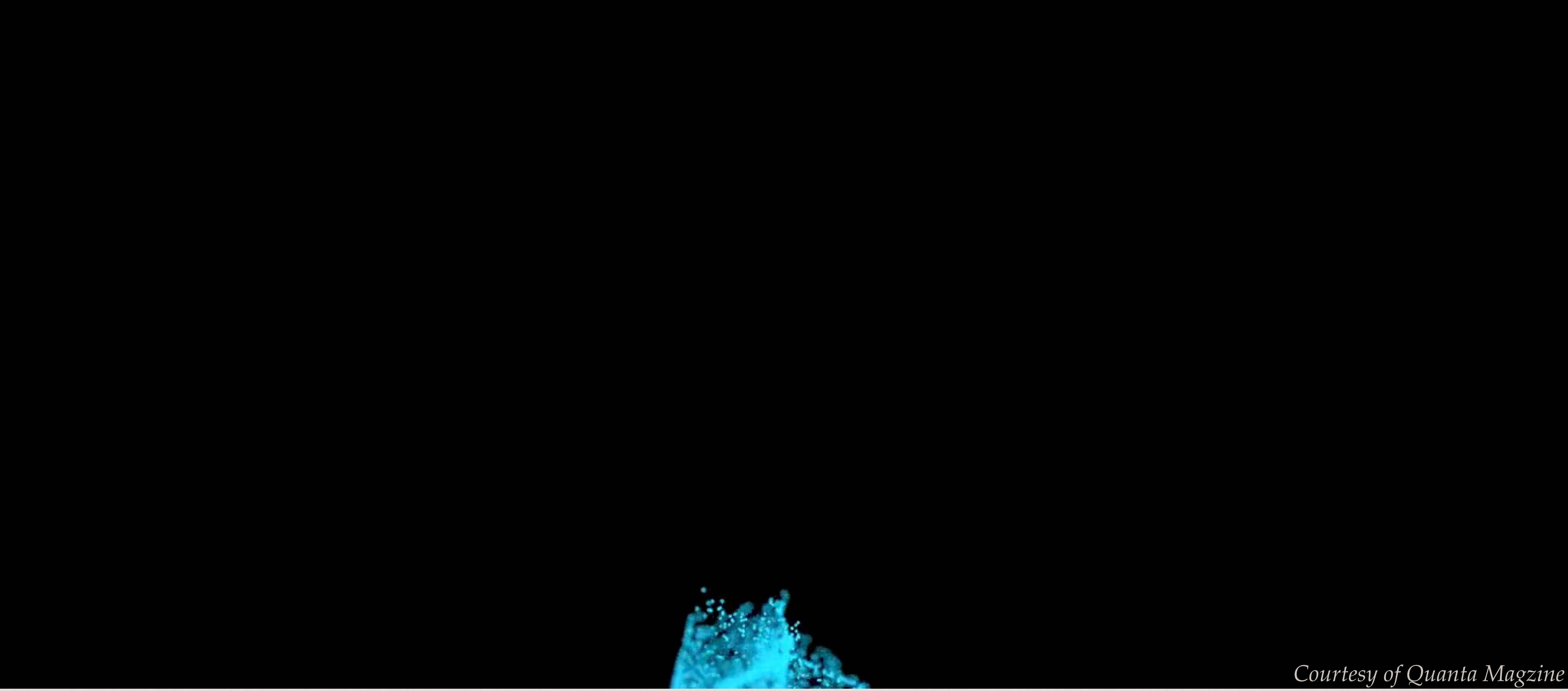


Courtesy of Quanta Magzine

Physics informed neural networks (PINNs) for blow-up solutions of Euler equations

Yongji Wang

Department of Geosciences, Princeton University

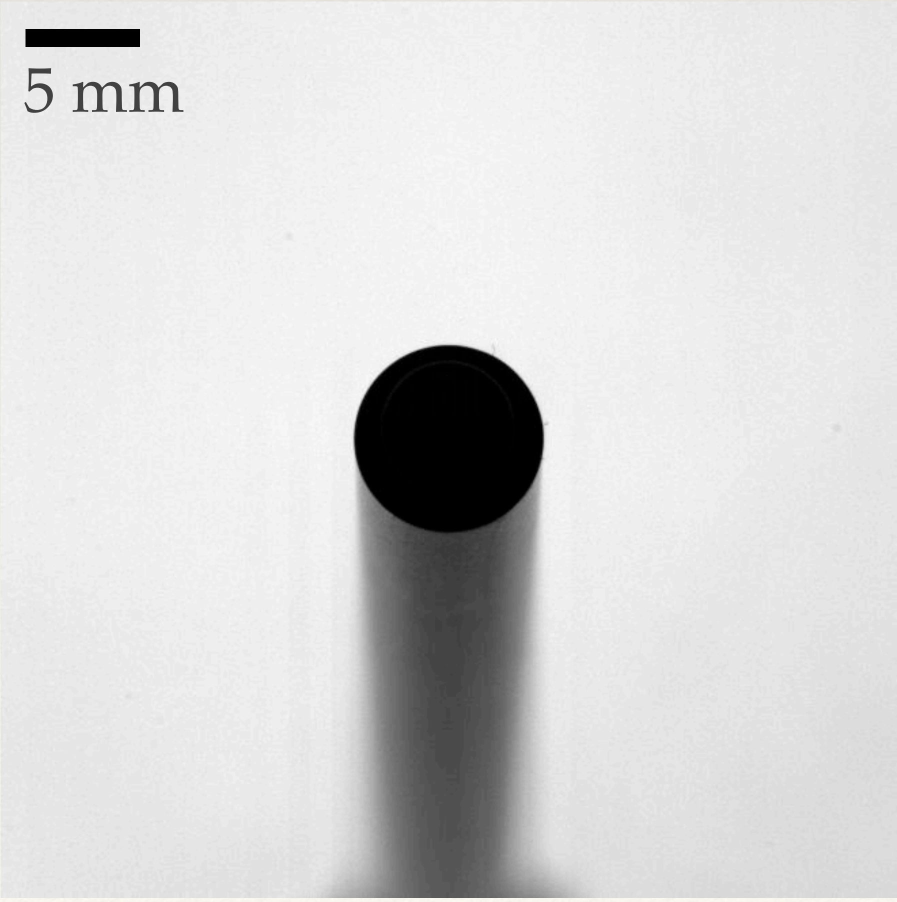
Mathematics, Physics & Machine Learning webinar, May 26th, 2022

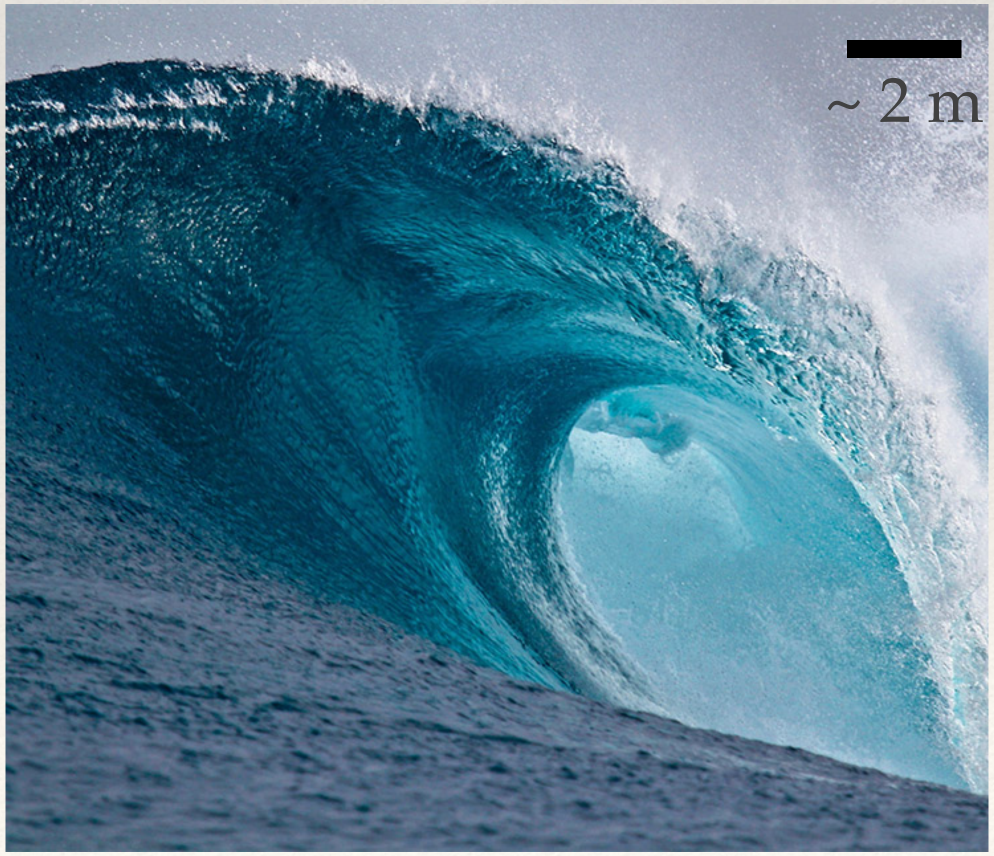
Navier-Stokes equations

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Navier-Stokes equations if

$$\underbrace{\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u}}_{\text{Momentum change}} + \overbrace{\nabla p}^{\text{Pressure}} = \underbrace{\mu \Delta \mathbf{u}}_{\text{Shear stress}}, \quad \operatorname{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$. Here μ is fluid viscosity





Euler equations

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Euler equations if

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \overset{\mathbf{0}}{\cancel{\mu \Delta \mathbf{u}}}, \quad \operatorname{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$.

Open Problem:

Does there exist smooth, finite energy initial condition $\mathbf{u_0}$ *leading to a solution* ***blowing up*** *in finite time?*

?

Euler equations

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Euler equations if

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \overset{\mathbf{0}}{\cancel{\mu \Delta \mathbf{u}}}, \quad \operatorname{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$.

Open Problem:

Does there exist smooth, finite energy initial condition $\mathbf{u_0}$ *leading to a solution blowing up in finite time?*

1-D example

$$\frac{du}{dt} = u$$

$$u(0) = 1$$

Blow-up solutions

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Euler equations if

$$\partial_t \mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \overset{\mathbf{0}}{\cancel{\mu\Delta\mathbf{u}}}, \quad \operatorname{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$.

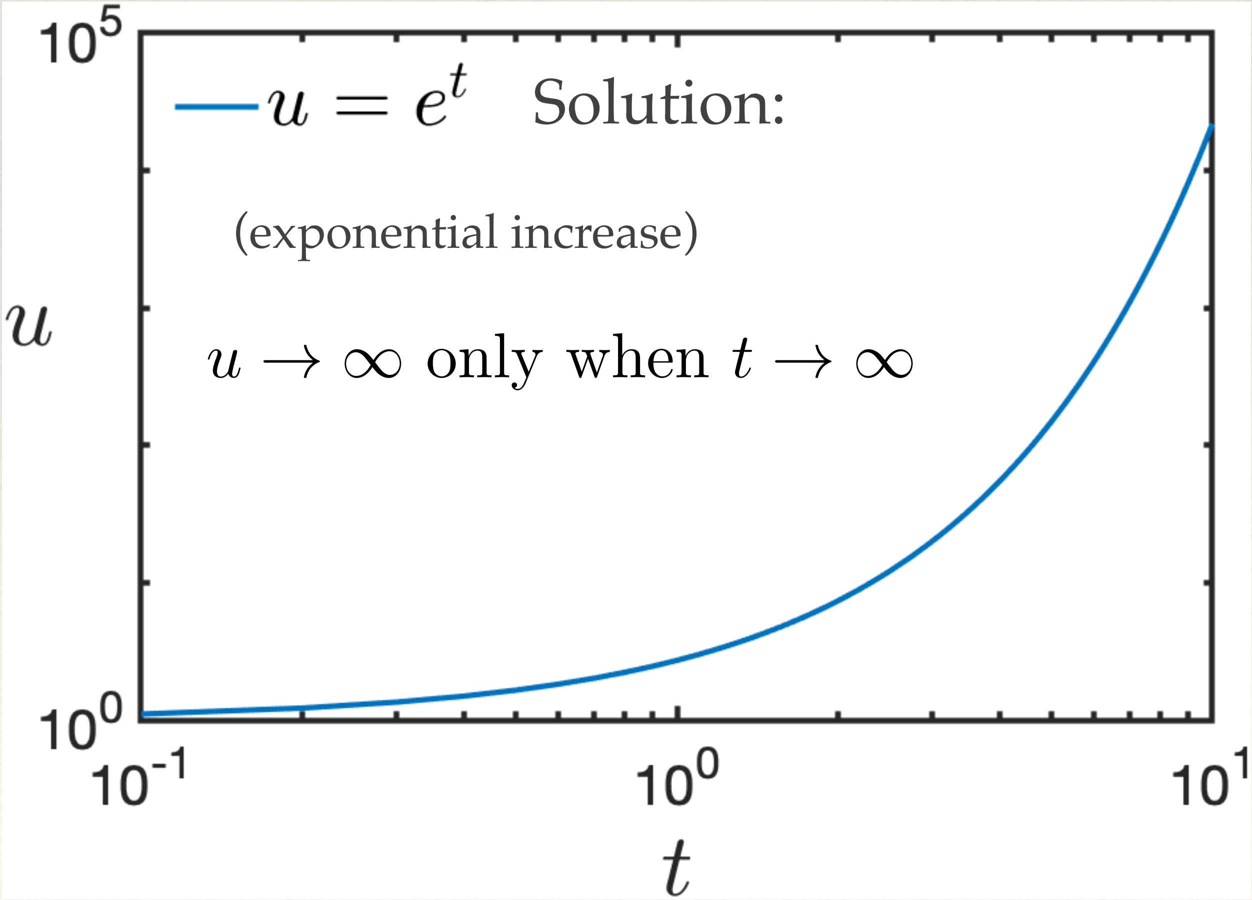
Open Problem:

Does there exist smooth, finite energy initial condition leading to a solution blowing up in finite time?

1-D example

$$\frac{du}{dt} = u$$

$$u(0) = 1$$



Blow-up solutions

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Euler equations if

$$\partial_t \mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \cancelto{\mathbf{0}}{\mu\Delta\mathbf{u}}, \quad \mathrm{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$.

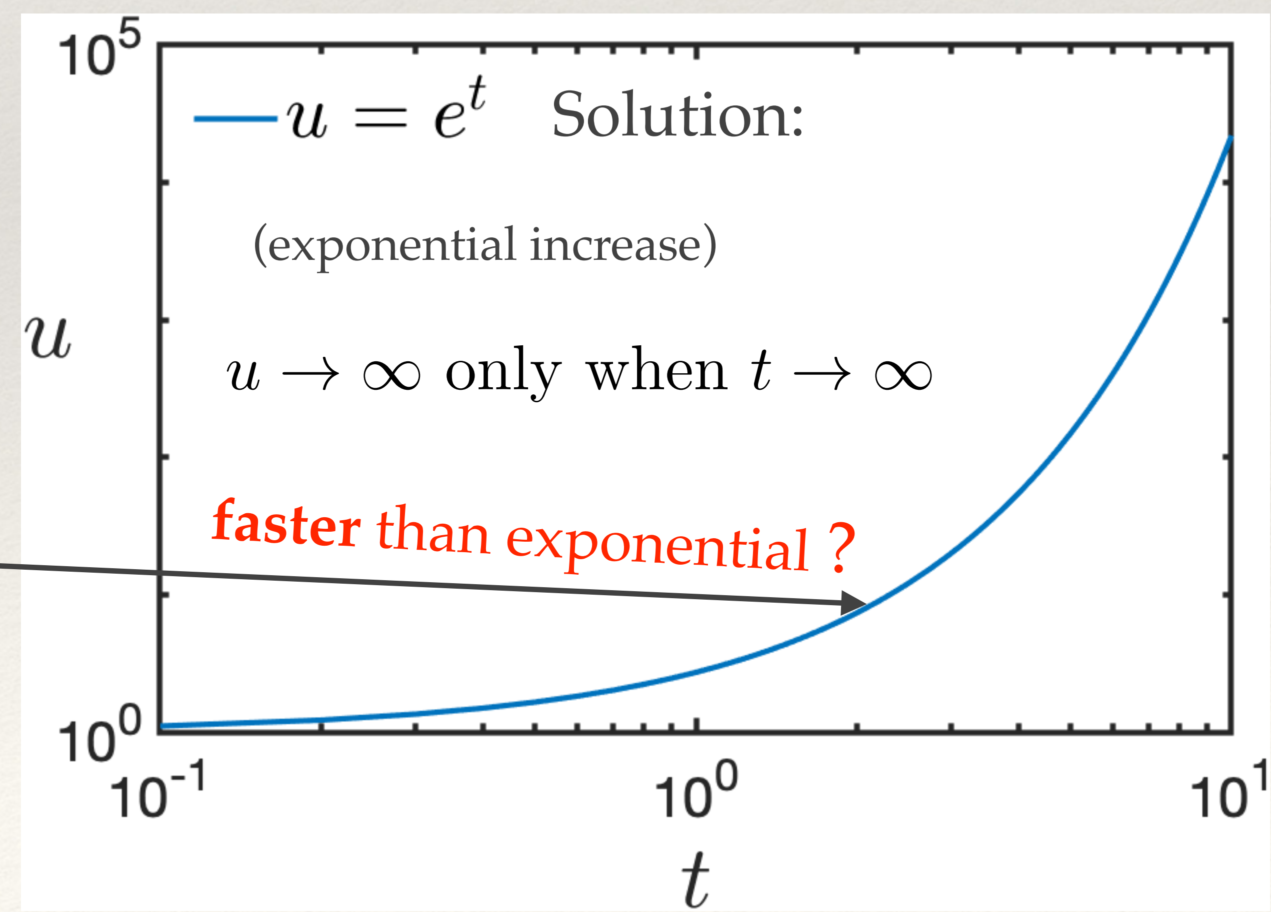
Open Problem:

Does there exist smooth, finite energy initial condition leading to a solution blowing up in finite time?

1-D example

$$\frac{du}{dt} = u^{\boxed{2}}$$

$$u(0) = 1$$



Blow-up solutions

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Euler equations if

$$\partial_t \mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \overset{\mathbf{0}}{\cancel{\mu \Delta \mathbf{u}}}, \quad \mathrm{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$.

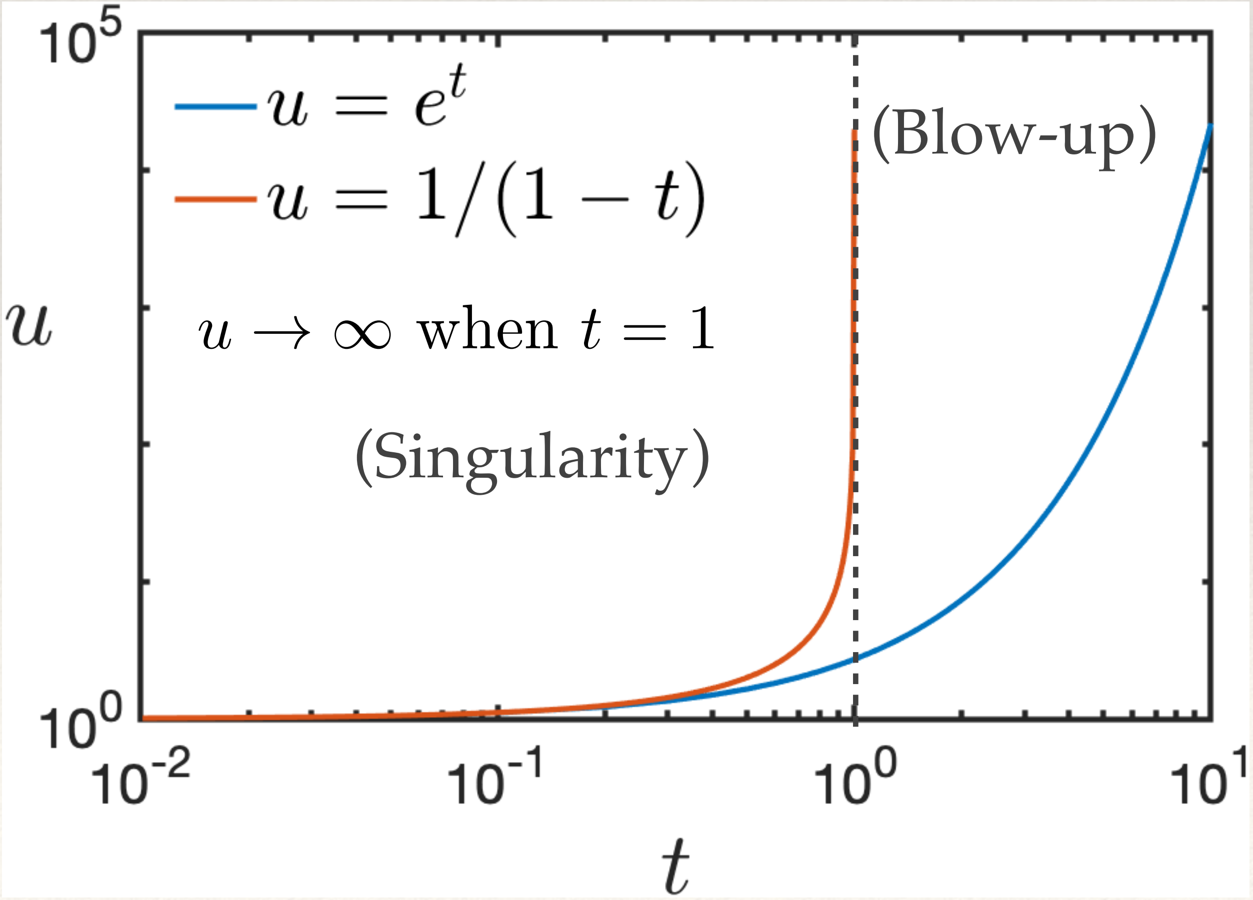
Open Problem:

Does there exist smooth, finite energy initial condition leading to a solution blowing up in finite time?

1-D example

$$\frac{du}{dt} = u^{2}$$

$$u(0) = 1$$



Blow-up solutions

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Euler equations if

0

$$\partial_t \mathbf{u} + \boxed{(\mathbf{u} \cdot \nabla)\mathbf{u}} + \nabla p = \cancel{\mu \Delta \mathbf{u}}, \quad \mathrm{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

Nonlinearity

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$.

