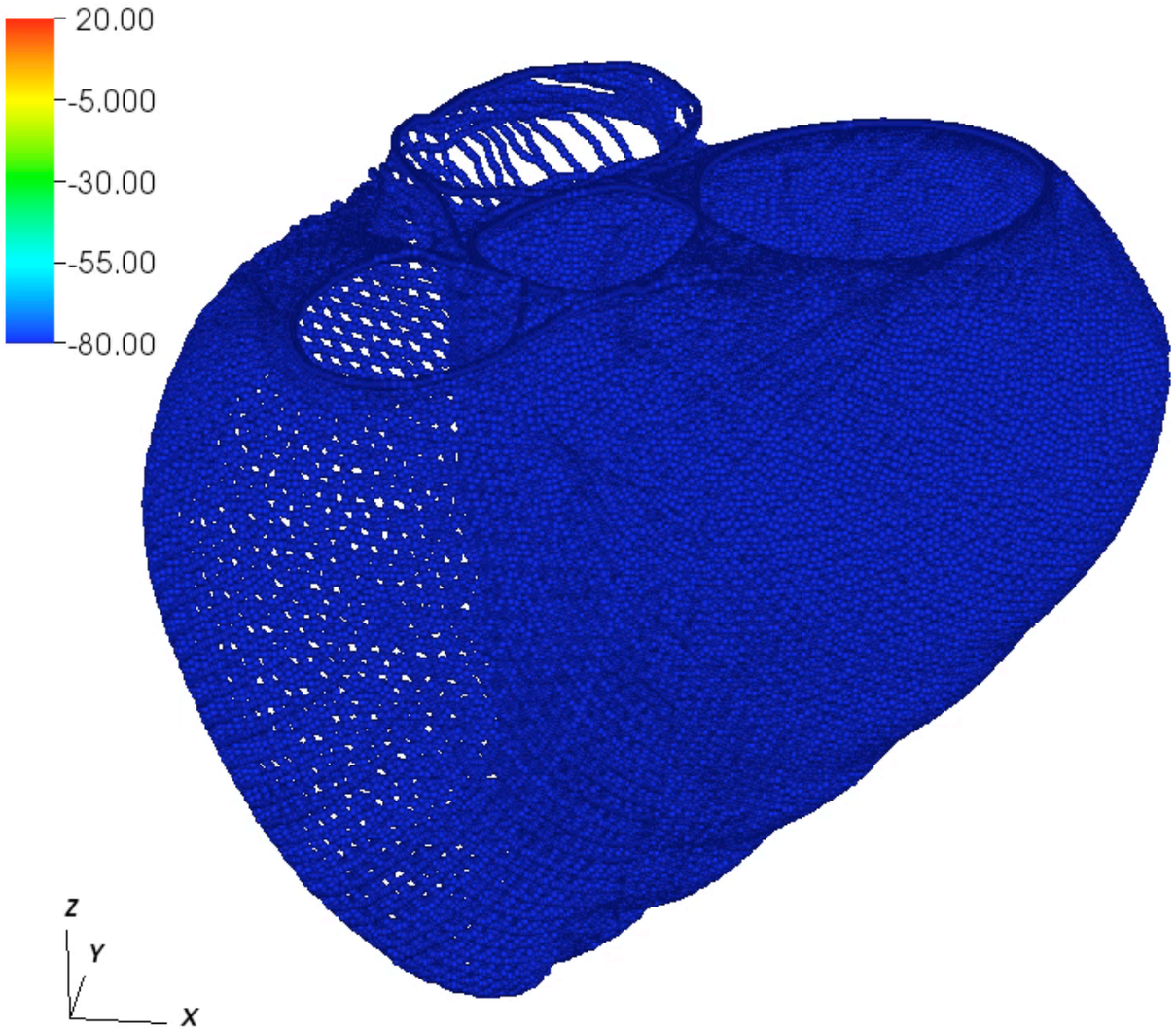
Interior structures of the heart

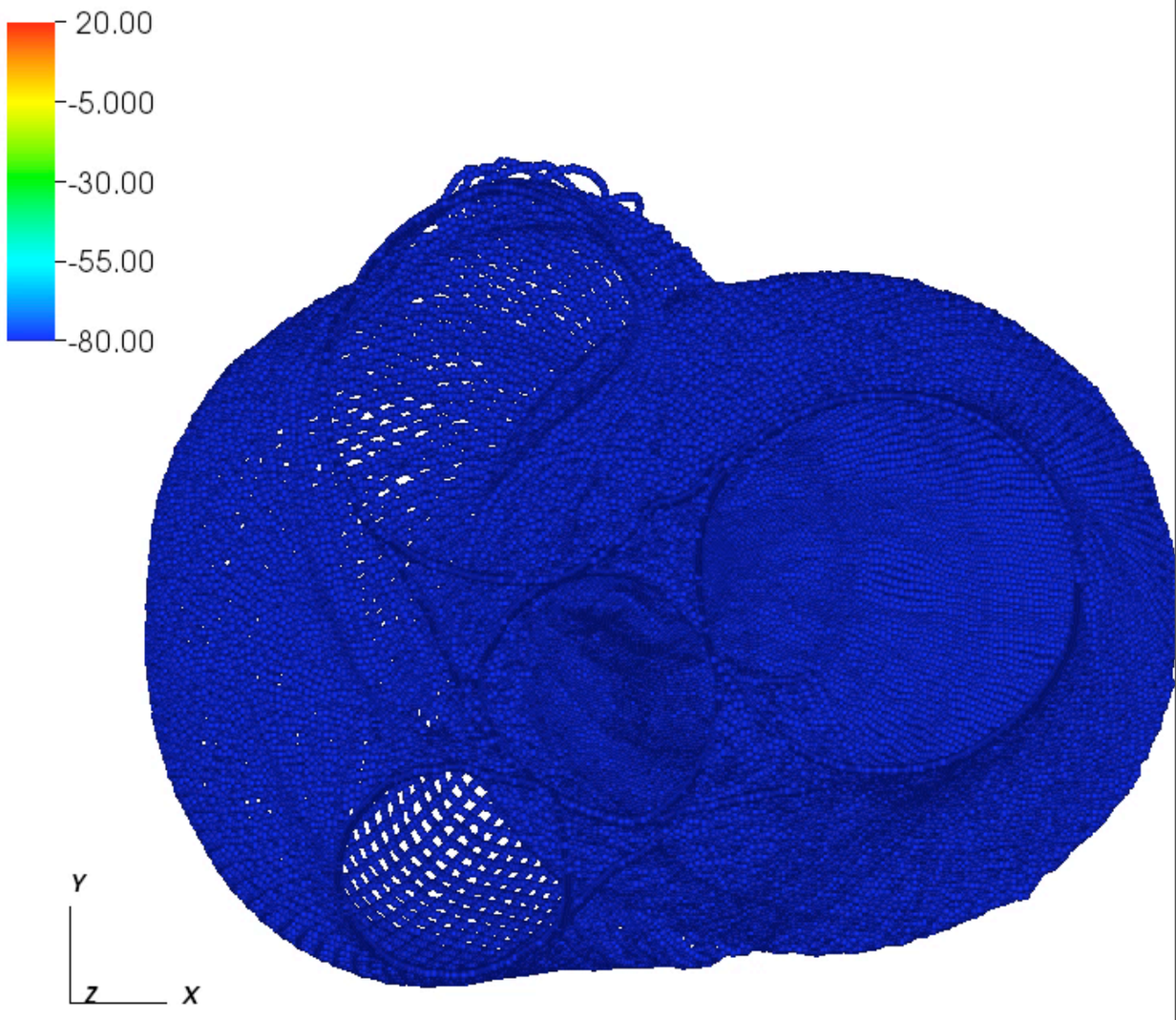







Pseudocolor
Var: curv_meshAV
20.00
5.000
-30.00
-55.00
-80.00
Max: 25.70
Min: -80.57
Y Axis
X Axis
Z Axis
2.0
1.0
0.0
-1.0
-2.0
-3.0
-4.0
Z
Y
X


20.00
-5.000
-30.00
-55.00
-80.00
Z
Y
X


20.00
-5.000
-30.00
-55.00
-80.00
Y
Z
X

Pseudocolor
Var: ventricles3d_24576_vertices/V
20.00
-5.000
-30.00
-55.00
-80.00
Pseudocolor
Var: v_e
10.00
5.000
0.000
-5.000
-10.00
Z
Y
X

Pseudocolor
Var: ventricles3d_24576_vertices/V
20.00
-5.000
-30.00
-55.00
-80.00
Pseudocolor
Var: v_e
10.00
5.000
0.000
-5.000
-10.00
Z
Y
X
Pseudocolor
Var: v_e
10.00
5.000
0.000
-5.000
-10.00
Y
X
Pseudocolor
Var: v_e
10.00
5.000
0.000
-5.000
-10.00
Z
X

An IB method with finite element (FE) mechanics

Fiber-based structure models

- are well-suited for applications involving extremely anisotropic materials;
- permit an especially simple discretization; and
- may be used to handle complex geometry.

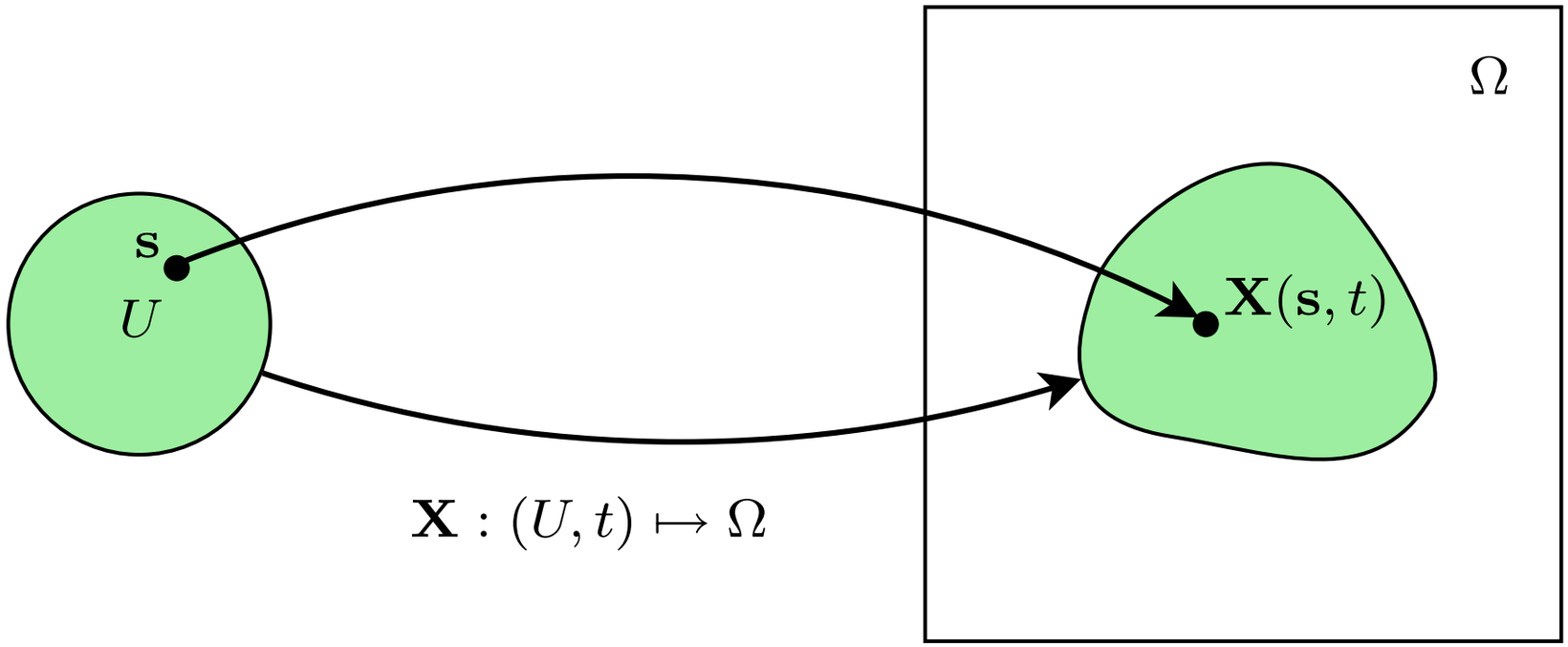
Additionally, with fiber models, the Lagrangian mesh must be approximately twice as fine as the background Eulerian grid to avoid *leaks* at fluid-structure interfaces.

Unlike fiber models, FE-based structure models

- can take advantage of modern mesh generation technology;
- can easily use isotropic, orthotropic, or anisotropic material models, including experimentally characterized constitutive models; and
- are straightforward to discretize in an adaptive manner.

