Autodifferentiation-Accelerated Methods for Finding Recurrent Solutions in Turbulent Flows

Stanisław Kowalski[1]
PHYS 6260 Computational Physics, Dr. John Wise[1]
Github: `https://github.com/KowalskiAnalytics/JAX-RPO-Finding`
[1]*Georgia Institute of Technology, Howey Physics Building, Atlanta, Georgia*

May 1st, 2023

ABSTRACT
Prominent in nonlinear physics, particularly that of turbulent fluid flows, is the problem of finding exactly recurrent flow states, which stay at, return to, or shift and return to their original condition through time. These states are very difficult to find because turbulent flows are inherently chaotic, so any minute perturbation can be amplified to completely change the flow in short order and the solutions are unstable, yet need to be integrated very stably at high resolution over durations the same order as their runaway timescales. We show for the simplest example of such a system, the Kuramoto-Sivashinsky laminar flame equation, that we can make use of certain properties of the system, such as its quadratic nonlinearity and finite number of unstable directions (both common among fluid systems), via specialized methods such as pseudospectral integrators and Newton-Krylov nonlinear regression, to efficiently and stably "train" the fluid flow into a recurrent state, and thus provide testing grounds for general methods of finding recurrent solutions.

Key words: Turbulence – RPOs – Newton-Krylov – Autodifferentiation – JAX

1 INTRODUCTION: TURBULENT FLOWS

Many nonlinear Partial Differential Equations, especially those representing turbulent fluid flows, will have recurrent solutions of forms:

- Relative Equilibrium (RE): $\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}(\mathbf{x} + \mathbf{c}t, t)$
- Periodic Orbit (PO): $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x}, t + \tau)$
- Relative Periodic Orbit (RPO): $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x} + \mathbf{c}\tau, t + \tau)$

Where $\mathbf{u}(\mathbf{x}, \mathbf{t})$ is the flow state at coordinate $\mathbf{x}$ and time t, and $\mathbf{c}$ and τ are constants representing the shift speed and period. Finding and analyzing recurrent states in general, and RPOs in particular, is an important problem in characterizing the state space of a PDE, in particular the geometry of its turbulent attractor as per Willis et al. (2016).

Perhaps the best known example of a turbulent attractor is the Lorenz attractor (Fig 1) of an idealized weather system with three variables, a well-studied example of a low-dimensional system exhibiting complex chaotic structure. The attractor has a dimension of approximately two[1] which is intuitively seen by observing that it forms a surface of solutions in 3d space. The attractor is stable in one dimension, so all trajectories off the attractor are "pulled" towards it as they are evolved in time Miles (1984), while trajectories on the surface are unstable in two directions (those of the surface), and any states $\mathbf{u}_0$ and $\mathbf{u}_0 + \delta\mathbf{u}_0$ initially separated by a small perturbation $\delta\mathbf{u}_0$ spanning either unstable dimension will diverge at an initially exponential rate according to a Lyapunov stability analysis. To be precise, the separation of states $\delta\mathbf{u}(t)$ will grow as $\|\delta\mathbf{u}(t)\| \approx e^{\lambda t}\,\|\delta\mathbf{u}_0\|$ for small t where λ is the Lyupanov exponent. Kaplan & Yorke (2006) This allows a precise method of characterizing the number of stable and unstable directions of a solution, i.e. by finding the number of $\lambda < 0$ and $\lambda > 0$ respectively.

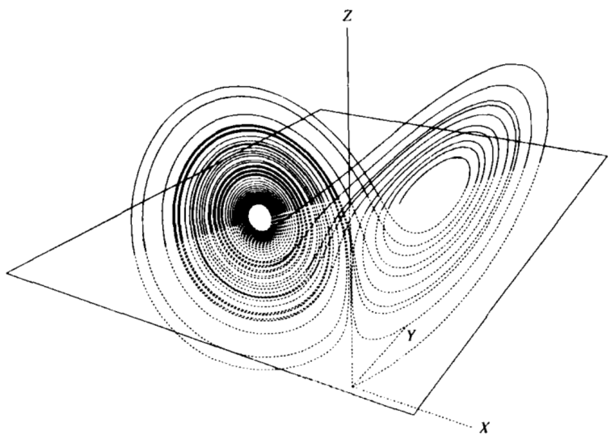

Figure 1. The Lorenz Attractor, Miles (1984)

Thus, efficient methods of finding RPOs are paramount to characterizing the dimensionality of turbulent attractors they span,

[1] A Lyapunov dimension of 2.06, but that is outside the scope of this paper. Kaplan & Yorke (2006)

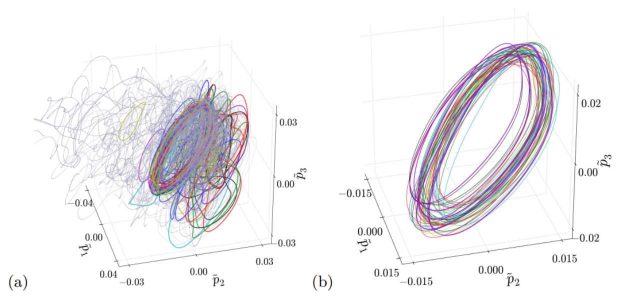

Figure 2. (a): Turbulent trajectories "shadowing" a collection of RPOs, (b) A low-dimensional projection of RPOs in turbulent pipe flow Willis et al. (2016)

as well as travel between regions of relative stability where a turbulent solution approaches close to an unstable RPO (a process called "shadowing" in Willis et al. (2016)) and behavior of chaotic regions such as the grey area of Fig. 2a. They can also be used to characterize the parameter space of a solution, e.g. the boundary conditions at which turbulence always occurs or never occurs, which can be determined by "continuing" a solution by slowly varying its boundary conditions and reconverging the recurrent solution to find a family of similar recurrent solutions across various boundary conditions. Of particular interest is the behavior of bifurcations of these families in the parameter space of Reynolds number, as in Crowley et al. (2023).

RPOs can also be used to put extremely good limits on time-averaged statistics of observables in turbulent systems, as the spectrum of any evolution operator for a dynamical system, and all its global properties such as observables, is dual to the (local) spectrum of periodic orbits and their properties, which is the key result from Cvitanović et al. (2016) making analysis of complex behaviors of unstable turbulent systems over long timeframes possible, but only through knowledge of the spectrum of recurrent flow states.

