

# Mathematical Methods for Data-Driven Modelling

Supplementary Module

SMSTC: Scottish Mathematical Sciences Training Centre

# Introduction

Debasish Das (Weeks 1–5)

Prabakaran Rajamanickam (Weeks 6–10)

University of Strathclyde

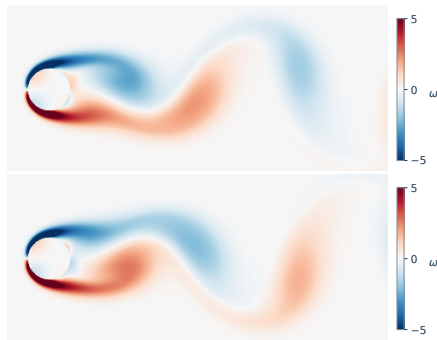
## Mathematical models from data

Experiments and simulations produce large collections of measurements.

We seek a small set of variables and equations that explain the behaviour and predict its evolution.

### A recurring example

Vortex shedding behind a cylinder: a complex flow with organised, repeating motion.



Two vorticity snapshots. Red and blue indicate opposite signs.

# What can we learn from measurements and simulations?

---

| Scientific task | Mathematical question                 | Examples of methods                  |
|-----------------|---------------------------------------|--------------------------------------|
| Representation  | Which variables capture structure?    | SVD, PCA, POD, autoencoders          |
| Prediction      | How does the system evolve?           | DMD, reduced models, neural dynamics |
| Inference       | Which equations fit the observations? | Regression, inverse problems         |
| Reliability     | Does the model generalise?            | Validation, sensitivity, uncertainty |

These ideas apply to fluid flows, deforming structures and many other dynamical systems.

# A research example: flexible filaments and sheets

## Reduced models of deforming objects in a fluid

A full simulation tracks many points on the object. Can a few learned variables capture its changing shape and motion?

### Encode the shape

$$\mathbf{z}_k = E(\mathbf{x}_k)$$

Represent the configuration by a small latent vector.

### Evolve the coordinates

$$\hat{\mathbf{z}}_{k+1} = F(\mathbf{z}_k)$$

Predict the next latent state.

### Reconstruct the shape

$$\hat{\mathbf{x}}_{k+1} = D(\hat{\mathbf{z}}_{k+1})$$

Recover a filament or sheet configuration.

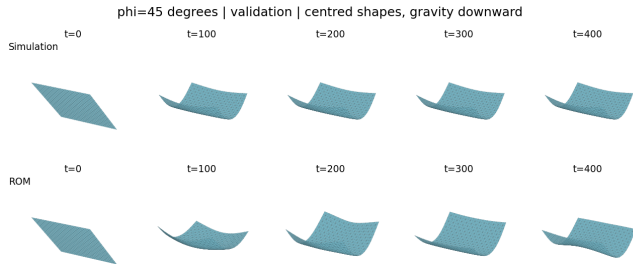
An **autoencoder** learns  $E$  and  $D$ . The dynamical model  $F$  predicts motion.

## In the simulation demonstrations

How well are shape and motion reproduced? Does the model work for new initial conditions?

# A research example: learning the dynamics of a flexible sheet

---



## From simulation data to a predictive reduced model

The full simulation evolves a high-dimensional deforming surface. A data-driven ROM instead evolves only a small number of latent variables and then reconstructs the sheet shape.

# Matrix factorisations and low-rank approximation

## Lectures 1–2

### Factorisations expose structure

$$PA = LU, \quad A = QR$$

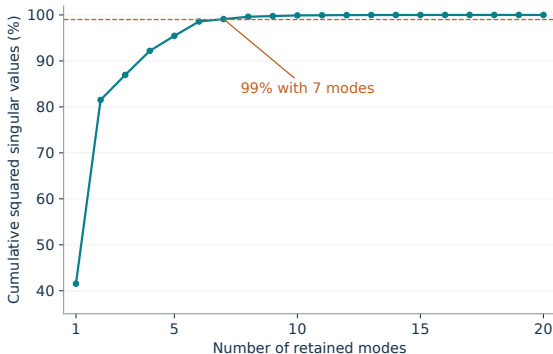
Linear systems, orthogonal bases and numerically stable calculations.

