Numerical computation of coherent structures in spatially-extended systems

Daniele Avitabile
Centre for Mathematical Medicine and Biology
School of Mathematical Sciences
University of Nottingham

ICMNS16, Antibes - Juan Les Pins 29/05/16

How to cite this tutorial and the accompanying codes:

Daniele Avitabile, "Numerical computation of coherent structures in spatially-extended systems", Second International Conference on Mathematical Neuroscience, Antibes Juan-les-Pins, 2016

Example of coherent structures

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations $\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$

Boundary conditions $\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$

Initial conditions $u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations $\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$

Boundary conditions $\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$

Initial conditions $u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

- Several numerical packages available

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations $\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$

Boundary conditions $\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$

Initial conditions $u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

- Several numerical packages available
  - Auto, MatCont, XPP, Coco, DDE-Biftool, Knut, etc.

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations $\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$

Boundary conditions $\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$

Initial conditions $u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

- Several numerical packages available
  - Auto, MatCont, XPP, Coco, DDE-Biftool, Knut, etc.
  - PDE2Path, LOCA, OOMPH

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

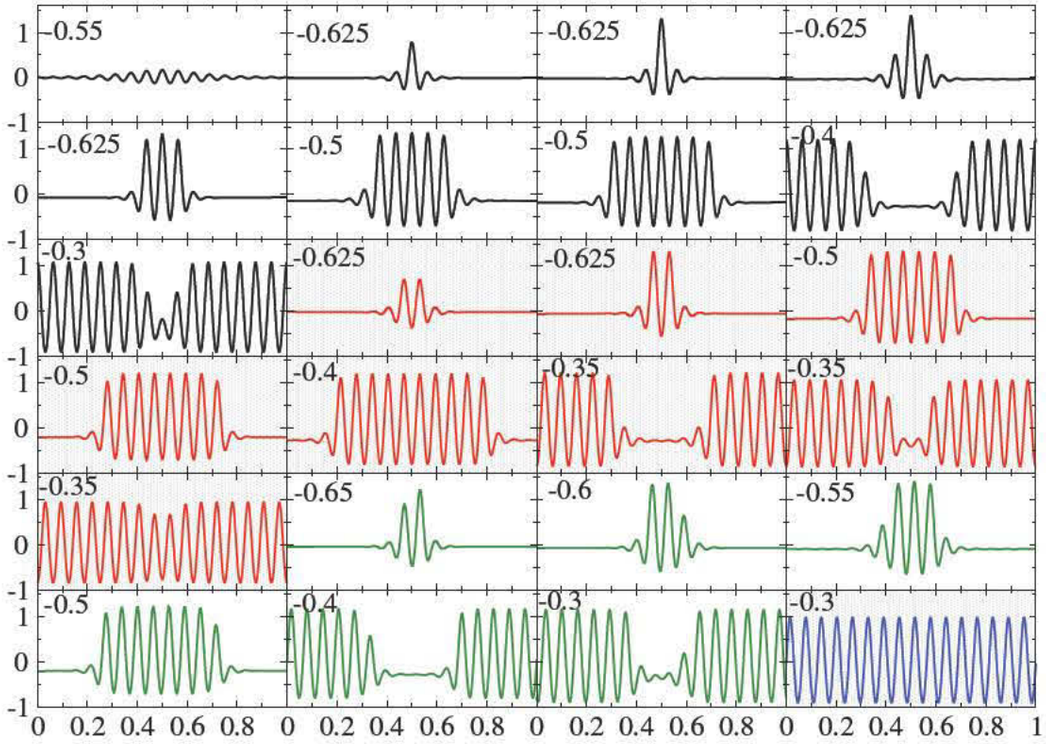
- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations	$\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$
Boundary conditions	$\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$
Initial conditions	$u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

- Several numerical packages available
  - Auto, MatCont, XPP, Coco, DDE-Biftool, Knut, etc.
  - PDE2Path, LOCA, OOMPH
  - DIY - version of the day [see Codes]

Localised steady states

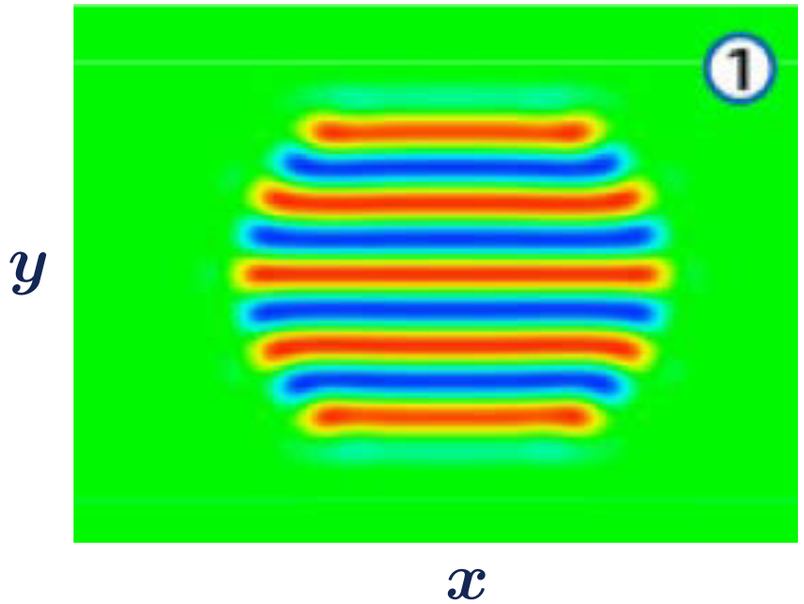
Steady states (Swift-Hohenberg Eq.)



[Thiele, Archer, Robbins, Gomez, Knobloch]

Localised steady states

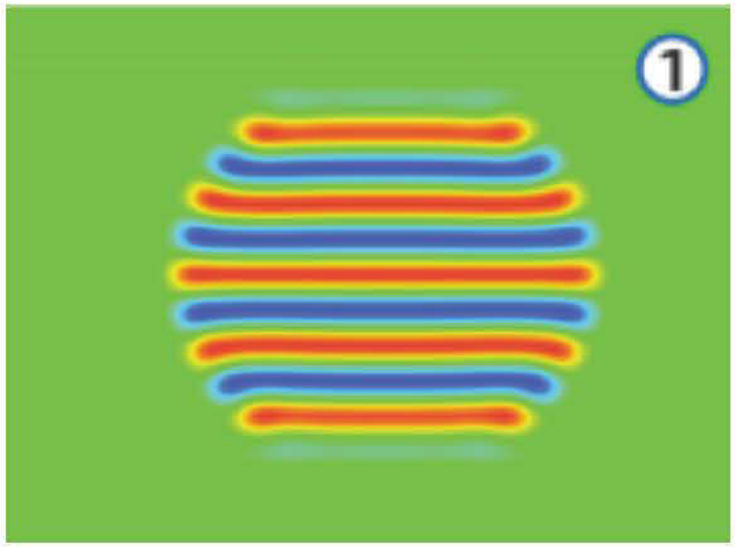
Localised steady states
(Swift-Hohenberg Eq.)



[A., Burke, Lloyd,
Sandstede, Knobloch]

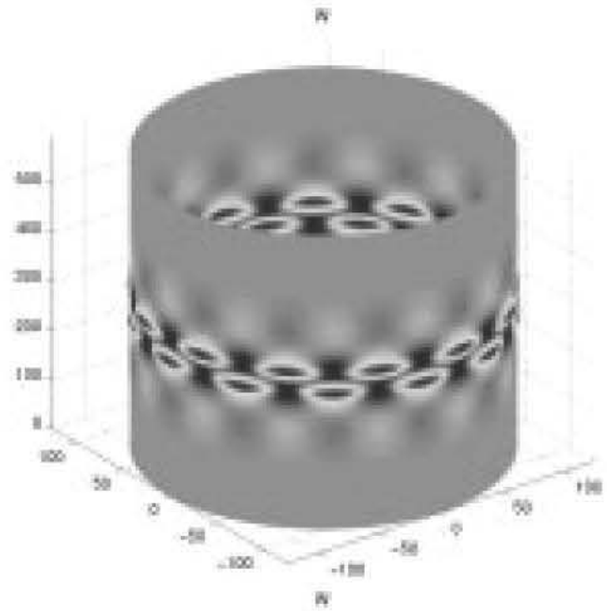
Localised steady states

Localised steady states (Swift-Hohenberg Eq.)



[A., Burke, Lloyd, Sandstede, Knobloch]

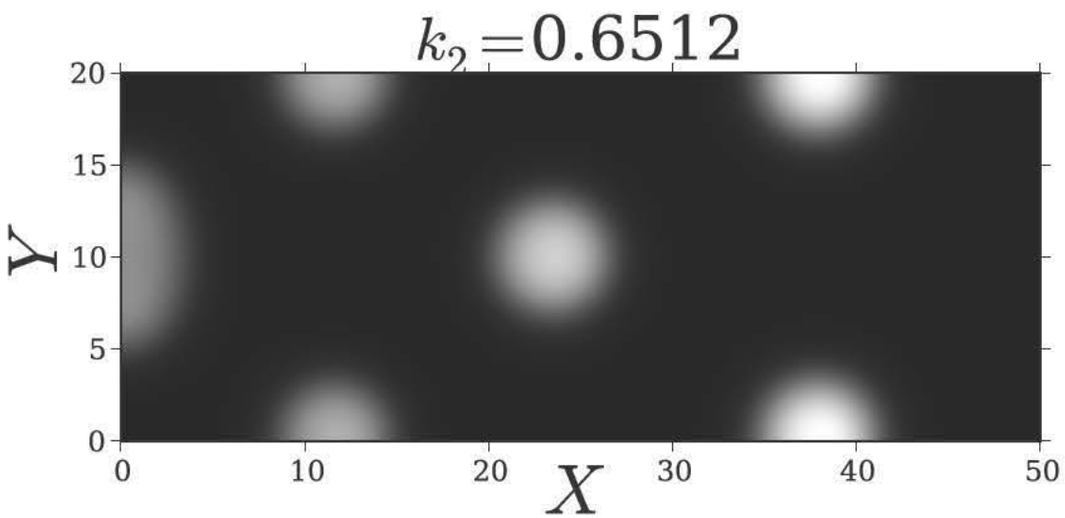
Localised steady state (von Kármán Eq.)



[Champneys, Hunt, Lord]

Steady states

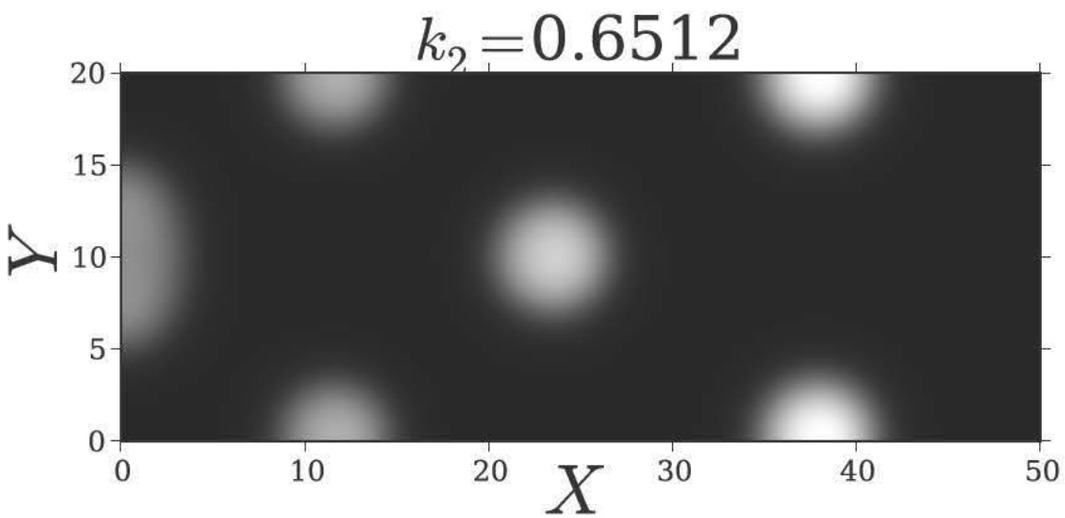
Stationary pattern
(Reaction-Diffusion-Advection system)



[Brena-Medina, A., Champneys, Ward]

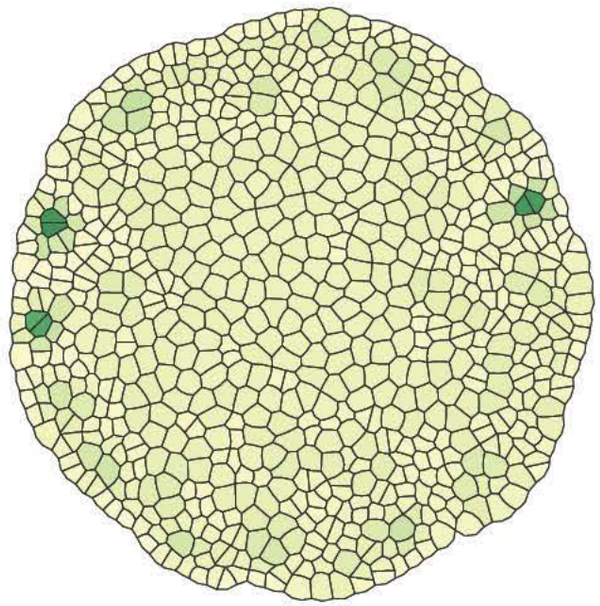
Steady states

Stationary pattern (Reaction-Diffusion-Advection system)



[Brena-Medina, A., Champneys, Ward]

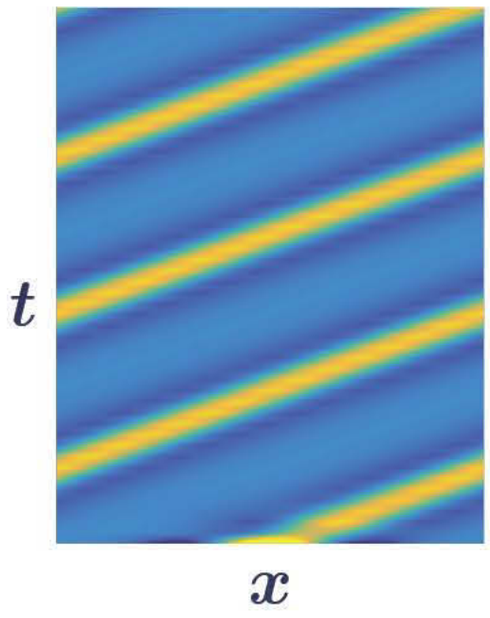
Localised auxin spots (plant root section)



[Draelants, A., Vanroose]

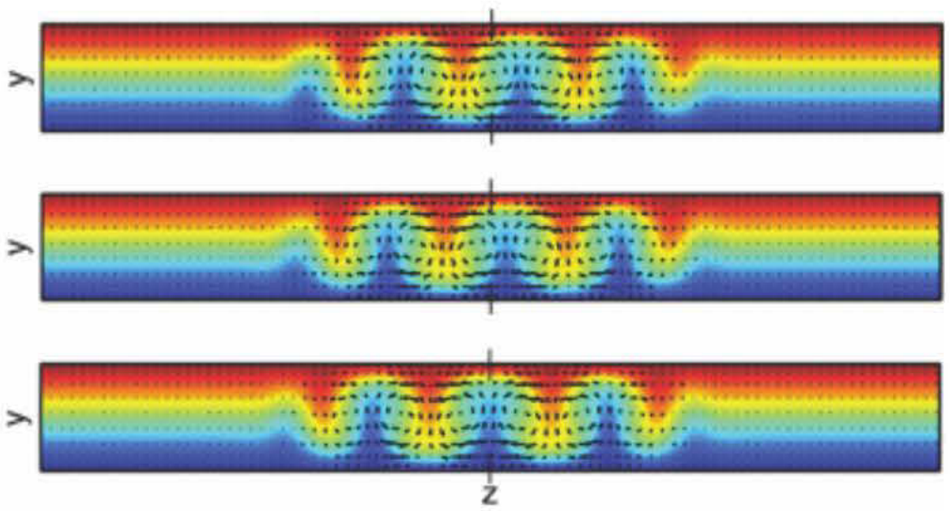
Travelling waves

Travelling wave (Neural Markov-chain)



[A., Wedgwood]

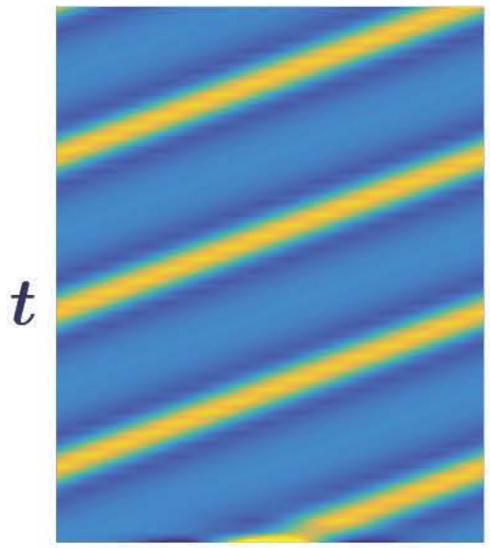
Travelling wave (Navier-Stokes)



[Schneider, Gibson, Burke]

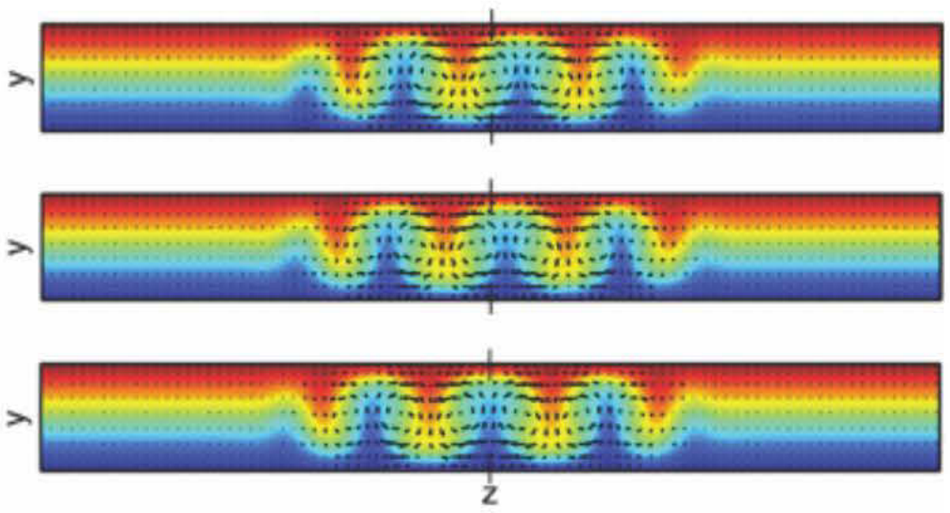
Travelling waves

Travelling wave (Neural Markov-chain)



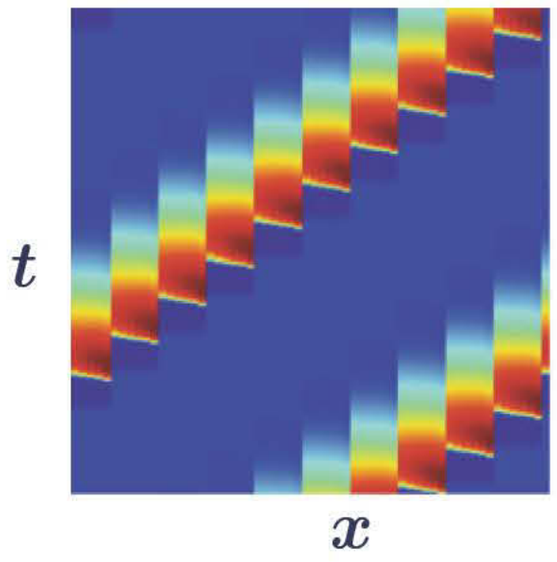
[A., Wedgwood]

Travelling wave (Navier-Stokes)



[Schneider, Gibson, Burke]

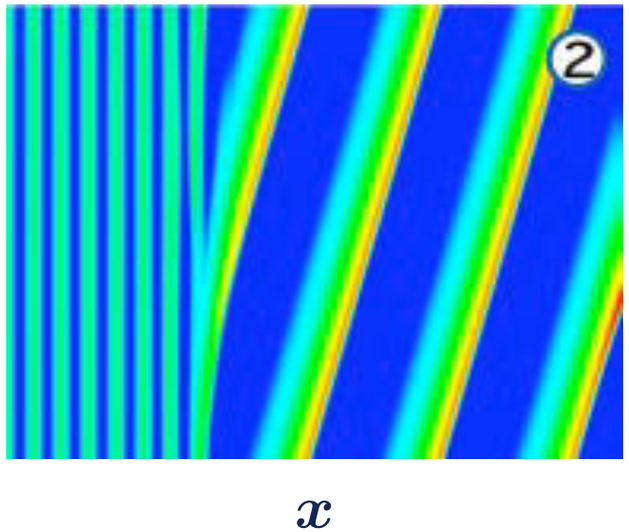
Lurching wave (Neural field model)



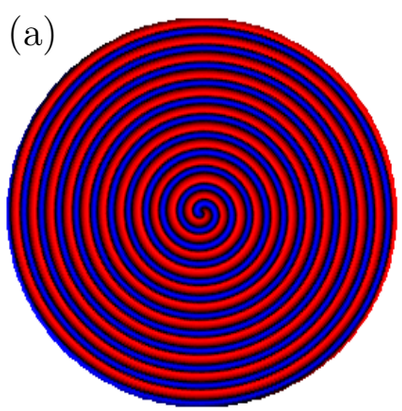
[Wasylenko, Cisternas, Laing, Kevrekidis]

Other examples of coherent structures

Defect/spiral waves
(Reaction-diffusion system)



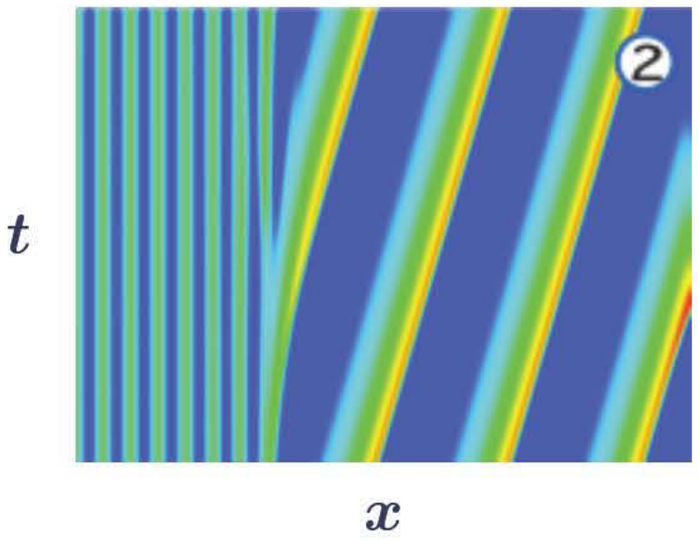
[A.]



