Physics informed neural networks

PINN

Ordinary differential equations

$$\begin{cases} \frac{dx}{dt} = \alpha x - \beta xy & \text{(Prey)} \\ \frac{dy}{dt} = \delta xy - \gamma y & \text{(Predator)} \end{cases}$$

An ODE contains one or more derivatives of a dependent variable y with respect to a single independent variable t (or x).

$$F(t, y, y', y'', \ldots, y^{(n)}) = 0$$

$$L\frac{d^2q}{dt^2} + R\frac{dq}{dt} + \frac{1}{C}q = E(t)$$

$$\frac{dN}{dt} = -\lambda N$$

$$\frac{dP}{dt} = rP\left(1 - \frac{P}{K}\right)$$

Key Classifications

- **Order:** The highest derivative present in the equation. (e.g., $y'' + y = 0$ is 2nd order).
- **Linearity:** An ODE is **linear** if the dependent variable y and its derivatives appear only to the first power and are not multiplied together. Otherwise, it is **non-linear**.
- **Homogeneity:**
  - **Homogeneous:** No term depends *only* on the independent variable (RHS = 0).
  - **Non-homogeneous:** Contains a term depending only on the independent variable (RHS $\neq$ 0).

Partial differential equations

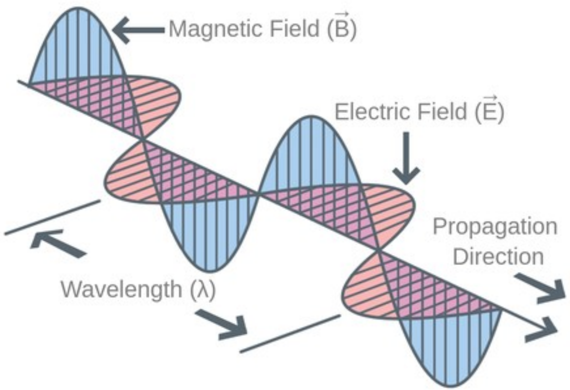
A PDE involves partial derivatives of a function u with respect to two or more independent variables (e.g., time t and space x, y, z).

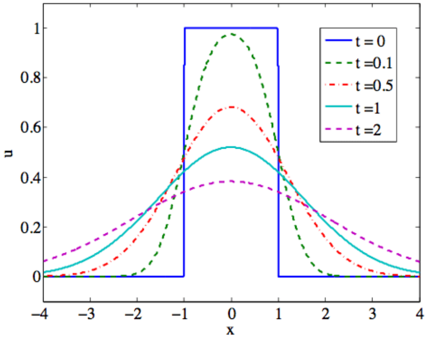
$$F(x, t, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial x^2}, \dots) = 0$$

Key Classifications (Second Order)

For a standard linear second-order PDE of the form $Au_{xx} + Bu_{xy} + Cu_{yy} + \cdots = 0$, the discriminant $B^2 - 4AC$ determines the type:

1. **Elliptic ($B^2 - 4AC < 0$):** Describes equilibrium states.
    - *Example:* Laplace's Equation ($\nabla^2 u = 0$).
2. **Parabolic ($B^2 - 4AC = 0$):** Describes diffusion processes.
    - *Example:* Heat Equation ($u_t = \alpha \nabla^2 u$).
3. **Hyperbolic ($B^2 - 4AC > 0$):** Describes wave propagation.
    - *Example:* Wave Equation ($u_{tt} = c^2 \nabla^2 u$).





General formulation

Following the original formulation of Raissi *et al.*, we begin with a brief overview of physics-informed neural networks (PINNs) [10] in the context of solving partial differential equations (PDEs). Generally, we consider PDEs taking the form

$$\mathbf{u}_t + \mathcal{N}[\mathbf{u}] = 0, \quad t \in [0, T], \ \mathbf{x} \in \Omega, \tag{2.1}$$