1.1 The KS Equation

The simplest spatially and temporally extended PDE is one which models thermal diffusion in a laminar flame front, the Kuramoto-Sivashinsky (1983) "KS" equation. In one dimension, it has a simple form:

$$\partial_t u + \partial_x^2 u + \nu\partial_x^4 u + u\partial_x u = 0 \tag{1}$$

Where u is flow velocity, ν is kinematic viscosity, and ∂_x^n denotes the nth partial derivative with respect to x, position. The last, non-linear term is frequent in fluid flows, as it describes convection, and is therefore of particular interest, as it is likewise the term which makes the Navier-Stokes equations nonlinear. Conveniently, it can be expressed as a quadratic nonlinearity, allowing us to re-express 1 as:

$$\partial_t u + \partial_x^2 u + \nu\partial^4 u + \tfrac{1}{2}\partial_x(u^2) = 0 \tag{2}$$

If we substitute a simple wavelike ansatz of form $e^{ikx}f(t)$, we observe a strong coupling between any spatial frequency and its double:

$$e^{ikx}f'(x) - k^2 e^{ikx}f(t) + \nu k^4 e^{ikx}f(t) + ike^{2ikx}f(t) = 0 \tag{3}$$

This clearly does not factor, indicating that a pure spectral method cannot trivially solve this equation. More interestingly however, this means that if any spatial frequency exists, its double will soon arise in any temporally extended solution, so any good integration scheme should exhibit frequency-doubling bifurcations.

Furthermore, it should be noted that in one dimension, the KS equation exhibits Gallilean-invariance, i.e. shifting perspective to any other inertial observer, including one with a different velocity, will also be a valid solution. Thus, per Cvitanović et al. (2010) if a velocity profile $u(x,t)$ is a solution, so is $u(x+ct,t)+c$, where c is a constant velocity. This dramatically simplifies finding Relative Periodic Orbit solutions, as it is possible express them as finding a regular Periodic Orbit of the equation

$$\begin{aligned}\partial_t u + \partial_x^2 u + \nu\partial^4 u + \tfrac{1}{2}\partial_x(u^2 + 2cu + c^2) = 0\\ \partial_t u + \partial_x^2 u + \nu\partial^4 u + \tfrac{1}{2}\partial_x(u^2) + c\partial_x u = 0\end{aligned} \tag{4}$$

Thus, we may develop our framework for finding RPOs simply by developing one for finding POs, a slightly easier problem. As we will see, Relative Equilibria can also arise out of our solving, but will generally be considered degenerate solutions of our optimizer, as they do not have temporally varying behavior, and thus are the trivial solution to regress to.

2 METHODS FOR TIME INTEGRATION

Any framework for finding recurrent flow states must have two parts, an integrator representing the time evolution of the physical system being studied, capable of stable integration over long timescales, and a non-physical nonlinear regression method, minimizing the difference between an initial condition and its propagation through time after one period. For reasons discussed below, and in Akrivis & Smyrlis (2004) these are respectively typically satisfied with pseudo-spectral Implicit-Explicit (IMEX) schemes exploiting the quadratic nonlinearity in some way, and matrix-free Newton-Krylov regression making use of the finite number of unstable directions (and thus large eigenvalues of the Jacobian of the residual error between an initial condition and its propagate).

2.1 Pseudospectral KS Integrator

For periodic boundary conditions $u(x,t) = u(x+L,t)$, the most convenient version of this problem, we can employ operator splitting to make the linear part of the KS equation diagonal in frequency space:

$$\begin{aligned}\partial_t u &= \mathcal{L}u + \mathcal{N}u\\ \mathcal{L} &= -\partial_x^2 - \nu\partial_x^4\\ \mathcal{N}u &= -\tfrac{1}{2}\partial_x(u^2)\\ \mathcal{F}[\mathcal{L}] &= -(ik)^2 - \nu(ik)^4 = k^2 - \nu k^4\end{aligned} \tag{5}$$

Using $\mathcal{F}[\mathcal{L}]$ to denote the Fourier transform of $\mathcal{L}$ for some spatial frequency k, which can be seen from the Fourier expression of $u(x)$ as $\int dk\hat{u}(k)e^{ikx}$, or in the discrete case, $\sum_k \hat{u}(k)e^{ikx}$, where $\hat{u} = \mathcal{F}\{u\}$. Because derivatives over x commute with integrals or sums over k, for each k, it is apparent that $\partial_x\hat{u} = ik\hat{u}$. Thus, if we use u^+, u, and u^- to denote the new, current, and previous timestep of the flow state in the temporal discretization, the implicit Crank-Nicholson method for solving $\partial_t u = f(u)$,

$$\frac{u^+ - u}{\Delta t} = \frac{1}{2}f(u^+) + \frac{1}{2}f(u) \tag{6}$$

Is diagonal for $\mathcal{F}[\mathcal{L}]$; it gives an algebraic relationship for $\hat{u}^+$ in terms of $\hat{u}$ for each spatial frequency k independent of any other spatial frequency. It can thus be accomplished by a simply array multiplication for a discrete Fourier representation of u. This gives a very stable and simple relationship that ensures the solution to the linear operator is time-reversible.

Because of the quadratic nonlinearity, a similar trick could be applied to the nonlinear part by representing it as $\frac{1}{2}\partial_x(u^+u)$, making it likewise time-reversible, but this representation is not diagonal in Fourier space, so a linear equation solving step would have to be included, negating the advantages of working with periodic boundary conditions. This is a smart discretization to use for finite walls or similar boundaries, and is also applicable to convective nonlinearities in Navier Stokes, and so would be a good target for extending these results before moving on to NS over nonperiodic boundaries.