[Wheeler, Barkley]

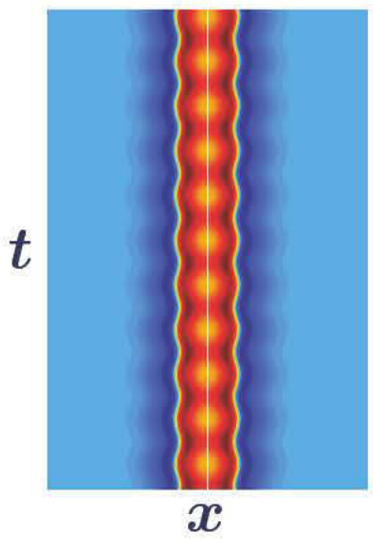
Other examples of coherent structures

Defect/spiral waves (Reaction-diffusion system)



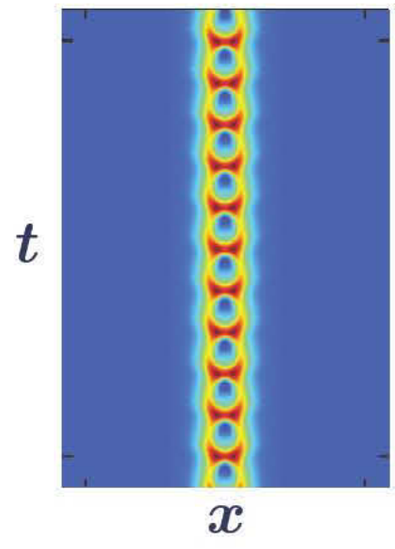
[A.]

Oscillon (Neural field)

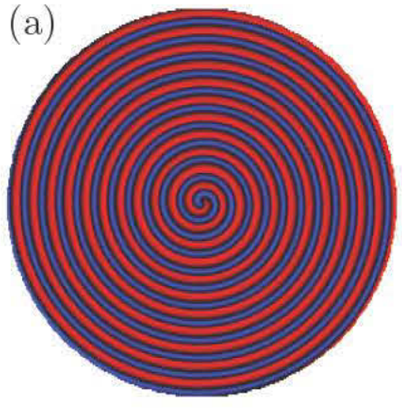


[Coombes, Schmidt, A.]

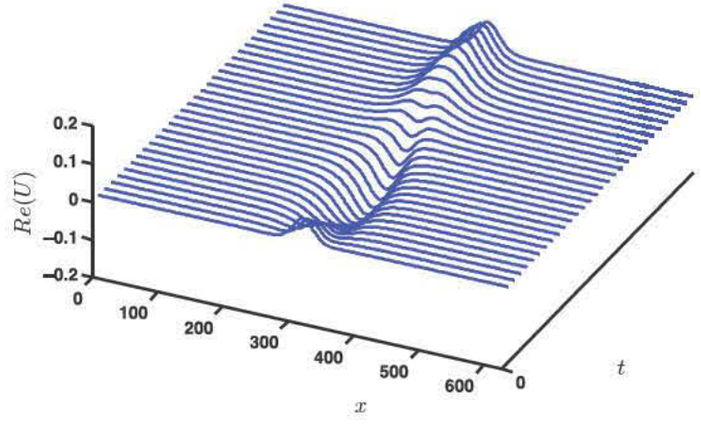
Oscillons (Complex Ginzburg-Landau Eq.)



[Burke, Yochelis, Knobloch]



[Wheeler, Barkley]



[Alnahdi, Niesen, Rucklidge]

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations $\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$

Boundary conditions $\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$

Initial conditions $u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations $\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$

Boundary conditions $\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$

Initial conditions $u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

- Several numerical packages available

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations $\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$

Boundary conditions $\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$

Initial conditions $u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

- Several numerical packages available
  - Auto, MatCont, XPP, Coco, DDE-Biftool, Knut, etc.

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations $\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$

Boundary conditions $\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$

Initial conditions $u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

- Several numerical packages available
  - Auto, MatCont, XPP, Coco, DDE-Biftool, Knut, etc.
  - PDE2Path, LOCA, OOMPH

Problem setting

- Coherent structures: solutions to nonlinear evolution equations with a particular spatial and temporal structure

- Problem setting: compute special solutions to an Initial-Boundary-Value problem

Evolution equations	$\partial_t u(x,t) = \mathcal{F}(u;p)(x,t), \qquad (x,t) \in \Omega \times (0,T]$
Boundary conditions	$\mathcal{B}(u;p)(x,t) = 0, \qquad (x,t) \in \partial\Omega \times (0,T]$
Initial conditions	$u(x,0) = u_0(x), \qquad x \in \Omega \cup \partial\Omega$

- Several numerical packages available
  - Auto, MatCont, XPP, Coco, DDE-Biftool, Knut, etc.
  - PDE2Path, LOCA, OOMPH
  - DIY - version of the day [see Codes]

Time stepping

Example of time stepping

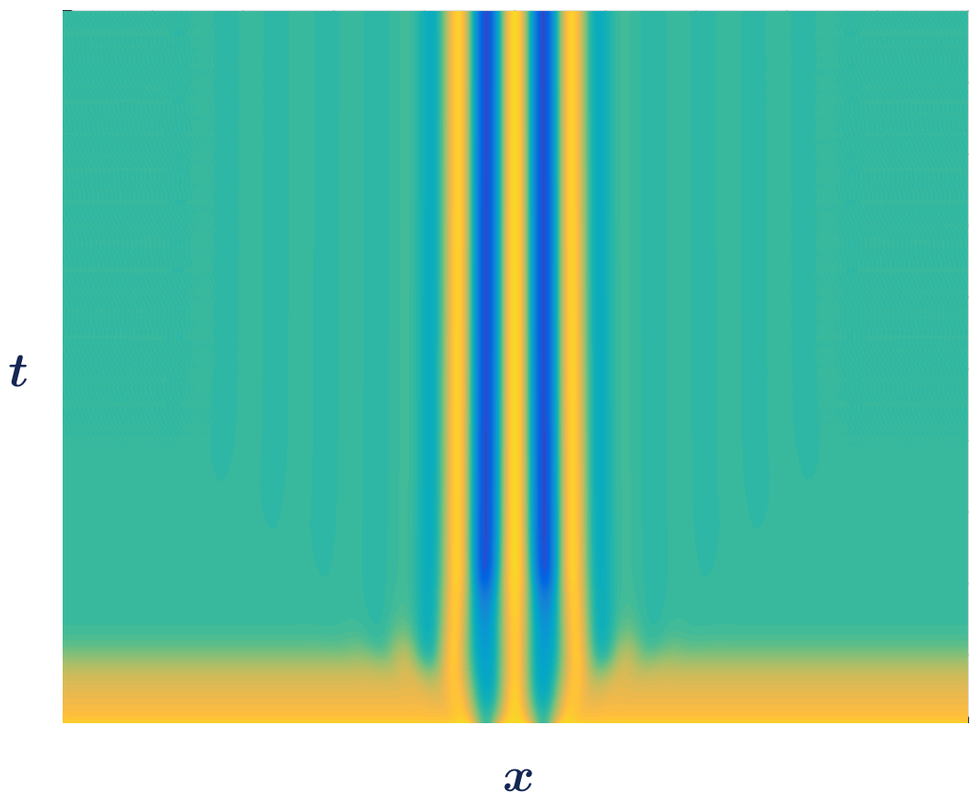
- 1D spatial domain $\Omega = [-L, L) \in \mathbb{R}$

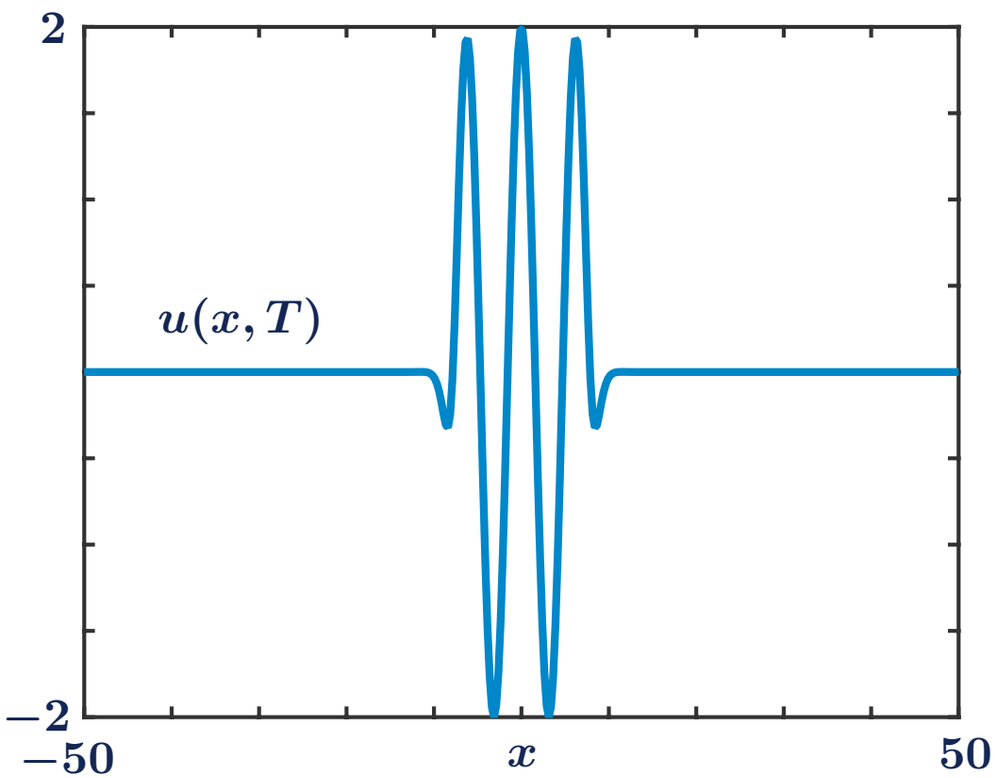
- Swift-Hohenberg equation with cubic-quintic nonlinearity

  $\partial_t u = -(1 + \partial_x^2)^2 u - \mu u + \nu u^3 - u^5$

- Periodic boundary conditions

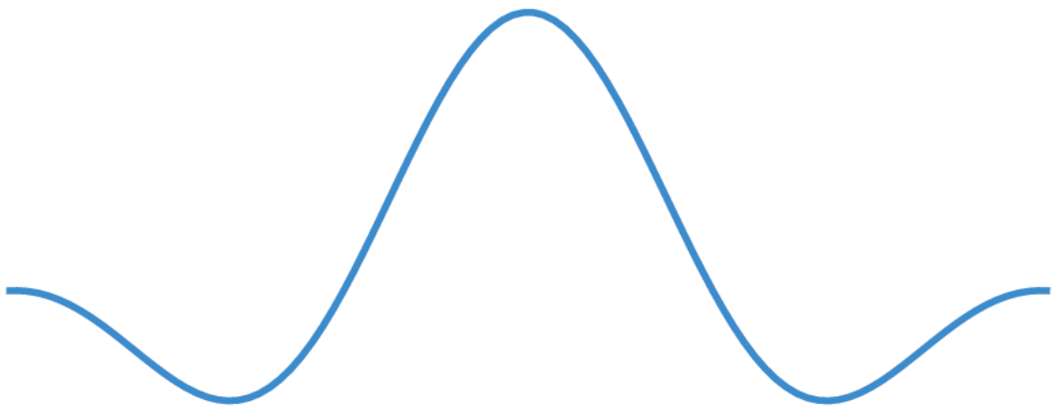
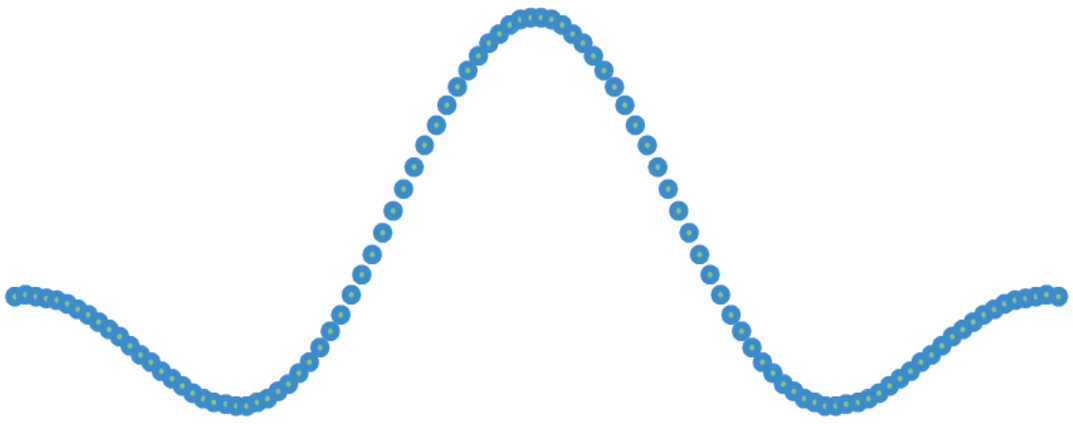
- Localised initial condition $u(x, 0) = 2 + \cos(x)[\tanh(x + 8) - \tanh(x - 8)]$





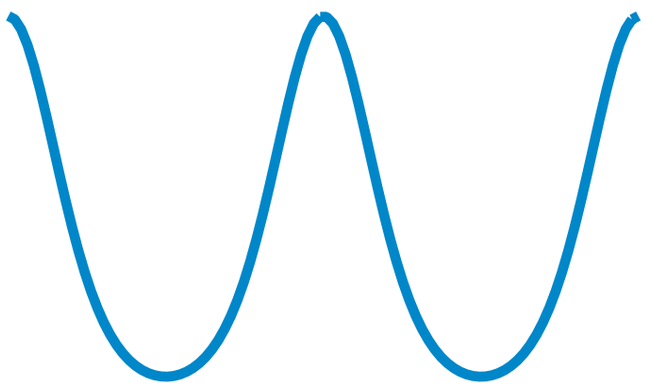
Method of lines

- A possible strategy is to use the method of lines
  1. Discretise spatial coordinate in the PDE
  2. Time step the resulting ODE

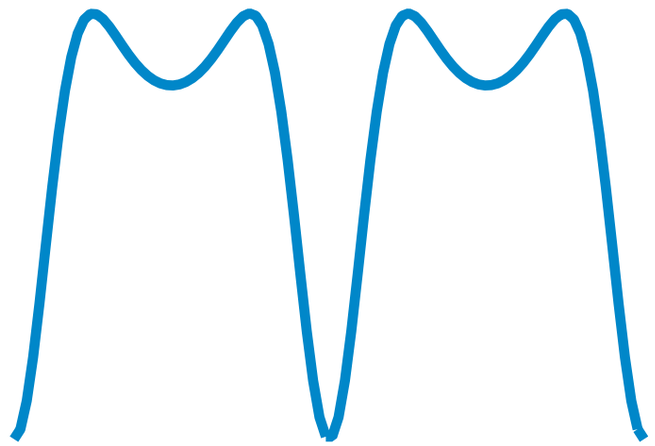
Continuum		Discrete
		
$u(x,t)$	State	$U(t), U_i(t) \approx u(x_i,t)$
$\mathcal{F}(u)$	Nonlinear operator	$F(U), F_i(U) \approx \mathcal{F}(u)(x_i)$
$\partial_t u = \mathcal{F}(u)$	Evolution equation	$\dot{U} = F(U)$

Approximation of derivatives [Trefethen]

- Differential operator $\partial_{xx} : u(x) \mapsto v(x) = u''(x)$



$$u \mapsto \partial_{xx} u$$

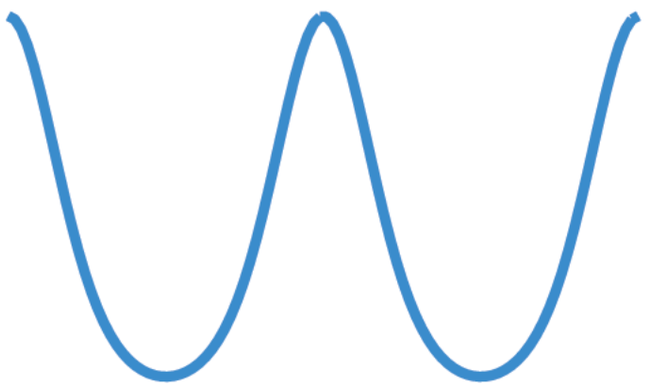


$$u(x) = \exp\left(\cos\frac{4\pi x}{L}\right)$$

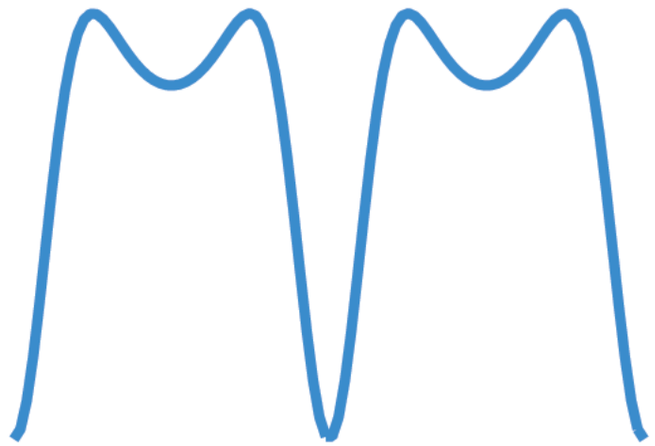
$$\partial_{xx} u(x) = \frac{16\pi^2}{L^2} e^{\cos\frac{4\pi x}{L}} \left[\sin^2\frac{4\pi x}{L} - \cos\frac{4\pi x}{L}\right]$$

Approximation of derivatives [Trefethen]

- **Differential operator** $\partial_{xx} : u(x) \mapsto v(x) = u''(x)$



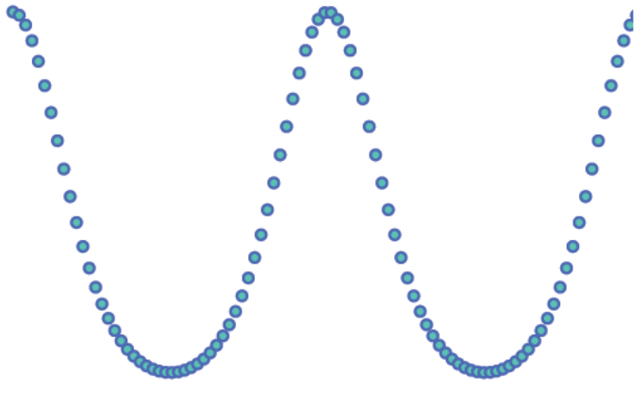
$$u \mapsto \partial_{xx} u$$



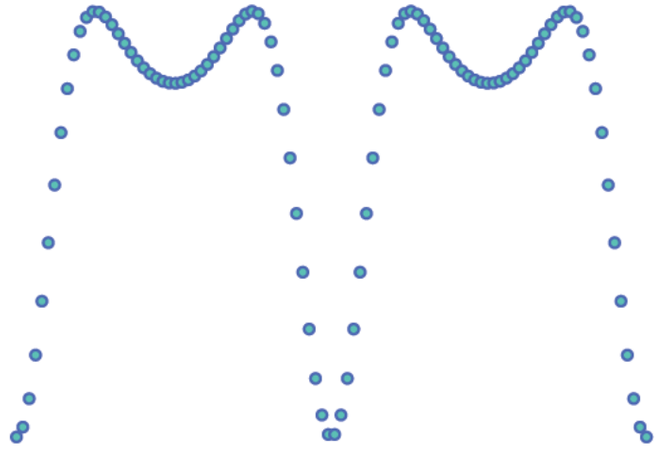
$$u(x) = \exp\left(\cos\frac{4\pi x}{L}\right)$$

$$\partial_{xx} u(x) = \frac{16\pi^2}{L^2} e^{\cos\frac{4\pi x}{L}} \left[\sin^2\frac{4\pi x}{L} - \cos\frac{4\pi x}{L}\right]$$

- **Differentiation matrix** $D_{xx} : U \mapsto V, \quad V_i = \dfrac{U_{i+1} - 2U_i + U_{i-1}}{h_x^2} \approx u''(x_i)$



$$U \mapsto D_{xx} U$$



Discretising the Swift-Hohenberg operator

- Swift-Hohenberg equation with cubic-quintic nonlinearity

$$\partial_t u = \mathcal{F}(u) \qquad \mathcal{F}(u) = -(1+\partial_x^2)^2 u - \mu u + \nu u^3 - u^5$$

- Discretised Swift-Hohenberg equation with cubic-quintic nonlinearity

$$\dot{U} = F(U) \qquad F(U) = -(I_n + D_{xx})^2 U - \mu U + \nu U^{\odot 3} - U^{\odot 5}$$

- Note the Hadamard product/power symbol (component-wise product/power)

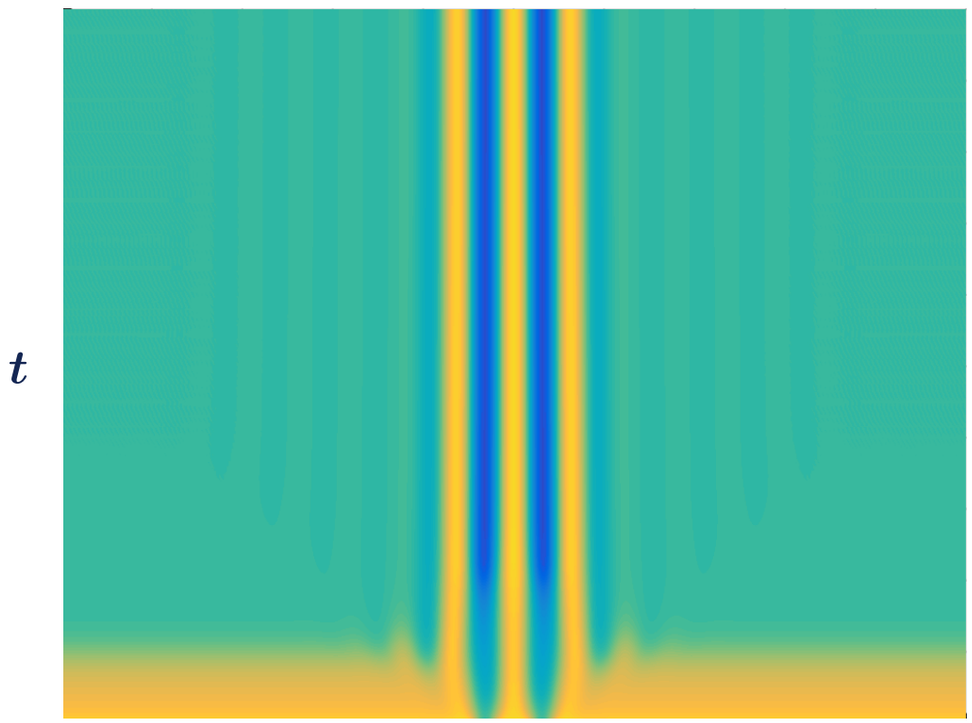
  Product between vectors $(U \odot V)_i = U_i V_i$