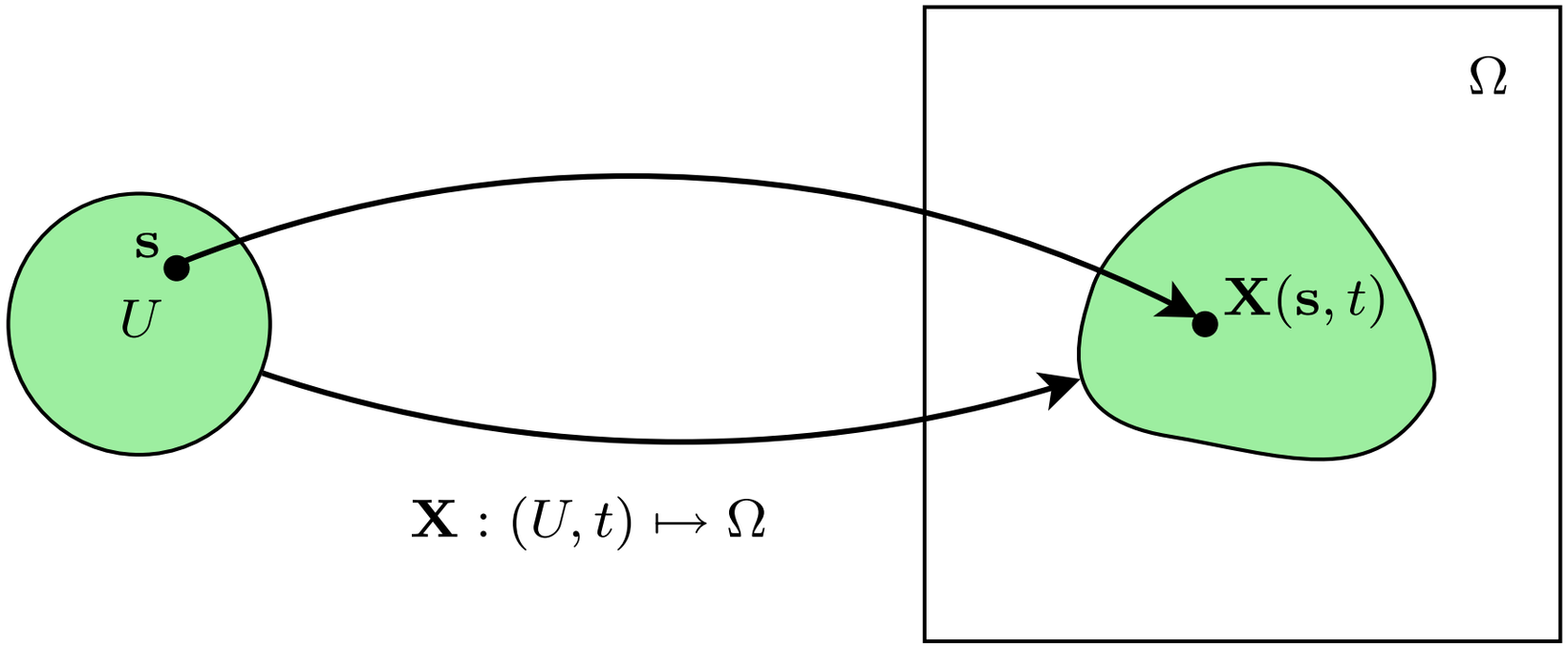
Moreover, using FE structure models, it becomes possible for the Lagrangian and Eulerian discretizations to be chosen *independently*.

Lagrangian and Eulerian notation



- $\mathbf{s} \in U$ are Lagrangian (material) coordinates, with $U \subset \mathbb{R}^3$ the Lagrangian coordinate domain,
- $\mathbf{x} \in \Omega$ are Eulerian (spatial) coordinates, with $\Omega \subset \mathbb{R}^3$ the physical domain,
- $\mathbf{X}(\mathbf{s},t) \in \Omega$ is the physical position of material point $\mathbf{s}$ at time t,
- $U_t = \mathbf{X}(U,t) \subseteq \Omega$ is the physical region occupied by the elastic structure at time t, and
- $\Omega \setminus U_t = \Omega \setminus \mathbf{X}(U,t)$ is the physical region occupied by the fluid at time t.

Stress in Eulerian and Lagrangian forms



If $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{x},t)$ is the Cauchy stress tensor of the fluid-structure system, then

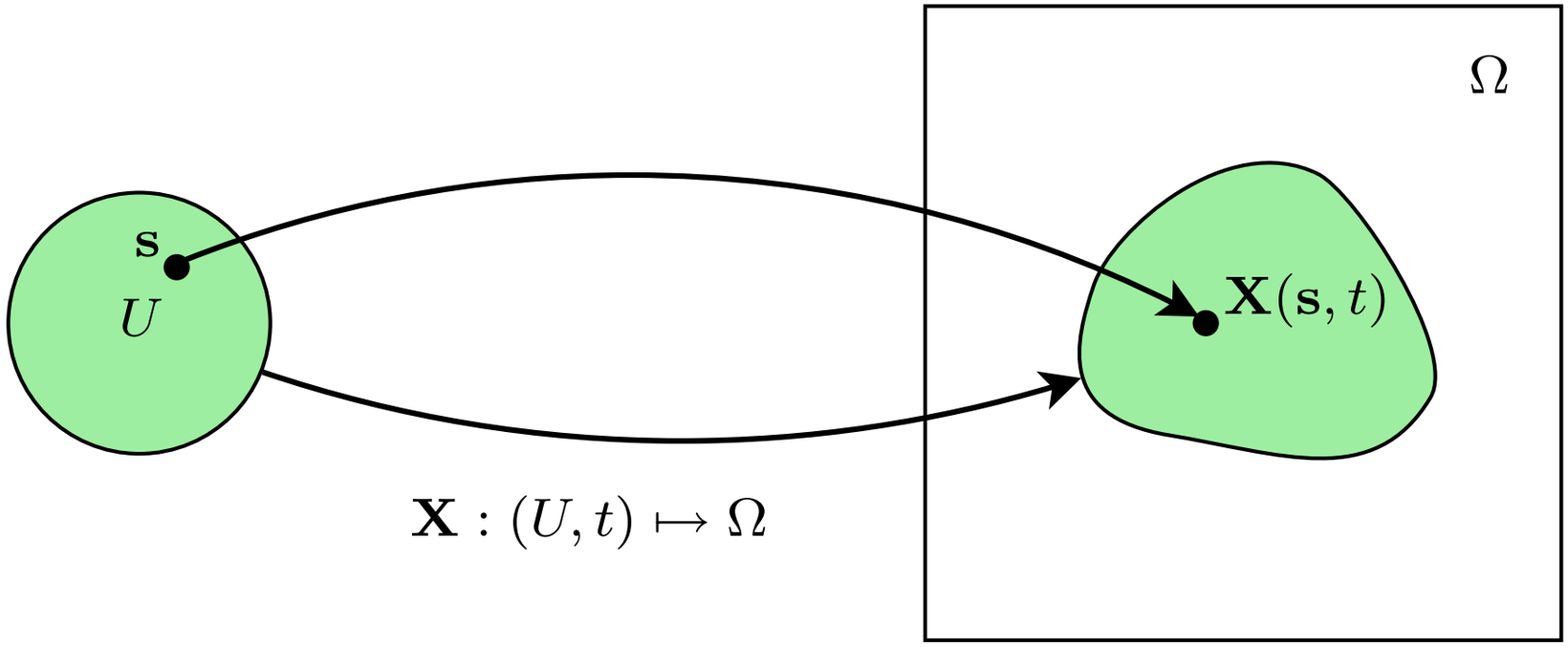
$$\boldsymbol{\sigma}(\mathbf{x},t) = \begin{cases} \boldsymbol{\sigma}^{\mathrm{f}}(\mathbf{x},t) + \boldsymbol{\sigma}^{\mathrm{s}}(\mathbf{x},t) & \text{for } \mathbf{x} \in \mathbf{X}(U,t), \\ \boldsymbol{\sigma}^{\mathrm{f}}(\mathbf{x},t) & \text{otherwise,} \end{cases}$$

in which

$$\boldsymbol{\sigma}^{\mathrm{f}} = -p\mathbb{I} + \mu\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right]$$

is the stress tensor of a viscous incompressible fluid, and $\boldsymbol{\sigma}^{\mathrm{s}}(\mathbf{x},t)$ is the stress tensor that describes the elasticity of the structure.

Stress in Eulerian and Lagrangian forms



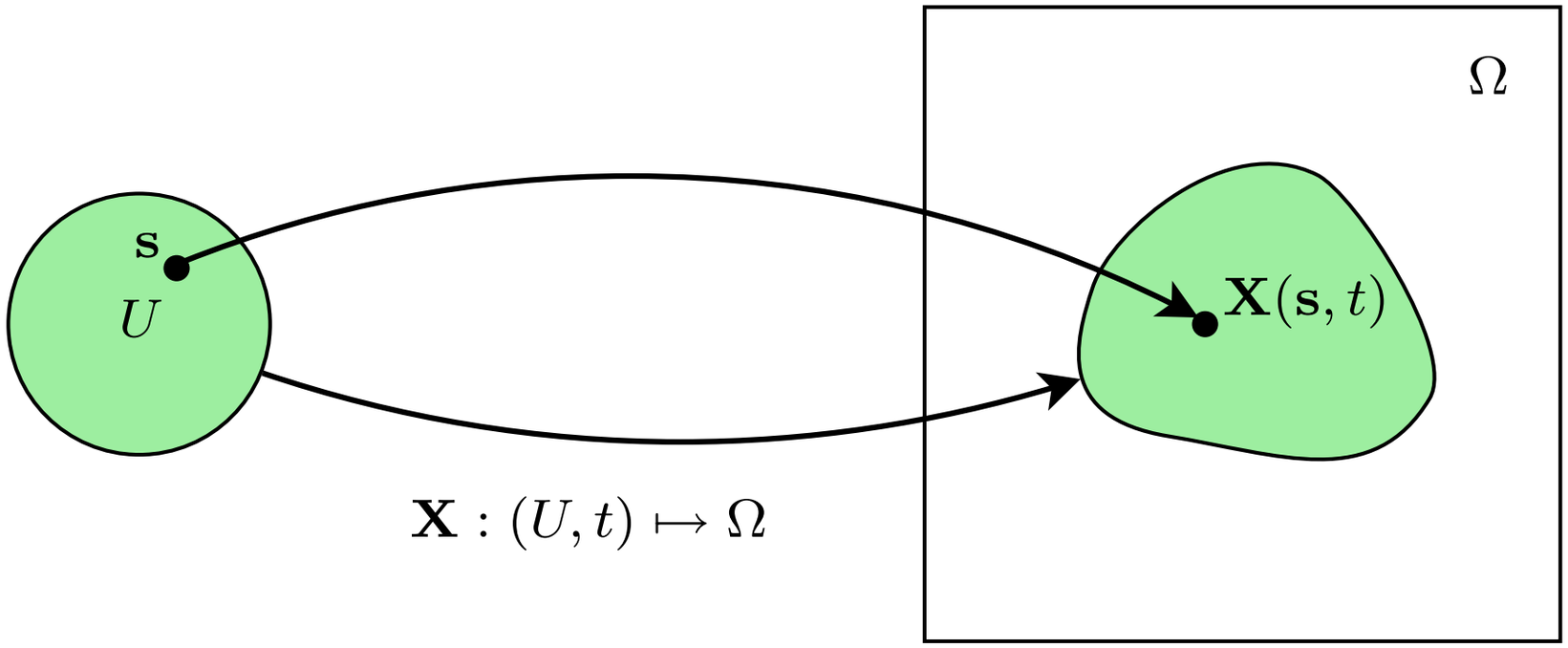
A convenient Lagrangian description of the elastic stress is the first Piola-Kirchhoff elastic stress tensor $\mathbb{P}^{\mathrm{s}}(\mathbf{s}, t)$.

Let $\mathbb{F} = \frac{\partial \mathbf{X}}{\partial \mathbf{s}}$ denote the deformation gradient tensor, and let $J = \det(\mathbb{F})$.

$\mathbb{P}^{\mathrm{s}}$ can be expressed in terms of $\boldsymbol{\sigma}^{\mathrm{s}}$ and $\mathbb{F}$ by:

$$\mathbb{P}^{\mathrm{s}} = J\, \boldsymbol{\sigma}^{\mathrm{s}}\, \mathbb{F}^{-T}.$$

Stress in Eulerian and Lagrangian forms



For a hyperelastic material with strain-energy functional $W = W(\mathbb{F})$,

$$\mathbb{P}^{\mathrm{s}} = \frac{\partial W}{\partial \mathbb{F}}.$$

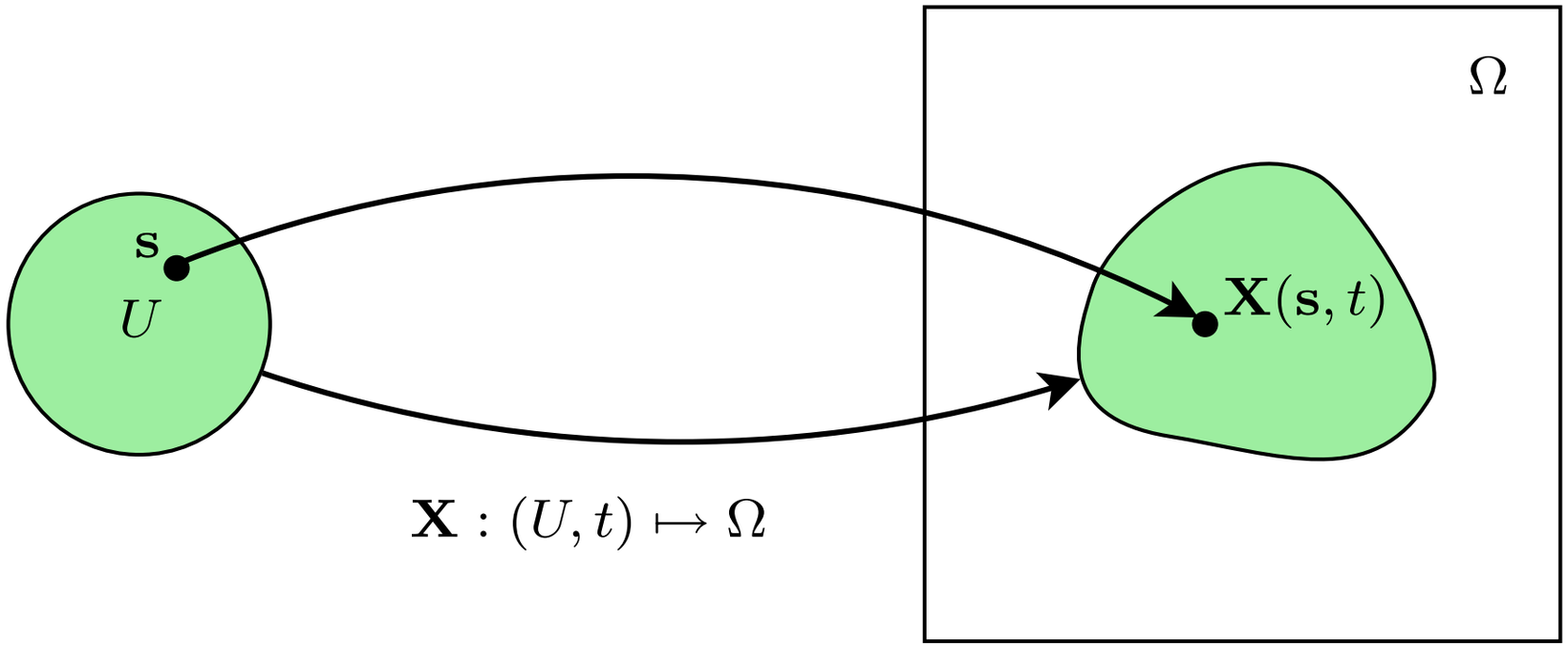
For an incompressible neo-Hookean material, the strain-energy functional is:

$$W(\mathbb{F}) = \frac{C}{2}\left(I_1\left(\mathbb{F}^T\mathbb{F}\right) - d\right) = \frac{C}{2}\left(\mathrm{trace}\left(\mathbb{F}^T\mathbb{F}\right) - d\right) = \frac{C}{2}\left(\mathbb{F} : \mathbb{F} - d\right),$$

for which

$$\mathbb{P}^{\mathrm{s}}(\mathbf{s},t) = \frac{\partial W}{\partial \mathbb{F}}(\mathbf{s},t) = C\mathbb{F} = C\frac{\partial \mathbf{X}}{\partial \mathbf{s}}.$$