In our setup, however, we can retain the benefit of a diagonal linear operator by using a two-step Adams-Bashforth method,

$$\frac{u^+ - u}{\Delta t} = \frac{3}{2}f(u) - \frac{1}{2}f(u^-) \tag{7}$$

And noting that

$$\begin{aligned} f(u) = \mathcal{N}u = \tfrac{1}{2}\partial_x(u^2) = -\frac{1}{2}(ik)\mathcal{F}\{u^2\} \\ \frac{3}{2}f(u) - \frac{1}{2}f(u^-) = -\frac{1}{2}(ik)\mathcal{F}\left\{\frac{3}{2}u^2 - \frac{1}{2}(u^-)^2\right\} \end{aligned} \tag{8}$$

Which, if u^2 and $(u^-)^2$ or their Fourier transforms are precomputed, allows us to express our relation between each timestep seperately for each k (once all Fourier transforms are computed). Thus, combining our expressions for linear and nonlinear parts results in:

$$\frac{\hat{u}^+ - \hat{u}}{\Delta t} = \mathcal{F}[\mathcal{L}]\frac{\hat{u}^+ + \hat{u}}{2} - \frac{1}{2}(ik)\mathcal{F}\left\{\frac{3}{2}u^2 - \frac{1}{2}(u^-)^2\right\} \tag{9}$$

Which can be re-arranged into an algebraic relationship for each k:

$$\left(1 + \frac{\Delta t}{2}\mathcal{F}[\mathcal{L}]\right)\hat{u}^+ = \left(1 - \frac{\Delta t}{2}\mathcal{F}[\mathcal{L}]\right)\hat{u} - \frac{1}{2}(ik)\mathcal{F}\left\{\frac{3}{2}u^2 - \frac{1}{2}(u^-)^2\right\}\Delta t$$

For each timestep, this requires one Fourier transform, one inverse Fourier transform (to compute u^2), and 4 array multiplications. Thus, we have arrived at our temporal propagator.

3 METHODS FOR FINDING RECURRENT STATES

There are two complimentary optimization-based methods of finding RPO solutions to a given differential equation $\mathcal{D}$: Gradient Descent, familiar in machine learning, and Adjoint Looping, which is perhaps less intuitive. Both follow a similar structure:

- Let $\mathbf{u}(\mathbf{x}, t)$ be a vector-valued function of a generalized coordinate $\mathbf{x}$ (Cartesian position, cylindrical position, etc.) and time t, as is the case for many functions determining the state of a physical system in turbulent flow (e.g. vorticity in a spatiotemporal domain describing incompressible flow, velocity, pressure, and density in a compressible flow, etc.)
- Let $\mathcal{D}$ represent the differential equation constraining the physics of the system such that $\mathcal{D}\mathbf{u} = 0$ for all $(\mathbf{x}, t)$ is equivalent to the system satisfying all relevant physical constraints except boundary conditions.
- Let $\mathbf{u}_0(\mathbf{x}; s) = \mathbf{u}(\mathbf{x}, 0)$ be an initial condition as a function of a regression parameter s, along which we will optimize our solution in a later-specified manner. In any discretization, we will consider this a vector-valued function of s, whose dimension is the number of spatial gridpoints, and whose entries are the respective values of the initial condition at those points.
- Let P^t be our time propagation operator, such that $P^t\mathbf{u}_0 = \mathbf{u}(\mathbf{x}, t)$, and $\mathcal{D}P^t = 0$. When discretized, this will be a numerical solution routine that continues the initial state through time, e.g. via RK4, trivially via spectral method, (e.g. if $\mathcal{D} = \partial_t - \lambda$ for some constant λ, $P^t = e^{\lambda t}$).
- Let $F(\mathbf{u}_0)$ be some vector-valued error function, e.g. if we seek an exactly temporally recurrent solution, we might set $F(\mathbf{u}_0) = \mathbf{u}_0 - P^\tau\mathbf{u}_0$ for some desired period τ.
- Let $G(\mathbf{u}_0) = \langle F, F\rangle = \|F\|^2 = \int d\mathbf{x}F(\mathbf{u}_0(\mathbf{x}))^2$ be our objective function (scalar-valued error function), which we may thus consider a function of s via $G(s) = G(\mathbf{u}_0(s))$. Minimization of this function will thus be our ultimate goal, which we can accomplish via the following two optimization methods. Note that $\frac{dG}{ds} = 2\langle F, \frac{\partial F}{\partial \mathbf{u}_0}\frac{d\mathbf{u}_0}{ds}\rangle$, which suggests the following definition:
- Finally, let $f(\mathbf{u}_0) = \frac{d\mathbf{u}_0}{ds}$, our first wholly-unphysical function, which describes how we adjust our discretized $\mathbf{u}_{0,n}$ each iteration to arrive at a better guess at a solution which minimizes our objective function G, e.g. via $\mathbf{u}_{0,n+1} = \mathbf{u}_{0,n} + f(\mathbf{u}_{0,n})\Delta s$, where Δs can be set variably using stepsize control algorithms. $f(\mathbf{u}_0)$ is the only function we are free to choose, and how we set it determines our regression model.

3.1 Gradient Descent

Setting $f(\mathbf{u}_0) = -\left(\frac{\partial F}{\partial \mathbf{u}_0}\right)^{-1}F(\mathbf{u}_0)$ yields perhaps the most famous model, a simple series of linear approximations and regressions to the root thereof, a generalization of the Newton-Raphson method to a multidimensional input and output. This has exponential convergence properties, as we see from $\frac{dG}{ds} = 2\langle F, \frac{\partial F}{\partial \mathbf{u}_0}f(\mathbf{u}_0)\rangle = -2\langle F, F\rangle = -2G$, but only converges when started with a guess close to the exact solution without extensive preconditioning. Choquet & Erhel (1993) Persson & Peraire (2008)

It should be noted that $(\frac{\partial F}{\partial \mathbf{u}_0})$ is the Jacobian matrix J of the linear map F from $\mathbf{u}_0$ to vectorized error, whose size and computational cost are $O(n_x^2)$ in the number of gridpoints n_x in the spatial discretization of $\mathbf{x}$, which is usually extremely expensive for minimal benefit (n_x can be hundreds of thousands of points for a 3d system, while only a few dozen largest principal components contribute significantly to the solution). Thus, matrix-free (Jacobian-free) methods such as Newton-GMRES are used to solve the problem of $Jf(\mathbf{u}_0) = F(\mathbf{u}_0)$ via the Generalized Minimal Residual Method (GMRES), which finds the vector $\mathbf{a}$ in the Krylov subspace of J for an arbitrary staring vector b, $\{b, Jb, J^2b, ...J^{n-1}b\}$ (which can be efficiently generated), which has minimal residual in satisfying the equation $J\mathbf{a} = F(\mathbf{u}_0)$ via Arnoldi iteration. Qin et al. (2000)

In our system, the size n_x (usually 256 or 1024) is not remotely close to justify GMRES over LU factorization solely on the basis of using a matrix-free Jacobian, but its ability to deal well with sparse matrices with a low number of high-value eigenvectors, which we show, and the fact that size constraints become even more relevant for more turbulent systems, which we also show, do make it an essential algorithm to demonstrate in developing a general framework for finding recurrent solutions.

Notably, convergence between Newton iterations can be accelerated considerably by use of the **LGMRES** algorithm from Baker et al. (2005), which carries over a set of "outer" vectors between iterations, making use of the similarity between the eigenspaces of Jacobians at nearby points on the search manifold, dramatically accelerating convergence of each Krylov iteration, as discussed in Results, especially where steps taken are small - the slow parts of a Newton loop.

3.2 Adjoint Looping

Setting $f(\mathbf{u}_0) = -(\frac{\partial F}{\partial \mathbf{u}_0})^\dagger F(\mathbf{u}_0)$, where $(\frac{\partial F}{\partial \mathbf{u}_0})^\dagger$ is the adjoint of J, defined such that $\langle a, Jb\rangle = \langle J^\dagger a, b\rangle$, and thus $\frac{dG}{ds} = -2\langle F, \frac{\partial F}{\partial \mathbf{u}_0} f(\mathbf{u}_0)\rangle = -2\langle J^\dagger F, f(\mathbf{u}_0)\rangle = -2\langle J^\dagger F, J^\dagger F\rangle = -2\left\|J^\dagger F\right\|^2$, which is likewise an always-negative quantity, but one with different convergence properties. Specifically, the $(\frac{\partial F}{\partial \mathbf{u}_0})^\dagger$ term is larger the further $\mathbf{u}_0$ is from our optimal solution, so intuitively, we would expect it to converge faster the further our guess is from our desired solution, and slower the closer it is, which is what has been shown in literature. Azimi et al. (2020)

This is an extremely convenient property, especially coupled with the fact that this method exhibits global convergence, so adjoint looping can be started from a completely randomized guess, regress quickly in the direction of a solution satisfying our conditions, and switch to our complementary method of gradient descent with opposite convergence rates once the adjoint method slows down, indicating proximity to an optimum. Azimi et al. (2020)

The difficulty with finding such an adjoint is that the adjoint of a differential operator under an integral norm (such as our $\langle,\rangle$) is that via integration by parts, shifting its action onto the other term requires mixing with boundary conditions Juniper (2010), while for a periodic flow state, it can simply be taken as the transpose of a vector-Jacobian product.

3.3 Stepsize Control

The second parameter we have control over, aside from descent direction $f(\mathbf{u}_0) = \frac{\partial F}{\partial \mathbf{u}_0} = p$ here labelled p, is step size s. Mere step-limiting such as dividing the magnitude of the descent vector p by 100 or 1000, and ensuring that it is less than a percent of the magnitude of the current state $\mathbf{u}$ failed to match scipy's **optimize.newton_krylov**, which uses Armijo's algorithm for a line search for a step size that both guarantees residual decrease and finds a *sufficient* decrease.

The numerical method specification Knoll & Keyes (2004) they cite only guarantees the former, as it tries halving s until the condition $\|\mathbf{F}(\mathbf{u}+s\mathbf{p})\| < \|\mathbf{F}(\mathbf{u})\|$ is satisfied. Kelley (1995) defines the formal Armijo condition for "sufficient" decrease, $\|\mathbf{F}(\mathbf{u}+s\mathbf{p})\| < (1-\alpha s)\|\mathbf{F}(\mathbf{u})\|$ for any $\alpha \in [0,1)$, and proves that such a step size always exists from the 2nd order Taylor expansion of the residual. The documentation does, however, specify a good rule for increasing the initial trial step size s_l of a step l by:

$$s_l = s_{l-1} * \frac{\left\|F(\mathbf{u}^{l-1})\right\|}{\left\|F(\mathbf{u}^{l-2})\right\|} \tag{10}$$

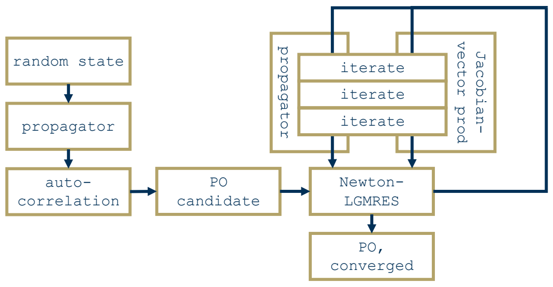

Figure 3. Structure of the program for finding Periodic Orbits (POs) of the KS equation using Newton-Krylov regression.

Which helps when residuals start getting smaller and derivatives (and thus p) generally start getting smaller (as a minimum is in an area of positive 2nd derivative), but does little to help when residual decrease starts to stagnate, at which point we suspect scipy's **newton_krylov** automatically bumps the initial stepsize to 1, which seems supported by our convergence plots. This proved unnecessary for almost exactly replicating the overall behavior of Newton-Krylov, and this "stagnation" condition is not formally defined in the documentation, so we did not try to replicate it, although it could be useful to closing the final convergence, especially as p gets very large when residuals get very close to 0 but stagnate.

3.4 Program Structure

The program structure for the main recurrent state solving loop is explained in Fig 3. We start with a random seed state, for example $u_0(x) = \cos(\frac{2\pi}{L}x) + 0.1\cos(\frac{4\pi}{L}x)$, and propagate it to get a timeseries of temporally connected states. This allows us to search for recurrent state candidates of our liking, by a process we term "autocorrelation," which refers to finding the difference between two candidate states at t_i, t_j, $\left\|u(t_i) - u(t_j)\right\|^2$, and picking out the two states with the minimum of such difference, for which the first is guaranteed to be temporally connected to the latter with minimum difference, the best candidate for a periodic orbit. We could also roll each of these pairs over a spatial shift to find the best candidate for both period and shift velocity, but this is unnecessary for the KS equation due to its Gallilean invariance - all that would be added is the ability to find slightly better, equivalent candidates.

