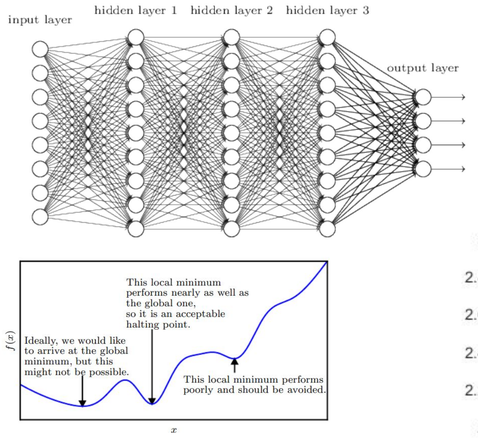
Comparing PINN to Classical Methods

Joey Fishback and Mark Sussman
Florida State University
Department of Mathematics,Tallahassee, Florida, USA

FSU UNDERGRADUATE RESEARCH OPPORTUNITY PROGRAM
CENTER FOR UNDERGRADUATE RESEARCH & ACADEMIC ENGAGEMENT

Background:

- Differential equations are equations that contain derivatives.
- They are challenging to solve because of their complexity.
- Modelling system with limited and noisy data is often a challenge, or can be computationally expensive.
- Neural networks with purely data driven approach often required large amounts of sample data to accurately train.
- Physics informed neural network incorporate known physical laws into loss function of the network to ensure that the network is not just training to the data.



Mathematical Representation

Let's define:

- Input vector: $\mathbf{x} \in R^n$ (n features)
- Weight matrix: $\mathbf{W} \in R^{m \times n}$ (m neurons, n inputs per neuron)
- Bias vector: $\mathbf{b} \in R^m$ (one bias term per neuron)
- Output vector: $\mathbf{y} \in R^m$ (output of m neurons)
- Activation function: f

The operation of a fully connected layer can be written as:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Mean Squared Error Loss $= \frac{1}{N}\sum_{i=1}^{N} \|Y_{\text{pred}}^{(i)} - Y_{\text{exact}}^{(i)}\|^2$

The L2 norm (Euclidean norm) is used: $\|x\| = \sqrt{\sum_i x_i^2}$

Methods:

- Begin with understanding and creating numerical methods
- Runge-Kutta 4 , Runge-kutta-6, Forward Euler, Backward Euler
- create Data-driven neural network to solve ODE/PDE
- Create Physics Informed Neural Network.
- Compare.

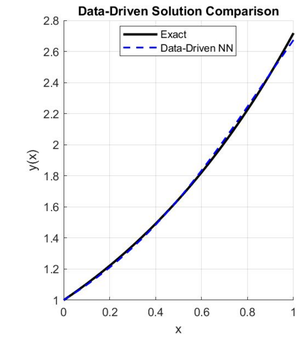
Numerical Methods for Solving Differential Equations

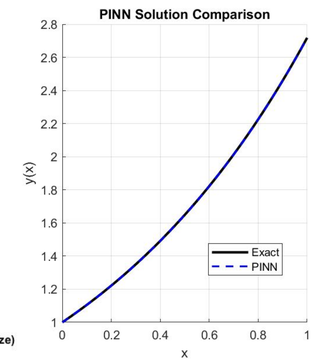
Main Equation: $\frac{dy}{dt} = f(t, y)$

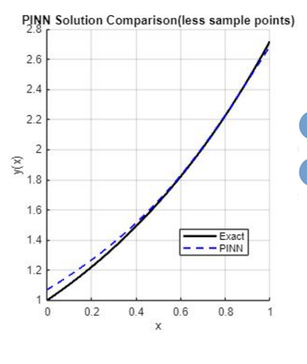
Forward Euler	Runge-Kutta 2	Backward Euler	Runge-Kutta 4	Runge-Kutta 6
$y_{n+1} = y_n + hf(t_n, y_n)$	$k_1 = f(t_n, y_n)$ $k_2 = f(t_n + h, y_n + hk_1)$ $y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2)$	$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$	$k_1 = f(t_n, y_n)$ $k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$ $k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$ $k_4 = f(t_n + h, y_n + hk_3)$ $y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$	$k_1 = f(t_n, y_n)$ $k_2 = f(t_n + \frac{h}{3}, y_n + \frac{h}{3}k_1)$ $k_3 = f(t_n + \frac{h}{3}, y_n + \frac{h}{6}(k_1 + k_2))$ $k_4 = f(t_n + \frac{h}{2}, y_n + \frac{h}{8}(k_1 + 3k_3))$ $k_5 = f(t_n + h, y_n + h(k_1 - 3k_3 + 4k_4))$ $k_6 = f(t_n + \frac{2h}{3}, y_n + \frac{h}{27}(7k_1 + 10k_2 + k_4 + 8k_5))$ $y_{n+1} = y_n + \frac{h}{48}(7k_1 + 6k_2 + 8k_4 + 23k_5 + 4k_6)$