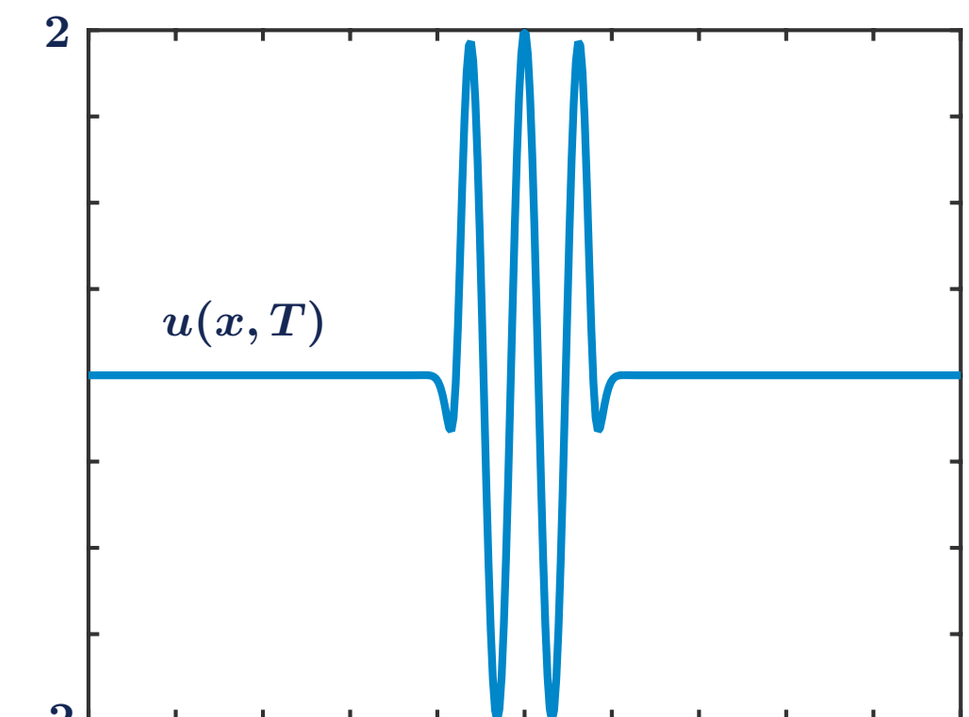
  Power of a vector $(U^{\odot 3})_i = (U \odot U \odot U)_i = U_i^3$

- The similarity between $\mathcal{F}(u)$ and $F(U)$ is striking (and useful in practice)!

Comments on time stepping

See Codes





- Discretised Swift-Hohenberg equation with cubic-quintic nonlinearity

$$\dot{U} = F(U) \qquad F(U) = -(I_n + D_{xx})^2 U - \mu U + \nu U^{\odot 3} - U^{\odot 5}$$

- I used standard stiff ODE integrators, but other choices are available/advisable
  1. Semi-implicit time-steppers
  2. Splitting methods
  3. Exponential time steppers
  4. Other problem-dependent time steppers (symplectic integrators, etc.)

Numerical continuation

Numerical continuation - introduction

- We want to study how steady states of $\partial_t u = \mathcal{F}(u; p)$ depend upon parameters

- In principle, we could change p and perform a new simulation. However
  1. There may be long transients before a new steady state is reached
  2. Only *stable* steady states are observable with time simulations
  3. For fixed p there may coexisting steady states, and which one is found depends on the initial condition

- Main idea behind numerical continuation:
  1. Start from (u_0, p_0) such that $\mathcal{F}(u_0, p_0) = 0$
  2. Generate a sequence $\{(u_k, p_k)\}_k$ such that $\mathcal{F}(u_k, p_k) = 0$

Numerical continuation - introduction

- We want to study how steady states of $\partial_t u = \mathcal{F}(u; p)$ depend upon parameters

- In principle, we could change p and perform a new simulation. However
  1. There may be long transients before a new steady state is reached
  2. Only *stable* steady states are observable with time simulations
  3. For fixed p there may coexisting steady states, and which one is found depends on the initial condition

- Main idea behind numerical continuation:
  1. Start from (u_0, p_0) such that $\mathcal{F}(u_0, p_0) = 0$
  2. Generate a sequence $\{(u_k, p_k)\}_k$ such that $\mathcal{F}(u_k, p_k) = 0$

Numerical continuation - introduction

- We want to study how steady states of $\partial_t u = \mathcal{F}(u; p)$ depend upon parameters

- In principle, we could change p and perform a new simulation. However
  1. There may be long transients before a new steady state is reached
  2. Only *stable* steady states are observable with time simulations
  3. For fixed p there may coexisting steady states, and which one is found depends on the initial condition

- Main idea behind numerical continuation:
  1. Start from (u_0, p_0) such that $\mathcal{F}(u_0, p_0) = 0$
  2. Generate a sequence $\{(u_k, p_k)\}_k$ such that $\mathcal{F}(u_k, p_k) = 0$

- Some advantages

Numerical continuation - introduction

- We want to study how steady states of $\partial_t u = \mathcal{F}(u; p)$ depend upon parameters

- In principle, we could change p and perform a new simulation. However
  1. There may be long transients before a new steady state is reached
  2. Only *stable* steady states are observable with time simulations
  3. For fixed p there may coexisting steady states, and which one is found depends on the initial condition

- Main idea behind numerical continuation:
  1. Start from (u_0, p_0) such that $\mathcal{F}(u_0, p_0) = 0$
  2. Generate a sequence $\{(u_k, p_k)\}_k$ such that $\mathcal{F}(u_k, p_k) = 0$

- Some advantages
  1. Gives access to stable and unstable states

Numerical continuation - introduction

- We want to study how steady states of $\partial_t u = \mathcal{F}(u; p)$ depend upon parameters

- In principle, we could change p and perform a new simulation. However
  1. There may be long transients before a new steady state is reached
  2. Only *stable* steady states are observable with time simulations
  3. For fixed p there may coexisting steady states, and which one is found depends on the initial condition

- Main idea behind numerical continuation:
  1. Start from (u_0, p_0) such that $\mathcal{F}(u_0, p_0) = 0$
  2. Generate a sequence $\{(u_k, p_k)\}_k$ such that $\mathcal{F}(u_k, p_k) = 0$

- Some advantages
  1. Gives access to stable and unstable states
  2. Stability calculations can be performed on-the-fly

Numerical continuation - introduction

- We want to study how steady states of $\partial_t u = \mathcal{F}(u; p)$ depend upon parameters

- In principle, we could change p and perform a new simulation. However
  1. There may be long transients before a new steady state is reached
  2. Only *stable* steady states are observable with time simulations
  3. For fixed p there may coexisting steady states, and which one is found depends on the initial condition

- Main idea behind numerical continuation:
  1. Start from (u_0, p_0) such that $\mathcal{F}(u_0, p_0) = 0$
  2. Generate a sequence $\{(u_k, p_k)\}_k$ such that $\mathcal{F}(u_k, p_k) = 0$

- Some advantages
  1. Gives access to stable and unstable states
  2. Stability calculations can be performed on-the-fly
  3. Bifurcations points can be detected

Numerical continuation - introduction

- We want to study how steady states of $\partial_t u = \mathcal{F}(u; p)$ depend upon parameters

- In principle, we could change p and perform a new simulation. However
  1. There may be long transients before a new steady state is reached
  2. Only *stable* steady states are observable with time simulations
  3. For fixed p there may coexisting steady states, and which one is found depends on the initial condition

- Main idea behind numerical continuation:
  1. Start from (u_0, p_0) such that $\mathcal{F}(u_0, p_0) = 0$
  2. Generate a sequence $\{(u_k, p_k)\}_k$ such that $\mathcal{F}(u_k, p_k) = 0$

- Some advantages
  1. Gives access to stable and unstable states
  2. Stability calculations can be performed on-the-fly
  3. Bifurcations points can be detected
  4. Solutions are collected into branches with possibly different symmetry properties

Numerical continuation - introduction

- We want to study how steady states of $\partial_t u = \mathcal{F}(u; p)$ depend upon parameters

- In principle, we could change p and perform a new simulation. However
  1. There may be long transients before a new steady state is reached
  2. Only *stable* steady states are observable with time simulations
  3. For fixed p there may coexisting steady states, and which one is found depends on the initial condition

- Main idea behind numerical continuation:
  1. Start from (u_0, p_0) such that $\mathcal{F}(u_0, p_0) = 0$
  2. Generate a sequence $\{(u_k, p_k)\}_k$ such that $\mathcal{F}(u_k, p_k) = 0$

- Some advantages
  1. Gives access to stable and unstable states
  2. Stability calculations can be performed on-the-fly
  3. Bifurcations points can be detected
  4. Solutions are collected into branches with possibly different symmetry properties
  5. Can be applied to periodic orbits, travelling waves, spiral waves

Continuation of steady states in ODEs

See Codes

- **Evolution equation**

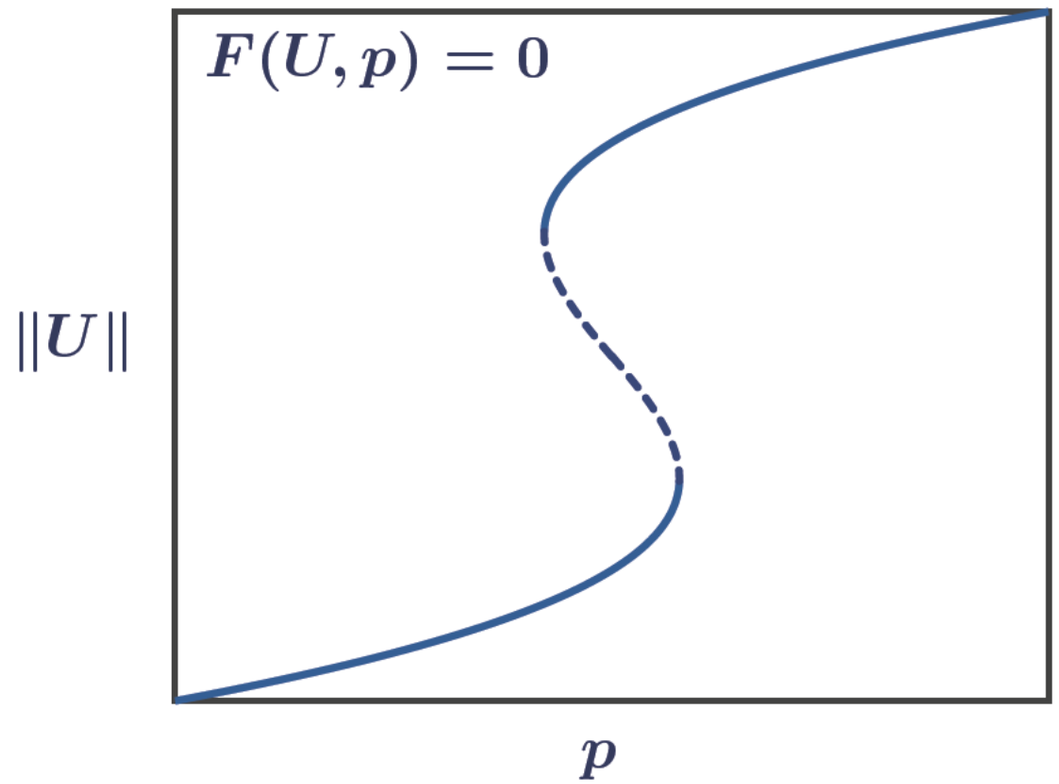
$$\dot{U} = F(U; p)$$

- **Branch of steady states**

$$F(U(s), p(s)) = 0$$

- **Stability of a steady state** (U_*, p_*)

$$D_U F(U_*, p_*)\tilde{U} = \lambda \tilde{U}$$



Continuation of steady states in ODEs

See Codes

- **Evolution equation**

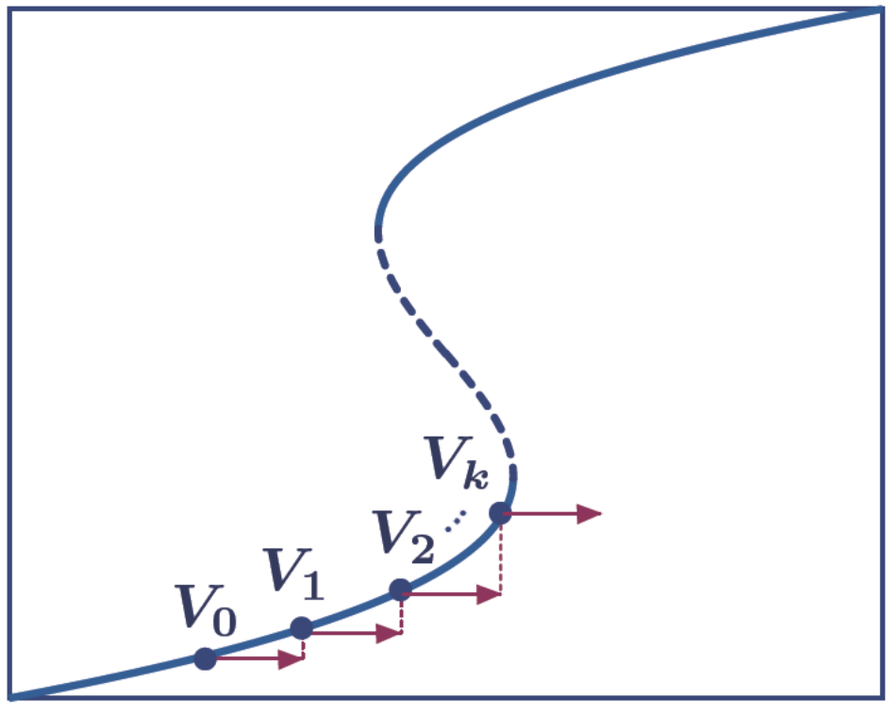
$$\dot{U} = F(U; p)$$

- **Branch of steady states**

$$F(U(s), p(s)) = 0$$

- **Stability of a steady state** (U_*, p_*)

$$D_U F(U_*, p_*)\tilde{U} = \lambda \tilde{U}$$



- **Assume knowledge of** $V_0 = (U_0, p_0) \in \mathbb{R}^{n+1}$ **such that** $F(U_0, p_0) = 0$

1. Fix h
2. For each $k > 0$ find (U, p) such that

$$\begin{aligned} F(U, p) &= 0 \\ p - p^{k-1} - h &= 0 \end{aligned} \qquad \text{or} \qquad G(V; p^{k-1}, h) = 0$$

Continuation of steady states in ODEs

See Codes

- **Evolution equation**

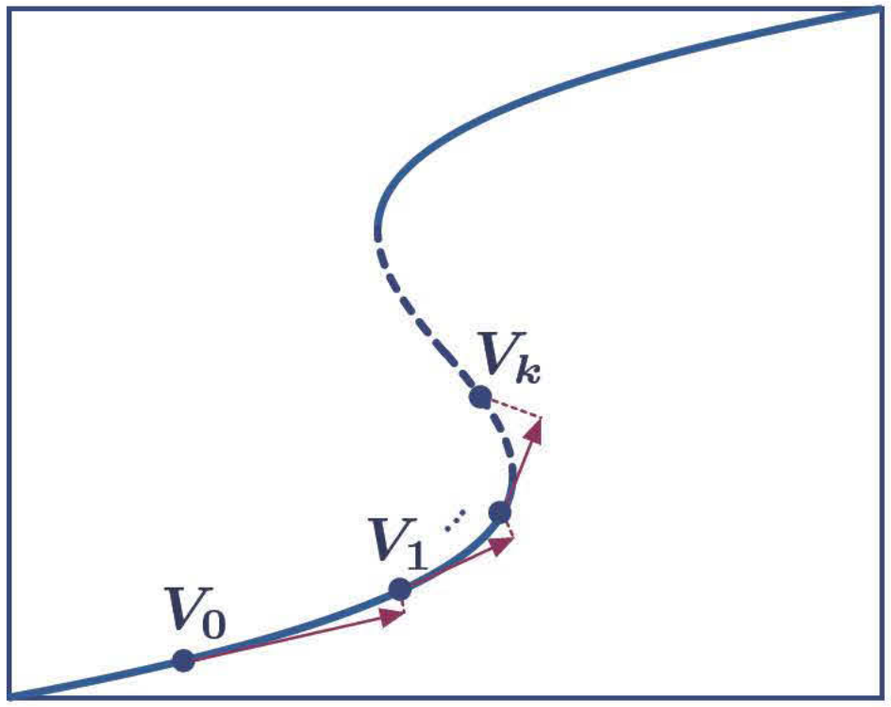
$$\dot{U} = F(U; p)$$

- **Branch of steady states**

$$F(U(s), p(s)) = 0$$

- **Stability of a steady state** (U_*, p_*)

$$D_U F(U_*, p_*)\tilde{U} = \lambda \tilde{U}$$



- **Assume knowledge of** $V_0 = (U_0, p_0) \in \mathbb{R}^{n+1}$ **such that** $F(U_0, p_0) = 0$
  1. Fix h
  2. For each $k > 0$
     A. Predict along the tangent $\tilde{V}_k = V_k + hT_k$
     B. Correct in the normal direction: find V such that

$$\begin{aligned} F(V) &= 0 \\ \langle V - \tilde{V}_k, hV_k \rangle &= 0 \end{aligned} \qquad \text{or} \qquad G(V; V_k, \tilde{V}_k, h) = 0$$

Newton's method

- At each continuation step, we solve the problem $G(V) = 0$
- Newton's method with initial guess V_0 : repeat until convergence
  1. Solve $D_V G(V_l)\tilde{V}_l = -G(V_l)$ for $\tilde{V}_l$
  2. Compute $V_{l+1} = V_l + \tilde{V}_l$
- At each continuation step we need to solve *several* linear problems
- In any continuation package, the functions $G(V)$, $D_V G(V)$ are constructed automatically from $F(U,p)$, $D_U F(U,p)$, $D_p F(U,p)$, which are provided by the user
- For PDEs, most computational time is spent in function evaluations and solutions of large linear problems

Application to the Swift-Hohenberg operator

- Fix μ, ν. Find u such that

$$-(1+\partial_x^2)^2 u - \mu u + \nu u^3 - u^5 = 0$$
$$u(L,t) - u(-L,t) = 0$$

or

$$\mathcal{F}(u) = 0$$
$$\mathcal{B}(u) = 0$$

Application to the Swift-Hohenberg operator

- Swift-Hohenberg operator and its Fréchet Derivative at u

$$\mathcal{F}(u) = -(1+\partial_x^2)^2 u - \mu u + \nu u^3 - u^5$$

$$\partial_u \mathcal{F}(u) = -(1+\partial_x^2)^2 - \mu + 3\nu u^2 - 5u^4$$

Application to the Swift-Hohenberg operator

- Swift-Hohenberg operator and its Fréchet Derivative at u

$$\mathcal{F}(u) = -(1+\partial_x^2)^2 u - \mu u + \nu u^3 - u^5$$
$$\partial_u \mathcal{F}(u) = -(1+\partial_x^2)^2 - \mu + 3\nu u^2 - 5u^4$$

- Discretised Swift-Hohenberg equation and its Jacobian at U

$$F(U) = -(I_n + D_{xx})^2 U - \mu U + \nu U^{\odot 3} - U^{\odot 5}$$
$$D_U F(U) = -(I_n + D_{xx})^2 - \mu I_n + \mathrm{diag}(3\nu U^{\odot 2} - 5U^{\odot 4})$$

Application to the Swift-Hohenberg operator

- Swift-Hohenberg operator and its Fréchet Derivative at u

$$\mathcal{F}(u) = -(1+\partial_x^2)^2 u - \mu u + \nu u^3 - u^5$$

$$\partial_u \mathcal{F}(u) = -(1+\partial_x^2)^2 - \mu + 3\nu u^2 - 5u^4$$

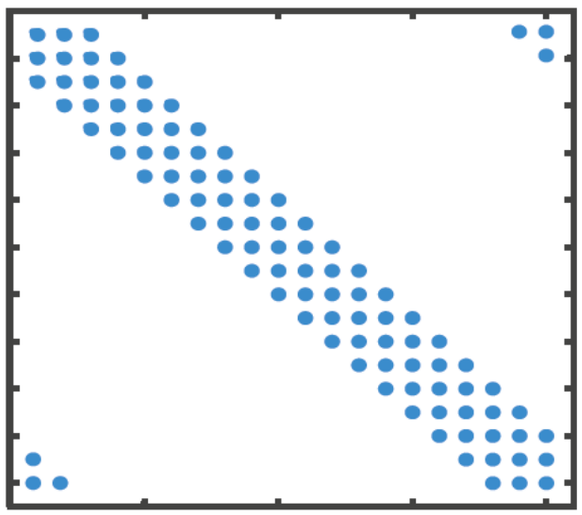
- Discretised Swift-Hohenberg equation and its Jacobian at U

$$F(U) = -(I_n + D_{xx})^2 U - \mu U + \nu U^{\odot 3} - U^{\odot 5}$$

$$D_U F(U) = -(I_n + D_{xx})^2 - \mu I_n + \mathrm{diag}(3\nu U^{\odot 2} - 5U^{\odot 4})$$

- Sparsity pattern example $n = 20$

$$D_U F(U) =$$



$$-(I_n + D_{xx})^2$$

Fixed!

