CP302

CAPSTONE PROJECT – 1

End – Semester Report

Under the guidance of

Dr. Sachin Kumar
Associate Professor
Department of Mechanical Engineering
IIT Ropar

Submitted by

Group No. 03

Mohit Sharma (*2022MEB1326*)

Priyanshu Gupta (*2022MEB1330*)

Acknowledgement

We extend our heartfelt gratitude to Dr. Sachin Kumar, our project guide, for his unwavering support and invaluable guidance, which have been instrumental in the success of our development project. We are grateful for his insightful assistance and directions in helping us understand the project better.

Furthermore, we sincerely appreciate our parents for their constant encouragement, enabling us to explore and learn beyond our horizons. We also acknowledge the contributions of numerous researchers and scholars whose work has inspired and informed our project, significantly shaping our ideas and approach.

A special thanks to our fellow mechanical batchmates for their support, collaboration, and assistance in various aspects of our project. Lastly, we express our gratitude to everyone who has contributed to our project's growth and progress.

With the grace of the Almighty, we have thoroughly enjoyed working on this endeavour as a team and look forward to continued learning and development in the future.

Contents

1 Title of Project

Machine Learning Based Fracture Characterisation Using Phase Field Model

2 Introduction

When materials like concrete, ceramics, or rocks start to break, they don't fail suddenly—they gradually soften before completely fracturing. This softening behaviour is a key characteristic of quasi-brittle materials, and understanding it is crucial for predicting material failure in structures. Traditionally, engineers and researchers rely on finite element methods (FEM) and phase-field models to study this behaviour. While these methods are accurate, they can be computationally expensive and require a lot of fine-tuning, especially for complex fractures.

The phase-field model is widely used because it doesn't require explicitly tracking cracks. Instead, it represents fractures using a continuous damage parameter, making it easier to simulate material degradation. However, solving these equations with standard numerical methods can be slow and resource-intensive, especially for large-scale problems.

This is where machine learning (ML) comes in. By integrating physics-informed ML techniques, we can make fracture simulations faster and more efficient while still respecting the fundamental governing equations of fracture mechanics. Our approach focuses on:

a) Predicting how displacement and strain evolve as the material softens.

b) Using a physics-based loss function, where the model learns to minimize deviations from expected behaviour.

c) Capturing the softening process and transition from an intact state to complete fracture.

By combining data-driven learning with physics-based modelling, we aim to develop a more efficient and scalable approach for predicting material degradation. If successful, this method could have real-world applications in areas like civil engineering, aerospace, and materials science, helping to design safer and more reliable structures.

3 Aim and Objectives

The aim of this project is to develop a machine learning (ML) based approach for predicting the softening behaviour of quasi-brittle materials using the phase-field model. The study focuses on characterizing material degradation under mechanical loading by leveraging physics-informed machine learning techniques.

The project involves:

a) Developing a computational framework that predicts displacement and strain evolution in a structure under applied loading.

b) Formulating a physics-informed loss function, where deviations from the governing equations guide model optimization.

c) Extending the approach to capture fracture propagation and material softening behaviour.

4 Background and Literature Review

4.1 Historical Context and Motivation for Fracture Modelling

Fracture mechanics has evolved over centuries, driven by the need to understand, predict, and prevent material failure. The significance of fracture behaviour can be traced back to ancient times, when early humans used controlled fracturing to shape tools from stone. However, the scientific study of fracture began much later, with foundational contributions from figures such as Leonardo da Vinci, Galileo Galilei, and Griffith, leading to modern fracture mechanics.

4.1.1 Early Observations and Classical Theories of Fracture

The first recorded scientific understanding of fracture can be attributed to Leonardo da Vinci (1452–1519), who observed that the strength of materials depends on their size and shape. Galileo Galilei (1638) further expanded on this concept by introducing scaling laws for materials under tension and bending, laying the groundwork for later theories.

During the 19th century, the increasing use of iron and steel in construction raised concerns about fracture-related failures. David Kirkaldy (1865) conducted large-scale experiments on wrought iron and steel to study their mechanical properties, providing early insights into the brittleness of steel structures. However, it wasn't until the 20th century that fracture mechanics matured into a structured scientific discipline.

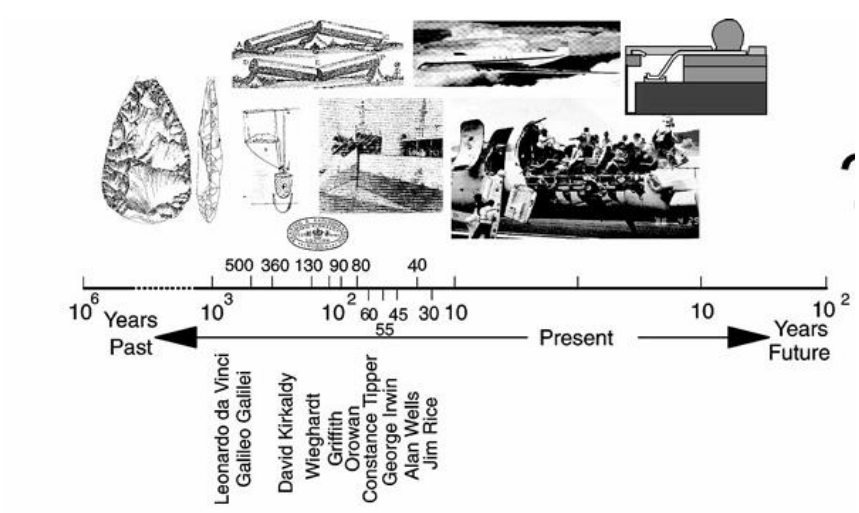

Figure 1 History of fracture mechanics [1]

4.1.2 The Birth of Modern Fracture Mechanics: Griffith's Theory

The field of fracture mechanics was revolutionized by Alan Arnold Griffith (1920), who introduced an energy-based theory to explain why materials fracture. His work, originally aimed at understanding the low strength of glass, introduced the concept that cracks propagate when the energy released by an advancing crack exceeds a critical threshold. This idea formed the foundation of Linear Elastic Fracture Mechanics (LEFM) and established the importance of crack length and stress intensity in determining fracture behaviour.

