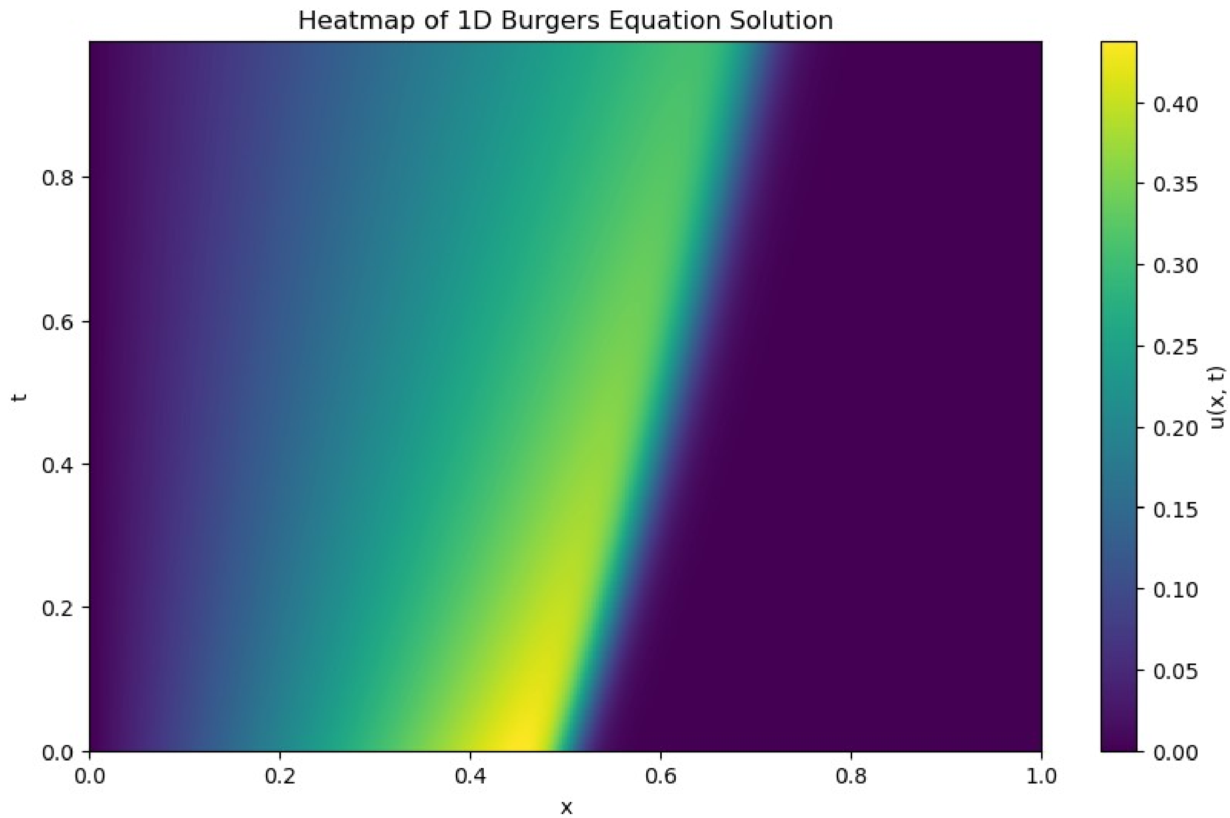
```python
import deepxde as dde
import numpy as np
from deepxde.backend import tf
import matplotlib.pyplot as plt
```

```
Using backend: tensorflow.compat.v1
Other supported backends: tensorflow, pytorch, jax, paddle.
paddle supports more examples now and is recommended.

WARNING:tensorflow:From d:\anaconda3\Lib\site-packages\deepxde\
backend\tensorflow_compat_v1\tensor.py:25: The name
tf.disable_v2_behavior is deprecated. Please use
tf.compat.v1.disable_v2_behavior instead.

WARNING:tensorflow:From d:\anaconda3\Lib\site-packages\tensorflow\
python\compat\v2_compat.py:98: disable_resource_variables (from
tensorflow.python.ops.resource_variables_toggle) is deprecated and
will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

Analytical Solution

```python
# Define the Reynolds number for the analytical solution
# Re = 1,50,100,300
Re_fixed = 300

# Define the spatial and temporal grid
N_t = 100  # Number of time points
N_x = 256  # Number of spatial points
t = np.linspace(0, 0.99, N_t)  # Time grid
x = np.linspace(0, 1, N_x)    # Spatial grid

