Equilibria, periodic orbits and computing them.

EPSRC Summer School on Modal decompositions in fluid mechanics.
DAMTP, Cambridge, 5-8 August 2019.

Ashley P. Willis,
School of Mathematics and Statistics, **University of Sheffield, U.K.**
a.p.willis@sheffield.ac.uk

In this short exposition, we describe equilibria and periodic orbits in terms of the flow map, $\mathbf{\Phi}$, and discuss the essentials of the Jacobian-free Newton–Krylov (JFNK) method that can be used to find them. This method requires little more than calls to an existing time stepping code, which $\mathbf{\Phi}$ can be considered to represent. Fortran90 / MATLAB code is available to try it out for yourself, where in the template/example it is applied to the Lorenz system. This code is problem-independent and can be applied to large systems, having initially been developed to find periodic orbits in simulations of pipe flow.

1 Using the flow-map

Let the point $\boldsymbol{x}_0$ be an n-vector representing the state of a system. For a dynamical system, as time t progresses, the point $\boldsymbol{x}_t$ traces out a trajectory, a one-dimensional curve, in an n-dimensional phase space $\mathcal{M}$.

We can describe the trajectory for $\boldsymbol{x}_t$ using the **flow-map** denoted $\mathbf{\Phi}^t$, which takes a point $\boldsymbol{x}_0$ and evolves it by a time period t: $\mathbf{\Phi}^t : \boldsymbol{x}_0 \to \boldsymbol{x}_t$ i.e.

$$\boldsymbol{x}_t = \mathbf{\Phi}^t(\boldsymbol{x}_0)\,. \tag{1.1}$$

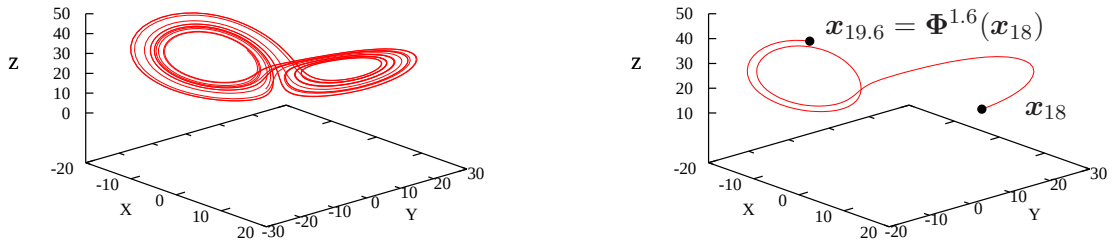
More generally, the flow-map simply advances a point along its trajectory:

$$\boldsymbol{x}_{t'+t} = \mathbf{\Phi}^t(\boldsymbol{x}_{t'})\,. \qquad \text{[flow-map]} \tag{1.2}$$

1.1 Example: Lorenz's model for convection

Lorenz (1963) derived the following system for three-time dependent amplitudes, $X(t)$, $Y(t)$ and $Z(t)$:

$$\begin{aligned}
\dot{X} &= -\sigma X + \sigma Y\,,\\
\dot{Y} &= -XZ + rX - Y\,,\\
\dot{Z} &= XY - bZ\,.
\end{aligned} \tag{1.3}$$



(a) (b)

Figure 1: (a) Lorenz attractor. (b) The flow-map $\boldsymbol{\Phi}$ is used to advance the state $\boldsymbol{x}$ by 1.6 time units from $t = 18$ to $t = 19.6$.

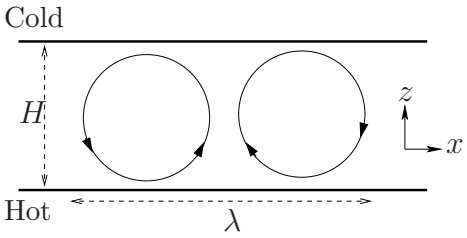


Figure 2: Rayleigh–Bénard convection. A pair of convection rolls, wavelength $\lambda = (2/a)\,H$; $b = 4/(1 + a^2)$.

At each instant in time, the current state $\boldsymbol{x} = (X, Y, Z)$ is a point in the phase space $\mathcal{M} = \mathbb{R}^3$. As time progresses, $\boldsymbol{x}_t = (X(t), Y(t), Z(t))$ traces out a trajectory, i.e. a curve, in $\mathbb{R}^3$. Lorenz focussed on parameter values $r = 28$, $b = 8/3$, $\sigma = 10$, which result in chaotic trajectories. The flow-map takes us along this trajectory, see figure 1.

We should not forget that each point $\boldsymbol{x}_t$ in phase-space corresponds to a whole convection flow pattern! Here it corresponds to a two-dimensional flow between two flat plates a distance H apart, with a temperature difference ΔT between the top and bottom plates (figure 2). The amplitudes $X(t)$, $Y(t)$, $Z(t)$ correspond to modulated variations in the temperature and velocity fields:

$$T(x,z,t) = \theta(x,z,t) - (z/H)\,\Delta T, \qquad u_x = -\frac{\partial \psi}{\partial z}, \quad u_z = \frac{\partial \psi}{\partial x}, \quad \text{(stream function } \psi)$$

$$\begin{aligned} \psi &= X(t) \times \sin(\pi a x/H)\sin(\pi z/H) \times c_1, \\ \theta &= Y(t) \times \cos(\pi a x/H)\sin(\pi z/H) \times c_2 \; - \; Z(t) \times \sin(2\pi z/H) \times c_3, \end{aligned}$$

where the c_i are scalar constants.

1.2 Invariant solutions

Equilibria and periodic orbits are topological features of the phase space $\mathcal{M}$. Irrespective of the coordinates and measure of distance used to visualise the phase space, an equilibrium remains a point, and a periodic orbit remains a closed loop. Properties, such as their eigenvalues, are also *invariant* to the measure of distance used.

An **equilibrium** $\boldsymbol{x}_0$ is a **fixed point** of the flow-map that satisfies

$$\boldsymbol{x}_0 = \boldsymbol{\Phi}^t(\boldsymbol{x}_0)\,, \qquad \text{[equilibrium = fixed point]} \tag{1.4}$$

for *any* time t. A point $\boldsymbol{x}_p$ on a **periodic orbit** satisfies

$$\boldsymbol{x}_p = \boldsymbol{\Phi}^T(\boldsymbol{x}_p)\,, \qquad \text{[periodic orbit]} \tag{1.5}$$

where T is the period of the orbit. In terms of solutions of the flow-map, we can consider an equilibrium to be a special case of a periodic orbit where T may be arbitrarily chosen.

