Name: Sneha Verma

SRN: PES1UG23AM309

Course: Machine Learning

Date: September 14, 2025

**1. Introduction**

**Purpose**

The primary objective of this lab was to build, train, and evaluate a neural network from scratch to perform a non-linear regression task. The goal was to approximate a complex polynomial function using a multi-layer perceptron, understand the impact of various hyperparameters on model performance, and identify key training phenomena like underfitting and convergence.

**Tasks Performed**

1. Implemented the core components of a neural network, including weight initialization (Xavier), forward propagation, the ReLU activation function, Mean Squared Error (MSE) loss, and backward propagation.
2. Generated a unique dataset based on a "Cubic + Sine" polynomial function determined by the assigned SRN.
3. Trained a baseline neural network model and evaluated its initial performance.
4. Conducted a series of hyperparameter tuning experiments by systematically varying the **learning rate**, **number of epochs**, and **activation function**.
5. Analyzed the results of each experiment by observing loss curves, prediction plots, and performance metrics like the R² score to draw conclusions about optimal model configuration.

**2. Dataset Description**

The dataset was programmatically generated based on a unique student ID.

* **Polynomial Type:** The assigned function was a **Cubic + Sine** polynomial:

y = 2.41x³ + 0.48x² + 3.54x + 11.97 + 12.6\*sin(0.040x)

* **Samples & Features:** The dataset consists of **100,000 samples**. It has a single input feature (x) and a single target variable (y). The data was split into 80,000 training samples and 20,000 test samples.
* **Noise Level:** Gaussian noise with a standard deviation of **2.37** (ε ~ N(0, 2.37)) was added to the target variable to simulate real-world data imperfections.

**3. Methodology**

The neural network was constructed with a feedforward architecture determined by the assigned SRN: **Input(1) → Hidden(96) → Hidden(96) → Output(1)**.

1. **Initialization:** Weights were initialized using **Xavier Initialization** to maintain variance across layers and prevent vanishing or exploding gradients. Biases were initialized to zero.
2. **Forward Propagation:** For a given input x, the network calculated a prediction ŷ by passing the data through two hidden layers. The **ReLU (Rectified Linear Unit)** activation function was applied after each hidden layer to introduce non-linearity, which is essential for learning complex curves. The output layer used a linear activation, standard for regression tasks.
3. **Loss Calculation:** The **Mean Squared Error (MSE)** was used as the loss function to quantify the difference between the predicted values (ŷ) and the actual values (y).
4. **Backward Propagation:** The gradient of the loss with respect to each weight and bias was calculated using the chain rule. This process determined the direction and magnitude of the adjustments needed to improve the model.
5. **Optimization:** A standard **Gradient Descent** algorithm was used to update the model's weights and biases. The update rule is: weight = weight - learning\_rate \* gradient.
6. **Hyperparameter Tuning:** A baseline model was first trained. Subsequently, four additional experiments were conducted to find a better model by tuning the **learning rate**, **number of training epochs**, and testing an alternative activation function (**Leaky ReLU**). Early stopping with a patience of 30-50 epochs was used to prevent unnecessary training.

**4. Results and Analysis**

Five experiments were conducted to analyze the model's performance.

**Experiment 1: Baseline Model (LR=0.003, Epochs=500)**

* **Final Test MSE:** 0.17558
* **R² Score:** 0.82577
* **Discussion:** The baseline model shows significant **underfitting**. The training loss curve indicates that the model has not fully converged after 500 epochs. The prediction plot clearly shows the model's failure to capture the complex, non-linear shape of the actual data, resulting in a low R² score.

**Experiment 2: Increased Epochs (LR=0.003, Epochs=1500)**

* **Final Test MSE:** 0.08867
* **R² Score:** 0.91201
* **Discussion:** By simply increasing the training duration to 1500 epochs, the model's performance improved dramatically. The loss curve converges successfully to a much lower value. The prediction plot shows an excellent fit, demonstrating that the model has now learned the underlying function. This highlights that the primary issue with the baseline was insufficient training time.

