A numerical study of the transition from turbulence in pipe flow

An introduction to the pipe flow problem, and some of the numerics behind it

David Moxey, Dwight Barkley

Mathematics Institute
University of Warwick

Postgraduate Seminar Series
21st January 2009

Introduction

- Turbulence is a well-known phenomenon in fluid mechanics, but its origins and behaviour are not generally understood. One approach is to look at how instabilities can cause a transition to turbulence.
- One of the most studied examples is pipe flow, first observed by Reynolds in 1883.
- Pipe flow is curious because its basic laminar solution is linearly stable. However, Reynolds noted that very small instabilities can still cause turbulence depending on the *Reynolds number*

$$\mathrm{Re} = \frac{U_B D}{\nu}.$$

  - U_B: average or bulk velocity
  - D: pipe diameter
  - ν: kinematic viscosity
- Most work on the transition problem concentrates on transition from **laminar to turbulent** flow. We wish to build a phase portrait of the fluid going from **turbulent to laminar** flow, and investigate states of **laminar-turbulent co-existance**.

Mathematical Framework

- Flows are governed by the *Navier-Stokes equations*:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}.$$

  $\mathbf{u} : \Omega \times [0, \infty) \to \mathbb{R}^n$ is the velocity field, $p : \Omega \times [0, \infty) \to \mathbb{R}$ is the scalar pressure, ρ is the fluid density and $\mathbf{f}$ is a forcing term (which may or may not depend on time).

- We also insist that the flow is *incompressible*, in which case it satisfies the additional restriction

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

  Also implies ρ is constant and so assume $\rho = 1$.

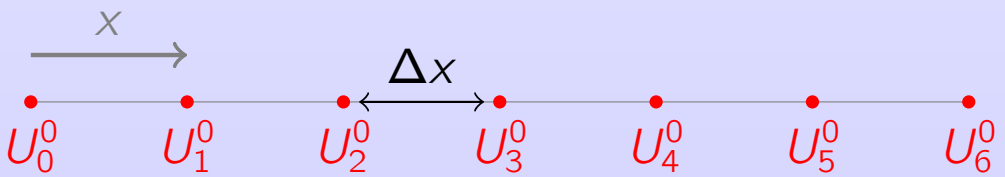
- $(\mathbf{u} \cdot \nabla)\mathbf{u}$ term provides extremely complex non-linear behaviour, and generally means we can't solve Navier-Stokes analytically.
- Instead we use numerical methods to obtain approximations of the solution by computer.
- In our case we perform a DNS - direct numerical simulation - of the equations.

Numerics

- How do we solve PDEs via computer? First step is to discretise the domain into a finite set of points; at each point we store an approximation to the solution.
- Also need to be able to calculate derivatives (and sometimes integrals) numerically, using only the data we have at each point.
- Easiest case is a uniform grid. Consider $x_j = j\Delta x, j = 0, \ldots, J$ where Δx is the grid spacing.
- Discretise time in a similar fashion by writing $t_n = n\Delta t$; Δt is the *timestep*.
- Set $U_j^n = u(x_j, t_n)$.

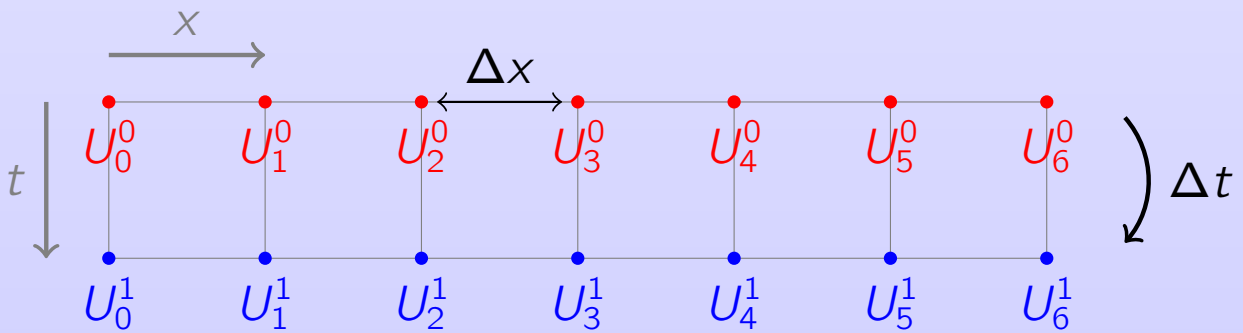
Numerics

- How do we solve PDEs via computer? First step is to discretise the domain into a finite set of points; at each point we store an approximation to the solution.
- Also need to be able to calculate derivatives (and sometimes integrals) numerically, using only the data we have at each point.
- Easiest case is a uniform grid. Consider $x_j = j\Delta x, j = 0, \ldots, J$ where Δx is the grid spacing.
- Discretise time in a similar fashion by writing $t_n = n\Delta t$; Δt is the *timestep*.
- Set $U_j^n = u(x_j, t_n)$.



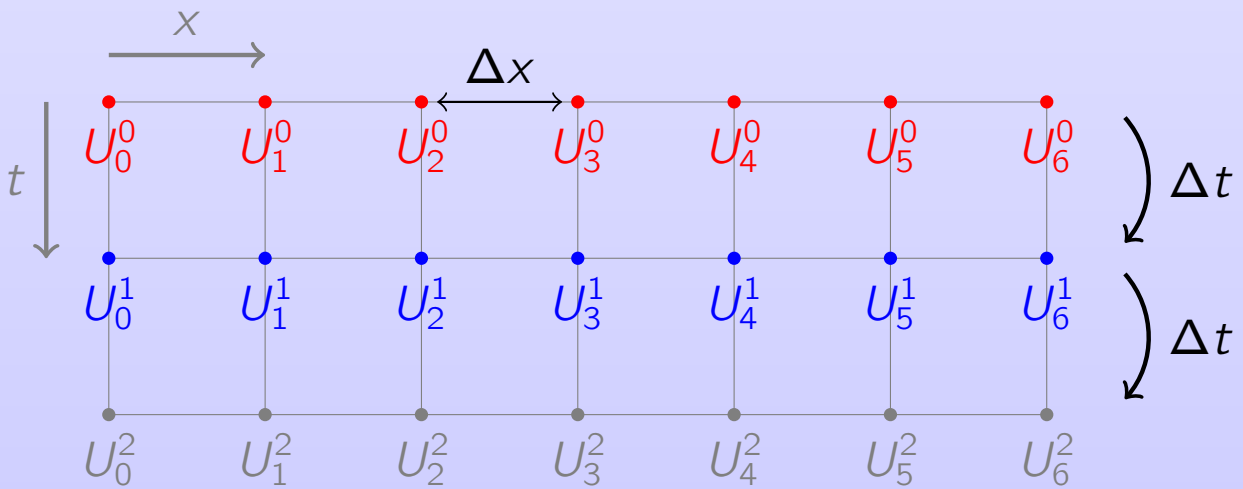
Numerics

- How do we solve PDEs via computer? First step is to discretise the domain into a finite set of points; at each point we store an approximation to the solution.
- Also need to be able to calculate derivatives (and sometimes integrals) numerically, using only the data we have at each point.
- Easiest case is a uniform grid. Consider $x_j = j\Delta x, j = 0, \ldots, J$ where Δx is the grid spacing.
- Discretise time in a similar fashion by writing $t_n = n\Delta t$; Δt is the *timestep*.
- Set $U_j^n = u(x_j, t_n)$.



Numerics

- How do we solve PDEs via computer? First step is to discretise the domain into a finite set of points; at each point we store an approximation to the solution.
- Also need to be able to calculate derivatives (and sometimes integrals) numerically, using only the data we have at each point.
- Easiest case is a uniform grid. Consider $x_j = j\Delta x, j = 0, \ldots, J$ where Δx is the grid spacing.
- Discretise time in a similar fashion by writing $t_n = n\Delta t$; Δt is the *timestep*.
- Set $U_j^n = u(x_j, t_n)$.



Finite Difference

- One of the easiest ways of numerically calculating derivatives. Essentially approximates the limit definition of the derivative for a rougher guess, or otherwise we can use Taylor expansions to gain accuracy.

$$\left.\frac{\partial u}{\partial x}\right|_{x=x_j} = \frac{U^n_{j+1} - U^n_j}{\Delta x} + O(\Delta x)$$
$$= \frac{U^n_{j+1} - U^n_{j-1}}{2\Delta x} + O(\Delta x^2)$$

- Easy to implement because the discretisation of an operator $\mathbb{L}$ can be represented by a matrix $\mathbf{L}$. For example, for $\mathbb{L} = \partial_x$ and using the second expansion above, we have

$$\mathbf{LU} = \frac{1}{2\Delta x}\begin{bmatrix} \ddots & & & & \\ & -1 & 0 & 1 & \\ & & & & \ddots \end{bmatrix}\begin{pmatrix} U^n_0 \\ \vdots \\ U^n_j \end{pmatrix}$$

- **Problem:** Not very accurate unless you use a large number of points. More points requires far more computation when doing matrix multiplication. Not very useful for pipe flow.

Spectral Methods

- **Basic idea:** Approximate the function by writing it as a sum of *modes* $\phi_k(x)$ weighted by *amplitudes* $\hat{u}_k(t)$:

$$u(x,t) \approx u^{\delta}(x,t) = \sum_{k=0}^{N-1} \hat{u}_k(t)\phi_k(x).$$

- In 'simple' geometries, this is one of the most commonly used methods because they have extremely good error properties.
- **Quick example:** 1D heat equation in $\Omega = [0, \pi]$ and homogeneous Dirichlet boundary conditions:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

Let $\phi_k(x) = \sin(kx)$ and substitute approximation into PDE:

$$\sum_{k=0}^{N-1} \frac{\mathrm{d}\hat{u}_k}{\mathrm{d}t}(t)\sin(kx) = \sum_{k=0}^{N-1} -k^2\hat{u}_k(t)\sin(kx) \;\Rightarrow\; \frac{\mathrm{d}\hat{u}_k}{\mathrm{d}t}(t) = -k^2\hat{u}_k(t)$$

Heat Equation Continued

- Although in this case, we can solve the equation explicitly, generally we discretise time:
$$\frac{\partial \hat{u}_k}{\partial t} \approx \frac{\hat{u}_k^{n+1} - \hat{u}_k^n}{\Delta t} = -k^2 \hat{u}_k^n \;\Rightarrow\; \hat{u}_k^{n+1} = (1 - k^2 \Delta t)\hat{u}_k^n.$$
- The general procedure then is something like:
$$\mathbf{U}^0 \xrightarrow{\text{transform}} \hat{\mathbf{u}}^0 \xrightarrow{\text{timestep}} \hat{\mathbf{u}}^1 \xrightarrow{\text{timestep}} \cdots \xrightarrow{\text{timestep}} \hat{\mathbf{u}}^T \xrightarrow{\text{transform}^{-1}} \mathbf{U}^T$$
- Notice that differentiation in spectral space is easy: to obtain the second order derivative, we simply multiplied each mode by k^2. This is a common theme amongst spectral methods.
- We can also extend this method into multiple dimensions. A common choice for the modes in this case is
$$\phi_{\mathrm{k}}(\mathbf{x}) = e^{i\mathrm{k}\cdot\mathrm{x}}.$$

Some Problems

- In this simple formulation, a lot of problems were glossed over:
  - The evolution equations for $\hat{u}_k$ are decoupled.
  - Boundary conditions are easy.
  - Geometry is simple.
  - No non-linearity!

Some Problems

- In this simple formulation, a lot of problems were glossed over:
  - The evolution equations for $\hat{u}_k$ are decoupled.
  - Boundary conditions are easy.
  - Geometry is simple.
  - No non-linearity!
- To deal with non-linear equations, we keep both amplitudes $\hat{u}_k$ **and** a spatial grid x_n.
- For instance, with the operator $u\partial_x$, we calculate ∂_x spectrally, transform this result to normal space and then multiply by u on the grid.

Some Problems

- In this simple formulation, a lot of problems were glossed over:
  - The evolution equations for $\hat{u}_k$ are decoupled.
  - Boundary conditions are easy.
  - Geometry is simple.
  - No non-linearity!
- To deal with non-linear equations, we keep both amplitudes $\hat{u}_k$ **and** a spatial grid x_n.
- For instance, with the operator $u\partial_x$, we calculate ∂_x spectrally, transform this result to normal space and then multiply by u on the grid.
- Such methods are called *pseudo-spectral*. In particular, Fourier pseudo-spectral methods using $\phi_k(x) = e^{ikx}$ are a popular choice if the problem has periodic boundary conditions.
- Each timestep requires a pair of transforms, which will usually slow things down. In the case of Fourier modes, we can use the *Fast Fourier Transform* (FFT) which only requires $O(n \log_2 n)$ operations.

Spectral/hp Element Methods

- Less 'simple' geometries require non-uniform grids, making spectral methods difficult to implement.
- Finite and spectral *element* methods solve this by partitioning the domain Ω into N_{el} elements Ω^e.
- The general goal is to solve the equation in each element (*locally*), and then somehow combine the solutions to obtain a *global* solution.
- Modes:
  - Finite element: piecewise linear.
  - Spectral element: something like a family of Jacobi polynomials of order $\leq P$.