Open Problem:

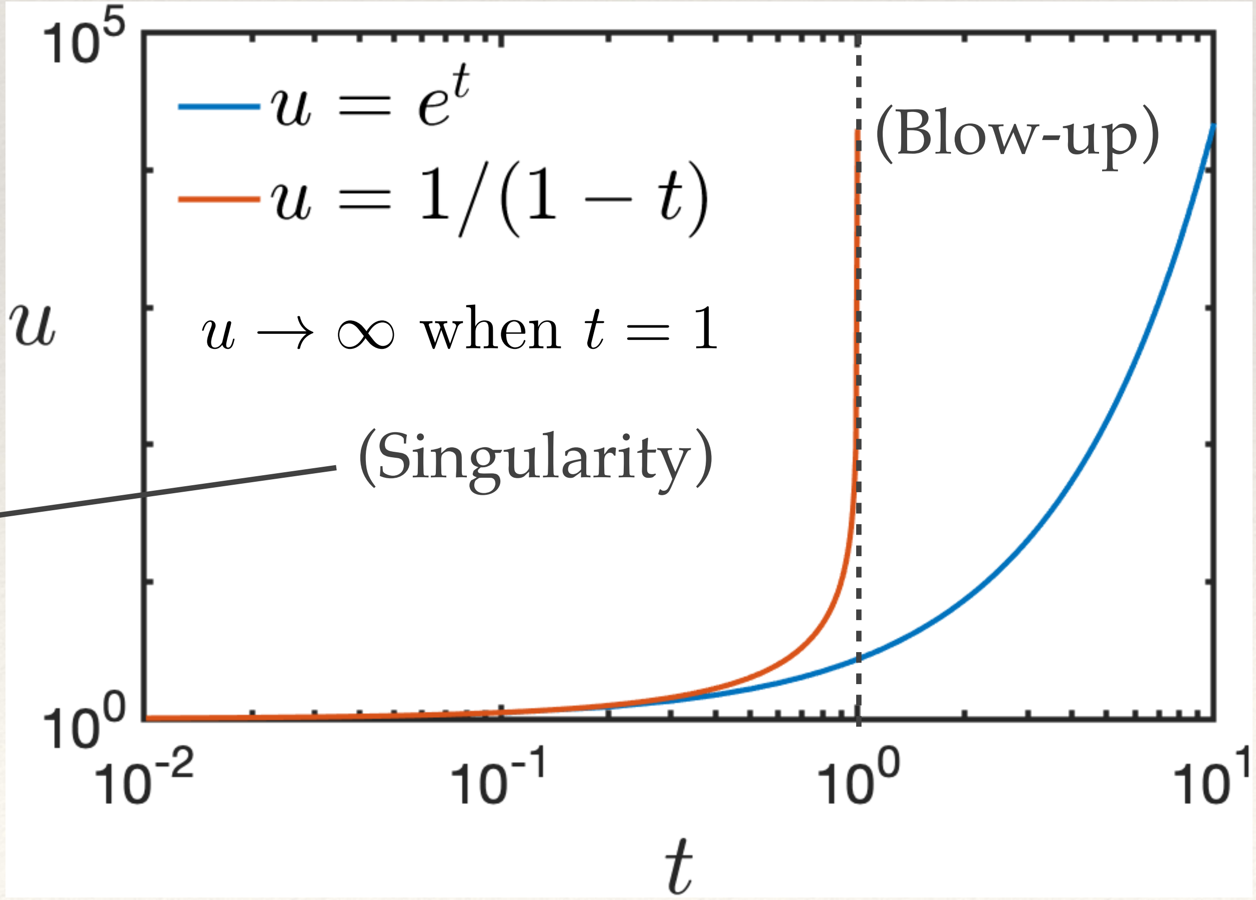
Does there exist smooth, finite energy initial condition leading to a solution blowing up in finite time?

Nonlinearity

1-D example

$$\frac{du}{dt} = \boxed{u^2}$$

$$u(0) = 1$$



Euler equations

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Euler equations if

$$\partial_t \mathbf{u} + \underbrace{\boxed{(\mathbf{u} \cdot \nabla)\mathbf{u}}}_{\text{Nonlinearity}} + \nabla p = \overset{\mathbf{0}}{\cancel{\mu \Delta \mathbf{u}}}, \quad \operatorname{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$.

Open Problem:

Does there exist smooth, finite energy initial condition $\mathbf{u_0}$ *leading to a solution blowing up in finite time?*

If it does exists ⟶ Local velocity goes infinity ✗

Euler equations

The pair $(\mathbf{u}, p)$ solves the incompressible 3-D Euler equations if

$$\partial_t \mathbf{u} + \boxed{(\mathbf{u} \cdot \nabla)\mathbf{u}} + \nabla p = \overset{\mathbf{0}}{\cancel{\mu \Delta \mathbf{u}}}, \quad \operatorname{div}(\mathbf{u}) = 0, \quad \text{and} \quad \mathbf{u}(\cdot, t) = \mathbf{u_0}$$

Nonlinearity

for velocity $\mathbf{u}$, pressure p and initial velocity $\mathbf{u_0}$.

Open Problem:

Does there exist smooth, finite energy initial condition $\mathbf{u_0}$ *leading to a solution blowing up in finite time?*

If it does exists ⟶ Local velocity goes infinity ✗

Numerical challenge: how to find the **blow-up** solution if it exits

Physics-informed neural networks (PINNs)

Raissi *et. al.* (2019), *J Comp.Phys.*, **378**

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Outlines

1. What is Physics-informed Neural Networks (PINNs)

- Basic and key components
- Understand PINNs from the mathematics point of view
- Comparison with classical numerical scheme

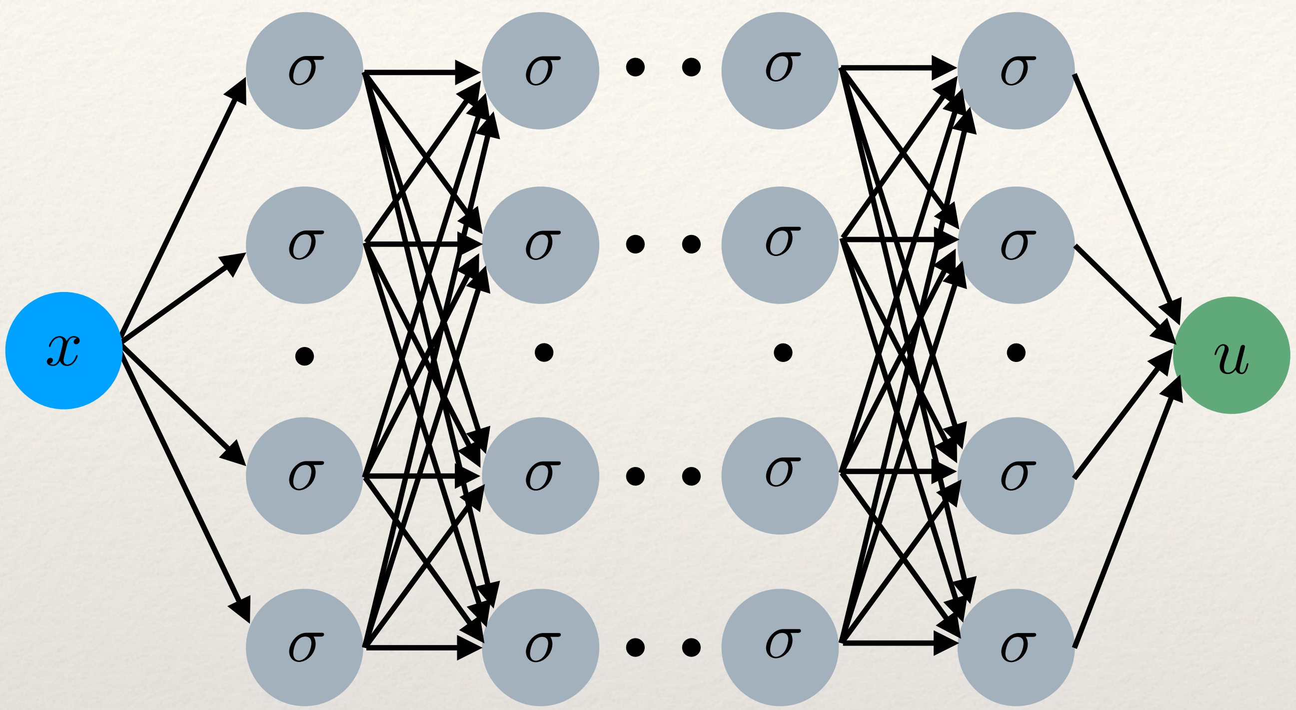
2. Why can PINNs find self-similar blow-up solutions

- Advantages of PINNs over classical numerical scheme
- Steps to set up the PINNs
- Robustness and universality of PINNs

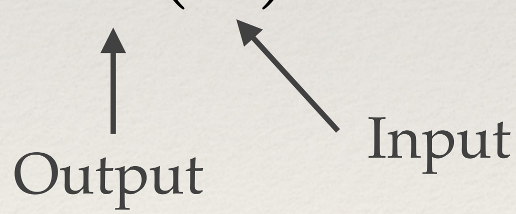
Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Fully-connected Neural network



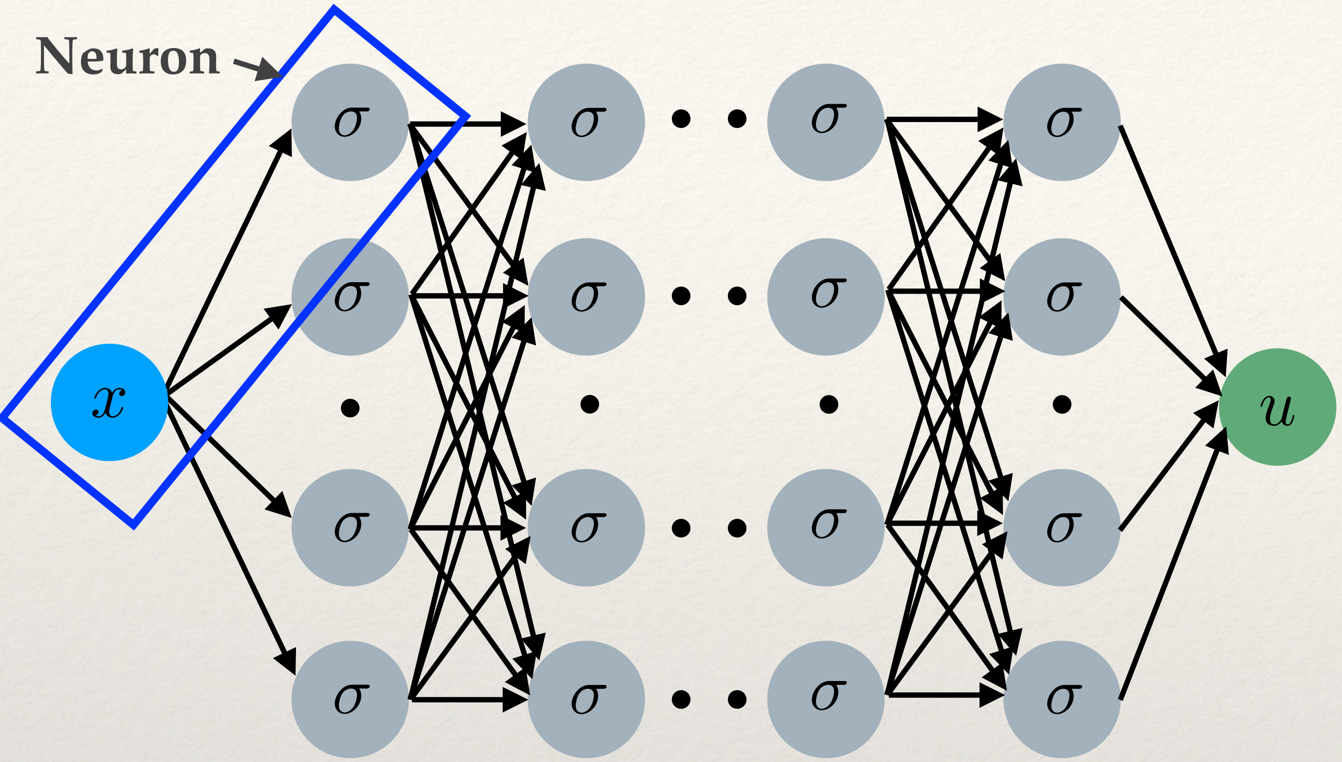
Function $u(x)$



Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Fully-connected Neural network



Function $u(x)$

Weights

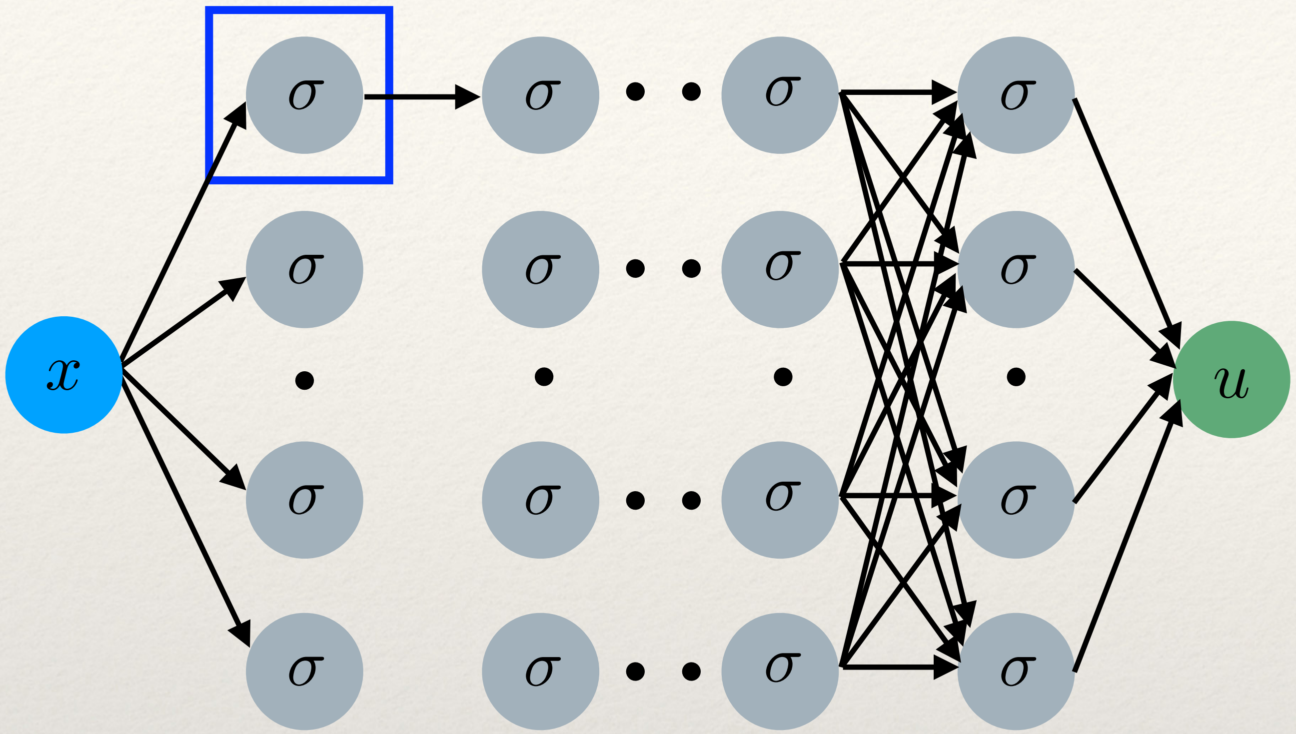
$$w_i^{(0)} x + b_i^{(0)}$$ *(both are free parameter)*

Biases

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Fully-connected Neural network



Activation function (*nonlinearity*)

Function $u(x)$

Common choice

$\sigma(x) = \tanh(x)$

$\sigma(x) = \sin(x)$

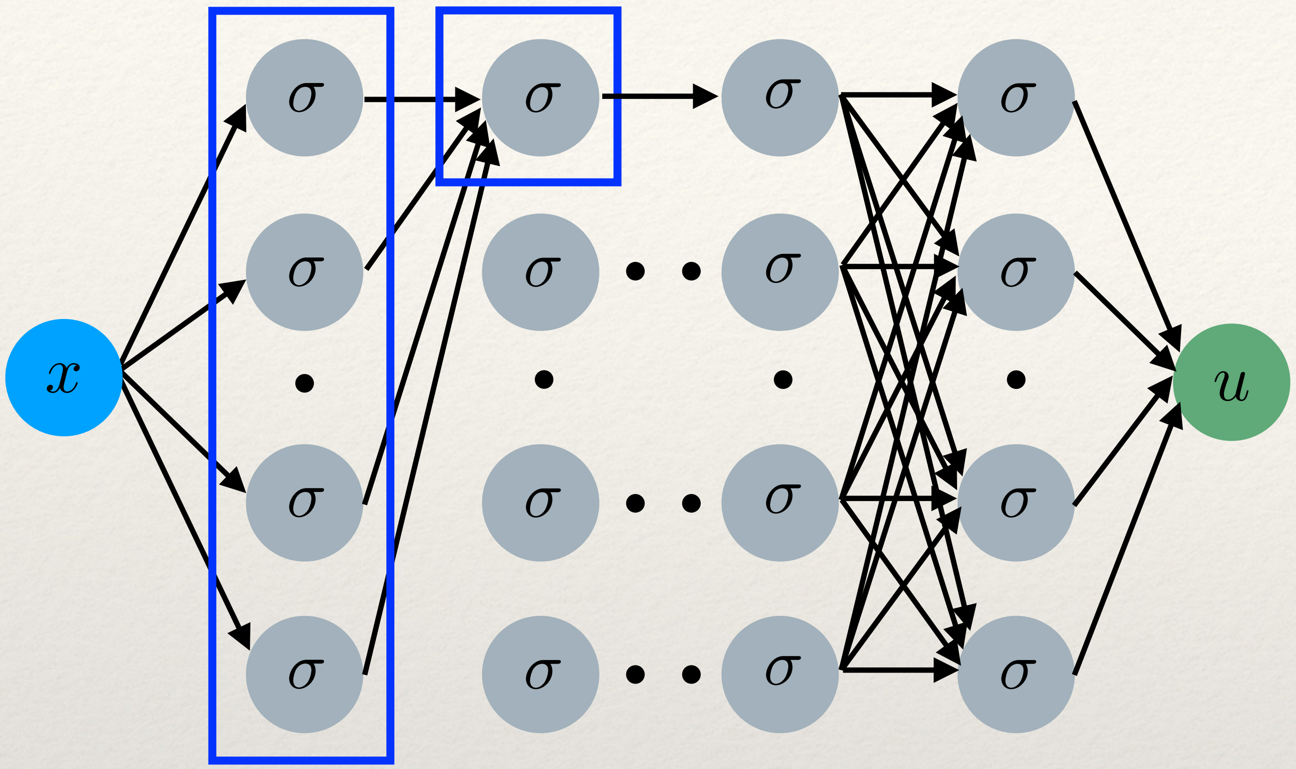
$$\sigma\left(w_i^{(0)} x + b_i^{(0)}\right)$$

Output of a neuron

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Fully-connected Neural network



Function $u(x)$

Sum of the outputs from previous layer

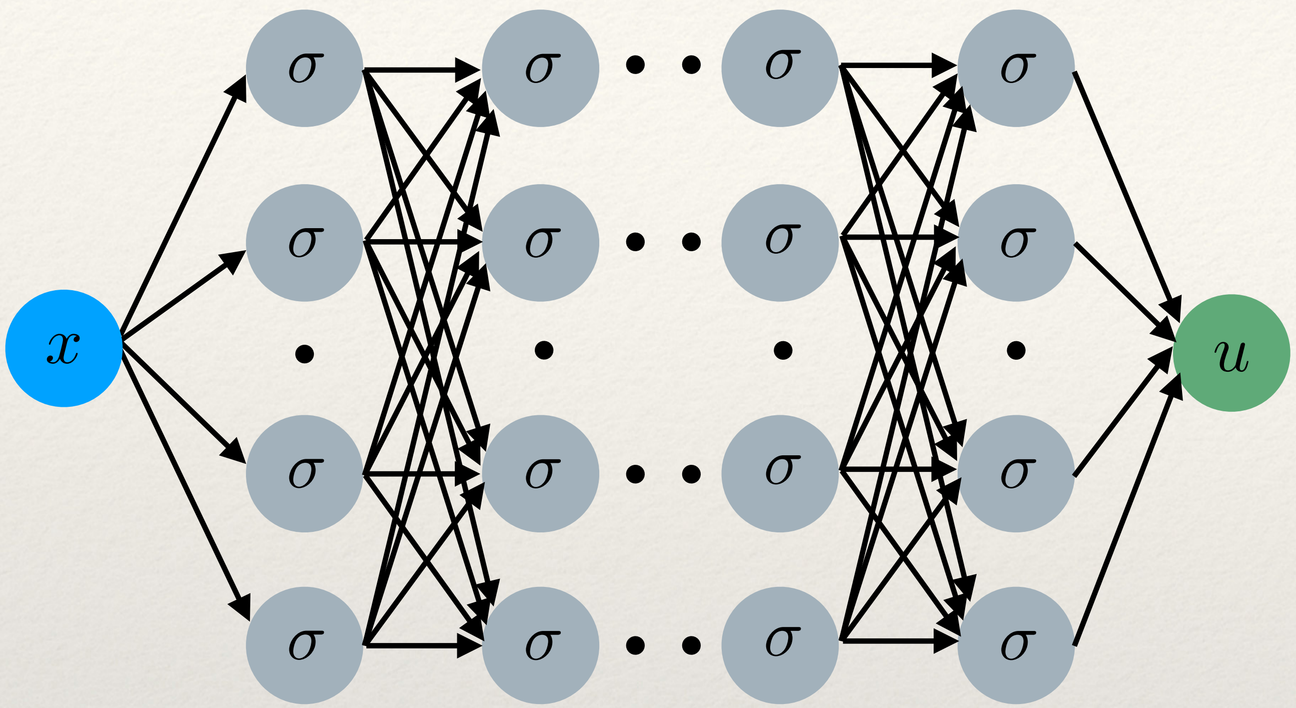
$$\sigma\left(\sum_{i=1} w_{ji}^{(1)}\,\sigma\left(w_i^{(0)}x + b_i^{(0)}\right) + b_j^{(1)}\right)$$

Output of a neuron in the second layer

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Fully-connected Neural network



Function $$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma\left(\sum_{i=1} w_{kj}^{(n-1)} \sigma\left(\dots \sigma\left(\sum_{i=1} w_{ji}^{(1)}\, \sigma\left(w_i^{(0)} x + b_i^{(0)}\right) + b_j^{(1)}\right) \dots\right) + b_k^{(n-1)}\right) + b_l^{(n)}$$