Once this "best pair" is computed, we pass it to our optimization loop as an initial guess. The optimization loop, for each during each iteration, performs an LGMRES residual minimization calculation to find the descent direction matching our specifications above, with each iteration for each subsequent Krylov vector involving time integration of an initial state over the prospective period, as well as a Jacobian-vector product calculation, computed by propagating automatic differentiation through each integration step, or via numerical differentiation, comparing the results of two separate integrations offset by a tiny step in the trial direction. Both methods have a similar constant-time overhead, but the former is full-precision accurate without truncation error. The optimal descent direction computed by LGMRES is then passed to the stepsize optimizer described above, which adjusts its guess in that direction by that step, the iterates again until it converges to a periodic orbit!

matrix	GMRES	LU
A	10^{-1}	10^{-10}
ADA^T	10^{-1}	10^{2}

Table 1. residual error for $N = 10,000$. A is a random NxN matrix, $D = e^{\Lambda}$ where Λ is a diagonal array of N uniform random numbers between $[-90, 10)$.

4 RESULTS

Each individual component of this project had to be rigorously tested, from the simplest building block such as manipulation in frequency space for spectral and pseudospectral methods, or GMRES for random matrices with finite spectra, all the way to testing the interactions of multiple algorithms and verifying them across different implementations and grid sizes. With everything verified, we were able to consistently regress to the same periodic orbits with better convergence properties than scipy's **newton_krylov** via JAX with automatic differentiation.

4.1 GMRES vs LU

GMRES operates over the Krylov subspace because it closely approximates the span of the eigenspaces with the largest eigenvalues, and so is a good representation for a sparse matrix, which our Jacobian often is, because Krylov subspaces quickly pick out the finite number of unstable directions in our nonlinear system.

We can demonstrate the utility of GMRES for sparse/singular matrices. First, we generate a random matrix A, which is in all likelihood invertible, because invertible matrices are dense over the space of square matrices of a given dimension. For this matrix, **gmres** solves $Ax = b$ faster but less accurately than **linsolve** (a wrapper for Fortran LU decomposition routine **_gsev()**), but, if we create a matrix which is 10% high-eigenvalue eigenspaces, and 90% very low-value eigenspaces, we find that **gmres** converges to an equally accurate result equally as fast, while **linsolve** provides a less accurate result in a longer time, struggling with a nearly-singular matrix.

Refer to Table 1 for order of magnitude of maximum error of $A\tilde{x} - b$ for approximate solutions $\tilde{x}$ generated by the two methods.

4.2 Newton-Krylov Implementations

We test our own Newton-Krylov implementations on the integro-differential equation from the scipy documentation's example for its usage:

$$(\partial_x^2 + \partial_y^2)P + 5\left(\int_0^1 \int_0^1 \cosh(P)\; dx\; dy\right)^2 = 0 \tag{11}$$

All produce the same result, visually depicted in Fig 4, with scipy's numerical differentiation algorithm with better stepsize line search converging in 8 iteration on the numpy residual code, and somehow 1 iteration on the compiled JAX residual code. Custom Newton loops using scipy's LGMRES with a custom LinearOperator defined by either JAX's automatic differentiation of a compiled residual function, or a numpy numerical differentiation function, both converging in 5 iterations, with almost exactly the same steps taken up to slight numerical implementation differences.

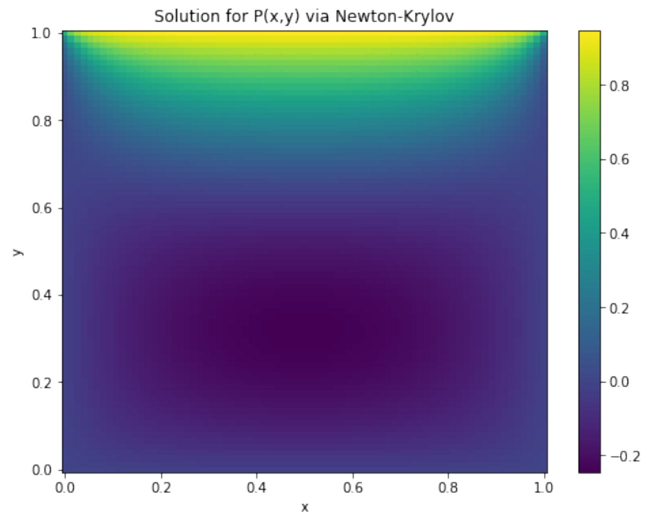

Figure 4. Standard test case for Newton-Krylov, solution for P in equation (11) with $P = 0$ at all boundaries except $y = 1$, where $P = 1$.

4.3 Adjoint Looping

Adjoint looping with $p = J^{\dagger}F$ evaluated as $p = (F^T J)^T$ via JAX's vector-Jacobian product, which is theoretically correct for the KS equation with periodic boundary conditions, but not for the test case, failed to converge in both. It failed to make a timestep that would satisfy the decrease condition for the KS equation, in a scenario where all three other methods managed to reduce residual error, and it managed to make 1 iteration in the test case, which happened to directly match the other methods' first step. This could support the idea that adjoint looping displays inferior convergence properties once close to the solution, as both cases were started as such, but it is more likely that $p = (F^T J)^T$ is simply an incorrect way of implementing adjoint looping, as per Juniper (2010). For the KS problem with periodic boundaries, however, it displays the same behavior.

4.4 The Heat Equation

A test case for spectral integrators, using numpy's FFT on periodic boundaries, and a comparison between spectral and FTCS integrators in both numpy and JAX's implementation thereof was performed on the heat equation, Fourier's original motivating problem, with very good agreement between methods. The heat equation, its finite difference implementation, and its spectral solution are:

$$\begin{aligned} &\partial_t \Theta = \alpha \partial_x^2 \Theta \quad \implies \Theta_n^+ = \Theta_n + \alpha \frac{L^2}{n_x^2}\left(-\Theta_{n-1} + 2\Theta_n - \Theta_{n+1}\right)\Delta t \\ &\partial_t \hat{\Theta} = \alpha (ik)^2 \hat{\Theta} \implies \hat{\Theta}(t) = e^{-\alpha k^2 t}\hat{\Theta}(0) \end{aligned} \tag{12}$$

We observe in Fig 5 very good agreement between these two methods on periodic boundaries, with low (3% maximum) error even in unphysical starting conditions with infinite derivative, and extremely low (0.6%) error after the initial smoothing of the nonphysical discontinuity.

We observe further in Fig 6 that when extended to non-periodic boundary conditions by making the modification $\Theta' = \Theta - a_1 x - a_2$,

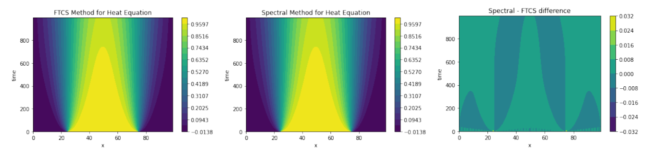

Figure 5. Comparison of FTCS and Spectral methods for heat equation on periodic bounds.