### The singular value decomposition

$$X = U\Sigma V^T, \quad X_r = U_r \Sigma_r V_r^T$$

Keep the dominant components and quantify what is lost.

**Key question:** how much can we discard?



Fraction of squared Frobenius norm captured in the centred vorticity data.

# Least squares, regularisation and model selection

## Lectures 3–4

### Fit a model to data

$$\min_{\mathbf{c}} \|A\mathbf{c} - \mathbf{b}\|_2^2$$

Least squares gives a systematic way to choose model parameters from observations.

### Control complexity

$$\min_{\mathbf{c}} \{ \|A\mathbf{c} - \mathbf{b}\|_2^2 + \lambda R(\mathbf{c}) \}$$

Regularisation balances agreement with the data against model complexity or sensitivity.

### Choose the model

Use training data to fit and validation data to choose model complexity and tuning parameters.  
Reserve test data for final assessment.



# PCA, POD and reduced representations

## Lecture 5: From a large state to a few coordinates

Many simulations have  $\mathbf{x}(t) \in \mathbb{R}^{10^6}$ . PCA/POD seeks

$$\mathbf{x}(t) \approx \bar{\mathbf{x}} + \Phi_r \mathbf{a}(t), \quad \mathbf{a}(t) \in \mathbb{R}^r, \quad r \ll 10^6.$$

If  $r = 10$ , a state described by a million numbers is replaced by only ten time-dependent coefficients.

$\bar{\mathbf{x}}$  is the mean/reference state; the columns of  $\Phi_r$  are the retained modes.

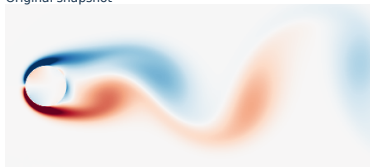
### Representation, not dynamics

These are reduced coordinates, not dynamics.

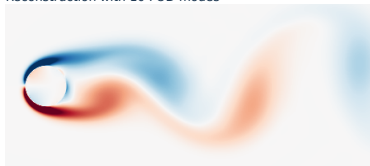
POD is **kinematic**: it represents the snapshots but gives no evolution law for  $\mathbf{a}(t)$ .

Mathematical Methods for Data-Driven Modelling

Original snapshot



Reconstruction with 10 POD modes



Cylinder-wake example. Both panels use the same colour scale.

# Dynamic mode decomposition

## Lecture 6: Learn a linear evolution law from data

From consecutive snapshots,

$$X_1 = [\mathbf{x}_1, \dots, \mathbf{x}_{m-1}], \quad X_2 = [\mathbf{x}_2, \dots, \mathbf{x}_m],$$

seek

$$X_2 \approx AX_1.$$

This is a least-squares problem:

$$A_\star = \arg \min_A \|X_2 - AX_1\|_F^2 = X_2 X_1^\dagger.$$

## Exploit low-rank structure

Rather than construct the enormous matrix  $A$ , use

$$X_1 = U_r \Sigma_r V_r^\top$$

to form the small matrix

$$\tilde{A} = U_r^\top X_2 V_r \Sigma_r^{-1}.$$

Then eigendecompose

$$\tilde{A}W = W\Lambda.$$

The eigenvalues give oscillation and growth/decay rates; the eigenvectors lead to the DMD modes.

**least squares + SVD + eigenvalues = DMD**

# Reduced-order models and nonlinear dynamics

## Lectures 7–8

PCA/POD  $\longrightarrow$  DMD  $\longrightarrow$  projection-based ROM  $\longrightarrow$  nonlinear dynamics from data

### Lecture 7: derive reduced dynamics

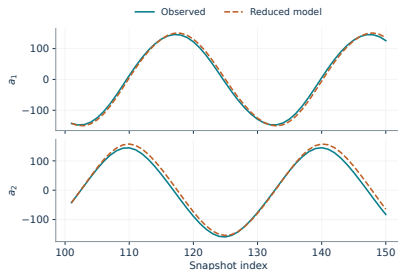
Start from  $\mathbf{x} \approx \bar{\mathbf{x}} + \Phi_r \mathbf{a}$  and project the known governing equations:

$$\dot{\mathbf{a}} = \Phi_r^T \mathbf{f}(\bar{\mathbf{x}} + \Phi_r \mathbf{a}).$$

The  $r$  POD coefficients now have their own reduced dynamical system.