Stress in Eulerian and Lagrangian forms



Notice: We handle the incompressibility of the immersed structure in the *Eulerian* fluid-like stress tensor. In the continuous formulation, there is *no need* also to account for incompressibility in Lagrangian form.

When the equations are discretized, however, the computed deformation is generally no longer exactly incompressible. In some cases, it may be useful to adopt a nearly incompressible approach (e.g., by adding some penalty terms to the strain-energy functional).

The equations of motion for the fluid-structure system

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}(\mathbf{x},t)+\mathbf{u}(\mathbf{x},t)\cdot\nabla\mathbf{u}(\mathbf{x},t)\right)=-\nabla p(\mathbf{x},t)+\mu\nabla^2\mathbf{u}(\mathbf{x},t)+\mathbf{f}(\mathbf{x},t)+\mathbf{t}(\mathbf{x},t),$$

$$\nabla\cdot\mathbf{u}(\mathbf{x},t)=0,$$

$$\mathbf{f}(\mathbf{x},t)=\int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s},$$

$$\mathbf{F}(\mathbf{s},t)=\nabla_{\mathbf{s}}\cdot\mathbb{P}^{\mathrm{s}}(\mathbf{s},t),$$

$$\mathbf{t}(\mathbf{x},t)=\int_{\partial U} \mathbf{T}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}A(\mathbf{s}),$$

$$\mathbf{T}(\mathbf{s},t)=-\mathbb{P}^{\mathrm{s}}(\mathbf{s},t)\,\mathbf{N}(\mathbf{s}),$$

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t)=\int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}.$$

The total force generated by the structure is comprised of two terms:

- an *internal force density* $\mathbf{F}(\mathbf{s},t)=\nabla_{\mathbf{s}}\cdot\mathbb{P}^{\mathrm{s}}(\mathbf{s},t)$, which is a volumetric force density distributed in the *interior* of the structure; and
- a *transmission force density* $\mathbf{T}(\mathbf{s},t)=-\mathbb{P}^{\mathrm{s}}(\mathbf{s},t)\,\mathbf{N}(\mathbf{s})$, which is a surface force density concentrated on the *boundary* of the structure.

Weak form of the equations of motion

To facilitate discretization of the Lagrangian equations via C^0 finite elements, we introduce the weak formulation:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f} + \mathbf{t},$$

$$\nabla\cdot\mathbf{u} = 0,$$

$$\mathbf{f}(\mathbf{x},t) = \int_U \mathbf{F}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s},$$

$$\int_U \mathbf{F}(\mathbf{s},t)\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} = -\int_U \mathbb{P}^{\mathrm{s}}(\mathbf{s},t) : \nabla_{\mathbf{s}}\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} + \int_{\partial U} \mathbb{P}^{\mathrm{s}}(\mathbf{s},t)\,\mathbf{N}(\mathbf{s})\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}A(\mathbf{s}), \quad \forall\mathbf{v}(\mathbf{x}),$$

$$\mathbf{t}(\mathbf{x},t) = \int_{\partial U} \mathbf{T}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}A(\mathbf{s}),$$

$$\mathbf{T} = -\mathbb{P}^{\mathrm{s}}\,\mathbf{N},$$

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}.$$

Weak form of the equations of motion

We can also adopt a weak formulation that eliminates the transmission force from the equations of motion:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{g},$$

$$\nabla\cdot\mathbf{u} = 0,$$

$$\mathbf{g}(\mathbf{x},t) = \int_U \mathbf{g}(\mathbf{s},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s},$$

$$\int_U \mathbf{G}(\mathbf{s},t)\cdot\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s} = -\int_U \mathbb{P}^{\mathrm{s}}(\mathbf{s},t) : \nabla_{\mathbf{s}}\mathbf{v}(\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}, \qquad \forall\mathbf{v}(\mathbf{x}),$$

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\,\delta(\mathbf{x}-\mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{x}.$$

Remark: The two weak formulations are equivalent in the continuous setting, but *not* when discretized. In practice, the "partitioned" formulation, in which the interior and transmission forces are treated separately, seems to yield higher accuracy, especially in the case in which the Lagrangian mesh is significantly coarser than the background Eulerian grid.

Finite element discretization

Let $\mathcal{T}_h$ be a triangulation of U, with elements U^e ($\mathcal{T}_h = \cup_e U^e$), nodes $\{\mathbf{s}_l\}$, and nodal basis functions $\{\phi_l(\mathbf{s})\}$.

Standard nodal FE basis functions are interpolatory:

$$\sum_l \phi_l(\mathbf{s}) = 1, \text{ and}$$

$$\phi_l(\mathbf{s}_m) = \begin{cases} 1, & \text{if } l = m, \\ 0, & \text{otherwise.} \end{cases}$$

Typically we use Q^1 or Q^2 basis functions.

We approximate the deformation in terms of the time-dependent nodal positions $\{\mathbf{X}_l(t)\}$ via

$$\mathbf{X}(\mathbf{s}, t) = \sum_l \mathbf{X}_l(t)\, \phi_l(\mathbf{s}),$$

and we approximate the internal force density in terms of the time-dependent nodal forces $\{\mathbf{F}_l(t)\}$ via

$$\mathbf{F}(\mathbf{s}, t) = \sum_l \mathbf{F}_l(t)\, \phi_l(\mathbf{s}).$$

Finite element discretization

We compute $\mathbf{F}(\mathbf{s}, t)$ in a standard way, e.g., by requiring

$$\int_U \mathbf{F}(\mathbf{s}, t)\, \phi_m(\mathbf{s})\, \mathrm{d}\mathbf{s} = -\int_U \mathbb{P}^{\mathrm{s}}(\mathbf{s}, t)\, \nabla_{\mathbf{s}} \phi_m(\mathbf{s})\, \mathrm{d}\mathbf{s} + \int_{\partial U} \mathbb{P}^{\mathrm{s}}(\mathbf{s}, t)\, \mathbf{N}(\mathbf{s})\, \phi_m(\mathbf{s}, t)\, \mathrm{d}A(\mathbf{s})$$

to hold for each basis function $\phi_m(\mathbf{s})$.

Plugging in $\mathbf{F}(\mathbf{s}, t) = \sum_l \mathbf{F}_l(t)\, \phi_l(\mathbf{s})$ leads to a linear system of equations for the nodal forces $\{\mathbf{F}_l(t)\}$,

$$\mathcal{M}\mathbf{F} = \mathbf{b},$$

in which $\mathcal{M}_{l,m} = \left(\int_U \phi_l(\mathbf{s})\, \phi_m(\mathbf{s})\, \mathrm{d}\mathbf{s}\right)$ is the *mass matrix* and $\mathbf{b}$ is a right-hand-side vector that depends on $\mathbb{P}^{\mathrm{s}} = \mathbb{P}^{\mathrm{s}}(\mathbb{F}) = \mathbb{P}^{\mathrm{s}}(\frac{\partial \mathbf{X}}{\partial \mathbf{s}})$.

In practice, we may replace $\mathcal{M}$ with a diagonal (lumped) mass matrix.

Lagrangian-Eulerian interaction

We wish to compute:

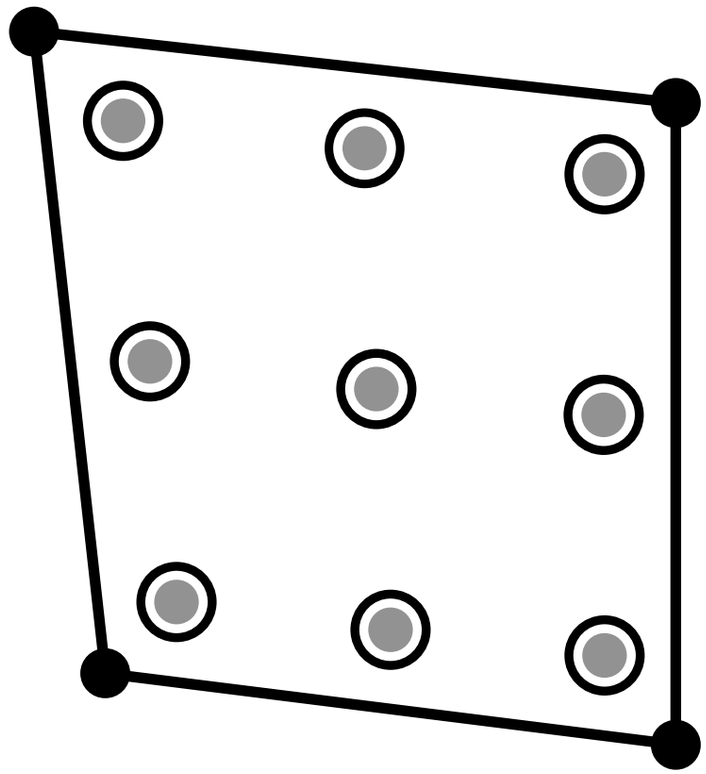
$$\mathbf{f}(\mathbf{x}_{i,j,k}, t) = \int_U \mathbf{F}(\mathbf{s}, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}.$$

Letting U^e denote the elements of the triangulation $\mathcal{T}_h$ of the Lagrangian domain U, we have:

$$\mathbf{f}(\mathbf{x}_{i,j,k}, t) = \sum_{U^e \in \mathcal{T}_h} \int_{U^e} \mathbf{F}(\mathbf{s}, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}.$$

We use Gaussian quadrature to approximate each of these element integrals.

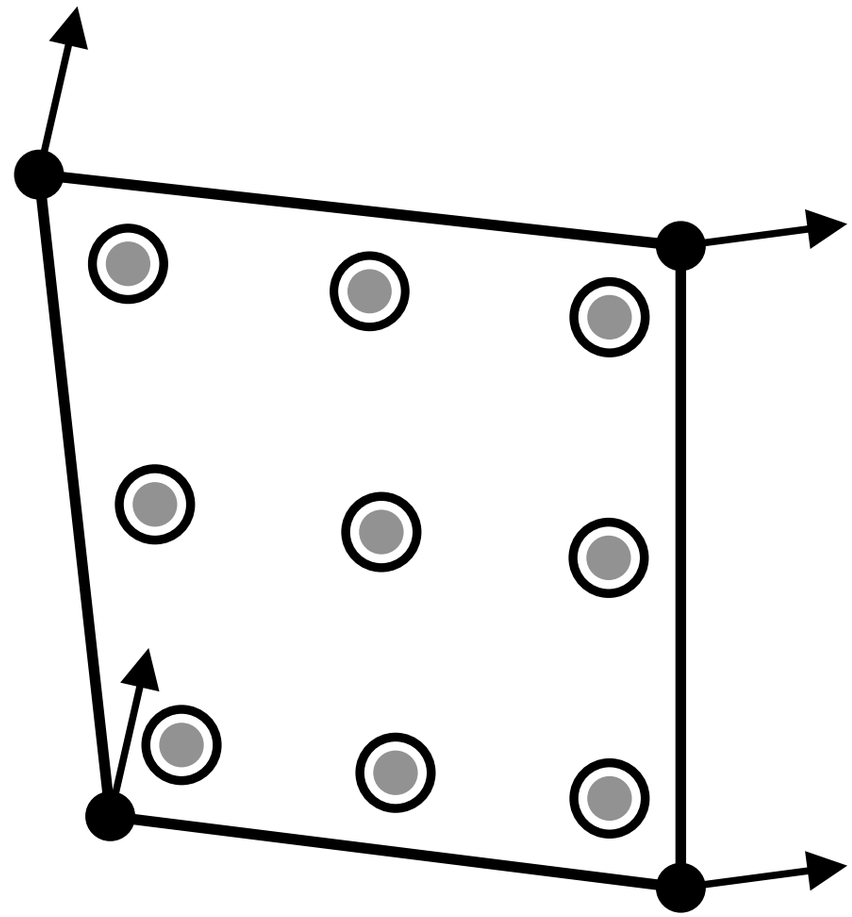
Lagrangian-Eulerian interaction



$$\mathbf{f}(\mathbf{x}_{i,j,k}, t) = \sum_{U^e \in \mathcal{T}_h} \int_{U^e} \mathbf{F}(\mathbf{s}, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}$$

$$\approx \sum_{U^e} \sum_{\mathbf{s}_Q^e \in U^e} \mathbf{F}(\mathbf{s}_Q^e, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}_Q^e, t))\, \omega_Q^e$$

