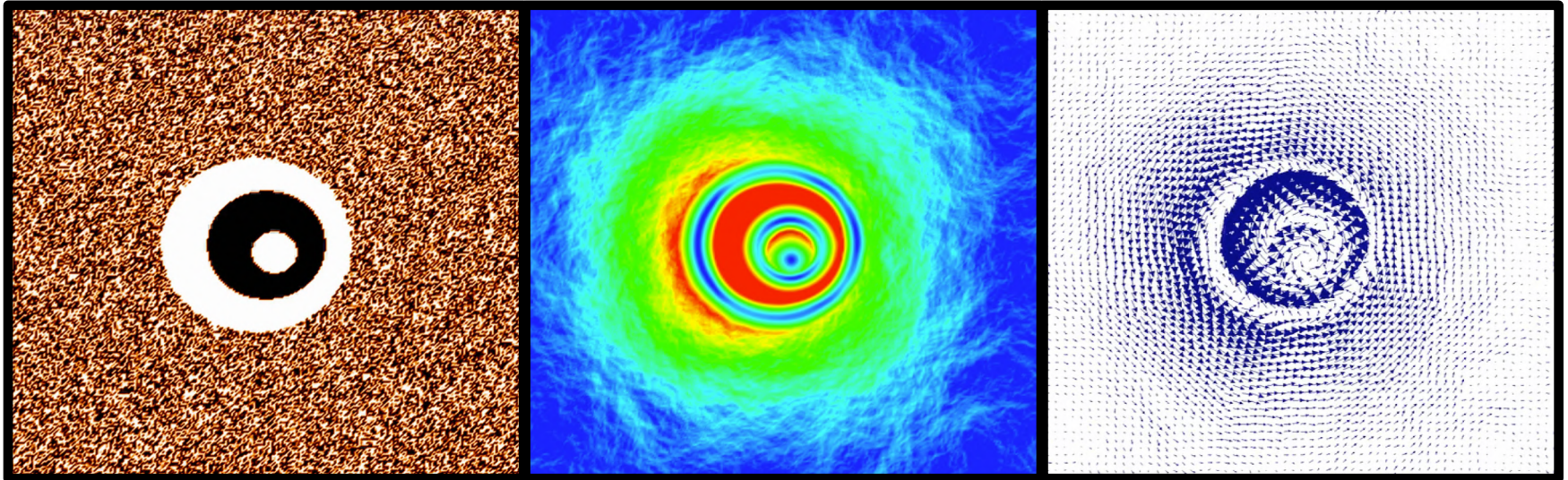


# suite-CFD:

## *Open Source Fluid Dynamics Codes in MATLAB / Python*



Nicholas A. Battista

[battistn@tcnj.edu](mailto:battistn@tcnj.edu)

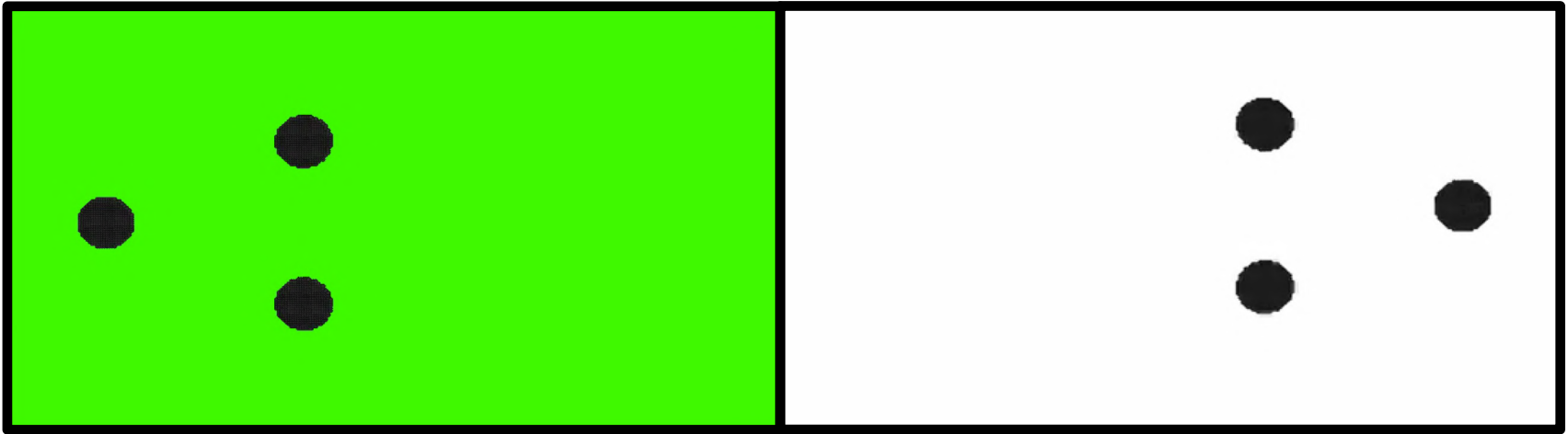
Dept. of Math and Statistics  
School of Science  
The College of New Jersey



**TCNJ**  
THE COLLEGE OF  
NEW JERSEY

# suite-CFD:

*Open Source Fluid Dynamics Codes in MATLAB / Python*



Nicholas A. Battista

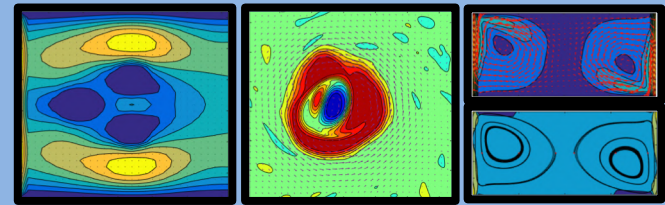
[battistn@tcnj.edu](mailto:battistn@tcnj.edu)


Dept. of Math and Statistics  
School of Science  
The College of New Jersey




**TCNJ**  
THE COLLEGE OF  
NEW JERSEY

# Numerical Fluid Solvers



 Search or jump to... Pull requests Issues Marketplace Explore



is nerdin' out!

## Nick Battista

nickabattista

Edit profile

Asst. Math. Prof. at TCNJ, interested in FSI, numerical PDE, mathematical biology, physiology, and educational tools/software.

TCNJ


Ewing, NJ

[nickabattista@gmail.com](mailto:nickabattista@gmail.com)

<http://battistn.pages.tcnj.edu>

Overview Repositories 13 Projects 0 Packages 0 Stars 16 Followers 61 Following 47


Pinned Customize your pins



### IB2d

An easy to use immersed boundary method in 2D, with full implementations in MATLAB and Python that contains over 60 built-in examples, including multiple options for fiber-structure models and adve...


MATLAB ★ 38 🍴 34



### Ark

An array of codes used for teaching various aspects of numerical analysis, covering Interpolation, Quadrature, basic ODE solves and Spectral solvers, as well as Monte Carlo methods.


MATLAB 🍴 1



### Holy\_Grail

An array of fluid solver codes, including Projection, Pseudo-Spectral (FFT), Lattice Boltzmann, and the Panel Method with implementations in both MATLAB and Python3


MATLAB ★ 8 🍴 9



### Peacocks\_Eye

Books, notes, and mathematical/scientific writings in progress.


Mathematica



### Sankara\_Stones

An array of codes for solving nonlinear elliptic PDEs and advection-diffusion equations using Chebyshev pseudo-spectral methods.

MATLAB 🍴 3



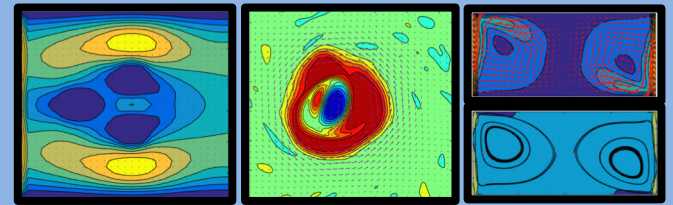
### Grail\_Tablet

MATLAB and Python 3.5 scripts for printing data (points, scalar, vector, etc) to VTK formats

MATLAB

Open Source Codes Available: [github.com/nickabattista](https://github.com/nickabattista)

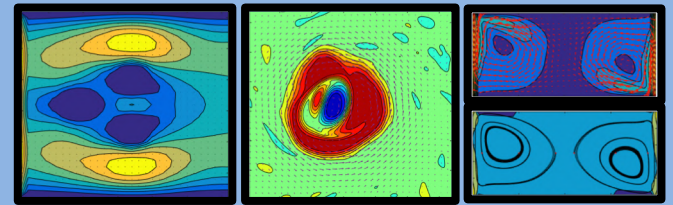
# Numerical Fluid Solvers



- **Fluid solvers for various applications**  
    (“pick your preference”)
- Projection Methods
- Spectral Methods
- Lattice Boltzmann

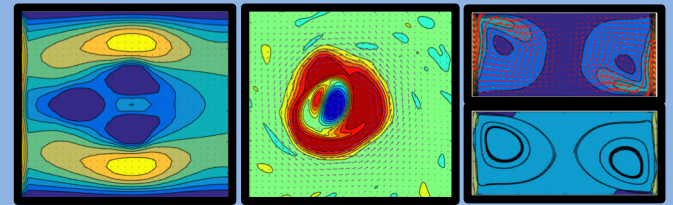


# Numerical Fluid Solvers



- Fluid solvers for various applications  
("pick your preference")
- **Projection Methods**
- **Spectral Methods**
- **Lattice Boltzmann**

# Numerical Fluid Solvers



- Fluid solvers for various applications  
("pick your preference")

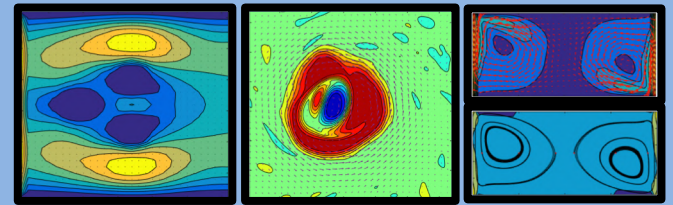
- **Projection Methods**

- **Spectral Methods**

- **Lattice Boltzmann**

Code available:

[github.com/nickabattista/Holy\\_Grail](https://github.com/nickabattista/Holy_Grail)

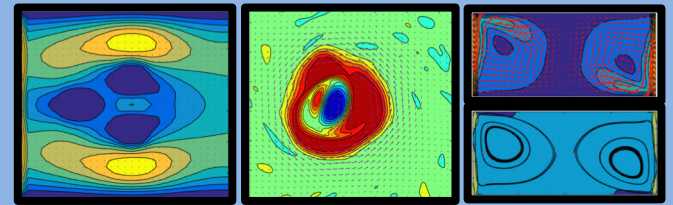


## PROJECTION METHODS

Examples:

1. Cavity Flow in Rectangular Domain
2. Circular Flow in Square Domain

***\*Overview of Numerical Scheme to follow\****



## PROJECTION METHODS

Examples:

1. Cavity Flow in Rectangular Domain

2. Circular Flow in Square Domain

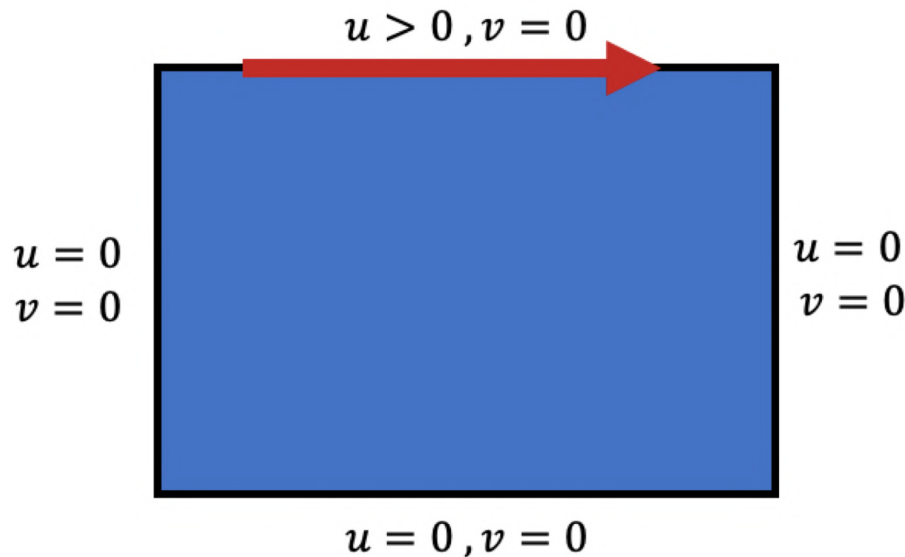
*\*Overview of Numerical Scheme to follow\**



# Projection Methods: Cavity Flow

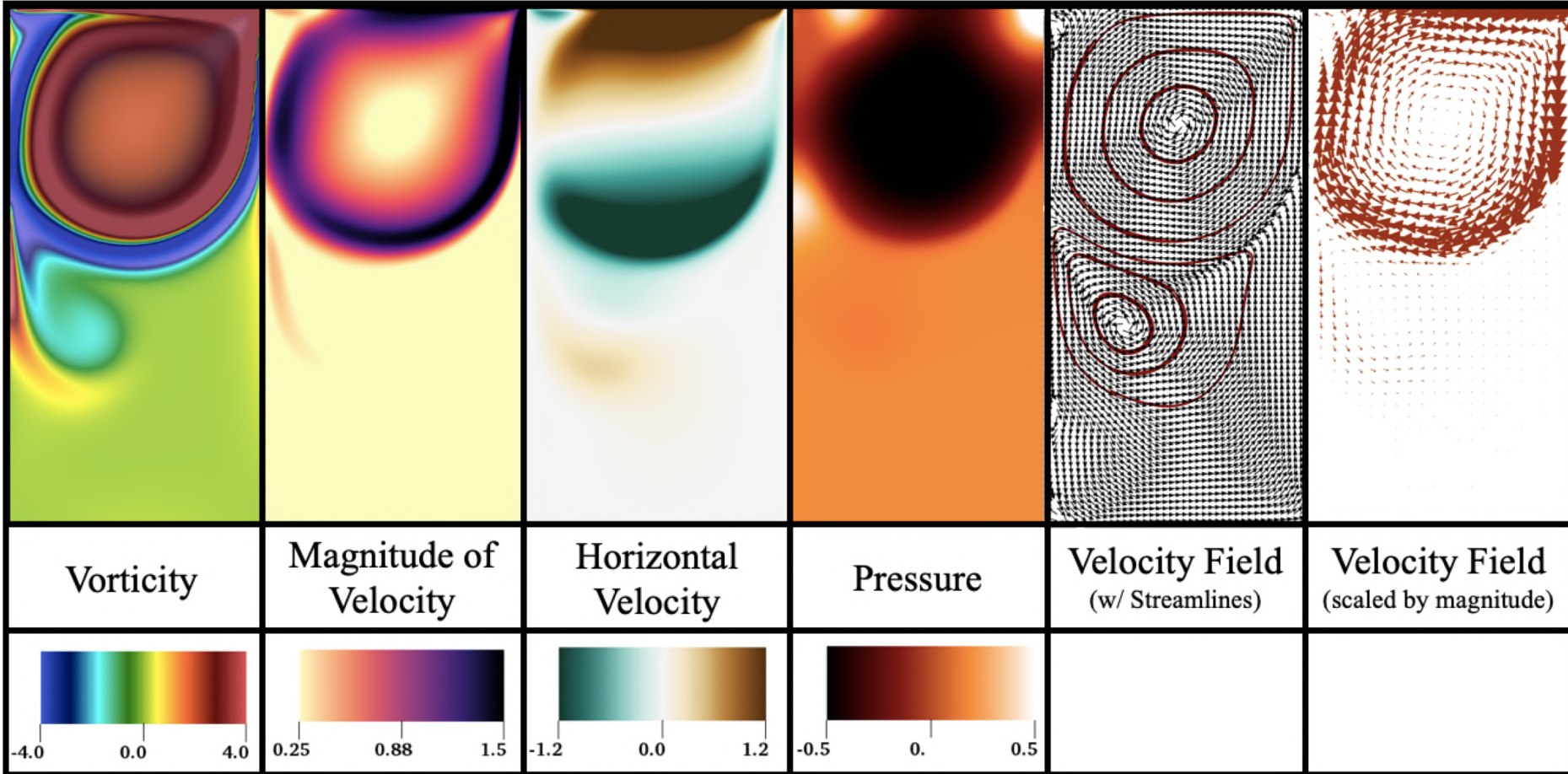


Cavity Flow Boundary Conditions



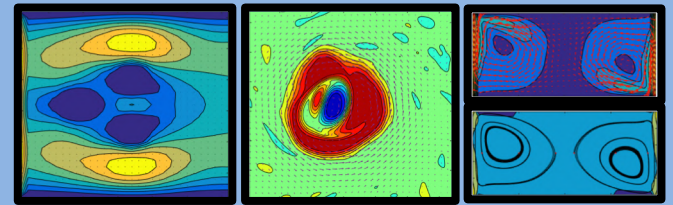
- Flow speed is ramped up across top of rectangular domain
- Observe qualitative differences due to differing Reynolds Numbers
- Measure horizontal velocity across middle of domain

# Projection Methods: Cavity Flow

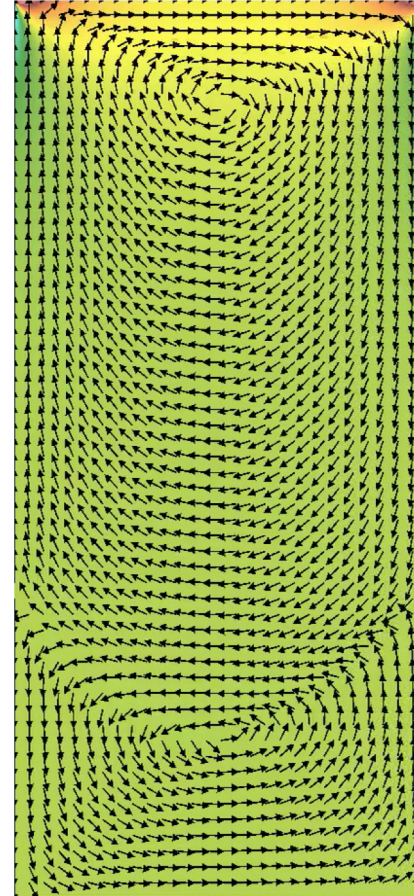
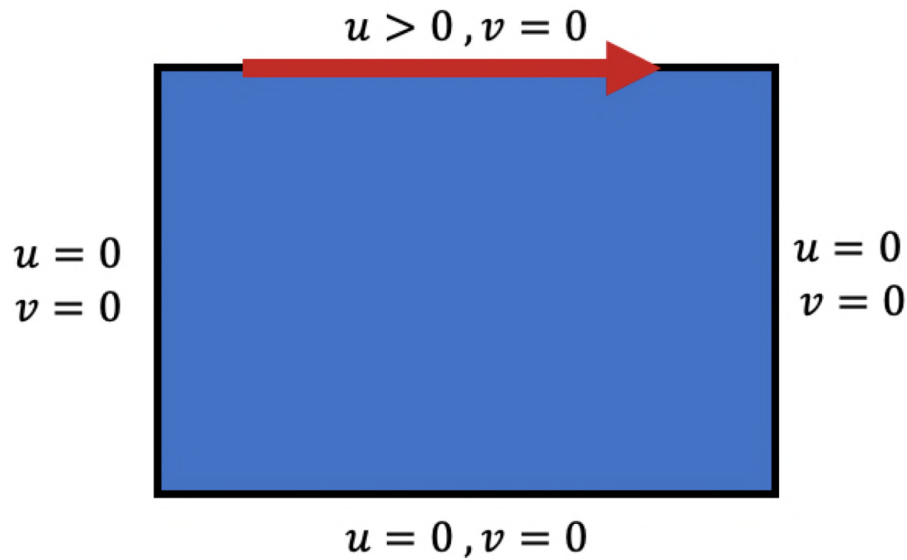


*Simulation Data: Re=4000*

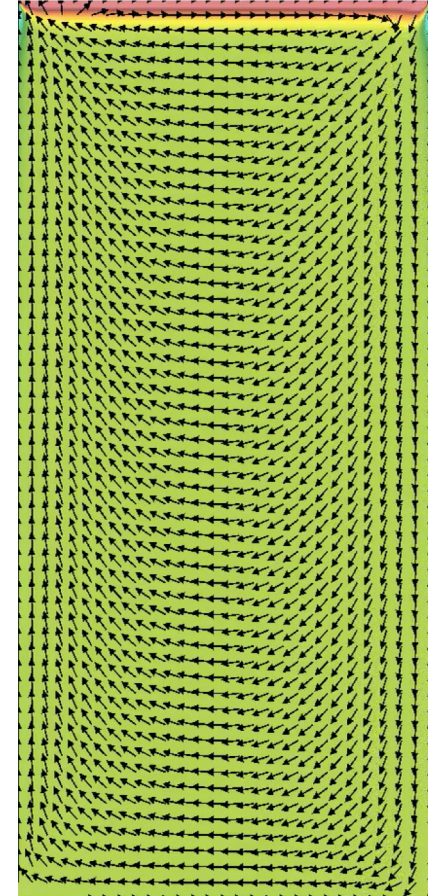
# Projection Methods: Cavity Flow



Cavity Flow Boundary Conditions



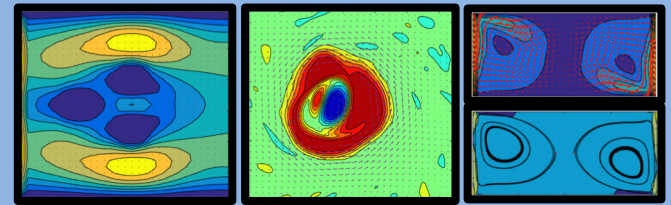
$Re = 4$



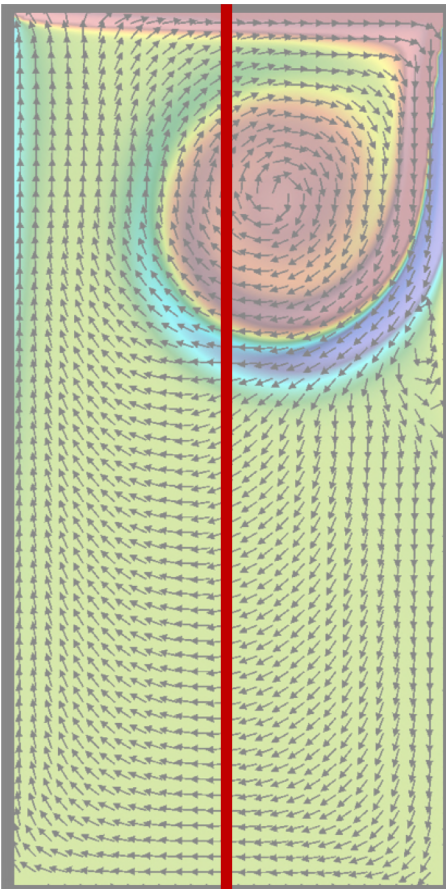
$Re = 4000$



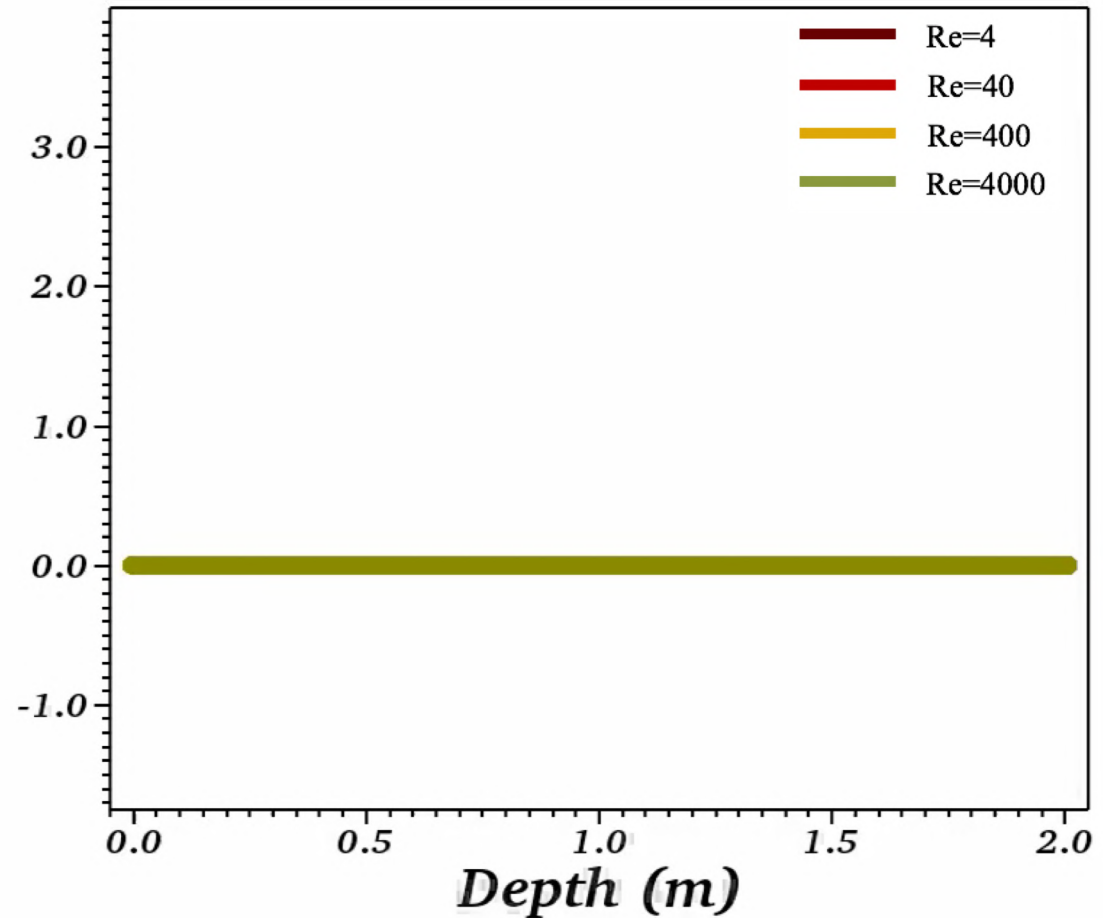
# Projection Methods: Cavity Flow



Measure **horizontal velocity**  
across *center of domain* for  
different Reynolds Numbers



