I. APPENDIXES

Here we have included the implementation of PINN through Tensorflow library.

A. Model Initialization

```
import tensorflow as tf

# Initialize model
def init_model(num_hidden_layers=6, num_neurons_per_layer=60):
model = tf.keras.Sequential()
model.add(tf.keras.Input(shape=(2,)))

for _ in range(num_hidden_layers):
model.add(tf.keras.layers.Dense(num_neurons_per_layer, activation='tanh',
    kernel_initializer='glorot_uniform'))
model.add(tf.keras.layers.Dropout(0.1))  # 10% Dropout

model.add(tf.keras.layers.Dense(1))  # Output layer
return model
```

B. Residual Computation

```
def get_r(model, X_r):
with tf.GradientTape(persistent=True, watch_accessed_variables=False) as tape:
t, x = X_r[:, 0:1], X_r[:, 1:2]
tape.watch([t, x])

u = model(tf.concat([t, x], axis=1))
u_t = tape.gradient(u, t)
u_x = tape.gradient(u, x)

u_xx = tape.gradient(u_x, x) if u_x is not None else tf.zeros_like(u)
u_tx = tape.gradient(u_t, x) if u_t is not None else tf.zeros_like(u)
u_tt = tape.gradient(u_t, t) if u_t is not None else tf.zeros_like(u)
u_xxx = tape.gradient(u_xx, x) if u_xx is not None else tf.zeros_like(u)
u_xxxx = tape.gradient(u_xxx, x) if u_xxx is not None else tf.zeros_like(u)

del tape
return fun_r(x, t, u, u_tt, u_xx, u_tx, u_xxxx)
```

C. Loss Computation

```
def compute_loss(model, X_r, X_0, u_0, u_b):
r = get_r(model, X_r)
phi_r = tf.reduce_mean(tf.square(r))
u_0p = model(X_0)
u_bp = model(X_b)
loss_ic = tf.reduce_mean(tf.square(u_0 - u_0p))
loss_bc = tf.reduce_mean(tf.square(u_b - u_bp))
return phi_r + 10 * loss_ic + 10 * loss_bc
```

D. Gradient Computation

```
def get_grad(model, X_r, X_0, u_0, u_b):
with tf.GradientTape() as tape:
loss = compute_loss(model, X_r, X_0, u_0, u_b)
grad_theta = tape.gradient(loss, model.trainable_variables)
return loss, grad_theta
```

E. Model and Optimizer Initialization

```
# Initialize model
model = init_model()

# Optimizer with learning rate decay
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
initial_learning_rate=0.001, decay_steps=1000, decay_rate=0.95, staircase=True)
optim = tf.keras.optimizers.Adam(learning_rate=lr_schedule)
```