Griffith's work remained largely theoretical until the Second World War, when catastrophic failures in welded steel structures, ships, and aircraft highlighted the need for a deeper understanding of fracture. The brittle fracture of Liberty ships and the failure of the Comet aircraft due to fatigue cracks in the 1950s led to the widespread adoption of fracture mechanics in engineering design.

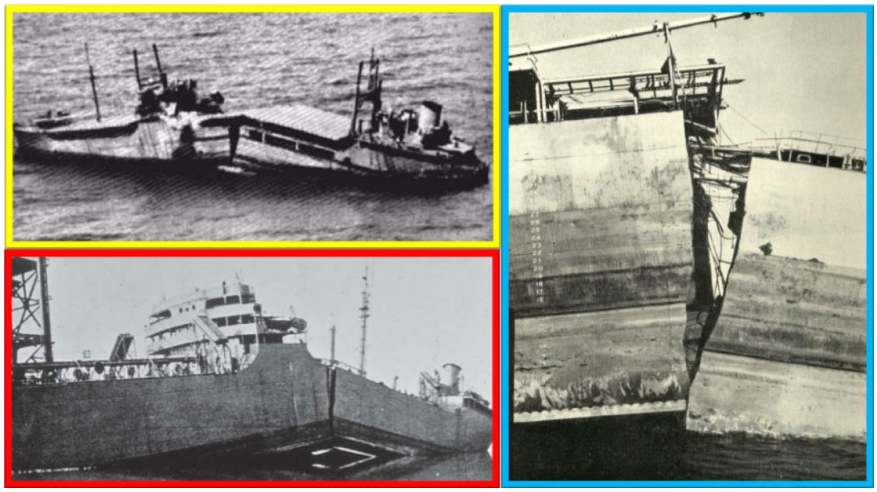

Figure 2 Brittle fracture caused in Liberty Ship, yellow box- top view, red box- side view, blue box- fracture at cantered

4.1.3 From LEFM to Nonlinear Fracture Mechanics

While LEFM was effective for brittle materials, it struggled to explain the fracture behaviour of ductile and quasi-brittle materials, which exhibit softening behaviour before failure. This limitation led to the development of Elasto-Plastic Fracture Mechanics (EPFM) and Cohesive Zone Models (CZM), which accounted for plastic deformation and damage evolution.

By the late 20th century, computational techniques such as Finite Element Methods (FEM) and Phase Field Models (PFM) emerged as powerful tools for simulating fracture. The phase field method in particular gained traction due to its ability to represent fractures as a continuous field, eliminating the need for explicit crack tracking. However, these methods are computationally expensive, especially for complex crack propagation scenarios.

4.2 Phase Field Model for Fracture Mechanics

Fracture modelling has traditionally relied on discrete methods, such as the cohesive zone model (CZM) and extended finite element method (XFEM), which explicitly track crack paths. However, these methods become computationally expensive and difficult to implement for complex fracture problems where crack initiation, branching, and merging occur. To overcome these challenges, the phase field model (PFM) has emerged as a powerful alternative, enabling the representation of fractures as a diffuse damage field rather than a sharp discontinuity.

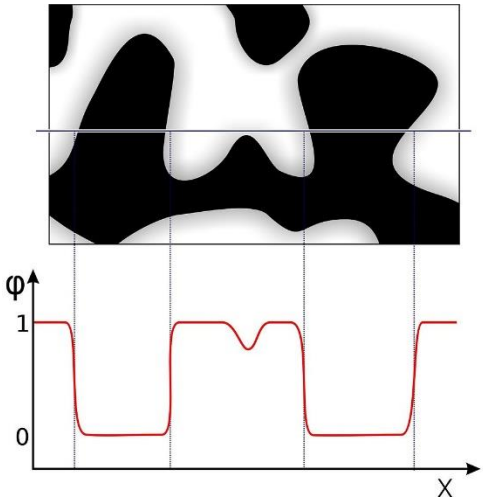

Figure 3 A two phase microstructure and the order parameter profile is shown on a line across the domain. Gradual change of order parameter from one phase to another show's diffuse nature of the interface.

The phase field model is based on thermodynamic principles and is widely used for predicting brittle and quasi-brittle fracture. Instead of defining a crack as a distinct discontinuity, the phase field approach introduces a continuous variable $\varphi(x,t)$, where

a) $\varphi = 0$ represents intact material,

b) $\varphi = 1$ represents a fully fractured region,

c) $0 < \varphi < 1$ represents the fracture process zone, where the material is progressively degrading.

This method eliminates the need for explicit crack tracking and allows for automatic crack initiation and propagation, making it particularly useful for modelling complex fracture patterns.

4.2.1 Mathematical Formulation of the Phase Field Model

The phase field approach is derived from Griffith's fracture theory, which states that a crack propagates when the energy released due to crack growth exceeds the material's fracture energy. The total energy functional for a solid undergoing fracture consists of two main components[2]:

$$\varepsilon = \int_{\Omega} \left(g(\varphi)\Psi_e + \frac{G_c}{c_\omega}\big(\omega(\varphi)\big) + l_o^2|\nabla\varphi|^2 \right) d\Omega \qquad (1)$$

where

a) Ψ_e is the elastic strain energy density,

b) $g(\varphi) = (1-\varphi)^2$ is the degradation function,

c) G_c is the critical energy release rate,

l_o is the length scale parameter, which controls the transition width between fractured and intact regions,

d) $\omega(\varphi)$ is a potential function that governs the crack evolution,

e) c_ω is a normalization constant ensuring proper energy scaling.

The governing Euler-Lagrange equations derived from this energy formulation define the coupled behaviour of elastic deformation and phase field evolution, leading to:

1. Equilibrium equation for mechanical deformation:

$$\nabla \cdot \sigma = 0 \tag{2}$$

where σ is the Cauchy stress tensor.

2. Phase field evolution equation:

$$\frac{G_c}{l_o}\varphi - G_c l_o \nabla^2 \varphi = -g'(\varphi)H(x,t) \tag{3}$$

where H(x,t) is the history field, ensuring crack irreversibility.