Horizontal Velocity (m/s)







## PROJECTION METHODS

Examples:

1. Cavity Flow in Rectangular Domain

**2. Circular Flow in Square Domain**

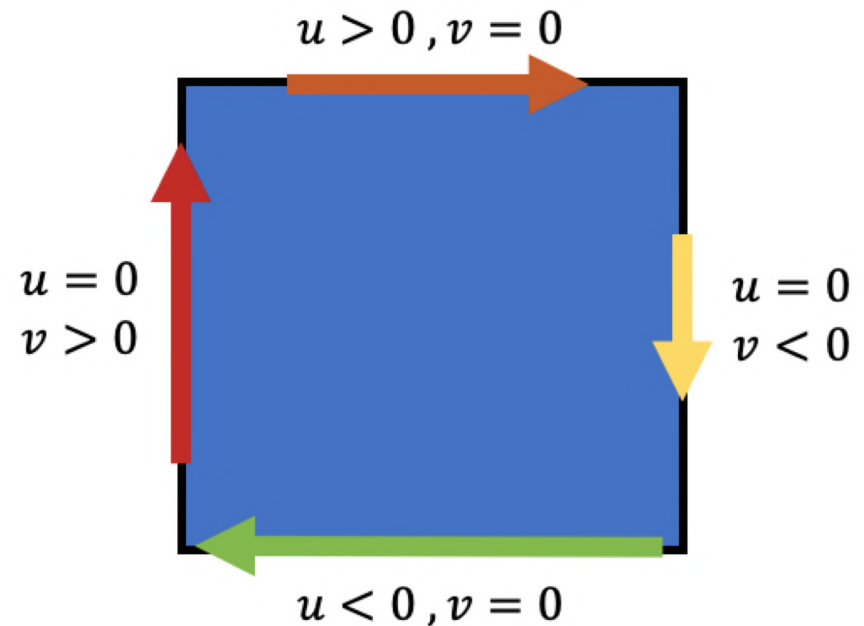
*\*Overview of Numerical Scheme to follow\**

# Projection Methods: Circular Flow

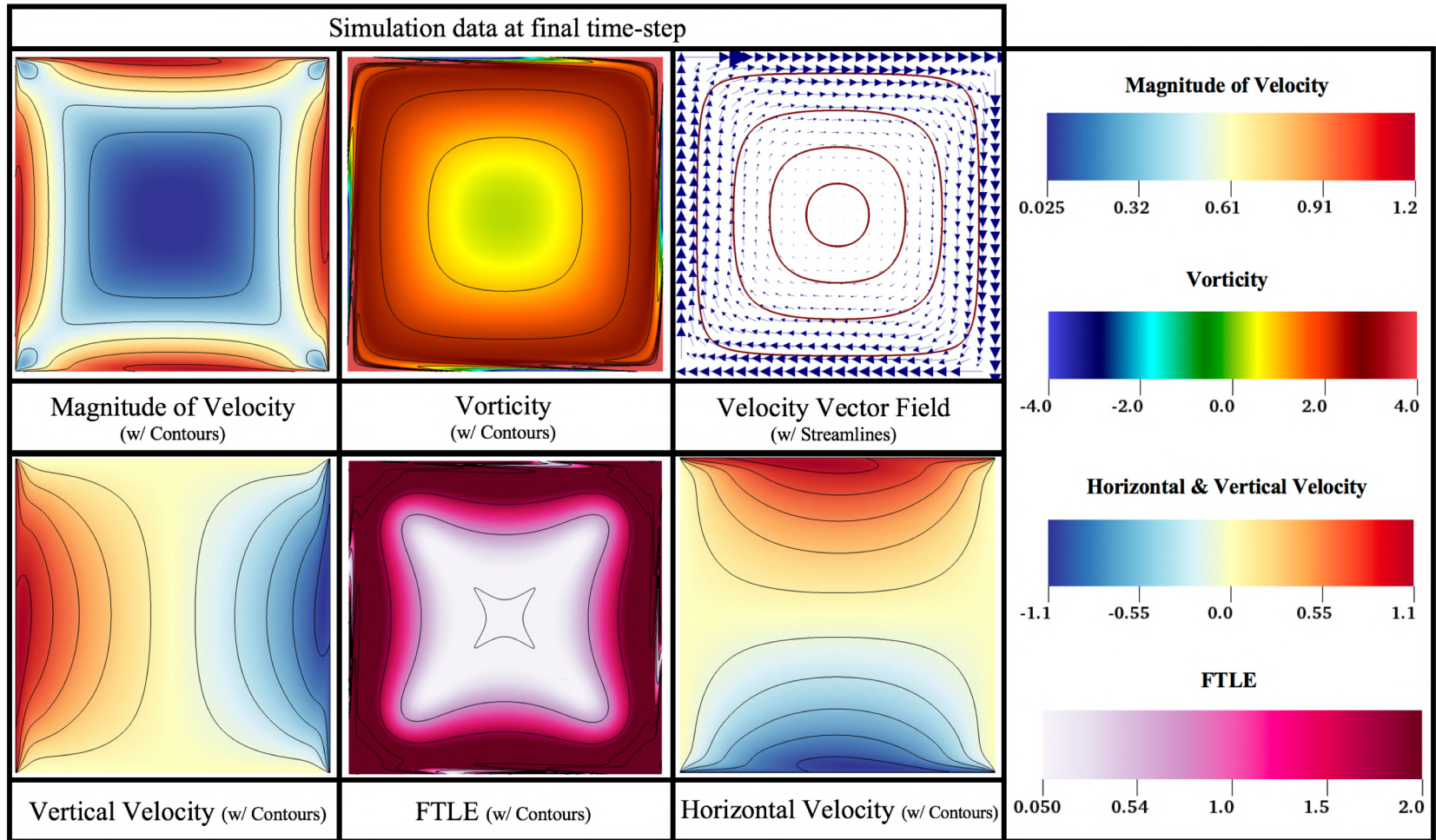
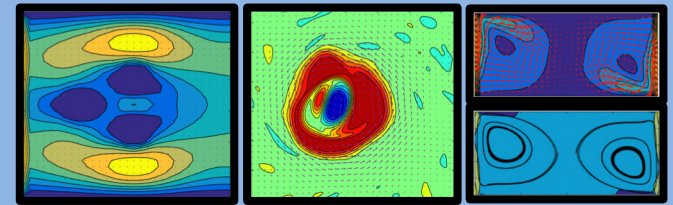


- Flow speed is ramped up across all sides of domain. Flow on each side is uniform in magnitude.
- Flow direction is **clockwise** around domain
- Observe qualitative differences due to differing Reynolds Numbers
- Measure horizontal velocity across middle of domain

Circular Flow Boundary Conditions

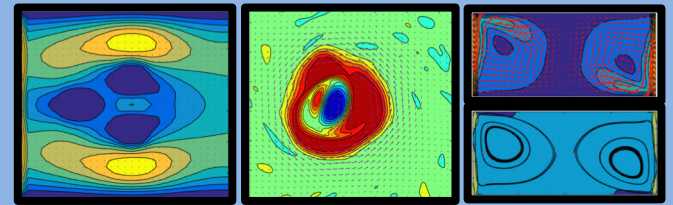


# Projection Methods: Circular Flow



$Re = 4000$

# Projection Methods: Circular Flow



## Magnitude of Velocity



$Re = 400$



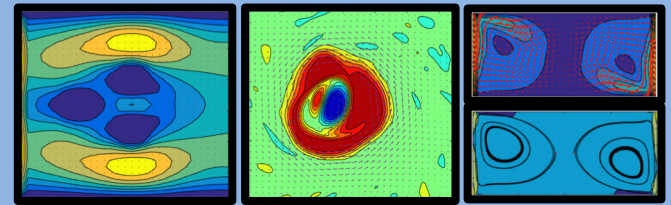
$Re = 1000$



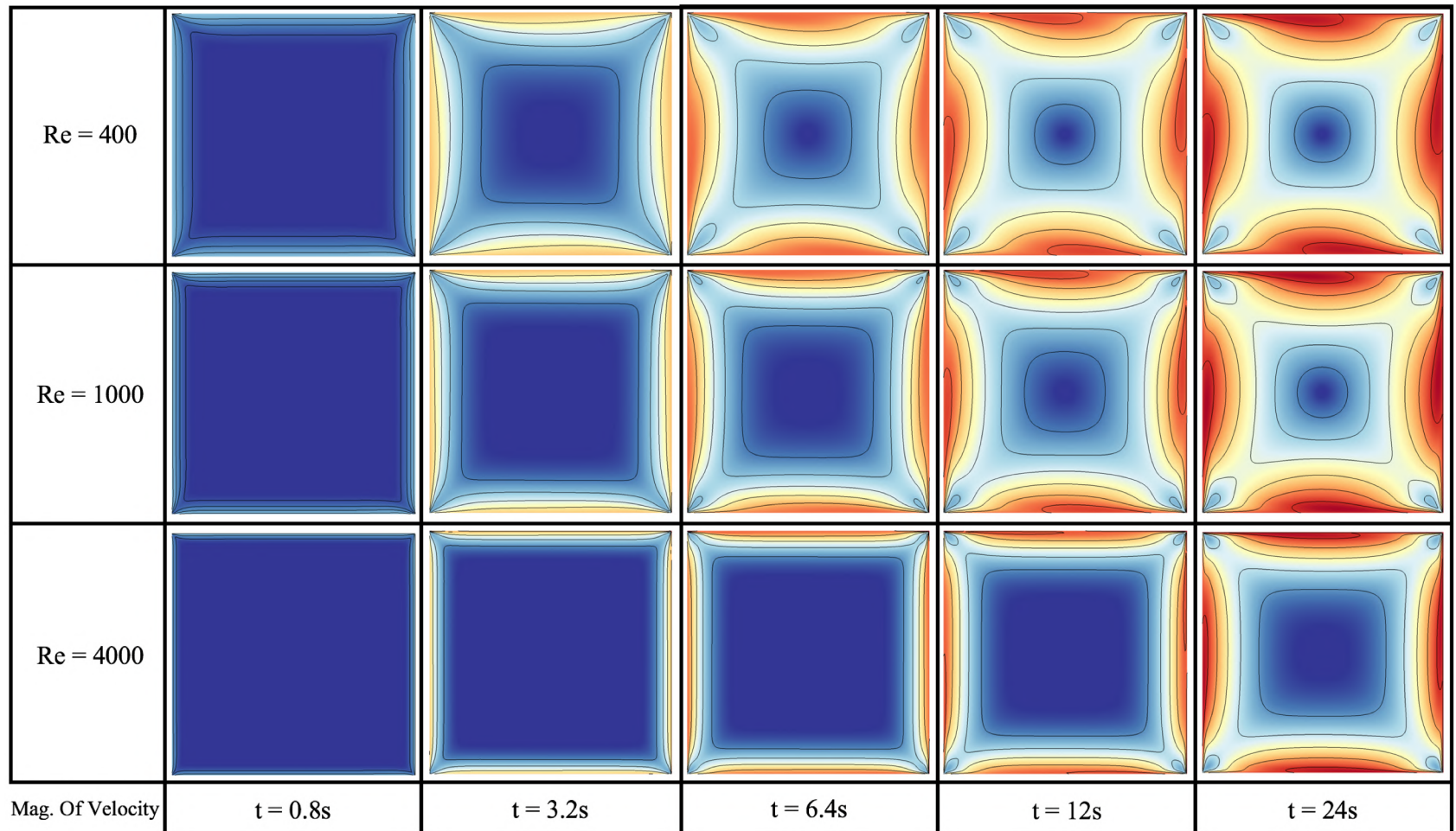
$Re = 4000$



# Projection Methods: Circular Flow



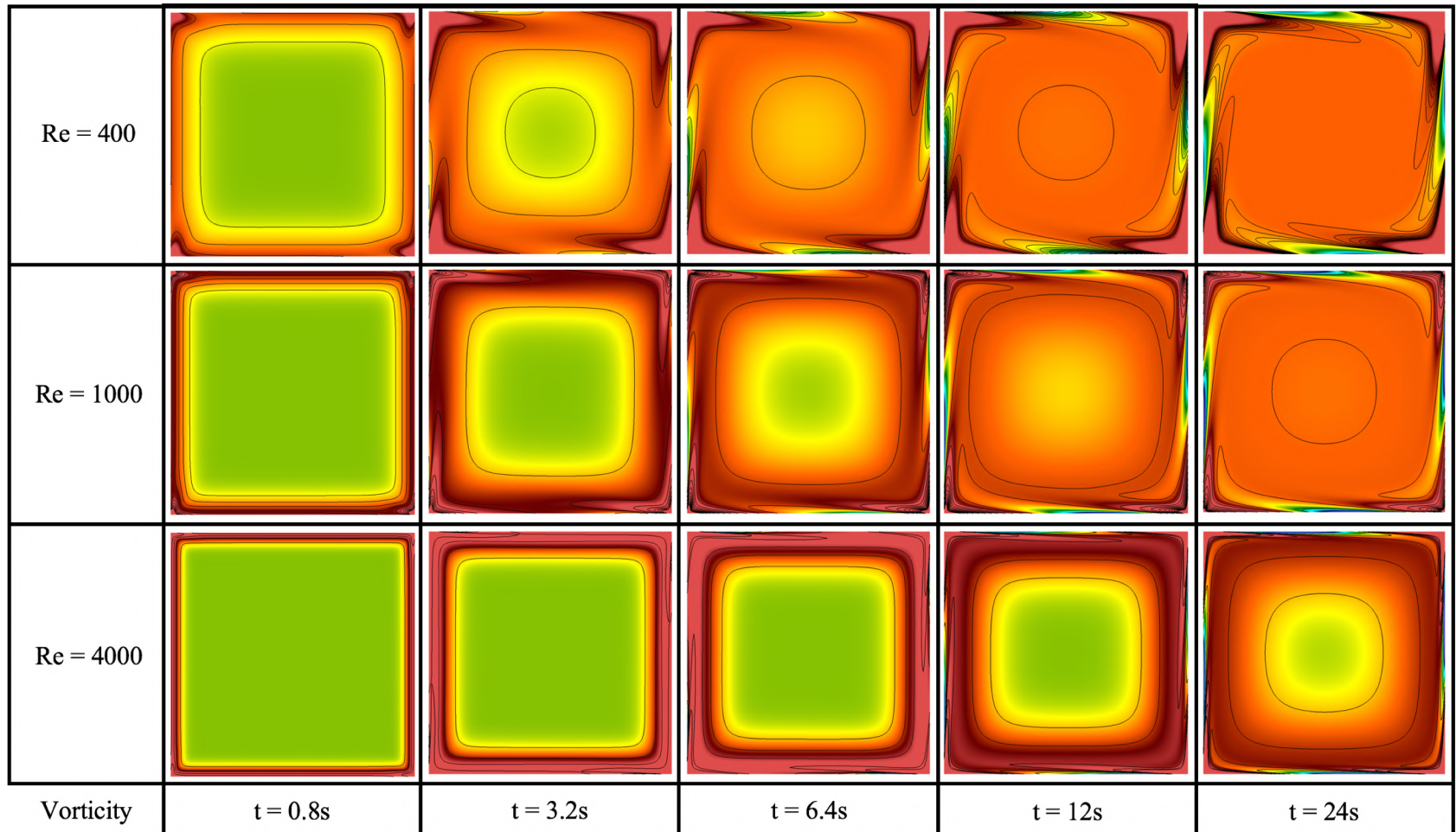
## Magnitude of Velocity



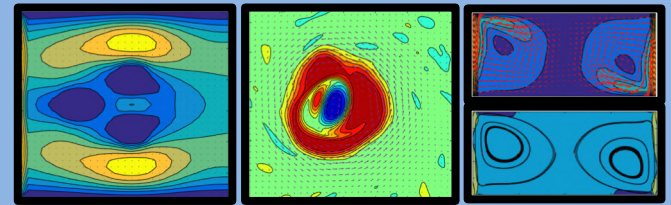
# Projection Methods: Circular Flow



## Vorticity

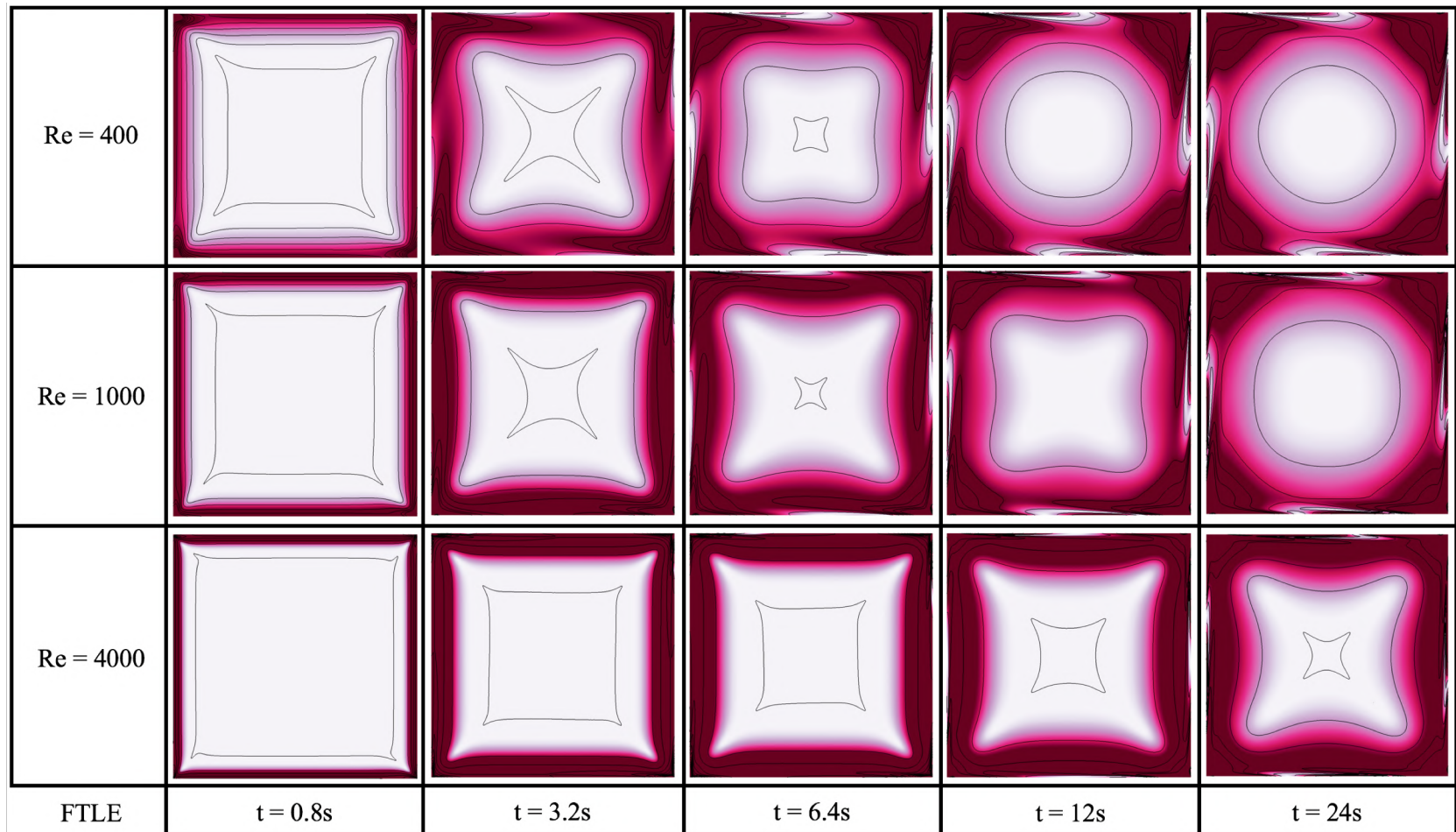


# Projection Methods: Circular Flow



## FTLE: finite-time Lyapunov exponents

(maximal ridges provide information regarding Lagrangian Coherent Structures)

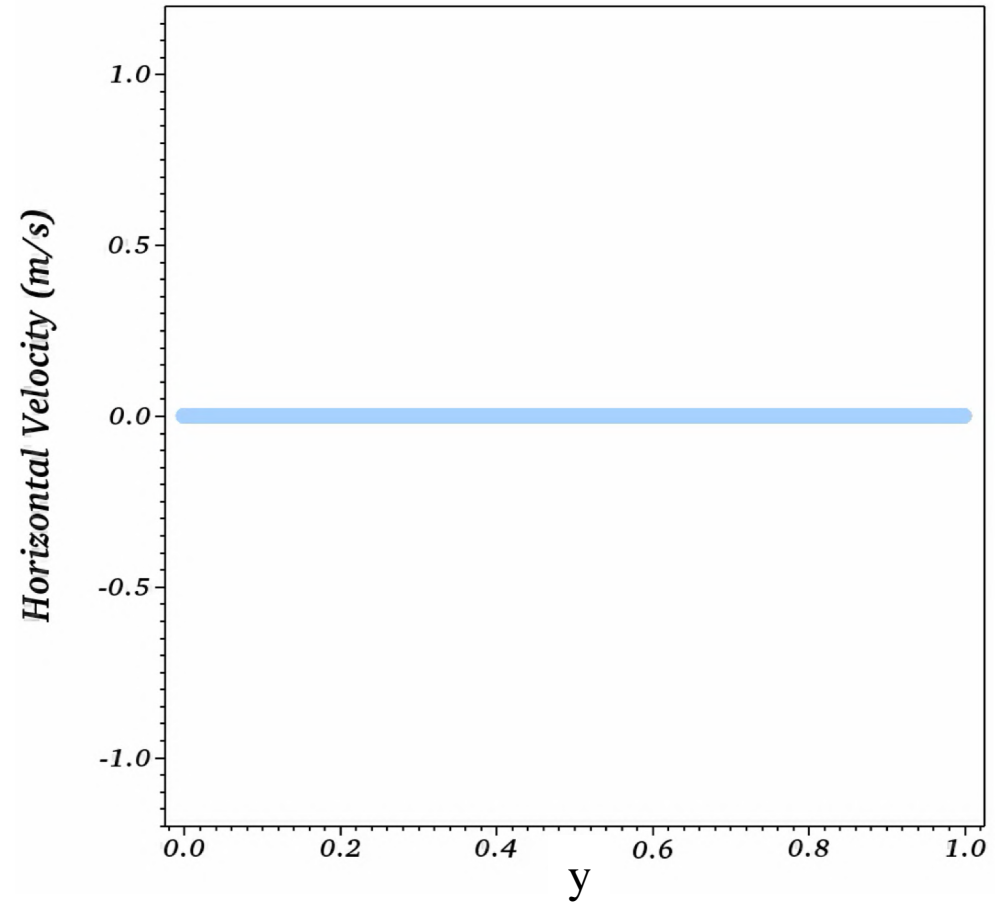
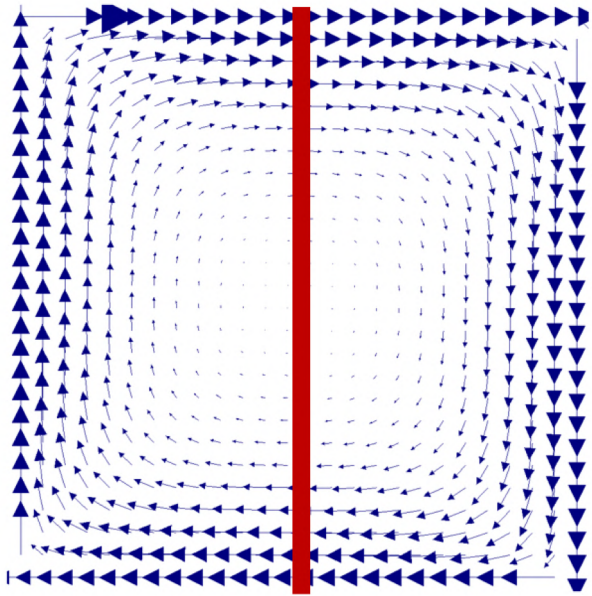




# Projection Methods: Circular Flow

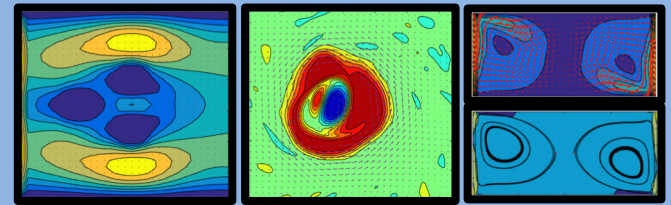


*Horizontal Velocity* measured  
along vertical line in across middle of domain





# Projection Methods: Circular Flow



## Diffusion Time Scales!

The **viscous diffusion time** can be thought of as the *time* it takes for a *fluid parcel to diffuse a particular distance on average*

Mean squared distance

Viscous Diffusion Time

$$\tilde{t} \sim \frac{\tilde{l}}{\nu} \sim \frac{1}{\nu}$$

Kinematic Viscosity

# Projection Methods: Circular Flow



## Diffusion Time Scales!

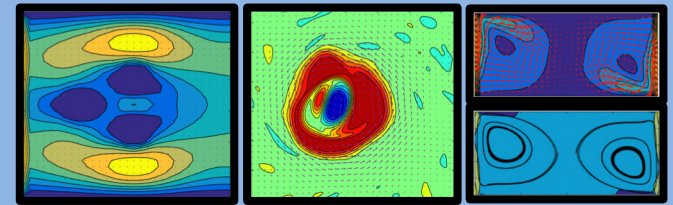
The **viscous diffusion time** can be thought of as the *time* it takes for a *fluid parcel to diffuse a particular distance on average*

$$\tilde{t}_{Re=400} \sim \frac{1}{2.5/1000} = 400 \text{ s}$$

$$\tilde{t}_{Re=4000} \sim \frac{1}{0.25/1000} = 4000 \text{ s}$$

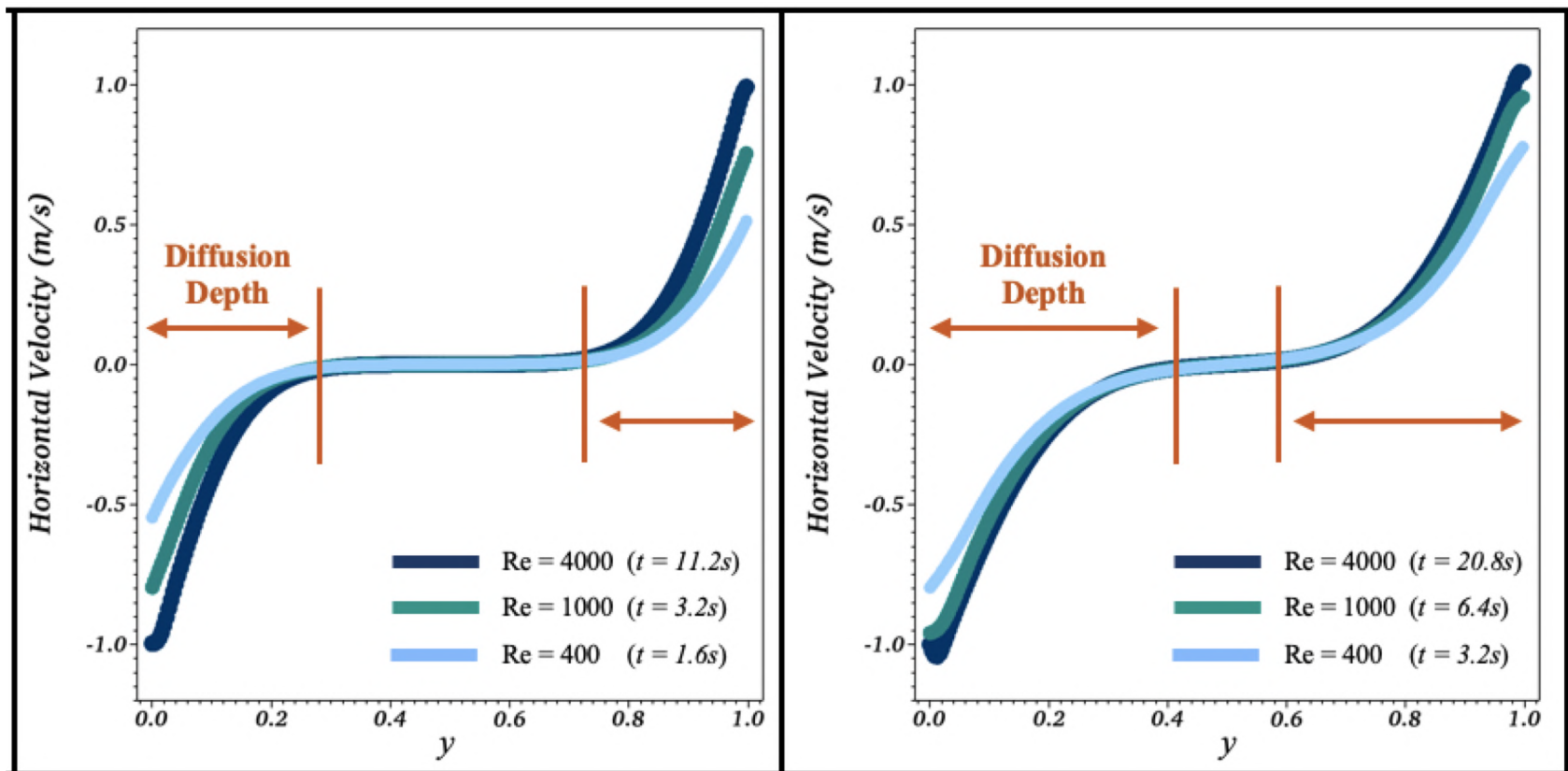
Dynamics evolve  
10x slower towards  
middle of domain in  
the Re=4000 case!