Application to the Swift-Hohenberg operator

- Swift-Hohenberg operator and its Fréchet Derivative at u

$$\mathcal{F}(u) = -(1+\partial_x^2)^2 u - \mu u + \nu u^3 - u^5$$

$$\partial_u \mathcal{F}(u) = -(1+\partial_x^2)^2 - \mu + 3\nu u^2 - 5u^4$$

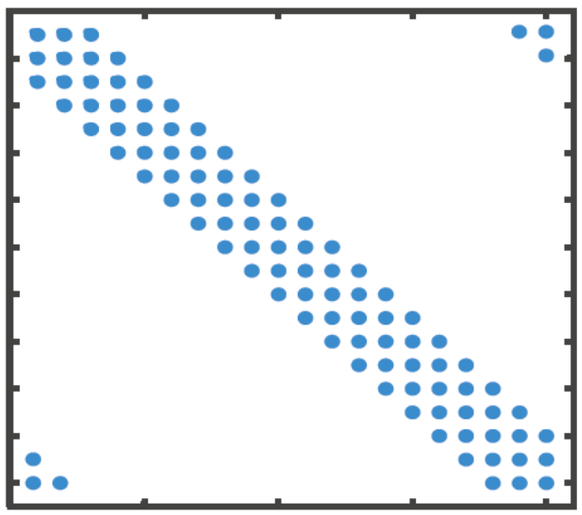
- Discretised Swift-Hohenberg equation and its Jacobian at U

$$F(U) = -(I_n + D_{xx})^2 U - \mu U + \nu U^{\odot 3} - U^{\odot 5}$$

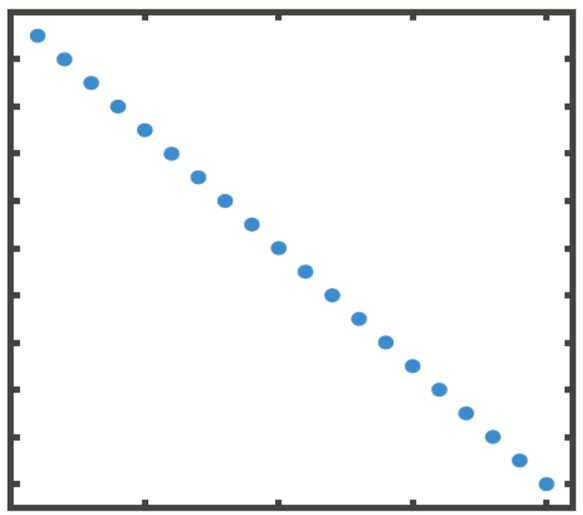
$$D_U F(U) = -(I_n + D_{xx})^2 - \mu I_n + \mathrm{diag}(3\nu U^{\odot 2} - 5U^{\odot 4})$$

- Sparsity pattern example $n = 20$

$$D_U F(U) =$$



$$+$$



$-(I_n + D_{xx})^2$

Fixed!

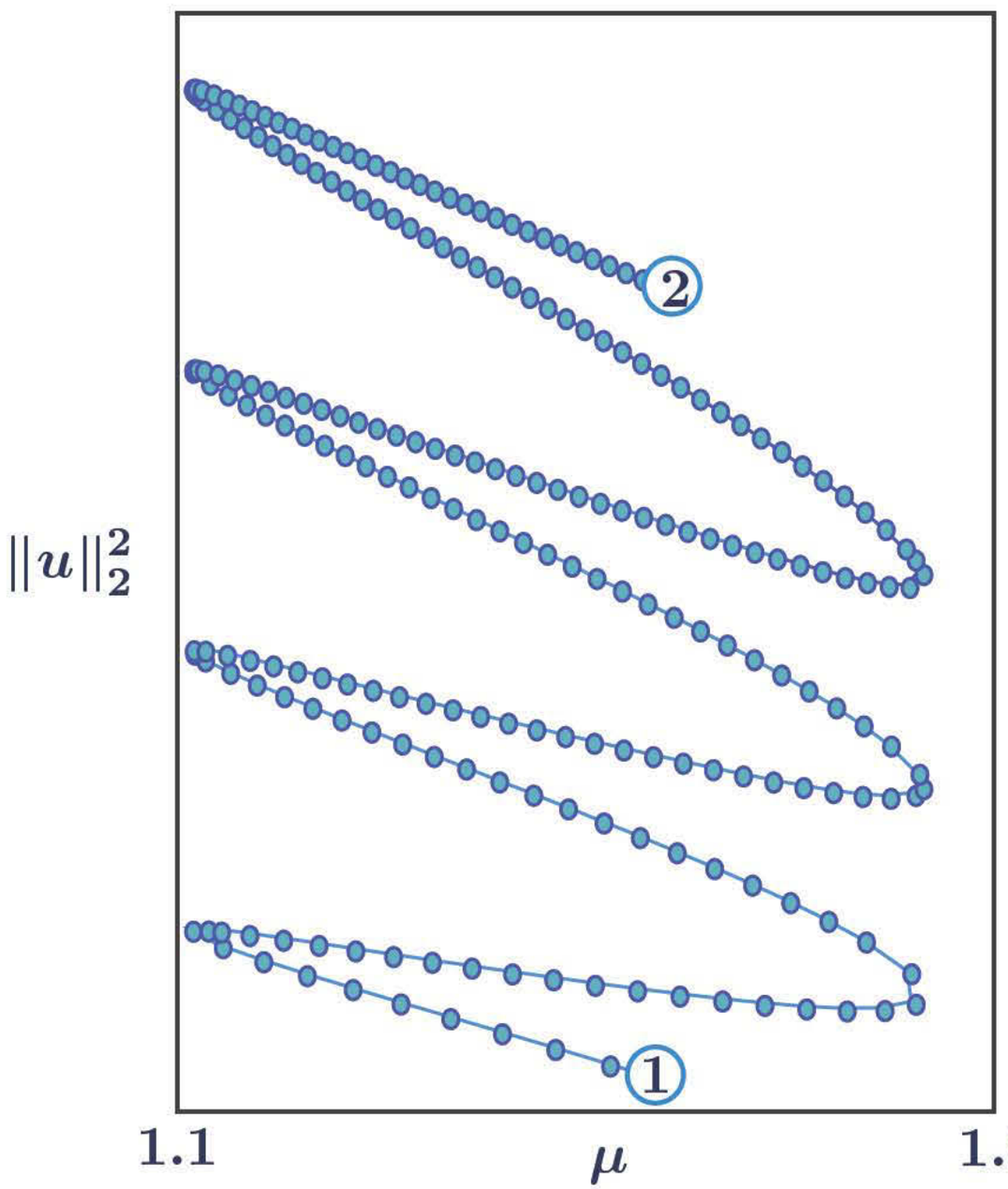
$-\mu I_n + \mathrm{diag}(3U^{\odot 2} - 5U^{\odot 4})$

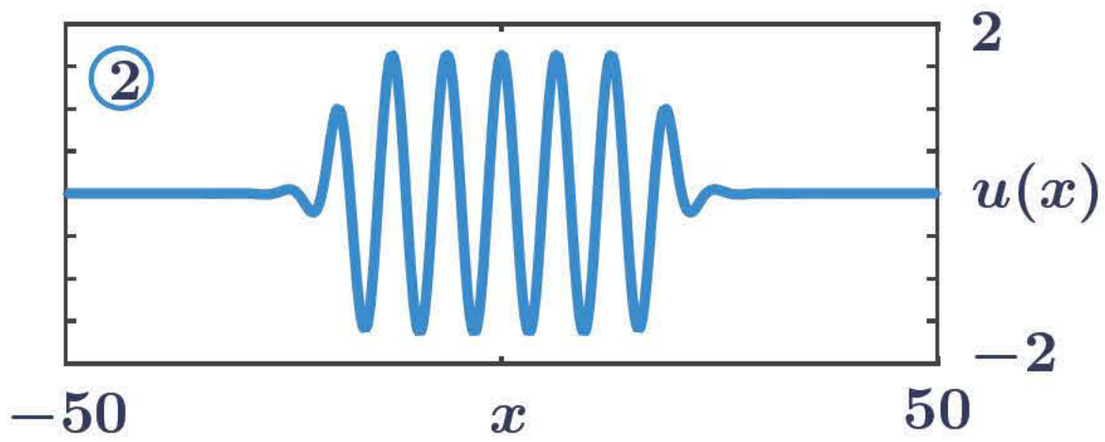
Variable

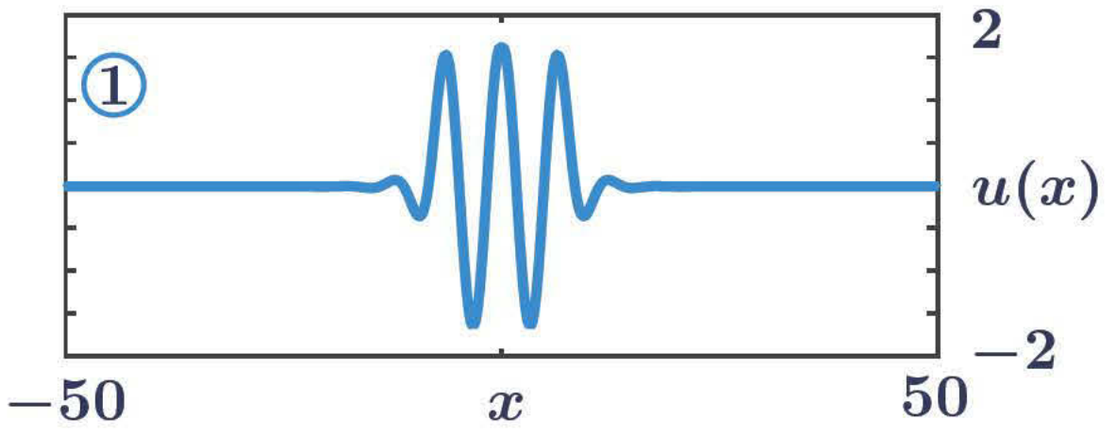
Example of numerical continuation

See Codes

$$\Omega = (-50, 50),\ \nu = 3,\ n = 400$$







Linear stability

Linear stability of steady states

- Equilibrium u_* of the evolution equation

$$\partial_t u = \mathcal{F}(u), \qquad \mathcal{F}(u_*) = 0$$

- Study the evolution of *small* perturbations to u_*

$$u(x,t) = u_*(x) + \varepsilon \tilde{u}(x,t), \qquad 0 < \varepsilon \ll 1, \quad \|\tilde{u}\| = \mathcal{O}(1)$$

- At $\mathcal{O}(\varepsilon)$, the perturbations evolve according to

$$\partial_t \tilde{u} = \partial_u \mathcal{F}(u_*)\tilde{u} := \mathcal{L}(u_*)\tilde{u}$$

- The ansatz $\tilde{u}(x,t) = e^{\lambda t}\psi(x)$ leads to the eigenvalue problem

$$\lambda \psi = \mathcal{L}(u_*)\psi$$

- The state u_* is linearly stable if $\mathrm{Spec}\left[\mathcal{L}(u_*)\right]$ is contained in the left half of the complex plane

Numerical calculations

See Codes

- Continuum problem

$$\lambda\psi = \mathcal{L}(u_*)\psi \qquad \mathcal{L}(u_*) = -(1+\partial_x^2)^2 - \mu + 3\nu u_*^2 - 5u_*^4$$

- Discrete problem

$$\lambda\Psi = L(U_*)\Psi \qquad L(U) = -(I_n + D_{xx})^2 - \mu I_n + \mathrm{diag}(3U^{\odot 2} - 5U^{\odot 4})$$

A function which computes L has already been used in the Newton iterations!

Numerical calculations

See Codes

- Continuum problem

$$\lambda\psi = \mathcal{L}(u_*)\psi \qquad \mathcal{L}(u_*) = -(1+\partial_x^2)^2 - \mu + 3\nu u_*^2 - 5u_*^4$$

- Discrete problem

$$\lambda\Psi = L(U_*)\Psi \qquad L(U) = -(I_n + D_{xx})^2 - \mu I_n + \mathrm{diag}(3U^{\odot 2} - 5U^{\odot 4})$$

A function which computes L has already been used in the Newton iterations!

- Numerical computation of eigenvalues

Numerical calculations

See Codes

- Continuum problem

$$\lambda\psi = \mathcal{L}(u_*)\psi \qquad \mathcal{L}(u_*) = -(1+\partial_x^2)^2 - \mu + 3\nu u_*^2 - 5u_*^4$$

- Discrete problem

$$\lambda\Psi = L(U_*)\Psi \qquad L(U) = -(I_n + D_{xx})^2 - \mu I_n + \mathrm{diag}(3U^{\odot 2} - 5U^{\odot 4})$$

A function which computes L has already been used in the Newton iterations!

- Numerical computation of eigenvalues
  1. Standard factorisations for dense matrices (compute full spectrum)

Numerical calculations

See Codes

- Continuum problem

$$\lambda\psi = \mathcal{L}(u_*)\psi \qquad \mathcal{L}(u_*) = -(1+\partial_x^2)^2 - \mu + 3\nu u_*^2 - 5u_*^4$$

- Discrete problem

$$\lambda\Psi = L(U_*)\Psi \qquad L(U) = -(I_n + D_{xx})^2 - \mu I_n + \mathrm{diag}(3U^{\odot 2} - 5U^{\odot 4})$$

A function which computes L has already been used in the Newton iterations!

- Numerical computation of eigenvalues
  1. Standard factorisations for dense matrices (compute full spectrum)
  2. Arnoldi iterations for sparse matrices (compute a few eigenvalues)

Numerical calculations

See Codes

- Continuum problem

$$\lambda\psi = \mathcal{L}(u_*)\psi \qquad \mathcal{L}(u_*) = -(1+\partial_x^2)^2 - \mu + 3\nu u_*^2 - 5u_*^4$$

- Discrete problem

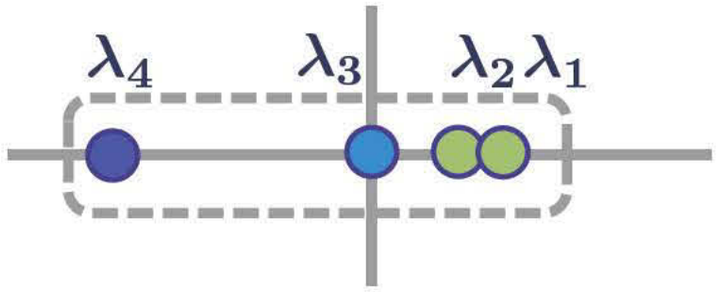
$$\lambda\Psi = L(U_*)\Psi \qquad L(U) = -(I_n + D_{xx})^2 - \mu I_n + \mathrm{diag}(3U^{\odot 2} - 5U^{\odot 4})$$

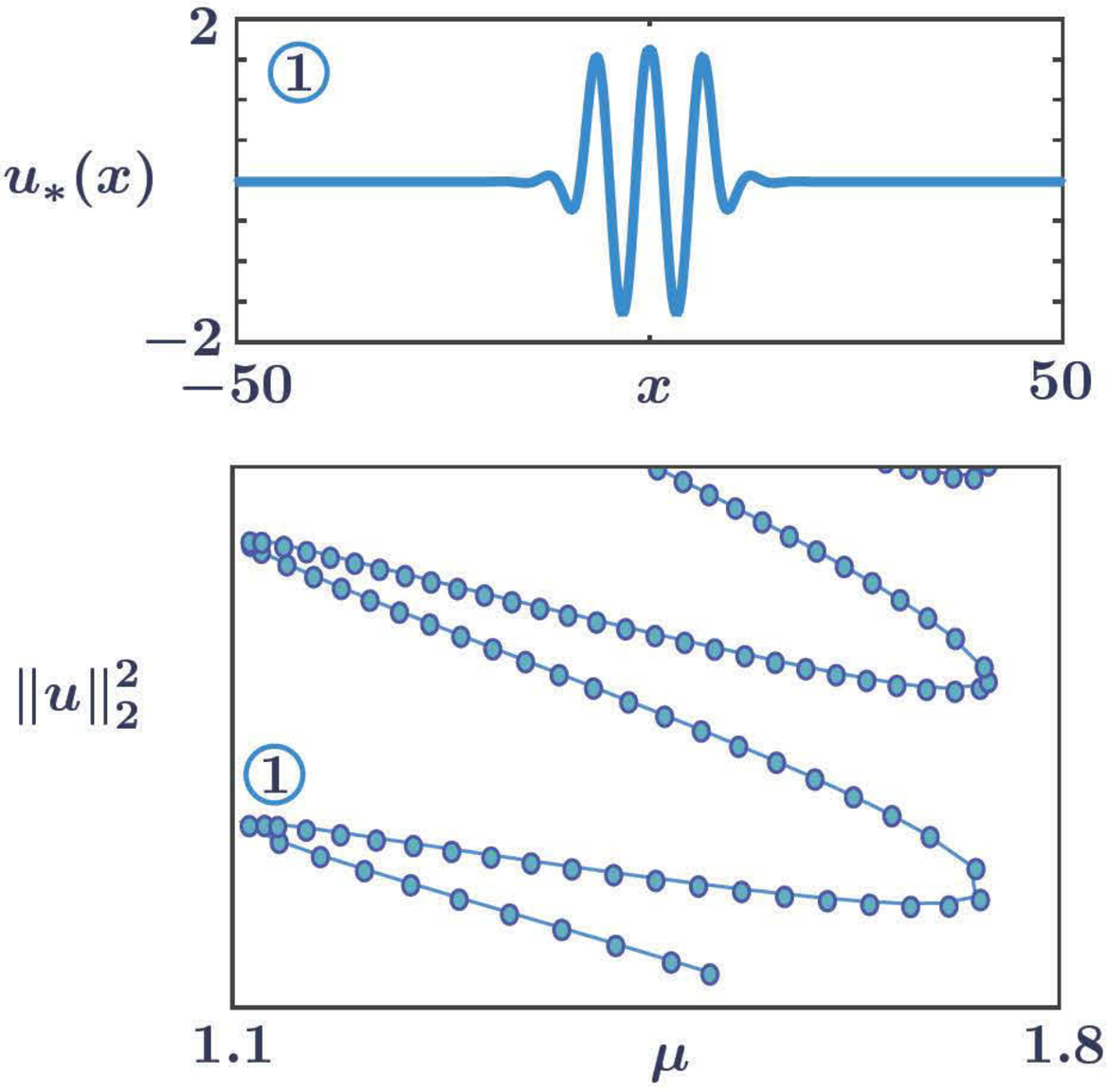
A function which computes L has already been used in the Newton iterations!

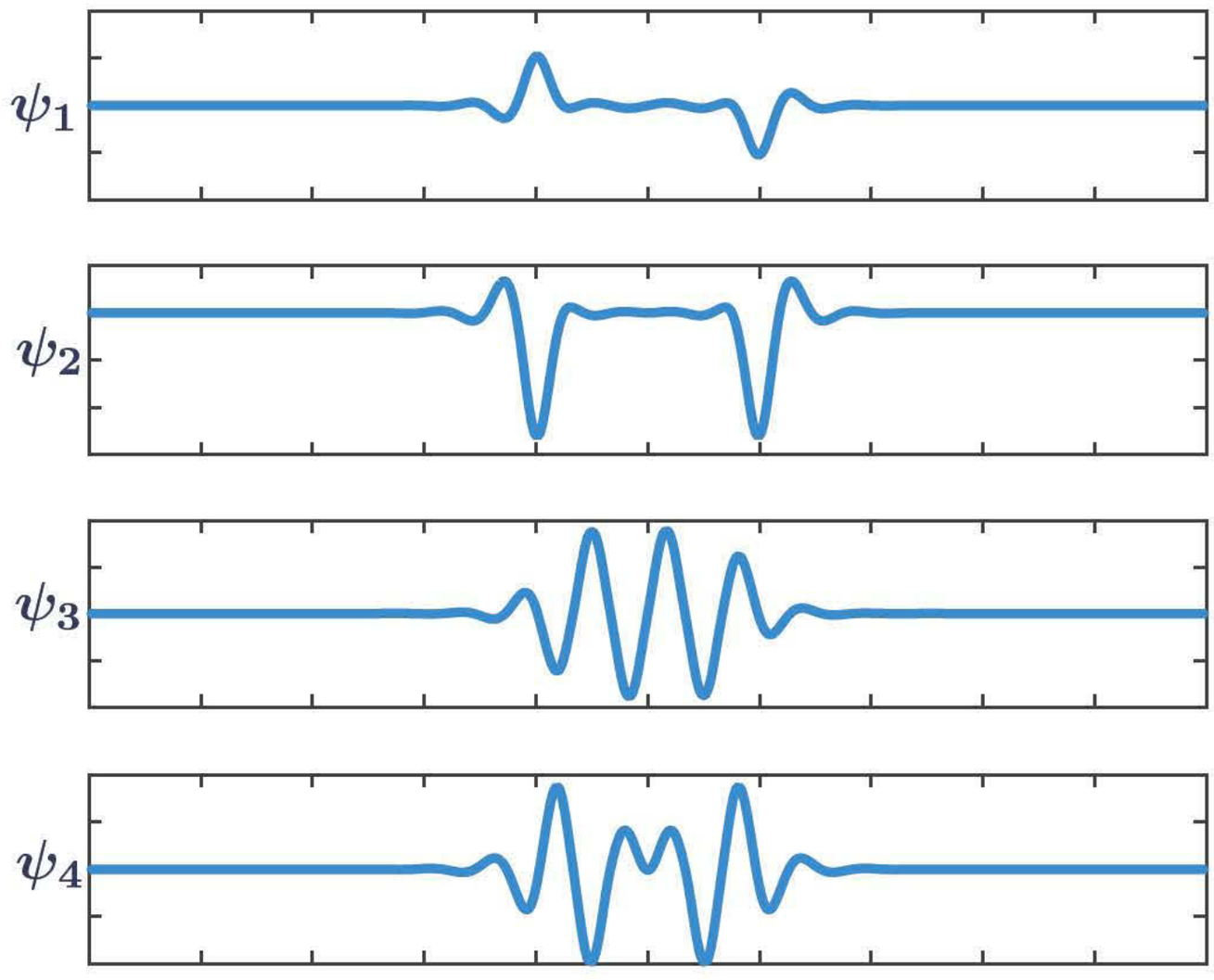
- Numerical computation of eigenvalues
  1. Standard factorisations for dense matrices (compute full spectrum)
  2. Arnoldi iterations for sparse matrices (compute a few eigenvalues)

- Computations of eigenvalues
  1. Can be done on-the-fly, during continuation
  2. May be performed as a post-processing step