**Experiment 3: Lower Learning Rate (LR=0.001, Epochs=1500)**

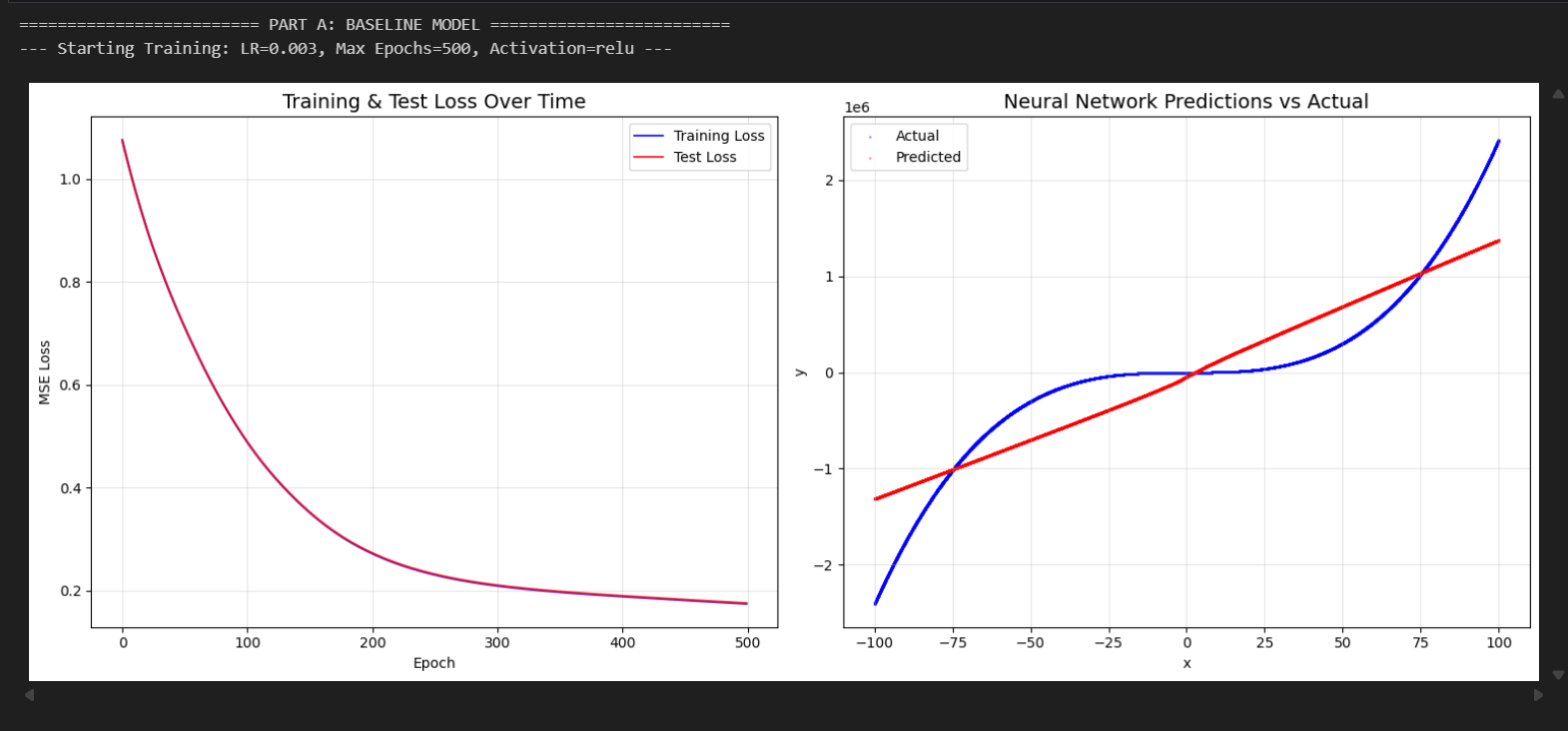
* **Final Test MSE:** 0.17560
* **R² Score:** 0.82576
* **Discussion:** This experiment with a lower learning rate performed poorly. The loss curve shows extremely slow convergence, failing to reach a good minimum even after 1500 epochs. The performance is almost identical to the undertrained baseline, indicating that the learning rate was too small for efficient training.