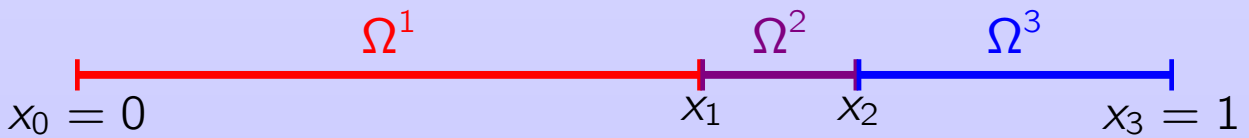
Spectral/hp Element Methods

- Less 'simple' geometries require non-uniform grids, making spectral methods difficult to implement.
- Finite and spectral *element* methods solve this by partitioning the domain Ω into N_{el} elements Ω^e.
- The general goal is to solve the equation in each element (*locally*), and then somehow combine the solutions to obtain a *global* solution.
- Modes:
  - Finite element: piecewise linear.
  - Spectral element: something like a family of Jacobi polynomials of order $\leq P$.
- In spectral/**hp** methods, we therefore have two ways of increasing accuracy:
  - p-refinement: increase the polynomial order in each element.
  - h-refinement: decrease the size of elements.

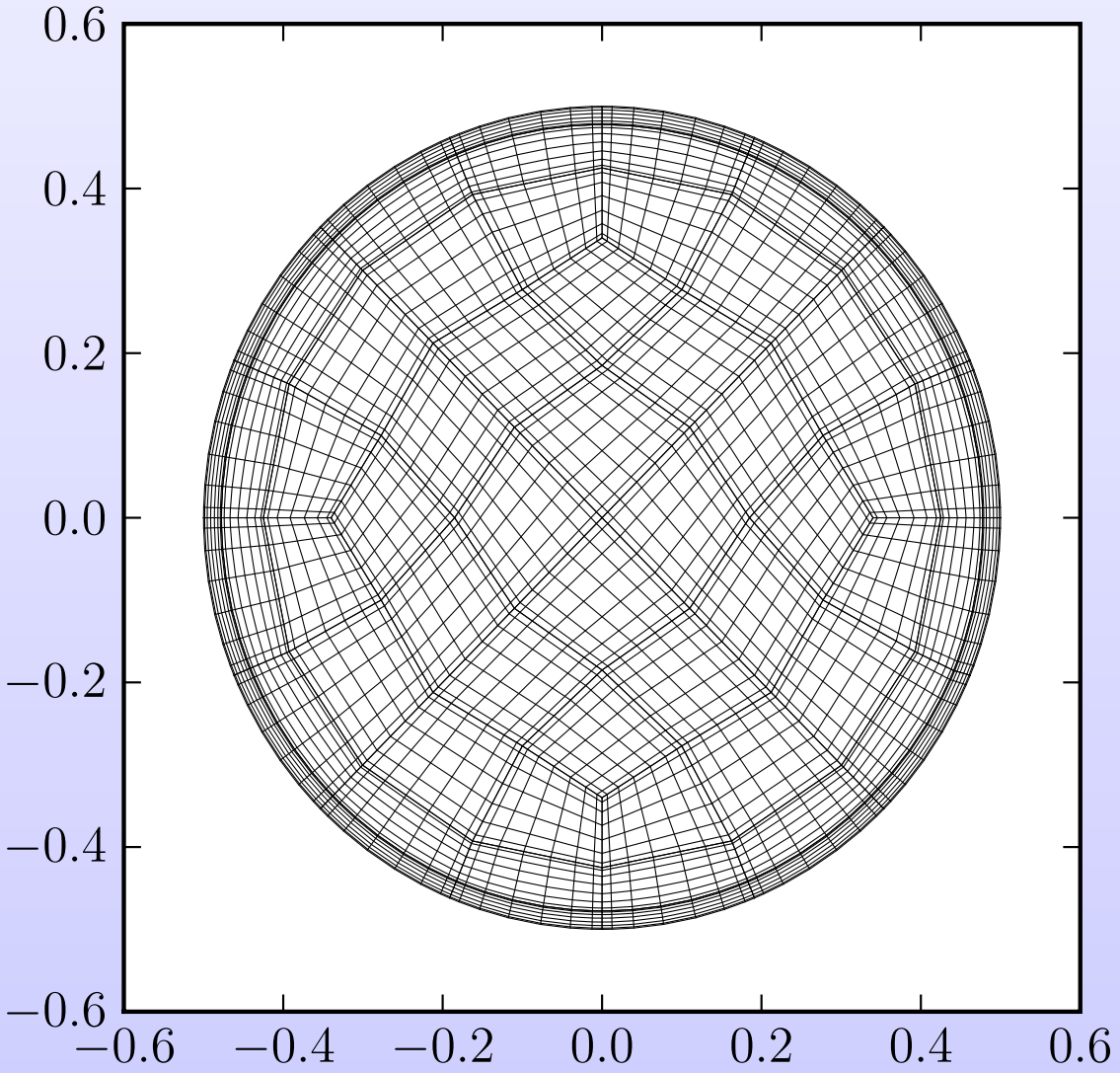
Spectral/hp Element Methods

- Less 'simple' geometries require non-uniform grids, making spectral methods difficult to implement.
- Finite and spectral *element* methods solve this by partitioning the domain Ω into N_{el} elements Ω^e.
- The general goal is to solve the equation in each element (*locally*), and then somehow combine the solutions to obtain a *global* solution.
- Modes:
  - Finite element: piecewise linear.
  - Spectral element: something like a family of Jacobi polynomials of order $\leq P$.
- In spectral/**hp** methods, we therefore have two ways of increasing accuracy:
  - *p*-refinement: increase the polynomial order in each element.
  - *h*-refinement: decrease the size of elements.

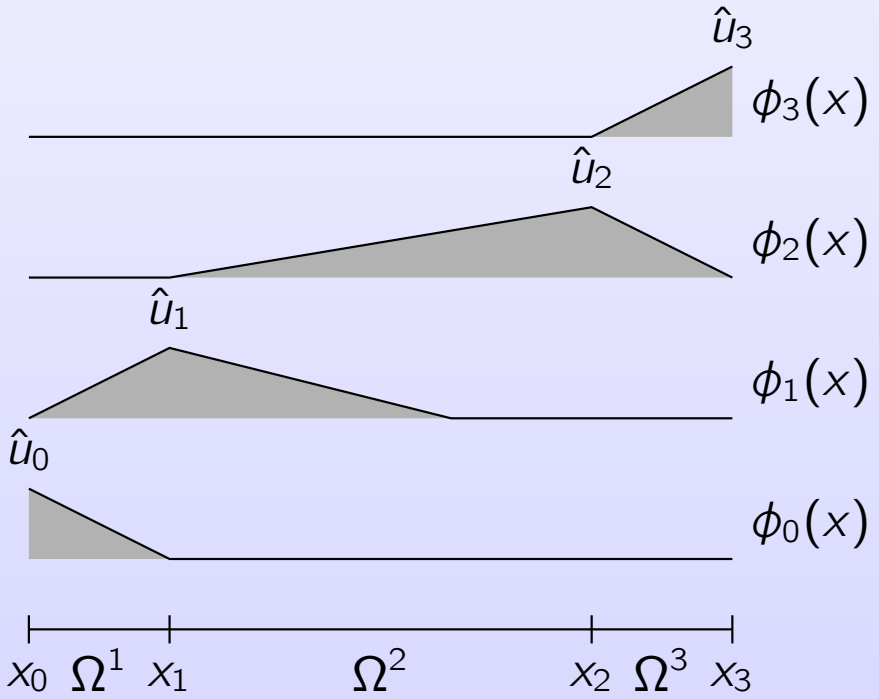
1D Example: $\Omega = [0, 1] = \bigcup_{e=1}^{3} \Omega^e$