These equations are solved simultaneously, making phase field models computationally intensive. However, they offer superior accuracy in predicting complex fracture phenomena compared to traditional methods.

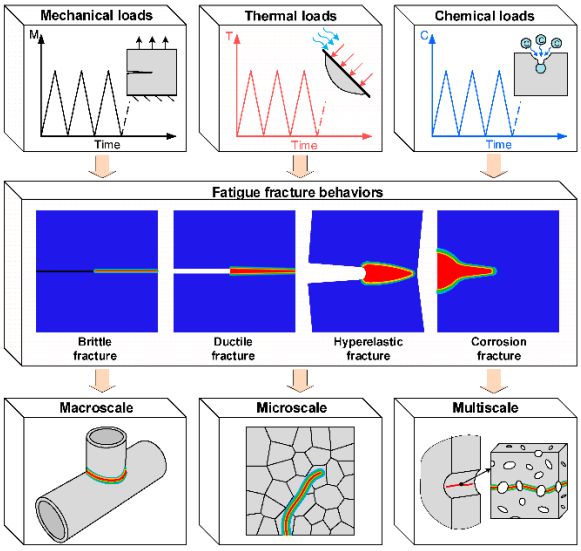

Figure 4 Fatigue fracture behaviours simulated with phase field methods[3]

4.3 Physics-Informed Neural Networks (PINNs) for Fracture Mechanics

Physics-Informed Neural Networks (PINNs) represent a significant advancement in the integration of machine learning with scientific computing. Unlike traditional machine learning models that require vast amounts of labelled data, PINNs incorporate physical laws directly into the training process. This approach allows the network to learn solutions to partial differential equations (PDEs) without the need for large datasets. The key advantage of PINNs

is that they combine the power of deep learning with the rigor of governing physical equations, ensuring that the predictions made by the model are consistent with the underlying physical principles.

In the context of fracture mechanics, PINNs have proven to be especially effective for solving complex problems related to damage propagation and crack growth. Traditional numerical methods, such as the Finite Element Method (FEM), are often computationally expensive, particularly when dealing with fine mesh resolutions required near crack tips or complex geometries. In contrast, PINNs are mesh-free and can handle arbitrary crack shapes and material properties, making them more versatile and computationally efficient for these types of problems.

PINNs work by approximating the solution $u(x,t)$ to a PDE, where eq. (4) represents

$$\mathcal{N}[u(x,t)] = 0 \quad x \in \Omega, \quad t \in [0,T] \tag{4}$$

the governing physics (e.g., equilibrium equations, stress-strain relationships, energy principles). Specifically, for fracture mechanics problems, PINNs are used to predict displacement fields $u(x,t)$ and the phase field variable $\phi(x,y)$, which describes the evolution of cracks in the material. The model incorporates boundary conditions and ensures the solution is physically valid by minimizing a combined loss function that includes:

1. Residual loss to ensure the predicted solution satisfies the governing PDEs,
2. Boundary loss to enforce boundary and loading conditions,
3. Energy loss to maintain consistency with the phase-field energy functional.

By training the network to minimize these losses, PINNs can predict solutions that not only approximate the displacement but also respect the material's physical behavior under load, including damage evolution.

The main advantage of using PINNs in fracture mechanics is their ability to integrate physical laws into the neural network's training, which results in a more accurate and efficient solution compared to traditional methods. PINNs do not require explicit meshing or resolution around crack tips, which makes them a promising tool for problems involving fracture and damage in materials. Moreover, once trained, the model can predict solutions much faster than traditional numerical solvers, making them more suitable for real-time predictions or simulations with large-scale geometries.

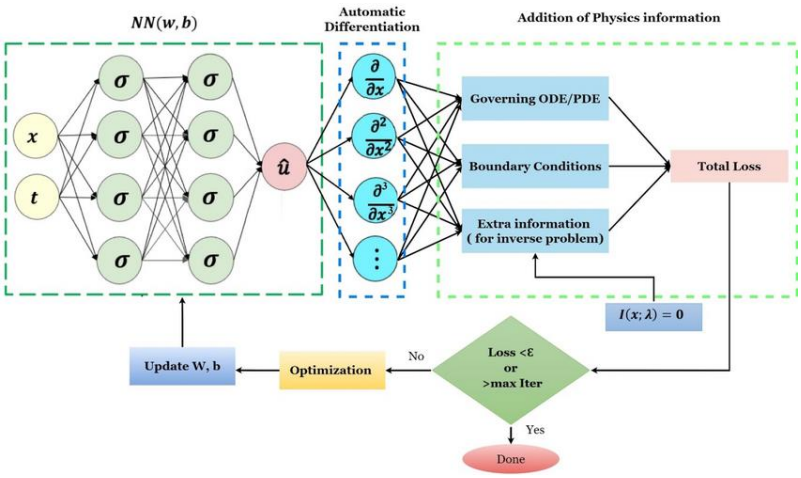

Figure 5 Schematic of PINN framework*[4]*

4.4 Introduction to Machine Learning in Fracture Mechanics

Fracture mechanics has traditionally relied on methods like the Finite Element Method (FEM) and Phase Field Models (PFM) to predict crack initiation and propagation. However, these approaches can be computationally expensive and require fine mesh resolutions, especially near crack tips. As fracture problems grow more complex in engineering applications, there is a need for faster and more adaptable methods to handle these challenges.

Machine Learning (ML) is increasingly being used to address these limitations in fracture mechanics. Unlike traditional methods, ML models can learn from data and predict crack behavior, stress distribution, and material degradation without needing detailed meshing or large datasets. This makes ML models highly efficient, especially for real-time predictions in structural health monitoring and material design. ML is applied in fracture mechanics in several ways: it can predict material degradation, estimating the softening behavior of quasi-brittle materials, which reduces the reliance on costly empirical testing. In structural health monitoring, ML-based crack detection algorithms analyze data from sensors or images of structures such as bridges, aircraft, and pipelines, helping assess their integrity. Moreover, ML-driven surrogate models are used to replace expensive FEM simulations, enabling faster predictions of crack growth and material failure, which is essential in engineering design where multiple fracture scenarios need to be evaluated rapidly.

