Automatic Differentiation in Developing LES Models

Alexey Miroshnikov

ICAM
Department of Mathematics
Virginia Polytechnic Institute & State University

Collaborators

- Paul D. Hovland and Boyana Norris
  - Argonne National Laboratory
- Jeff Borggaard, Traian Iliescu, and Grant Boquet
  - ICAM/Virginia Tech

Automatic Differentiation

Abstractly, computer programs can be regarded as

$$a \equiv (a_1, a_2, \ldots, a_m)$$

$$\downarrow$$

$$z \equiv (z_1, z_2, \ldots, z_p), \quad p >> m + n$$

$$\downarrow$$

$$u \equiv (u_1, u_2, \ldots, u_n)$$

where

$$z_k = f^k_{\text{elem}}(z_i), \qquad i < k,$$

(-, pow(·), sin(·), . . .),

Automatic Differentiation

Abstractly, computer programs can be regarded as

$$a \equiv (a_1, a_2, \ldots, a_m)$$

$$\downarrow$$

$$z \equiv (z_1, z_2, \ldots, z_p), \quad p >> m + n$$

$$\downarrow$$

$$u \equiv (u_1, u_2, \ldots, u_n)$$

or

$$z_k = f^k_{\text{elem}}(z_i, z_j), \qquad i < k, \quad j < k,$$

(+, /, . . .).

Automatic Differentiation

- Forward (Direct) Mode:

$$\frac{dz_k}{da} = \frac{\partial f^k_{\text{elem}}}{\partial z_i}\frac{dz_i}{da} + \frac{\partial f^k_{\text{elem}}}{\partial z_j}\frac{dz_j}{da}.$$

(*computes* all of the $\frac{dz_k}{da}$ terms)

Automatic Differentiation

- Forward (Direct) Mode:

$$\frac{dz_k}{da} = \frac{\partial f^k_{\text{elem}}}{\partial z_i}\frac{dz_i}{da} + \frac{\partial f^k_{\text{elem}}}{\partial z_j}\frac{dz_j}{da}.$$

(*computes* all of the $\frac{dz_k}{da}$ terms)

- Reverse (Adjoint) Mode:

$$\frac{du}{dz_i} = \frac{\partial f^k_{\text{elem}}}{\partial z_i}\frac{du}{dz_k} \quad \text{and} \quad \frac{du}{dz_j} = \frac{\partial f^k_{\text{elem}}}{\partial z_j}\frac{du}{dz_k}.$$

(requires that $\frac{\partial f^k_{\text{elem}}}{\partial z_j}$ be *stored*)

Automatic Differentiation

Example: $u = a_1 2 a_2 + \sin(a_2 a_3)$

$$
\begin{array}{lll l}
z_1 & \leftarrow & a_1 & \text{(input)} \\
z_2 & \leftarrow & a_2 & \text{(input)} \\
z_3 & \leftarrow & a_3 & \text{(input)} \\
z_4 & \leftarrow & \mathrm{pow}(z_1, 2) & (a_1 2) \\
z_5 & \leftarrow & z_4 * z_2 & (a_1 2 a_2) \\
z_6 & \leftarrow & z_2 * z_3 & (a_2 a_3) \\
z_7 & \leftarrow & \sin(z_6) & (\sin(a_2 a_3)) \\
z_8 & \leftarrow & z_5 + z_7 & (u)
\end{array}
$$

AD: Forward Mode

Forward Mode: (Find $\frac{\partial u}{\partial a_1}$, i.e. $a = a_1$)

$$\frac{\partial z_1}{\partial a} \leftarrow \frac{\partial a_1}{\partial a} \qquad \text{(input = 1)}$$

$$\frac{\partial z_2}{\partial a} \leftarrow \frac{\partial a_2}{\partial a} \qquad \text{(input = 0)}$$

$$\frac{\partial z_3}{\partial a} \leftarrow \frac{\partial a_3}{\partial a} \qquad \text{(input = 0)}$$

$$\frac{\partial z_4}{\partial a} \leftarrow 2z_1 \frac{\partial z_1}{\partial a} \qquad (2z_1)$$

$$\frac{\partial z_5}{\partial a} \leftarrow \frac{\partial z_4}{\partial a} z_2 + z_4 \frac{\partial z_2}{\partial a} \qquad (2z_1 z_2)$$

$$\frac{\partial z_6}{\partial a} \leftarrow \frac{\partial z_2}{\partial a} z_3 + z_2 \frac{\partial z_3}{\partial a} \qquad (0)$$

$$\frac{\partial z_7}{\partial a} \leftarrow \cos(z_6) \frac{\partial z_6}{\partial a} \qquad (0)$$

$$\frac{\partial z_8}{\partial a} \leftarrow \frac{\partial z_5}{\partial a} + \frac{\partial z_7}{\partial a} \qquad (2z_1 z_2)$$

AD: Forward Mode

Forward Mode

- Implemented by *operator overloading*

  z_2 (and $\frac{\partial z_2}{\partial a}$)

  z_3 (and $\frac{\partial z_3}{\partial a}$)

  $z_2 * z_3$ (and $\frac{\partial z_2}{\partial a} z_3 + z_2 \frac{\partial z_3}{\partial a}$)

- **ADMAT**:**A**utomatic **D**ifferentiation for MATLAB
  - Coleman and Verma
- Or implemented by *source transformation*

AD: Reverse Mode

Reverse Mode: (Find $\frac{\partial u}{\partial a}$, a is an arbitrary input)

$$
\begin{array}{ccccc}
\frac{\partial z_8}{\partial a} & \leftarrow & \frac{\partial z_8}{\partial z_5}\frac{\partial z_5}{\partial a} + \frac{\partial z_8}{\partial z_7}\frac{\partial z_7}{\partial a} & = & \underline{1}\,\frac{\partial z_5}{\partial a} + \underline{1}\,\frac{\partial z_7}{\partial a} \\
\frac{\partial z_7}{\partial a} & \leftarrow & \frac{\partial z_7}{\partial z_6}\frac{\partial z_6}{\partial a} & = & \underline{\cos(z_6)}\,\frac{\partial z_6}{\partial a} \\
\frac{\partial z_6}{\partial a} & \leftarrow & \frac{\partial z_6}{\partial z_2}\frac{\partial z_2}{\partial a} + \frac{\partial z_6}{\partial z_3}\frac{\partial z_3}{\partial a} & = & \underline{z_3}\,\frac{\partial z_2}{\partial a} + \underline{z_2}\,\frac{\partial z_3}{\partial a} \\
\frac{\partial z_5}{\partial a} & \leftarrow & \frac{\partial z_5}{\partial z_4}\frac{\partial z_4}{\partial a} + \frac{\partial z_5}{\partial z_2}\frac{\partial z_2}{\partial a} & = & \underline{z_2}\,\frac{\partial z_4}{\partial a} + \underline{z_4}\,\frac{\partial z_2}{\partial a} \\
\frac{\partial z_4}{\partial a} & \leftarrow & \frac{\partial z_4}{\partial z_1}\frac{\partial z_1}{\partial a} & = & \underline{2z_1}\,\frac{\partial z_1}{\partial a}
\end{array}
$$

With the stored values of $\frac{\partial z_8}{\partial z_5}$, $\frac{\partial z_8}{\partial z_7}$, etc. on the forward sweep, we can determine $\frac{\partial u}{\partial a_1}$ by prescribing

$$
\frac{\partial z_1}{\partial a} = 1 \qquad \frac{\partial z_2}{\partial a} = 0 \qquad \frac{\partial z_3}{\partial a} = 0.
$$

AD: Reverse Mode

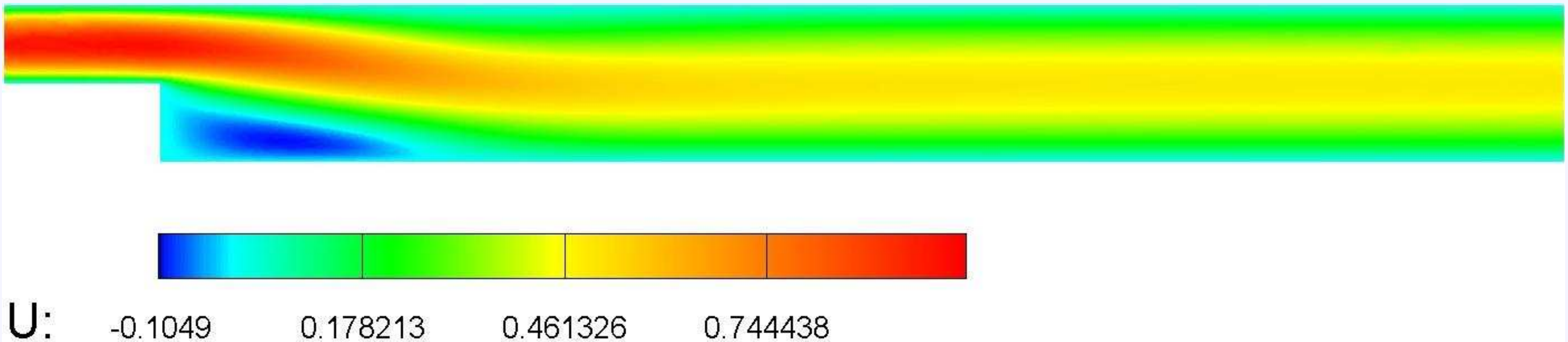
Reverse Mode

- Implemented by storing (underlined) variables on the “forward” solve.
- Preprocessing can be done to remove unneccessary storage.
- We can find $\frac{\partial u}{\partial a_1}$, $\frac{\partial u}{\partial a_2}$ and $\frac{\partial u}{\partial a_3}$ for the same work
- The *Cheap Gradient Theorem* states that we can find the gradient wrt an arbitrary number of inputs for $4\times$ function cost.
- These two modes can be combined.

