Training the FNO to Solve the 1D Wave Equation

In this exercise, you will train a model to approximate the solution of the 1D wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \quad t \in (0, 1], \quad x \in [0, 1], \quad c = 0.5$$

with boundary conditions

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

and initial conditions

$$u(x, 0) = u_0(x), \quad u_t(x, 0) = 0$$

where u_t is the partial derivative with respect to time. Note that the initial conditions are sampled from an unknown distribution. Knowing the exact expression for u_0 is not necessary to complete the tasks below.

You should use a **Fourier Neural Operator (FNO)** for all the tasks.

Dataset Details

You are provided with the following datasets in this **folder**:

1. **Training Dataset:** `train_sol.npy`
   - Shape: $(128, 5, 64)$
   - Description:
     - 128: Number of trajectories.
     - 5: Time snapshots of the solution. For a given trajectory u, the time snapshots are:

       $u[0]$: Initial condition u_0 at $t = 0.0$,
       $u[1]$: Solution at $t = 0.25$,
       $u[2]$: Solution at $t = 0.50$,
       $u[3]$: Solution at $t = 0.75$,
       $u[4]$: Solution at $t = 1.0$.

     - 64: Spatial resolution of the data.
     - Please see Figure 1 for visualization.

2. **Testing Datasets:**
   - `test_sol.npy`: Similar shape as the training dataset $(128, 5, 64)$. Contains 128 trajectories with all 5 time snapshots.
   - `test_sol_res_{s}.npy`: Testing datasets at varying spatial resolutions $s \in \{32, 64, 96, 128\}$. Shape: $(128, 2, s)$, where:

     $u[0]$: Initial condition u_0 at $t = 0.0$,
     $u[1]$: Solution at $t = 1.0$.

     Note: Intermediate time snapshots are not included.

- `test_sol_OOD.npy`: Out-of-distribution (OOD) testing dataset. Shape: $(128, 2, 64)$, where:

$$u[0]\text{: Initial condition } u_0 \text{ at } t = 0.0,$$
$$u[1]\text{: Solution at } t = 1.0.$$

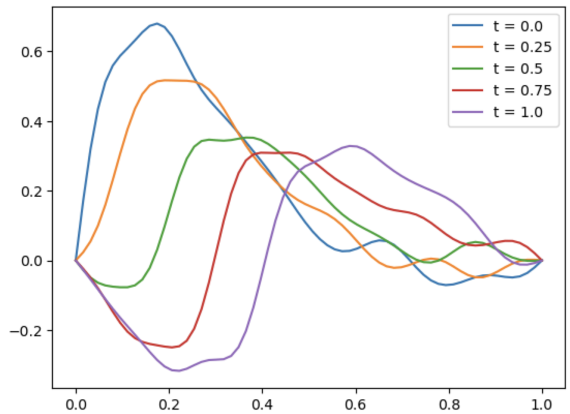

Figure 1: Training Dataset - One trajectory

Tasks

Task 1: One-to-One Training

1. Use **64** trajectories from the training dataset.
2. Select the first ($t = 0.0$) and last ($t = 1.0$) time snapshots for these trajectories.
3. Train an FNO model to learn the mapping:

$$G : u_0 \to u(t = 1.0)$$

4. Use the remaining trajectories for validation.
5. Test the trained model on the `test_sol.npy` dataset, focusing **only** on predictions at $t = 1.0$ (the map $u_0 \to u(t = 1.0)$).
6. Report the **average relative L2 error**:

$$\text{err} = \frac{1}{128} \sum_{n=1}^{128} \frac{\|u^{(n)}_{\text{pred}}(t = 1.0) - u^{(n)}_{\text{true}}(t = 1.0)\|_2}{\|u^{(n)}_{\text{true}}(t = 1.0)\|_2}$$

Task 2: Testing on Different Resolutions

1. Test the trained model from Task 1 on the datasets `test_sol_res_{s}.npy` for $s \in \{32, 64, 96, 128\}$.
2. Compute and report the **average relative L2 error** for each dataset.
3. What do you observe about the model's performance across different resolutions?

Task 3: Testing on Out-of-Distribution (OOD) Dataset

1. Test the trained model from Task 1 on the OOD dataset `test_sol_OOD.npy`.
2. Compute and report the **average relative L2 error**.
3. Compare the error to the one obtained in Task 1. What do you observe? Is the error higher or lower?

Task 4: All2All Training

1. Use 64 trajectories from the training dataset.
2. Use **all provided time snapshot**s ($t = 0.0, 0.25, 0.50, 0.75, 1.0$) for these trajectories to train a time-dependent FNO model. Note that this is similar to the task that we had in time-dependent CNO tutorial. *Hint: Use time-conditional batch normalization and include time as one of the input channels.*
3. What is the total number of samples used for training in the **All2All** approach?
4. Test the trained model on the `test_sol.npy` dataset, focusing **only** on predictions at $t = 1.0$.
5. Report the **average relative L2 error**.
6. Compare the error to the one obtained in Task 1. What do you observe?

Bonus Task

1. Use the model from Task 4 to make predictions at multiple time steps: $t = 0.25, t = 0.50, t = 0.75, t = 1.0$.
2. Compute the **average relative L2 error** for each time step.
3. What do you observe about the model's performance over time?
4. Use the model from Task 4 to make predictions on the OOD dataset at $t = 1.0$.
5. What do you observe about the model's performance?