If a system has a homogeneous dimension, x, then it can have **travelling wave** solutions. (See figure 3a.) In a frame moving at some phase speed c, the solution looks steady. Equivalently, we can keep shifting the solution so that it looks steady. Let $g(l)$ be an operator that shifts a state by a distance l in the x direction. Then, a travelling wave satisfies

$$\boldsymbol{x}_0 = g(-ct)\,\boldsymbol{\Phi}^t(\boldsymbol{x}_0)\,, \qquad \text{[relative equilibrium = travelling wave]} \tag{1.6}$$

for any time t. The state $\boldsymbol{x}_0$ is an equilibrium solution of the slightly modified map for any t, hence we also call a travelling wave a **relative equilibrium**. Similarly, a periodic solution that recurs up to a spatial shift,

$$\boldsymbol{x}_p = g(-\bar{c}T)\,\boldsymbol{\Phi}^T(\boldsymbol{x}_p)\,, \qquad \text{[relative periodic orbit]} \tag{1.7}$$

for some $\bar{c}$, we call a **relative periodic orbit**.

Suppose the dimension x has a mirror symmetry about $x = 0$. Let σ be the flip operator: $\sigma\,\boldsymbol{x}(x) = \boldsymbol{x}(-x)$. The second half of an orbit satisfying (1.5) might just be a reflection of the first half:

$$\boldsymbol{x}_p = \sigma\,\boldsymbol{\Phi}^{T/2}(\boldsymbol{x}_p)\,. \qquad \text{[pre-periodic orbit]} \tag{1.8}$$

(See figure 3b.) Such orbits are called **pre-periodic orbits.** The shortest/simplest periodic orbits of a system with discrete symmetries are typically pre-periodic. Figure 4 shows the shortest periodic orbit of the Lorenz system, where the first half of the orbit is related to the second half by the 180 degree rotation $(X, Y, Z) \to (-X, -Y, Z)$.

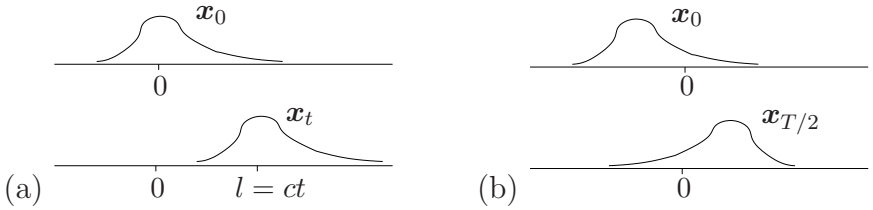


Figure 3: (a) A travelling wave: $\boldsymbol{x}_t = g(ct)\,\boldsymbol{x}_0$. By shifting back, $\boldsymbol{x}_0 = g(-l)\,\boldsymbol{x}_t = g(-l)\,\boldsymbol{\Phi}^t(\boldsymbol{x})$. (b) A pre-periodic orbit: $\boldsymbol{x}_0 = \boldsymbol{x}_T$, but also $\boldsymbol{x}_0 = \sigma\,\boldsymbol{x}_{T/2}$, where σ flips the state about 0.

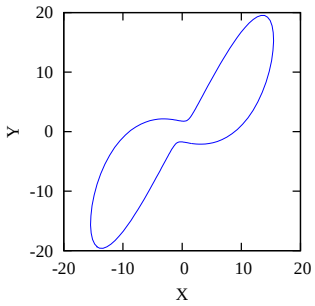


Figure 4: The shortest periodic orbit of the Lorenz system (1.3) has the rotational symmetry $(X, Y, Z) \to (-X, -Y, Z)$.

1.3 Poincaré sections

Let $\boldsymbol{x}'$ be a point and $\boldsymbol{t}'$ be a normal vector that together define a hypersurface $\mathcal{P}$. Crossings of $\mathcal{P}$ can be defined by times t when $\langle \boldsymbol{x}_t - \boldsymbol{x}' | \boldsymbol{t}' \rangle = 0$. (See figure 5a.) We might restrict to when crossings occur in a particular direction, say when the inner product goes from negative to positive.

We can now let $\boldsymbol{\Phi}$ be the map that takes one point on $\mathcal{P}$ to the next crossing point on $\mathcal{P}$. If a periodic orbit has a point $\boldsymbol{x}_p$ on $\mathcal{P}$, then it satisfies

$$\boldsymbol{x}_p = \boldsymbol{\Phi}(\boldsymbol{x}_p)\,. \tag{1.9}$$

The advantage is that we no longer need to worry about the period T for periodic orbits. The disadvantage is that we know nothing about what happens to the orbit off $\mathcal{P}$, and in general, not all periodic orbits cross a single $\mathcal{P}$.

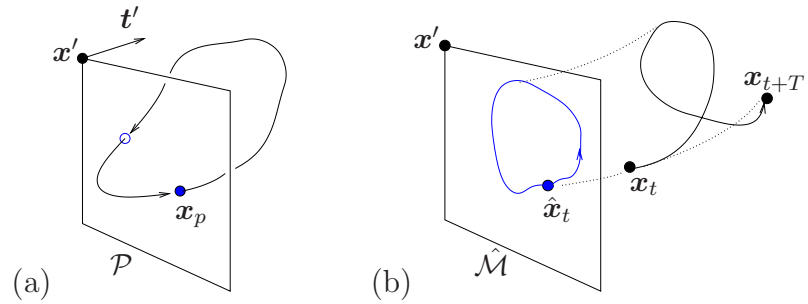


Figure 5: (a) A Poincaré section $\mathcal{P}$, defined by a point $\boldsymbol{x}'$ and a normal vector $\boldsymbol{t}'$, is pierced by a periodic orbit at the periodic point $\boldsymbol{x}_p$. (b) The projection of a relative periodic orbit onto a slice $\hat{\mathcal{M}}$, is a periodic orbit, $\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t+T}$. The whole orbit is projected onto $\hat{\mathcal{M}}$. Each state $\boldsymbol{x}_t$ is projected by applying shifts along the dotted lines onto $\hat{\mathcal{M}}$.

1.4 Slicing

For a homogeneous spatial dimension x, the freedom of a pattern to appear at any location is awkward when we want to compare states. Slicing is an automatic shifting procedure that removes this degree of freedom.