# Projection Methods: Circular Flow

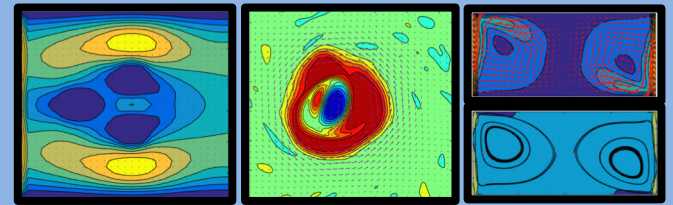


## Diffusion Time Scales!

The **viscous diffusion time** can be thought of as the *time* it takes for a *fluid parcel* to *diffuse* a *particular distance* on *average*



(look at the time in which these depths are achieved for each case)



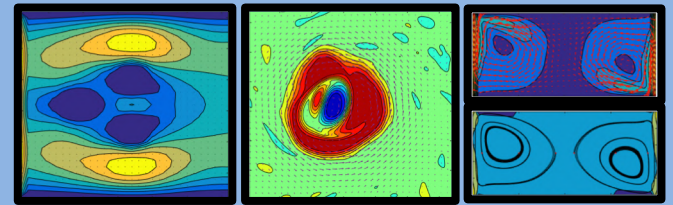
## PROJECTION METHODS

Examples:

1. Cavity Flow in Rectangular Domain
2. Circular Flow in Square Domain

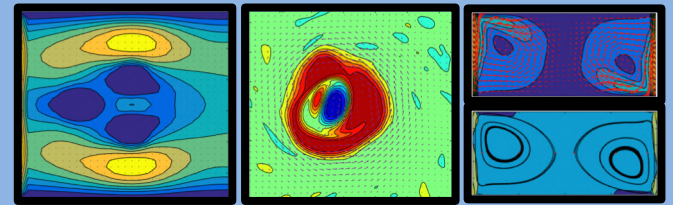
***\*Overview of Numerical Scheme to follow\****

# Projection Methods



- Use Operator Splitting + Helmholtz-Hodge Decomp.

# Projection Methods



- Use **Operator Splitting** + Helmholtz-Hodge Decomp.

1. Compute auxiliary velocity -> IGNORE the pressure terms

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla) \mathbf{u}^n + \nu \Delta \mathbf{u}^n$$

2. “Projection Step”

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho} \nabla p^{n+1}$$



# Projection Methods



- Use **Operator Splitting** + Helmholtz-Hodge Decomp.

1. Compute auxiliary velocity -> IGNORE the pressure terms

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla) \mathbf{u}^n + \nu \Delta \mathbf{u}^n$$

2. “Projection Step”

NEXT STEP PRESSURE...  
HOW TO GET IT?!

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho} \nabla p^{n+1}$$

# Projection Methods



- Use Operator Splitting + **Helmholtz-Hodge Decomp.**

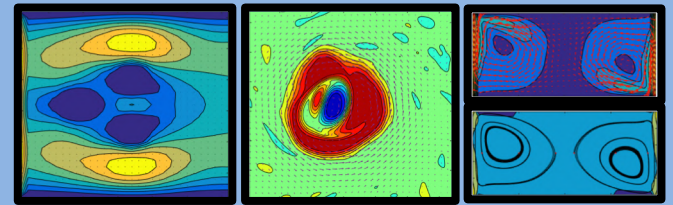
$$\mathbf{V} = \mathbf{V}_{sol} + \mathbf{V}_{irr} = \mathbf{V}_{sol} + \nabla\phi$$

$$\nabla \cdot \mathbf{V} = \Delta\phi$$

$$\mathbf{V}_{sol} = \mathbf{V} - \nabla\phi$$

$$\mathbf{u}_{incompressible} = \mathbf{u}' - \nabla\phi$$

# Projection Methods



- Use Operator Splitting + **Helmholtz-Hodge Decomp.**

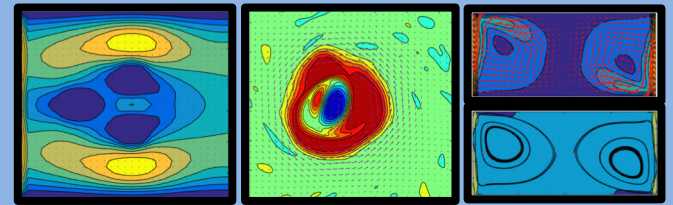
$$\mathbf{V} = \mathbf{V}_{sol} + \mathbf{V}_{irr} = \mathbf{V}_{sol} + \nabla\phi$$

$$\nabla \cdot \mathbf{V} = \Delta\phi$$

$$\mathbf{V}_{sol} = \mathbf{V} - \nabla\phi$$

$$\mathbf{u}_{incompressible} = \mathbf{u}' - \nabla\phi$$

# Projection Methods



- Use Operator Splitting + **Helmholtz-Hodge Decomp.**

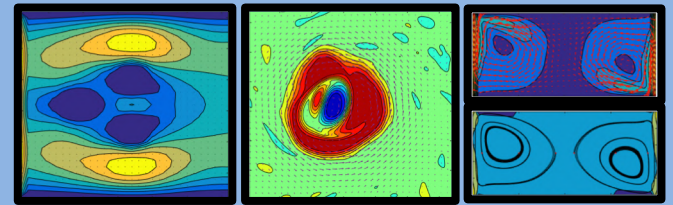
$$\mathbf{V} = \mathbf{V}_{sol} + \mathbf{V}_{irr} = \mathbf{V}_{sol} + \nabla\phi$$

$$\nabla \cdot \mathbf{V} = \Delta\phi$$

$$\mathbf{V}_{sol} = \mathbf{V} - \nabla\phi$$

$$\mathbf{u}_{incompressible} = \mathbf{u}' - \nabla\phi$$

# Projection Methods



- Use Operator Splitting + **Helmholtz-Hodge Decomp.**

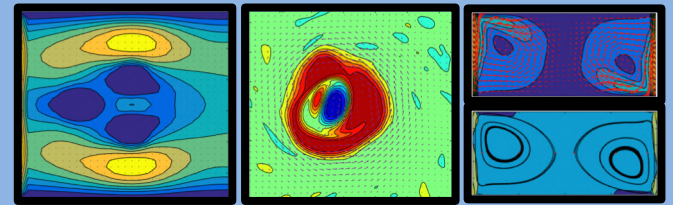
2. “Projection Step”

$$\nabla \cdot (\mathbf{u}^{n+1} - \mathbf{u}^*) = -\frac{\Delta t}{\rho} \nabla \cdot \nabla p^{n+1}$$

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

$$\frac{\Delta t}{\rho} \Delta p^{n+1} = \nabla \cdot \mathbf{u}^*$$

# Projection Methods



- Use Operator Splitting + **Helmholtz-Hodge Decomp.**

2. “Projection Step”

$$\nabla \cdot (\mathbf{u}^{n+1} - \mathbf{u}^*) = -\frac{\Delta t}{\rho} \nabla \cdot \nabla p^{n+1}$$

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

$$\frac{\Delta t}{\rho} \Delta p^{n+1} = \nabla \cdot \mathbf{u}^*$$



# Projection Methods



- Use Operator Splitting + **Helmholtz-Hodge Decomp.**

2. “Projection Step”

$$\nabla \cdot (\mathbf{u}^{n+1} - \mathbf{u}^*) = -\frac{\Delta t}{\rho} \nabla \cdot \nabla p^{n+1}$$

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

$$\frac{\Delta t}{\rho} \Delta p^{n+1} = \nabla \cdot \mathbf{u}^*$$

# Projection Methods



- Use **Operator Splitting + Helmholtz-Hodge Decomp.**

1. Compute auxiliary velocity -> IGNORE the pressure terms

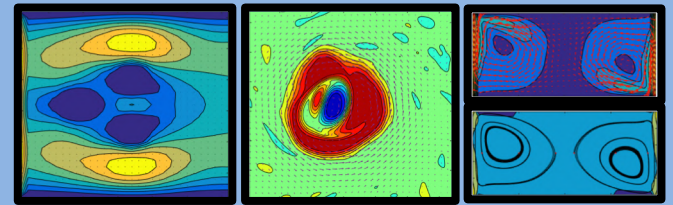
$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla) \mathbf{u}^n + \nu \Delta \mathbf{u}^n$$

2a. Elliptic Solve for Pressure Field

$$\frac{\Delta t}{\rho} \Delta p^{n+1} = \nabla \cdot \mathbf{u}^*$$

2b. Projection Step

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho} \nabla p^{n+1}$$



## Spectral Method (FFT)

Examples:

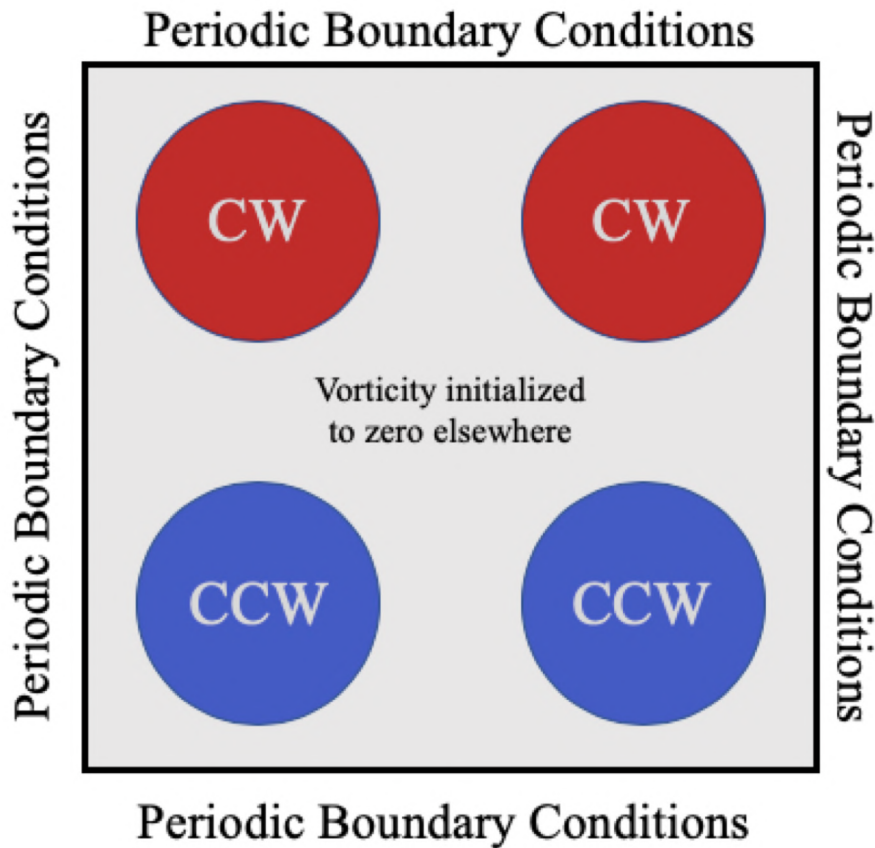
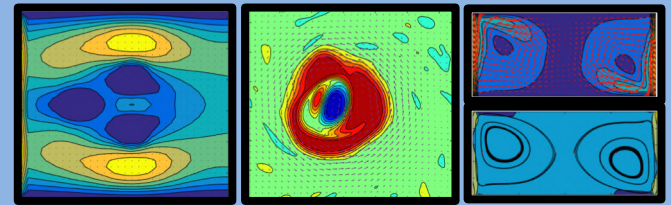
**1. Interacting Regions of Vorticity**

2. Overlapping Regions of Vorticity

3. Evolution of vorticity from an initial vector velocity field

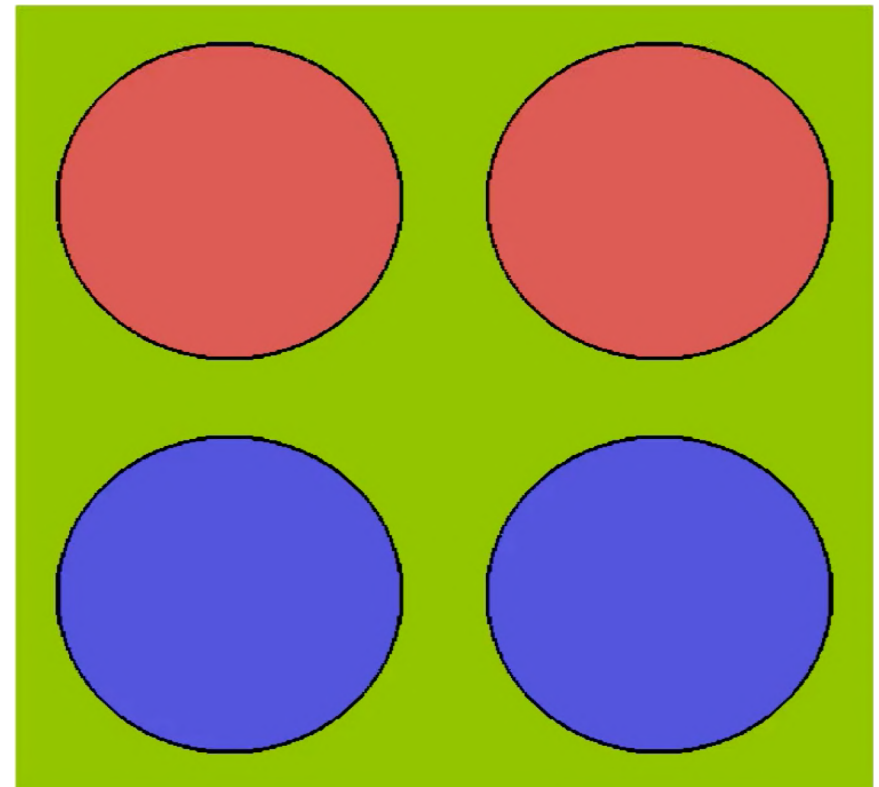
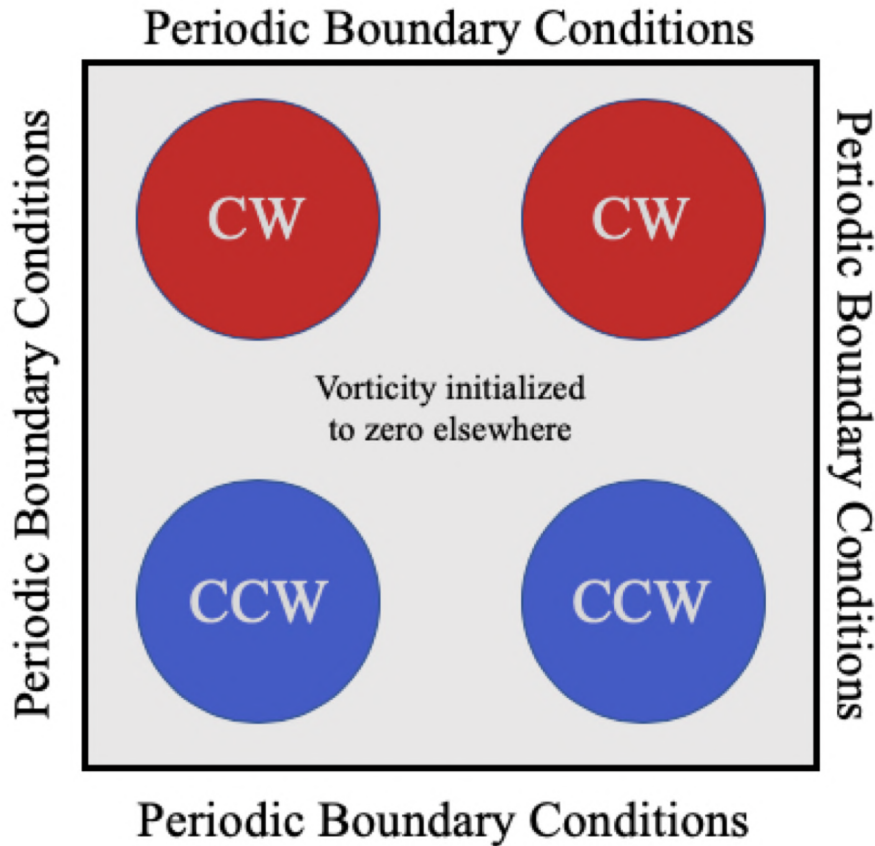
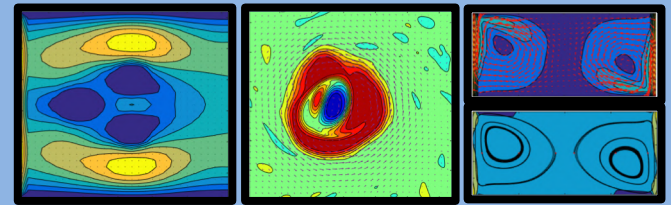
***\*Overview of Numerical Scheme to follow\****

# FFT: Interacting Regions



- Vorticity is initialized into each circular region uniformly, as either a positive or negative vorticity value
- CW = Positive initialization, CCW = Negative initialization
- Vorticity elsewhere is set to zero
- Periodic boundary conditions on all sides of domain

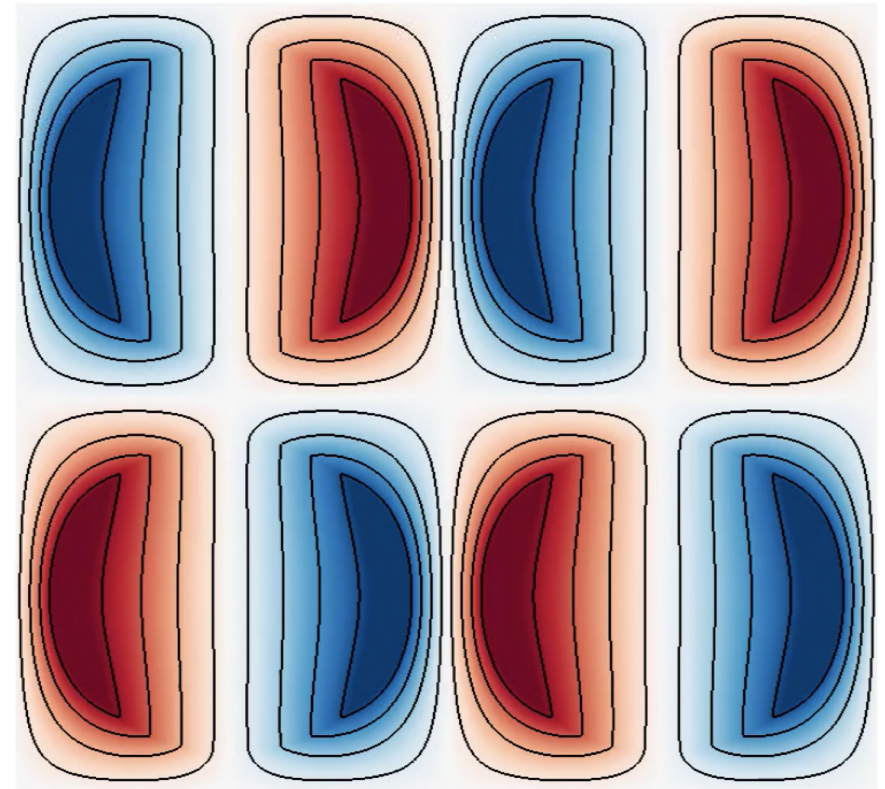
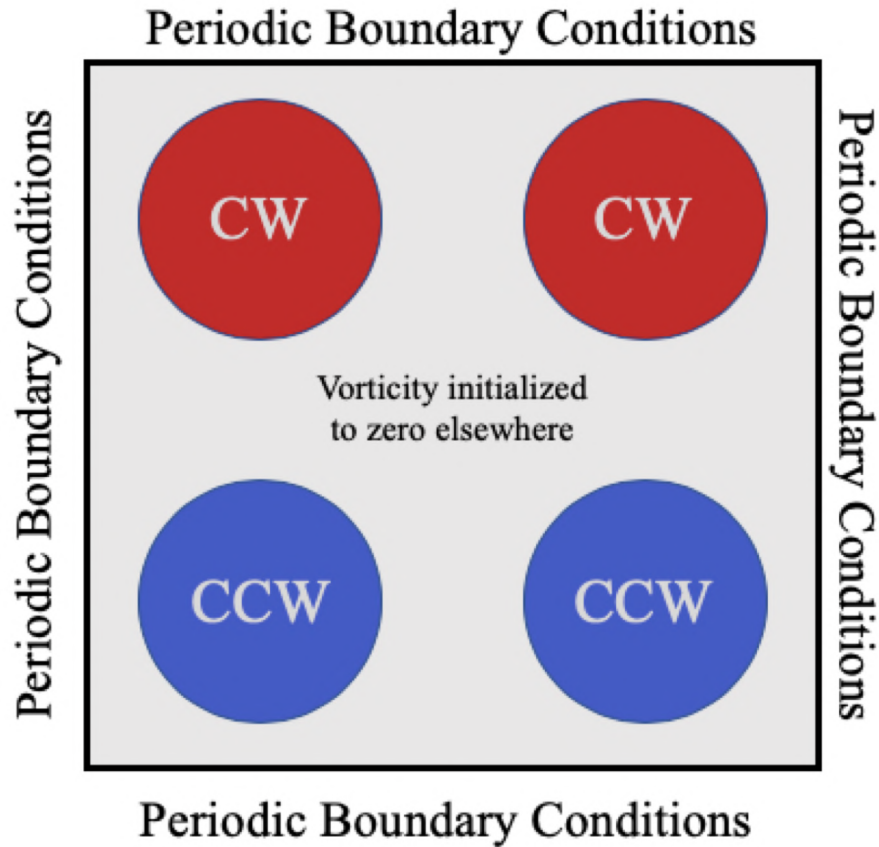
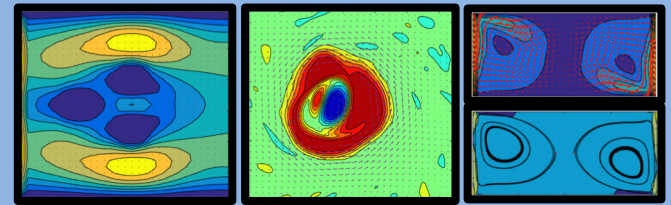
# FFT: Interacting Regions



