5.8 Computational Exercise: Physics Informed Neural Networks (PINNs)

In this exercise, you will use feed-forward networks to solve the nonlinear diffusion-advection-reaction equation

$$\mathcal{L}u \equiv u''(x) - \mathrm{Pe}\, u'(x) + \mathrm{Da}\, u(1-u) = 0 \qquad x \in (0,1) \tag{5.24}$$

$$u(0) = 0 \tag{5.25}$$

$$u(1) = 1. \tag{5.26}$$

In this equation, the non-dimensional number Pe is the Peclet number which measures the strength of advection relative to diffusion, and Da is the Damkohler number, which measures the strength of reaction to diffusion.

1. Use the MLP class developed previously to create a feed-forward network of `width=18` and `depth=6`, and with input and output dimensions set to 1. This will be used to represent the function $u = u(x;\boldsymbol{\theta})$ which will be solution to the PDE above. The weights and biases ($\boldsymbol{\theta}$) should be initialized using a standard normal distribution (zero mean and unitary standard deviation). All hidden layers should make use of a `tanh` activation function, while no activation should be used in the output layer. Use l_2 regularization in all layers with a parameter 1e-7.
2. Create an array of $N = 100$ uniformly spaced points in $[0, 1]$. Train a neural network with the following loss function

$$\Pi(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} (\mathcal{L}u(x_i;\boldsymbol{\theta}))^2 + \lambda_b\left(u(0;\boldsymbol{\theta})^2 + (u(1;\boldsymbol{\theta}) - 1)^2\right)$$

which is the sum of the interior residual and a scaled boundary residual.

```
def loss_fun(x_train, model, Pe, Da):
    u = model(x_train)
    u_x = torch.autograd.grad(
            u, x_train,
            grad_outputs=torch.ones_like(x_train),
            create_graph=True)[0]
    u_xx = torch.autograd.grad(
            u_x, x_train,
            grad_outputs=torch.ones_like(x_train),
            create_graph=True)[0]
    R_int = torch.mean(torch.square(u_xx - Pe * u_x + Da * u *
        (1.0 - u)))
    R_bc  = torch.square(u[0]) + torch.square(u[-1] - 1.0)

    return R_int,R_bc
```

3. Use a learning rate of 1e-4 and $\lambda_b = 10$ for the training. Consider four different sets of values for the non-dimensional parameters:

   - $P_e = 0.01$, $Da = 0.01$ (diffusion dominates).
   - $P_e = 20$, $Da = 0.01$ (advection dominates).
   - $P_e = 0.01$, $Da = 60$ (reaction dominates).
   - $P_e = 15$, $Da = 40$ (advection and reaction dominate).

4. For each set of parameter values with $\lambda_b = 10$

   - Train the network for 40,000 epochs **without mini-batches**. Save the history of the interior loss and boundary loss.
   - Generate a plot of the interior and boundary as a function of epoch number. One plot for each set of parameter values.
   - Using the fully trained network (at the end of 40,000 epochs) evaluate predicted solution at the test points contained in the first column of the reference solution files provided to you.
   - Generate a plot of the network solution and the reference solution as a function of the spatial coordinate, x. One plot for each set of parameter values.

5. Repeat the previous step for $\lambda_b = 300$ and $\lambda_b = 10{,}000$.
6. Based on the results obtained answer the following questions:

   - Which set of parameter values is the hardest to solve for?
   - Does the same value of λ_b work for all sets of parameter values? Provide a justification for your observations.
   - Extra credit: try different options (activation functions, network size, learning rate, regularization parameter, and the hyperparameters) to see if you can arrive at a good solution for the most challening set of parameters.