Pre Lab:-

1. What are TensorFlow operations, and how do they relate to mathematical computations?
TensorFlow operations (ops) are units of computation, like addition or matrix multiplication, that execute mathematical functions on tensors, allowing for complex computations in machine learning models.

2. Explain evaluation methods of the model's performance.
Common methods include metrics like accuracy for classification, mean squared error (MSE) for regression, and cross-validation to assess model generalization on unseen data.

3. What is a sequential model in PyTorch?
A sequential model in PyTorch is a container that allows layers to be stacked in sequence, where the output of one layer becomes the input for the next, simplifying the construction of feed-forward neural networks.

VIVA:-

1. PyTorch supports both symbolic and numerical computation. Can you explain the difference between these two approaches and their applications?
Symbolic computation deals with abstract mathematical symbols (e.g., derivatives or algebraic expressions), while numerical computation evaluates these expressions using real numbers. PyTorch focuses on numerical computation, using automatic differentiation for deep learning models.

2. Why would you use a PyTorch placeholder, and how does it facilitate the dynamic input of data into a computational graph?
PyTorch does not use traditional placeholders like TensorFlow. Instead, dynamic input is handled directly with tensors, which allows for flexible and dynamic graph construction as data changes during runtime.

3. Polynomial equations can be non-linear. How does PyTorch handle the solution of non-linear equations, and what considerations should be taken into account?
PyTorch solves non-linear equations using optimization techniques like gradient descent. Care must be taken with convergence, initialization, and loss functions to ensure accurate solutions.

4. If given a polynomial equation of a higher degree, how would you extend your PyTorch-based solution? Are there limitations to the degree of polynomials that PyTorch can effectively handle?
You can extend the solution by increasing the complexity of the model or the number of parameters. PyTorch can handle high-degree polynomials, but as the degree increases, issues like overfitting and computational complexity can arise, requiring regularization and careful design.