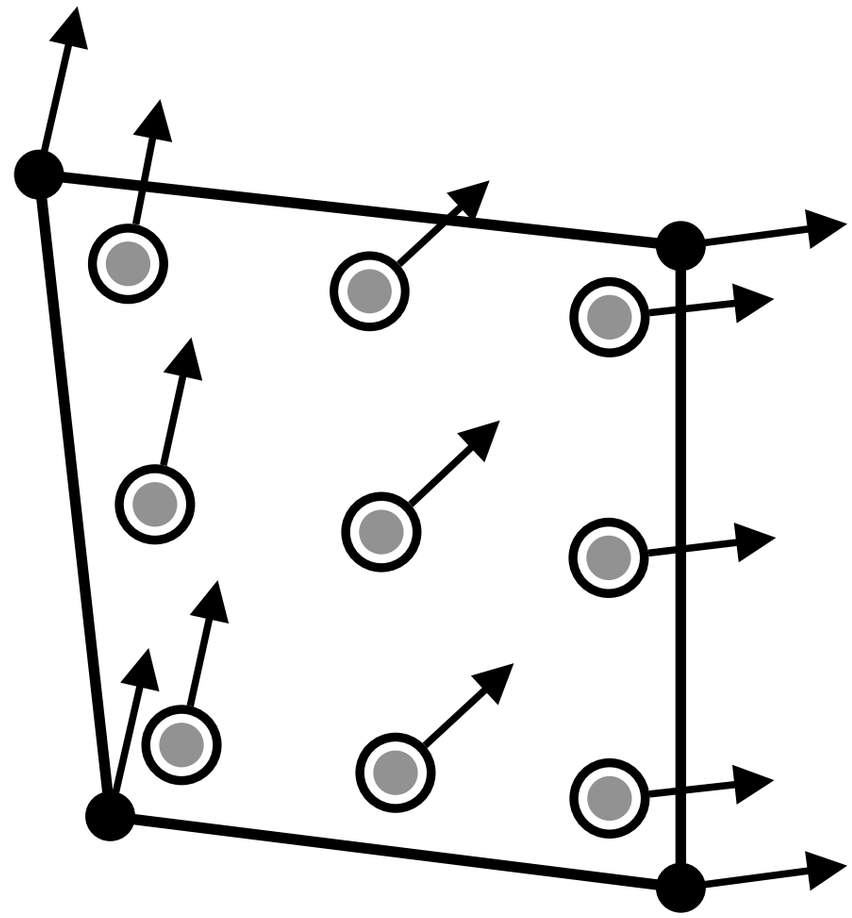
Lagrangian-Eulerian interaction



$$\mathbf{f}(\mathbf{x}_{i,j,k}, t) = \sum_{U^e \in \mathcal{T}_h} \int_{U^e} \mathbf{F}(\mathbf{s}, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}$$

$$\approx \sum_{U^e} \sum_{\mathbf{s}_Q^e \in U^e} \mathbf{F}(\mathbf{s}_Q^e, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}_Q^e, t))\, \omega_Q^e$$

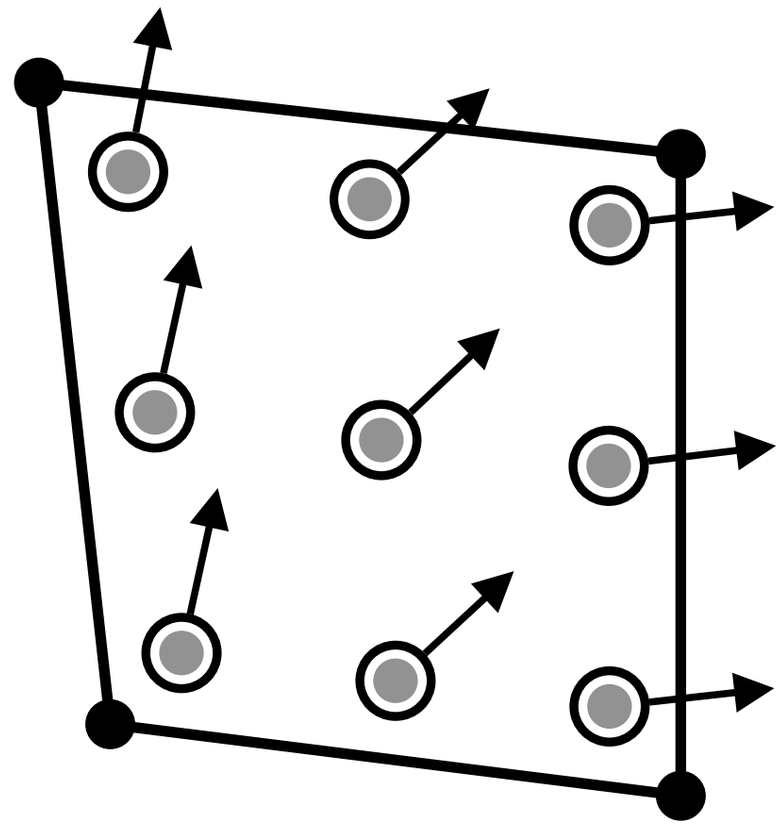
Lagrangian-Eulerian interaction



$$\mathbf{f}(\mathbf{x}_{i,j,k}, t) = \sum_{U^e \in \mathcal{T}_h} \int_{U^e} \mathbf{F}(\mathbf{s}, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}$$

$$\approx \sum_{U^e} \sum_{\mathbf{s}_Q^e \in U^e} \mathbf{F}(\mathbf{s}_Q^e, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}_Q^e, t))\, \omega_Q^e$$

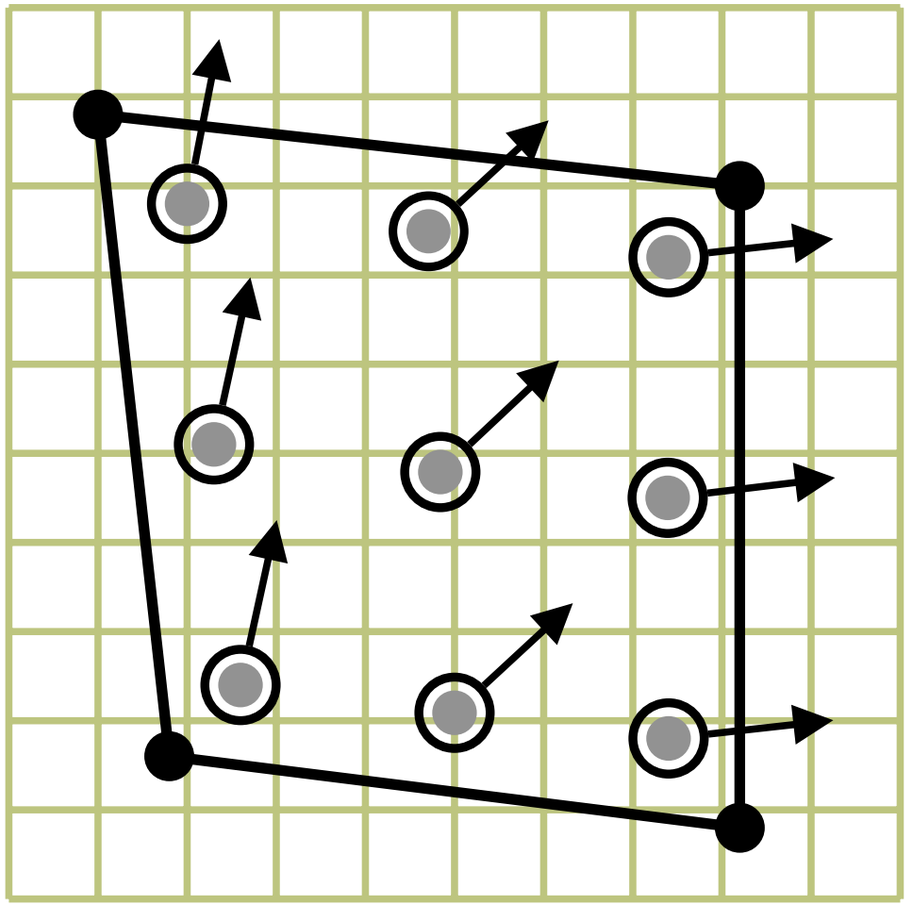
Lagrangian-Eulerian interaction



$$\mathbf{f}(\mathbf{x}_{i,j,k}, t) = \sum_{U^e \in \mathcal{T}_h} \int_{U^e} \mathbf{F}(\mathbf{s}, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}$$

$$\approx \sum_{U^e} \sum_{\mathbf{s}_Q^e \in U^e} \mathbf{F}(\mathbf{s}_Q^e, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}_Q^e, t))\, \omega_Q^e$$

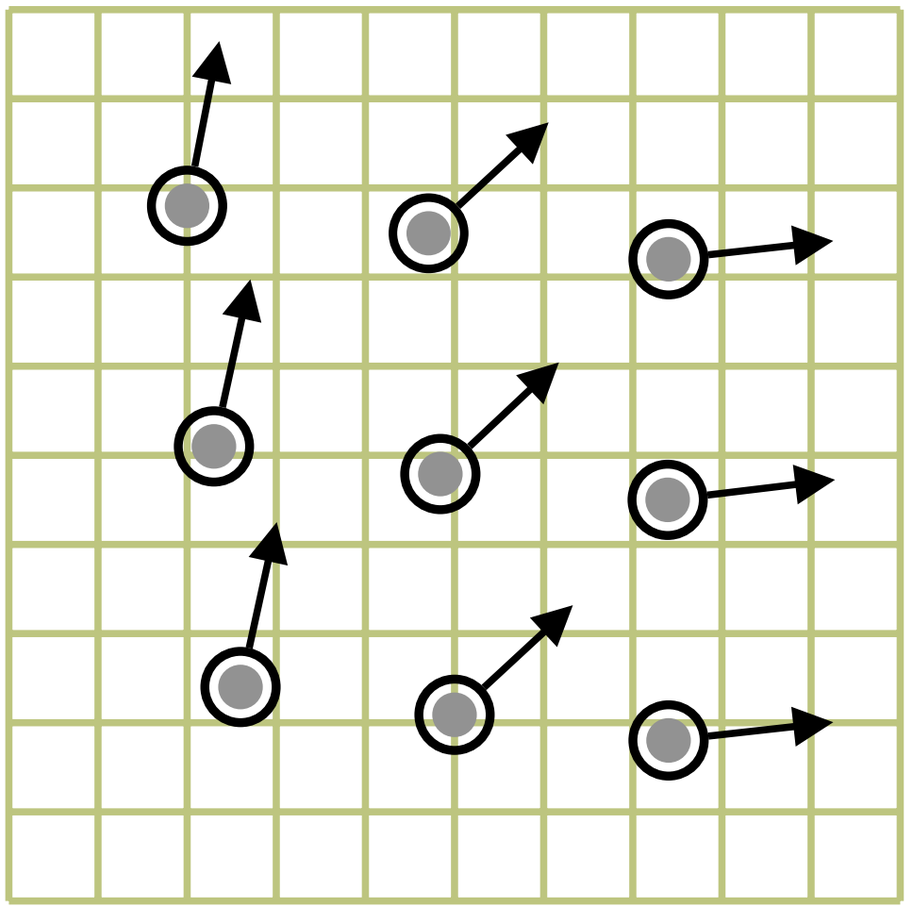
Lagrangian-Eulerian interaction



$$\mathbf{f}(\mathbf{x}_{i,j,k}, t) = \sum_{U^e \in \mathcal{T}_h} \int_{U^e} \mathbf{F}(\mathbf{s}, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}, t))\, \mathrm{d}\mathbf{s}$$

