2. Training an FNO to approximate a dynamical system (50 + 10 points)

In this exercise, you will train a **Fourier Neural Operator** to approximate an unknown dynamical system described by

$$\frac{\partial u}{\partial t} = \mathcal{D}(u(x,t)), \quad t \in (0,1], \quad x \in [0,1],$$

with boundary conditions

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

and initial conditions

$$u(x,0) = u_0(x).$$

The initial conditions are sampled from an unknown distribution. As the exact physics of the problem is unknown, you will have to approximate it from the available data.

Note: 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:** `data_train_128.npy`
   - Shape: $(1024, 5, 128)$
   - Description:
     - 1024: 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 $t = 1.0$.

     - 128: Spatial resolution of the data.
     - Please see Figure 2 for visualization.
2. **Validation Dataset:** `data_val_128.npy`
   - Shape: $(32, 5, 128)$
   - This dataset should be used as a validation dataset during the training.
   - The initial conditions $u[0]$ are sampled from the same distribution as in `data_train_128.npy`.
3. **Testing Datasets:** `data_test_{s}.npy`:
   - Testing datasets at varying spatial resolutions $s \in \{32, 64, 96, 128\}$ of the shape $(128, 5, s)$

- The initial conditions $u[0]$ are sampled from the same distribution as in `data_train_128.npy`.

4. **Different Initial Distribution Datasets:**

   (a) `data_finetune_train_unknown_128.npy`:

   - Shape: $(32, 5, 128)$
   - Dataset to be used for **finetunning**.
   - The initial conditions $u[0]$ are sampled from **an unknown distribution, different from** `data_train_128.npy`.

   (b) `data_finetune_val_unknown_128.npy`:

   - Shape: $(8, 5, 128)$
   - Dataset to be used for validation during finetunning.
   - The initial conditions $u[0]$ are sampled from the same distribution as in `data_finetune_train_unknown_128.npy`.

   (c) `data_test_unknown_128.npy`:

   - Shape: $(128, 5, 128)$
   - Dataset to be used for **testing the models on the unknown distribution** of inputs.
   - The initial conditions $u[0]$ are sampled from the same distribution as in `data_finetune_train_unknown_128.npy`.

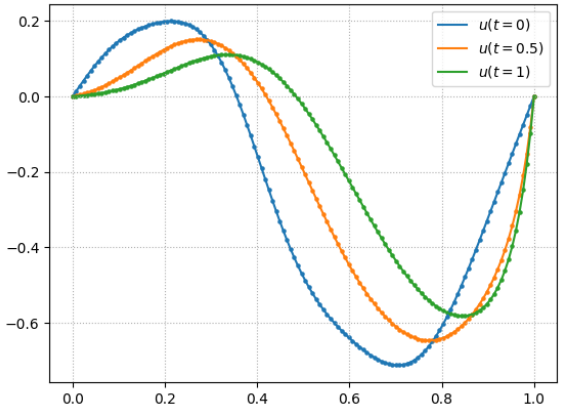

Figure 2: Training Dataset - One trajectory

Tasks

Task 1: One-to-One Training (10 points)

From the **all 1024 trajectories** in the training dataset, keep only the first and last time snapshots, corresponding to $t = 0.0$ and $t = 1.0$. Train an FNO model (**one2one** training) to learn the mapping

$$G : u_0 \mapsto u(t = 1.0).$$

After training, evaluate the model on the `data_test_128.npy` dataset, considering *only* predictions at the final time $t = 1.0$. Report the average relative L^2 error over the 128 test trajectories, defined as

$$\text{err} = \frac{1}{128} \sum_{n=1}^{128} \frac{\left\| u_{\text{pred}}^{(n)}(t = 1.0) - u_{\text{true}}^{(n)}(t = 1.0) \right\|_2}{\left\| u_{\text{true}}^{(n)}(t = 1.0) \right\|_2}.$$

Task 2: Testing on Different Resolutions (10 points)

Test the trained model **from Task 1** on the datasets `data_test_{s}.npy` for $s \in \{32, 64, 96, 128\}$. Compute and report the average relative L2 error for each dataset (at $t = 1.0$). What do you observe about the model's performance across different resolutions?

Task 3: All2All Training (15 points)

Now, use **all provided time snapshot**s ($t = 0.0, 0.25, 0.50, 0.75, 1.0$) of the 1024 training trajectories to **train a time-dependent** FNO model (**all2all** training). Note that this is similar to the task that we had in time-dependent CNO tutorial. *Hint: Use time-conditional normalizations and include time as one of the input channels.*

- **(10 points)** Test the trained model on the `data_test_128.npy` dataset, focusing **only** on predictions at $t = 1.0$. Report the average relative L2 error. Compare the error to the one obtained in Task 1. What do you observe?
- **(5 points)** Now use the model to make predictions at multiple time steps: $t = 0.25, t = 0.50, t = 0.75, t = 1.0$. Compute the average relative L2 error **for each time step**. What do you observe about the model's performance over time?

Task 4: Finetunning (15 points + 10 points)

- **(5 points)** Test the model model trained in Task 3 at $t = 1.0$ of the dataset drawn from an unknown distribution `data_test_unknown_128.npy` in a zero-shor manner. Report the average relative L2 error. How does the error compare to the one obtained in Task 1?
- **(10 points)** Use 32 trajectories from `data_finetune_train_unknown_128.npy` to **finetune the model trained in Task 3** in all2all fashion on the unknown data. Test the finetuned model on `data_test_unknown_128.npy`. Report the average relative L2 error. What is the influence of the finetunning?
- **(BONUS - 10 points)** Use 32 trajectories from `data_finetune_train_unknown_128.npy` to **train a new model from scratch** in all2all fashion on the unknown data. Test the finetuned model on `data_test_unknown_128.npy`. How does the error of the model trained from scratch compare to the error of the finetuned model? Is transfer learning successful?