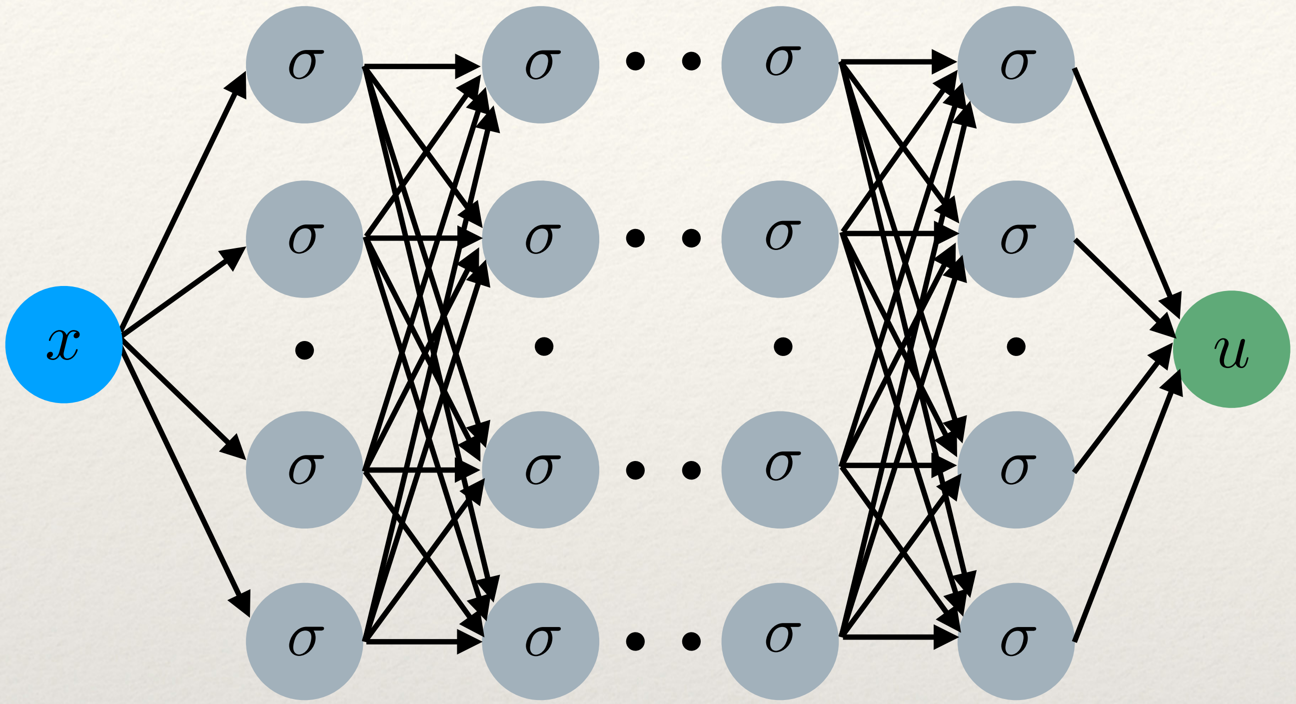
$\mathbf{w}$: weights $\quad$ $\mathbf{b}$: biases $\qquad$ $\sigma(x)$: activation function

(free parameters to be trained) $\qquad$ *(fixed and selected by users)*

Neural network

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Fully-connected Neural network



$$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \, \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_l^{(n)}$$

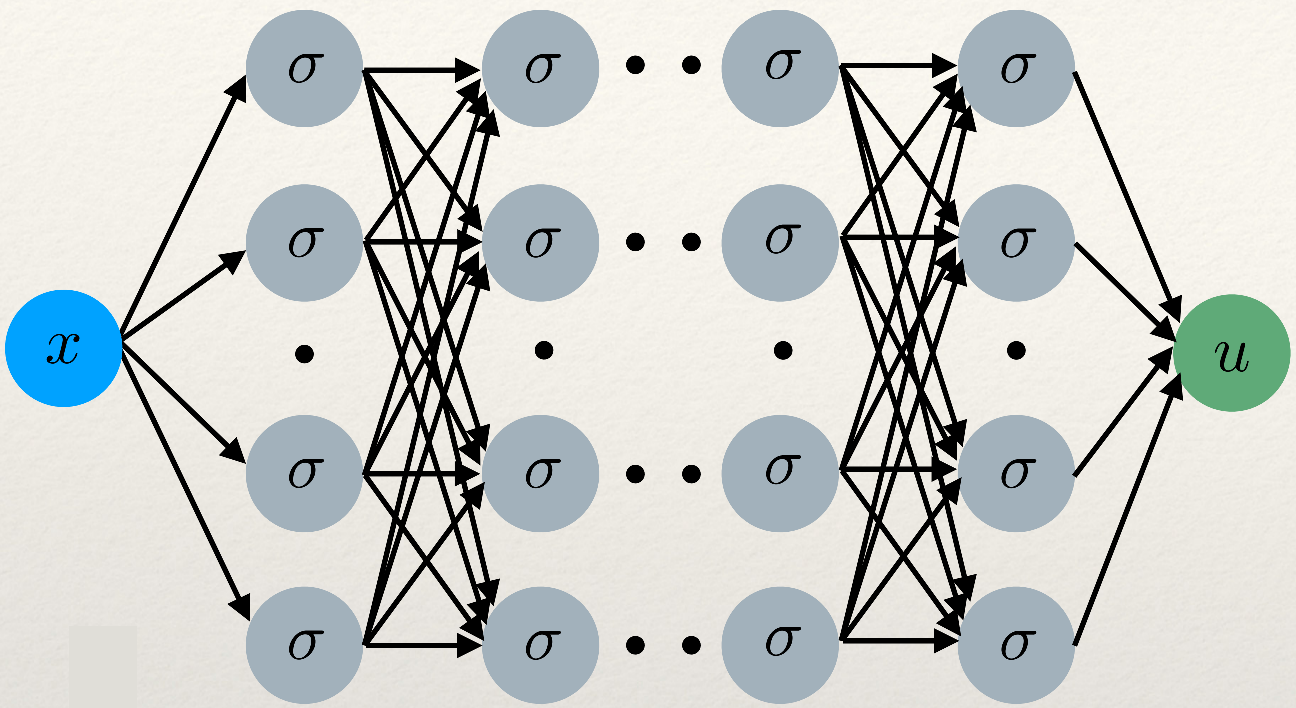
$\mathbf{w}$: weights $\mathbf{b}$: biases $\sigma(x)$: activation function

Universal function approximator

Hornik et. al. (1989), *Neural Netw.* **2**

Neural network

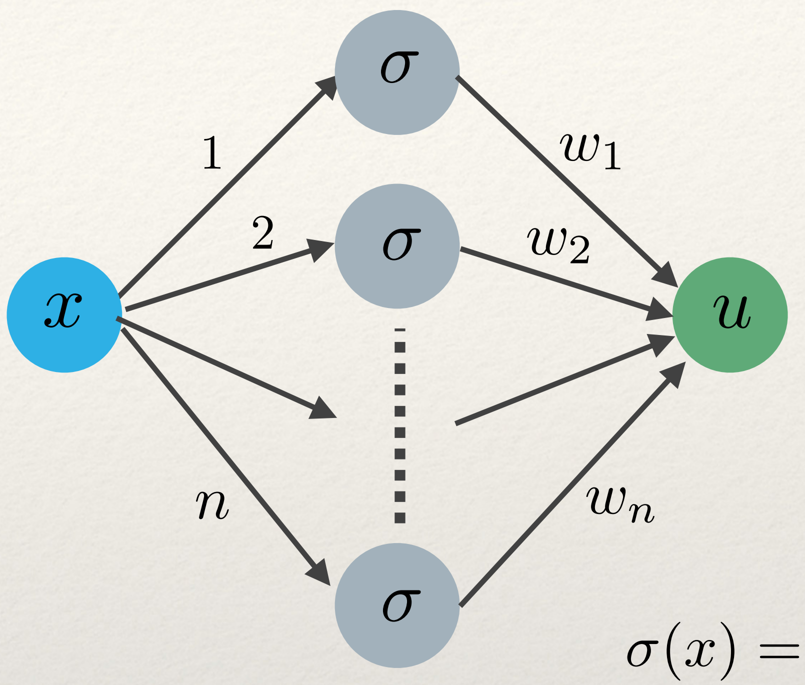
Fully-connected Neural network



$$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \, \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_l^{(n)}$$

$\mathbf{w}$: weights $\mathbf{b}$: biases $\sigma(x)$: activation function

Fourier series: $u(x, w_n, b_n) = \sum_{n=0}^{N} w_n \sin(nx + b_n)$



$\sigma(x) = \sin(x)$

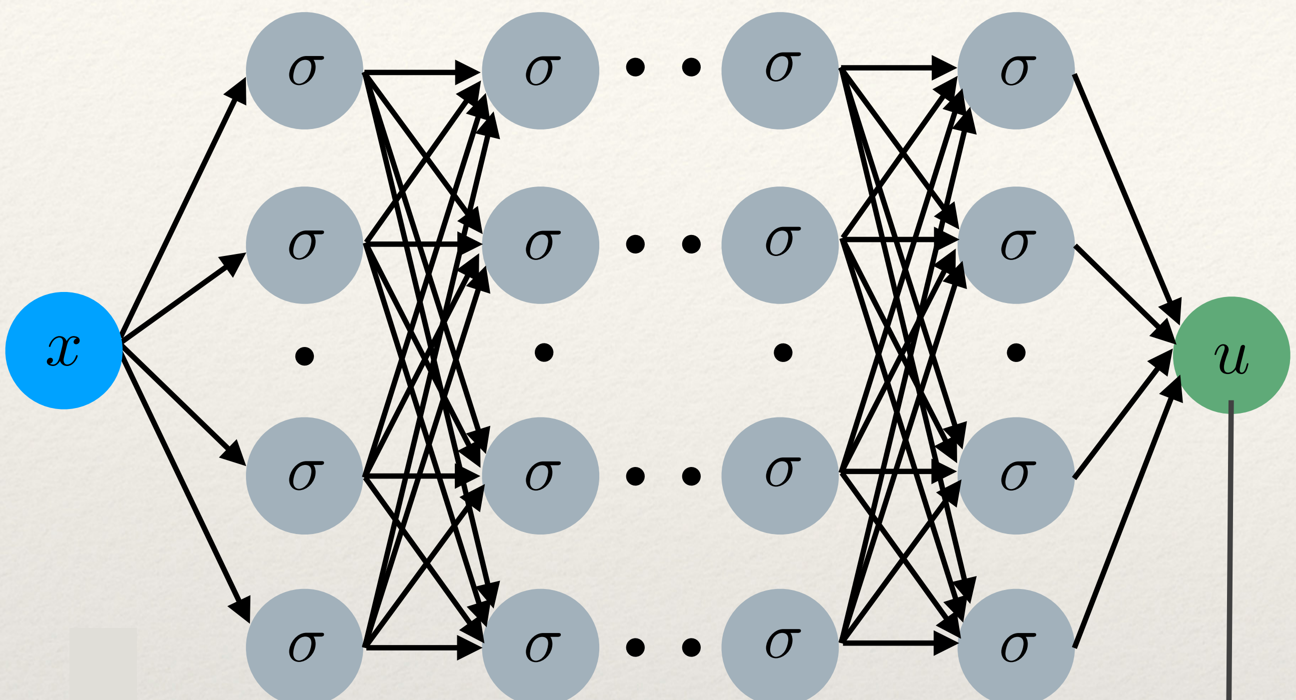
Universal function approximator

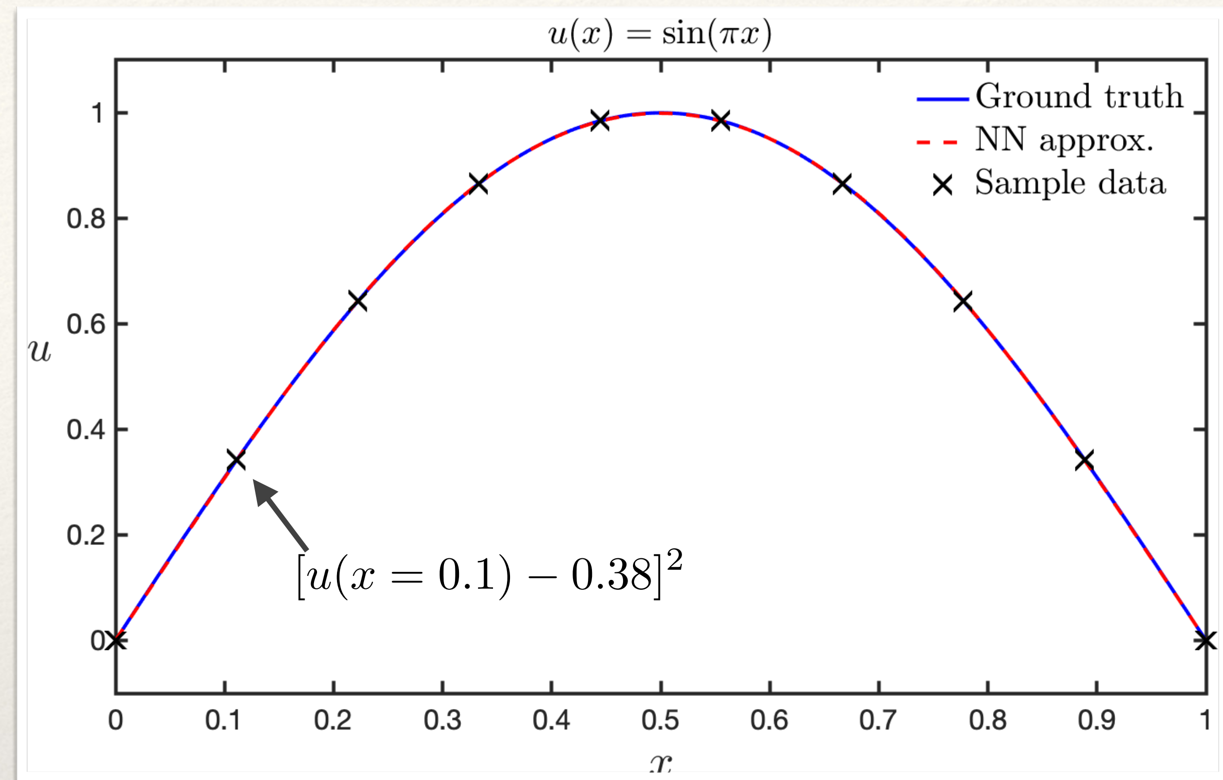
Hornik et. al. (1989), *Neural Netw.* **2**

Neural network for regression

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Fully-connected Neural network





$$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \, \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_l^{(n)}$$

Updating variables: $\mathbf{w}$: weights $\quad \mathbf{b}$: biases

Optimization ← Loss

data of u at $x = x_i$

Cost function: *mean squared error*

$$J(x, \mathbf{w}, \mathbf{b}) = loss_d = \frac{1}{N_d} \sum_{i=1}^{N_d} [u(x_i, \mathbf{w}, \mathbf{b}) - u_i^{(d)}]^2$$

Gradient descent

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \, \nabla_{\mathbf{w}} J(x, \mathbf{w}^{(i)}, \mathbf{b}^{(i)})$$

$$\mathbf{b}^{(i+1)} = \mathbf{b}^{(i)} - \eta \, \nabla_{\mathbf{b}} J(x, \mathbf{w}^{(i)}, \mathbf{b}^{(i)})$$

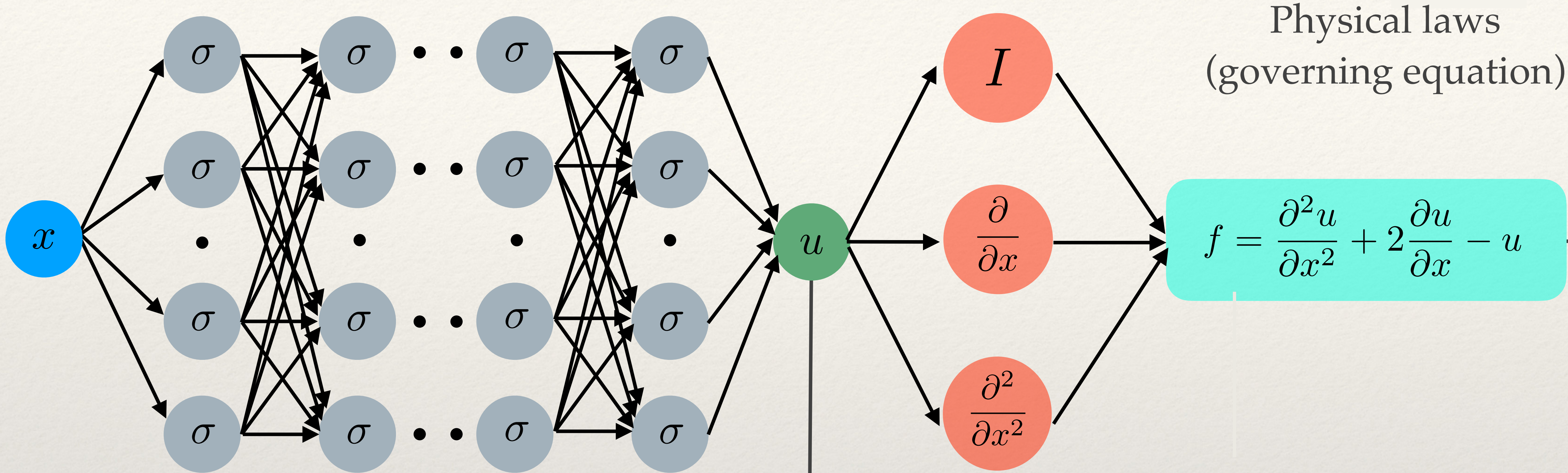
$\mathbf{w}^{(i)}, \mathbf{b}^{(i)}$: value at the i-th iteration $\quad \eta$: learning rate

Physics-informed neural networks

Karniadakis *et. al.* (2021), *Nat. Rev. Phys.*, **3**

Fully-connected Neural network

Physical laws (governing equation)



$$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \, \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_l^{(n)}$$

Updating variables: $\mathbf{w}$: weights $\mathbf{b}$: biases

Data loss: $$loss_d = \frac{1}{N_d} \sum_{i=1}^{N_d} [u(x_i, \mathbf{w}, \mathbf{b}) - u_i^{(d)}]^2$$

Equation loss $$loss_f = \frac{1}{N_f} \sum_{i=1}^{N_f} f^2(x_j, u(x_j, \mathbf{w}, \mathbf{b}))$$

Optimization

Loss

Gradient descent

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \, \nabla_{\mathbf{w}} J(x, \mathbf{w}^{(i)}, \mathbf{b}^{(i)})$$

$$\mathbf{b}^{(i+1)} = \mathbf{b}^{(i)} - \eta \, \nabla_{\mathbf{b}} J(x, \mathbf{w}^{(i)}, \mathbf{b}^{(i)})$$

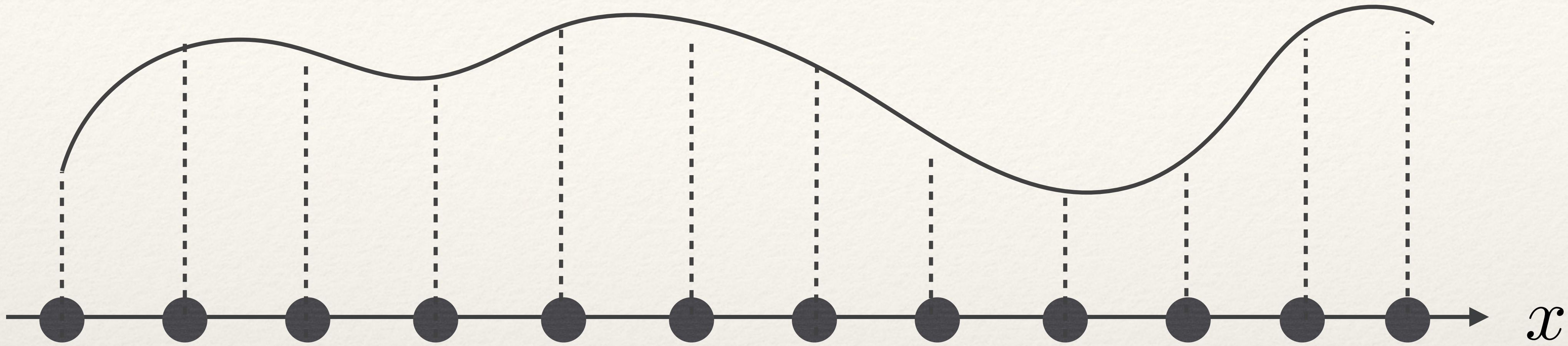
$\mathbf{w}^{(i)}, \mathbf{b}^{(i)}$: value at the i-th iteration $\quad \eta$: learning rate

Cost function: *data + equation loss*

$$J(x, \mathbf{w}, \mathbf{b}) = loss_d + loss_f$$

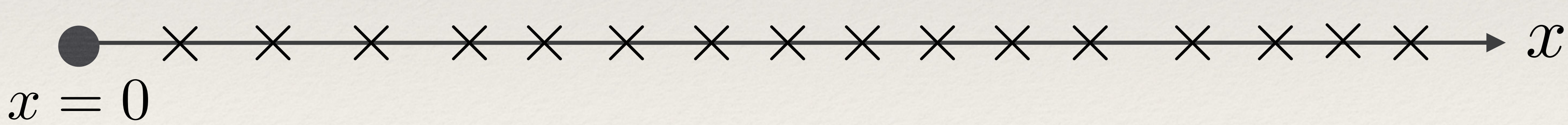
Physics-informed neural networks

Neural network for regression (with data only)



● **Finite** data points (evaluate difference between NN and data)

Physics-informed Neural network $f = \frac{\partial u}{\partial x} - u$ **Differential equation solver**



× **Infinite** collocation points (evaluate equation balance)

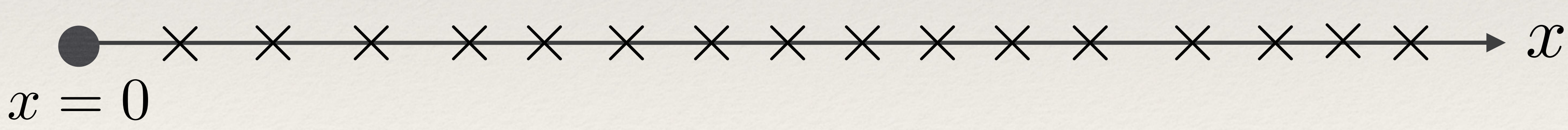
● Only one data point (boundary condition i. e. $u(0) = 1$)

Physics-informed neural networks

How does PINN evaluate the differential equation?

How does it different from classical numerical method?