Here we will discuss the simplest form of slicing – 'Fourier' slicing. (See Willis et al., 2016, and references therein.) When a system has a homogeneous dimension, it is commonplace to work

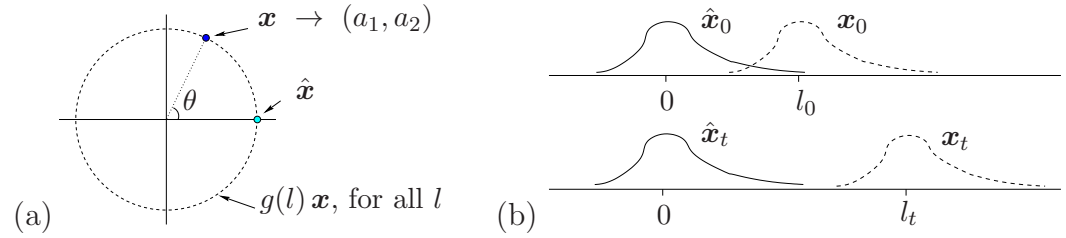


Figure 6: (a) Fourier slicing: A state $\boldsymbol{x}$ is mapped onto the (a_1, a_2)-plane, where all shifted versions of $\boldsymbol{x}$ map onto a circle. All versions are reduced to the single copy $\hat{\boldsymbol{x}} = g(-(\theta/2\pi)L)\,\boldsymbol{x}$. (b) A travelling wave is reduced to an equilibrium. l_t is automatically determined by the slicing algorithm, from which a phase speed c can be inferred.

with a periodic domain of length $L = 2\pi/\alpha$. In this case, construct $\boldsymbol{x}' = \boldsymbol{x}_c \cos\alpha x + \boldsymbol{x}_s \sin\alpha x$, where $\boldsymbol{x}_c$ and $\boldsymbol{x}_s$ are arbitrary states independent of x; they must not both be zero.

Any state $\boldsymbol{x}$ may be projected onto a plane (a_1, a_2) via $a_1 = \langle \boldsymbol{x} | \boldsymbol{x}' \rangle$ and $a_2 = \langle \boldsymbol{x} \,|\, g(L/4)\,\boldsymbol{x}' \rangle$. (See figure 6a.) In this projection, the set of shifted states $\{g(l)\boldsymbol{x} \text{ for all } l\}$ lie on a circle centred on the origin. By shifting the states, we can rotate all points on the circle to the *unique* point on the circle where it crosses the positive a_1 axis. All possible shifted versions of $\boldsymbol{x}$ are then mapped to the unique version $\hat{\boldsymbol{x}} = g(-l)\,\boldsymbol{x}$, where $l = (\theta/2\pi)L$ and θ is the polar angle to (a_1, a_2).

This operation is a **symmetry-reduction**, and we say that the symmetry-reduced state $\hat{\boldsymbol{x}}$, indicated by the hat, lies on a **slice**. Arbitrary shifts have been eliminated, so the slice has dimension one less than that of the original system. The slice $\hat{\mathcal{M}}$ is a hypersurface within the original space of states $\mathcal{M}$.

The slice is different from a Poincaré section because the symmetry reduction can be applied to $\boldsymbol{x}_t$ for all times t. We can compute sliced dynamics with trajectories $\hat{\boldsymbol{x}}_t$ that lie within the slice. (See figure 5b.) Meanwhile, trajectories only pierce a Poincaré section.

Relative equilibria (travelling waves) are reduced to equilibria automatically:

$$\boldsymbol{x}_0 = g(-ct)\,\boldsymbol{\Phi}^t(\boldsymbol{x}_0) \quad \to \quad \hat{\boldsymbol{x}}_0 = \hat{\boldsymbol{\Phi}}^t(\hat{\boldsymbol{x}}_0)\,. \tag{1.10}$$

All possible shifts of a state are reduced by shifting to one particular version on the slice $\hat{\mathcal{M}}$, i.e. the travelling wave is 'pinned' by the shifting. (See figure 6b.) Here, $\hat{\boldsymbol{\Phi}}$ is the flow-map of the symmetry reduced dynamics. Note that *all* travelling waves of the system are reduced to equilibria, while they typically will have a range of different phase speeds c.

Similarly, **a relative periodic orbit becomes a closed periodic orbit**, because the start and end point are shifted to a single point on $\hat{\mathcal{M}}$. The relative periodic orbit now satisfies the simpler form:

$$\boldsymbol{x}_p = g(-\bar{c}T)\,\boldsymbol{\Phi}^T(\boldsymbol{x}_p) \quad \to \quad \hat{\boldsymbol{x}}_p = \hat{\boldsymbol{\Phi}}^T(\hat{\boldsymbol{x}}_p)\,, \tag{1.11}$$

for any symmetry-reduced point $\hat{\boldsymbol{x}}_p$ on the orbit.

2 Periodic orbits

All periodic orbits (POs) in a chaotic attractor must be unstable, otherwise the behaviour would eventually be attracted to the orbit and become periodic, not chaotic. So why is it useful to find POs when they're all unstable!?

2.1 Why periodic orbits?

Firstly, the dynamics is always very close to a PO! A chaotic attractor is **dense** in POs: For any chaotic point $\boldsymbol{x}_0$ and $0 < \epsilon \ll 1$, there exists a periodic point $\boldsymbol{x}_p$ within ϵ of $\boldsymbol{x}_0$. See figure 7. Also...

- Unlike equilibria, POs exhibit dynamics! — they capture the time-dependent dynamic processes of the system and organise the chaotic set.
- The chaotic dynamics tends to follow the least unstable POs.
- A PO is a closed loop in state space. It will appear as a closed loop irrespective of the coordinates used to visualise the state space.
- The shortest, most fundamental, POs provide an alphabet for symbolic dynamics.
- Statistical properties of a chaotic attractor can be calculated in terms of sums over the POs, using their relative stability.

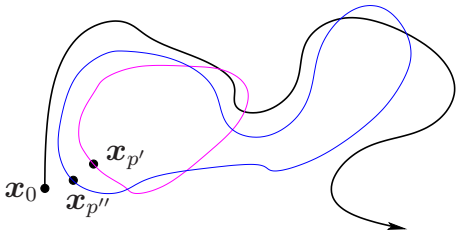


Figure 7: 'Shadowing' of a chaotic trajectory by periodic orbits. By considering periodic orbits of increasing length, it is possible to find a periodic point $\boldsymbol{x}_p$ arbitrarily close to a point $\boldsymbol{x}_0$ on the chaotic attractor.