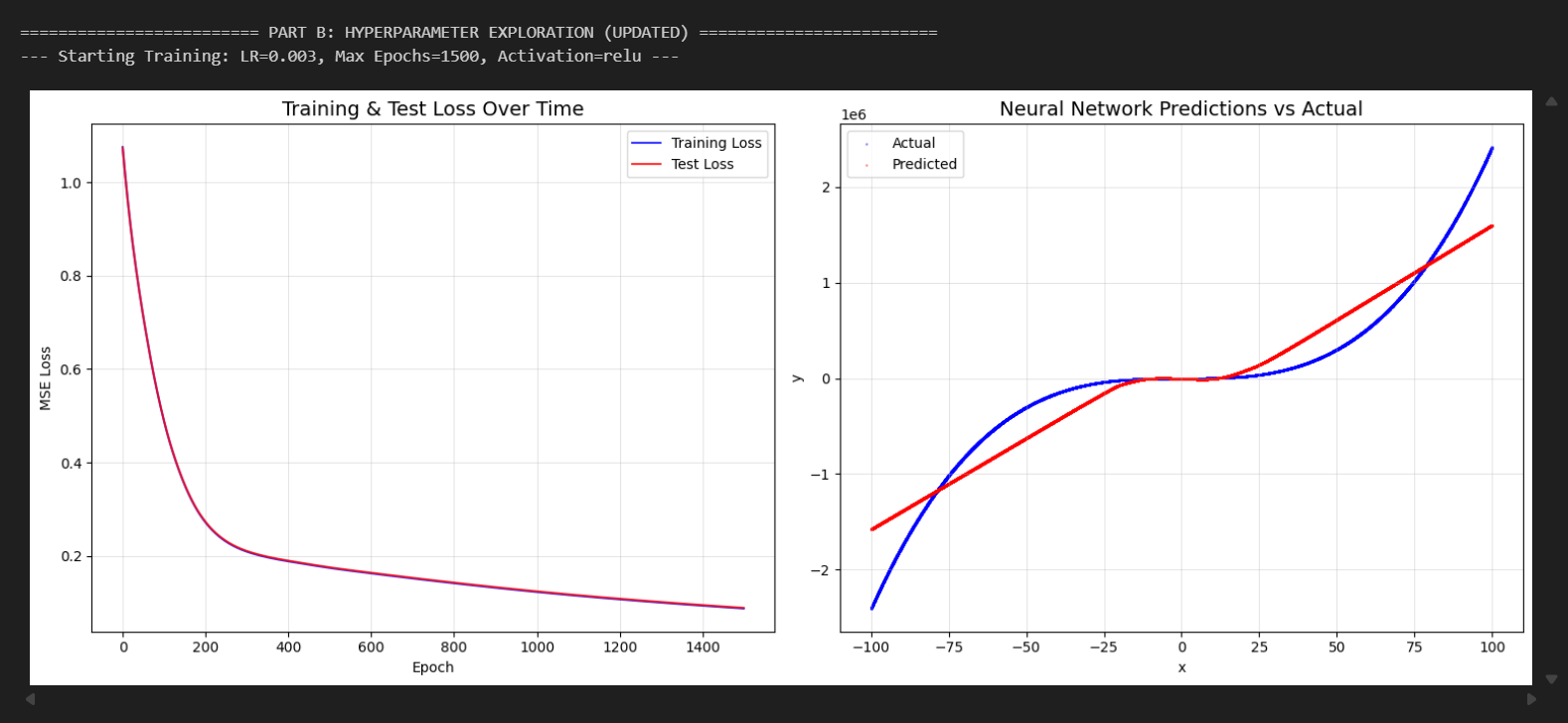
**Experiment 4: Leaky ReLU Activation (LR=0.003, Epochs=1500)**

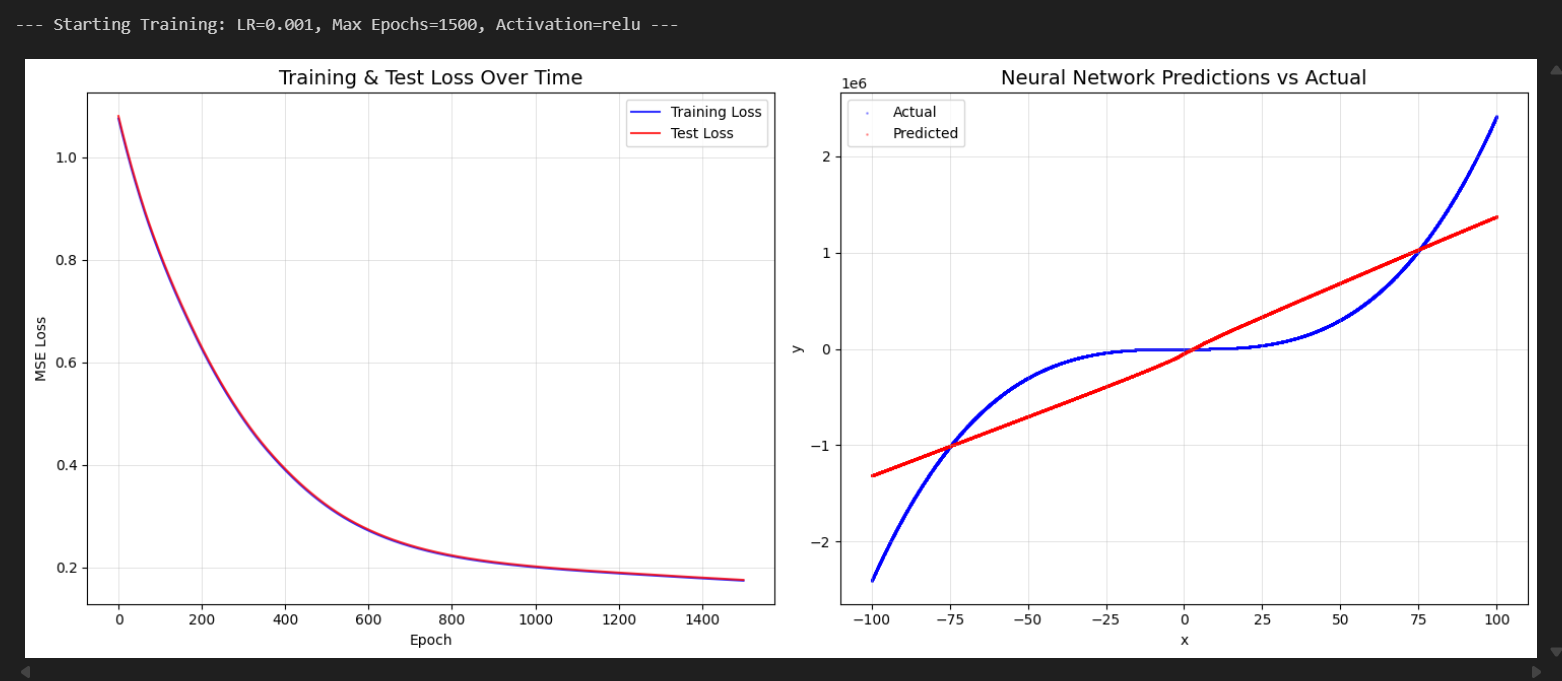
* **Final Test MSE:** 0.08892
* **R² Score:** 0.91176
* **Discussion:** Using the Leaky ReLU activation function produced results that were nearly identical to the standard ReLU in Experiment 2. The model converged well and achieved a high R² score. This suggests that for this problem, both activation functions are equally effective.