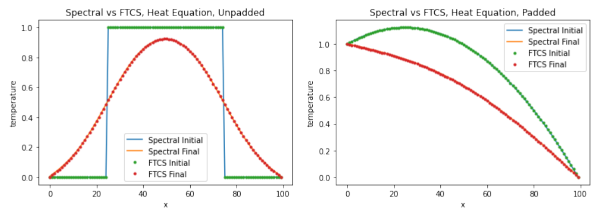

Figure 6. Comparison of FTCS and Spectral methods for heat equation on periodic and nonperiodic bounds.

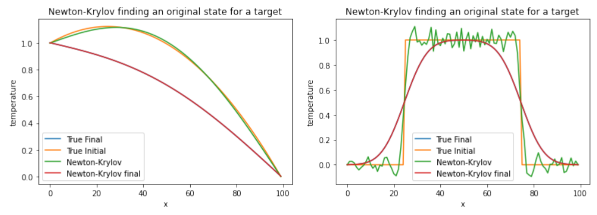

Figure 7. Newton-Krylov regression to a parabolic and square wave initial condition given their propagates.

which does not affect the temporal derivative nor the second spatial derivative, solving the Fourier problem in frequency space on Θ' with periodic boundaries, and reintroducing the linear ramp between boundaries, we still maintain good agreement. It should be noted, however, that the transformed input Θ' must be padded with a negative copy of itself to fix odd symmetry at the bounds, lest an initial computation including even modes decay to make the bounds nonzero over time.

Additionally, we can see in Fig 7 how Newton-Krylov behaves on a spectral method such as this by constructing a target state that is the time evolution of a parabola or a square wave, and then we attempt to use scipy's builtin Newton-Krylov solver to regress to the correct initial state. Since the heat equation is the definition of unstable in reverse (it's a forward-time smoothing operator, so in reverse it exponentially amplifies high frequency noise of any kind), we unsurprisingly get a large amount of high-frequency noise for the discontinuous square wave solution, but not for the parabolic solution. It should be noted, however, that the final states of our regressed initial states exactly match the target final states, even if the regressed and true initial states have slight (especially high-frequency) discrepancies. This is partly out of a tendency for overzealous optimization (analogous to overfitting if we treat the initial state as a parameter set for optimization, and the target state the data), and partly because the higher frequencies decay by the time the final state is reached, making them irrelevant in the computation of the residual.

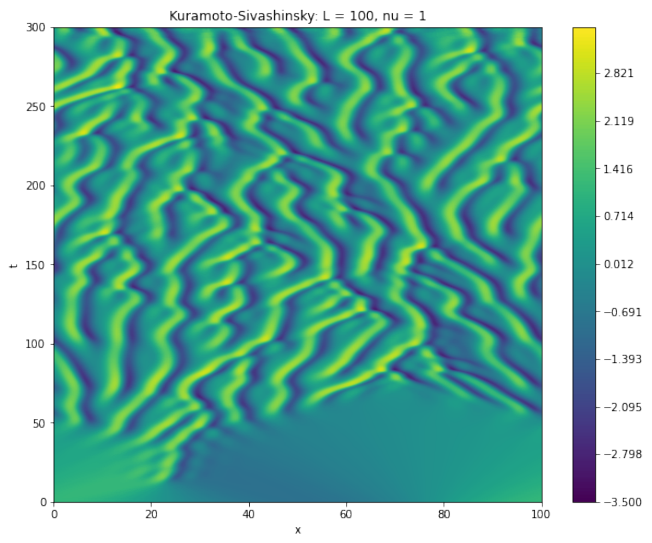

Figure 8. KS equation for $L = 100$, demonstrating clear chaotic dynamics and frequency-doubling bifurcations.

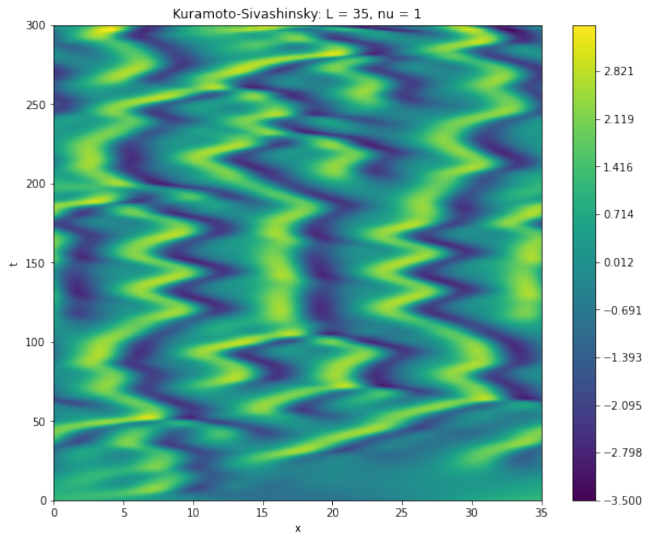

Figure 9. KS equation for $L = 35$, demonstrating a very promising candidate for a periodic orbit.

4.5 Kuramoto-Sivashinsky

Equipped with that successful test case of both spectral integrators and nonlinear optimization, we proceed to implement the pseudospectral CN+AB integrator for the KS equation described in Section 2.1, and we indeed observe frequency-doubling bifurcations (Fig 8) and recurrent states (Fig 9), focusing on the $L = 35$ regime for the most immediately promising candidates. However, we should restrict ourselves to periods of less than 50 non-dimensional time units, as our scheme breaks down after that for $L = 35$ regardless of the temporal resolution (Fig 10). The integrator is however not very sensitive to spatial resolution change, and yields consistent results within 50 time units across many different resolutions, indicating that the one we've selected is sufficient for examining this state.

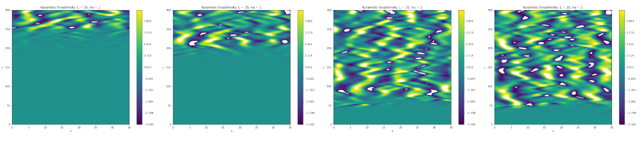

Figure 10. Half and Quarter spatial resolution residuals for the same integration at L=35, Double and Half temporal resolution integration residuals.