2.2 Examples of periodic orbits

- The logistic map $x_{t+1} = r\,x_t\,(1 - x_t)$ is chaotic for the case $r = 4$ but has the (unstable) period-2 orbit

$$(5 - \sqrt{5})/8 \;\to\; (5 + \sqrt{5})/8 \;\to\; (5 - \sqrt{5})/8 \;\to\; \ldots$$

- In a remarkable methodical and computational feat, Viswanath (2003) calculated 111011 periodic orbits of the Lorenz attractor, that is all periodic orbits with itineraries of up to length 20 (number of windings around the left/right 'wings'), with an accuracy of 14 decimal digits. A few of them are shown in figure 8. The periodic orbit $A^{14}B$ was found to be the least unstable among all orbits calculated. (It is the periodic orbit with the smallest Lyapunov/Floquet exponent.)

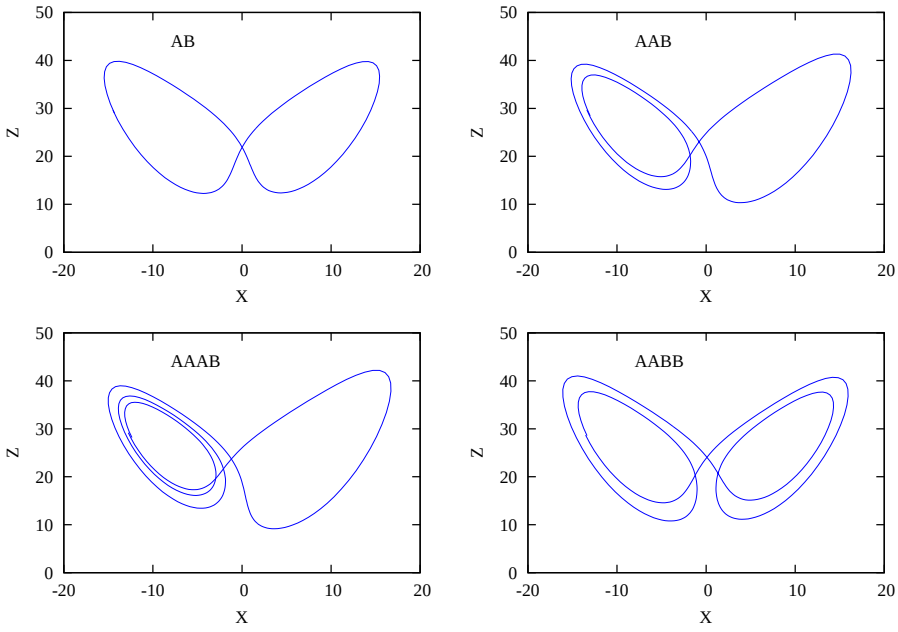


Figure 8: A few of the shortest periodic orbits of the Lorenz system. (Reproduction of figure 3 of Viswanath, 2003)

- For an $n = 154755$-dimensional model of turbulent flow in a pipe, Willis et al. (2016) calculated periodic orbits and travelling wave solutions. Travelling waves are stationary in a moving frame, so they correspond to fixed points in the sliced phase space. Two 'clouds' of solutions were observed, shown in figure 9, and the relationship between them found to be the reflection symmetry. The phase-space visualisation reveals that trajectories don't like to switch orientation with respect to the symmetry, due to the presence of a strongly repelling (unstable) fixed point, marked 'A', that lives between the two clouds. For beautiful examples from Couette flow, see Cvitanović and Gibson (2010).

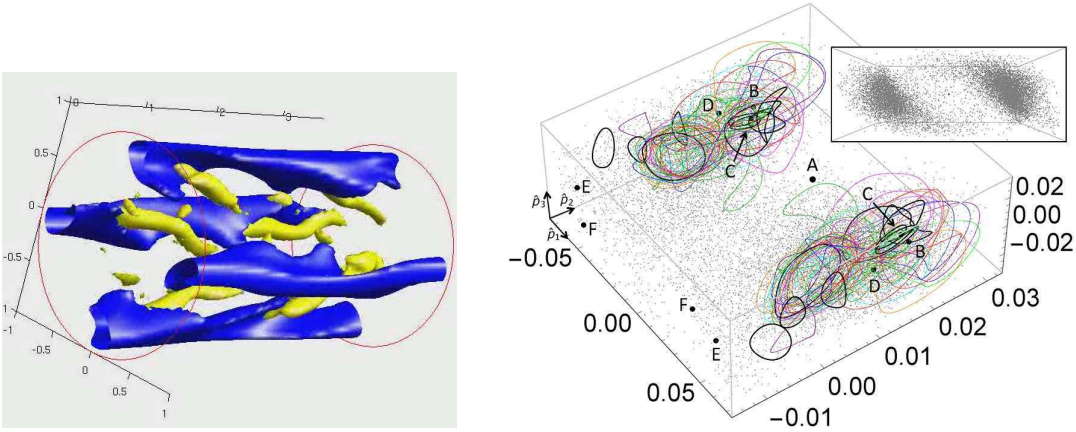


Figure 9: (*left*) Visualisation of pipe flow with slow streaks (blue) and vortices (yellow). (*right*) Periodic orbits of pipe flow. Inset is view from bottom left side of box. (Reproduction of figure 4 of Willis et al., 2016)

2.3 Searching for recurrences

At present, the standard approach to finding near-recurrences is pretty crude. It involves recurrence plots, with distance measures tailored for the case at hand, and, if not automated, visual inspection of the plot for close recurrences. Nevertheless, this is the usual means to find points close to periodic orbits that can be refined to exact recurrences using the Newton method (described in the next section).

For a recurrence plot we compute something like

$$\frac{||\boldsymbol{x}_t - \boldsymbol{x}_{t-\Delta t}||}{||\boldsymbol{x}_{t-\Delta t}||} \,. \tag{2.1}$$

Figure 10 is an example for pipe flow. We then look for local minima in the plot that provide candidate recurrent points, $\boldsymbol{x}_p \approx \boldsymbol{x}_{t-\Delta t}$ and $T \approx \Delta t$. The normalisation factor might be chosen to depend on both $||\boldsymbol{x}_t||$ and $||\boldsymbol{x}_{t-\Delta t}||$, or might not be necessary at all. A complication is that we might need to minimise over discrete symmetries, such as the flip operator σ, or over shifts, e.g.

$$\min(||\boldsymbol{x}_t - \boldsymbol{x}_{t-\Delta t}||,\ ||\boldsymbol{x}_t - \sigma\,\boldsymbol{x}_{t-\Delta t}||) \quad \text{or} \quad \min_l(||\boldsymbol{x}_t - g(-l)\boldsymbol{x}_{t-\Delta t}||) \tag{2.2}$$

Minimisation over shifts can be avoided if slicing is applied, see section 1.4.

The norm itself might need tinkering with. For example, in the sheared flow of fluid, perturbations in the streamwise dimension are typically an order of magnitude larger than the crossflow components. A 'compensatory' norm helps in this case, where the components are scaled to be more similar in magnitude.

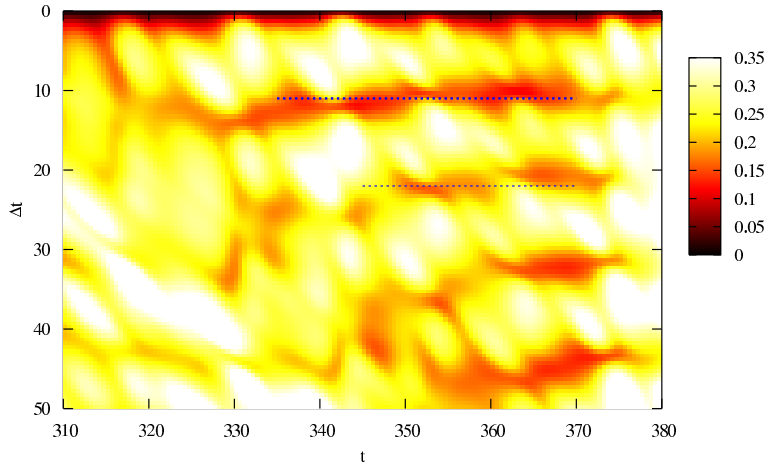


Figure 10: Search for recurrences: colour plot of $||\hat{\boldsymbol{x}}_t - \hat{\boldsymbol{x}}_{t-\Delta t}||_c \,/\, ||\hat{\boldsymbol{x}}_{t-\Delta t}||_c$ for pipe flow in a 'minimal' box (Willis et al., 2013). Axial shifts have been eliminated by slicing, indicated by the hat; section 1.4. Minima around the horizontal lines suggest shadowing of a periodic orbit with period $T \approx 11$. Subscript c indicates that compensation has been applied to pick up a signal from cross-flow components, which are smaller in magnitude but as important as the streamwise perturbations.

3 The Newton–Krylov method

The Jacobian-free Newton–Krylov (JFNK) method is a variant of the Newton–Raphson method. In its raw form, the Newton–Raphson method for an n-dimensional system involves an $n \times n$ Jacobian matrix, which can be tricky to evaluate. It is possible to avoid this evaluation using a Krylov-subspace method.

3.1 The Newton–Raphson method

To find roots x such that $f(x) = 0$ in one dimension, given an initial guess x_0, the Newton-Raphson method generates improvements using the iteration

$$x_{i+1} = x_i - f(x_i)/f'(x_i)\,. \tag{3.1}$$

Re-arranging, we may re-express the iteration as

$$x_{i+1} = x_i + \delta x_i \quad \text{where} \quad f'(x_i)\,\delta x_i = -f(x_i)\,. \tag{3.2}$$

Our task is to find fixed points of the map such that $\boldsymbol{x_p} = \boldsymbol{\Phi}(\boldsymbol{x}_p)$, i.e.

$$\boldsymbol{F}(\boldsymbol{x}_p) = \boldsymbol{0} \quad \text{where} \quad \boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{x}. \tag{3.3}$$

(The fixed points could correspond to equilibria, periodic orbits, or their relative equivalents. Augmentations, if necessary, to find a period T or spatial shift l are delayed to section 3.4.) The extension of Newton's method (3.2) to an n-dimensional system is then

$$(a)\;\; \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{\delta x}_i \quad \text{where} \quad (b)\;\; \left.\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial x}}\right|_{\boldsymbol{x}_i} \boldsymbol{\delta x}_i = -\boldsymbol{F}(\boldsymbol{x}_i)\,. \tag{3.4}$$

In order to apply the update (3.4a), the linear system (3.4b) needs to be solved for the unknown $\boldsymbol{\delta x}_i$.

In (3.4b), the matrix part is given by

$$\left.\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial x}}\right|_{\boldsymbol{x}_i} \;=\; \left.\frac{\boldsymbol{\partial \Phi}}{\boldsymbol{\partial x}}\right|_{\boldsymbol{x}_i} - I \;=\; J - I \tag{3.5}$$

where J is the **Jacobian** matrix for $\boldsymbol{\Phi}(\boldsymbol{x})$ and I is the identity matrix. For the case $n = 3$,

$$\boldsymbol{x} = (x_1,\, x_2,\, x_3)\,, \quad \boldsymbol{\Phi}(\boldsymbol{x}) = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \end{bmatrix}, \quad J = \begin{bmatrix} \frac{\partial \Phi_1}{\partial x_1} & \frac{\partial \Phi_1}{\partial x_2} & \frac{\partial \Phi_1}{\partial x_3} \\ \frac{\partial \Phi_2}{\partial x_1} & \frac{\partial \Phi_2}{\partial x_2} & \frac{\partial \Phi_2}{\partial x_3} \\ \frac{\partial \Phi_3}{\partial x_1} & \frac{\partial \Phi_3}{\partial x_2} & \frac{\partial \Phi_3}{\partial x_3} \end{bmatrix}. \tag{3.6}$$