Physics-informed Neural network $f = \frac{\partial u}{\partial x} - u$ **Differential equation solver**



× **Infinite** collocation points (evaluate equation balance)

● Only one data point (boundary condition i. e. $u(0) = 1$)

Differential equations

(derivatives)

Difficulty $$\boxed{\frac{du}{dx}} = u$$

Differential equations

Numeric

PINNs

↓

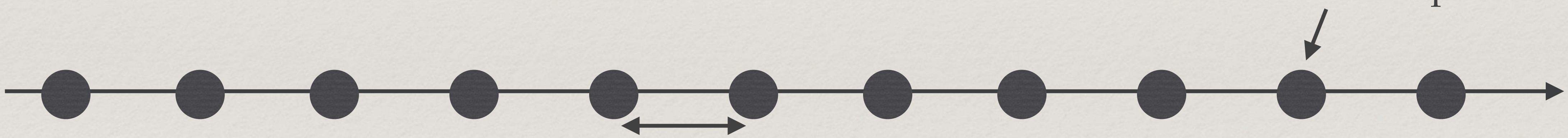
Algebraic equations

Numerical method

$$\frac{du}{dx} = u$$ Boundary condition $$u(0) = 1$$

Differential equation ⟶ Algebraic equation

Finite difference



$$\frac{du(x_{n-1})}{dx} \approx \frac{u_n - u_{n-1}}{h}$$

Finite difference

$$\frac{du}{dx} = u \quad \Longrightarrow \quad \frac{u_n - u_{n-1}}{h} = u_{n-1}$$

algebraic equations

$$u_n = u(x_n) \qquad x_n = x_{n-1} - h$$

Output: value at each discretized points

Differential equations

(derivatives)

Difficulty $$\boxed{\frac{du}{dx}} = u$$

Differential equations

Numeric

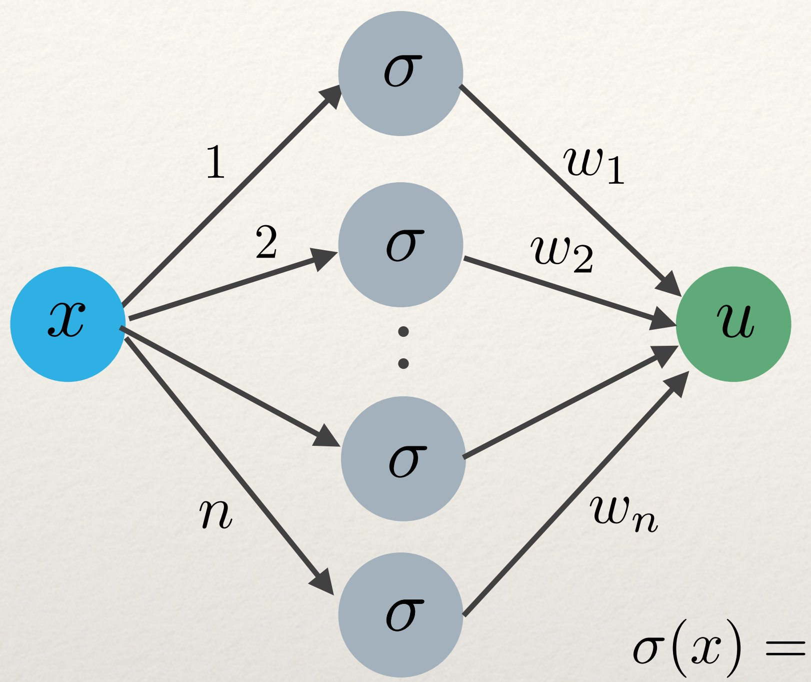
(Finite difference)

PINNs

Algebraic equations

Physics-informed neural networks

Fourier series: 1-hidden layer network



$\sigma(x) = \sin(x)$

$$u(x) = \sum_{n=0}^{N} w_n \sin\left(\frac{2\pi}{L} n x + b_n\right)$$

Elementary base function: sin(x)

$$\frac{du}{dx}(x) = \sum_{n=0}^{N} \frac{2\pi}{L} n w_n \cos\left(\frac{2\pi}{L} n x + b_n\right)$$

Explicit expression for its **exact** derivative

↓

Evaluate the derivative
(no truncation error)

$$\frac{du(x_{n-1})}{dx} \approx \frac{u_n - u_{n-1}}{h}$$

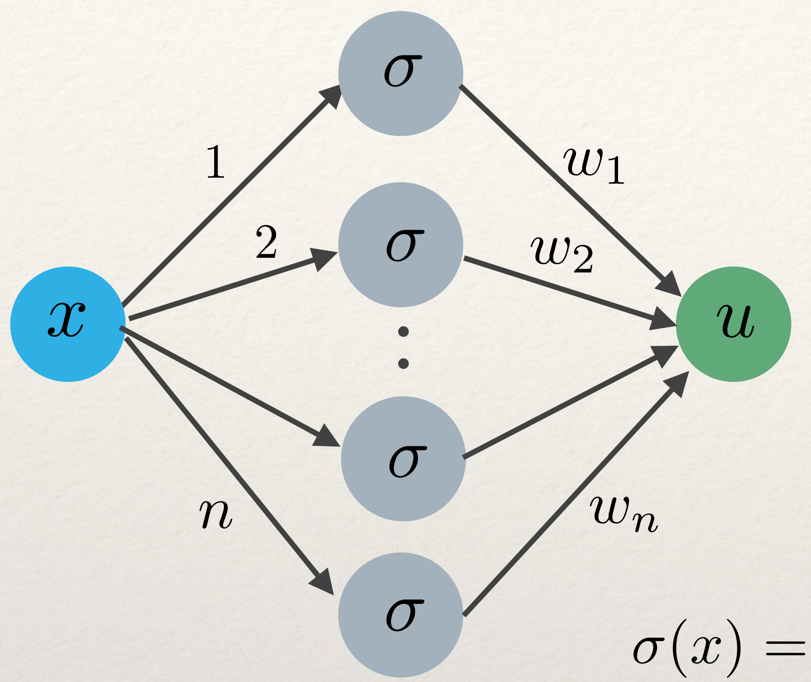
↑

Truncation error

Output: a continuous function

Physics-informed neural networks

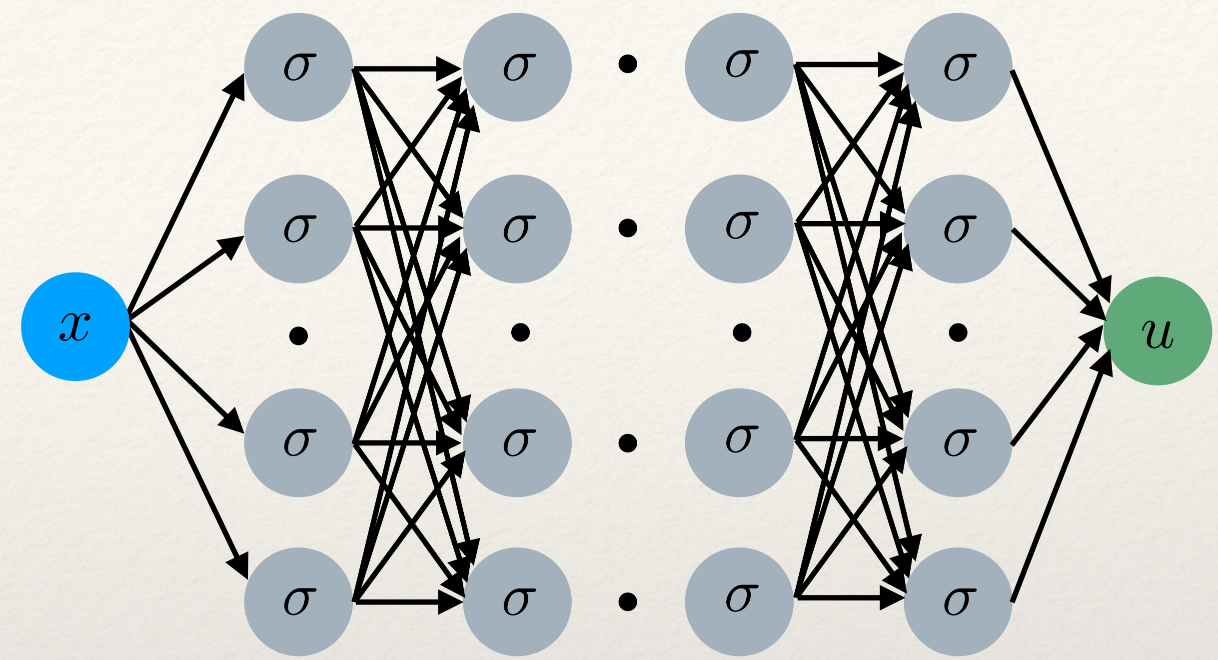
Fourier series: $u(x) = \sum_{n=0}^{N} w_n \sin\left(\frac{2\pi}{L} nx + b_n\right)$



$\sigma(x) = \sin(x)$

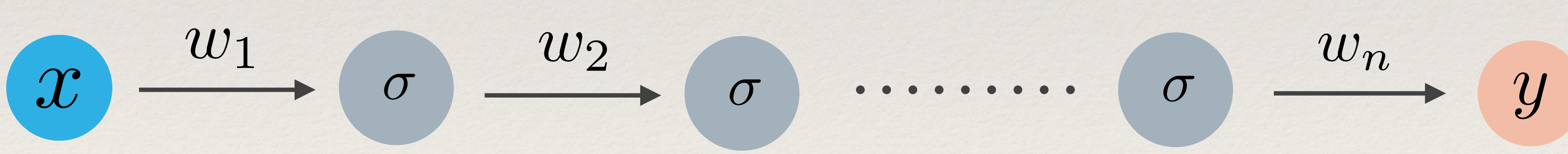
1-hidden layer network

Multi-layer Neural network



$$u(x) = \sum_{j=1} w_{lk}^{(n)} \sigma \left(\dots \sigma \left(\sum_{i=1} w_{ji}^{(1)} \sigma \left(w_i^{(0)} x + b_i^{(0)} \right) + b_j^{(1)} \right) \dots \right) + b_l^{(n)}$$

$\mathbf{w}$: weights $\mathbf{b}$: biases $\sigma(x)$: activation function

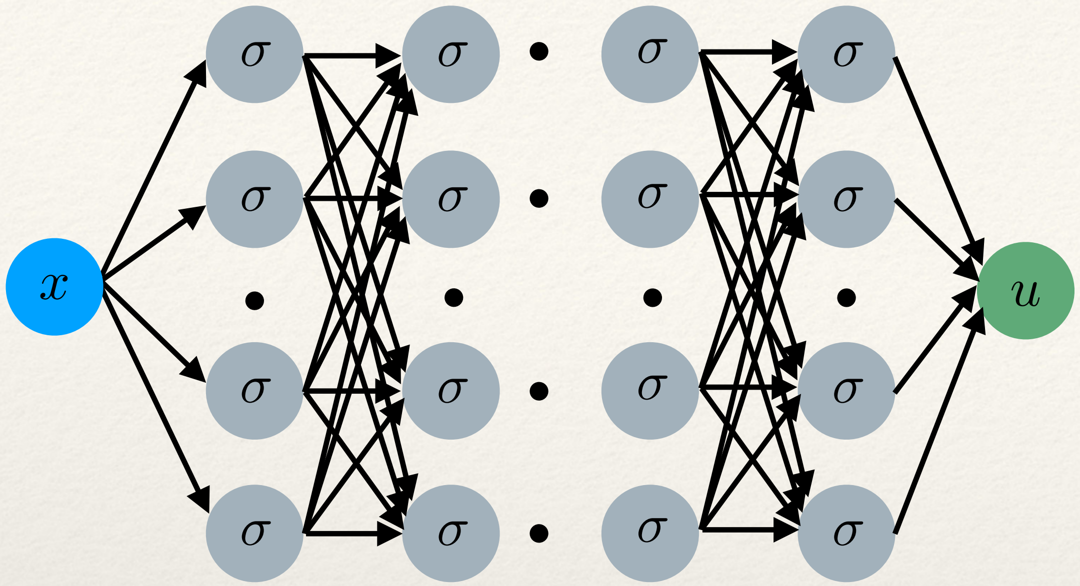


$a_1 = \sigma(w_1 \cdot x)$ $a_2 = \sigma(w_2 \cdot a_1)$ $a_{n-1} = \sigma(w_{n-1} \cdot a_{n-2})$ $y = w_n a_{n-1}$

Chain rule $\frac{dy}{dx} = \boxed{\frac{dy}{da_{n-1}}} \cdot \boxed{\frac{da_{n-1}}{da_{n-2}}} \dots \boxed{\frac{da_2}{da_1}} \cdot \boxed{\frac{da_1}{dx}}$ ← *each derivative is known exactly*

Automatic differentiation

Comparison between two methods



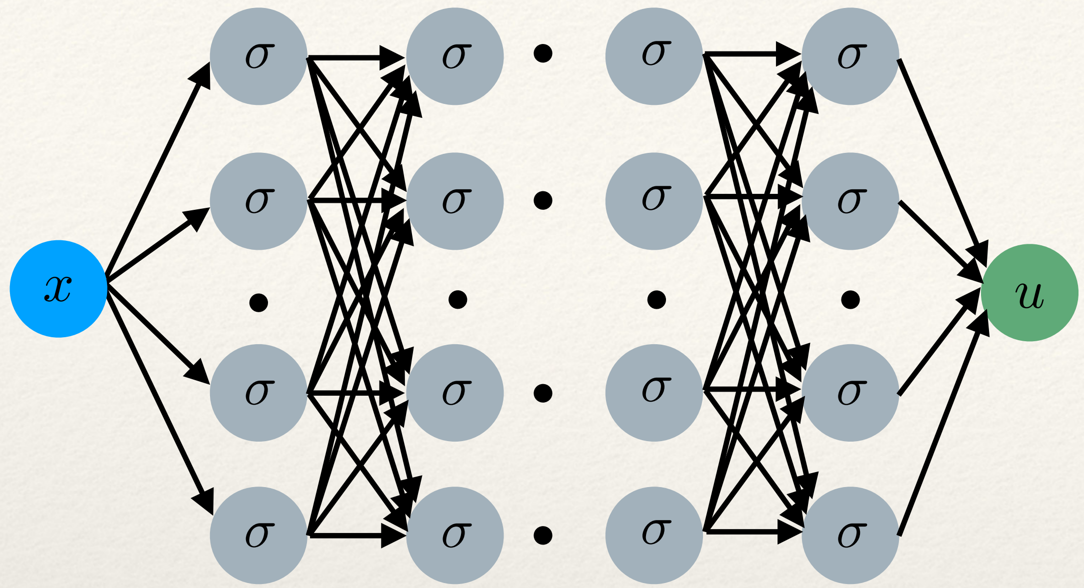
Classical numerical scheme

$$\frac{du(x_{n-1})}{dx} \approx \frac{u_n - u_{n-1}}{h}$$

Truncation error

PINNs	Numerical
Automatic differentiation	Finite difference
No truncation error	Has truncation error
Continuous function	Discretized points

Comparison between two methods



Classical numerical scheme

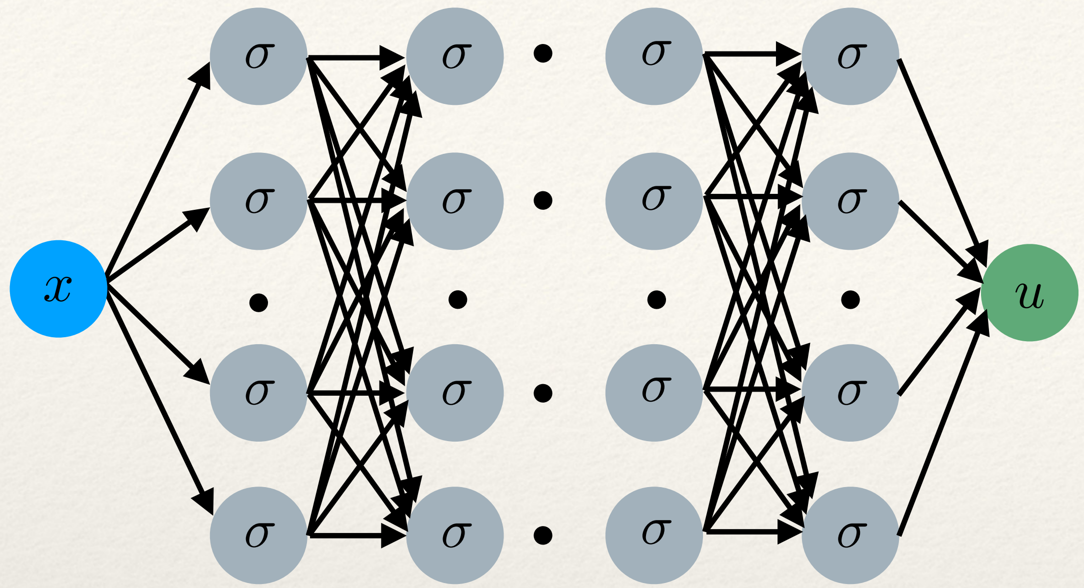
$$\frac{du(x_{n-1})}{dx} \approx \frac{u_n - u_{n-1}}{h}$$

Truncation error

PINNs	Numerical
Automatic differentiation	Finite difference
No truncation error	Has truncation error
Continuous function	Discretized points
Trapped in local minimal	Fast convergence rate
Higher computational cost	Computational efficient

$O(\text{min})$	*For a linear ODE*	$O(0.1)$ sec
$O(\text{hour})$	*For a linear PDE*	$O(10)$ sec

Comparison between two methods



Classical numerical scheme

$$\frac{du(x_{n-1})}{dx} \approx \frac{u_n - u_{n-1}}{h}$$

Truncation error

PINNs	Numerical
Automatic differentiation	Finite difference
No truncation error	Has truncation error
Continuous function	Discretized points
Trapped in local minimal	Fast convergence rate
Higher computational cost	Computational efficient
Newly-developed method	Well-developed and documented

Why is PINN able to find self-similar blow-up solutions?

Incompressible Euler equation

The pair (u, p) solves the incompressible 3-D Euler equations if

$$\partial_t u + (u \cdot \nabla) u + \nabla p = 0, \quad \mathrm{div}(u) = 0, \quad \text{and} \quad u(\cdot, t) = u_0$$

for velocity u, pressure p and initial velocity u_0.

Open Problem:

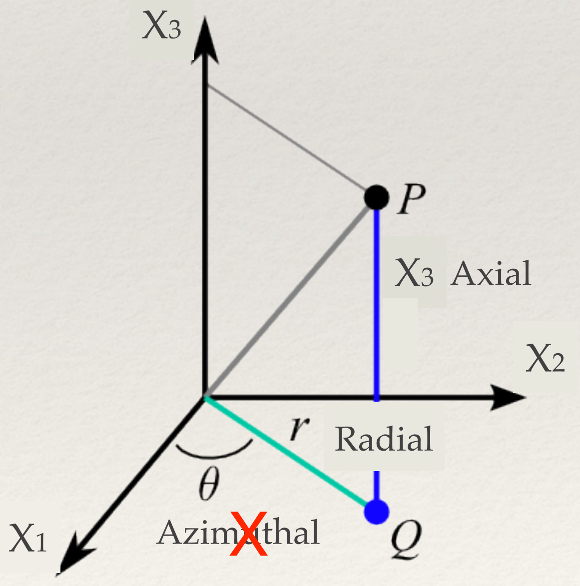
Does there exist smooth, finite energy initial data u_0 *leading to a singularity in finite time?*

Under axi-symmetry, the equations become

$$\left(\partial_t + u_r \partial_r + u_3 \partial_{x_3}\right)\left(\frac{\omega_\theta}{r}\right) = \frac{1}{r^4}\partial_{x_3}\left(r u_\theta\right)^2$$

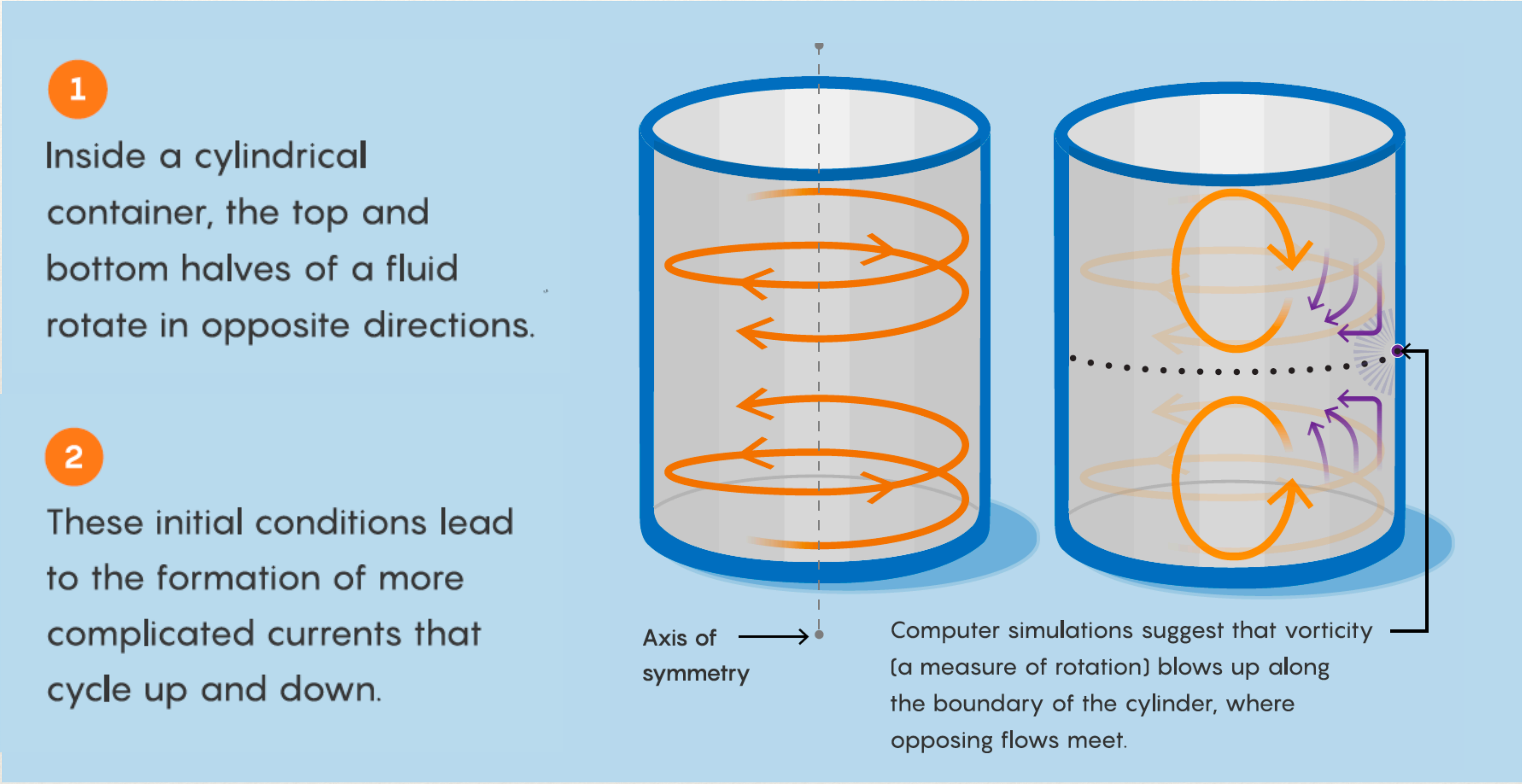
$$\left(\partial_t + u_r \partial_r + u_3 \partial_{x_3}\right)\left(r u_\theta\right) = 0$$

$$\partial_r u_r + \frac{u_r}{r} + \partial_{x_3} u_3 = 0 \qquad \omega_\theta = \partial_{x_3} u_r - \partial_r u_3$$



where (u_r, u_θ, u_3) is the velocity in cylindrical coordinates and ω_θ is the angular component of the vorticity (curl of the velocity).

Luo-Hou Scenario



Luo-Huo '14 provided compelling numerical evidence for singularity formation in this setting (growth by a factor of 3×10^8). The numerics suggest an asymptotic self-similar scaling at the time of singularity.