Outline

- Problem Description
- Software
  - ViTLES
  - ADIC
  - PETSc
- Numerical Experiments
- Future Work

Problem Description



- Navier-Stokes Equation
- LES Models
- Sensitivities

Problem Description

Navier-Stokes equation

$$
\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla \cdot \tau(\mathbf{u}) + \nabla p &= \mathbf{f} \\
\nabla \cdot \mathbf{u} &= \mathbf{0}
\end{aligned}
$$

Problem Description

Navier-Stokes equation

$$
\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla \cdot \tau(\mathbf{u}) + \nabla p &= \mathbf{f} \\
\nabla \cdot \mathbf{u} &= \mathbf{0}
\end{aligned}
$$

Let $\bar{\mathbf{u}}(t) := [\mathbf{g}_\delta * \mathbf{u}](\mathbf{t})$.

Problem Description

Navier-Stokes equation

$$
\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \nabla\cdot\tau(\mathbf{u}) + \nabla p &= \mathbf{f} \\
\nabla\cdot\mathbf{u} &= \mathbf{0}
\end{aligned}
$$

Let $\bar{\mathbf{u}}(t) := [\mathbf{g}_\delta * \mathbf{u}](\mathbf{t})$. Then filtered N.S.

$$
\begin{aligned}
\frac{\partial \bar{\mathbf{u}}}{\partial t} + \nabla\cdot(\overline{\mathbf{u}\mathbf{u}}) - \nabla\cdot\tau(\bar{\mathbf{u}}) + \nabla \bar{p} &= \bar{\mathbf{f}} \\
\nabla\cdot\bar{\mathbf{u}} &= \mathbf{0}
\end{aligned}
$$

Problem Description

Navier-Stokes equation

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla \cdot \tau(\mathbf{u}) + \nabla p &= \mathbf{f} \\ \nabla \cdot \mathbf{u} &= \mathbf{0} \end{aligned}$$

Let $\bar{\mathbf{u}}(t) := [\mathbf{g}_\delta * \mathbf{u}](\mathbf{t})$. Then filtered N.S.

$$\begin{aligned} \frac{\partial \bar{\mathbf{u}}}{\partial t} + \nabla \cdot (\overline{\mathbf{u}\mathbf{u}}) - \nabla \cdot \tau(\bar{\mathbf{u}}) + \nabla \bar{p} &= \bar{\mathbf{f}} \\ \nabla \cdot \bar{\mathbf{u}} &= \mathbf{0} \end{aligned}$$

To close the system we need to model the $\nabla \cdot (\overline{\mathbf{u}\mathbf{u}})$ term since $\overline{\mathbf{u}\mathbf{u}} \neq \bar{\mathbf{u}}\bar{\mathbf{u}}$.

Problem Description

- **Some LES Models**
  - Smagorinsky Model
  - Deconvolution Model

Problem Description

Smagorinsky Model

Approximate $\nabla \cdot (\overline{\mathbf{uu}})$ term by

$$\nabla \cdot (\overline{\mathbf{uu}}) \approx \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) - \nabla \cdot \left[(C_s\delta)^2 \|\mathbf{D}(\bar{u})\|\mathbf{D}(\bar{u})\right]$$

where $\mathbf{D}(\bar{u}) := \frac{1}{2}\left(\nabla u + \nabla u^T\right)$

Problem Description

Smagorinsky Model

Approximate $\nabla \cdot (\overline{\mathbf{uu}})$ term by

$$\nabla \cdot (\overline{\mathbf{uu}}) \approx \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) - \nabla \cdot \left[(C_s \delta)^2 \|\mathbf{D}(\bar{u})\| \mathbf{D}(\bar{u}) \right]$$

where $\mathbf{D}(\bar{u}) := \frac{1}{2} \left(\nabla u + \nabla u^T \right)$

Deconvolution Model

$$\nabla \cdot (\overline{\mathbf{uu}}) \approx \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) + \nabla \cdot \tilde{\tau}$$

where $\tilde{\tau} = (\tilde{\tau}_{ij})$ with $\tilde{\tau}_{ij}$ solving

$$(I - \delta^2 \Delta) \tilde{\tau}_{ij} = 2\delta^2 \nabla \bar{u}_i \nabla \bar{u}_j$$

Problem Description

Sensitivities

Flow computations are dependent on closure model and their parameters.

Problem Description

Sensitivities

Flow computations are dependent on closure model and their parameters.

Question: How sensitive are computations to closure model parameters?

Problem Description

Sensitivities

Flow computations are dependent on closure model and their parameters.

Question: How sensitive are computations to closure model parameters?

Problem: Determine sensitivity of turbulent flow to closure model parameters.

Problem Description

Smagorinsky and Deconvolution Model Sensitivities

Problem Description

Smagorinsky and Deconvolution Model Sensitivities

Parameter:

A filter radius δ

Problem Description

Smagorinsky and Deconvolution Model Sensitivities

Parameter:

A filter radius δ

Determine:

Sensitivities $\frac{\partial \bar{\mathbf{u}}(x,t)}{\partial \delta}$ and $\frac{\partial \bar{\mathbf{p}}(x,t)}{\partial \delta}$

Software

ViTLES (Virginia Tech Large Eddy Simulation)

Software

ViTLES (Virginia Tech Large Eddy Simulation)

- FEM (Finite Element Method)

Software

ViTLES (Virginia Tech Large Eddy Simulation)

- FEM (Finite Element Method)
- PETSc(Portable, Extensible Toolkit for Scientific Computation)

Software

ViTLES (Virginia Tech Large Eddy Simulation)

- FEM (Finite Element Method)
- PETSc(Portable, Extensible Toolkit for Scientific Computation)
- ADIC(Automatic Differentiation of ANSI C)

Software

ViTLES (Virginia Tech Large Eddy Simulation)

- FEM (Finite Element Method)
- PETSc(Portable, Extensible Toolkit for Scientific Computation)
- ADIC(Automatic Differentiation of ANSI C)
- The Virginia Tech system X is used for computations

Software

Applying AD

Applying AD

Let δ be a closure model parameter.

Applying AD

Let δ be a closure model parameter.

One time step of discretized N.S. (Crank-Nicholson)

$$\mathbf{F}(\mathbf{x}(\delta), \tilde{x}(\delta), \delta) + \mathbf{G}(\mathbf{y}(\delta), \tilde{y}(\delta), \delta) = 0$$

Applying AD

Let δ be a closure model parameter.

One time step of discretized N.S. (Crank-Nicholson)

$$\mathbf{F}(\mathbf{x}(\delta), \tilde{x}(\delta), \delta) + \mathbf{G}(\mathbf{y}(\delta), \tilde{y}(\delta), \delta) = 0$$

where

$\mathbf{x}$ and $\mathbf{y}$ are state vectors at $n+1$ and n time steps

$\tilde{x}$ and $\tilde{y}$ are vectors of D.B.C. at $n+1$ and n time steps

Applying AD

We look for a sensitivity $\frac{\partial \mathbf{x}}{\partial \delta}$.

$$\mathbf{F}_{\mathbf{x}}(\mathbf{x},\tilde{x},\delta)\frac{\partial \mathbf{x}}{\partial \delta} = -\left[\mathbf{F}_{\tilde{x}}(\mathbf{x},\tilde{x},\delta)\frac{\partial \tilde{x}}{\partial \delta} + \mathbf{F}_{\delta}(\mathbf{x},\tilde{x},\delta) + \mathbf{G}_{\mathbf{y}}(\mathbf{y},\tilde{y},\delta)\frac{\partial \mathbf{y}}{\partial \delta} + \mathbf{G}_{\tilde{y}}(\mathbf{y},\tilde{y},\delta)\frac{\partial \tilde{y}}{\partial \delta} + \mathbf{G}_{\delta}(\mathbf{y},\tilde{y},\delta)\right]$$

Applying AD

We look for a sensitivity $\frac{\partial \mathbf{x}}{\partial \delta}$.

$$\mathbf{F}_{\mathbf{x}}(\mathbf{x}, \tilde{x}, \delta)\frac{\partial \mathbf{x}}{\partial \delta} = -\left[\mathbf{F}_{\tilde{x}}(\mathbf{x}, \tilde{x}, \delta)\frac{\partial \tilde{x}}{\partial \delta} + \mathbf{F}_{\delta}(\mathbf{x}, \tilde{x}, \delta) + \right.$$
$$\left. \mathbf{G}_{\mathbf{y}}(\mathbf{y}, \tilde{y}, \delta)\frac{\partial \mathbf{y}}{\partial \delta} + \mathbf{G}_{\tilde{y}}(\mathbf{y}, \tilde{y}, \delta)\frac{\partial \tilde{y}}{\partial \delta} + \mathbf{G}_{\delta}(\mathbf{y}, \tilde{y}, \delta)\right]$$

Set dependent variables and gradients and evaluate RHS by finding

$$\frac{\partial}{\partial \delta}\mathbf{F}(\mathbf{x}, \tilde{x}(\delta), \delta) = \mathbf{F}_{\tilde{x}}(\mathbf{x}, \tilde{x}, \delta)\frac{\partial \tilde{x}}{\partial \delta} + \mathbf{F}_{\delta}(\mathbf{x}, \tilde{x}, \delta)$$

$$\frac{\partial}{\partial \delta}\mathbf{G}(\mathbf{y}(\delta), \tilde{y}(\delta), \delta) = \mathbf{G}_{\mathbf{y}}(\mathbf{y}, \tilde{y}, \delta)\frac{\partial \mathbf{y}}{\partial \delta} + \mathbf{G}_{\tilde{y}}(\mathbf{y}, \tilde{y}, \delta)\frac{\partial \tilde{y}}{\partial \delta} + \mathbf{G}_{\delta}(\mathbf{y}, \tilde{y}, \delta)$$