4.4.1 Key Concepts in ML for Fracture Mechanics

1. Data-Driven Approaches:
   Supervised learning models, such as neural networks and random forests, are trained on simulation or experimental data to predict key variables like stress, strain, and crack growth. These methods are especially useful when large datasets are available.
2. Physics-Informed Machine Learning (PINNs):
   Unlike traditional ML models, Physics-Informed Neural Networks (PINNs) integrate fracture mechanics principles directly into the network's training process. This ensures that the network's predictions are physically consistent with the governing equations, such as those describing crack propagation. PINNs can solve partial differential equations (PDEs) efficiently, without needing large training datasets.
3. Surrogate Models:
   ML can also be used to build surrogate models that replace computationally expensive FEM simulations. These models can quickly predict crack growth and material

failure, which is crucial in engineering design when multiple fracture scenarios need to be evaluated rapidly.

In summary, Machine Learning, particularly PINNs, offers a promising solution to the challenges in fracture mechanics by combining data-driven approaches with physics-based constraints. This allows for faster, more accurate predictions of material behavior, crack evolution, and damage, without the high computational costs of traditional methods. Furthermore, ML's ability to adapt to complex geometries, integrate experimental data, and speed up simulations makes it a powerful tool for advancing structural health monitoring and material design in the field of fracture mechanics.

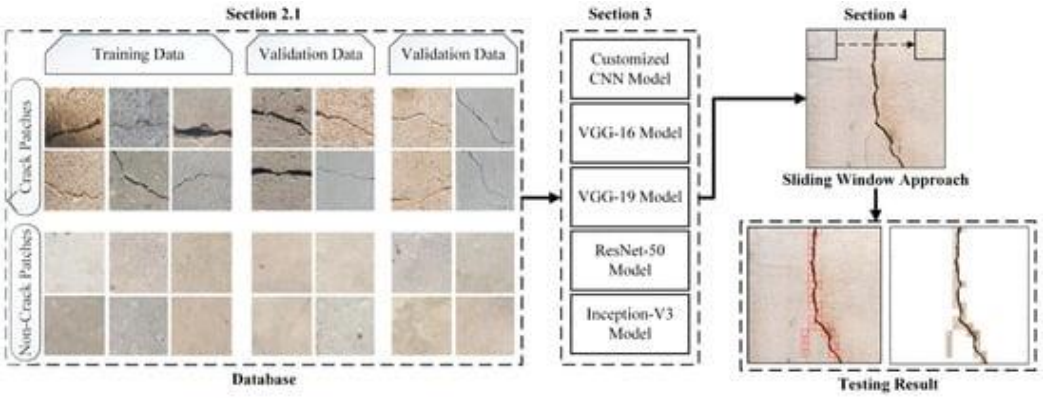

Figure 6 Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures

4.5 Softening Parameters in Fracture Mechanics

In fracture mechanics, particularly for quasi-brittle materials like concrete, ceramics, and some metals, failure does not occur suddenly but through a gradual softening process. This means that as cracks form and propagate, the material experiences a progressive loss of stiffness and load-bearing capacity before complete failure. Accurately modelling this softening behaviour is critical for predicting material degradation and structural failure.

Softening in fracture mechanics is often represented through damage models that reduce material stiffness based on crack evolution. Traditional numerical methods like Finite Element Method (FEM) and Cohesive Zone Models (CZM) have been used to model softening behaviour, but they require careful calibration of material parameters. With the rise of Phase Field Models (PFM) and Machine Learning (ML), researchers now aim to predict softening parameters more efficiently.

4.5.1 Mathematical Representation of Softening

Softening behaviour in fracture mechanics is often captured using stress-strain relationships or degradation functions in the phase field model. The most commonly used degradation function is:

$$g(\varphi) = (1-\varphi)^2 \qquad (8)$$

Here, φ is the phase field variable that represents material damage and the values represent the material degradation state as follows:

a) φ=0 → Intact material

b) φ=1 → Fully fractured material

The softening behaviour is characterized by material parameters such as:

a) Critical energy release rate (G_c) – The energy required for crack propagation.

b) Fracture toughness (K_c) – The resistance of a material to crack growth.

c) Softening modulus (H_s) – Determines how quickly a material loses stiffness as damage progresses.

In phase field modelling, the softening parameter controls how the material transitions from elastic to fully fractured, influencing crack speed and shape.

5 Methodology

5.1 Modelling Approach

In this project, the primary objective is to predict displacement and damage evolution in materials under applied loads, using a Physics-Informed Neural Network (PINN) model integrated with the Phase-Field method. The Modelling Approach combines traditional fracture mechanics principles with the flexibility of machine learning to predict crack initiation, growth, and the resulting material degradation. The Phase-Field method is used to model the damage evolution in materials, while PINNs are employed to incorporate physical laws directly into the learning process, ensuring that the predictions are both accurate and physically consistent.

The Phase-Field model and PINNs are both essential for capturing the complex interactions between material properties, external loads, and crack propagation, which can be computationally expensive with traditional methods like Finite Element Method (FEM). By using PINNs, the model can learn the solution to partial differential equations (PDEs) governing fracture mechanics, without requiring explicit meshing or large amounts of labeled data, thus offering a more computationally efficient alternative.

5.1.1 Phase-Field

The Phase-Field method is a powerful numerical approach used for modelling fracture in materials. Unlike traditional methods, which require explicit tracking of crack surfaces, the Phase-Field method represents cracks using a continuous damage variable, denoted as $\phi(x, y)$, that smoothly transitions from 0 (representing intact material) to 1 (representing fully fractured material). This smooth representation of cracks simplifies the computational process, especially in dynamic fracture problems where cracks evolve over time.

In this project, we apply the Phase-Field model to simulate the damage evolution in the material as it undergoes loading. The crack propagation is governed by the energy balance between the elastic energy of the material and the fracture energy required for the formation of new cracks.

