Neural Network for Solving PDE

MLP for solving beam dynamic problems

# Procedure:

1. [SSB vibration](#_Experiment_1)
   1. [EI=1→4](#_Experiment_1.1)
      1. [t∈[0,1]→[0,0.5]](#_Experiment_1.1.1)
         1. [t∈[0,0.5]→[0,0.3]](#_Experiment_1.1.1.1)
         2. [[2, 100, 100, 100, 1] → [2, 100, 100, 100, 100, 1]](#_Experiment_1.1.1.2)
2. String vibration

# Vibration of Simple Supported Beam (SSB)

Boundary conditions:

## Experiment 1

This experiment is for examining PINN’s capability of solving 4th order PDE.

### Setup:

* Governing equation:
* Initial conditions:
* Loss:

self.loss = tf.reduce\_mean(tf.square(self.w0\_tf - self.w0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_t0\_tf - self.w\_t0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_lb\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_lb\_pred)) + \

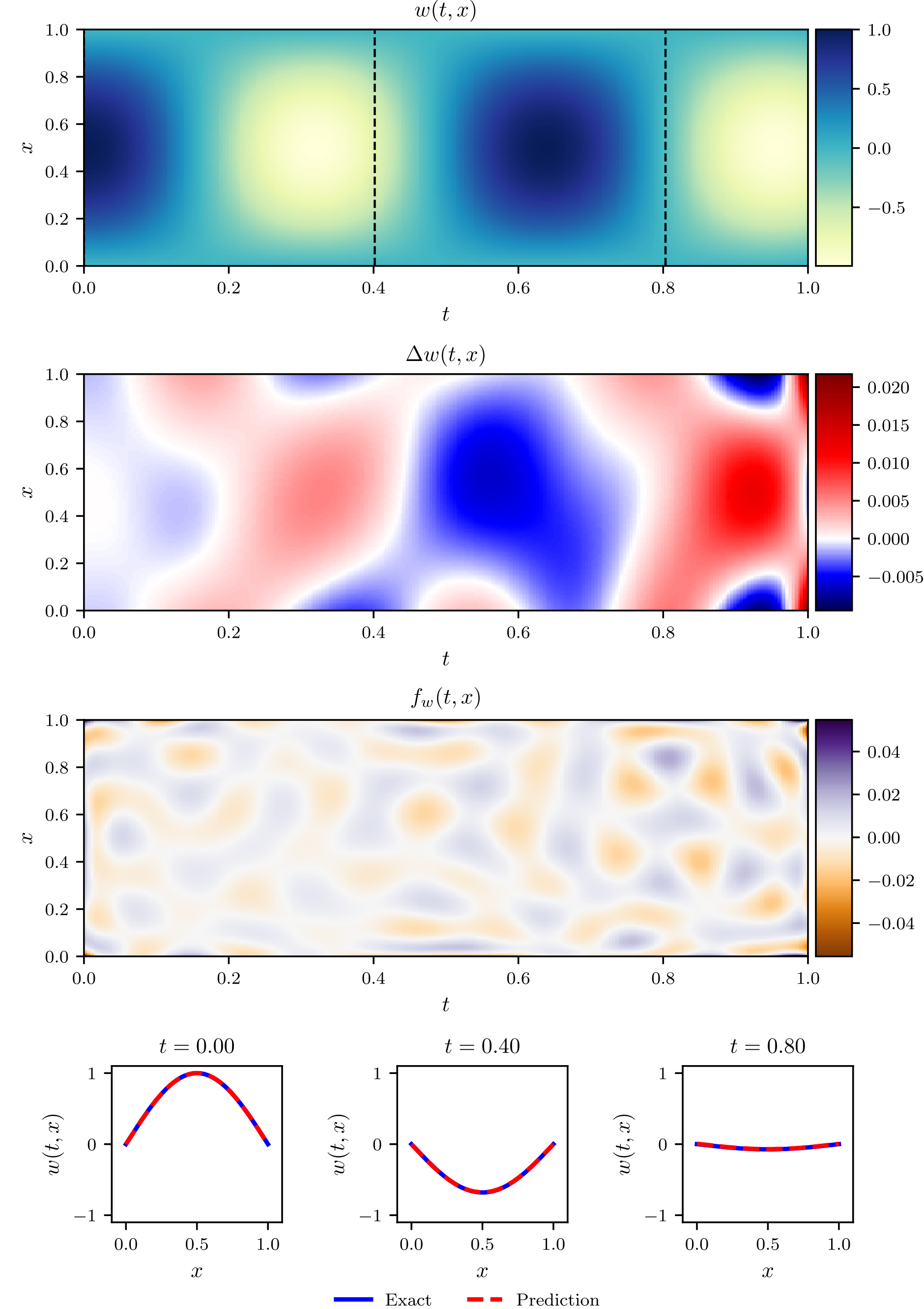
tf.reduce\_mean(tf.square(self.w\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.f\_w\_pred))

