Finding a Universal Model of Nonlinear Pattern Forming PDEs in terms of the Phase Parameter

Edward McDugald

University of Arizona

November 2, 2022

Motivation

- We are interested in pattern forming systems, characterized by modulated stripe patterns. The canonical example of such patterns is Rayleigh-Bénard convection.
- Such patterns are observed as solutions to various non-linear PDEs.
- However they are thought to all obey a universal diffusion equation in terms of the *phase* of the pattern. We would like to derive such an equation.

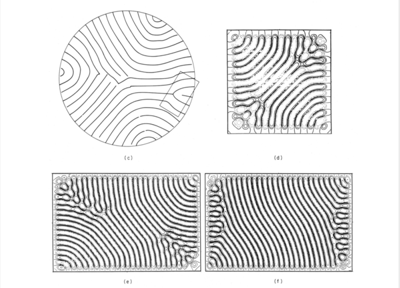


Fig. 1. Configuration of convective rolls in large aspect ratio cells. a) and b)

Cross-Newell Phase Diffusion Equation

- Many nonlinear PDEs admit an exact straight roll solution of the form

$$w = F(\theta = \vec{k} \cdot \vec{x}, A) = \sum A_n(k) \cos n\theta.$$

- For most natural patterns, the wave number $k = |\vec{k}|$ is not constant, but varies slowly over the box. So we seek a solution where the wave number is slowly modulating over the box

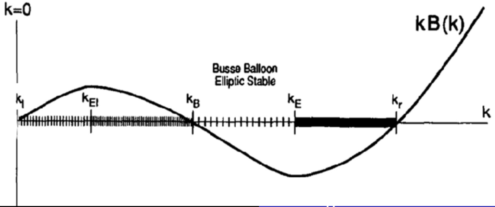
$$w = F(\theta = \frac{1}{\epsilon}\Theta(\vec{X} = \epsilon\vec{x}, T = \epsilon^2 t) = \frac{1}{\epsilon}\int \vec{k} d\vec{X}) + \epsilon w_1 + \ldots .$$

- Using this ansatz, and imposing some restrictive assumptions, one can show that the phase Θ obeys the Cross-Newell equation

$$\tau(k)\frac{\partial\Theta}{\partial T} = -\nabla\vec{k}B(k) - \epsilon^2\eta\nabla^4\Theta.$$

Cross-Newell as Regularization of a Diffusion Equation

- The Cross-Newell equation can be written as
$$\tau(k)\frac{\partial\Theta}{\partial T} = -\nabla^2\Theta B(k) - \epsilon^2\eta\nabla^4\Theta.$$
- The quantities $B(k), (kB(k))'$ determine whether or not $\tau(k)\frac{\partial\Theta}{\partial T} = -\nabla^2\Theta B(k)$ is ill-posed. It turns out that this is a proper diffusion equation only for $k_B < k < K_E$.
- For most natural patterns, the preffered wave number k is something like $k_b - c$, ie, just to the left of K_b.
- Thus, the biharmonic term is added to make the PDE well-posed.



Cross-Newell Equation as variation of Energy

- Cross-Newell can be expressed as

$$-\frac{\delta F}{\delta \Theta} = \tau(k)\frac{\partial \Theta}{\partial T}$$
$$\epsilon F = \int \left(1 - |\nabla \Theta|^2\right)^2 d\vec{X} + \epsilon^2 \int \left(\nabla^2 \Theta\right)^2.$$

- The left term measures energy due to stretching, the right term measures energy due to bending.
- If we look for "self-dual" solutions by "balancing" the bending energy with the stretching energy, then such solutions solve the Cross-Newell equation if the Guassian curvature of the solution is 0.
- For a range of pattern defects/instabilities, one can assocaite a cost given by the energy functional.

Bringing in some Machine Learning

- The Cross-Newell equation is limited in the pattern defects it can predict.
- It's derivation dates back to the 80s, and there has been no proposed modification to capture more patterns.
- I want to use some data driven methods to analyze this problem.
- I am starting with simple methods for data driven discovery of PDEs.
- I will learn how to "rediscover" known non-linear PDEs that generate patterns of interest, and then concoct an experiment to propose governing equations of the corresponding phase surfaces.

PDE Discovery via Sparse Optimization

- I combined ideas from Kutz et. al. and Hayden Schaeffer for finding governing equations via sparse regression.
- Suppose we have access to solutions $u(x, y, t)$ on a spatio-temporal grid, and in addition, we are able to estimate time derivatives for each solution, $u_t(x, y, t)$. Let us collect the time derivatives into one vector, U_t.
- We can then form a feature matrix of tall, skinny feature vectors:

$$F_u(t) = \begin{bmatrix} | & | & | & | & | & | & | & | & \dots \\ 1 & u & u^2 & u_x & u_x^2 & uu_x & u_x^2 & u_{xx} & \dots \\ | & | & | & | & | & | & | & | & \dots \end{bmatrix}$$

