PINN to Nonlinear PDE Models Info

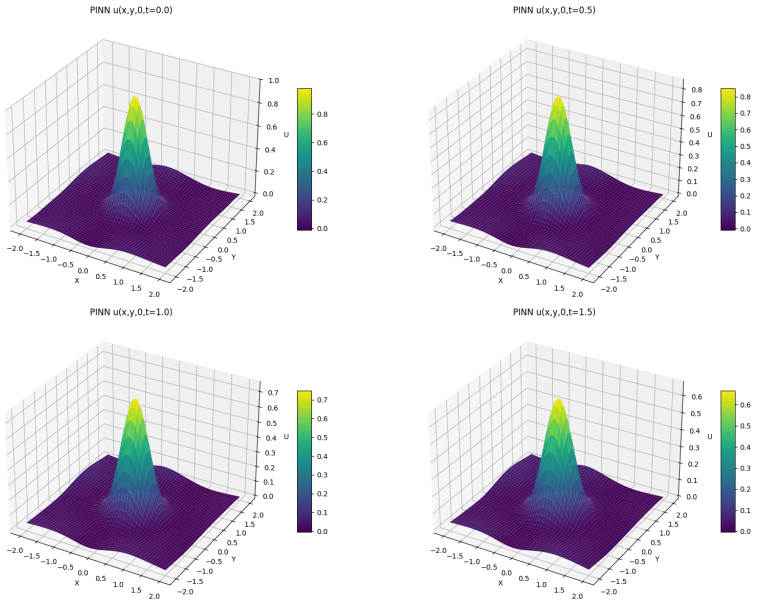
Riemann problem:

1. Riemann_first.pt {Not Stable}
    - Riemann_first_plt.png
    - 10k epochs
    - Tanh activation
    - No loss epochs
    - Trained in domain x = [-1, 1], t = [0, 1]
2. 0Riemann_40k_PINN.pt
    - 0Loss_Riemann_40k_PINN_plt.png
    - 40k epochs
    - Tanh Activation
    - No loss Weights
    - Trained in domain x = [-1, 1], t = [0, 1]
3. Riemann_40k_PINN.pt
    - Loss_Riemann_40k_PINN_plt.png
    - 40k epochs
    - Tanh Activation
    - 4 Step loss weights
    - Trained in domain x = [-1, 1], t = [0, 1]
4. 2Riemann_40k_PINN.pt
    - 2Loss_Riemann_40k_PINN_plt.png
    - 40k epochs
    - Tanh Activation
    - Gradually changing loss weights
    - Trained in domain x = [-1, 1], t = [0, 1]
5. Riemann_50k_ReLU_PINN.pt
    - Riemann_50k_ReLU_PINN.png
    - Not Stable graph but stable Results in range
    - 50k epochs
    - ReLU Activation
    - No loss Weights
    - Trained in domain x = [-5, 5], t = [0, 3]
6. Riemann_50k_ELU_PINN.pt
    - Riemann_50k_ELU_PINN.png
    - **Stable Loss plot but not good results**
    - 50k epochs
    - ELU Activation
    - No loss Weights
    - Trained in domain x = [-5, 5], t = [0, 3]

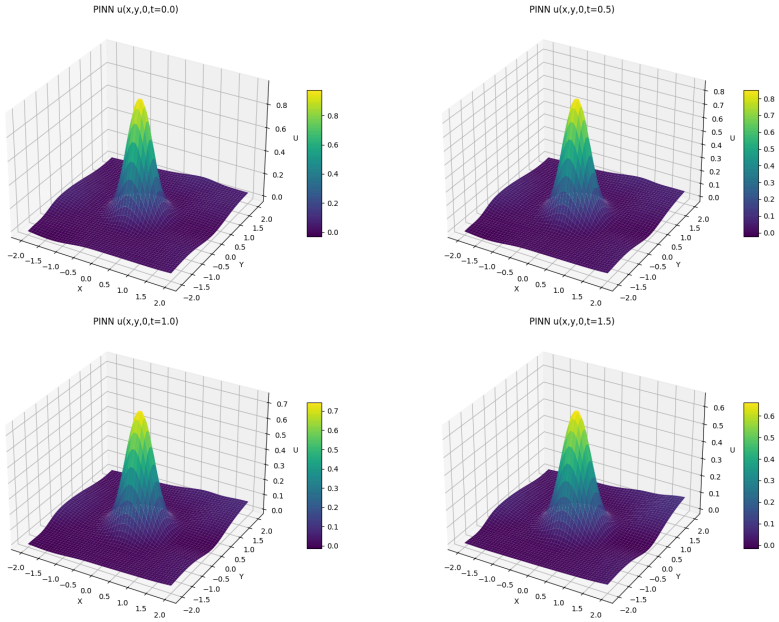
7. Riemann_50k_SELU_PINN.pt
    - Riemann_50k_SELU_PINN.png
    - **Stable Loss plot but not good results**
    - 50k epochs
    - SELU Activation
    - No loss Weights
    - Trained in domain x = [-5, 5], t = [0, 3]
8. Riemann_50k_LeakyReLU_PINN.pt
    - Riemann_50k_LeakyReLU_PINN.png
    - *UnStable Loss plot but good results*
    - 50k epochs
    - Leaky ReLU Activation
    - No loss Weights
    - Trained in domain x = [-5, 5], t = [0, 3]
9. Riemann_50k_ParametricReLU_PINN.pt
    - Riemann_50k_ParametricReLU_PINN.png
    - ***UnStable Loss plot and not good results***
    - 50k epochs
    - Parametric ReLU Activation
    - No loss Weights
    - Trained in domain x = [-5, 5], t = [0, 3]

Heat Equation

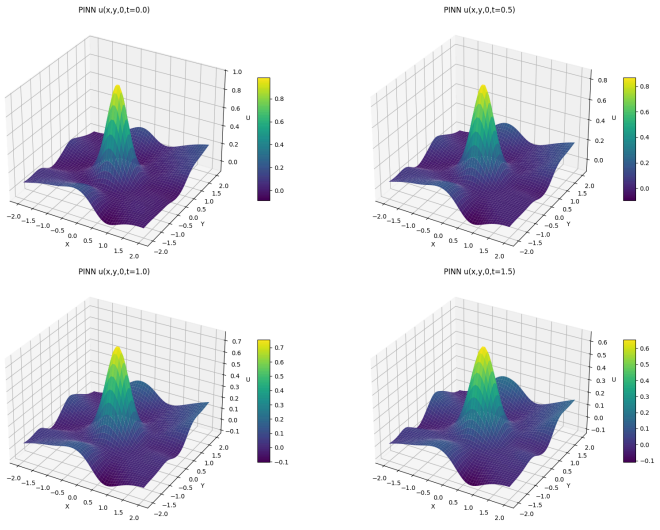
1. 1HeatEqn_30k_tanh_NN.pt
    - 1HeatEqn_30k_tanh_NN.png
    - Layers = [4, 64, 64, 64, 64, 1]
    - 30k epochs; {Stable Loss after 25k}
    - Tanh activation
    - X , Y , Z = (-1.0, 1.0)^3, T = [0,1]
    - Matches with analytical results at t = 1.5 also
    - But doesn't match for **{X , Y , Z} > 1 and < -1**



2. [2HeatEqn_30k_tanh_NN.pt]
   - [2HeatEqn_30k_tanh_NN.png]
   - Layers = [4, 32, 16, 8, 1]
   - 30k epochs; {Very Stable Loss}
   - Similar Results to above model

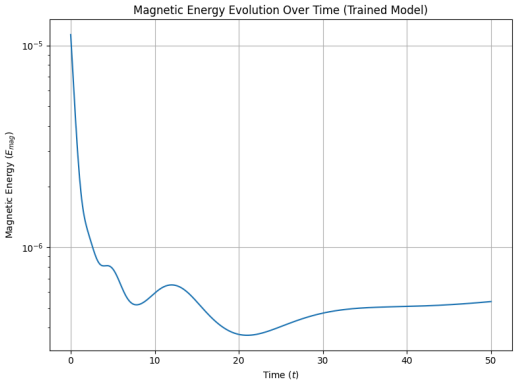


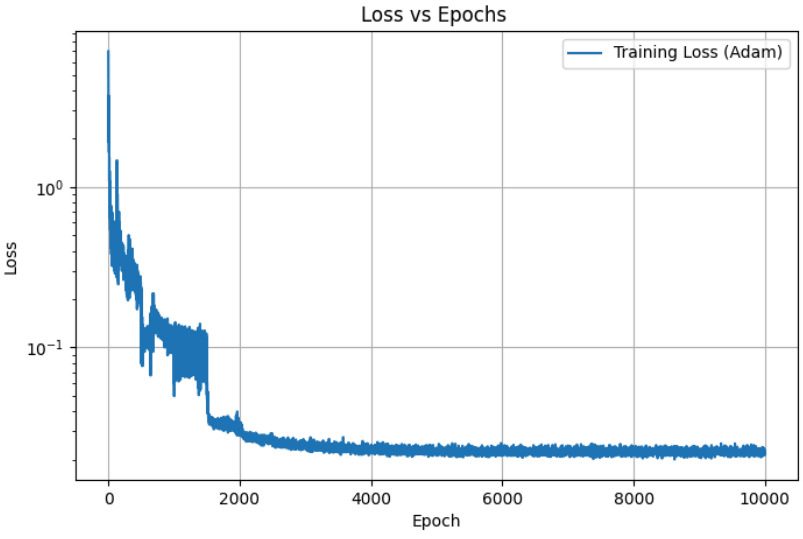
3. [3HeatEqn_30k_tanh_NN.pt]
   - [3HeatEqn_30k_tanh_NN.png]
   - Layers = [4, 32, 8, 1]
   - 30k epochs; {Very Stable Loss}
   - Diverges more at **{X , Y , Z} > 1 and < -1** (Outside the trained domain)