*Visualization of Vorticity  
w/ Vorticity Contours*

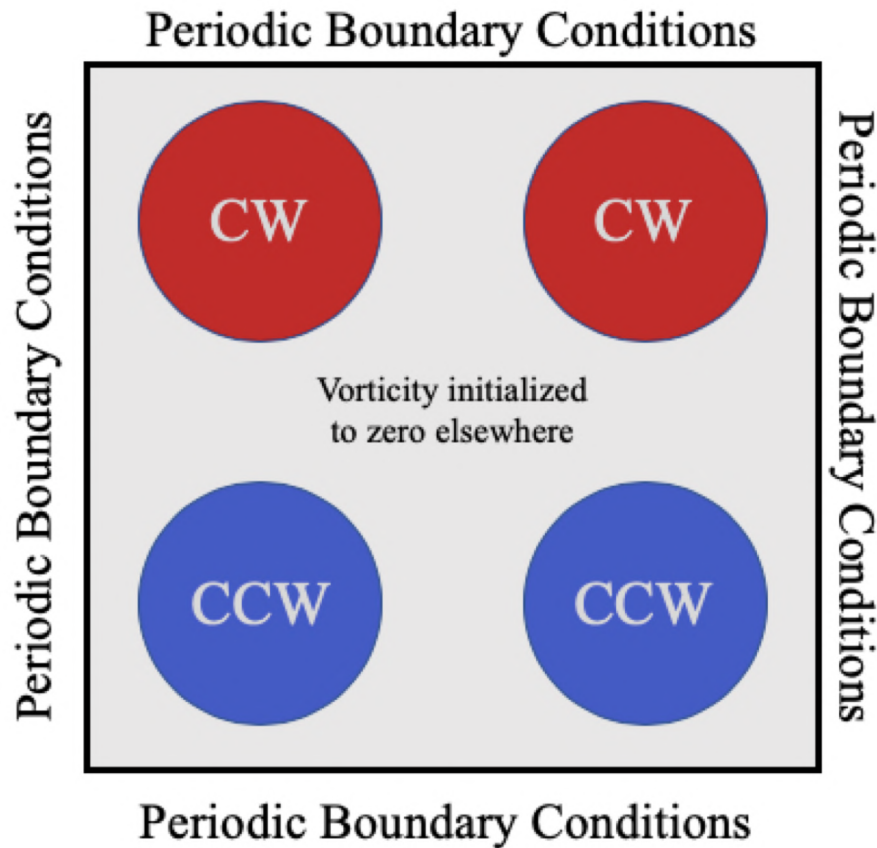
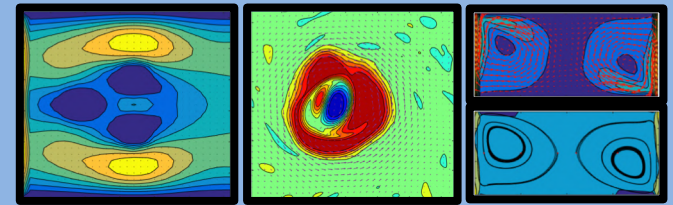


# FFT: Interacting Regions

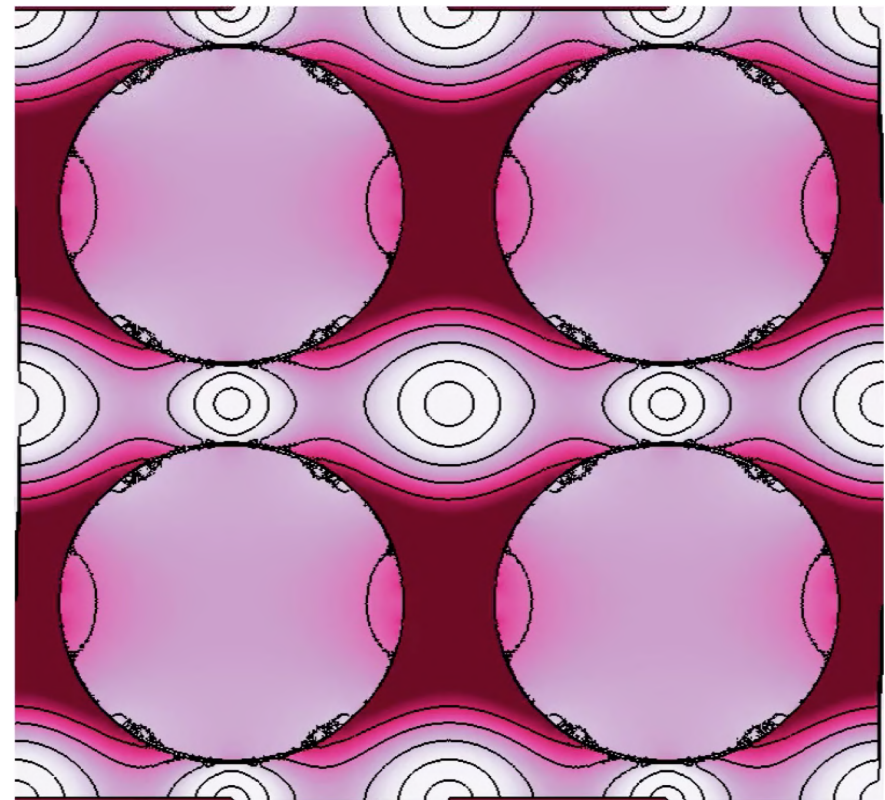


***Visualization of Vertical Velocity  
 $w$ / its Contours***

# FFT: Interacting Regions



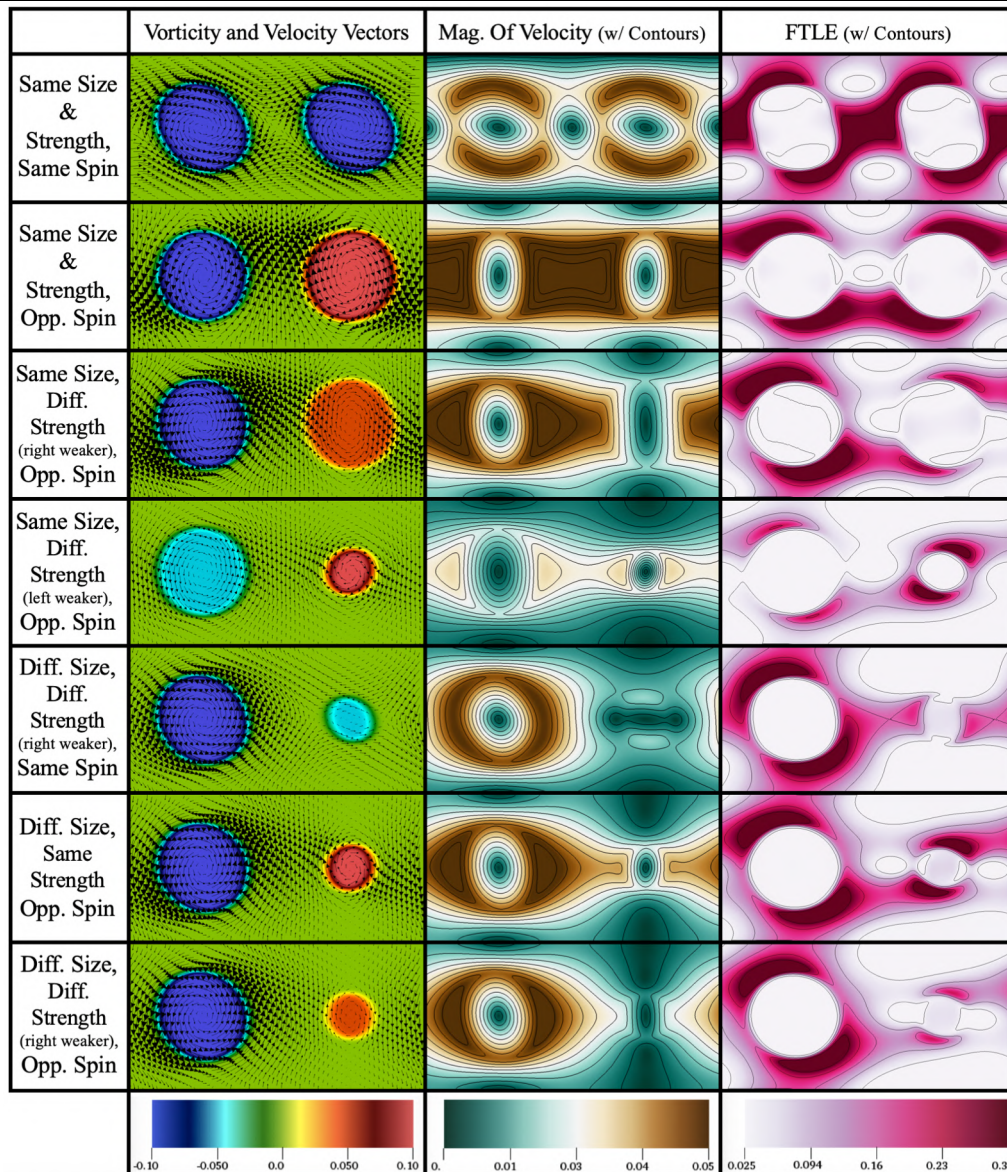
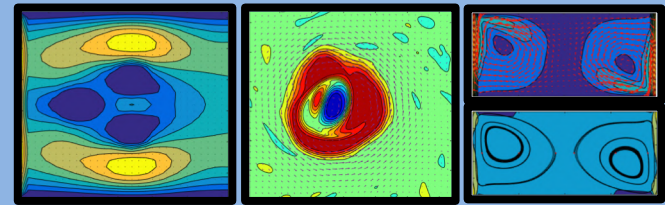
*High FTLE values can be thought to be regions of high fluid mixing*



***Visualization of FTLE  
w/ its Contours***



# FFT: Interacting Regions



Instead of 4 interacting regions, place 2 Side-By-Side and simulate all possible cases of:

1. Region size
2. Region's vorticity magnitude
3. Region's vorticity initialization value (both positive, both negative, mixed)



## Spectral Method (FFT)

Examples:

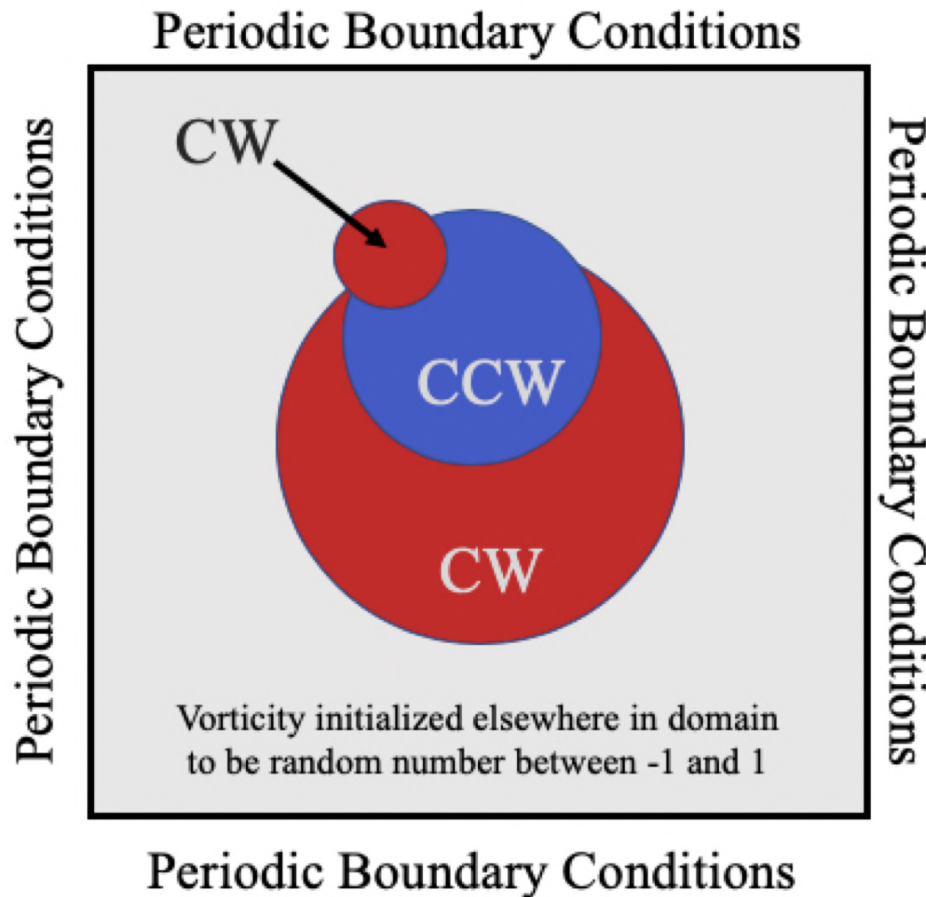
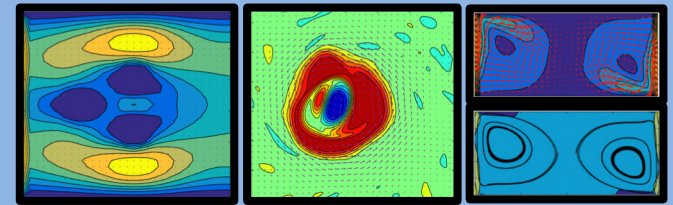
1. Interacting Regions of Vorticity

**2. Overlapping Regions of Vorticity**

3. Evolution of vorticity from an initial vector velocity field

*\*Overview of Numerical Scheme to follow\**

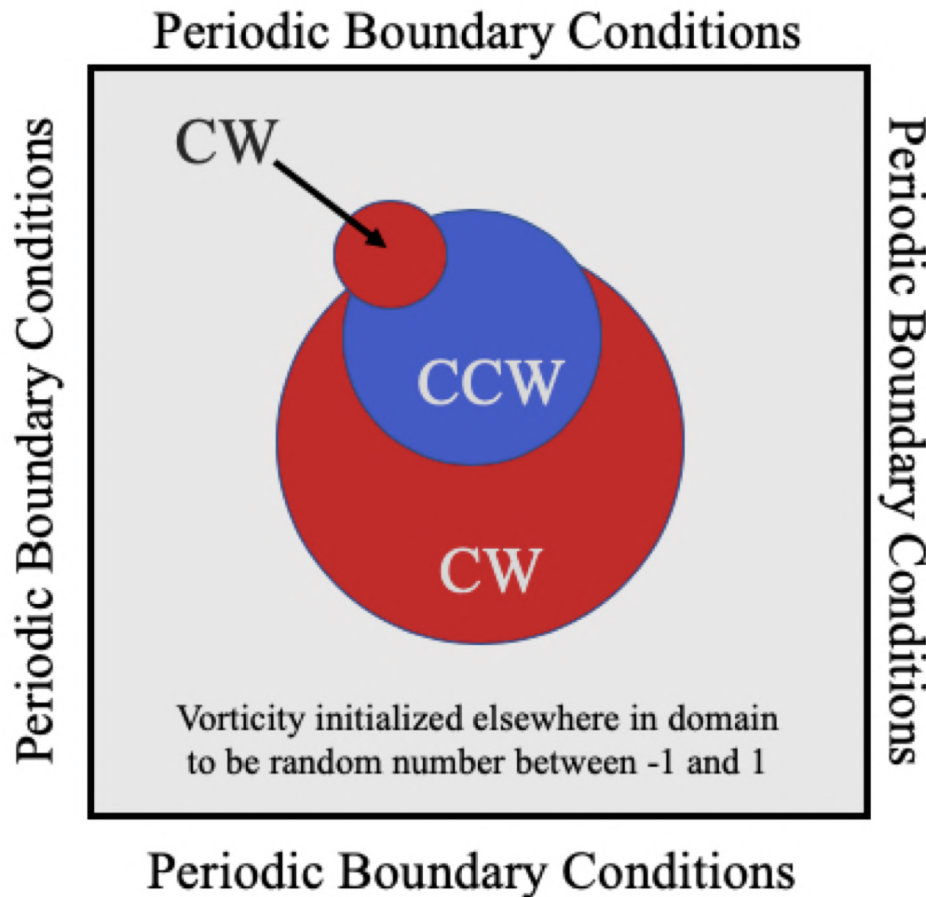
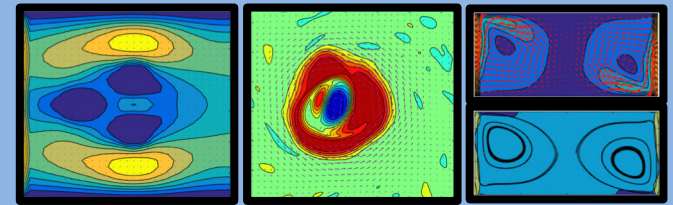
# FFT: Overlapping Regions



- Vorticity is initialized into each circular region uniformly, as either a positive or negative vorticity value
- CW = Positive initialization, CCW = Negative initialization
- Vorticity elsewhere is set a uniformly distributed random number between  $[-1,1]$
- Periodic boundary conditions on all sides of domain

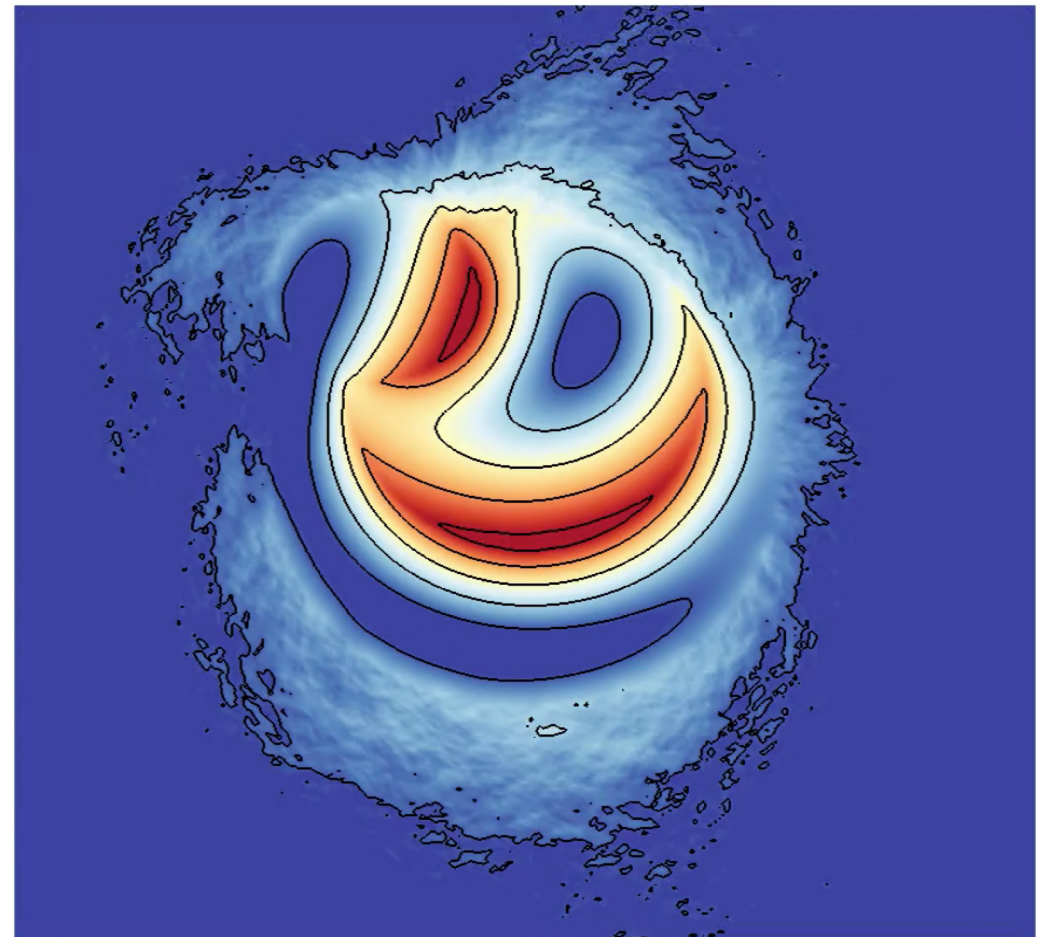
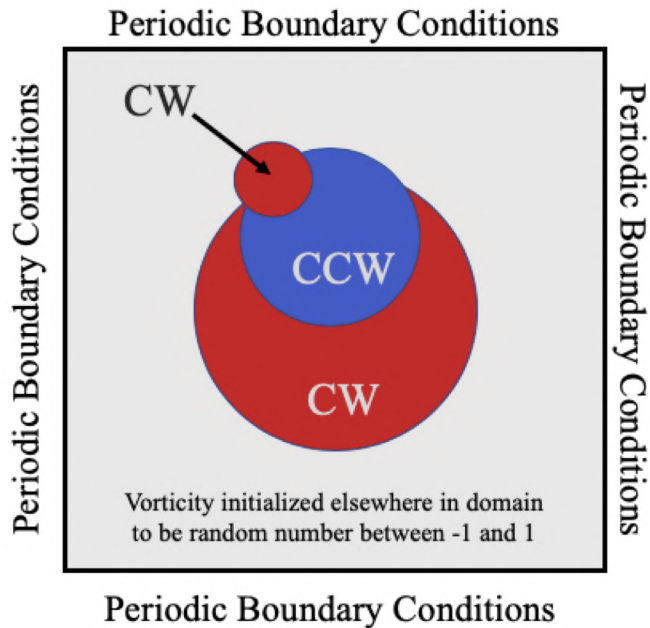
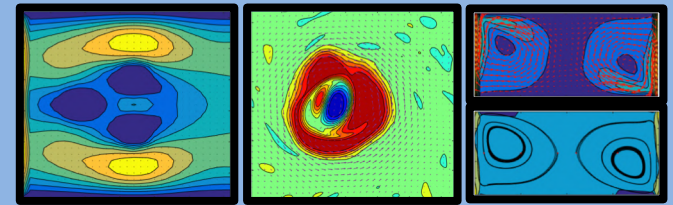


# FFT: Overlapping Regions



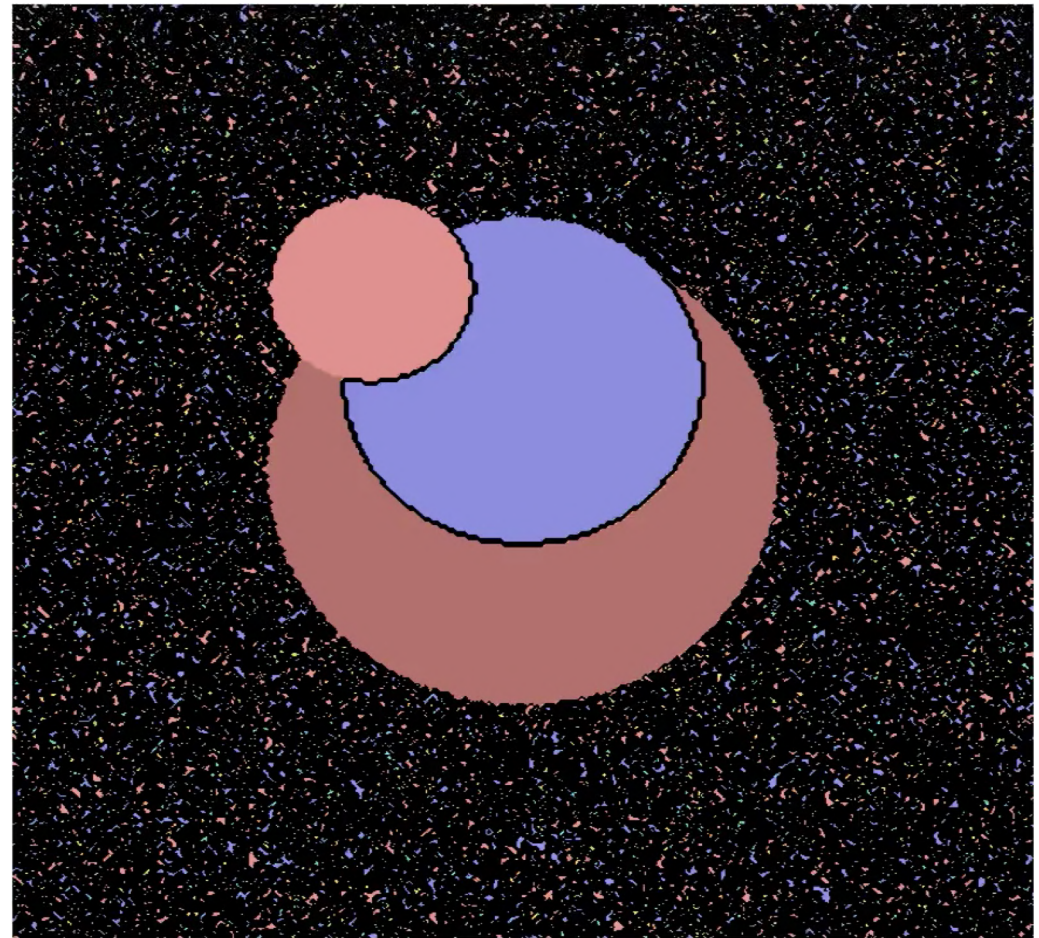
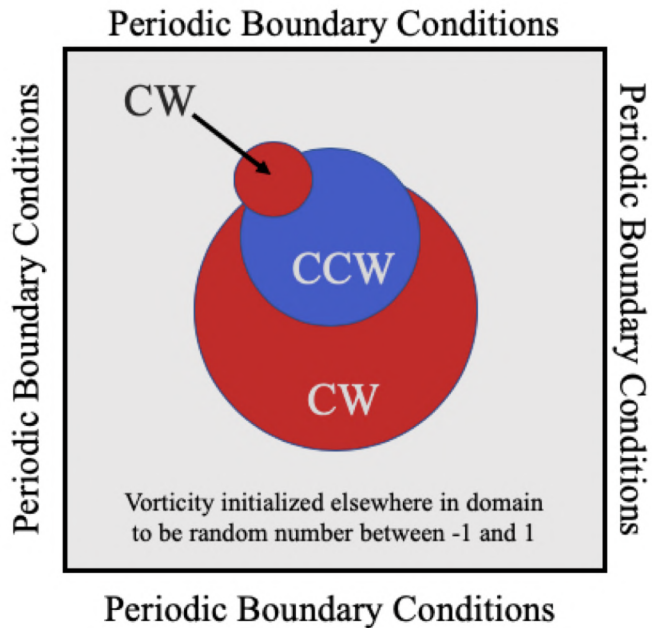
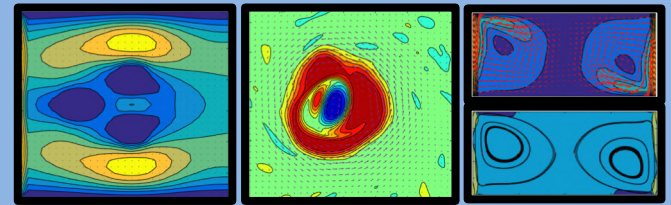
- Vorticity is initialized into each circular region uniformly, as either a positive or negative vorticity value
- CW = Positive initialization, CCW = Negative initialization
- Vorticity elsewhere is set a uniformly distributed random number between  $[-1,1]$
- Periodic boundary conditions on all sides of domain