The total energy functional is the central quantity being minimized. It captures both the elastic energy stored in the material and the fracture energy dissipated due to crack evolution.

$$E_{total}(u,\phi) = \int_{\Omega} g(\phi)\,\Psi\big(\epsilon(u)\big)d\Omega + \int_{\Omega} \frac{G_c}{C_o}\left(\frac{\alpha(\phi)}{l_c} + l_c|\nabla\phi|^2\right)d\Omega \quad (9)$$

Where u is displacement field, ϕ Phase-field variable, ϕ=0 (intact), ϕ=1 (fully broken).$\epsilon(u)$ Linearized strain tensor. $\Psi(\epsilon) = \frac{1}{2}\epsilon_T . C . \epsilon$: Strain energy density.$g(\phi)$ Degradation function G_c : Fracture toughness (critical energy release rate). $\alpha(\phi)$ Crack density function. l_c Length scale parameter that controls the width of the diffused crack.

The evolution of crack is characterized by the crack geometric function [5]

$$\alpha(\phi) = 2\phi - \phi^2 \;\forall\; \alpha \in [0,1] \quad (10)$$

The energy degradation function, $g(\phi) \in [0,1]$, has a direct effect on the elastic stiffness of the material , and satisfy the following properties,[5], [6]

$$g(\phi) = \frac{(1-\phi)^p}{(1-\phi)^p + a_1(1 + a_2\phi + a_2a_3\phi^2)} \quad (11)$$

Here for linear softening type , $a_1 = \frac{4E_oG_c}{f_t^2 l_c C_o}$, $a_2 = -0.5$, $a_3 = 0$, $p = 2$ taken from[7]

Given , $C_o = \pi$, $Young's\ modulus\ (E_o) = 3 \times 10^4 \frac{N}{mm^2}$, $l_c = 10$, $tensile\ strength\ (f_t) = 3$, $G_c = 0.008$

Residual equation with respect to displacement –

$$R_u(x) = \nabla(g(\phi)\sigma) \quad (12)$$

where σ is the Cauchy stress tensor.

Phase-field residual equation –

$$R_\phi(x) = g(\acute{\phi}).\varphi(\epsilon) - \frac{G_c}{C_o}\left(\frac{\alpha(\acute{\phi})}{l_c} - 2l_c\nabla\phi\right) \quad (13)$$

5.1.2 PINNS

Physics-Informed Neural Networks (PINNs) offer a robust framework for solving partial differential equations (PDEs), which are typically involved in problems like fracture mechanics. PINNs work by incorporating the governing physical laws into the loss function of the neural network, ensuring that the network's predictions are consistent with the physical behavior of the system. This is especially useful when solving problems that require simulating

complex physical phenomena, such as damage propagation and displacement fields in materials under load.

In this project, PINNs are used to predict both the displacement field u(x,y) and the phase-field damage variable $\phi(x,y)$ with the loss function consisting of several key terms:

1. Residual loss: Ensures the predicted solution satisfies the governing PDEs for displacement and damage evolution.
2. Boundary loss: Enforces boundary conditions, such as prescribed displacement or force values.
3. Energy loss: Ensures the predicted solution is consistent with the energy principles governing material behavior.

By minimizing this combined loss function, the neural network is trained to predict displacement and damage evolution, while adhering to the underlying physics of fracture mechanics. The model is trained on scaled input data, such as the coordinates and applied load, and produces the corresponding displacement and damage predictions.

PINNs significantly reduce the computational burden associated with traditional FEM or FDM by eliminating the need for explicit meshing and solving the governing equations directly through the neural network's optimization process. This approach makes it well-suited for complex geometries and boundary conditions, as it can handle intricate crack paths and material heterogeneity more efficiently than conventional methods.

5.2 Data Collection and Preprocessing

The dataset was obtained from an FEM-based numerical simulation of an axial bar under load. The input feature consisted of the applied force at the last node, while the target variables were the displacements recorded at 46 different nodes along the bar. Given the boundary condition displacement at the first node was zero means fixed from one end. The displacement values were significantly small, requiring appropriate scaling before training the model.

To ensure numerical stability and improve model training, these preprocessing steps were applied:

Preprocessing Steps:

1. Scaling the Data:
    - The displacement values were significantly small, which could lead to numerical instability during model training.
    - Initially, the displacement values were manually scaled by multiplying them with a factor to bring them into a manageable range. However, the results were not as effective as anticipated.
    - To resolve this, the StandardScaler from Scikit-learn was used to normalize the data. The StandardScaler ensures that all features are transformed to have a mean of zero and a standard deviation of one, making it easier for the model to learn efficiently.
2. Feature Scaling:

- MinMaxScaler was used on features like X and Y coordinates and load to scale them between the range of [0, 1]. This standardization helps avoid the model being biased towards features with larger magnitudes.
- This scaling ensures that all the input features are in the same numerical range, preventing the model from giving undue importance to variables with larger numerical values.

3. Train-Test Split:
    - The data was split into training and validation sets using an 80-20 split ratio. This means:
        - 80% of the data was used for training the model.
        - 20% of the data was kept aside for validation to test the model's performance on unseen data.
    - The train-test split is crucial to ensure that the model does not overfit to the training data and can generalize well to new, unseen data.
4. Shuffling and Batching:
    - Shuffling was applied to the training dataset before batching to ensure that the model does not learn the order of the data, which could lead to overfitting.
    - The dataset was divided into batches of size 32 to improve training efficiency. Training in batches helps the model converge faster and reduces memory usage during training.
5. Tensor Conversion:
    - All input data was converted into TensorFlow tensors to facilitate seamless training with TensorFlow's neural network architecture. This ensures efficient handling of the data during the training process, especially when performing automatic differentiation.
6. Outcome of Preprocessing:
    - By applying these preprocessing steps, the dataset was scaled appropriately and split into training and validation sets, providing a solid foundation for training the model.
    - These steps were crucial in ensuring that the model would train efficiently, without being overwhelmed by large disparities in feature values or overfitting to the training data.

5.3 Neural Network Architecture

The Neural Network (NN) architecture was designed to model the complex relationships between the applied load and the resulting displacements at various nodes along the bar. Given the nature of the problem, where we need to predict continuous values (displacement) based on input features (applied load, node positions), a regression-based neural network was used.

The model was built with multiple dense layers to learn non-linear mappings, and an output layer that provides the final predictions.

1. Model Structure

- Input Layer:
- The input layer receives three variables: (1) the applied load at the terminal node of the bar, (2) the X-coordinate, and (3) the Y-coordinate of the point at which the displacement and damage values are to be predicted.
- Hidden Layers:
    - The architecture consists of six hidden layers, each fully connected with tanh activation functions. The ReLU was used in output layer to ensure non negative outputs.
    - The number of neurons in each hidden layer decreases progressively to reduce the model's complexity while still capturing the relevant patterns:
        - First hidden layer: 2048 neurons.
        - Second hidden layer: 1024 neurons.
        - Third hidden layer: 512 neurons.
        - Fourth hidden layer: 256 neurons.
        - Fifth hidden layer: 128 neurons.
        - Sixth hidden layer: 64 neurons.
- Output Layer:
    - The output layer consists of 2 neurons one for predicting displacement and other for predicting damage at specified point.

2. Model Training

- The model was compiled using the Adam optimizer with a learning rate of 0.0001. After initial training with Adam, L-BFGS optimizer was applied to fine-tune the weights. L-BFGS is a quasi-Newton method known for its fast and precise convergence in smooth optimization landscapes, making it effective for refining solutions found by stochastic optimizers.

3. Physics-Informed Loss

To ensure the model's predictions respect the governing physics of the fracture mechanics problem, the Physics-Informed Loss was incorporated into the model. This loss includes terms for both data loss (MSE between predicted and true values) and physics-based loss (ensuring the model satisfies the PDEs that govern displacement and damage evolution).

The Physics Loss function combines:

- Residual loss: Enforces the governing equations of the material, ensuring that the predicted displacement and damage fields satisfy the equilibrium equations.

- Boundary loss: Ensures the model adheres to boundary conditions (e.g., displacement at the first node is zero).
- Energy loss: Guarantees consistency with the phase-field energy functional, which models crack propagation.

The physics loss is calculated using the physics_loss function in the code, which computes the residuals for both the displacement field and phase-field damage variable. These residuals are then added to the standard loss function (Huber loss) during training to enforce physical consistency.

4. Model Evaluation

- Metrics: Metric computes the relative difference between the predicted displacement values and the actual displacement values across all nodes. The R^2 score and mean squared error (MSE) were calculated during training to monitor the model's performance.

5. Training Process

The model was trained using batch gradient descent. For each batch of data, the gradients were computed using backpropagation with the physics-informed loss. The Adam optimizer updated the model's weights, minimizing the combined loss function (physical loss + data loss). The process was repeated for 50 epochs, with the model evaluated on the validation set after each epoch. After completion of training model weights were re-computed using L-BFGS.

- Shuffling and Batching: The data was shuffled and batched to ensure the model does not learn from the sequence of the data and to improve convergence during training.

5.4 Computational Tools and Environment

The model was developed and trained using a combination of popular libraries and frameworks to ensure efficient computation and effective training of the Physics-Informed Neural Network (PINN). Below is an overview of the computational tools and environment used in this project:

1. Programming Language and Frameworks

- Python: The primary programming language used for developing and implementing the neural network model. Python is widely used for machine learning tasks due to its extensive libraries and frameworks.
- TensorFlow: The TensorFlow 2.0 framework, an open-source deep learning library developed by Google, was used to build and train the neural network. TensorFlow allows for easy creation of deep learning models and supports GPU acceleration, which significantly reduces training time for large datasets.
    - Keras: Built on top of TensorFlow, Keras is a high-level neural network API that simplifies the model-building process, offering easy-to-use tools for defining layers, loss functions, and optimizers.

- NumPy: Used for efficient numerical computations and array manipulations. NumPy provides essential support for working with matrices and tensors in Python, which is crucial for handling the input data and model parameters.
- Scikit-learn: Used for preprocessing the dataset, particularly the MinMaxScaler and train_test_split functions. Scikit-learn also provides essential tools for splitting data into training and validation sets and scaling the features.

2. Hardware
   - CPU/GPU: The model was trained using the GPU available on the system i.e, NVIDIA Geforce RTX 3050 GPU 4 GB, which accelerated the training process significantly. TensorFlow automatically utilizes available GPUs if configured correctly, speeding up the backpropagation and optimization steps.

3. Development Environment
   - Jupyter Notebook: For experimentation and prototyping, Jupyter Notebook was used to test and visualize intermediate results, including the output of the model at each epoch, loss functions, and predictions. This environment allows for step-by-step development and visualization, aiding in debugging and model tuning.
   - Version Control: Git was used for version control to manage changes to the code and track the progress of model development. GitHub was employed as a remote repository for collaboration and backup.
4. Software Environment Configuration
   - Python Version: The code was written in Python 3.8, which is compatible with TensorFlow 2.0 and all associated libraries.
   - TensorFlow Version: The code was executed using TensorFlow 2.0, which provides an intuitive and flexible interface for building deep learning models with support for eager execution, dynamic computation graphs, and other enhancements.
   - Operating System: The project was developed on Windows 11 .

6 Results and Discussion

6.1 Model Evaluation

The model was trained for 50 epochs with a batch size of 128. The performance was assessed using:

a) Loss and Validation Loss: The loss function values were monitored over training epochs to evaluate convergence.

b) Mean Percentage Error: Used to assess prediction accuracy.