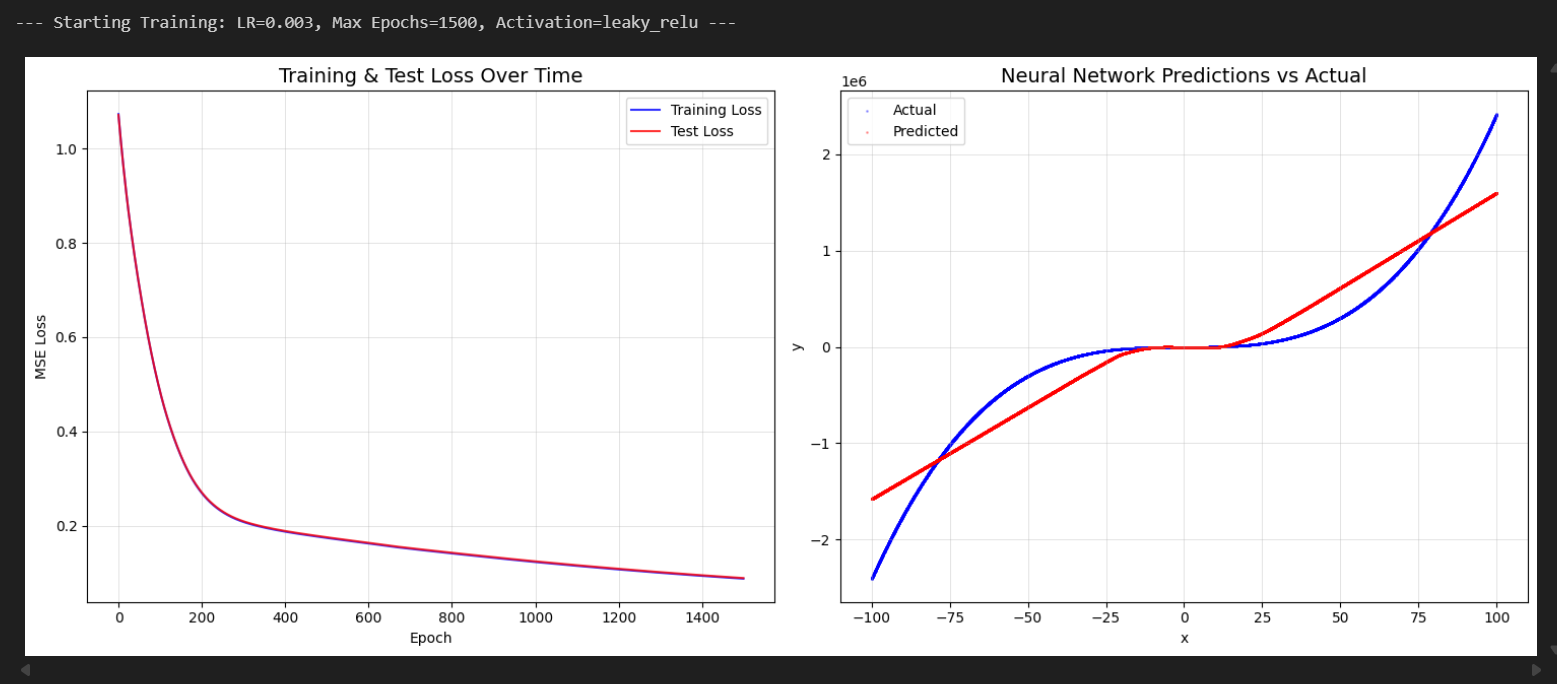
**Experiment 5: Higher Learning Rate (LR=0.006, Epochs=600)**

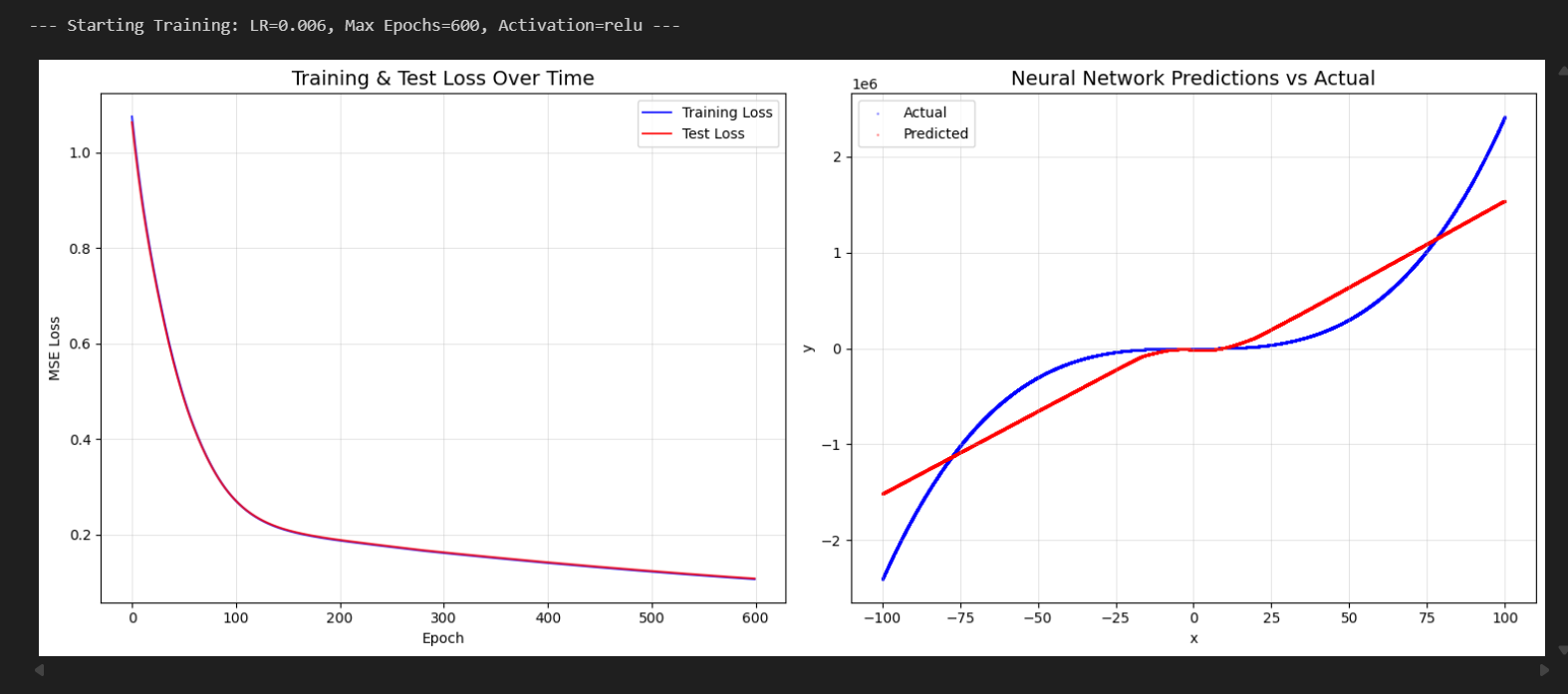
* **Final Test MSE:** 0.10813
* **R² Score:** 0.89271
* **Discussion:** A more aggressive learning rate allowed the model to converge much faster. However, the final R² score, while good, was not as high as the best-performing models. This suggests the higher rate may have caused the optimizer to slightly overshoot the true minimum loss value.

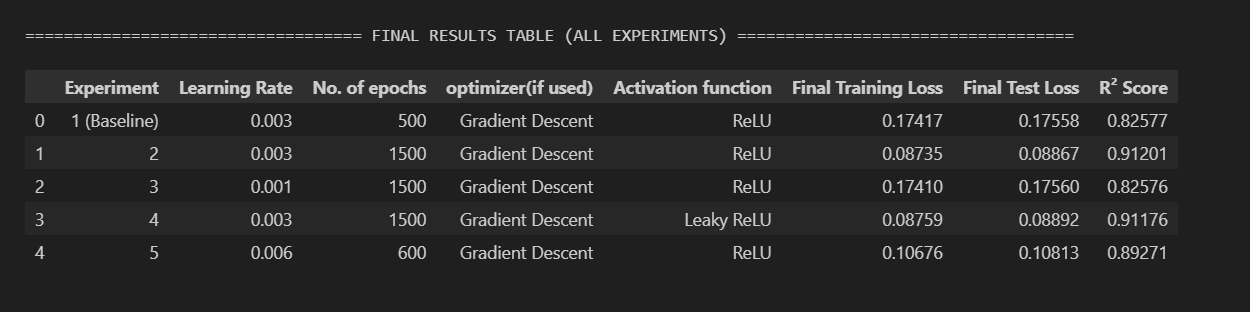












**5. Conclusion**

Based on the five experiments, several key conclusions can be drawn:

1. **Sufficient Training is Crucial:** The most significant performance gain came from increasing the number of **epochs**. The baseline model was severely undertrained, and extending the training from 500 to 1500 epochs was the primary factor in moving from a poor fit to an excellent one.
2. **Learning Rate is a Trade-off:** A learning rate of **0.003** proved to be the most effective. A lower rate (0.001) was too slow to converge, while a higher rate (0.006) converged quickly but settled for a slightly less optimal result.
3. **Activation Function Choice Was Not Critical Here:** Both **ReLU** and **Leaky ReLU** performed exceptionally well, with nearly identical top scores. This indicates that the choice of activation function was less critical than ensuring the model had enough time to train with an appropriate learning rate.

Overall, **Experiment 2 (LR=0.003, 1500 epochs, ReLU)** provided the best result, achieving the highest R² score and demonstrating a well-converged, accurate model.