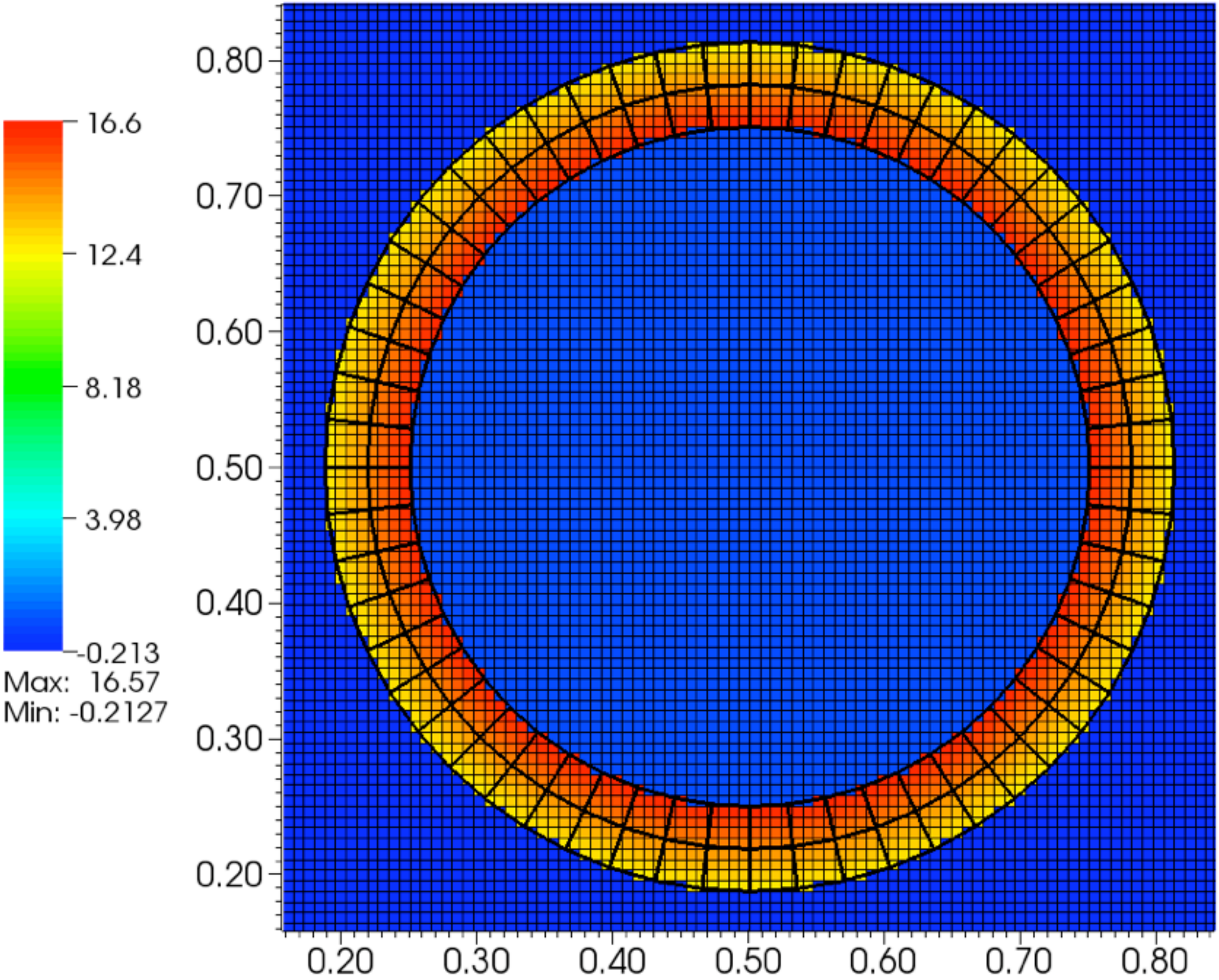
$$\approx \sum_{U^e} \sum_{\mathbf{s}_Q^e \in U^e} \mathbf{F}(\mathbf{s}_Q^e, t)\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}_Q^e, t))\, \omega_Q^e$$

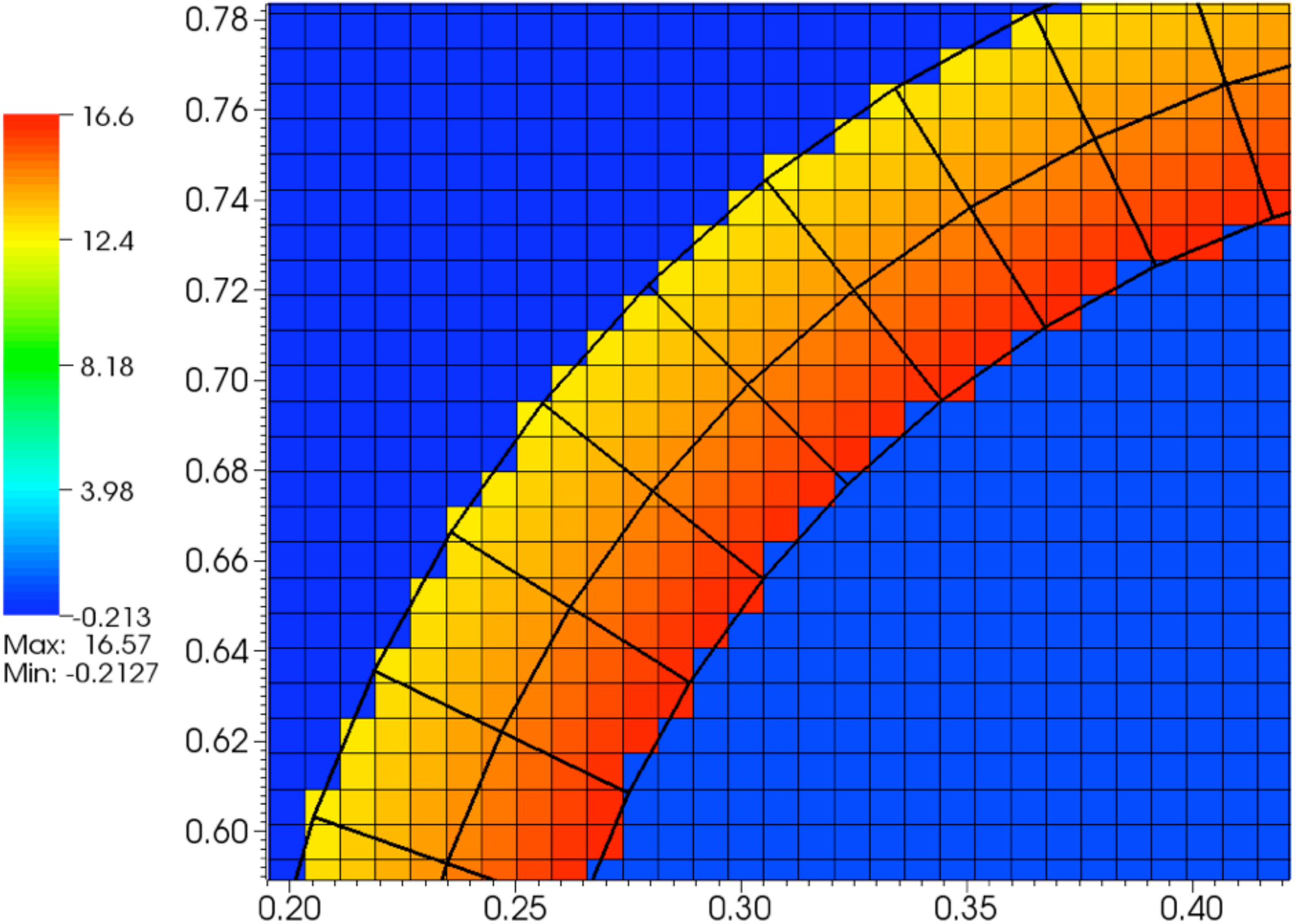
Lagrangian-Eulerian interaction



$$\mathbf{f}(\mathbf{x}_{i,j,k},t) = \sum_{U^e \in \mathcal{T}_h} \int_{U^e} \mathbf{F}(\mathbf{s},t)\,\delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s},t))\,\mathrm{d}\mathbf{s}$$

$$\approx \sum_{U^e} \sum_{\mathbf{s}_Q^e \in U^e} \mathbf{F}(\mathbf{s}_Q^e,t)\,\delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s}_Q^e,t))\,\omega_Q^e$$


16.6
12.4
8.18
3.98
-0.213
Max: 16.57
Min: -0.2127
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.20
0.30
0.40
0.50
0.60
0.70
0.80


16.6
12.4
8.18
3.98
-0.213
Max: 16.57
Min: -0.2127
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
0.60
0.20
0.25
0.30
0.35
0.40

16.6
12.4
8.18
3.98
-0.213
Max: 16.57
Min: -0.2127
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90

Velocity restriction with the IB/FE method

We need to construct a *velocity-restriction operator* $\mathcal{R}$ to determine the deformation of the structure from the Eulerian velocity field:

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathcal{R}\,\mathbf{u}.$$

We remark that, if $\mathbf{T} \equiv 0$ and if

$$\left(\mathbf{F}, \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t}\right)_{\mathbf{s}} = (\mathbf{f}, \mathbf{u})_{\mathbf{x}}\,,$$

then the IB/FE method conserves energy during Lagrangian-Eulerian interaction. (It is also possible to incorporate $\mathbf{T}$ into $\mathbf{F}$ in a manner that also yields energy conservation if $\mathbf{T} \not\equiv 0$.)

We are thus motivated to construct $\mathcal{R}$ so that it is the adjoint of the force-spreading operator $\mathcal{S}$:

$$\left(\mathbf{F}, \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t}\right)_{\mathbf{s}} = (\mathbf{f}, \mathbf{u})_{\mathbf{x}} \iff (\mathbf{F}, \mathcal{R}\,\mathbf{u})_{\mathbf{s}} = (\mathcal{S}\,\mathbf{F}, \mathbf{u})_{\mathbf{x}} \iff \mathcal{R} = \mathcal{S}^*.$$

Velocity restriction with the IB/FE method

We rewrite the adjoint equation $(\mathbf{F}, \mathcal{R}\,\mathbf{u})_{\mathbf{s}} = (\mathcal{S}\,\mathbf{F}, \mathbf{u})_{\mathbf{x}}$ in matrix form:

$$(\mathbf{F}, \mathcal{R}\,\mathbf{u})_{\mathbf{s}} =: \mathbf{F}^T \mathcal{M}\,\mathcal{R}\,\mathbf{u} = (\mathcal{S}\,\mathbf{F})^T\,\mathbf{u}\,h^3 := (\mathcal{S}\,\mathbf{F}, \mathbf{u})_{\mathbf{x}}\,,$$

in which $\mathcal{M}$ is the mass matrix with entries $\mathcal{M}_{l,m} = \left(\int_U \phi_l(\mathbf{s})\,\phi_m(\mathbf{s})\,\mathrm{d}\mathbf{s}\right)$.

For this to hold for all $\mathbf{F}$ and $\mathbf{u}$, $\mathcal{R}$ must satisfy

$$\mathcal{M}\,\mathcal{R} = \mathcal{S}^T h^3,$$

i.e.,

$$\mathcal{R} = \mathcal{M}^{-1}\,\mathcal{S}^T h^3.$$

The construction of the restriction operator $\mathcal{R}$ is purely algebraic; it may not be clear what $\mathcal{R}$ *actually does*.

Velocity restriction with the IB/FE method

Let the *continuous* function $\mathbf{U}(\mathbf{s},t)$ be defined by the usual IB restriction procedure:

$$\mathbf{U}(\mathbf{s},t) = \sum_{i,j,k} \mathbf{u}_{i,j,k}\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s},t))\, h^3.$$

We wish to determine $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t}(\mathbf{s},t)$ from $\mathbf{U}(\mathbf{s},t)$; however, generally $\mathbf{U}(\mathbf{s},t) \notin \operatorname{span}\{\phi_l(\mathbf{s})\}$.

Although we *could* use $\mathbf{U}(\mathbf{s},t)$ to determine the motion of a discrete set of points in the FE mesh (e.g., the nodes of the mesh), , we *cannot* set $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t}(\mathbf{s},t) = \mathbf{U}(\mathbf{s},t)$ for *all* $\mathbf{s} \in U$, because

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t}(\mathbf{s},t) = \sum_l \frac{\mathrm{d}\mathbf{X}_l}{\mathrm{d}t}(t)\, \phi_l(\mathbf{s}).$$

Velocity restriction with the IB/FE method

If we wish to use

$$\mathbf{U}(\mathbf{s},t) = \sum_{i,j,k} \mathbf{u}_{i,j,k}\, \delta_h(\mathbf{x}_{i,j,k} - \mathbf{X}(\mathbf{s},t))\, h^3.$$

to determine $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t}(\mathbf{s},t)$, one approach would be to *project* $\mathbf{U}(\mathbf{s},t)$ onto $\operatorname{span}\{\phi_l(\mathbf{s})\}$. To do this, we compute $\mathbf{V}(\mathbf{s},t) \in \operatorname{span}\{\phi_l(\mathbf{s})\}$ such that

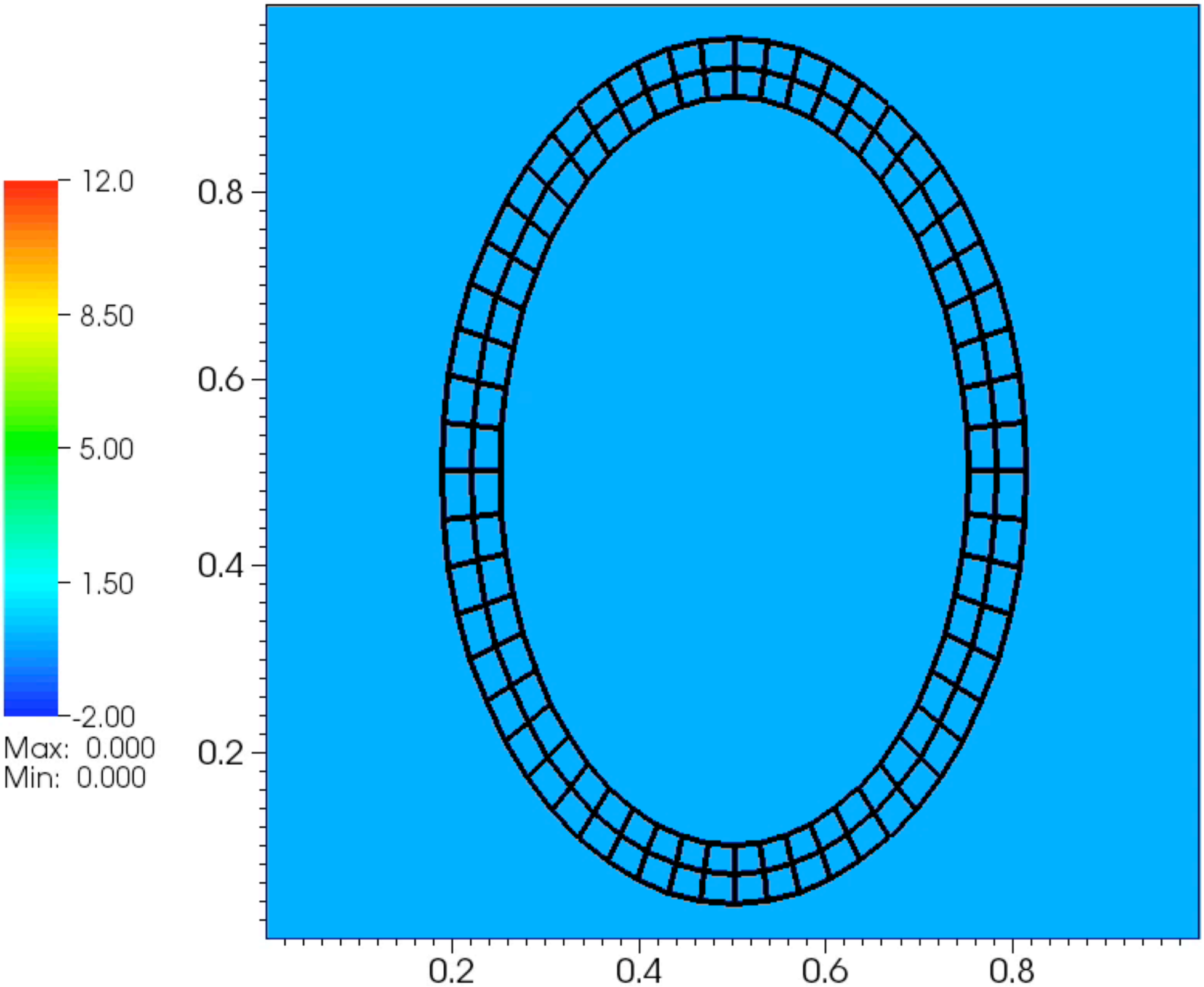
$$\int_U \mathbf{V}(\mathbf{s},t)\, \phi_m(\mathbf{s})\, \mathrm{d}\mathbf{s} = \int_U \mathbf{U}(\mathbf{s},t)\, \phi_m(\mathbf{s})\, \mathrm{d}\mathbf{s}.$$