3.2 Jacobian-Free method

The $n \times n$ Jacobian matrix J is usually difficult to evaluate. We might not even have sufficient computer memory to store it for a high dimensional system. The problem (3.4b), however, is

in the form

$$A\,\boldsymbol{\delta x} = \boldsymbol{b}\,, \tag{3.7}$$

where A is an $n \times n$ matrix and $\boldsymbol{\delta x}$ and $\boldsymbol{b}$ are n-vectors. This can be solved for $\boldsymbol{\delta x}$ using the **Krylov-subspace method GMRES(m)**. The GMRES algorithm does not need to know the matrix A itself, only the result of multiplying a given vector by A. The method seeks a solution for $\boldsymbol{\delta x}$ in $\mathrm{span}\{\boldsymbol{K}_1, \boldsymbol{K}_2, \ldots, \boldsymbol{K}_m\}$, i.e. $\boldsymbol{\delta x} = c_1\,\boldsymbol{K}_1 + c_2\,\boldsymbol{K}_2 + ... + c_m\,\boldsymbol{K}_m$. It is common to start with $\boldsymbol{K}_1 = \boldsymbol{b}/||\boldsymbol{b}||$. The next vector is generated by evaluating $\tilde{\boldsymbol{K}}_{i+1} = A\,\boldsymbol{K}_i$, then $\boldsymbol{K}_{i+1}$ is obtained by orthonormalising $\tilde{\boldsymbol{K}}_{i+1}$ against the previous $\boldsymbol{K}_j$ $(j \leq i)$ using the Gram-Schmidt method. Next, $\mathrm{error} = ||A\ \boldsymbol{\delta x} - \boldsymbol{b}\,||$ is minimised over the coefficients c_j $(j \leq i+1)$ and the process repeated if error is too large.

Iterations of the GMRES algorithm for the problem (3.4b) involve calculating matrix-vector products with given $\boldsymbol{\delta x}$ that may be approximated:

$$\left.\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial x}}\right|_{\boldsymbol{x}_i} \boldsymbol{\delta x} \quad \approx \quad \frac{1}{\epsilon}\left(\boldsymbol{F}(\boldsymbol{x}_i + \epsilon\,\boldsymbol{\delta x}) - \boldsymbol{F}(\boldsymbol{x}_i)\right). \tag{3.8}$$

ϵ is a small scalar value; a typical value is ϵ such that $(\epsilon||\boldsymbol{\delta x}||)\,/\,||\boldsymbol{x}_i|| = 10^{-6}$. The important point is that we do not need to know the Jacobian — **only a routine for evaluating $\boldsymbol{F}(\boldsymbol{x})$ is required**.

Note that provided that each step of the Newton method, $\boldsymbol{\delta x}$, is approximately in the correct direction, the method is expected to converge. Therefore the tolerance specified in the accuracy of the solution for $\boldsymbol{\delta x}$ in each Newton step (calculated via the GMRES method) typically need not be so stringent as the tolerance placed on the Newton method itself for the solution $\boldsymbol{x}$. For example, we might seek a relative error for the Newton solution $||\boldsymbol{F}(\boldsymbol{x})||/||\boldsymbol{x}|| = O(10^{-8})$, but a relative error for the GMRES solution $||A\,\boldsymbol{\delta x} - \boldsymbol{b}||/||\boldsymbol{\delta x}|| = O(10^{-3})$ is likely to be sufficient for calculation of the steps $\boldsymbol{\delta x}$.

3.3 Hookstep approach

To improve the domain of convergence of the Newton method, it is commonplace to limit the size of the step taken. One approach is simply to take a 'damped' step in the direction of the solution to 3.4(b), i.e. step by $\alpha\ \boldsymbol{\delta x}_i$, where $\alpha \in (0, 1]$. In the "'hookstep approach"', we minimise subject to the condition that the magnitude of the Newton step is limited, $||\boldsymbol{\delta x}_i|| < \delta$, where δ is the size of the "'trust region"':

$$\min_{\boldsymbol{\delta x}_i:\ ||\boldsymbol{\delta x}_i|| < \delta} \left|\left| \left.\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial x}}\right|_{\boldsymbol{x}_i} \boldsymbol{\delta x}_i + \boldsymbol{F}(\boldsymbol{x}_i) \right|\right| . \tag{3.9}$$

Given the minimisation, the hookstep $\boldsymbol{\delta x}_i$ is expected to produce a better result than a simple damped step of the same size. It is also expected to perform much better in 'valleys', where it produces a bent/hooked step to a point along the valley, rather than jumping from one side of the valley to the other; see figure 11.

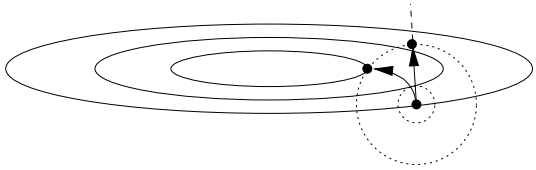


Figure 11: Hookstep versus 'damped'/line-search step of the same size in minimising $||\boldsymbol{F}(\boldsymbol{x})||^2$. The radius of the circle corresponds to the size of the step / trust region δ.

The hookstep can be calculated with little extra work to the GMRES method, provided that the size of Krylov-subspace, m, is chosen sufficiently large to solve to the desired accuracy within m GMRES iterations; for details see Viswanath (2007) [particularly **v1** on arxiv.org].

For a given $\boldsymbol{\delta x}_i$, the reduction in error predicted by the linearisation (3.9) can be compared with the actual reduction in $||\boldsymbol{F}(\boldsymbol{x}_i + \boldsymbol{\delta x}_i)||$. According to the accuracy of the prediction, the size of the trust region δ can be adjusted automatically; see Dennis and Schnabel (1996).

3.4 Adding constraints

3.4.1 Time constraint

When looking for a periodic orbit, the period T is an extra unknown. One way to eliminate needing to find T is to work within a Poincaré section, as described in section 1.3. We can then attempt to solve the function $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{x}$ as it stands to find a point $\boldsymbol{x}_p$ on the Poincaré section that corresponds to a periodic orbit.