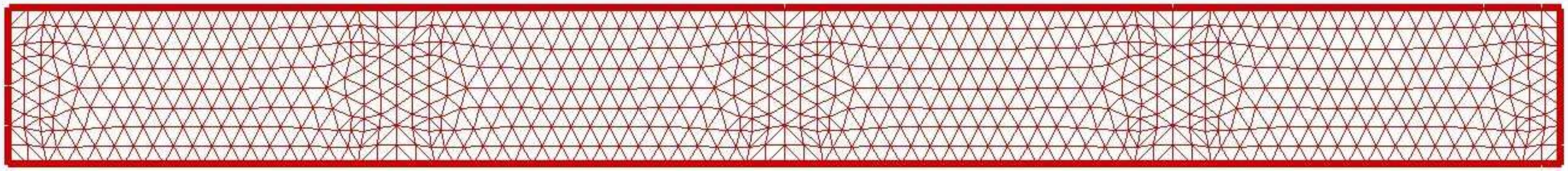
Numerical Experiments

2D Smagorinsky Model

Numerical Experiments

2D Smagorinsky Model

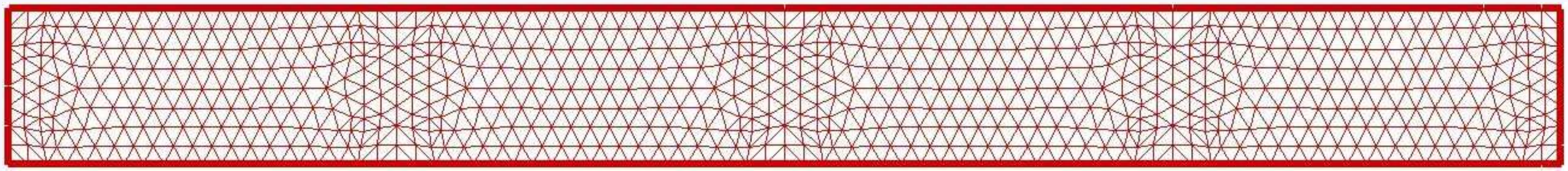
- Channel



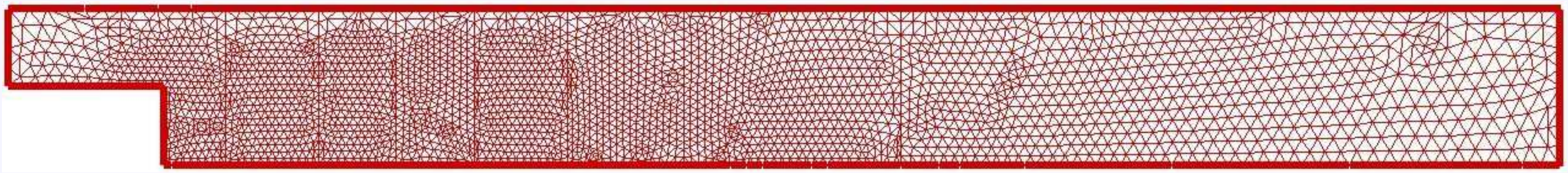
Numerical Experiments

2D Smagorinsky Model

- Channel



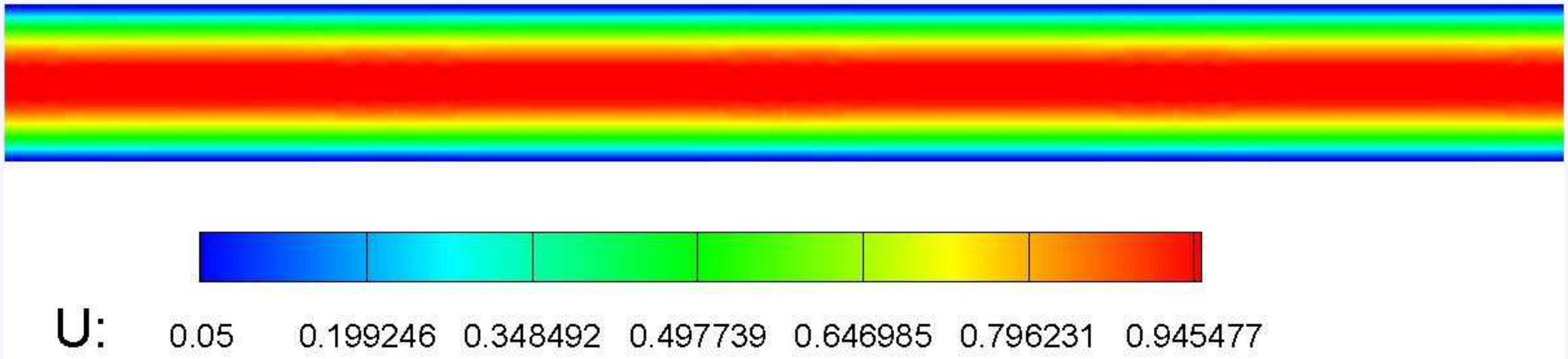
- Backward Facing Step



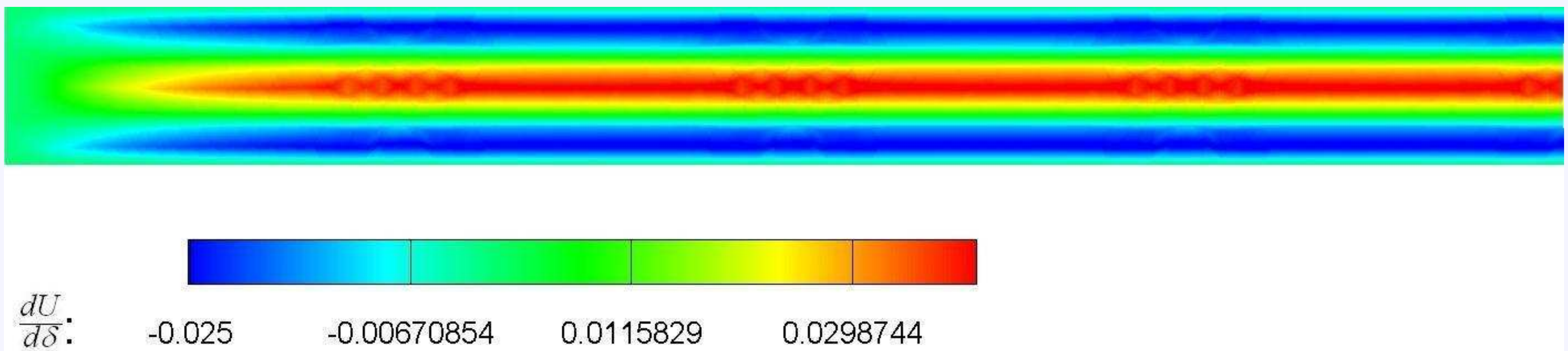
Numerical Experiments

Channel

1st velocity component



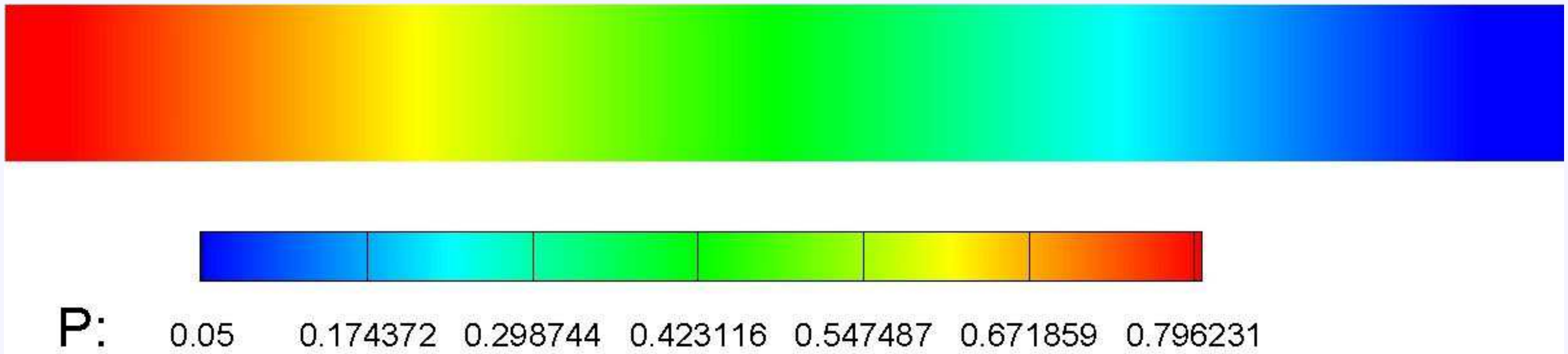
