In [1]:

```
import deepxde as dde
import numpy as np
import matplotlib.pyplot as plt
from deepxde.backend import tf
```

```
Using backend: tensorflow
```

I intend to solve the Helmholtz equation in 2D for an unbounded, homogeneous, source free domain. The PDE satisfies:

$$\nabla^2 u(x, y) + k^2 u(x, y) = 0$$

where $k^2 = \omega^2/c^2$ is the free space wave number, wherein $\omega = 2\pi f$ and c is the speed of light. Moreover, $u(x, y) = e^{-jkx}$, satisfies the given PDE, is a plane wave propagating along the $x-$direction in an unbounding medium.

In [2]:

```
# General parameters
c    = 1    # assuming speed of light to be 1 m/s. Infact c = 3x10^8 m/s
freq = 2
w    = 2*np.pi*freq
k    = w/c
wave_len = 2*np.pi / k

weights  = 1
epochs   = 5000
precision_train   = 10
precision_test    = 5
learning_rate     = 1e-3
num_dense_layers = 3
num_dense_nodes  = 350
activation        = "tanh"
```

In [3]:

```
def pde(X,u):
    ur,ui = u[:,0:1],u[:,1:2]

    ur_xx = dde.grad.hessian(u,X,component = 0, i=0,j=0)
    ur_yy = dde.grad.hessian(u,X,component = 0, i=1,j=1)

    ui_xx = dde.grad.hessian(u,X,component = 1, i=0,j=0)
    ui_yy = dde.grad.hessian(u,X,component = 1, i=1,j=1)

    return [ ur_xx + ur_yy + k**2 * ur,
             ui_xx + ui_yy + k**2 * ui
           ]
```

In [8]:

```
geom     = dde.geometry.Rectangle([0, 0], [2*wave_len, 2*wave_len]) # 2 x 2 wave ler

hx_train = wave_len / precision_train
nx_train = int(1 / hx_train)

hx_test  = wave_len / precision_test
nx_test  = int(1 / hx_test)

data     = dde.data.PDE(
            geom,
            pde,
            [],
            num_domain   = nx_train ** 2,
            num_boundary = 4 * nx_train,
            num_test     = nx_test ** 2,
)

net      = dde.nn.FNN(
            [2] + [num_dense_nodes] * num_dense_layers + [2], activation, "Glorot ur
)

model    = dde.Model(data, net)

loss_weights = [1, weights]
model.compile(
            "adam",
            lr=learning_rate,
            loss_weights=loss_weights,
            )

losshistory, train_state = model.train(iterations=epochs)
```

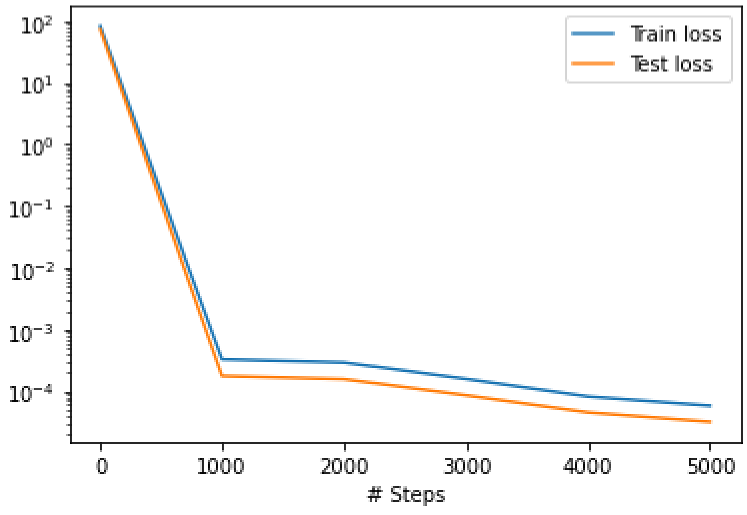
```
Compiling model...
'compile' took 0.000513 s


/Users/sandhua/opt/anaconda3/envs/DeepXDE/lib/python3.9/site-packages/
skopt/sampler/sobol.py:246: UserWarning: The balance properties of Sob
ol' points require n to be a power of 2. 0 points have been previously
generated, then: n=0+402=402.
  warnings.warn("The balance properties of Sobol' points require "
/Users/sandhua/opt/anaconda3/envs/DeepXDE/lib/python3.9/site-packages/
skopt/sampler/sobol.py:246: UserWarning: The balance properties of Sob
ol' points require n to be a power of 2. 0 points have been previously
generated, then: n=0+84=84.
  warnings.warn("The balance properties of Sobol' points require "
2022-08-18 15:37:58.904010: I tensorflow/core/platform/cpu_feature_gua
rd.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural
Network Library (oneDNN) to use the following CPU instructions in perf
ormance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the approp
riate compiler flags.

Training model...

Step      Train loss                Test loss                 Test metric
0         [5.63e+01, 2.72e+01]      [5.32e+01, 2.02e+01]      []
1000      [3.53e-05, 2.94e-04]      [1.83e-05, 1.58e-04]      []
2000      [3.54e-05, 2.57e-04]      [1.83e-05, 1.39e-04]      []
```