We might not want to restrict ourselves to Poincaré sections. We must then solve

$$\boldsymbol{F}(\boldsymbol{x}, T) = \boldsymbol{\Phi}^T(\boldsymbol{x}) - \boldsymbol{x} = \boldsymbol{0}\,, \tag{3.10}$$

for $(\boldsymbol{x}, T)$. We augment the whole system. Let

$$\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i,\, T_i) \quad \text{and} \quad \tilde{\boldsymbol{b}} = (-\boldsymbol{F}(\tilde{\boldsymbol{x}}_i),\, 0)\,. \tag{3.11}$$

We now want to solve a system of the form

$$A\,\tilde{\boldsymbol{\delta x}}_i = \tilde{\boldsymbol{b}}\,, \tag{3.12}$$

for $\tilde{\boldsymbol{\delta x}}_i = (\boldsymbol{\delta x}_i,\, \delta T_i)$, but need an extra constraint because we have an extra unknown. We choose that the update $\boldsymbol{\delta x}_i$ has no component that points along the trajectory, i.e. $\langle \dot{\boldsymbol{x}}_i | \boldsymbol{\delta x}_i \rangle = 0$. Following the ethos of matrix-free methods, that we do not need to know the matrix A itself, we only need to state the result of multiplication by A:

$$A\,\tilde{\boldsymbol{\delta x}} = \left(\left.\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial \tilde{x}}}\right|_{\tilde{\boldsymbol{x}}_i} \tilde{\boldsymbol{\delta x}},\, \langle \dot{\boldsymbol{x}}_i | \boldsymbol{\delta x} \rangle \right)\,. \tag{3.13}$$

We use the approximation (3.8) to evaluate the first part of the result. The augmented system (3.12) can be solved for $\tilde{\boldsymbol{\delta x}}_i$ using the GMRES algorithm by applying multiplications (3.13). The update for both the state and the period is then $\tilde{\boldsymbol{x}}_{i+1} = \tilde{\boldsymbol{x}}_i + \tilde{\boldsymbol{\delta x}}_i$.

3.4.2 Shift constraints

For relative equilibria (travelling waves) and relative periodic periodic orbits we need to solve

$$\boldsymbol{F}(\boldsymbol{x}, T, l) = g(-l)\,\boldsymbol{\Phi}^T(\boldsymbol{x}) - \boldsymbol{x} = \boldsymbol{0}\,, \tag{3.14}$$

where l is an unknown spatial shift in the homogeneous x-dimension.

One way to avoid the extra unknown is to work with the sliced dynamics (section 1.4) so that arbitrary shifts are automatically eliminated. Alternatively, we can augment the system again. Let

$$\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i,\, T_i, l_i) \quad \text{and} \quad \tilde{\boldsymbol{b}} = (-\boldsymbol{F}(\tilde{\boldsymbol{x}}_i),\, 0,\, 0)\,. \tag{3.15}$$

We now want to solve a system of the form

$$A\,\tilde{\boldsymbol{\delta x}}_i = \tilde{\boldsymbol{b}}\,, \tag{3.16}$$

for $\tilde{\boldsymbol{\delta x}}_i$, but need another constraint to match the extra unknown. This time we choose that the update $\boldsymbol{\delta x}_i$ has no component that just corresponds to a spatial shift, i.e. $\langle \partial_x \boldsymbol{x}_i | \boldsymbol{\delta x}_i \rangle = 0$. We assert that multiplication by A is:

$$A\,\tilde{\boldsymbol{\delta x}} = \left(\left.\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial \tilde{x}}}\right|_{\tilde{\boldsymbol{x}}_i} \tilde{\boldsymbol{\delta x}},\, \langle \dot{\boldsymbol{x}}_i | \boldsymbol{\delta x} \rangle,\, \langle \partial_x \boldsymbol{x}_i | \boldsymbol{\delta x} \rangle \right)\,. \tag{3.17}$$

3.5 Preconditioning

The good news is that you can ignore preconditioning and skip this section if you are combining Newton–Krylov with timestepping to evaluate the flow-map $\boldsymbol{\Phi}^T(\boldsymbol{x})$.

3.5.1 Exponentiation and timestepping

The GMRES algorithm is closely related to another Krylov-subspace method, the Arnoldi method, which is used to calculated eigenvalues of a matrix A. It tends to find the eigenvalues most separated in the complex plane first, but those might be of little interest. For example, the Laplacian ∇^2 has a spectrum of very negative eigenvalues corresponding to high frequency oscillations that rapidly decay. Basically, we do not wish to build a Kyrlov-subspace involving such modes.

It may be better to work with $\tilde{A} = \mathrm{e}^A = 1 + A + \frac{1}{2!}A^2 + ...$, corresponding to the eigenproblem $\mathrm{e}^\sigma \boldsymbol{x} = \mathrm{e}^A \boldsymbol{x}$. This problem shares the same eigenvectors as the problem $\sigma\,\boldsymbol{x} = A\,\boldsymbol{x}$, but has more suitable eigenvalues, $\tilde{\sigma} = \mathrm{e}^\sigma$. The negative eigenvalues σ then correspond to eigenvalues $\tilde{\sigma}$ bunched close to the origin. The Arnoldi method then favours the $\tilde{\sigma}$ most distant from the origin, corresponding to the σ with largest real parts.

Note that for the system $\partial_t \boldsymbol{x} = A\,\boldsymbol{x}$, time integration corresponds to exponentiation: Taking eigenvector $\boldsymbol{x}$ with growth rate σ as an initial condition, the result of time integration from 0 to T is $\mathrm{e}^{\sigma T}\boldsymbol{x}$. We therefore have that $\mathrm{e}^{\sigma T}\boldsymbol{x} = \boldsymbol{x} + \int_0^T A\,\boldsymbol{x}\,dt = \mathrm{e}^{AT}\boldsymbol{x}$, which can be written $\tilde{\sigma}\,\boldsymbol{x} = B\,\boldsymbol{x}$ where $\tilde{\sigma} = \mathrm{e}^{\sigma T}$ is the eigenvalue of the time integration operator $B = \mathrm{e}^{AT}$.

3.5.2 Explicit preconditioning

GMRES is likely to find it easier to solve $M^{-1}A\,\boldsymbol{x} = M^{-1}\boldsymbol{b}$ than the original system, if M^{-1} is an approximate inverse for A. For example, if A is dominated by its diagonal elements, we might take M to be the banded matrix consisting of the diagonal and the first sub- and super-diagonals of A. Each GMRES iteration applied to the modified system now requires a muliplication by A then by M^{-1}. This is fine, as, for a banded matrix, it is quick and easy to solve $M\boldsymbol{x}' = \boldsymbol{x}$ for $\boldsymbol{x}'$. Like A, we don't need to know the matrix M^{-1} itself, only the result of multiplication by the matrix.

4 Try it yourself! Application of the Newton–Krylov method to the Lorenz system

Given that the Newton–Krylov method is designed to cope with high-dimensional systems (the same code has been used to find travelling waves in pipe flow), this is somewhat overkill, but it helps illustrates how we can use the solver as a **black box**...

Please cite `Openpipeflow.org` (Willis, 2017) if you use this code in your research. Thanks!