subject to the initial and boundary conditions

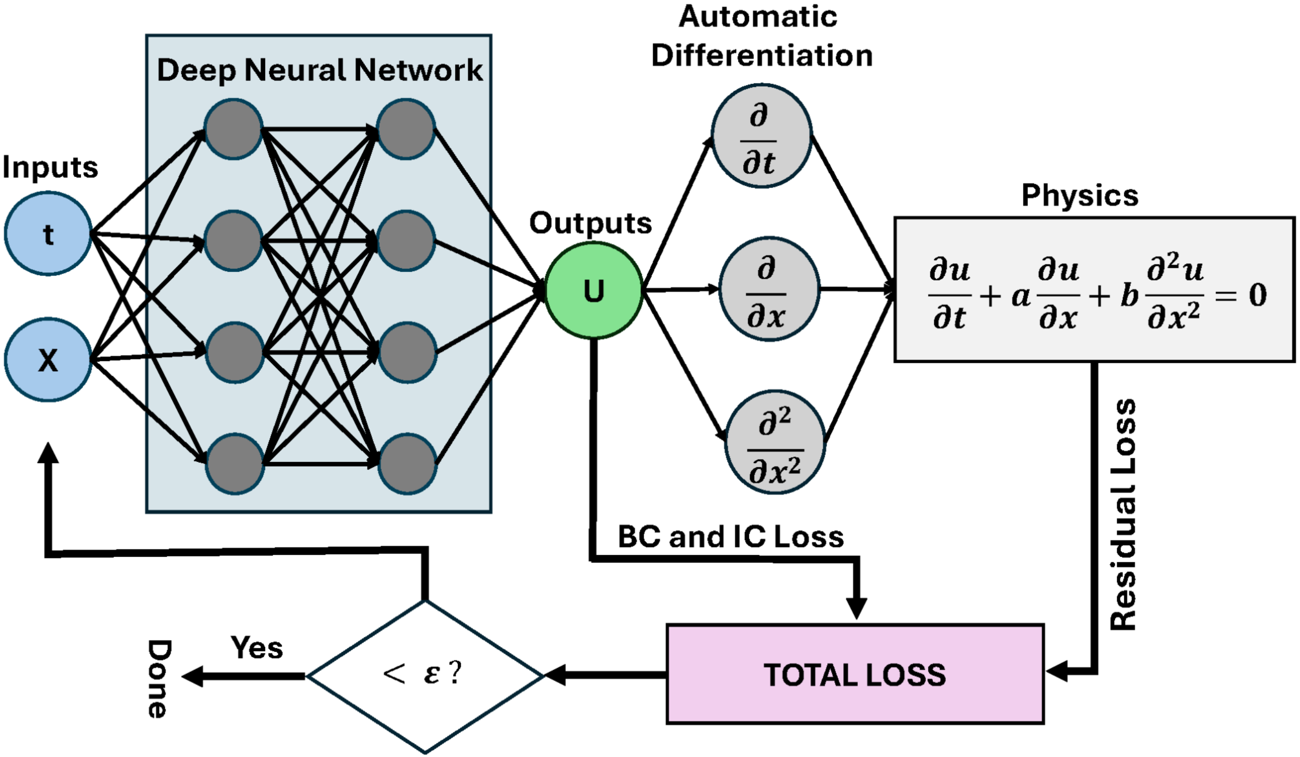
$$\mathbf{u}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{2.2}$$

$$\mathcal{B}[\mathbf{u}] = 0, \quad t \in [0, T], \ \mathbf{x} \in \partial\Omega, \tag{2.3}$$

where $\mathcal{N}[\cdot]$ is a linear or nonlinear differential operator, and $\mathcal{B}[\cdot]$ is a boundary operator corresponding to Dirichlet, Neumann, Robin, or periodic boundary conditions. In addition, $\mathbf{u}$ describes the unknown latent solution that is governed by the PDE system of Equation (2.1).

https://arxiv.org/pdf/2308.08468

Solving with neural networks



Residual

We proceed by representing the unknown solution $\mathbf{u}(t, \mathbf{x})$ by a deep neural network $\mathbf{u}_\theta(t, \mathbf{x})$, where θ denotes all tunable parameters of the network (e.g., weights and biases). This allows us to define the PDE residuals as

$$\mathcal{R}_\theta(t, \mathbf{x}) = \frac{\partial \mathbf{u}_\theta}{\partial t}(t_r, \mathbf{x}_r) + \mathcal{N}[\mathbf{u}_\theta](t_r, \mathbf{x}_r)$$

If we compare this residual with Equation (2.1), we know that it must be 0

Residual loss

```python
u_t = torch.autograd.grad(
    u, t,
    grad_outputs=torch.ones_like(u),
    create_graph=True,
    retain_graph=True
)[0]
```

```python
u_x = torch.autograd.grad(
    u, x,
    grad_outputs=torch.ones_like(u),
    create_graph=True,
    retain_graph=True
)[0]
```

$$\mathcal{R}_\theta(t, \mathbf{x}) = \frac{\partial \mathbf{u}_\theta}{\partial t}(t_r, \mathbf{x}_r) + \mathcal{N}[\mathbf{u}_\theta](t_r, \mathbf{x}_r)$$

Using automatic differentiation, we can calculate the residual for each (t,x) in out training database, giving us the residual loss:

$$\mathcal{L}_r(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}_\theta(t_r^i, \mathbf{x}_r^i) \right|^2$$

Initial condition

$$\mathbf{u}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \Omega,$$

The initial condition is a function that must be replicated on t = 0 (ODE or PDE)

The loss here can be defined as the distance on t=0 to g(x):

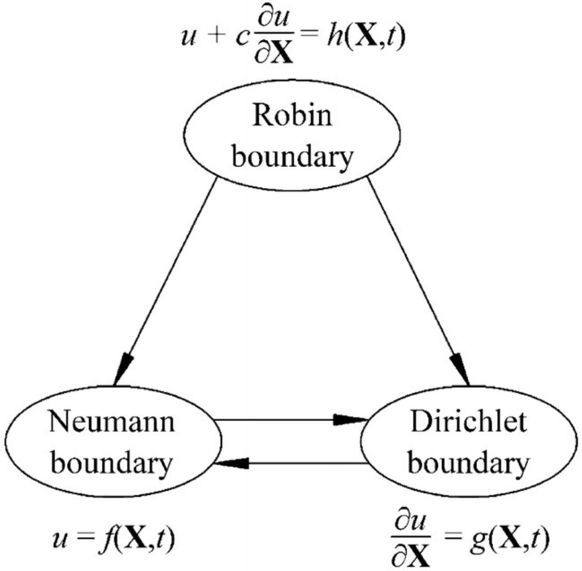
$$\mathcal{L}_{ic}(\theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left| \mathbf{u}_\theta(0, \mathbf{x}_{ic}^i) - \mathbf{g}(\mathbf{x}_{ic}^i) \right|^2$$

```
# Initial condition points (t=0)
x_ic = torch.rand(n_initial, 1).to(device)
t_ic = torch.zeros(n_initial, 1).to(device)
u_ic = initial_condition(x_ic)
```

Boundary conditions

$$\mathcal{B}[\mathbf{u}] = 0, \quad t \in [0, T], \ \mathbf{x} \in \partial\Omega$$

In a PDE, the boundary condition define how the function behaves in the boundary of the spatial domain. B[u(x,t)] must be zero for all t,x in boundary.



$$\mathcal{L}_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \left| \mathcal{B}[\mathbf{u}_\theta](t_{bc}^i, \mathbf{x}_{bc}^i) \right|^2$$

```
# Boundary condition points (x=0 and x=1)
t_bc = torch.rand(n_boundary, 1).to(device)
x_bc_left = torch.zeros(n_boundary, 1).to(device)
x_bc_right = torch.ones(n_boundary, 1).to(device)
u_bc = boundary_condition(t_bc)
```

Error function

$$\mathcal{L}(\theta) = \mathcal{L}_{ic}(\theta) + \mathcal{L}_{bc}(\theta) + \mathcal{L}_r(\theta)$$

Equation

$$\mathcal{L}_r(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{R}_\theta(t_r^i, \mathbf{x}_r^i) \right|^2$$

```
# 1. PDE residual loss
residual = compute_pde_residual(model, x_pde, t_pde, alpha)
loss_pde = torch.mean(residual ** 2)
```

Initial condition

$$\mathcal{L}_{ic}(\theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left| \mathbf{u}_\theta(0, \mathbf{x}_{ic}^i) - \mathbf{g}(\mathbf{x}_{ic}^i) \right|^2$$

```
# 2. Initial condition loss
u_pred_ic = model(x_ic, t_ic)
loss_ic = torch.mean((u_pred_ic - u_ic) ** 2)
```

Boundary conditions

$$\mathcal{L}_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \left| \mathcal{B}[\mathbf{u}_\theta](t_{bc}^i, \mathbf{x}_{bc}^i) \right|^2$$

```
# 3. Boundary condition loss
u_pred_bc_left = model(x_bc_left, t_bc)
u_pred_bc_right = model(x_bc_right, t_bc)
loss_bc = torch.mean((u_pred_bc_left - u_bc) ** 2) +
          torch.mean((u_pred_bc_right - u_bc) ** 2)
```

```
# Total loss
loss = loss_pde + loss_ic + loss_bc
```

Training loop (see implementation)

For each step:

1. Compute points for boundary and initial conditions so we can evaluate the losses
2. Compute points uniformly on the domain so we can evaluate residual loss
3. Backpropagate the gradients from the loss
4. Update the parameters

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations - ScienceDirect

Burgers equation

In one space dimension, the Burger's equation along with Dirichlet boundary conditions reads as

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$

$$u(0, x) = -\sin(\pi x),$$

$$u(t, -1) = u(t, 1) = 0.$$

Burgers equation

In one space dimension, the Burger's equation along with Dirichlet boundary conditions reads as

$$
\begin{aligned}
& u_t + u u_x - (0.01/\pi) u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\
& u(0, x) = -\sin(\pi x), \\
& u(t, -1) = u(t, 1) = 0.
\end{aligned}
$$

Let us define $f(t, x)$ to be given by

$$
f := u_t + u u_x - (0.01/\pi) u_{xx},
$$

Burgers equation

In one space dimension, the Burger's equation along with Dirichlet boundary conditions reads as

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx},$$

$$MSE = MSE_u + MSE_f,$$

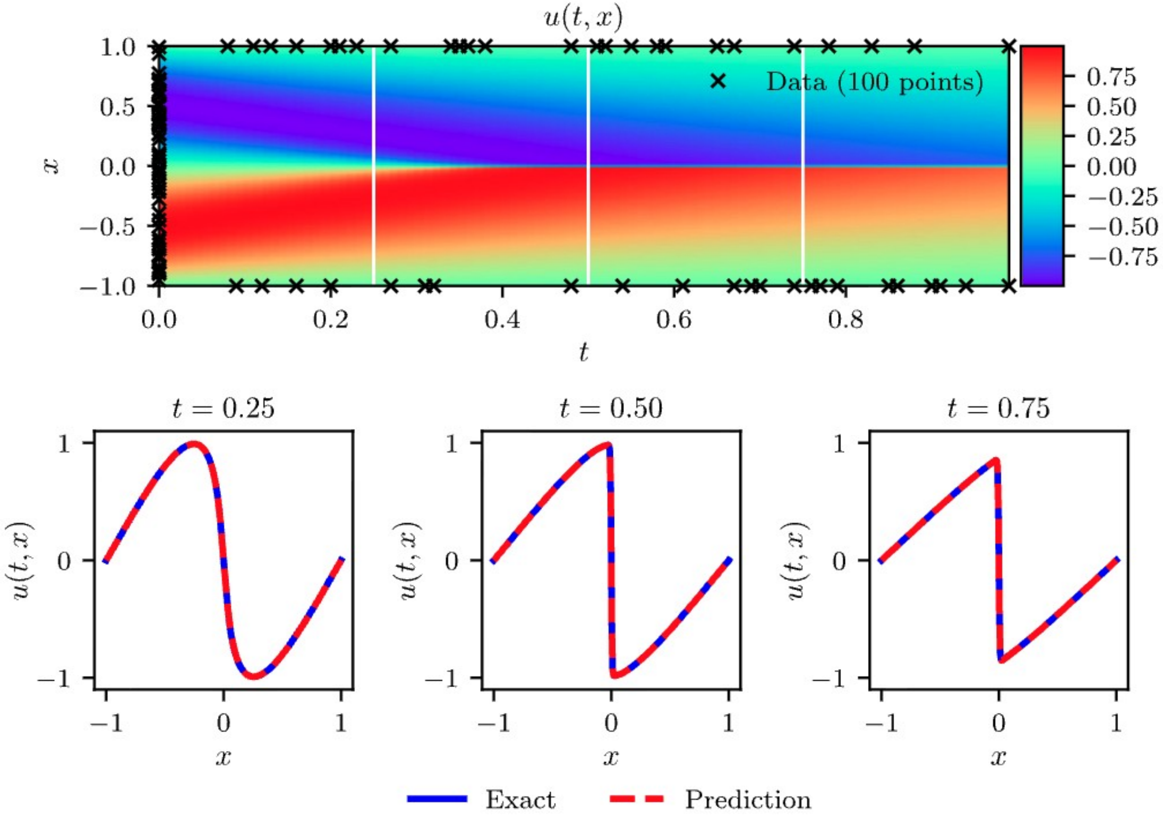
where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