Results for a profile close to a saddle-node







Translation invariance

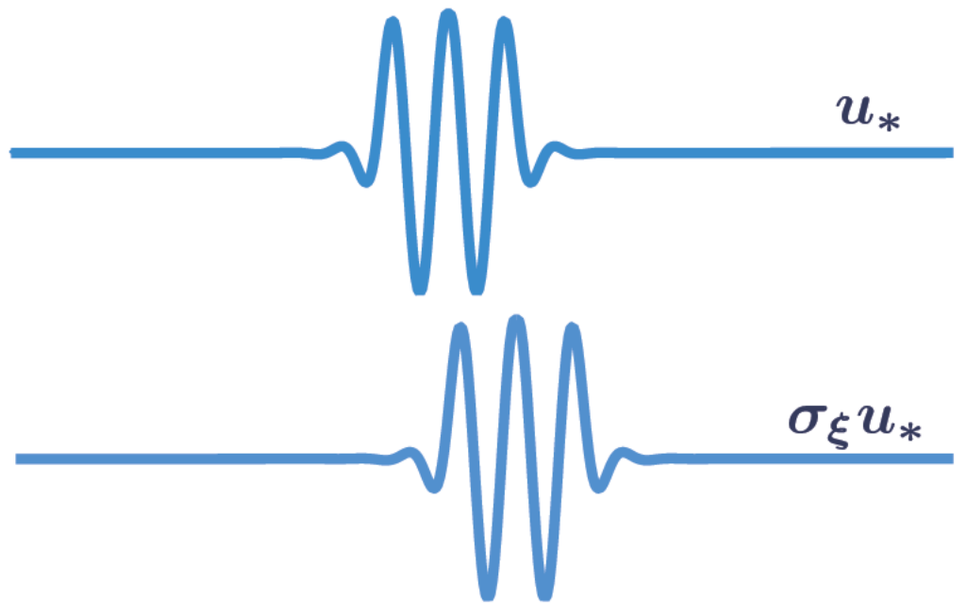
Translation invariance

- SH equation with periodic boundary conditions

$$\partial_t u = \mathcal{F}(u)$$

- Consider spatial translations via the action

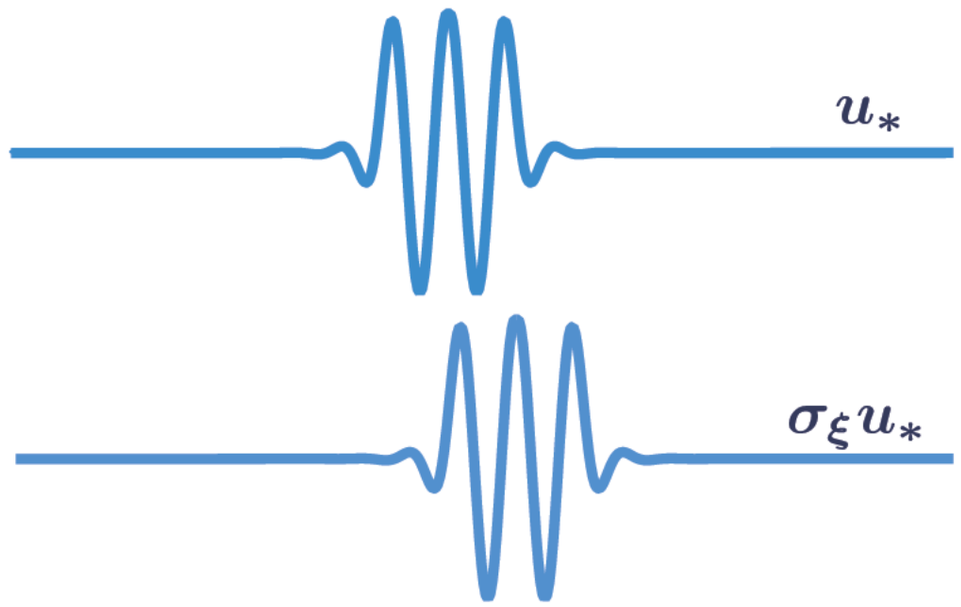
$$\sigma_\xi : v(\cdot) \mapsto v(\cdot + \xi), \quad \xi \in \mathbb{R}$$



Translation invariance

- SH equation with periodic boundary conditions

$$\partial_t u = \mathcal{F}(u)$$



- Consider spatial translations via the action

$$\sigma_\xi : v(\cdot) \mapsto v(\cdot + \xi), \quad \xi \in \mathbb{R}$$

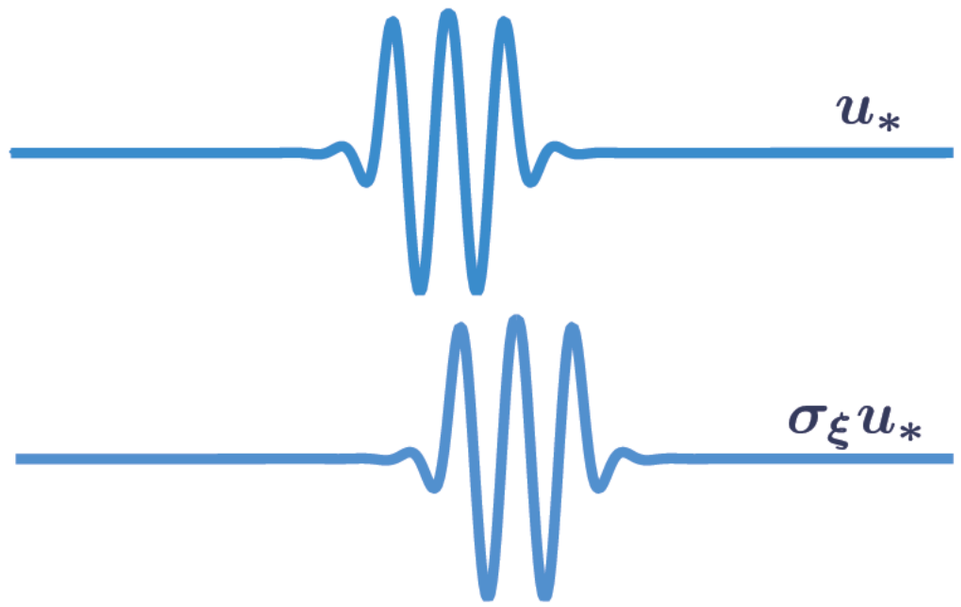
- The SH operator is equivariant under this action, that is, $\sigma_\xi \mathcal{F}(u) = \mathcal{F}(\sigma_\xi u)$

$$\begin{aligned}
\sigma_\xi \mathcal{F}(u) &= \sigma_\xi \big[-(1+\partial_x^2)^2 u - \mu u + \nu u^3 - u^5 \big] \\
&= -(1+\partial_x^2)^2 \sigma_\xi u - \mu \sigma_\xi u + \nu (\sigma_\xi u)^3 - (\sigma_\xi u)^5 \\
&= \mathcal{F}(\sigma_\xi u)
\end{aligned}$$

Translation invariance

- SH equation with periodic boundary conditions

$$\partial_t u = \mathcal{F}(u)$$



- Consider spatial translations via the action

$$\sigma_\xi : v(\cdot) \mapsto v(\cdot + \xi), \quad \xi \in \mathbb{R}$$

- The SH operator is equivariant under this action, that is, $\sigma_\xi \mathcal{F}(u) = \mathcal{F}(\sigma_\xi u)$

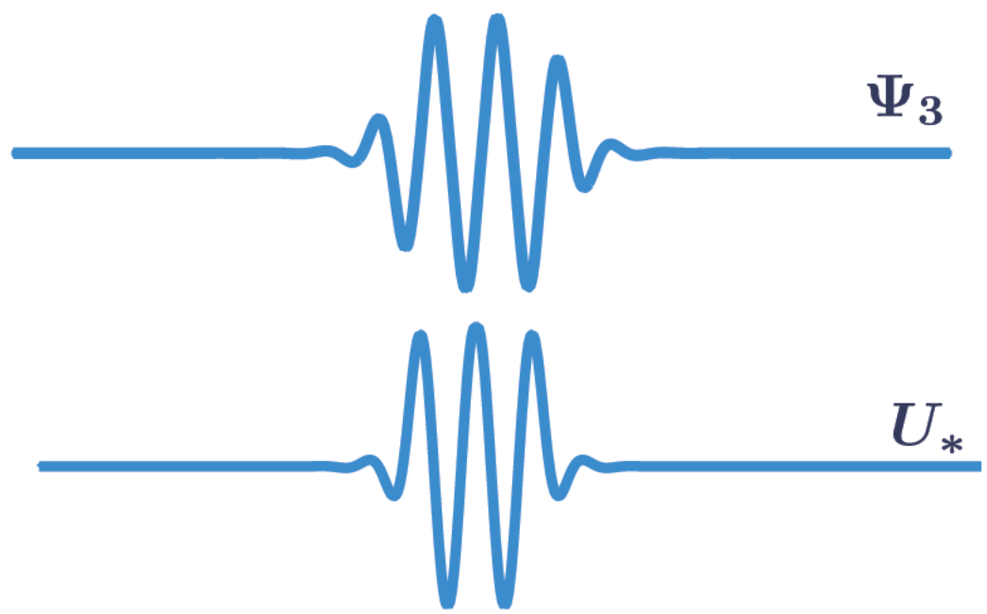
$$\begin{aligned}\sigma_\xi \mathcal{F}(u) &= \sigma_\xi\big[-(1+\partial_x^2)^2 u - \mu u + \nu u^3 - u^5\big] \\ &= -(1+\partial_x^2)^2 \sigma_\xi u - \mu \sigma_\xi u + \nu(\sigma_\xi u)^3 - (\sigma_\xi u)^5 \\ &= \mathcal{F}(\sigma_\xi u)\end{aligned}$$

- Consequences of translation invariance:
  1. If $u_*(x)$ is a steady state, so is an arbitrary translation $u_*(x+\xi)$
  2. The operator $\partial_u \mathcal{F}(u_*)$ has an eigenvalue 0 corresponding to eigenfunction $\partial_x u_*$

Numerical considerations

- Our differentiation matrices impose periodicity
- We expect an eigenvalue $\lambda \approx 0$ with eigenvector $\Psi \approx D_x U_*$

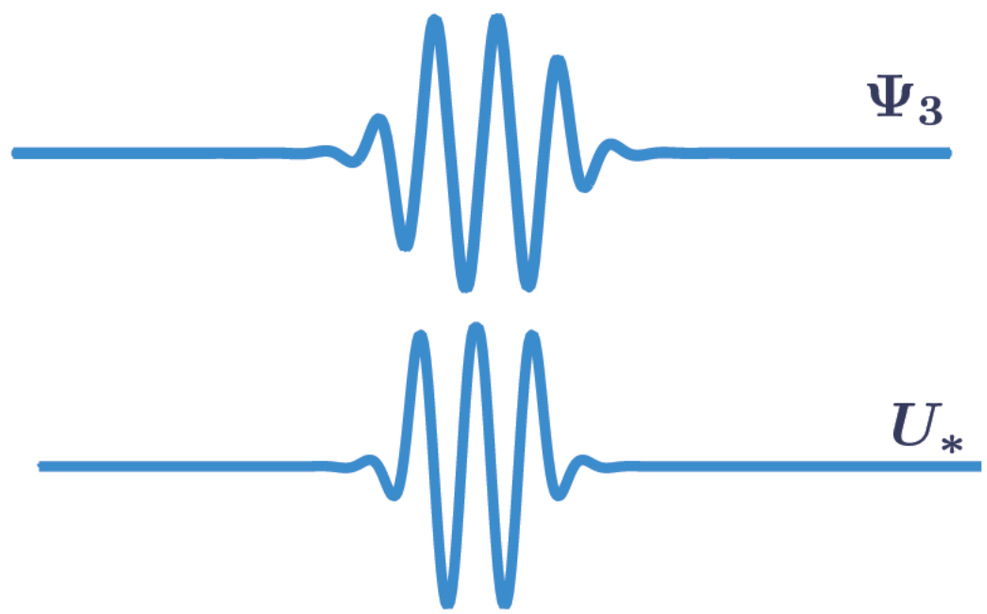
$\lambda_3 \approx 7 \times 10^{-11}$



Numerical considerations

- Our differentiation matrices impose periodicity
- We expect an eigenvalue $\lambda \approx 0$ with eigenvector $\Psi \approx D_x U_*$
- This means $D_U F(U_*)$ is nearly singular.

$\lambda_3 \approx 7 \times 10^{-11}$

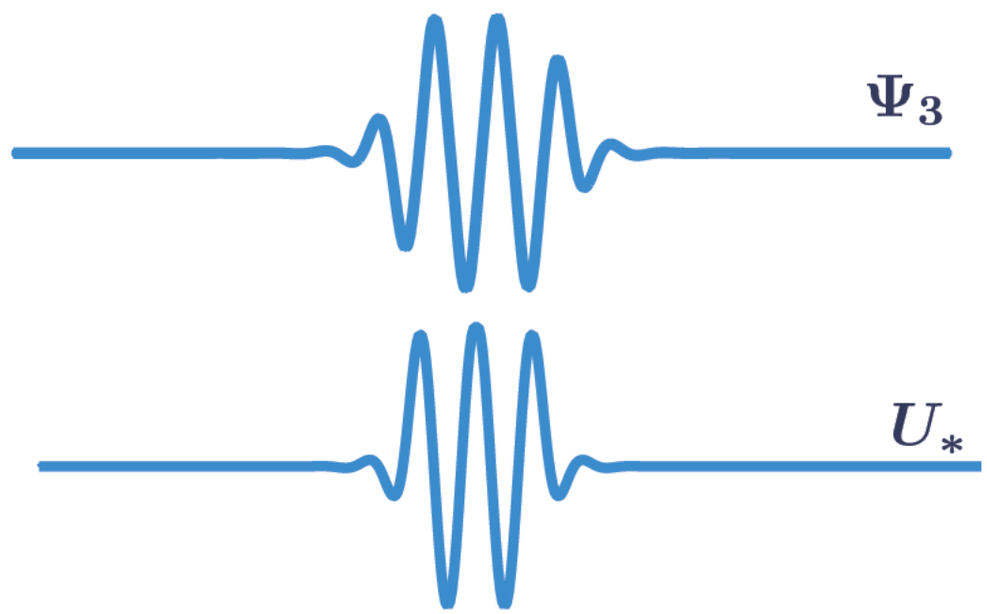


Numerical considerations

- Our differentiation matrices impose periodicity
- We expect an eigenvalue $\lambda \approx 0$ with eigenvector $\Psi \approx D_x U_*$
- This means $D_U F(U_*)$ is nearly singular.
- The solution U_* is not a regular zero of F

This is a disaster for Newton's iterates!

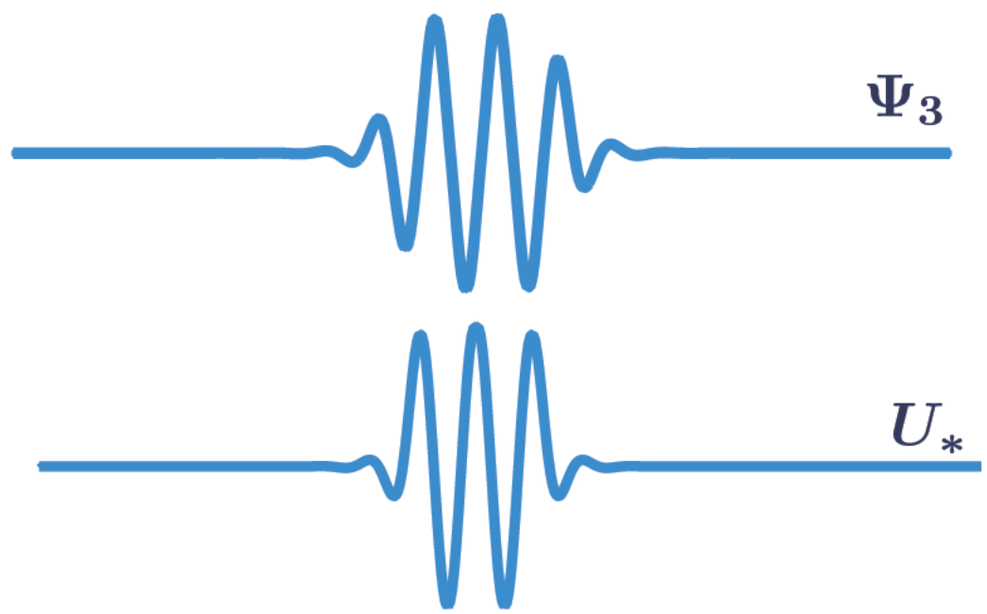
$\lambda_3 \approx 7 \times 10^{-11}$



Numerical considerations

- Our differentiation matrices impose periodicity
- We expect an eigenvalue $\lambda \approx 0$ with eigenvector $\Psi \approx D_x U_*$
- This means $D_U F(U_*)$ is nearly singular.

$\lambda_3 \approx 7 \times 10^{-11}$



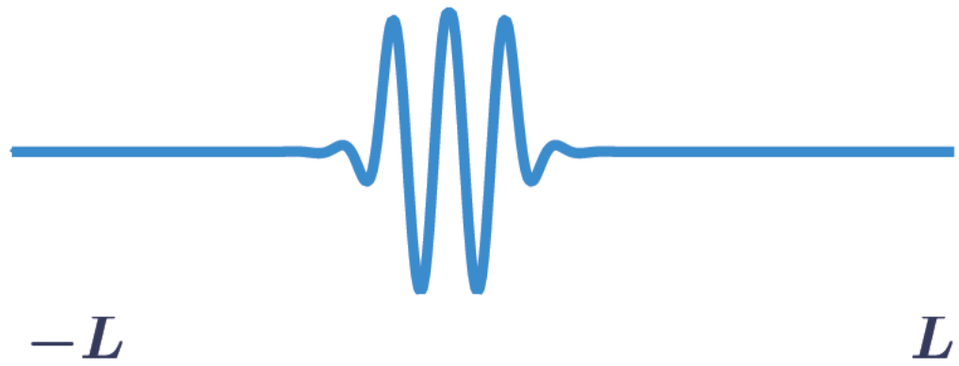
- The solution U_* is not a regular zero of F

This is a disaster for Newton's iterates!

- Possible remedies:
  1. Remove invariance via boundary conditions
  2. Regularise the problem via a phase condition

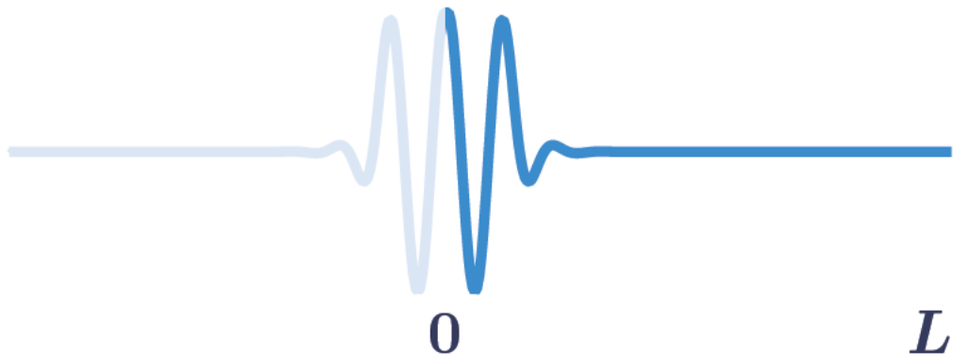
Regularisation via boundary conditions

See Codes



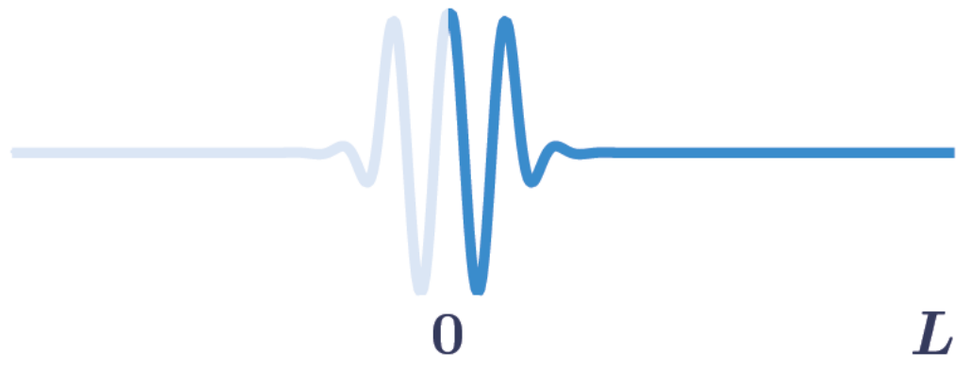
Regularisation via boundary conditions

See Codes



Regularisation via boundary conditions

See Codes



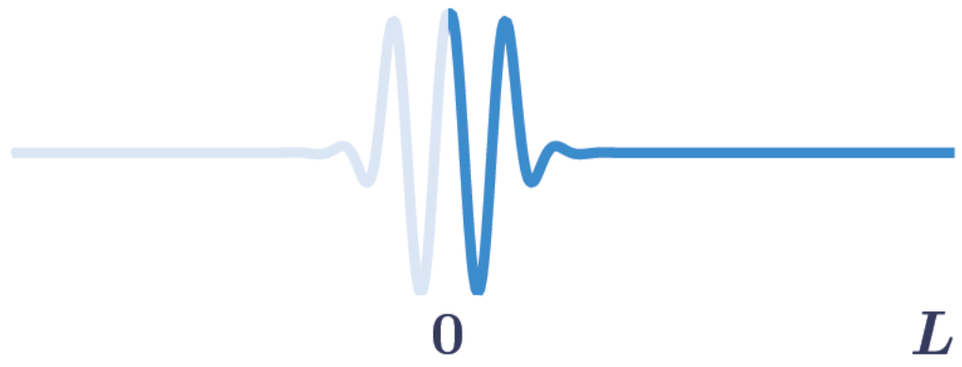
- Pose the problem on half-domain with Neumann boundary conditions

$$-(1+\partial_x^2)^2 u - \mu u + u^3 - u^5 = 0 \qquad x \in (0, L)$$

$$\partial_x u = 0, \quad \partial_x^3 u = 0 \qquad x = 0, L$$

Regularisation via boundary conditions

See Codes



- Pose the problem on half-domain with Neumann boundary conditions

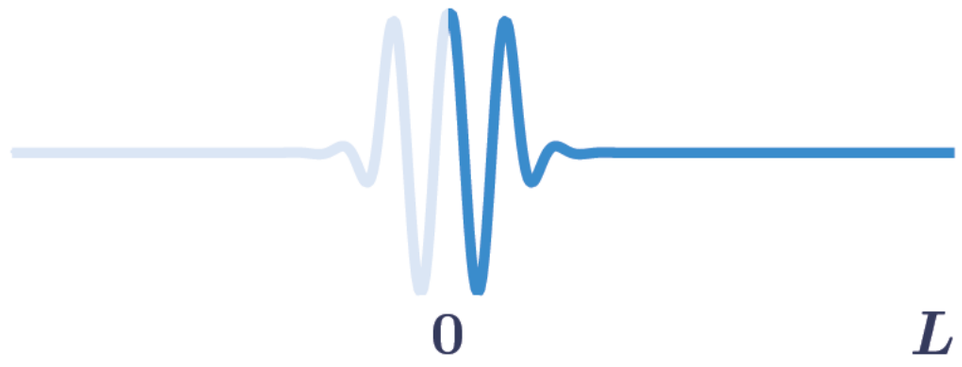
$$-(1+\partial_x^2)^2 u - \mu u + u^3 - u^5 = 0 \qquad x \in (0, L)$$

$$\partial_x u = 0, \quad \partial_x^3 u = 0 \qquad x = 0, L$$

- Advantages:
  1. Only a change in differentiation matrix is required
  2. The same accuracy is obtained with half the number of grid points