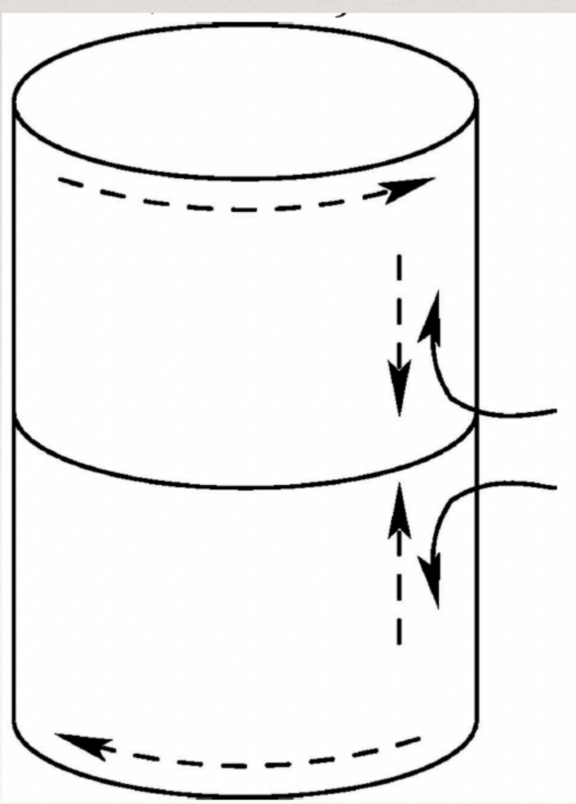
Self-similar Euler equation with boundary

Considering the Euler exterior to the cylindrical boundary

$$(u_r, u_3) = (1-t)^{\lambda}\mathbf{U}(\mathbf{y}, s) = (1-t)^{\lambda}(U_1(\mathbf{y}, s),\ U_2(\mathbf{y}, s)),$$
$$\omega_\theta = (1-t)^{-1}\Omega(\mathbf{y}, s), \quad \partial_r\,(ru_\theta)^2 = (1-t)^{-2}\Psi(\mathbf{y}, s),$$
$$\partial_{x_3}\,(ru_\theta)^2 = (1-t)^{-2}\Phi(\mathbf{y}, s)$$

For self-similar coordinates

$$\mathbf{y} = (y_1, y_2) = \frac{(x_3, r-1)}{(1-t)^{1+\lambda}}, \quad s = -\log(1-t)$$



Self-similar Euler equation with boundary

Considering the Euler exterior to the cylindrical boundary

$$(u_r, u_3) = (1-t)^{\lambda}\mathbf{U}(\mathbf{y}, s) = (1-t)^{\lambda}(U_1(\mathbf{y}, s),\ U_2(\mathbf{y}, s)),$$
$$\omega_\theta = (1-t)^{-1}\Omega(\mathbf{y}, s), \quad \partial_r (ru_\theta)^2 = (1-t)^{-2}\Psi(\mathbf{y}, s),$$
$$\partial_{x_3} (ru_\theta)^2 = (1-t)^{-2}\Phi(\mathbf{y}, s)$$

For self-similar coordinates

$$\mathbf{y} = (y_1, y_2) = \frac{(x_3, r-1)}{(1-t)^{1+\lambda}}, \quad s = -\log(1-t)$$

We obtain the self-similar equations

$$(\partial_s + 1)\Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Omega = \Phi + \mathcal{E}_1$$
$$(\partial_s + 2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi = -\partial_{y_1} U_2 \Psi$$
$$(\partial_s + 2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi = -\partial_{y_2} U_1 \Phi$$
$$\Omega = \partial_{y_1} U_2 - \partial_{y_2} U_1 \qquad \operatorname{div} \mathbf{U} = \mathcal{E}_2$$

Exist at least one λ, equations have **smooth** and **finite energy** solutions

Self-similar equation for Euler

$$\boxed{\mathcal{E}_1} = -y_2 e^{-(1+\lambda)s} \frac{(y_2 e^{-(1+\lambda)s} + 2)(y_2^2 e^{-2(1+\lambda)s} + 2y_2 e^{-(1+\lambda)s} + 2)}{(1 + y_2 e^{-(1+\lambda)s})^4} \Phi$$

$$\boxed{\mathcal{E}_2} = -e^{-(1+\lambda)s} \frac{U_2}{1 + y_2 e^{-(1+\lambda)s}}$$

where $s = -\log(1-t) \longrightarrow -\infty$

So long as $\lambda > -1$ then these errors act like decaying forcing.

$$\Downarrow$$

$$(\partial_s + 1)\Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Omega = \Phi + \boxed{\mathcal{E}_1}$$

$$(\partial_s + 2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi = -\partial_{y_1} U_2 \Psi$$

$$(\partial_s + 2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi = -\partial_{y_2} U_1 \Phi$$

$$\Omega = \partial_{y_1} U_2 - \partial_{y_2} U_1 \qquad \operatorname{div} \mathbf{U} = \boxed{\mathcal{E}_2}$$

Euler blow-up = Bousinessq blow-up

$$\mathcal{E}_1 = -y_2 e^{-(1+\lambda)s} \frac{(y_2 e^{-(1+\lambda)s} + 2)(y_2^2 e^{-2(1+\lambda)s} + 2y_2 e^{-(1+\lambda)s} + 2)}{(1 + y_2 e^{-(1+\lambda)s})^4} \Phi$$

$$\mathcal{E}_2 = -e^{-(1+\lambda)s} \frac{U_2}{1 + y_2 e^{-(1+\lambda)s}} \qquad \text{where} \quad s = -\log(1-t)$$

So long as $\lambda > -1$ then these errors act like decaying forcing.

$$\Downarrow$$

$$(\partial_s + 1)\Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Omega = \Phi + \cancel{\mathcal{E}_1}$$

$$(\partial_s + 2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Phi = -\partial_{y_1} U_2 \Psi$$

$$(\partial_s + 2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Psi = -\partial_{y_2} U_1 \Phi$$

$$\Omega = \partial_{y_1} U_2 - \partial_{y_2} U_1 \qquad \text{div}\, \mathbf{U} = \cancel{\mathcal{E}_2}\ 0$$

$$\Uparrow$$

Equal to the self-similar equations for the 2-D **Bousinessq** equations

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = (0, \theta), \quad \text{div}(\mathbf{u}) = 0 \quad \text{and} \quad \partial_t \theta + \mathbf{u} \cdot \nabla \theta = 0$$

Self-similar equations

Steady self-similar equations for axisymmetric Euler with boundary (Bousinessq)

$$\Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Omega = \Phi$$

$$(2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi = -\partial_{y_1} U_2 \Psi$$

$$(2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi = -\partial_{y_2} U_1 \Phi$$

$$\Omega = \partial_{y_1} U_2 - \partial_{y_2} U_1 \qquad \text{div } \mathbf{U} = 0$$

In addition, we impose

6. Solution smooth everywhere

1. U_1, Φ, Ω are odd in y_1
2. U_2, Ψ are even in y_1 → Symmetry of the solutions (1 and 2)
3. $U_2(y_1, 0) = 0$ → No-penetration condition
4. $\partial_{y_1}\Omega(0) = -1$ → Rescaling constraint
5. $\nabla\mathbf{U}$, Φ and Ψ all vanish at infinity → Finite energy

Challenges to numerical method

Steady self-similar equations for axisymmetric Euler with boundary (Bousinessq)

$$\Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Omega = \Phi$$
$$(2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi = -\partial_{y_1} U_2 \Psi$$
$$(2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi = -\partial_{y_2} U_1 \Phi$$
$$\Omega = \partial_{y_1} U_2 - \partial_{y_2} U_1 \qquad \text{div}\,\mathbf{U} = 0$$

Two big challenges:

to be determined by the constraint of solution

1. Governing equation involves **unknown** parameter λ

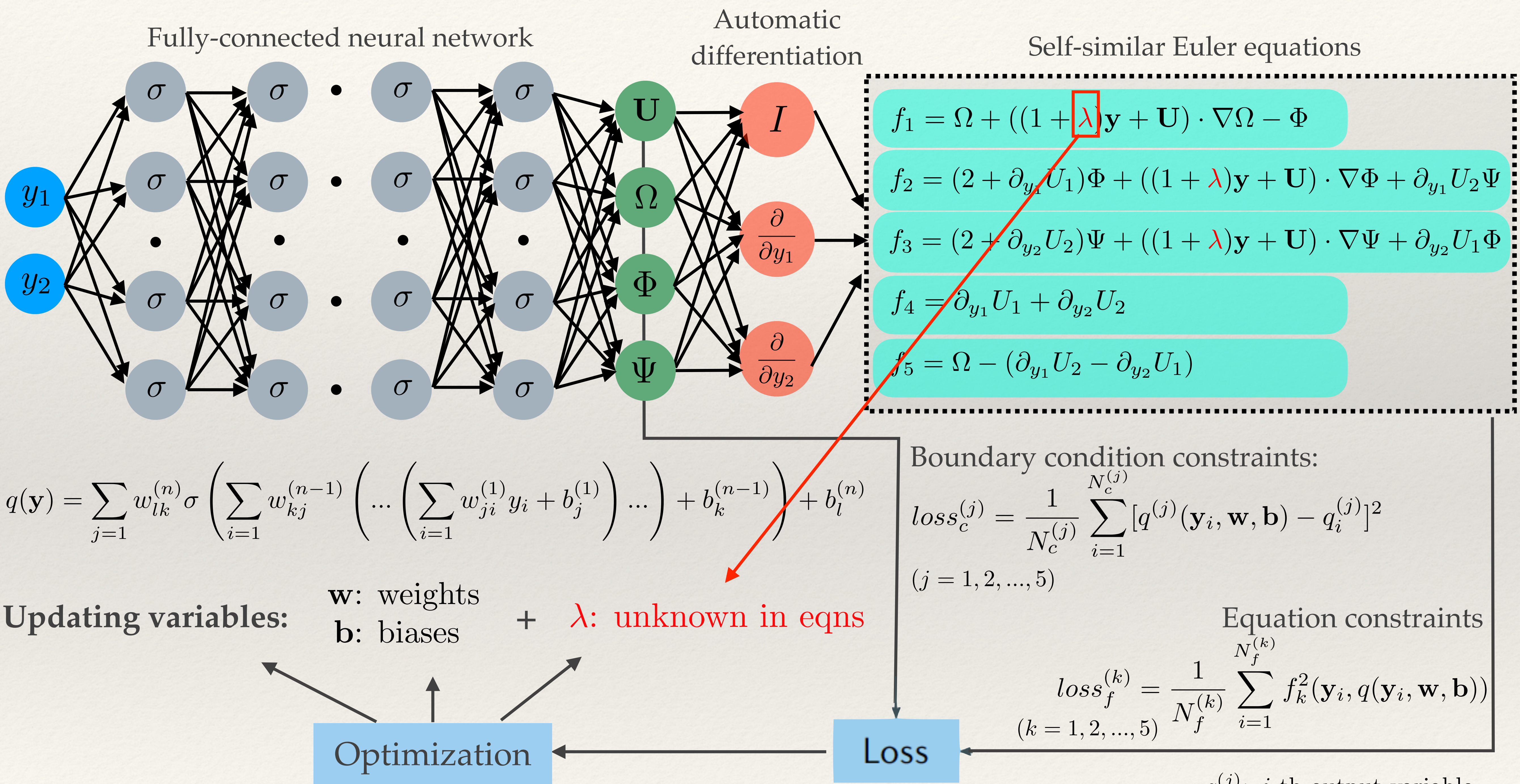
   (Numerical method is only efficient at solving **fully-known** equations)

2. Solution should be **smooth** everywhere

   (Numerical method is hard to deal with the smoothness condition due to **discretization**)

Advantages of PINNs

Physics-informed Neural network for self-similar Euler equation with boundary



$q^{(j)}$: j-th output variable

$q_i^{(j)}$: data of $q^{(j)}$ at $\mathbf{y} = \mathbf{y}_i$

PINN solves the **first** challenges inherently

Challenges to numerical method

Steady self-similar equations for axisymmetric Euler with boundary (Bousinessq)

$$\Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Omega = \Phi$$
$$(2 + \partial_{y_1}U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi = -\partial_{y_1}U_2\Psi$$
$$(2 + \partial_{y_2}U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi = -\partial_{y_2}U_1\Phi$$
$$\Omega = \partial_{y_1}U_2 - \partial_{y_2}U_1 \qquad \text{div}\,\mathbf{U} = 0$$

Two big challenges:

to be determined by the constraint of solution ↓

1. Governing equation involves **unknown** parameter λ ✔

   (Numerical method is only efficient at solving **fully-known** equations)

2. Solution should be **smooth** everywhere

   (Numerical method is hard to deal with the smoothness condition due to **discretization**)

1-D example - Burgers

Burgers' equation

$$u_t + uu_x = 0$$

Assuming the self-similar ansatz

$$u = (1-t)^\lambda U\left(\frac{x}{(1-t)^{1+\lambda}}\right)$$

we obtain the self-similar Burgers' equation

$$-\lambda U + ((1+\lambda)y + U)\partial_y U = 0$$

Using a nice trick, the self-similar Burgers' equation can be implicitly solved:

$$y = -U - CU^{1+\frac{1}{\lambda}}$$

for some constant C. In order to obtain a **smooth symmetric** self-similar solution,

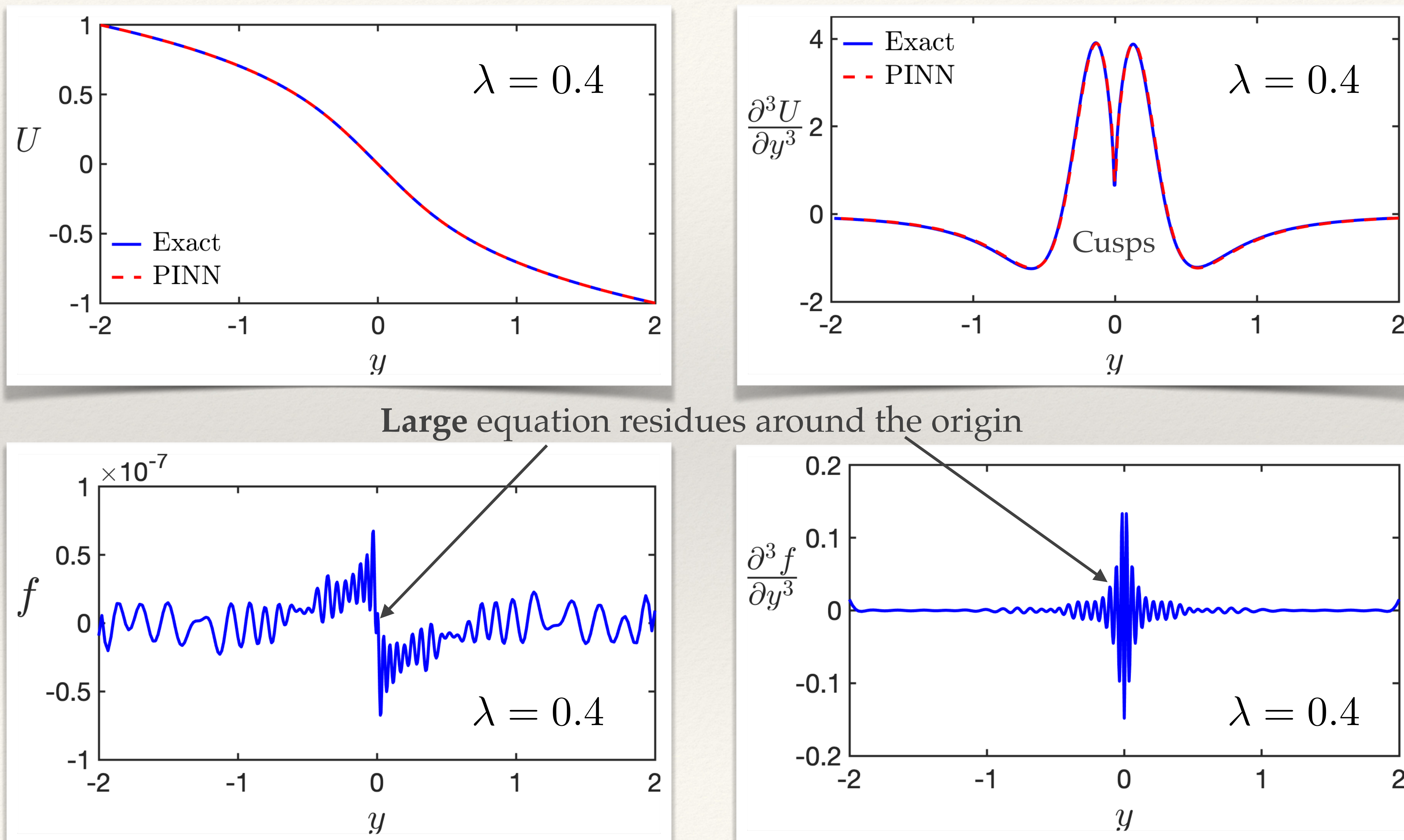
then λ must be chosen such that

$$\lambda = \frac{1}{2i+2} \quad \text{for } i = 0, 1, 2, \ldots$$

Non-smooth solution

Self-similar equation for Burgers: $f = -\lambda U + ((1+\lambda)y + U)\partial_y U$

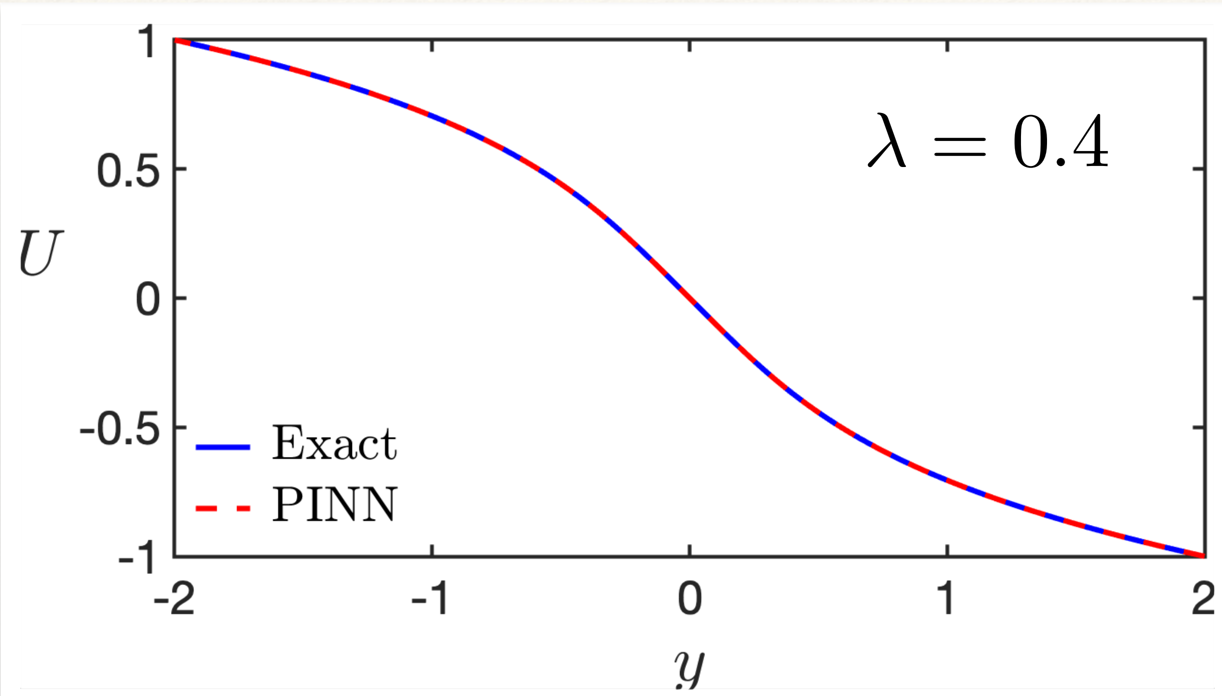
Impose symmetry: $y = -\text{sgn}(y)|U| - \text{sgn}(y)|U|^{1+\frac{1}{\lambda}}$



Non-smooth solution

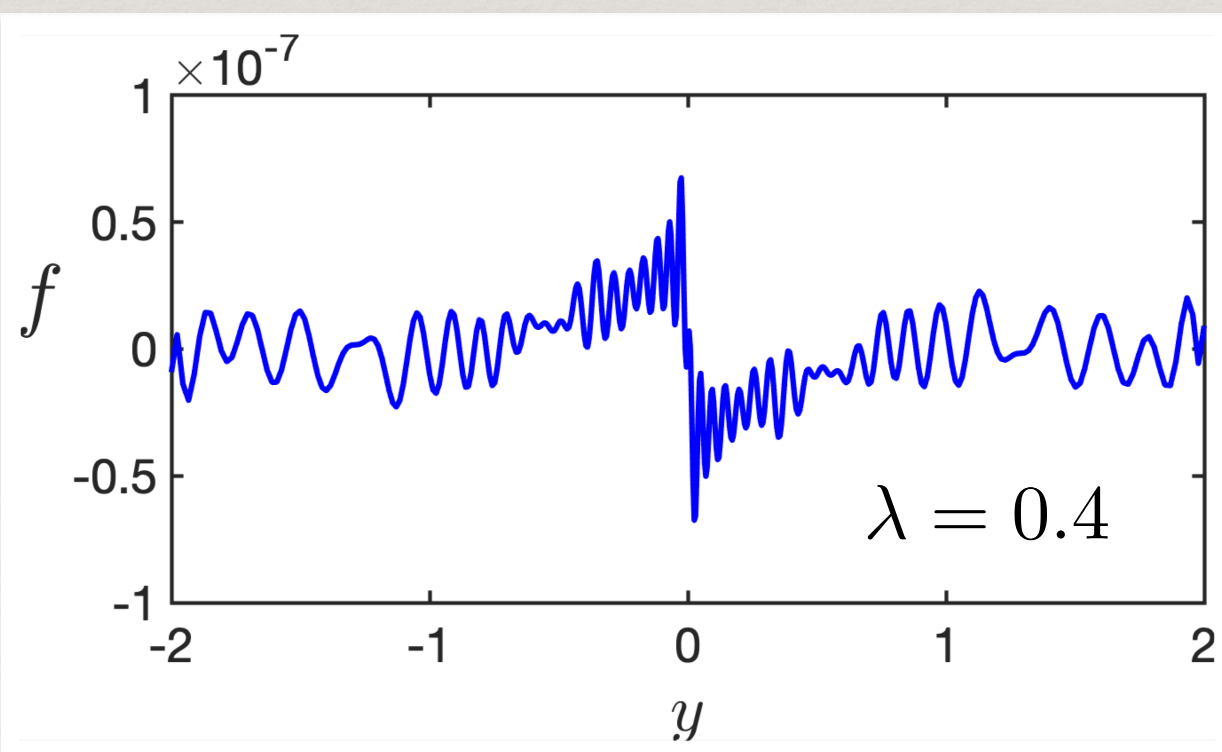
Self-similar equation for Burgers: $f = -\lambda U + ((1+\lambda)y + U)\partial_y U$