# Define the analytical solution for the 1D Burgers equation
def analytical_solution(x, t, Re_fixed):
    to = np.exp(Re_fixed / 8)  # Parameter in the equation
    u = x / (t + 1) / (1 + np.sqrt((t + 1) / to) * np.exp(Re_fixed *
x**2 / (4 * (t + 1))))
    return u

# Compute the solution
usol = np.array([[analytical_solution(xi, ti, Re_fixed) for xi in x]
for ti in t])

# Explicitly enforce boundary conditions
usol[:, 0] = 0  # u(0, t) = 0
usol[:, -1] = 0  # u(1, t) = 0

# Verify the boundary conditions
assert np.allclose(usol[:, 0], 0), "Boundary condition u(0, t) = 0 not
```

```
satisfied"
assert np.allclose(usol[:, -1], 0), "Boundary condition u(1, t) = 0
not satisfied"

# Save the data to a .npz file
np.savez("dataset/Burgers.npz", t=t, x=x, usol=usol)

# Create a new figure window with a size of 10x6 inches
plt.figure(figsize=(10, 6))

# Plot the heatmap
# `usol` is the 2D array containing the solution, `extent` specifies
the range for x and t axes,
# `origin='lower'` ensures the heatmap starts from the bottom-left
corner,
# `aspect='auto'` adjusts the aspect ratio automatically, and
`cmap='viridis'` sets the colormap
plt.imshow(usol, extent=[x.min(), x.max(), t.min(), t.max()],
           origin='lower', aspect='auto', cmap='viridis')

# Add a colorbar to indicate the value of u(x, t) for each color
plt.colorbar(label="u(x, t)")

plt.title("Heatmap of 1D Burgers Equation Solution")
plt.xlabel("x")
plt.ylabel("t")

plt.show()
```



Part (a): Forward Problem

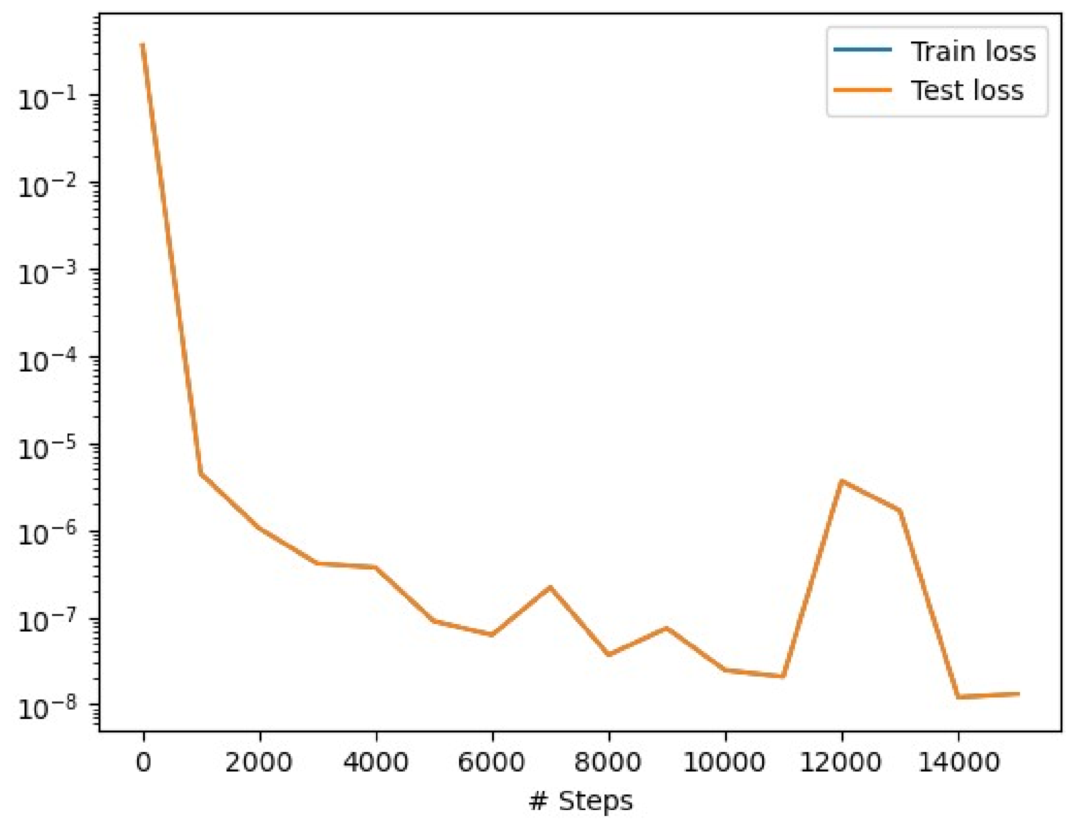
```
# Define fixed Reynolds number
# Re = 1,50,100,300
Re_fixed = 300