Regularisation via boundary conditions

See Codes



- Pose the problem on half-domain with Neumann boundary conditions

$$-(1+\partial_x^2)^2 u - \mu u + u^3 - u^5 = 0 \qquad x \in (0, L)$$

$$\partial_x u = 0, \quad \partial_x^3 u = 0 \qquad x = 0, L$$

- Advantages:
  1. Only a change in differentiation matrix is required
  2. The same accuracy is obtained with half the number of grid points

- Disadvantages:
  1. Only *even* solutions to the original BVP are computed
  2. Stability calculations require some amendments

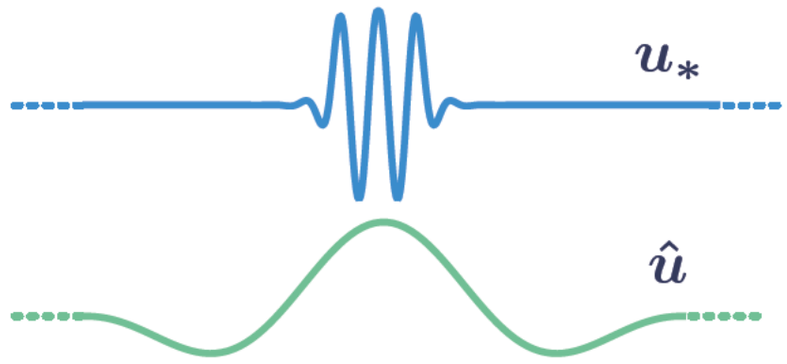
Regularisation via phase conditions

- Solve for (u, c) the following BVP with periodic boundary conditions

$$-(1+\partial_x^2)^2 u - \mu u + u^3 - u^5 + c\partial_x u = 0, \qquad x \in [-L, L)$$
$$\mathcal{B}_{\text{per}}(u) = 0$$
$$\varphi(u, \hat{u}) = 0$$

- Where
  1. The original BVP has been perturbed by a term $c\partial_x u$
  2. The solution $\hat{u}$ is a reference/template solution
  3. The last equation is a phase condition

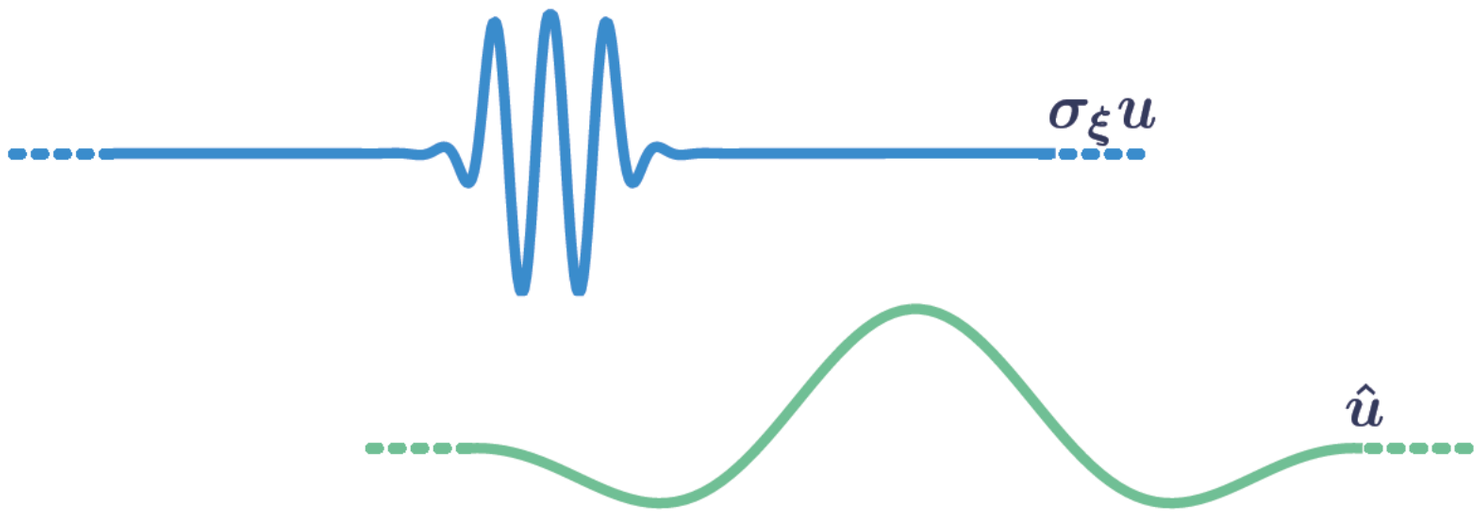
$$\varphi(u, \hat{u}) = \int_{-L}^{L} \partial_x \hat{u}(x)\big(u(x) - \hat{u}(x)\big)\, dx$$



- Things to note:
  1. $(u_*, 0)$ is a regular solution to the BVP above (Bordering Lemma)
  2. The phase condition tries to minimise the L_2-distance to the template solution

Phase condition

See Codes

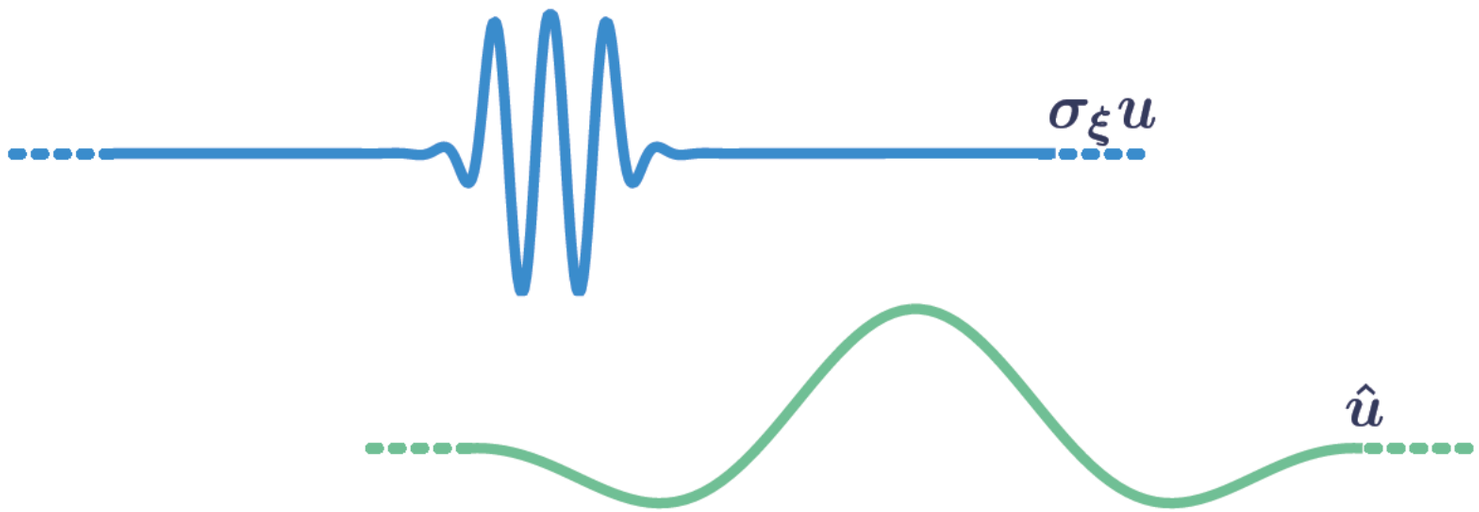


- Distance between shifted solution and reference function

$$\mathcal{D}(u,\xi) = \int_{-L}^{L} |u(x-\xi) - \hat{u}(x)|^2 \, dx = \|\sigma_\xi u - \hat{u}\|_{L_2}^2$$

Phase condition

See Codes



- Distance between shifted solution and reference function

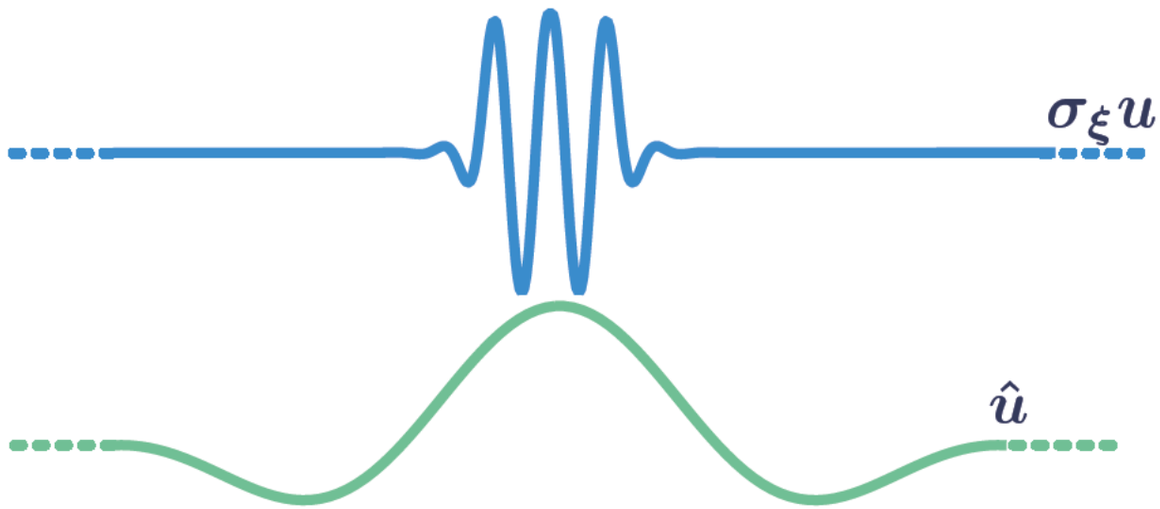
$$\mathcal{D}(u,\xi) = \int_{-L}^{L} |u(x-\xi) - \hat{u}(x)|^2 \, dx = \|\sigma_\xi u - \hat{u}\|_{L_2}^2$$

- Differentiating w.r.t. ξ we obtain a necessary condition for a local minimum

$$\int_{-L}^{L} \partial_x \hat{u}(x) \big(u(x-\xi) - \hat{u}(x)\big) \, dx = 0$$

Phase condition

See Codes



- Distance between shifted solution and reference function

$$\mathcal{D}(u,\xi) = \int_{-L}^{L} |u(x-\xi) - \hat{u}(x)|^2 \, dx = \|\sigma_\xi u - \hat{u}\|_{L_2}^2$$

- Differentiating w.r.t. ξ we obtain a necessary condition for a local minimum

$$\int_{-L}^{L} \partial_x \hat{u}(x)\big(u(x-\xi) - \hat{u}(x)\big) \, dx = 0$$

- The phase condition $\varphi(u,\hat{u}) = 0$ selects a solution for which the above equation holds at $\xi = 0$

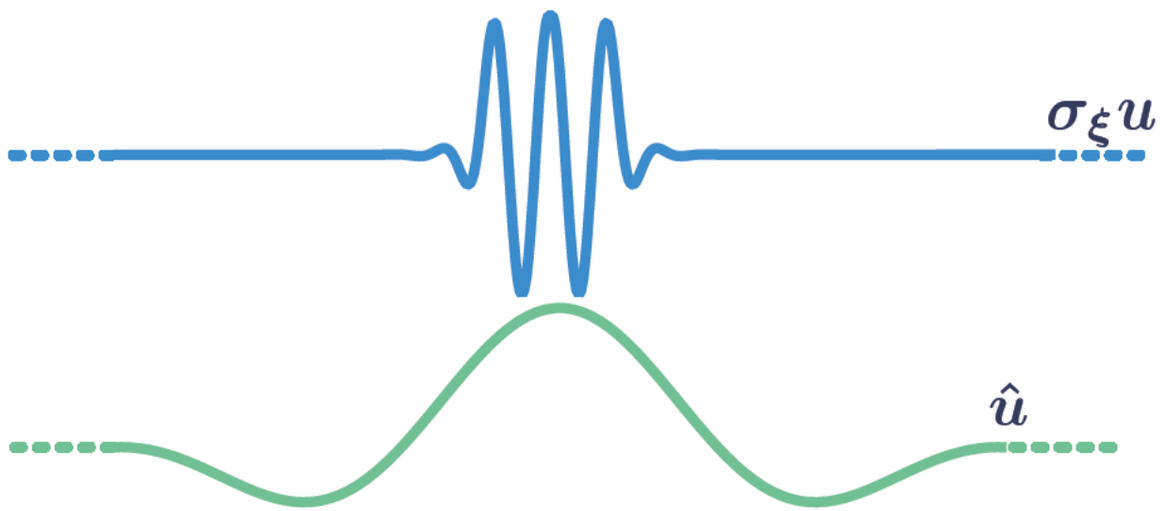
Phase condition

See Codes

[Doedel]

[Beyn, Thümmler]

[Champneys, Sandstede]



- Distance between shifted solution and reference function

$$\mathcal{D}(u,\xi) = \int_{-L}^{L} |u(x-\xi) - \hat{u}(x)|^2 \, dx = \|\sigma_\xi u - \hat{u}\|_{L_2}^2$$

- Differentiating w.r.t. ξ we obtain a necessary condition for a local minimum

$$\int_{-L}^{L} \partial_x \hat{u}(x)\big(u(x-\xi) - \hat{u}(x)\big) \, dx = 0$$

- The phase condition $\varphi(u,\hat{u}) = 0$ selects a solution for which the above equation holds at $\xi = 0$

- Phase conditions (and borderings) can be constructed for more general symmetries

Neural fields: PDE and integral form

Integral Neural Field Model

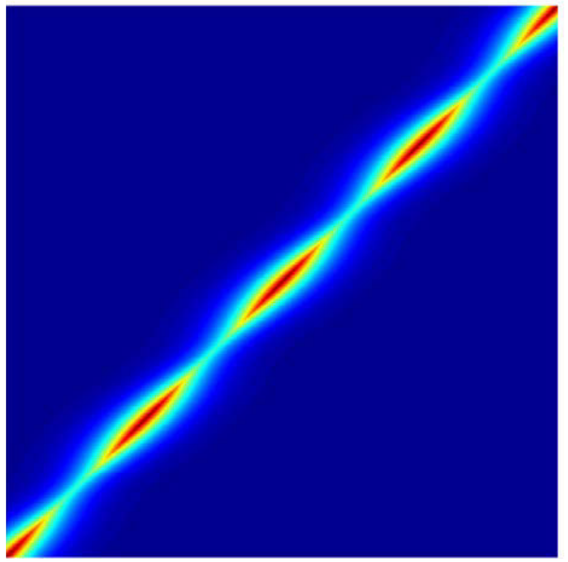
- **Wilson-Cowan-Amari neural field equation**

$$\partial_t u(x,t) = -u(x,t) + \int_{-\infty}^{\infty} W(x,y) f(u(y,t))\,\mathrm{d}y, \quad x \in \mathbb{R}$$

- **Heterogeneous connectivity function**

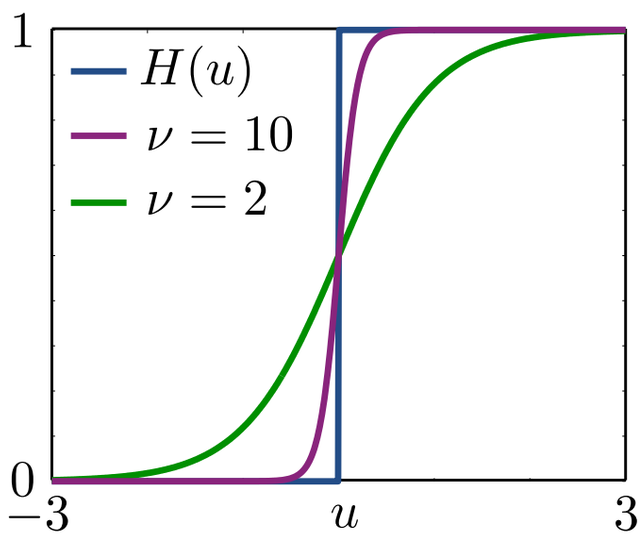
$$\begin{aligned} W(x,y;a,\varepsilon) &= w(|x-y|)A(y;a,\varepsilon) \\ &= \frac{1}{2} e^{-|x-y|}\big(1 + a\cos(y/\varepsilon)\big), \end{aligned}$$

$W(x,y)$



- **Firing rate function**

$$f(u;h,\nu) = \frac{1}{1+\exp(-\nu(u-h))}$$



[A. & Schmidt, Bressloff, Schmidt et al, Coombes & Laing]

PDE method [Laing]

- Fourier transform

$$\partial_t \hat{u}(\xi,t) = -\hat{u}(\xi,t) + \hat{w}(\xi)\widehat{\big(Af(u)\big)}(\xi,t),$$

where $\hat{w}(\xi) = P(\xi)/Q(\xi)$

- Inverse Fourier transform

$$\mathcal{L}_Q\left[\partial_t u + u\right] = \mathcal{L}_P\left[A(x)f(u;h,\nu)\right]$$

- For our choice of the kernel:

Evolution equation $$(1-\partial_x^2)\partial_t u = (\partial_x^2 - 1)u + A(x)f(u)$$

Steady states $$0 = (\partial_x^2 - 1)u + A(x)f(u)$$

Stability $$\lambda(1-\partial_x^2)\psi = \left[\partial_x^2 - 1 + A(x)f'(u)\right]\psi$$

PDE discretisation

See Codes

- For our choice of the kernel:

  Evolution equation $$(1-\partial_x^2)\partial_t u = (\partial_x^2 - 1)u + A(x)f(u)$$

  Steady states $$0 = (\partial_x^2 - 1)u + A(x)f(u)$$

  Stability $$\lambda(1-\partial_x^2)\psi = \left[\partial_x^2 - 1 + A(x)f'(u)\right]\psi$$

- Form the matrix $M = I_n - D_{xx}$

  Time step using mass matrix $$M\dot{U} = -MU + A \odot f(U)$$

  Steady states continuation $$0 = -MU + A \odot f(U)$$

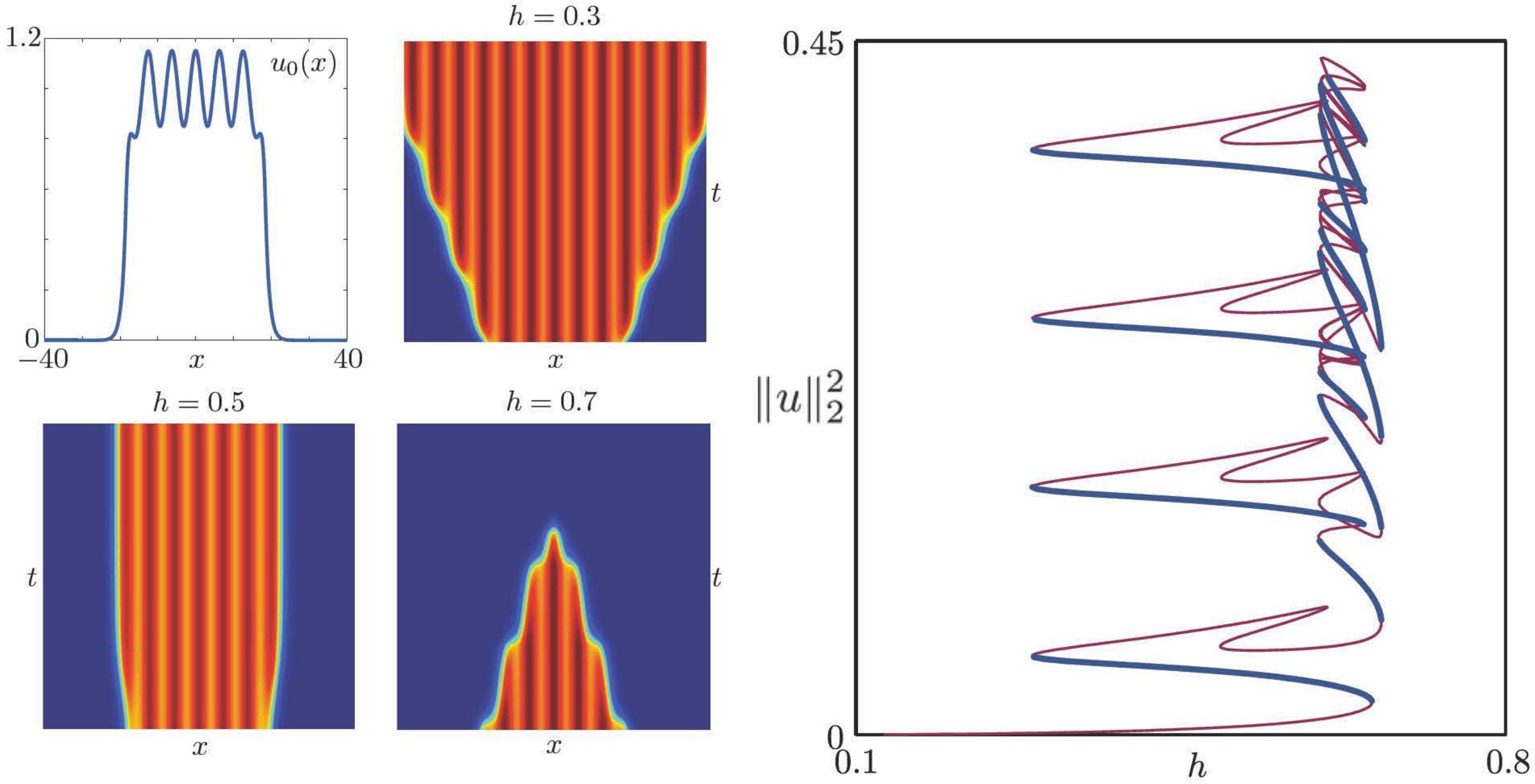
  Steady states stability $$\lambda M\Psi = \left[-M + \mathrm{diag}(A \odot f'(U))\right]\Psi$$

- This fits into the framework used for the SH equation!

- Read on for continuation of neural field models in integral form.

Numerical computations

See Codes

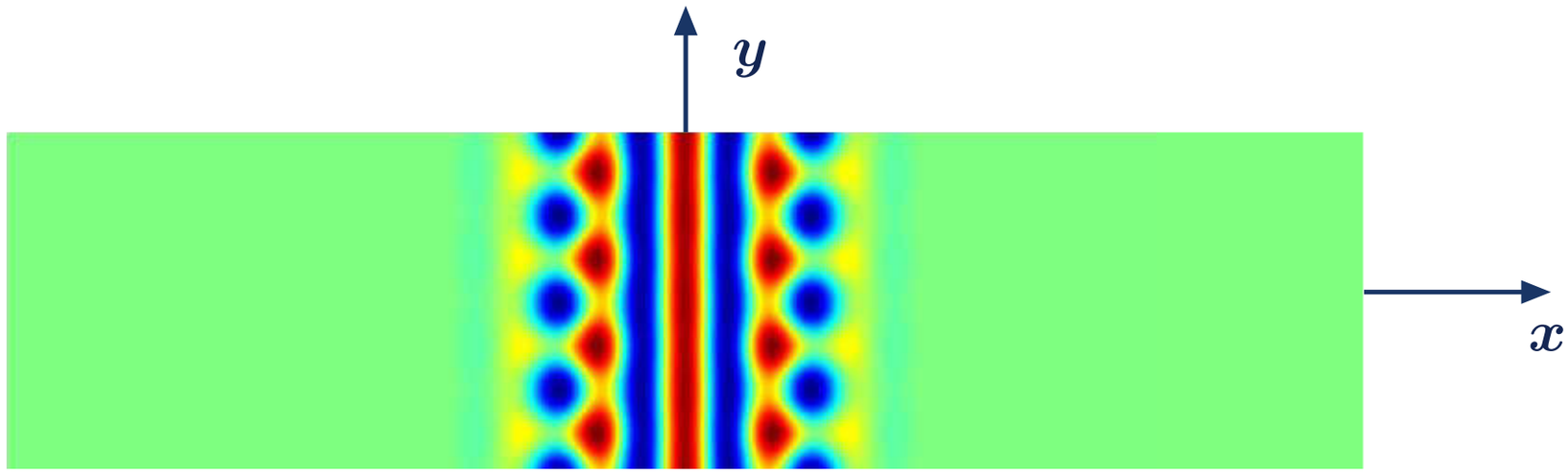


[A. & Schmidt]

From 1D to 2D

Swift-Hohenberg on a plane

See Codes



- Swift-Hohenberg equation posed on a 2D domain

$$\partial u_t = -(1+\Delta)^2 u - \mu u + \nu u^3 - u^5, \qquad (x,y) \in \mathbb{R} \times \mathbb{R}/2L_y\mathbb{Z}$$

- We would like to obtain a discrete form of this type

$$\dot{U} = -(I+L)^2 U - \mu U + \nu U^{\odot 3} - U^{\odot 5}$$

- The matrix L is the *differentiation matrix* for the Laplacian
  1. It can be computed elegantly for tensor grids
  2. Different discretisations (or discretisation methods!) are allowed for x and y

Building the Laplacian for differentiation matrices

- Consider a grid of $N_x \times N_y$ nodes, and pose $u(x_i, y_j) \approx U_{i,j}$

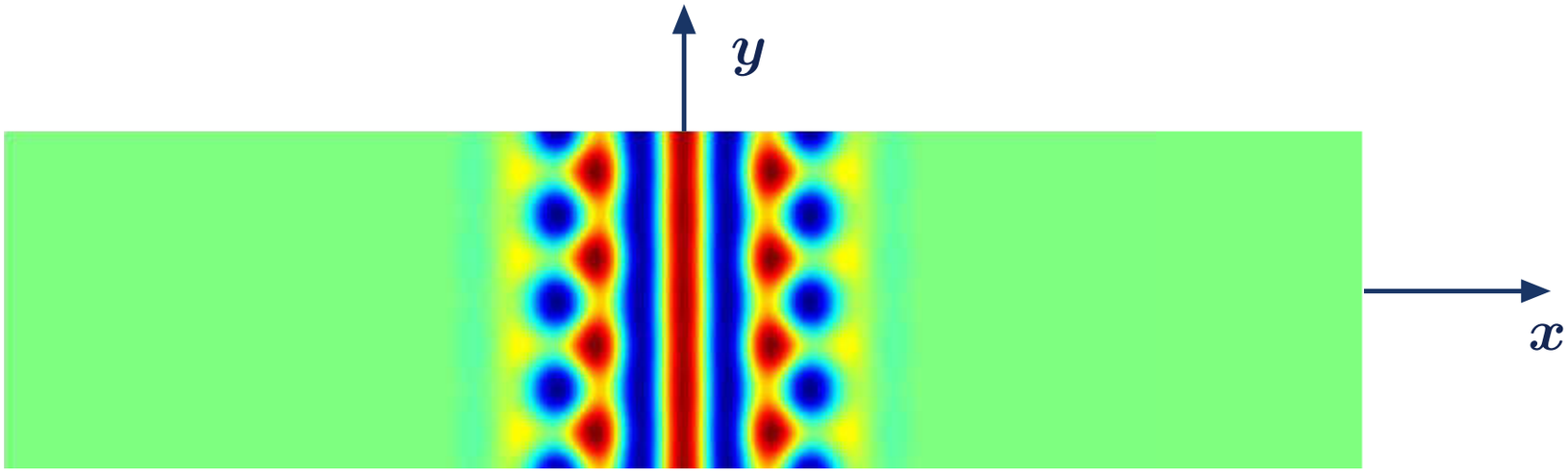
- If the component of the unknown vector $U \in \mathbb{R}^{N_x N_y \times 1}$ are in lexicographic order, the discrete Laplacian can be computed using the Kronecker product $\otimes$

$$L = D_{xx} \otimes I_y + I_x \otimes D_{yy}$$

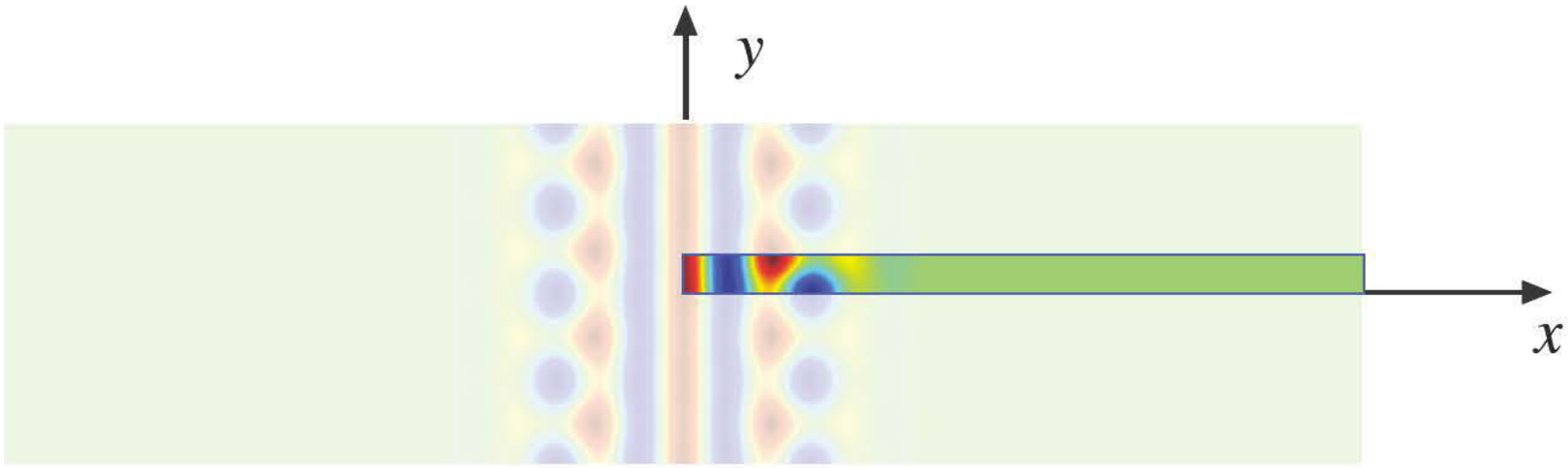
- Where $D_{xx}, I_x \in \mathbb{R}^{N_x \times N_x}$ and $D_{yy}, I_y \in \mathbb{R}^{N_y \times N_y}$ are calculated independently

- The framework seen in 1D is easy to adapt to the 2D case

Notes on 2D calculations

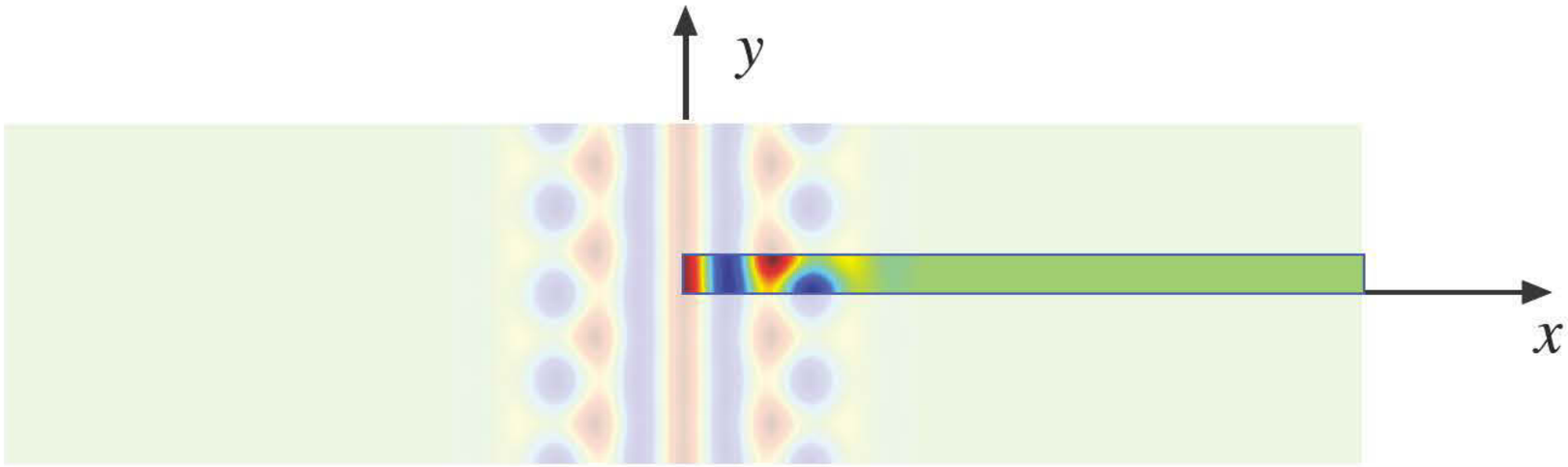


Notes on 2D calculations

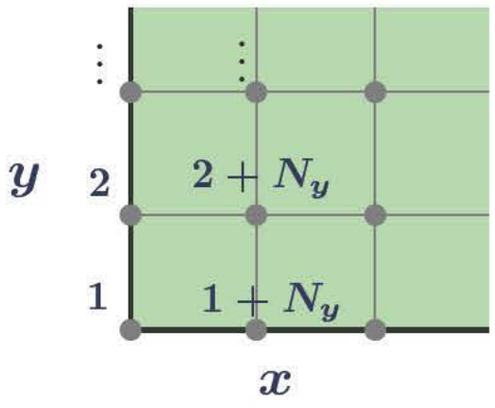


- Use symmetries whenever possible

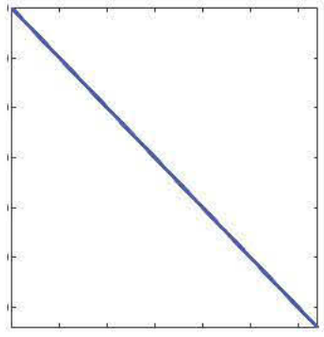
Notes on 2D calculations

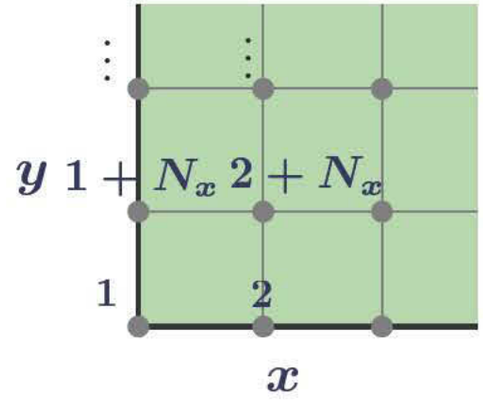


- Use symmetries whenever possible
- Ordering may affect the sparsity pattern/matrix bandwidth

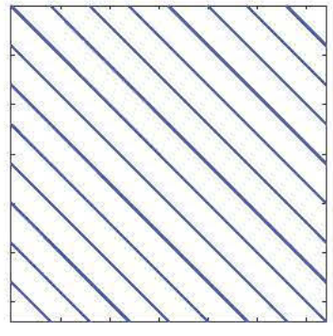


$$L = D_{xx} \otimes I_y + I_x \otimes D_{yy}$$





$$L = D_{yy} \otimes I_x + I_y \otimes D_{xx}$$



Notes on 2D calculations

- Linear algebra routines could be a bottleneck. Sometimes we need to
  1. Use of efficient solvers for bordered systems
  2. Use of Matrix-Free methods (Krylov methods)
  3. Use preconditioners for iterative linear systems

- Finite Element methods may be more efficient:
  1. Allow a greater flexibility in allocating nodes
  2. Mesh adaptation is possible
  3. The tensor-product trick will not work in general

Spatio-temporal structures

Coherent structures

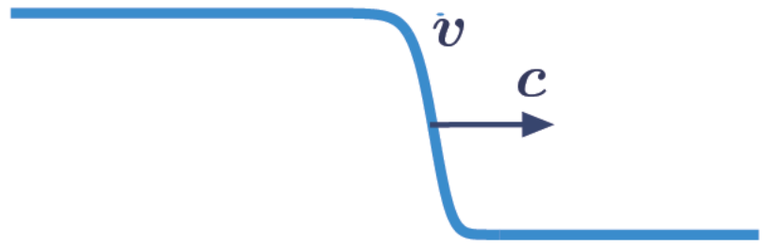
- *Coherent structures*: solutions to nonlinear evolution equations with a special spatio-temporal structure
  1. Travelling waves and solitons
  2. Spiral waves in 2D
  3. Breathers
  4. Oscillons

- Strategy to compute coherent structures
  1. Prescribe a defining system of equations (a well-posed BVP, often symmetries require phase conditions)
  2. Setup an eigenvalue problem for linear stability
  3. Use the methods seen in the past sections

Travelling wave continuation

- We derive the defining problem for TW of

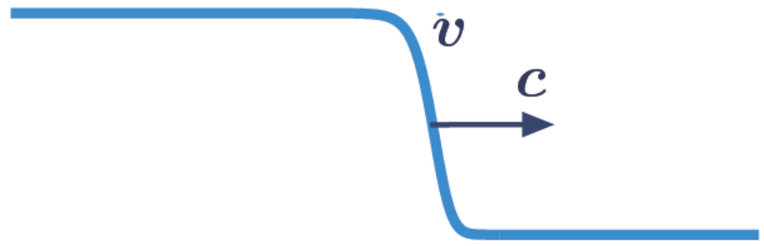
$$\partial_t u = \mathcal{F}(u)$$
$$0 = \mathcal{B}(u)$$



Travelling wave continuation

- We derive the defining problem for TW of

$$\partial_t u = \mathcal{F}(u)$$
$$0 = \mathcal{B}(u)$$



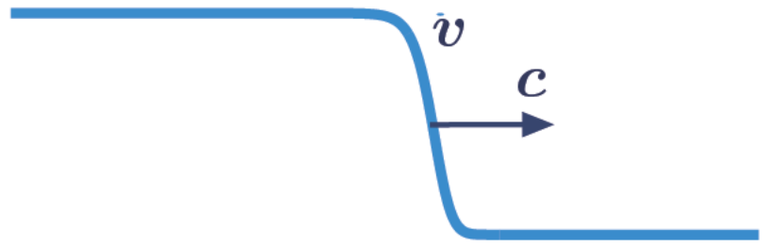
- Travelling waves are special solutions to the evolution equation above

$$u(x,t) = v(x - ct)$$

Travelling wave continuation

- We derive the defining problem for TW of

$$\partial_t u = \mathcal{F}(u)$$
$$0 = \mathcal{B}(u)$$



- Travelling waves are special solutions to the evolution equation above

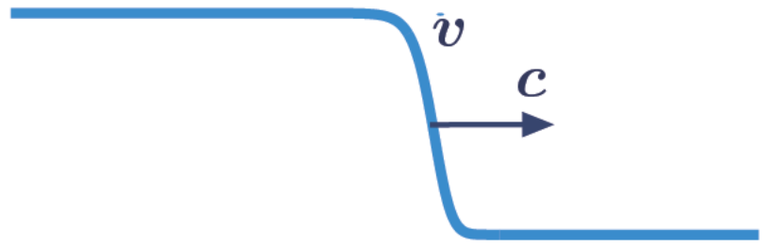
$$u(x,t) = v(x - ct)$$

- Need to determine the *profile* v and the *wave speed* c

Travelling wave continuation

- We derive the defining problem for TW of

$$\partial_t u = \mathcal{F}(u)$$
$$0 = \mathcal{B}(u)$$



- Travelling waves are special solutions to the evolution equation above

$$u(x,t) = v(x - ct)$$

- Need to determine the *profile* v and the *wave speed* c

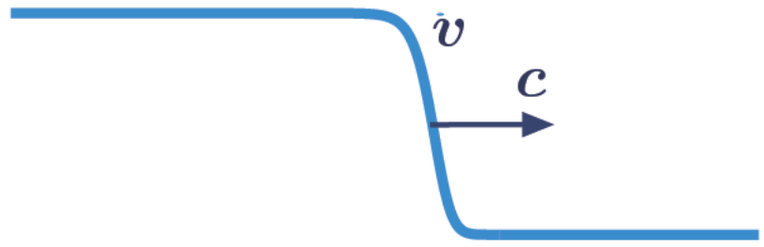
- Let $z = x - ct$ and substitute the TW ansatz in the evolution equation

$$c\partial_z v(z) + \mathcal{F}(v)(z) = 0$$

Travelling wave continuation

- We derive the defining problem for TW of

$$\partial_t u = \mathcal{F}(u)$$
$$0 = \mathcal{B}(u)$$



- Travelling waves are special solutions to the evolution equation above

$$u(x,t) = v(x - ct)$$

- Need to determine the *profile* v and the *wave speed* c

- Let $z = x - ct$ and substitute the TW ansatz in the evolution equation

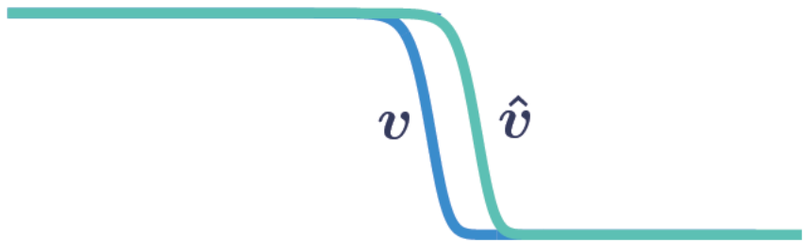
$$c\partial_z v(z) + \mathcal{F}(v)(z) = 0$$

- This is not a valid defining system!
  1. We need one extra equation to determine c
  2. The linearised operator has a 0 eigenvalue, with eigenfunction $\partial_z v$

Travelling waves - defining system

- Defining system: find (v, c) such that

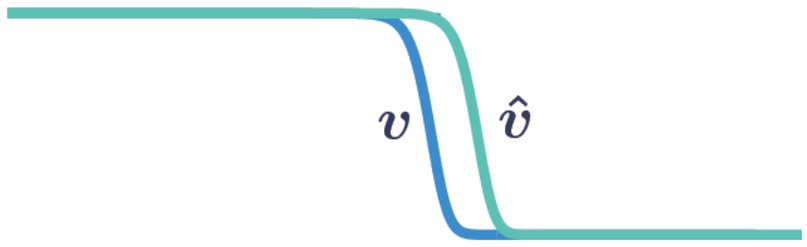
$$c\partial_z v + \mathcal{F}(v) = 0$$
$$\mathcal{B}(v) = 0$$
$$\varphi(v, \hat{v}) = 0$$



Travelling waves - defining system

- Defining system: find (v, c) such that

$$c\partial_z v + \mathcal{F}(v) = 0$$
$$\mathcal{B}(v) = 0$$
$$\varphi(v, \hat{v}) = 0$$



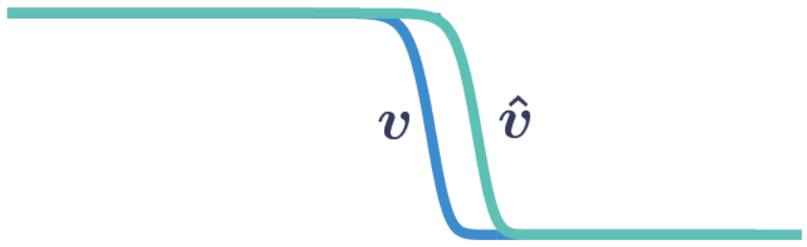
- Stability of a travelling wave (v_*, c_*)

$$u(x,t) = v_*(x - ct) + \varepsilon\tilde{v}(x - ct, t), \qquad 0 < \varepsilon \ll 1, \quad \|\tilde{v}\| = \mathcal{O}(1)$$

Travelling waves - defining system

- Defining system: find (v, c) such that

$$c\partial_z v + \mathcal{F}(v) = 0$$
$$\mathcal{B}(v) = 0$$
$$\varphi(v, \hat{v}) = 0$$



- Stability of a travelling wave (v_*, c_*)

$$u(x,t) = v_*(x - ct) + \varepsilon\tilde{v}(x - ct, t), \qquad 0 < \varepsilon \ll 1, \quad \|\tilde{v}\| = \mathcal{O}(1)$$