- We then approximate a solution of

$$\xi = \text{argmin} \| F_u(t)\xi - U_t \|_2^2 + \lambda \|\xi\|_0$$

"Discovering" Swift Hohenberg: Derivatives, Sampling and Feature Generation

- I used Operator Splitting and Exponential Time Differencing to obtain numerical solutions of Swift-Hohenberg $u_t = -\Delta^2 u - 2\Delta u + (R-1)u - u^3$.
- I used backward finite differences for my time derivatives, and spectral derivatives for all spatial derivatives.
- I used a "coarse" grid for sampling. Ie, I took 20 time steps, and down sampled the spatial grids by a factor of 8.

```python
num_t = 20
x_subsample = 8
y_subsample = 8
t_vals = np.arange(1,len(t),math.floor(len(t)/num_t))
x_vals = np.arange(0,nx,x_subsample)
y_vals = np.arange(0,ny,y_subsample)
```

```python
for t_idx in t_vals:
    print(t_idx)
    uu = U[:,:,t_idx]
    uu_t = BackwardDiff(U[:,:,t_idx],U[:,:,t_idx-1],dt)
    uu_x = SpectralDerivs(U[:,:,t_idx],Lx,Ly,'x')
    uu_y = SpectralDerivs(U[:,:,t_idx],Lx,Ly,'y')
    uu_xx = SpectralDerivs(U[:,:,t_idx],Lx,Ly,'xx')
    uu_yy = SpectralDerivs(U[:,:,t_idx],Lx,Ly,'yy')
    uu_xy = SpectralDerivs(U[:,:,t_idx],Lx,Ly,'xy')
    lapuu = SpectralDerivs(U[:,:,t_idx],Lx,Ly,'laplacian')
    biharmuu = SpectralDerivs(U[:,:,t_idx],Lx,Ly,'biharmonic')
    for x_idx in x_vals:
        for y_idx in y_vals:
            u[i] = uu[x_idx,y_idx]
            u_t[i] = uu_t[x_idx,y_idx]
            u_x[i] = uu_x[x_idx,y_idx]
            u_y[i] = uu_y[x_idx,y_idx]
            u_xx[i] = uu_xx[x_idx,y_idx]
            u_yy[i] = uu_yy[x_idx,y_idx]
            u_xy[i] = uu_xy[x_idx,y_idx]
            lapu[i] = lapuu[x_idx,y_idx]
            biharmu[i] = biharmuu[x_idx,y_idx]
            i+=1
```

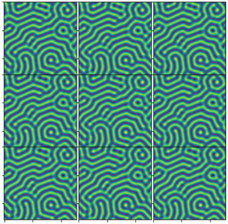
I used Sequentially Thresholded Ridge Regression. This is an iterative method which solves $\xi = \text{argmin}\|F_u(t)\xi - U_t\|_2^2 + \lambda\|\xi\|_2^2$ at each step, and applies a "hard" threshold to coefficients below a certain size, before repeating. The threshold tolerance is determined empirically by using an 80/20 test/train split. We used $\lambda = 1e - 5$.

Algorithm 1: STRidge($\mathbf{\Theta}, \mathbf{U}_t, \lambda, tol,$ iters)

$\hat{\xi} = arg\,min_\xi \|\mathbf{\Theta}\xi - \mathbf{U}_t\|_2^2 + \lambda\|\xi\|_2^2$ # ridge regression
bigcoeffs = $\{j : |\hat{\xi}_j| \geq tol\}$ # select large coefficients
$\hat{\xi}[\sim$ bigcoeffs$]$ = 0 # apply hard threshold
$\hat{\xi}$[bigcoeffs] = STRidge($\mathbf{\Theta}[:,$ bigcoeffs$], \mathbf{U}_t, tol,$ iters $- 1$)
recursive call with fewer coefficients
return $\hat{\xi}$

"Discovering" Swift Hohenberg- Results

Notebook Demo



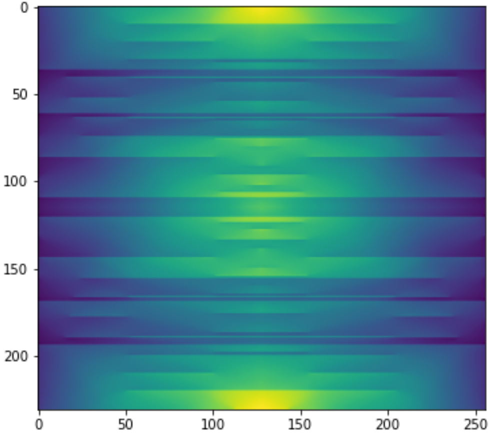
```
c = TrainSTRidge(X,u_t,10**-5,1)
print_pde(c, description)

u_t = (-0.515199 +0.000000i)u
    + (-1.030736 +0.000000i)u^3
    + (-2.061575 +0.000000i)lapu
    + (-1.030717 +0.000000i)biharmu
```

Next Steps

- Having successfully "rediscovered" Swift-Hohenberg, I am ready to extend the method to analyze the Cross-Newell Equation.
- A simple first experiment would be to generate Swift-Hohenberg solutions that obey the restrictions imposed in the Cross-Newell derivation. If I can generate phase surfaces from the solutions, I can run the experiment on the phase surfaces, and see if such an experiment predicts Cross-Newell.
- It would be neat to derive Cross-Newell, or a variant of it, directly from temperature surface simulations. I am wondering if there is a way to combine a neural network, or other machine learning technique to make this work.
- First step is to find a reliable method for estimating phase surfaces, and return to the mathematical analysis to understand the Cross-Newell limitations.

Phase Surface Depiction



References

- MC Cross and Alan C Newell. Convection patterns in large aspect ratio systems. 1984
- Alan C. Newell and Shankar C. Venkataramani. The universal behavior of modulated stripe patterns 2022
- Kutz Et. Al. Data-driven discovery of partial differential equations. 2017
- Hayden Schaeffer Learning partial differential equations via data discovery and sparse optimization, 2017
- My Codes