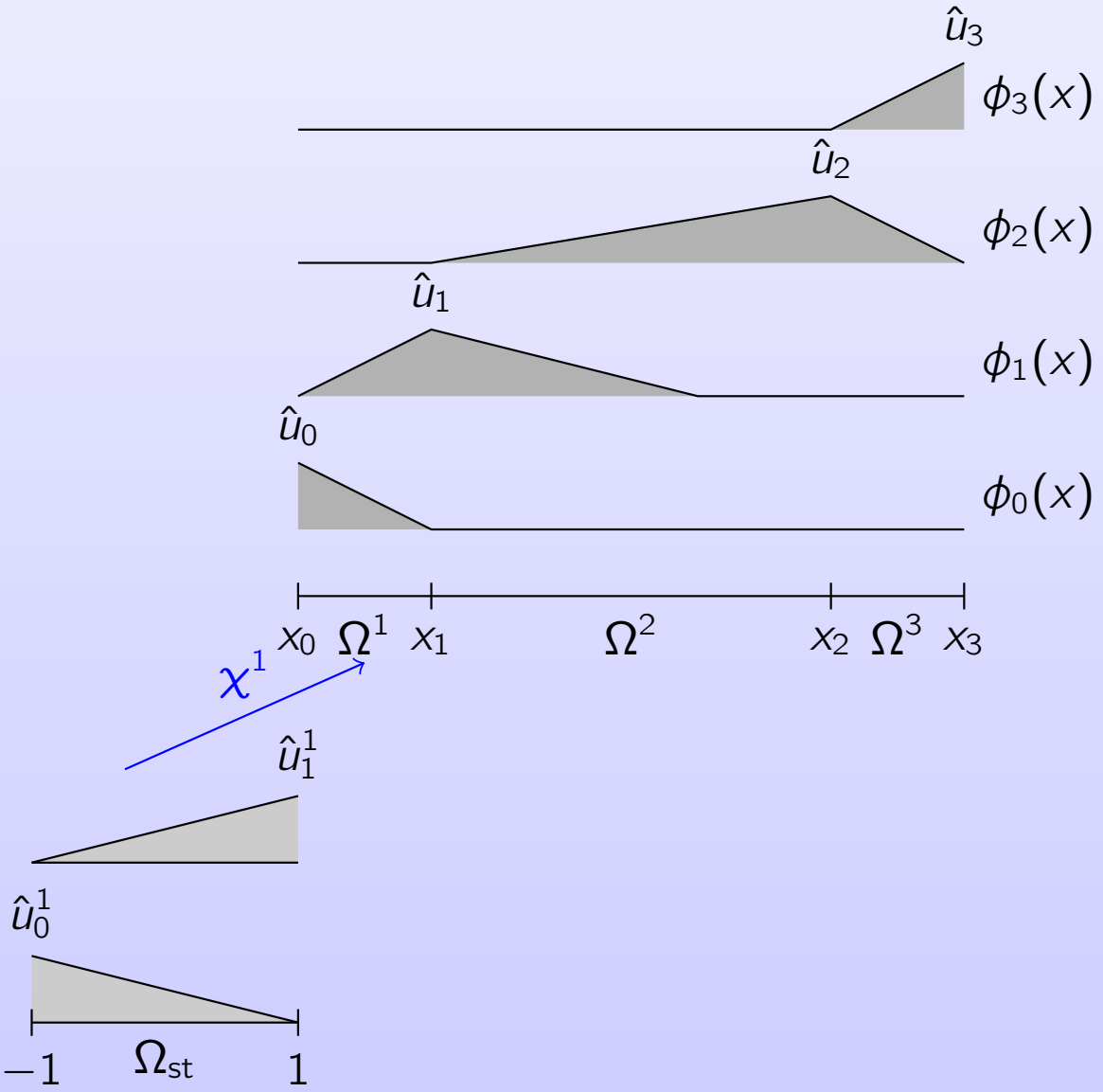
A more complex example



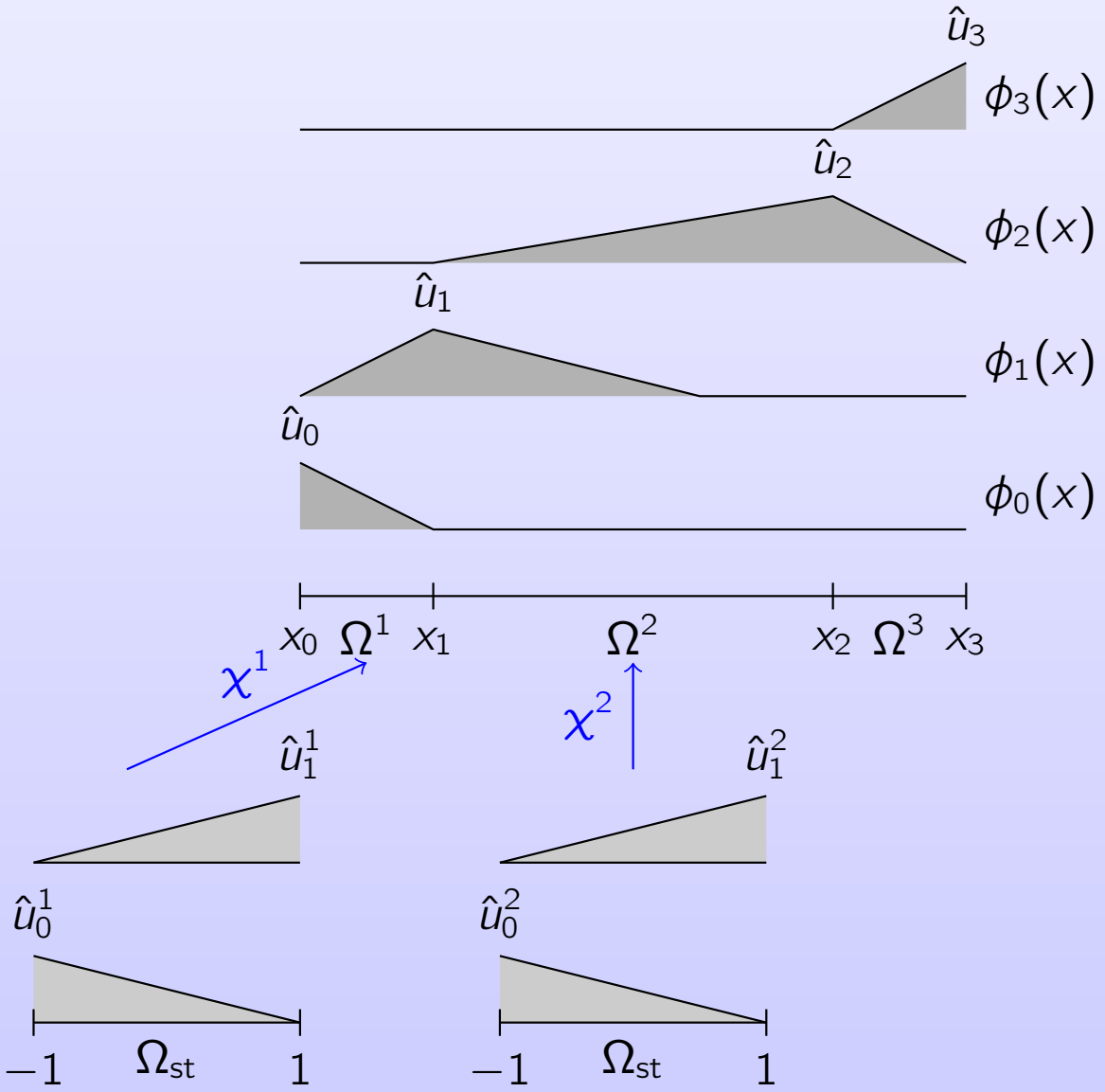
Local and Global Modes: 1D Example



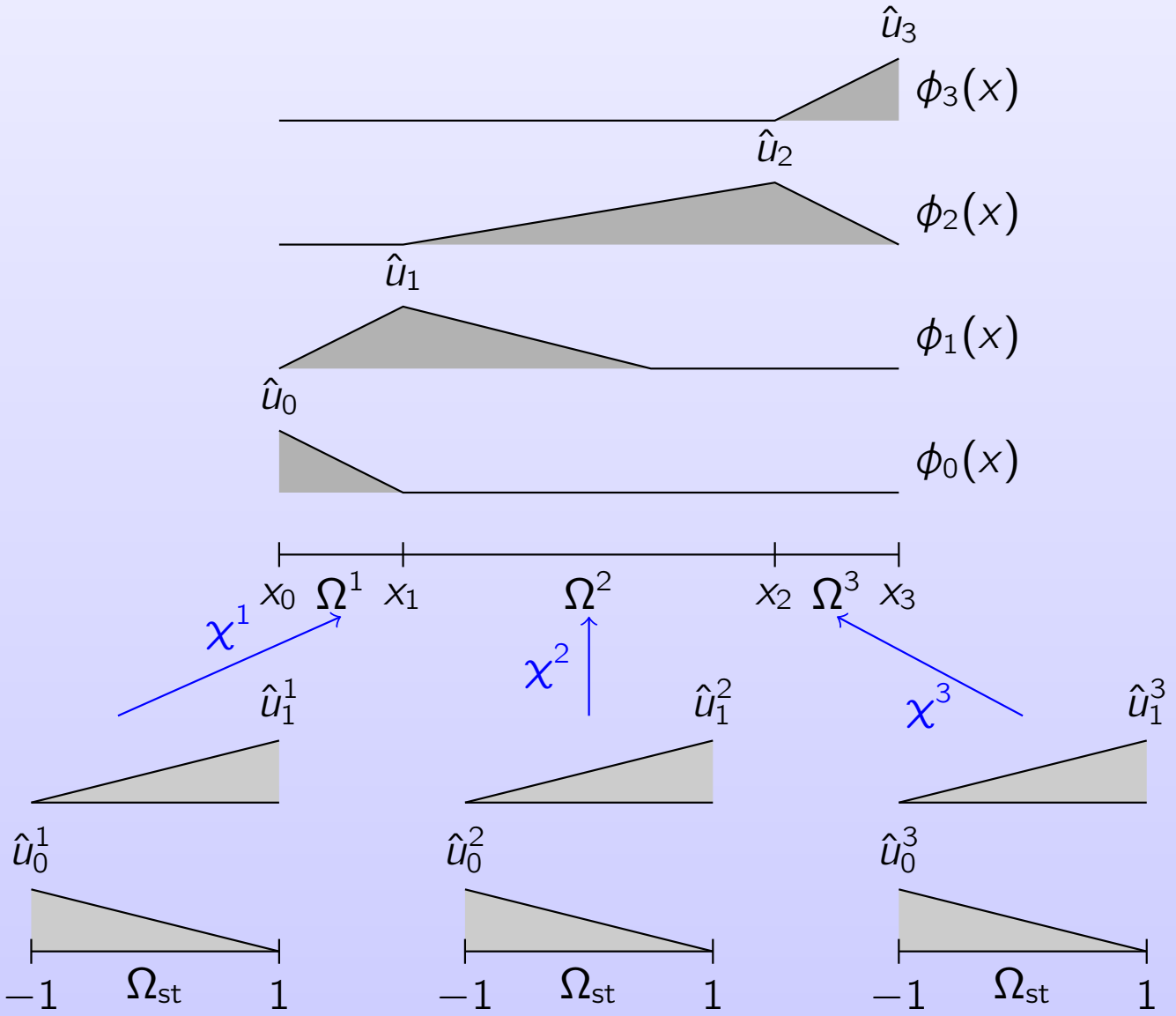
Local and Global Modes: 1D Example



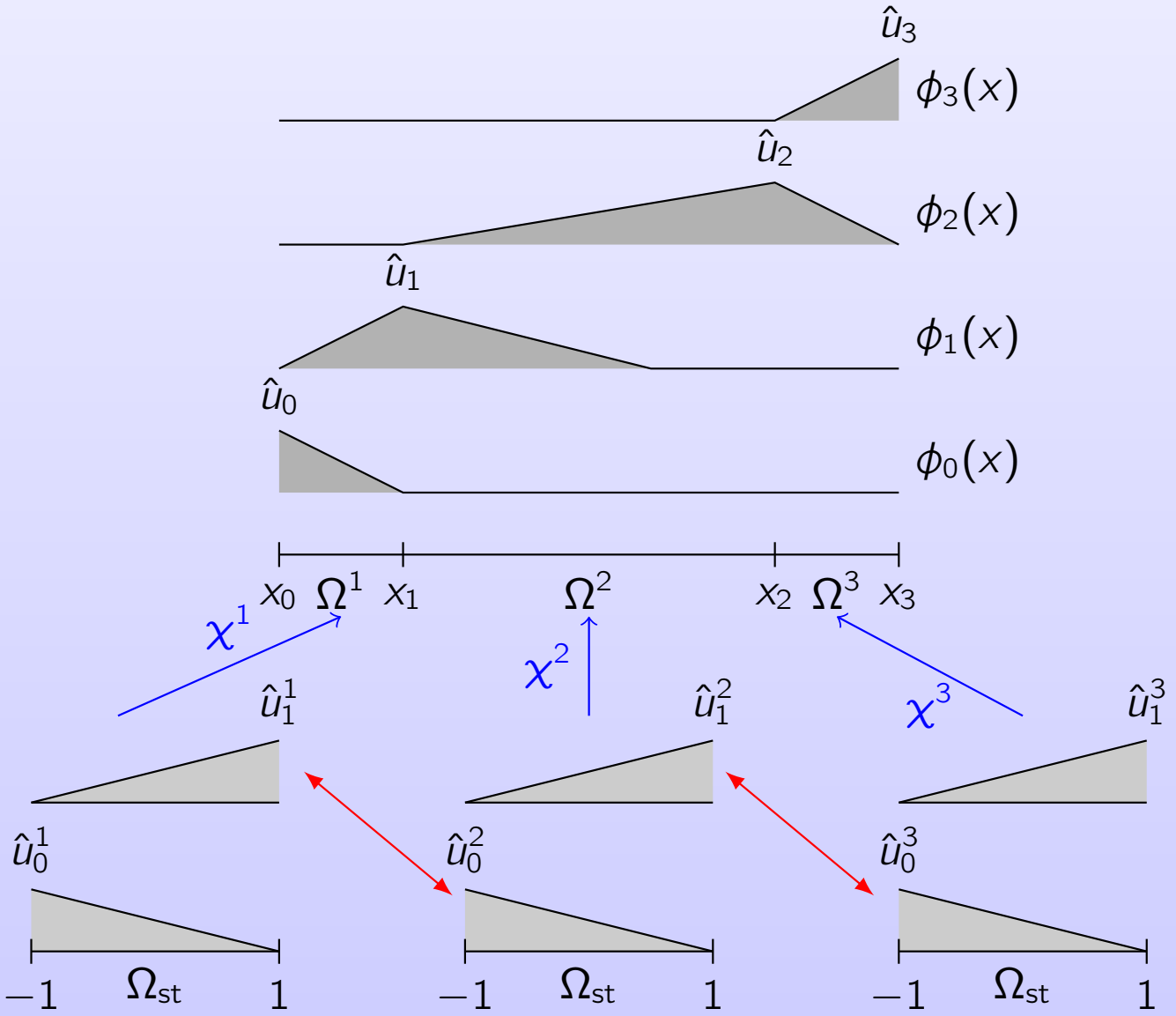
Local and Global Modes: 1D Example



Local and Global Modes: 1D Example



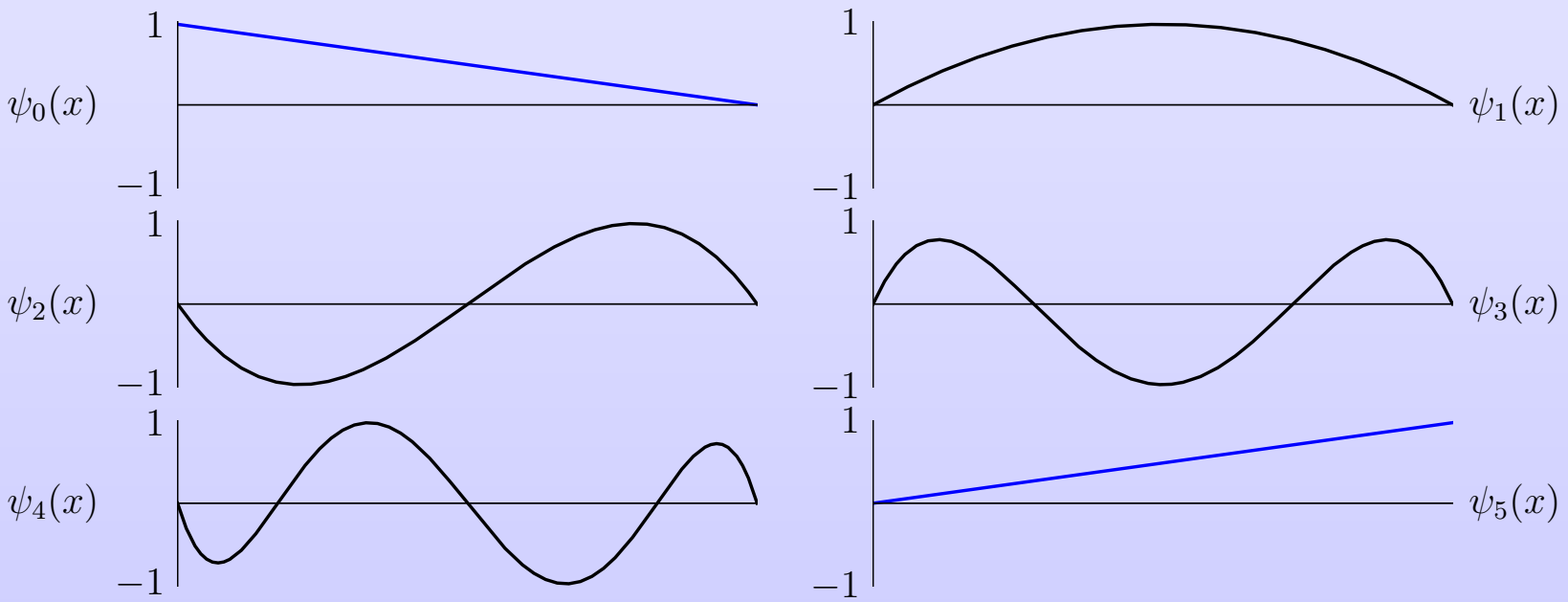
Local and Global Modes: 1D Example



Choosing Modes

Choosing modes is a non-trivial task. Mostly we choose subsets of the Jacobi polynomials $J_p^{\alpha,\beta}(\xi)$. Typical 'modal' basis for $\xi \in \Omega_{\text{st}}$:

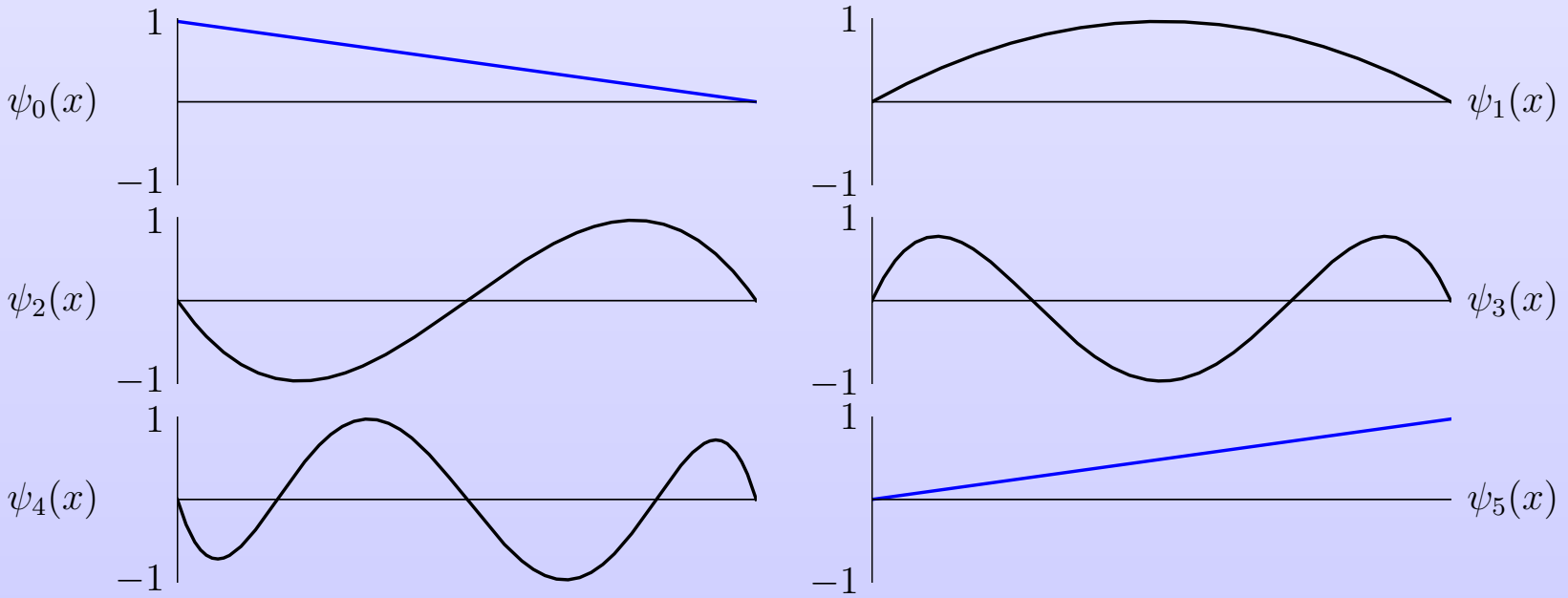
$$\psi_p(\xi) = \begin{cases} \frac{1}{2}(1-\xi), & p = 0 \\ \frac{1}{4}(1+\xi)(1-\xi)J_{p-1}^{1,1}(x), & 0 < p < P \\ \frac{1}{2}(1+\xi), & p = P \end{cases}$$



Choosing Modes

Choosing modes is a non-trivial task. Mostly we choose subsets of the Jacobi polynomials $J_p^{\alpha,\beta}(\xi)$. Typical 'modal' basis for $\xi \in \Omega_{\text{st}}$:

$$\psi_p(\xi) = \begin{cases} \frac{1}{2}(1-\xi), & p = 0 \quad \text{boundary mode} \\ \frac{1}{4}(1+\xi)(1-\xi)J_{p-1}^{1,1}(x), & 0 < p < P \\ \frac{1}{2}(1+\xi), & p = P \quad \text{boundary mode} \end{cases}$$



Putting it all together

- To relate the global and local modes, we write

$$\hat{\mathbf{u}}_l = \mathcal{A}\hat{\mathbf{u}}_g.$$

  $\mathcal{A}$ is called the *assembly matrix*.
- Consider the problem of *Galerkin projection*: find $u^\delta \in X^\delta$ such that

$$(v^\delta, u^\delta) = (v^\delta, f), \quad \forall\, v^\delta \in X^\delta.$$

Putting it all together

- To relate the global and local modes, we write

$$\hat{\mathbf{u}}_l = \mathcal{A}\hat{\mathbf{u}}_g.$$

  $\mathcal{A}$ is called the *assembly matrix*.
- Consider the problem of *Galerkin projection*: find $u^\delta \in X^\delta$ such that

$$(v^\delta, u^\delta) = (v^\delta, f), \quad \forall\, v^\delta \in X^\delta.$$

- We substitute our approximation (in terms of the global modes $\phi_k(x)$) and re-write the problem in matrix form:

$$\mathbf{v}^\top(\mathbf{M}^G\hat{\mathbf{u}} = \mathbf{f}) \;\Rightarrow\; \mathbf{M}^G\hat{\mathbf{u}} = \mathbf{f} \;\Rightarrow\; \hat{\mathbf{u}} = [\mathbf{M}^G]^{-1}\mathbf{f}$$

  where

$$\mathbf{M}^G_{pq} = (\phi_p, \phi_q), \hat{\mathbf{u}} = (\hat{u}_0, \ldots, \hat{u}_{N-1}), \mathbf{f}_p = (\phi_p, f)$$

- $\mathbf{M}^G$ is called the *mass matrix*.

Putting it all together

- To relate the global and local modes, we write

$$\hat{\mathbf{u}}_l = \mathcal{A}\hat{\mathbf{u}}_g.$$

  $\mathcal{A}$ is called the *assembly matrix*.
- Consider the problem of *Galerkin projection*: find $u^\delta \in X^\delta$ such that

$$(v^\delta, u^\delta) = (v^\delta, f), \quad \forall\, v^\delta \in X^\delta.$$

- We substitute our approximation (in terms of the global modes $\phi_k(x)$) and re-write the problem in matrix form:

$$\mathbf{v}^\top(\mathbf{M}^G\hat{\mathbf{u}} = \mathbf{f}) \;\Rightarrow\; \mathbf{M}^G\hat{\mathbf{u}} = \mathbf{f} \;\Rightarrow\; \hat{\mathbf{u}} = [\mathbf{M}^G]^{-1}\mathbf{f}$$

  where

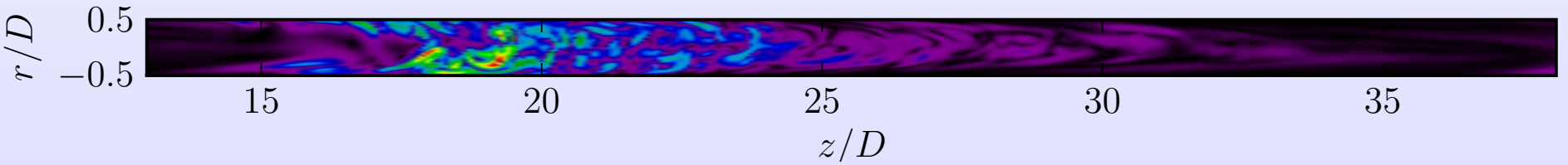
$$\mathbf{M}^G_{pq} = (\phi_p, \phi_q), \hat{\mathbf{u}} = (\hat{u}_0, \ldots, \hat{u}_{N-1}), \mathbf{f}_p = (\phi_p, f)$$

- $\mathbf{M}^G$ is called the *mass matrix*.
- In reality, we construct the *elemental* mass matrices $\mathbf{M}^e$ in a similar fashion, and then use $\mathcal{A}$ to construct $\mathbf{M}^G$:

$$\mathbf{M}^G = \mathcal{A}^\top\mathbf{M}^e\mathcal{A}.$$

Large-scale structures

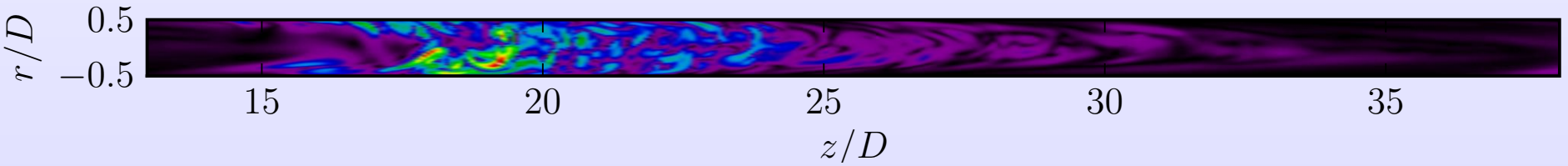
- For $1900 \leq \mathrm{Re} \leq 2200$, we observe *puffs* in the fluid. These turbulent structures co-exist with laminar flow.



- This puff was recorded at $\mathrm{Re} = 2000$ and has length $L \approx 25D$, so larger pipe lengths are needed to capture their behaviour.

Large-scale structures

- For $1900 \leq \mathrm{Re} \leq 2200$, we observe *puffs* in the fluid. These turbulent structures co-exist with laminar flow.



- This puff was recorded at $\mathrm{Re} = 2000$ and has length $L \approx 25D$, so larger pipe lengths are needed to capture their behaviour.
- Because correctly resolved DNS is computationally expensive, most turbulence simulations use short pipes.
- The most recent numerical simulations of these structures have been in a pipe of length $L = 16\pi D \approx 50D$ [Pringle & Kerswell, *PRL*, 2007].
- However, experimental pipes are typically several hundred diameters long [Darbyshire & Mullin, *JFM*, 1995]. Thanks to Moore's law, we can now begin to study these pipe lengths using DNS.

Simulations

- Parameters for the simulations were:

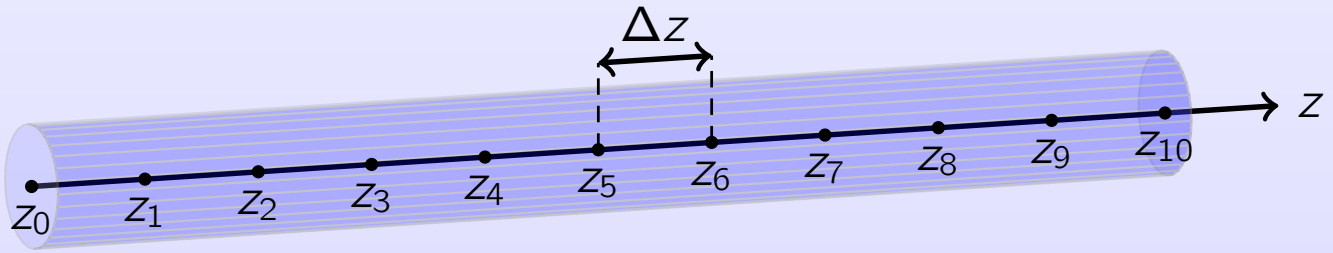
L	$16\pi D \approx 50D$	$48\pi D \approx 150D$
N_z	768	2048
N_{proc}	64	128
Re	$1900 \leq \text{Re} \leq 2150$	$2000 \leq \text{Re} \leq 2500$
Δt	2×10^{-3}	2×10^{-3}

- The flow is driven using a constant volumetric flux $q = 1$ through a circular cross-section of the pipe $\Rightarrow$ U_B is fixed.
- The simulation is started with uniform turbulence using a short run at $\text{Re} = 5000$. We then reduce Re as follows:

$5000 \rightarrow 4000 \rightarrow 3000$	500 time units
$2800 \rightarrow 2650 \rightarrow 2500 \rightarrow 2350$	1000 time units
$2200 \rightarrow 2150 \rightarrow 2100 \rightarrow 2050 \rightarrow \cdots$	2000 time units

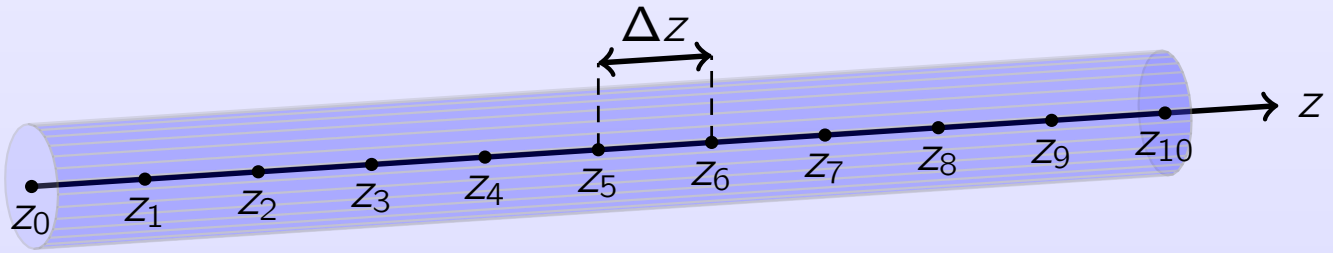
History Plots

- Field data is too large to record regularly. Instead we measure *history data*: every 0.1 time units, we record velocity field data from points along the axis of the pipe.