### Lecture 8: learn reduced dynamics from data

Alternatively, infer  $\dot{\mathbf{a}} = \mathbf{g}(\mathbf{a})$  directly from trajectories. Sparse regression selects the few selected linear and nonlinear terms.



POD coefficients from the cylinder wake example. Dynamics are fitted to the earlier snapshots.

# Three ways to obtain reduced dynamics

## DMD, projection-based ROM and sparse regression

|                           | DMD                                               | POD–Galerkin ROM                                          | Sparse regression / SINDy                           |
|---------------------------|---------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------|
| Evolution law             | Fitted from time-shifted snapshot data            | Derived by projecting the governing equations             | Fitted from data using a library of candidate terms |
| Typical form              | $\mathbf{a}_{k+1} \approx \tilde{A} \mathbf{a}_k$ | $\dot{\mathbf{a}} = \mathbf{g}_{\text{proj}}(\mathbf{a})$ | $\dot{\mathbf{a}} = \Theta(\mathbf{a})\Xi$          |
| Dynamics                  | Linear                                            | Linear or nonlinear                                       | Linear or nonlinear; sparse in the chosen library   |
| Need governing equations? | No                                                | Yes                                                       | No                                                  |

**DMD:** learn linear dynamics from data

**Galerkin:** derive reduced dynamics from equations

**SINDy:** learn sparse nonlinear dynamics from data

# What does each method produce?

## Different routes to a reduced dynamical model

|                        | DMD                                                              | POD–Galerkin ROM                                                 | Sparse regression / SINDy                                                               |
|------------------------|------------------------------------------------------------------|------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <b>Role of POD/SVD</b> | Gives a low-dimensional space for fitting and spectral analysis  | Gives the basis onto which the governing equations are projected | <b>Not required.</b> May be used first when the original state is very high-dimensional |
| <b>Main output</b>     | Modes with growth/decay rates and frequencies                    | Coupled equations for the reduced coefficients                   | Selected candidate terms and their coefficients, defining the dynamical equations       |
| <b>Main limitation</b> | A single linear map may not capture strongly nonlinear evolution | Truncation may omit important interactions and affect stability  | Depends on the candidate library and sufficiently informative data                      |

**POD/SVD answers:** what coordinates should we use?

**DMD, Galerkin and SINDy answer:** how do those variables evolve?

# Gradient-based optimisation and adjoints

## Lectures 9–10

### Estimate parameters by comparing a model with observations

$$\min_{\mathbf{p}} J(\mathbf{p}), \quad J(\mathbf{p}) = \frac{1}{2} \sum_k \|\mathbf{h}(\mathbf{x}(t_k; \mathbf{p})) - \mathbf{y}_k\|_2^2, \quad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{p}).$$

Fit parameters  $\mathbf{p}$  to measurements  $\mathbf{y}_k$  through the observation map  $\mathbf{h}$ .

#### Gradient-based optimisation

$$\mathbf{p}_{n+1} = \mathbf{p}_n - \alpha_n \nabla J(\mathbf{p}_n)$$

Use derivatives to improve the fit. Study step sizes and convergence.

#### Adjoint methods

Differentiate an objective that depends on a dynamical trajectory.

Adjoint methods are efficient when there are many parameters and few objectives.

# Assessment: individual computational project

---

In consultation with the lecturers, choose a project from the list released after the first few weeks.

Apply methods from the module to a data-driven modelling problem. Explain the mathematical choices and assess the model's accuracy and limitations.

## Submission

- A written report of approximately 10 pages.
- Reproducible computational code.

## What the project assesses

- Understanding of the underlying mathematics.
- Appropriate numerical implementation.
- Interpretation of results and clear figures.
- Critical evaluation of accuracy and limitations.

Python is the main teaching language. MATLAB is also acceptable.