1st velocity component sensitivity



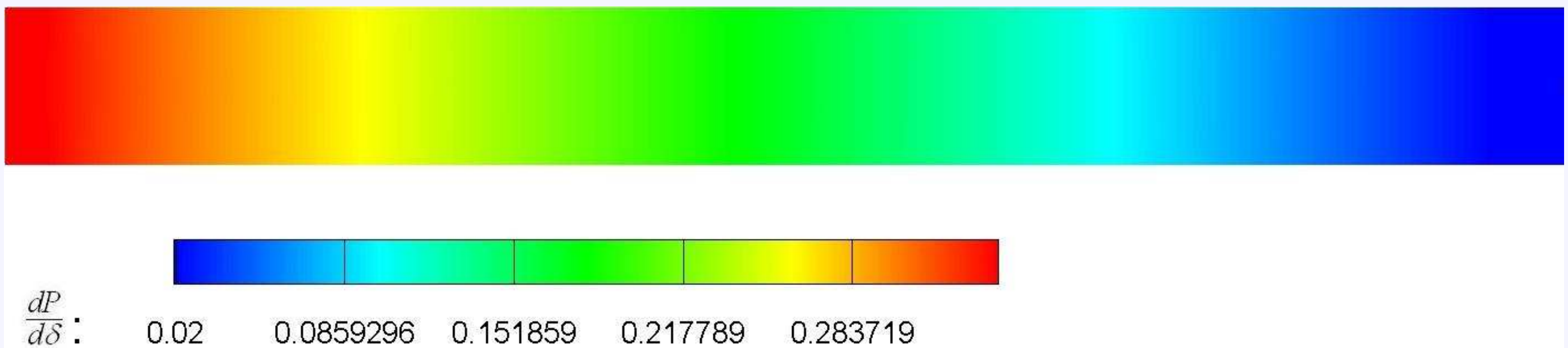
Numerical Experiments

Channel

pressure

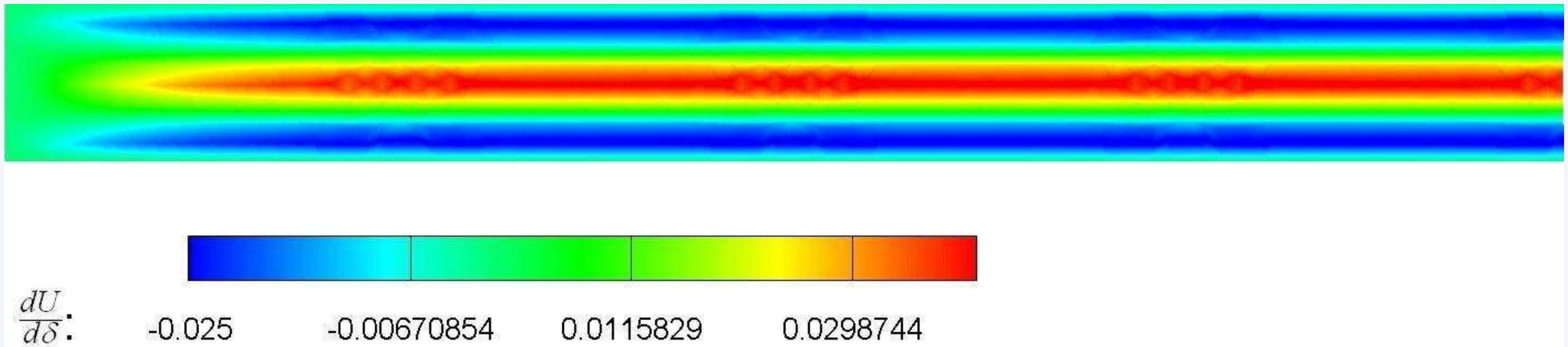


pressure sensitivity

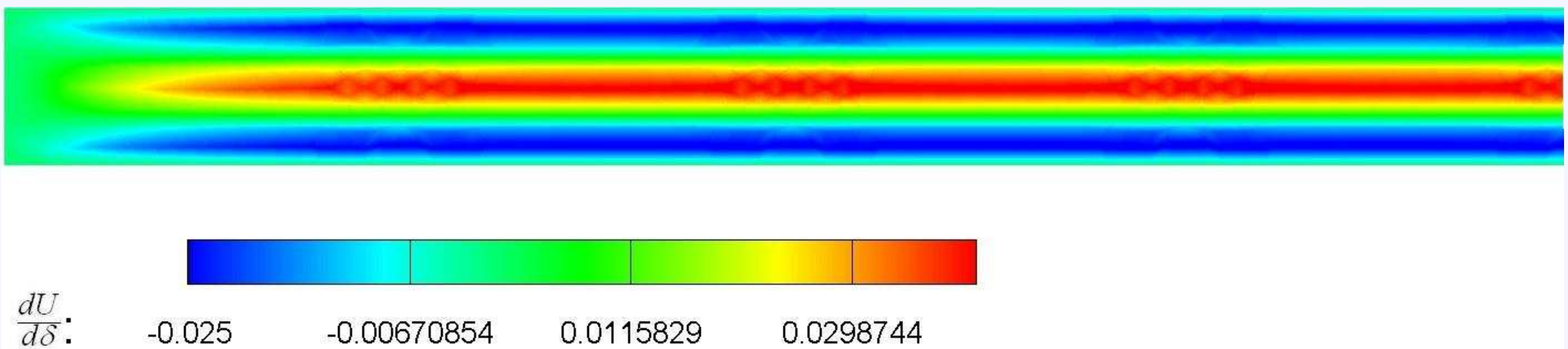


Channel: Finite Difference Test (Velocity U Sensitivity)

automatic differentiation

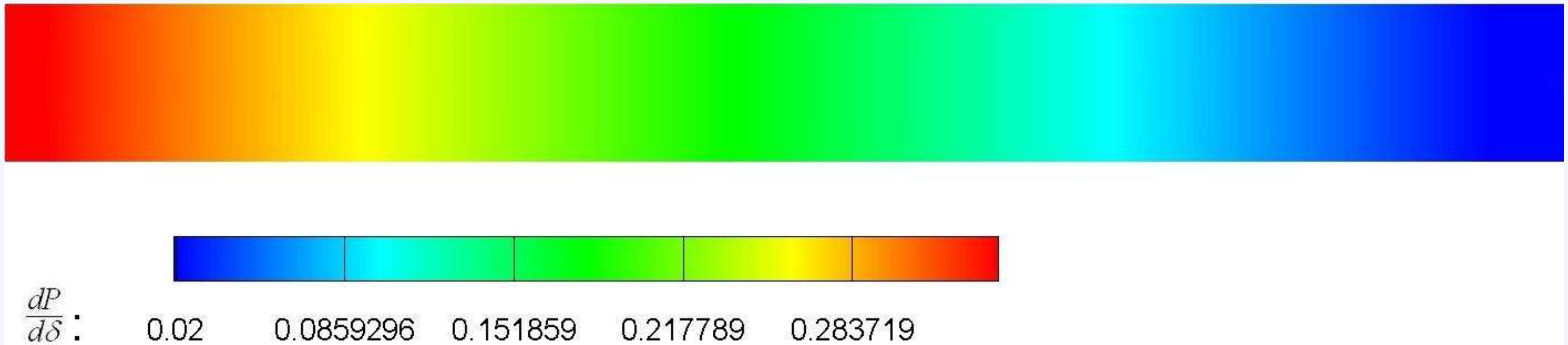


finite difference

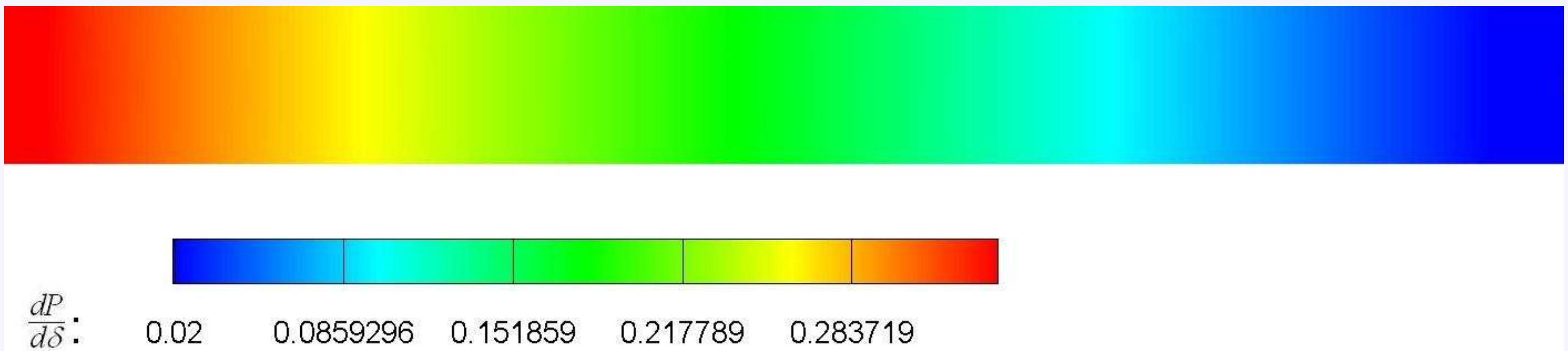


Channel: Finite Difference Test (Pressure Sensitivity)