* Structure of MLP:
* Domain:
* Exact solution:
* Training:  
  The parameters of the MLP are trained with Adam for 10000 iterations at first. During each epoch, the collocation points are randomly sampled from the domain of definition to conduct stochastic gradient descent. For both the initial and boundary conditions there are 50 points sampled, and 2500 points are sampled for the internal region which should satisfy the governing equation.  
  After 10000 iterations, the parameters are then fine-tuned by L-BFGS-B until the loss converges. The data points are taken from a 150 x 150 grid.

### Result:



* Remark:  
  The function approximation tends to have larger error the further away from the initial boundary. This tendency aggravates as the frequency of the vibration increases.

## Experiment 1.1

This experiment is for examining PINN’s performance when the vibration frequency increases.

### Setup:

* Governing equation:
* Initial condition:
* Loss:

self.loss = tf.reduce\_mean(tf.square(self.w0\_tf - self.w0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_t0\_tf - self.w\_t0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_lb\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_lb\_pred)) + \

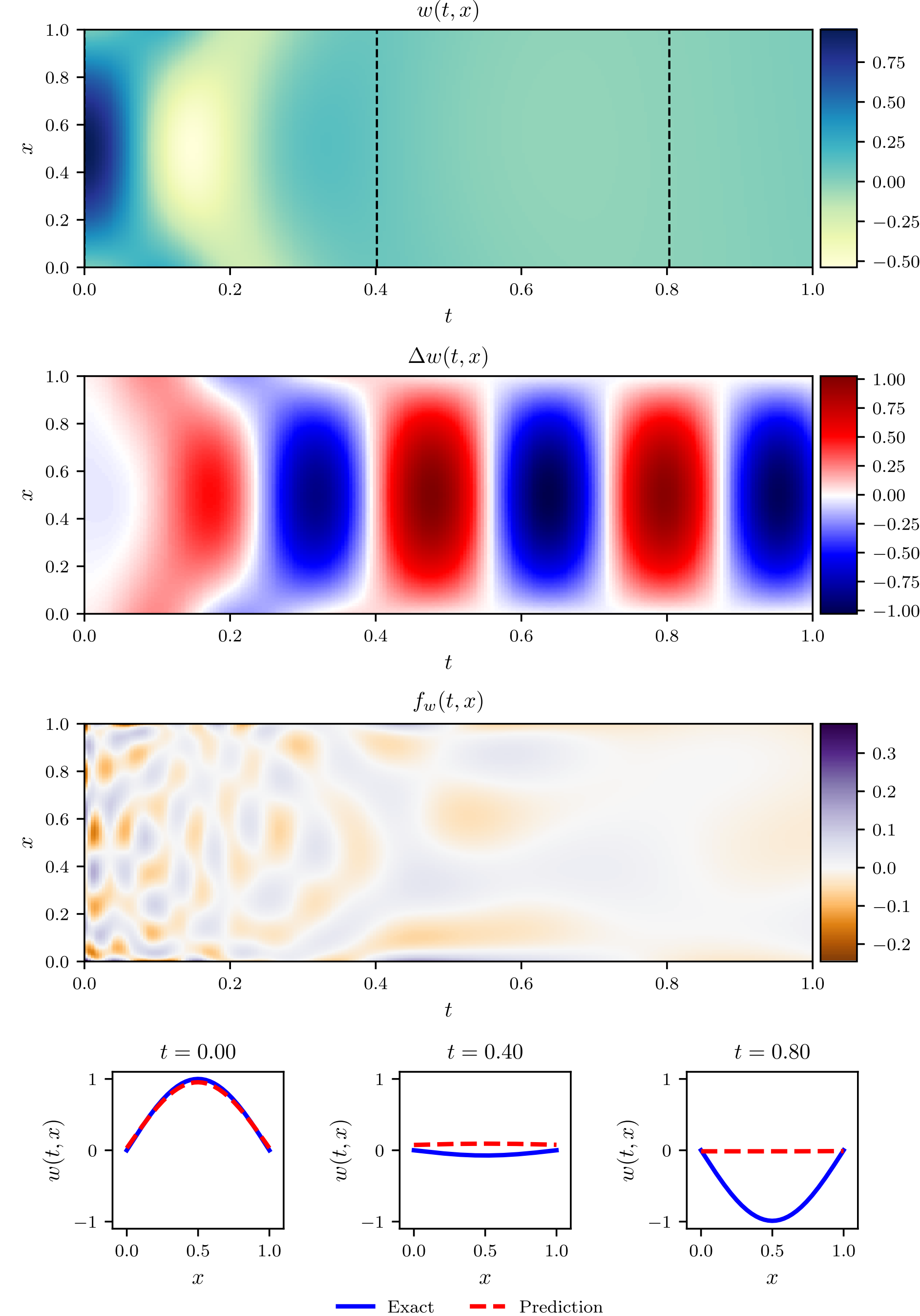
tf.reduce\_mean(tf.square(self.w\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.f\_w\_pred))

* Structure of MLP:
* Domain:
* Exact solution:
* Training:  
  The parameters of the MLP are trained with Adam for 10000 iterations at first. During each epoch, the collocation points are randomly sampled from the domain of definition to conduct stochastic gradient descent. For both the initial and boundary conditions there are 50 points sampled, and 2500 points are sampled for the internal region which should satisfy the governing equation.  
  After 10000 iterations, the parameters are then fine-tuned by L-BFGS-B until the loss converges. The data points are taken from a 150 x 150 grid.

### Result:



* Remark:  
  The collocation points locating at the rear part of the time axis become a hindrance to the convergence of the network. As constant value is a trivial solution to the governing equation, whereas the network cannot distinguish that from a meaningful solution, the rear part is inclined to gravitate towards a constant in order to lower down the overall loss.

## Experiment 1.1.1

As a follow-up to experiment 1.1, this experiment is for examining whether the unideal result last time can be attributed to the overlong time domain.

### Setup:

* Governing equation:
* Initial condition:
* Loss:

self.loss = tf.reduce\_mean(tf.square(self.w0\_tf - self.w0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_t0\_tf - self.w\_t0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_lb\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_lb\_pred)) + \

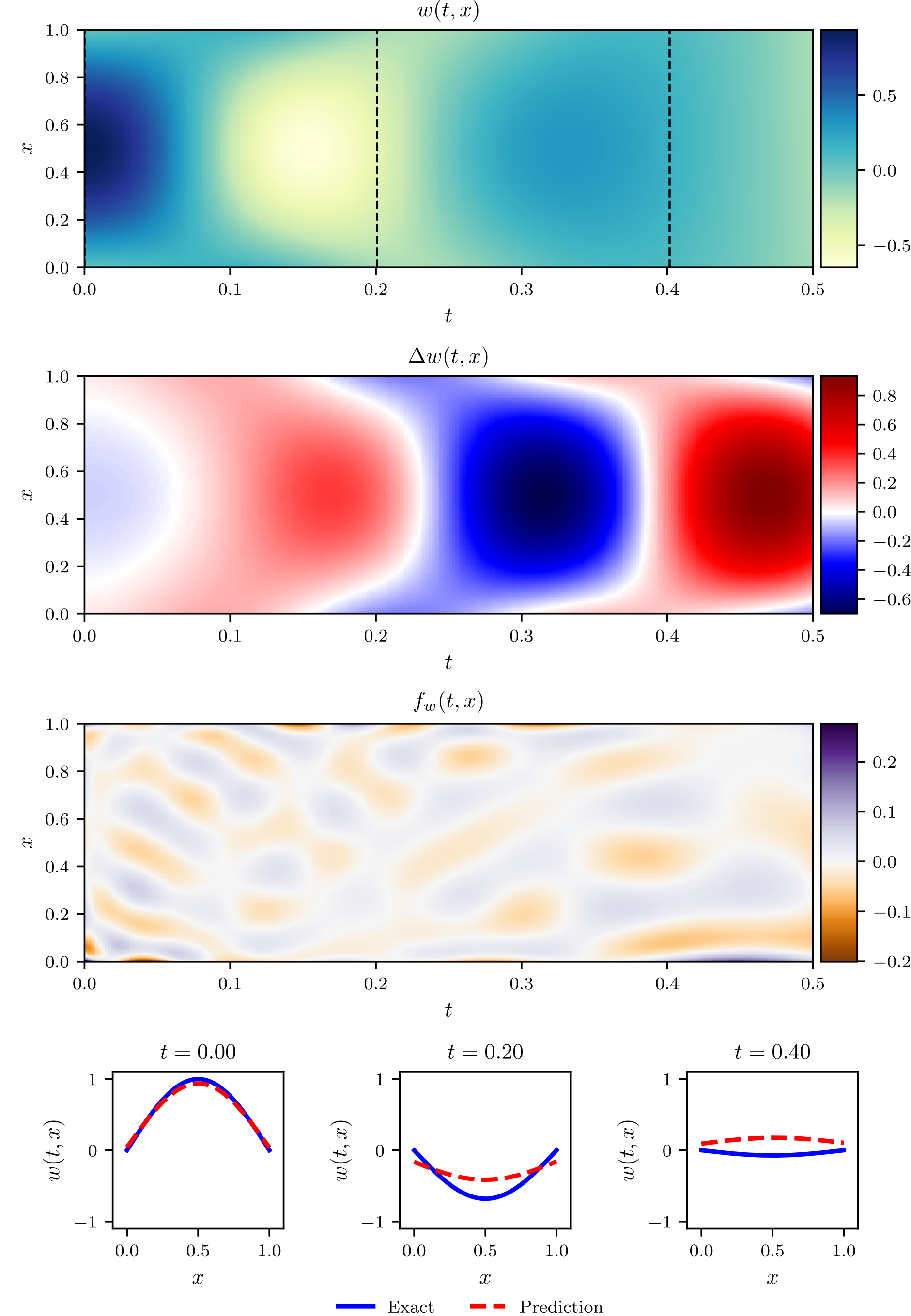
tf.reduce\_mean(tf.square(self.w\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.f\_w\_pred))

* Structure of MLP:
* Domain:
* Exact solution:
* Training:  
  The parameters of the MLP are trained with Adam for 10000 iterations at first. During each epoch, the collocation points are randomly sampled from the domain of definition to conduct stochastic gradient descent. For both the initial and boundary conditions there are 50 points sampled, and 2500 points are sampled for the internal region which should satisfy the governing equation.  
  After 10000 iterations, the parameters are then fine-tuned by L-BFGS-B until the loss converges. The data points are taken from a 150 x 150 grid.