# Generate analytical test data
def gen_testdata():
    data = np.load("dataset/Burgers.npz")  # Ensure the file is
present in the correct location
    t, x, exact = data["t"], data["x"], data["usol"].T
    xx, tt = np.meshgrid(x, t)
    X = np.vstack((np.ravel(xx), np.ravel(tt))).T
    y = exact.flatten()[:, None]
    return X, y

# Define the PDE
def pde(x, y, Re):
    dy_x = dde.grad.jacobian(y, x, i=0, j=0)
    dy_t = dde.grad.jacobian(y, x, i=0, j=1)
    dy_xx = dde.grad.hessian(y, x, i=0, j=0)
    return dy_t + y * dy_x - 3 / Re * dy_xx # Re=Re_fixed/100

# Define the domain and conditions
geom = dde.geometry.Interval(0, 1)
```

```
timedomain = dde.geometry.TimeDomain(0, 0.99)
geomtime = dde.geometry.GeometryXTime(geom, timedomain)
bc = dde.DirichletBC(geomtime, lambda x: 0, lambda _, on_boundary:
on_boundary)
ic = dde.IC(
    geomtime, lambda x: x[:, 0:1] / (1 + np.sqrt(np.exp(Re_fixed / 8))
* np.exp(Re_fixed * x[:, 0:1]**2 / 4)),
    lambda _, on_initial: on_initial
)

# Solve the forward problem for fixed Re
print(f"Training for fixed Re = {Re_fixed}")

# Define dataset for fixed Re
data_fixed = dde.data.TimePDE(
    geomtime, lambda x, y: pde(x, y, Re_fixed), [bc, ic],
num_domain=2540, num_boundary=80, num_initial=160
)

# Define the neural network
net_fixed = dde.maps.FNN([2] + [20] * 3 + [1], "tanh", "Glorot
normal")
model_fixed = dde.Model(data_fixed, net_fixed)
model_fixed.compile("adam", lr=1e-3)

# Train the model
model_fixed.train(epochs=15000)
model_fixed.compile("L-BFGS")
losshistory_fixed, train_state_fixed = model_fixed.train()

# Save results
dde.saveplot(losshistory_fixed, train_state_fixed, issave=True,
isplot=True)

# Test the model
X_fixed, y_true_fixed = gen_testdata()
y_pred_fixed = model_fixed.predict(X_fixed)
f_fixed = model_fixed.predict(X_fixed, operator=lambda x, y: pde(x, y,
Re_fixed))

# Print errors for fixed Re
print(f"Fixed Re = {Re_fixed}, Mean residual:
{np.mean(np.absolute(f_fixed))}")
print(f"Fixed Re = {Re_fixed}, L2 relative error:
{dde.metrics.l2_relative_error(y_true_fixed, y_pred_fixed)}")
np.savetxt(f"test_fixed_Re_{Re_fixed}.dat", np.hstack((X_fixed,
y_true_fixed, y_pred_fixed)))
```

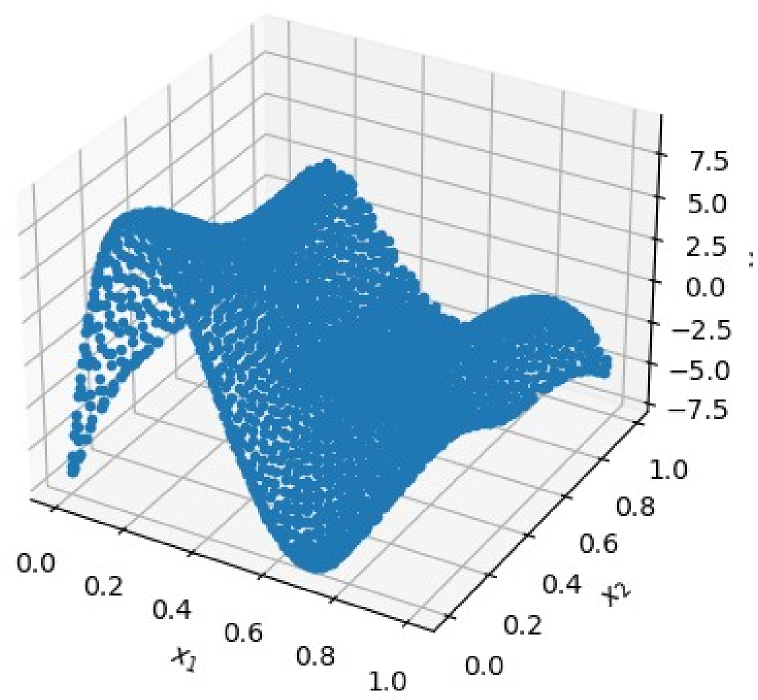
```
Training for fixed Re = 300
Compiling model...
```

```
Building feed-forward neural network...
'build' took 0.063288 s

WARNING:tensorflow:From d:\anaconda3\Lib\site-packages\deepxde\
model.py:168: The name tf.train.Saver is deprecated. Please use
tf.compat.v1.train.Saver instead.