automatic differentiation



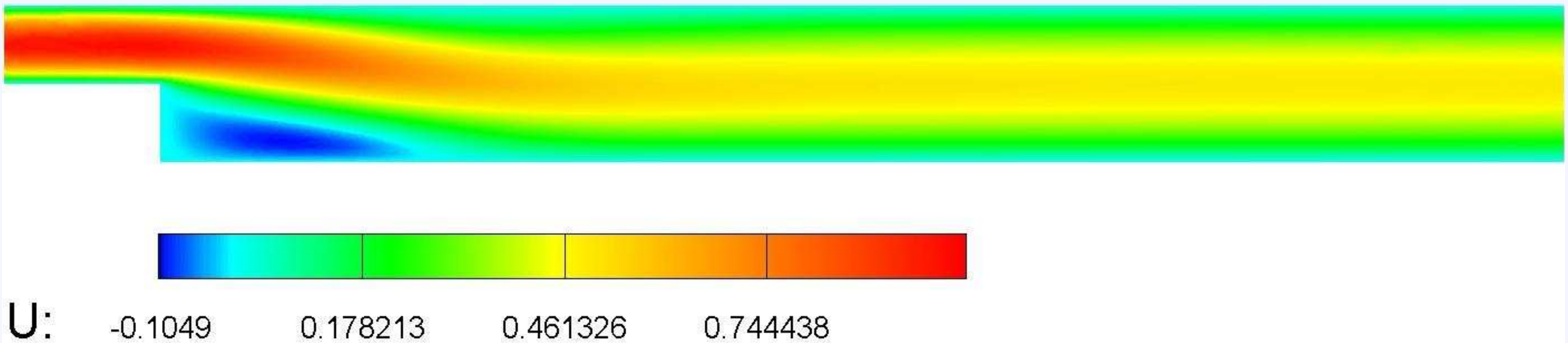
finite difference



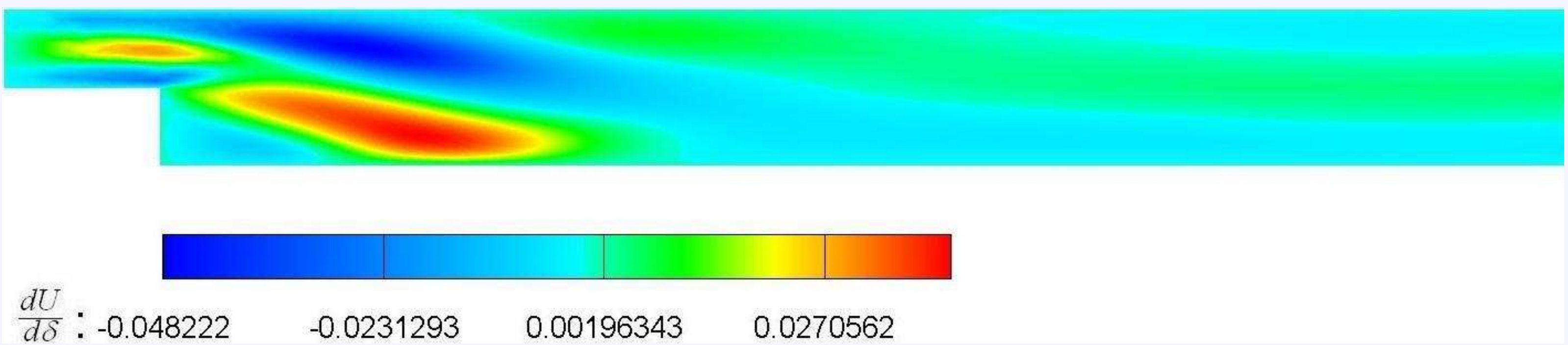
Numerical Experiments

Backward Facing Step

1st velocity component



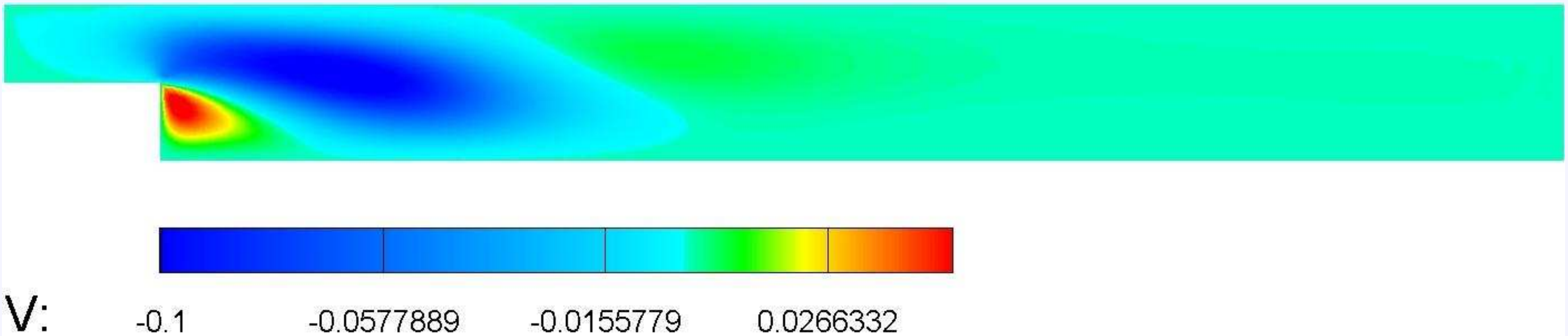
1st velocity component sensitivity



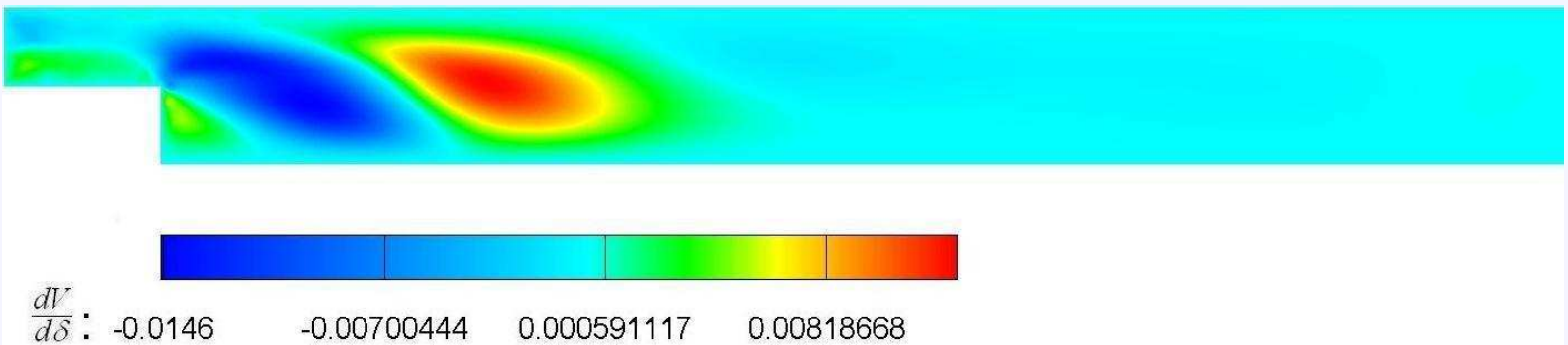
Numerical Experiments

Backward Facing Step

2nd velocity component



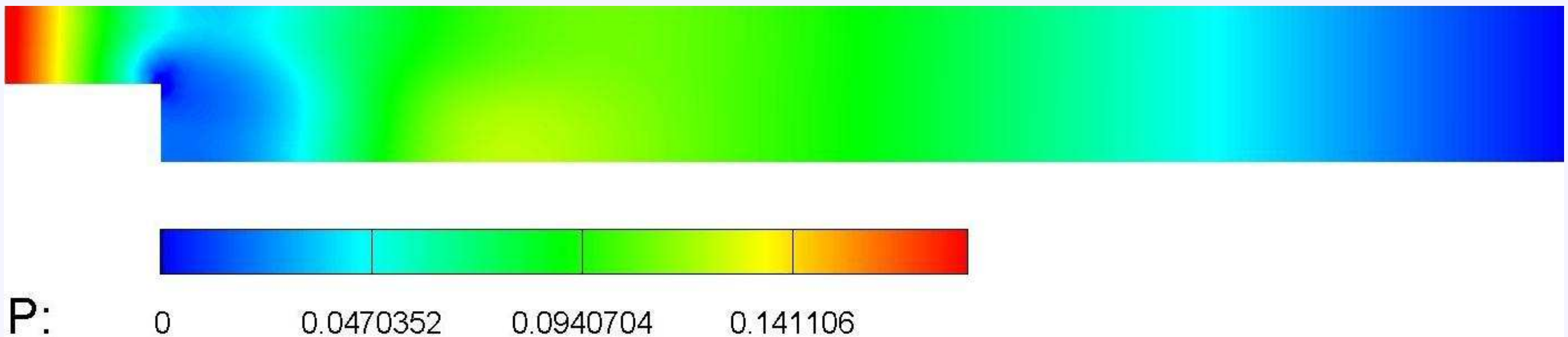
2nd velocity component sensitivity



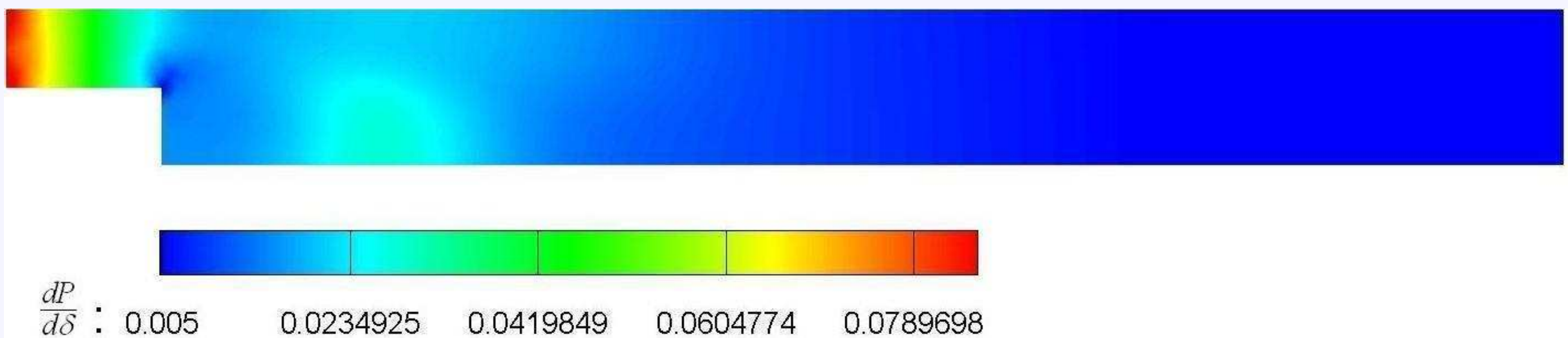
Numerical Experiments

Backward Facing Step