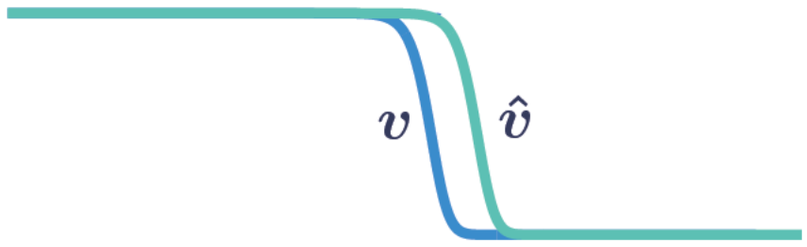
- At $\mathcal{O}(\varepsilon)$, the perturbations evolve according to

$$\partial_t \tilde{v} = \left[c_*\partial_z + \partial_u \mathcal{F}(v_*)\right]\tilde{v} := \mathcal{L}(v_*, c_*)\tilde{v}$$

Travelling waves - defining system

- Defining system: find (v, c) such that

$$c\partial_z v + \mathcal{F}(v) = 0$$
$$\mathcal{B}(v) = 0$$
$$\varphi(v, \hat{v}) = 0$$



- Stability of a travelling wave (v_*, c_*)

$$u(x,t) = v_*(x - ct) + \varepsilon\tilde{v}(x - ct, t), \qquad 0 < \varepsilon \ll 1, \quad \|\tilde{v}\| = \mathcal{O}(1)$$

- At $\mathcal{O}(\varepsilon)$, the perturbations evolve according to

$$\partial_t \tilde{v} = \left[c_*\partial_z + \partial_u \mathcal{F}(v_*)\right]\tilde{v} := \mathcal{L}(v_*, c_*)\tilde{v}$$

- The travelling wave is linearly stable if $\mathrm{Spec}\left[\mathcal{L}(v_*, c_*)\right] \setminus \{0\}$ is contained in the left half of the complex plane

Travelling waves in a neural field

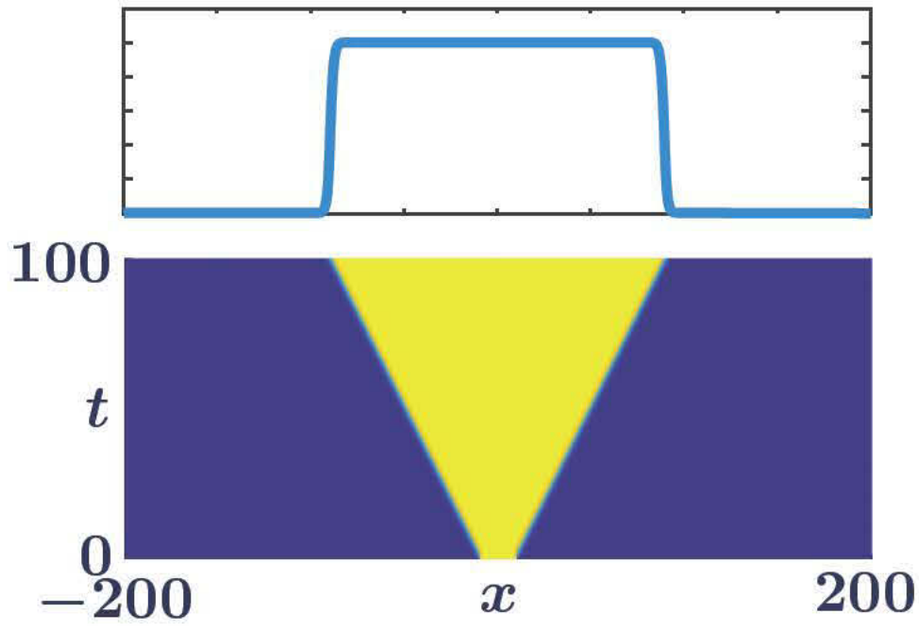
See Codes

- **Neural field example** $x \in \mathbb{R}$

$$\partial_u(x,t) = -u(x,t) + \int_{\mathbb{R}} w(x-y) S(u(y,t))\, dy,$$

- **Synaptic kernel and firing-rate function**

$$w(x) = \frac{1}{2} e^{-|x|}, \qquad S(u) = \frac{1}{1+e^{-\beta(u-h)}}$$

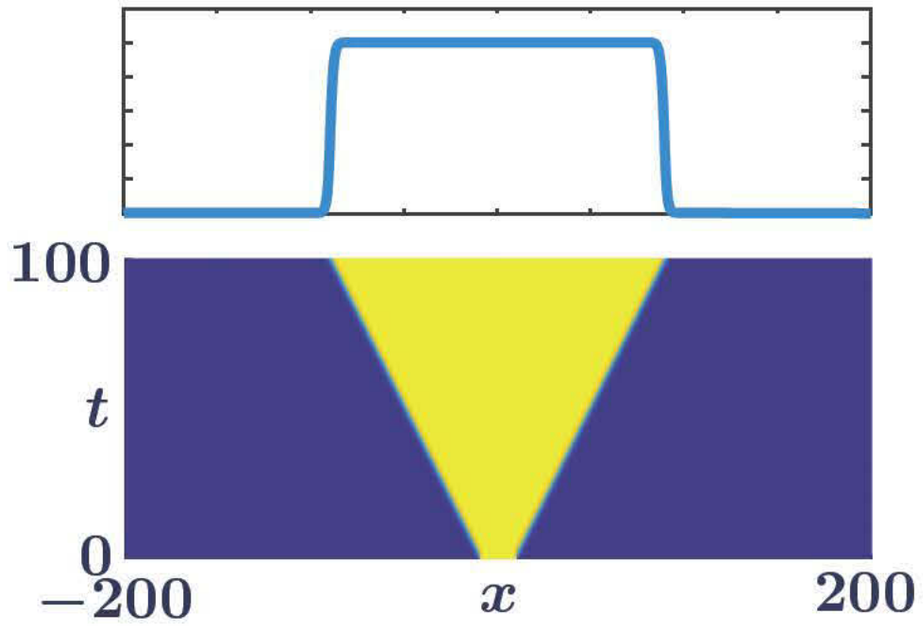


Travelling waves in a neural field

See Codes

- **Neural field example** $x \in \mathbb{R}$

$$\partial_u(x,t) = -u(x,t) + \int_{\mathbb{R}} w(x-y)S(u(y,t))\,dy,$$



- **Synaptic kernel and firing-rate function**

$$w(x) = \frac{1}{2}e^{-|x|}, \qquad S(u) = \frac{1}{1+e^{-\beta(u-h)}}$$

- **Our framework can be applied with**

$$\mathcal{F}: u \mapsto -u + w * S(u)$$

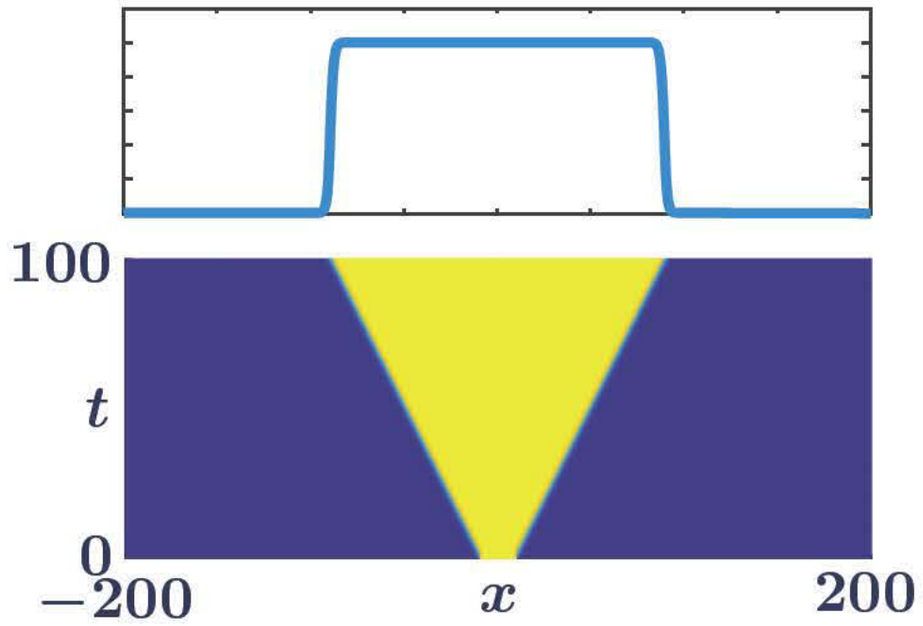
$$\partial_u\mathcal{F}(u): v \mapsto -v + w * \big(S'(u)v\big)$$

Travelling waves in a neural field

See Codes

- **Neural field example** $x \in \mathbb{R}$

$$\partial_u(x,t) = -u(x,t) + \int_{\mathbb{R}} w(x-y)S(u(y,t))\,dy,$$



- **Synaptic kernel and firing-rate function**

$$w(x) = \frac{1}{2}e^{-|x|}, \qquad S(u) = \frac{1}{1+e^{-\beta(u-h)}}$$

- **Our framework can be applied with**

$$\mathcal{F}: u \mapsto -u + w * S(u)$$
$$\partial_u\mathcal{F}(u): v \mapsto -v + w * \big(S'(u)v\big)$$

- **Discrete version**

$$F(U) = -U + WS(U)$$
$$D_U F(U) = -I + W\,\mathrm{diag}(S'(U))$$

Travelling waves in a neural field

See Codes

- Neural field example $x \in \mathbb{R}$

$$\partial_u(x,t) = -u(x,t) + \int_{\mathbb{R}} w(x-y)S(u(y,t))\,dy,$$

- Synaptic kernel and firing-rate function

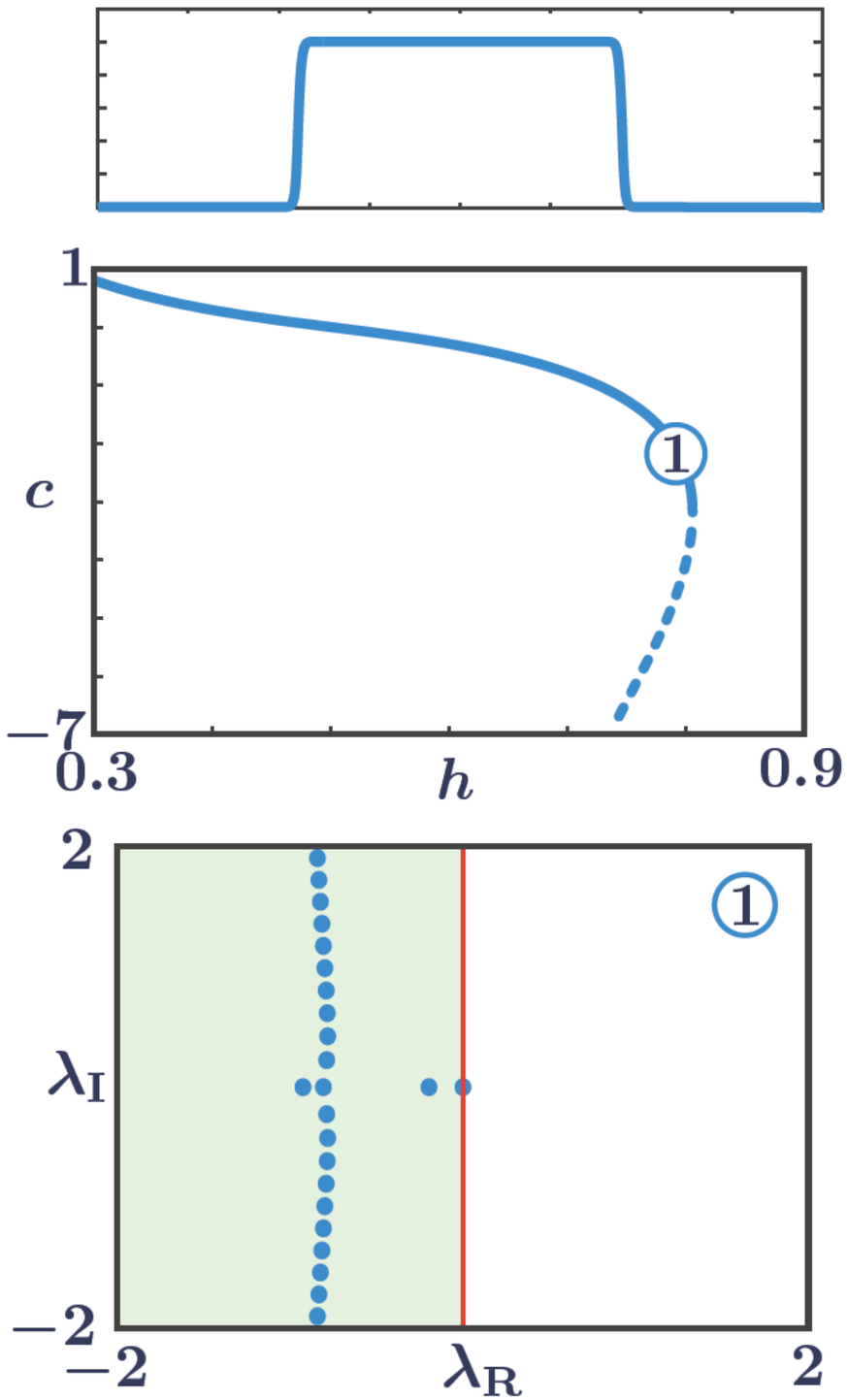
$$w(x) = \frac{1}{2}e^{-|x|}, \qquad S(u) = \frac{1}{1+e^{-\beta(u-h)}}$$

- Our framework can be applied with

$$\mathcal{F}: u \mapsto -u + w * S(u)$$
$$\partial_u \mathcal{F}(u): v \mapsto -v + w * (S'(u)v)$$

- Discrete version

$$F(U) = -U + WS(U)$$
$$D_U F(U) = -I + W\,\mathrm{diag}(S'(U))$$



Travelling waves in a neural field

See Codes

- **Neural field example** $x \in \mathbb{R}$

$$\partial_u(x,t) = -u(x,t) + \int_{\mathbb{R}} w(x-y)S(u(y,t))\,dy,$$

- **Synaptic kernel and firing-rate function**

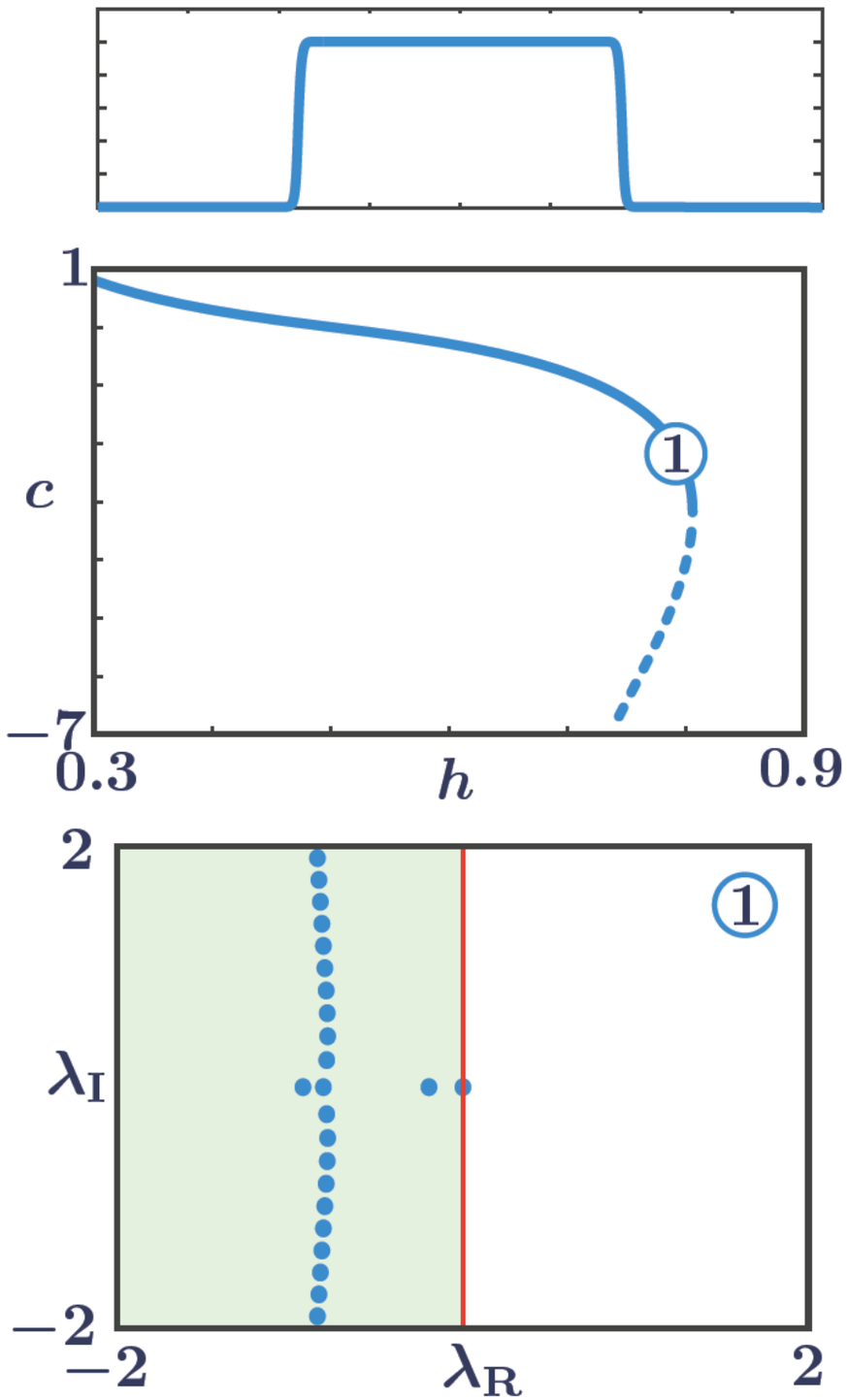
$$w(x) = \frac{1}{2}e^{-|x|}, \qquad S(u) = \frac{1}{1+e^{-\beta(u-h)}}$$

- **Our framework can be applied with**

$$\mathcal{F}: u \mapsto -u + w * S(u)$$
$$\partial_u\mathcal{F}(u): v \mapsto -v + w * \left(S'(u)v\right)$$

- **Discrete version**

$$F(U) = -U + WS(U)$$
$$D_U F(U) = -I + W\,\mathrm{diag}(S'(U))$$



- **Convolution integrals can be evaluated efficiently using FFTs. See the NewtonGMRES code in the heterogeneous example [Rankin, A., Faye, Lloyd]**

See Codes

Conclusions

- Coherent structures can be analysed following structured steps
  1. Prescribe a well-defined BVP
  2. Discretise the BVP using finite differences, spectral methods, FEM
  3. Use standard numerical continuation techniques for path-following
  4. Prescribe an eigenvalue problem
  5. Use standard eigenvalue solvers on the discretised eigenvalue problem

Conclusions

- Coherent structures can be analysed following structured steps
  1. Prescribe a well-defined BVP
  2. Discretise the BVP using finite differences, spectral methods, FEM
  3. Use standard numerical continuation techniques for path-following
  4. Prescribe an eigenvalue problem
  5. Use standard eigenvalue solvers on the discretised eigenvalue problem

- Almost all the steps above lead to (or wrap around) *a single numerical subroutine*

$$(V, c) \mapsto \Big(G(V, p), D_V G(V, p) \Big)$$

"These codes sometimes remind me of a Russian doll"

- Joshua Davis

Conclusions

- Coherent structures can be analysed following structured steps
  1. Prescribe a well-defined BVP
  2. Discretise the BVP using finite differences, spectral methods, FEM
  3. Use standard numerical continuation techniques for path-following
  4. Prescribe an eigenvalue problem
  5. Use standard eigenvalue solvers on the discretised eigenvalue problem

- Almost all the steps above lead to (or wrap around) *a single numerical subroutine*

$$(V, c) \mapsto \Big(G(V, p), D_V G(V, p) \Big)$$

"These codes sometimes remind me of a Russian doll"

- Joshua Davis

- In 2D and 3D, convergence properties of the Newton/Linear Algebra/Eigenvalue solver are often crucial to obtain numerical solution

Conclusions

- Coherent structures can be analysed following structured steps
  1. Prescribe a well-defined BVP
  2. Discretise the BVP using finite differences, spectral methods, FEM
  3. Use standard numerical continuation techniques for path-following
  4. Prescribe an eigenvalue problem
  5. Use standard eigenvalue solvers on the discretised eigenvalue problem

- Almost all the steps above lead to (or wrap around) *a single numerical subroutine*

$$(V, c) \mapsto \Big(G(V, p), D_V G(V, p) \Big)$$

"These codes sometimes remind me of a Russian doll"

- Joshua Davis

- In 2D and 3D, convergence properties of the Newton/Linear Algebra/Eigenvalue solver are often crucial to obtain numerical solution

- Choices of Newton/Linear Algebra/Eigenvalue solver may be problem dependent

Conclusions

- Coherent structures can be analysed following structured steps
  1. Prescribe a well-defined BVP
  2. Discretise the BVP using finite differences, spectral methods, FEM
  3. Use standard numerical continuation techniques for path-following
  4. Prescribe an eigenvalue problem
  5. Use standard eigenvalue solvers on the discretised eigenvalue problem

- Almost all the steps above lead to (or wrap around) *a single numerical subroutine*

$$(V, c) \mapsto \Big(G(V, p), D_V G(V, p) \Big)$$

"These codes sometimes remind me of a Russian doll"

- Joshua Davis

- In 2D and 3D, convergence properties of the Newton/Linear Algebra/Eigenvalue solver are often crucial to obtain numerical solution

- Choices of Newton/Linear Algebra/Eigenvalue solver may be problem dependent

- The steps above can also wrap around a *numerical time stepper*

$$\dot{U} = F(U; p), \qquad \Phi_T : U(t) \mapsto U(t + T)$$

[Keller] [Lust] [Tuckermann, Barkley] [Cvitanovich et al.] [Sanchez-Umbria]

Acknowledgments

- Bjorn Sandstede
- David Lloyd
- Eric Phipps
- Andrew Cliffe
- Andy Salinger
- Edgar Knobloch
- Mathieu Desroches
- Eusebius Doedel
- Alan Champneys
- Frank Schilder
- Jan Sieber
- Giovanni Samaey
- Stephen Coombes
- Rebecca Hoyle
- Wim Vanroose
- Victor Breña-Medina
- Helmut Schmidt
- Kyle Wedgwood
- Joshua Davis