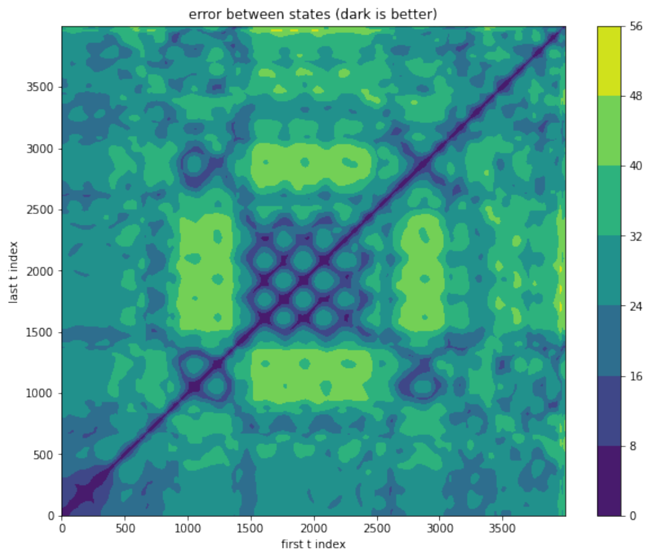

Figure 11. Autocorrelation plot for $L = 35$, demonstrating mathematically our clear visial intuition for the presence and location of recurrent solutions.

4.6 Autocorrelation

We can formalize this visual notion of Fig 9 containing a lot of states that are "like one another" with Fig 11, the tool described in Section 3.4 for finding the best two similar temporally separated states. It's obvious that a state and the one immediately after it will be very similar to one another, so we prescribe a 10% exclusion region (not shown here), and then take the minimum of the autocorrelation array excluding that 10% diagonal to find the best periodic orbit candidate.

In the autocorrelation plot, we see a four-looped central formation corresponding to the set of four back-and-forth wiggles in Fig 9, with the deepest off-axis minima corresponding to a single period of this periodic orbit, and another deep minimum corresponding to a period-doubled version of this periodic orbit, which is what is selected by our autocorrelation program (as excluding the 10% nearest the axis was too restrictive here).

4.7 Regressed Periodic Orbits

If we run our previously-tested implementations of Newton-Krylov regression, both Scipy's numerical differentiation-powered one, as well as our own with a JAX backend for automatic differentiation, on this or similar periodic orbit candidates, we get very similar period-doubled, converged periodic orbits (Fig 13, Fig 14)

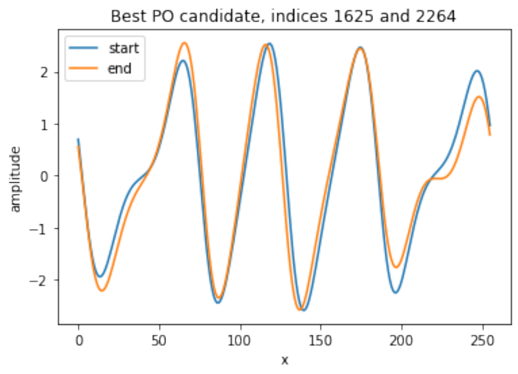

Figure 12. Quite good agreement between these two states.

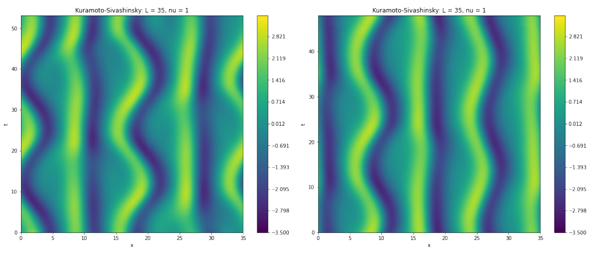

Figure 13. Similar period-doubled periodic orbits found by Scipy's numerical Newton-Krylov and our JAX-autodiff powered reimplementation

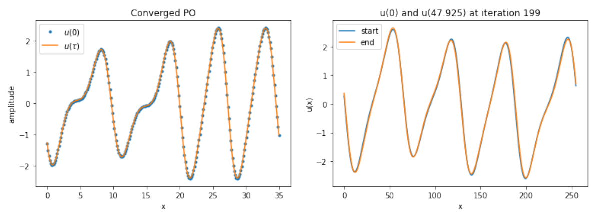

Figure 14. The corresponding initial states and states after a full period.

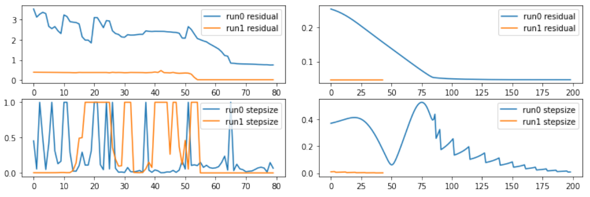

Figure 15. The residual plots and corresponding stepsize vs iteration for Scipy's and our Newton-Krylov implementations, regressing a similar state. Run restarted in both cases to allow further decrease in residual.

4.8 Convergence and Stepsize Control

By far the most important factor in getting our JAX implementation to compete with Scipy's was stepsize selection algorithms. Upon successful implementation of Armijo, our JAX algorithm went from only being able to regress to relative eqilibria (Fig 16) to being able to compete with Scipy in finding the RPO in Fig 13 Fig 14, converging at a more stable rate with an even stabler stepsize, as Scipy's routine tends to be extremely aggressive, defaulting to 1 whenever convergence slows, which could potentially benefit our algorithm if implemented.

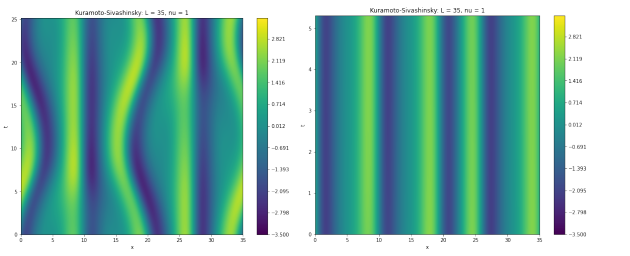

Figure 16. The resultant single-period PO if autocorrelation is re-ran on the output of the first optimization run, and a degenerate solution of a Relative Equilibirum if a lower resolution is selected and no better periodic orbit candidates appear in the initial search.