C:\Users\jhyang\AppData\Local\Temp\ipykernel_18744\2581919726.py:27:
RuntimeWarning: overflow encountered in multiply
  geomtime, lambda x: x[:, 0:1] / (1 + np.sqrt(np.exp(Re_fixed / 8)) *
np.exp(Re_fixed * x[:, 0:1]**2 / 4)),

'compile' took 0.529124 s

Warning: epochs is deprecated and will be removed in a future version.
Use iterations instead.
Training model...

Step      Train loss                          Test loss
Test metric
0         [2.52e-01, 6.14e-02, 5.72e-02]    [2.52e-01, 6.14e-02,
5.72e-02]    []
1000      [4.15e-06, 2.04e-07, 1.27e-07]    [4.15e-06, 2.04e-07,
1.27e-07]    []
2000      [9.16e-07, 9.16e-08, 4.54e-08]    [9.16e-07, 9.16e-08,
4.54e-08]    []
3000      [3.73e-07, 2.54e-08, 1.66e-08]    [3.73e-07, 2.54e-08,
1.66e-08]    []
4000      [1.55e-07, 1.61e-07, 5.58e-08]    [1.55e-07, 1.61e-07,
5.58e-08]    []
5000      [8.21e-08, 3.78e-09, 4.51e-09]    [8.21e-08, 3.78e-09,
4.51e-09]    []
6000      [5.61e-08, 2.60e-09, 3.95e-09]    [5.61e-08, 2.60e-09,
3.95e-09]    []
7000      [5.19e-08, 1.33e-07, 3.66e-08]    [5.19e-08, 1.33e-07,
3.66e-08]    []
8000      [3.10e-08, 1.65e-09, 4.33e-09]    [3.10e-08, 1.65e-09,
4.33e-09]    []
9000      [3.38e-08, 1.74e-08, 2.38e-08]    [3.38e-08, 1.74e-08,
2.38e-08]    []
10000     [1.71e-08, 1.37e-09, 6.24e-09]    [1.71e-08, 1.37e-09,
6.24e-09]    []
11000     [1.36e-08, 1.21e-09, 5.90e-09]    [1.36e-08, 1.21e-09,
5.90e-09]    []
12000     [4.84e-07, 1.98e-06, 1.21e-06]    [4.84e-07, 1.98e-06,
1.21e-06]    []
13000     [1.19e-06, 2.51e-07, 2.50e-07]    [1.19e-06, 2.51e-07,
2.50e-07]    []
14000     [7.49e-09, 1.13e-09, 3.36e-09]    [7.49e-09, 1.13e-09,
```

```
3.36e-09]    []
15000     [6.39e-09, 2.41e-09, 4.35e-09]    [6.39e-09, 2.41e-09,
4.35e-09]    []

Best model at step 14000:
  train loss: 1.20e-08
  test loss: 1.20e-08
  test metric: []

'train' took 40.350875 s

Compiling model...
'compile' took 0.203661 s

Training model...

Step      Train loss                        Test loss
Test metric
15000     [6.39e-09, 2.41e-09, 4.35e-09]    [6.39e-09, 2.41e-09,
4.35e-09]    []
WARNING:tensorflow:From d:\anaconda3\Lib\site-packages\deepxde\
optimizers\tensorflow_compat_v1\scipy_optimizer.py:398: The name
tf.logging.info is deprecated. Please use tf.compat.v1.logging.info
instead.

