1. **SOLVING NONLINEAR EQUATIONS**
   1. **Bisection:**
      1. **Steps:**
         1. Choose and as two guesses for the root such that [It’s better to chooses two x values on the same side where the root is]
         2. Estimate the root, of the equation as the mid-point between and as,
         3. Now check the following

If , then the root lies between and ; then ; .

If , then the root lies between and ; then ; .

If , then the root is ; Stop the algorithm.

* + - 1. Find the new estimate of the root, . Find the absolute relative approximate error,
      2. Compare the absolute relative approximate error with the pre-specified relative error tolerance . Also, check if the number of iterations has exceeded the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user.
    1. **Pros:**
       1. Always convergent
       2. The root bracket gets halved with each iteration - guaranteed.
    2. **Cons:**
       1. This method will work only when f(x) changes sign. If a function f(x) is such that it just touches the x-axis it will be unable to find the lower and upper guesses.
          1. If f(x) changes sign there will be odd(1,3,5..) number of roots
          2. If f(x) doesn’t changes sign there will 0/1/2/4/6.. roots
       2. Slow convergence
       3. If one of the initial guesses is close to the root, the convergence is slower
       4. Have to guess two points
       5. When function changes sign but root doesn’t exist
  1. **Newton-Raphson:**
     1. **Steps:** 
        1. Evaluate symbolically
        2. Use an initial guess of the root, , to estimate the new value of the root, [It’s better to choose the point on the side of x axis where the root is]
        3. Find the absolute relative approximate error,
        4. Compare the absolute relative approximate error with the pre-specified relative error tolerance . Also, check if the number of iterations has exceeded the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user.
     2. **Pros:**
        1. Requires only one guess
        2. Converges fast
     3. **Cons:** 
        1. Division by zero: if we guess the point, where the slope is 0 (Root can’t be found)
        2. Divergence at inflection points
        3. Oscillations near local maximum and minimum (Program won’t stop until max iteration limit exceeded)
        4. Root Jumping

1. **SOLVING LINEAR EQUATIONS**
   1. **Gaussian Elimination:**
      1. **Steps:**
         1. Forward Elimination
            1. Transform coefficient matrix into upper triangular matrix
            2. (n-1) steps of forward elimination
         2. Back Substitution
            1. Solve each equation starting from the last equation
      2. **Cons:**
         1. Division by zero
         2. Large round off error
2. **INTERPOLATION**
   1. **Newton’s Divided Difference Polynomial Method**
      1. **Pros:**
         1. Just a new term is added with the change in degree
      2. **Cons:**
         1. It’s hard to find the constants
   2. **Lagrange**
      1. **Pros:**
         1. Constants can be found easily
      2. **Cons:**
         1. All the terms changes with the change in degree