- 

Kinematic dynamo action

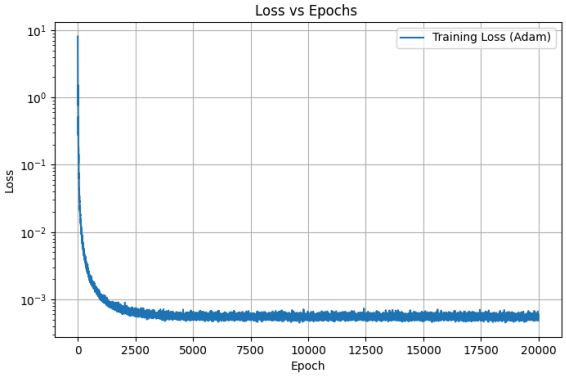
1. [kinematic_dynamo_action_10k_tanh_NN_10lossicW_Rem100.pt]

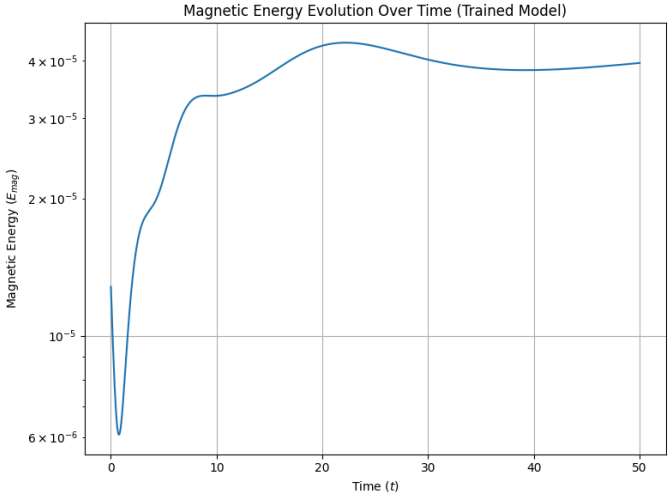
- 

- 

- 10k epochs
  - Re_m = 100.0
  - A, B, C = 1.0, 1.0, 1.0
  - k_abc = 2
  - domain_min = 0.0
  - domain_max = 2*np.pi
  - T_max = 40.0
  - loss_pde = torch.mean(torch.sqrt(fBx**2 + fBy**2 + fBz**2))

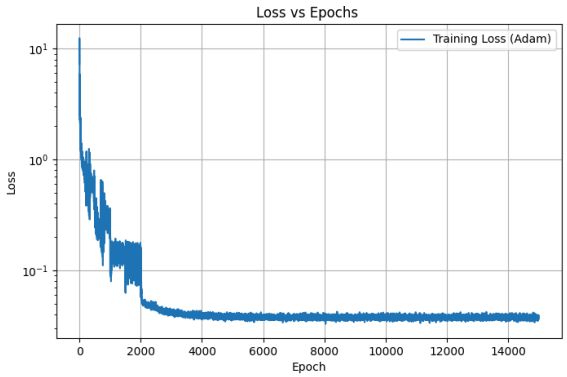
kinematic_dynamo_action_20k_tanh_NN_10lossicW_.pt

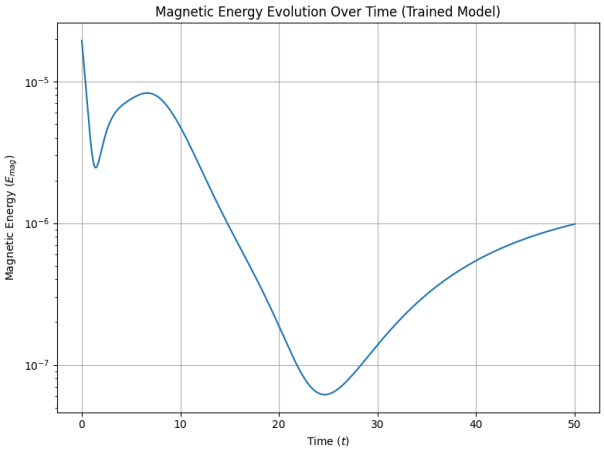




- For Rem = 1600
- loss_pde = torch.mean(fBx**2 + fBy**2 + fBz**2)