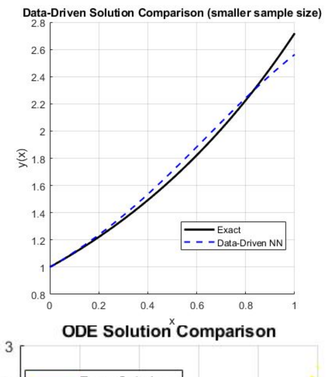
Variable Definitions

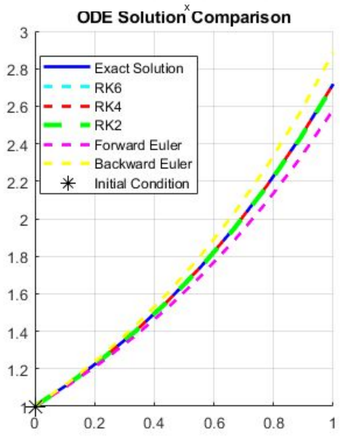
h: Step size — t_n: Current time
y_n: Current solution value — $f(t, y)$: Right-hand side of the differential equation

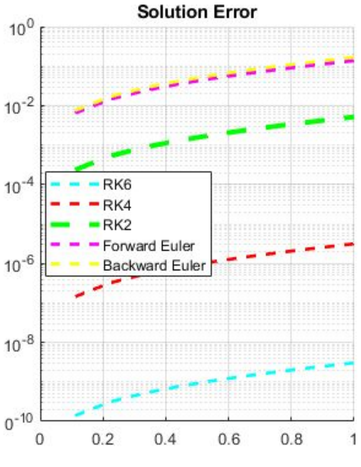




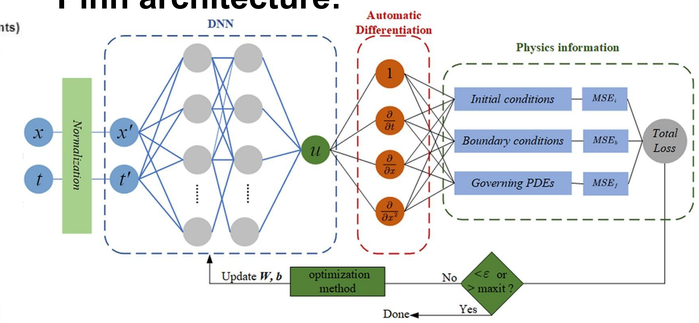








Pinn architecture:



Comparison:

```
PINN Network Information and Training Parameters
=================================================================
Parameter                | Value
-----------------------------------------------------------------
Network Size             | 3 layers (input→32→16→1)
Activation Functions     | tanh (after layer 1 and 2)
Number of Weights        | 578
Training Size            | 10000
Learning Rate            | Initial: 0.0100, Drop Factor: 0.70
Drop Period: 10 epochs
Momentum                 | 0.95
Mini-batch Size          | 200
Number of Epochs         | 50
Loss Weights             | Physics: 1.0, IC: 10.0

PINN ODE Solver Performance:
=================================================================
Compute Time (s) | Memory (MB) |    MSE     | IC Error
-----------------------------------------------------------------
      55.812615 |       14.93 |  4.24e-07 | 2.81e-05

Data-Driven ODE Solver Performance:
=================================================================
Compute Time (s) | Memory (MB) |    MSE     | IC Error
-----------------------------------------------------------------
      22.006378 |       38.98 |  1.88e-04 | 8.76e-04
```

```
Network Information and Training Parameters
=================================================================
Parameter                | Value
-----------------------------------------------------------------
Network Size             | 3 layers (input→32→16→1)
Activation Functions     | tanh (after layer 1 and 2)
Number of Weights        | 578
Training Size            | 220
Learning Rate            | Initial: 0.0100, Drop Factor: 0.70
Drop Period: 10 epochs
Momentum                 | 0.95
Mini-batch Size          | 20
Number of Epochs         | 50 (with early stopping)
IC Coefficient           | 10

Numerical Method Performance:
=========================================
Method          | Compute Time (s) |    MSE
-----------------------------------------
RK6             |         0.018249 |  0.000000
RK4             |         0.006231 |  0.000000
RK2             |         0.004946 |  0.000007
Forward Euler   |         0.004177 |  0.004701
Backward Euler  |         0.004848 |  0.006965
```

Expected Results:

- For low dimensional problems Classical methods will often be a superior choice, as is the case with this simple ODE.
- As the dimension of the problems increase classical methods often become to computationally expensive to use.
- PINN ability to interpolate solutions on their domain, while also working with noisy datasets in higher dimensional problem may give them an advantage when solving certain classes of PDEs

References: