# Coupled PINN Model for Heat Transfer in Material Extrusion

This project implements two coupled Physics-Informed Neural Networks (PINNs) to solve the heat transfer problem in a material extrusion (MatEx) additive manufacturing process involving an amorphous polymer. The first model predicts the temperature field in the solid region, and the second model computes the melt region's temperature field using the interface predicted by the first.

Copyright: Prof. Cheng Luo, University of Texas at Arlington, [chengluo@uta.edu](mailto:chengluo@uta.edu)

## Files Included

- `heat\_transfer\_PINN.py`: Main training script (includes data generation, training, evaluation, and plotting)

- `README.md`: Project overview and usage instructions

##Recommendation

Copy the code from heat\_transfer\_PINN.py and run it on Google Colab using an A100 GPU. The implementation takes less than 10 minutes and automatically satisfies the requirements listed at the end.

## Model Summary

### Model 1: Solid Region PINN

- Solves a 2D convection-diffusion PDE using auto-differentiation

- Boundary conditions: inlet temperature, symmetry at axis, constant wall temperature

- Predicts the solid/melt interface as the contour where temperature equals `T\_th`

### Model 2: Melt Region PINN

- Solves a non-linear PDE that includes viscous heating and temperature-dependent viscosity

- Incorporates the predicted interface as a boundary condition

## Saved models after implementation

-Model 1 is saved  as 'model1.keras' in the current working directory.

-Model 2 is saved  as 'model2.keras' in the current working directory.

## To upload and reuse the two models, follow the simple example below:

-------------------------

import numpy as np

import tensorflow as tf

# Parameters for dimensional conversion

T0 = 25      # initial/reference temperature for Model 1

Tw = 230    # wall temperature

Tth = 164    # threshold temperature (reference) for Model 2

# Load the models

model1 = tf.keras.models.load\_model('model1.keras')

model2 = tf.keras.models.load\_model('model2.keras')

# Points to predict (R, Z)

points1 = np.array([[0.0, 0.1], [0.0, 0.2]], dtype=np.float32)

points2 = np.array([[0.0, 0.4], [0.0, 0.6]], dtype=np.float32)

# Predict dimensionless temperatures

preds\_model1\_dimless = model1.predict(points1)

preds\_model2\_dimless = model2.predict(points2)

# Convert dimensionless predictions to dimensional temperatures for Model 1

def to\_dimensional\_model1(T\_dimless):

    return T\_dimless \* (T0 - Tw) + Tw

# Convert dimensionless predictions to dimensional temperatures for Model 2

def to\_dimensional\_model2(T\_dimless):

    return T\_dimless \* (Tth - Tw) + Tw

preds\_model1\_dim = to\_dimensional\_model1(preds\_model1\_dimless)

preds\_model2\_dim = to\_dimensional\_model2(preds\_model2\_dimless)

# Print dimensional temperatures

print("\nModel 1 Dimensional Temperature Predictions (°C):")

for i in range(len(points1)):

    R, Z = points1[i]

    print(f"At (R={R}, Z={Z:.2f}): {preds\_model1\_dim[i, 0]:.2f} °C")

print("\nModel 2 Dimensional Temperature Predictions (°C):")

for i in range(len(points2)):

    R, Z = points2[i]

    print(f"At (R={R}, Z={Z:.2f}): {preds\_model2\_dim[i, 0]:.2f} °C")

------------------------------

## Requirements

- Python 3.8+

- TensorFlow 2.x

- NumPy

- SciPy

- Matplotlib

Install with:

```bash

pip install tensorflow numpy scipy matplotlib