INFO:tensorflow:Optimization terminated with:
  Message: CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
  Objective function value: 0.000000
  Number of iterations: 1
  Number of functions evaluations: 36
15017     [6.39e-09, 2.41e-09, 4.35e-09]    [6.39e-09, 2.41e-09,
4.35e-09]    []

Best model at step 14000:
  train loss: 1.20e-08
  test loss: 1.20e-08
  test metric: []

'train' took 0.380802 s

Saving loss history to c:\Users\jhyang\OneDrive\文档\GitHub_Projects\
ME_964\Final_Project\loss.dat ...
Saving training data to c:\Users\jhyang\OneDrive\文档\GitHub_Projects\
ME_964\Final_Project\train.dat ...
Saving test data to c:\Users\jhyang\OneDrive\文档\GitHub_Projects\
ME_964\Final_Project\test.dat ...
```



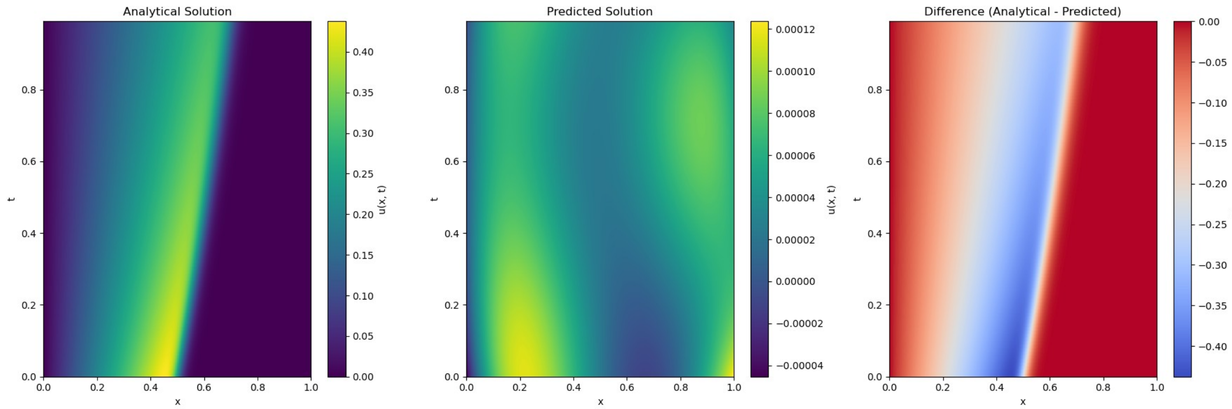
Train loss
Test loss
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
0
2000
4000
6000
8000
10000
12000
14000
Steps



7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0.0
0.2
0.4
0.6
0.8
1.0
x_1
0.0
0.2
0.4
0.6
0.8
1.0
x_2

```
Fixed Re = 300, Mean residual: 5.804481770610437e-05
Fixed Re = 300, L2 relative error: 0.9998068515099797
```

```
# Reshape y_true and y_pred back into the shape of the grid for
plotting
y_true_reshaped = y_true_fixed.reshape(len(t), len(x))
y_pred_reshaped = y_pred_fixed.reshape(len(t), len(x))

# Create a figure with subplots for comparison
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# Plot analytical solution heatmap
# im1 = axs[0].imshow(y_true_reshaped, extent=[x.min(), x.max(),
t.min(), t.max()],
#                     origin='lower', aspect='auto', cmap='viridis')
im1 = axs[0].imshow(usol, extent=[x.min(), x.max(), t.min(), t.max()],

           origin='lower', aspect='auto', cmap='viridis')
axs[0].set_title("Analytical Solution")
axs[0].set_xlabel("x")
axs[0].set_ylabel("t")
fig.colorbar(im1, ax=axs[0], label="u(x, t)")

# Plot predicted solution heatmap
im2 = axs[1].imshow(y_pred_reshaped, extent=[x.min(), x.max(),
t.min(), t.max()],
                    origin='lower', aspect='auto', cmap='viridis')
axs[1].set_title("Predicted Solution")
axs[1].set_xlabel("x")
axs[1].set_ylabel("t")
fig.colorbar(im2, ax=axs[1], label="u(x, t)")

# Plot difference heatmap
diff = y_pred_reshaped - usol
im3 = axs[2].imshow(diff, extent=[x.min(), x.max(), t.min(), t.max()],
                    origin='lower', aspect='auto', cmap='coolwarm')
axs[2].set_title("Difference (Analytical - Predicted)")
axs[2].set_xlabel("x")
axs[2].set_ylabel("t")
fig.colorbar(im3, ax=axs[2], label="Difference")