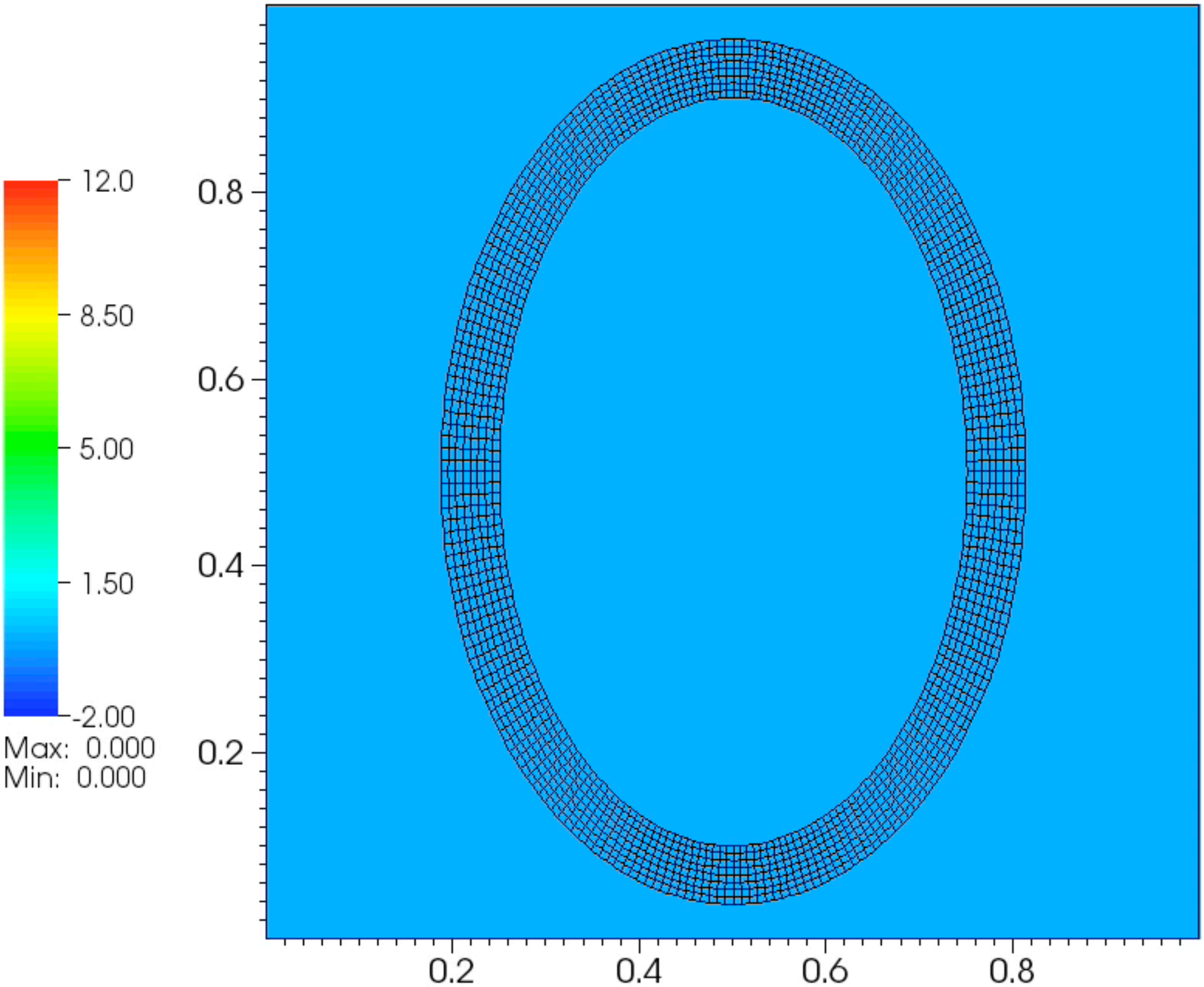
for all m. In matrix form,

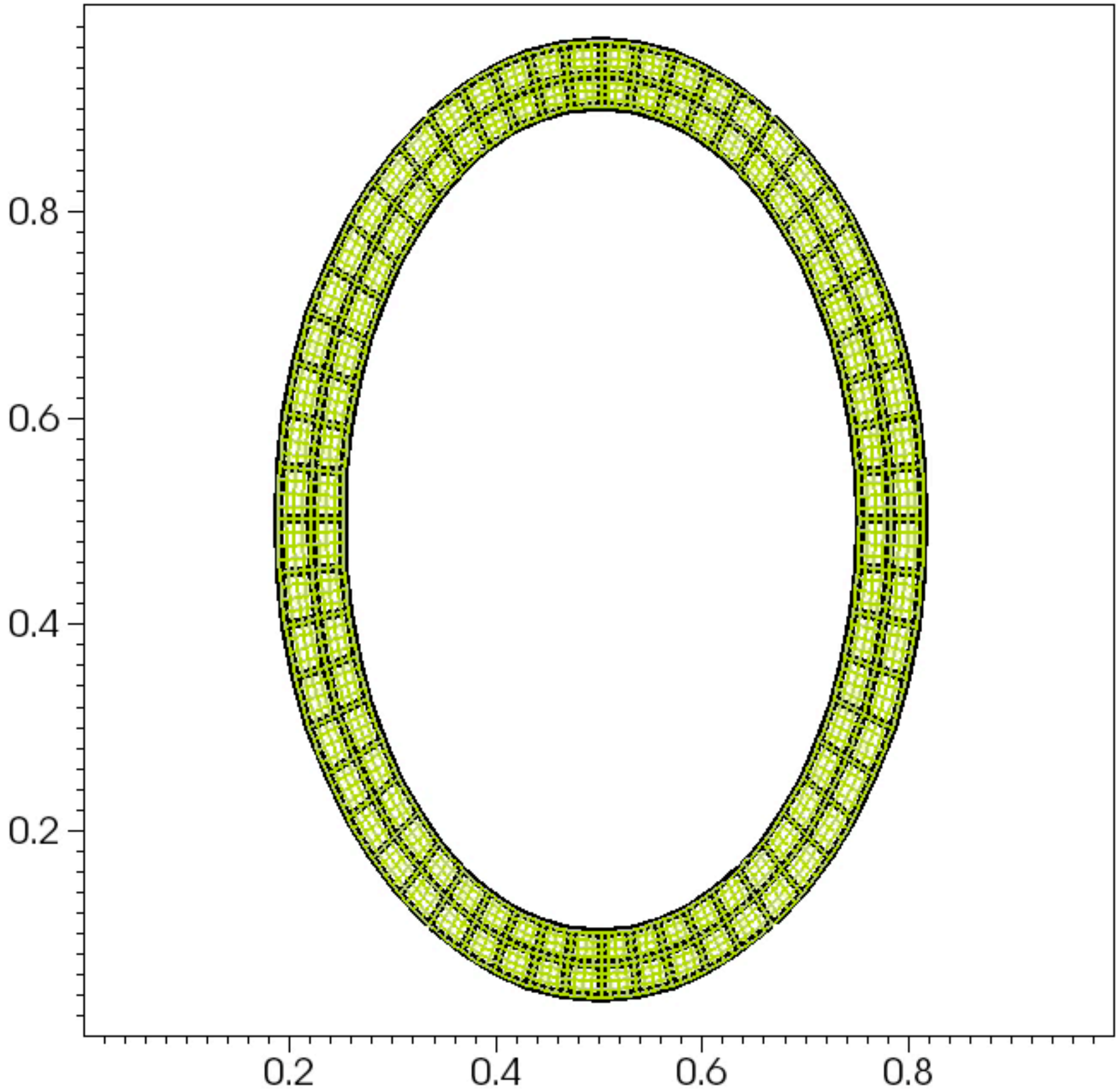
$$\mathbf{V} = \mathcal{M}^{-1}\mathbf{U}.$$

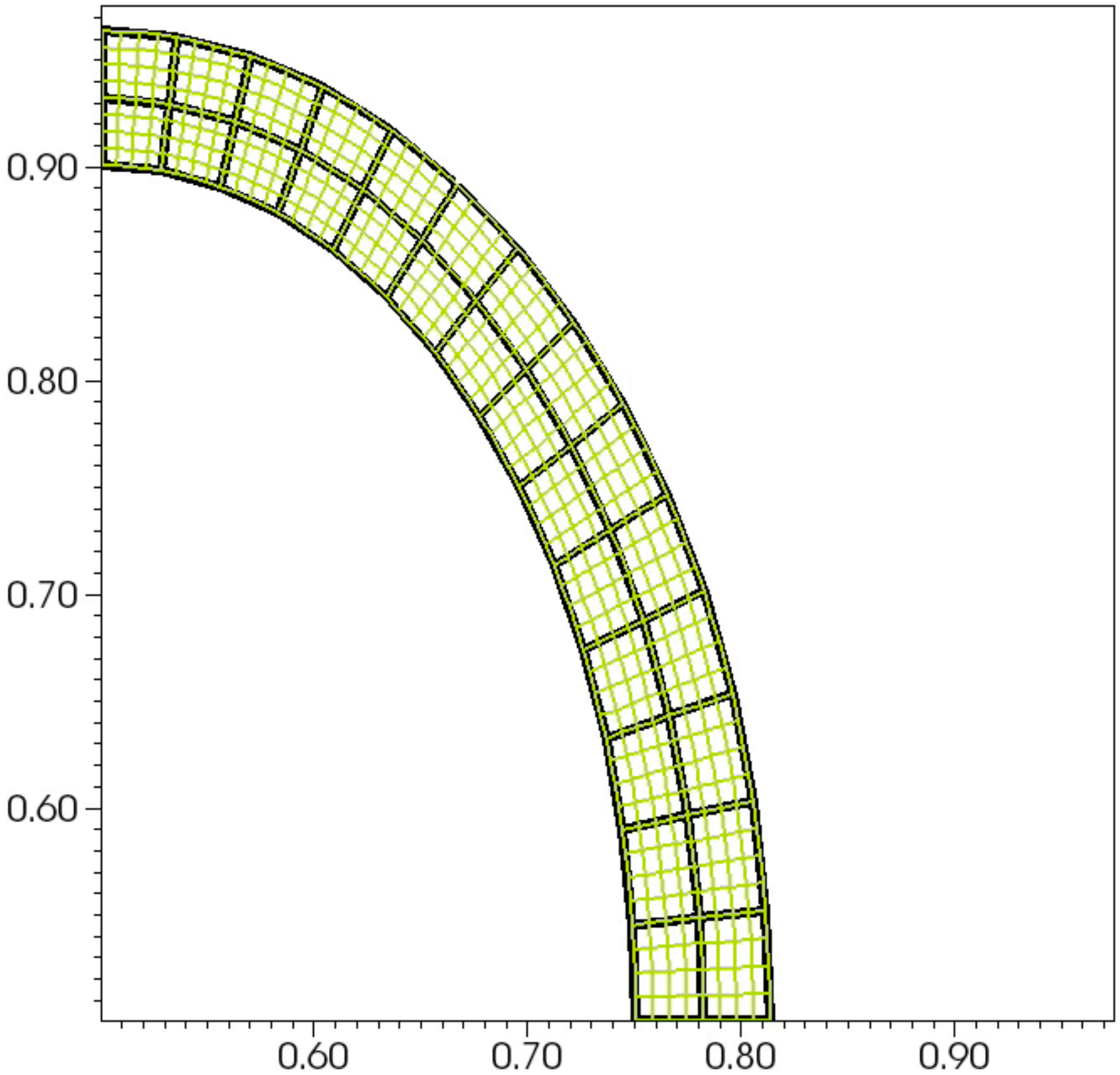
Because $\mathbf{V}(s,t) \in \operatorname{span}\{\phi_l(\mathbf{s})\}$, we *can* use $\mathbf{V}(\mathbf{s},t)$ to determine the deformation for each $\mathbf{s} \in U$.