Burgers equation



Schrödinger equations

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2],$$

$$h(0, x) = 2\,\mathrm{sech}(x),$$

$$h(t, -5) = h(t, 5),$$

$$h_x(t, -5) = h_x(t, 5),$$

Schrödinger equations

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2],$$

$$h(0, x) = 2\,\mathrm{sech}(x),$$

$$h(t, -5) = h(t, 5),$$

$$h_x(t, -5) = h_x(t, 5),$$

$$f := ih_t + 0.5h_{xx} + |h|^2 h$$

Schrödinger equations

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2],$$

$$h(0, x) = 2\,\mathrm{sech}(x),$$

$$h(t, -5) = h(t, 5),$$

$$h_x(t, -5) = h_x(t, 5),$$

$$f := ih_t + 0.5h_{xx} + |h|^2 h$$

$$MSE = MSE_0 + MSE_b + MSE_f,$$

where

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} |h(0, x_0^i) - h_0^i|^2,$$

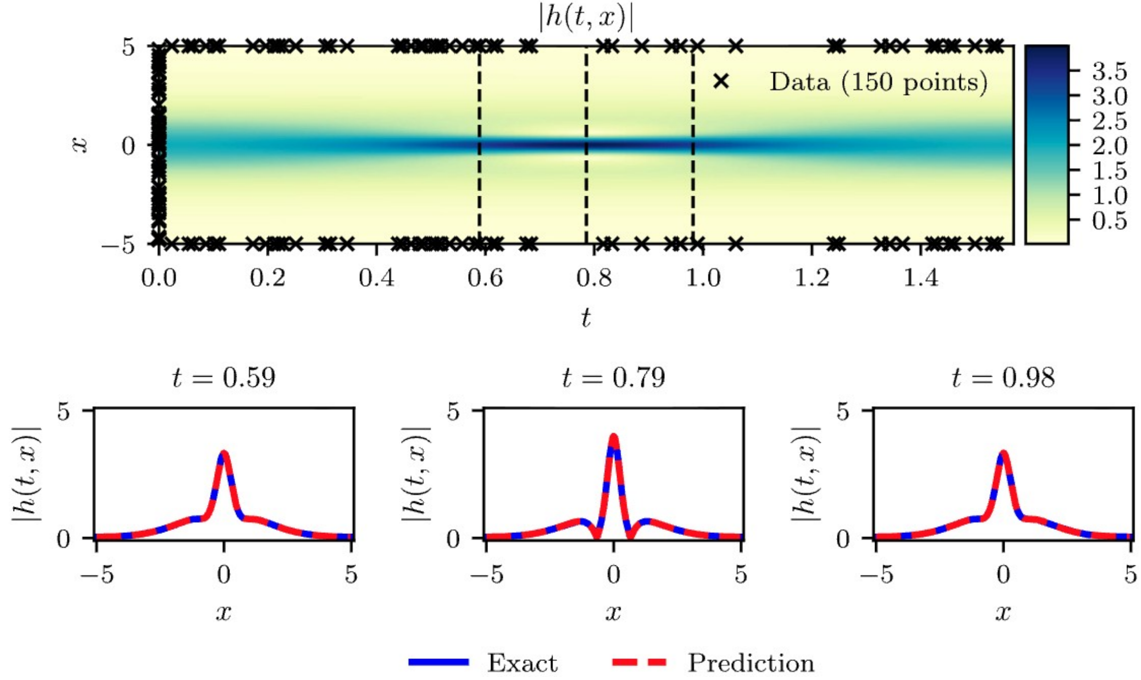
$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b} \left(|h^i(t_b^i, -5) - h^i(t_b^i, 5)|^2 + |h_x^i(t_b^i, -5) - h_x^i(t_b^i, 5)|^2\right),$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