# Adjust layout and show the plot
plt.tight_layout()
plt.show()
```



Part (b): Combined Inverse-Forward Problem

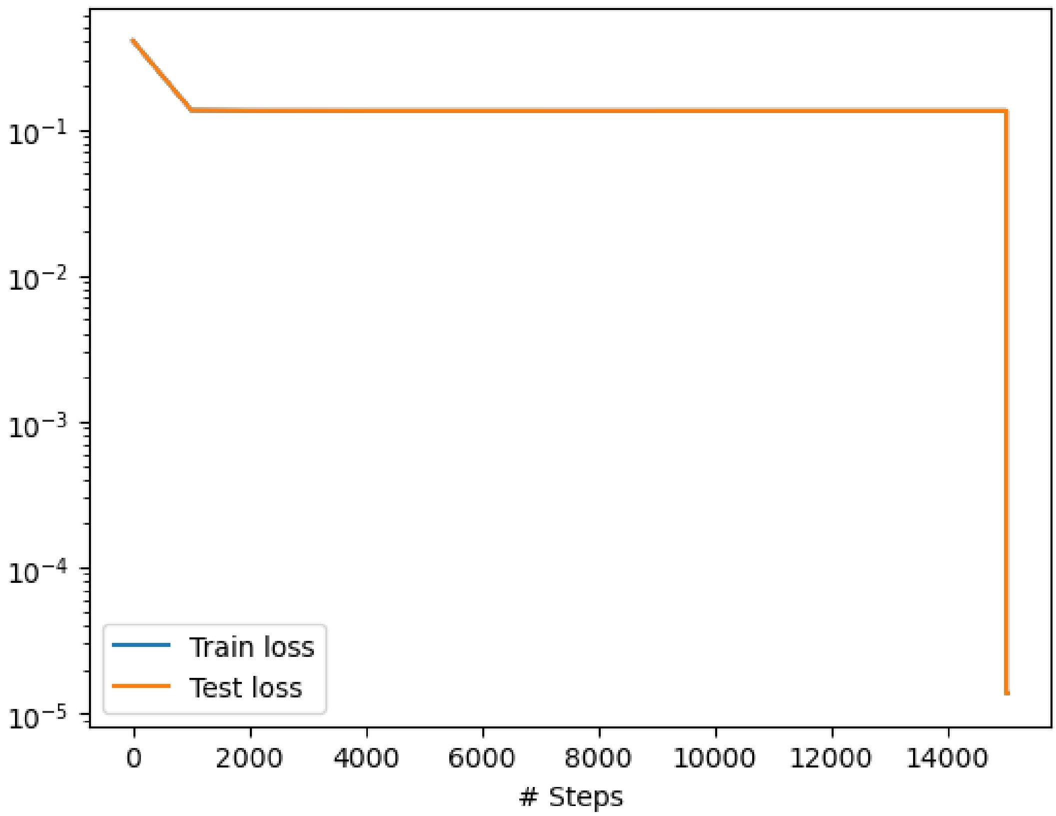
```
# Define Reynolds number as a trainable variable
Re_trainable = tf.Variable(300.0, trainable=True, dtype=tf.float32)

# Define the PDE
def pde_trainable(x, y):
    dy_x = dde.grad.jacobian(y, x, i=0, j=0)
    dy_t = dde.grad.jacobian(y, x, i=0, j=1)
    dy_xx = dde.grad.hessian(y, x, i=0, j=0)
    return dy_t + y * dy_x - 3 / Re_trainable * dy_xx

# Define the initial condition
to = tf.exp(Re_trainable / 8)
ic = dde.IC(
    geomtime,
    lambda x: x[:, 0:1] / (1 + tf.sqrt(tf.minimum(10.0, to)) *
tf.exp(tf.minimum(10.0, Re_trainable * x[:, 0:1]**2 / 4))),
    lambda _, on_initial: on_initial,
)

# Dirichlet boundary conditions
bc = dde.DirichletBC(geomtime, lambda x: 0, lambda _, on_boundary:
on_boundary)

# Define the dataset
data_trainable = dde.data.TimePDE(
    geomtime, pde_trainable, [bc, ic], num_domain=3000,
num_boundary=100, num_initial=200
)

# Define the neural network
net_trainable = dde.maps.FNN([2] + [40] * 5 + [1], "tanh", "Glorot
normal")

# Define custom loss function with L2 regularization
def custom_loss(y_true, y_pred):
    # Compute Mean Squared Error (MSE)
```

```
    mse = tf.reduce_mean(tf.square(y_true - y_pred))
    # Collect trainable variables
    trainable_vars = tf.compat.v1.trainable_variables()
    # Apply L2 regularization
    l2_reg = 1e-6 * tf.reduce_sum([tf.nn.l2_loss(v) for v in
trainable_vars])
    return mse + l2_reg

# Compile and train the model
model_trainable = dde.Model(data_trainable, net_trainable)
model_trainable.compile("adam", lr=1e-5, loss=custom_loss)
model_trainable.train(epochs=15000)
model_trainable.compile("L-BFGS")
losshistory_trainable, train_state_trainable = model_trainable.train()