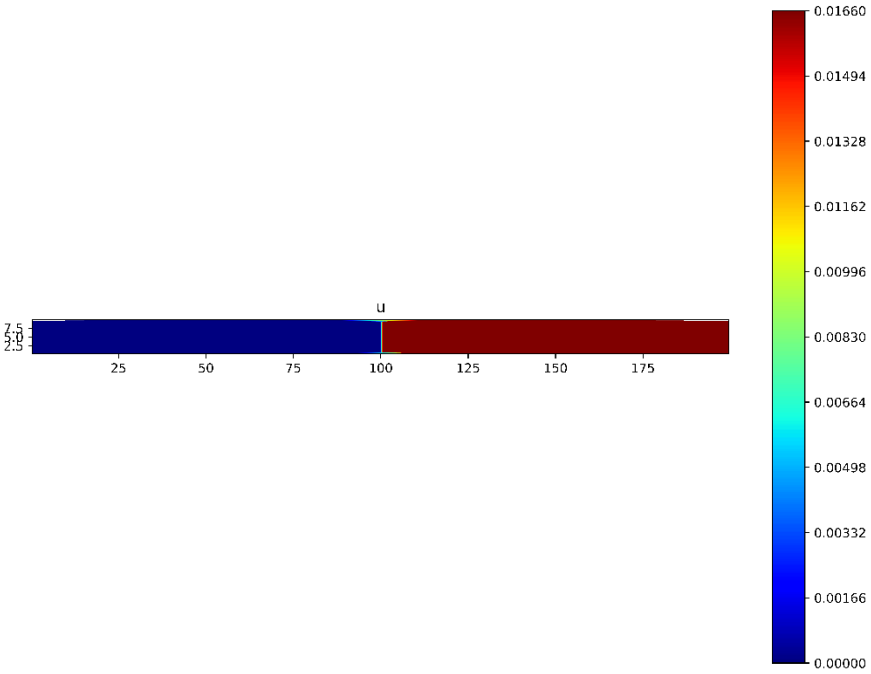

Figure 7 Actual displacement (u) along the bar

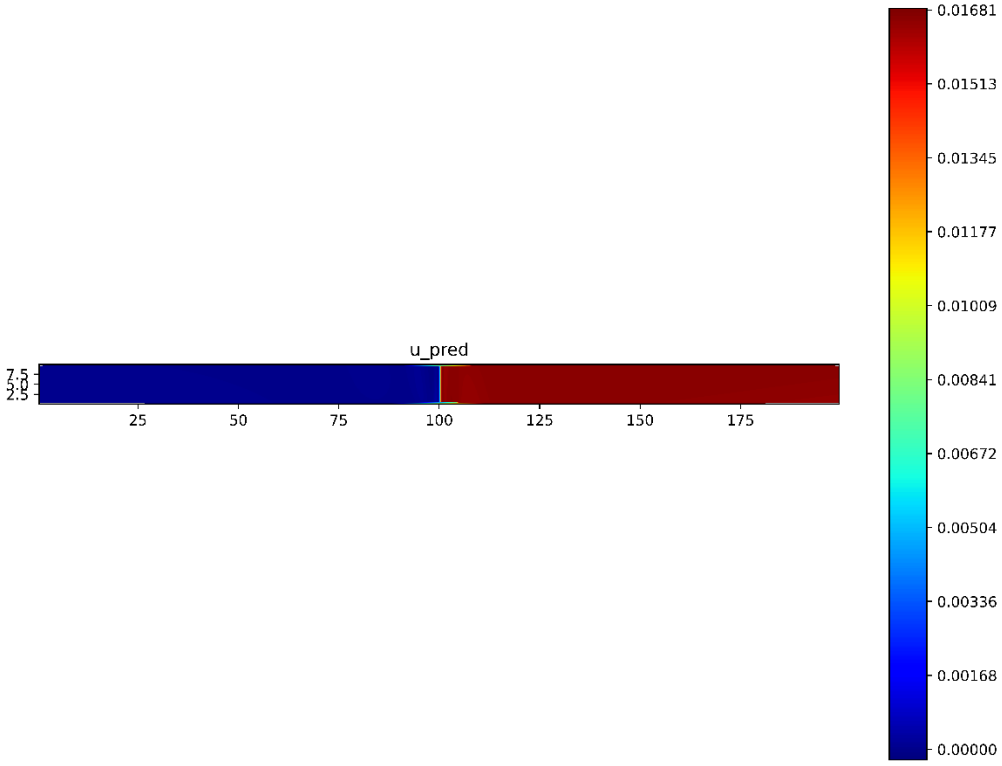

Figure 8 Predicted displacement distribution (u_pred) along the bar

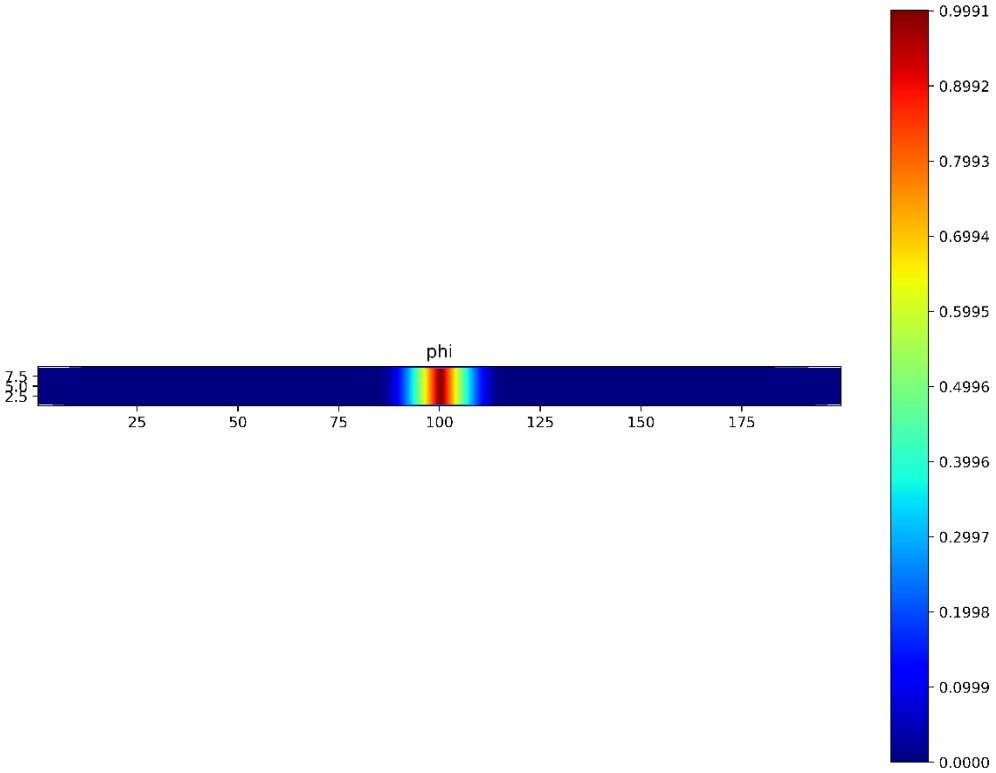

Figure 9 True phase-field damage distribution (ϕ) along the bar, indicating crack initiation and propagation.