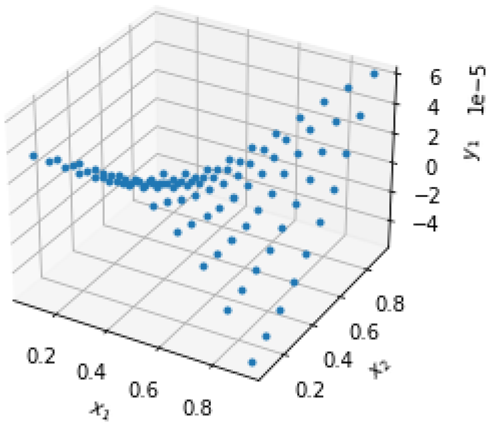
```
3000      [3.67e-05, 1.21e-04]    [1.92e-05, 6.62e-05]    []
4000      [3.49e-05, 4.68e-05]    [1.84e-05, 2.66e-05]    []
5000      [2.49e-05, 3.31e-05]    [1.31e-05, 1.87e-05]    []

Best model at step 5000:
  train loss: 5.80e-05
  test loss: 3.17e-05
  test metric: []

'train' took 350.919102 s
```

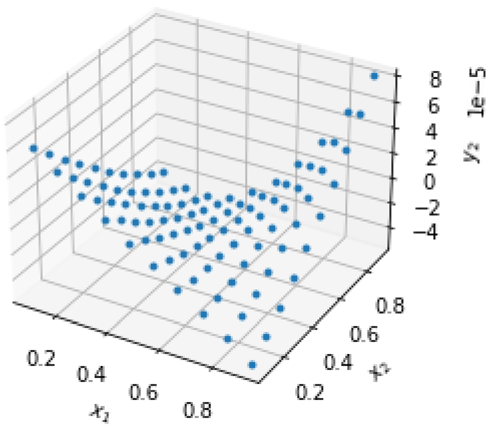
In [9]:

```
dde.saveplot(losshistory, train_state, issave=True, isplot=True)
```

```
Saving loss history to /Users/sandhua/My Drive/Academics@KFUPM/Researc
h_on_PINNs_Applied_to_EM/Codes/DeepXDE/examples/pinn_forward/loss.dat
...
Saving training data to /Users/sandhua/My Drive/Academics@KFUPM/Resear
ch_on_PINNs_Applied_to_EM/Codes/DeepXDE/examples/pinn_forward/train.da
t ...
Saving test data to /Users/sandhua/My Drive/Academics@KFUPM/Research_o
n_PINNs_Applied_to_EM/Codes/DeepXDE/examples/pinn_forward/test.dat ...
```







In [20]:

```
def func(X):
    """
    This could be the true solution. Although it can also be in the direction X[:,1:
    """
    u = np.exp(-1j*k*X[:,0:1])
    return u
```

In [ ]:

```
```