History Plots

- Field data is too large to record regularly. Instead we measure *history data*: every 0.1 time units, we record velocity field data from points along the axis of the pipe.



- For the velocity field $\mathbf{u} = (u, v, w)$, we construct the quantity

$$q(z,t) = \sqrt{u^2 + v^2}\Big|_{(x=0,y=0,z,t)}$$

and then change to a moving frame of reference by applying

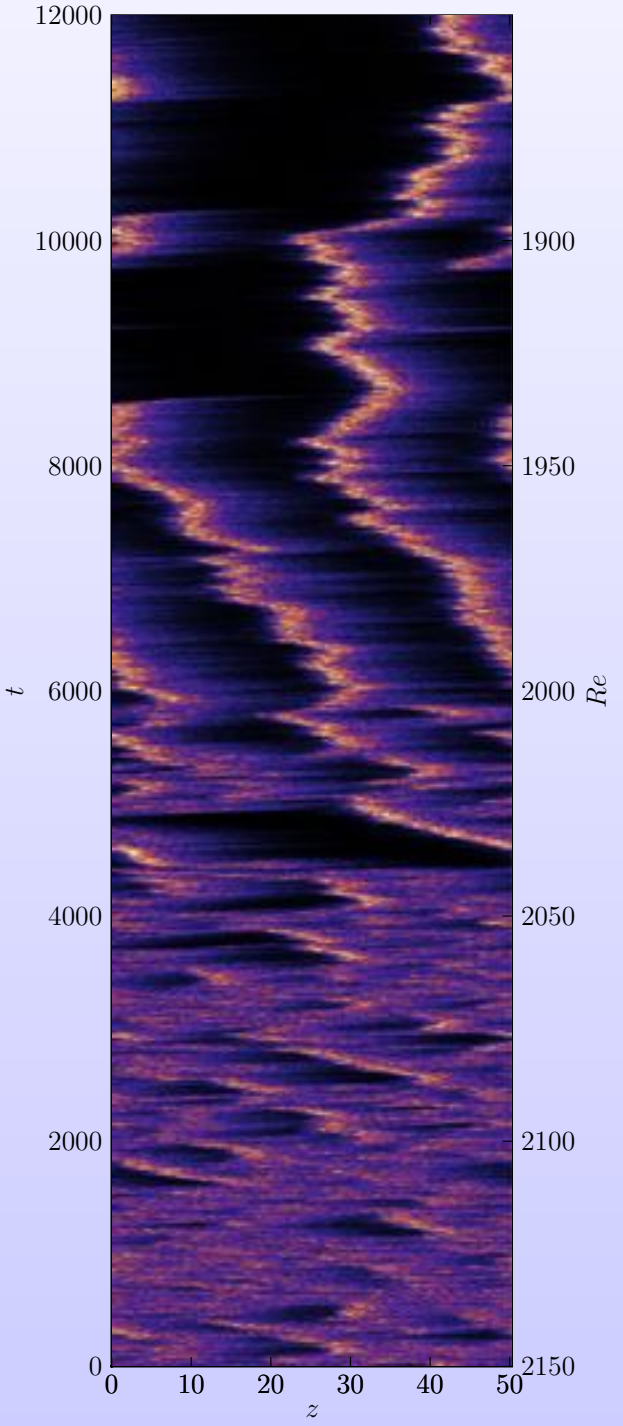
$$q(z,t) \to q(z - ct, t)$$

where c is, for example, the speed of a puff.

- This data produces a space-time contour plot which is useful for analysing general qualities of the fluid.

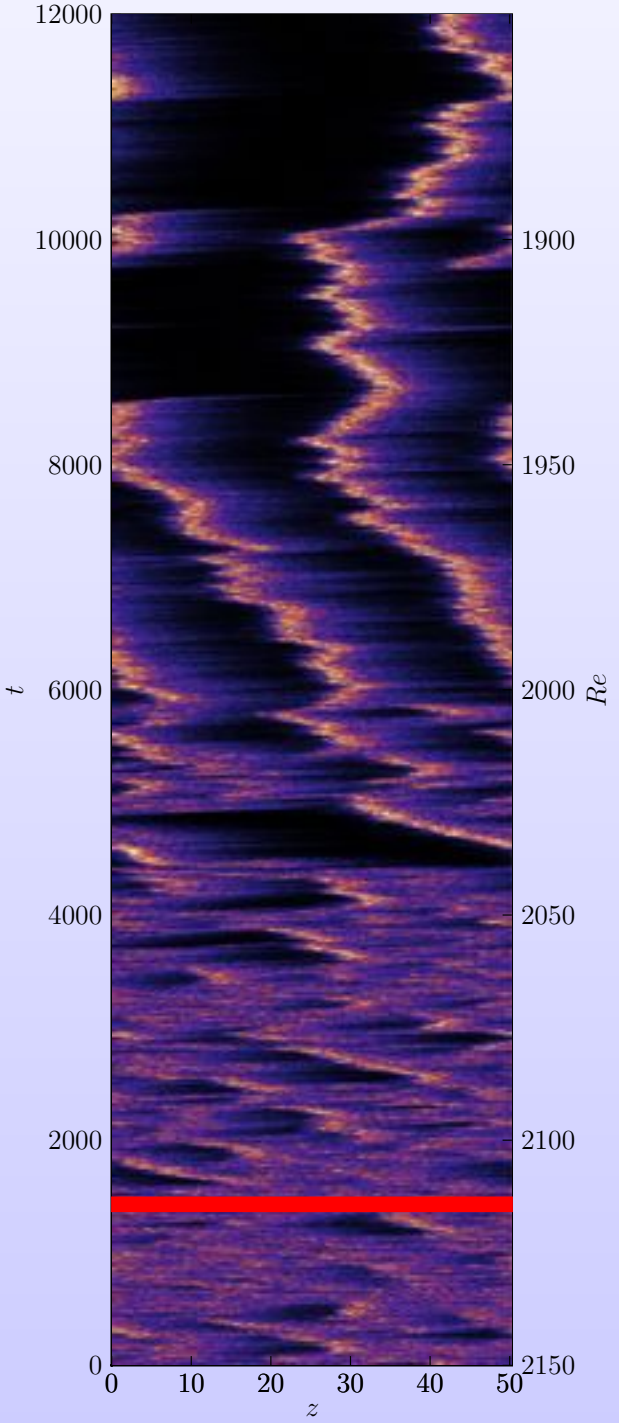
$L = 50D$ History Data

- $1900 \leq \mathrm{Re} \leq 2150$.
- $0 \leq t \leq 12000$.
- $c = U_B$.
- Re-laminarization occurs at $\mathrm{Re} = 1750$.
- The data took roughly 4.1 CPU years to generate.

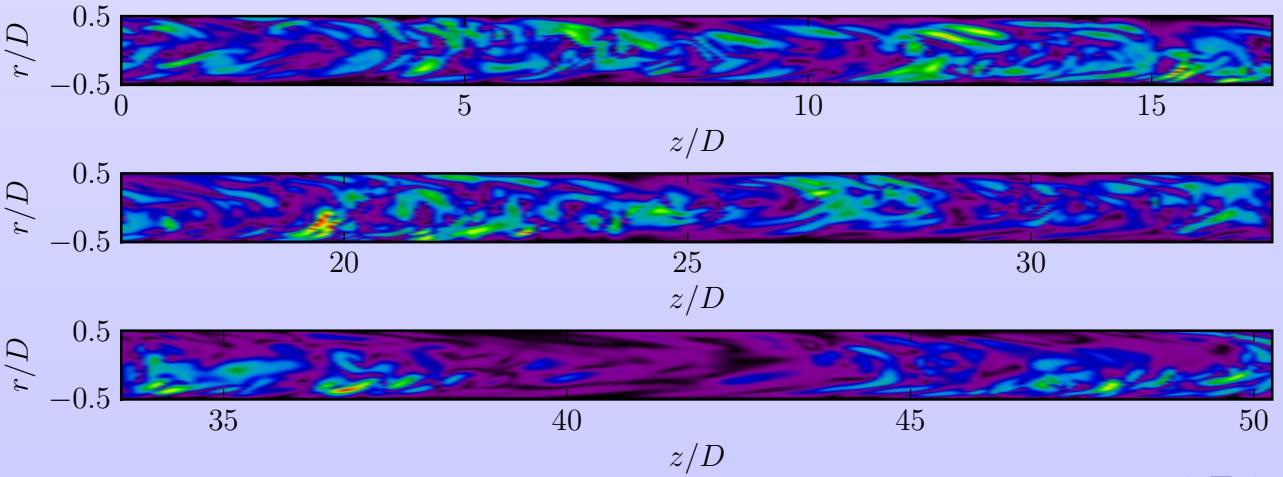


$L = 50D$ History Data

- $1900 \leq \mathrm{Re} \leq 2150$.
- $0 \leq t \leq 12000$.
- $c = U_B$.
- Re-laminarization occurs at $\mathrm{Re} = 1750$.
- The data took roughly 4.1 CPU years to generate.

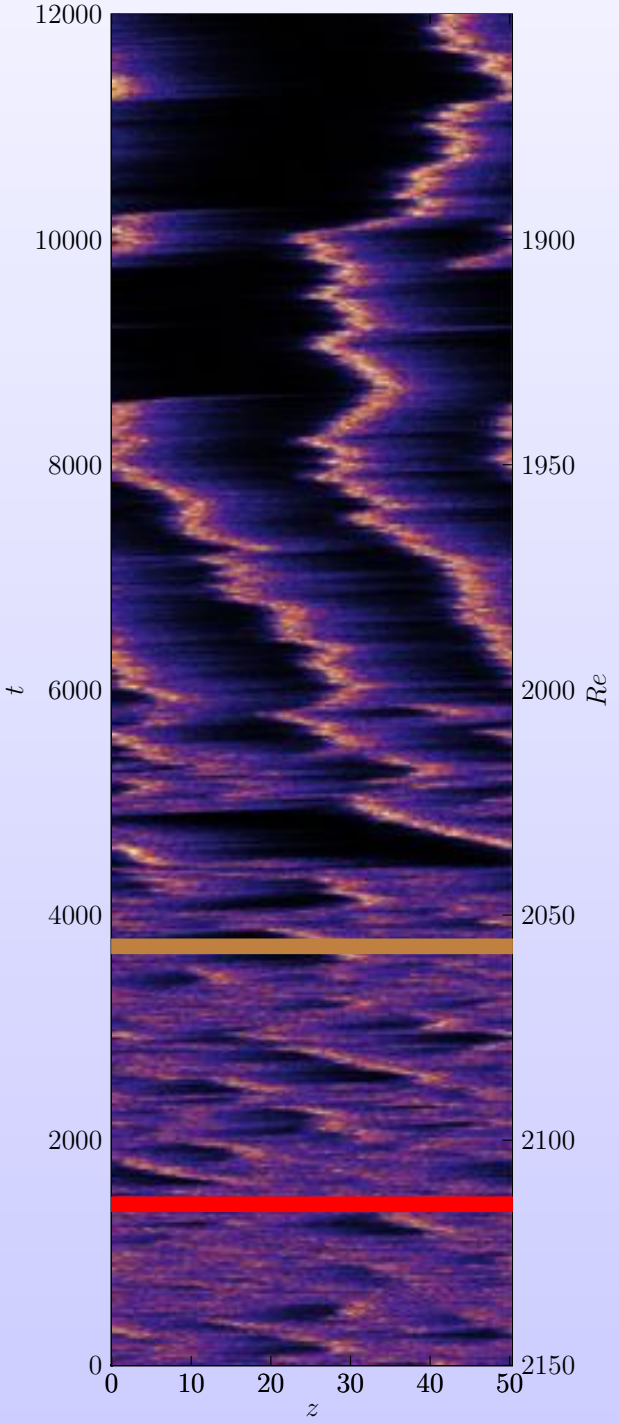


$\mathrm{Re} = 2150, t = 1500$

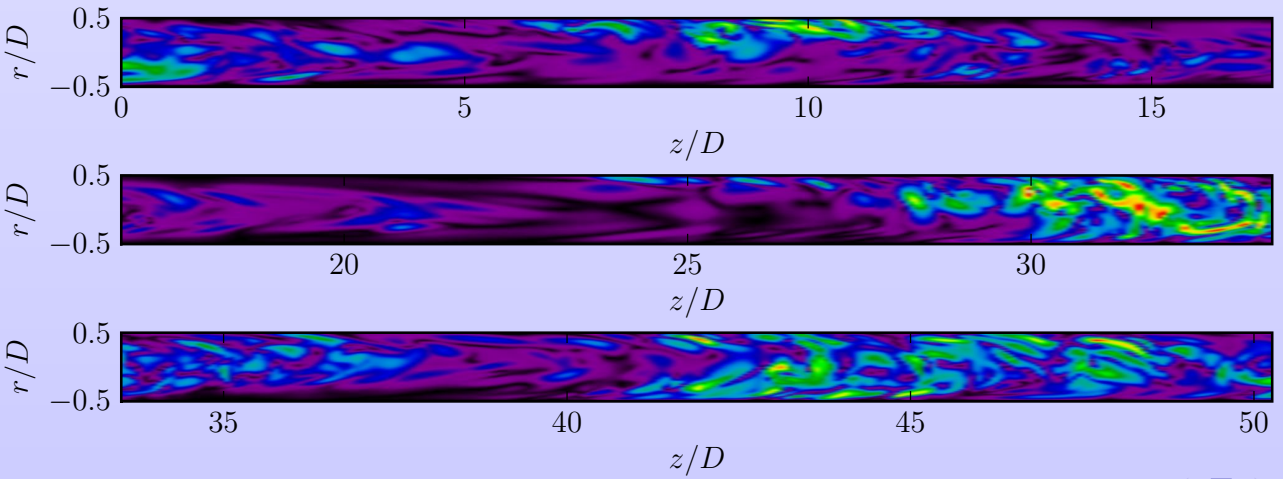


$L = 50D$ History Data

- $1900 \leq \mathrm{Re} \leq 2150$.
- $0 \leq t \leq 12000$.
- $c = U_B$.
- Re-laminarization occurs at $\mathrm{Re} = 1750$.
- The data took roughly 4.1 CPU years to generate.