### Result:



* Remark:  
  Though in expectation the result shouldn’t be too different from experiment 1, as this problem is simply a contracted version of the one in experiment 1, surprisingly the rear part of the time axis failed to converge to the exact solution.

## Experiment 1.1.1.1

As a follow-up to experiment 1.1.1, this experiment investigates if truncating the time domain even more can improve the approximation.

### Setup:

* Governing equation:
* Initial condition:
* Loss:

self.loss = tf.reduce\_mean(tf.square(self.w0\_tf - self.w0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_t0\_tf - self.w\_t0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_lb\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_lb\_pred)) + \

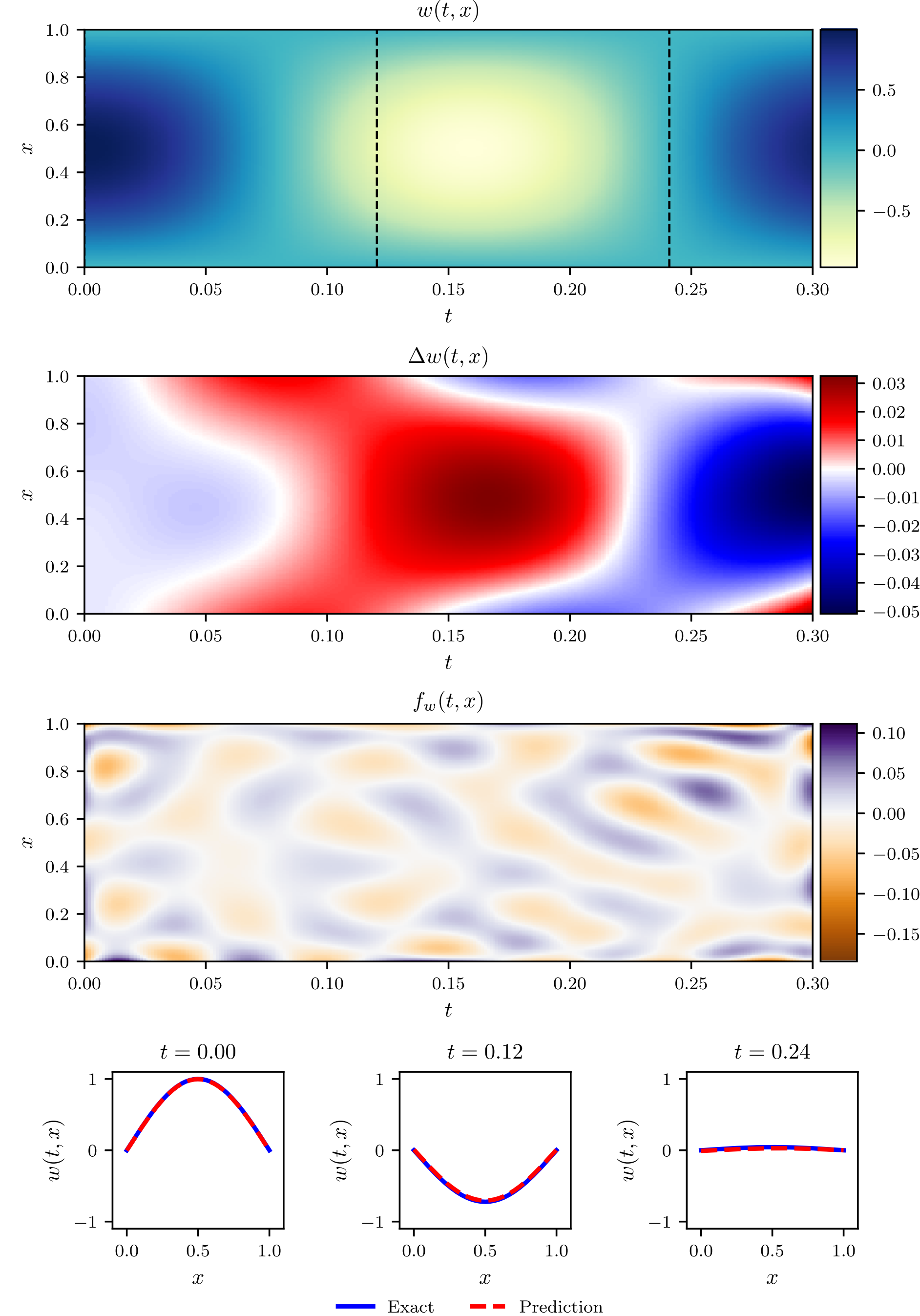
tf.reduce\_mean(tf.square(self.w\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.f\_w\_pred))