# FFT: Overlapping Regions



*Visualization of Mag. of Velocity  $w$  / its Contours*

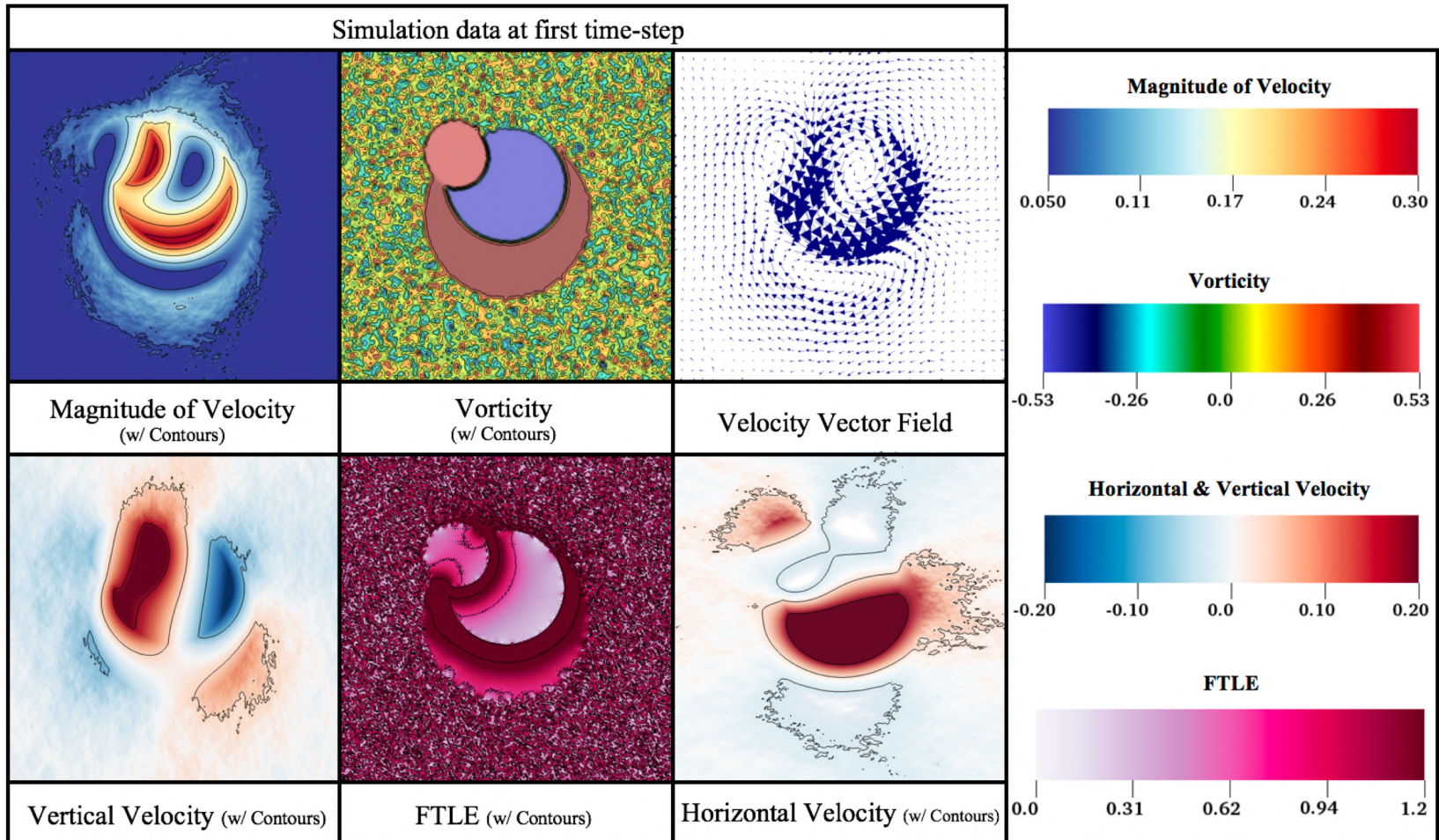
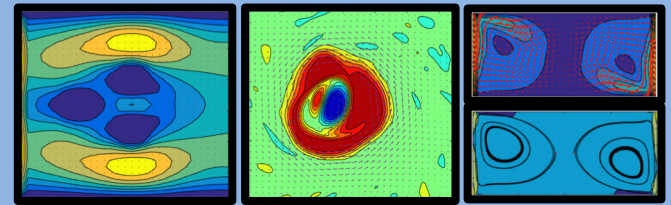
# FFT: Overlapping Regions



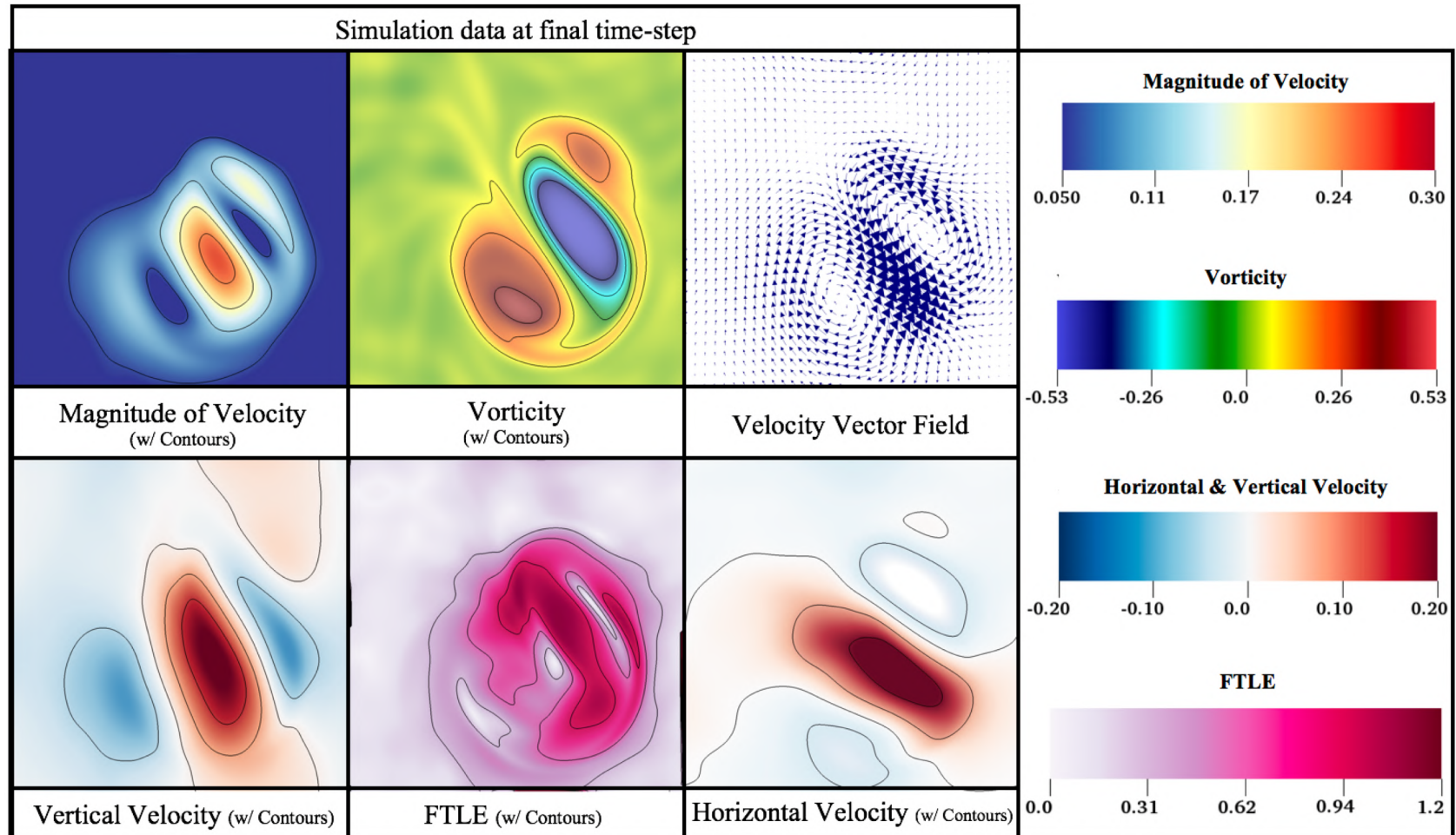
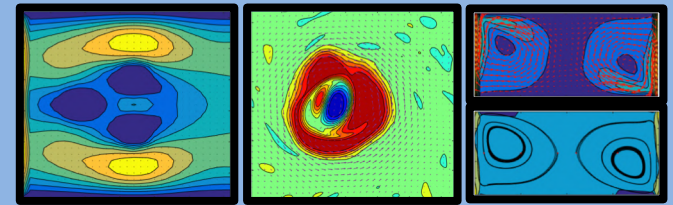
*Visualization of Vorticity  $w$  / its Contours*



# FFT: Overlapping Regions



# FFT: Overlapping Regions







## Spectral Method (FFT)

Examples:

1. Interacting Regions of Vorticity
2. Overlapping Regions of Vorticity

**3. Evolution of vorticity from an initial vector velocity field**

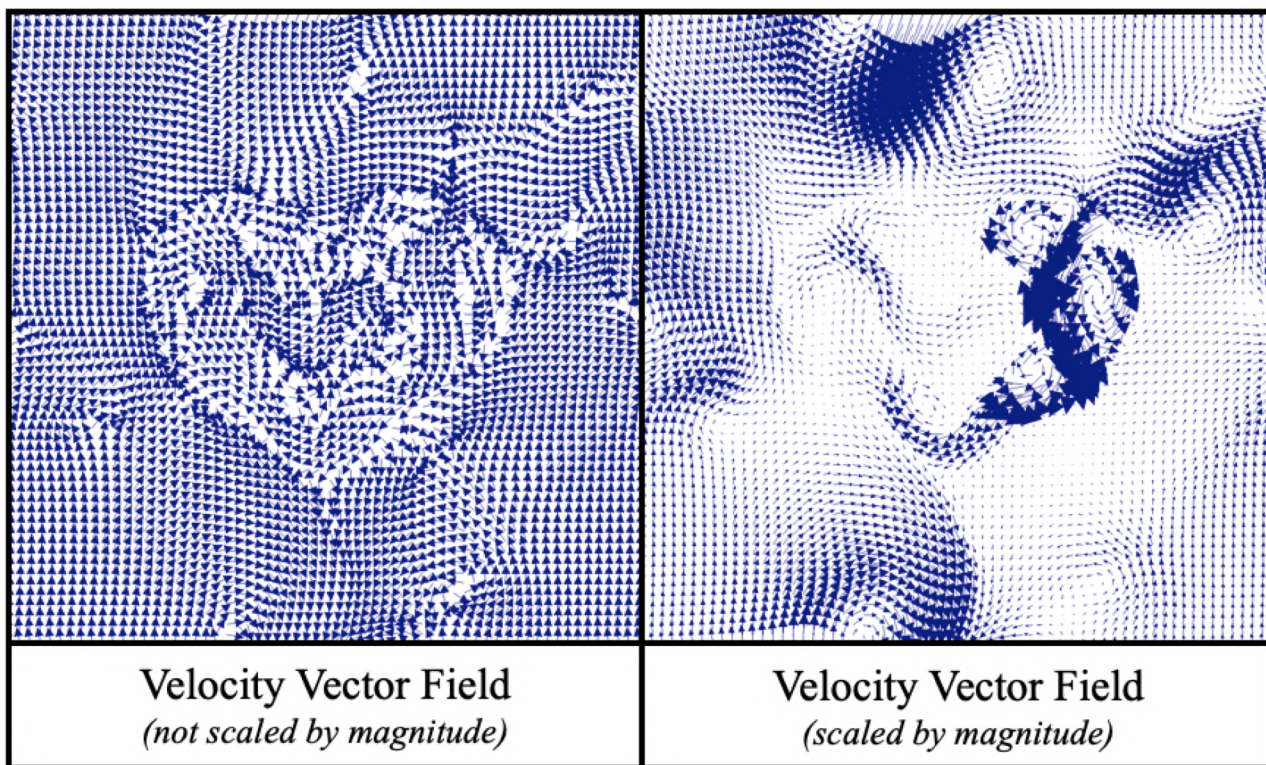
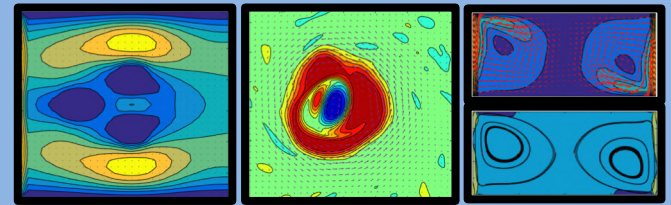
*\*Overview of Numerical Scheme to follow\**

# FFT: Evolve Vorticity from Initial Velocity Field



The **initial vorticity** for this simulation will be **computed from a velocity field** from an independent simulation and ***then evolved forward in time!***

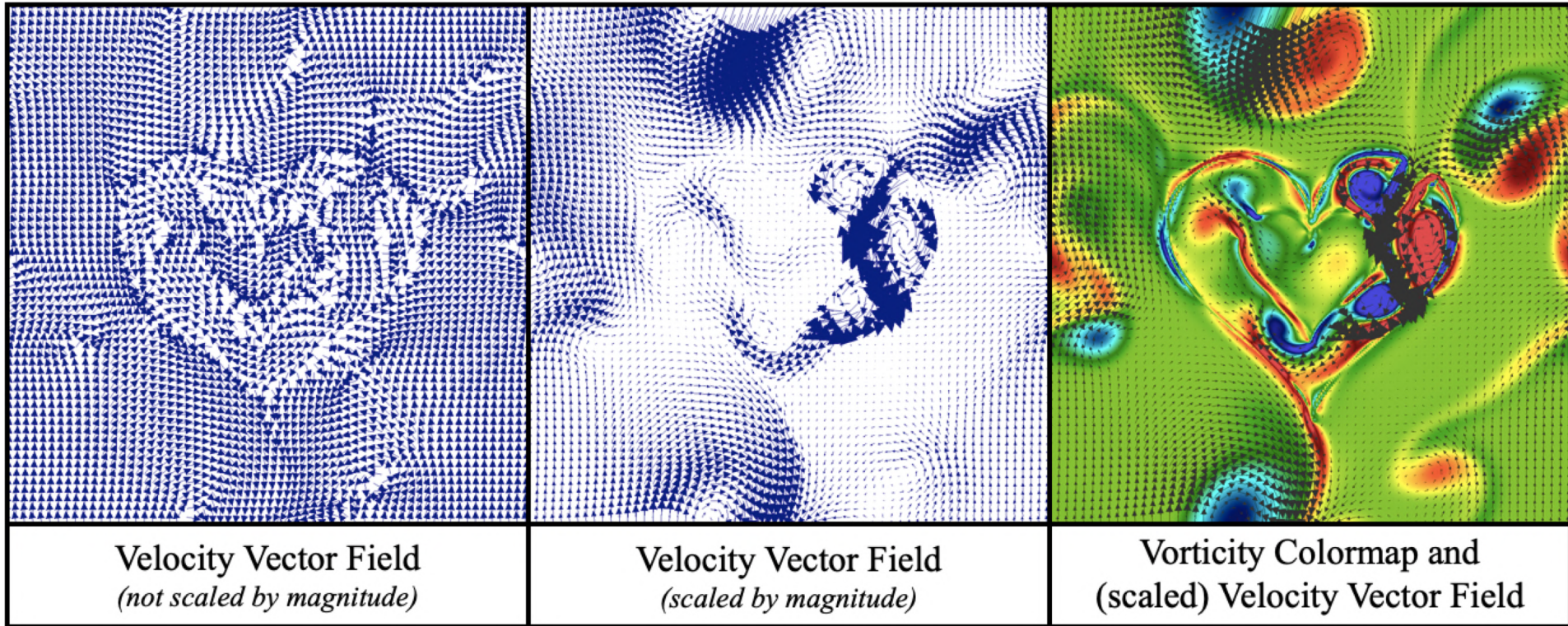
# FFT: Evolve Vorticity from Initial Velocity Field



**Initial Velocity Field taken from  
a timepoint for an independent  
fluid simulation**



# FFT: Evolve Vorticity from Initial Velocity Field



Velocity Vector Field  
(not scaled by magnitude)

Velocity Vector Field  
(scaled by magnitude)

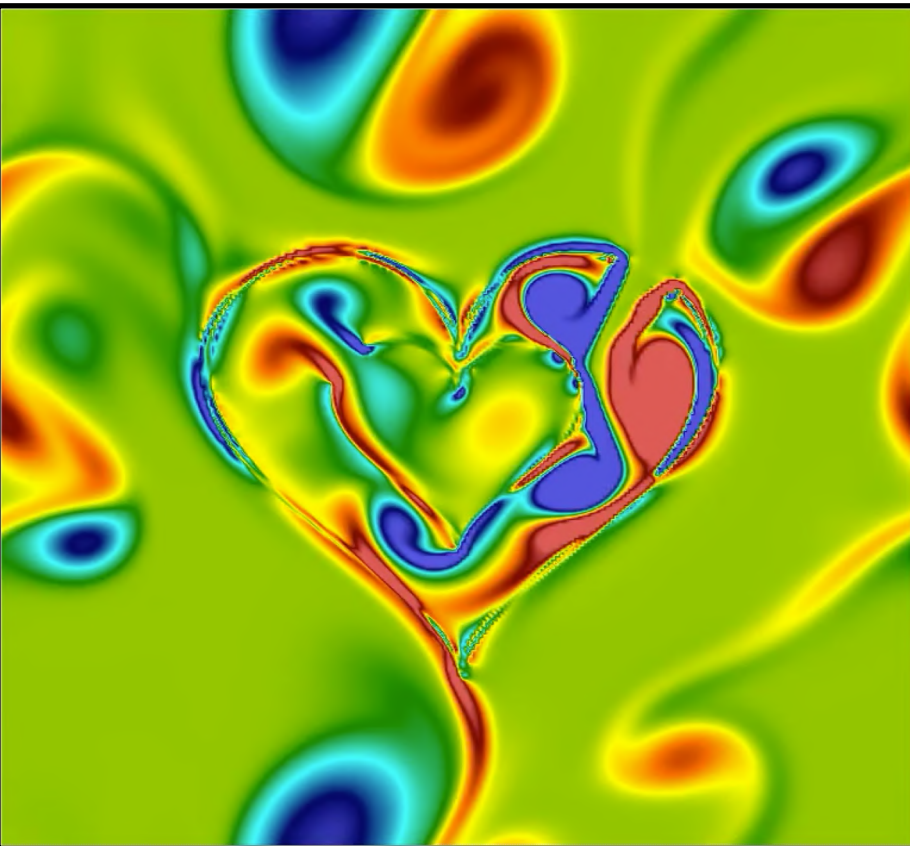
Vorticity Colormap and  
(scaled) Velocity Vector Field

**Initial Velocity Field taken from  
a timepoint for an independent  
fluid simulation**

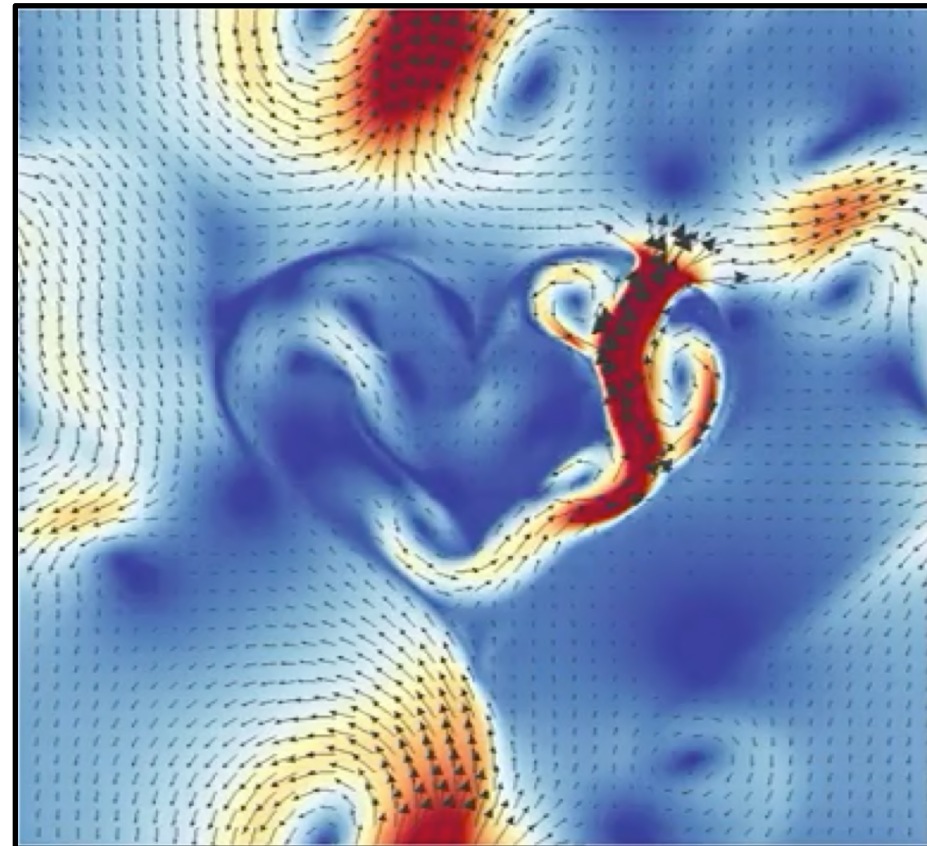
**Initial Vorticity  
associated with such  
velocity field**



# FFT: Evolve Vorticity from Initial Velocity Field



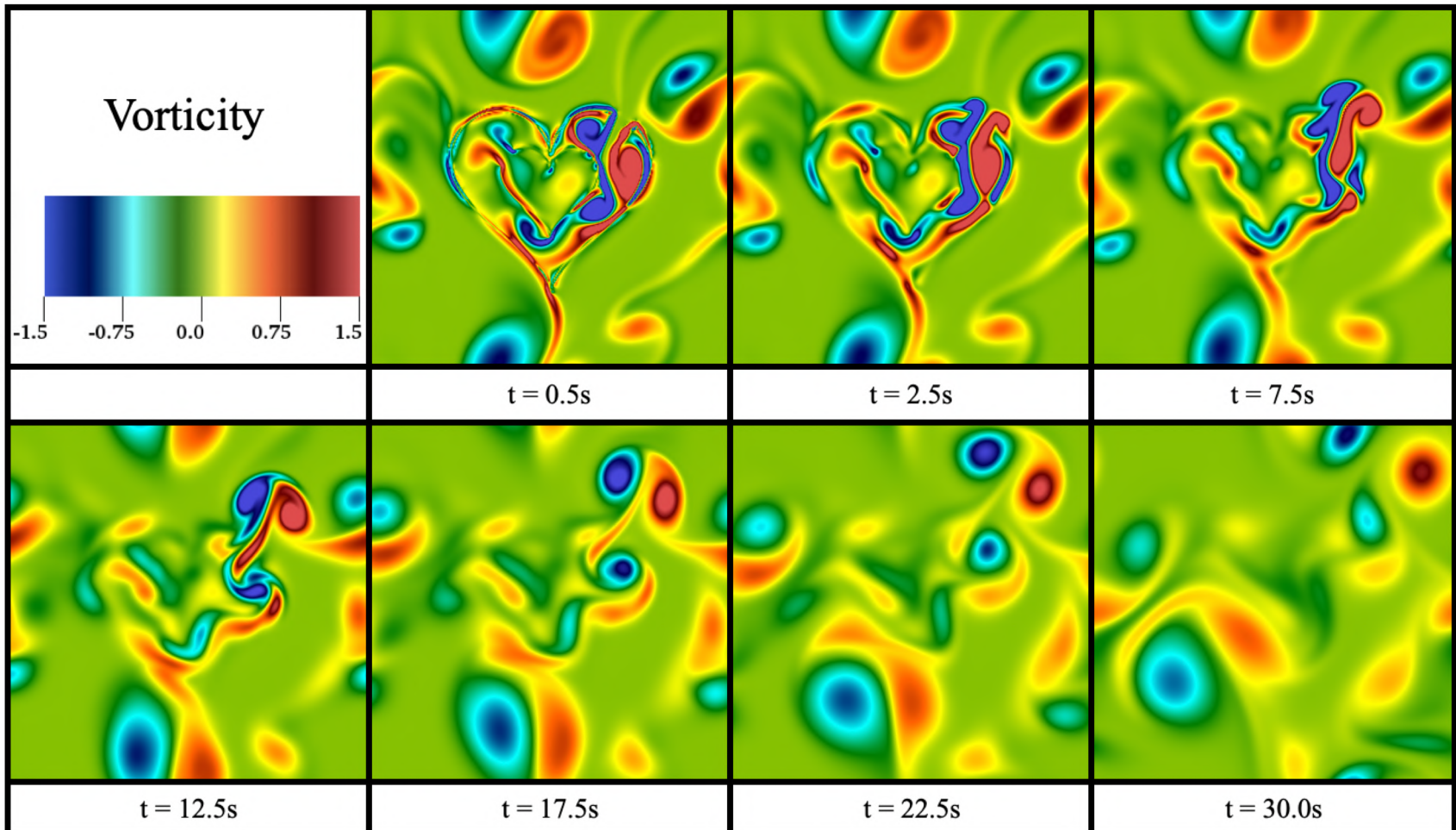
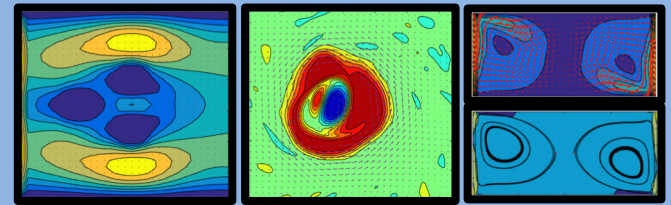
*Vorticity*



*Magnitude of Velocity  
w/ Velocity Field (scaled)*



# FFT: Evolve Vorticity from Initial Velocity Field





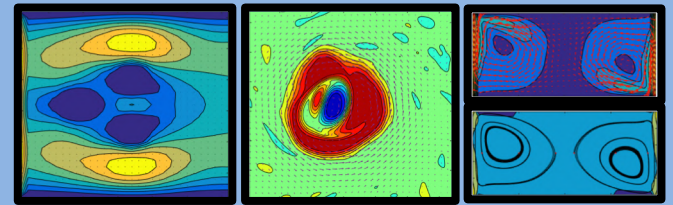
## Spectral Method (FFT)

Examples:

1. Interacting Regions of Vorticity
2. Overlapping Regions of Vorticity
3. Evolution of vorticity from an initial vector velocity field

***\*Overview of Numerical Scheme to follow\****

# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

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

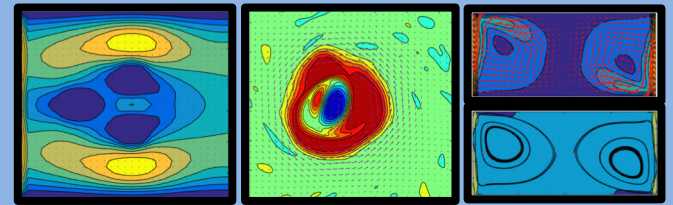
Define vorticity:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}$$

Vorticity Form. Of NS:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \times (\boldsymbol{\omega} \times \mathbf{u}) = \nu \Delta \boldsymbol{\omega}$$

# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

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

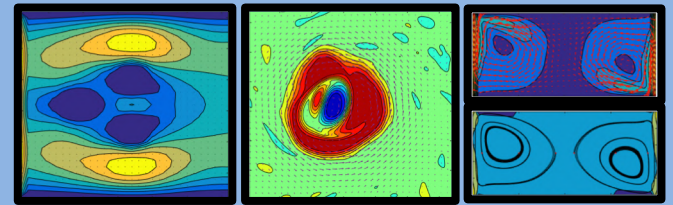
Define vorticity:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}$$

Vorticity Form. Of NS:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \times (\boldsymbol{\omega} \times \mathbf{u}) = \nu \Delta \boldsymbol{\omega}$$

# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

$$\frac{\partial \omega}{\partial t} + \nabla \times (\omega \times \mathbf{u}) = \nu \Delta \omega$$

---

$$\nabla \times (\mathbf{A} \times \mathbf{B}) = (\nabla \cdot \mathbf{B} - \mathbf{B} \cdot \nabla) \mathbf{A} - (\nabla \cdot \mathbf{A} + \mathbf{A} \cdot \nabla) \mathbf{B}$$

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega - \omega \cdot \nabla \mathbf{u} = \nu \Delta \omega$$



# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

$$\frac{\partial \omega}{\partial t} + \nabla \times (\omega \times \mathbf{u}) = \nu \Delta \omega$$

---

$$\nabla \times (\mathbf{A} \times \mathbf{B}) = (\nabla \cdot \mathbf{B} - \mathbf{B} \cdot \nabla) \mathbf{A} - (\nabla \cdot \mathbf{A} + \mathbf{A} \cdot \nabla) \mathbf{B}$$

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega - \omega \cdot \nabla \mathbf{u} = \nu \Delta \omega$$

# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

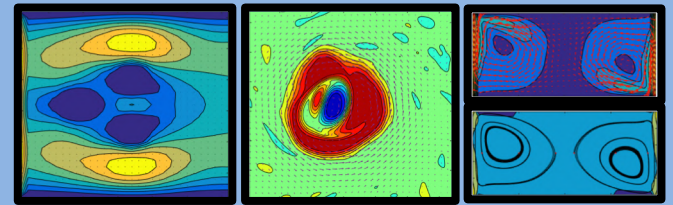
$$\frac{\partial \omega}{\partial t} + \nabla \times (\omega \times \mathbf{u}) = \nu \Delta \omega$$

---

$$\nabla \times (\mathbf{A} \times \mathbf{B}) = (\nabla \cdot \mathbf{B} - \mathbf{B} \cdot \nabla) \mathbf{A} - (\nabla \cdot \mathbf{A} + \mathbf{A} \cdot \nabla) \mathbf{B}$$

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega - \cancel{\omega \cdot \nabla} \mathbf{u} = \nu \Delta \omega$$

# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

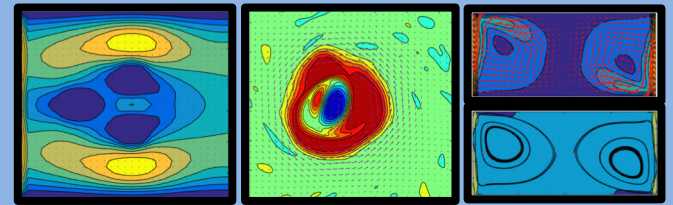
$$\frac{\partial \omega}{\partial t} + \nabla \times (\omega \times \mathbf{u}) = \nu \Delta \omega$$

$$\nabla \times (\mathbf{A} \times \mathbf{B}) = (\nabla \cdot \mathbf{B} - \mathbf{B} \cdot \nabla) \mathbf{A} - (\nabla \cdot \mathbf{A} + \mathbf{A} \cdot \nabla) \mathbf{B}$$

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega - \omega \cdot \nabla \mathbf{u} = \nu \Delta \omega$$

B/c only considering  
2D flows!

# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \Delta \omega$$

Define “Vector Potential”, aka *Stream-function*:

$$\mathbf{u} = \nabla \times \psi \hat{k}$$

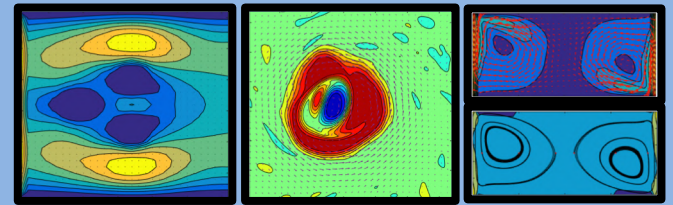
$$u = \frac{\partial \psi}{\partial y}$$

$$v = -\frac{\partial \psi}{\partial x}$$

Hence we have the Poisson problem for the stream-function,

$$\Delta \psi = -\omega$$

# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \Delta \omega$$

Define “Vector Potential”, aka *Stream-function*:

$$\mathbf{u} = \nabla \times \psi \hat{k}$$

$$u = \frac{\partial \psi}{\partial y}$$

$$v = -\frac{\partial \psi}{\partial x}$$

Hence we have the Poisson problem for the stream-function,

$$\Delta \psi = -\omega$$



# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \Delta \omega$$

Define “Vector Potential”, aka *Stream-function*:

$$\mathbf{u} = \nabla \times \psi \hat{k}$$

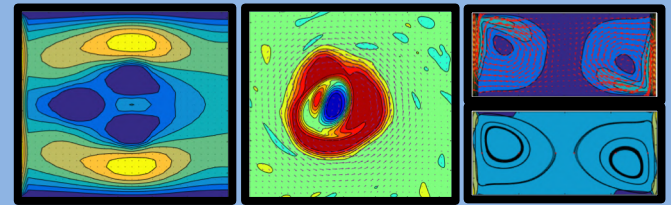
$$u = \frac{\partial \psi}{\partial y}$$

$$v = -\frac{\partial \psi}{\partial x}$$

Hence we have the Poisson problem for the stream-function,

$$\Delta \psi = -\omega$$

# Spectral Methods (FFT)



- First put into vorticity formulation of Navier-Stokes

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \Delta \omega$$

Define “Vector Potential”, aka *Stream-function*:

$\mathbf{u} =$

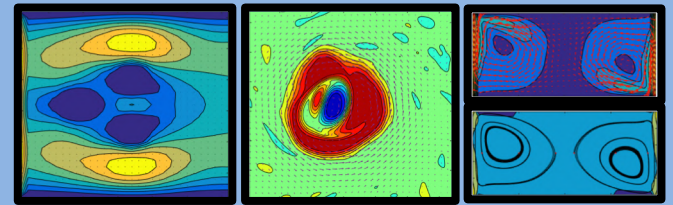
So if we can find the vorticity...we can find the velocity field and it will be automatically incompressible (divergence free)

$$v = -\frac{\partial \psi}{\partial x}$$

Hence we have the Poisson problem for the stream-function,

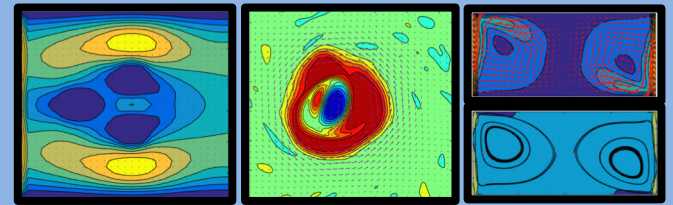
$$\Delta \psi = -\omega$$

# Spectral Methods (FFT)



- Steps For Algorithm: (**EVERYTHING IN FREQUENCY SPACE!**)
  1. Solve Poisson Problem for Stream-Function from previous time-step vorticity
  2. Compute x,y derivatives of stream-function and vorticity (**in real space**), then compute advection term, and finally transform it into frequency space
  3. Crank-Nicholson (**semi-implicit**: explicit for nonlinear advection term, implicit for viscous term)

# Spectral Methods (FFT)



- Steps For Algorithm: (**EVERYTHING IN FREQUENCY SPACE!**)
  1. Solve Poisson Problem for Stream-Function from previous time-step vorticity
  2. Compute x,y derivatives of stream-function and vorticity (**in real space**), then compute advection term, and finally transform it into frequency space
  3. Crank-Nicholson (**semi-implicit**: explicit for nonlinear advection term, implicit for viscous term)



# Spectral Methods (FFT)



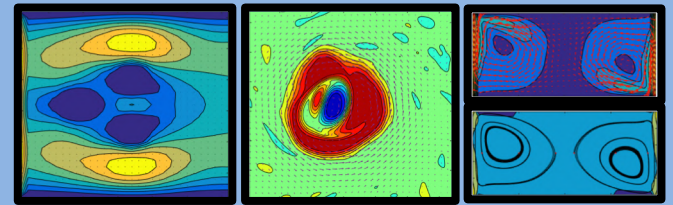
- Steps For Algorithm: (**EVERYTHING IN FREQUENCY SPACE!**)
  1. Solve Poisson Problem for Stream-Function from previous time-step vorticity
  2. Compute x,y derivatives of stream-function and vorticity (**in real space**), then compute advection term, and finally transform it into frequency space
  3. Crank-Nicholson (**semi-implicit**: explicit for nonlinear advection term, implicit for viscous term)

# Spectral Methods (FFT)



- Steps For Algorithm: (**EVERYTHING IN FREQUENCY SPACE!**)
  1. Solve Poisson Problem for Stream-Function from previous time-step vorticity
  2. Compute  $x, y$  derivatives of stream-function and vorticity (**in real space**), then compute advection term, and finally transform it into frequency space
  3. Crank-Nicholson (**semi-implicit**: explicit for nonlinear advection term, implicit for viscous term)

# Spectral Methods (FFT)

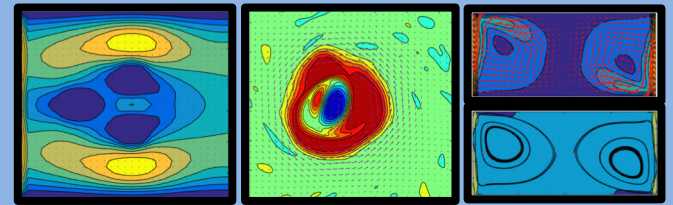


- Steps For Algorithm: (**EVERYTHING IN FREQUENCY SPACE!**)

1. Solve Poisson Problem for Stream-function from previous time-step vorticity

$$\hat{\psi}_{ij}^n = \frac{\omega_{ij}^n}{k_{X_i}^2 + k_{Y_j}^2}$$

# Spectral Methods (FFT)



- Steps For Algorithm: (**EVERYTHING IN FREQUENCY SPACE!**)

2. Compute x,y derivatives of stream-function and vorticity (**in real space**), then compute advection term, and finally transform it into frequency space

$$u_{ij}^n = \mathcal{F}^{-1} \left\{ K_Y \hat{\psi}_{ij}^n \right\}$$

$$v_{ij}^n = \mathcal{F}^{-1} \left\{ -K_X \hat{\psi}_{ij}^n \right\}$$

$$\omega_{x_{ij}}^n = \mathcal{F}^{-1} \left\{ K_X \hat{\omega}_{ij}^n \right\}$$

$$\omega_{y_{ij}}^n = \mathcal{F}^{-1} \left\{ K_Y \hat{\omega}_{ij}^n \right\}$$

$$F_{\text{adv}_{ij}}^n = u_{ij}^n \cdot \omega_{x_{ij}}^n + v_{ij}^n \cdot \omega_{y_{ij}}^n$$

$$\hat{F}_{\text{adv}_{ij}}^n = \mathcal{F} \left\{ F_{\text{adv}_{ij}}^n \right\}$$



# Spectral Methods (FFT)

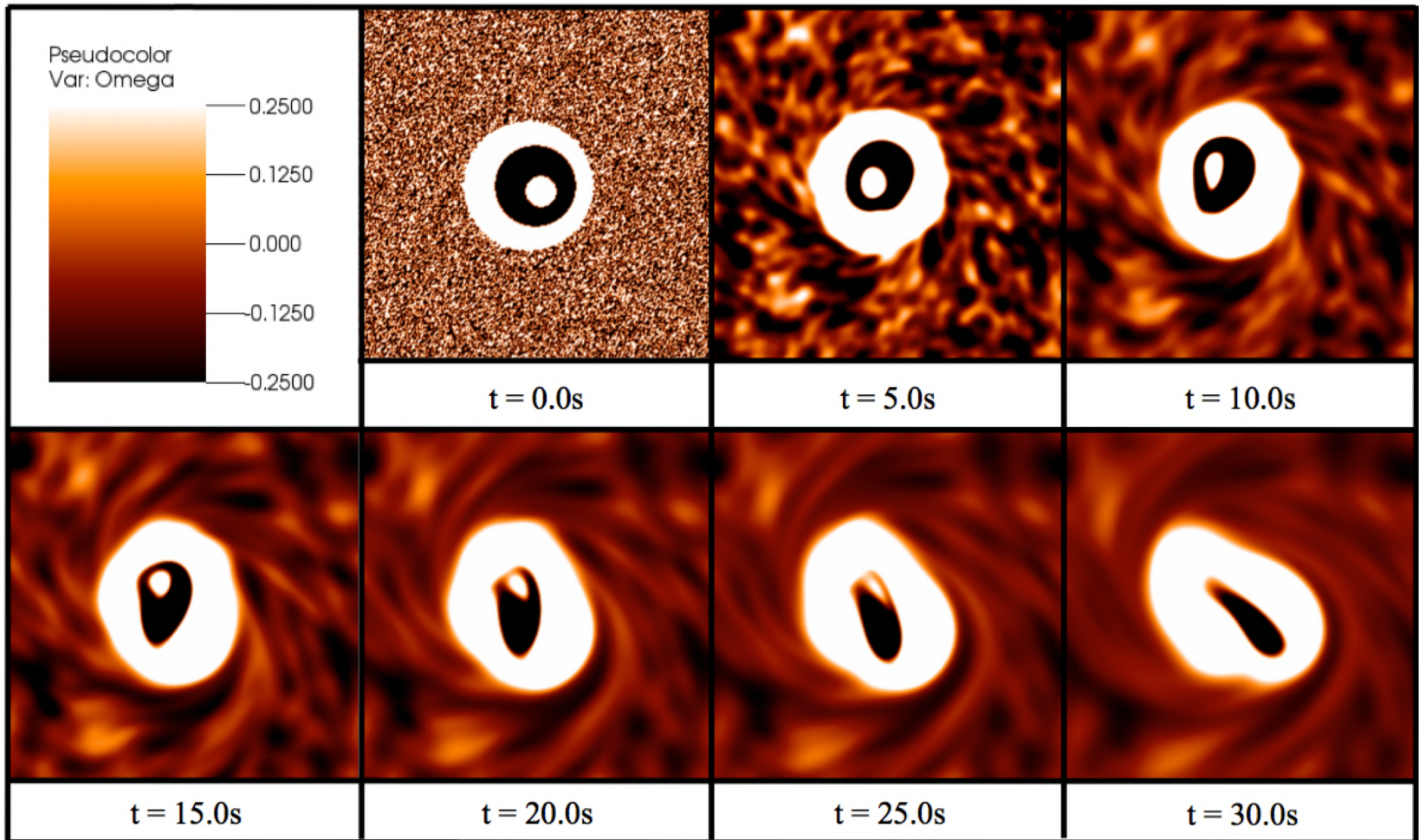
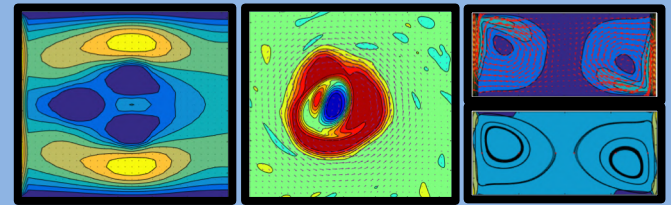


- Steps For Algorithm: (**EVERYTHING IN FREQUENCY SPACE!**)

3. Crank-Nicholson (**semi-implicit**: explicit for nonlinear advection term, implicit for viscous term)

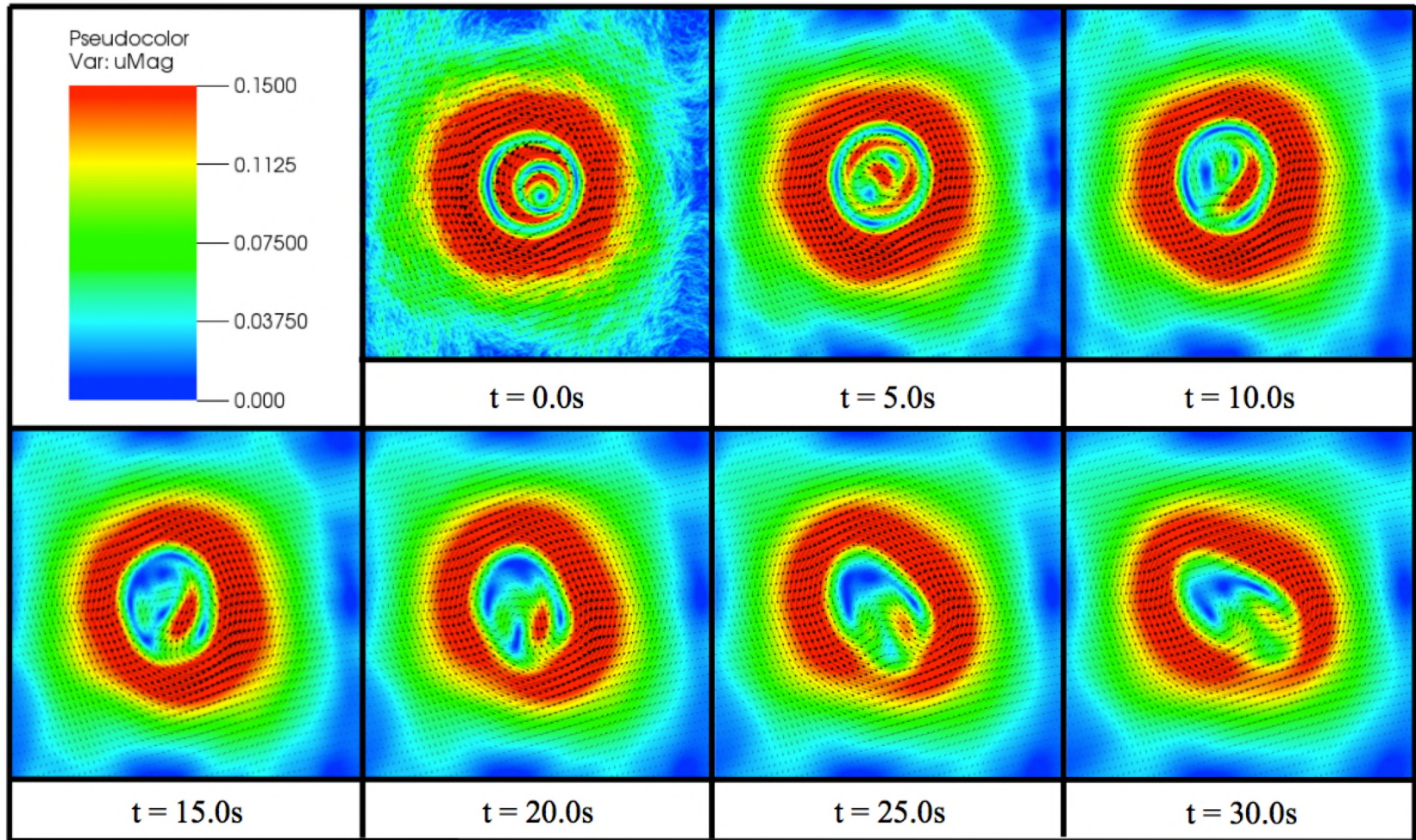
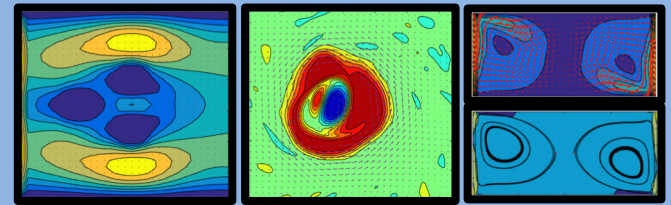
$$\hat{\psi}_{ij}^{n+1} = \frac{\overbrace{\left[ 1 + \frac{\nu \Delta t}{2} \left( k_{X_{ij}}^2 + k_{Y_{ij}}^2 \right) \right]}^{\text{implicit}} \hat{\psi}_{ij}^n - \underbrace{\Delta t \hat{F}_{\text{adv}_{ij}}^n}_{\text{explicit}}}{1 - \frac{\nu \Delta t}{2} \left( k_{X_{ij}}^2 + k_{Y_{ij}}^2 \right)}$$

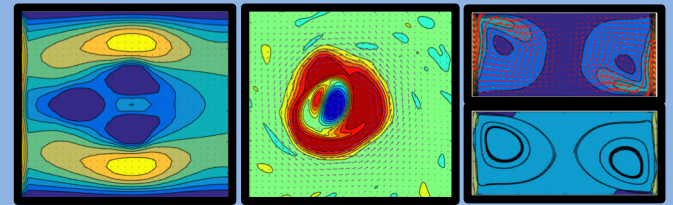
# Spectral Methods (FFT)





# Spectral Methods (FFT)





# Lattice Boltzmann Method (LBM)

Examples:

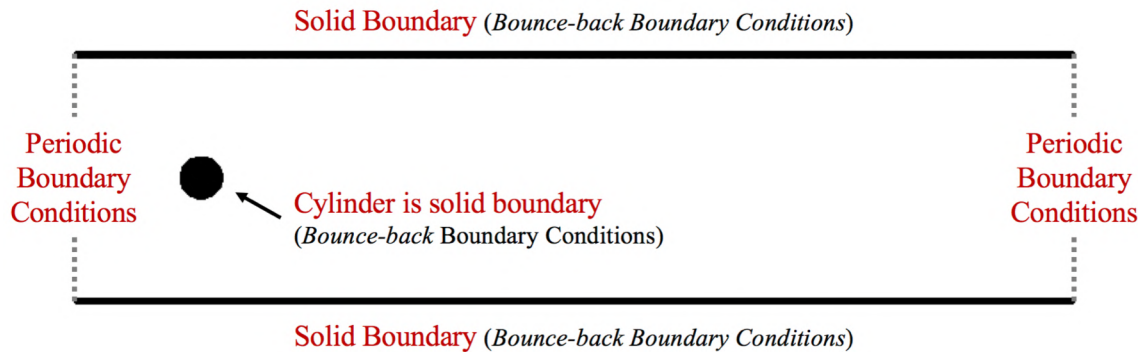
**1. Flow around one or multiple cylinders**

2. Flow around porous cylinder

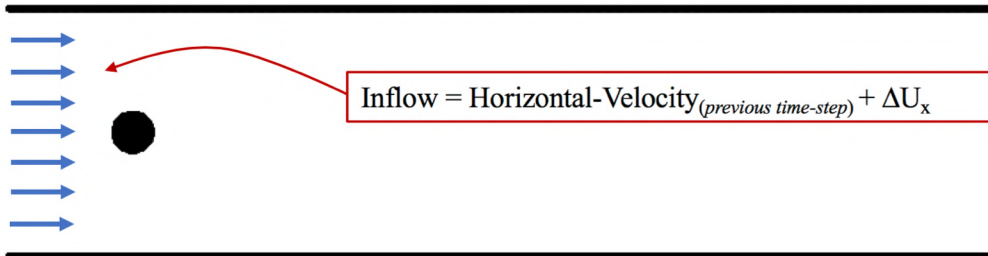
*\*Overview of Numerical Scheme to follow\**



# LBM: Flow around cylinders

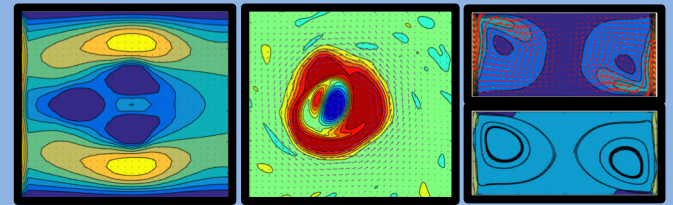


**Boundary conditions**  
along domain and on  
cylinder(s)



**Inflow:** left to right, gets  
ramped up during  
simulation

# LBM: Flow around cylinders



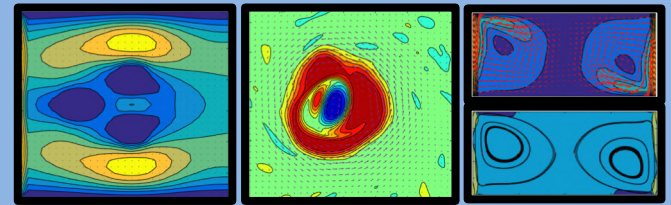
## Flow around a single cylinder:

*Vorticity + contours*



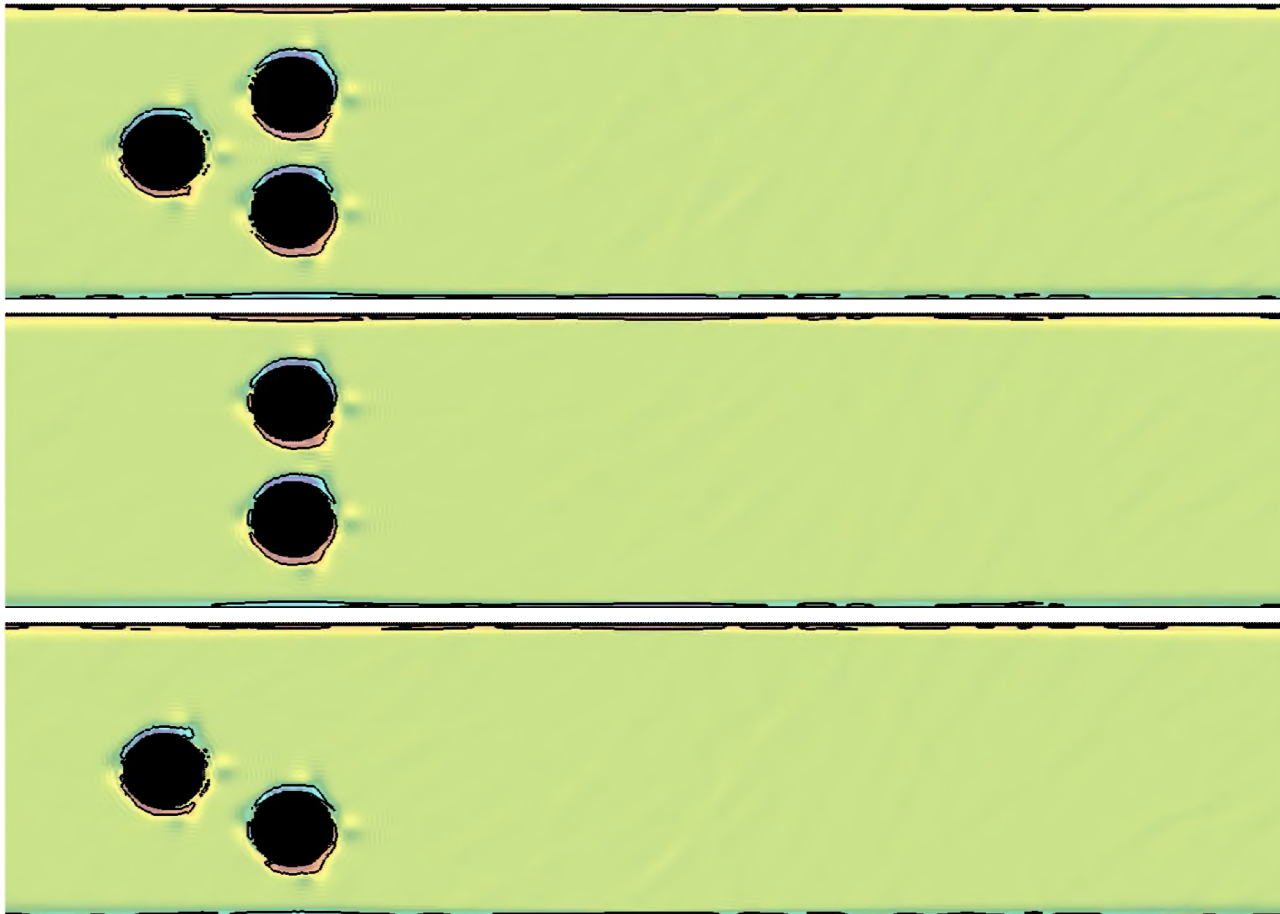
- Flow remains symmetric until instability forms in which there is a *transition from smooth flow to vortical flow patterns* and
- Vortices begin to be *shed* off of cylinder (*e.g., vortex shedding*)

# LBM: Flow around cylinders

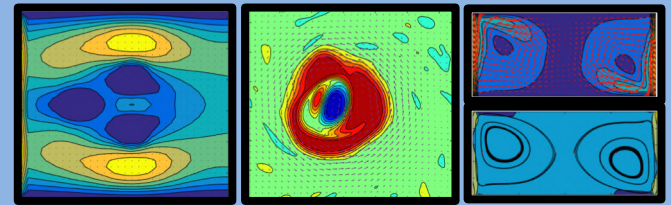


## Flow around multiple cylinders:

*Vorticity + contours*

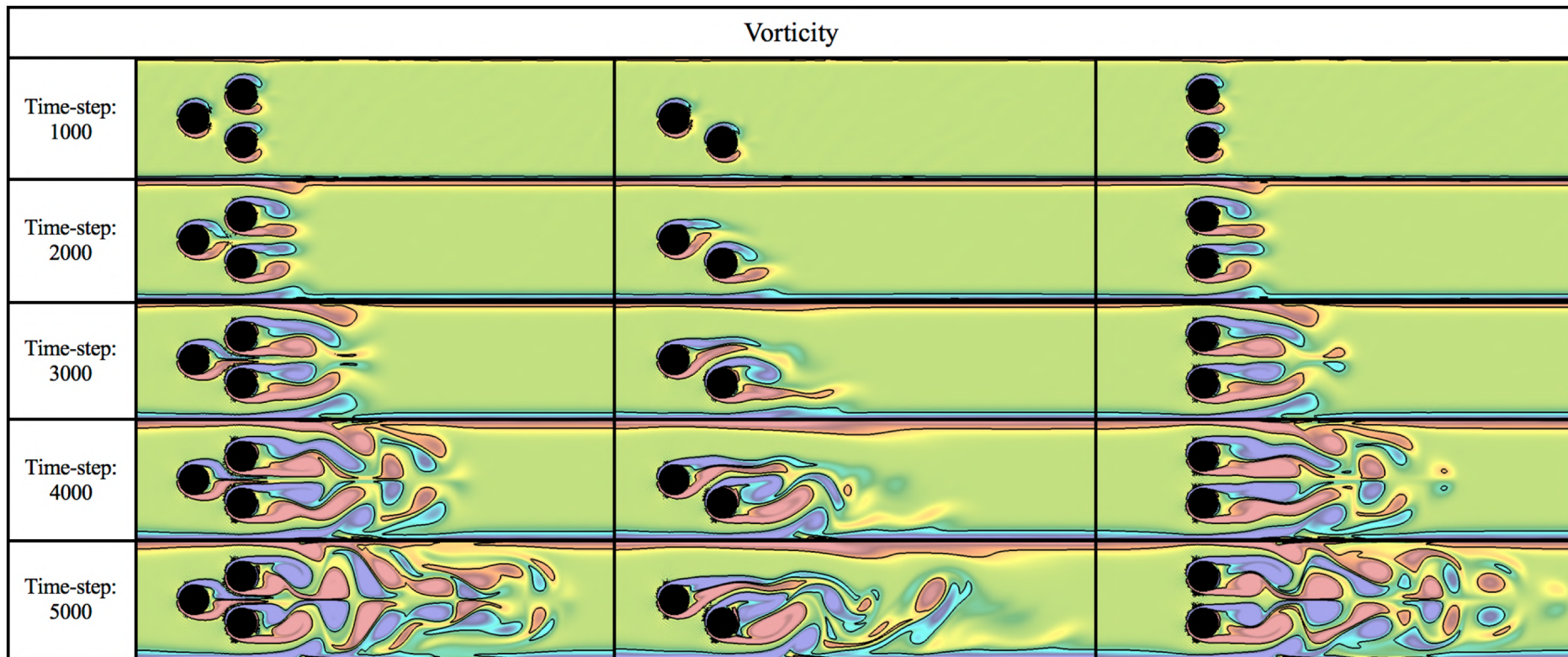


# LBM: Flow around cylinders



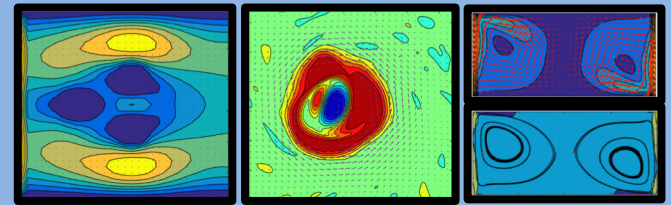
## Flow around multiple cylinders:

*Vorticity + contours*



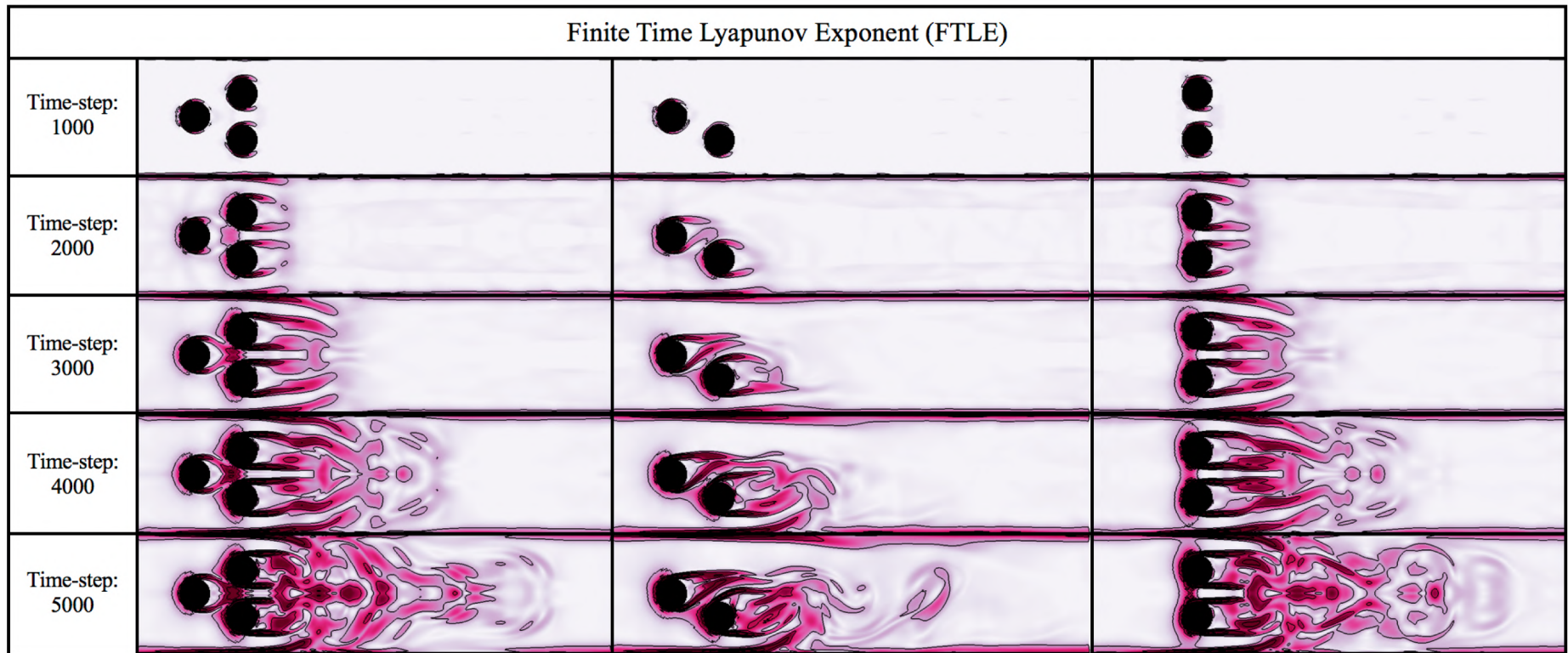


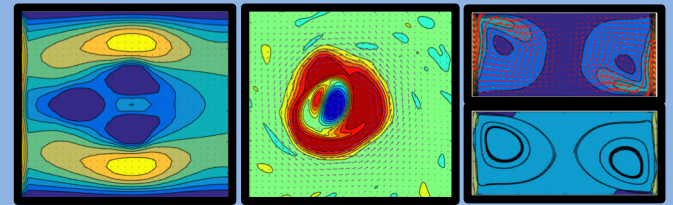
# LBM: Flow around cylinders



## Flow around multiple cylinders:

*FTLE + contours*





## Lattice Boltzmann Method (LBM)

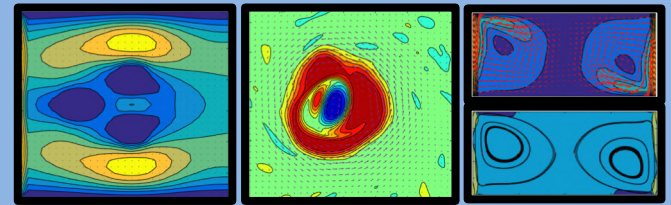
Examples:

1. Flow around one or multiple cylinders

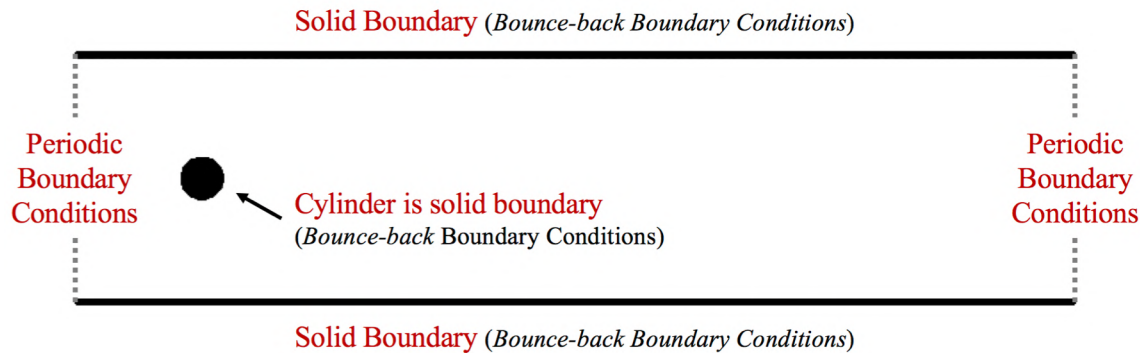
2. Flow around porous cylinder

*\*Overview of Numerical Scheme to follow\**

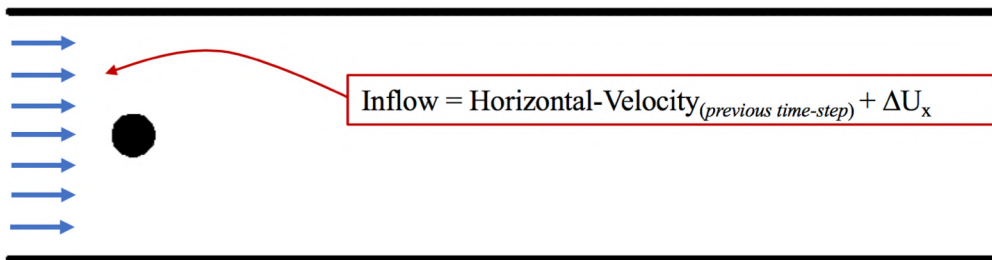
# LBM: Flow around POROUS cylinders



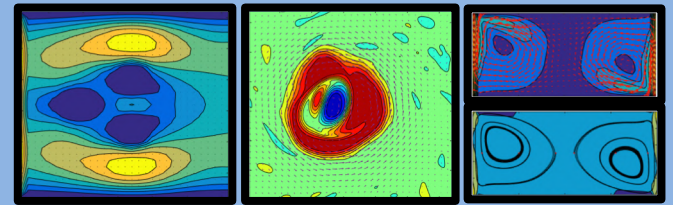
## SAME BOUNDARY CONDITIONS AND INFLOW CONDITIONS AS PREVIOUS LBM CASES!



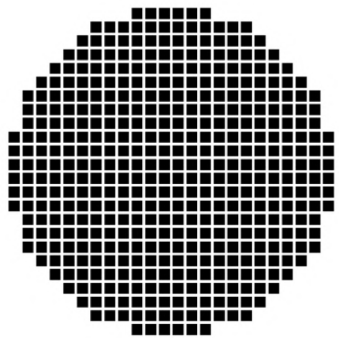
**Boundary conditions**  
along domain and on  
cylinder(s)



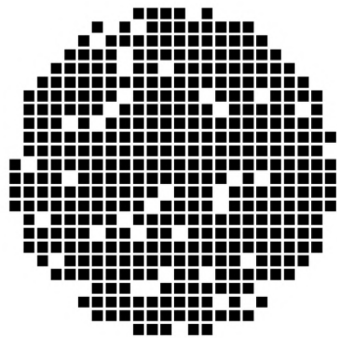
**Inflow:** left to right, gets  
ramped up during  
simulation



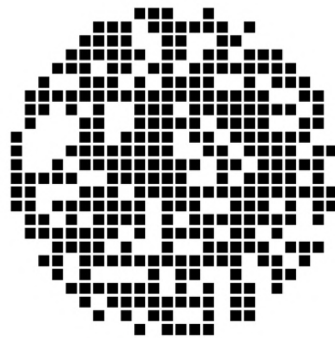
## Porous Cylinder Geometries



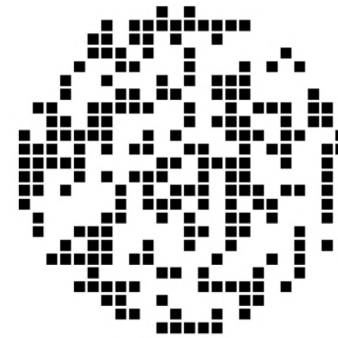
VF=0%



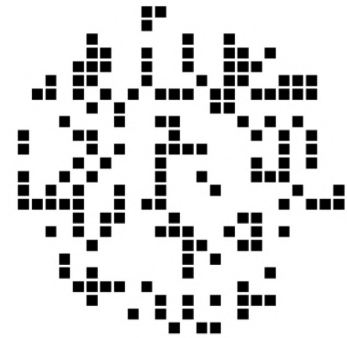
VF=10%



VF=25%



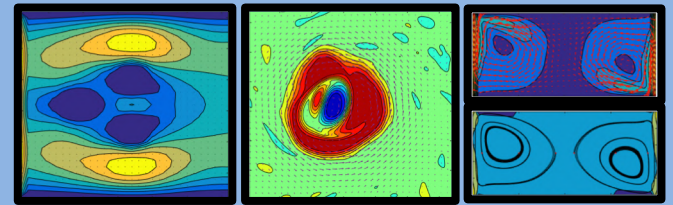
VF=50%



VF=62.5%

- VF = void fraction
- Higher VF = lower amount of grid cells blocked out for geometry
- *Porosity is modeled randomly*; no particular void networks through cylinder.

# LBM: Flow around POROUS cylinders

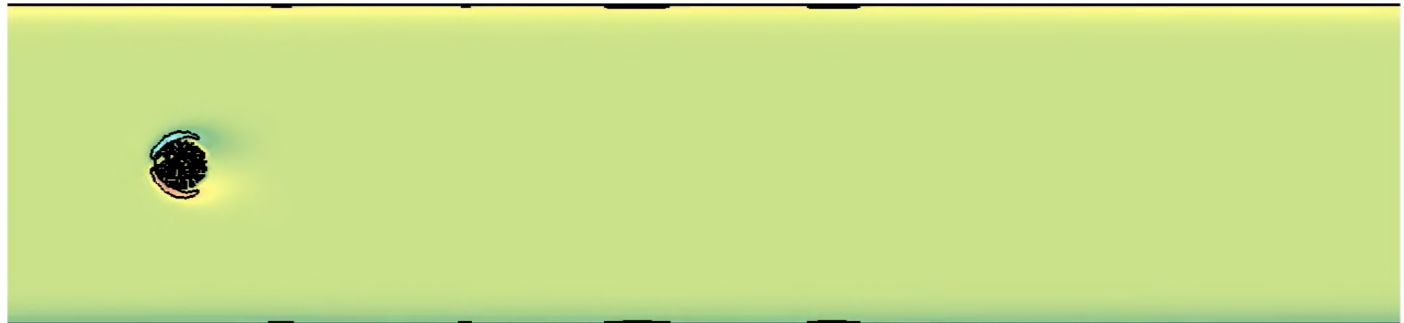


## Flow around POROUS cylinders:

**VF = 0%**  
(solid)



**VF = 25%**

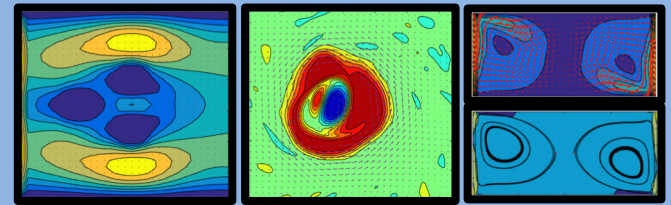


**VF = 62.5%**

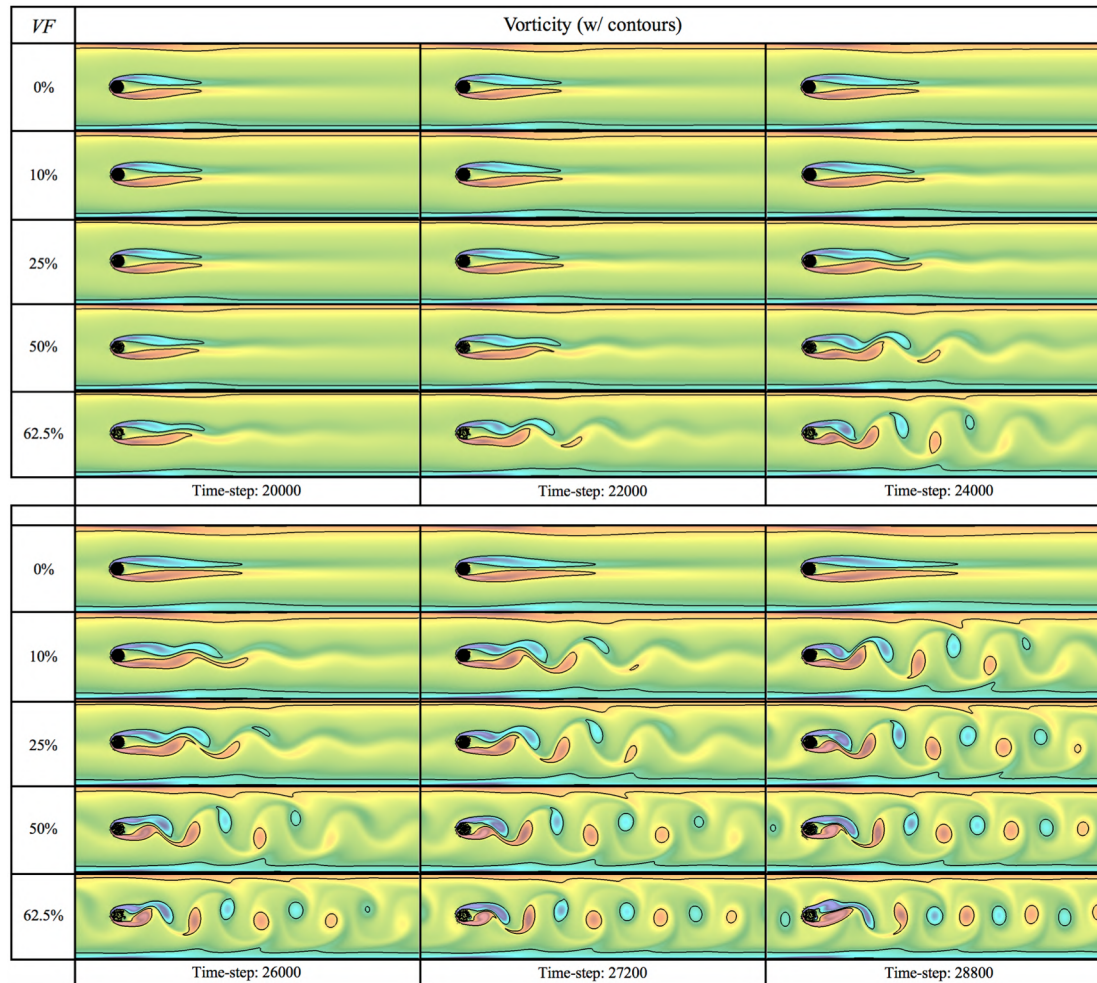




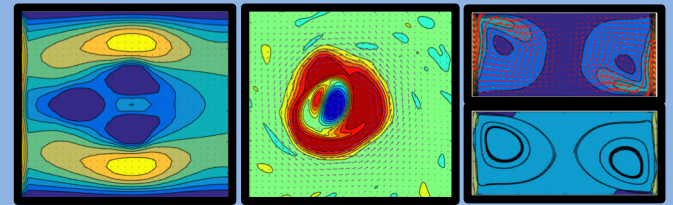
# LBM: Flow around POROUS cylinders



## Flow around POROUS cylinders:



- There is a quicker transition to vortex shedding in all cases of porous cylinders when compared to the solid cylinder
- Cases with higher  $VF$ , see such transition earlier than lower cases