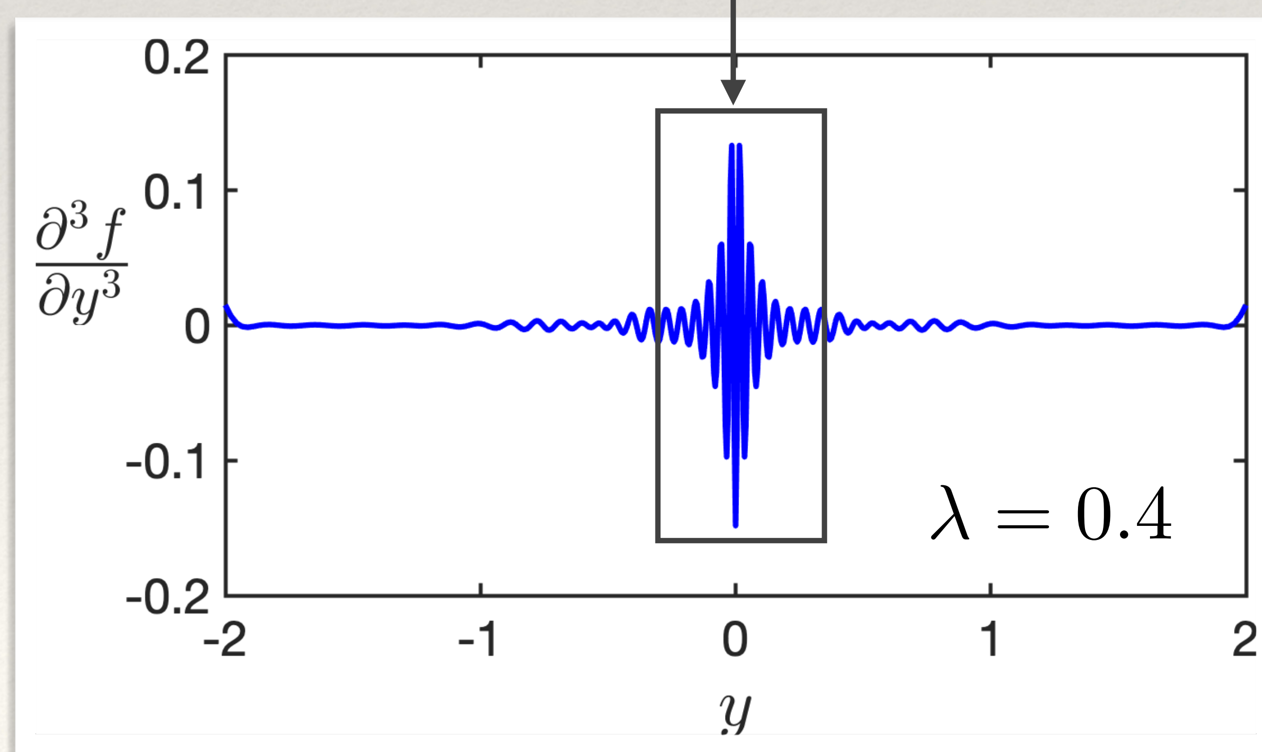
Impose symmetry: $y = -\text{sgn}(y)|U| - \text{sgn}(y)|U|^{1+\frac{1}{\lambda}}$



Additional constraint for smooth solution

$loss_s = [\partial_{xxx} f(x)]^2 \to 0$ around the origin

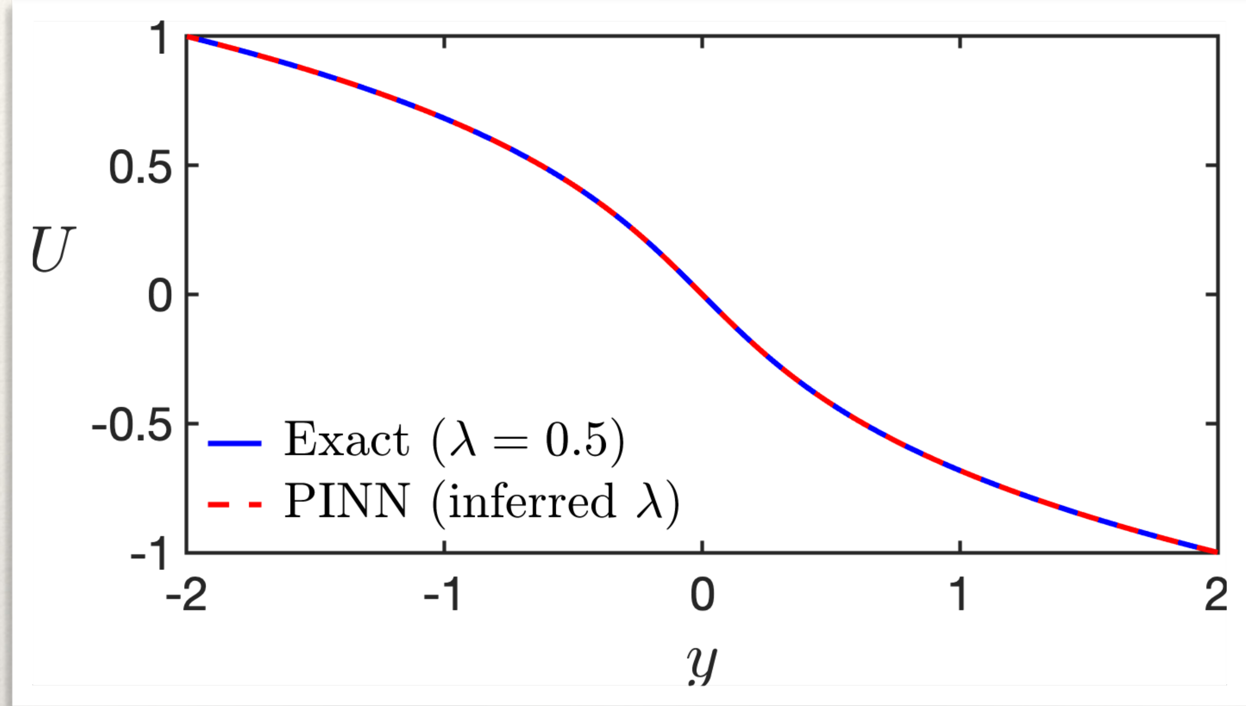




Smooth solution inferred

Self-similar equation for Burgers: $f = -\lambda U + ((1+\lambda)y + U)\partial_y U$

Impose symmetry: $y = -\text{sgn}(y)|U| - \text{sgn}(y)|U|^{1+\frac{1}{\lambda}}$



Additional constraint for smooth solution

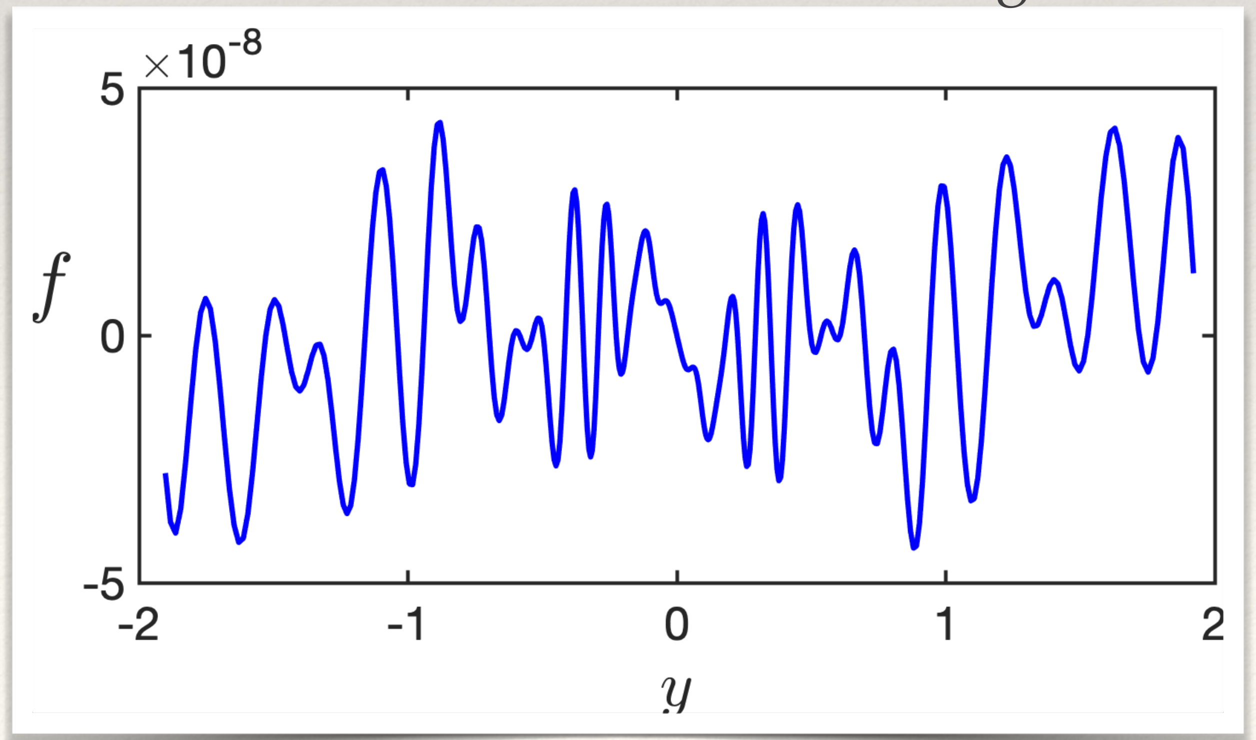
$loss_s = [\partial_{xxx} f(x)]^2 \to 0$ around the origin

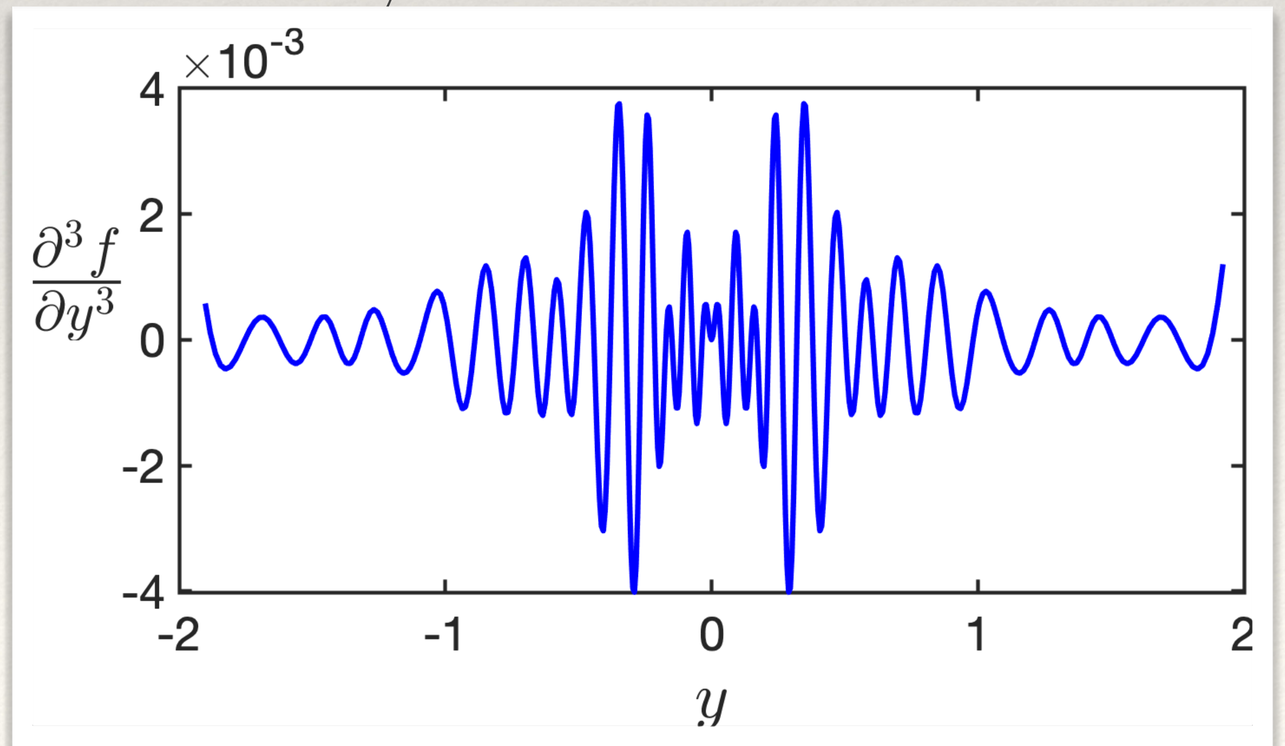
theoretical $\lambda = 0.5$

inferred $\lambda = 0.49995$

Very precise

Uniform higher-order derivatives everywhere





Challenges to numerical method

Steady self-similar equations for axisymmetric Euler with boundary (Bousinessq)

$$\Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Omega = \Phi$$
$$(2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Phi = -\partial_{y_1} U_2 \Psi$$
$$(2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Psi = -\partial_{y_2} U_1 \Phi$$
$$\Omega = \partial_{y_1} U_2 - \partial_{y_2} U_1 \qquad\qquad \operatorname{div} \mathbf{U} = 0$$

Two big challenges:

to be determined by the constraint of solution ↓

1. Governing equation involves **unknown** parameter λ ✔

   (Numerical method is only efficient at solving **fully-known** equations)

2. Derived solution should be **smooth** everywhere ✔

   (Numerical method is hard to deal with the smoothness condition due to **discretization**)

Non-smooth solution for Euler (Bousinessq)

Self-similar equations for axisymmetric Euler with boundary (Bousinessq)

Fixing $\lambda = 5$

$$f_1 = \Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Omega - \Phi$$

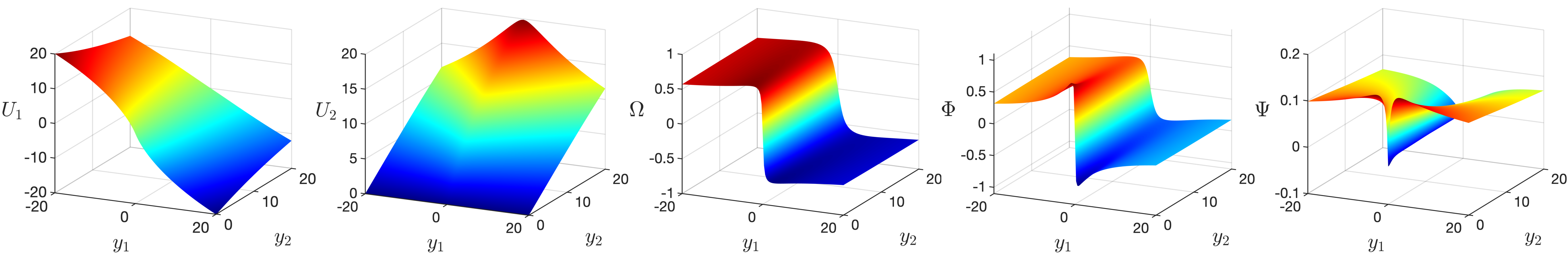
$$f_2 = (2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi + \partial_{y_1} U_2 \Psi$$

$$f_3 = (2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi + \partial_{y_2} U_1 \Phi$$

$$f_4 = \partial_{y_1} U_1 + \partial_{y_2} U_2$$

$$f_5 = \Omega - (\partial_{y_1} U_2 - \partial_{y_2} U_1)$$

Non-smooth self-similar solution at $\lambda = 5$



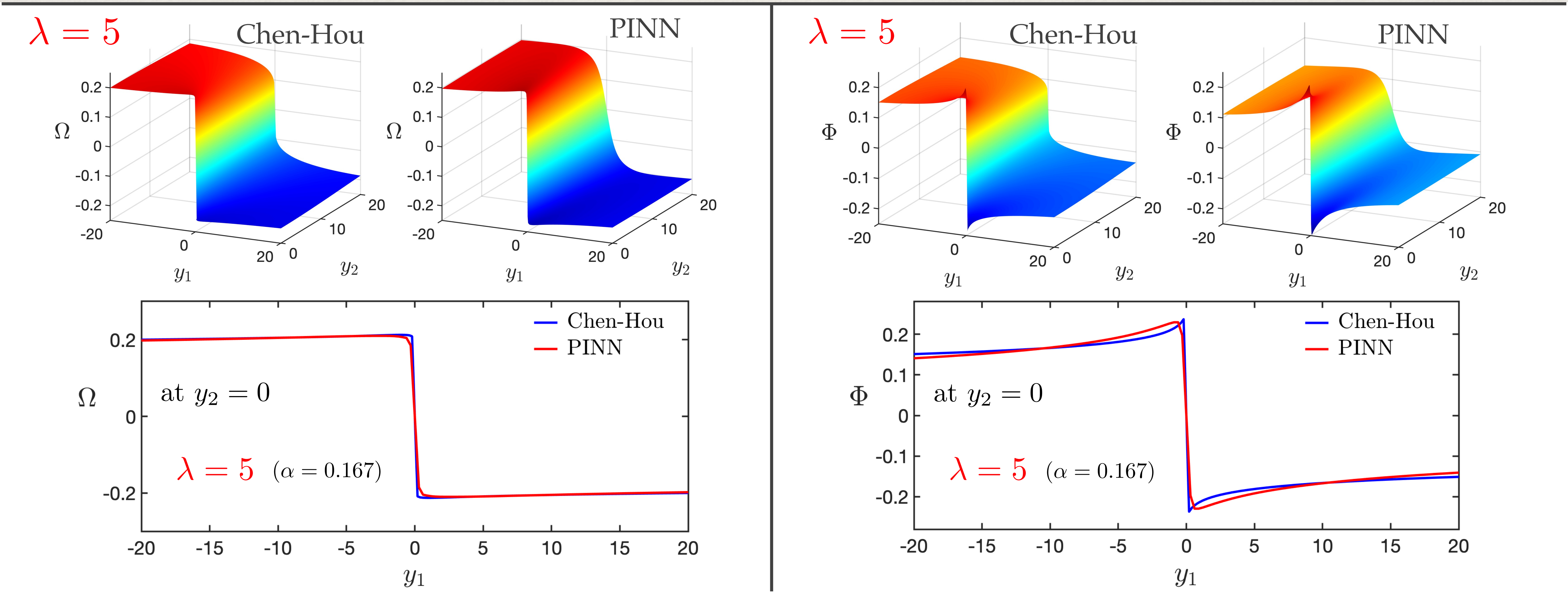
Comparison with literature

Define $R = (y_1^2 + y_2^2)^{\frac{\alpha}{2}}$ and $\gamma = \arctan\left(\frac{y_2}{y_1}\right)$ with $\alpha = \frac{1}{1+\lambda}$

Chen-Hou '21 constructed an approximate self-similar solution for $\alpha \ll 1$ $(\lambda \gg 1)$

$$\Omega = -\frac{\alpha}{c}(\cos(\gamma))^{\alpha}\frac{3R}{(1+R)^2}, \qquad \Phi = -\frac{\alpha}{c}(\cos(\gamma))^{\alpha}\frac{6R}{(1+R)^3}$$

for $\gamma \in [0, \frac{\pi}{2}]$ (or equivalently $y_1 \geq 0$) and $c = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} (\cos(\theta))^{\alpha} \sin(2\theta)\, d\theta$



Non-smooth solution for Euler (Bousinessq)

Self-similar equations for axisymmetric Euler with boundary (Bousinessq)

Fixing $\lambda = 5$

$$f_1 = \Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Omega - \Phi$$

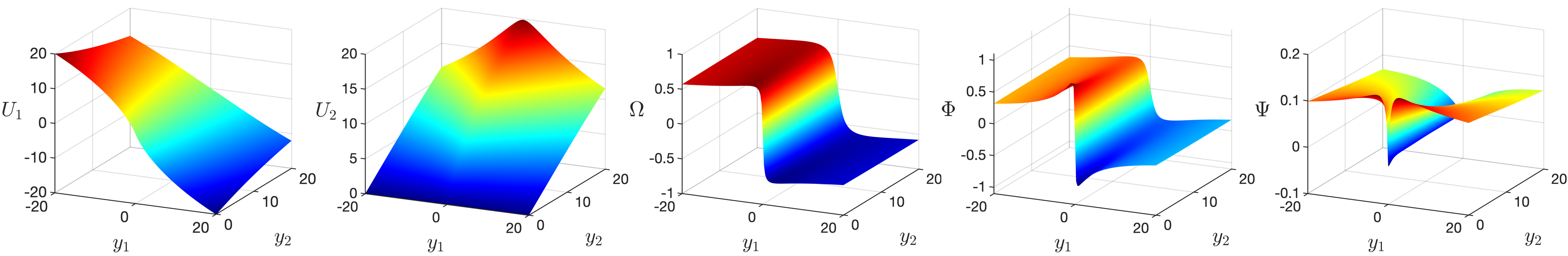
$$f_2 = (2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi + \partial_{y_1} U_2 \Psi$$

$$f_3 = (2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi + \partial_{y_2} U_1 \Phi$$

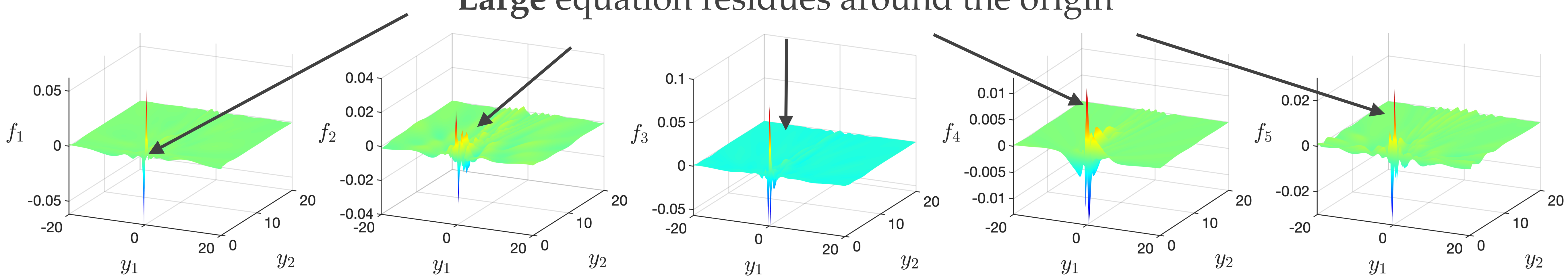
$$f_4 = \partial_{y_1} U_1 + \partial_{y_2} U_2$$

$$f_5 = \Omega - (\partial_{y_1} U_2 - \partial_{y_2} U_1)$$

Non-smooth self-similar solution at $\lambda = 5$



Large equation residues around the origin



Smooth solution for Euler (Bousinesq)