* Structure of MLP:
* Domain:
* Exact solution:
* Training:  
  The parameters of the MLP are trained with Adam for 10000 iterations at first. During each epoch, the collocation points are randomly sampled from the domain of definition to conduct stochastic gradient descent. For both the initial and boundary conditions there are 50 points sampled, and 2500 points are sampled for the internal region which should satisfy the governing equation.  
  After 10000 iterations, the parameters are then fine-tuned by L-BFGS-B until the loss converges. The data points are taken from a 150 x 150 grid.

### Result:



* Remark:  
  The result is way better than that in E1.1.1. Truncating the time domain can help improve the approximation quality, but it still doesn’t explain the question in E1.1.1.  
  A new experiment w.r.t the other variables like the complexity of the neural network should be carried out.

## Experiment 1.1.1.2

As a follow-up to experiment 1.1.1, this experiment investigates if increasing the complexity of the MLP can improve the approximation.

### Setup:

* Governing equation:
* Initial condition:
* Loss:

self.loss = tf.reduce\_mean(tf.square(self.w0\_tf - self.w0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_t0\_tf - self.w\_t0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_lb\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_lb\_pred)) + \

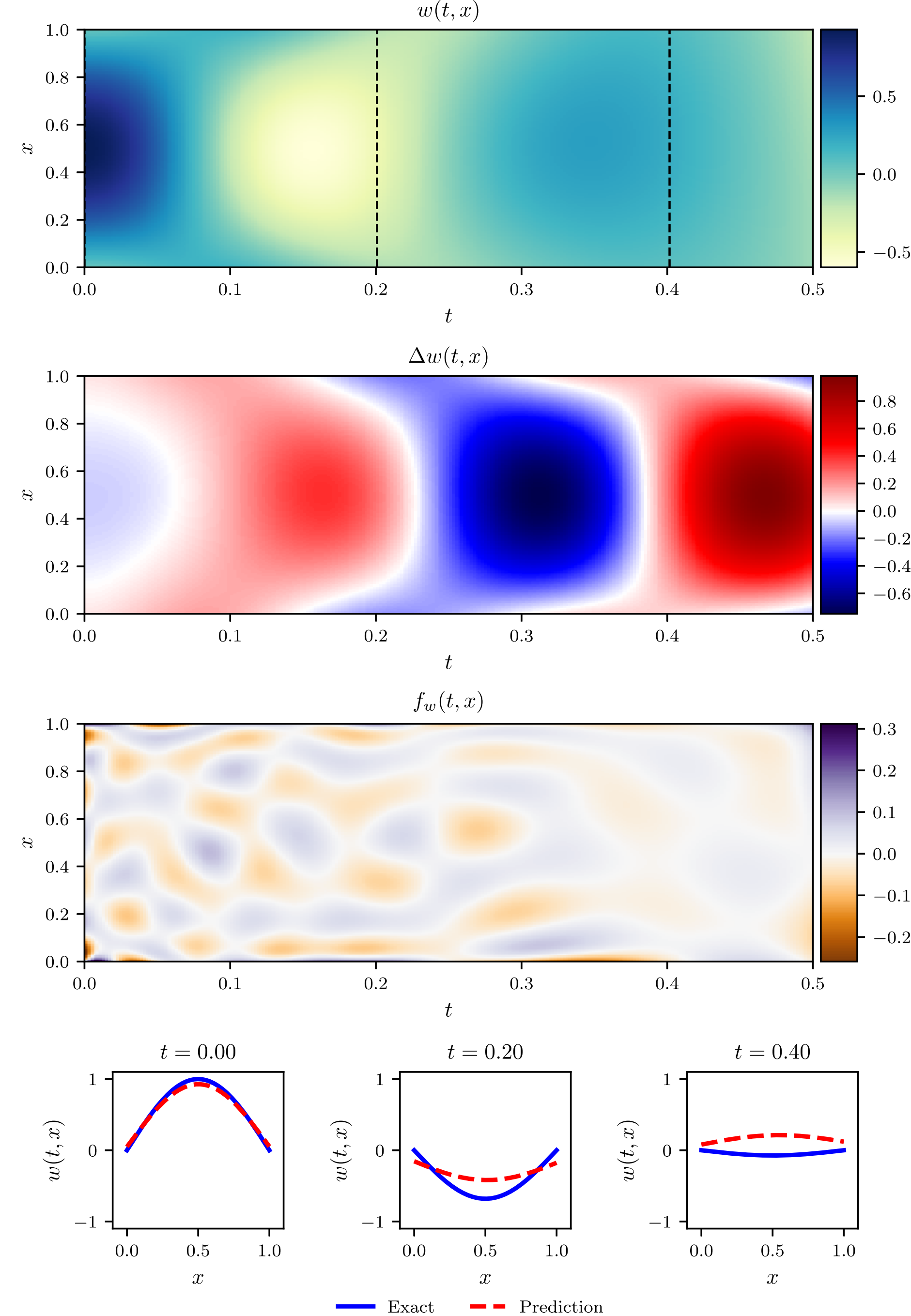
tf.reduce\_mean(tf.square(self.w\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_xx\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.f\_w\_pred))

* Structure of MLP:
* Domain:
* Exact solution:
* Training:  
  The parameters of the MLP are trained with Adam for 10000 iterations at first. During each epoch, the collocation points are randomly sampled from the domain of definition to conduct stochastic gradient descent. For both the initial and boundary conditions there are 50 points sampled, and 2500 points are sampled for the internal region which should satisfy the governing equation.  
  After 10000 iterations, the parameters are then fine-tuned by L-BFGS-B until the loss converges. The data points are taken from a 150 x 150 grid.

### Result:



* Remark:  
  The increase of complexity hasn’t improved the approximation.

# Vibration of String

Boundary conditions:

## Experiment 1

This experiment is for examining PINN’s capability of solving 2th order hyperbolic PDE.

### Setup:

* Governing equation:
* Initial conditions:
* Loss:

self.loss = tf.reduce\_mean(tf.square(self.w0\_tf - self.w0\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_t0\_tf - self.w\_t0\_pred)) + \