Schrödinger equations

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2],$$
$$h(0, x) = 2\,\mathrm{sech}(x),$$
$$h(t, -5) = h(t, 5),$$
$$h_x(t, -5) = h_x(t, 5),$$



$$f := ih_t + 0.5h_{xx} + |h|^2 h$$

(f evaluated on 20.000 points)

Navier-Stokes system of equations

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy})$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy})$$

where $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and $p(t, x, y)$ the pressure.

$$u_x + v_y = 0$$

Navier-Stokes system of equations

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy})$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy})$$

where $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and $p(t, x, y)$ the pressure.

$$u_x + v_y = 0 \quad \longrightarrow \quad u = \psi_y, \quad v = -\psi_x$$

Navier-Stokes system of equations

$$u = \psi_y, \quad v = -\psi_x$$

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy})$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy})$$

where $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and $p(t, x, y)$ the pressure.

$$f := u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy})$$
$$g := v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy})$$

Navier-Stokes system of equations

$$u = \psi_y, \quad v = -\psi_x$$

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy})$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy})$$

where $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and $p(t, x, y)$ the pressure.

$$f := u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy})$$
$$g := v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy})$$

We approximate $\left[\psi(t, x, y) \quad p(t, x, y)\right]$ using:

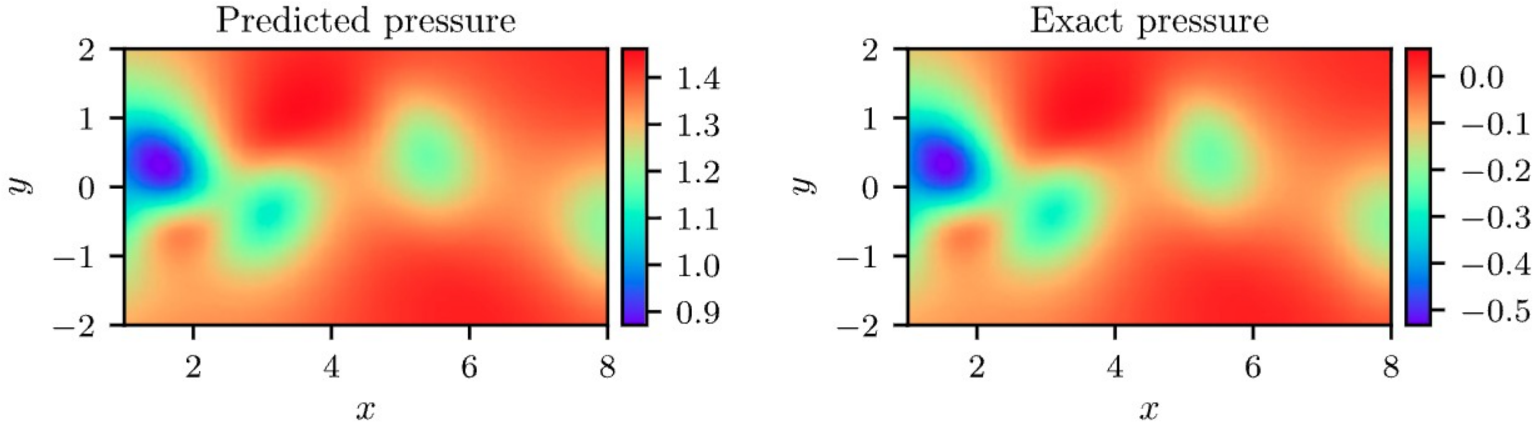
$$MSE := \frac{1}{N}\sum_{i=1}^{N}\left(|u(t^i, x^i, y^i) - u^i|^2 + |v(t^i, x^i, y^i) - v^i|^2\right) + \frac{1}{N}\sum_{i=1}^{N}\left(|f(t^i, x^i, y^i)|^2 + |g(t^i, x^i, y^i)|^2\right)$$

Navier-Stokes system of equations

$$u = \psi_y, \quad v = -\psi_x$$

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy})$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy})$$

where $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and $p(t, x, y)$ the pressure.



Exercises

1. Finish the implementation for burgers equation
2. Solve the following system of equations, known as Lotka-Volterra:

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y$$