# Save results
dde.saveplot(losshistory_trainable, train_state_trainable,
issave=True, isplot=True)

# Test and evaluate the model
X_trainable, y_true_trainable = gen_testdata()
y_pred_trainable = model_trainable.predict(X_trainable)
f_trainable = model_trainable.predict(X_trainable,
operator=pde_trainable)

# Print errors and learned Reynolds number
print("Mean residual for trainable Re:",
np.mean(np.absolute(f_trainable)))
print("L2 relative error for trainable Re:",
dde.metrics.l2_relative_error(y_true_trainable, y_pred_trainable))

with tf.compat.v1.Session() as sess:
    sess.run(tf.compat.v1.global_variables_initializer())
    learned_Re = sess.run(Re_trainable)
print("Learned Reynolds number:", learned_Re)

# Save test results
np.savetxt(f"test_trainable_Re_{learned_Re:.1f}.dat",
np.hstack((X_trainable, y_true_trainable, y_pred_trainable)))
```

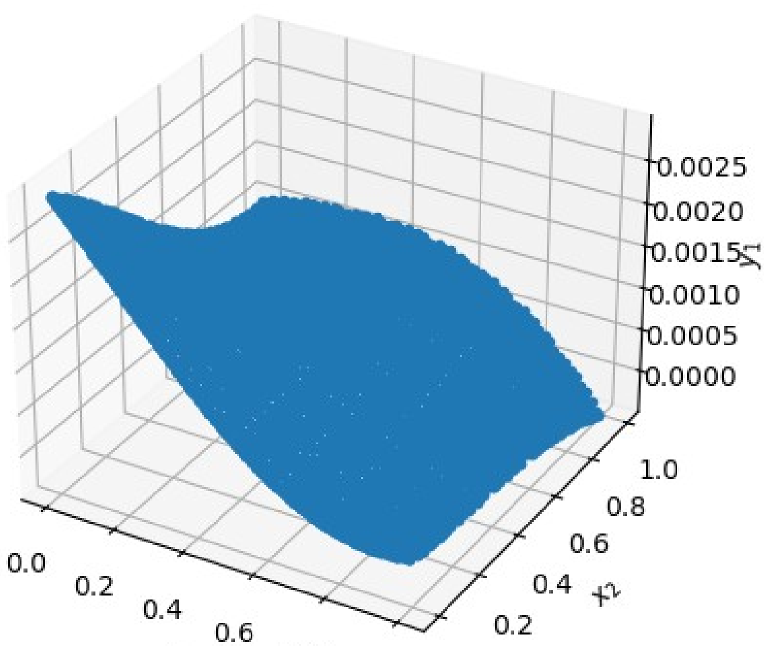
```
Compiling model...
Building feed-forward neural network...
'build' took 0.088369 s

WARNING:tensorflow:From C:\Users\jhyang\AppData\Local\Temp\
ipykernel_18744\2858174054.py:35: The name tf.trainable_variables is
deprecated. Please use tf.compat.v1.trainable_variables instead.

'compile' took 1.155185 s
```

```
Warning: epochs is deprecated and will be removed in a future version.
Use iterations instead.
Training model...

Step      Train loss                        Test loss
Test metric
0         [5.48e-02, 2.08e-01, 1.45e-01]    [5.48e-02, 2.08e-01,
1.45e-01]    []
1000      [4.53e-02, 4.59e-02, 4.52e-02]    [4.53e-02, 4.59e-02,
4.52e-02]    []
2000      [4.52e-02, 4.51e-02, 4.51e-02]    [4.52e-02, 4.51e-02,
4.51e-02]    []
3000      [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
4000      [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
5000      [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
6000      [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
7000      [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
8000      [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
9000      [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
10000     [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
11000     [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
12000     [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
13000     [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
14000     [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []
15000     [4.51e-02, 4.51e-02, 4.51e-02]    [4.51e-02, 4.51e-02,
4.51e-02]    []

Best model at step 15000:
  train loss: 1.35e-01
  test loss: 1.35e-01
  test metric: []

'train' took 117.012841 s

Compiling model...
'compile' took 0.347285 s

Training model...
```

```
Step      Train loss                       Test loss
Test metric
15000     [8.40e-07, 1.83e-06, 1.10e-05]    [8.40e-07, 1.83e-06,
1.10e-05]    []
INFO:tensorflow:Optimization terminated with:
  Message: CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
  Objective function value: 0.000014
  Number of iterations: 1
  Number of functions evaluations: 28
15019     [8.40e-07, 1.83e-06, 1.10e-05]    [8.40e-07, 1.83e-06,
1.10e-05]    []