pressure

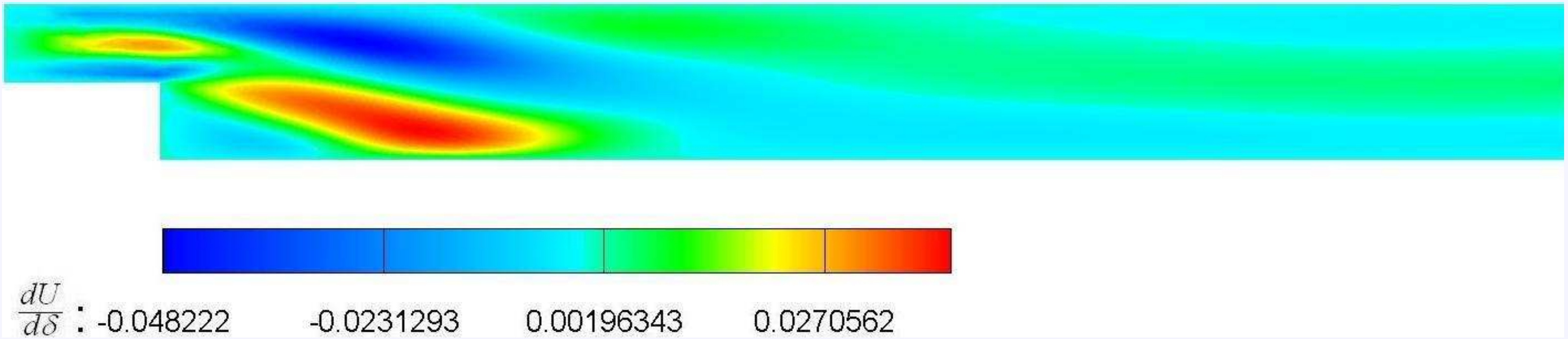


pressure sensitivity

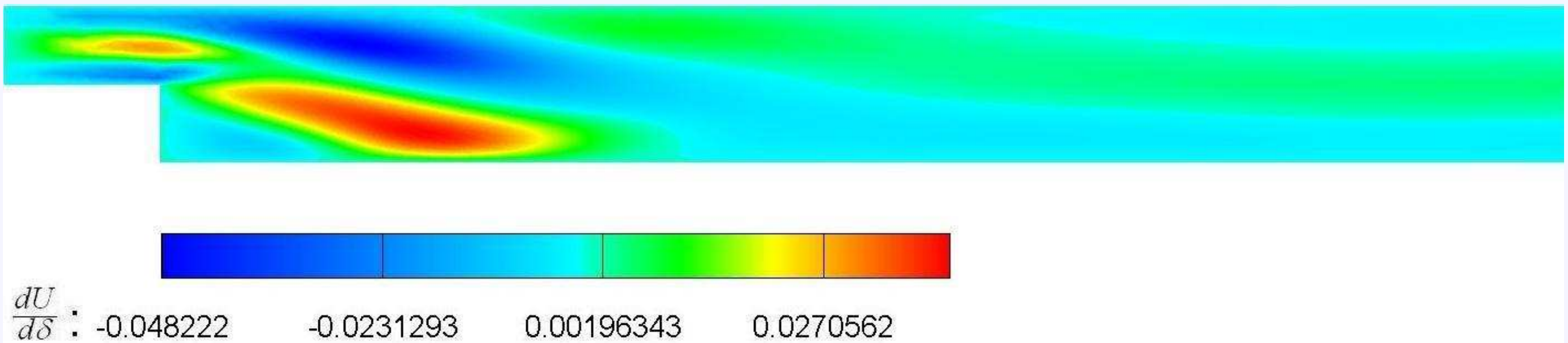


BFS: Finite Difference Test (Velocity U Sensitivity)

automatic differentiation

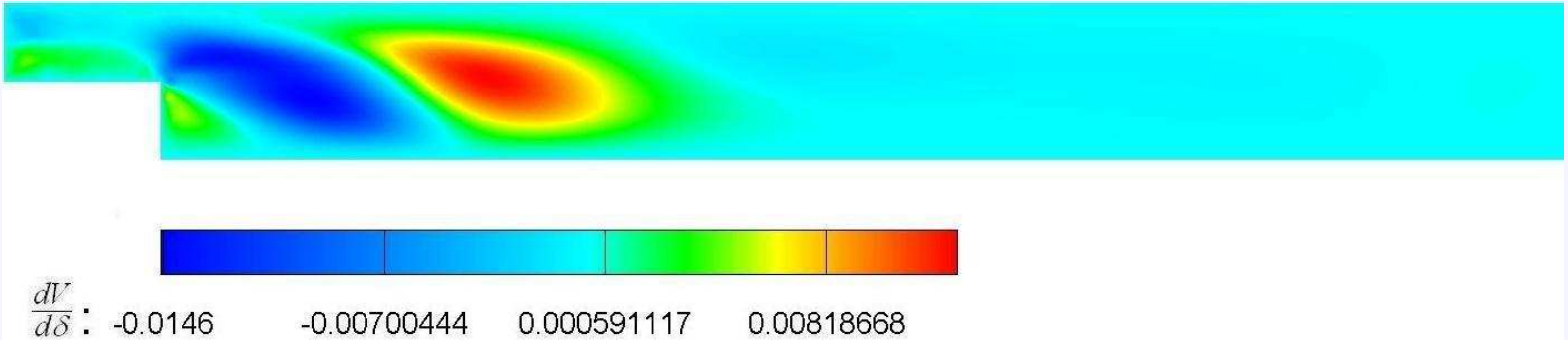


finite difference

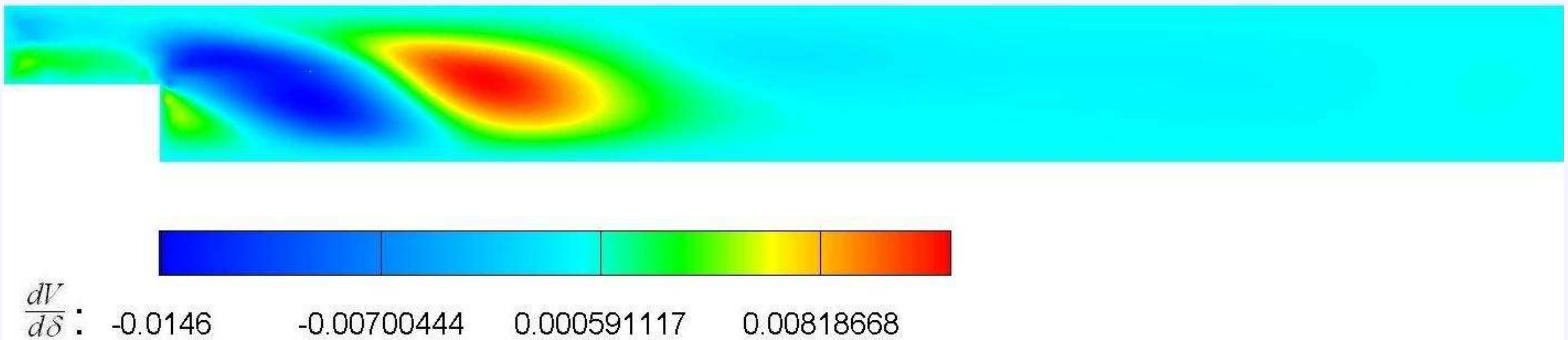


BFS: Finite Difference Test (Velocity V Sensitivity)

automatic differentiation



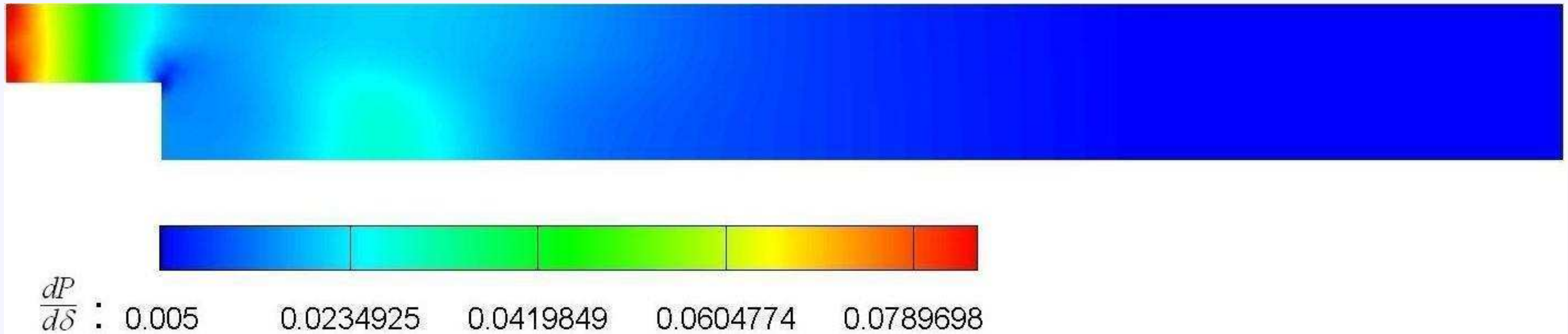
finite difference



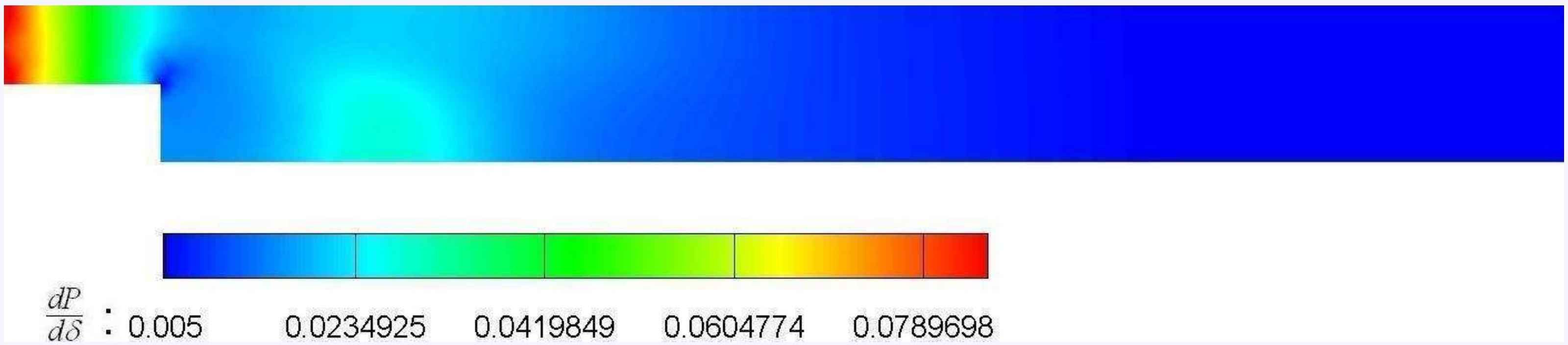
Numerical Experiments

BFS: Finite Difference Test (Pressure Sensitivity)

automatic differentiation



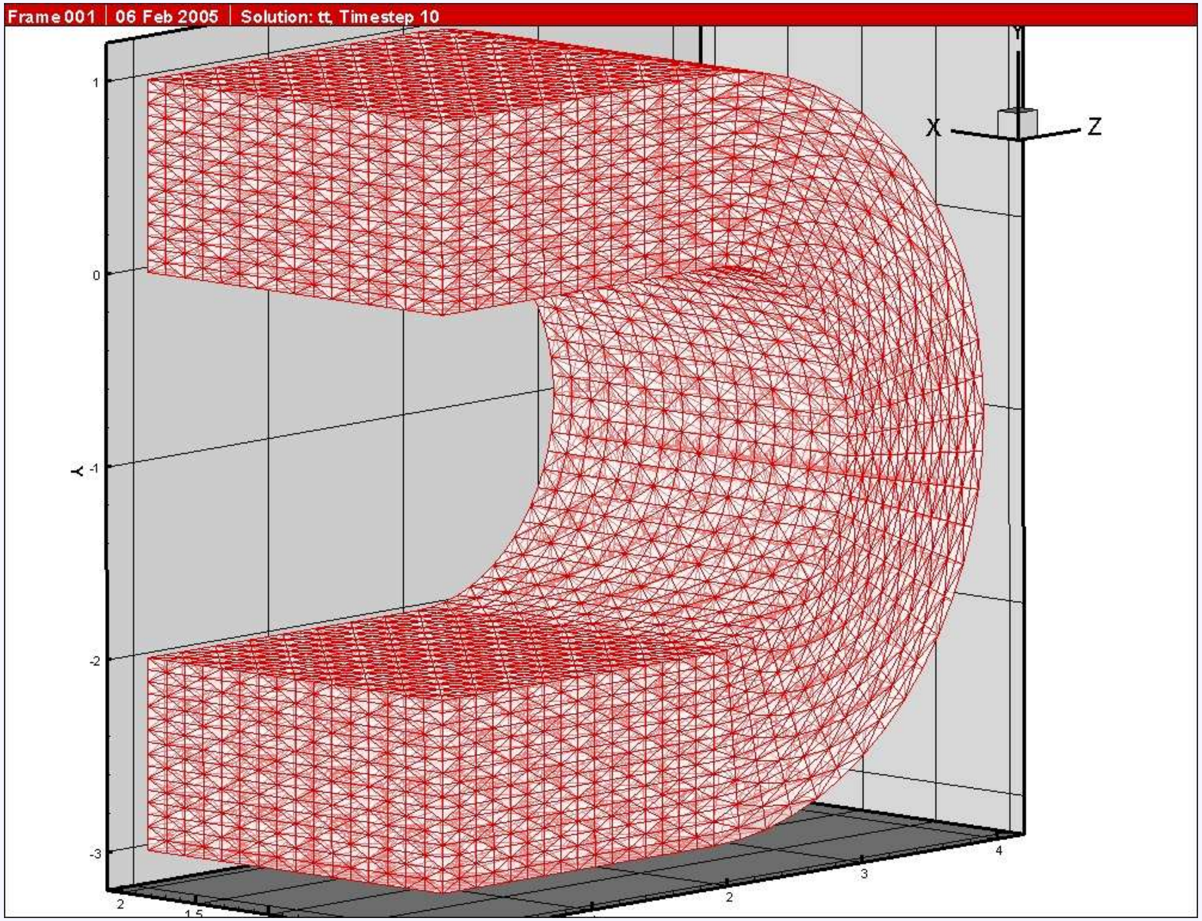
finite difference



Future Work

3D Navier-Stokes and Sensitivity Computations

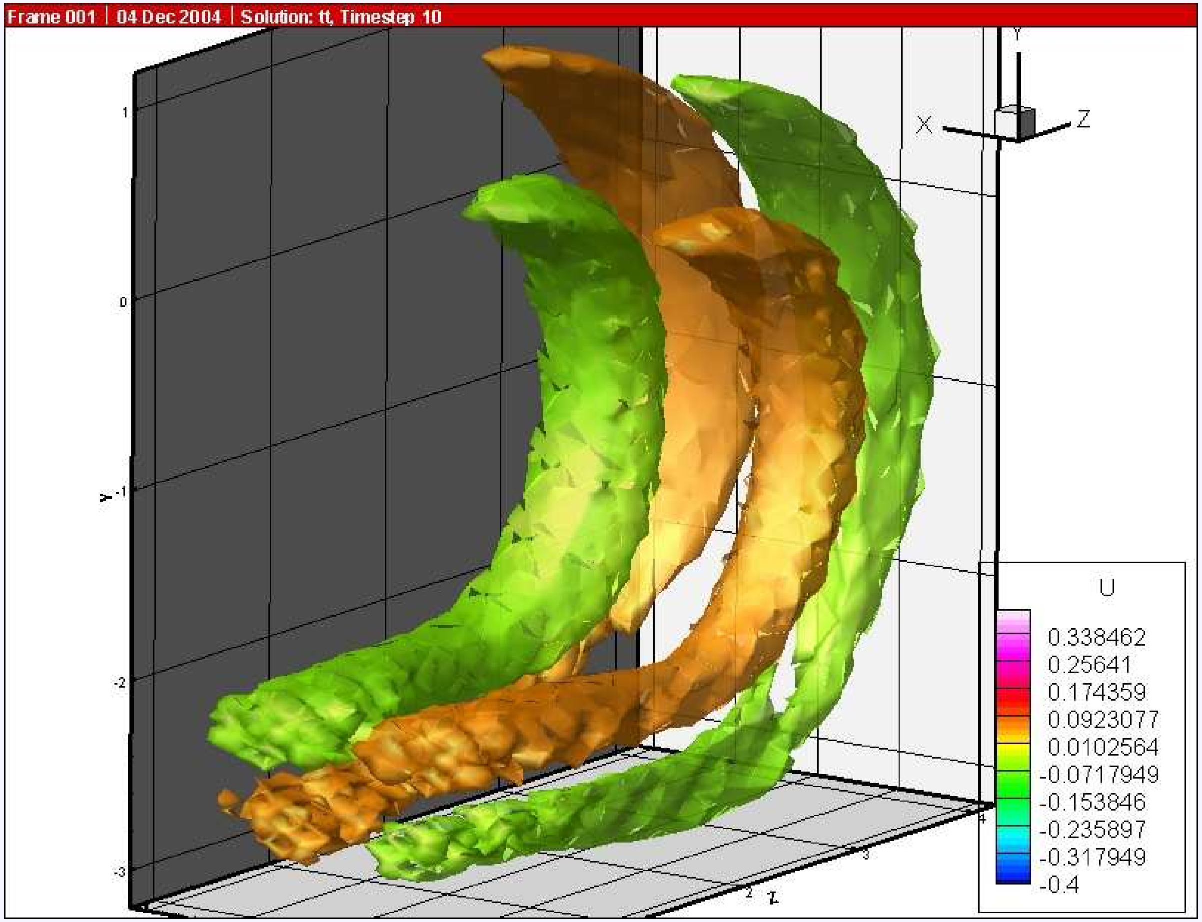
mesh



Future Work

3D Navier-Stokes and Sensitivity Computations

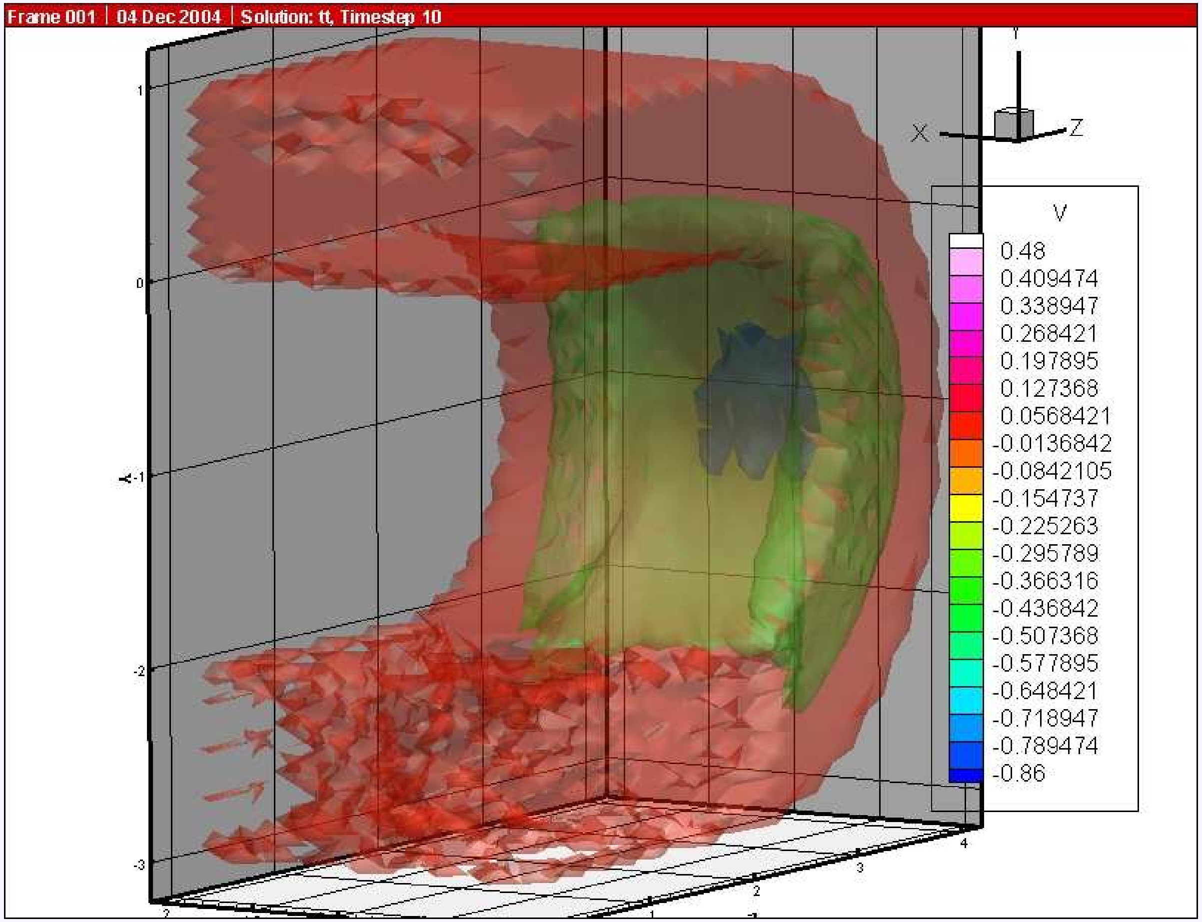
1st velocity component iso-surfaces



Future Work

3D Navier-Stokes and Sensitivity Computations

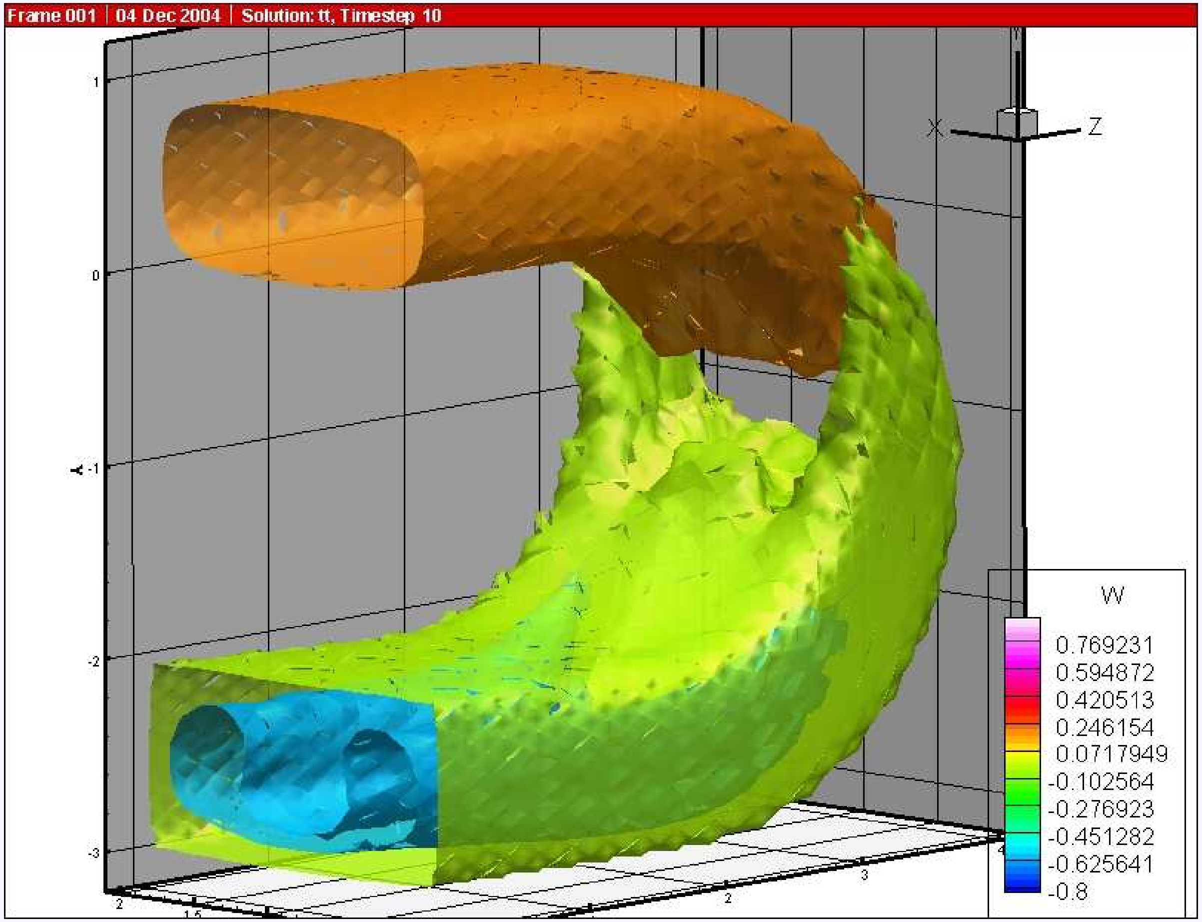
2nd velocity component iso-surfaces



Future Work

3D Navier-Stokes and Sensitivity Computations

3rd velocity component iso-surfaces



Future Work

Sensitivity Computation for Deconvolution Model

We approximate $\nabla \cdot (\overline{\mathbf{u}\mathbf{u}})$ term by

$$\nabla \cdot (\overline{\mathbf{u}\mathbf{u}}) \approx \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) + \nabla \cdot \tilde{\tau}$$

where $\tilde{\tau} = (\tilde{\tau}_{ij})$ with $\tilde{\tau}_{ij}$ solving

$$(I - \delta^2 \Delta)\tilde{\tau}_{ij} = 2\delta^2 \nabla \cdot \bar{u}_i \nabla \bar{u}_j$$

Future Work

Sensitivity Computation for Deconvolution Model

Let $s_i = \frac{\partial \bar{\mathbf{u}}}{\partial \delta}$. Then at n-th time step solve for $\frac{\partial \tilde{\tau}_{ij}}{\partial \delta}$

$$-2\delta\Delta\tilde{\tau}_{ij} + (I - \delta^2\Delta)\frac{\partial \tilde{\tau}_{ij}}{\partial \delta} = 2\delta\nabla\bar{u}_i\nabla\bar{u}_j + \delta^2(\nabla s_i\nabla\bar{u}_j + \nabla\bar{u}_i\nabla s_j)$$

Future Work

Sensitivity Computation for Deconvolution Model

Let $s_i = \frac{\partial \bar{\mathbf{u}}}{\partial \delta}$. Then at n-th time step solve for $\frac{\partial \tilde{\tau}_{ij}}{\partial \delta}$

$$-2\delta\Delta\tilde{\tau}_{ij} + (I - \delta^2\Delta)\frac{\partial \tilde{\tau}_{ij}}{\partial \delta} = 2\delta\nabla\bar{u}_i\nabla\bar{u}_j + \delta^2(\nabla s_i\nabla\bar{u}_j + \nabla\bar{u}_i\nabla s_j)$$

Let $\tilde{T}$ denote a discretized version of $\tilde{\tau}_{ij}$ computed at n-th time step.

One time step of discretized N.S. (Crank-Nicholson)

$$\mathbf{F}(\mathbf{x}(\delta), \tilde{x}(\delta), \delta) + \mathbf{G}(\mathbf{y}(\delta), \tilde{y}(\delta), \tilde{T}(\delta), \delta) = 0$$

Future Work

Sensitivity Computation for Deconvolution Model

Using data $\tilde{T}_{ij}$ and $\frac{\partial \tilde{T}_{ij}}{\partial \delta}$ we set dependent variables and gradients.

Then solve for a $\frac{\partial \mathbf{x}}{\partial \delta}$

$$\begin{aligned}\mathbf{F}_{\mathbf{x}}(\mathbf{x},\tilde{x},\delta)\frac{\partial \mathbf{x}}{\partial \delta} = -\Bigg[&\mathbf{F}_{\tilde{x}}(\mathbf{x},\tilde{x},\delta)\frac{\partial \tilde{x}}{\partial \delta} + \mathbf{F}_{\delta}(\mathbf{x},\tilde{x},\delta) + \\ &\mathbf{G}_{\mathbf{y}}(\mathbf{y},\tilde{y},\tilde{T},\delta)\frac{\partial \mathbf{y}}{\partial \delta} + \mathbf{G}_{\tilde{y}}(\mathbf{y},\tilde{y},\tilde{T},\delta)\frac{\partial \tilde{y}}{\partial \delta} \\ &+\mathbf{G}_{\delta}(\mathbf{y},\tilde{y},\tilde{T},\delta) + \mathbf{G}_{\tilde{T}}(\mathbf{y},\tilde{y},\tilde{T},\delta)\frac{\partial \tilde{T}}{\partial \delta}\Bigg]\end{aligned}$$