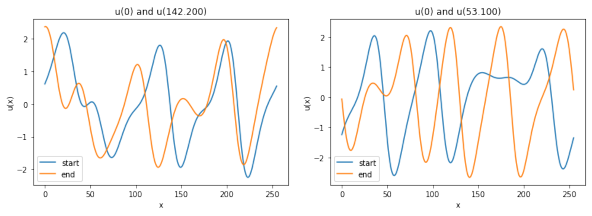

Figure 17. Very bad starts which still allowed convergence to the PO state.

4.9 Robustness and Degenerate Solutions

Fig 12 shows a very, very good initial state to use as a guess, but the methods (particular's the ones using Scipy's stepsize control algorithm) are robust enough to handle the incorrect guesses displayed in Fig 17, which arose from using too small a stepsize for the integration period being handled, and regress them to almost the same periodic orbit as shown in Fig 13, while our implementations struggled to reduce poor guesses like these to anything more complex than relative equilibria (such as that shown in Fig 16), likely due to an inferior stepsize control algorithm.

4.10 LGMRES iterations

In its current implementation, LGMRES takes between 20-120 Jacobian-vector product iterations per Newton step, which is by far faster than GMRES (40 minimum for stability), while LU factorization would take the full size of the spatial discretization, in the best case, 256. The computation of the Jacobian-vector product is the limiting factor by a large margin, with LGMRES taking essentially no overhead. Krylov subspaces are inherently only generated by iterative algorithms, so they cannot be parallelized while LU factorization could be, with minimal modification to the integrator. This advantage would, however, quickly die off as the system size increased, so this is not tested.

5 SUMMARY & CONCLUSIONS

All taken, we were able to demonstrate compatibility of a wide range of techniques and algorithms (implicit-explicit CN+AB schemes, pseudospectral methods, automatic differentiation, LGMRES iteration, Newton rootfinding) and how they can be applied together to take advantage of key properties of fluid flows and efficiently pick out the rare recurrent solutions to chaos.

Our method was competitive with Scipy's nonlinear optimizer, despite its inferior stepsize selection algorithm, as evidenced by the similar behavior of our own numerical differentiation Newton-Krylov optimizer, which should perform the exact same operations as Scipy's, up to step size selection, and its failure to converge. Finding good test cases was by far the hardest problem in this project, and once they were found, finding and resolving issues was largely simple. Going forward, the improving and reverse-engineering Scipy's Newton-Krylov algorithm for use with JAX is one of the few remaining improvements that could be done on the KS system, aside from slightly extending the regression capabilities to include a variable period and shift velocity.

If this were to be used as a testing ground for further development of tools that could be used in nonlinear flow analysis and finding of recurrent solutions, useful tools to test would include low-dimensional projection of timeseries data (to allow visualization of a turbulent attractor), computation of Lyupanov exponents and number of unstable directions for a converged flow state (which can be done with a modification of GMRES giving eigenvalues), and implementation of adjoint looping via some method other than simple vector-Jacobian multiplication.

If the methods here developed were to be extended to other systems, the most readily apparent would be Kolmogorov flow, one of the simpler 2d flow systems with capacity for rich recurrent state behaviors (as it is magnetically driven). The pseudospectral integrator herein developed could be modified for use with incompressible (purely solenoidal) flow fields, and thus the whole project could be applied with relatively minimal modification to that system. Of potential interest could be to develop a time-reversible integrator for the quadratic nonlinearity as mentioned in Section 2.1, which could be useful for the Kolmogorov problem with nonperiodic boundary conditions.

As we can see from all the options above described, as well as the performance of the algorithms and their promising results, these methods have proved highly successful as a model problem and algorithm for finding recurrent states in turbulent flows, and offers considerable extensibility into more complex future projects.

ACKNOWLEDGEMENTS

The author would like to thank Matthew Golden and Wesley Toler for their eager assistance in the mathematics and practical debugging skillset of finding recurrent solutions, their help and advice proved invaluable during the many times the author had gotten stuck in a local minimum in this extensive optimization process of a project.

DATA AVAILABILITY

All data used in this article, including figures, test cases, and any claims made, can be reproduced via codes in the respository https://github.com/KowalskiAnalytics/JAX-RPO-Finding.

REFERENCES

Akrivis G., Smyrlis Y.-S., 2004, Applied Numerical Mathematics, 51, 151

Azimi S., Ashtari O., Schneider T. M., 2020, arXiv preprint arXiv:2007.06427

Baker A. H., Jessup E. R., Manteuffel T., 2005, SIAM Journal on Matrix Analysis and Applications, 26, 962

Choquet R., Erhel J., 1993, PhD thesis, INRIA

Crowley C. J., Pughe-Sanford J. L., Toler W., Grigoriev R. O., Schatz M. F., 2023, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 381, 20220137

Cvitanović P., Davidchack R. L., Siminos E., 2010, SIAM Journal on Applied Dynamical Systems, 9, 1

Cvitanović P., Artuso R., Mainieri R., Tanner G., Vattay G., 2016, Chaos: Classical and Quantum. Niels Bohr Inst., Copenhagen, `http://ChaosBook.org/`

Juniper M., 2010, in Int. Workshop on Nonnormal and Nonlinear Effects in Aero-and Thermoacoustics.

Kaplan J. L., Yorke J. A., 2006, in , Functional Differential Equations and Approximation of Fixed Points: Proceedings, Bonn, July 1978. Springer, pp 204–227

Kelley C. T., 1995, Iterative Methods for Linear and Nonlinear Equations. Society for Industrial and Applied Mathematics (https://epubs.siam.org/doi/pdf/10.1137/1.9781611970944), doi:10.1137/1.9781611970944, `https://epubs.siam.org/doi/abs/10.1137/1.9781611970944`

Knoll D., Keyes D., 2004, Journal of Computational Physics, 193, 357

Miles J., 1984, Elsevier, pp 189–214, doi:https://doi.org/10.1016/S0065-2156(08)70045-0, `https://www.sciencedirect.com/science/article/pii/S0065215608700450`

Persson P.-O., Peraire J., 2008, SIAM Journal on Scientific Computing, 30, 2709

Qin N., Ludlow D. K., Shaw S. T., 2000, International Journal for Numerical Methods in Fluids, 33, 223

Sivashinsky G. I., 1983, Annual Review of Fluid Mechanics, 15, 179

Willis A. P., Short K. Y., Cvitanović P., 2016, Physical Review E, 93