Best model at step 15000:
  train loss: 1.37e-05
  test loss: 1.37e-05
  test metric: []

'train' took 0.813819 s

Saving loss history to c:\Users\jhyang\OneDrive\文档\GitHub_Projects\
ME_964\Final_Project\loss.dat ...
Saving training data to c:\Users\jhyang\OneDrive\文档\GitHub_Projects\
ME_964\Final_Project\train.dat ...
Saving test data to c:\Users\jhyang\OneDrive\文档\GitHub_Projects\
ME_964\Final_Project\test.dat ...
```



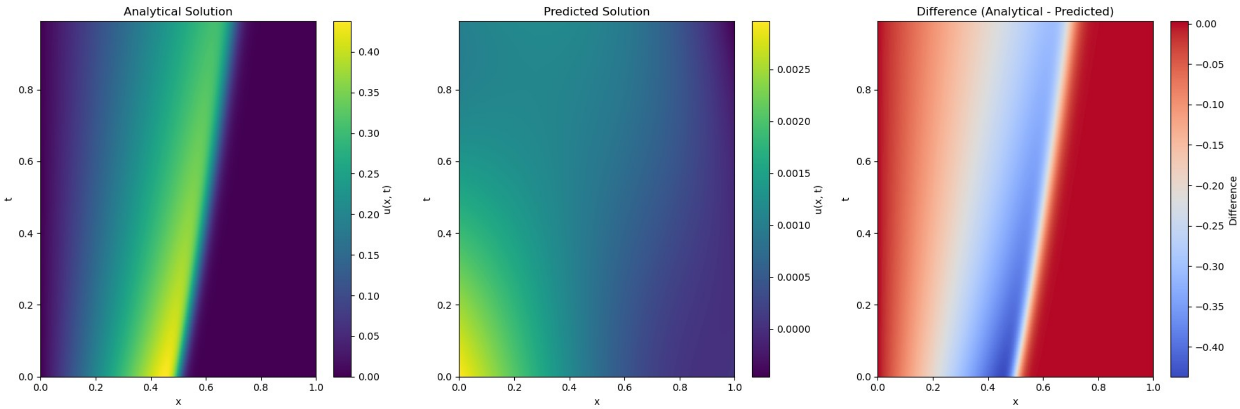
Train loss
Test loss
Steps



x_1
x_2
y_1

```
Mean residual for trainable Re: 0.0006394048
L2 relative error for trainable Re: 0.9965252271207028
Learned Reynolds number: 300.0
```

```
# Reshape y_true and y_pred back into the shape of the grid for
plotting
y_true_reshaped = y_true_trainable.reshape(len(t), len(x))
y_pred_reshaped = y_pred_trainable.reshape(len(t), len(x))

# Create a figure with subplots for comparison
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# Plot analytical solution heatmap
# im1 = axs[0].imshow(y_true_reshaped, extent=[x.min(), x.max(),
t.min(), t.max()],
#                     origin='lower', aspect='auto', cmap='viridis')
im1 = axs[0].imshow(usol, extent=[x.min(), x.max(), t.min(), t.max()],

           origin='lower', aspect='auto', cmap='viridis')
axs[0].set_title("Analytical Solution")
axs[0].set_xlabel("x")
axs[0].set_ylabel("t")
fig.colorbar(im1, ax=axs[0], label="u(x, t)")

# Plot predicted solution heatmap
im2 = axs[1].imshow(y_pred_reshaped, extent=[x.min(), x.max(),
t.min(), t.max()],
                    origin='lower', aspect='auto', cmap='viridis')
axs[1].set_title("Predicted Solution")
axs[1].set_xlabel("x")
axs[1].set_ylabel("t")
fig.colorbar(im2, ax=axs[1], label="u(x, t)")

# Plot difference heatmap
diff = y_pred_reshaped - usol
im3 = axs[2].imshow(diff, extent=[x.min(), x.max(), t.min(), t.max()],
                    origin='lower', aspect='auto', cmap='coolwarm')
axs[2].set_title("Difference (Analytical - Predicted)")
axs[2].set_xlabel("x")
axs[2].set_ylabel("t")
fig.colorbar(im3, ax=axs[2], label="Difference")

# Adjust layout and show the plot
plt.tight_layout()
plt.show()
```


Analytical Solution
Predicted Solution
Difference (Analytical - Predicted)
u(x, t)
Difference
x
t