3.

To find polynomials that serve as upper and lower bounds on the curve obtained from the code, we can use the coefficients a, b, and c obtained from the curve fitting:

The fitted curve is of the form:

f(n) = a. n^2 + b. n + c

From the curve fitting, we have obtained the coefficients a,b, and c.

Upper Bound Polynomial: For the upper bound, we can use a polynomial that grows faster than the fitted curve. We can take a polynomial with a higher degree, such as O(n^3)

Lower Bound Polynomial: For the lower bound, we can use a polynomial that grows slower than the fitted curve. We can take a polynomial with a lower degree, such as O(n) or (O(n log n)

Big-O Notation: The upper bound of the curve is O(n^3). This indicates that the time complexity of the algorithm is bounded above by a cubic polynomial. In other words, the algorithm has a worst-case time complexity of O(n^3).

Big-Omega Notation : The lower bound of the curve can be approximated as O(n)) or (O(n log n). This indicates that the algorithm has a best-case time complexity of at least (O(n)) or (O(n log n)

Big-Theta Notation: The big-Theta notation provides both upper and lower bounds. Since the upper bound is O(n^3) and the lower bound is O(n) or O(n log n) , we can conclude that the algorithm's time complexity is Theta(n^3) in the worst case and Omega(n) or Omega(n log n) in the best case. Therefore, the tightest bound is Theta(n^3), indicating that the algorithm's time complexity grows cubically with the input size.