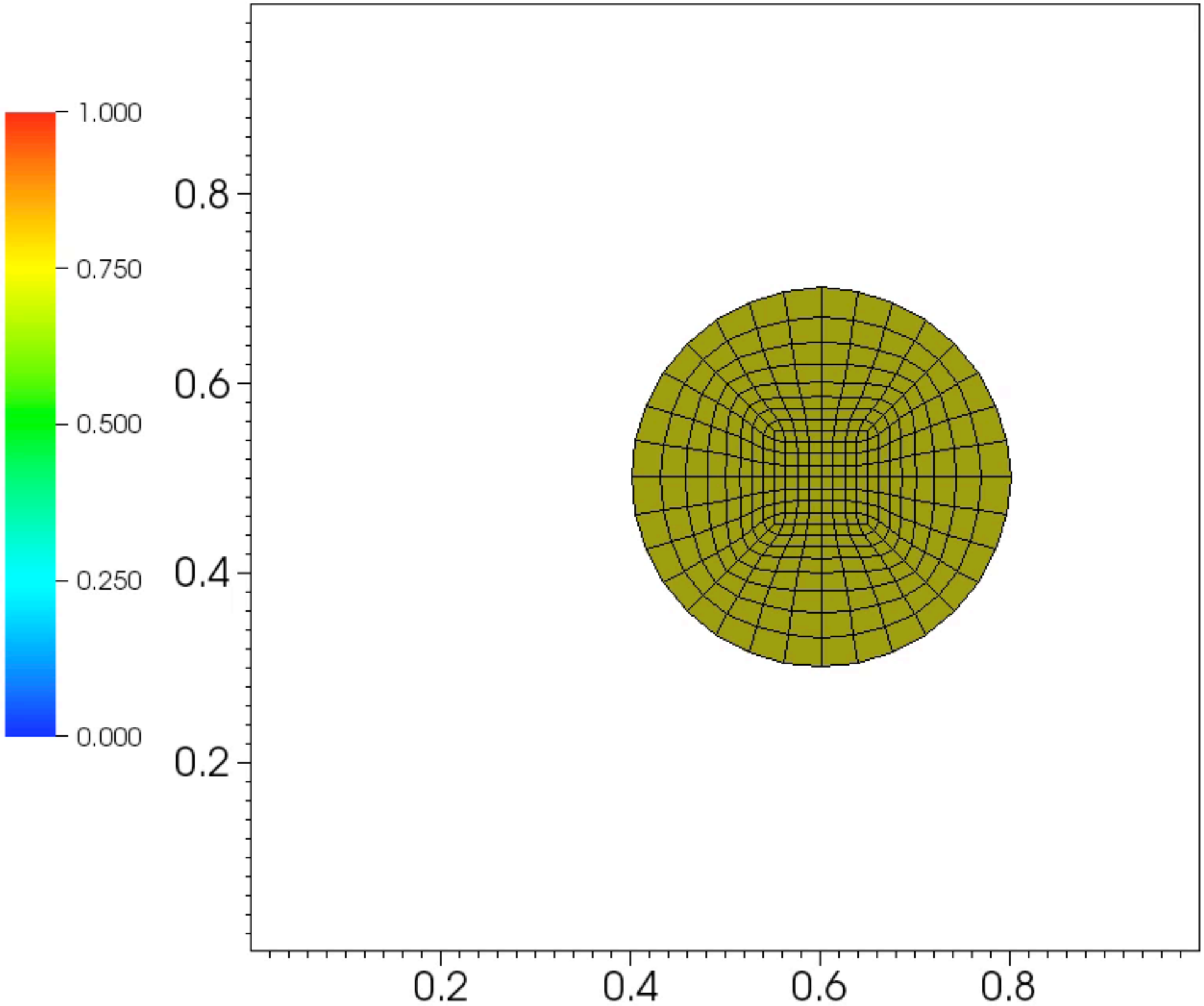
In fact, this is precisely what is done by our (continuous) restriction operator $\mathcal{R}$: $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathcal{R}\mathbf{u}$ is the L^2 projection of $\mathbf{U}(\mathbf{s},t)$ onto the FE basis functions.

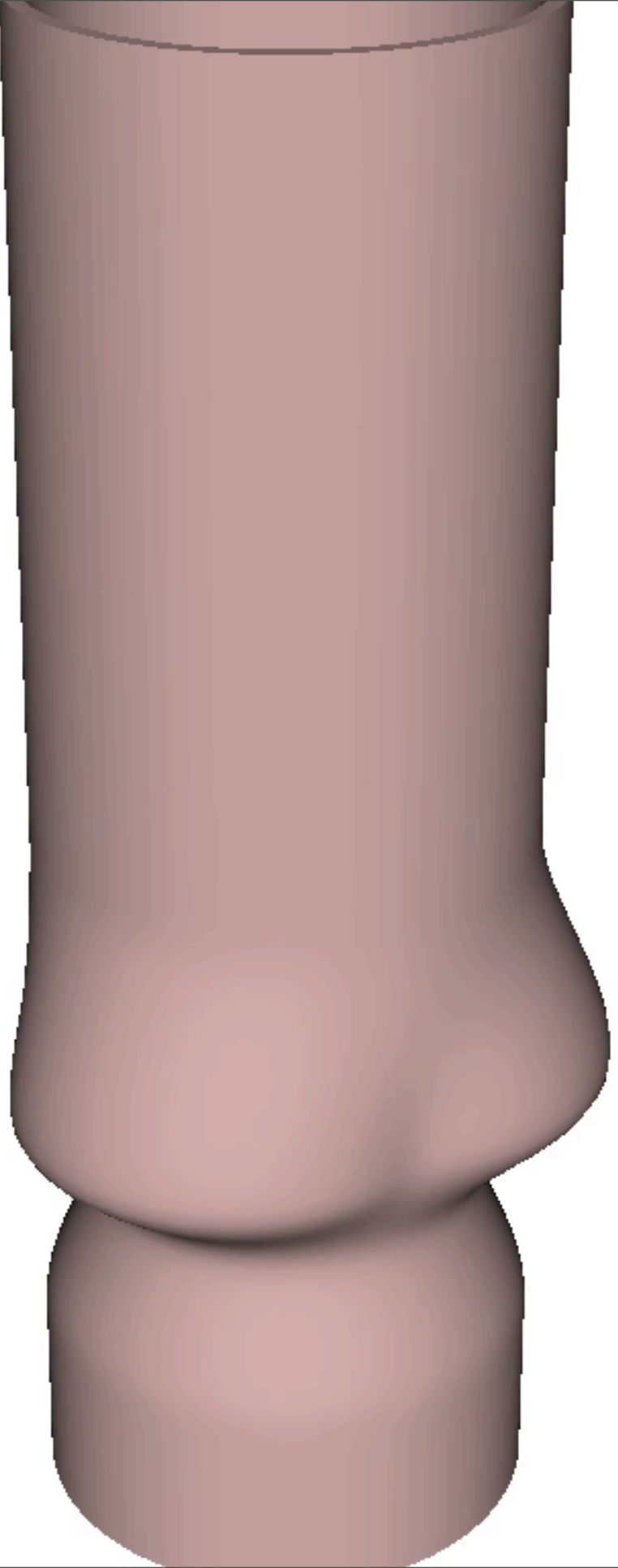

12.0
8.50
5.00
1.50
-2.00
Max: 0.000
Min: 0.000
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8

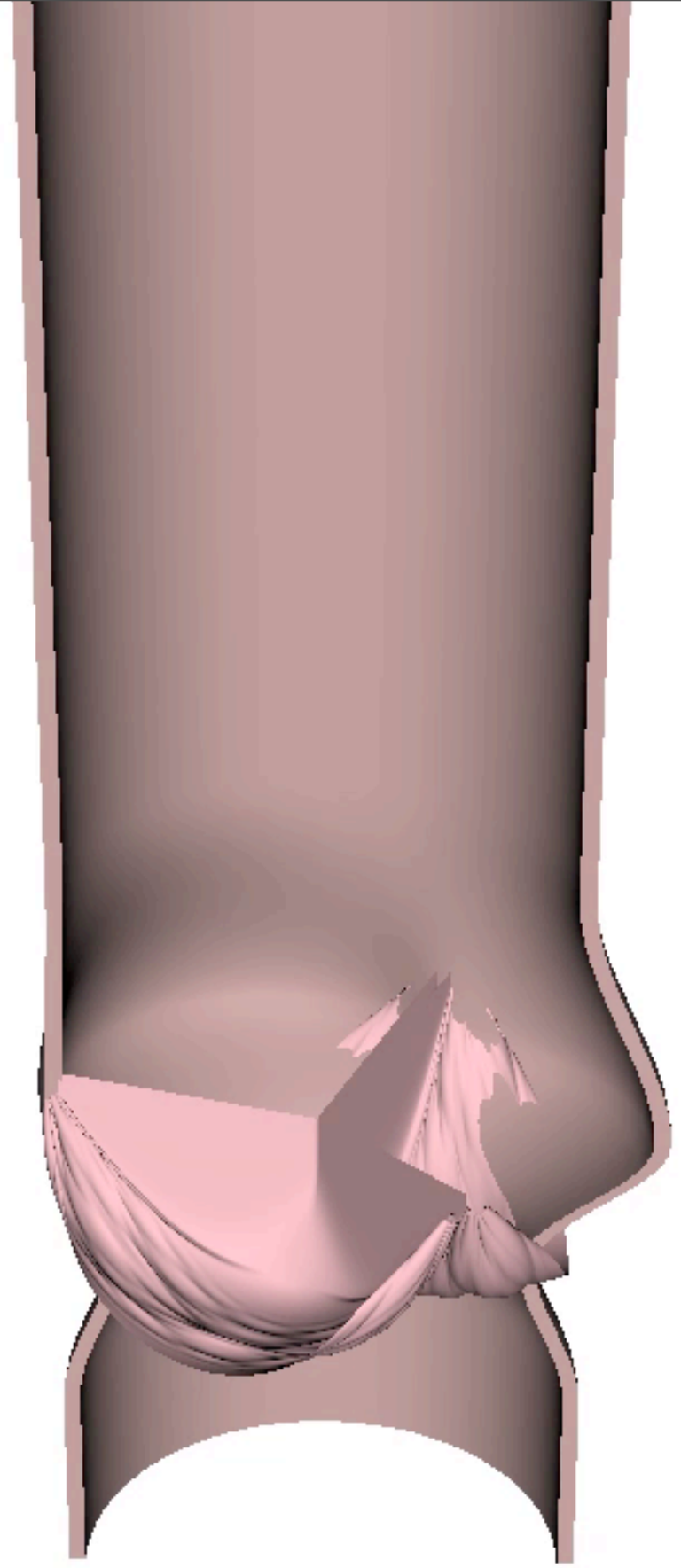

12.0
8.50
5.00
1.50
-2.00
Max: 0.000
Min: 0.000
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8