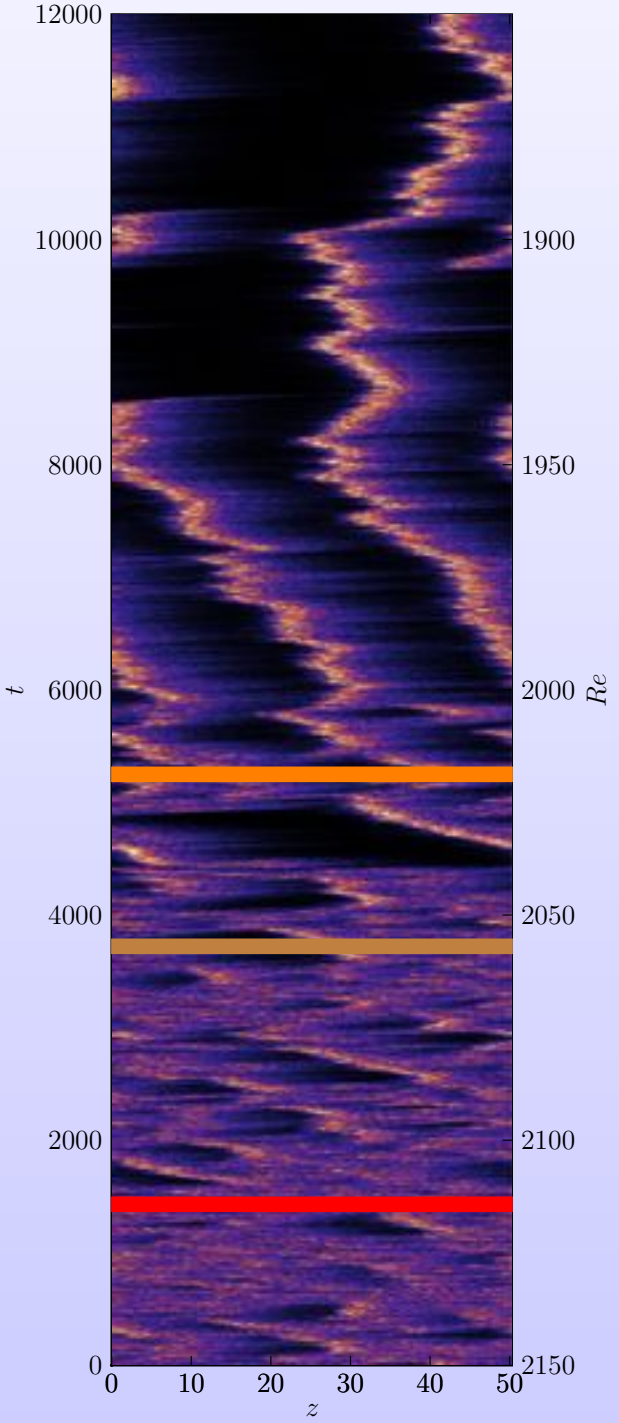


$\mathrm{Re} = 2100, t = 3900$

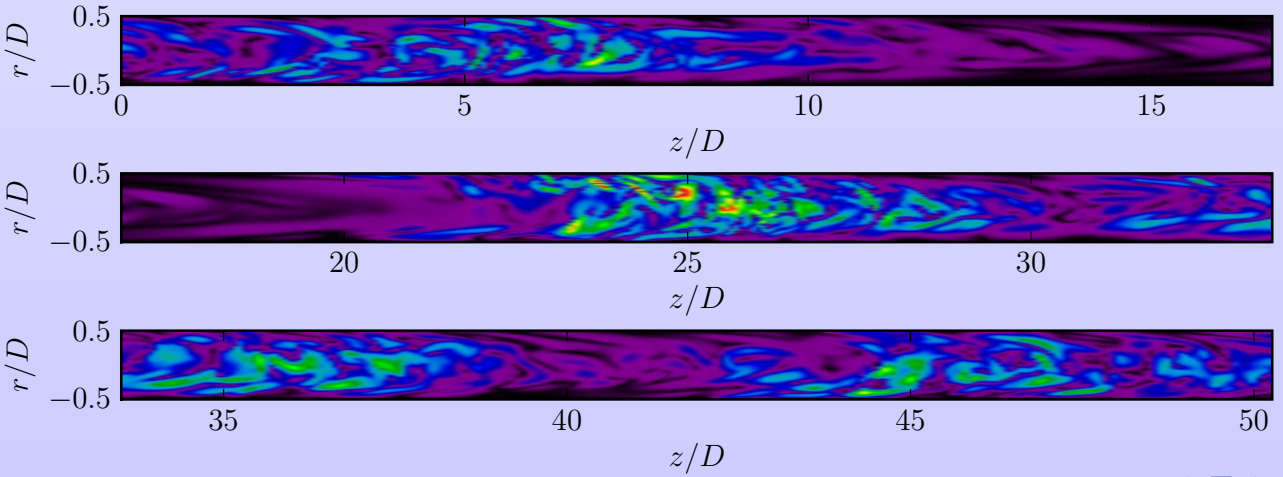


$L = 50D$ History Data

- $1900 \leq \mathrm{Re} \leq 2150$.
- $0 \leq t \leq 12000$.
- $c = U_B$.
- Re-laminarization occurs at $\mathrm{Re} = 1750$.
- The data took roughly 4.1 CPU years to generate.

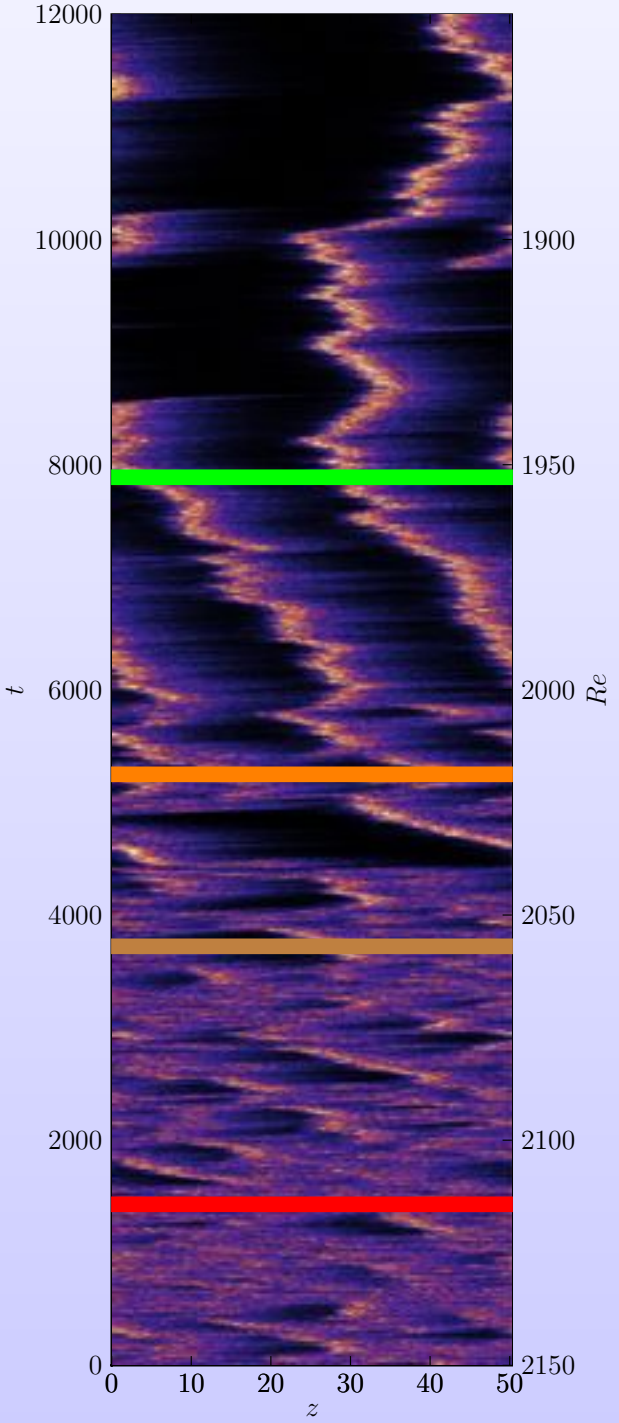


$\mathrm{Re} = 2050,\ t = 5400$

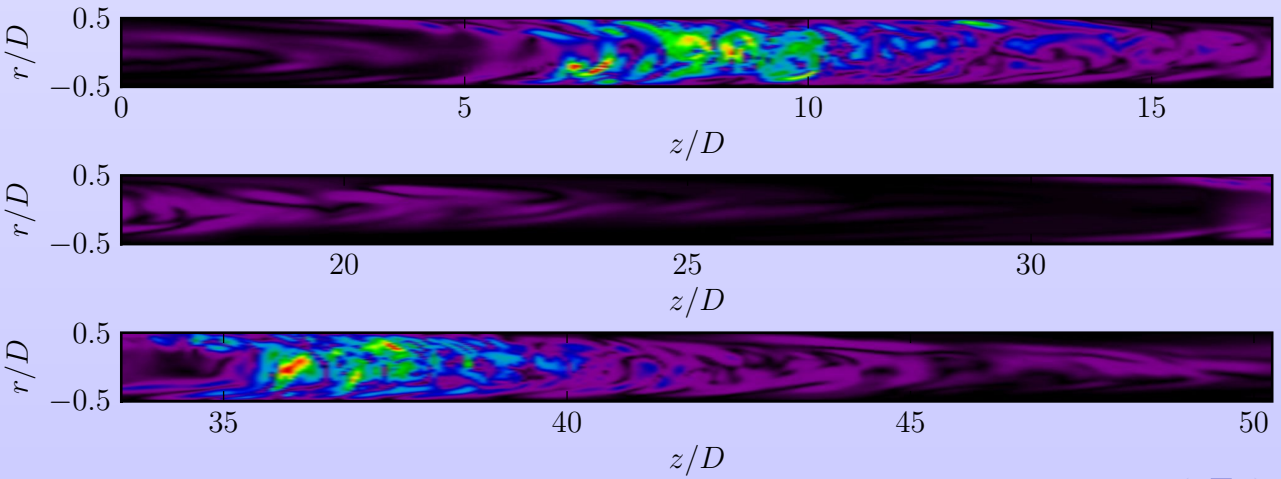


$L = 50D$ History Data

- $1900 \leq \mathrm{Re} \leq 2150$.
- $0 \leq t \leq 12000$.
- $c = U_B$.
- Re-laminarization occurs at $\mathrm{Re} = 1750$.
- The data took roughly 4.1 CPU years to generate.

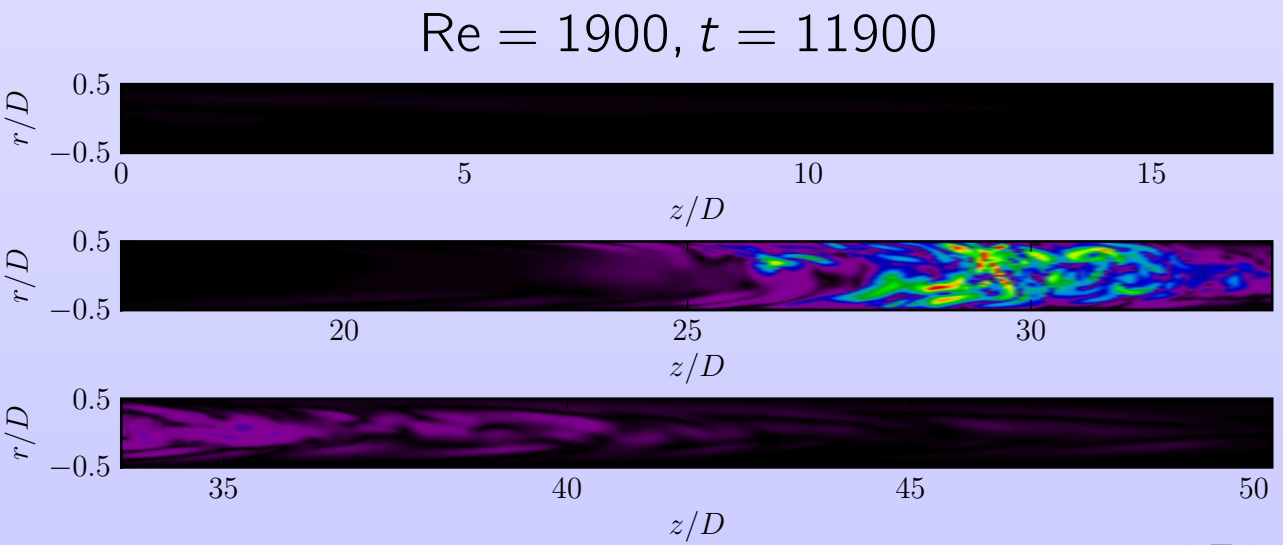


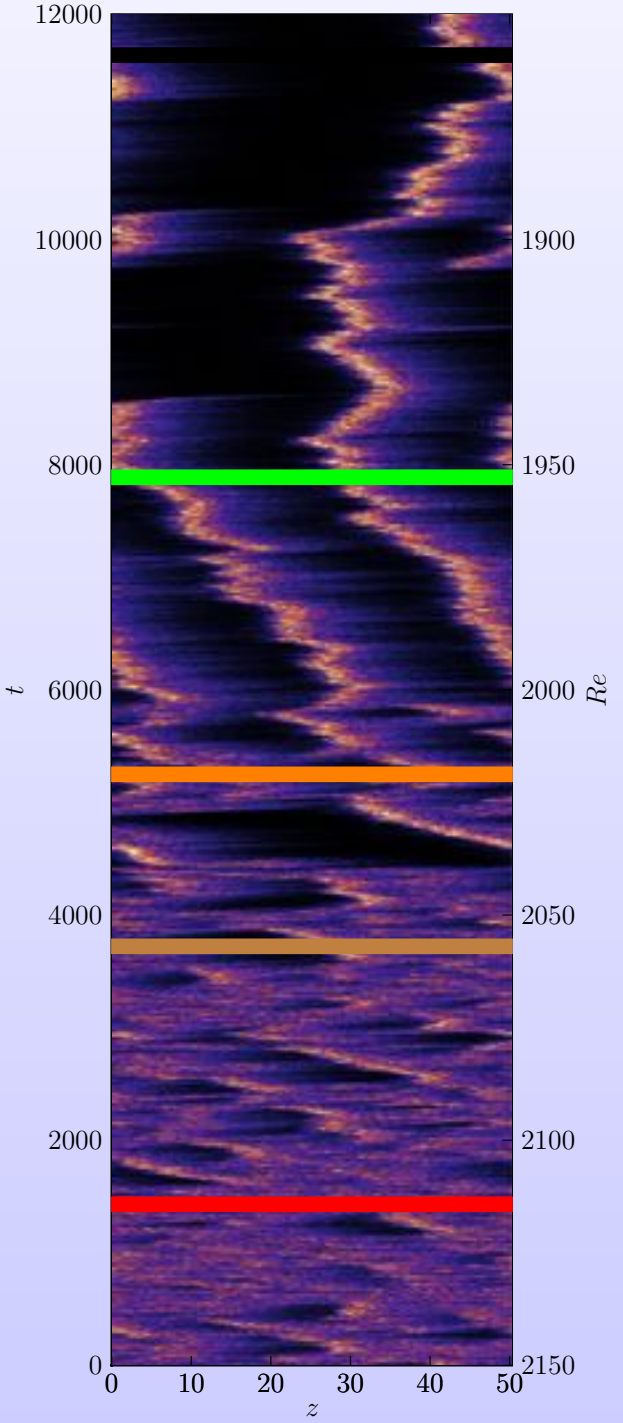
$\mathrm{Re} = 2000$, $t = 7950$



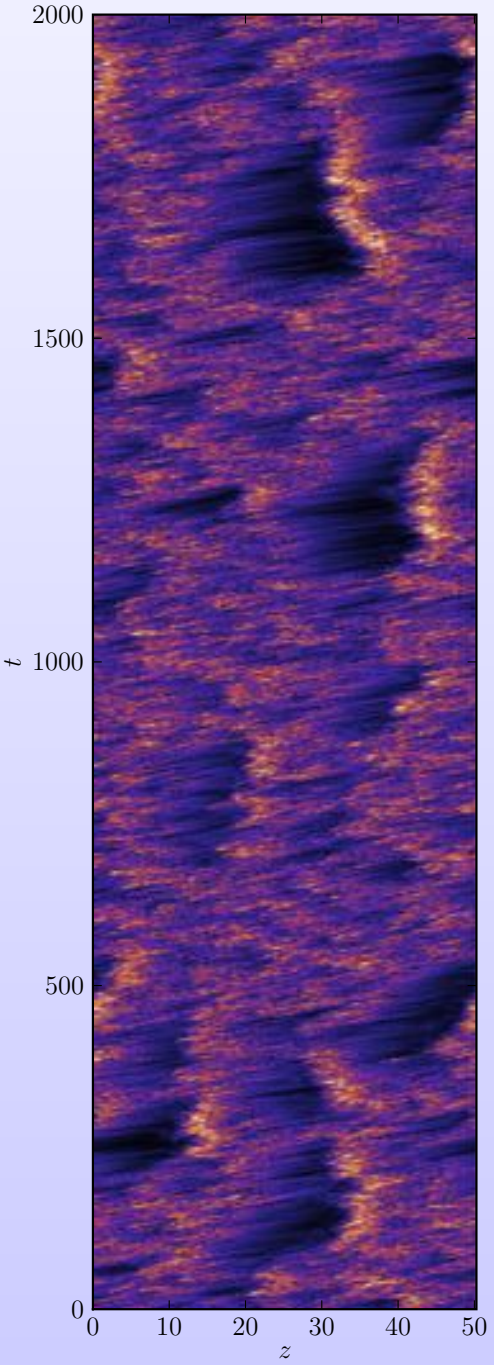
$L = 50D$ History Data

- $1900 \leq \mathrm{Re} \leq 2150$.
- $0 \leq t \leq 12000$.
- $c = U_B$.
- Re-laminarization occurs at $\mathrm{Re} = 1750$.
- The data took roughly 4.1 CPU years to generate.

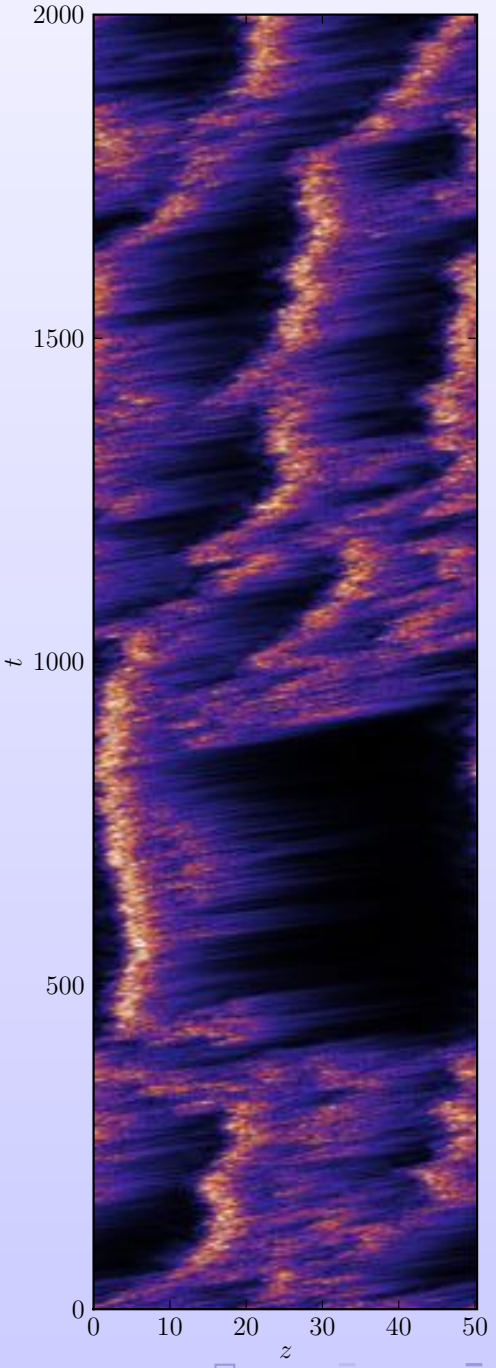




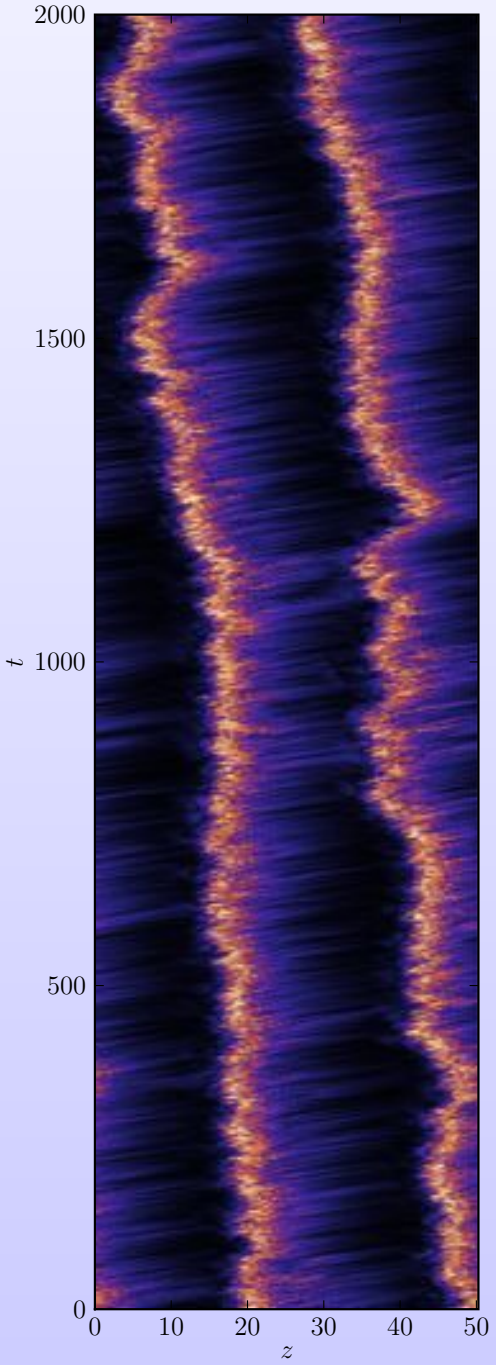
Re = 2150



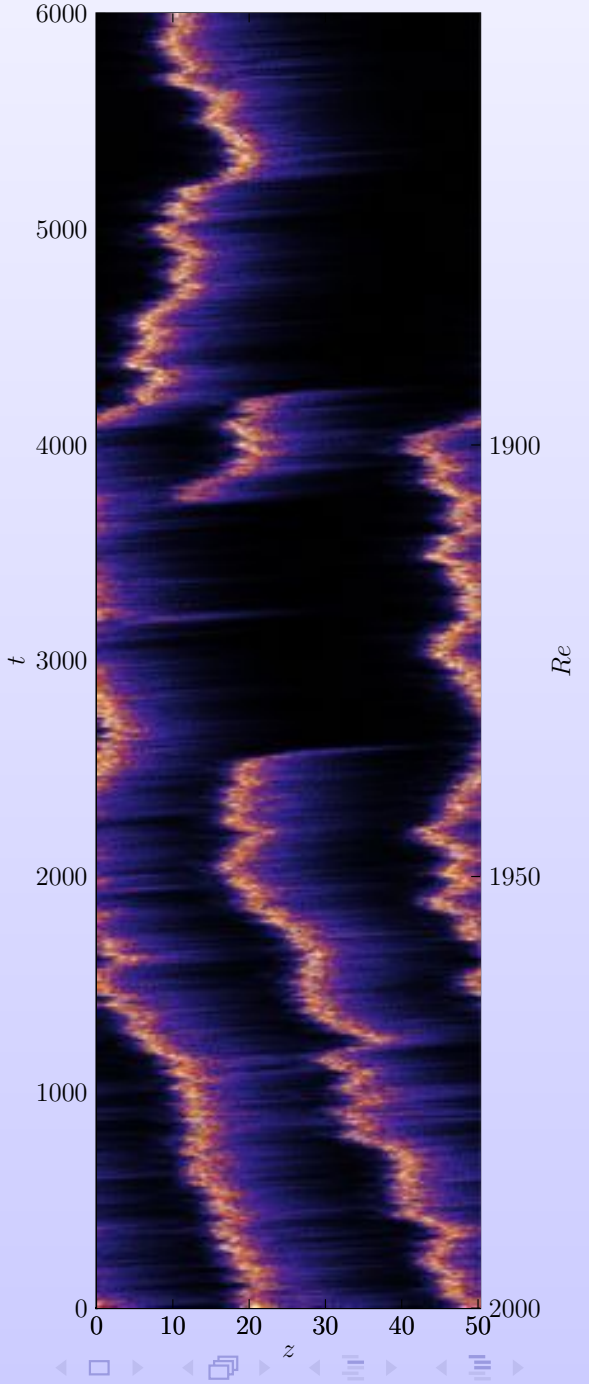
Re = 2050

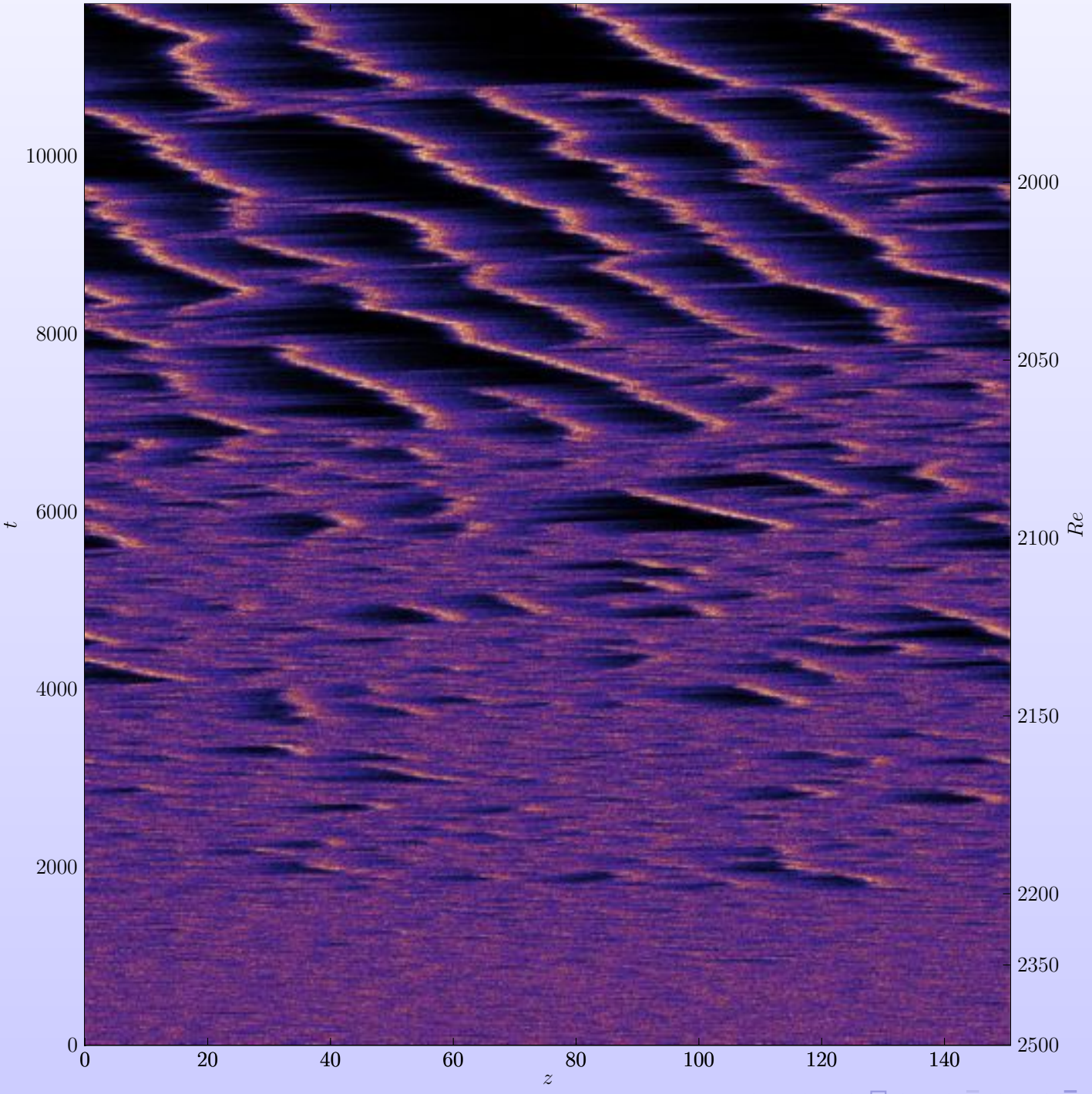


$\mathrm{Re} = 2000$



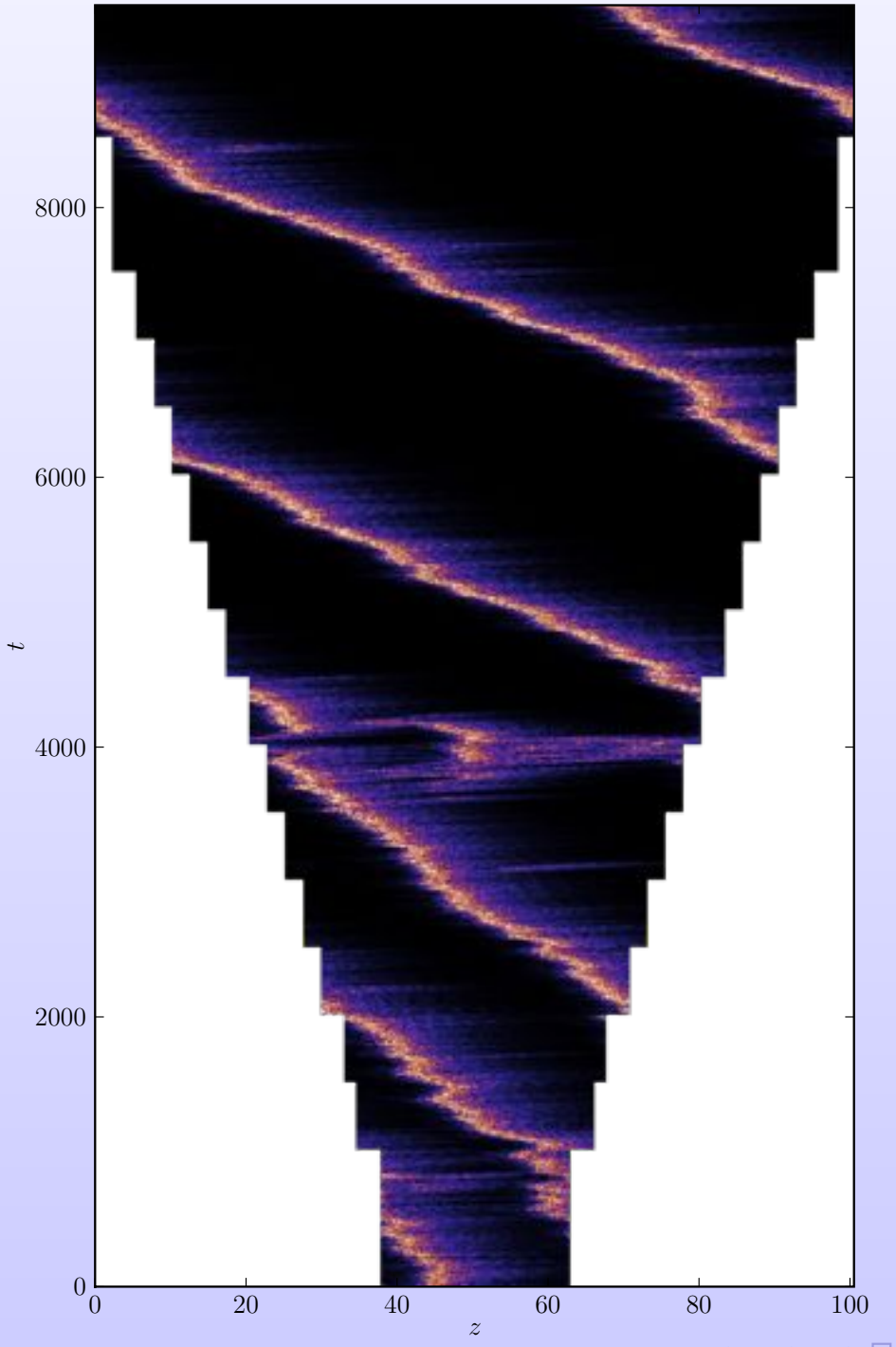
$1900 \leq \mathrm{Re} \leq 2000$

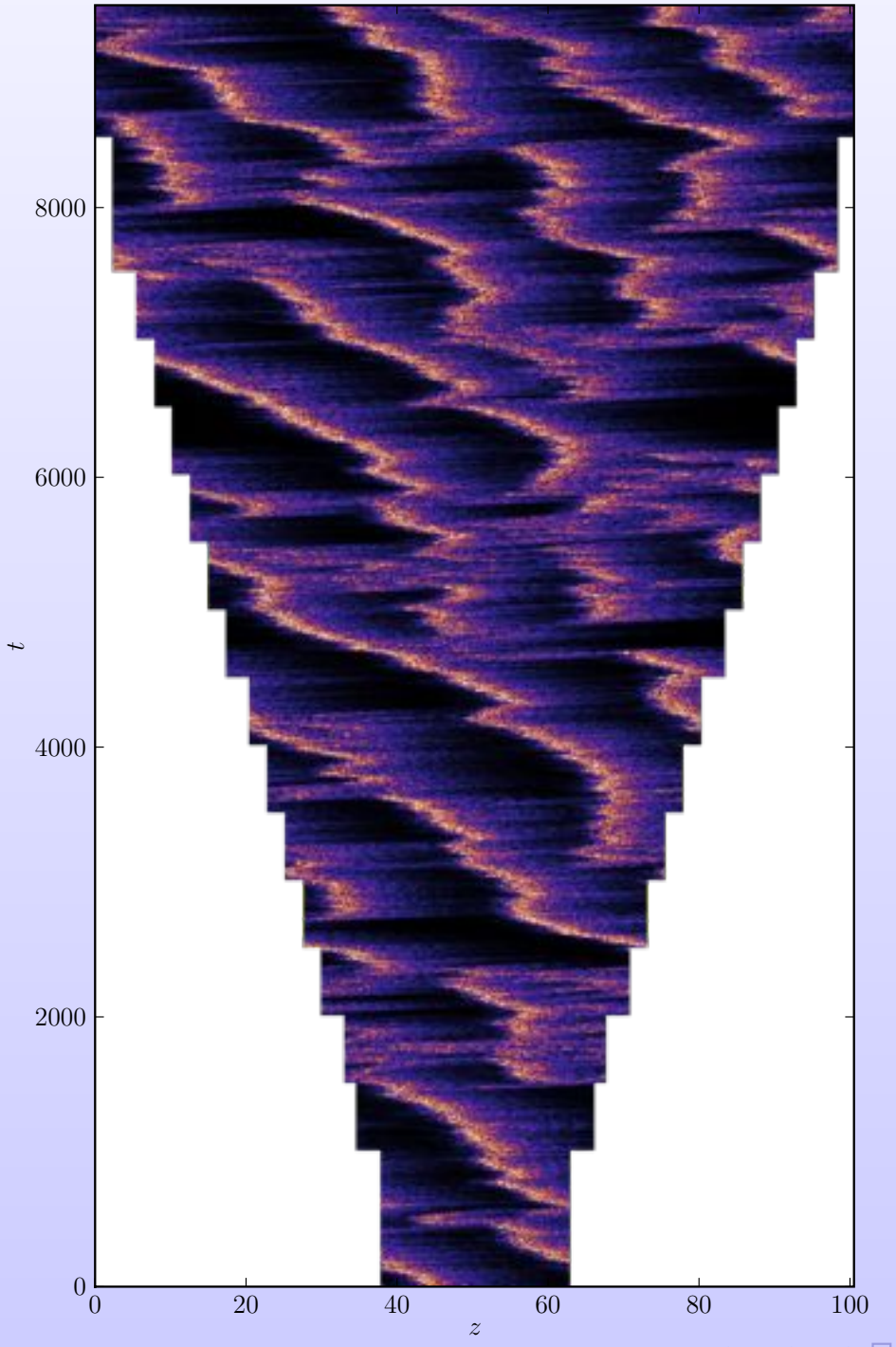



10000
8000
6000
4000
2000
0
t
0
20
40
60
80
100
120
140
z
2000
2050
2100
2150
2200
2350
2500
Re

Domain Expansion

- We saw that puffs tended to split apart in the $\mathrm{Re} = 2050$ case. Is this just a co-incidence or more of a natural behaviour of Navier-Stokes?
- To answer the question, we take a puff of $L = 25D$.
- Then we expand the domain every 500 time units by $L = 5D$ until we reach $L = 100D$.
- N_z is increased with L so that the domain remains correctly resolved.
- This was done for two separate simulations at both $\mathrm{Re} = 2000$ and $\mathrm{Re} = 2050$.
- Again we plot the history data with the quantity q.


8000
6000
4000
2000
0
t
0
20
40
60
80
100
z


8000
6000
4000
2000
0
t
0
20
40
60
80
100
z

Conclusions

- Computing power has increased enough over the past decade to simulate long pipes over large periods of time using highly accurate spectral and spectral element methods.
- Space-time plots reveal interesting behaviour in the transition from turbulent to laminar flow. Previously unseen pattern formation with trains of puffs.
- Domain expansion of the puffs reveals that the cases $\mathrm{Re} = 2000$ and $\mathrm{Re} = 2050$ are far different from one another.
- We believe that the more 'intermittant' behaviour found in the latter case is worth investigating in far greater detail.
- The puff splitting seen at $\mathrm{Re} = 2050$ is previously unseen at such low Reynolds numbers: usually this only happens for $\mathrm{Re} \geq 2400$.
- Future work will involve developing a quantitative method for identifying and classifying different states found in the Navier-Stokes attractor during transition.