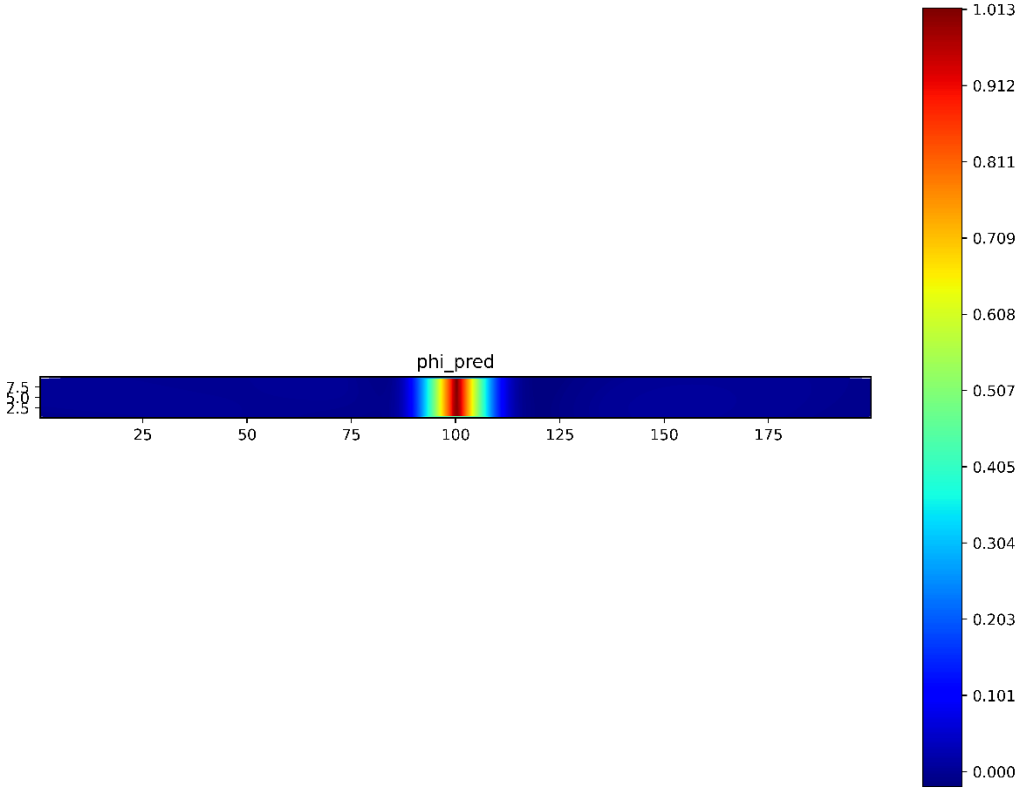

Figure 10 Predicted phase-field damage distribution (ϕ) along the bar

7 Discussion

In this work, all the computations were done for a single load step, meaning the load was applied only once. However, in real-life situations, loads can change over time. So, this method needs to be extended to handle multiple load steps to better represent how structures behave in real conditions.

8 Conclusion and future work

This project successfully developed a Physics-Informed Neural Network (PINN) model to predict displacement and damage evolution in a material under axial loading. The model effectively captured the relationship between applied load, displacement, and crack propagation, using the Phase-Field method and physics-informed loss functions to ensure physical consistency. The predictions for displacement and phase-field damage closely matched the true values from the simulation, offering a promising alternative to traditional methods like FEM for fracture mechanics problems.

Future Work:

Future work will focus on predicting the softening behavior of quasi-brittle materials, which is essential for modelling material failure as cracks propagate. The next steps include:

- Incorporating Softening Behavior into the Phase-Field model, adjusting the damage evolution equations to account for decreasing material stiffness as cracks grow.
- Training the Model to predict softening parameters along with displacement and damage.
- Validating the enhanced model using experimental or simulation data to assess its accuracy in predicting softening under different loading conditions.

Expanding the model to include softening behavior will improve its ability to simulate material failure and enhance its applicability in engineering design and structural health monitoring.

9 References

[1] B. Cotterell, “The past, present, and future of fracture mechanics.” [Online]. Available: www.elsevier.com/locate/engfracmech

[2] S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk, “Transfer learning enhanced physics informed neural network for phase-field modeling of fracture,” *Theoretical and Applied Fracture Mechanics*, vol. 106, Apr. 2020, doi: 10.1016/j.tafmec.2019.102447.

[3] H. Cui, C. Du, and H. Zhang, “Applications of Phase Field Methods in Modeling Fatigue Fracture and Performance Improvement Strategies: A Review,” Apr. 01, 2023, *MDPI*. doi: 10.3390/met13040714.

[4] A. Fallah and M. M. Aghdam, “Physics-informed neural network for bending and free vibration analysis of three-dimensional functionally graded porous beam resting on elastic foundation,” *Eng Comput*, vol. 40, no. 1, pp. 437–454, Feb. 2024, doi: 10.1007/s00366-023-01799-7.

[5] J. Y. Wu, “A unified phase-field theory for the mechanics of damage and quasi-brittle failure,” *J Mech Phys Solids*, vol. 103, pp. 72–99, Jun. 2017, doi: 10.1016/j.jmps.2017.03.015.

[6] E. Lorentz and V. Godard, “Gradient damage models: Toward full-scale computations,” *Comput Methods Appl Mech Eng*, vol. 200, no. 21–22, pp. 1927–1944, May 2011, doi: 10.1016/j.cma.2010.06.025.

[7] A. Pandey and S. Kumar, “A multi-level adaptive mesh refinement strategy for unified phase field fracture modeling using unstructured conformal simplices,” *Comput Methods Appl Mech Eng*, vol. 433, Jan. 2025, doi: 10.1016/j.cma.2024.117514.