kinematic_dynamo_action_0k_tanh_NN_10lossicW_K=3.pt

- 

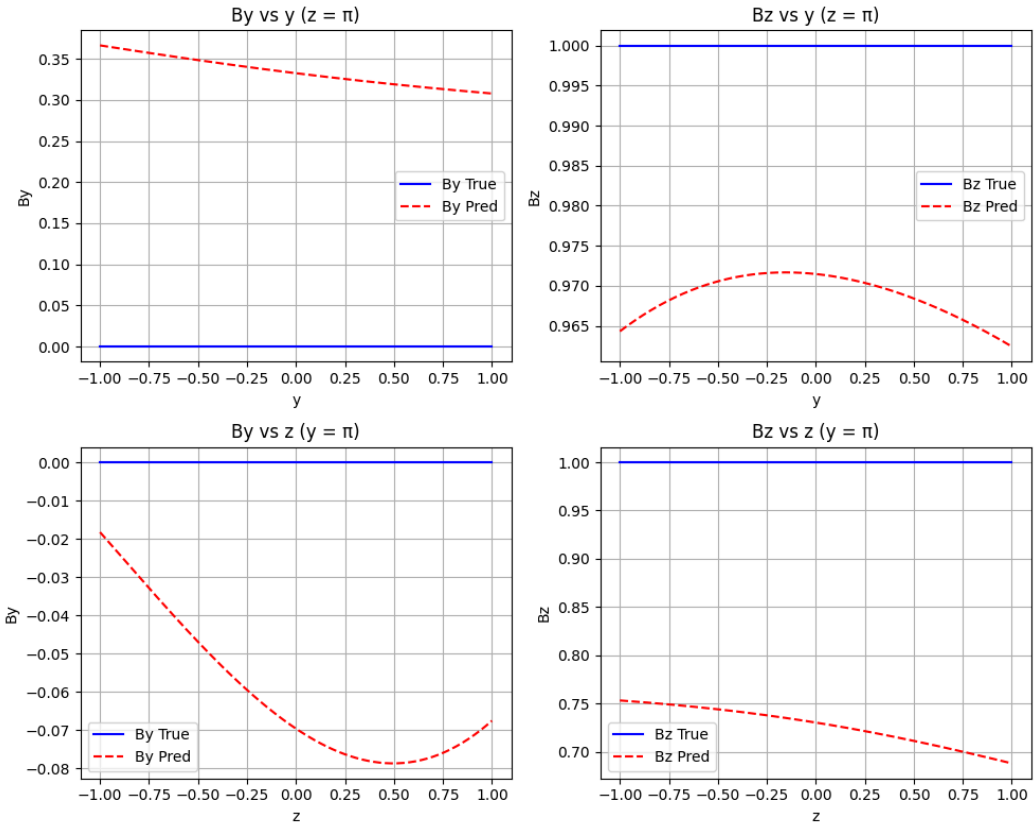
- 

-
- loss_pde = torch.mean(torch.sqrt(fBx**2 + fBy**2 + fBz**2))
- loss = 10.0*loss_pde + 10.0*loss_div + 1.0*loss_ic # weights can be tuned
- K = 3, Rem = 100

2D Kinematic dynamo action

B(t = 0) = B0 zcap {at t = 0}
u = uo*sin(z/2) ycap

1. 2D_kinematic_dynamo_action_10k_tanh_x,y,z.pt
    - 2D_kinematic_dynamo_action_10k_tanh_x,y,z_Loss.png
    - model = PINN_Magnetic3D([4, 64, 64, 64, 3]).to(device)
    - 10k epochs
    - Tanh layer



2. [2D_kinematic_dynamo_action_10k.pt]
    - [2D_kinematic_dynamo_action_10k_Loss.png]
3. [2D_kinematic_dynamo_action_10k_64*4-hiddenlayers.pt]
    - [2D_kinematic_dynamo_action_10k_64*4-hiddenlayers_Loss.png]
4. [2D_kinematic_dynamo_action_10k_128*4-hiddenlayers.pt]
    - [2D_kinematic_dynamo_action_10k_128*4-hiddenlayers_Loss.png]
5. [2D_kinematic_dynamo_action_10k_64*2-hiddenlayers.pt]
    - [2D_kinematic_dynamo_action_10k_64*2-hiddenlayers_Loss.png]
6. [2D_kinematic_dynamo_action_20k_[3, 128, 64, 32, 16, 8, 4, 2]-cos-Relu.pt]
    - [2D_kinematic_dynamo_action_20k_[3, 128, 64, 32, 16, 8, 4, 2]-cos-Relu_Loss.png]
    - Layers = [3, 128, 64, 32, 16, 8, 4, 2]
    - Epochs = 20k
    - u = u0 * cos(pi * z / 2)
    - Activation function = ReLu

IMPROVE:

1. Adaptive λ_pde, λ_ic like the [paper]
2. Time-slab training

3. Causal time weighting
4. Residual connections
5. Fourier feature mapping
6. Higher-dimensional PDE handling