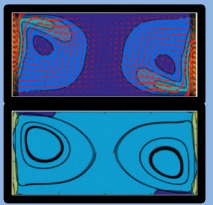
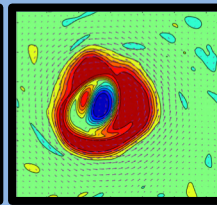
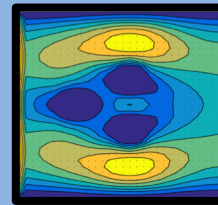
## Lattice Boltzmann Method (LBM)

Examples:

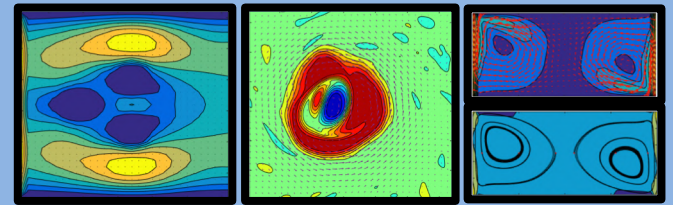
1. Flow around one or multiple cylinders
2. Flow around porous cylinder

*\*Overview of Numerical Scheme \**

# Lattice Boltzmann

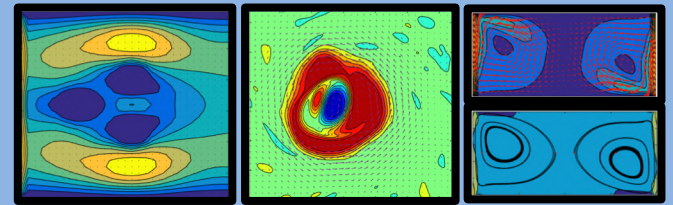


# Lattice Boltzmann

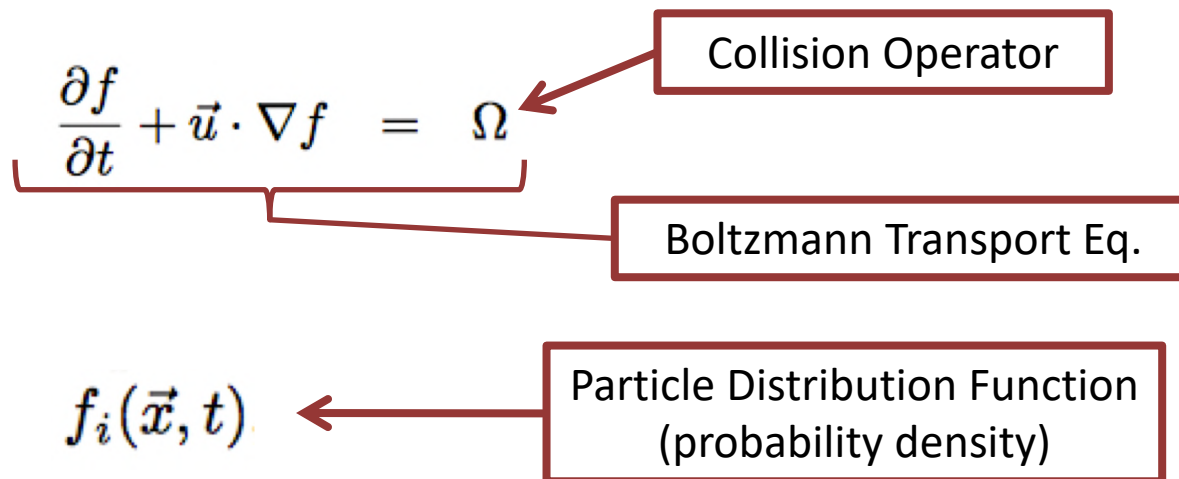


- **Uses discrete Boltzmann equations to model fluid dynamics (does not explicitly solve Navier-Stokes)**
  - Think of it as tracks fictitious particles of fluid in the flow
  - Then we have a constitutive propagation relation and consider collision processes over a discrete lattice mesh

# Lattice Boltzmann

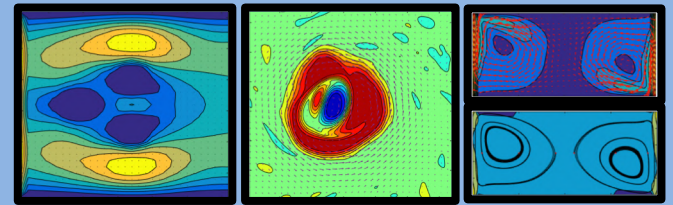


- **Uses discrete Boltzmann equations to model fluid dynamics (does not explicitly solve Navier-Stokes)**
  - Think of it as tracks fictitious particles of fluid in the flow
  - Then we have a constitutive propagation relation and consider collision processes over a discrete lattice mesh

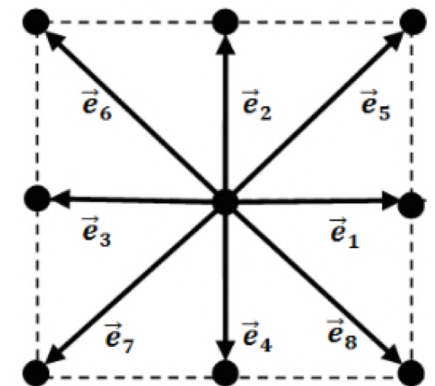
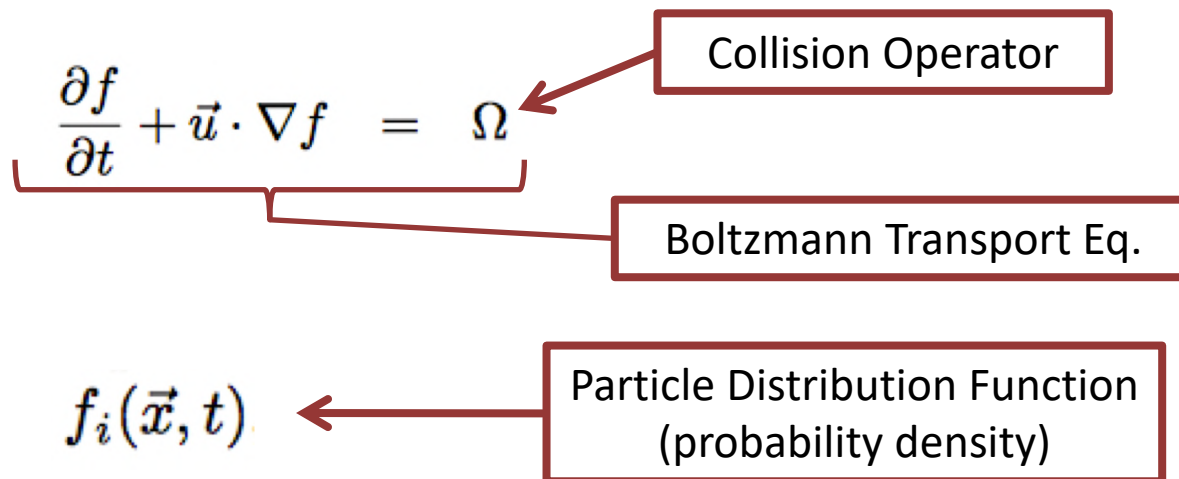




# Lattice Boltzmann

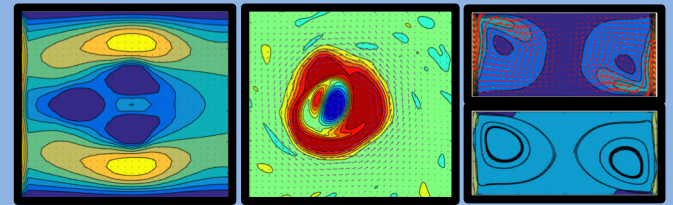


- Uses discrete Boltzmann equations to model fluid dynamics (does not explicitly solve Navier-Stokes)
  - Think of it as tracks fictitious particles of fluid in the flow
  - Then we have a constitutive propagation relation and consider collision processes over a discrete lattice mesh



"D2Q9"

# Lattice Boltzmann

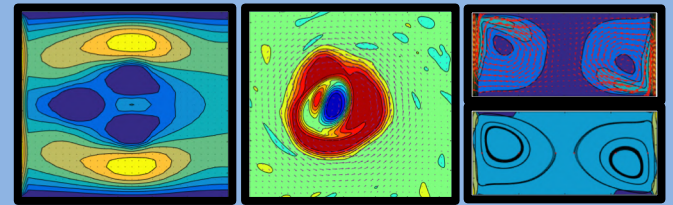


- **Uses discrete Boltzmann equations to model fluid dynamics (does not explicitly solve Navier-Stokes)**

$$\rho(\vec{x}, t) = \sum_{i=0}^8 f_i(\vec{x}, t) \quad \vec{u}(\vec{x}, t) = \frac{1}{\rho} \sum_{i=0}^8 c f_i \vec{e}_i$$

- **Steps:**
  1. Stream the particle densities propagate in each direction
  2. Find an equivalent “equilibrium” density
  3. Relax the densities towards that EQ. state, in proportion governed by  $\tau$  (‘viscosity’)

# Lattice Boltzmann



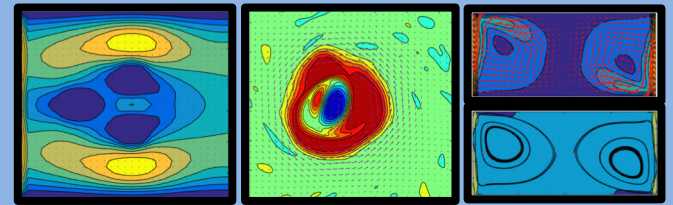
- Uses discrete Boltzmann equations to model fluid dynamics (does not explicitly solve Navier-Stokes)

$$\rho(\vec{x}, t) = \sum_{i=0}^8 f_i(\vec{x}, t) \quad \vec{u}(\vec{x}, t) = \frac{1}{\rho} \sum_{i=0}^8 c f_i \vec{e}_i$$

- **Steps:**

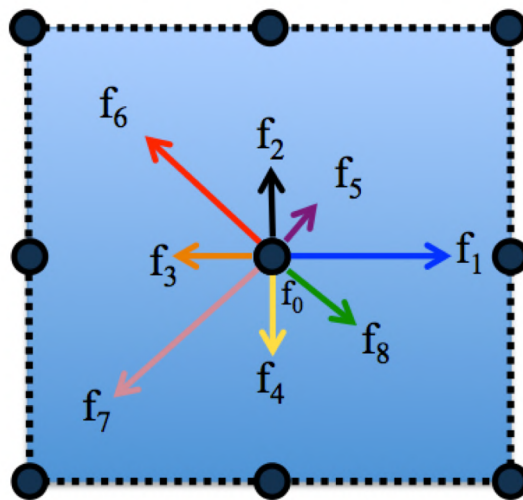
1. Stream the particle densities propagate in each direction
2. Find an equivalent “equilibrium” density
3. Relax the densities towards that EQ. state, in proportion governed by  $\tau$  (‘viscosity’)

# Lattice Boltzmann

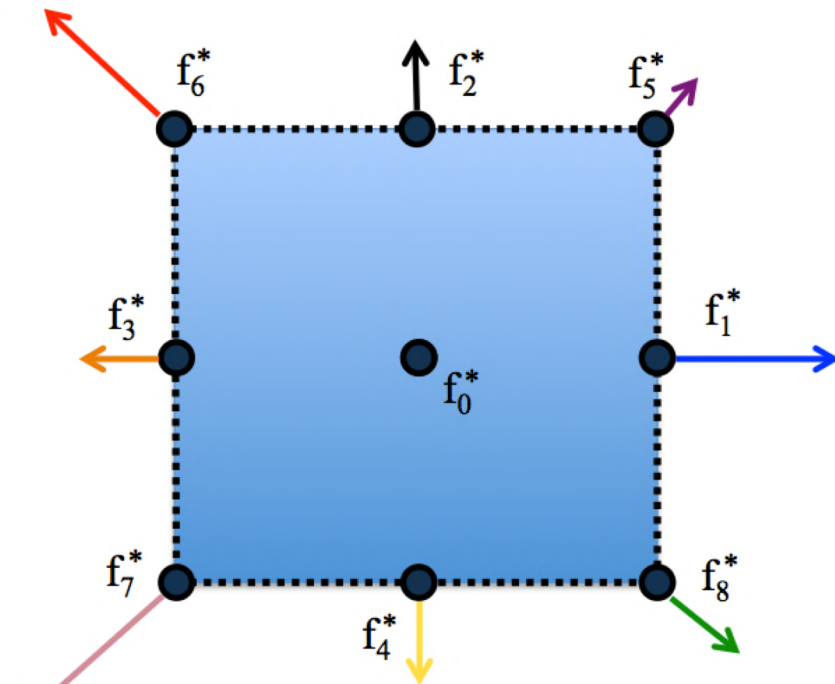


- **Steps:**

1. Stream the particle densities propagate in each direction

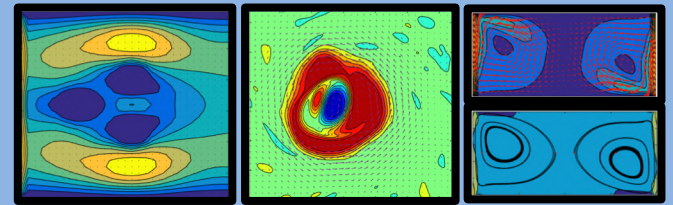


**Pre-Streaming**



**Post-Streaming**

# Lattice Boltzmann



- **Steps:**

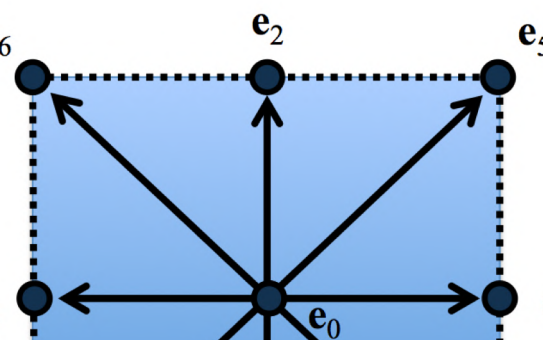
1. Stream the particle densities propagate in each direction

$$f_2^*(x_i, y_j) = f_2(x_i, y_{j-1})$$

$$f_6^*(x_i, y_j) = f_6(x_{i+1}, y_{j-1}) e_6$$

$$f_5^*(x_i, y_j) = f_5(x_{i-1}, y_{j-1})$$

$$f_3^*(x_i, y_j) = f_3(x_{i+1}, y_j) e_3$$



$$f_1^*(x_i, y_j) = f_1(x_{i-1}, y_j) e_1$$

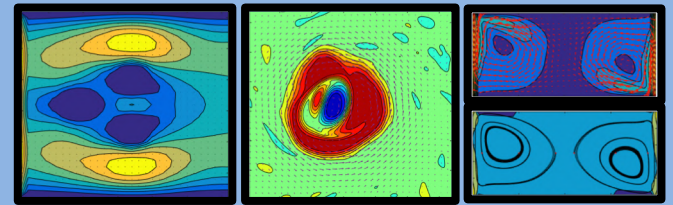
$$f_7^*(x_i, y_j) = f_7(x_{i+1}, y_{j+1}) e_7$$

$$f_4^*(x_i, y_j) = f_4(x_i, y_{j+1})$$

$$f_8^*(x_i, y_j) = f_8(x_{i-1}, y_{j+1})$$



# Lattice Boltzmann



- **Steps:**

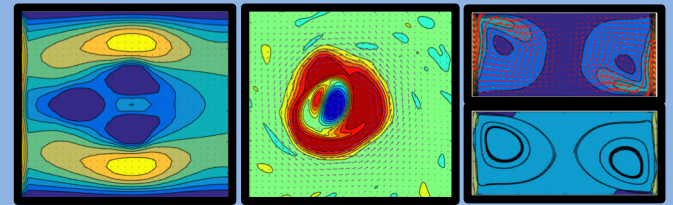
2. Find an equivalent "equilibrium" density (BGK Collision Model, PRL 1954)

$$f_i^{eq}(\vec{x}, t) = w_i \rho + \rho s_i(\vec{u}(\vec{x}, t))$$

$$s_i(\vec{u}) = w_i \left[ 3 \frac{\vec{e}_i \cdot \vec{u}}{c} + \frac{9}{2} \frac{(\vec{e}_i \cdot \vec{u})^2}{c^2} - \frac{3}{2} \frac{\vec{u} \cdot \vec{u}}{c^2} \right] \quad c = \frac{\Delta x}{\Delta t}$$

$$w_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i = 1, 2, 3, 4 \\ 1/36 & i = 5, 6, 7, 8 \end{cases}$$

# Lattice Boltzmann



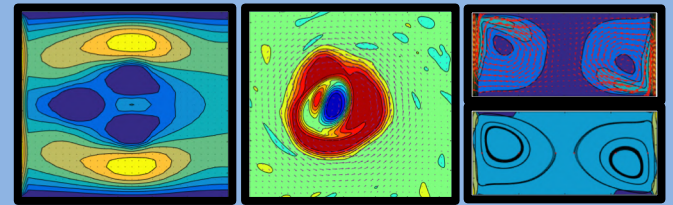
- **Steps:**

3. Relax the densities towards that EQ. state, in proportion governed by  $\tau$ , (BGK Collision Model, PRL 1954)

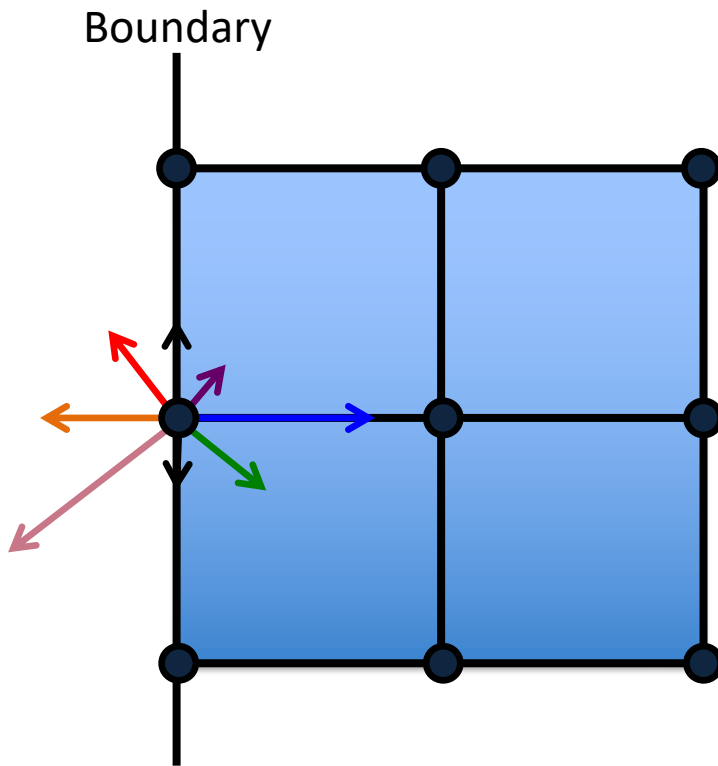
$$f_i = f_i^* - \frac{f_i^* - f_i^{eq}}{\tau}$$

$$\nu = \frac{2\tau - 1}{6} \frac{(\Delta x)^2}{\Delta t}$$

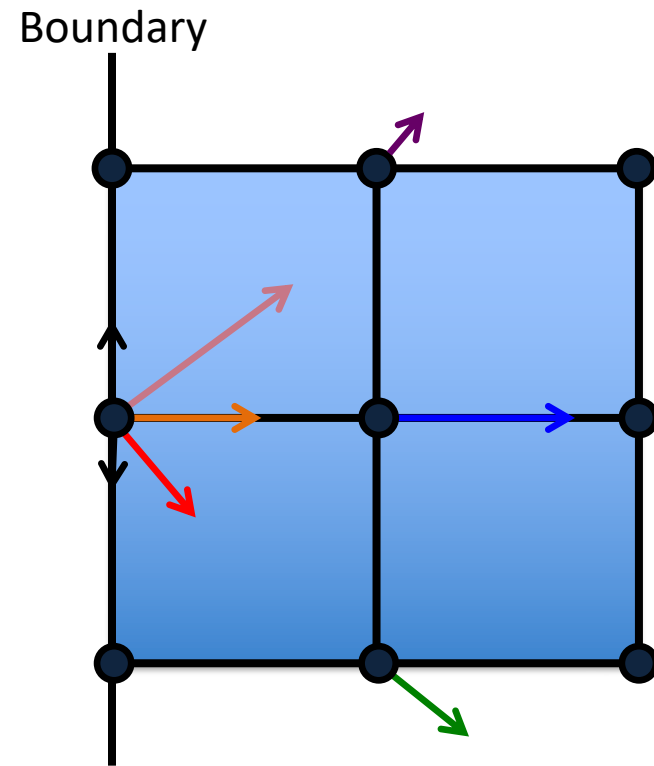
# Lattice Boltzmann



## Boundary Conditions: *“Reflective Conditions”*

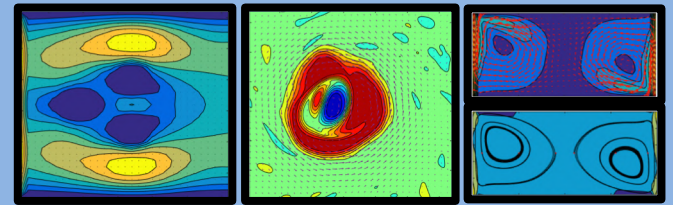


Pre-Streaming



Post-Streaming

# Lattice Boltzmann



Pseudocolor  
Var: Omega

0.02000

0.01000

0.000

-0.01000

-0.02000

$n = 0$

$n = 1000$

$n = 1500$

$n = 2000$

$n = 2500$

$n = 3000$

$n = 3000$

$n = 3200$

$n = 3400$

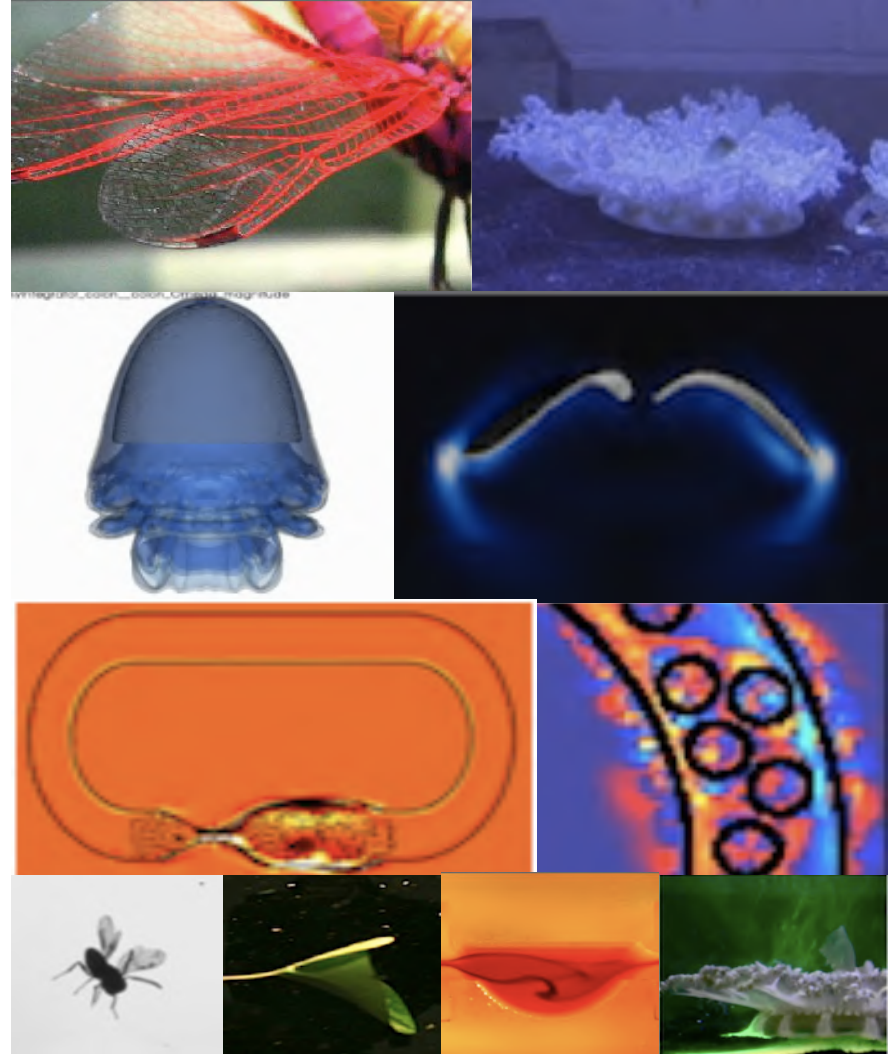
$n = 3600$

$n = 3800$

$n = 4000$

# Acknowledgements:

- **Grants:**
  - TCNJ SoS SOSA (2018-2020)
  - NSF OAC-1828163
- **The BICEP Lab at TCNJ:**
  - **Jason Miles** (uGrad Math)
  - **Erin Haus** (uGrad Bio)
  - **Chris Jakuback** (uGrad Stats/Physics)
  - **Michael Mongelli** (uGrad Bio/CS)
- **Laura Miller (UNC-CH) Lab:**
  - **Dr. Laura Miller**, PI
  - Dr. Austin Baird (Math)
  - Dr. Shannon Jones (Comp. Bio.)
  - Dr. Julia Samson (Bio)
  - Michael Senter (Math)
  - **Andrea Lane** (former uGrad, now at Emory)
  - Grace McClaughlin (former uGrad)
  - Alex Davis (former uGrad, now at Duke)
- **UTK:**
  - Dr. W. Christopher Strickland
  - Leigh B. Percy (Math)
- **Bucknell University**
  - Dr. Christina Hamlet
- **Chapman University**
  - Dr. Lindsay Waldrop
- **University of Akron**
  - Dr. Alex Hoover
- **Oklahoma State U.**
  - Dr. Arvind Santhanakrishnan





# Questions?

[battistn\[at\]tcnj.edu](mailto:battistn[at]tcnj.edu)

 @mostnerdy

[battistn.pages.tcnj.edu](http://battistn.pages.tcnj.edu)