Self-similar equations for axisymmetric Euler with boundary (Bousinessq)

$$f_1 = \Omega + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Omega - \Phi$$

$$f_2 = (2 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi + \partial_{y_1} U_2 \Psi$$

$$f_3 = (2 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi + \partial_{y_2} U_1 \Phi$$

$$f_4 = \partial_{y_1} U_1 + \partial_{y_2} U_2$$

$$f_5 = \Omega - (\partial_{y_1} U_2 - \partial_{y_2} U_1)$$

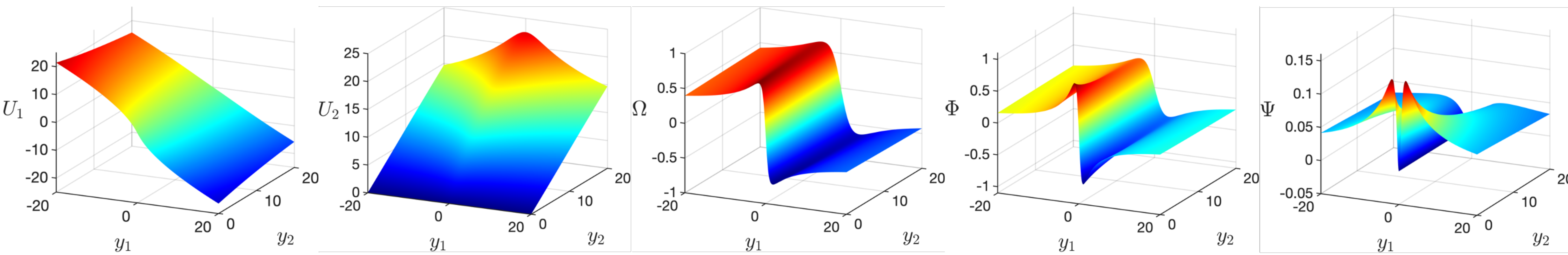
Additional constraint for **smooth** solution

$loss_s = [\partial_x f(x)]^2 \to 0$ around the origin

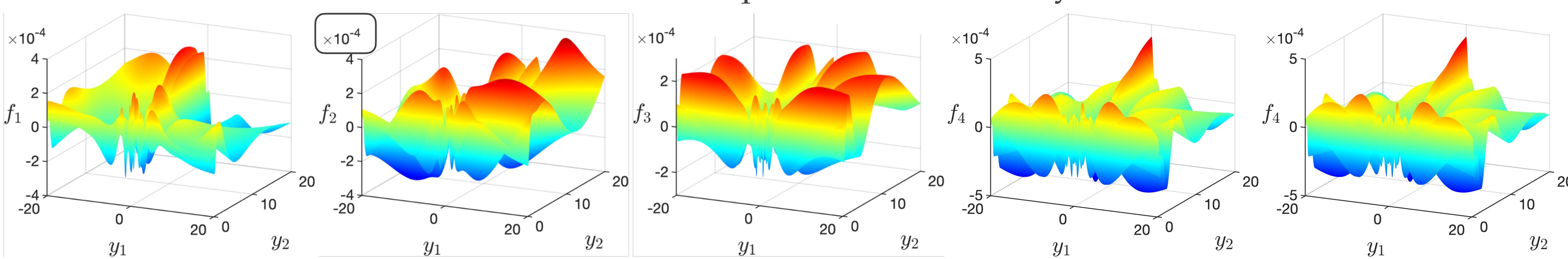
Inferred $\lambda = 1.90$

(Luo-Hou $\lambda = 1.91$)

Smooth self-similar solution at $\lambda = 1.90$



Uniform and small equation residues everywhere



Universality - other 1-D examples

Given a constant $a \in \mathbb{R}$, the **generalized De Gregorio** equations are

$$\omega_t + au\omega_x = \omega u_x, \qquad \text{where } u = -\int_0^x (H\omega)(s)\,ds = -\Lambda^{-1}\omega$$

(Hilbert transform)

Setting $U = -\Lambda^{-1}\Omega$ and $y = \dfrac{x}{(1-t)^{1+\lambda}}$, leads to the equations

$$\Omega + ((1+\lambda)y - aU)\partial_y\Omega + \Omega\partial_y U = 0$$

We assume Ω and U are odd and we fix

$$\partial_y\Omega(0) = 2$$

Literature review

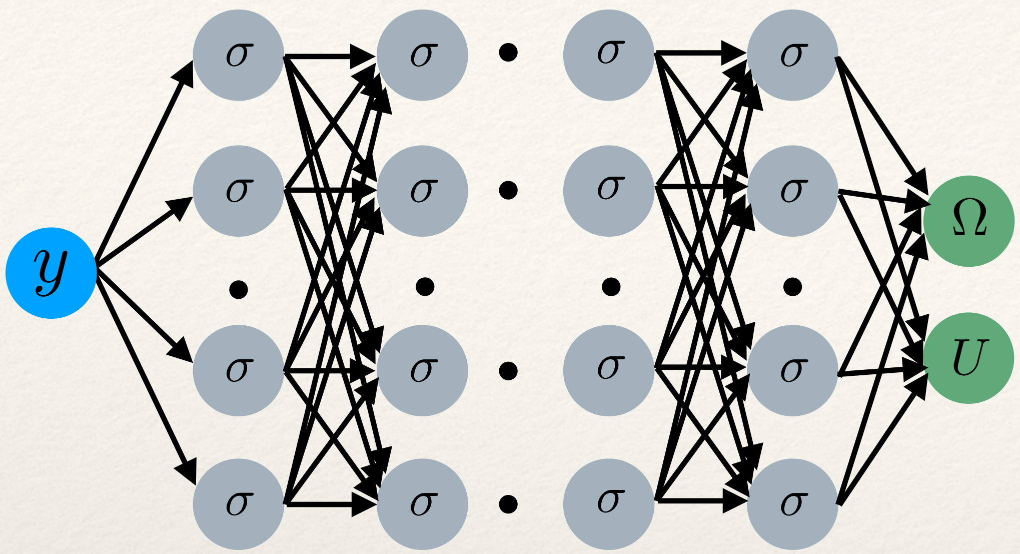
Rigorous results:

1. The case $a = 0$ is the Constantin-Lax-Majda equation. Explicit self-similar blow up solutions can be constructed Constantin-Lax-Majda '85
2. The case $a = -1$ is the Córdoba-Córdoba-Fontelos model, singularity formulation is known (Córdoba-Córdoba-Fontelos '05).
3. For $a < 0$, blowup (Castro-Córdoba '10).
4. For $a > 0$, a small, self-similar blow-up was proven by Elgindi '19.
5. The case $a = 1$ is the De Gregorio equation. Self-similar blow-up was proven in Chen-Hou-Huang '19 via a computer assisted proof.

Numerical results:

Numerical results: In Lushnikov-Silantyev-Siegel '21, numerical self-similar solutions were found for $a \in [-1, 1]$ and beyond.

Generalized De Gregorio equation



Equations:

$$f_1 = \Omega + ((1+\lambda)y - aU)\partial_y\Omega + \Omega\partial_y U$$

$$f_2 = \frac{dU}{dy} + \tilde{H}\Omega$$

(numerical Hilbert Transform)

Conditions: $\partial_y\Omega(0) = 2$ $\quad$ Ω and U are odd

Boundary condition constraints:

$$loss_c = \left[\frac{d\Omega}{dy}(y=0, \mathbf{w}, \mathbf{b}) - 2\right]^2$$

$\mathbf{w}$: weights $\quad$ $\mathbf{b}$: biases

Equation constraints (entire domain)

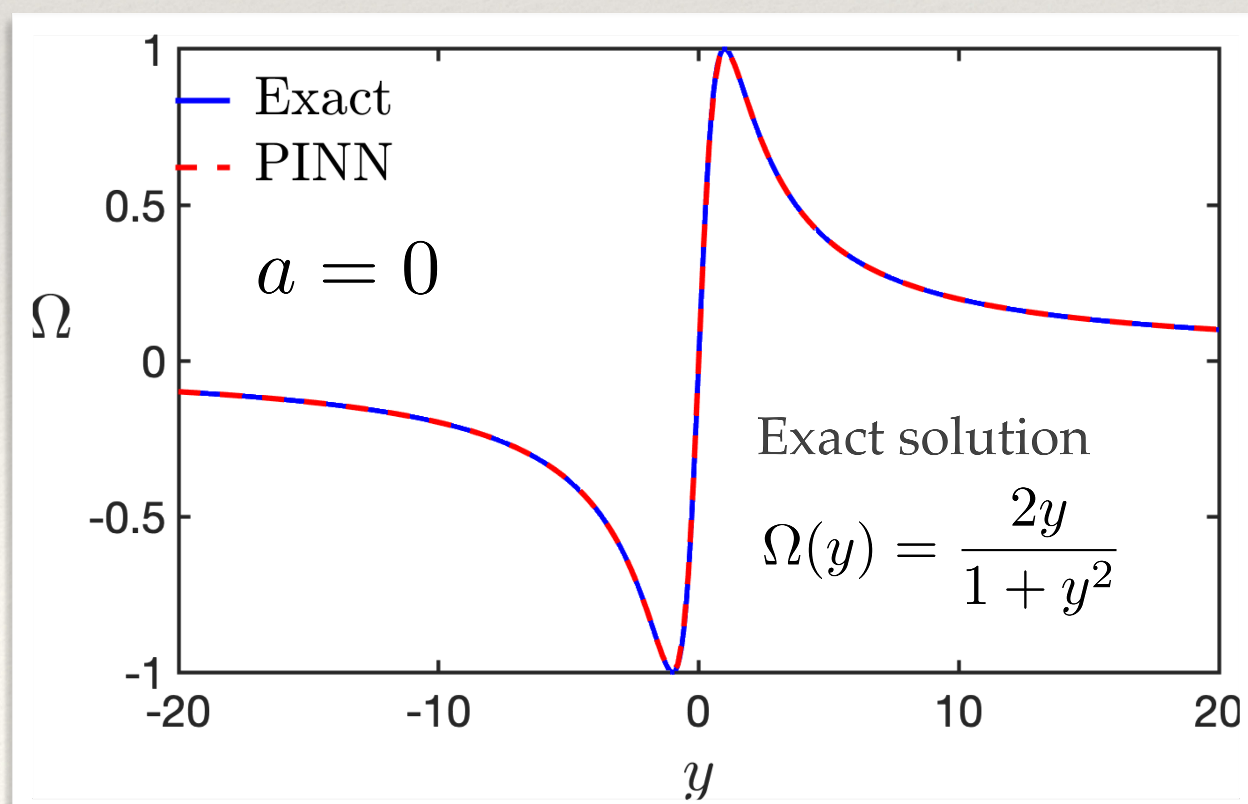
$$loss_f^{(k)} = \frac{1}{N_f}\sum_{i=1}^{N_f} f_k^2(y_i, \mathbf{w}, \mathbf{b})$$

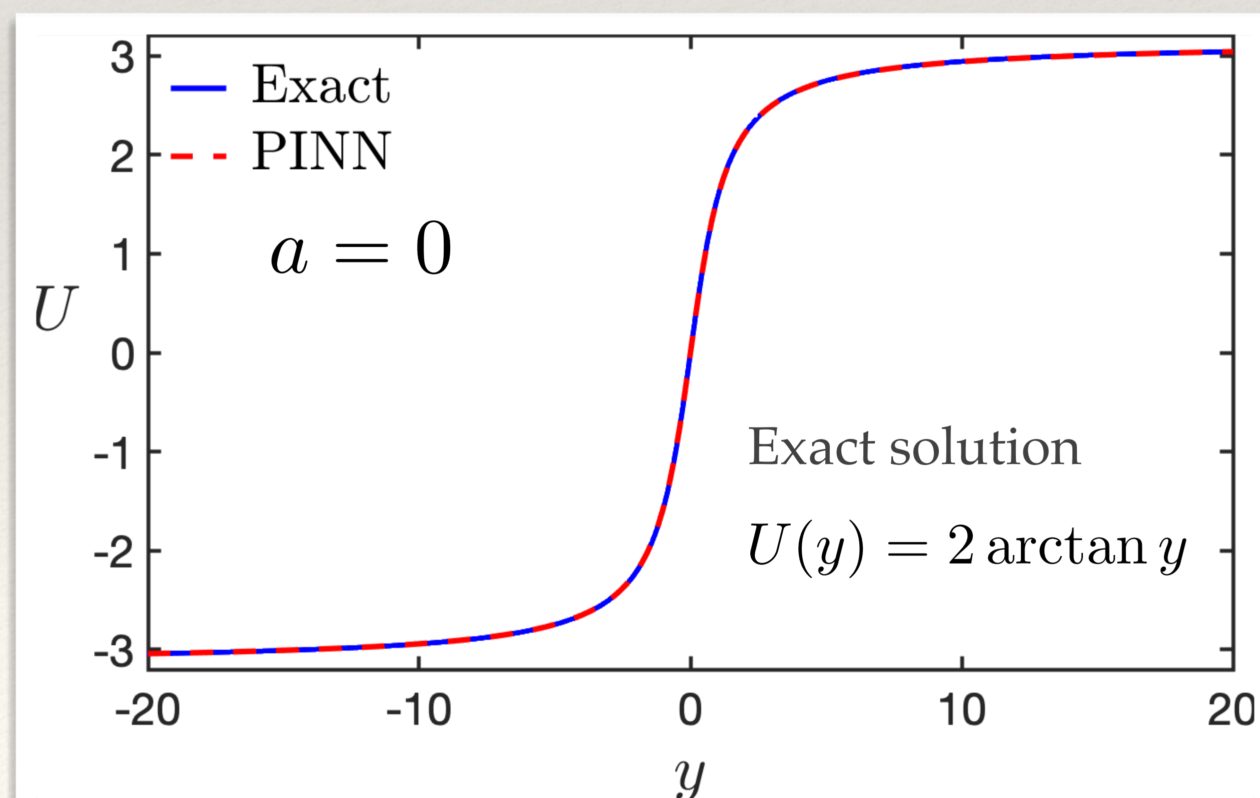
N_f: number of collocation points

Smoothness constraint (near origin)

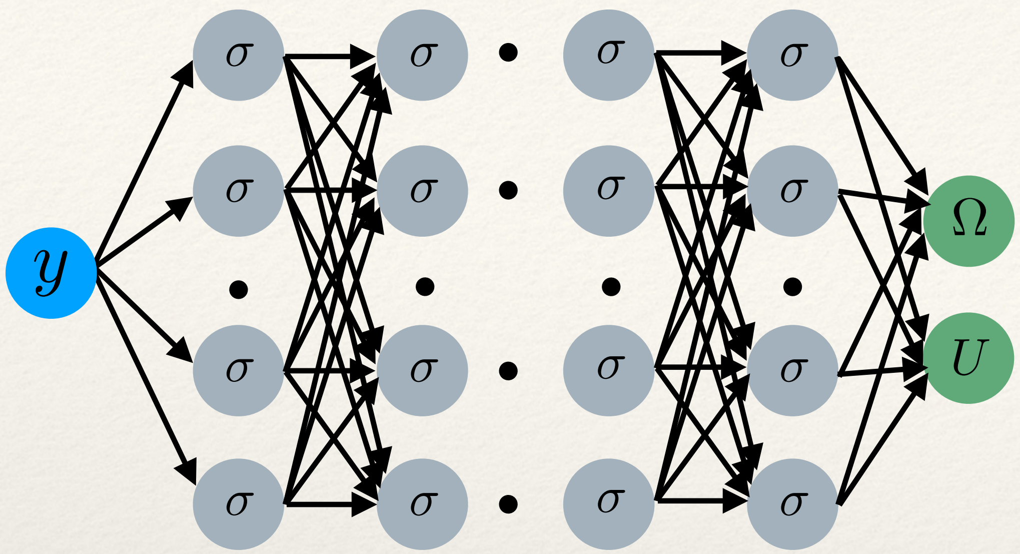
$$loss_s = \frac{1}{N_s}\sum_{i=1}^{N_s}\left[\frac{df_1}{dy}(y_i, \mathbf{w}, \mathbf{b})\right]^2$$

N_s: number of collocation points around origin





Generalized De Gregorio equation



Equations:

$$f_1 = \Omega + ((1+\lambda)y - aU)\partial_y\Omega + \Omega\partial_y U$$

$$f_2 = \frac{dU}{dy} + \tilde{H}\Omega$$ (numerical Hilbert Transform)

(Zhou *et. al.* 2009)

Conditions: $\partial_y\Omega(0) = 2$ $\quad\Omega$ and U are odd

Boundary condition constraints:

$$loss_c = \left[\frac{d\Omega}{dy}(y=0,\mathbf{w},\mathbf{b}) - 2\right]^2$$

$\mathbf{w}$: weights $\quad\mathbf{b}$: biases

Equation constraints (entire domain)

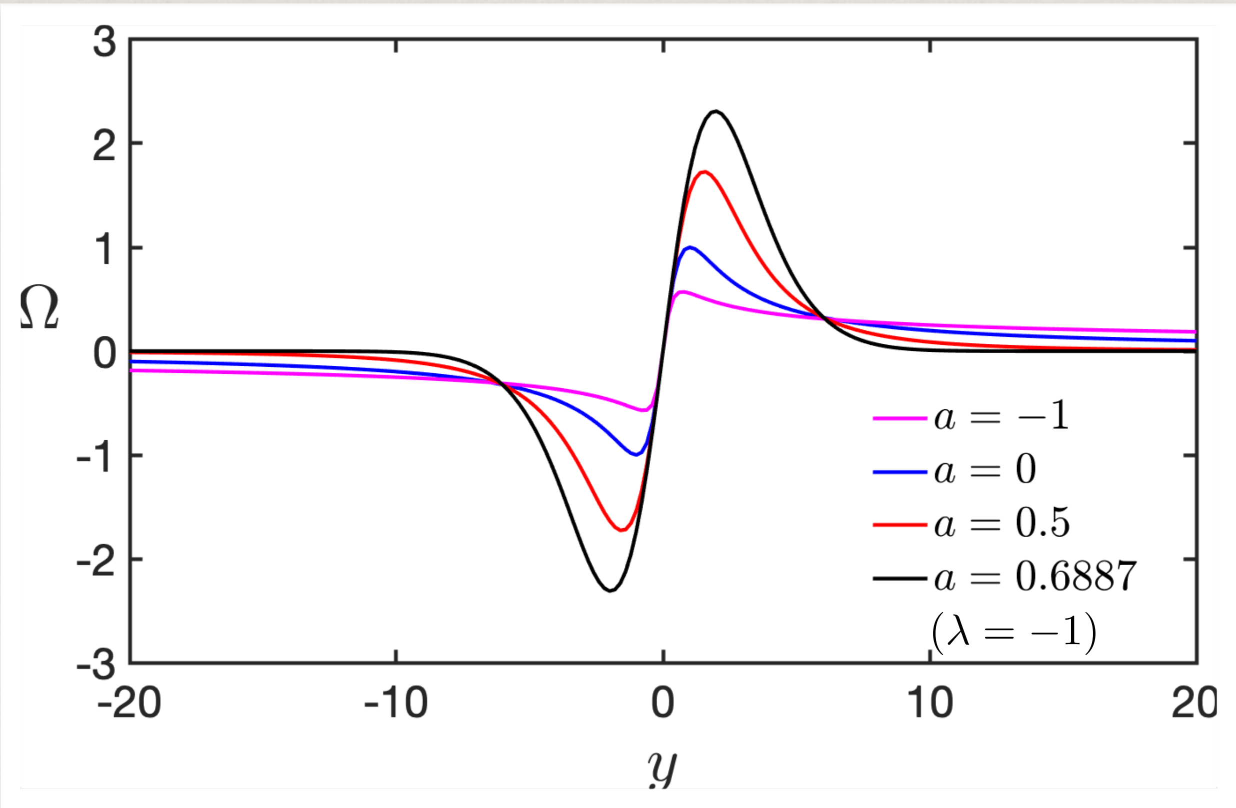
$$loss_f^{(k)} = \frac{1}{N_f}\sum_{i=1}^{N_f} f_k^2(y_i,\mathbf{w},\mathbf{b})$$

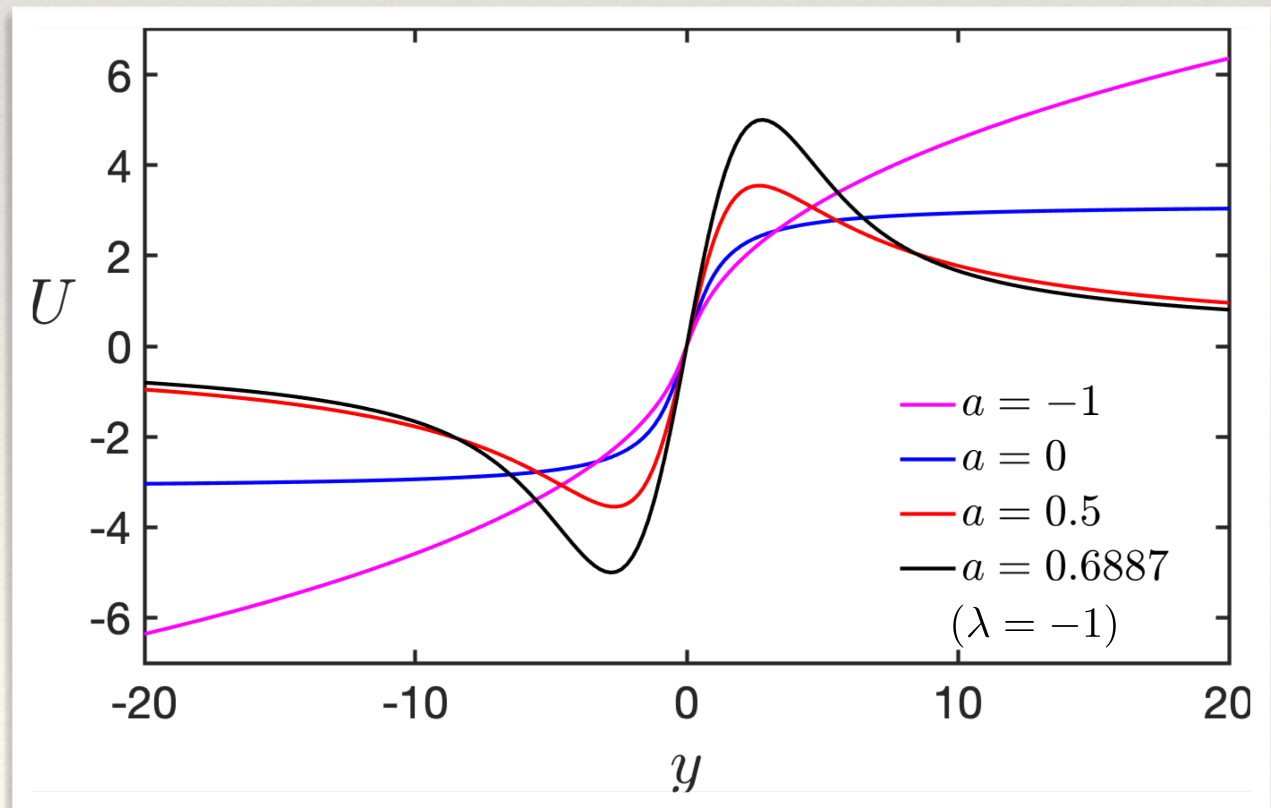
N_f: number of collocation points

Smoothness constraint (near origin)

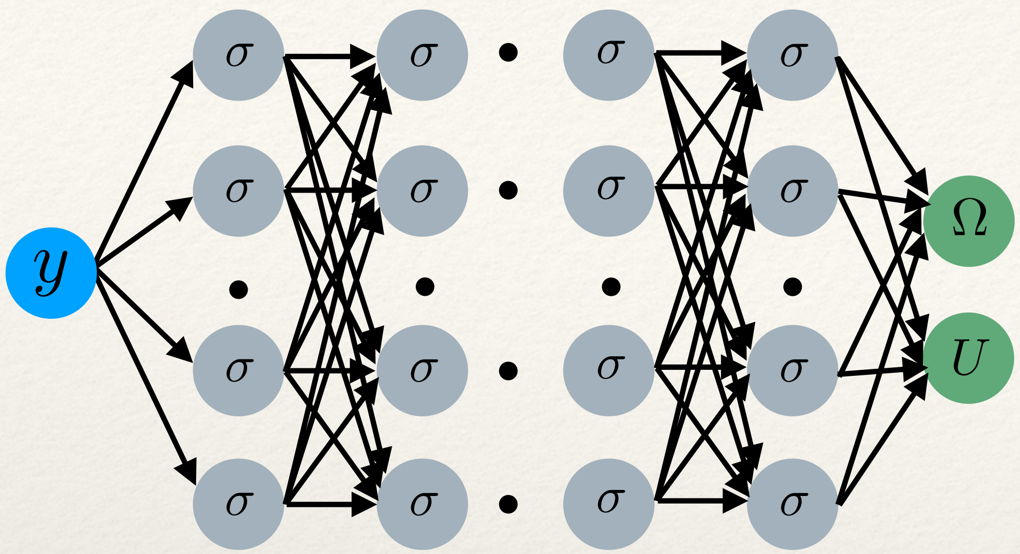
$$loss_s = \frac{1}{N_s}\sum_{i=1}^{N_s}\left[\frac{df_1}{dy}(y_i,\mathbf{w},\mathbf{b})\right]^2$$

N_s: number of collocation points around origin





Generalized De Gregorio equation



Equations:

$$f_1 = \Omega + ((1+\lambda)y - aU)\partial_y \Omega + \Omega \partial_y U$$

$$f_2 = \frac{dU}{dy} + \tilde{H}\Omega$$

(numerical Hilbert Transform)

(Zhou *et. al.* 2009)

Conditions: $\partial_y \Omega(0) = 2$ $\quad$ Ω and U are odd

Boundary condition constraints:

$$loss_c = \left[\frac{d\Omega}{dy}(y=0,\mathbf{w},\mathbf{b}) - 2\right]^2$$

$\mathbf{w}$: weights $\quad$ $\mathbf{b}$: biases

Equation constraints (entire domain)

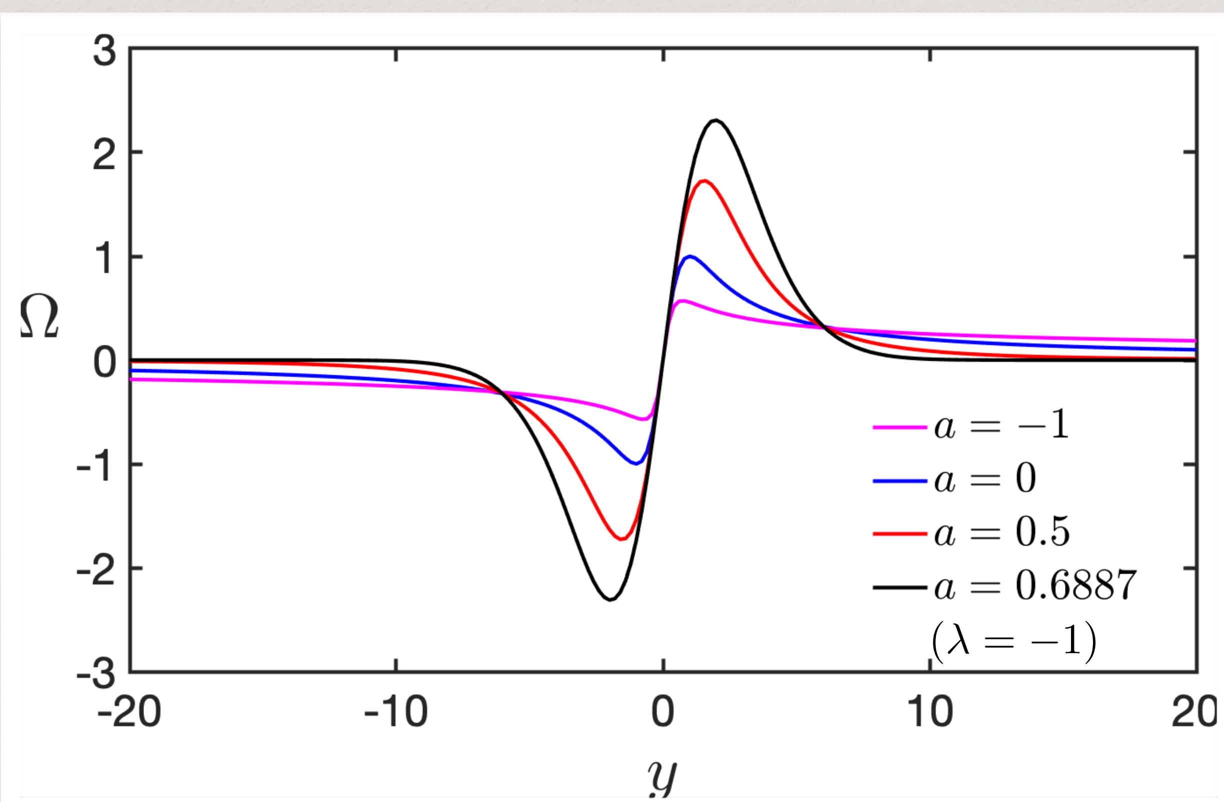
$$loss_f^{(k)} = \frac{1}{N_f}\sum_{i=1}^{N_f} f_k^2(y_i,\mathbf{w},\mathbf{b})$$

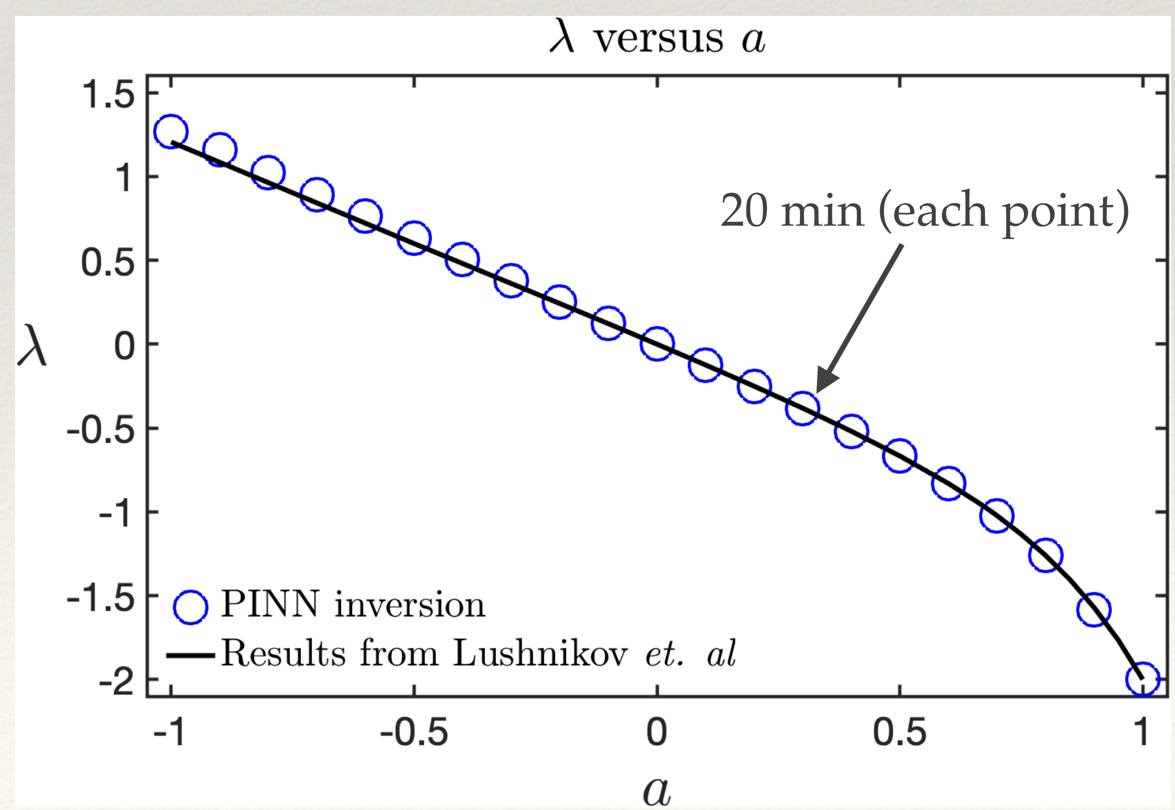
N_f: number of collocation points

Smoothness constraint (near origin)

$$loss_s = \frac{1}{N_s}\sum_{i=1}^{N_s}\left[\frac{df_1}{dy}(y_i,\mathbf{w},\mathbf{b})\right]^2$$

N_s: number of collocation points around origin





Universality - other 2-D examples

The **incompressible porous media (IPM)** equations are written

$$\partial_t \rho + \operatorname{div}(\rho \mathbf{u}) = 0\,, \quad \operatorname{div} \mathbf{u} = 0\,, \quad \text{and} \quad \mathbf{u} + \nabla p = (0, \rho)$$

where the 2D vector $\mathbf{u}(\mathbf{x}, t)$ is the velocity and the scalar $\rho(x, t)$ is the density

we introduce $\phi = \partial_{x_1} \rho$ and $\psi = \partial_{x_2} \rho$ and assume self-similar ansatz

$$\mathbf{u} = (1-t)^{\lambda} \mathbf{U}(\mathbf{y})\,, \quad \Phi = (1-t)^{-1} \phi(\mathbf{y}) \quad \text{and} \quad \Psi = (1-t)^{-1} \psi(\mathbf{y})$$

with self-similar coordinates $\mathbf{y} = (y_1, y_2) = \dfrac{\mathbf{x}}{(1-t)^{1+\lambda}}$

We obtain the self-similar equations

$$\begin{gathered}
(1 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Phi = -\partial_{y_1} U_2 \Psi \\
(1 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla \Psi = -\partial_{y_2} U_1 \Phi \\
\Phi = \partial_{y_1} U_2 - \partial_{y_2} U_1 \qquad \operatorname{div} \mathbf{U} = 0
\end{gathered}$$

Smooth solution for IPM

Self-similar equations for IPM

$$f_1 = (1 + \partial_{y_1} U_1)\Phi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Phi + \partial_{y_1} U_2 \Psi$$

$$f_2 = (1 + \partial_{y_2} U_2)\Psi + ((1+\lambda)\mathbf{y} + \mathbf{U}) \cdot \nabla\Psi + \partial_{y_2} U_1 \Phi$$

$$f_3 = \partial_{y_1} U_1 + \partial_{y_2} U_2$$

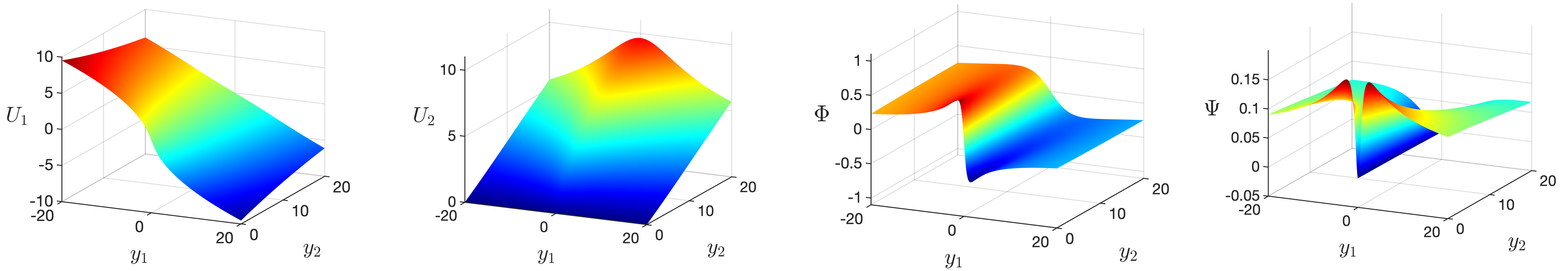
$$f_4 = \Phi - (\partial_{y_1} U_2 - \partial_{y_2} U_1)$$

Additional constraint for **smooth** solution

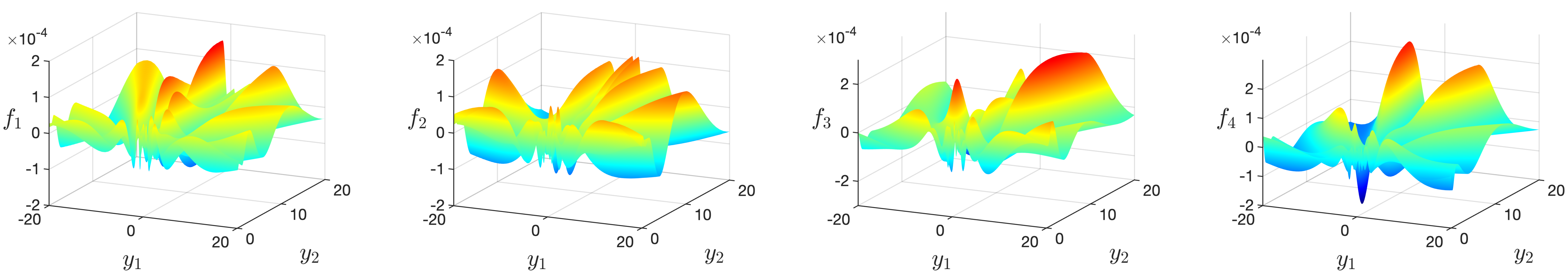
$loss_s = [\partial_x f(x)]^2 \to 0$ around the origin

Inferred $\lambda = 1.03$

Smooth self-similar solution at $\lambda = 1.03$



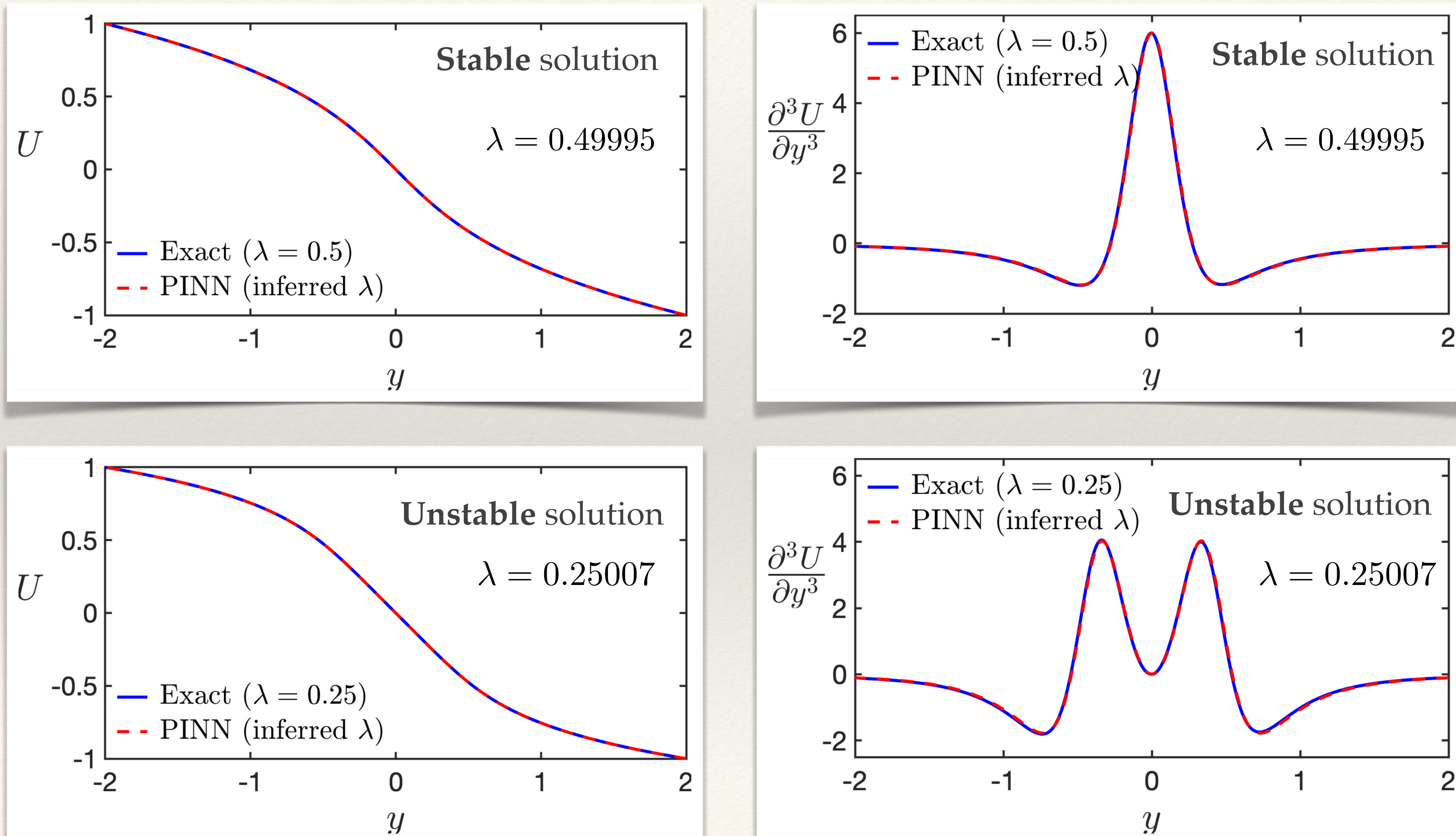
Uniform and small equation residues everywhere



Advantages of PINNs

Self-similar equation for Burgers: $f = -\lambda U + ((1+\lambda)y + U)\partial_y U$

Impose symmetry: $y = -\mathrm{sgn}(y)|U| - \mathrm{sgn}(y)|U|^{1+\frac{1}{\lambda}}$



Summary

- PINNs is a differential equation solver *(giving continuous function)*
- PINNs solves equation with unknowns *(as long as well-posed)*
- PINNs can deal with the smoothness constraint *(find blow-up solution)*

Future works

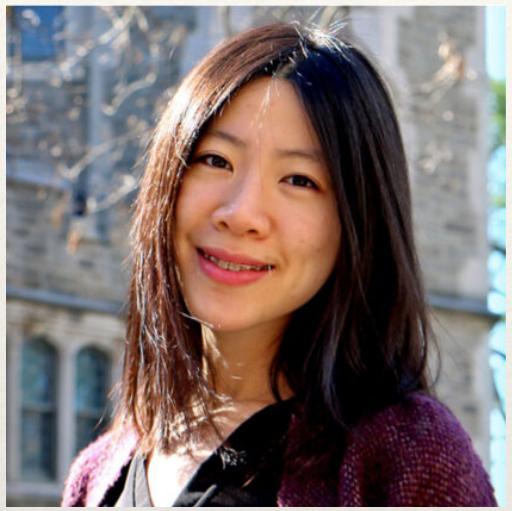
Theoretical: make a rigorous proof $\Rightarrow$ Computer-assisted

Numerical:

1. find self-similar blow-up solution for Euler without boundary
2. find unstable solution for 2-D equations

Acknowledgements

Prof. Ching-Yao Lai



Department of Geoscience,
Princeton University

Prof. Tristan Buckmaster

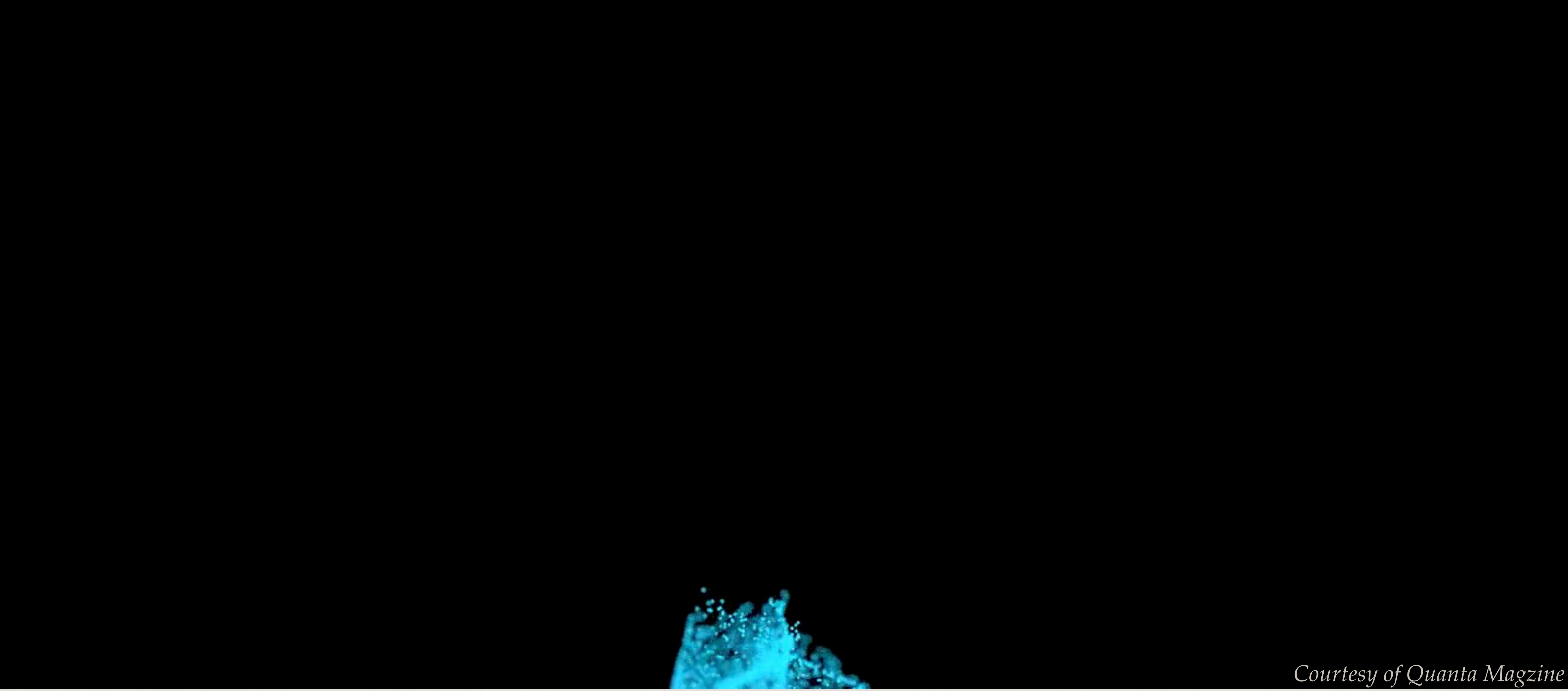
Department of Mathematics,
Princeton University &
Institute of Advanced Study

Prof. Javier Gómez-Serran

Department of Mathematics
Brown University &
Universitat de Barcelona

Charlie Cowen-Breen '22
Department of Mathematics,
Princeton University

Ray (Ming-Ruey) Chou
Department of Geosciences,
Princeton University


Courtesy of Quanta Magzine

Thank you and questions

Yongji Wang

yw1705@princeton.edu