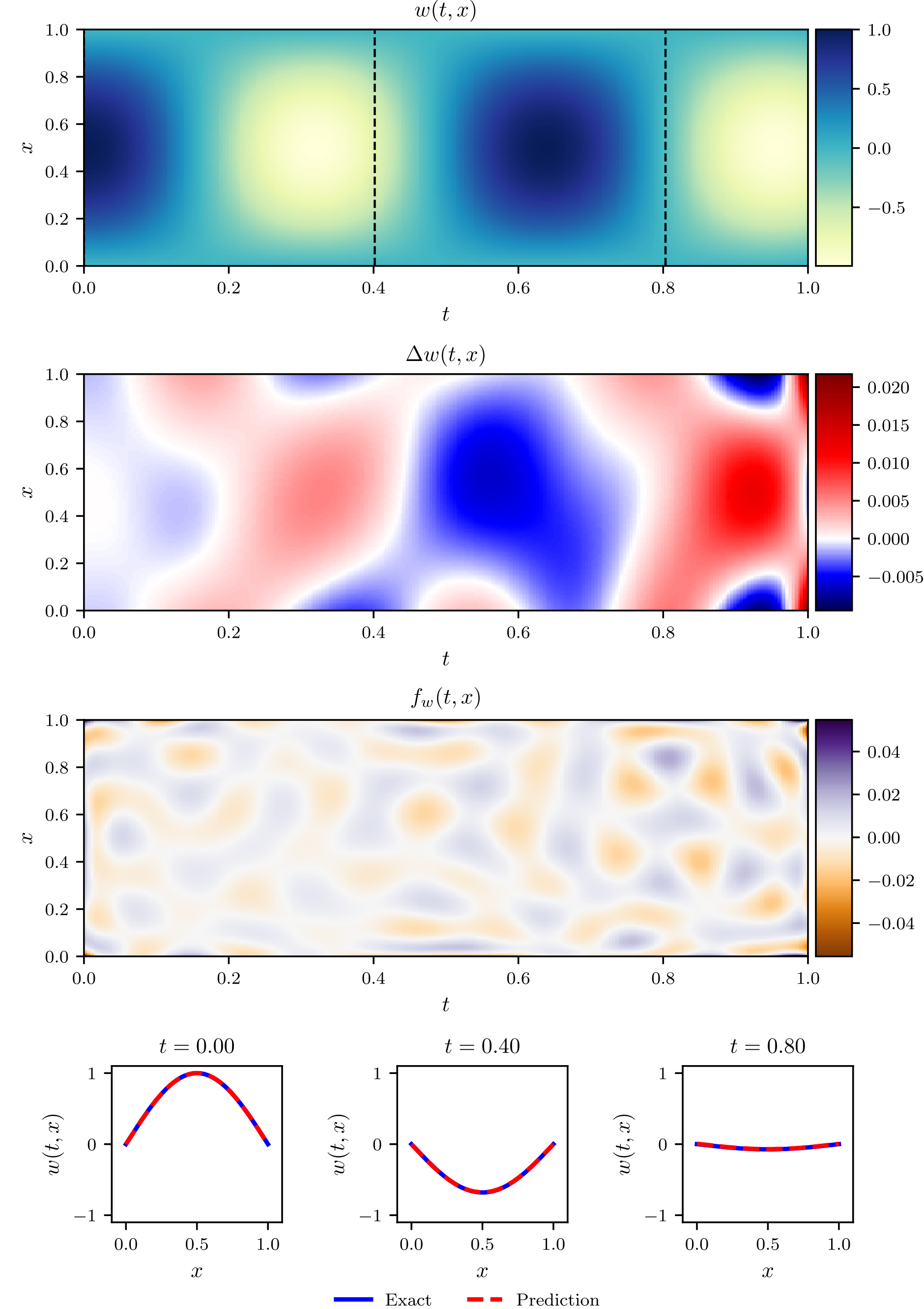
tf.reduce\_mean(tf.square(self.w\_lb\_pred)) + \

tf.reduce\_mean(tf.square(self.w\_ub\_pred)) + \

tf.reduce\_mean(tf.square(self.f\_w\_pred))

* Structure of MLP:
* Domain:
* Exact solution:
* Training:  
  The parameters of the MLP are trained with Adam for 10000 iterations at first. During each epoch, the collocation points are randomly sampled from the domain of definition to conduct stochastic gradient descent. For both the initial and boundary conditions there are 50 points sampled, and 2500 points are sampled for the internal region which should satisfy the governing equation.  
  After 10000 iterations, the parameters are then fine-tuned by L-BFGS-B until the loss converges. The data points are taken from a 150 x 150 grid.

### Result:



* Remark:  
  The function approximation tends to have larger error the further away from the initial boundary. This tendency aggravates as the frequency of the vibration increases.