- **Download the Template/Example** (Fortran90 / MATLAB / Octave)
  `http://www.openpipeflow.org/index.php?title=Newton-Krylov_method`
- For MATLAB, the unpacked `tgz/zip` file has separate `.m` files for each function.
  Take a look at
  - `Lorenz_f.m`: Lorenz evolution rule $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$.
  - `steporbit.m`: Evaluate $\boldsymbol{\Phi}^T(\boldsymbol{x})$, i.e. step $\boldsymbol{f}$ by `ndts_` timesteps, where the input `x(1)`$= T$, and `x(2:4)`$= (X, Y, Z)$. The timestep size is `dt`$= T/$`ndts_`.
  - `saveorbit.m`: Output at end of each Newton iteration. `relative_err` $= ||\boldsymbol{F}(\boldsymbol{x})|| \,/\, ||\boldsymbol{x}||$.
  - `MAIN.m`: Set up initial guess $\boldsymbol{x}_0$ and call the **black box** `NewtonHook.m`.

- Other functions are called by `NewtonHook.m`, and are unlikely to need changing for a problem of this type, where shifts and other spatial symmetries are ignored:
  - `getrhs.m`: Evaluate right-hand side $\tilde{\boldsymbol{b}}$ (3.11) i.e. $\boldsymbol{F}(\tilde{\boldsymbol{x}}) = \boldsymbol{\Phi}^T(\boldsymbol{x}) - \boldsymbol{x}$.
  - `multJ.m` : Evaluate multiplication (3.13), i.e. multiplication by the Jacobian.
  - `multJp.m`: Preconditioner for multiplication (here an empty function).
  - `dotprd.m`: Evaluate inner product $\langle \boldsymbol{a} | \boldsymbol{b} \rangle$.
  - `GMRESm.m`: Method of section 3.2.
  - `GMREShook.m`: Calculate hookstep, section 3.3.

- The following data are points on the periodic orbits of figure 8, taken from Viswanath (2003). $Z = 27$ in call cases.

	X	Y	T
AB	13.763610682134	19.578751942452	1.5586522107162
AAB	12.595115397689	16.970525307084	2.3059072639399
AAAB	11.998523280062	15.684254096883	3.0235837034339
AABB	12.915137970311	17.673100172646	3.0842767758221

($Z = 27$)

- In MATLAB, call `MAIN`. It will plot the result of timestepping the initial guess for the AB orbit (green), call the `NewtonHook` subroutine, then plot the converged solution (blue). Scroll back through the output, and compare `relative_err` for the initial guess at `iteration 0` with the final relative error.

- Comment/uncomment other initial guesses `new_x` $= \boldsymbol{x}_0$, or experiment with your own. How do they affect the number of Newton iterations taken?
  [Typically convergence takes $O(10)$ iterations, otherwise it will never converge.]

- Uncomment the initial guess for an equilibrium. Here we assume a short fixed T, too short for a PO; T is not permitted to change, otherwise $||\boldsymbol{\Phi}^T(\boldsymbol{x}) - \boldsymbol{x}||$ could be reduced by simply taking $T \to 0$. Check that `MAIN` can find the analytic equilibrium solution $(\pm\alpha, \pm\alpha, r-1)$, where $\alpha = \sqrt{(r-1)\,b}$.

4.1 Adapting the code for your own use

- For a very large system, for which you might consider parallelization, you should probably use the Fortran90 version.

- Experiment with the Template/Example first, to get used to how the code is set up. The initial guess is put in `new_x`.

- Note that at present, `new_x(1)` $= T$ (the period), and `new_x(2:end)` $= \boldsymbol{x}$ (the state).

- The place to start is then `steporbit`. If you already have an existing timestepping code, it could do something as simple as call it externally via system calls:

```
function y = steporbit(ndts_,x)
  persistent dt

  if ndts_ ~= 1                % Set timestep size dt=T/ndts_
     dt = x(1) / ndts_ ;       % If only doing one step to calc \dot{x},
  end                          % then use previously set dt.

  a = x(2:end) ;

  WRITE DATA TO FILES:
     dt     timestep size
     ndts_  number of steps to take
     a      initial condition

  LOAD STATE, TIMESTEP, SAVE STATE:
     system('run_my_code.exe')

  LOAD TIMESTEPPED STATE: --> a

  y = zeros(size(x)) ;
  y(2:end) = a ;
end
```

- `saveorbit` is called at the end of each Newton iteration. Add code here to save the current state `new_x`.
- If your inner product corresponds to $\langle \boldsymbol{a}|\boldsymbol{b}\rangle = \boldsymbol{a}^T W \boldsymbol{b}$ where W is a diagonal matrix of positive weights, and here T is the transpose, then pass $\boldsymbol{x}' = W^{\frac{1}{2}}\boldsymbol{x}$ to the code. The existing functions that take inner products then need no modification.
- For parallel use with MPI+Fortran, the `NewtonHook` and `GMRES` codes do not need changing: Split vectors over threads and let each thread pass its section to `NewtonHook`. The only place where an MPI call is required is an MPI_Allreduce in the `dotprod` function. To avoid all threads outputting information, set `info=1` on rank 0, and `info=0` on all other ranks.

Acknowledgements

AW would like to thank Rich Kerswell, Predrag Cvitanović (`chaosbook.org`), John Gibson (`channelflow.org`), Marc Avila and many others for their generous support in many forms. Developed under EPSRC grants EP/K03636X/1, EP/P000959/1.

References

Cvitanović, P. and J. F. Gibson (2010). Geometry of turbulence in wall-bounded shear flows: Periodic orbits. *Phys. Scr. T 142*, 014007.

Dennis, J. and R. Schnabel (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *J. Atmos. Sci. 20*, 130–141.

Viswanath, D. (2003). Symbolic dynamics and periodic orbits of the Lorenz attractor. *Nonlinearity 16*, 1035–1056.

Viswanath, D. (2007). Recurrent motions within plane Couette turbulence. *J. Fluid Mech. 580*, 339–358.

Willis, A. (2017). The Openpipeflow NavierStokes solver. *SoftwareX 6*, 124–127.

Willis, A. P., P. Cvitanović, and M. Avila (2013). Revealing the state space of turbulent pipe flow by symmetry reduction. *J. Fluid Mech. 721*, 514–540.

Willis, A. P., K. Y. Short, and P. Cvitanović (2016). Symmetry reduction in high dimensions, illustrated in a turbulent pipe. *Phys. Rev. E 93*, 022204.