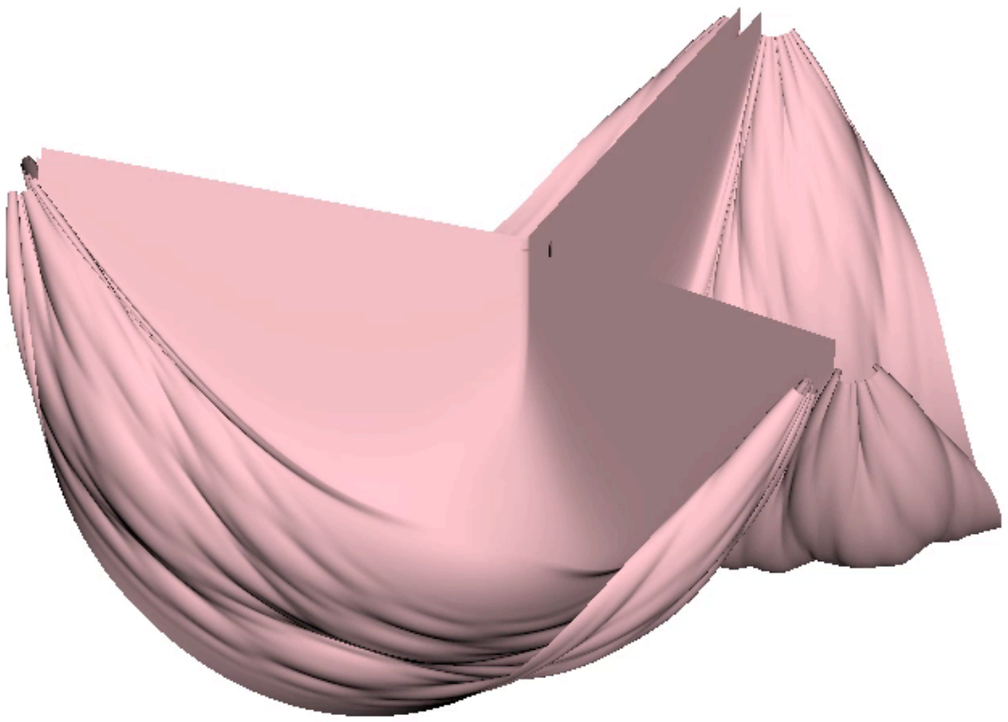

0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8

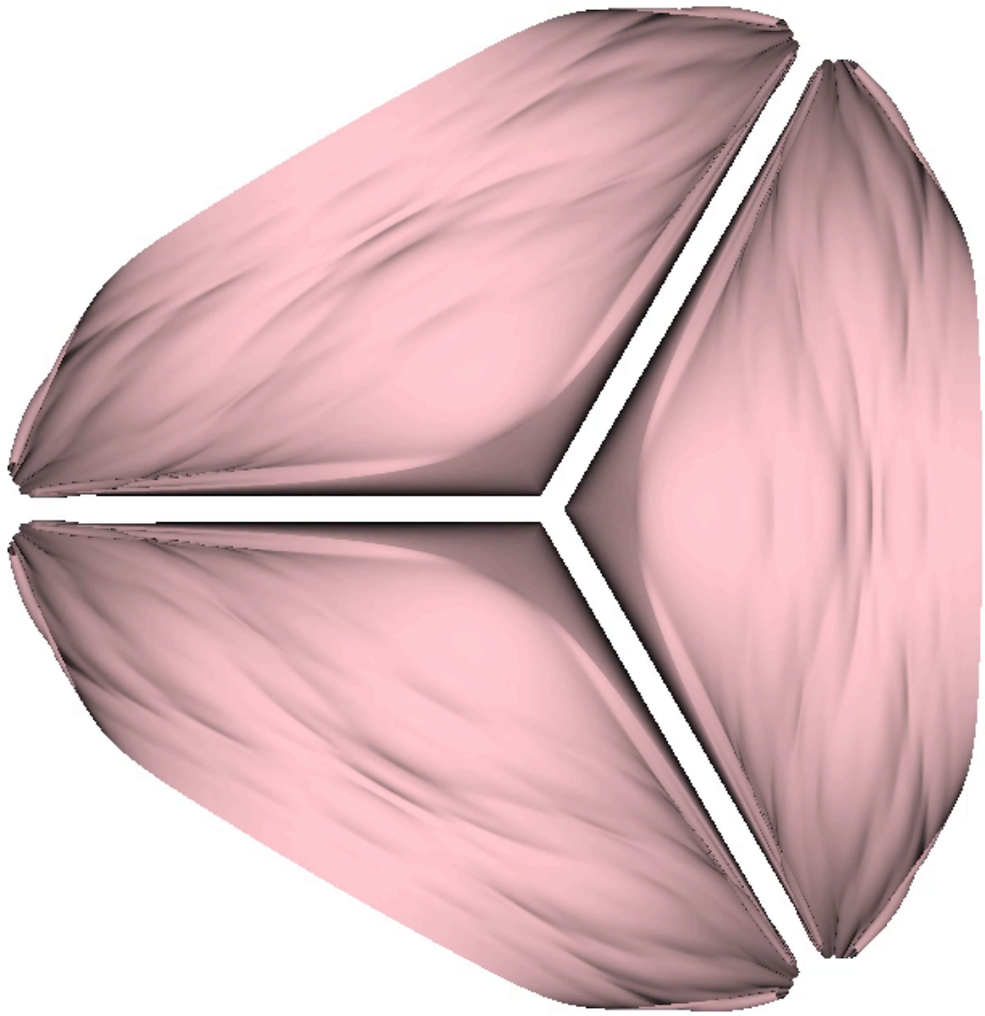

0.90
0.80
0.70
0.60
0.60
0.70
0.80
0.90

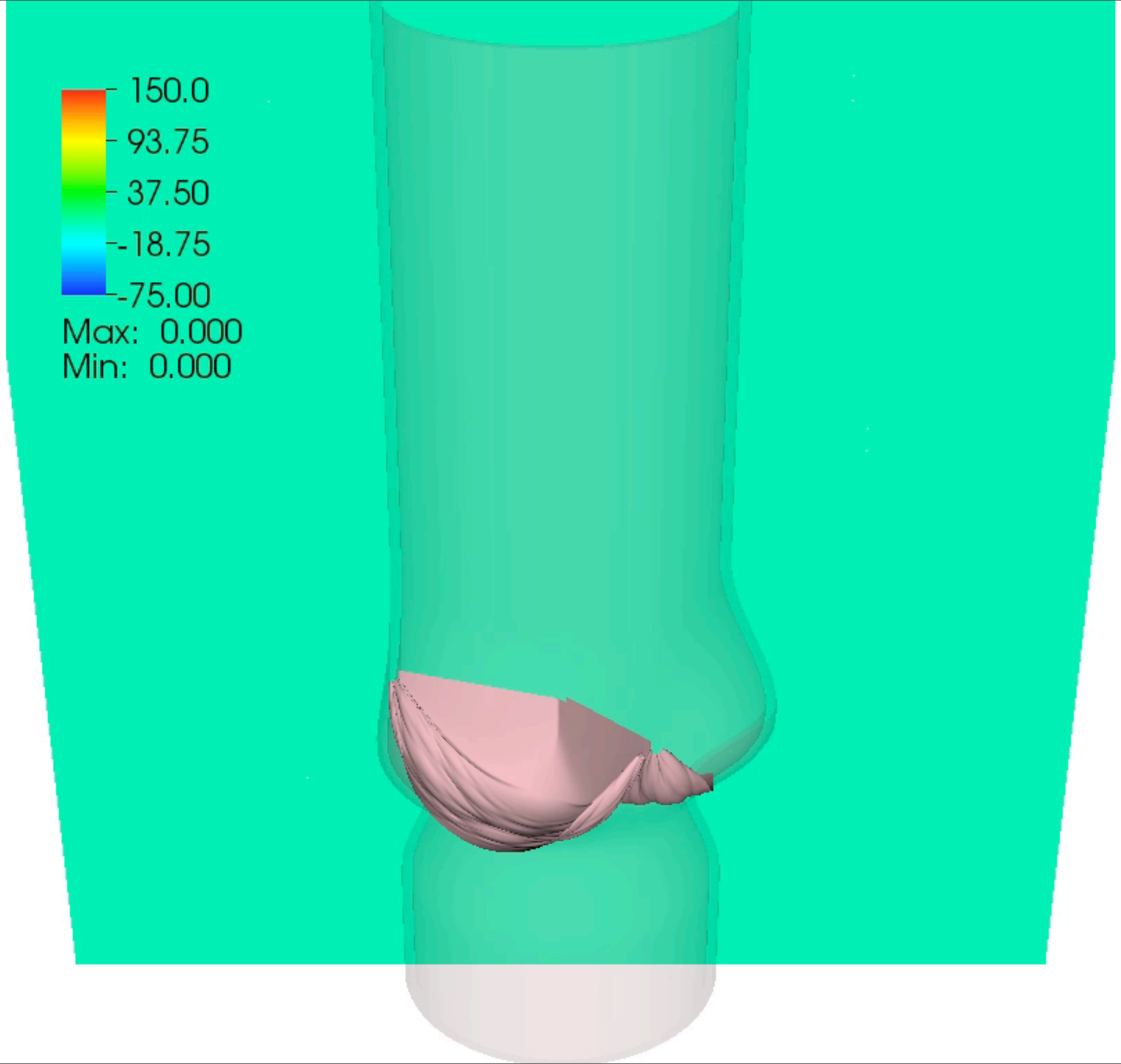

1.000
0.750
0.500
0.250
0.000
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8

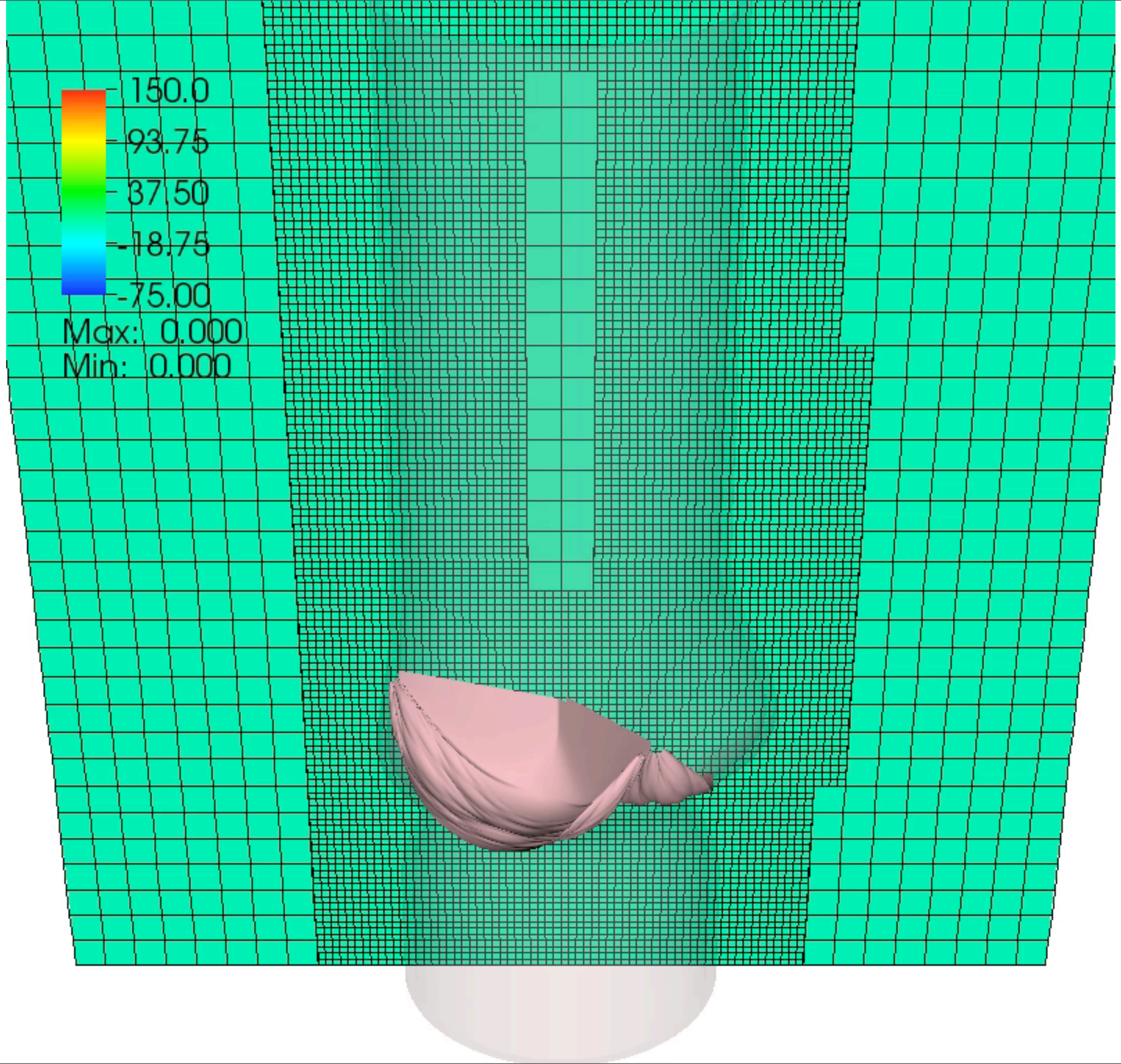









150.0
93.75
37.50
-18.75
-75.00
Max: 0.000
Min: 0.000


150.0
93.75
37.50
-18.75
-75.00
Max: 0.000
Min: 0.000

IBAMR: An adaptive and distributed-memory parallel implementation of the IB method

IBAMR is a C++-based IB software framework that supports distributed-memory parallelism via MPI.

Development supported by an NSF *Software Infrastructure for Sustained Innovation* award.

Built on freely available, high-quality software libraries, including:

- SAMRAI (Lawrence Livermore National Lab)
- PETSc (Argonne National Lab)
- `libMesh` (University of Texas)
- *hypre* (Lawrence Livermore National Lab)
- other utility libraries (Blitz++, HDF5, Silo)

Used at NYU along with Northwestern University, Tulane University, University of Cincinnati, University of Glasgow, University of Montana, University of North Carolina-Chapel Hill, University of Utah, and others.

`http://ibamr.googlecode.com`

References

The immersed boundary method and its extensions:

- C. S. Peskin, The immersed boundary method, *Acta. Numer.*, 11: 479–517 (2002).
- B. E. Griffith and C. S. Peskin, On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems, *J. Comput. Phys.*, 208: 75–105 (2005).
- E. P. Newren, A. L. Fogelson, R. D. Guy, and R. M. Kirby, Unconditionally stable discretizations of the immersed boundary equations, *J. Comput. Phys.*, 222: 702–719 (2007).
- Y. Kim and C. S. Peskin, Penalty immersed boundary method for an elastic boundary with mass, *Phys. Fluid.*, 19: 053103 (18 pages) (2007).
- S. Lim and A. Ferent and X. S. Wang and C. S. Peskin, Dynamics of a closed rod with twist and bend in fluid, *SIAM J. Sci. Comput.*, 31: 273–302 (2008).
- B. E. Griffith and X. Luo, Immersed boundary method with finite element elasticity, submitted.
- B. E. Griffith, On the volume conservation of the immersed boundary method, submitted.

(Preprints and reprints available from `http://www.cims.nyu.edu/~griffith`.)

References

Recent applications of adaptive IB methods to cardiac fluid-structure interaction:

- B. E. Griffith, R. D. Hornung, D. M. McQueen, and C. S. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, *J. Comput. Phys.*, 223: 10–49 (2007).
- B. E. Griffith, X. Luo, D. M. McQueen, and C. S. Peskin, Simulating the fluid dynamics of natural and prosthetic heart valves using the immersed boundary method, *Int. J. Appl. Mech.*, 1: 137–177 (2009).
- B. E. Griffith, Immersed boundary model of aortic heart valve dynamics with physiological driving and loading conditions, *Int. J. Numer. Meth. Biomed. Eng.*, to appear.

(Preprints and reprints available from `http://www.cims.nyu.edu/~griffith`.)

References

Related numerical methods:

- Z.-L. Li and M.-C. Lai, The immersed interface method for the Navier-Stokes equations with singular forces, *J. Comput. Phys.*, 171: 822–842 (2001).
- L. Lee and R. J. LeVeque, An immersed interface method for incompressible Navier-Stokes equations, *SIAM J. Sci. Comput.*, 25: 832–856 (2003).
- L. Zhang, A. Gerstenberger, X. Wang, and W. K. Liu, Immersed finite element method, *Comput. Meth. Appl. Mech. Eng.*, 193: 2051–2067 (2004).
- D. Boffi, L. Gastaldi, L. Heltai, and C. S. Peskin, On the hyper-elastic formulation of the immersed boundary method, *Comput. Meth. Appl. Mech. Engrg.*, 197: 2210–2231 (2008).